Quattro Pro Functions Help

Click **Help Topics** to return to the list of **Functions Help** topics.

@@ - Contents of Cell

Syntax

@@(Cell)

Cell	A single cell address that contains another cell address or cell name that is written as a label.

@@ is used to reference a cell that contains another cell address or cell name that is written as a label. @@ translates the label into a cell or single-cell reference and returns the contents of that cell. @@ does not accept a cell name for a selection that is not a single-cell.

Examples

@@("A15") = the contents of A15

@@("BLOCK_NAME") = the contents of the single-cell named BLOCK_NAME

@@(A3) = 50 if A3 contains the label 'A1 and cell A1 contains the value 50

@@(A3) = the label 'Total if A3 contains the label 'Block, which is the name of cell C9, which contains the label 'Total

@@(A1) = ERR, where A1 contains the label 'B1..B5

@@(A1) = B1, where A1 contains the label 'B1

@@("A1") = B1..B5, where A1 contains the label 'B1..B5

@@("[NOTEBK1]A1") = B1..B5, where A1 in the current sheet of NOTEBK1 contains the label 'B1..B5 and NOTEBK1 is open

@SUM(@@(A1)) = the sum of the values in B1..B5, where A1 contains the label 'B1..B5, because Quattro Pro translates the label into cell coordinates for a non-single cell selection

@@("B1..B5") = ERR (not a single-cell selection)

@SUM(@@("B1..B5")) = 5 if cells B1 through B5 each contain 1

Related topics

@ABDAYS--Add/Subtract Business Days

Syntax

@ABDAYS(Date, Days, <Holidays>, <Saturday>, <Sunday>)

Date	Number representing the date to which a number of business days should be added. See "Using dates and times in Quattro Pro."
Days	Integer representing number of business days to add; can be negative.
Holidays	Cells containing dates that are holidays or the date of a single holiday or 0 to indicate no holidays (the default is 0).
Saturday	0 to specify that Saturday is not a business day; 1 to specify that Saturday is a business day (the default is 0).
Sunday	0 to specify that Sunday is not a business day; 1 to specify that Sunday is a business day (the default is 0).

@ABDAYS adds or subtracts Days business days from Date and returns a serial date number.

If Date falls on a weekend or a holiday, one business day out of Days is used to bring Date forward to the next business day; if Days is negative, Date is taken backward to the previous business day. For example, if 20 business days are added to June 5, 1993, the result is the same as adding 19 business days to June 7, 1993 since June 5 falls on a Saturday.

Example

This formula calculates the date that precedes January 12, 1994 by 90 business days, assuming that Saturday, Sunday, and the dates in the cells A7..C9 are holidays:

@ABDAYS(@DATE(94,1,12),-90,A7..C9) = 34213 (September 1, 1993)

Related topics

@ABS - Absolute Value

Syntax

@ABS(X)

X	A numeric value.

@ABS returns the absolute (positive) value of X.

Examples

@ABS(-100) = 100
@ABS(100) = 100
@ABS(0) = 0

Related topics

@ACCRINT - Accrued Interest (Bond)

Syntax

@ACCRINT(Settle, Maturity, Coupon, <Issue>, <FirstCpn>, <Par>, <Freq>, <Calendar>)

Settle	Number representing the settlement date.
Maturity	Number representing the maturity date.
Coupon	Coupon rate; 0 ≤ Coupon ≤ 1.
Issue	Number representing the issue date.
FirstCpn	Number representing the first coupon date.
Par	Par value (the default is 1000).
Freq	Frequency of coupon payments in the number of payments per year (can be 1, 2, 3, 4, 6, or 12; the default is 2).
Calendar	Flag specifying which calendar to observe: 0 = US (NASD) 30/360; default 1 = Actual/actual 2 = Actual/360 3 = Actual/365 4 = European 30/360

@ACCRINT returns the accrued interest for a bond. Accrued interest represents an amount paid to the bond seller as compensation for owning the bond for a fraction of a coupon period. Interest accrues from the last coupon date to the settlement date. @ACCRINT returns the accrued interest per 1000 face value.

Dates for @ACCRINT must follow this pattern:

Issue < Settle < FirstCpn < Maturity

Example

This formula returns the accrued interest, as of May 15, 1993, on an 8.875% bond with a $100,000 face value, maturing February 15, 1998, dated November 22, 1992, and paying its first coupon on August 15, 1993:

@ACCRINT(@DATE(93,5,15),@DATE(98,2,15),0.08875,@DATE(92,11,22),@DATE(93,8,15), 100000) = $4,264.93

Related topics

@ACCRINTXL - Accrued Interest (Security)

Syntax

@ACCRINTXL(Issue, FirstCpn, Settle, Coupon, <Par>, <Freq>, <Calendar>)

Issue	Number representing the issue date. (Issue, FirstInt, Settle, Freq, and Basis are truncated to integers.) The Issue value must be < FirstCpn and <Settle.
FirstCpn	Number representing the first interest date
Settle	Number representing the settlement date.
Coupon	Interest rate; 0 ≤ Coupon.
Par	Par value (the default is 1000).
Freq	Number of coupon payments per year. For annual payments, frequency = 1; for semiannual, frequency = 2; for quarterly, frequency = 4.
Calendar	Flag specifying which calendar to observe: 0 = US (NASD) 30/360; default 1 = Actual/actual 2 = Actual/360 3 = Actual/365 4 = European 30/360

@ACCRINTXL returns the accrued interest for a security that pays periodic interest.

@ACCRINTXL uses the formula

$$I_a = par \times \frac{r}{f} \times \sum_{i=1}^{NC} \frac{A_i}{NL_i}$$

where

Ia	accrued interest
par	par value
r	coupon rate
f	frequency of coupon payments
NC	number of quasi-coupon periods that fit in odd period, rounded to next integer
Nli	normal in days of the ith quasi-coupon period within the odd period
Ai	number of accrued days for ith quasi-coupon period within the odd period

When you use any optional argument, you must also use the ones before it.

Example

This formula returns the accrued interest, as of May 15, 1996, on an 8.875% bond with a $10,000 par value, issued November 22, 1995 and paying its first interest on August 15, 1996. The US 30/360-day year is used and coupon payments are twice a year.

@ACCRINTXL(@DATE(92,11,22),@DATE(93,8,15),@DATE(93,5,15), 0.08875, 100000, 2, 0) = $4264.93

Related topics

@ACCRINTM - Accrued Interest (CD)

Syntax

@ACCRINTM(Issue, Settle, Coupon, <Par>, <Calendar>)

Issue	Number representing the issue date; must be < Settle.
Settle	Number representing the settlement date.
Coupon	Coupon rate; 0 ≤ Coupon ≤ 1.
Par	Par value (the default is 1000).
Calendar	Flag specifying which calendar to observe: 0 = US (NASD) 30/360; default 1 = Actual/actual 2 = Actual/360 3 = Actual/365 4 = European 30/360

@ACCRINTM calculates the amount of interest accrued per par value between the issue date of the coupon and the settle date.

Example

This formula returns the accrued interest on a certificate of deposit (CD) with $1,000,000 face value settling March 11, 1990, dated December 15, 1989, and paying a coupon of 10% on an actual/360 basis:

@ACCRINTM(@DATE(89,12,15),@DATE(90,3,11),0.10,1000000,2) = $23,888.89

Related topics

@ACCRUED - Accrued Interest (Security)

Syntax

@ACCRUED(Settle, Issue, FirstInt, Coupon, <Par>, <Freq>, <Calendar>)

Settle	Number representing the settlement date; must be greater than Issue.
Issue	Number representing the issue date.
FirstInt	Number representing the first interest date; must be greater than Issue.
Coupon	Coupon rate; can be any positive value, including 0.
Par	Par value, or the principal to be paid at maturity (optional); the default is 100.
Freq	Frequency of coupon payments (optional) in number of payments per year; can be 1, 2, 4, or 12; the default is 2.
Calendar	Flag specifying which calendar to observe: 0 = US (NASD) 30/360; default 1 = Actual/actual 2 = Actual/360 3 = Actual/365 4 = European 30/360

@ACCRUED calculates the accrued interest for a security with periodic interest payments. Short, standard, and long coupon periods can also be used.

@ACCRUED uses the formula

$$I_a = par \times \frac{r}{f} \times \sum_{i=1}^{NC} \frac{A_i}{NL_i}$$

where

Ia	accrued interest
par	par value
r	coupon rate
f	frequency of coupon payments
NC	number of quasi-coupon periods that fit in odd period, rounded to next integer
Nli	normal in days of the ith quasi-coupon period within the odd period
Ai	number of accrued days for ith quasi-coupon period within the odd period

When you use any optional argument, you must also use the ones before it.

Example

This formula returns the accrued interest, as of May 15, 1996, on an 8.875% bond with a $10,000 par value, issued November 22, 1995 and paying its first interest on August 15, 1996. The US 30/360-day year is used and coupon payments are twice a year.

@ACCRUED(@DATE(93,5,15), @DATE(92,11,22), @DATE(93,8,15), 0.08875, 10000, 2, 0) = 426.4931

Related topics

@ACDAYS - Add Calendar Days

Syntax

@ACDAYS(Date, Days, <Calendar>, <EndMnth>)

Date	Number representing the date to add days to. See "Using dates and times in Quattro Pro."
Days	Integer representing number of days to add; can be negative.
Calendar	Flag specifying which calendar to observe:
	0 = US (NASD) 30/360; default
	1 = Actual/actual
	2 = Actual/360
	3 = Actual/365
	4 = European 30/360
EndMnth	1 to indicate adherence to ends of months; 0 to indicate that ends of months are ignored (the default is 1).

@ACDAYS adds Days days to Date using an actual or 30/360 calendar and returns a serial date number. If Days is negative, @ACDAYS subtracts the absolute value of Days from Date.

With the 30/360 calendar, if the ending month is February and the remaining days push the result to the 29th or 30th, the result is forced to the true end of the month (28th or 29th, depending on whether Date is in a leap year).

More than one result is possible when using the 30/360 calendar. For example, adding 90 days to April 30 can yield either July 30 or July 31.

Examples

@ACDAYS(@DATE(93,1,1),120) = 34090 (May 1, 1993)

@ACDAYS(@DATE(93,11,15),-270) = 34015 (February 15, 1993)

Related topics

@ACOS - Arc Cosine

Syntax

@ACOS(X)

X	A numeric value between -1 and 1.

@ACOS returns the arc cosine of X. The result is the angle (in radians) whose cosine is X. To convert radians to degrees, use @DEGREES.

Examples

@ACOS(1) = 0

@ACOS(0.5) = 1.047198

@DEGREES(@ACOS(0.5)) = 60

@ACOS(@ABS(B10)) = the arc cosine of the absolute value of B10

@ACOS(2) = ERR (means that X is greater than 1)

Related topics

@ACOSH - Arc Hyperbolic Cosine

Syntax

@ACOSH(X)

X	The hyperbolic cosine of an angle. X must be greater than or equal to 1 but less than approximately 1.34078E+154.

@ACOSH returns the arc, or inverse, hyperbolic cosine of a number. The arc hyperbolic cosine is the value whose hyperbolic cosine is X, so @ACOSH(@COSH(X)) = X.

@ACOSH returns the result in radians; to convert to degrees, use @DEGREES.

Examples

@ACOSH(1) = 0

@ACOSH(2) = 1.316958

@ROUND(@DEGREES(@ACOSH(1.600287)), 2) = 60 degrees. Or: "The angle whose hyperbolic cosine is 1.600287, rounded to 2 decimal places."

@ACOSH(@ABS(D33)) = the arc hyperbolic cosine of the absolute value of D33

@ACOSH(0.5) = ERR (means that X is less than 1)

Related topics

@ACOT - Arc Cotangent

Syntax

@ACOT(X)

X	The cotangent of an angle. X can be any value from approximately -1.789E+308 through 1.789E+308.

@ACOT calculates the arc, or inverse, cotangent using the cotangent X of an angle. The result of @ACOT is an angle, in radians, from 0 through π. This represents an angle between 0 and 180 degrees. To convert radians to degrees, use @DEGREES.

Examples

@ACOT(0.5) = 1.107149

@ACOT(1) = 0.785398

@ROUND(@DEGREES(@ACOT(1)),2) = 45

Related topics

@ACOTH - Arc Hyperbolic Cotangent

Syntax

@ACOTH(X)

X	The hyperbolic cotangent of an angle. X can be any value between approximately -1.79E+308 and less than -1 and between greater than 1 and approximately 1.79E+308.

@ACOTH calculates the arc, or inverse, hyperbolic cotangent using the hyperbolic cotangent X of an angle. The result is in radians; to convert to degrees, use @DEGREES.

Examples

@ACOTH(4) = 0.255413

@ACOTH(@PI/4) = ERR, because π /4 is between -1 and 1

@ACOTH(@PI/3) = 1.884943

@ROUND(@DEGREES(@ACOTH(1.524869)), 2) = 45 degrees. Or: "The angle whose hyperbolic cotangent is 1.524869, rounded to 2 decimal places."

Related topics

@ACSC - Arc Cosecant

Syntax

@ACSC(X)

X	The cosecant of an angle. X can be any value between approximately -1.79E+308 and -1 and between 1 and approximately 1.79E+308.

@ACSC calculates the arc, or inverse, cosecant using the cosecant X of an angle. The result of @ACSC is an angle, in radians, from - π/2 through π/2 (from -90 through 90 degrees). To convert radians to degrees, use @DEGREES.

Examples

@ACSC(1) = 1.570796

@ACSC(-2) = - 0.5236

@ROUND(@DEGREES(@ACSC(-2)),2) = - 30 degrees

@ACSC(0.25) = ERR, because X is between -1 and 1

Related topics

@ACSCH - Arc Hyperbolic Cosecant

Syntax

@ACSCH(X)

X	The hyperbolic cosecant of an angle. X can be any value between approximately -1.34078E+154 and 1.34078E+154, but not 0.

@ACSCH calculates the arc, or inverse, hyperbolic cosecant using the hyperbolic cosecant X of an angle. The result is in radians; to convert to degrees, use @DEGREES.

Examples

@ACSCH(@RADIANS(30)) = 1.402581

@ACSCH(@RADIANS(75)) = 0.704265

@ROUND(@DEGREES(@ACSCH(1.825306)), 2) = 30 degrees. Or: "The angle whose hyperbolic cosecant is 1.825306, rounded to 2 decimal places."

Related topics

@ADDB - Add Binary Numbers

Syntax

@ADDB(Binary1, <Binary2>, <BitIn>, <Bits>)

Binary1	First binary number.
Binary2	Second binary number.
BitIn	Input carry bit; can be either 0 (the default) or 1.
Bits	Number of binary bits used for both input and output; if omitted, Bits = number of bits in Binary1 or Binary2, whichever is greater; must be in the range 0 <n ≤ 64.

@ADDB returns the sum of two binary numbers. If Binary2 is omitted, @ADDB counts the bits in Binary1 that are set to 1; this bit counting operation is called addition reduction.

Use two's complement notation (see Quattro Pro glossary) to represent negative numbers. If BitIn is 1, @ADDB adds one extra bit to the result.

Example

@ADDB(100,100) = 1000

@ADDB(100,100,1,4) = 1001

@ADDB(101) = 2

@ADDB(1100,1,1,5) = 01110

Related topics

@ADDBO - Overflow of Binary Addition

Syntax

@ADDBO(Binary1, Binary2, <BitIn>, <Bits>)

Binary1	First binary number.
Binary2	Second binary number.
BitIn	Input carry bit; can be either 0 (the default) or 1.
Bits	Number of binary bits used for input; if omitted, Bits = number of bits in Binary1 or Binary2, whichever is greater; must be in the range 0 <n ≤ 64.

@ADDBO returns the overflow bit (either 0 or 1) of the sum of two binary numbers. An overflow occurs when a bit is carried out of the word size specified by Bits. For example, if Binary1 = 10 and Binary2 = 10, the sum of the two numbers is 00, with 1 carry bit in the third place not shown.

Use two's complement notation (see Quattro Pro glossary) to represent negative numbers. If BitIn is 1, @ADDBO adds one extra bit to the sum of the two numbers before returning the overflow.

Example

@ADDBO(1000,111) = 0

@ADDBO(1000,111,1) = 1

@ADDBO(1100,100,1,4) = 1

Related topics

@ADDH - Add Hexadecimal Numbers

Syntax

@ADDH(Hex1, <Hex2>, <BitIn>, <Bits>)

Hex1	First hexadecimal number.
Hex2	Second hexadecimal number.
BitIn	Input carry bit; can be either 0 (the default) or 1.
Bits	Number of binary bits used for both input and output; if omitted, Bits = number of bits in Hex1 or Hex2, whichever is greater; must be in the range 0 <n ≤ 64.

@ADDH returns the sum of two hexadecimal numbers. If Hex2 is omitted, @ADDH counts the bits in Hex1 that are set to 1; this bit counting operation is called addition reduction.

Use two's complement notation (see Quattro Pro glossary) to represent negative numbers. If BitIn is 1, @ADDH adds one extra bit to the result.

Example

@ADDH("E00","100") = F00

@ADDH("100","100",1,16) = 0201

@ADDH("9") = 2

@ADDH("C","1",1,8) = 0E

Related topics

@ADDHO - Overflow of Hexadecimal Addition

Syntax

@ADDHO(Hex1, Hex2, <BitIn>, <Bits>)

Hex1	First hexadecimal number.
Hex2	Second hexadecimal number.
BitIn	Input carry bit; can be either 0 (the default) or 1.
Bits	Number of binary bits used for both input and output; if omitted, Bits = number of bits in Hex1 or Hex2, whichever is greater; must be in the range $0 < n \leq 64$.

@ADDHO returns the overflow bit (either 0 or 1) of the sum of two hexadecimal numbers. An overflow occurs when a bit is carried out of the word size specified by Bits. For example, if the binary equivalents for Hex1 and Hex2 are 10 and 10, the sum of the two numbers is 00 with 1 carry bit in the third place not shown.

Use two's complement notation (see Quattro Pro glossary) to represent negative numbers. If BitIn is 1, @ADDHO adds one extra bit to the sum of the two numbers before returning the overflow.

Examples

@ADDHO("8","F") = 1

@ADDHO("8","F",1,5) = 0

@ADDHO("C","4",1,4) = 1

Related topics

@ADDRESS - Cell Reference

Syntax

@ADDRESS(RowNum, ColNum, <RefType>, <Format>, <Page>)

RowNum	Row number of the cell for which you want the reference.
ColNum	Column number.
RefType	Type of cell reference to return (optional).
Format	Logical value indicating A1 or R1C1 reference style (optional):
	TRUE = A1 (default, if omitted)
	FALSE = R1C1
Page	Name of notebook sheet (optional).

@ADDRESS returns, as text, a cell reference for which you specify row and column numbers. Optionally, you can specify another sheet and choose reference type and style.

RefType	Format	Meaning
1	$A:$A$1	All absolute
2	$A:A$1	Absolute sheet and row, relative column
3	$A:$A1	Absolute sheet and column, relative row
4	$A:A1	Absolute sheet, relative column and row
5	A:$A$1	Relative sheet, absolute column and row
6	A:A$1	Relative sheet and column, absolute row
7	A:$A1	Relative sheet and row, absolute column
8	A:A1	All relative

You can use @ADDRESS with @INDEX, @VLOOKUP, or @HLOOKUP to create cell references from tables of values in the current file. Use @ADDRESS with @@ to return values in cell references.

Examples

@ADDRESS(5,4) = "$D$5" - absolute reference to Cell D5

@ADDRESS(5,4,3) = "$D5" - absolute reference to column D, relative reference to row 5

@ADDRESS(5,4,3,FALSE) = "R5C[4]" - relative row 5, absolute column 4, in R1C1 style

@ADDRESS(5,4,3,TRUE,"AREAS") = "$AREAS **:** $D5" - absolute sheet name, absolute column D, and relative row 5

If B7 contains 7, and C7 contains 6, @@(@ADDRESS(B7,C7)) returns the value in cell F7.

Related topics

@AMAINT - Amortized Accumulated Interest

Syntax

@AMAINT(Principal, Int, Term, n, <Part>, <Residual>, <ResOff>, <Adv>, <Odd>, <Simp>)

Principal	Initial loan principal.
Int	Periodic interest rate.
Term	Term of the loan, expressed in number of total payments.
n	Number of payments made; must be an integer from 0 to Term.
Part	Part of (n+1)th period passed (must be from 0 to 1; the default is 0).
Residual	Remaining balance on loan at end of loan term (the default is 0).
ResOff	Number of periods after last periodic payment that residual is to be paid; can have fractional component (the default is 0).
Adv	Number of advance payments made at loan inception (the default is 0); n - Adv must be an integer.
Odd	Number of periods between loan inception and date of first payment (not including advance payments); can have fractional component (the default is 1).
Simp	0 to specify compounded interest or 1 to specify simple interest (the default is 0).

@AMAINT calculates the accumulated interest paid on a loan after n payments. The accumulated interest is the sum of the interest portions of the first n payments plus the interest of an optional partial payment (specified by Part).

Term and n should include advance payments made at the beginning of the loan. Part handles payoff situations where the payoff date does not coincide with a periodic payment date. For example, if 20 out of 60 monthly payments of a loan have been made and 10 days have passed since the 20th payment date, Part = 10/31, assuming there are 31 days between the 20th and 21st monthly payments.

If Simp = 0, @AMAINT uses this formula:

$$I = n * Pa - Pr + B * (1 + Int)^P$$

If Simp = 1, @AMAINT uses this formula:

$$I = n * Pa - Pr + B * (1 + Int * P)$$

where

I	interest
Pa	payment
Pr	principal
B	balance
P	part

where Payment is the periodic payment and Balance is the remaining balance on the loan after n payments. Balance, like Principal, is the present value of an annuity paying Payment.

Examples

A loan of $10,000 was made on September 11, 1992 to be repaid in 48 monthly installments. The annual interest rate is 8.4% (8.4%/12 = 0.7% monthly rate). The first payment was paid in advance. This formula calculates the amount of paid interest after 15 regular payments:

@AMAINT(10000,0.007,48,15,0,0,0,1) = $840.74

For the same loan, this formula calculates how much interest accumulated on the loan as of March 3, 1993,

assuming the borrower made regular payments. The previous payment (the 18th) fell on February 11, 1993; there are 21 days between the previous payment and March 3. Interest accrues as simple interest over fractional periods.

@AMAINT(10000,0.007,48,18,21/29,0,0,1,1,1) = $1,020.76

Related topics

@AMINT - Amortized Interest Rate

Syntax

@AMINT(Principal, Term, Payment, <Residual>, <ResOff>, <Adv>, <Odd>, <Simp>, <Prec>)

Principal	Initial loan principal.
Term	Term of loan, expressed in number of total payments.
Payment	Periodic payment (for example, if Term is expressed in months, Payment must be a monthly payment).
Residual	Remaining balance on loan at end of loan term (the default is 0).
ResOff	Number of periods after last periodic payment that Residual is to be paid; can have fractional component (the default is 0).
Adv	Number of advance payments made at loan inception (the default is 0).
Odd	Number of periods between loan inception and date of first payment (not including advance payments); can have fractional component (the default is 1).
Simp	0 to specify compounded interest or 1 to specify simple interest (the default is 0).
Prec	Required precision of result (the default is 0.000001); must be ≥ 0.

@AMINT calculates the interest rate for one payment of an amortized loan. For example, if the arguments passed correspond to a monthly loan, the interest rate returned represents a monthly rate. Use Prec to specify how close @AMINT must be to the actual interest rate.

Example

A loan for $50,000 was made on March 15, 1993. Repayment terms stipulate ten annual payments of $7,500, each to be made on July 31, beginning 1993 and ending 2002, along with a final payment of $2,500 on December 31, 2003. Assuming interest is compounded during fractional periods, this formula calculates the interest rate at which the financing is performed:

@AMINT(50000,10,7500,2500,1.4180,0,0.3781) = 0.098913

A normal period is one year long, but the first period is 138 days long. Since there are 365 days between March 15, 1993 and March 15, 1994, the first period is 138/365 the length of a normal period (Odd = 0.3781). There is also a delay between the last periodic payment and the residual payment of $2,500. The length of the delay is one period (July 31, 2002 to July 31, 2003) plus 153 days (July 31, 2003 to December 31, 2003), so ResOff = 1.4180 (which equals 1+(153/366)).

Related topics

@AMNTHS - Add Months

Syntax

@AMNTHS(Date, Months, <EndMnth>)

Date	Number representing the date to add number of months to. See "Using dates and times in Quattro Pro."
Months	Integer representing number of months to add; can be negative.
EndMnth	1 to indicate adherence to ends of months; 0 to indicate that ends of months are ignored (the default is 1).

@AMNTHS adds the number of months specified by Months to Date and returns a serial date number. If Months is negative, @AMNTHS subtracts the absolute value of Months from Date.

Adding one month usually means going from a day in one month to the same day in the next month. However, adding one month to March 31 cannot result in April 31, since April 31 does not exist. In this case, the last day of the month, April 30, is returned. If Date falls on the 31st of a month, the result also falls on the last day of a month.

If Date falls on the last day of a month with less than 31 days, you can use EndMnth to specify one of two different results. To move ahead to the same day of the specified month, specify EndMnth as 0. To move ahead to the last day of the specified month, omit EndMnth or specify it as 1.

Examples

@AMNTHS(@DATE(93,4,30),3) = 34181 (July 31, 1993), which is the last day of the month three months from April 30, 1993.

Consider a loan with 120 payments that pays on the 30th of each month starting on June 30, 1993. In February, it pays on the last day of the month. This formula calculates the date of the 43rd payment:

@AMNTHS(@DATE(93,6,30),42,0) = 35429 (December 30, 1996)

Related topics

@AMPMT - Amortized Payment

Syntax

@AMPMT(Principal, Int, Term, <Residual>, <ResOff>, <Adv>, <Odd>, <Simp>)

Principal	Initial loan principal.
Int	Periodic interest rate (for example, if Term is expressed in months, Int must be a monthly rate).
Term	Term of loan, expressed in number of total payments.
Residual	Remaining balance on loan at end of loan term (the default is 0).
ResOff	Number of periods after last periodic payment that residual is to be paid; can have fractional component (the default is 0).
Adv	Number of advance payments made at loan inception (the default is 0).
Odd	Number of periods between loan inception and date of first payment (not including advance payments); can have fractional component (the default is 1).
Simp	0 to specify compounded interest or 1 to specify simple interest (the default is 0).

@AMPMT calculates the payment (monthly, annual, and so on) for an amortized loan.

Examples

A loan for $35,000 has 48 monthly payments and a residual payment of $7,500 that is due three months after the last monthly payment. This formula calculates the monthly payment if the annual interest rate is 9% (9%/12 = 0.75% monthly rate):

@AMPMT(35000,0.0075,48,7500,3) = $743.48

An annuity with an investment of $250,000 makes quarterly payments starting five years from the date of investment for a period of 20 years. It also pays a lump sum of $50,000 three and a half years after the last quarterly payment. If the annualized yield from the investment is 8.4% (8.4%/4 = 2.1% quarterly rate), this formula calculates the quarterly payment:

@AMPMT(250000,0.021,80,50000,14,0,20) = $9,431.83

The term of the annuity is 80 quarters. A value of 14 for ResOff specifies, in quarters, the three-and-a-half year delay between the last quarterly payment and the residual payment. A value of 20 for Odd specifies the five year delay between investment and first payment, in quarters.

Related topics

@AMPMTI - Amortized Interest Portion of Payment

Syntax

@AMPMTI(Principal, Int, Term, n, <Residual>, <ResOff>, <Adv>, <Odd>, <Simp>)

Principal	Initial loan principal.
Int	Periodic interest rate (for example, if Term is expressed in months, then Int must be a monthly rate).
Term	Term of loan, expressed in number of total payments.
n	Number of payments made; must be an integer from 0 to Term.
Residual	Remaining balance on loan at end of loan term (the default is 0).
ResOff	Number of periods after last periodic payment that residual is to be paid; can have fractional component (the default is 0).
Adv	Number of advance payments made at loan inception (the default is 0).
Odd	Number of periods between loan inception and date of first payment (not including advance payments); can have fractional component (the default is 1).
Simp	0 to specify compounded interest or 1 to specify simple interest (the default is 0).

@AMPMTI calculates the interest portion of the nth payment of an amortized loan. Term and n should include any advance payments made at the beginning of the loan.

If Simp = 0, @AMPMTI uses this formula:

$$I_n = B_{n-1} * [(1+Int)^{tn} - 1]$$

If Simp = 1, @AMPMTI uses this formula:

$$I_n = B_{n-1} * [((1+Int)^{I(t_n)} * (1+F(t_n) * Int)) - 1]$$

I	interest
B	balance

Balance n-1, like Principal, is the present value of an annuity. Both correspond to annuities with the same payment size but differing in number of payments. If Principal corresponds to an annuity with Term payments, Balance n-1 corresponds to annuity with Term n + 1 payments. tn equals 1 except when n equals Adv + 1, in which case tn equals Odd.

Example

This formula calculates the portion of the 15th payment (the 15th after any advanced payments) of a 120 payment loan that constitutes interest, if the original principal of $100,000 is financed at a periodic rate of 4.5% and the first two payments are made in advance:

@AMPMTI(100000,0.045,120,17,0,0,2) = $4,106.92.

The value of 17 passed for n specifies the 15th payment after the two advance payments.

Related topics

@AMPRN - Amortized Initial Principal

Syntax

@AMPRN(Int, Term, Payment, <Residual>, <ResOff>, <Adv>, <Odd>, <Simp>)

Int	Periodic interest rate (for example, if Term is expressed in half-years, Int must be a semiannual rate).
Term	Term of loan, expressed in number of total payments.
Payment	Periodic payment.
Residual	Remaining balance on loan at end of loan term (the default is 0).
ResOff	Number of periods after last periodic payment that residual is to be paid; can have fractional component (the default is 0).
Adv	Number of advance payments made at loan inception (the default is 0).
Odd	Number of periods between loan inception and date of first payment (not including advance payments); can have fractional component (the default is 1).
Simp	0 to specify compounded interest or 1 to specify simple interest (the default is 0).

@AMPRN calculates the initial principal of an amortized loan.

Examples

A loan has an annualized monthly compounded interest rate of 10.8% (10.8%/12 = 0.9% monthly rate) over a period of 48 months, and the monthly payment is $525. Each payment is due at the beginning of the month, including the first payment which coincides with the loan's start date. This formula calculates the amount financed:

@AMPRN(0.009,48,525,0,0,1) = $20,573.04

A savings plan requires a monthly contribution of $1,000 for a period of 25 years. If the plan pays an annual interest rate of 6.6% (6.6%/12 = 0.55% monthly rate), this formula calculates what initial deposit (not payment), if any, is needed in order to accumulate $1,000,000 two and a half years after the last monthly payment:

@AMPRN(0.0055,300,-1000,1000000,30) = $16,907.93.

If the annuity is viewed as a loan and the investor as the lender, the original investment can be treated as the loan principal, the monthly payments as payments from lender to borrower, and the $1,000,000 future value as a residual payment from the borrower to the lender. The interest rate is 0.55%. The negative payment means payment from lender to borrower. A value of 30 for ResOff specifies the thirty month delay between the last monthly contribution and the date on which to measure the end balance.

Related topics

@AMRES - Amortized Residual Payment

Syntax

@AMRES(Principal, Int, Term, Payment, <ResOff>, <Adv>, <Odd>, <Simp>)

Principal	Initial loan principal.
Int	Periodic interest rate (for example, if Term is expressed in months, then Int must be a monthly rate).
Term	Term of loan, expressed in number of total payments.
Payment	Periodic payment.
ResOff	Number of periods after last periodic payment that Residual is to be paid; can have fractional component (the default is 0).
Adv	Number of advance payments made at loan inception (the default is 0).
Odd	Number of periods between loan inception and date of first payment (not including advance payments); can have fractional component (the default is 1).
Simp	0 to specify compounded interest or 1 to specify simple interest (the default is 0).

@AMRES calculates the residual (or balloon) payment of an amortized loan or the future value of an annuity.

Example

A $10,000,000 loan is paid back in 20 payments of $1,000,000 and a final payment. The first payment is made at the start of the loan. The remaining 19 payments are made annually, starting 9 months after the first payment. The final payment is made 20 months after the last annual payment. If the loan has an annual interest rate of 9.68%, this formula calculates the final payment (assume compounding of interest over fractional periods):

@AMRES(10000000,0.0968,20,1000000,20/12,1,0.75) = $1,681,942.54.

ResOff is set to 20/12 to specify the 20 month delay between the last annual payment and the residual payment. Adv is set to 1 to specify the advance payment. Odd is set to 9/12 (0.75) to specify the nine month period between the start of the loan and the second payment.

Related topics

@AMRPRN - Amortized Remaining Principal

Syntax

@AMRPRN(Principal, Int, Term, n, <Part>, <Residual>, <ResOff>, <Adv>, <Odd>, <Simp>)

Principal	Initial loan principal.
Int	Periodic interest rate (for example, if Term is expressed in years, then Int must be a yearly rate).
Term	Term of loan, expressed in number of total payments.
n	Number of payments made; must be an integer from 0 to Term.
Part	Part of (n+1)th period passed; $0 \leq$ Part ≤ 1 (the default is 0).
Residual	Remaining balance on loan at end of loan term (the default is 0).
ResOff	Number of periods after last periodic payment that residual is to be paid; can have fractional component (the default is 0).
Adv	Number of advance payments made at loan inception (the default is 0).
Odd	Number of periods between loan inception and date of first payment (not including advance payments); can have fractional component (the default is 1).
Simp	0 to specify compounded interest or 1 to specify simple interest (the default is 0).

@AMRPRN computes the balance remaining after n payments, accounting for possible interest accrual over part of the following payment period.

Term and n should include advance payments made at the beginning of the loan. Part handles payoff situations where the payoff date (date on which the remaining balance on a loan is fully paid) does not coincide with a periodic payment date. For example, if 15 out of 36 monthly payments of a loan have been made and 17 days have passed since the 15th payment date, Part = 17/30, assuming there are 30 days between the 15th and 16th monthly payments.

If Simp = 0, @AMRPRN uses this formula:

$$B_{n+p} = B_n * (1+Int)^P$$

If Simp = 1, @AMRPRN uses this formula:

$$B_{n+p} = B_n * (1+Int*P)$$

B	Balance
P	Part

Examples

A loan of $100,000 has an annual interest rate of 9.6% (9.6%/12 = 0.8% monthly rate). Repayment consists of monthly payments over ten years. This formula calculates the balance remaining after the 57th monthly payment:

@AMRPRN(100000,0.008,120,57) = $64,108.38

A loan of $2,000,000 was made on March 16, 1993, to be paid back in 40 quarterly payments and a final payment of $100,000. The annual interest rate is 9.96% (9.96%/4 = 2.49% quarterly rate). The first four payments are due at the start of the loan. The fifth payment is due July 1, 1993. Thereafter, a payment is due every three months. The final residual payment of $100,000 is due June 15, 2002. This formula calculates the remaining balance due on the loan as of September 23, 1996, assuming timely payments and simple interest accrual over fractional periods:

@AMRPRN(2000000,0.0249,40,17,84/92,100000,75/91,4,1+16/90,1) = $1,322,015.26.

September 23, 1994 falls in the middle of the payment period following the 17th quarterly payment. It falls 84 days into the quarter, which is 92 days long, so Part = 84/92. The residual is paid 75 days after the 40th quarterly payment. The corresponding quarter (April 1, 2002 to July 1, 2002) contains 91 days, so ResOff = 75/91. The first quarterly payment after the advance payments is paid one period and 16 days after loan inception. The 16 days correspond to the quarter containing the loan inception date, March 16, 1993. That quarter contains 90 days, so Odd = 1+(16/90).

Related topics

@AMTERM - Amortized Term

Syntax

@AMTERM(Principal, Int, Payment, <Residual>, <ResOff>, <Adv> <Odd>, <Simp>)

Principal	Initial loan principal.
Int	Periodic interest rate (for example, if term is expressed in quarters, Int must be a quarterly rate).
Payment	Periodic payment.
Residual	Remaining balance on loan at end of loan term (the default is 0).
ResOff	Number of periods after last periodic payment that residual is to be paid; can have fractional component (the default is 0).
Adv	Number of advance payments made at loan inception (the default is 0).
Odd	Number of periods between loan inception and date of first payment (not including advance payments); can have fractional component (the default is 1).
Simp	0 to specify compounded interest or 1 to specify simple interest (the default is 0).

@AMTERM calculates the duration of an amortized loan, expressed in number of payments.

Example

A loan for $50,000 was made on April 1, 1993. The loan is repaid monthly, starting on May 1, 1993, except for the first three payments, which are due at the start of the loan (April 1, 1993). If the monthly interest rate is 1.15%, this formula calculates the smallest number of payments that would allow repayment with a maximum allowable monthly payment of $600:

@AMTERM(50000,0.0115,600,0,0,3) = 228.18

The smallest number of payments to support such a loan is 229, which results in a monthly payment of $599.56. Using 228 payments results in a monthly payment of $600.10.

Related topics

@AND - Logical And

Syntax

@AND(List)

List	True-or-false conditions to test.

@AND returns 1 (true) if all arguments are true, 0 (false) if even one argument is false.

Arguments in List must be logical values or references.

@AND ignores text, numbers, or empty cells.

Examples

Given the following data:

	A	B	C
1	$2	$101	$12
2	$50	$115	$22
3	$127	$130	$45

@AND(A1>10,A2>10,A3>10) = 0 (false)

@AND(B1>10,B2>10,B3>10) = 1 (true)

To find which values in column A are less than 100, enter in cell A4 the formula +A1..A3<100. Quattro Pro enters the formula as an array and returns the array {1|1|0} in cells A4..A6, showing the first two values in column A are less than 100. You can do the same in cell B4 for the amounts in column B, and C4 for the amounts in column C.

Suppose a $5 service charge is deducted if the daily account balance falls below the $100 minimum for three consecutive days. Use @AND to test the true-or-false conditions in A4..A6, B4..B6, and C4..C6, and @IF to subtract $5 or not, depending on the results.

For account A, @IF(@AND(A4..A6), "$5", "$0") = $0, no service charge

For account B, @IF(@AND(B4..B6), "$5", "$0") = $0, no service charge

For account C, @IF(@AND(C4..C6), "$5", "$0") = $5

Related topics

@ANDB - Binary AND

Syntax

@ANDB(Binary1, <Binary2>, <Bits>)

Binary1	First binary number.
Binary2	Second binary number.
Bits	Number of binary bits used for both input and output; if omitted, Bits = number of bits in Binary1 or Binary2, whichever is greater; must be in the range 0 <n ≤ 64.

@ANDB performs a bit-by-bit logical AND of each bit in Binary1 and Binary2. Use @ANDB to set bits to 0; any bit that is 0 in either Binary1 or Binary2 causes the resulting output bit to be 0.

If only one number is specified, then @ANDB performs an all-ones test, or AND reduction, on Binary1; @ANDB returns 1 if all bits in Binary1 are set to 1; otherwise, it returns 0.

Examples

@ANDB(10,1) = 00

@ANDB(11,10) = 10

@ANDB(11) = 1

@ANDB(1100,111,5) = 00100

Related topics

@ANDH - Hexadecimal AND

Syntax

@ANDH(Hex1, <Hex2>, <Bits>)

Hex1	First hexadecimal number.
Hex2	Second hexadecimal number.
Bits	Number of binary bits used for both input and output; if omitted, Bits = number of bits in Hex1 or Hex2, whichever is greater; 4 binary digits = 1 hexadecimal digit; must be in the range 0 <n ≤ 64.

@ANDH performs a bit-by-bit logical AND of each bit in Hex1 and Hex2. Use @ANDH to set bits to 0; any binary bit that is 0 in either Hex1 or Hex2 causes the resulting output bit to be 0.

If only one number is specified, then @ANDH performs an all-ones test, or AND reduction, on Hex1; @ANDH returns 1 if all bits in Hex1 are set to 1; otherwise, it returns 0.

Examples

@ANDH("A","F") = A

@ANDH("A") = 0

@ANDH("C","4",8) = 04

Related topics

@ARRAY - Array Formula

Syntax

@ARRAY(Expression, <Columns>, <Rows>)

Expression	Formula or @function using array syntax; @functions can be nested, that is, you can have more than one @function in a single statement.
Columns	Number of columns in the output range, including the column of the current cell (the default Columns depends on dimensions of input array(s) in Expression).
Rows	Number of rows in the output range, including the row of the current cell (the default Rows depends on dimensions of input array(s) in Expression).

@ARRAY returns the result of Expression, which can be either a formula with array operands or an @function with array arguments. An array is a selection of values treated as a single group. You do not need to type @ARRAY to create an array formula or @function; if an Expression requires array output, Quattro Pro converts it by surrounding it with the @ARRAY @function.

Columns and Rows are optional arguments; the size of an output array is dependent on the size of the input array(s) in Expression. Specify values for Columns and Rows only if you want to truncate or replicate portions of the output array.

By using arrays in formulas and @functions, you can perform an operation on multiple values or cells. You also save time by not having to repeat the same formula or @function in multiple cells. Arrays also save memory by reducing the number of formulas in a notebook.

If an array expression returns an array, the @ARRAY @function appears only in the current cell, which is also the upper left cell of the output array; the other cells in the array contain calculated values. Also, array formulas do not recognize 3-D syntax; if you specify a 3-D selection in Expression, @ARRAY recognizes only the cells on the first sheet of the series of consecutive sheets.

If you use many array formulas in a notebook, recalculation may become noticeably slower. Also, if you plan to share a notebook with other people, keep in mind that array formulas can make notebooks difficult to understand.

Examples

The next figure shows several examples using @ARRAY.

	A	B	C	D
1	B1=@ARRAY({1;2;3}*12)	12	24	36
2				
3	B3=@ARRAY({1|2|3}*12)	12		
4		24		
5		36		
6				
7	B7=@ARRAY(D7..D9*2)	16		8
8		20		10
9		24		12
10				
11	B11=@ARRAY(D7..D9/{4;5;6})	2	1.6	1.333333
12		2.5	2	1.666667
13		3	2.4	2
14				
15		1.23	-6.43	9
16	B16=@ARRAY(@ABS(B15..D15))	1.23	6.43	9

17			
18	B18=@ARRAY(@SQRT(C18..C20))	6.78233	46
19		7.348469	54
20		6	36
21			
22	B22=@ARRAY(@UPPER(C22..C25))	THIS	This
23		IS	is
24		A	a
25		TEST	Test

Related topics

@ASCTOHEX - Convert ASCII to Hexadecimal

Syntax

@ASCTOHEX(ASCII, <Places>)

ASCII	ASCII character string to convert; can be up to 20 ASCII characters.
Places	Number of characters to return; can be from 1 to 40 characters.

@ASCTOHEX returns the hexadecimal string equivalent of an ASCII number.

If the ASCII value includes nonnumeric characters, enclose it in quotation marks.

Examples

@ASCTOHEX("A") = 41

@ASCTOHEX("A",4) = 0041

@ASCTOHEX("01ABCDEF") = 3031414243444546

@ASCTOHEX("QUATTRO",5) = 4524F

Related topics

@ASEC - Arc Secant

Syntax

@ASEC(X)

X	The secant of an angle. X can be any value from approximately -1.789E+308 through -1 and from 1 through approximately 1.789E+308.

@ASEC calculates the arc, or inverse, secant using the secant X of an angle. The result of @ASEC is an angle, in radians, from 0 through π (from 0 through 180 degrees). To convert radians to degrees, use @DEGREES.

Examples

@ASEC(1) = 0

@ASEC(-2) = 2.094395

@DEGREES(@ASEC(-2)) = 120 degrees

@ASEC(0.25) = ERR, because X is between -1 and 1

Related topics

@ASECH - Arc Hyperbolic Secant

Syntax

@ASECH(X)

X	The hyperbolic secant of an angle. X must be greater than 0 and less than or equal to 1.

@ASECH calculates the arc, or inverse, hyperbolic secant using the hyperbolic secant X of an angle. The result is in radians; to convert to degrees, use @DEGREES.

Examples

@ASECH(@RADIANS(30)) = 1.263277

@ASECH(@RADIANS(75)) = ERR, because the hyperbolic secant of a 75-degree angle is between 0 and 1

@ASECH(@RADIANS(45)) = 0.723368

@ROUND(@DEGREES(@ASECH(0.624888)), 2) = 60 degrees. Or: "The angle whose hyperbolic secant is 0.624888, rounded to 2 decimal places."

Related topics

@ASIN - Arc Sine

Syntax

@ASIN(X)

X	A numeric value between -1 and 1.

@ASIN calculates the arc sine of X. The result is the angle (in radians) whose sine is X. To convert radians to degrees, use @DEGREES.

Examples

@ASIN(1) = 1.570796

@ASIN(0.25) = 0.25268

@DEGREES(@ASIN(0.5)) = 30

@ASIN(-2) = ERR (X is less than -1)

Related topics

@ASINH - Arc Hyperbolic Sine

Syntax

@ASINH(X)

X	The hyperbolic sine of an angle. X can be any value from approximately -1.34078E+154 to 1.34078E+154.

@ASINH calculates the arc, or inverse, hyperbolic sine using the hyperbolic sine X of an angle. The result is in radians; to convert to degrees, use @DEGREES.

Examples

@ASINH(1) = 0.881374

@ASINH(0.25) = 0.247466

@ROUND(@DEGREES(@ASINH(0.547853)), 2) = 30 degrees. Or: "The angle whose hyperbolic sine is 1.600287, rounded to 2 decimal places."

Related topics

@ATAN - Arc Tangent

Syntax

@ATAN(X)

X	A numeric value.

@ATAN calculates the arc tangent of X. The result is the angle (in radians) whose tangent is X. To convert radians to degrees, use @DEGREES.

Examples

@ATAN(0.5) = 0.463648

@ATAN(1) = 0.785398

@DEGREES(@ATAN(1)) = 45

Related topics

@ATAN2 - Arc Tangent of Two Points

Syntax

@ATAN2(X, Y)

X	A numeric value.
Y	A numeric value.

@ATAN2 calculates the arc tangent of the angle represented by the point with (x,y) coordinates X and Y. The result is the angle (in radians) whose tangent is Y/X. The result is between -pi and pi, with the quadrant chosen appropriately according to the sign of the result. If both X and Y are 0, the result is ERR.

The order of arguments is the same as for 1-2-3, but opposite that of the ATAN2 function in FORTRAN and other programming languages.

To convert radians to degrees, use @DEGREES.

Examples

@ATAN2(1,2) = 1.107149

@DEGREES(@ATAN2(1,1)) = 45

Related topics

@ATANH - Arc Hyperbolic Tangent

Syntax

@ATANH(X)

X	The hyperbolic tangent of an angle. X must be greater than -1 and less than 1.

@ATANH calculates the arc, or inverse, hyperbolic tangent using the hyperbolic tangent X. The result is in radians; to convert to degrees, use @DEGREES.

Examples

@ATANH(0.5) = 0.549306

@ATANH(1) = ERR, because X is not between -1 and 1

@ATANH(0.999999) = 7.254329

@ROUND(@DEGREES(@ATANH(0.655794)), 2) = 45 degrees. Or: "The angle whose hyperbolic tangent is 0.655794, rounded to 2 decimal places."

Related topics

@AVEDEV - Mean Absolute Deviation

Syntax

@AVEDEV(List)

List	One or more numeric or cell values.

@AVEDEV returns the mean absolute deviation, that is, the average of the absolute deviation of the data points in List from their mean. Use @AVEDEV to measure the variability of a data set around the mean. @AVEDEV uses this formula:

$$\mathrm{AVEDEV}(x_1..x_n) = \frac{1}{N}\sum_{j=1}^{N}\left|x_j - \bar{X}\right|$$

Example

@AVEDEV(10,11,12,13,12,11,10) = 0.897959

Related topics

@AVG - Average

Syntax

@AVG(List)

List	One or more numeric or cell values.

@AVG calculates the arithmetic mean of all values in List. It uses the formula: Sum of List divided by N

If List contains more than one item, they must be separated by commas. If any of the cells referenced contains ERR, the resulting value is ERR.

@AVG ignores blank cells when it makes its calculations. Cells containing labels, however, are treated as containing 0; thus assure that all blank cells are truly blank, and do not contain string operators such as apostrophes.

Examples

	B	**C**	**D**
1			
2	January	February	March
3	$652	$833	$599
4	$456	$305	$522
5	$68	$59	$73

@AVG(5,20,10,5) = 10

@AVG(B3..D3) = $694.67

@AVG(225,B3..D5) = $379.20

@AVG(B3..B5,PART) = $395.50 when C3..C5 is named PART

Related topics

@BASE - Convert Number to Another Base

Syntax

@BASE(Decimal, <Base>, <Precision>)

Decimal	Any decimal value to convert.
Base	Indicates the target base in which to express Decimal; can be any integer from 2 to 36, inclusive (the default is 16).
Precision	Indicates the number of desired digits after the decimal point; can be any integer from 0 to 15, inclusive (the default is 0).

@BASE converts a number from base-10 to a string value in a target base from 2 to 36.

Examples

@BASE(128,8) = 200

@BASE(123.47,16,6) = 7B.7851EB

Related topics

@BDAYS - Business Days

Syntax

@BDAYS(StartDate, EndDate, <Holidays>, <Saturday>, <Sunday>)

StartDate	Number representing the start date. See "Using dates and times in Quattro Pro."
EndDate	Number representing the end date.
Holidays	Cells containing dates that are holidays or the date of a single holiday or 0 to indicate no holidays (the default is 0).
Saturday	0 to specify that Saturday is not a business day; 1 to specify that Saturday is a business day (the default is 0).
Sunday	0 to specify that Sunday is not a business day; 1 to specify that Sunday is a business day (the default is 0).

@BDAYS returns the number of business days between StartDate and EndDate, including EndDate in the total. If EndDate is less than StartDate, the result is negative.

If neither StartDate nor EndDate falls on a weekend or holiday, @BDAYS returns the number of business days from StartDate to EndDate, including EndDate.

If StartDate and EndDate are two consecutive business days, the result is 1. If StartDate and EndDate both fall on weekends or holidays, @BDAYS returns the number of business days between the two dates, excluding EndDate.

If StartDate or EndDate (but not both) falls on a weekend or holiday, the result depends on which date falls on a business day. If StartDate falls on the weekend, the result includes EndDate. For example, if StartDate is a Saturday and EndDate is the following Thursday, the result includes the Thursday and is 4. If EndDate falls on the weekend, the result does not include EndDate. For example, if StartDate is a Thursday and EndDate is the following Saturday, the result is 1.

Examples

This formula calculates how many business days pass from November 30, 1993 to November 14, 1993, assuming that the dates in the cells A7..C9 are holidays:

@BDAYS(@DATE(93,11,30),@DATE(93,11,14),A7..C9) = -9

This formula calculates how many business days pass from June 2, 1993 to June 10, 1993, assuming no holidays other than weekends:

@BDAYS(@DATE(93,6,2),@DATE(93,6,10)) = 6

Related topics

@BESSELI - Modified Bessel I*n*(*x*)

Syntax

@BESSELI(x, n)

x	Numeric value at which to evaluate the function.
n	Number ≥ 0 representing the order of the Bessel function; if n is not an integer, it is truncated to an integer.

@BESSELI calculates the nth order modified Bessel function of the variable x. It uses this formula:

$$I_n(x) = (i)^{-n} J_n(ix)$$

@BESSELI is equivalent to the Bessel function J_n, but is evaluated for purely imaginary arguments.

Example

@BESSELI(1.5,0) = 1.646723

Related topics

@BESSELJ - Bessel J*n*(*x*)

Syntax

@BESSELJ(x, n)

x	Numeric value at which to evaluate the function.
n	Number ≥ 0 representing the order of the Bessel function; if n is not an integer, it is truncated to an integer.

@BESSELJ calculates the Bessel function J*n*(*x*). It uses this formula:

$$J_n(X)\sum_{k=0}^{\infty}\frac{(-1)^k}{k!\,\Gamma(n+k+1)}\left(\frac{x}{2}\right)^{n+2k}$$

, where

$$\Gamma(n+k+1)=\int_0^{\infty}e^{-x}x^{n+k}dx$$

is the gamma function.

Example

@BESSELJ(1.5,0) = 0.511828

Related topics

@BESSELK - Modified Bessel K*n*(*x*)

Syntax

@BESSELK(x, n)

x	Numeric value at which to evaluate the function; must be > 0.
n	Integer ≥ 0 representing the order of the Bessel function; if n is not an integer, it is truncated to an integer.

@BESSELK calculates the nth order modified Bessel function of the variable x. It uses this formula:

$$K_n(x) = \frac{\pi}{2} i^{n+1}\left[J_n(ix) + iY_n(ix)\right]$$

where Jn and Yn are @BESSELJ and @BESSELY, respectively.

Example

@BESSELK(1.5,0) = 0.213806

Related topics

@BESSELY - Bessel Y*n*(*x*)

Syntax

@BESSELY(x, n)

x	Non-negative numeric value at which to evaluate the function.
n	Integer ≥ 0 representing the order of the Bessel function; if n is not an integer, it is truncated to an integer.

@BESSELY calculates the Bessel function Y*n*(*x*) (also called the Neumann or Weber function). It uses this formula:

$$Y_n(x) = \lim_{v \to n} \frac{J_v(x)\cos(v\pi) - J_{-v}(x)}{\sin(v\pi)}$$

where:

$$J_{-n}(x) = (-1)^n J_n(x)$$

Example

@BESSELY(1.5,0) = 0.382449

Related topics

@BETA - Beta Function

Syntax

@BETA(Z, W)

Z	α parameter to the function; must be > 0.
W	β parameter to the function; must be > 0.

@BETA returns the value of the beta function, which is widely used in mathematics and statistics. @BETA uses this formula:

$$\beta(z, w) = \beta(w, z) = \int_0^1 t^{z-1}(1 - t)^{w-1}\,dt$$

Examples

@BETA(4,3) = 0.016667

@BETA(2,3) = 0.083333

@BETA(9,0.4) = 0.93348

@BETA(12,0.3) = 1.432072

Related topics

@BETADIST - Beta Distribution

Syntax

@BETADIST(X, Z, W, <A>, <B>)

X	Value at which to evaluate the function over the interval A ≤ X ≤ B.
Z	α distribution parameter; Z > 0.
W	β distribution parameter; W > 0.
A	Optional lower bound to the interval of X (the default is 0); A cannot equal B and must be ≤ X.
B	Optional upper bound to the interval of X (the default is 1); B cannot equal A and must be ≥ X.

@BETADIST returns the cumulative beta probability density function. The cumulative beta probability density function is a bounded distribution that is useful for studying variables such as percentages that may only take on values within a restricted range. The optional arguments A and B set those bounds.

Examples

@BETADIST(0.5,3,4,0,1) = 0.65625

@BETADIST(0.4,3,4,0,1) = 0.45568

Related topics

@BETAI - Incomplete Beta Function

Syntax

@BETAI(Z, W, X)

Z	α parameter to the function; if W = 0, Z > 0.
W	β parameter to the function; if Z = 0, W > 0.
X	Value at which to evaluate the function; cannot exceed 1.

@BETAI computes the incomplete beta function, that is, the probability that a standard beta-distributed variable will be less than X. @BETAI uses this formula:

$$I_x(z,w) = 1 - I_{1-x}(w,z) = \int_0^x t^{z-1}(1-t)^{w-1}\,dt$$

Examples

@BETAI(3,4,0.5) = 0.65625

@BETAI(3,4,0.1) = 0.01585

@BETAI(3,4,0.98) = 0.999998

@BETAI(7,8,0.7) = 0.968531

Related topics

@BETAINV - Inverse of Beta Distribution

Syntax

@BETAINV(Prob, Z, W, <A>, <B>)

Prob	Cumulative probability value; $0 \leq$ Prob ≤ 1.
Z	α parameter to the Beta distribution; must be > 0.
W	β parameter to the Beta distribution; must be > 0.
A	Optional lower bound to the interval of X (the default is 0); A cannot equal B and must be $\leq$ X.
B	Optional upper bound to the interval of X (the default is 1); B cannot equal A and must be $\geq$ X.

@BETAINV computes the inverse of the cumulative beta distribution function. If Prob = @BETADIST(X...), then @BETAINV(Prob...) = X.

Examples

@BETAINV(0.65625,3,4,0,1) = 0.5

@BETAINV(0.45568,3,4,0,1) = 0.4

Related topics

@BINOMDIST - Binomial Distribution

Syntax

@BINOMDIST(Successes, Trials, Prob, Cumulative)

Successes	Number of successes in number of trial runs; must be ≥ 0.
Trials	Number of independent trial runs in sample; must be > Successes.
Prob	Probability of a success on each trial run; must be ≥ 0 and ≤ 1.
Cumulative	1 to return the cumulative distribution function; 0 to return the probability that there are exactly Successes successes.

@BINOMDIST returns the binomial probability mass function, which is the probability that the number of successes in the independent trials equals Successes. Use @BINOMDIST when the outcome of experiments is success or failure, when the experiments are independent of one another, and when the probability of success does not change in successive trials. For example, a coin toss experiment is a binomial experiment.

Example

Using a random sample, a polling organization asks 50 voters if they favor Candidate A for reelection. Given that 55% of the city's voters favor Candidate A, this formula returns the probability that 40 people from the sample will favor her:

@BINOMDIST(40,50,.55,0) = 0.000144

Related topics

@BINTOHEX - Binary to Hexadecimal

Syntax

@BINTOHEX(Binary)

@BINTOHEX("1010") = A

Binary	Binary number to convert; denote negative numbers using a minus sign.

@BINTOHEX returns the hexadecimal string equivalent of a binary number.

Examples

@BINTOHEX("10000") = 10

@BINTOHEX("11110") = 1E

Related topics

@BINTOHEX64 -Binary to Hexadecimal

Syntax

@BINTOHEX64(Binary, <Places>)

Binary	Binary number to convert; must be positive.
Places	Number of characters to return; must be ≤ 16.

@BINTOHEX64 returns the hexadecimal string equivalent of a binary number (up to 64 bits).

Examples

@BINTOHEX64(1001) = 9
@BINTOHEX64(1010,2) = 0A
@BINTOHEX64("11110000001111000") = 1E078
@BINTOHEX64("11110000001111000",2) = 78

Related topics

@BINTONUM - Binary to Decimal

Syntax

@BINTONUM(Binary)

Binary	Binary number to convert; denote negative numbers using a minus sign.

@BINTONUM returns the decimal equivalent of a binary number.

Examples

@BINTONUM("1010") = 10

@BINTONUM("10000") = 16

@BINTONUM("11110") = 30

Related topics

@BINTONUM64 - Binary to Decimal

Syntax

@BINTONUM64(Binary, <Signed>)

Binary	Binary number to convert.
Signed	1 if the most significant bit of Binary is a sign bit; 0 (the default) if Binary is positive.

@BINTONUM64 returns the decimal equivalent of a binary number (up to 64 bits).

If Signed is 1, the most significant bit of Binary is the sign bit. If the sign bit is 0, the number is positive; if it is 1, the number is negative.

Examples

@BINTONUM64(100) = 4

@BINTONUM64(1010) = 10

@BINTONUM64("11110000001111000") = 123000

@BINTONUM64("11110000001111000",1) = -8072

Related topics

@BINTOOCT - Binary to Octal

Syntax

@BINTOOCT(Binary)

Binary	Binary number to convert; denote negative numbers using a minus sign.

@BINTOOCT returns the octal string equivalent of a binary number.

Examples

@BINTOOCT("1010") = 12

@BINTOOCT("10000") = 20

@BINTOOCT("11110") = 36

Related topics

@BINTOOCT64 - Binary to Octal

Syntax

@BINTOOCT64(Binary, <Places>)

Binary	Binary number to convert; must be positive.
Places	Number of characters to return; must be ≤ 22.

@BINTOOCT returns the octal string equivalent of a binary number (up to 64 bits).

Examples

@BINTOOCT64("0111") = 07
@BINTOOCT64("1000",3) = 010
@BINTOOCT64("11110000001111000") = 360170
@BINTOOCT64("11110000001111000",3) = 170
@BINTOOCT64("00000101001110010111011 1") = 01234567

Related topics

@BITRB - Binary Bit Reset

Syntax

@BITRB(Binary, Position)

Binary	Binary number.
Position	Bit position; must be ≥ 0 and ≤ number of bits in Binary - 1.

@BITRB resets to 0 the specified Position bit of a binary value.

Examples

@BITRB(1010,1) = 1000

@BITRB(1010,3) = 0010

@BITRB(1100,2) = 1000

Related topics

@BITRH - Hexadecimal Bit Reset

Syntax

@BITRH(Hex, Position)

Hex	Hexadecimal number.
Position	Bit position; must be ≥ 0 and ≤ number of bits in Hex - 1.

@BITRH resets to 0 the specified Position bit of a hexadecimal value.

Examples

@BITRH("A",1) = 8

@BITRH("A",3) = 2

@BITRH("C",2) = 8

Related topics

@BITSB - Binary Bit Set

Syntax

@BITSB(Binary, Position)

Binary	Binary number.
Position	Bit position; must be ≥ 0 and ≤ number of bits in Binary - 1.

@BITSB sets to 1 the specified Position bit of a binary number.

Examples

@BITSB(1010,2) = 1110

@BITSB(1010,0) = 1011

@BITSB(1100,0) = 1101

Related topics

@BITSH - Hexadecimal Bit Set

Syntax

@BITSH(Hex, Position)

Hex	Hexadecimal number.
Position	Bit position; must be ≥ 0 and ≤ number of bits in Hex - 1.

@BITSH sets to 1 the specified Position bit of a hexadecimal number.

Examples

@BITSH("A",2) = E

@BITSH("C",0) = D

Related topics

@BITTB - Binary Bit Test

Syntax

@BITTB(Binary, Position)

Binary	Binary number.
Position	Bit position; must be ≥ 0 and ≤ number of bits in Binary - 1.

@BITTB returns the value of the bit of a binary number in the specified Position.

Examples

@BITTB(1010,2) = 0

@BITTB(1001,0) = 1

@BITTB(1100,1) = 0

Related topics

@BITTH - Hexadecimal Bit Test

Syntax

@BITTH(Hex, Position)

Hex	Hexadecimal number.
Position	Bit position; must be ≥ 0 and ≤ number of bits in Hex - 1.

@BITTH returns the value of the bit of a hexadecimal number in the specified Position.

Examples

@BITTH("A",2) = 0

@BITTH("9",0) = 1

@BITTH("C",0) = 0

Related topics

@BLOCKNAME - Block Name

Syntax

@BLOCKNAME(Block)

Block	Cell or reference (for example, A1 or B1..B5).

@BLOCKNAME returns the name of a cell or selection specified by Block. If Block does not contain a name, @BLOCKNAME returns ERR; if Block contains more than one name, @BLOCKNAME arbitrarily returns one of the names.

If the name for a selection was created in another notebook, use @BLOCKNAME2.

Example

@BLOCKNAME(D2..D15) = SALES (selection D2..D15 is named SALES)

Related topics

@BLOCKNAME2 - Block Name in Specified Notebook

Syntax

@BLOCKNAME2(NotebookLink, Block)

NotebookLink	A reference to a sheet, cell, or cells in another notebook (for example, [BUDGET]A:A1).
Block	Cell or reference (for example, A1 or B1..B5).

@BLOCKNAME2 returns the cell name created in the notebook specified by NotebookLink that refers to Block, which can be in another notebook. If Block does not contain a name, @BLOCKNAME2 returns ERR; if Block contains more than one name, @BLOCKNAME2 arbitrarily returns one of the names.

Example

@BLOCKNAME2([BUDGET]A:A1,A:D2..D15) = SALES (selection A:D2..D15 of the active notebook is named SALES in the name table of notebook BUDGET)

Related topics

@BLOCKNAMES - Block Names

Syntax

@BLOCKNAMES(Block)

Block	Cell or reference (for example, A1 or B1..B5).

@BLOCKNAMES returns a two-column table showing the cell names that intersect with Block. The left column of the table contains cell names, and the right column contains corresponding coordinates.

If Block does not contain a name, @BLOCKNAMES returns ERR.

Because @BLOCKNAMES returns an array, it is automatically enclosed within an @ARRAY @function.

Make sure there is enough room for a two-column table, with one row for each cell name. Quattro Pro overwrites existing data in cells it uses for the table.

If cell names for a notebook were created in another notebook, use @BLOCKNAMES2.

Example

This example refers to cells in the next figure. Selections B3..B7, C3..C7, D3..D7, and B3..D7 are named HOTEL, TRANS, MEALS, and TOTAL, respectively. The example is entered in cell A12.

@ARRAY(@BLOCKNAMES(B3..D7)) = table in A12..B15 shown in the next figure.

	A	B	C	D
1	WEEKLY EXPENSE REPORT			
2	DATE	HOTEL	TRANS	MEALS
3	05/11	$99.70	$774.23	$67.34
4	05/12	$99.70	$15.00	$89.50
5	05/13	$99.70	$23.00	$97.78
6	05/14	$99.70	$13.00	$75.41
7	05/15	$99.70	$32.00	$63.20
8		$498.50	$857.23	$393.23
9				
10			TOTAL	$1,748.96
11				
12	HOTEL	[C:\COREL\QUATTRO\EXPENSES.QPW]A:B3..B7		
13	TRANS	[C:\COREL\QUATTRO\EXPENSES.QPW]A:C3..C7		
14	MEALS	[C:\COREL\QUATTRO\EXPENSES.QPW]A:D3..D7		
15	TOTAL	[C:\COREL\QUATTRO\EXPENSES.QPW]A:B3..D7		

Related topics

@BLOCKNAMES2 - Block Names in Specified Notebook

Syntax

@BLOCKNAMES2(NotebookLink, Block)

NotebookLink	A reference to a sheet, cell, or cells in another notebook (for example, [BUDGET]A:A1).
Block	Cell or reference (for example, A1 or B1..B5).

@BLOCKNAMES2 returns a two-column table showing the cell names created in the notebook specified by NotebookLink that refer to selections that intersect with Block. Use @BLOCKNAMES2 instead of @BLOCKNAMES if the cell names for a notebook were created in another notebook. The left column of the output table contains cell names, and the right column contains corresponding cell coordinates.

If Block does not contain a name, @BLOCKNAMES2 returns ERR.

Because @BLOCKNAMES2 returns an array, it is automatically enclosed within an @ARRAY @function.

Make sure there is enough room for a two-column table, with one row for each cell name. Quattro Pro overwrites existing data in cells it uses for the table.

Example

This example refers to cells in the next figure. Selections B3..B7, C3..C7, D3..D7, and B3..D7 in the active notebook EXPENSES are named HOTEL, TRANS, MEALS, and TOTAL, respectively, in the notebook TRAVEL. The example is entered in cell A12.

@ARRAY(@BLOCKNAMES2([TRAVEL]A:A1,B3..D7)) = table in A12..B15 shown in the next figure

	A	B	C	D
1	WEEKLY EXPENSE REPORT			
2	DATE	HOTEL	TRANS	MEALS
3	05/11	$99.70	$774.23	$67.34
4	05/12	$99.70	$15.00	$89.50
5	05/13	$99.70	$23.00	$97.78
6	05/14	$99.70	$13.00	$75.41
7	05/15	$99.70	$32.00	$63.20
8		$498.50	$857.23	$393.23
9				
10			TOTAL	$1,748.96
11				
12	HOTEL	[C:\COREL\QUATTRO\EXPENSES.QPW]A:B3..B7		
13	TRANS	[C:\COREL\QUATTRO\EXPENSES.QPW]A:C3..C7		
14	MEALS	[C:\COREL\QUATTRO\EXPENSES.QPW]A:D3..D7		
15	TOTAL	[C:\COREL\QUATTRO\EXPENSES.QPW]A:B3..D7		

Related topics

@BUSDAY - Closest Business Day

Syntax

@BUSDAY(Date, <Direction>, <Holidays>, <Saturday>, <Sunday>)

Date	Number representing a date. See "Using dates and times in Quattro Pro."
Direction	Flag specifying direction of adjustment; 0 = forward; 1 = backward; 2 = forward if in same month as Date, otherwise backward (the default is 0).
Holidays	Cells containing dates that are holidays or the date of a single holiday or 0 to indicate no holidays (the default is 0).
Saturday	0 to specify that Saturday is not a business day; 1 to specify that Saturday is a business day (the default is 0).
Sunday	0 to specify that Sunday is not a business day; 1 to specify that Sunday is a business day (the default is 0).

@BUSDAY returns Date if it is a valid business day. If Date falls on a Saturday (and Saturday is set to 0 or omitted), Sunday (and Sunday is set to 0 or omitted), or holiday, @BUSDAY returns the date of the closest valid business day in the direction specified by Direction.

Example

This formula calculates the closest business day after December 25, 1993, assuming that the 25th is a holiday:

@BUSDAY(@DATE(93,12,25),0,@DATE(93,12,25)) = 34330 (December 27, 1993)

Related topics

@CATB - Concatenate Binary

Syntax

@CATB(Binary1, <HiBit1>, <LoBit1>, <Binary2>, <HiBit2>, <LoBit2>, <Bits>)

Binary1	First binary number.
HiBit1	Highest bit of the first number to use for concatenation; the default is the most significant bit.
LoBit1	Lowest bit of the first number to use for concatenation; the default is 0.
Binary2	Second binary number.
HiBit2	Highest bit of the second number to use for concatenation; the default is the most significant bit.
LoBit2	Lowest bit of the second number to use for concatenation; the default is 0.
Bits	Number of binary digits to return; must be in the range $0 < n \leq 64$.

@CATB joins together two specified binary numbers or extracts selected bits from one binary number. Specify high bit and low bit values if you want to use only a portion of a number for concatenation. For example, if HiBit1 = 2 and LoBit1 = 0, only the first three bits of Binary1 are joined with Binary2.

Examples

@CATB("1100",2,0,"0011",1,0) = 10011

@CATB("1100",2,0,"0011",1,0,3) = 011

@CATB("1100",3,0,"11",1,0,8) = 00110011

@CATB("101101",4,1) = 0110

Related topics

@CATH - Concatenate Hexadecimal

Syntax

@CATH(Hex1, <HiBit1>, <LoBit1>, <Hex2>, <HiBit2>, <LoBit2>, <Bits>)

Hex1	First hexadecimal number.
HiBit1	Highest bit of the first number to use for concatenation; the default is the most significant bit.
LoBit1	Lowest bit of the first number to use for concatenation; the default is 0.
Hex2	Second hexadecimal number.
HiBit2	Highest bit of the second number to use for concatenation; the default is the most significant bit.
LoBit2	Lowest bit of the second number to use for concatenation; the default is 0.
Bits	Number of equivalent binary digits to return; 4 binary digits = 1 hexadecimal digit; must be in the range 0 <n ≤ 64.

@CATH joins together two specified hexadecimal numbers or extracts selected bits from one hexadecimal number. Specify high bit and low bit values if you want to use only a portion of a number for concatenation. For example, if HiBit1 = 2 and LoBit1 = 0, only the first three bits of Hex1 are joined with Hex2.

Examples

@CATH("C",2,0,"3",1,0) = 13

@CATH("C",2,0,"3",1,0,3) = 3

@CATH("C",3,0,"3",1,0,8) = 33

@CATH("CA",6,2) = 12

Related topics

@CATNB - Concatenate *n* Binary

Syntax

@CATNB(n, Binary1, <Binary2>, <Binary3>, ..., <BinaryN>, <Bits>)

n	Number of binary numbers being concatenated; n ≤ 64.
Binary1	First binary number.
Binary2,Binary3, . . . , BinaryN	Second through the nth binary numbers.
Bits	Number of binary digits to return; must be in the range 0 <n ≤ 64.

@CATNB joins together n binary numbers.

Examples

@CATNB(3,1,0,1010) = 101010

@CATNB(3,1,0,1010,4) = 1010

@CATNB(3,11,"00",11,8) = 00110011

Related topics

@CATNH - Concatenate *n* Hexadecimal

Syntax

@CATNH(n, Hex1, <Hex2>, <Hex3>, ..., <HexN>, <Bits>)

n	Number of hexadecimal numbers being concatenated; n ≤ 16.
Hex1	First hexadecimal number.
Hex2,Hex3, . . . , HexN	Second through the nth hexadecimal numbers.
Bits	Number of equivalent binary digits to return; 4 binary digits = 1 hexadecimal digit; must be in the range 0 <n ≤ 64.

@CATNH joins together n hexadecimal numbers.

Examples

@CATNH(3,"1","0","A") = 10A

@CATNH(3,"1","0","A",4) = A

@CATNH(3,"A","B","C",16) = 0ABC

Related topics

@CDAYS - Calendar Days

Syntax

@CDAYS(StartDate, EndDate, <Calendar>, <February>)

StartDate	Number representing the start date. See "Using dates and times in Quattro Pro."
EndDate	Number representing the end date.
Calendar	Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).
February	0 to use 30-day treatment of February for 30/360 calendar; 1 to use the actual-day treatment (the default is 0).

@CDAYS returns the number of calendar days between StartDate and EndDate, including EndDate in the total. If EndDate precedes StartDate, the result is negative.

You can use Calendar to specify whether the actual or 30/360 day calendar is used. Under the actual calendar, Quattro Pro calculates the number of days by subtracting one date from the other.

To handle months with more than 30 days (and February), @CDAYS sometimes adjusts StartDate or EndDate before the sum is calculated. @CDAYS adjusts StartDate to fall on the 30th if either of the following two conditions are true: the day of the month on which StartDate falls is greater than 30, or StartDate falls on the last day of February (28th or 29th, depending on year) and February is 0.

If StartDate falls on the 30th (either because @CDAYS adjusted it or it falls on the 30th) and the day of the month on which EndDate falls is greater than 30, then EndDate also adjusts to fall on the 30th. By default, @CDAYS treats the last day of February as the 30th. To prevent this, set February to 1.

Example

@CDAYS(@DATE(93,1,23),@DATE(95,6,28)) = 875

Related topics

@CELL - Cell Attribute

Syntax

@CELL(Attribute, Block)

Attribute	Any one of the attributes listed for @CELL.
Block	A cell reference or name.

@CELL returns the requested attribute of the upper left cell in Block. For details on types of attributes, see Attribute Arguments.

If you type in or point to a single-cell address when entering Block, Quattro Pro converts it to a cell reference.

You can enter attributes in either upper- or lowercase, but you must surround them with double quotes. You can also reference a cell containing an attribute.

@CELL does not recalculate automatically; press F9 to obtain the current value.

Examples

	A	B	C	D
1				
2		January	February	March
3	Advertising	$652	$833	$599
4	Car expenses	$456	$305	$522
5	Cleaning	$80	$80	$80

@CELL("prefix",A3) = '

@CELL("format",B5) = C0

@CELL("type",D4) = v

@CELL("address",A3) = $A$3

@CELL("row",B4) = 4

Related topics

Attribute Arguments

You may enter any of these as Attribute arguments for @CELL, @CELLINDEX, and @CELLPOINTER:

"address"

The address of the upper left cell in Block.

"row"

The row number of the upper left cell in Block (1 to 8192).

"col"

The column number of the upper left cell in Block (1 to 256, corresponding to notebook sheets A through IV).

"sheet"

Sheet number of the upper left cell in Block (1 to 256, corresponding to notebook sheets A through IV).

"NotebookName"

Referenced notebook name, 8 characters or fewer.

"NotebookPath"

Full path name of the referenced notebook.

"TwoDAddress"

2-D address of the referenced cell--$G$23, for example. The sheet name is never returned, even if the referenced cell is on another sheet or in another notebook.

"ThreeDAddress"

3-D address of the referenced cell--$A:$G23, for example. The sheet name is always returned.

"FullAddress"

Full address of the referenced cell--[NOTEBK1]$A:$G$23, for example. The notebook and sheet names are always returned.

"contents"

The contents of the upper left cell in Block.

"type"

The type of data in the upper left cell in Block: b if the cell is blank, v if the cell contains a number or any formula, l if the cell contains a label.

"prefix"

The label-prefix character of the upper left cell in Block: ' if label is left-aligned, ^ if label is centered, " if label is right-aligned, \ if label is repeating.

"protect"

The protected status of the upper left cell in Block: 0 if cell is not protected, 1 if cell is protected.

"width"

The width of the column containing the upper left cell in Block (between 1 and 1024).

"rwidth"

The width of the cells.

"format"

The numeric format code of the upper left cell in Block:

Fn	Fixed (n = 0-15)
Sn	Scientific (n = 0-15)
Cn	Currency (n = 0-15)

,n	Commas used to separate thousands (n = 0-15)
G	General
+	+/- (bar chart format)
Pn	Percent (n = 0-15)
D1-D5	Date
	D1 = DD-MMM-YY
	D2 = DD-MMM
	D3 = MMM-YY
	D4 = MM/DD/YY, DD/MM/YY, DD.MM.YY, YY-MM-DD
	D5 = MM/DD, DD/MM, DD.MM, MM-DD
D6-D9	Time
	D6 = HH:MM:SS AM/PM
	D7 = HH:MM AM/PM
	D8 = HH:MM:SS-24hr, HH.MM.SS-24hr, HH,MM,SS-24hr, HHhMMmSSs
	D9 = HH:MM-24hr, HH.MM-24hr, HH,MM, HHhMMm.
T	Show Formulas (Text)
H	Hidden
U	User-defined

Related topics

@CELLINDEX - Cell Attribute of Table Index

Syntax

@CELLINDEX(Attribute, Block, Col, Row, <Page>)

Attribute	Any one of the attributes listed for @CELL.
Block	A cell reference or name.
Col	The number of the referenced column, from 0 to 255 (the first column in Block = 0, the second = 1, and so on).
Row	The number of the referenced row; if an offset, the first row in Block = 0, the second = 1, and so on.
Page	The number of the referenced sheet, from 0 to 255 (the first sheet in Block = 0, the second = 1, and so on).

@CELLINDEX is the same as @CELL, but returns the requested attribute of the cell in the specified column and row of Block on optional sheet. (For details on types of attributes, see Attribute Arguments.) The upper left corner of Block is column 0, row 0.

@CELLINDEX does not recalculate automatically. Press F9 to obtain the current value.

Examples

	A	B	C	D
1				
2		January	February	March
3	Advertising	$652	$833	$599
4	Car expenses	$456	$305	$522
5	Cleaning	$80	$80	$80

@CELLINDEX("prefix",A1..D5,0,2) = '

@CELLINDEX("format",B3..D5,0,2) = C0

@CELLINDEX("type",B3..D5,2,1) = v

@CELLINDEX("address",A1..D5,0,2) = $A$3

@CELLINDEX("row",A1..D5,1,3) = 4

Related topics

@CELLPOINTER - Selected Cell Attribute

Syntax

@CELLPOINTER(Attribute)

Attribute	Any one of the attributes listed for @CELL.

@CELLPOINTER is similar to @CELL in that it returns the requested attribute of a cell. The only difference is that it reads the cell containing the selector. You cannot specify another cell. However, if you move the selector to a different cell and then press F9, the results of the @CELLPOINTER formula are updated.

You can enter attribute names in either upper- or lowercase, but each must be enclosed by double quotes. For details on types of attributes, see Attribute Arguments.

This @function is useful in macros and @IF statements for quickly determining certain characteristics about the current cell, such as whether there is a label or a value currently in it. For example, this function statement tells Quattro Pro to write "value" in the cell if the current cell is a value; otherwise, it writes "label":

@IF(@CELLPOINTER("type")="v","value","label")

Examples

These examples refer to cell A1, which contains the date value 11/19/91.

@CELLPOINTER("address") = $A$1

@CELLPOINTER("col") = 1

@CELLPOINTER("contents") = 33561

@CELLPOINTER("format") = D4

@CELLPOINTER("type") = v

Related topics

@CEILING - Round Up to Nearest Multiple

Syntax

@CEILING(X, Y)

X	Value to round.
Y	Value to make rounded x evenly divisible by.

@CEILING rounds X up (away from zero) to the nearest value that is evenly divisible by Y. If X and Y have different signs, the result of @CEILING is ERR.

Examples

@CEILING(22,5) = 25

@CEILING(5.7,0.2) = 5.8

@CEILING(-3.2,-2) = -4

@CEILING(-3.2,2) = ERR

Related topics

@CHAR - ANSI or ASCII Character

Syntax

@CHAR(Code)

Code	A numeric value between 1 and 255.

@CHAR returns the onscreen character corresponding to the given code. This is useful in generating symbols not found on the keyboard.

Refer to any standard ANSI table for the codes corresponding to each character.

@CHAR can be used to set up an ANSI table in your notebook. Fill a column of cells with values from 1 to 255, using Block|Fill. In the cell to the right of the first number, use @CHAR to show the screen character for 1, for example, @CHAR(A1). Then copy the formula down the column for the next 255 cells. The copied formulas will display the screen character for each number. (The first 128 will be the same characters as in the ASCII character set.)

Examples

@CHAR(33) = !

@CHAR(34) = "

@CHAR(35) = #

@CHAR(36) = $

Related topics

@CHIDIST - Chi-squared Distribution

Syntax

@CHIDIST(X, DegFreedom)

X	Value at which to evaluate the function; must be ≥ 0.
DegFreedom	Integer number of degrees of freedom in the distribution; must be ≥ 1.

@CHIDIST returns the cumulative chi-square distribution, which is associated with a chi-square test. Chi-square tests allow you to compare the differences between observed and expected frequencies.

If DegFreedom is not an integer, @CHIDIST rounds it to the nearest integer.

Examples

@CHIDIST(36.41503,24) = 0.05

@CHIDIST(17.53455,8) = 0.025

Related topics

@CHIINV - Inverse of Chi-squared Distribution

Syntax

@CHIINV(Prob, DegFreedom)

Prob	Cumulative probability value; must be ≥ 0 and ≤ 1.
DegFreedom	Integer number of degrees of freedom; must be ≥ 1.

@CHIINV computes the inverse of the cumulative one-tailed chi-square distribution. Use @CHIINV to compute the critical value for a test involving a chi-square variable.

If DegFreedom is not an integer, @CHIINV rounds it to the nearest integer.

Examples

@CHIINV(0.05,24) = 36.41503

@CHIINV(0.025,8) = 17.53455

Related topics

@CHITEST - Test for Independence

Syntax

@CHITEST(Actual, Expected)

Actual	Cells containing actual values.
Expected	Cells containing expected values.

@CHITEST computes the probability that the actual and expected frequencies are similar by chance. @CHITEST returns the probability for a chi-square test distribution with (r - 1)(c - 1) degrees of freedom, where r = number of rows, and c = number of columns.

Actual and Expected must have the same number of values and must contain multiple rows or columns of data.

Example

This example refers to cells in the next figure. The chi-square statistic for the data in the next figure is 16.25813 and the degrees of freedom is 4.

@CHITEST(C3..E5,C7..E9) = 0.002692

	A	B	C	D	E
1		Soft Drink Flavors			
2		Age Ranges	Cola	Orange	Lemon-lime
3	Actual	Under 25	120	65	55
4		26-50	100	45	85
5		Over 50	75	35	70
6					
7	Expected	Under 25	108.93	53.53	77.54
8		26-50	104.38	51.31	74.31
9		Over 50	81.69	40.16	58.15

Related topics

@CHOOSE - Choose Value from List

Syntax

@CHOOSE(Number, List)

Number	A positive integer equal to or less than the number of items in List - 1.
List	One or more numeric or string values, cell addresses, and cell references or names, separated by commas.

@CHOOSE selects and enters a value from the supplied list. The value it chooses depends on the value of Number: 0 chooses the first value in the list; 1 chooses the second; 2 chooses the third, and so on. If you specify a cell address for Number, Quattro Pro uses the number contained in the cell. If the cell is blank, the first value is chosen.

The List values can be cell addresses, strings, numbers, or a mixture of the three. The total characters entered cannot exceed 1024.

@CHOOSE operates on integers only. If you supply a non-integer (such as 1.6433), the decimal values are disregarded. @VLOOKUP and @HLOOKUP perform similar tasks in tables.

Examples

@CHOOSE(0,"Howie","Sarah","Chris") = Howie

@CHOOSE(1,"Howie","Sarah","Chris") = Sarah

@CHOOSE(2,"Howie","Sarah","Chris") = Chris

@CHOOSE(A15,"Howie","Sarah","Chris") = Howie, if A15 is 0; Sarah if A15 is 1; Chris if A15 is 2.

@CHOOSE(3,"Howie","Sarah","Chris") = ERR (Number is too large).

@CHOOSE(@MOD(@NOW,7),"Saturday","Sunday","Monday","Tuesday","Wednesday","Thursday","Friday") = Wednesday when @NOW has DateTimeNumber = 33625

Related topics

@CLEAN - Remove Nonprintable Characters

Syntax

@CLEAN(String)

String	A string value.

@CLEAN removes all nonprintable characters (0-31) from a string.

Related topics

@CODE - ANSI Code

Syntax

@CODE(String)

String	A string value.

@CODE returns the ANSI code of the first character in a string. This is the opposite of @CHAR, which returns the character corresponding to the given code.

Examples

@CODE("!") = 33

@CODE("Sam") = 83 (code for S)

@CODE("#") = 35

@CODE("$") = 36

@CODE("?") = 63

@CODE(hello) = syntax error (missing quotes)

Related topics

@COLS - Columns

Syntax

@COLS(Block)

Block	A cell reference or name.

@COLS returns the number of columns within the specified cells.

Examples

@COLS(A1..IV1) = 256

@COLS(A1..A1) = 1

@COLS(NAME) = 30 (if the NAME selection contains 30 columns)

Related topics

@COLUMN - Column Number

Syntax

@COLUMN(<Block>)

Block	The cell or cells for which you want the column number(s).

@COLUMN returns the column number(s) for a cell or cells.

Block can be a cell name.

If you omit Block, Quattro Pro assumes you want the column number of the cell where you entered @COLUMN.

Block cannot refer to non-contiguous areas.

Examples

@COLUMN(C1..C7) = 3

@COLUMN(K1..M20) = {11| 12| 13}

If F2..F7 is a cell named APRIL, @COLUMN(APRIL) = 6

Entered in D3 without an argument, @COLUMN = 4

Related topics

@COMB - Combinations

Syntax

@COMB(N, R)

N	Number of elements in the group; must be ≥ 0.
R	Number of elements in each subgroup selected from group N; must be ≥ 0.

@COMB calculates the number of combinations (unordered subgroups of size R) that you can form out of a group of size N. If N < R @COMB returns ERR.

The formula for calculating the number of combinations, if R ≤ N, is

$$@COMB(N,R) = \frac{N!}{R!(N-R)!}$$

Example

Given eleven marbles, this formula calculates how many ways a subset of 5 marbles can be constructed such that no two constructions contain the same 5 marbles:

@COMB(11,5)= 462

Related topics

@COMMAND - Value of Command Equivalent

Syntax

@COMMAND(CommandEquivalent)

CommandEquivalent	A Quattro Pro command equivalent; to display a list, press Shift+F3 and choose Command Equivalents.

@COMMAND returns the current value of a Quattro Pro command equivalent. It is most often used in macros to base the next action on a particular menu setting or to save current settings so they can be restored later.

CommandEquivalent must be enclosed in double quotes. To view a list of acceptable arguments, press Shift+F3 and choose Command Equivalents.

@COMMAND returns strings; even if the setting is a number, it is returned as a string. Not all CommandEquivalent entries return a useful value. In general, @COMMAND only returns values for command equivalents that take arguments, usually menu commands that display a current setting or status.

Like @CELL, @COMMAND statements do not recalculate automatically as many other @functions do. Press F9 to obtain the current value.

A related @function that uses Quattro Pro for DOS menu equivalents is @CURVALUE. Another related @function, @PROPERTY, returns settings for requested object properties.

Examples

@COMMAND("Print.Block") = the currently specified print selection

@COMMAND("Print.Copies") = the number entered after Copies in the Spreadsheet Print dialog box

Related topics

@COMPLEX - Complex Number

Syntax

@COMPLEX(X, Y)

X	Numeric value representing real coefficient of complex number.
Y	Numeric value representing imaginary coefficient of complex number.

@COMPLEX converts X and Y into a complex number.

Example

@COMPLEX(5,7) = "5+7i"

Related topics

@CONCATENATE - Link Text Items

Syntax

@CONCATENATE(List)

List	One or more values to link together; can be labels, numbers, or cell references.

@CONCATENATE links several items together.

You can also use the "&" operator instead of @CONCATENATE to join text items.

Examples

Suppose you have a database where dates are stored in three fields:

	F	G	H
1	Day	Month	Year
2	31	October	1995

@CONCATENATE(G2," ",F2,", ",H2) = October 31, 1995

	A	B	C
1	First	Init.	Last
2	John	K.	Doe

If parts of names are always entered into cells with a space following, you can concatenate first name, initial, and last name by simply referring to the cells: @CONCATENATE(A2..C2) = John K. Doe

@CONCATENATE("Lucky ","Day") = Lucky Day

(+"Lucky"&" "&"Day") = Lucky Day

Related topics

@CONFIDENCE - Confidence Interval for Population Mean

Syntax

@CONFIDENCE(Alpha, SDev, Size)

Alpha	Significance level; the percentage of the normal curve that is outside the confidence interval (1 - Alpha); for example, if the confidence interval is 95%, Alpha = 5%; must be > 0 and < 1.
SDev	Population standard deviation; must be > 0.
Size	Sample size; must be ≥ 1.

@CONFIDENCE computes the confidence interval around the mean for a specified sample size, using the normal distribution function. Given a specified degree of confidence, the confidence interval indicates that the population mean will be within that interval. Use @CONFIDENCE to apply levels of confidence to sample data and to determine margins of error.

Example

Out of 1000 people sampled, 490 said they would vote for Candidate A. If the population standard deviation is 0.5, this formula returns the 95% confidence interval for the population mean:

@CONFIDENCE(0.05,0.5,1000) = 0.03099

Pollsters can report that Candidate A will receive 49% of the vote with a margin of error of 3.1%.

Related topics

@CONVERT - Convert Number

Syntax

@CONVERT(X, FromUnit, ToUnit)

X	Numeric value in FromUnit to convert, in the units specified by FromUnit.
FromUnit	Unit type of the value X (must be on the list of supported unit names).
ToUnit	Units to convert the value X into; must be on the list of supported unit names.

@CONVERT changes X, which is expressed in FromUnit units, to the equivalent value in ToUnit units. Column Unit of the following tables lists the measurement units that you can specify in FromUnit and ToUnit. Each argument is case sensitive.

Mass measurement units

Mass	Unit
Gram	"g"
Slug	"sg"
Pound mass (avoirdupois)	"lbm"
U (atomic mass unit)	"u"
Ounce mass (avoirdupois)	"ozm"

Pressure measurement units

Pressure	Unit
Pascal	"p"
Atmosphere	"at"

Distance measurement units

Distance	Unit
Meter	"m"
Statute mile	"mi"
Nautical mile	"Nmi"
Inch	"in"
Foot	"ft"
Yard	"yd"
Angstrom	"ang"

Time measurement units

Time	Unit
Year	"yr"
Day	"day"
Hour	"hr"
Minute	"mn"
Second	"sec"

Force measurement units

Force	Unit
Newton	"N"
Dyne	"dy"
Pound force	"lbf"

Energy measurement units

Energy	Unit
Joule	"J"
Erg	"e"
Thermodynamic calorie	"c"
IT calorie	"cal"
Electron volt	"ev"
Horsepower-hour	"hh"
Watt-hour	"wh"
Foot-pound	"flb"
BTU	"btu"

Power measurement units

Power	Unit
Horsepower	"h"
Watt	"w"

Magnetic measurement units

Magnetism	Unit
Tesla	"T"
Gauss	"ga"

Temperature measurement units

Temperature	Unit
Celsius	"cel"
Fahrenheit	"fah"
Kelvin	"kel"
Rankine	"ran"

Liquid measurement units

Liquid	Unit
Teaspoon	"tsp"
Tablespoon	"tbs"
Fluid ounce	"oz"
Cup	"cup"
Pint	"pt"
Quart	"qt"
Gallon	"gal"
Liter	"lt"

If a metric unit is used (for example, Gram, Meter, or Liter), you can preface it with one of the prefixes listed in the next table. Use the metric prefixes to multiply a metric unit by a power of 10.

Metric Prefixes

Metric Prefix	Multiplier	Unit Prefix
exa	1E+18	"E"
peta	1E+15	"P"
tera	1E+12	"T"
giga	1E+09	"G"
mega	1E+06	"M"
kilo	1E+03	"k"
deka	1E+01	"e"
deci	1E-01	"d"
centi	1E-02	"c"
milli	1E-03	"m"
micro	1E-06	"u"
nano	1E-09	"n"
pico	1E-12	"p"
femto	1E-15	"f"
atto	1E-18	"a"

Whenever @CONVERT is used, both FromUnit and ToUnit must come from the same table.
To determine a specific conversion factor, use 1 for X.

Examples

@CONVERT(2,"day","hr") = 48
@CONVERT(3.5,"kg","lbm") = 7.71618
@CONVERT(2.5,"oz","mlt") = 73.94991

Related topics

@CORREL - Correlation

Syntax

@CORREL(Array1, Array2)

Array1	First array of numeric values.
Array2	Second array of numeric values.

@CORREL computes the correlation coefficient of the numeric values in Array1 and Array2. Use @CORREL to ascertain the relationship between two sets of data. If two data sets change in a related matter based on the input that generates them, they are said to be correlated.

Array1 and Array2 must have the same number of values. Also, the values in Array1 and Array2 must show some variance.

@CORREL uses this formula:

$$r = \frac{n\left(\sum XY\right) - \left(\sum X\right)\left(\sum Y\right)}{\sqrt{\left[n\sum X^2 - \left(\sum X\right)^2\right]\left[n\sum Y^2 - \left(\sum Y\right)^2\right]}}$$

Examples

These examples refer to cells in the next figure.

@CORREL(A2..A9,B2..B9) = 0.994135

@CORREL(A2..A9,C2..C9) = 0.460718

@CORREL(A2..A9,D2..D9) = -0.52494

@CORREL(B2..B9,C2..C9) = 0.547422

	A	B	C	D
1	X1	X2	X3	X4
2	1	2	-1	-9
3	2	3	-7	-3
4	3	4	2	6
5	4	5	8	3
6	5	6	-4	-2
7	6	7	0	-21
8	7	8	-12	0
9	8	10	45	-33

Related topics

@COS - Cosine

Syntax

@COS(X)

X	A numeric value.

@COS returns the cosine of the angle X. X must be given in radians, not degrees. To convert degrees to radians, use @RADIANS.

Examples

@COS(@RADIANS(60)) = 0.5

@COS(@RADIANS(75)) = 0.258819

@COS(@RADIANS(45)) = 0.707107

@COS(@PI/3) = 0.5

Related topics

@COSH - Hyperbolic Cosine

Syntax

@COSH(X)

X	Any value from approximately -710.47558 to approximately 710.47558.

@COSH calculates the hyperbolic cosine of the angle X. X must be specified in radians, not degrees. To convert degrees to radians, use @RADIANS.

@COSH returns a value greater than or equal to 1.

Examples

@COSH(@RADIANS(60)) = 1.600287

@COSH(@RADIANS(75)) = 1.986274

@COSH(@RADIANS(45)) = 1.324609

@COSH(@PI/3) = 1.600287

Related topics

@COT - Cotangent

Syntax

@COT(X)

X	An angle measured in radians. X can be any value from approximately -9.00719E+15 through 9.00719E+15.

@COT calculates the cotangent of angle X. X must be specified in radians, not degrees. To convert degrees to radians, use @RADIANS.

In a right triangle, the cotangent of an acute angle is the ratio side adjacent **:** side opposite.

Examples

@COT(4) = 0.863691

@COT(@PI/4) = 1

@COT(@RADIANS(45)) = 1

Related topics

@COTH - Hyperbolic Cotangent

Syntax

@COTH(X)

X	Any value from approximately -708.39599 through 708.39599, but not 0.

@COTH calculates the hyperbolic cotangent of X. X must be specified in radians, not degrees. To convert degrees to radians, use @RADIANS.

Examples

@COTH(4) = 1.000671

@COTH(@PI/4) = 1.524869

@COTH(@RADIANS(45)) = 1.524869

Related topics

@COUPDAYBS - Coupon Days from Beginning to Settlement

Syntax

@COUPDAYBS(Settle, Maturity, <Freq>, <Calendar>)

Settle	Number representing the settlement date; must be < Maturity.
Maturity	Number representing the maturity date.
Freq	Frequency of coupon payments in number of payments per year (can be 1, 2, 3, 4, 6, or 12; the default is 2).
Calendar	Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

@COUPDAYBS returns the number of days from the beginning of the coupon period of a bond to the settlement date.

Example

A bond's settlement date is May 15, 1992 and its maturity date is February 15, 1996. This formula calculates the number of days from the beginning of the coupon period to the settlement date:

@COUPDAYBS(@DATE(92,5,15),@DATE(96,2,15)) = 90

Related topics

@COUPDAYS - Coupon Days

Syntax

@COUPDAYS(Settle, Maturity, <Freq>, <Calendar>)

Settle	Number representing the settlement date; must be < Maturity.
Maturity	Number representing the maturity date.
Freq	Frequency of coupon payments in number of payments per year (can be 1, 2, 3, 4, 6, or 12; the default is 2).
Calendar	Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

@COUPDAYS returns the number of days in the coupon period of a bond that contains the settlement date.

Example

A bond's settlement date is May 15, 1992 and its maturity date is February 15, 1996. This formula calculates the number of days in the coupon period that contains the settlement date.

@COUPDAYS(@DATE(92,5,15),@DATE(96,2,15)) = 180

Related topics

@COUPDAYSNC - Coupon Days from Settlement to Next Coupon

Syntax

@COUPDAYSNC(Settle, Maturity, <Freq>, <Calendar>)

Settle	Number representing the settlement date; must be < Maturity.
Maturity	Number representing the maturity date.
Freq	Frequency of coupon payments in number of payments per year (can be 1, 2, 3, 4, 6, or 12; the default is 2).
Calendar	Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

@COUPDAYSNC returns the number of days from the settlement date of a bond to the next coupon date.

Example

A bond's settlement date is May 15, 1992 and its maturity date is February 15, 1996. This formula calculates the number of days between the settlement date and the next coupon date:

@COUPDAYSNC(@DATE(92,5,15),@DATE(96,2,15)) = 90

Related topics

@COUPNCD - Next Coupon Date after Settlement

Syntax

@COUPNCD(Settle, Maturity, <Freq>, <Calendar>)

Settle	Number representing the settlement date; must be < Maturity.
Maturity	Number representing the maturity date.
Freq	Frequency of coupon payments in number of payments per year (can be 1, 2, 3, 4, 6, or 12; the default is 2).
Calendar	Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

@COUPNCD returns the serial date number for the next coupon date after the settlement date of a bond.

Example

A bond pays a coupon semiannually and matures on August 31, 2003. This formula calculates the date of the next coupon payment after December 17, 1992:

@COUPNCD(@DATE(92,12,17),@DATE(103,8,31)) = 34028 (February 28, 1993)

Related topics

@COUPNUM - Number of Coupons

Syntax

@COUPNUM(Settle, Maturity, <Freq>, <Calendar>)

Settle	Number representing the settlement date; must be < Maturity.
Maturity	Number representing the maturity date.
Freq	Frequency of coupon payments in number of payments per year (can be 1, 2, 3, 4, 6, or 12; the default is 2).
Calendar	Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

@COUPNUM returns the number of coupons payable between the settlement date and maturity date of a bond.

Example

A bond's settlement date is March 15, 1994 and its maturity date is April 15, 2004. This formula calculates the number of annual coupon payments there are until the maturity date:

@COUPNUM(@DATE(94,3,15),@DATE(104,4,15),1) = 11

Related topics

@COUPPCD - Previous Coupon Date

Syntax

@COUPPCD(Settle, Maturity, <Freq>, <Calendar>)

Settle	Number representing the settlement date; must be < Maturity.
Maturity	Number representing the maturity date.
Freq	Frequency of coupon payments in number of payments per year (can be 1, 2, 3, 4, 6, or 12; the default is 2).
Calendar	Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

@COUPPCD returns the serial date number for the coupon date just before the settlement date of a bond.

Example

A bond's settlement date is December 17, 1992 and its maturity date is August 31, 1999. This formula calculates the date of the previous semiannual coupon payment before the settlement date:

@COUPPCD(@DATE(92,12,17),@DATE(99,8,31)) = 33847 (August 31, 1992)

Related topics

@COVAR - Covariance (Population Covariance)

Syntax

@COVAR(Array1, Array2)

Array1	First array of numeric values.
Array2	Second array of numeric values.

@COVAR returns the covariance, which is the joint variability/degree of association of two random variables, by taking the deviations for each corresponding element in Array1 and Array2, computing their products, and taking the average of their average. Array1 and Array2 must have the same number of values. Use @COVAR to analyze the relationship between two data sets. The covariance is calculated using this formula:

$$\mathrm{COVAR}(X,Y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_x)(y_i - \mu_y)$$

Example

A high school professor recorded test scores for a class of nine students, along with their number of study hours, their estimated 'stress level' on a scale of 1 to 10, and their amount of sleep the night before the test. To examine the relationship between the attained test score and any of these three factors, the following formulae would be used:

@COVAR(A2..A9,B2..B9) = 22.265625

@COVAR(A2..A9,C2..C9) = 3.84375

@COVAR(A2..A9,D2..D9) = 17.796875

	A	B	C	D
1	SCORE	STUDY HRS.	STRESS	HRS. SLEEP
2	84	7	5	8
3	77	6	5	7
4	88	7.5	7	7
5	65	5.5	4	5
6	92	9	6	8.5
7	57	3	8	5
8	61	4	3	4
9	66	4.5	5	5

Related topics

@COUNT - Count Non-Blank Cells

Syntax

@COUNT(List)

List	One or more numeric or string values, cell addresses, and cell references or names, separated by commas.

@COUNT returns the number of non-blank cells in List. If more than one selection is listed, they must be separated by commas.

Examples

	A	B	C	D
1		January	February	March
2	John	$652	$833	$599
3	Mary	$456	$305	$522
4	Ralph	$68	$59	$73
5	Anna	$80	$80	$80

@COUNT(B2..B5) = 4

@COUNT(B1..B6) = 5

@COUNT(A6) = 0

@COUNT(A6..B6) = 0

@COUNT(C1..C5, D3..D6) = 8

Related topics

@COUNTBLANK - Count Blank Cells

Syntax

@COUNTBLANK(Block)

Block	The cells where you want to count blank cells.

@COUNTBLANK counts blank cells in specified cells.

@COUNTBLANK includes cells with formulas that return "" (or empty text), but it does not count cells with zero values.

Example

	A	B
1	One	
2	2	
3		
4		
5	410	

Cell A3 contains the formula @IF(A2<5,"",A2), which returns "".

Cell A4 is empty.

@COUNTBLANK(A1..A5) = 2

Related topics

@COUNTIF - Count Matching Cells

Syntax

@COUNTIF(Block, <Criteria>)

Block	Range of one or more cell addresses, a cell reference, or a name to include in the count.
Criteria	Numeric or string values that determine whether a cell is counted. If Criteria is omitted, @COUNTIF counts all cells containing logical values greater than 0 (text equals 0).

@COUNTIF returns the number of cells in Block that meet a specified set of criteria.

Example

	A	B	C	D
1		January	February	March
2	John	Transport	Entertainment	Finance
3	Mary	Food	Transport	Entertainment
4	Ralph	Finance	Food	Transport
5	Anna	Entertainment	Finance	Food

@COUNTIF(B2..D5,"Transport") = 3

@COUNTIF(B2..D5,"Food") = 3

@COUNTIF(B2..D5,"Finance") = 3

Related topics

@CRITBINOM - Critical Probability of Binomial Distribution

Syntax

@CRITBINOM(Trials, Prob, Alpha)

Trials	Integer number of Bernoulli trials; must be ≥ 0.
Prob	Probability of success per trial; must be ≥ 0 and ≤ 1.
Alpha	Critical probability to test; must be ≥ 0 and ≤ 1.

@CRITBINOM calculates the maximum number of successes that can occur before the cumulative probability expressed by Alpha is exceeded for the number of Trials. @CRITBINOM has applications in quality assurance. For example, you could use @CRITBINOM to calculate the maximum number of defects allowed in a shipment.

Example

Company A tests a sample of 100 electrical circuits received from Company B. The probability that a circuit is defective is 7%. Using an Alpha value of 7.4%, this formula calculates the maximum number of defective circuits that can be expected.

@CRITBINOM(100,0.07,0.074) = 3

Related topics

@CSC - Cosecant

Syntax

@CSC(X)

X	An angle measured in radians. X can be any value from approximately -9.00719E+15 through 9.00719E+15, excluding 0.

@CSC returns the cosecant of angle X, in radians. X must be specified in radians, not degrees. To convert degrees to radians, use @RADIANS.

The cosecant is the reciprocal of the sine. The result of @CSC is a value greater than or equal to 1, or less than or equal to -1.

Examples

@CSC(@RADIANS(30)) = 2

@CSC(@RADIANS(75)) = 1.035276

@CSC(@RADIANS(45)) = 1.414214

@CSC(@PI/6) = 2

Related topics

@CSCH - Hyperbolic Cosecant

Syntax

@CSCH(X)

X	Any value from approximately -708.39599 through 708.39599, but not 0.

@CSCH(X) calculates the hyperbolic cosecant of X. X must be specified in radians, not degrees. To convert degrees to radians, use @RADIANS.

The hyperbolic cosecant is the reciprocal of the hyperbolic sine.

Examples

@CSCH(@RADIANS(30)) = 1.825306

@CSCH(@RADIANS(75)) = 0.582688

@CSCH(@RADIANS(45)) = 1.151184

@CSCH(@PI/6) = 1.825306

Related topics

@CTERM - Compounding Periods

Syntax

@CTERM(Rate, Fv, Pv)

Rate	A numeric value > -1, representing the periodic interest rate (the fixed interest rate per compounding period).
Fv	A numeric value representing the future value of an investment (the value the investment will reach at some point).
Pv	A numeric value representing the current value of an investment (the present value).

@CTERM calculates the number of time periods required for an investment, while earning interest per compounding period, assuming that the investment is an ordinary annuity. It uses the formula

$$\frac{\ln(Fv/Pv)}{\ln(1 + R)}$$

where

Fv	future value
Pv	present value
R	periodic interest rate

An equivalent for this formula using @NPER is

@NPER(Rate, 0, - Pv, Fv)

@CTERM assumes that the investment is an ordinary annuity. @NPER, which is calculated differently than but is related to @CTERM, uses an optional argument, Type, to indicate whether the investment is an ordinary annuity or an annuity due.

Examples

Assuming that your savings account has an annual interest rate of 7%, how long would it take a $3000 deposit to reach $5000? The answer is

@CTERM(7%,5000,3000) = 7.55 years

If the Rate figure is given for years, the result is in years as well. If you are working with monthly interest, multiply the answer by 12 to get a result in months.

You can also use @NPER to solve this problem:

@NPER(7%,0,-3000,5000,0) = 7.55

Other examples:

@CTERM(0.07,5000,3000) = 7.550042

@CTERM(0.10,5000,3000) = 5.359612

@CTERM(0.12,5000,3000) = 4.50747

@CTERM(0.12,10000,7000) = 3.147261

Related topics

@CUMIPMT - Cumulative Interest Paid

Syntax

@CUMIPMT(Rate, Nper, Pv, StartPeriod, EndPeriod, Type)

Rate	Interest rate.
Nper	Total number of payment periods.
Pv	Present value.
StartPeriod	First period in the calculation.
EndPeriod	Last period in the calculation.
Type	Timing of payment: 0 = payment at the end of the period 1 = payment at the beginning of the period

@CUMIPMT returns the cumulative interest paid on a loan between specified periods or in a single period.

Nper, StartPeriod, EndPeriod, and Type are truncated to integers.

Make sure you are consistent about the units you use for specifying Rate and Nper. For example:

- For annual payments on a four-year loan at 12% annual interest, use 12% for Rate and 4 for Nper.
- For monthly payments on the same loan, use 12%/12 for Rate and 4*12 for Nper.
- @CUMIPMT returns ERR if:
- Rate <= 0, Nper <= 0, or Pv <= 0.
- StartPeriod < 1, EndPeriod < 1, or StartPeriod > EndPeriod.
- Type is any number other than 0 or 1.
- Any argument is non-numeric.

Examples

Your $250,000 home has a 30-year mortgage at an interest rate of 8.25%. To find out how much interest you will pay this year, the fourth year of the loan:

@CUMIPMT(0.0825/12,30*12, 250000,37,48,0) = -19995.14

The interest you will pay the first month of this year is:

@CUMIPMT(0.0825/12,30*12, 250000,37,37,0) = -1674.16

Related topics

@CUMPRINC - Cumulative Principal Paid

Syntax

@CUMPRINC(Rate, Nper, Pv, StartPeriod, EndPeriod, Type)

Rate	Interest rate.
Nper	Total number of payment periods.
Pv	Present value.
StartPeriod	First period in the calculation.
EndPeriod	Last period in the calculation.
Type	Timing of the payment: 0 = payment at the end of the period 1 = payment at the beginning of the period

@CUMPRINC returns the cumulative principal paid on a loan between specified periods or in a single period.

Nper, StartPeriod, EndPeriod, and Type are truncated to integers.

Make sure you are consistent about the units you use for specifying Rate and Nper. For example:

- For annual payments on a four-year loan at 12% annual interest, use 12% for Rate and 4 for Nper.
- For monthly payments on the same loan, use 12%/12 for Rate and 4*12 for Nper.

@CUMIPMT returns ERR if:

- Rate <= 0, Nper <= 0, or Pv <= 0.
- StartPeriod < 1, EndPeriod < 1, or StartPeriod > EndPeriod.
- Type is any number other than 0 or 1.
- Any argument is non-numeric.

Examples

Your $250,000 home has a 30-year mortgage at an interest rate of 8.25%. To find out how much principal you will pay this year, the fourth year of the loan:

@CUMPRINC(0.0825/12,30*12, 250000,37,48,0) = -2542.86

The principal you will pay the first month of this year is:

@CUMPRINC(0.0825/12,30*12, 250000,37,37,0) = -204.01

Related topics

@CURVALUE - Current Value of Quattro Pro/DOS Command

Syntax

@CURVALUE(GeneralAction, SpecificAction)

GeneralAction	A general menu category.
SpecificAction	A menu item that requires setting.

@CURVALUE returns the current value of a Corel Quattro Pro for DOS menu command setting. It is used in macros, usually to base the next action on a particular menu setting, and is included for compatibility with Corel Quattro Pro for DOS. To view a list of acceptable arguments, press Shift+F3 and choose / Commands.

A related @function that uses Corel Quattro Pro for Windows command equivalents is @COMMAND. Another related @function, @PROPERTY, returns settings for requested properties.

Both GeneralAction and SpecificAction must be enclosed in double quotes. They must together create one of the Corel Quattro Pro for DOS menu-equivalent commands.

Not all GeneralAction/SpecificAction combinations return a useful value. In general, only menu commands that display a current setting or status have menu equivalents that are useful for @CURVALUE. Some settings previously controlled with Corel Quattro Pro for DOS commands are now set through Windows, particularly hardware options and printer settings. These don't return a setting in Corel Quattro Pro for Windows.

Like @CELL, @CURVALUE statements do not recalculate automatically as many other @functions do. Press F9 to obtain the current value.

Examples

@CURVALUE("print","block") = the currently specified print selection

@CURVALUE("file","save") = the name of the last file saved

Related topics

@D360 - Difference Between Dates

Syntax

@D360(StartDate, EndDate)

StartDate	Date number. See "Using dates and times in Quattro Pro."
EndDate	Date number.

@D360 returns the number of days between two dates, based on a 360-day year (twelve 30-day months). Use this function to help compute payments if your accounting system is based on this calendar. The formula conforms to the 1990 modifications to the Securities Industry Association's 1986 edition of Standard Security Calculation Methods.

A similar function, @DAYS360, differs from @D360 by adding a third argument, Method. When you specify Method as FALSE (the default), you can in some cases get a different result using @DAYS360 from the one returned by @D360. See @DAYS360 for details.

For a conventional year of 365 days (366 in leap years), use @DATEDIF or subtraction.

Examples

To calculate the number of days between March 3 and May 31 of 1996:

@D360(@DATE(96,3,3), @DATE(96,5,31)) = 87

The same calculation using date values: @D360(35127, 35216) = 87

However, @DAYS360 gives a different result when no Method is specified:

@DAYS360(@DATE(96,3,3), @DATE(96,5,31)) = 88

Related topics

@DATE - Date

Syntax

@DATE(Yr, Mo, Day)

Yr	For years 1900 through 3199, the four-digit year is valid. Or, use a numeric value between -300 and 1299 (-300 = 1600, 0 = 1900, 1299 = 3199)
Mo	A numeric value between 1 and 12.
Day	A numeric value between 1 and 31.

@DATE(40,12,31) and @DATE(1940,12,31) both return 14976 (31-Dec-1940).

@DATE(140,12,31) and @DATE(2040,12,31) both return 51501 (31-Dec-2040)

@DATE returns the date/time "serial number" of the date specified with year, month, and day arguments. This serial number can range from -109,571 (January 1,1600) to 474,816 (December 31, 3199). December 30, 1899 is 0, so a positive number represents the number of days from December 30, 1899 up to the date referenced in the formula.

Date/time serial numbers are used in notebook calculations. (The fractional portion of a serial number is used for the time @functions.)

To display a date/time serial number in a date format, right-click cells, click Cell Properties, then click Numeric Format. This shows the date in its more common form (for example, Jan-1-94 instead of 34335).

Any illegal dates return ERR as their value, for example, @DATE(87,2,29). (This date corresponds to February 29, 1987, which is impossible; 1987 was not a leap year.) See "Using dates and times in Quattro Pro."

Examples

@DATE(1940,12,31)= 14976 (December 31, 1940)

@DATE(2040,12,31)= 51501 (December 31, 2040)

@DATE(93,1,1) = 33970 (January 1, 1993)

@DATE(91,9,13) = 33494 (September 13, 1991)

@DATE(0,1,1) = 2 (January 1, 1900)

@DATE(-300,1,1) = -109571 (January 1, 1600)

@DATE(1299,12,31) = 474816 (December 31, 3199)

Related topics

@DATEDIF - Days, Months, or Years Between Dates

Syntax

@DATEDIF(StartDate, EndDate, Format)

StartDate	Date number. See "Using dates and times in Quattro Pro."
EndDate	Date number.
Format	Code, entered as text, specifying format of the result: y = Years m = Months d = Days md = Days, disregarding months and years ym = Months, disregarding years yd = Days, disregarding years

@DATEDIF calculates the number of years, months, or days between two dates. @DATEDIF uses a 365-day year and a 366-day leap year.

To find the number of days between two dates using a 360-day financial year, use @D360 or @DAYS360.

Examples

@DATEDIF(@DATE(96,3,3), @DATE(96,5,31),"d") = 89

Similarly, @DATE(96,5,31)- @DATE(96,3,3) = 89

@DATEDIF(@DATE(94,3,3), @DATE(96,5,31),"md") = 28, the number of days from the 3rd to and the 31st of any month, disregarding the difference in years

@DATEDIF(@DATE(94,3,3), @DATE(96,5,31),"ym") = 2, the number of months from March to May, disregarding the difference in years

Related topics

@DATEINFO - Information About a Date

Syntax

@DATEINFO(Date, Attribute)

Date	Date number. See "Using dates and times in Quattro Pro."
Attribute	Code for the type of information you want:
	1 = Day of the week as a label, in short format (Mon)
	2 = Day of the week as a label, long format (Monday)
	3 = Day of the week as an integer from 0 (Monday) through 6 (Sunday)
	4 = Week of the year as an integer from 1 to 53
	5 = Month of the year as a label, in short format (Jan)
	6 = Month of the year as a label, in long format (January)
	7 = Number of days in the month specified by date
	8 = Number of days left in the month specified by date
	9 = Last day of the month specified by date
	10 = The Quarter date is in, as an integer from 1 (Q1) through 4 (Q4)
	11 = 1 if the year specified by date is a leap year; 0 if the year is not a leap year
	12 = Day of the year specified by date, as a number from 1 to 366
	13 = Days left in the year specified by date, as a number

@DATEINFO returns information about a date number. @DATEINFO uses a 365-day year and a 366-day leap year.

@DATEINFO supports dates ranging from 1900-2099.

The valid date calculation range for this function is 01/01/1900 through 12/31/2099.

Examples

@DATEINFO(@DATE(96,3,3),2) = Sunday

@DATEINFO(@DATE(96,3,3),4) = 9, because March 3, 1996 is in the 9th week of the year; weeks start on Monday and end on Sunday.

@DATEINFO(@DATE(96,3,3),13) = 303, the number of days left in the year 1996

Related topics

@DATEVALUE - Value Corresponding to Date

Syntax

@DATEVALUE(DateString)

DateString	A numeric or string value in any valid date format, enclosed by quotation marks (or coordinates or a cell name for a selection that contains a date string).

@DATEVALUE returns a serial date value that corresponds to the value in DateString. If the value in DateString is not in the correct format or is not enclosed in quotes, ERR or a syntax error message is returned. If DateString is entered using the international format, the year, month, and day must be in the same order as the current international date format (set in Tools ▸ Settings ▸ International) and the separator character must also agree.

You can display resulting date string values in standard date formats by right-clicking cells, clicking Cell Properties, then clicking Numeric Format.

There are five valid formats for DateString:

- DD-MMM-YY ("04-Jul-92").
- DD-MMM ("04-Jul") (assumes the current year).
- MMM-YY ("Jul-92") (assumes the first of the month).
- DD-MMM-YYYY ("04-Jul-1992").
- MM-YYYY ("Jul-1992") (assumes the first of the month).
- The Long International date format specified as the system default, two of which are MM/DD/YY ("07/04/92") and MM/DD/YYYY ("07/04/1992").
- The Short International date format specified as the system default, one of which are MM/DD ("07/18"). This assumes the current year. See "Using dates and times in Quattro Pro."

@DATEVALUE is included for compatibility with other products.

Examples

@DATEVALUE("12-Mar-2010") = 40249

@DATEVALUE("07/04/92") = 33789

@DATEVALUE("JUL-92") = 33786 (July 1, 1992)

@DATEVALUE("04-may-93") = 34093

@DATEVALUE(07/04/94) = 0.018617 (no quotes makes Quattro Pro divide the numbers)

@DATEVALUE("May-04-1992") = ERR

Related topics

@DAVG - Database Average

Syntax

@DAVG(Block, Column, Criteria)

Block	The 2-D cells (reference or name) containing the database, including field names.
Column	The number of the column containing the field you want to average (the first column in Block is 0, the second is 1, and so on).
Criteria	2-D cells containing search criteria; the first row must be field names.

@DAVG averages selected field entries in a database. It includes only those entries in Column whose records meet the criteria specified in Criteria.

The field specified in your criteria and the field being averaged need not be the same. The field averaged is that contained within the column you specify as Column.

You can specify all or part of your database as Block, but field names must be included for each field you include in the cells.

Examples

These examples refer to the database and criteria tables.

	A	B	C	D
1				
2	DATE	LOCATION	REP	AMOUNT
3	Jul-91	San Fran	CJ	$14,999
4	Jul-91	LA	RX	$28,725
5	Jul-91	Chicago	RX	$18,600
6	Jul-91	NY	CJ	$15,600
7	Aug-91	Chicago	CJ	$23,769
8	Aug-91	LA	RX	$34,345
9				
10		Criteria Table		
11	Date	Location	Rep	
12	Jul-91	San Fran	CJ	
13		LA		

@DAVG(A2..D8,3,A11..A12) = $19,481 (average of July sales)

@DAVG(A2..D8,3,B11..B13) = $26,023 (average of California sales)

@DAVG(A2..D8,3, C11..C12) = $18,123 (average of CJ's sales)

@DAVG(A2..D8,4,A11..C13) = ERR (Column figure too high)

@DAVG(A2..D8,2,A11..A12) = 0 (labels are treated as 0)

Related topics

@DAY - Day Portion of Date Serial Number

Syntax

@DAY(DateTimeNumber)

DateTimeNumber	A numeric value between -109571 and 474816.9999999, representing a date/time serial number: -109571 = January 1, 1600; 0 = December 31, 1899; 474816 = December 31, 3199; the decimal = time (24 hr).

@DAY converts the date/time serial number you supply as DateTimeNumber into the number associated with that day (1-31). Decimal (time) portions of the number are ignored. See "Using dates and times in Quattro Pro."

Examples

@DAY(33508) = 27 (9/27/91)

@DAY(32134) = 23 (12/23/87)

@DAY(@DATE(93,9,10)) = 10

@DAY(474817) = ERR because the number you entered was larger than 474816.9999999

Related topics

@DAYS360 - Difference Between Dates

Syntax

@DAYS360(StartDate, EndDate, <Method>)

StartDate	Date number. See "Using dates and times in Quattro Pro."
EndDate	Date number.
Method	Optional logical value that specifies US or European method: FALSE or 0 = US (NASD); the default if you do not specify a method TRUE or 1 = European method

@DAYS360 returns the number of days between two dates based on a 360-day year (twelve 30-day months). Use this function to help compute payments if your accounting system is based on this calendar. The formula conforms to the 1990 modifications to the Securities Industry Association's 1986 edition of Standard Security Calculation Methods.

You can express dates either as text strings or date values.

- Method affects the result only in some cases when StartDate or EndDate is the 31st of the month:
- Using the US method, if StartDate is the 31st, @DAYS360 makes it the 30th of the same month. If the EndDate is the 31st and the StartDate is less than the 30th of a month, @DAYS360 makes EndDate the 1st of the next month; otherwise it makes EndDate the 30th of the same month.
- Using the European method, if StartDate or EndDate is the 31st, @DAYS360 makes it the 30th of the same month.

If StartDate occurs after EndDate, DAYS360 returns a negative number.

A similar function, @D360, is available. It omits the optional Method argument and will in some cases give a different result from the one returned by @DAYS360.

For a conventional year of 365 days (366 in leap years), use @DATEDIF or subtraction.

Examples

@DAYS360(@DATE(96,3,3), @DATE(96,5,31),FALSE) = 88

@DAYS360(@DATE(96,3,3), @DATE(96,5,31)) = 88

@DAYS360(@DATE(96,3,3), @DATE(96,5,31),1) = 87

The same calculation using date values:

DAYS360(35127, 35216,TRUE) = 87

Using subtraction,

@DATE(96,5,31) - @DATE(96,3,3) = 89, using a 364-day year

Related topics

@DB - Declining Balance Depreciation

Syntax

@DB(Cost, Salvage, Life, Period,<Month>)

Cost	Amount originally paid for an asset.
Salvage	Estimated value at end of asset life.
Life	Number of periods the asset takes to depreciate to its salvage value.
Period	Length of time for which you want to know the depreciation allowance.
Month	Number of months in the first year (optional); If Month is omitted, @DB uses 12.

@DB calculates the depreciation of an asset over a specified period using the fixed-declining balance method. @DB uses the following formulas to calculate depreciation for a period:

(cost - total depreciation from prior periods) * rate

rate = $1 - ((S / C)^{(1/L)})$, rounded to three decimal places

where

S	salvage
C	cost
L	life

Depreciation for the first and last periods are special cases:

For the first period, DB uses the formula cost * rate * month / 12

For the last period, DB uses this formula:

((cost - total depreciation from prior periods) * rate * (12 - month)) / 12

- Cost can be any positive value, including 0. If Cost is 0, the result of @DB is 0.
- Salvage can be any positive value, **but must be greater than 0**, since assets should have a salvage value. If Salvage is greater than Cost, the result of @DB is negative.
- Life and Period can be any positive value, but not 0. They must be in the same units, usually years. Life cannot be greater than Period.

The fixed-declining balance method slows the rate of depreciation in comparison to the double-declining balance method, so more depreciation expense can be written off in later periods. Depreciation ends when the asset's book value (asset cost minus accumulated depreciation) reaches its salvage value.

Example

Your office uses a machine purchased on July 29, 1993, for $13,250. It has an expected life of 10 years and expected salvage value of $100. Your fiscal years ends December 31, so the first period has 5 months. To calculate fourth-period depreciation for 1996:

@DB(13250,100,10,4,5) = 1616.145

Related topics

@DCOUNT - Database Count

Syntax

@DCOUNT(Block, Column, Criteria)

Block	The 2-D cells (reference or name) containing the database, including field names.
Column	The number of the column containing the field you want to count (the first column in Block is 0, the second is 1, and so on).
Criteria	2-D cells containing search criteria; the first row must be field names.

@DCOUNT counts selected field entries in a database. It includes only those non-blank entries in Column whose records meet the criteria specified in Criteria. (If you specifically want to count blank cells, use @COUNTBLANK instead of @DCOUNT.)

The field specified in your criteria and the field being counted need not be the same. The field counted is that contained within the column you specify as Column.

You can specify all or part of your database as Block, but field names must be included for each field you include in the cells.

Examples

	A	B	C	D
1				
2	DATE	LOCATION	REP	AMOUNT
3	Jul-91	San Fran	CJ	$14,999
4	Jul-91	San Jose	RX	$25,000
5	Jul-91	Chicago	RX	$18,998
6	Jul-91	NY	CJ	$15,600
7	Aug-91	Chicago	CJ	$23,769
8	Aug-91	LA	RX	$34,345
9				
10		CRITERIA TABLE		
11	Date	Location	Rep	
12	Jul-91	San Fran	CJ	
13		LA		

These examples refer to the database and criteria tables.

@DCOUNT(A2..D8,3,A11..A12) = 4 (number of July sales)

@DCOUNT(A2..D8,3,B11..B13) = 2 (number of San Fran and LA sales)

@DCOUNT(A2..D8,3,C11..C12) = 3 (number of CJ's sales)

@DCOUNT(A2..D8,4,A11..C13) = ERR (Column figure too high)

@DCOUNT(A3..D8,3,A11..A12) = 6 (incorrect--field names not included)

Related topics

@DDB - Double-Declining Balance Depreciation

Syntax

@DDB(Cost, Salvage, Life, Period)

Cost	A numeric value representing the amount paid for an asset.
Salvage	A numeric value representing the value of an asset at the end of its useful life.
Life	A numeric value representing the expected useful life of an asset (in years).
Period	A numeric value representing the time period for which you want to calculate depreciation.

@DDB determines accelerated depreciation values for an asset, given the initial cost, life expectancy, end value, and depreciation period. It calculates depreciation using the double-declining balance method.

Depreciation value (DDB) and book value (BV) are calculated by:

BV	Cost
DDB	2BV/Life
BV	BV - DDB

These statements must be true:

Life ≥ Period ≥ 1

Life and Period must be integers

Cost ≥ Salvage ≥ 0

@SLN and @SYD offer other depreciation methods. @VDB uses the variable-rate declining balance method.

Examples

Suppose you just bought a new $4000 computer. The dealer says you can sell it back to the store for $350 after eight years, but no one would want to buy it after that. In other words, Salvage is $350 and Life is 8. To calculate the double-declining depreciation allowance of this computer by the second year, enter this formula:

@DDB(4000,350,8,2) = $750

These examples show depreciation values for the first five years of a $15,000 investment with a salvage value of $3000 and a life of 10 years:

@DDB(15000,3000,10,1) = $3,000

@DDB(15000,3000,10,2) = $2,400

@DDB(15000,3000,10,3) = $1,920

@DDB(15000,3000,10,4) = $1,536

@DDB(15000,3000,10,5) = $1,229

Related topics

@DDELINK - DDE Link

Syntax

@DDELINK([AppName|Topic]"DataToReceive", <nCols>, <nRows>, <nSheets>)

AppName	The DDE-server application to contact. The entire path to each file must be included.
Topic	The table, spreadsheet, document, or other file in the DDE-server application from which to retrieve data. The entire path to each file must be included.
DataToReceive	The field, cells, or other information to receive from the application (DDE Item string).
nCols	The number of columns in the data cells (optional).
nRows	The number of rows in the data cells (optional).
nSheets	The number of sheets in the data cells (optional).

@DDELINK creates a "live" data link from another Windows application that supports DDE (Dynamic Data Exchange). Using @DDELINK is equivalent to choosing Edit|Paste special with data from another application copied to the Clipboard, and then choosing Paste Link from the Paste special dialog box.

AppName and Topic are the same arguments used in the macro {INITIATE} except that "System" is not accepted as a substitute for Topic. DataToReceive is a string indicating the location of the target data in the server application.

When you enter @DDELINK into a cell, the linked data appears there. Unless you indicate otherwise, the data takes up as much space as it did in the original application. You can use nCols, nRows, and nSheets to specify smaller dimensions. If any of the arguments is 0 or omitted, the original dimension applies.

@DDELINK sets up a zone of cells that can be overwritten whenever data changes in the DDE-server application. Avoid storing other data near @DDELINK, and consider using the limit arguments.

Examples

The maximum size of the data cells for the following formula is 5 cells by 5 cells:

@DDELINK([EXCEL|FILE1]"R1C1:R5C5")

With nRows = 3, the maximum size of the data cells drops to 5 cells by 3 cells:

@DDELINK([EXCEL|FILE1]"R1C1:R5C5",0,3)

Related topics

@DEGREES - Convert Radians to Degrees

Syntax

@DEGREES(X)

X	A numeric value representing radians.

@DEGREES converts the given number of radians to degrees. It uses this formula:
180 times X divided by pi

Examples

@DEGREES(0.5) = 28.64789
@DEGREES(0.017) = 0.974028
@DEGREES(@PI/2) = 90

Related topics

@DELTA - Test If Two Numbers are Equal

Syntax

@DELTA(X, <Y>)

X	Numeric value to check.
Y	Numeric value that X must equal for the function to return 1 (if omitted, assumed to be zero).

@DELTA tests whether X and Y are equal. If they are, @DELTA returns 1 (True); if not, @DELTA returns 0 (False).

Examples

@DELTA(1,2) = 0

@DELTA(2,2) = 1

@IF(@DELTA(2,2),"Equal","Not Equal") = Equal

Related topics

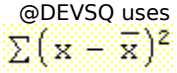
@DEVSQ - Sum of the Squares of the Deviations

Syntax

@DEVSQ(List)

List	One or more numeric or cell values.

@DEVSQ returns the sum of the squares of the deviations of the numbers in List from their mean value.
@DEVSQ uses this formula:



Example

@DEVSQ(9,10,12,14,15) = 26

Related topics

@DFRAC - Decimal to Fraction

Syntax

@DFRAC(Dec, Denom)

Dec	Number to be converted, expressed as a decimal.
Denom	Denominator; must be an integer > 0.

@DFRAC converts a number expressed as a decimal to a fraction using the specified denominator. @DFRAC reverses the effect of @FRACD.

The result looks like a decimal, but the portion to the right of the decimal point is actually the numerator of the fraction using the specified denominator. For example, you can use @DFRAC to convert a decimal to 32nds.

Converting 99.375 to 32nds results in 99.12, representing 99 12/32.
Format the cell that contains the @function to show the same number of decimal places as the number of digits in the desired Denom. For example, if Denom is 32, set the cell format to display two decimal places.

Example

@DFRAC(106.4375,32) = 106.14; the 14 to the right of the decimal place signifies 14/32.

Related topics

@DGET - Database Value

Syntax

@DGET(database-block,column,criteria-block)

Block	The 2-D cells (reference or name) containing the database, including field names.
Column	The number of the column containing the field for which you want to return a value or label (the first column in Block is 0, the second is 1, and so on).
Criteria	2-D cells containing search criteria; the first row must be field names.

@DGET returns a value or label from a field of a database table that meets specified criteria.

If no record matches the criteria, @DGET returns ERR.

If more than one record matches the criteria, @DGET returns ERR.

Examples

	A	B	C	D
1				
2	DATE	LOCATION	REP	AMOUNT
3	Jul-91	San Fran	CJ	$14,999
4	Jul-91	LA	RX	$28,725
5	Jul-91	Chicago	RX	$18,600
6	Jul-91	NY	CJ	$15,600
7	Aug-91	Chicago	CJ	$23,769
8	Aug-91	LA	RX	$34,345
9				
10		CRITERIA TABLE		
11	Date	Location	Rep	
12	Jul-91	San Fran	CJ	
13		LA		

@DGET(A2..D8,3,B11..B12) = $14,999 (Amount for San Fran)

@DGET(A2..D8,3,C11..C12) = ERR (Returns ERR because CJ has more than one Amount entry)

@DGET(A2..D8,3,B11..C12) = $14,999 (Amount in July for San Fran)

Related topics

@DISC - Discount Rate

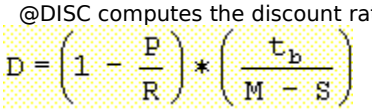
Syntax

@DISC(Settle, Maturity, Price, <Redemption>, <Calendar>)

Settle	Number representing the settlement date; must be < Maturity.
Maturity	Number representing the maturity date; must be > Settle.
Price	Settlement price per 100 face value; must be ≥ 0 and ≤ 100.
Redemption	Redemption value per 100 face value (must be > 0; the default is 100).
Calendar	Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

@DISC computes the discount rate for a security, which is the percentage discount offered on a security for a 360-day or 365-day term.

@DISC computes the discount rate using this formula:



D	discount rate
P	price
R	redemption
b	basis
M	maturity
S	settle

tb is the number of days over which the discount rate applies (360 or 365).

Example

This formula calculates the discount rate for a bond with the following terms: Settle is May 27, 1995, Maturity is November 24, 1995, Price is 96.2492, Redemption is 100, and Calendar is 2 (actual/360).

@DISC(@DATE(95,5,27),@DATE(95,11,24),96.2492,100,2) = 0.074602

Related topics

@DMAX - Database Maximum Value

Syntax

@DMAX(Block, Column, Criteria)

Block	The 2-D cell cells (reference or name) containing the database, including field names.
Column	The number of the column containing the field for which you want to find the maximum value (the first column in Block is 0, the second is 1, and so on).
Criteria	2-D cells containing search criteria; the first row must be field names.

@DMAX finds the maximum value of selected field entries in a database. It includes only those entries in Column whose records meet the criteria specified in Criteria.

The field specified in your criteria and the field you are finding the maximum value for need not be the same. The field you are finding the maximum value for is that contained within the column you specify as Column.

You can specify all or part of your database as Block, but field names must be included for each field you include in the cells.

Examples

	A	B	C	D
1				
2	DATE	LOCATION	REP	AMOUNT
3	Jul-91	San Fran	CJ	$14,999
4	Jul-91	LA	RX	$28,725
5	Jul-91	Chicago	RX	$18,600
6	Jul-91	NY	CJ	$15,600
7	Aug-91	Chicago	CJ	$23,769
8	Aug-91	LA	RX	$34,345
9				
10		CRITERIA TABLE		
11	Date	Location	Rep	
12	Jul-91	San Fran	CJ	
13		LA		

@DMAX(A2..D8,3,A11..A12) = $28,725 (highest July sale)

@DMAX(A2..D8,3,B11..B13) = $34,345 (highest California sale)

@DMAX(A2..D8,3,C11..C12) = $23,769 (highest of CJ's sales)

@DMAX(A2..D8,4,A11..C13) = ERR (Column figure too high)

@DMAX(A3..D8,3,A11..A12) = $34,345 (incorrect--field names not included)

Related topics

@DMIN - Database Minimum Value

Syntax

@DMIN(Block, Column, Criteria)

Block	Cells (reference or name) containing the database, including field names.
Column	The number of the column containing the field for which you want to find the minimum value (the first column in Block is 0, the second is 1, and so on).
Criteria	Cells containing search criteria; the first row must be field names.

@DMIN finds the minimum value of selected field entries in a database. It includes only those entries in Column whose records meet the criteria specified in Criteria.

The field specified in your criteria and the field for which you are finding the minimum value need not be the same. The field for which you are finding the minimum value is that contained within the column you specify as Column.

You can specify all or part of your database as Block, but field names must be included for each field you include in the cells.

Examples

	A	B	C	D
1				
2	DATE	LOCATION	REP	AMOUNT
3	Jul-91	San Fran	CJ	$14,999
4	Jul-91	LA	RX	$28,725
5	Jul-91	Chicago	RX	$18,600
6	Jul-91	NY	CJ	$15,600
7	Aug-91	Chicago	CJ	$23,769
8	Aug-91	LA	RX	$34,345
9				
10		CRITERIA TABLE		
11	Date	Location	Rep	
12	Jul-91	San Fran	CJ	
13		LA		

@DMIN(A2..D8,3,A11..A12) = $14,999 (smallest July sale)

@DMIN(A2..D8,3,B11..B13) = $14,999 (smallest California sale)

@DMIN(A2..D8,3,C11..C12) = $14,999 (smallest of CJ's sales)

@DMIN(A3..D8,3,A11..A12) = $15,600 (incorrect--field names not included)

@DMIN(A2..D8,4,A11..C13) = ERR (Column figure too high)

Related topics

@DOLLAR - Dollars as Text

Syntax

@DOLLAR(Num, <Dec>)

Num	Numeric value, reference to a cell that contains a numeric value, or formula that returns a numeric value.
Dec	How many digits you want to display to the right of the decimal point. If Dec is negative, @DOLLAR rounds off Num to the left of the decimal point. If you omit Dec, @DOLLAR rounds off to 2 decimal places.

@DOLLAR converts a numeric value to text, using currency format. Decimals are rounded to the specified place:

0	$##,###
2	$##,###.##
-3	$##,000

You can also format numeric values by right-clicking the cell and choosing Cell Properties, then Currency. The result looks the same, but the value remains a numeric value when you use Cell Properties, while @DOLLAR converts it to text.

Examples

@DOLLAR(9449.985, 2) = "$9449.99"

@DOLLAR(9449.985, 0) = "$9450"

@DOLLAR(9449.985, -3) = "$9000"

@DOLLAR(9449.985) = "$9449.99"

Related topics

@DOLLARDE - Fractional Price Into Dollars

Syntax

@DOLLARDE(FracDollar, Denom)

FracDollar	Number and numerator of the fraction, expressed as number.numerator.
Denom	Denominator of the fraction: ▪ Denom must be a numeric value >0 ▪ If Denom is not an integer, @DOLLARDE truncates it

@DOLLARDE converts a fractional price into dollars. Use @DOLLARDE to convert fractional values like stock prices into dollars.

Examples

@DOLLARDE(12.3,4) = 12.75

@DOLLARDE(12.5,8) = 12.625

Related topics

@DOLLARFR - Dollar Price Into Fraction

Syntax

@DOLLARFR(DecDollar, Denom)

DecDollar	Dollar price.
Denom	Denominator of the fraction: ▪ Denom must be a numeric value >0 ▪ If Denom is not an integer, @DOLLARFR truncates it

@DOLLARFR converts a dollar price into a fractional price. Use @DOLLARFR to convert dollars into fractional values like stock prices. The result is expressed in the format dollar**.**numerator; the denominator is the number you specified for Denom.

Examples

@DOLLARFR(12.75,4) = 12.3, meaning "12 and 3/4"

@DOLLARFR(12.625,8) = 12.5, meaning "12 and 5/8"

Related topics

@DOLLARTEXT - Dollar Numeric Value Into Dollar Text

Syntax

@DOLLARTEXT(Number, <Format>)

Number	Numeric value, reference to a cell that contains a numeric value, or formula that returns a numeric value.
Format	1 = Displays dollar value in text; ignores decimal values.
	2 = Displays dollar value in text, followed by "Dollars".
	3 = Displays dollar value in text, followed by cent value in numbers. The decimal is rounded to two decimal places.
	4 = Displays dollar value in text, followed by cent value in numbers, followed by "Dollars". This is the default if no Format is specified.
	5 = Displays dollar value in text, followed by "Dollars", followed by cent value in text, followed by "Cents".

@DOLLARTEXT converts the numeric value to a cardinal number in text, similar to how numbers are written out on checks or official documents.

Examples

@DOLLARTEXT(100.25,1) = One Hundred

@DOLLARTEXT(100.25,2) = One Hundred Dollars

@DOLLARTEXT(956600.55,3) = Nine Hundred Fifty-Six Thousand Six Hundred and 55/100

@DOLLARTEXT(66500.70,4) = Sixty-Six Thousand Five Hundred and 70/100 Dollars

@DOLLARTEXT(66500.70,5) = Sixty-Six Thousand Five Hundred Dollars and Seventy Cents

Related topics

@DPURECOUNT - Database count numbers

Syntax

@DPURECOUNT (database-block,column,criteria-block)

Block	The 2-D cells (reference or name) containing the database, including field names.
Column	The number of the column containing the field for which you want to count all number field entries (the first column in Block is 0, the second is 1, and so on).
Criteria	2-D cells containing search criteria; the first row must be field names.

@DPURECOUNT counts all number field entries in a database that match the criteria.

You can specify all or part of your database as Block, but field names must be included for each field you include in the cells.

The field specified in your criteria and the field to be counted do not need to be the same. The field that is counted is the column you specify for Column.

Examples

	A	B	C	D
1				
2	DATE	LOCATION	REP	AMOUNT
3	Jul-91	San Fran	CJ	$14,999
4	Jul-91	LA	RX	$28,725
5	Jul-91	Chicago	RX	$18,600
6	Jul-91	NY	CJ	$15,600
7	Aug-91	Chicago	CJ	$23,769
8	Aug-91	LA	RX	$34,345
9				
10		CRITERIA TABLE		
11	Date	Location	Rep	
12	Jul-91	San Fran	CJ	
13		LA		

@DPURECOUNT(A2..D8,3,B11..B12) = 1 (number of San Fran sales)

@DPURECOUNT(A2..D8,3,B11..B13) = 3 (number of San Fran and LA sales)

@DPURECOUNT(A2..D8,3,A11..A12) = 4 (number of July sales)

@DPURECOUNT(A2..D8,3,C11..C12) = 3 (number of CJ's sales)

@DPURECOUNT(A2..D8,3,A11..C13)= 2 (number of CJ's sales in San Fran in July and LA sales)

@DPURECOUNT(A2..D9,3,A11..B13)= 2 (same as above example because it ignores blank cells)

@DPURECOUNT(A2..D8,1,A11..B13)= 0 (no numbers in the column to count)

@DPURECOUNT(A3..D8,3,A11..A12)= 5 (incorrect, no field names were included)

Related topics

@DPRODUCT - Database product value

Syntax

@DPRODUCT(database-block,column,criteria-block)

Block	The 2-D cells (reference or name) containing the database, including field names.
Column	The number of the column containing the field for which you want to multiply all matching values (the first column in Block is 0, the second is 1, and so on).
Criteria	2-D cells containing search criteria; the first row must be field names.

@DPRODUCT multiplies the values in a specified field from all records in a database that match the specified criteria.

@DPRODUCT returns 1 if no cells in Block match the values in Criteria.

Related topics

@DSTD - Database Standard Deviation

Syntax

@DSTD(Block, Column, Criteria)

Block	Cells (reference or name) containing the database, including field names.
Column	The number of the column containing the field for which you want to find the standard deviation (the first column in Block is 0, the second is 1, and so on).
Criteria	Cells containing search criteria; the first row must be field names.

@DSTD finds the population standard deviation for selected field entries in a database. @DSTDS computes the standard deviation of sample data.

@DSTD includes only those entries in Column whose records meet the criteria specified in Criteria.

The field specified in Criteria and the field for which you are finding the standard deviation need not be the same. The field for which you are finding the standard deviation is the field contained within Column.

You can specify all or part of your database as Block, but field names must be included for each field in the cells.

Examples

	A	B	C	D
1				
2	DATE	LOCATION	REP	AMOUNT
3	Jul-91	San Fran	CJ	$14,999
4	Jul-91	LA	RX	$28,725
5	Jul-91	Chicago	RX	$18,600
6	Jul-91	NY	CJ	$15,600
7	Aug-91	Chicago	CJ	$23,769
8	Aug-91	LA	RX	$34,345
9				
10		CRITERIA TABLE		
11	Date	Location	Rep	
12	Jul-91	San Fran	CJ	
13		LA		

@DSTD(A2..D8,3,B11..B13) = $8,126 (population SD of California sales)

@DSTD(A2..D8,3,C11..C12) = $4,000 (population SD of CJ's sales)

@DSTD(A2..D8,4,A11..C13) = ERR (Column figure too high)

Related topics

@DSTDS - Database Sample Standard Deviation

Syntax

@DSTDS(Block, Column, Criteria)

Block	Cells (reference or name) containing the database, including field names.
Column	The number of the column containing the field for which you want to find the standard deviation (the first column in Block is 0, the second is 1, and so on).
Criteria	Cells containing search criteria; the first row must be field names.

@DSTDS finds the sample standard deviation for selected field entries in a database. @DSTD computes the standard deviation of population data.

This @function is not compatible with 1-2-3. If your file must be compatible with 1-2-3, use @DSTD instead.

Examples

	A	B	C	D
1				
2	DATE	LOCATION	REP	AMOUNT
3	Jul-91	San Fran	CJ	$14,999
4	Jul-91	LA	RX	$28,725
5	Jul-91	Chicago	RX	$18,600
6	Jul-91	NY	CJ	$15,600
7	Aug-91	Chicago	CJ	$23,769
8	Aug-91	LA	RX	$34,345
9				
10		CRITERIA TABLE		
11	Date	Location	Rep	
12	Jul-91	San Fran	CJ	
13		LA		

@DSTDS(A2..D8,3,B11..B13) = $9,952 (sample SD of California sales)

@DSTDS(A2..D8,3,C11..C12) = $4,899 (sample SD of CJ's sales)

Related topics

@DSUM - Database Sum

Syntax

@DSUM(Block, Column, Criteria)

Block	Cells (reference or name) containing the database, including field names.
Column	The number of the column containing the field you want to total (the first column in Block is 0, the second is 1, and so on).
Criteria	Cells containing search criteria; the first row must be field names.

@DSUM totals selected field entries in a database. It includes only those entries in Column whose records meet the criteria specified in Criteria.

The field specified in Criteria and the field you are finding the sum of need not be the same. The field you are finding the sum of is that contained within Column.

You can specify all or part of your database as Block, but field names must be included for each field you include in the cells.

Examples

	A	B	C	D
1				
2	DATE	LOCATION	REP	AMOUNT
3	Jul-91	San Fran	CJ	$14,999
4	Jul-91	LA	RX	$28,725
5	Jul-91	Chicago	RX	$18,600
6	Jul-91	NY	CJ	$15,600
7	Aug-91	Chicago	CJ	$23,769
8	Aug-91	LA	RX	$34,345
9				
10		CRITERIA TABLE		
11	Date	Location	Rep	
12	Jul-91	San Fran	CJ	
13		LA		

@DSUM(A2..D8,3,A11..A12) = $77,924 (total of July sales)

@DSUM(A2..D8,3,B11..B13) = $78,069 (total of California sales)

@DSUM(A2..D8,3,C11..C12) = $54,368 (total of CJ's sales)

@DSUM(A2..D8,1,A11..A12) = 0

@DSUM(A2..D8,4,A11..C13) = ERR (Column figure too high)

Related topics

@DURAT - Duration

Syntax

@DURAT(Discrate, Flows, <Initial>, <[Odd|Periods]>, <Simp>, <Pathdep>, <Filter>, <Start>, <End>)

Discrate	Discount rate or cells containing discount rates that correspond to cash flows stored in Flows.
Flows	Cells containing cash flows associated with the discount rates in Discrate.
Initial	Initial cash flow (the default is 0).
Odd|Periods	Delay between initial and first cash flow, in number of periods (the default is 1) or cells containing lengths of periods between cash flows (the default is 1).
Simp	Flag specifying how to discount: 0 = compounded discounting (default) 1 = mixed compounded and simple discounting 2 = simple discounting
Pathdep	Flag specifying whether to apply path-dependent compounding to each flow; 0 = no path (default); 1 = path.
Filter	Flag specifying filter type: 0 = no filter (default); 1 = cashflow < Start; 2 = cashflow ≤ Start; 3 = cashflow > Start; 4 = cashflow ≥ Start; 5 = Start < cashflow < End; 6 = Start ≤ cashflow ≤ End
Start	A starting cash flow amount to compare against individual flows.
End	An ending cash flow amount to compare against individual flows.

@DURAT calculates the duration of a specified cash flow structure. Duration (also called Macaulay duration) is defined as the weighted average time to receipt of a cash flow where the present values of the cash flows are the weights. Each weight in the sum is the present value of a cash flow divided by the net present value of all the cash flows.

@DURAT computes Macaulay duration using this formula:

$$Du = \frac{\sum_{i=1}^{n} Di_i * DF_i * Fl_i}{I + \sum_{i=1}^{n} DF_i * Fl_i}$$

where

$$Di_i = \sum_{j=1}^{i} Fl_i$$

Di	Distance
Du	Duration
Fl	Flows
I	Initial

n is the number of cash flows. DFi is the discount factor corresponding to the ith flow.

Modified (or Hicks) duration is defined as a sensitivity of present value to change in the internal rate of return. Modified duration is not defined for multiple discount rates (when Discrate is a selection of discount rates).

To convert Macaulay duration to Modified (or Hicks) duration, use this formula:

modified duration = Macaulay duration / (1 + Discrate)

Example

Consider a cash flow stream comprising four flows of $5, followed by four flows of $10, followed by seven flows of $11, followed by a final flow of $110. The first flow is 0.56745 periods away. The next 11 flows occur one period apart. The last four flows are 1.5 periods apart. This formula calculates the duration, assuming compound discounting, no initial cash flow, and the data shown in the next figure:

@DURAT(D8,A2..B5,B8,C2..D4) = 10.28273

	A	B	C	D
1	Cash Flows		Periods	
2	4	$5	1	0.56745
3	4	$10	11	1
4	7	$11	4	1.5
5	1	$110		
6				
7	Initial		Discount Rate	
8		0		7.85%

Related topics

@DURATION - Macaulay Duration

Syntax

@DURATION(Settle, Maturity, Coupon, Yield, <Freq>, <Calendar>)

Settle	Number representing the settlement date; must be < Maturity.
Maturity	Number representing the maturity date; must be > Settle.
Coupon	Coupon rate; must be ≥ 0.
Yield	Annual yield; must be > 0 and ≤ 1.
Freq	Frequency of coupon payments in the number of payments per year (can be 1, 2, 3, 4, 6, or 12; the default is 2).
Calendar	Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

@DURATION returns the Macaulay duration for a bond with an assumed par value of 100. Macaulay duration is the weighted average maturity of a bond's cash flow stream where the present values of all future cash receipts are used as weights.

Example

The following formula calculates the Macaulay duration of a bond with these terms: Settle is August 8, 1992, Maturity is November 15, 1998, Coupon is 9%, and Yield is 8.816%.

@DURATION(@DATE(92,8,8),@DATE(98,11,15),0.09,0.08816) = 4.836099

Related topics

@DVAR - Database Variance

Syntax

@DVAR(Block, Column, Criteria)

Block	Cells (reference or name) containing the database, including field names.
Column	The number of the column containing the field for which you want to compute variance (the first column in Block is 0, the second is 1, and so on).
Criteria	Cells containing search criteria; the first row must be field names.

@DVAR calculates the population variance for selected field entries in a database. @DVARS computes the variance of sample data.

@DVAR includes only those entries in Column whose records meet the criteria specified in Criteria.

The field specified in Criteria and the field for which you are calculating the variance need not be the same. The field analyzed is the field contained within Column.

You can specify all or part of your database as Block, but field names must be included for each field you include in the cells.

Examples

	A	B	C	D
1				
2	DATE	LOCATION	REP	AMOUNT
3	Jul-91	San Fran	CJ	$14,999
4	Jul-91	LA	RX	$28,725
5	Jul-91	Chicago	RX	$18,600
6	Jul-91	NY	CJ	$15,600
7	Aug-91	Chicago	CJ	$23,769
8	Aug-91	LA	RX	$34,345
9				
10		CRITERIA TABLE		
11	Date	Location	Rep	
12	Jul-91	San Fran	CJ	
13		LA		

@DVAR(A2..D8,3,B11..B13) = $66,028,355 (pop. variance of Calif. sales)

@DVAR(A2..D8,3,C11..C12) = $16,000,740 (pop. variance of CJ's sales)

@DVAR(A2..D8,4,A11..C12) = ERR (Column figure too high)

Related topics

@DVARS - Database Sample Variance

Syntax

@DVARS(Block, Column, Criteria)

Block	Cells (reference or name) containing the database, including field names.
Column	The number of the column containing the field for which you want to compute variance (the first column in Block is 0, the second is 1, and so on).
Criteria	Cells containing search criteria; the first row must be field names.

@DVARS calculates the sample variance for selected field entries in a database. @DVAR computes variance with population data.

This @function is not compatible with 1-2-3. To use the file in 1-2-3, use @DVAR instead.

Examples

	A	B	C	D
1				
2	DATE	LOCATION	REP	AMOUNT
3	Jul-91	San Fran	CJ	$14,999
4	Jul-91	LA	RX	$28,725
5	Jul-91	Chicago	RX	$18,600
6	Jul-91	NY	CJ	$15,600
7	Aug-91	Chicago	CJ	$23,769
8	Aug-91	LA	RX	$34,345
9				
10		CRITERIA TABLE		
11	Date	Location	Rep	
12	Jul-91	San Fran	CJ	
13		LA		

@DVARS(A2..D8,3,B11..B13) = $99,042,532 (sample variance of Calif. sales)

@DVARS(A2..D8,3,C11..C12) = $24,001,110 (sample variance of CJ's sales)

Related topics

@EFFECT - Effective Interest Rate

Syntax

@EFFECT(NomRate, Nper)

NomRate	Nominal interest rate.
Nper	Number of compounding periods per year, truncated to an integer.

@EFFECT calculates the effective annual interest rate for a specified nominal rate and number of compounding periods a year.

@EFFECT is related to @NOMINAL in the following way:

$$Re = \left(1 + \frac{Rn}{Nper}\right)^{(Nper - 1)}$$

where

Re	effective rate
Rn	nominal rate
Nper	number of compounding periods per year

@EFFECT returns ERR if either argument is non-numeric, if NomRate <= 0, or if Nper < 1.

Example

@EFFECT(7.18%,4) = 0.073756 or 7.3756%

Related topics

@EMNTH - Ending Day in Month

Syntax

@EMNTH(Date)

Date	Number representing a date. See "Using dates and times in Quattro Pro."

@EMNTH returns the serial date number for the date of the last day of the month in which Date falls.

Example

@EMNTH(@DATE(96,2,14)) = 35124 (February 29, 1996), the last day of the month in which February 14, 1996 falls.

Related topics

@EOMONTH - Last Day of Month

Syntax

@EOMONTH(StartDate, Months)

StartDate	Serial number of start date. See "Using dates and times in Quattro Pro."
Months	Number of months before or after StartDate: If Months is positive, @EOMONTHS returns a date after StartDate. If Months is negative, @EOMONTHS returns a date before StartDate. If Months is not an integer, @EOMONTHS truncates it.

@EOMONTH returns the serial date number for the last day of the month a specified number of months before or after StartDate. @EOMONTH lets you calculate maturity dates falling on the last day of the month.

@EOMONTH returns ERR if:

- StartDate is not a valid serial date number.
- StartDate plus Months yields an invalid serial date number.
- Either argument is non-numeric.

Examples

@EOMONTH(@DATE(96,2,14),4) = 35246 or 6/30/96

@EOMONTH(@DATE(96,2,14),-2) = 35064 or 12/31/95

Related topics

@ERFD - Error Function Derivative

Syntax

@ERFD(X)

X	A value from -26.6417 to 26.6417.

@ERFD(X) returns the derivative of the error function. It uses the following formula:

(2/@SQRT(@PI)) * @EXP(-X^2)

If X is less than -26.6417 or greater than 26.6417, @ERFD returns ERR because the calculation is too large to store.

Example

@ERFD(1) = 0.415107

Related topics

@ERF - Error Function

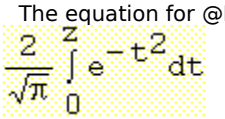
Syntax

@ERF(Lower, <Upper>)

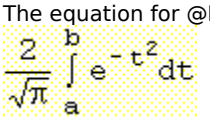
Lower	Lower bound for integrating @ERF; must be ≥ 0.
Upper	Upper bound for integrating @ERF; if omitted, @ERF integrates the error function between 0 and Lower; must be ≥ 0.

@ERF returns the error function integrated between Lower and Upper. The error function helps solve partial differential equations that involve convection or diffusion.

The equation for @ERF(z) is



The equation for @ERF(a,b) is



This is the same as @ERF(b) minus @ERF(a).

Example

@ERF(0,1) = 0.842701

Related topics

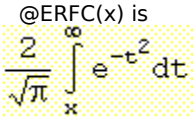
@ERFC - Complementary Error Function

Syntax

@ERFC(Lower)

Lower	Lower bound for integrating @ERF; must be ≥ 0.

@ERFC returns the complementary error function, which derives from the error function @ERF. The formula for @ERFC(x) is



This is the same as 1 - @ERF(x).
Therefore, @ERFC(Lower) = 1 - @ERF(Lower,Upper).

Example

@ERFC(1) = 0.157299

Related topics

@ERR - Error Value

Syntax

@ERR

@ERR returns the value ERR in the current cell and in any other cells that reference the current cell, either directly or indirectly. (Exceptions to this are @COUNT, @DCOUNT, @ISERR, @ISNA, @ISNUMBER, @ISSTRING, and @CELL formulas; these will not result in ERR if they reference an ERR cell.)

The ERR value resulting from this @function is the same as the ERR value produced by Quattro Pro when it encounters an error. It is often used with @IF to bring attention to error conditions.

ERR is a unique number, not to be confused with the label ERR.

Examples

@ERR = ERR

@IF(B6>B7,0,@ERR) = 0 (if B6>B7) or ERR (if B6<B7)

Related topics

@EVEN - Round Up to Even Number

Syntax

@EVEN(X)

X	Value to round.

@EVEN rounds X up (away from zero) to the nearest even integer, ignoring the sign of X. If X is already an even integer, @Even returns x unchanged.

Examples

@EVEN(3.2) = 4

@EVEN(-3.2) = -4

@EVEN(8) = 8

Related topics

@EXACT - Test If Values Are Exactly Alike

Syntax

@EXACT(String1, String2)

String1	A valid string value.
String2	A valid string value.

@EXACT compares the values of String1 and String2. If the values are exactly identical, including capitalization and diacritical marks (such as ~), it returns 1. If there are any differences, it returns 0.

If you are comparing literal strings, surround them with double quotes. If you use a cell name or cell address, no quotes are necessary. You can compare the contents of label cells only. If you try to compare one or more numbers or empty cells, the result is ERR. When you compare labels, label prefixes are ignored.

To compare strings or cell contents without regard to capitalization or diacritical marks, use @IF. For example, @IF(C3=B3,1,0) returns 1 if the contents of the cells are the same but are capitalized differently.

Examples

@EXACT("client","Client") = 0

@EXACT("client","client") = 1

@EXACT(29,"29") = ERR (the first string is a value)

@EXACT(A1,"yes") = 1 (if A1 contains the label yes)

@EXACT(client,client) = syntax error (no quotation marks)

@EXACT("client","client","client ") = syntax error (more than two strings)

Related topics

@EXP - *e* Raised to *X* Power

Syntax

@EXP(X)

X	A numerical value equal to or less than 709.

@EXP returns the mathematical constant e, raised to the Xth power. This @function is the inverse of a natural logarithm, @LN.

Examples

@EXP(3.4) = 29.9641000474

@EXP(1) = 2.718281828459 (the actual value of e)

@SQRT(@EXP(2)) = 2.71828183

@LN(@EXP(2.5)) = 2.5

Related topics

@EXP2 - e Raised to -X^2 Power

Syntax

@EXP2(X)

X	A value from -26.6417 to 26.6417.

@EXP2 calculates the value of the constant e raised to the power (-X^2). The constant e equals 2.718281828459.

If X is less than -26.6417 or greater than 26.6417, @EXP2 returns ERR.

Example

@EXP2(1) = 0.367879

Related topics

@EXPONDIST - Exponential Distribution

Syntax

@EXPONDIST(X, Lambda, Cum)

X	Value at which to evaluate the function; must be ≥ 0.
Lambda	Value to indicate; Lambda = 1/Mean; must be > 0.
Cum	1 to perform cumulative distribution function; 0 to perform the probability density function.

The exponential distribution, sometimes called the waiting-time distribution, describes the amount of time or distance between the occurrence of random events. For example, it can be used to find the time between major earthquakes or the time between no hitters pitched in major league baseball. The exponential distribution calculated by @EXPONDIST is a continuous distribution with a probability density function whose formula is:

$$f(X; \lambda) = \lambda e^{-\lambda x}$$

For the cumulative distribution function, the formula is:

$$F(X; \lambda) = 1 - e^{-\lambda x}$$

Use this distribution in connection with estimating the length of material life, or the length of time a process might take.

Examples

On average, customers at a certain bank must wait 2 minutes before being served by a teller. This formula calculates the probability that someone would have to wait 3 minutes:

@EXPONDIST(3,1/2,0) = 0.111565

This formula calculates the probability that someone would wait only 1 minute for a teller:

@EXPONDIST(1,1/2,1) = 0.393469

Related topics

@FACT - Factorial

Syntax

@FACT(N)

N	Integer ≥ 0 specifying the factorial to calculate.

@FACT calculates the factorial of a number. N! is defined as follows: if N ≥ 0,
N! = N × (N-1) × (N-2) × (N-3) ×...× (2) × (1)
@FACT(0) returns 1. If N is a non-integer or negative number, @FACT returns ERR.

Examples

@FACT(10) = 3628800
@FACT(128) = 3.9E+215

Related topics

@FACTDOUBLE - Double Factorial

Syntax

@FACTDOUBLE(N)

N	Value ≥ 0 to calculate factorial of.

@FACTDOUBLE returns the double factorial of N. N!! is defined as follows:
If N is even, N!! = N(N-2)(N-4)...(4)(2)
If N is odd, N!! = N(N-2)(N-4)...(3)(1)
If N is negative, @FACTDOUBLE returns ERR.

Examples

@FACTDOUBLE(12) = 46080
@FACTDOUBLE(13) = 135135

Related topics

@FACTLN - Natural Logarithm of Factorial

Syntax

@FACTLN(n)

n	Integer from 0 through 170.

@FACTLN returns the natural logarithm of the factorial of n. The factorial of n is the product of all positive integers from 1 to n. The factorial of 0 is 1 by definition.

Example

@FACTLN(4) = 3.178054

Related topics

@FALSE - Logical Value 0

Syntax

@FALSE

@FALSE returns the logical value 0 and is usually used in @IF formulas. The zero that it returns is the same as any other zero, but @FALSE makes the formula easier to read.

@TRUE is a related @function.

Examples

@FALSE = 0

@IF(C3=100,10,@FALSE) = 10 (if C3 = 100) or 0 (if C3 is not equal to 100)

@IF(C3=100,@TRUE,@FALSE) = 1 (if C3 = 100) or 0 (if C3 is not equal to 100)

Related topics

@FBDAY - First Business Day

Syntax

@FBDAY(Date, <Holidays>, <Saturday>, <Sunday>)

Date	Number representing a date. See "Using dates and times in Quattro Pro."
Holidays	Cells containing dates that are holidays or the date of a single holiday or 0 to indicate no holidays (the default is 0).
Saturday	0 to specify that Saturday is not a business day; 1 to specify that Saturday is a business day (the default is 0).
Sunday	0 to specify that Sunday is not a business day; 1 to specify that Sunday is a business day (the default is 0).

@FBDAY returns the serial date number of the first business day of the month in which Date falls. If the first of the month is not a business day, @FBDAY returns the business day closest to it within the same month.

Example

This formula calculates the first business day in January 1997, assuming that Saturdays, Sundays, and some dates are holidays:

	A
1	01/01/97
2	01/02/97
3	01/03/97

@FBDAY(@DATE(97,1,1),A1..A3,0,0) = 35436 (which is Monday, January 6, 1997)

Related topics

@FDIST - F-Distribution

Syntax

@FDIST(X, DegFreedom1, DegFreedom2)

X	Positive value at which to evaluate the function.
DegFreedom1	Numerator degrees of freedom; must be ≥ 1.
DegFreedom2	Denominator degrees of freedom; must be ≥ 1.

@FDIST returns the cumulative F-distribution function, which is the probability that a random variable will be less than X. Use @FDIST to compare two population variances.

Example

@FDIST(6.256057,5,4) = 0.05

Related topics

@FEETBL - Fee Table

Syntax

@FEETBL(Tu, Ppu, [StdTbl|Val], <[MinTbl|Val]>, <[MaxTbl|Val]>, <RndPlcs>)

Tu	Total units; if Tu is negative, @FEETBL uses its absolute value.
Ppu	Price per unit.
StdTbl|Val	Fee table or a single value that defines the standard fee calculation.
MinTbl|Val	Fee table or a single value that defines the minimum fee calculation (if omitted, MinTbl equals StdTbl).
MaxTbl|Val	Fee table or a single value that defines the maximum fee calculation (if omitted, MaxTbl equals StdTbl).
RndPlcs	Number of places to which the final result is rounded; can be from 0 to 10 places (the default is no rounding).

@FEETBL returns fee calculations from tables. You can use @FEETBL to calculate fees or commissions for many types of stock transactions, taxes, sales commissions, and other types of fees and charges. To use @FEETBL, you need to create a table (or tables) that describes the fees.

@FEETBL is more powerful than other table lookup @functions such as @HLOOKUP and @VLOOKUP because it allows you to

- Compare the standard fee with minimum and maximum values
- Multiply the lookup value by the number of units or total price
- Add a fixed value to the fee
- Round the result to a specified number of decimal places

If the fee table is indexed by values of total units or price per unit, Tu or Ppu must be greater than the smallest value in the index; otherwise, @FEETBL cannot find a lookup value. If either Tu or Ppu is zero, @FEETBL returns zero.

If you specify an optional argument, such as RndPlcs, you must also specify all preceding optional arguments. If MinTbl and MaxTbl are not pertinent to the fee calculation, use StdTbl again for MinTbl and MaxTbl, or enter values that have no effect on the final result. For example, enter 0 for MinTbl and 1E+99 for MaxTbl.

The upper left cell of a fee table must contain a table header string that identifies the row index, column index, and cell contents of the table, and also specifies if the table contains an additive factor for the fee calculation. The table header string consists of three or four parameters separated by a space; each parameter has several possible values.

For valid comparison, values for StdTbl, MinTbl, and MaxTbl arguments must have the same units.

Table header parameters

Parameter	Description	Values
1	Row index	tu, ppu, tp, na
2	Column index	tu, ppu, tp, na
3	Cell contents	fpu, fpct, luo
4	Additive factor	fa

Description Values

tu	total units
fpu	fee per unit
ppu	price per unit
fpct	fee percentage

tp	total price
luo	lookup only
na	not applicable
fa	fixed adder

The first parameter of the table header identifies the contents of the row index, which appears in the first column of the table below the table header. The second parameter identifies the contents of the column index, which appears in the first row of the table to the right of the table header.

The third parameter of the table header determines if @FEETBL multiplies the lookup value from the table by another value. For example, "fpu" (fee per unit) indicates that @FEETBL multiplies the lookup value by the number of units; "fpct" (fee percentage) indicates that the lookup value is a percentage that @FEETBL multiplies by the total price; "luo" (lookup only) indicates that @FEETBL uses the lookup value without modification.

The fourth parameter of the table header is an optional additive factor; specify "fa" (fixed adder) to add a value to the result of the operation specified by the third parameter. If the fee table has no additive factor, omit the fourth parameter.

In the next figure, the table header in cell A3 is "tp na fpct fa"; "tp" indicates that A4..A9 represents the row index values for total price; "na" indicates that B3..C3 has no column index values; "fpct" indicates that the lookup values in B4..B9 are percentages that must be multiplied by the total price; "fa" indicates that the values in C4..C9 are "fixed adders", that is, one of these values must be added to the product of the fee percentage and the total price.

	A	**B**	**C**
1	Standard Commission Rate Table		
2	Principal	%Fee +	Fixed Adder
3	tp na fpct fa		
4	$0	1.60%	$26.00
5	$2,500	0.60%	$51.00
6	$6,000	0.30%	$69.00
7	$22,000	0.20%	$91.00
8	$50,000	0.10%	$141.00
9	$500,000	0.08%	$241.00

In the next figure, the table header in cell A2 is "tu tp luo"; "tu" indicates that A3..A7 represents the row index values for total units; "tp" indicates that B2..E2 represents the column index values for total price; "luo" indicates that @FEETBL uses the lookup values in B3..E7.

	A	**B**	**C**	**D**	**E**
1	Total Units			Total Price	
2	tu tp luo	$0	$10,000	$15,000	$20,000
3	0	$250	$500	$750	$1,000
4	2	$200	$400	$600	$800
5	5	$175	$350	$525	$700
6	10	$150	$300	$450	$600
7	20	$125	$250	$375	$500

@FEETBL treats all string values (other than the table header) or empty cells in fee tables as zero.

Examples

A furniture manufacturer sells 100 bookcases at a price of $150 each to a retailer. This formula calculates the handling fee for the order based on the fee table in the next figure.

@FEETBL(100,150,A1..D5) = $300

Cell A4 is the row index value for 100 total units. Cell C1 is the column index value for $150 price per unit. Cell C4 is the lookup value, which is multiplied by the total units: $3 * 100 = $300.

This formula calculates the handling fee based on the fee table in the next figure for a sale of 5 end tables at a price of $75 each:

@FEETBL(5,75,A1..D5) = $25

Cell A2 is the row index value for 5 units (between 0 and 9). Cell B1 is the column index value for $75 price per unit. Cell B2 is the lookup value, which is multiplied by the total units: $5 * 5 = $25.

	A	B	C	D
1	tu ppu fpu	$0	$100	$500
2	0	$5	$7	$8
3	10	$4	$5	$6
4	100	$2	$3	$4
5	1000	$2	$1	$2

Related topics

@FIB - Fibonacci Sequence

Syntax

@FIB(N)

N	Integer ≥ 0 specifying the desired term of a Fibonacci sequence.

@FIB calculates the Nth term of a Fibonacci sequence (1, 1, 2, 3, 5, 8, 13, 21...), in which each number, after the first two, is the sum of the two numbers immediately preceding it. @FIB(0) is defined to be 0.

Examples

@FIB(4) = 3

@FIB(9) = 34

@FIB(15) = 610

Related topics

@FIELD - Nth Substring in a String

Syntax

@FIELD(String,N,<Delimiter>)

String	A string value containing two or more delimited substrings, or a cell reference to a delimited string value.
N	Number of the substring you want to find (the first substring is numbered 1, the second 2, and so on).
Delimiter	Optional delimiter character; if you do not specify a delimiter, Quattro Pro uses the delimiter character specified in the Application International property.

@FIELD finds the nth substring in String, a delimited list of strings. You can use @FIELD to find a single property value from a delimited list of properties returned by the @PROPERTY function. You can also use it to return a specific field from an imported text file.

Example

The cell in A1 contains the following formula:

@PROPERTY("Active_Block.Shading")

The shading property has not been changed from the default setting, so the formula returns

3,0,Blend7

You can use @FIELD with @PROPERTY to return the third substring in the comma-delimited string:

@FIELD(@PROPERTY("Active_Block.Shading"),3) = Blend7

@FIELD(@PROPERTY("Active_Block.Shading"),3,",") = Blend7

Related topics

@FILEEXISTS - Test If File Exists

Syntax

@FILEEXISTS(FileName)

FileName	Any file name.

@FILEEXISTS returns a 1 if a file named FileName exists in the current file directory, and returns a 0 if it does not. FileName can be a cell name containing a path or file name string. If entered as a literal string, FileName must be enclosed by quotes and must include any extension attached to the file name. To search for a file in a directory other than the default directory, include the directory path in FileName.

Examples

@FILEEXISTS("EXAMPLE.QPW") = 1 (if EXAMPLE.QPW is in the working directory)

@FILEEXISTS("C:\DATA\EXAMPLE.QPW") = 1 (if EXAMPLE.QPW is in the specified directory)

@FILEEXISTS(FILE_NAME) = 1 (if the selection FILE_NAME contains a path and file-name label and if the file exists in that directory)

Related topics

@FIND - Search for String

Syntax

@FIND(Substring, String, StartNumber)

Substring	A valid string value, representing the value to search for.
String	A valid string value, representing the value to search through.
StartNumber	A numeric value ≥ 0, representing the character position to begin searching with; 0 = the first character.

@FIND searches through String from left to right for Substring. If it finds Substring, it returns the character position of the first occurrence. StartNumber indicates where to begin the search: 0 = the first character in the string, 1 = the second, and so on. The value of StartNumber must not be more than the number of characters in String minus 1.

@FIND is case-sensitive and is also sensitive to diacritical marks used in non-English languages. You can overcome the case sensitivity of this @function by using @UPPER to force one or more of the strings into all uppercase letters. For example, the following formula forces both the substring in cell C3 and the string in cell C4 to uppercase, then searches for the substring:

@FIND(@UPPER(C3),@UPPER(C4),0)

@FIND is most often used in conjunction with two other string functions: @REPLACE (to perform "search and replace" operations on strings) and @MID (to access substrings).

If @FIND fails to find any occurrences of Substring, or if the StartNumber given is invalid, the result is ERR.

Examples

@FIND("i","find",0) = 1

@FIND("nd","find",2) = 2

@FIND("F","find",0) = ERR

@FIND("f","find",3) = ERR

@FIND("d","find",4) = ERR

@FIND(n,find,0) = syntax error (quote marks omitted from strings)

@FIND("hi",C4,0) = 1 (if C4 contains ship)

Related topics

@FINV - Inverse of F-Distribution

Syntax

@FINV(Prob, DegFreedom1, DegFreedom2)

Prob	Cumulative probability value; must be ≥ 0 and ≤ 1.
DegFreedom1	Numerator degrees of freedom; must be ≥ 1.
DegFreedom2	Denominator degrees of freedom; must be ≥ 1.

@FINV returns the inverse of the cumulative F-distribution function. Use this function to measure the degree of variability in two data sets.

Example

@FINV(0.05,5,4) = 6.256057

Related topics

@FIRSTBLANKPAGE - Name of First Blank Page

Syntax

@FIRSTBLANKPAGE(Block)

Block	A cell or reference; can be a link to another opened notebook (for example, [BUDGET]A:A1).

@FIRSTBLANKPAGE returns a string that contains the letters for the first unnamed blank sheet in a notebook that is not part of a group.

Quattro Pro searches for the first unnamed blank sheet (that is not in a group) starting at sheet A and continuing toward sheet IV. If there are no unnamed blank sheets (or they are all in groups), @FIRSTBLANKPAGE returns ERR.

Example

@FIRSTBLANKPAGE(B17) = "AA" (if it is the first sheet that is blank and unnamed)

Related topics

@FIRSTINGROUP - First Sheet in Group

Syntax

@FIRSTINGROUP(Block, GroupName)

Block	A block of cells to check in the notebook.
GroupName	A string value representing a group name.

@FIRSTINGROUP returns a string that contains the letters for the first sheet in the group named GroupName. @FIRSTINGROUP searches the notebook referenced by Cell for the group. If the group does not exist, @FIRSTINGROUP returns ERR.

Example

@FIRSTINGROUP([REPORTQ4]A:C12,"Totals") = "A" (if the notebook REPORTQ4 contains a group named Totals that starts with sheet A)

Related topics

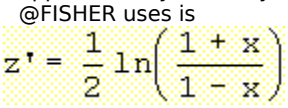
@FISHER - Fisher Transformation

Syntax

@FISHER(X)

X	Numeric value; -1 < X < 1.

@FISHER returns the Fisher transformation at the value X. Fisher's z-transformation is used to produce an approximately normally distributed variable (rather than skewed) from the correlation coefficient. The formula @FISHER uses is



Example

@FISHER(0.25) = 0.255413

Related topics

@FISHERINV - Inverse of Fisher Transformation

Syntax

@FISHERINV(Y)

Y	Numeric value ≤ 354 for which you want the inverse of the Fisher transformation.

@FISHERINV returns the inverse of the Fisher transformation. Use @FISHERINV to determine the confidence limits for a correlation coefficient.

Example

@FISHERINV(0.255413) = 0.25

Related topics

@FIXED - Decimal Number as Text

Syntax

@FIXED(Num, <Dec>, <NoCommas>)

Num	Number to be rounded and converted to text.
Dec	Number of decimal places to be displayed. If Dec is negative, @FIXED rounds off Num to the left of the decimal point. If you omit Dec, @FIXED rounds off to 2 decimal places.
NoCommas	A logical value: 1 = do not display thousands separators 0 = display thousands separators (the default, if omitted, or if NoCommas ≠ 1)

@FIXED rounds a number to a specified number of decimals, formats it, and displays the result as text. Your display will depend on your Application International Property settings.

Num cannot have more than 15 significant digits.

You can also format numeric values by right-clicking the cell and choosing Cell Properties, then Fixed, then entering the number of decimal places to display. The result looks the same, but the value remains a numeric value when you use Cell Properties, while @FIXED converts it to text.

When you use any optional argument, you must also use the ones before it.

Examples

@FIXED(9449.985, 2) = "9,449.99"

@FIXED(9449.985, 2, 1) = "9449.99"

@FIXED(9449.985, 0) = "9,450"

@FIXED(9449.985, -3) = "9,000"

Related topics

@FLOOR - Round Down to Nearest Multiple

Syntax

@FLOOR(X, Y)

X	Value to round.
Y	Value to make rounded X evenly divisible by.

@FLOOR rounds X down (toward zero) to the nearest value that is evenly divisible by Y. If X and Y have different signs, the result of @FLOOR is ERR.

Examples

@FLOOR(3.2,3) = 3

@FLOOR(-3.2,-3) = -3

Related topics

@FORECAST - Linear Regression Forecast

Syntax

@FORECAST(X, KnownY, KnownX)

X	Numeric value at which to evaluate the function.
KnownY	Dependent range of values.
KnownX	Independent range of values.

@FORECAST returns a predicted Y value corresponding to X based upon a linear regression of KnownY and KnownX.

KnownY and KnownX must contain the same number of values. The variance of KnownX must not be 0.

Example

This example refers to cells in the figure below.

@FORECAST(1000,C2..C16,B2..B16) = $15,868.50

	A	B	C
1	Date	Advertising	Sales
2	04/30/93	$435	$7,000
3	05/07/93	$400	$6,000
4	05/14/93	$505	$7,767
5	05/21/93	$470	$7,800
6	05/28/93	$610	$9,534
7	06/04/93	$540	$7,750
8	06/11/93	$575	$8,945
9	06/18/93	$715	$11,301
10	06/25/93	$645	$9,465
11	07/02/93	$680	$10,760
12	07/09/93	$785	$13,000
13	07/16/93	$750	$11,890
14	07/23/93	$855	$12,980
15	07/30/93	$820	$13,068
16	08/06/93	$890	$14,246

Related topics

@FRACD - Fraction to Decimal

Syntax

@FRACD(Frac, Denom)

Frac	Number to be converted.
Denom	Denominator; must be an integer > 0.

@FRACD converts the fraction Frac to a decimal number. For example, you can use this @function to convert a number with a fractional portion in 32nds to a decimal number. @FRACD reverses the effect of @DFRAC.

Frac looks like a decimal, but @FRACD does not use it that way. The portion to the right of the decimal point is the numerator of the fraction using the denominator specified by Denom. For example, if Denom is 32 and you want to find the decimal equivalent of 99 12/32, set Frac to 99.12. If Denom were 100, setting Frac to 99.12 represents 99 12/100.

Using a value of @FRACD(1.1,32) computes as 1 10/32. If you want 1 1/32 you must use @FRACD(1.01,32).

Example

This formula finds the decimal equivalent of 106 14/32.

@FRACD(106.14,32) = 106.4375

This formula finds the decimal equivalent of 1 1/32.

@FRACD(1.01,32)= 1.03125

Related topics

@FRACTION - Decimal to Fraction

Syntax

@FRACTION(Value, <Denom>, <ForceDenom>)

Value	Decimal value to be converted.
Denom	Denominator; must be an integer > 0. If you specify a denominator that doesn't allow an exact fraction, the numerator is rounded to the nearest whole number. For example, @FRACTION(100.25, 5) would display 100 1/5.
ForceDenom	When ForceDenom is not specified, the fraction is displayed in its lowest common denominator form; use 1 to display the denominator value specified in <Denom>.

@FRACTION converts a decimal value to a fraction using the specified denominator.

If you specify @FRACTION(Value, <Denom>), the fraction is created based on the denominator value, then displayed in the lowest common denominator form. If you want the denominator value displayed, even if it's not the lowest common denominator, specify 1 for ForceDenom.

If no <Denom> is specified, @fraction will round to the nearest 64th.

The fraction created from @FRACTION is a string value. You can reference the fraction in a formula by using it with the @VALUE function. For example, if the fraction is computed in B4, you can use it in a formula as @VALUE(B4).

Example

@FRACTION(100.5,4) = 100 1/2

@FRACTION(100.63,8) = 100 5/8

@FRACTION(100.5,8,1) = 100 4/8

@FRACTION(100.75,16,1) = 100 12/16

Related topics

@FREQDIST - Frequency Distribution

Syntax

@FREQDIST(Data, Intervals)

Data	Cells of values for which you want to count frequencies.
Intervals	Array of or reference to intervals into which you want to group the values.

@FREQDIST calculates a frequency distribution, displaying it as a vertical array. A frequency distribution reports how many of the specified values occur in each of the specified intervals.

The Intervals argument tells @FREQDIST the upper boundary of each interval, so the result will be an array one greater than the number of cells in Intervals.

- Intervals must be in ascending order.
- If Data contains no values, @FREQDIST returns an array of zeros.
- If Intervals contains no values, @FREQDIST returns the number of elements in Data.
- @FREQDIST ignores blank cells and text.

Examples

Your consulting income is recorded in a cell area named INCOME. You want to know how many months your income is $100 or below, from $100 to $500, from $500 to $800, and over $800.

INCOME	**A**	**B**	**C**	**D**
1				
2	January	February	March	Intervals
3	$652	$833	$599	100
4	$456	$305	$522	500
5	$68	$59	$73	800

@FREQDIST(A3..C5,D3..D5) returns

7	3
8	2
9	3
10	1

If you named the cells A3..C5 INCOME and cells D3..D5 INTERVALS, you could also enter the formula as:

@FREQDIST(INCOME,INTERVALS) = {3| 2| 3| 1}

If the intervals are not in cells, you can establish them in the formula:

@FREQDIST(INCOME,{100,500,800}) = {3| 2| 3| 1}

Related topics

@FTEST - F-test

Syntax

@FTEST(Array1, Array2)

Array1	First array of numeric values.
Array2	Second array of numeric values.

@FTEST returns the results of an F-test run against the samples in Array1 and Array2. An F-test is a one-tailed probability that the differences in the sample variances in Array1 and Array2 are different. Use @FTEST to determine if two samples have significantly different variances (that is, if data sets were drawn from different parent populations).

Array1 and Array2 must have more than two values. The variance of Array1 or Array2 must not be zero.

Example

@FTEST({75,82,83,85,85,90},{80,86,92,93,95,96}) = 0.637248

Related topics

@FULLP - Convert Half-Width String

Syntax

@FULLP(String)

String	The single-byte (half-width) character string

@FULLP converts a single-byte character string to a full-width, double-byte character string in a label. Double-byte characters are used in software localized to most Far Eastern languages (for example, Japanese, Chinese, and Korean). The localized machine will display a toolbar that lets you select various single and double-byte character sets from within Quattro Pro.

Using @FULLP in a cell label allows you to convert ASCII text characters to double-byte characters. @FULLP does not convert double-byte character set (DBCS) characters.

You cannot use @FULLP to convert single-byte Katakana characters to double-byte Katakana.

Related topics

@FUTV - Future Value of Cash Flow

Syntax

@FUTV(Intrate, Flows, <[Odd|Periods]>, <Simp>, <Pathdep>, <Filter>, <Start>, <End>)

Intrate	Interest rate or cells containing interest (discount) rates.
Flows	Cells containing cash flows.
Odd|Periods	Delay after last cash flow in number of periods (the default is 0) or cells containing lengths of periods between cash flows (the default is 1).
Simp	Flag specifying how to discount: 0 = compounded discounting (default) 1 = mixed compounded and simple discounting 2 = simple discounting
Pathdep	Flag specifying whether to apply path-dependent compounding to each flow; 0 = no path (default); 1 = path.
Filter	Flag specifying filter type: 0 = no filter (default); 1 = cashflow < Start; 2 = cashflow ≤ Start; 3 = cashflow > Start; 4 = cashflow ≥ Start; 5 = Start < cashflow < End; 6 = Start ≤ cashflow ≤ End.
Start	A starting cash flow amount to compare against individual flows.
End	An ending cash flow amount to compare against individual flows.

@FUTV calculates the future value of a specified cash flow structure. The future value of a stream of cash flows is the sum of the future values of each cash flow.

By default, @FUTV computes the future value at the time of the last cash flow. If you specify Periods, @FUTV calculates the future value at a time one period after the last cash flow. If you specify Odd, @FUTV calculates the future value at a time Odd periods after the last cash flow.

@FUTV computes future value using this formula:

$$FV = \sum_{i=1}^{n} Fl_i * IF_i$$

where n is the number of cash flows, and IFi is the interest factor associated with the ith cash flow, Fli. IFi is the @FUTV counterpart of the discount factor, DFi used in @NETPV. Unlike DFi, which reduces the value of a flow, IFi increases the value.

FV	Future Value
Fl	Flows

Example

Suppose a portfolio has a bond that will make 15 annual interest payments of $1,500, and pay $20,000 in principal along with the last interest payment. If the interest earned on investing the annual interest payments (the reinvestment rate) is 8.5%, this formula calculates the amount in the portfolio at the end of 15 years, using the data shown in the next figure:

@FUTV(D2,A2..B3) = $62,348.40

	A	B	C	D
1	Cash Flows		Interest Rate	
2	14	$1,500		8.5%
3	1	$21,500		

Related topics

@FV - Future Value of Investment

Syntax

@FV(Pmt, Rate, Nper)

Pmt	A numeric value representing the amount of the periodic payment.
Rate	A numeric value > -1, representing the periodic interest rate (the fixed interest rate per compounding period).
Nper	Number of periods, which should be an integer ≥ 2.

@FV returns the future value of an investment where Pmt is invested for Nper periods at the rate of Rate per period. @FV calculates the future value with this formula:

$$P\frac{(1+R)^N-1}{R}$$

where

P	amount of periodic payment
R	periodic interest rate
N	number of periods

An equivalent for this formula using @FVAL is

@FVAL(Rate, Nper, - Pmt, 0)

@FV assumes that the investment is an ordinary annuity. @FVAL, a related @function, uses an optional argument, Type, to indicate whether the investment is an ordinary annuity or an annuity due.

Examples

Assume you want to set aside $500 at the end of each year in a savings account that earns 15% annually. To determine what the account will be worth at the end of six years, enter this formula:

@FV(500,15%,6)

Your yearly payment of $500 will be worth $4,376.87 in six years. You could also use @FVAL:

@FVAL(15%,6,-500,0,0)

Note that in @FVAL, you have to be precise about whether a payment is out of your pocket (a negative number) or paid to you (a positive number).

Other examples:

@FV(200,0.12,5) = $1,270.57

@FV(500,0.9,4) = $6,684.50

@FV(800,0.9,3) = $5,208.00

@FV(800,0.9,A3) = $40,929.67 (if A3 = 6)

Related topics

@FVAL - Future Value of Investment

Syntax

@FVAL(Rate, Nper, Pmt, <Pv>, <Type>)

Rate	A numeric value > -1, representing the periodic interest rate (the fixed interest rate per compounding period).
Nper	Number of periods, which should be an integer > 0.
Pmt	A numeric value representing the amount of the periodic payment.
Pv	A numeric value representing the current value of an investment (the present value).
Type	An optional numeric value that indicates whether payments or cash flows occur at the beginning (1) or the end (0) of the period; default = 0.

Like the related @function @FV, @FVAL returns the future value of an investment. The last two arguments, Pv and Type, are optional. If you omit the last one or both of them, Quattro Pro assumes their values are zero. These arguments let you define the problem as an annuity due (putting money into an account before it earns its interest for that year means you have an annuity due to you, which increases the future value). Be sure to enter negative numbers for money going out and positive numbers for money coming in to you.

This @function is not compatible with 1-2-3. If your file must be compatible with 1-2-3, use @FV instead.

Examples

Assume you want to set aside $500 at the start of each year in a savings account that earns 15% annually. To determine what the account will be worth at the end of six years, starting at a present value of zero, enter this formula:

@FVAL(15%,6,-500,0,1)

Note that the payment is out of your pocket, so you enter a negative number. Your yearly payment of $500 will be worth $5,033.40 in six years, or $656.53 more than if you deposited the money at the last day of the year as in the example for @FV.

If the account already had $340 in it before your yearly deposits of $500, you could calculate the future value after six years with this formula:

@FVAL(15%,6,-500,-340,1) = $5,819.84

Related topics

@GAMMA - Gamma Function

Syntax

@GAMMA(X)

X	Any positive number, not 0.

@GAMMA calculates the gamma function. It approximates the gamma function accurately to within six significant figures.

The gamma function has the property that @GAMMA(X+1) = X * @GAMMA(X). Or, @GAMMA(n+1) = n!

@GAMMA returns ERR if X is greater than approximately 171.6242.

Examples

@GAMMA(3) = 2

@GAMMA(4) = 6

@GAMMA(5) = 24

Related topics

@GAMMADIST - Gamma Distribution

Syntax

@GAMMDIST(X, Alpha, Beta, Cum)

X	Value at which to evaluate the function; must be ≥ 0.
Alpha	Parameter to the gamma distribution; must be > 0.
Beta	Parameter to the gamma distribution; must be > 0.
Cum	1 to return the cumulative gamma distribution function; 0 to return the probability density function.

@GAMMDIST returns the gamma distribution function, which is the probability that a random variable will be less than X. Use @GAMMADIST to study random variables characterized by skewed and asymmetric distributions.

When Alpha = 1, @GAMMADIST returns the exponential distribution; see @EXPONDIST.

Examples

@GAMMADIST(18,8,2,1) = 0.676103

@GAMMADIST(18,8,2,0) = 0.058558

Related topics

@GAMMAINV - Inverse of Gamma Distribution

Syntax

@GAMMAINV(Prob, Alpha, Beta)

Prob	Probability associated with the gamma cumulative function; must be ≥ 0 and ≤ 1.
Alpha	A parameter to the gamma distribution; must be > 0.
Beta	A parameter to the gamma distribution; must be > 0.

@GAMMAINV returns the inverse of the cumulative gamma distribution function.

Example

@GAMMAINV(0.676103,8,2) = 18

Related topics

@GAMMALN - Natural Logarithm of Gamma Function

Syntax

@GAMMALN(X)

X	Value for which you want to calculate @GAMMALN; must be > 0.

@GAMMALN returns the natural logarithm of the gamma function. Use @GAMMALN to build other common statistical functions such as the beta function (see @BETA) and the factorial function (see @FACT).

Example

@GAMMALN(6) = 4.787492

Related topics

@GAMMAP - Incomplete Gamma Function

Syntax

@GAMMAP(A, X)

A	Parameter to the function; must be > 0.
X	Value at which to evaluate the function; must be ≥ 0.

@GAMMAP returns the incomplete gamma function, also known as the standard cumulative gamma distribution. @GAMMAP is equal to the cumulative gamma distribution when $\beta = 1$.

Example

@GAMMAP(3,4) = 0.761897

Related topics

@GAMMAQ - Complement to Incomplete Gamma Function

Syntax

@GAMMAQ(A, X)

A	Parameter to the function; must be > 0.
X	Value at which to evaluate the function; must be ≥ 0.

@GAMMAQ is a complement to the incomplete gamma function and equals (1 - @GAMMAP).

Example

@GAMMAQ(3,4) = 0.238103

Related topics

@GCD - Greatest Common Divisor

Syntax

@GCD(X, Y)

X	Integer to find greatest common divisor of.
Y	Integer to find greatest common divisor of.

@GCD returns the greatest common divisor of X and Y (the largest integer that both numbers can be divided by without a remainder.

@GCD should not be confused with the greatest common denominator which is @LCM.

Examples

@GCD(96,78) = 6

@GCD(112,42) = 14

@GCD(-9,-3) = 3

Related topics

@GEOMEAN - Geometric Mean

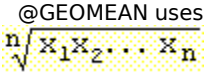
Syntax

@GEOMEAN(List)

List	One or more numeric or values; values in List must be positive.

@GEOMEAN returns the geometric mean of a positive range of values. The geometric mean is the nth root of the product of a series of numbers. Use @GEOMEAN when you are interested in an average rate of change of values in a data set given a varying rate of change.

@GEOMEAN uses this formula:



Example

@GEOMEAN(3,4,5,6,7) = 4.789389

@GEOMEAN(C10..C14) = 1.129486, where C10=1.15, C11=1.08, C12=1.13, C13=1.18, and C14=1.11

Related topics

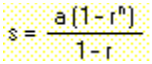
@GEOSUM - Geometric Series

Syntax

@GEOSUM(FirstTerm, Terms, Ratio)

FirstTerm	First term of the series.
Terms	Number of terms in the series.
Ratio	Common ratio of the series.

@GEOSUM calculates the geometric series that is sum of the terms of a geometric sequence of a number of terms (n) based on the first term and common ratio. The notion of a geometric series is the basis of the mathematical model of an annuity. @GEOSUM uses the formula:



where

s	geometric series
a	first term
r	common ratio
n	number of terms

The formula assumes: $r \neq 1$; if r = 1, @GEOSUM returns NA.

Examples

If you invest $2,000 at 3.5% interest compounded annually, the list of compound amounts at the end of each year for 5 years is

2000*1.035, 2000*1.035^2, 2000*1.035^3, 2000*1.035^4, 2000*1.035^5

This is a geometric series with the common ration 1.035.

@GEOSUM(2000*1.035,5,1.035) = $11,100.30

Related topics

@GESTEP - Test if X >= Y

Syntax

@GESTEP(X, <Y>)

X	Numeric value to check.
Y	Numeric value that X must exceed for function to return 1 (if omitted, assumed to be 0).

@GESTEP tests whether X is greater than or equal to Y. If it is, @GESTEP returns 1 (true); if not, @GESTEP returns 0 (false).

Examples

@GESTEP(1,2) = 0

@GESTEP(2,1) = 1

@GESTEP(1) = 1

@GESTEP(-2) = 0

You can sum several @GESTEP functions to count the number of values that exceed a certain threshold (Y).

Related topics

@GETGROUP - Name of Group Containing Sheet

Syntax

@GETGROUP(Block, <PageName>)

Block	A cell or cells of the notebook to check.
PageName	A string value representing a sheet name or an address specifying the sheet name to check (optional).

@GETGROUP returns a string that is the name of the group that includes the sheet containing specified cells.

If Block is used in conjunction with the optional argument PageName, @GETGROUP searches the notebook referenced by Block for the group that contains the sheet specified by PageName. PageName is a string or cell address.

If the sheet is not part of a group, @GETGROUP returns ERR.

Example

@GETGROUP([REPORTQ4]A:C12,"April") searches the notebook REPORTQ4 for the name of the group that contains the sheet named April

@GETGROUP([REPORTQ4]A:C12) searches the notebook REPORTQ4 for the name of the group that contains the sheet named A

Related topics

@GETREGISTRYKEY - Return Value of Windows Registry Key

Syntax

@GETREGISTRYKEY(Registry Key, Registry Value)

Registry Key	A string value representing the path in the registry.
Registry Value	A string value representing the stored value in the registry, at the specified path.

@GETREGISTRYKEY returns the value of the specified key in the registry.

You can use @GETREGISTRYKEY to open a file specified in the registry.

Example

```
{FileOpen +@GETREGISTRYKEY("HKEY_LOCAL_MACHINE\SOFTWARE\Corel\QuattroPro\9\Location of Files\
EN","Template Folder")&"\amortize.qpw"}
```

Related topics

@GRANDTOTAL123 - Sum of Subtotals

Syntax

@GRANDTOTAL123 (List)

List	Any combination of cells; separated by valid argument separators.

@GRANDTOTAL123 sums all cells in a designated area that contain @SUBTOTAL123 in their formulas.

Example

In the following, Cells A3 and C3 contain @SUBTOTAL123(A1..A2) and @SUBTOTAL123(C1..C2), respectively. Cell A4 contains the formula @GRANDTOTAL123(A1..C3) and sums only the subtotals in the cells. To omit possible subtotals in Column B, you could also write @SUM(A1..A2,B1..B2).

	A	B	C	D
1	$30		$18	
2	$65		$22	
3	$95		$40	@SUBTOTAL123 in A3 and C3
4	$135			@GRANDTOTAL123 in A4

Related topics

@GROWTH - Fits Exponential Curve to Data

Syntax

@GROWTH(KnownYs, <KnownXs>, <NewXs>, <Const>)

KnownYs	Array of known y-values for the curve y = b*m^x.
KnownXs	Array of known x-values (optional).
NewXs	Array of new x-values for which you want the corresponding y-values (optional).
Const	Logical value (optional) that tells @GROWTH whether to force the constant b = 1: ▪ If Const is TRUE or omitted, @GROWTH uses the actual value of b. ▪ If Const is FALSE, @GROWTH sets b = 1, then adjusts the m-values so that y = m^x.

@GROWTH fits an exponential curve to the data KnownYx and KnownXs, and returns the y-values along that curve for the array of NewXs that you specify.

- If known y-values are in one column, @GROWTH takes each column of known x-values to be a separate variable. If known y-values are in one row, @GROWTH takes each row of known x-values to be a separate variable.
- If any of the known y-values are 0 or negative, @GROWTH returns ERR.
- The argument KnownXs can include more than one set of variables. If you use only one variable, KnownYs and KnownXs can be cell areas of any shape, but must have the same dimensions. If you use more than one variable, KnownYs must be a single-column or single-row. Use commas to separate x-values in the same row and pipes (|) to separate rows.
- The argument NewXs must follow the pattern of KnownXs: It must include a row or column for each independent variable. If you omit the argument NewXs, @GROWTH assumes it is the same as KnownXs. If you omit both KnownXs and NewXs, @GROWTH assumes they are the array {1, 2, 3,...} of a size equal to KnownYs.

Example

Sales for your company in its first four quarters are entered in a selection named Sales:

	A	B
1	Quarter	Sales
2	1	$75,000
3	2	$90,000
4	3	$115,000
5	4	$140,000

To predict second-year sales, @GROWTH(Sales,A2..A5,A6..A9) = {$173,359, $214,246, $264,775, $327,222}

Related topics

@HALFP - Convert Full-Width String

Syntax

@HALFP(String)

String	The double-byte (full-width) character string

@HALFP converts a double-byte character string to a half-width (single-byte) character string in a label. Double-byte characters are used in software localized to most Far Eastern languages (for example, Japanese, Chinese, and Korean). The localized machine will display a toolbar that lets you select various single and double-byte character sets from within Quattro Pro.

@HALFP allows for converting double-byte character set (DBCS) alphanumeric characters to ASCII characters. In cases where there is no matching ASCII character, these characters are not converted and the results displays the same format as the original.

You cannot convert use @HALFP to convert double-byte Katakana characters to single-byte Katakana.

Related topics

@HARMEAN - Harmonic Mean

Syntax

@HARMEAN(List)

List	One or more numeric or cell values; none of the values in List can equal 0.

@HARMEAN returns the harmonic mean of a data set. The harmonic mean is the reciprocal of the arithmetic mean of the reciprocals of a set of numbers.

@HARMEAN uses this formula:

$$\frac{1}{\frac{1}{n}\left(\frac{1}{x_1}+\frac{1}{x_2}+\ldots+\frac{1}{x_n}\right)}$$

Example

@HARMEAN(3,4,5,6,7) = 4.575163

Related topics

@HEXTOASC - Hexadecimal to ASCII

Syntax

@HEXTOASC(Hex)

Hex	Hexadecimal number to convert; can be up to 40 hexadecimal digits.

@HEXTOASC returns the ASCII equivalent of a hexadecimal number.

If the hexadecimal number includes nonnumeric characters, enclose it in quotation marks.

Examples

@HEXTOASC("2B") = +

@HEXTOASC("3031414243444546") = 01ABCDEF

@HEXTOASC("5155415454524F") = QUATTRO

Related topics

@HEXTOBIN - Hexadecimal to Binary

Syntax

@HEXTOBIN(Hex)

Hex	Hexadecimal number to convert.

@HEXTOBIN returns the binary string equivalent of a hexadecimal number. To convert a negative number, precede Hex with a minus sign.

Examples

@HEXTOBIN("A") = 1010

@HEXTOBIN("10") = 10000

@HEXTOBIN("1E") = 11110

Related topics

@HEXTOBIN64 - Hexadecimal to Binary

Syntax

@HEXTOBIN64(Hex, <Places>)

Hex	Hexadecimal number to convert, must be >0.
Places	Number of characters to return; must be ≤ 64.

@HEXTOBIN64 returns the binary string equivalent of a hexadecimal number (up to 64 bits).
If the hexadecimal number includes nonnumeric characters, enclose it in quotation marks.

Examples

@HEXTOBIN64("A",2) = 10
@HEXTOBIN64("A",6) = 001010
@HEXTOBIN64("1E078") = 00011110000001111000
@HEXTOBIN64("1E078",7) = 1111000

Related topics

@HEXTONUM - Hexadecimal to Decimal

Syntax

@HEXTONUM(Hex)

Hex	A hexadecimal number enclosed by double quotes, either positive or negative.

@HEXTONUM converts the hexadecimal number in the string to the corresponding decimal value. @NUMTOHEX performs the opposite conversion, from decimal to hexadecimal.

Examples

@HEXTONUM("a") = 10

@HEXTONUM("10") = 16

@HEXTONUM("00FF") = 255

@HEXTONUM(A1) = 10 (if cell A1 contains the label 'a)

Related topics

@HEXTONUM64 - Hexadecimal to Decimal

Syntax

@HEXTONUM64(Hex, <Signed>)

Hex	Hexadecimal number to convert.
Signed	1 if the most significant bit of Hex is a sign bit; 0 if Hex is positive (the default is 0).

@HEXTONUM64 returns the decimal equivalent of a hexadecimal number (up to 64 bits).

If Signed is 1, the most significant bit of Hex is the sign bit. If the sign bit is 0, the number is positive; if it is 1, the number is negative.

If the hexadecimal number includes nonnumeric characters, enclose it in quotation marks.

Examples

@HEXTONUM64("A") = 10

@HEXTONUM64("123456789ABCDEF0") = 1311768467463790320

@HEXTONUM64("FE4FA1",1) = -110687

Related topics

@HEXTOOCT - Hexadecimal to Octal

Syntax

@HEXTOOCT(Hex)

Hex	Hexadecimal number to convert.

@HEXTOOCT returns the octal string equivalent of a hexadecimal number. To convert a negative number, precede Hex with a minus sign.

Hexadecimal strings must be <14 characters.

Examples

@HEXTOOCT("A") = 12

@HEXTOOCT("10") = 20

@HEXTOOCT("1E") = 36

Related topics

@HEXTOOCT64 - Hexadecimal to Octal

Syntax

@HEXTOOCT64(Hex, <Places>)

Hex	Hexadecimal number to convert, must be > 0.
Places	Number of characters to return; must be ≤ 22.

@HEXTOOCT64 returns the octal string equivalent of a hexadecimal number (up to 64 bits).
If the hexadecimal number includes nonnumeric characters, enclose it in quotation marks.

Examples

@HEXTOOCT64("A") = 12
@HEXTOOCT64("7",2) = 07
@HEXTOOCT64("1E078",6) = 360170
@HEXTOOCT64("0123456789ABCDEF") = 0004432126361152746757

Related topics

@HLOOKUP - Horizontal Lookup

Syntax

@HLOOKUP (X, Block, Row, <Type>)

X	The numeric or string value you want to search for.
Block	The range of cells.
Row	The number of the referenced row. The rows are referenced from 0 to the number of rows in Block minus 1.
	The first row (index row) in Block = 0
	The second row in Block = 1, ...
Type <optional>	Lets you specify whether or not the match must be exact.
	0 Does not need to be an exact match
	1 Must be an exact match

@HLOOKUP searches horizontally through the index row of Block for the value X. When @HLOOKUP finds the value X, it returns the value displayed Row rows beneath it.

All values in the index column must be sorted in ascending order for the function to work correctly. Otherwise, ERR or an incorrect answer may be returned.

@HLOOKUP returns 0 if the referenced cell is blank. ERR is returned if:

- Row is less than 0 or greater than the number of rows minus 1 in Block.
- X is less than the smallest value in the topmost row of Block.
- X and the first row entries are string values and @HLOOKUP fails to find a match in the top row of Block.
- X is a string or label and the index row entries are numeric values.

If X is a string, @HLOOKUP looks for an exact case-sensitive match. If X is a number and @HLOOKUP cannot find an equal number, it locates the highest number, not more than X, in the row.

If X is a number and the index row contains only labels, @HLOOKUP stops at the rightmost column.

Each cell of the index row must contain a value.

Examples

In the following example, @HLOOKUP searches across the index row (1) of the Block (A1..D4), looking for the largest number equal to or less than X (17). It stops at cell D1, then moves down Row rows (3). It stops at cell D4 and returns the value 47.

	A	B	C	D
1	1	5	10	15
2	43	3	32	67
3	92	42	18	22
4	45	83	76	47

@HLOOKUP(17, A1..D4, 3)

Returns: 47

Related topics

@HOLS - Holidays

Syntax

@HOLS(StartDate, EndDate, Holidays, <Saturday>, <Sunday>)

StartDate	Number representing the start date. See "Using dates and times in Quattro Pro."
EndDate	Number representing the end date.
Holidays	Cells containing dates that are holidays; to indicate no holidays, enter an empty cell or cells.
Saturday	0 to specify that Saturday is not a business day; 1 to specify that Saturday is a business day (the default is 0).
Sunday	0 to specify that Sunday is not a business day; 1 to specify that Sunday is a business day (the default is 0).

@HOLS returns the number of holidays between StartDate and EndDate, including the specified dates (if they appear in Holidays).

By default, @HOLS does not include holidays that fall on a Saturday or Sunday; if either Saturday or Sunday is passed as 1, the count also includes holidays falling on that day.

Example

This formula calculates the number of holidays between April 1, 1993 and December 14, 1993, assuming that the dates contained in selection A7..C9 are holidays.

@HOLS(@DATE(93,4,1),@DATE(93,12,14),A7..C9) = 5

Related topics

@HOUR - Hour Portion of Date Serial Number

Syntax

@HOUR(DateTimeNumber)

DateTimeNumber	A numeric value between -109571 and 474816.9999999, representing a date/time serial number: -109571 = January 1, 1600; 0 = December 31, 1899; 474816 = December 31, 3199; the decimal = time (24 hr).

See "Using dates and times in Quattro Pro."

@HOUR returns the hour portion of DateTimeNumber. DateTimeNumber must be a valid date/time serial number. Because only the decimal portion of a serial number pertains to time, the integer portion of the number is disregarded. The result is between 0 (12:00AM) and 23 (11:00PM).

To extract the hour portion of a string that is in time format (instead of serial format), use @TIMEVALUE with @HOUR to translate the time into a serial number. To return standard hours (1-12) instead of military hours (1-24), use @MOD with a parameter of 12.

Examples

@HOUR(.25) = 6

@HOUR(.5) = 12

@HOUR(.75) = 18

@HOUR(@TIMEVALUE("10:08am")) = 10

@MOD(@HOUR(@TIMEVALUE("9:31:52 PM")),12) = 9

Related topics

@HYPGEOMDIST - Hypergeometric Distribution

Syntax

@HYPGEOMDIST(SampleSuccess, SampleSize, PopSuccess, PopSize)

SampleSuccess	Successes in the sample; must be ≥ 0.
SampleSize	Sample size; must be ≥ 0 and ≤ PopSize.
PopSuccess	Successes in the population; must be ≥ 0 and ≤ PopSize.
PopSize	Population size; must be ≥ 0.

@HYPGEOMDIST returns the hypergeometric distribution of a sample. It gives the probability of successes in a sample given the sample's size, the total population, and the number of successful trials in that population. Use @HYPGEOMDIST to determine the probability that a distribution contains exactly SampleSuccess items of a particular type.

SampleSuccess must be greater than or equal to 0, greater than the lesser of SampleSize or PopSuccess, and greater than the larger of 0 or (SampleSize-PopSize+PopSuccess)

@HYPGEOMDIST uses this formula:

$$P(X = d) = h(d; n, D, N) = \frac{\binom{D}{d}\binom{N - D}{n - d}}{\binom{N}{n}}$$

where

d	SampleSuccess
n	SampleSize
D	PopSuccess
N	PopSize

Examples

Five cards are drawn from a deck of 52 playing cards. This formula calculates the probability that one of the five cards drawn is an ace (assuming there are only four aces in the deck):

@HYPGEOMDIST(1,5,4,52) = 0.299474

Related topics

@IF - Perform Logical Test

Syntax

@IF(Cond, TrueExpr, FalseExpr)

Cond	A logical expression representing the condition to be tested.
TrueExpr	A numeric or string value representing the value to use if Cond is true.
FalseExpr	A numeric or string value representing the value to use if Cond is false.

@IF evaluates the logical condition specified as Cond. If the condition is found to be true, it returns the value specified as TrueExpr. If the condition is false, it returns the value specified as FalseExpr. Cond is true if it evaluates to any nonzero numeric value.

The formula entered as Cond can be any logical expression that can be evaluated as true or false; for example, B6<0 or C3*D2=53.

You can use compound conditions by connecting expressions with #AND# or #OR#. If you use #AND#, both conditions specified must be met to evaluate true. If you use #OR#, the expression is true if either of the conditions is met. For example, A3<10#OR#A3>5 means that the value in A3 must be *either* less than 10 *or* greater than 5 to evaluate true--6,9,1, and 15 are all true; A3<10#AND#A3>5 means that the value in A3 must be *between* 5 and 10 to evaluate true. If A3 contains a label, the expression evaluates true because labels have a value of 0 (zero).

You can also use the #NOT# operator to negate a condition. For example, #NOT#(B3>10) evaluates true if B3 is not greater than 10.

TrueExpr and FalseExpr can be numbers, formulas resulting in numbers, or text. If text, the string must be enclosed by double quotes; for example, @IF(D6=5,"John","Harry "). You can also use cell references to use the contents of other cells in the notebook. For example, @IF(B10<18,D5,C4) enters the contents of D5 if the condition is true, and enters the contents of C4 if the condition is false.

If the condition you specify with Cond searches a cell for a number and the cell contains a label, the label is evaluated as having a value of 0 and FalseExpr is returned. Likewise, if you search for a label and find a numeric value, TrueExpr results if the value of the referenced cell is 0; FalseExpr results if it is nonzero.

Although logical expressions typically reference other cells, this is not required. Any expression resulting in a numeric value is accepted; for example, A1=1 or A1="Fred". If the result of Cond is nonzero, TrueExpr is the result; otherwise, FalseExpr is the result.

@IF statements can be nested, or used within one another. In other words, TrueExpr can contain yet another test to further validate Cond.

For example, @IF(B5>C6,@IF(B5>C7,1,2),3) tells Quattro Pro to see if the contents of B5 are greater than C6. If they are, it then checks to see if B5 is greater than C7; if so, it enters a 1 in the cell. If not, it enters a 2. If B5 is not greater than C6, it enters a 3. There is no limit on the number of levels @IF expressions that you can nest, as long as the entire expression does not exceed 1024 characters.

Examples

@IF(8=7,4,5) = 5

@IF(B4<100,"Yes","No") = Yes if B4 < 100; otherwise, No

@IF(C10=BLOCK,45,50) = 45 if C10 = the cell named BLOCK; otherwise, 50

@IF(C10,1,0) = 0 if C10 = 0; otherwise, 1

Related topics

@IMABS - Absolute Value of Complex Number

Syntax

@IMABS(Complex)

Complex	Complex number in the format *x* + *y*i, *x* + i*y*, *x* + *y*j, or *x* + j*y* for which you want the absolute (modulus) value.

@IMABS returns the absolute value (modulus) of a complex number. It uses this formula:

Given Complex = *x* + *y*i

$$|C| = \sqrt{x^2 + y^2}$$

C = Complex

Example

@IMABS("-10+25.6i") = 27.48381

Related topics

@IMAGINARY - Imaginary Coefficient of Complex Number

Syntax

@IMAGINARY(Complex)

Complex	Complex number in the format *x* + *y*i, *x* + i*y*, *x* + *y*j, or *x* + j*y* from which you want to extract the imaginary coefficient.

@IMAGINARY returns the imaginary coefficient of a complex number.

Examples

@IMAGINARY("2+8i") = 8

@IMAGINARY("-i") = -1

Related topics

@IMARGUMENT - Angle of Complex Number

Syntax

@IMARGUMENT(Complex)

Complex	Complex number in the format *x* + *y*i, *x* + i*y*, *x* + *y*j, or *x* + j*y* for which you want to calculate the angle in the complex plane.

@IMARGUMENT returns the angle Θ, in radians, of a number in the complex plane. It uses this formula:

$$x + yi = |x + yi|e^{i\Theta} = |x + yi|(\cos\Theta + i\sin\Theta)$$

Example

@IMARGUMENT("5+12i") = 1.176005

Related topics

@IMCONJUGATE - Complex Conjugate of Complex Number

Syntax

@IMCONJUGATE(Complex)

Complex	Complex number in the format *x* + *y*i, *x* + i*y*, *x* + *y*j, or *x* + j*y* for which you want to calculate the complex conjugate.

@IMCONJUGATE returns the complex conjugate of a complex number.

Example

@IMCONJUGATE("5+12i") = "5-12i"

Related topics

@IMCOS - Cosine of Complex Number

Syntax

@IMCOS(Complex)

Complex	Complex number in the format *x* + *y*i, *x* + i*y*, *x* + *y*j, or *x* + j*y* for which you want to calculate the cosine.

@IMCOS returns the cosine of the complex number Complex. @IMCOS uses this formula:

Given Complex = *x* + *y*i

$$@IMCOS(C) = \frac{e^{iC} + e^{-iC}}{2}$$

C = Complex

Example

@IMCOS("5+12i") = "23083.7+78034.8i"

Related topics

@IMDIV - Quotient of Complex Numbers

Syntax

@IMDIV(Complex1, Complex2)

Complex1	Complex numerator or dividend in the format *x* + *y*i, *x* + i*y*, *x* + *y*j, or *x* + j*y*.
Complex2	Complex denominator or divisor in the format *x* + *y*i, *x* + i*y*, *x* + *y*j, or *x* + j*y*.

@IMDIV returns the quotient of two complex numbers (Complex1 and Complex2). It uses this formula:

Given: Complex1 = a + bi and Complex2 = c + di

$$\frac{C1}{C2} = \frac{a + bi}{c + di} = \frac{(ac + bd) + (bc - ad)i}{c^2 + d^2}$$

C = Complex

Example

@IMDIV("5+6i","3+4i") = "1.56-0.08i"

Related topics

@IMEXP - Exponential of Complex Number

Syntax

@IMEXP(Complex)

Complex	Complex number in the format *x* + *y*i, *x* + i*y*, *x* + *y*j, or *x* + j*y* for which you want to calculate the exponential.

@IMEXP returns the exponential of a complex number. It uses this formula:

Given Complex = x + yi,

$$e^{C} = e^{x}(\cos y + i * \sin y)$$

C = Complex

Example

@IMEXP("5+12i") = "125.239-79.6345i"

Related topics

@IMLN - Natural Logarithm of Complex Number

Syntax

@IMLN(Complex)

Complex	Complex number in the format *x* + *y*i, *x* + i*y*, *x* + *y*j, or *x* + j*y* for which you want to calculate the natural logarithm.

@IMLN returns the natural logarithm of the complex number Complex. It uses this formula:

$$\ln(x + yi) = \ln\sqrt{x^2 + y^2} + i*\tan^{-1}\left(\frac{y}{x}\right)$$

Example

@IMLN("5+12i") = "2.56495+1.17601i"

Related topics

@IMLOG10 - Base 10 Logarithm of Complex Number

Syntax

@IMLOG10(Complex)

Complex	Complex number in the format *x* + *y*i, *x* + i*y*, *x* + *y*j, or *x* + j*y* for which you want to calculate the base 10 log.

@IMLOG10 returns the base 10 (common) logarithm of the complex number Complex. It uses this formula:

Given Complex = x + yi

$$\log\ 10(C) = \frac{\log(C)}{\log(10)}$$

C = Complex

Example

@IMLOG10("5+12i") = "1.11394+0.510733i"

Related topics

@IMLOG2 - Base 2 Logarithm of Complex Number

Syntax

@IMLOG2(Complex)

Complex	Complex number in the format *x* + *y*i, *x* + i*y*, *x* + *y*j, or *x* + j*y* for which you want to calculate the base 2 log.

@IMLOG2 returns the base 2 logarithm of the complex number Complex. It uses this formula:

Given Complex = x + yi

$$\log\ 2(C) = \frac{\log(C)}{\log(2)}$$

C = Complex

Example

@IMLOG2("5+12i") = "3.70044+1.69662i"

Related topics

@IMPOWER - Complex Number Raised to a Power

Syntax

@IMPOWER(Complex, Power)

Complex	Complex number in the format *x* + *y*i, *x* + i*y*, *x* + *y*j, or *x* + j*y*.
Power	The power to which you want to raise Complex; can be a complex number in the format *x* + *y*i, *x* + i*y*, *x* + *y*j, or *x* + j*y*.

@IMPOWER returns the complex number Complex raised to the power Power. Power can be a value or a complex number.

Examples

@IMPOWER("5+12i",3) = "-2035-828i"

@IMPOWER("5+12i","3+2i") = "-150.575+145.094i"

Related topics

@IMPRODUCT - Product of Complex Numbers

Syntax

@IMPRODUCT(Complex1, Complex2)

Complex1	Complex number in the format *x* + *y*i, *x* + i*y*, *x* + *y*j, or *x* + j*y*.
Complex2	Complex number in the format *x* + *y*i, *x* + i*y*, *x* + *y*j, or *x* + j*y*.

@IMPRODUCT returns the product of two complex numbers (Complex1 and Complex2). It uses this formula:

Given Complex1 = a + bi and Complex2 = c + di

(Complex1)(Complex2) = (ac - bd) + (ad + bc)i

Example

@IMPRODUCT("5+12i","2-i") = "22+19i"

@IMPRODUCT("10+2i",5) = "50+10i"

Related topics

@IMREAL - Real Coefficient of a Complex Number

Syntax

@IMREAL(Complex)

Complex	Complex number in the format *x* + *y*i, *x* + i*y*, *x* + *y*j, or *x* + j*y* from which you want to extract the real coefficients.

@IMREAL returns the real coefficient of a complex number.

Examples

@IMREAL("2+8i") = 2

@IMREAL("-i") = 0

Related topics

@IMSIN - Sine of a Complex Number

Syntax

@IMSIN(Complex)

Complex	Complex number in the format *x* + *y*i, *x* + i*y*, *x* + *y*j, or *x* + j*y* for which you want to calculate the sine.

@IMSIN returns the sine of the complex number Complex. It uses this formula:

Given Complex = *x* + *y*i

$$@IMSIN(C) = \frac{e^{iC} - e^{-iC}}{2i}$$

C = Complex

Examples

@IMSIN("5+12i") = "-78034.8+23083.7i"

@IMSIN("1+i") = "1.29846+0.634964i"

Related topics

@IMSQRT - Square Root of a Complex Number

Syntax

@IMSQRT(Complex)

Complex	Complex number in the format *x* + *y*i, *x* + i*y*, *x* + *y*j, or *x* + j*y* to calculate square root of.

@IMSQRT returns the square root of a complex number. It uses this formula:

Given Complex = x + yi

$$\sqrt{C} = \sqrt{|C|}\left(\cos\left(\frac{\arg(C)}{2}\right) + \sin\left(\frac{\arg(C)}{2}\right)\right)$$

C = Complex

Example

@IMSQRT("5+12i") = "3+2i"

Related topics

@IMSUB - Difference of Complex Numbers

Syntax

@IMSUB(Complex1, Complex2)

Complex1	Complex number in the format *x* + *y*i, *x* + i*y*, *x* + *y*j, or *x* + j*y* from which to subtract Complex2.
Complex2	Complex number in the format *x* + *y*i, *x* + i*y*, *x* + *y*j, or *x* + j*y* to subtract from Complex1.

@IMSUB returns the difference of two complex numbers (Complex1 and Complex2). It uses this formula:

Given Complex1 = (a + bi) and Complex2 = (c + di)

(a + bi) - (c + di) = (a - c) + (b - d)i

Example

@IMSUB("5+12i","2-i") = "3+13i"

Related topics

@IMSUM - Sum of Complex Numbers

Syntax

@IMSUM(List)

List	One or more complex numbers in the format *x* + *y*i, *x* + i*y*, *x* + *y*j, or *x* + j*y*, separated by commas.

@IMSUM returns the sum of a list of complex numbers. It uses this formula:

Given Complex1 = (a + bi) and Complex2 = (c + di)

Complex1 + Complex2 = (a + c) + (b + d)i

Example

@IMSUM("5+12i","7+14i") = "12+26i"

Related topics

@INDEX - Return Value from Table Index

Syntax

@INDEX(Block, Column, Row, <Page>)

Block	A cell reference or name.
Column	The number of the referenced column, from 0 to 255 (the first column in Block = 0, the second = 1, and so on).
Row	The number of the referenced row; if an offset, the first row in Block = 0, the second = 1, and so on.
Page	The number of the referenced sheet, from 0 to 255 (the first sheet in Block = 0, the second = 1, and so on).

@INDEX searches through the table specified as Block and returns the value specified with the Column, Row, and optional Page values. The upper left cell in Block is column 0, row 0. Likewise, the first sheet is 0. The Column and Row values are not the actual coordinates of the resulting cell, but instead are offset values. In other words, @INDEX begins in the top left cell of the specified cells, moves right the number of columns specified by Column, moves down the number of rows specified by Row, and through the number of sheets specified by Page (if you have specified a Page). It then returns the value in the current cell.

The Column, Row, and Page values must be numbers equal to or greater than zero and less than the number of rows, columns, or sheets in the cells. If a fractional number is used (for example, 2.35), the fractional part is dropped (not rounded).

@HLOOKUP and @VLOOKUP are related functions.

Examples

	A	B	C	D
1	1	5	10	15
2	43	53	32	67
3	92	42	18	22
4	45	83	76	47

These examples reference cells in the data table:

@INDEX(A1..D4,3,2) = 22

@INDEX(A1..D4,1,2) = 42

@INDEX(C2..D3,0,1) = 18

@INDEX(C2..D3,1,3) = ERR (too many rows)

@INDEX(A1..D4,-2,3) = ERR (negative column number)

Related topics

@INDEXTOLETTER - Letter(s) Corresponding to Sheet/Column Index

Syntax

@INDEXTOLETTER(Index)

Index	An integer number from 0 to 18277 inclusive.

@INDEXTOLETTER returns a one-, two-, or three-character string equivalent (for example, A, B, AA, AB, and ZZZ) for the index number of a sheet or column.

If Index is < 0 or > 18277, @INDEXTOLETTER returns ERR. If Index is not an integer, it is rounded to the nearest integer.

Examples

@INDEXTOLETTER(0) = A

@INDEXTOLETTER(1) = B

@INDEXTOLETTER(18277) = ZZZ

Related topics

@INT - Integer

Syntax

@INT(X)

X	A numeric value.

@INT drops the fractional portion of X, returning its integer value. @ROUND rounds X to the nearest integer.

Examples

@INT(499.99) = 499
@INT(0.1245) = 0
@INT(-2.3) = -2
@INT(C4) = 5 if C4 contains a value between 5 and 6

Related topics

@INTERCEPT - Y-Intercept

Syntax

@INTERCEPT(KnownY, KnownX)

KnownY	Dependent range of values.
KnownX	Independent range of values.

@INTERCEPT returns the y-intercept of the linear regression line through two data sets. KnownY and KnownX must contain the same number of values. The formula @INTERCEPT uses is

$$q = \overline{Y} - a_1 \overline{X}$$

a_i, the slope, is calculated using this formula:

$$a_1 = \frac{n \sum xy - \left(\sum x\right)\left(\sum y\right)}{n \sum x^2 - \left(\sum x\right)^2}$$

Example

@INTERCEPT({16,28,30,35,52,65},{11,15,18,22,35,43}) = 3.56304

Related topics

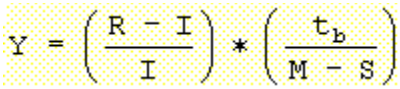
@INTRATE - Annualized Interest Rate

Syntax

@INTRATE(Settle, Maturity, Investment, Redemption, <Calendar>)

Settle	Number representing the settlement date; must be < Maturity.
Maturity	Number representing the maturity date; must be > Settle.
Investment	Amount invested; must be > 0.
Redemption	Redemption value; must be > 0.
Calendar	Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

@INTRATE returns the simple annualized yield for a fully invested security. @INTRATE computes yield using this formula:



Y	yield
R	redemption
I	investment
b	basis
M	maturity
S	settle

tb is the number of days over which the discount rate applies (360 or 365).

Example

This formula calculates the interest rate for a bond with the following terms: Settle is November 11, 1995, Maturity is May 27, 1996, Investment is $10,000, Redemption is $10,397.50, and Calendar is 1 (actual/actual).

@INTRATE(@DATE(95,11,11),@DATE(96,5,27),10000,10397.50,1) = 0.073277

Related topics

@INVB - Binary Bit Inversion

Syntax

@INVB(Binary, <Bits>)

Binary	Binary number.
Bits	Number of binary bits used for both input and output; if omitted, Bits = number of bits in Binary; must be in the range 0 <n ≤ 64.

@INVB inverts the bits of a binary number. All bits that are 1 change to 0, and all bits that are 0 change to 1.

Examples

@INVB(0) = 1

@INVB(1010,5) = 10101

@INVB(1100,5) = 10011

Related topics

@INVH - Hexadecimal Bit Inversion

Syntax

@INVH(Hex, <Bits>)

Hex	Hexadecimal number.
Bits	Number of binary bits used for both input and output; if omitted, Bits = number of bits in Hex; 4 binary digits = 1 hexadecimal digit; must be in the range 0 <n ≤ 64.

@INVH inverts the binary bits of a hexadecimal number. All bits that are 1 change to 0, and all bits that are 0 change to 1.

Example

@INVH("A") = 5

@INVH("C",8) = F3

Related topics

@INTXL - Integer

Syntax

@INTXL(X)

X	A numeric value.

@INTXL rounds X down to an integer value. @ROUND rounds X to the nearest integer. @INT drops the fractional portion of X, returning its integer value.

Examples

@INTXL(499.99) = 499

@INTXL(0.1245) = 0

@INTXL(-2.3) = -3

@INTXL(D4) = -6 if D4 contains a value between -5 and -6

@INTXL(C4) = 5 if C4 contains a value between 5 and 6

Related topics

@IPAYMT - Interest Portion of Payment

Syntax

@IPAYMT(Rate, Per, Nper, Pv, <Fv>, <Type>)

Rate	A numeric value > -1, representing the periodic interest rate (the fixed interest rate per compounding period).
Per	The number of the loan period for which the interest is desired (where Nper is the total number of periods).
Nper	A numeric value > 0, representing the number of periods of the loan (the number of payments to be made) or investment (the number of compounding periods).
Pv	A numeric value representing the amount borrowed (the principal).
Fv	A numeric value representing the future value of an investment (the value the investment will reach at some point).
Type	An optional numeric value that indicates whether payments or cash flows occur at the beginning (1) or the end (0) of the period; default = 0.

@IPAYMT and @PPAYMT tell what portion of a particular loan payment is interest and what portion is principal, respectively. For each month in the transaction period,

@PAYMT(Rate, Nper, Pv, Fv, Type) = @IPAYMT(Rate, Per, Nper, Pv, Fv, Type) + @PPAYMT(Rate, Per, Nper, Pv, Fv, Type).

@IPAYMT is calculated by computing the simple interest on the outstanding principal from the previous month. @PPAYMT then gives the principal portion of the payment for the current month, and is computed by subtracting @IPAYMT from @PAYMT. The calculation starts by using Pv as the outstanding principal at the beginning of the first month:

@IPAYMT(Rate, Per, Nper, Pv, Fv, Type) = Rate ∗ @FVAL(Rate, Per + (Type - 1), @PAYMT(Rate, Nper, PV, Fv, Type), Pv, Type)

The last two arguments, Fv and Type, are optional. If you omit one or both of them, their values are assumed to be zero.

Examples

If you are two years into a 30-year, 10% mortgage on a $100,000 loan and your interest payment is tax-deductible, then @IPAYMT(.1/12,2*12,30*12,100000) returns your current month's deduction: -824.03.

Related topics

@IRATE - Interest Rate

Syntax

@IRATE(Nper, Pmt, Pv, <Fv>, <Type>)

Nper	A numeric value > 0, representing the number of periods of the loan (the number of payments to be made) or investment (the number of compounding periods).
Pmt	A numeric value representing the amount of the periodic payment.
Pv	A numeric value representing the current value of an investment (the present value).
Fv	A numeric value representing the future value of an investment (the value the investment will reach at some point).
Type	An optional numeric value that indicates whether payments or cash flows occur at the beginning (1) or the end (0) of the period; default = 0.

@IRATE calculates the interest rate required to pay off a principal (Pv) or reach an investment goal (Fv) in Nper payments of Pmt amount.

The last two arguments, Fv and Type, are optional. If you omit one or both of them, their values are assumed to be zero.

@IRATE requires that the initial cash flow (Pv + Type * Pmt) and the last cash flow (Fv + (1-Type) * Pmt) have opposite signs. Otherwise, @IRATE returns ERR because the transaction is not simple and there may not be a meaningful rate.

@IRATE is not compatible with 1-2-3. If your file must be compatible with 1-2-3, use @RATE instead.

Be sure to enter a negative number for money that is out of your pocket and a positive number for money that's coming in to you.

Examples

Assume you are negotiating to buy a $15,000 new car. The salesperson says you can have the car for $500 a month for the next five years. To calculate the monthly percentage rate:

@IRATE(5*12,-500,15000,0,0) = 0.02632

Another example: Assume that you plan to deposit $2000 a year into a savings account that currently contains only $2.38. What interest rate must the account earn to generate $15,000 at the end of 5 years? Use this formula:

@IRATE(5,-2000,-2.38,15000,0) = 0.203773

Related topics

@IRR - Internal Rate of Return

Syntax

@IRR(Guess, Block)

Guess	A numeric value that estimates the internal rate of return on an investment.
Block	Cells (reference or name) containing cash flow information for the investment.

@IRR determines the internal rate of return on an investment. It references cells in your notebook that contains cash flow information and uses the supplied internal rate of return estimate to calculate the results.

Before using @IRR, you must set up a cash flow table, showing expected cash flow amounts over a period of time. Quattro Pro assumes that the amounts are received at regular intervals. Negative amounts are interpreted as cash outflows, and positive amounts as inflows. The first amount must be a negative number, to reflect the initial investment. These amounts can all be the same for each time period, or they can be different (including a mixture of negatives, positives, or zeros).

@IRR always returns ERR or a rate of return greater than or equal to -1. Some cash flows have no rate of return, and some have several. If it can be determined that the cash flow has a unique rate of return, then the Guess is ignored and @IRR gives that unique value.

Quattro Pro can determine unique rates of return for simple transactions or cash flow. A simple cash flow has two sets of values: a series of nonpositive values (negative values and zero) that is your cash outflow, and a series of nonnegative values (positive values and zero) that is your cash inflow. A simple cash flow must contain both a negative value (cash outflow) and a positive value (cash inflow).

Simple values have a unique rate of return, so you can safely use @NA as the Guess argument in most cases. Quattro Pro then tries to determine whether the root is unique and returns that rate without using a Guess argument. If Quattro Pro cannot find a unique root value, @IRR returns ERR.

Typically, you make an investment (a negative cash flow) and then receive several dividends (positive cash flows). This is an example of a simple transaction, and @IRR gives the unique rate of return for this without requiring a Guess. More complex transactions, in which the direction of money changes several times, often do not have a meaningful value for @IRR. For more information, see @IRR with Multiple Rates of Return.

@IRR(Guess,Block) gives the number Rate which satisfies @NPV(Rate,Block,0) = 0. For a simple transaction, @NPV(@IRR(Block),Block,0) will give a number close to 0 (it may not be exactly 0 due to how numbers are rounded off).

Guess can be any value greater than -1. Values that are NA or less than or equal to -1 are ignored. Use @NA for Guess unless your cash flow has multiple rates of return. If you use @NA, you will get ERR if your cash flow has more than one rate of return, rather than the rate of return that happens to be near your Guess.

There is no Type parameter to @IRR because the rate of return is the same regardless of whether the payments are at the end or the beginning of each period.

Examples

	A	B	C
1	3000	-50000	-10000
2	700	-8000	1000
3	600	2000	1000
4	750	4000	1200
5	900	6000	2000

@IRR(0,A1..A5) = -1

@IRR(0,B1..B5) = -38.09%

@IRR(0,C1..C5) = -19.90%

Related topics

@IRR with Multiple Rates of Return

In unusual cases, @IRR may have as many as N-1 roots, where N is the number of terms in the cells. Consider the selection that has the values (-10, +150, -145). @IRR(@NA,Block) returns ERR because it is not simple. The two roots are 3.86% and 1296%, obtainable from guesses of 0 and 10, respectively. Both of these values are meaningful, if interpreted properly.

Maybe you invested $10 in an oil well. It struck oil, paying you $150, but then it ran into legal difficulties and required you to pay back $145. You had a net loss of $5, but your rate of return is quite large, as you had the use of a relatively large amount of money for a small investment. Or, maybe the real purpose of the transaction was to get a $150 loan from the bank. The bank required you to pay a $10 application fee ahead of time. After you got the loan, you paid back $145. Because you only paid back $155 on a $150 loan, the interest rate is fairly low. The difference in these two interpretations is that in one you're the lender, and in the other you're the borrower.

If you find a transaction with two roots, there is a mechanical way to determine which is the lender rate and which is the borrower rate. Pick a positive term in the Block, and increase it by a small amount. If the rate increases, it is a lender rate, and if the rate decreases, it is a borrower rate.

If Block has the values (-10,+150,-145), @IRR(0,Block) = 3.86%

If Block has the values (-10,+150,-145), @IRR(10,Block) = 1296%

If Block has the values (-10,+150.1,-145), @IRR(0,Block) = 3.78%

If Block has the values (-10,+150.1,-145), @IRR(10,Block) = 1297%

Because 3.78% < 3.86% and 1297 > 1296%, it follows from this rule that 3.86% is a borrower rate and 1296% is a lender rate.

Most uses of @IRR are for analyzing an investment in which the first cash flow is negative, and the rate is a lender rate.

Some transactions have no rate of return at all. @IRR(Guess,Block), with Block having the values (-1,+1,-1), returns ERR regardless of the Guess. There is no rate of return that is meaningful for this cash flow.

If there are more than two roots, the above analysis can still be used to determine if a particular root is a lender rate or a borrower rate. In some cases, it might still be possible to assign meaning to a root, but it is much more likely that the transaction should be interpreted as several transactions, with a rate of return for each. For example, the cash flow (-1,+6,-11,+6) has three roots, 0%, 100%, 200%. It is difficult to interpret such a transaction in terms of interest rates, and the roots are sensitive to small fluctuations.

Related topics

@ISBDAY - Business Day Test

Syntax

@ISBDAY(Date, <Holidays>, <Saturday>, <Sunday>)

Date	Number representing a date. See "Using dates and times in Quattro Pro."
Holidays	Cells containing dates that are holidays or the date of a single holiday or 0 to indicate no holidays (the default is 0).
Saturday	0 to specify that Saturday is not a business day; 1 to specify that Saturday is a business day (the default is 0).
Sunday	0 to specify that Sunday is not a business day; 1 to specify that Sunday is a business day (the default is 0).

@ISBDAY tests whether Date is a business day. To qualify as a business day, Date cannot fall on a Saturday or Sunday (unless Saturday and Sunday are designated as business days by Saturday and Sunday), and cannot appear in the cells specified by Holidays. If Date is a business day, @ISBDAY returns 1; otherwise, @ISBDAY returns 0.

Example

Given the cells of holidays, A7..C9, and treating Saturday as a business day (except when it is a date included in A7..C9), this formula tests whether May 31, 1993 is a business day:

@ISBDAY(@DATE(93,5,31),A7..C9,1) = 0, since May 31, 1993 is Memorial Day.

Related topics

@ISBLANK - Blank Test

Syntax

@ISBLANK(Location)

Location	Name or address of a cell.

@ISBLANK tests a specified cell to see if it is empty.
If the cell is empty, @ISBLANK returns 1 (true).
If the cell is not empty, @ISBLANK returns 0 (false).

Examples

	A	B
1		Profits
2	$90	

@ISBLANK(A1) = 1
@ISBLANK(B1) = 0
@ISBLANK(A2) = 0
@ISBLANK(B2) = 1
@ISBLANK(A1..B2) = returns the array {1|0|0|1}

Related topics

@ISBLOCK - Block Test

Syntax

@ISBLOCK(Block)

Block	Cell address or presumed cell name.

@ISBLOCK tests input to see if it is a defined cell name or valid address. Valid addresses have sheet name A to IV, column letters A to IV, and row numbers 1 to 8192. @ISBLOCK searches only files in memory.

- If Block is a defined cell name or valid address, @ISBLOCK returns 1 (true).
- If Block is not a defined cell name or valid address, @ISBLOCK returns 0 (false).

You can use @ISBLOCK with @IF to find out if an entry is a valid cell name for subroutine calls and branching with {DISPATCH}.

Examples

@ISBLOCK(C3) = 1

@ISBLOCK(3) = 0

@ISBLOCK(C3..C5) = 1

@ISBLOCK(PROFITS) = 1, if PROFITS is the name of a selection

@ISBLOCK("PROFITS") = 0, because the cell name is enclosed in quotation marks; arguments to @IS functions are not converted from text

Related topics

@ISERR - Error Test

Syntax

@ISERR(X)

X	A cell address or expression.

@ISERR is normally used to check the contents of a cell for errors. If the cell contains ERR, 1 is returned; otherwise, 0 is returned. You can also use formulas or numeric values with @ISERR.

Examples

@ISERR(C2)=1 if C2 contains ERR; otherwise, 0

@ISERR(10/0)=1

@ISERR(45+C3)=1 if C3 is ERR; otherwise, 0

@ISERR(C2/B3)=1 if B3 is 0 or ERR, or if C2 is ERR; otherwise, 0

@IF(@ISERR(A2),0,A5)=0 if A2 is ERR; otherwise, it returns the value in A5

Related topics

@ISEVEN - Even Number Test

Syntax

@ISEVEN(Number)

Number	Value to test.

@ISEVEN returns 1 (true) if a specified number is even, 0 (false) if it is odd.

- If Number is not an integer, it is truncated.
- If Number is non-numeric, @ISEVEN returns ERR.

Examples

@ISEVEN(4) = 1
@ISEVEN(4.9) = 1
@ISEVEN(5) = 0
@ISEVEN(-5) = 0

Related topics

@ISLEGALPAGENAME - Legal Sheet Name Test

Syntax

@ISLEGALPAGENAME(PageName)

PageName	A string value.

@ISLEGALPAGENAME returns 1 if PageName is a valid sheet name (even if the sheet name does not currently exist). Otherwise, it returns 0.

Examples

@ISLEGALPAGENAME("A") = 1

@ISLEGALPAGENAME("1st Qtr") = 1

@ISLEGALPAGENAME("1st Qtr: Net Profit") = 0 (name contains a colon, an invalid character)

Files that contain custom spreadsheet names may not open in Quattro Pro if characters used in the names are not recognized by Quattro Pro. In this event, Quattro Pro displays a message warning you about the invalid spreadsheet name. You can change the spreadsheet name to make it compatible with Quattro Pro. Valid Quattro Pro characters include the following: ~ ` ! % _ | \ ' ?.

Spreadsheet names must not exceed 63 characters.

Related topics

@ISLOGICAL - Logical Value Test

Syntax

@ISLOGICAL(Value)

Value	Empty cell, logical value, text, number, ERR, cell reference, or cell name to test.

@ISLOGICAL tests if the value is a logical value (0,1, TRUE, FALSE).; it returns 0 (false) if its argument refers to any other number.

Examples

@ISLOGICAL(A1) = 1 if Cell A1 evaluates to either 0 or 1

@ISLOGICAL(5) = 0

Related topics

@ISNA - NA Test

Syntax

@ISNA(X)

X	A cell address or expression.

@ISNA tests for the special value NA in a cell. If the cell contains an NA value, it returns 1; otherwise, it returns 0. NA is considered a special value; it appears in the notebook only through the use of @NA. Cells containing the label "NA" typed directly (not produced by @NA) are not recognized by @ISNA.

Examples

@ISNA("NA") = 0

@ISNA(@NA) = 1

@ISNA(A18) = 1 if A18 contains NA produced by @NA

Related topics

@ISNONTEXT - Nontext Test

Syntax

@ISNONTEXT(Value)

Value	Empty cell, logical value, text, number, ERR, cell reference, or cell name to test.

@ISNONTEXT tests if the argument is not text. @ISNONTEXT also returns 1 if Value refers to an empty cell.

Examples

	A	B
1		Profits
2	8/31/95	"8/31/95"

Cell A1 above is empty; Cell A2 contains the formula @TODAY().

@ISNONTEXT(A1) = 1

@ISNONTEXT(B1) = 0

@ISNONTEXT(A2) = 1

@ISNONTEXT(B2) = 0

@ISNONTEXT(A1..B2) = returns the array {1|0|1|0}

Related topics

@ISNUMBER - Number Test

Syntax

@ISNUMBER(X)

X	A cell address or expression.

@ISNUMBER examines X and determines if it contains a numeric value. If X is blank or contains a numeric value, ERR, or NA, @ISNUMBER returns a 1. If X is a label or text, @ISNUMBER returns a 0. @ISNUMBER is usually used with @IF to determine whether an entry is a value.

Examples

@ISNUMBER(88) = 1

@ISNUMBER("88") = 0 (quotes signify a text string)

@ISNUMBER(9/15/87) = 1

@ISNUMBER(@ERR) = 1 (ERR and NA are numeric values)

Related topics

@ISODD - Odd Number Test

Syntax

@ISODD(Number)

Number	Value to test.

@ISODD returns 1 (true) if a specified number is odd, 0 (false) if it is even.
If Number is not an integer, it is truncated.
If Number is non-numeric, @ISODD returns ERR.

Examples

@ISODD(4) = 0
@ISODD(4.9) = 0
@ISODD(5) = 1
@ISODD(-5) = 1

Related topics

@ISSTRING - String Test

Syntax

@ISSTRING(X)

X	A cell address or expression.

@ISSTRING examines X and determines if it contains a label or text string. If X does (even if the string is empty), @ISSTRING returns 1. If X is blank or contains a numeric or date value, @ISSTRING returns 0.

Usually, @ISSTRING is used to test the contents of a cell. You can test any expression, however. Literal string arguments must be enclosed by double quotes.

Examples

@ISSTRING(55) = 0

@ISSTRING(2/5/88) = 0

@ISSTRING("Hello, world.") = 1

@ISSTRING("Hello, "&"world.") = 1

@ISSTRING("55") = 1

@ISSTRING(A15) = 1 if A15 contains a label or formula that results in a string, otherwise 0

@ISSTRING(A15&A16&"!!!") = 1 if A15 and A16 contain labels or formulas that result in strings

@ISSTRING("") = 1 ("" is an empty string)

@ISSTRING(@NA) = 0 (NA and ERR are considered numeric values)

Related topics

@KANSUUJI - Convert Kanji Number to Arabic Number

Syntax

@KANSUUJI(Kanji Number)

Kanji Number	The kanji number

@KANSUUJI converts a kanji number to its Arabic representation.

Related topics

@KURT - Kurtosis

Syntax

@KURT(List)

List	One or more numeric or cell values.

@KURT returns the kurtosis of List. The kurtosis of a data set measures a distribution's closeness to normality, indicating relative peakedness or flatness. A kurtosis greater than zero is referred to as leptokurtic. A kurtosis less than zero is referred to as platykurtic.

List must have four or more values. The standard deviation of List must not be 0.

@KURT uses this formula:

$$\left\{\frac{n(n+1)}{(n-1)(n-2)(n-3)}\sum\left(\frac{x_i-\bar{x}}{s}\right)^4\right\}-\frac{3(n-1)^2}{(n-2)(n-3)}$$

where s is the sample standard deviation.

Examples

@KURT(5,7,9,12,14,15,4,9,5,6) = -1.11117

@KURT(9.7,10,9.5,9.3,10.2,10,9.5,11) = 1.780277

@KURT(20,25,27,22,35,28) = 0.876754

Related topics

@LARGEST - *N*th Largest Number

Syntax

@LARGEST(Array, N)

Array	A numeric array or cells of values.
N	Number that indicates the rank in size from the data set Array; must be greater than 0 and less than or equal to the number of values in Array.

@LARGEST returns the Nth largest number in Array. Use @LARGEST to determine a value's rank in a data set from the top of that set.

If there are duplicates in Array, @LARGEST treats them as separate numbers.

Examples

@LARGEST({1,2,3,4,5,6,7,8,9,10},2) = 9

@LARGEST({1,2,3,4,5,6,7,8,9,10},4) = 7

@LARGEST({1,2,3,4,5,6,7,8,9,10},6) = 5

Related topics

@LASTBLANKPAGE - Last Blank Page

Syntax

@LASTBLANKPAGE(Block)

Block	A cell or reference; can be a link to another opened notebook (for example, [BUDGET]A:A1).

@LASTBLANKPAGE returns a string that contains the letters for the last unnamed blank sheet in a notebook that is not part of a group.

Quattro Pro searches for the last unnamed blank sheet (that is not in a group) starting at sheet IV and continuing toward sheet A. If there are no unnamed blank sheets (or if they are all in groups), @LASTBLANKPAGE returns ERR.

Example

@LASTBLANKPAGE(B17) = IG (if it is the last sheet that is blank and unnamed)

Related topics

@LASTCELLVALUE - Return Contents of Last Cell in Block

Syntax

@LASTCELLVALUE(block, <type>)

Block	A cell or reference.
Type	Number 1 (column) or 2 (row); the default type is 1.

@LASTCELLVALUE returns the contents of the last non-blank cell in the cells.

Quattro Pro searches for the last non-blank cell in the row or column. If there is no content to return, @LASTCELLVALUE returns 0.

Example

@LASTCELLVALUE(A1..I34,2) (the last cell in the row that is not blank)

Related topics

@LASTINGROUP - Last Sheet in Group

Syntax

@LASTINGROUP(Block, GroupName)

Block	A cell or cells of the notebook to check.
GroupName	A string value representing a group name.

@LASTINGROUP returns a string that contains the letters for the last sheet in the group named GroupName. @LASTINGROUP searches the notebook referenced by Block for the group. If the group does not exist, @LASTINGROUP returns ERR.

Example

@LASTINGROUP([REPORTQ4]A:C12,"Totals") = "C" (if the notebook named REPORTQ4 contains a group named Totals that ends with sheet C)

Related topics

@LBDAY - Last Business Day in Month

Syntax

@LBDAY(Date, <Holidays>, <Saturday>, <Sunday>)

Date	Number representing a date. See "Using dates and times in Quattro Pro."
Holidays	Cells containing dates that are holidays or the date of a single holiday or 0 to indicate no holidays (the default is 0).
Saturday	0 to specify that Saturday is not a business day; 1 to specify that Saturday is a business day (the default is 0).
Sunday	0 to specify that Sunday is not a business day; 1 to specify that Sunday is a business day (the default is 0).

@LBDAY returns the serial date number for the date of the last business day of the month in which Date falls.

Example

This formula calculates the last business day in November 1993, assuming that Sundays and the dates contained in cells A7..C9 are holidays.

@LBDAY(@DATE(93,11,1),A7..C9,1) = 34303 (November 30, 1993)

Related topics

@LCM - Least Common Multiple

Syntax

@LCM(X, Y)

X	Integer to find least common multiple of.
Y	Integer to find least common multiple of.

@LCM returns the least common multiple of X and Y (the smallest integer into which both X and Y can divide without leaving a remainder).

@LCM is otherwise known as the greatest common denominator, which is not to be confused with @GCD (the greatest common divisor).

Examples

@LCM(9,6) = 18

@LCM(24,12) = 24

Related topics

@LEFT - Leftmost Characters

Syntax

@LEFT(String, Num)

String	A string value.
Num	A numeric value equal to or greater than 0.

@LEFT returns the leftmost Num characters of String. It lets you extract a specified number of characters starting from the left end of a string or label.

If String is a numeric or date value or a blank cell, @LEFT returns ERR. If Num is longer than the length of String, all of String is returned. The number of characters returned is never greater than the length of the string.

Examples

@LEFT("Jennifer",5) = Jenni

@LEFT("Jennifer",15) = Jennifer

@LEFT("155",1) = 1

@LEFT(" Jennifer",6) = J (including five leading spaces)

@LEFT(123,1) = ERR (123 is a value)

@LENGTH(@LEFT("Jennifer",255)) = 8

Related topics

@LENGTH - Number of Characters

Syntax

@LENGTH(String)

String	A string value.

@LENGTH returns the number of characters in String, including spaces. You can combine strings or cell addresses with an ampersand (&). When String is a text string, it must be enclosed by double quotes.

If you try to reference a blank cell with this @function, Quattro Pro returns ERR.

Examples

@LENGTH("Hello, world.") = 13

@LENGTH(" Jennifer") = 9 (including preceding space)

@LENGTH("Greetings "&"earthling") = 19 (including space after Greetings)

@LENGTH(29584949) = ERR (29584949 is a value, not a string)

@LENGTH(A6&B10) = total number of characters in A6 and B10

@LENGTH(B10) = ERR (if B10 is blank or a value)

Related topics

@LETTERTOINDEX - Sheet/Column Index Corresponding to Letter(s)

Syntax

@LETTERTOINDEX(Letters)

Letters	A one- or two-character string enclosed in quotation marks; column and sheet letters run in sequence from A to Z, and continue from AA to AZ, up to IV.

@LETTERTOINDEX returns the index number (from 0 to 255) for column letters or sheet letters.

If Letters is a character string outside the range of sheet and column letters (for example, "IW"), @LETTERINDEX returns ERR.

Examples

@LETTERTOINDEX("A") = 0

@LETTERTOINDEX("B") = 1

@LETTERTOINDEX("IV") = 255

Related topics

@LINEST - Fits Line to Data

Syntax

@LINEST(KnownYs, <KnownXs>, <Const>, <Stats>)

KnownYs	Array of known y-values for the line y = mx + b.
KnownXs	Array of known x-values (optional).
Const	Logical value (optional) that tells @LINEST whether to force the constant b = 0: If Const is TRUE or omitted, @LINEST uses the actual value of b. If Const is FALSE, @LINEST sets b = 0, then adjusts the m-values so that y = mx.
Stats	Logical value (optional) that tells @LINEST whether to return more regression statistics. If Stats is TRUE, @LINEST returns the array $\{m_n, m_{n-1}, \ldots, m_1, b; se_n, se_{n-1}, \ldots, se_1, se_b; r^2, se_y; F, df; ss_{reg}$ If Stats is FALSE or omitted, @LINEST returns only the m-coefficients and b.

@LINEST uses the "least squares" method to calculate a straight line that best fits your data and returns an array to describe the line. @LINEST returns additional regression statistics when the argument Stats = TRUE.

- If known y-values are in one column, @LINEST takes each column of known x-values to be a separate variable. If known y-values are in one row, @LINEST takes each row of known x-values to be a separate variable.
- The argument KnownXs can include more than one set of variables. If you use more than one variable, KnownYs must be a single-column or single-row selection. Use commas to separate x-values in the same row and semicolons to separate rows.
- If you omit the argument KnownXs, @LINEST assumes it is the array {1, 2, 3,...} of a size equal to KnownYs.

The equation for the line is:

$\{m_n, m_{n-1}, \ldots, m_1, b; se_n, se_{n-1}, \ldots, se_1, se_b; r^2, se_y; F, df; ss_{reg}, ss_{resid}\}$

or

y = mx + b

where y is a function of x, the independent variable. The m-values are coefficients that correspond to the x-values. The value b is a constant. Values y, x, and m can be vectors. @LINEST returns the array

$\{m_n, m_{n-1}, \ldots, m_1, b\}$

- For @LINEST to work, the x variables need to be in one contiguous block of data.
- Slope and y-intercept define a straight line.

Slope (m): Given any two points on a line (x_1, y_1) and (x_2, y_2),
slope m = $(y_2 - y_1)/(x_2 - x_1)$.
Y-intercept (b): The y-intercept of a line = the value of y where the line crosses the y-axis.
The equation of a straight line is y = mx + b. Knowing the values of m and b, you can find any point on the line if you know either y or x. You can also use @TREND.

- If there is only one independent x-variable, use the following formulas to find the slope and y-intercept values directly:

  Slope: @INDEX(@LINEST(KnownYs, KnownXs),1)

  Y-intercept: @INDEX(@LINEST(KnownYs, KnownXs),2)
- The accuracy of @LINEST's calculation depends on the scatter of your data. The more linear your data, the more accurate the @LINEST calculation. @LINEST determines the best fit for the data by the least squares method. Given only one independent x-variable, @LINEST uses the following formulas to calculate m and b:

$$m = \frac{n\left(\sum xy\right)-\left(\sum x\right)\left(\sum y\right)}{n\left(\sum\left(x^2\right)\right)-\left(\sum x\right)^2}$$

$$b = \frac{\left(\sum y\right)\left(\sum\left(x^2\right)\right)-\left(\sum x\right)\left(\sum xy\right)}{n\left(\sum\left(x^2\right)\right)-\left(\sum x\right)^2}$$

- @LINEST fits the best straight line to your data; @LOGEST fits the best exponential curve. To decide which best fits your data, use @TREND for a straight line, or @GROWTH for an exponential curve, without the NewXs argument, to see an array of y-values predicted along that line or curve at your actual data points. Then compare predicted values with actual values. You can chart them for a visual comparison.
- Note that y-values predicted by the regression equation might not be valid if they are outside the cells of y-values you used to determine the equation.

Example 1 - Slope and Y-intercept

@LINEST({4,3,2,1},{0,1,2,3}) = {-1, 4}, meaning the slope = -1 and y-intercept = 4

Example 2 - Simple Linear Regression

Sales for your company in its first four quarters are entered in a selection named Sales:

	A	B
1	Quarter	Sales
2	1	$80,000
3	2	$90,000
4	3	$95,000
5	4	$105,000

To predict third-quarter sales for the following year, enter the formula

@SUM(@LINEST(B2..B5)*{7,1})

The projected sales are displayed in the selected cells:

6	5	
7	6	
8	7	$128,500
9	8	

In general, @SUM({m,b}*{x,1}) = mx + b, the estimated y-value for a specified x-value. You can also use @TREND.

Related topics

Regression Statistics

@LINEST and @LOGEST return additional regression statistics when the argument Stats = TRUE:

Statistic	Description
se1, se2,..., sen	Standard error values for coefficients m1, m2,..., mn.
seb	Standard error value for constant b (NA when Const is FALSE).
r2	Coefficient of determination, ranging from 0 to 1. Compares the estimated and actual y-values. When r2 = 1, there is a perfect correlation in the sample (no difference between the estimated and actual y-values). As r2 approaches 0, y-values become unpredictable. See note below for information about how r2 is calculated.
sey	Standard error for the y estimate.
F	F-observed value (F statistic). The F statistic measures whether the relationship observed between the y and x could occur by chance alone.
df	Degrees of freedom, used to find F-critical values in a statistical table. Compare these F-critical values to the F statistic to determine a confidence level for the model.
ssreg	Regression sum of squares.
ssresid	Residual sum of squares.

The additional regression statistics are displayed as follows:

m_n	m_{n-1}	. . .	m_2	m_1	b
se_n	se_{n-1}	. . .	se_2	se_1	se_b
r^2	se_y				
F	dF				
ssreg	ssresid				

In regression analysis, Quattro Pro calculates for each point the squared difference between the y-value estimated for that point and its actual y-value. The sum of these squared differences is called the residual sum of squares. Quattro Pro then calculates the sum of the squared differences between the actual y-values and the average of the y-values, which is called the total sum of squares (regression sum of squares + residual sum of squares). The smaller the residual sum of squares is, as compared with the total sum of squares, the larger the value of the coefficient of determination, r2, a measure of how well the equation resulting from the regression analysis explains the relationship among the variables.

Related topics

Multiple Linear Regression

Example 1

High school grade point averages (GPA) are thought to depend on factors like the number of credit hours taken in the semester, the student's year in school, and hours worked at an outside job. Data for a small sample of students is in the following table:

	A	B	C	D
1	GPA	Credit hrs	Yrs	Hrs on job
2	3	15	4	2
3	2	16	3	10
4	4	12	4	0
5	3.5	15	2	0

For the full regression display, enter the formula

@LINEST(A2..A5,B2..D5,TRUE,TRUE)

The result is the following array:

	A	B	C	D
6	-0.11	-0.14	-0.26	7.68
7	0.00	0.00	0.00	0.00
8	1.00	0.00	NA	NA
9	0.00	0.00	NA	NA
10	2.19	0.00	NA	NA

Numbers in a full regression array are:

m_n	m_{n-1}	. . .	m_2	m_1	b
se_n	se_{n-1}	. . .	se_2	se_1	se_b
r^2	se_y				
F	dF				
ssreg	ssresid				

The multiple regression equation, y = m1*x1 + m2*x2 + m3*x3 + b, can now be obtained using the values from row 6:

y = - 0.26*x1 + (- 0.14)*x2 + (- 0.11)*x3 + 7.68

where

x1	credit hours
x2	year in school
x3	hours on job

To estimate the expected GPA of a third-year student who takes 14 credit hours and works 5 hours a week at an outside job, substitute:

y = - 0.26*14 - 0.14*3 - 0.11*5 + 7.68 = 3.1

Example 2

You have gathered data on 10 homes recently sold in your neighborhood. The table lists the amount each home sold for, the number of bedrooms and baths, total floor space in square feet, and lot size:

	A	B	C	D	E
1	Sold for	Bedrms	Baths	Sq. ft.	Acres
2	$255,000	2	2	1500	1
3	$435,500	3	2.5	1900	5

4	$395,500	2	2	1200	10
5	$495,000	4	3	2400	1
6	$125,000	1	1	950	0.25
7	$270,000	2	2	1280	2
8	$595,000	4	1.5	2100	4
9	$249,500	2	2	1350	0.5
10	$255,000	2	1	1100	1
11	$244,500	2	1	1080	0.25

@LINEST(A2..A11,B2..E11,TRUE,TRUE) returns the following table of the full regression statistics:

	A	**B**	**C**	**D**	**E**
12	x4=acres	x3=sq ft	x2=baths	x1=bedrooms	constant b
13	17373.966	46.18235	-29475.5	118123.37	-10502.1
14	1411.8687	39.42453	11029.34	16402.137	13092.08
15	0.9967254	10962.52	NA	NA	NA
16	380.471	5	NA	NA	NA
17	1.829E+11	6.01E+08	NA	NA	NA

Note that these values are rounded off, because the default column width does not allow display of all digits, though full precision is stored in the cell. In further calculations, reference the cell for the value.

The multiple regression equation

$$y = m_1 x_1 + m_2 x_2 + m_3 x_3 + m_4 x_4 + b$$

can now be obtained using the values from row 13:
y = 118123.37*x1 - 29475.5*x2 + 46.18235*x3 + 17373.966*x4 - 10502.1
where

x1	number of bedrooms
x2	number of baths
x3	square feet of floor space
x4	lot size in acres

To find the expected selling price of a 3-bedroom, 2-bathroom house of 2250 square feet on 3.5 acres, enter

+D13*3 + C13*2 + B13*2250 + A13*3.5 + E13 = $449,636

You can also calculate this value using @TREND.

Using F and R2 Statistics

The coefficient of determination R-squared, in A15, is 0.9967254, showing a strong relationship between the independent variables and sales price.

To determine if such a strong correlation could occur by chance alone, compare the F statistic in A16 with the F-critical value. Tables of F-critical values can be found in many statistics textbooks. To read the table, assume a single-tailed test and use an alpha value of 0.05, meaning there is less than a 5% chance the correlation is accidental. For the degrees of freedom (abbreviated in most tables as v1 and v2), use

v1 = 4, the number of variables, and

v2 = s - (v1 + 1) = 10 - (4 + 1) = 5

where s = the number of houses in the sample.

The F-critical value in the table, for alpha = 0.05, v1 = 4, and v2 = 5, is 5.19. The F-observed value in Cell A16 is 380.471, which is much larger than 5.19. This means the regression equation can be used with assurance to predict expected sales prices for other homes in the area.

Calculating T-Statistics

You can also test each independent variable to see how well it predicts the expected selling price of a home. You do this by comparing the t-observed value for each variable with t-critical from a statistics table.

First divide slope m for each variable by its estimated standard error, se. For lot size, m appears in A13 and se is in A14, so

+A13/A14 (or 17373.966 / 1411.8687) = 12.30565

From a table of critical values of t, single tail, for alpha = 0.05 and 5 degrees of freedom (Cell B16 in the regression array above), t-critical = 2.015

The absolute value of t-observed for lot size, 12.30565, is above t-critical, showing lot size to be an important variable in predicting sales price for homes in the area. The other variables can be tested in the same way.

Variable	**t-observed value**
Number of bedrooms	7.201712
Number of baths	-2.67246
Floor space	1.171412
Lot size	12.30565

Number of bedrooms, number of baths, and lot size all have absolute values of t-observed > t-critical, so these variables are useful in predicting the sales price of a home in the area.

Related topics

@LINTERP - Linear Interpolation

Syntax

@LINTERP(KnownX, KnownY, X)

KnownX	One-dimensional selection containing X values.
KnownY	One-dimensional selection containing Y values corresponding to the X values in KnownX.
X	Number for which the corresponding Y value is desired.

@LINTERP interpolates a Y value corresponding to X using the XY pairs specified by KnownX (which contains the X coordinates) and KnownY (which contains the Y coordinates). If X lies between two values in KnownX, @LINTERP interpolates using those two values and their respective KnownY counterparts. If X is outside the range of KnownX, the Y value is extrapolated based on the slope of the line between the two closest points.

KnownX and KnownY do not have to be the same size. If KnownY is smaller than KnownX, the last value in KnownY is used as the corresponding Y value for extra KnownX values. If KnownY is larger than KnownX, its extra values are ignored.

Example

This formula calculates the Y value for the X value 6.7 if the data in the next figure is used.

@LINTERP(A3..A9,B3..B9,6.7) = 17.5976

	A	B
1	x values	y values
2		
3	-28.345	-9.7821
4	-17.89	-5.6667
5	0.9232	2.891
6	1.212	2.9978
7	4.552	13.67
8	10.75	25.003
9	30.8	33.33

Related topics

@LLDEC - Latitude and Longitude to Decimal

Syntax

@LLDEC(Degrees, Minutes, Seconds, Direction)

Degrees	Degrees of Latitude or Longitude.
Minutes	Minutes of Latitude or Longitude.
Seconds	Seconds of Latitude or Longitude.
Direction	For Latitude, North (1) or South (2) of the equator; for Longitude, East (3) or West (4) of the prime meridian at Greenwich, England.

@LLDEC converts a latitude or longitude coordinate to decimal. Latitude south of the equator is represented as a negative number; longitude west of the prime meridian is represented as a negative number.

Examples

@LLDEC(38, 45, 15, 1) = 38.75417

@LLDEC(38, 45, 15, 2) = -38.75417

@LLDEC(143, 15, 25, 4) = -143.257

Related topics

@LN - Natural Logarithm

Syntax

@LN(X)

X	A numeric value > 0.

@LN returns the natural logarithm of X. A natural logarithm uses the mathematical constant e as a base. @LN produces the inverse of @EXP.

Examples

@LN(3) = 1.098612289

@LN(@EXP(10)) = 10

@LN(16)/@LN(2) = 4

@LN(-4) = ERR (-4 is less than 0)

Related topics

@LOG - Base 10 Logarithm

Syntax

@LOG(X)

X	A numeric value > 0.

@LOG returns the logarithm of a number in base 10.

Examples

@LOG(1000) = 3

@LOG(10^23.8) = 23.8

@LOG(16)/@LOG(2) = 4 (log to base 2 of 16)

Related topics

@LOGBASE - Logarithm to Base X

Syntax

@LOGBASE(Number, <Base>)

Number	Positive real number.
Base	Base of the logarithm (optional); the default is 10.

@LOGBASE calculates the logarithm of a specified number to the specified base.
<Base> must be a value greater than 1; otherwise ERR is returned.

Examples

@LOGBASE(100) = 2
@LOGBASE(27,3) = 3
@LOGBASE(18,2) = 4.169925

Related topics

@LOGCONV - Converts Logarithm to Another Base

Syntax

@LOGCONV(Number, FromBase, ToBase)

Number	Value to be converted = the log of a number m to the base b.
FromBase	Positive integer greater than 1.
ToBase	Positive integer greater than 1.

@LOGCONV converts a specified logarithm from one specified base to another. For Number = $\log_b m$ @LOGCONV uses the formula:

$\log_a m = \text{Number} * \log_a b$

where

m	a number
b	original base
a	new base

If FromBase = 1, @LOGCONV returns ERR because the log of b to the base a is zero.

If FromBase or ToBase are negative or non-integer, @LOGCONV returns ERR.

Examples

@LOGCONV(2,4,2) = 4

@LOGCONV(3,4,2) = 6

Related topics

@LOGEST - Fits Curve to Data

Syntax

@LOGEST(KnownYs, <KnownXs>, <Const>, <Stats>)

KnownYs	Array of known y-values for the curve y = b*m^x.
KnownXs	Array of known x-values.
Const	Logical value (optional) that tells @LOGEST whether to force the constant b = 1: If Const is TRUE or omitted, @LOGEST uses the actual value of b. If Const is FALSE, @LOGEST sets b = 1, then adjusts the m-values so that y = m^x.
Stats	Logical value (optional) that tells @LOGEST whether to return more regression statistics. If Stats is TRUE, @LOGEST returns the array $\{m_n, m_{n-1}, \ldots, m_1, b; se_n, se_{n-1}, \ldots, se_1, se_b; r^2, se_y; F, df; ss_{reg}$ If Stats is FALSE or omitted, @LOGEST returns only the m-coefficients and b.

@LOGEST calculates an exponential curve that fits your data and returns an array to describe the curve. @LOGEST returns additional regression statistics when the argument Stats = TRUE.

- If known y-values are in one column, @LOGEST takes each column of known x-values to be a separate variable. If known y-values are in one row, @LOGEST takes each row of known x-values to be a separate variable.
- If any of the known y-values are 0 or negative, @LOGEST returns ERR.
- The argument KnownXs can include more than one set of variables. If you use only one variable, KnownYs and KnownXs can be selections of any shape, but must have the same dimensions. If you use more than one variable, KnownYs must be a single-column or single-row selection. Use commas to separate x-values in the same row and semicolons to separate rows.
- If you omit the argument KnownXs, @LOGEST assumes it is the array {1, 2, 3,...} of a size equal to KnownYs.

The equation for the curve is $y = (b * (m_1^{x_1}) * (m_2^{x_2}) * \ldots)$ or $y = b * m^x$

where y is a function of x, the independent variable. The m-values are bases that correspond to the exponent x-values. The value b is a constant. Values y, x, and m can be vectors. @LOGEST returns the array $\{m_n, m_{n-1}, \ldots, m_1, b\}$.

If there is only one independent x-variable, use the following formulas to find the slope and y-intercept values directly:

Slope m: @INDEX(@LOGEST(KnownYs, KnownXs),1)

Y-intercept b: @INDEX(@LOGEST(KnownYs, KnownXs),2)

The accuracy of @LOGEST's calculation depends on how close the plot of your data comes to an exponential curve. @LINEST fits the best straight line to your data; @LOGEST fits the best exponential curve. To decide which best fits your data, use @TREND for a straight line, or @GROWTH for an exponential curve, without the NewXs argument, to see an array of y-values predicted along that line or curve at your actual data points. Then compare predicted values with actual values. You can chart them for a visual comparison.

The y-values predicted by the regression equation might not be valid if they are outside the cells of the y-values you used to determine the equation.

Using @LOGEST to test an equation is similar to using @LINEST. However, the additional statistics @LOGEST returns are based on the following linear model:

$\ln y = x_1 \ln m_1 + \ldots + x_n \ln m_n + \ln b$

Remember this when evaluating the additional statistics, especially the sei and seb values, which should be compared to ln mi and ln b, not to mi and b.

Example

Sales for your company in its first four quarters are entered in a selection named Sales:

	A	B
1	Quarter	Sales
2	1	$75,000
3	2	$90,000
4	3	$115,000
5	4	$140,000

For the full regression display, enter the formula

@LOGEST(Sales,A2..A5,TRUE,TRUE)

The regression statistics are returned as follows:

6	1.235849	60133.78
7	0.008186	0.022419
8	0.99702	0.018305
9	669.1028	2
10	0.224208	0.00067

The equation for the curve is y = b * m1^x1.

So, using the values from the array:

y = 60133.78 * 1.23585^x

To estimate sales for future months, substitute the month number for x in this equation, or use @GROWTH. For example, for the fifth month:

y = 60133.78 * 1.23585^5 = $173,359

You can use the additional regression statistics (cells A6..B10 in this example) to determine how useful the equation is for predicting future values. For instance, the value in cell A8 in this example is the r-squared (r2) value. When r-squared = 1, there is a perfect correlation in the sample (no difference between the estimated and actual y-values). As r-squared approaches 0, y-values become unpredictable.

Related topics

@LOGINV - Inverse of Cumulative Lognormal Distribution

Syntax

@LOGINV(Prob, Mean, SDev)

Prob	Probability associated with the cumulative lognormal distribution function; $0 \leq Prob < 1$.
Mean	Mean of ln(x).
SDev	Standard deviation of ln(x); must be > 0.

@LOGINV returns the inverse of the cumulative lognormal distribution.

Example

@LOGINV(0.027985,2.5,0.8) = 2.640543

Related topics

@LOGNORMDIST - Cumulative Lognormal Distribution

Syntax

@LOGNORMDIST(X, Mean, SDev)

X	Value to evaluate the function; must be > 0.
Mean	Mean of ln(x).
SDev	Standard deviation of ln(x); must be > 0.

@LOGNORMDIST returns the cumulative lognormal distribution.

Example

@LOGNORMDIST(3,2.5,0.8) = 0.03991

Related topics

@LOOKUP - Looks Up Values

Syntax

@LOOKUP(Value, LookupVector, ResultVector)

Value	Value to look for in LookupVector; can be a number, text, logical value, or reference to a value.
LookupVector	Cells containing only one row or column.
ResultVector	Selection of the same dimensions as LookupVector and containing corresponding values.

@LOOKUP looks up values in a specified row or column. It looks in a designated row or column for a specified value, moves to the corresponding cell in another specified row or column, and returns the value it finds there.

Values in LookupVector and ResultVector must be in ascending order for @LOOKUP to return the correct value (upper- and lowercase text have the same values).

@LOOKUP lets you specify row or column to look in, as well as the one containing the value to return.

Examples

From the wind chill table below, you want to find the apparent temperature at a certain wind speed.

	A	**B**	**C**	**D**	**E**	**F**	**G**
1				Wind Speed			
2	Deg. F	5	10	15	20	25	30
3	10	6	-9	-18	-25	-29	-33
4	20	16	4	-5	-10	-15	-18
5	30	27	16	9	4	0	-2
6	40	37	28	22	18	16	13
7	50	48	40	36	32	30	28

To find the apparent temperature when the thermometer reads 25 degrees Fahrenheit and the wind is blowing at 10 miles an hour, enter

@LOOKUP(25,A2..A7,C2..C7)

The result is 4 degrees F, which is the apparent temperature corresponding to a thermometer reading of 20 degrees, the next lower value.

At 20 degrees, when the wind is twice as strong (20 mph), the apparent temperature is

@LOOKUP(20,A2..A7,E2..E7) = -10

Related topics

@LOWER - String in Lowercase

Syntax

@LOWER(String)

String	A string value.

@LOWER returns String in lowercase characters. Numbers and symbols within a string are unaffected. Numeric and date values return ERR.

Examples

@LOWER("UPPER") = upper

@LOWER("Hello, world.") = hello, world.

@LOWER("145 Bancroft Lane") = 145 bancroft lane

@LOWER(4839) = ERR

@LOWER(@LEFT("Johnson",1)) = j

Related topics

@LWKDAY - Last Weekday

Syntax

@LWKDAY(Wkday, Month, Year, <AuxWkday>)

Wkday	Number from 1 (Saturday) to 7 (Friday).
Month	Number from 1 (January) to 12 (December).
Year	Number from 0 (1900) to 199 (2099) or a standard year like 1993.
AuxWkday	Auxiliary day of the week that must fall in the same week as Wkday; 0 for no auxiliary day or a number from 1 (Saturday) to 7 (Friday) indicating the auxiliary day (the default is 0).

@LWKDAY returns the serial date number for the date of the last occurrence of Wkday in Month of Year (for example, the last Tuesday in November 1994).

See "Using dates and times in Quattro Pro."

You can use AuxWkday to specify that both Wkday and AuxWkday must fall in the same week of the same month. (See the second example.)

The valid date calculation range for this function is 01/01/1900 through 12/31/2099.

Examples

/@LWKDAY(3,6,115)= 42184 (June 29, 2015), the date of the last Monday in June 2015.

@LWKDAY(4,11,94,7)= 34660 (November 22, 1994), the last Tuesday on which both the last Tuesday and a Friday fall on the same week of November 1994.

Related topics

@MATCH - Position of Matching Cell

Syntax

@MATCH(Cell Contents, Block, <Match Type>)

Cell Contents	Numeric or string value to be matched.
Block	Cells, contiguous selections, an array, or array reference.
<Match Type>	-1, 0, or 1. Match Type specifies which cell positions are returned: -1 = smallest, 1 = largest, 0 = first found.

@MATCH returns the relative position of the cell in Block whose contents match the Cell Contents argument.

@MATCH returns the position of the item rather than the item itself.

@MATCH returns ERR or 0 if no matches are found.

Match Type = -1, returns the position of the smallest value that is greater than or equal to Cell Contents. Selections must be arranged in descending order.

Match Type = 0, returns the position of the first value that is exactly equal to Cell Contents. Blocks may be arranged in any order.

Match Type = 1, returns the position of the largest value that is less than or equal to Cell Contents. Selections must be arranged in ascending order.

Match Type =1 is the Default Value.

Examples

	A	B
1	Name	Grade
2	Fred	92
3	Mary	84
4	Reno	75
5	Anne	67
6	John	54

In cell selections, or arrays of cells, @MATCH uses zero-based numbering (the first cell in the selection equals 0).

@MATCH(75,B1..B4,-1) = 3

@MATCH(75,B1..B4,0) = 3

@MATCH(75,B1..B4,1) = 0 (the selection is ordered incorrectly)

@MATCH("RENO",A1..A4,0) = 3 (Reno is third in his class)

Also, @MATCH("b",{"a","b","c"},0) = 1 (the first item in the array equals 0).

Related topics

@MAX - Maximum Value

Syntax

@MAX(List)

List	One or more numeric values, cell addresses, and references or names, separated by commas.

@MAX returns the largest numeric or data value in List. If more than one selection is listed, commas must separate the selections. If any of the cells referenced contain ERR, the resulting value is ERR.

Examples

	A	B	C	D
1		Jan.	Feb.	Mar.
2	JA	$652	$833	$599
3	MH	$456	$305	$522
4	RB	$68	$59	$73
5	PD	$379	$379	$379
		---	---	---
6		$1,555	$1,576	$1,573

@MAX(B3..B5) = $456

@MAX(C3..C5,D3..D5) = $522

@MAX(A1..D6) = $1,576

@MAX(B2..C5,D3) = $833

Related topics

@MAXLOOKUP - Cell Containing Largest Value

Syntax

@MAXLOOKUP(BlockList)

BlockList	Cells or list of selections containing numeric values.

@MAXLOOKUP returns the address of the cell containing the largest value in specified cells or list of selections. @MAXLOOKUP return ERR if no cells in BlockList contain values.

Separate the entries in BlockList with argument separators.

Labels and blank cells in BlockList are ignored.

Example

Suppose you keep lists of contributors and the amounts they gave in separate notebooks for each year. The notebooks have the same layout, because you built them from the same template. To find the cell location of the maximum amount anyone gave over all the years for which you have files, enter

@MAXLOOKUP([YR1992]AMOUNT, [YR1993]AMOUNT, [YR1994]AMOUNT, [YR1995]AMOUNT)

Related topics

@MDAYS - Calendar Days in Month

Syntax

@MDAYS(Month, Year)

Month	Number from 1 (January) to 12 (December).
Year	Number from 0 (1900) to 199 (2099) or a standard year like 1993.

@MDAYS returns the number of calendar days in Month of Year.

The valid date calculation range for this function is 01/01/1900 through 12/31/2099.

See "Using dates and times in Quattro Pro."

Example

@MDAYS(2,1996)= 29, the number of days in February 1996.

Related topics

@MDET - Determinant of a Matrix

Syntax

@MDET(Array)

Array	A numeric array or a selection of values specifying a square matrix; must have an equal number of rows and columns, and cannot contain blank cells.

@MDET calculates the determinant of a matrix (Array). The determinant is obtained by taking any row or column of the matrix, forming the products of each element and its cofactor, and taking the sum of the products; @MDET uses this formula:

$$|A| = \sum_{j=1}^{n} a_{ij} A_{ij}$$

where aij is the element in the ith row and jth column of A and the cofactor aij is the product of the determinant of the minor matrix Mij, formed by deleting row i and column j of A, and a power of -1:

$$A_{if} = (-1)^{i+j} \left| M_{ij} \right|$$

If Array does not contain the same number of rows and columns, or if Array contains any blank cells, @MDET returns ERR. If any two rows or columns in Array are equal or have proportional elements, @MDET returns 0.

The result of @MDET is accurate to approximately 16 digits, which can lead to a small numeric error when the cancellation is not complete.

Examples

@MDET({12,15,21|8,13,17|16,32,44}) = 144

This formula calculates the determinant of the data shown in the next figure:

@MDET(C3..F6) = -2869.95

	C	D	E	F
3	2.908	-2.253	6.775	3.97
4	1.212	1.995	2.266	8.008
5	4.552	5.681	8.85	1.302
6	5.809	-5.03	0.099	7.832

Related topics

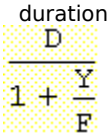
@MDURATION - Modified Duration

Syntax

@MDURATION(Settle, Maturity, Coupon, Yield, <Freq>, <Calendar>)

Settle	Number representing the settlement date; must be < Maturity.
Maturity	Number representing the maturity date; must be > Settle.
Coupon	Coupon rate; 0 ≤ Coupon ≤ 1.
Yield	Annual yield; 0 < Yield ≤ 1.
Freq	Frequency of coupon payments in number of payments per year (can be 1, 2, 3, 4, 6, or 12; the default is 2).
Calendar	Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

@MDURATION returns the modified Macaulay duration for a bond with assumed par value of 100. Modified duration is calculated using this formula:



D	Duration
Y	Yield
F	Frequency

Example

This formula calculates the modified duration of a bond with these terms: Settle is August 8, 1992, Maturity is November 15, 1998, Coupon is 9%, and Yield is 8.816%.

@MDURATION(@DATE(92,8,8),@DATE(98,11,15),0.09,0.08816) = 4.631923

Related topics

@MEDIAN -Middle Value

Syntax

@MEDIAN(List)

List	One or more numeric or cell values.

@MEDIAN returns the middle value in a range of values in a data set arranged in ascending or descending order. If the number of values in the data set is even, the median is the mean of the two middle values. Use @MEDIAN when you want a more robust estimation of the central value in a distribution than you obtain with @AVG.

Examples

@MEDIAN(10,12,15,25,30) = 15

@MEDIAN(2,4,5,5,6,8,9,9) = 5.5

Related topics

@MEMAVAIL - Memory Available

Syntax

@MEMAVAIL

@MEMAVAIL returns the number of bytes of memory currently available.

Example

@MEMAVAIL = 47819 (if 47,819 bytes of memory are available)

Related topics

@MEMEMSAVAIL - Expanded Memory Available

Syntax

@MEMEMSAVAIL

@MEMEMSAVAIL is included for compatibility with Quattro Pro for DOS; it always returns NA under Windows.

Related topics

@MID - Extract Characters from String

Syntax

@MID(String, StartNumber, Num)

String	A string value.
StartNumber	A numeric value equal to or greater than 0.
Num	A numeric value equal to or greater than 0.

@MID extracts the first Num characters of String starting at StartNumber, which is the number of characters to the right of the first character (character 0). It is similar to @LEFT, which extracts Num characters of String beginning with the first character. The difference is that you can specify a character other than the first character in the string.

String can be any text string (enclosed by quotes) or reference to a cell containing a label. If StartNumber is greater than or equal to the length of String or if Num is 0, the result is "", or an empty string.

Examples

@MID("Abraham Lincoln",8,7) = Lincoln

@MID("George Washington",7,4) = Wash

@MID("Theodore Roosevelt",19,5) = ""

@MID(A23,@FIND("Roosevelt",A23,0),@LENGTH("Roosevelt")) = Roosevelt (if A23 = Franklin Roosevelt)

Related topics

@MIN - Minimum Value

Syntax

@MIN(List)

List	One or more numeric values, cell addresses, and references or names, separated by commas.

@MIN returns the smallest numeric or data value in List. If List contains more than one value, commas must separate the values. Labels are treated in all statistical functions as 0 and should therefore be excluded from List.

If List is entered as cells and one or more cells in the cells are blank, the blanks are excluded from the calculation; otherwise, blanks are treated as 0.

Examples

	A	B	C	D
1		Jan.	Feb.	Mar.
2	JA	$652	$833	$599
3	MH	$456	$305	$522
4	RB	$68	$59	$73
5	PD	$379	$379	$379
		---	---	---
6		$1,555	$1,576	$1,573

@MIN(B3..B6) = $68

@MIN(B2..D2,B4..D4) = $59

@MIN(B3..D3) = $305

Related topics

@MINLOOKUP - Cell Containing the Smallest Value

Syntax

@MINLOOKUP(BlockList)

BlockList	Cells or list of selections containing numeric values.

@MINLOOKUP returns the address of the cell containing the smallest value in specified cells or list of selections. @MINLOOKUP return ERR if no cells in BlockList contain values.

Separate the entries in BlockList with argument separators.

Labels and blank cells in BlockList are ignored.

Example

Suppose you keep lists of contributors and the amounts they gave in separate notebooks for each year. The notebooks have the same layout, because you built them from the same template. To find the cell location of the minimum amount anyone gave over all the years for which you have files, enter

@MINLOOKUP([YR1992]AMOUNT, [YR1993]AMOUNT, [YR1994]AMOUNT, [YR1995]AMOUNT)

Related topics

@MINUTE - Minute Portion of Date Serial Number

Syntax

@MINUTE(DateTimeNumber)

DateTimeNumber	A numeric value between -109571 and 474816.9999999, representing a date/time serial number: -109571 = January 1, 1600; 0 = December 31, 1899; 474816 = December 31, 3199; the decimal = time (24 hr).

See "Using dates and times in Quattro Pro."

@MINUTE returns the minute portion of DateTimeNumber. DateTimeNumber must be a valid date/time serial number. Because only the decimal portion of a serial number pertains to time, the integer portion of the number is disregarded. The result is between 0 and 59.

To extract the minute portion of a string that is in time format (instead of serial format), use @TIMEVALUE with @MINUTE to translate the time into a serial number. You can also use @TIME to enter a time value instead of a serial number.

Examples

@MINUTE(.36554) = 46

@MINUTE(.2525) = 3

@MINUTE(35) = 0

@MINUTE(@TIME(3,15,22)) = 15

@MINUTE(@TIMEVALUE("10:08 am")) = 8

Related topics

@MINVERSE - Inverse Matrix

Syntax

@MINVERSE(Array)

Array	Square numeric array (same number of rows as columns); you can use a cell reference or cell name or an array constant like {1,2|3,4}.

@MINVERSE returns the inverse matrix for a matrix stored in a square array. Matrices are helpful in solving problems with many variables in mathematics and economics.

@MINVERSE returns ERR if:

- Any cells in Array are empty or contain text.
- Array does not have an equal number of rows and columns.

To show how the inverse of a 2 x 2 matrix is calculated, suppose cells A1..B2 contains the letters a, b, c, and d, representing any four numbers:

	A	**B**
1	a	c
2	b	d

The inverse of the matrix in A1..B2 is:

	A	**B**
1	d/(a*d - b*c)	b/(b*c - a*d)
2	c/(b*c - a*d)	a/(a*d - b*c)

The product of a matrix and its inverse is the identity matrix, in which all diagonal values = 1 and all other values = 0.

The result of @MINVERSE is accurate to approximately 16 digits; this can lead to a small numeric error when the cancellation is not complete.

Some square matrices cannot be inverted, and @MINVERSE returns ERR. The determinant is 0 for a matrix that cannot be inverted.

Use @INDEX to get individual elements from an inverse matrix.

Examples

@MINVERSE({1,2|3,4}) = {-2,1|1.5,-0.5}

@MINVERSE({2,1,0|0.5,1,-0.5|3,2,1}) = {1,-0.5,-0.25|-1,1,0.5|-1,-0.5,0.75}

Related topics

@MIRR - Modified Internal Rate of Return

Syntax

@MIRR(Block, FinRate, ReinvRate, <Type>)

Block	Cells containing cash flows; negative = outflow, positive = inflow.
FinRate	Interest rate paid for funds used in cash flows.
ReinvRate	Interest rate received on reinvested funds used in cash flows.
Type	Timing of the cash flows (optional):
	0 = end of each period (default)
	1 = beginning of each period

@MIRR calculates the modified internal rate of return on an investment consisting of payments made at regular intervals.

The cells must contain at least one positive value and one negative value. Normally, the first cash-flow amount in the cells is a negative number (a cash outflow) that represents the investment. Quattro Pro assigns the value 0 to all blank cells and labels in Block and includes them in the calculation.

@MIRR helps you determine the profitability of an investment. To assess an investment, combine @MIRR with other financial functions, like @NPV.

MIRR relates to NPV by the following formula:

$$MIRR = \left(\frac{-NPV(r,ci) * (1+r)^{n}}{NPV(f,co) * (1+f)} \right)^{\frac{1}{n-1}} - 1$$

where

ci	cash received in (positive)
co	cash paid out (negative)
n	number of cash flows
f	finance rate
r	reinvestment rate

Examples

You bought a small cafe at the end of 1990 for $250,000. The first year of operation (1992) you spent a lot on restoration and advertising, but since then profits have grown. You want to know when the business started to pay for itself and what the return is now. Your finance rate is 9.5%, your reinvestment rate is 11.5%, and you reinvest your profits at the end of the period.

	A	B	C	D
1	Year	Profit/Loss	end/per	beg/per
2	1990	($250,000)		
3	1991	($50,000)		
4	1992	$112,500	-25.31%	-38.32%
5	1993	$120,500	-2.31%	-5.95%
6	1994	$128,500	8.32%	8.03%
7	1995	$136,500	13.77%	14.64%

For each year of profit, enter @MIRR in Column C for the cash flow up through that year. For example, in Cell C4,

@MIRR(B2..B4,0.095,0.115) = -25.31% and in Cell C7,

@MIRR(B2..B7,0.095,0.115) = 13.77%. Your investment began to pay off in 1994.

If you could reinvest your profits at the beginning of the period instead of the end, your return would be higher. You found this out by entering the same formulas in Column D but with Type = 1. For example, in Cell D7,

@MIRR(B2..B7,0.095,0.115,1) = 14.64%

Related topics

@MMULT - Matrix Product of Two Arrays

Syntax

@MMULT(Array1, Array2)

Array1, Array2 Arrays to be multiplied.

@MMULT calculates the matrix product of two arrays. The resulting array has the same number of rows as Array1 and the same number of columns as Array2.

- The number of columns in Array1 must equal the number of rows in Array2.
- Array1 and Array2 must contain only numbers.
- Array1 and Array2 can be specified as cell names or references, or array constants.

The matrix product array a of two arrays b and c is:

$$a_{ij} = \sum_{k=1}^{n} b_{jk} c_{kj}$$

Examples

@MMULT({2,3|1,0},{1,0|0,1}) = {2,3|1,0}

Given the following arrays,

	A	B	C
1	1	2	0
2	3	2	1
3	1	0	3
4			
5	2	1	
6	0	2	
7	1	0	

Enter @MMULT({A1..C3},{A5..B7})

The result is displayed as follows:

	A	B	C
8	2	5	
9	7	7	
10	5	1	

Related topics

@MNTHS - Months

Syntax

@MNTHS(StartDate, EndDate, <EndMnth>)

StartDate	Number representing the start date. See "Using dates and times in Quattro Pro."
EndDate	Number representing the end date.
EndMnth	1 to indicate adherence to ends of months; 0 to indicate that ends of months are ignored; the default is 1.

@MNTHS calculates the number of whole months between StartDate and EndDate. A whole month is the day of the month on which a specified date falls to that same day in the next month, such as March 11 to April 11.

If the day of the month on which StartDate falls does not exist in the month in which EndDate falls, and EndDate falls on the last day of that month, the number of months returned includes that month. For example, the number of months returned for March 31, 1993 to June 30, 1993 is 3 and not 2.

If StartDate falls on the last day of a month with less than 31 days, and EndDate precedes StartDate, the result depends on the value of EndMnth; for example, when evaluating the period from February 29, 1992, to January 31, 1992, if EndMnth is 1, -1 is returned; if EndMnth is 0, 0 is returned.

Examples

@MNTHS(@DATE(93,4,9),@DATE(94,9,15)) = 17, the number of whole months between April 9, 1993 and September 15, 1994.

@MNTHS(@DATE(93,4,30),@DATE(93,1,31)) = -3

Related topics

@MOD - Modulus (Remainder)

Syntax

@MOD(X, Y)

X	A numeric value.
Y	A numeric value not equal to 0.

@MOD divides the X value by Y and returns the remainder, or modulus, value. Because you cannot divide a number by zero, ERR results if the value of Y is zero.

Examples

@MOD(3,1) = 0 (3 divided by 1 leaves no remainder)

@MOD(5,2) = 1 (5 divided by 2 leaves a remainder of 1)

@MOD(3,1.1) = 0.8

@MOD(4,0) = ERR

Related topics

@MODE - Most Frequent Value

Syntax

@MODE(List)

List	One or more numeric or cell values.

@MODE returns the value in a sample or population that appears more frequently than any other value. The mode emphasizes data concentration and is best used to describe large data sets. It is commonly used to decide which resulting value is correct when the same measuring or computing process is repeated several times.

If the data set contains no duplicate data points, @MODE returns NA.

Examples

@MODE(2,2,5,7,9,9,9,10,10,11,12,18) = 9

@MODE(91,87,83,80,86,55,83,68,79,83) = 83

@MODE(1,2,3,4,5) = NA

Related topics

@MODULO - Remainder (Modulus)

Syntax

@MODULO(x, y)

x	Numeric value.
y	Numeric value, but not 0.

@MODULO returns the remainder, or modulus, of x/y. @MODULO works like @MOD, with this difference:
@MODULO uses the formula
x - y * @ROUNDDOWN(x/y)
and the sign of the result (+ or -) is always the same as the sign of y.
@MOD uses the formula
x - y * @INT(x/y)
and the sign of the result (+ or -) is always the same as the sign of x.
If x is 0, @MODULO returns 0.

Examples

@MODULO(15,6) = 3
@MOD(15,6) = 3
@MODULO(-7,3) = 2
@MOD(-7,3) = -1

Related topics

@MONTH - Month Portion of Date Serial Number

Syntax

@MONTH(DateTimeNumber)

DateTimeNumber	A numeric value between -109571 and 474816.9999999, representing a date/time serial number: -109571 = January 1, 1600; 0 = December 31, 1899; 474816 = December 31, 3199; the decimal = time (24 hr).

See "Using dates and times in Quattro Pro."

@MONTH returns the month portion of DateTimeNumber. DateTimeNumber must be a valid date/time serial number. Only the integer portion is used. The result is between 1 (January) and 12 (December).

To extract the month portion of a string that is in date format (instead of serial format), use @DATEVALUE with @MONTH to translate the date into a serial number. You can also use @DATE to enter a date value instead of a serial number.

Examples

@MONTH(69858) = 4

@MONTH(58494) = 2

@MONTH(.3773) = 12

@MONTH(@DATEVALUE("3/5/88")) = 3

@MONTH(@DATE(88,3,5)) = 3

@MOD(@MONTH(@DATEVALUE("3/5/88")),12) = 3

Related topics

@MROUND - Round to Nearest Multiple

Syntax

@MROUND(X, Y)

X	Value to round.
Y	Value to make rounded X divisible by.

@MROUND rounds X to the nearest value that is evenly divisible by Y. If Y is zero, @MROUND returns zero. If X and Y do not have the same sign, @MROUND returns ERR.

Examples

@MROUND(2.36,0.25) = 2.25

@MROUND(2.47,0.25) = 2.5

Related topics

@MTGACC - Mortgage Acceleration

Syntax

@MTGACC(Int, TtlPer, Principal, Residual, ExtraPrin, <Fper>, <Lper>, <Rper>, <Option>)

Int	Number ≥ 0 representing the periodic interest rate.
TtlPer	Total periods in the loan from start to finish, or the total periods remaining from the chosen starting period forward.
Principal	Original loan balance; also can be any starting point in the loan.
Residual	Remaining balance on loan at end of loan term; enter 0 if the loan will be paid in full.
ExtraPrin	Extra principal amount to be paid each period (must be positive).
Fper	Number of the first period, relative to the starting point, in which extra principal is paid; the default is 1 (the first period).
Lper	Number of the last period, relative to the starting point, in which extra principal is paid; the default is until the end of the loan; you can set Lper to any number greater than or equal to the last period number when extra principal payments last the life of the loan (for example, Lper can be 400 for a loan which lasts 360 periods).
Rper	Period for which the loan status is reported; the default is at loan end (any number greater than the end of the loan defaults to loan end); Rper does not affect the value @MTGACC returns if Option is 0 or 10.
Option	Specifies the output value type (the default is 0): 0 = number of periods to loan end, when balance equals Residual 1 = balance of loan at the Rper 2 = cumulative interest paid at Rper 3 = cumulative principal paid at Rper 10 = number of fewer periods in loan life, due to payment of extra principal 11 = balance reduction at Rper due to payment of extra principal 12 = reduction in cumulative interest paid at Rper due to payment of extra principal 13 = increase in cumulative principal paid at Rper due to payment of extra principal

@MTGACC calculates the effects of paying extra monthly principal for amortized loans. The value that @MTGACC returns depends on the Option you specify. For the specified Rper, you can find the loan balance, cumulative interest, or cumulative principal. You can also find how the number of periods, loan balance, and cumulative interest have been reduced, and how cumulative principal has increased, as a result of paying extra principal.

@MTGACC returns values for the end of the loan or for the end of any payment period. During calculation, @MTGACC rounds currency values to 2 decimal places, giving currency answers in whole cents, as is common with mortgage institutions.

You can use @MTGACC to return information about a loan on which you make no extra principal payments (ExtraPrin = 0); for example, @MTGACC returns exact values of the loan balance, cumulative interest paid, and cumulative principal paid.

Examples

For a mortgage with a yearly interest rate of 9%, monthly payments for 30 years, an original balance of $150,000, and a future value of $80,000, the following formulas calculate the effects of paying $300 extra principal per month starting at the beginning of the second year and continuing through the fourth year. The reporting period is the tenth year (when the home will be sold and the mortgage paid-off).

Number of periods to loan end:

@MTGACC(.09/12,30*12,150000,80000,300,2*12,4*12,10*12,0) = 357

Balance at Rper:

@MTGACC(.09/12,30*12,150000,80000,300,2*12,4*12,10*12,1) = $128,635.26

Cumulative interest paid at Rper:

@MTGACC(.09/12,30*12,150000,80000,300,2*12,4*12,10*12,2) = $125,724.06

Cumulative principal paid at Rper:

@MTGACC(.09/12,30*12,150000,80000,300,2*12,4*12,10*12,3) = $21,364.74

Number of fewer periods in loan life:

@MTGACC(.09/12,30*12,150000,80000,300,2*12,4*12,10*12,10) = 3

Reduction in balance at Rper due to ExtraPrin:

@MTGACC(.09/12,30*12,150000,80000,300,2*12,4*12,10*12,11) = $13,964.70

Reduction in cumulative interest paid at Rper due to ExtraPrin:

@MTGACC(.09/12,30*12,150000,80000,300,2*12,4*12,10*12,12) = $6,464.70

Related topics

@MULT - Cumulative Product

Syntax

@MULT(List)

List	One or more numbers or selections of numbers, separated by commas.

@MULT calculates the cumulative product of a set of numbers (List). List is a comma separated list of numbers, selections containing numbers, or both. The numbers in this list are multiplied by each other. Blank cells and cells containing strings in any of the selections passed as arguments are given a value of 1.

Example

This formula calculates the cumulative product 3.4, 5.7, -1.2, and the numbers shown in the next figure.

@MULT(A6..C10,3.4,5.7,-1.2) = 30037.32

	A	B	C
6	3.467	0.123	
7	134.23	0.034	1.238
8	87.65	6.54%	0.987
9	-2.35		79.11
10	101.93		0.005

Related topics

@MULTINOMIAL - Sum of Terms

Syntax

@MULTINOMIAL(List)

List	One or more numbers to calculate multinomial of; each number in List must be ≥ 0.

@MULTINOMIAL returns the multinomial of a list of values. It uses this formula for @MULTINOMIAL(a,b,c):

$$\frac{(a + b + c)!}{a!b!c!}$$

If any value in List is negative, @MULTINOMIAL returns ERR.
If any value in List is non-integer, it will be truncated to an integer.

Example

@MULTINOMIAL(3,4,5) = 27720

Related topics

@N - Numeric Value of Upper Left Cell

Syntax

@N(Block)

Block	A cell reference or name.

@N inspects Block and returns the numeric value of the upper left cell. If that cell contains a label or is blank, it returns a 0.

This @function is used by other spreadsheet programs to avoid unnecessary ERR values resulting from labels included in calculations. This is unnecessary with Quattro Pro, however, because labels are already considered zero values in calculations. @N is included in Quattro Pro only for compatibility with other products.

Related topics

@NA - NA Value (Not Available)

Syntax

@NA

@NA returns the special value NA (not available). Formulas that depend on a value entered as @NA return the value NA, unless there is an error, in which case they return ERR. NA is a unique number, not to be confused with the label NA.

@NA is used to indicate values not yet available (it will not work with labels). It ensures that formulas relying on information that is not provided do not display inaccurate data.

Examples

	A	B	C	D
1	QTR	North	South	West
2	1	$187,681	$151,136	$131,123
3	2	$170,072	NA	$149,181
4				
5	YTD	$357,753	NA	$280,304
6	AVG	$178,877	NA	$140,152

@NA has been entered for the South's Qtr 2 results. As you can see, the NA cascades through to the totals. When the @NA is replaced with a valid value, the totals will immediately reflect the correct figures.

@NA = NA

@IF(B3=0,@NA,B3) = NA if B3 = 0; otherwise, the value of B3

Related topics

@NBDAY - Next Business Day

Syntax

@NBDAY(Date, <Holidays>, <Saturday>, <Sunday>)

Date	Number representing a date. See "Using dates and times in Quattro Pro."
Holidays	Cells containing dates that are holidays or the date of a single holiday or 0 to indicate no holidays (the default is 0).
Saturday	0 to specify that Saturday is not a business day; 1 to specify that Saturday is a business day (the default is 0).
Sunday	0 to specify that Sunday is not a business day; 1 to specify that Sunday is a business day (the default is 0).

@NBDAY returns the serial date number of the next business day after Date.

Examples

@NBDAY(@DATE(93,2,26)) = 34029 (March 1, 1993)

@NBDAY(@DATE(93,12,24),A7..C9,0,1) = 34329 (December 26, 1993), assuming that Saturdays and the dates in cells A7..C9 are holidays.

Related topics

@NEGBINOMDIST - Negative Binomial Distribution

Syntax

@NEGBINOMDIST(Failures, Successes, Prob)

Failures	Number of failures.
Successes	Threshold of successes.
Prob	Probability of a success; $0 \leq Prob \leq 1$.

@NEGBINOMDIST returns the negative binomial distribution. Use @NEGBINOMDIST to determine the distribution of the number of failures you experience before achieving a specified number of successes.

Example

A polling organization asks a sampling of voters if they favor Candidate A for reelection. Given that 55% of the city's voters favor Candidate A, this formula calculates the probability that the polling organization will contact 10 voters who do not favor her for reelection before contacting 1 voter who does favor her:

@NEGBINOMDIST(10,1,0.55) = 0.000187

Related topics

@NENGO - Convert Date to Kanji

Syntax

@NENGO(Date)

Date	The date, comprised of the elements Imperial Year, Month, and Day.

@NENGO converts a date to its kanji representation.

Related topics

@NETPV - Net Present Value

Syntax

@NETPV(Discrate, Flows, <Initial>, <[Odd|Periods]>, <Simp>, <Pathdep>, <Filter>, <Start>, <End>)

Discrate	Discount rate or cells containing discount rates corresponding to cells of cash flows.
Flows	Cells containing cash flows.
Initial	Initial cash flow (the default is 0).
Odd|Periods	Delay between initial and first cash flow in number of periods (the default is 1) or cells containing lengths of periods between cash flows (the default is 1).
Simp	Flag specifying how to discount: 0 = compounded discounting (default) 1 = mixed compounded and simple discounting 2 = simple discounting
Pathdep	Flag specifying whether to apply path-dependent compounding to each flow; 0 = no path (default); 1 = path.
Filter	Flag specifying filter type: 0 = no filter (default); 1 = cashflow < Start; 2 = cashflow ≤ Start; 3 = cashflow > Start; 4 = cashflow ≥ Start; 5 = Start < cashflow < End; 6 = Start ≤ cashflow ≤ End.
Start	A starting cash flow amount to compare against individual flows.
End	An ending cash flow amount to compare against individual flows.

@NETPV computes the net present value of a stream of cash flows.

Example

A firm is considering the purchase of two machines. Machine A requires an initial outlay of $40,000 and produces a cash flow of $23,000 for three years. Machine B requires an outlay of $50,000 and produces a cash flow of $22,000 for four years. The following formulas calculate the net present values of both machines, using the data shown in the next figure and a discount rate of 12%. The results are useful for determining which machine is a better purchase:

Machine A: @NETPV(0.12,A13..B14,B12) = $15,242.12

Machine B: @NETPV(0.12,C13..D14,D12) = $16,821.69

	A	B	C	D
11	Machine A Income		Machine B Income	
12	Initial	($40,000)	Initial	($50,000)
13	3	$23,000	4	$22,000

The net present value of the flows associated with Machine B is greater, so it should be purchased.

Related topics

@NETWORKDAYS - Number of Working Days

Syntax

@NETWORKDAYS(StartDate, EndDate, <Holidays>, <Weekends>)

StartDate	Date number representing start date. See "Using dates and times in Quattro Pro."
EndDate	Date number representing end date.
Holidays	Optional cell name or reference containing serial date numbers of holidays to exclude from the calculation.
Weekends	Optional argument, in quotation marks, to tell @NETWORKDAYS which days are weekend days. Use 0 through 6 (Monday through Sunday); for example, "45" means Friday and Saturday. The default, if you omit Weekends, is Saturday and Sunday. To specify no weekends, use "7".

@NETWORKDAYS returns the number of days between two dates, excluding weekends and holidays. StartDate is excluded from the result; EndDate is included in the result. For example, to find the number of days, excluding weekends and holidays, between April 10, 2001 and May 10, 2001, use April 9, 2001 as the StartDate and May 10, 2001 as the EndDate (the answer is 23).

You cannot use any optional argument without using all the ones preceding it. To specify weekends but not holidays, refer to a blank cell for holidays.

Example

You want to know how many working days there will be between mid-November, 1996, and the project due date early the next January. Your company holidays are stored in a cell named Holidays, and your weekend days are Saturday and Sunday.

	A
1	35397
2	35398
3	35424
4	35425
5	35426
6	35431
7	35432
8	35433

@NETWORKDAYS(@DATE(96,11,16), @DATE(97,1,5),Holidays) = 27

Related topics

@NOMINAL - Nominal Interest Rate

Syntax

@NOMINAL(EffectRate, Nper)

EffectRate	Effective interest rate.
NperY	Number of compounding periods per year, truncated to an integer.

@NOMINAL calculates the nominal annual interest rate for a specified effective rate and number of compounding periods a year.

@NOMINAL is related to @EFFECT in the following way:

$$Rn = Nper\left((Re+1)^{\frac{1}{Nper}}-1\right)$$

where

Rn	nominal rate
Re	effective rate
Nper	number of compounding periods per year

@NOMINAL returns ERR if either argument is non-numeric, if EffectRate <= 0, or if NperY < 1.

Example

@NOMINAL(7.3756%,4) = 0.0718 or 7.18%

Related topics

@NORMDIST - Normal Distribution

Syntax

@NORMDIST(X, Mean, SDev, Cum)

X	Value at which to evaluate function.
Mean	Mean of the normal distribution.
SDev	Standard deviation of the normal distribution; must be > 0.
Cum	1 to return the cumulative normal distribution function; 0 (the default) to return the probability density function.

@NORMDIST computes the normal distribution function. A normal distribution is one that is perfectly symmetrical about its mean, and its spread is determined by the value of the standard deviation. The normal distribution describes many statistical phenomena, including the distribution of population means.

@NORMDIST uses this formula to calculate the cumulative normal distribution function:

$$\int_{-\infty}^{x} f(t)dt, f(t) = \frac{e^{-(t-\mu)/2\sigma^2}}{\sqrt{2\pi\sigma^2}}$$

To calculate the probability mass function for a normal distribution, @NORMDIST uses this formula:

$$f(t) = \frac{e^{-(t-\mu)/2\sigma^2}}{\sqrt{2\pi\sigma^2}}$$

Examples

@NORMDIST(50,48,1.2,1) = 0.95221

@NORMDIST(50,48,1.2,0) = 0.082898

Related topics

@NORMINV - Inverse of Normal Distribution

Syntax

@NORMINV(Prob, Mean, SDev)

Prob	Probability corresponding to the normal distribution; 0 < Prob < 1.
Mean	Mean of the normal distribution.
SDev	Standard deviation of the normal distribution; must be > 0.

@NORMINV returns the inverse of the cumulative normal distribution function.

Example

@NORMINV(0.95221,48,1.2) = 50

Related topics

@NORMSDIST - Standard Normal Distribution

Syntax

@NORMSDIST(X)

X	Value at which to evaluate the function.

@NORMSDIST returns the standard normal cumulative distribution function. The standard normal cumulative distribution function has a mean of 0 and a standard deviation of 1.

@NORMSDIST uses this formula:

$$\int_{-\infty}^{x} f(t)dt, f(t) = \frac{e^{-t^2/2}}{\sqrt{2\pi}}$$

Example

@NORMSDIST(1.66667) = 0.95221

Related topics

@NORMSINV - Inverse of Standard Normal Distribution

Syntax

@NORMSINV(Prob)

Prob	Probability corresponding to the normal distribution; must be > 0 and < 1.

@NORMSINV returns the inverse of the standard normal cumulative distribution function. The standard normal cumulative distribution function has a mean of 0 and a standard deviation of 1.

Example

@NORMSINV(0.95221) = 1.66667

Related topics

@NOT - Logical Not

Syntax

@NOT(List)

List	Logical value or expression that can be evaluated to TRUE or FALSE.

@NOT reverses the value of its argument. If Logical is FALSE, @NOT returns TRUE; if Logical is TRUE, @NOT returns FALSE.

Use @NOT when you need to make sure a value is not equal to a certain value.

Example

Enter the following formula in Cell B2 and copy it into Cells B3 through B6. The relative cell address will change as you copy the function, to refer to the cell next to it.

@IF(@NOT(A2=100%),"Keep trying","HOORAY!!")

	A	B
1	Grades	
2	87%	Keep trying
3	92%	Keep trying
4	75%	Keep trying
5	98%	Keep trying
6	100%	HOORAY!!

Related topics

@NOW - Current Date and Time

Syntax

@NOW

@NOW returns the serial number corresponding to the current date and time. To display the number as a date or time, right-click the cell, choose Cell Properties, then click Numeric Format.

The value generated by @NOW is updated to the current date and time each time you press the Calc key (F9), or perform any operation that recalculates the notebook.

The integer part of a date/time serial number pertains to the date; the decimal portion pertains to time. To extract just the date portion, use @INT(@NOW) or @TODAY. To extract just the time portion, use @MOD(@NOW,1).

Examples

@NOW = 31905.572338 (5/8/87, 1:45 PM)

@INT(@NOW) = 31905 (5/8/87)

@MOD(@NOW,1) = 0.572338 (1:45 PM)

@INT(@MOD(@NOW,7)) = 6 (the number of the day of the week)

Related topics

@NPER - Number of Periods

Syntax

@NPER(Rate, Pmt, Pv, <Fv>, <Type>)

Rate	A numeric value > -1, representing the periodic interest rate (the fixed interest rate per compounding period).
Pmt	A numeric value representing the amount of the periodic payment.
Pv	A numeric value representing the current value of an investment (the present value).
Fv	A numeric value representing the future value of an investment (the value the investment will reach at some point).
Type	An optional numeric value that indicates whether payments or cash flows occur at the beginning (1) or the end (0) of the period; default = 0.

@NPER calculates the number of time periods required for an investment, using an optional argument, Type, to indicate whether the investment is an ordinary annuity or an annuity due. Like @CTERM and @TERM, @NPER computes the number of payments needed to reach Fv, given Pv, Pmt, and Rate. The last two arguments of @NPER, Fv and Type, are optional. If you omit one or both of them, Quattro Pro assumes their values are zero.

Be sure to enter a negative number for money that is out of your pocket and a positive number for money that is coming in to you.

This @function is not compatible with 1-2-3. If your file must be compatible, use @CTERM or @TERM instead.

Examples

Assume you have an IRA account that earns 11.5% interest paid annually at the start of the year, and you deposit $2000 into the account at the end of each year. The present account balance is $633. To determine how many payment periods it will take to reach a nest egg of $50,000, use @NPER:

@NPER(11.5%,-2000,-633,50000,0) = 12.12

The fractional part of the answer is not very meaningful; you cannot be sure of having your nest egg until the end of the 13th year.

Related topics

@NPV - Present Value of Future Cash Flow

Syntax

@NPV(Rate, Block, <Type>)

Rate	A numeric value > -1, representing the periodic interest rate (the fixed interest rate per compounding period).
Block	Cells (reference or name) containing cash flow information for the investment.
Type	An optional numeric value that indicates whether payments or cash flows occur at the beginning (1) or the end (0) of the period; default = 0.

@NPV calculates the current value of estimated cash flow values (Block), discounted at the given interest rate (Rate). It is helpful in determining how much an investment is currently worth, based on expected earnings, although its accuracy is entirely dependent on the accuracy of the cash flow table.

The optional third argument, Type, can be 0 or 1, depending on whether the cash flows are at the beginning or the end of the period. The default value is 0, end of the period.

The formula for @NPV(Rate,Block,Type)--if Block consists of $v_1,\ldots,v_n$--is given by

If Type = 0

$$\frac{v_1}{(1+Rate)}+\ldots+\frac{v_n}{(1+Rate)^n}$$

If Type = 1

$$v_1+\frac{v_2}{(1+Rate)}+\ldots+\frac{v_n}{(1+Rate)^{n-1}}$$

The cash flow table in Block should show expected income and debits over a period of time. If Type is 0, Quattro Pro assumes that the amounts are received at the end of regular intervals. If Type is 1, it assumes the amounts are received at the beginning of regular intervals. Quattro Pro also assumes that the length of this interval is the same as the period on which interest is compounded. In other words, if monthly cash flow is estimated, Rate needs to show monthly interest. To convert annual interest to monthly interest, simply divide by 12.

Examples

	A	B	C	D	E
1	1992	1993	1994	1995	1996
2	-5000	+2000	+2000	+2000	+2000

Suppose you are considering investing $5000 this year, and you expect a return of $2000 in each of the next four years. Put the values -5000,+2000,+2000,+2000,+2000 in the cells A2..E2. The net present value, using a discount rate of 10%, is @NPV(.1,A2..E2,1) which equals $1,340. Or, combine the initial investment with the present value of the four returns with +A2+@NPV(.1,B2..E2,0). The result is the same.

	A	B	C	D
1	Jan	8000	200	3500
2	Feb	9000	350	4000
3	Mar	8500	-300	3000
4	Apr	9500	600	5000

@NPV(1.25%,B1..B4) = $33,908.92

@NPV(15%/12,C1..C4) = $820.83

@NPV(15%/12,D1..D4) = $15,006.51

-2000+@NPV(15%/12,D1..D4) = $13,006.51 (assumes an initial cash outflow of $2,000)

Related topics

@NUMTOBIN - Decimal to Binary

Syntax

@NUMTOBIN(Decimal)

Decimal	Decimal number to convert.

@NUMTOBIN returns the binary string equivalent of a decimal number. To convert a negative number, precede Decimal with a minus sign.

Examples

@NUMTOBIN(10) = 1010

@NUMTOBIN(16) = 10000

@NUMTOBIN(30) = 11110

Related topics

@NUMTOBIN64 - Decimal to Binary

Syntax

@NUMTOBIN64(Decimal, <Places>)

Decimal	Decimal number to convert.
Places	Number of characters to return; must be ≤ 64.

@NUMTOBIN64 returns the binary string equivalent of a decimal number (up to 64 bits).

Examples

@NUMTOBIN64(10) = 1010
@NUMTOBIN64(10,5) = 01010
@NUMTOBIN64(123000) = 11110000001111000
@NUMTOBIN64(123000,7) = 1111000

Related topics

@NUMTOHEX - Decimal to Hexadecimal

Syntax

@NUMTOHEX(Decimal)

Decimal	Decimal number to convert.

@NUMTOHEX converts the decimal number Decimal to its corresponding hexadecimal string value. @HEXTONUM performs the opposite conversion, from hexadecimal to decimal.

Examples

@NUMTOHEX(10) = 'A

@NUMTOHEX(16) = '10

@NUMTOHEX(65535) = 'FFFF

Related topics

@NUMTOHEX64 - Decimal to Hexadecimal

Syntax

@NUMTOHEX64(Decimal, <Places>)

Decimal	Decimal number to convert.
Places	Number of characters to return; must be ≤ 16.

@NUMTOHEX64 returns the hexadecimal string equivalent of a decimal number (up to 64 bits).

Examples

@NUMTOHEX64(10) = A
@NUMTOHEX64(10,2) = 0A
@NUMTOHEX64(123000) = 1E078
@NUMTOHEX64(1000000000) = 3B9ACA00

Related topics

@NUMTOOCT - Decimal to Octal

Syntax

@NUMTOOCT(Decimal)

Decimal	Decimal number to convert.

@NUMTOOCT returns the octal string equivalent of a decimal number. To convert a negative number, precede Decimal with a minus sign.

Examples

@NUMTOOCT(10) = 12

@NUMTOOCT(16) = 20

@NUMTOOCT(30) = 36

Related topics

@NUMTOOCT64 - Decimal to Octal

Syntax

@NUMTOOCT64(Decimal, <Places>)

Decimal	Decimal number to convert.
Places	Number of characters to return; must be ≤ 22.

@NUMTOOCT64 returns the octal string equivalent of a decimal number (up to 64 bits).

Examples

@NUMTOOCT64(8) = 10
@NUMTOOCT64(10,3) = 012
@NUMTOOCT64(123000) = 360170
@NUMTOOCT64(123000,3) = 170
@NUMTOOCT64(2^63) = 1000000000000000000000

Related topics

@NWKDAY - *N*th Weekday in Month

Syntax

@NWKDAY(N, Wkday, Month, Year, <AuxWkday>)

N	Number from 1 to 5.
Wkday	Number from 1 (Saturday) to 7 (Friday).
Month	Number from 1 (January) to 12 (December).
Year	Number from 0 (1900) to 199 (2099) or a standard year like 1993.
AuxWkday	Auxiliary day of the week that must fall in the same week as Wkday; 0 for no auxiliary day or a number from 1 (Saturday) to 7 (Friday) indicating the auxiliary day (the default is 0).

@NWKDAY returns the serial date number for the date of the Nth occurrence of Wkday in Month. If there is not an Nth occurrence, @NWKDAY returns ERR.

See "Using dates and times in Quattro Pro."

You can use AuxWkday to specify another day that must fall in the same week and month as Wkdday; see the second example.

The valid date calculation range for this function is 01/01/1900 through 12/31/2099.

Examples

@NWKDAY(2,3,4,99) = 36262 (April 12, 1999), the date of the second Monday in April 1999.

@NWKDAY(1,7,12,93,3) = 34313 (December 10, 1993), the first Friday on which both the first Friday and a Monday fall in the same week of December 1993.

Related topics

@OCTTOBIN - Octal to Binary

Syntax

@OCTTOBIN(Oct)

Oct	Octal number to convert; denote negative numbers using a minus sign.

@OCTTOBIN returns the binary string equivalent of an octal number.

Examples

@OCTTOBIN("12") = 1010

@OCTTOBIN("20") = 10000

@OCTTOBIN("36") = 11110

Related topics

@OCTTOHEX - Octal to Hexadecimal

Syntax

@OCTTOHEX(Oct)

Oct	Octal number to convert; denote negative numbers using a minus sign.

@OCTTOHEX returns the hexadecimal string equivalent of an octal number.

Examples

@OCTTOHEX("12") = A

@OCTTOHEX("20") = 10

@OCTTOHEX("36") = 1E

Related topics

@OCTTONUM - Octal to Decimal

Syntax

@OCTTONUM(Oct)

Oct	Octal number to convert; denote negative numbers using a minus sign.

@OCTTONUM returns the decimal equivalent of an octal number.

Examples

@OCTTONUM("12") = 10

@OCTTONUM("20") = 16

@OCTTONUM("36") = 30

Related topics

@ODD - Round Up to Nearest Odd Integer

Syntax

@ODD(X)

X	Value to round.

@ODD rounds X up (away from zero) to the nearest odd integer. If X is already an odd integer, @ODD returns X.

Examples

@ODD(3.2) = 5
@ODD(3) = 3
@ODD(-3.2) = -5

Related topics

@ODDFPRICE - Price of Bond with Odd First Period

Syntax

@ODDFPRICE(Settle, Maturity, Issue, FirstCpn, Coupon, Yield, <Redemption>, <Freq>, <Calendar>)

Settle	Number representing the settlement date.
Maturity	Number representing the maturity date.
Issue	Number representing the issue date.
FirstCpn	Number representing the first coupon date.
Coupon	Coupon rate; must be ≥ 0.
Yield	Annual yield; 0 < Yield ≤ 1.
Redemption	Redemption value per 100 face value (must be > 0; the default is 100).
Freq	Frequency of coupon payments in number of payments per year (can be 1, 2, 3, 4, 6, or 12; the default is 2).
Calendar	Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

@ODDFPRICE returns the price per 100 face value of a bond having an odd (short or long) first period. In an odd first coupon period, the first coupon payment is a prorated multiple of a normal coupon payment.

Dates for @ODDFPRICE must follow this pattern:

Issue < Settle < Maturity

Issue < FirstCpn < Maturity

Example

This formula returns the price per 100 face value of a bond with the following terms: Settle is March 15, 1993, Maturity is November 15, 1995, Issue is January 4, 1992, FirstCpn is May 15, 1993, Coupon is 8.5%, Yield is 8.7%, Redemption is 100, Freq 2, and Calendar is 0 (30/360).

@ODDFPRICE(@DATE(93,3,15),@DATE(95,11,15),@DATE(92,1,4),@DATE(93,5,15),0.085, 0.087,100,2,0) = 99.40933

Related topics

@ODDFYIELD - Yield of Bond with Odd First Period

Syntax

@ODDFYIELD(Settle, Maturity, Issue, FirstCpn, Coupon, Price, <Redemption>, <Freq>, <Calendar>)

Settle	Number representing the settlement date.
Maturity	Number representing the maturity date.
Issue	Number representing the issue date.
FirstCpn	Number representing the first coupon date.
Coupon	Coupon rate; must be ≥ 0.
Price	Price of the security; must be > 0.
Redemption	Redemption value per 100 face value (must be > 0; the default is 100).
Freq	Frequency of coupon payments in number of payments per year (can be 1, 2, 3, 4, 6, or 12; the default is 2).
Calendar	Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

@ODDFYIELD returns the yield of a security having an odd (short or long) first period. In an odd first coupon period, the first coupon payment is a prorated multiple of a normal coupon payment.

Dates for @ODDFYIELD must follow this pattern:

Issue < Settle < Maturity

Issue < FirstCpn < Maturity

Example

This formula calculates the yield for a security with the following terms: Settle is March 15, 1993, Maturity is November 15, 1995, Issue is January 4, 1992, FirstCpn is May 15, 1993, Coupon is 8.5%, Price is 100, Redemption is 100, Freq is 2, and Calendar is 0 (30/360).

@ODDFYIELD(@DATE(93,3,15),@DATE(95,11,15),@DATE(92,1,4),@DATE(93,5,15),0.085, 100,100,2,0) = 0.084487

Related topics

@ODDLPRICE - Price of Bond with Odd Last Period

Syntax

@ODDLPRICE(Settle, Maturity, LastCpn, Coupon, Yield, <Redemption>, <Freq>, <Calendar>)

Settle	Number representing the settlement date; must be < Maturity.
Maturity	Number representing the maturity date.
LastCpn	Number representing the last coupon date; must be < Maturity.
Coupon	Coupon rate; must be ≥ 0.
Yield	Annual yield; 0 < Yield ≤ 1.
Redemption	Redemption value per 100 face value (must be > 0; the default is 100).
Freq	Frequency of coupon payments in number of payments per year (can be 1, 2, 3, 4, 6, or 12; the default is 2).
Calendar	Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

@ODDLPRICE returns the price per 100 face value of a bond having an odd (short or long) last period. In an odd last coupon period, the last coupon payment is a prorated multiple of a normal coupon payment.

Example

This formula calculates the price per 100 face value of a bond with the following terms: Settle is June 1, 1992, Maturity is December 15, 2012, LastCpn is September 15, 2012, Coupon is 7.5%, and Yield is 5.25%.

@ODDLPRICE(@DATE(92,6,1),@DATE(112,12,15),@DATE(112,9,15), 0.075,0.0525) = 128.0663

Related topics

@ODDLYIELD - Yield of Bond with Odd Last Period

Syntax

@ODDLYIELD(Settle, Maturity, LastCpn, Coupon, Price, <Redemption>, <Freq>, <Calendar>)

Settle	Number representing the settlement date; must be < Maturity.
Maturity	Number representing the maturity date.
LastCpn	Number representing the last coupon date; must be < Maturity.
Coupon	Coupon rate; must be ≥ 0.
Price	Price; must be > 0.
Redemption	Redemption value per 100 face value (must be > 0; the default is 10 0).
Freq	Frequency of coupon payments in number of payments per year (can be 1, 2, 3, 4, 6, or 12; the default is 2).
Calendar	Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

@ODDLYIELD returns the yield of a bond having an odd (short or long) last period. In an odd last coupon period, the last coupon payment is a prorated multiple of a normal coupon payment.

Example

This formula calculates the yield of a bond with the following terms: Settle is June 1, 1992, Maturity is December 15, 2012, LastCpn is September 15, 2012, Coupon is 7.5%, and Price is 128.0663.

@ODDLYIELD(@DATE(92,6,1),@DATE(112,12,15),@DATE(112,9,15),0.075, 128.0663) = 0.0525

Related topics

@OFFSET - Offset Reference

Syntax

@OFFSET(Reference, Rows, Cols, <Height>, <Width>)

Reference	Reference on which you want to base the offset; cannot refer to non-contiguous areas.
Rows	Number of rows from Reference you want the offset to refer to. If Rows = 5, the upper left cell in the offset is five rows below the upper left cell in Reference. Negative Rows are above, positive are below.
Cols	Number of columns from Reference you want the offset to refer to. If Cols = 5, the upper left cell in the offset is five columns to the right of the upper left cell in Reference. Negative Cols are to the left, positive are to the right.
Height	Height (optional) of the returned offset, in rows. Height must be a positive number. If omitted, Height is the same height as Reference.
Width	Width (optional) of the returned offset, in columns. Width must be a positive number. If omitted, Width is the same width as Reference.

@OFFSET returns a reference that is offset from another reference by a specified number of rows and columns; dimensions of the offset can also be specified. The upper left cell of the returned reference is offset from the upper left cell of Reference by the number of Cols and Rows you specify.

@OFFSET does not actually move cells or change the selection; it just returns a reference. Use @OFFSET with any function that expects a reference argument.

If Reference is a selection and you do not specify Height = 1 and Width = 1, you must select a selection of the appropriate size to display the result before entering @OFFSET as an array.

@OFFSET returns ERR if Rows and Cols offset Reference over the edge of the notebook sheet.

Examples

The notebook EXPENSES.QPW contains the following cells:

	A	B	C
1	January	February	March
2	$652	$833	$599
3	$456	$305	$522
4	$68	$59	$73

@OFFSET(A1,2,1,1,1) = [EXPENSES.qpw]A:B3..B3

@OFFSET(A1..C1,3,0,1,3) = [EXPENSES.qpw]A:A4..C4

Related topics

@OR - Logical Or

Syntax

@OR(List)

List	True-or-false conditions to test.

@OR returns 1 (true) if any argument is true, 0 (false) only if all arguments are false.

Arguments must be logical values, or references or arrays that contain logical values.

@OR ignores text or empty cells.

You can use an @OR array formula to see if a certain value occurs in a list of cells.

Examples

Given the following data:

	A	B
1	$2	$101
2	$50	$115
3	$127	$130

@OR(A1>10,A2>10,A3>10) = 1 (true)

@OR(A1>200,A2>200,A3>200) = 0 (false)

To find which values in column A are less than 100, enter in cell A4 the formula +A1..A3<100. Quattro Pro enters the formula as an array and returns the array {1|1|0} in cells A4..A6, showing the first two values in column A are less than 100. You can do the same in cell B4 for the amounts in column B.

Suppose a $5 service charge is deducted if the daily account balance falls below the $100 minimum. Use @OR to test the true-or-false conditions in A4..A6 and B4..B6, and @IF to subtract $5 or not, depending on the results:

For account A, @IF(@OR(A4..A6), "$5","$0") = $5

For account B, @IF(@OR(B4..B6), "$5","$0") = $0 (no service charge)

Related topics

@ORB - Binary OR

Syntax

@ORB(Binary1, <Binary2>, <Bits>)

Binary1	First binary number.
Binary2	Second binary number.
Bits	Number of binary bits used for both input and output; if omitted, Bits = number of bits in Binary1 or Binary2, whichever is greater; must be in the range 0 <n ≤ 64.

@ORB performs a bit-by-bit logical OR of each bit in Binary1 and Binary2. Any bit that is set to 1 in either Binary1 or Binary2 causes the resulting output bit to be set to 1.

If only one number is specified, then @ORB performs an any-ones test, or OR reduction, on Binary1; @ORB returns 1 if any bits in Binary1 are set to 1; otherwise, it returns 0.

Examples

@ORB(10,1) = 11

@ORB(10,10) = 10

@ORB(10) = 1

@ORB(1100,1,5) = 01101

Related topics

@ORH - Hexadecimal OR

Syntax

@ORH(Hex1, <Hex2>, <Bits>)

Hex1	First hexadecimal number.
Hex2	Second hexadecimal number.
Bits	Number of binary bits used for both input and output; if omitted, Bits = number of bits in Hex1 or Hex2, whichever is greater; 4 binary digits = 1 hexadecimal digit; must be in the range 0 <n ≤ 64.

@ORH performs a bit-by-bit logical OR of each bit in Hex1 and Hex2. Any binary bit that is set to 1 in either Hex1 or Hex2 causes the resulting output bit to be set to 1.

If only one number is specified, then @ORH performs an any-ones test, or OR reduction, on Hex1; @ORH returns 1 if any bits in Hex1 are set to 1; otherwise, it returns 0.

Examples

@ORH("A","F") = F

@ORH("A") = 1

@ORH("C","1",8) = 0D

Related topics

@PAGEINDEX - Index Number for Notebook Sheet

Syntax

@PAGEINDEX(Name)

Name	A string corresponding to the name of a sheet; must be enclosed in quotation marks.

@PAGEINDEX returns the index number (from 0 to 255) for a specified sheet name. If no sheet names match the string Name, @PAGEINDEX returns ERR.

To return the index number for a sheet in another notebook, use @PAGEINDEX2.

Example

@PAGEINDEX("EXPENSES") = 0 (the index number for the sheet named EXPENSES is 0)

Related topics

@PAGEINDEX2 - Index Number for Sheet in Another Notebook

Syntax

@PAGEINDEX2(NotebookLink, Name)

NotebookLink	A reference to a sheet, cell, or cells in another notebook (for example, [BUDGET]A:A1).
Name	A string corresponding to the name of a sheet; must be enclosed in quotation marks.

@PAGEINDEX2 returns the index number (from 0 to 255) for a specified sheet name in a notebook specified by NotebookLink. If no sheet names match the string Name, @PAGEINDEX2 returns ERR.

Example

@PAGEINDEX2([BUDGET]A:A1,"EXPENSES") = 0 (the index number for the sheet named EXPENSES in notebook BUDGET is 0)

Related topics

@PAGENAME - Name of a Notebook Sheet

Syntax

@PAGENAME(Index)

Index	A number from 0 to 255 inclusive.

@PAGENAME returns the name of a sheet specified by Index. If there is not a name for the specified sheet, @PAGENAME returns ERR.

To return a sheet name from another notebook, use @PAGENAME2.

Example

@PAGENAME(3) = EXPENSES (the sheet with index number 3 is named EXPENSES)

Related topics

@PAGENAME2 - Name of Sheet in Another Notebook

Syntax

@PAGENAME2(NotebookLink, Index)

NotebookLink	A reference to a sheet, cell, or cells in another notebook (for example, [BUDGET]A:A1).
Index	A number from 0 to 255 inclusive.

@PAGENAME2 returns the name of a sheet specified by Index in a notebook specified by NotebookLink. If there is not a name for the specified sheet, @PAGENAME2 returns ERR.

Example

@PAGENAME2([BUDGET]A:A1,3) = EXPENSES (the sheet with index number 3 in notebook BUDGET is named EXPENSES)

Related topics

@PAGENAMES - Table of Sheet Names

Syntax

@PAGENAMES(<ExcludedNames>)

ExcludedNames	A optional argument specifying sheet names to exclude from the table; enclose each sheet name in quotation marks, and separate sheet names with a comma.

@PAGENAMES returns a two-column table showing the sheet letters and corresponding sheet names for the active notebook. The left column of the table contains sheet letters (from A to IV), and the right column contains corresponding sheet names.

Because @PAGENAMES returns an array, it is automatically enclosed within an @ARRAY @function.

If the active notebook does not have any named sheets, @PAGENAMES returns ERR.

Make sure there is enough room for a two-column table, with one row for each sheet name. Quattro Pro overwrites existing data in cells it uses for the table.

To return sheet names for another notebook, use @PAGENAMES2.

Example

The active notebook consists of five named sheets, Qtr1, Qtr2, Qtr3, Qtr4, and Totals, whose sheet letters are A, B, C, D, and E, respectively.

@ARRAY(@PAGENAMES) = table in A1..B5 shown in the next figure

	A	B
1	A	Qtr1
2	B	Qtr2
3	C	Qtr3
4	D	Qtr4
5	E	Totals

Related topics

@PAGENAMES2 - Table of Sheet Names in Another Notebook

Syntax

@PAGENAMES2(NotebookLink, <ExcludedNames>)

NotebookLink	A reference to a sheet, cell, or cells in another notebook (for example, [BUDGET]A:A1).
ExcludedNames	A optional argument specifying sheet names to exclude from the table; enclose each sheet name in quotation marks, and separate sheet names with a comma.

@PAGENAMES2 returns a two-column table showing the sheet letters and corresponding sheet names for the notebook specified by NotebookLink. The left column of the table contains sheet letters (from A to IV), and the right column contains corresponding sheet names.

Because @PAGENAMES2 returns an array, it is automatically enclosed within an @ARRAY @function.

If the active notebook does not have any named sheets, @PAGENAMES2 returns ERR.

Make sure there is enough room for a two-column table, with one row for each sheet name. Quattro Pro overwrites existing data in cells it uses for the table.

Example

A notebook named BUDGET consists of five named sheets, Qtr1, Qtr2, Qtr3, Qtr4, and Totals, whose sheet letters are A, B, C, D, and E, respectively.

@ARRAY(@PAGENAMES2([BUDGET]F:A1)) = table in A1..B5 shown in the next figure

	A	B
1	A	Qtr1
2	B	Qtr2
3	C	Qtr3
4	D	Qtr4
5	E	Totals

Related topics

@PBDAY - Prior Business Day

Syntax

@PBDAY(Date, <Holidays>, <Saturday>, <Sunday>)

Date	Number representing a date. See "Using dates and times in Quattro Pro."
Holidays	Cells containing dates that are holidays or the date of a single holiday or 0 to indicate no holidays (the default is 0).
Saturday	0 to specify that Saturday is not a business day; 1 to specify that Saturday is a business day (the default is 0).
Sunday	0 to specify that Sunday is not a business day; 1 to specify that Sunday is a business day (the default is 0).

@PBDAY returns the serial date number of the first business day before Date.

Examples

@PBDAY(@DATE(93,3,1)) = 34026 (February 26, 1993)

@PBDAY(@DATE(93,12,26),A7..C9,0,1) = 34325 (December 22, 1993), assuming that Saturdays and the dates in the cells A7..C9 are holidays.

Related topics

@PAYMT - Amortized Payment

Syntax

@PAYMT(Rate, Nper, Pv, <Fv>, <Type>)

Rate	A numeric value > -1, representing the periodic interest rate (the fixed interest rate per compounding period).
Nper	A numeric value > 0, representing the number of periods of the loan (the number of payments to be made) or investment (the number of compounding periods).
Pv	A numeric value representing the amount borrowed (the principal).
Fv	A numeric value representing the future value of an investment (the value the investment will reach at some point).
Type	An optional numeric value that indicates whether payments or cash flows occur at the beginning (1) or the end (0) of the period; default = 0.

@PAYMT calculates the periodic payment needed to reach Fv, given Rate, Nper, and Pv. The last two arguments of @PAYMT, Fv and Type, are optional. If you omit the last one or both of them, Quattro Pro assumes that their values are zero. Enter negative numbers for out-of-pocket money and positive numbers for money coming in.

Related @function @PMT is not as flexible, but can be used if your file must be compatible with 1-2-3.

Examples

Assume you want to take out a 30-year $175,000 mortgage with a 17.5% annual interest rate with 12 payments a year, and you would like to see the difference in your monthly payments if you paid at the start or at the end of the month. All you have to do is enter these two @functions:

@PAYMT(17.5%/12,12*30,175000,0,0) = -2566.07

@PAYMT(17.5%/12,12*30,175000,0,1) = -2529.19

If, on the other hand, your mortgage has a "balloon payment" that leaves you with unpaid principal at the end of the mortgage, you can still calculate the payment. Just insert the balloon payment amount (say, $80,000) as the future value component:

@PAYMT(17.5%/12,12*30,175000,-80000,0) = -2559.68

Related topics

@PEARSON - Correlation

Syntax

@PEARSON(Array1, Array2)

Array1	Array of independent values.
Array2	Array of dependent values.

@PEARSON returns the Pearson product moment correlation coefficient, which measures the linear association of two data sets. Array1 and Array2 must have the same number of values. @PEARSON uses this formula:

A value of r near or equal to 0 implies little or no linear relationship exists between the two lists of numbers. A value of r near or equal to 1 or -1 indicates a very strong linear relationship.

Example

This example refers to cells in the next figure.

@PEARSON(B2..B16,C2..C16) = 0.989324

	A	B	C
1	Date	Advertising	Sales
2	04/30/93	$435	$7,000
3	05/07/93	$400	$6,000
4	05/14/93	$505	$7,767
5	05/21/93	$470	$7,800
6	05/28/93	$610	$9,534
7	06/04/93	$540	$7,750
8	06/11/93	$575	$8,945
9	06/18/93	$715	$11,301
10	06/25/93	$645	$9,465
11	07/02/93	$680	$10,760
12	07/09/93	$785	$13,000
13	07/16/93	$750	$11,890
14	07/23/93	$855	$12,980
15	07/30/93	$820	$13,068
16	08/06/93	$890	$14,246

Related topics

@PERCENTILE - Percentile

Syntax

@PERCENTILE(Array, X)

Array	A numeric array or cells of values.
X	A percentile value between 0 and 1, inclusive.

@PERCENTILE returns a number from Array at the percentile indicated by X.

Examples

@PERCENTILE({4,5,7,9,10,12,13,16},0) = 4

@PERCENTILE({4,5,7,9,10,12,13,16},0.25) = 6.5

@PERCENTILE({4,5,7,9,10,12,13,16},0.50) = 9.5

@PERCENTILE({4,5,7,9,10,12,13,16},0.75) = 12.25

@PERCENTILE({4,5,7,9,10,12,13,16},1) = 16

The examples above return values from percentile increments of 0.25, which are equal to quartiles. See @QUARTILE.

Related topics

@PERCENTRANK - Percentage Rank

Syntax

@PERCENTRANK(Array, X, <Digits>)

Array	A numeric array or cells of values.
X	Number to rank in Array; if X does not match a value in Array, @PERCENTRANK interpolates to return a percentage rank.
Digits	Number of significant digits for returned percentage value; must be ≥ 1 (the default is 3).

@PERCENTRANK returns the percentage rank of X in Array. Use @PERCENTRANK to see where a value stands within a list of values based on a percentage.

Example

This example refers to cells in the next figure. This formula returns the rank of a student's score among the scores of all test takers in the cells A2..A11, where the student's score is in A4:

@PERCENTRANK(A2..A11,A4,3) = 0.222

@PERCENTRANK sample data

	A
1	Test Scores
2	78
3	80
4	85
5	85
6	86
7	87
8	91
9	92
10	95
11	98

Related topics

@PERMUT - Permutations

Syntax

@PERMUT(N, R)

N	Number of different objects; n ≥ 0.
R	Number of objects taken at a time; R ≤ N.

@PERMUT returns the total number of arrangements of objects taken R at a time from a set of N objects. The formula @PERMUT uses is:

$$PERMUT(N,R) = \frac{N!}{(N-R)!}$$

@PERMUT is similar to @COMB except that it takes into account the order that objects are selected.

Example

Given 11 different colored marbles, this formula calculates how many different ways an ordered subset of five marbles can be constructed such that no two constructions contain the same five marbles in the same order. (Different constructions can contain the same five marbles, but they cannot share the same ordering.)

@PERMUT(11,5) = 55,440

Related topics

@PI

Syntax

@PI

@PI returns the value of pi (3.141592653589794...), the classic ratio of a circle's circumference to its diameter.

To figure the area of a circle, given the radius in cell A1, enter this formula:

@PI*A1^2

Examples

@PI*13 = 40.84 (circumference of circle with a diameter of 13)

@PI*(7.5)^2 = 176.7146 (area of circle with a radius of 7.5)

@PI*B3 = the circumference of a circle whose diameter is in B3

Related topics

@PIRATE - Internal Rate of Return

Syntax

@PIRATE(Npv, Flows, <Initial>, <[Odd|Periods]>, <Simp>, <Pathdep>, <Guess>, <Precision>, <Maxiter>, <Filter>, <Start>, <End>)

Npv	Net present value.
Flows	Cells containing cash flows.
Initial	Initial cash flow (the default is 0).
Odd|Periods	Delay between initial and first cash flow, in number of periods (the default is 1) or cells containing lengths of periods between cash flows (the default is 1).
Simp	Flag specifying how to discount:
	0 = compounded discounting (default)
	1 = mixed compounded and simple discounting
	2 = simple discounting
Pathdep	Flag specifying whether to apply path-dependent compounding to each flow; 0 = no path (default); 1 = path.
Guess	IRR guess for numerical search (useful for locating multiple roots); must be > -100%; the default is 0.10.
Precision	Minimum required precision; Precision > 0; the default is 0.000001.
Maxiter	Maximum number of iterations for search; Maxiter > 0; the default is 50.
Filter	Flag specifying filter type: 0 = no filter (default); 1 = cashflow < Start; 2 = cashflow ≤ Start; 3 = cashflow > Start; 4 = cashflow ≥ Start; 5 = Start < cashflow < End; 6 = Start ≤ cashflow ≤ End.
Start	A starting cash flow amount to compare against individual flows.
End	An ending cash flow amount to compare against individual flows.

@PIRATE computes the internal rate of return for a stream of cash flows. It is similar to @IRR, but @PIRATE accommodates more complex cash flow structures.

The initial Guess for the discount rate is 10%. For some cash flow streams, particularly those with both positive and negative flows, multiple solutions are possible. By specifying a different Guess, it is possible to locate other solutions. If no solution exists, @PIRATE returns ERR.

The default value of Precision is 0.000001; smaller values need more search iterations and may require a larger value for Maxiter, which specifies the maximum number of iterations to use when attempting to find a solution. If the net present value of the cash flows does not converge within Precision to the target value specified by Npv (within Maxiter iterations), @PIRATE returns ERR.

Example

In the next figure, the stream consists of four flows, specified in cells A12..A15. The time lengths of the periods preceding each flow are specified in cells C12..C15. The Npv is $98.34. This formula calculates the internal rate of return, assuming compounded interest:

@PIRATE(B17,A12..A15,0,C12..C15) = 0.050041

	A	B	C
11	Cash Flows		Periods
12	$4.50		0.3455
13	$4.50		1.2

14	$4.50		1
15	$104.50		1.5
16			
17	npv	$98.34	

Related topics

@PMT - Amortized Payment

Syntax

@PMT(Pv, Rate, Nper)

Pv	A numeric value representing the amount borrowed (the principal).
Rate	A numeric value > -1, representing the periodic interest rate (the fixed interest rate per compounding period).
Nper	A numeric value > 0, representing the number of periods of the loan (the number of payments to be made) or investment (the number of compounding periods).

@PMT calculates the fully amortized periodic payment needed to repay a loan with a principal of Pv dollars at Rate percent per period over Nper periods. It assumes that interest is paid at the end of each period and the investment is an ordinary annuity (not an annuity due).

@PMT uses this formula:

$$\frac{P * R}{1 - (1 + R)^{-N}}$$

where

P	principal
R	periodic interest rate
N	number of periods

An equivalent for this formula using @PAYMT is

@PAYMT(Rate, Nper, - Pv, 0)

You can enter the value for Rate as a percent or a decimal; for example, 9.5% or .095. The amount you specify for Rate must correlate with the unit used for Nper. In other words, if payments are made and interest calculated annually, the amount entered for Nper must represent years. If monthly, Nper must represent the number of months the loan covers. To calculate monthly payments using an annual interest rate, divide the interest rate by 12.

@PMT assumes that the investment is an ordinary annuity. Related @functions @PAYMT, @IPAYMT, and @PPAYMT let you use an optional argument, Type, to indicate whether the investment is an ordinary annuity or an annuity due. @PMTC calculates payments based on semi-annual compounding.

Examples

To calculate a monthly payment (paid on the last day of the month) for a three-year loan of $10,000 at an annual 15% interest rate, enter

@PMT(10000,15%/12,3*12) = $346.65

You can also use @PAYMT to figure this payment (the negative result means the money is out of your pocket):

PAYMT(15%/12,3*12,10000,0,0) = $-346.65

Other examples:

@PMT(1000,0.12,5) = $277.41

@PMT(500,0.16,12) = $96.21

@PMT(5000,16%/12,12) = $453.65

@PMT(12000,0.11,15) = $1,668.78

@PMT(10000,15%/12,36) calculates a monthly payment for a three-year loan of $10,000 at an annual 15% interest rate

Related topics

@PMTC - Monthly Loan Payment

Syntax

@PMTC(Pv,Rate,Nper)

Pv	A numeric value representing the amount borrowed (the principal).
Rate	A numeric value > -1, representing the yearly interest rate (the fixed interest rate per compounding period).
Nper	A numeric value > 0, representing the number of months of the loan (the number of payments to be made).

@PMTC calculates the monthly loan payments according to Canadian mortgage conventions.

@PMTC uses this formula:

P*((R/12) / (1-(1+(R/12))^-(N*12)))

where

P	principal
R	yearly interest rate
N	number of months

Example

@PMTC(10000,15%,36) = 344.46

Related topics

@POISSON - Poisson Probability Distribution

Syntax

@POISSON(N, Mean, Cum)

N	Number of events; must be ≥ 0.
Mean	Expected numeric value for the mean over the distribution; must be > 0.
Cum	1 to return the cumulative Poisson probability distribution that the number of random events will be in the range from zero to N; 0 to return the Poisson probability mass function that the number of events will be N.

@POISSON returns the Poisson probability distribution, that is, the probability that N number of events will occur over a specified time period. The Poisson distribution function uses this formula:

$$p(r) = e^{-\mu} \frac{\mu^r}{r!}$$

with
$r \geq 0$

Example

On average, Company Z receives 30 customer service phone calls per hour. What is the probability that Company Z will receive 35 calls in one hour?

@POISSON(35,30,0) = 0.045308

Related topics

@POWER - Number Raised to a Power

Syntax

@POWER(Num, Power)

Num	Number (base) to be raised to a power; can be any real number.
Power	Power (exponent), to which Num is to be raised.

@POWER calculates the result of a specified number raised to a power.

Examples

@POWER(4,3) = 64

@POWER(3.14159,2) = 9.8695877

@POWER(2,1/2) = 1.4142136

Related topics

@PPAYMT - Principal Portion of Payment

Syntax

@PPAYMT(Rate, Per, Nper, Pv, <Fv>, <Type>)

Rate	A numeric value > -1, representing the periodic interest rate (the fixed interest rate per compounding period).
Per	The number of the loan period for which the principal is desired (where Nper is the total number of periods).
Nper	A numeric value > 0, representing the number of periods of the loan (the number of payments to be made) or investment (the number of compounding periods).
Pv	A numeric value representing the amount borrowed (the principal).
Fv	A numeric value representing the future value of an investment (the value the investment will reach at some point).
Type	An optional numeric value that indicates whether payments or cash flows occur at the beginning (1) or the end (0) of the period; default = 0.

@PPAYMT calculates the amount of a particular payment that is going toward the loan principal or investment Pv and is not interest.

@IPAYMT gives the part of the payment which is interest; @PAYMT calculates the total payment for each period.

Examples

Assume you are two years into a 30-year, 10% mortgage on a $100,000 loan. To determine what portion of this month's payment is principal, enter

@PPAYMT(.1/12,2*12,30*12,100000) = $-53.54

The negative result indicates the money is out of your pocket.

Another example:

@PPAYMT (.15/4,24,40,10000,0,1) = $-250.83 quarterly payments for a $10,000 loan at 15% annual percentage rate adjusted to a quarterly basis over a 10-year term

Related topics

@PRICE - Price of a Bond

Syntax

@PRICE(Settle, Maturity, Coupon, Yield, <Redemption>, <Freq>, <Calendar>)

Settle	Number representing the settlement date; must be < Maturity.
Maturity	Number representing the maturity date; must be > Settle.
Coupon	Coupon rate; must be ≥ 0.
Yield	Annual yield; must be > 0 and ≤ 1.
Redemption	Redemption value per 100 face value; must be > 0; the default is 100.
Freq	Frequency of coupon payments in number of payments per year (can be 1, 2, 3, 4, 6, or 12; the default is 2).
Calendar	Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

@PRICE returns the price per 100 face value of a security that pays periodic interest.

Example

An 8.85% bond that matures August 17, 2017 has a yield-to-maturity of 7.5% for September 22, 1993 settlement. This formula calculates the price of the bond:

@PRICE(@DATE(93,9,22),@DATE(117,8,17),0.0885,0.075) = 114.8902

Related topics

@PRICEDISC - Price of a Bill

Syntax

@PRICEDISC(Settle, Maturity, Discount, <Redemption>, <Calendar>)

Settle	Number representing the settlement date; must be < Maturity.
Maturity	Number representing the maturity date; must be > Settle.
Discount	Rate of discount; 0 ≤ Discount ≤ 1.
Redemption	Redemption value per 100 face value (must be > 0; the default is 100).
Calendar	Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

@PRICEDISC returns the dollar price per 100 face value of a discounted security. The dollar price of a bill is its value less the applicable dollar discount. The discount is the product of the redemption value and the quoted discount rate, prorated for the number of days between settlement and maturity. @PRICEDISC uses this formula:

$$P = R * \left(1 - D * \left(\frac{M - S}{t_b}\right)\right)$$

P	price
R	redemption
D	discount
M	maturity
S	settle
b	basis

tb is the number of days over which the discount rate applies (360 or 365).

Example

This formula calculates the dollar price of a bill with the following terms: Settle is January 17, 1993, Maturity is August 15, 1993, Discount is 8.897%, Redemption is 100, and Calendar is 2 (actual/360).

@PRICEDISC(@DATE(93,1,17),@DATE(93,8,15),0.08897,100,2) = 94.81008

Related topics

@PRICEMAT - Price of a CD

Syntax

@PRICEMAT(Settle, Maturity, Issue, Coupon, Yield, <Calendar>)

Settle	Number representing the settlement date; must be < Maturity.
Maturity	Number representing the maturity date; must be > Settle.
Issue	Number representing the issue date; must be < Settle.
Coupon	Coupon rate; 0 ≤ Coupon ≤ 1.
Yield	Annual yield; 0 ≤ Yield ≤ 1.
Calendar	Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

@PRICEMAT returns the price per 100 face value (not including accrued interest) of a security that pays interest at maturity. The invoice price of a security that pays interest at maturity is the sum of the quoted price and accrued interest between issue and settlement dates.

Example

This formula calculates the price per 100 face value of a security with the following terms: Settle is February 1, 1993, Maturity is April 1, 1993, Issue is January 2, 1993, Coupon is 10%, Yield is 10%, and Calendar is 2 (actual/360).

@PRICEMAT(@DATE(93,2,1),@DATE(93,4,1),@DATE(93,1,2),0.10,0.10,2) = 99.986

Related topics

@PROB - Probability

Syntax

@PROB(XData, ProbRange, LowerLimit, <UpperLimit>)

Xdata	Values of X associated with the probabilities.
ProbRange	Cells or an array of probability values associated with XData; each value in ProbRange must be ≥ 0 and ≤ 1; the sum of ProbRange values must equal 1.
LowerLimit	Lower limit on the value for the desired probability.
UpperLimit	Upper limit on the value for the desired probability. (optional) (default = lower limit)

@PROB determines the probability that XData values are between two limits. If XData and ProbRange do not have the same number of values, @PROB returns ERR.

Example

@PROB({10,13,15,18,25},{0.1,0.2,0.4,0.2,0.1},10,13) = 0.3

Related topics

@PROPER - Initial Capital Letters

Syntax

@PROPER(String)

String	A string value.

@PROPER converts the first letter of every word in String to uppercase, and the rest of the characters to lowercase. A word is defined as an unbroken string of alphabetic characters. Any blank spaces, punctuation symbols, or numbers mark the end of a word.

Examples

@PROPER("GEORGE washINGTON") = George Washington

@PROPER("FIRST QUARTER") = First Quarter

@PROPER("JOHN J. SMITH") = John J. Smith

@PROPER("1979's results") = 1979'S Results

@PROPER(A1) = John J. Smith (where cell A1 contains JOHN J. SMITH)

Related topics

@PROPERTY - Current Property Setting

Syntax

@PROPERTY(Object.Property)

Object	The name of the object whose property settings you are requesting.
Property	The property whose settings you are requesting.

Returns the current setting of Property for an Object. See Property Reference for lists of objects and properties you can enter as arguments.

@PROPERTY returns a string, even if the setting is a number. Object.Property must be enclosed in double quotes.

Examples

@PROPERTY("Active_Block.Selection")

Returns: the coordinates of the currently selected cells.

@PROPERTY("Sales:A1..D12.Protection")

Returns: Protect if cells A1..D12 on sheet Sales is protected; otherwise, it returns Unprotect.

Related topics

@PUREAVG - Average, Ignoring Labels and Blanks

Syntax

@PUREAVG(List)

List	One or more numeric or cell values.

@PUREAVG calculates the average of values in a list, ignoring blank cells and labels. Compare this to @AVG, which calculates the average of all values in a list.

@Functions that ignore blank cells and labels are extremely important when using a spreadsheet for statistical analysis.

Examples

	A
1	Sales
2	$80,000
3	$90,000
4	$95,000
5	$105,000

@PUREAVG(A1..A5) = $92,500

@AVG(A1..A5) = $74,000, because @AVG includes the column heading and divides the total by 5

@AVG(A2..A5) = $92,500, when only the 4 cells containing numbers are in the list

Related topics

@PURECOUNT - Counts Non-Blank Cells, Ignoring Labels

Syntax

@PURECOUNT(List)

List	One or more numeric or string values, cell addresses, and cell references or names, separated by commas.

@PURECOUNT returns the number of entries and cells in a list, excluding blank cells and labels. Compare this to @COUNT, which returns the number of nonblank cells in a list. @COUNT includes cells with entries of any kind, including labels, a label-prefix character, or the values ERR and NA.

You can use both @PURECOUNT and @COUNT to divert or stop a macro that performs a task on a series of selections when the cell pointer reaches an empty cell.

@Functions that ignore blank cells and labels are extremely important when using a spreadsheet for statistical analysis.

Examples

	A
1	Sales
2	$80,000
3	$90,000
4	$95,000
5	$105,000

@PURECOUNT(A1..A6) = 4

@COUNT(A1..A6) = 5 (includes the label)

@PURECOUNT(1,"hello",A1..A3) = 3

@COUNT(1,"hello",A1..A3) = 5

Related topics

@PUREMAX - Maximum Value, Ignoring Labels and Blanks

Syntax

@PUREMAX(List)

List	One or more numeric or string values, cell addresses, and cell references or names, separated by commas.

@PUREMAX returns the largest numeric value in a list, ignoring blank cells and labels. Compare this to @MAX, which returns the largest numeric or data value in a list.

@Functions that ignore blank cells and labels are extremely important when using a spreadsheet for statistical analysis.

Examples

	A
1	Expenses
2	($80)
3	($90)
4	($95)
5	($105)

@PUREMAX(A1..A6) = ($80)

@MAX(A1..A6) = 0, the value of the label, Expenses

Related topics

@PUREMIN - Minimum Value, Ignoring Labels and Blanks

Syntax

@PUREMIN(List)

List	One or more numeric or string values, cell addresses, and cell references or names, separated by commas.

@PUREMIN returns the smallest numeric value in a list, ignoring blank cells and labels. Compare this to @MIN, which returns the smallest numeric or data value in a list.

@Functions that ignore blank cells and labels are extremely important when using a spreadsheet for statistical analysis.

Examples

	A
1	Sales
2	$80,000
3	$90,000
4	$95,000
5	$105,000

@PUREMIN(A1..A6) = $80,000

@MIN(A1..A6) = 0, the value of the label, Expenses

Related topics

@PURESTD - Population Standard Deviation, Ignoring Labels and Blanks

Syntax

@PURESTD(List)

List	One or more numeric or string values, cell addresses, and cell references or names, separated by commas.

@PURESTD returns the population standard deviation (square root of the population variance) of numeric values in a list, ignoring blank cells and labels. Compare this to @STD, which returns the population standard deviation of all values in a list. @PURESTDS and @STDS return sample standard deviation.

@STD and @PURESTD use the *n* method to calculate standard deviation of population data. This method assumes that the sample reflects the entire population. If the sample is small, the standard deviation is biased because of sampling errors, so @STDS or @PURESTDS should be used instead.

@Functions that ignore blank cells and labels are extremely important when using a spreadsheet for statistical analysis.

Examples

	A
1	Grades
2	4.0
3	3.4
4	3.7
5	3.6

@PURESTD(A1..A6) = 0.216506, ignoring the label and the blank

@STD(A1..A6) = 1.4827, because the argument includes the label, which to @STD = 0

@STD(A2..A6) = 0.216506, excluding the label from the argument

@PURESTDS(A1..A6) = 0.25, the sample standard deviation

Related topics

@PURESTDS - Sample Standard Deviation, Ignoring Labels and Blanks

Syntax

@PURESTDS(List)

List	One or more numeric or string values, cell addresses, and cell references or names, separated by commas.

@PURESTDS returns the sample standard deviation (square root of the sample variance) of numeric values in a list, ignoring blank cells and labels. Compare this to @STDS, which returns the sample standard deviation of all values in a list. @PURESTD and @STD return population standard deviation.

@STDS and @PURESTDS use the *n-1* method to calculate standard deviation of sample population data. This method compensates for sampling errors, returning a slightly larger standard deviation. If the sample is large enough, @STD or @PURESTD can be used.

Examples

	A
1	Grades
2	4.0
3	3.4
4	3.7
5	3.6

@PURESTDS(A1..A6) = 0.25, ignoring the label and the blank

@STDS(A1..A6) = 1.657709, because the argument includes the label, which to @STDS = 0

@STDS(A2..A6) = 0.25, excluding the label from the argument

@PURESTD(A1..A6) = 0.216506, the population standard deviation

Related topics

@PUREVAR - Population Variance, Ignoring Labels and Blanks

Syntax

@PUREVAR(List)

List	One or more numeric or string values, cell addresses, and cell references or names, separated by commas.

@PUREVAR calculates the population variance of numeric values in a list, ignoring blank cells and labels. Compare this to @VAR, which calculates the population variance of all values in a list. @PUREVARS and @VARS calculate sample population variance.

Variance @functions perform a statistical test called analysis of variance (anova). @VAR and @PUREVAR use the *n* method to calculate variance. This method assumes that the sample reflects the entire population. If the sample is small, the variance is biased because of sampling errors, so @VARS and @PUREVARS should be used instead.

@Functions that ignore blank cells and labels are extremely important when using a spreadsheet for statistical analysis.

Examples

	A
1	Grades
2	4.0
3	3.4
4	3.7
5	3.6

@PUREVAR(A1..A6) = 0.046875, ignoring the label and the blank

@VAR(A1..A6) = 2.1984, because the argument includes the label, which to @VAR = 0

@VAR(A2..A6) = 0.046875, excluding the label from the argument

@PUREVARS(A1..A6) = 0.0625, the sample population variance

Related topics

@PUREVARS - Sample Population Variance, Ignoring Labels and Blanks

Syntax

@PUREVARS(List)

List	One or more numeric or string values, cell addresses, and cell references or names, separated by commas.

@PUREVARS calculates the sample population variance of numeric values in a list, ignoring blank cells and labels. Compare this to @VARS, which calculates the sample population variance of all values in a list. @PUREVAR and @VAR calculate population variance.

Variance @functions perform a statistical test called analysis of variance (anova). @VARS and @PUREVARS use the *n-1* method to calculate variance. This method compensates for sampling errors, returning a slightly larger variance. If the sample is large enough, @VAR or @PUREVAR can be used.

@Functions that ignore blank cells and labels are extremely important when using a spreadsheet for statistical analysis.

Examples

	A
1	Grades
2	4.0
3	3.4
4	3.7
5	3.6

@PUREVARS(A1..A6) = 0.0625, ignoring the label and the blank

@VARS(A1..A6) = 2.748, because the argument includes the label, which to @VARS = 0

@VARS(A2..A6) = 0.0625, excluding the label from the argument

@PUREVAR(A1..A6) = 0.046875, the population variance

Related topics

@PV - Present Value

Syntax

@PV(Pmt, Rate, Nper)

Pmt	A numeric value representing the amount of the periodic payment.
Rate	A numeric value > -1, representing the periodic interest rate (the fixed interest rate per compounding period).
Nper	A numeric value > 0, representing the number of periods of the loan (the number of payments to be made) or investment (the number of compounding periods).

@PV calculates the present value of an investment where Pmt is received for Nper periods and is discounted at the rate of Rate per period. Present value is calculated using this formula:

$$P\frac{1-(1+R)^{-N}}{R}$$

where

P	amount of periodic payment
R	periodic interest rate
N	number of periods

An equivalent for this formula using @PVAL is

@PVAL(Rate, Nper, - Pmt, 0)

@PV assumes that the investment is an ordinary annuity. Related @function @PVAL lets you use an optional argument, Type, to indicate whether the investment is an ordinary annuity or an annuity due.

Examples

Assume you want to buy a new van that costs $12,000. The dealer presents two offers: Pay $12,000 cash up front, or pay $350 per month for the next five years with 7% interest. The present value of the loan is

@PV(350,7%/12,5*12) = $17,675.70

The loan is worth over $5000 more than paying the cost all at once.

You can also use @PVAL. The car loan example becomes

@PVAL(7%/12,5*12,-350,0,0) = $17,675.70

Other examples:

@PV(277,0.12,5) = $998.52

@PV(600,0.17,10) = $2,795.16

@PV(100,0.11,12) = $649.24

Related topics

@PVAL - Present Value

Syntax

@PVAL(Rate, Nper, Pmt, <Fv>, <Type>)

Rate	A numeric value > -1, representing the periodic interest rate (the fixed interest rate per compounding period).
Nper	A numeric value > 0, representing the number of periods of the loan (the number of payments to be made) or investment (the number of compounding periods).
Pmt	A numeric value representing the amount of the periodic payment.
Fv	A numeric value representing the future value of an investment (the value the investment will reach at some point).
Type	An optional numeric value that indicates whether payments or cash flows occur at the beginning (1) or the end (0) of the period; default = 0.

@PVAL calculates the present value of an investment where Pmt is received for Nper periods and is discounted at the rate of Rate per period.

Enter negative numbers for money that is out of your pocket and positive numbers for money coming in to you. The last two arguments, Fv and Type, are optional. If you omit the last one or both of them, Quattro Pro assumes their values are zero.

This @function is not compatible with 1-2-3. If your file must be compatible, use the related @function @PV instead.

Examples

Your grandfather leaves you $24,000 in cash over the next 12 years ($2000 a year) or you can have all his government bonds, which mature in 15 years to a worth of $30,000. To determine which is worth more, compute the present value of the $24,000. Assume you can invest the money as you accumulate it in a 10% money market account.

@PVAL(10%,12,2000,0,0) = -13,627.38

The result is negative because the money you invest is considered an outgoing cash flow. Now compare this figure with the present value of the $30,000, which you will not receive for 15 years:

@PVAL(10%,15,0,30000,0) = -7,181.76

These results tell you that the $24,000 spread over 12 years is the more valuable choice.

Related topics

@S - String Value of Upper Left Cell

Syntax

@S(Block)

Block	A cell reference or name.

@S returns the string value of the upper left cell of Block. If that cell contains a numeric or date value or is blank, it returns "" (an empty string).

Quattro Pro transforms cells prefixed with an exclamation point (as used in 1-2-3) to a one-cell range (!C3 changes to C3).

Examples

	A	B	C	D
1	COMPANY	REP	SALES	COMMISSION
2	ABC Inc.	Jones	$123,630	$3,115
3	Rogers Co.	Marcus	$160,330	$4,040
4	Klein Sales	Wong	$145,330	$3,662

@S(A1..A6) = COMPANY

@S(A2..A2) = ABC Inc.

@S(C2..C4) = (blank)

@S(B1..B1)&" = "&@S(B2..B2) = REP = Jones

@S(B3)&@S(C3) = Marcus

Related topics

@SCMARG - Discount Scenario Margin

Syntax

@SCMARG(Npv, Discrate, Flows, <Initial>, <[Odd|Periods]>, <Simp>, <Pathdep>, <Guess>, <Precision>, <Maxiter>, <Filter>, <Start>, <End>)

Npv	Net present value.
Discrate	Discount rate or cells containing discount rates corresponding to cells of cash flows.
Flows	Cells containing cash flows.
Initial	Initial cash flow (the default is 0).
Odd|Periods	Delay between initial and first cash flow, in number of periods (the default is 1) or cells containing lengths of periods between cash flows (the default is 1).
Simp	Flag specifying how to discount:
	0 = compounded discounting (default)
	1 = mixed compounded and simple discounting
	2 = simple discounting
Pathdep	Flag specifying whether to apply path-dependent compounding to each flow; 0 = no path (default); 1 = path.
Guess	Initial margin for numerical search (useful for locating multiple roots) (must be > -100%; the default is 0).
Precision	Minimum required precision (must be > 0; the default is 0.000001).
Maxiter	Maximum number of iterations for search (must be > 0; the default is 50).
Filter	Flag specifying filter type: 0 = no filter (default); 1 = cashflow < Start; 2 = cashflow ≤ Start; 3 = cashflow > Start; 4 = cashflow ≥ Start; 5 = Start < cashflow < End; 6 = Start ≤ cashflow ≤ End.
Start	A starting cash flow amount to compare against individual flows.
End	An ending cash flow amount to compare against individual flows.

@SCMARG computes the discount scenario margin, a value that must be added to the Discrate series so that the net present value of the cash flow series equals Npv. An initial guess is made for the margin and is refined until the difference between the computed net present value and Npv is less than Precision. If @SCMARG does not find a solution within Maxiter search iterations, it returns ERR.

If Discrate is a number, @SCMARG returns the difference between the internal rate of return and Discrate. If Discrate is a selection of discount rates, the margin is the amount to add to each discount rate to make the net present value equal Npv.

Example

A stream of cash flows is indexed off the current London InterBank Offer Rate (LIBOR). The time distances between flows is specified in cells C11..C15. The first flow is 1 period away, the second is 1.5 periods after that, and so on. If the current LIBOR is 8.5%, this formula calculates the margin to LIBOR if the net present value is $167,000 and compounded discounting is used:

@SCMARG(167000,0.085,A11..A15,0,C11..C15) = 0.02599

	A	B	C
10	Cash Flows		Time Periods
11	25000		1

12	35000	1.5
13	23000	1.2
14	50000	1
15	134500	1.3

Related topics

@SEC - Secant

Syntax

@SEC(X)

X	An angle measured in radians. X can be any value from approximately -9.00719E+15 through 9.00719E+15.

@SEC returns the secant of angle X, in radians. X must be specified in radians, not degrees. To convert degrees to radians, use @RADIANS.

In a right triangle, the secant of an acute angle is the ratio hypotenuse **:** side adjacent. Secant is the reciprocal of cosine.

Examples

@SEC(@RADIANS(60)) = 2

@SEC(@RADIANS(75)) = 3.863703

@SEC(@RADIANS(45)) = 1.414214

@SEC(@PI/3) = 2

Related topics

@SECOND - Second Portion of Date Serial Number

Syntax

@SECOND(DateTimeNumber)

DateTimeNumber	A numeric value between -109571 and 474816.9999999, representing a date/time serial number: -109571 = January 1, 1600; 0 = December 31, 1899; 474816 = December 31, 3199; the decimal = time (24 hr).

See "Using dates and times in Quattro Pro."

@SECOND returns the second portion of DateTimeNumber. DateTimeNumber must be a valid date/serial number. Because only the decimal portion of a serial number pertains to time, the integer portion of the number is disregarded. The result is between 0 and 59.

To extract the second portion of a string that is in time format (instead of serial format), use @TIMEVALUE with @SECOND to translate the time into a serial number. You can also use @TIME to enter a time value instead of a serial number.

Examples

@SECOND(.3655445) = 23

@SECOND(.2543222) = 13

@SECOND(35) = 0

@SECOND(@TIME(3,15,22)) = 22

@SECOND(@TIMEVALUE("10:08:45 am")) = 45

@SECOND(@TIMEVALUE("10:08 am")) = 0

Related topics

@SECH - Hyperbolic Secant

Syntax

@SECH(X)

X	A value from approximately -708.39599 to approximately 708.39599.

@SECH calculates the hyperbolic secant of X.

The hyperbolic secant is the reciprocal of the hyperbolic cosine. X must be specified in radians, not degrees. To convert degrees to radians, use @RADIANS.

@SECH returns a value greater than 0 or less than or equal to 1.

Examples

@SECH(@RADIANS(60)) = 0.624888

@SECH(@RADIANS(75)) = 0.503455

@SECH(@RADIANS(45)) = 0.75494

@SECH(@PI/3) = 0.624888

Related topics

@SEMEAN - Standard Error of Sample Mean

Syntax

@SEMEAN(Block)

Block	Cell reference or name.

@SEMEAN returns the standard error of the sample mean for values in specified cells.

Examples

@SEMEAN({5,3,7,8}) = 1.108678

Given cells H1.. H4 containing the values 4.0, 3.4, 3.7, and 3.6,

@SEMEAN(H1..H4) = 0.125

Related topics

@SERIESSUM - Sum of a Power Series

Syntax

@SERIESSUM(X, N, M, Coefficients)

X	Value in the power series.
N	Initial power to raise X to.
M	Increment of the power N for each successive term in the power series.
Coefficients	Cells or array of one or more numeric values by which each power of X is multiplied; the number of values in Coefficients sets the number of terms in the power series.

@SERIESSUM returns the sum of a power series. It uses this formula for @SERIESSUM(*X*,*N*,*M*,*A*):

$$A_1X^N+A_2X^{(N+M)}+A_3X^{(N+2M)}+\ldots+A_iX^{(N+(i-1)M)}$$

Example

If the cells A1..A3 contain the values 1, 2, and 3, then @SERIESSUM(0.5,1,2,A1..A3) = 0.84375

Related topics

@SETSTRING - Label of Given Length

Syntax

@SETSTRING(Text, Length, <Alignment>)

Text	Label text, in quotation marks.
Length	Integer from 1 through 1022 specifying label length.
Alignment	Optional argument specifying text alignment:
	0 = align text left; default if you omit the argument
	1 = center text
	2 = align text right

@SETSTRING returns a label as long as the number of characters you specify. The label consists of the specified text plus enough blank spaces to total Length, aligning the text as you specify in Alignment.

- If Length is smaller than the number of characters in Text, @SETSTRING still returns the entire text.
- With proportionally spaced fonts, blank spaces are narrower than most letters.

Examples

Each dot represents a blank space in these examples; the dots are not displayed by Quattro Pro.

@SETSTRING("Cost Estimate",18) = Cost Estimate·····

@SETSTRING("Cost Estimate",18,1) = ···Cost Estimate··

@SETSTRING("Cost Estimate",18,2) = ·····Cost Estimate

Related topics

@SHEETS - Number of Notebook Sheets

Syntax

@SHEETS(Block)

Block	A cell reference or name.

@SHEETS returns the number of sheets within the given cells. This is most often used with 3-D selections. @SHEETS always returns 1 for 2-D selections. This @function is similar to @COLS and @ROWS.

Examples

@SHEETS(B:A1..D:IV1) = 3

@SHEETS(A1..C7) = 1

@SHEETS(A..E:NAME) = 5

Related topics

@SHLB - Binary Shift Left

Syntax

@SHLB(Binary, <ShiftBits>, <BitIn>, <Bits>)

Binary	Binary number.
ShiftBits	Number of bits to shift; 0 ≤ ShiftBits ≤ 64; the default is 1.
BitIn	Binary bit inserted during the shift (can be 0, 1, "S" or "I"; "S" = same as the least significant bit before shifting; "I" = inverse of the least significant bit before shifting; the default is 0).
Bits	Number of binary bits used for both input and output; if omitted, Bits = number of bits in Binary; must be in the range 0 <n ≤ 64.

@SHLB shifts the specified binary number left by ShiftBits bits. @SHLB inserts the BitIn bit at the least significant bit (LSB).

Examples

@SHLB(1000,1,0,5) = 10000

@SHLB(1000,1,1,6) = 010001

@SHLB(1000,2,1,6) = 100011

Related topics

@SHLBO - Overflow of Binary Shift Left

Syntax

@SHLBO(Binary, <Bits>)

Binary	Binary number.
Bits	Number of input binary digits used during the shift operation; if omitted, Bits = the number of bits in Binary; must be in the range 0 <n ≤ 64.

@SHLBO returns the overflow bit (either 0 or 1) of the specified binary number after it has been shifted left by one bit.

An overflow occurs when a bit is shifted out of the word size specified by Bits. For example, if Binary = 1000 and Bits = 4, shifting the number left one bit results in 0000 with an overflow of 1 bit shifted to the fifth place not shown.

Examples

@SHLBO(1000) = 1

@SHLBO(100,4) = 0

@SHLBO(1100,4) = 1

Related topics

@SHLH - Hexadecimal Shift Left

Syntax

@SHLH(Hex, <ShiftBits>, <BitIn>, <Bits>)

Hex	Hexadecimal number.
ShiftBits	Number of bits to shift; 0 ≤ ShiftBits ≤ 64; the default is 1.
BitIn	Binary bit inserted during the shift (can be 0, 1, "S" or "I"; "S" = same as the least significant bit before shifting; "I" = inverse of the least significant bit before shifting; the default is 0).
Bits	Number of binary bits used for both input and output; if omitted, Bits = number of bits in Hex; 4 binary digits = 1 hexadecimal digit; must be in the range 0 <n ≤ 64.

@SHLH shifts the specified hexadecimal number left by ShiftBits bits. @SHLH inserts the BitIn bit at the least significant bit (LSB).

Examples

@SHLH("41",1) = 82

@SHLH("41",1,0,12) = 082

@SHLH("C",1,1,12) = 019

Related topics

@SHLHO - Overflow of Hexadecimal Shift Left

Syntax

@SHLHO(Hex, <Bits>)

Hex	Hexadecimal number.
Bits	Number of equivalent input binary digits used during the shift operation; if omitted, Bits = the number of bits in Hex; 4 binary digits = 1 hexadecimal digit; must be in the range 0 <n ≤ 64.

@SHLHO returns the overflow bit (either 0 or 1) of the specified hexadecimal number after it has been shifted left by one bit.

An overflow occurs when a bit is shifted out of the word size specified by Bits. For example, if the binary equivalent of Hex = 1000 and Bits = 4, shifting the number left one bit results in 0000 with an overflow of 1 bit shifted to the fifth place not shown.

Examples

@SHLHO("A") = 1

@SHLHO("A",5) = 0

@SHLHO("C",4) = 1

Related topics

@SHRB - Binary Shift Right

Syntax

@SHRB(Binary, <ShiftBits>, <BitIn>, <Bits>)

Binary	Binary number.
ShiftBits	Number of bits to shift; 0 ≤ ShiftBits ≤ 64; the default is 1.
BitIn	Binary bit inserted during the shift (can be 0, 1, "S" or "I"; "S" = same as the most significant bit before shifting; "I" = inverse of the most significant bit before shifting; the default is "S").
Bits	Number of binary bits used for both input and output; if omitted, Bits = number of bits in Binary; must be in the range 0 <n ≤ 64.

@SHRB returns the result of shifting the specified binary number right by ShiftBits bits. @SHRB inserts the BitIn bit at the most significant bit (MSB).

Examples

@SHRB(1000,1) = 1100

@SHRB(1000,1,1,5) = 10100

@SHRB(1100,1,0,6) = 000110

Related topics

@SHRBO - Overflow of Binary Shift Right

Syntax

@SHRBO(Binary, <Bits>)

Binary	Binary number.
Bits	Number of input binary digits used during the shift operation; if omitted, Bits = the number of bits in Binary; must be in the range 0 <n ≤ 64.

@SHRBO returns the overflow bit (either 0 or 1) of the specified binary number after it has been shifted right by one bit.

An overflow occurs when a bit is shifted out of the word size specified by Bits. For example, if Binary = 1001 and Bits = 4, shifting the number right one bit results in 0100 with an overflow of 1 bit shifted off the right side.

Examples

@SHRBO(1001) = 1

@SHRBO(10010,4) = 0

@SHRBO(1100,4) = 0

Related topics

@SHRH - Hexadecimal Shift Right

Syntax

@SHRH(Hex, <ShiftBits>, <BitIn>, <Bits>)

Hex	Hexadecimal number.
ShiftBits	Number of bits to shift; 0 ≤ ShiftBits ≤ 64; the default is 1.
BitIn	Binary bit inserted during the shift (can be 0, 1, "S" or "I"; "S" = same as the most significant bit before shifting; "I" = inverse of the most significant bit before shifting; the default is "S").
Bits	Number of binary bits used for both input and output; if omitted, Bits = number of bits in Hex; 4 binary digits = 1 hexadecimal digit; must be in the range 0 <n ≤ 64.

@SHRH returns the result of shifting the specified hexadecimal number right by ShiftBits bits. @SHRH inserts the BitIn bit at the most significant bit (MSB).

Examples

@SHRH("41",1,1) = A0

@SHRH("41",1,1,4) = 8

@SHRH("C",1,0,12) = 006

Related topics

@SHRHO - Overflow of Hexadecimal Shift Right

Syntax

@SHRHO(Hex, <Bits>)

Hex	Hexadecimal number.
Bits	Number of equivalent input binary digits used during the shift operation; if omitted, Bits = the number of bits in Hex; 4 binary digits = 1 hexadecimal digit; must be in the range 0 <n ≤ 64.

@SHRHO returns the overflow bit (either 0 or 1) of the specified hexadecimal number after it has been shifted right by one bit.

An overflow occurs when a bit is shifted out of the word size specified by Bits. For example, if the binary equivalent of Hex = 1001 and Bits = 4, shifting the number right one bit results in 0100 with an overflow of 1 bit shifted off the right side.

Examples

@SHRHO("A") = 0

@SHRHO("B",5) = 1

@SHRHO("C",4) = 0

Related topics

@SIGN - Tests for Sign

Syntax

@SIGN(X)

X	A numeric value.

@SIGN returns 1 if X is positive, 0 if X is zero, and -1 if X is negative.

Examples

@SIGN(2*4) = 1

@SIGN(0*4) = 0

@SIGN(-2*4) = -1

Related topics

@SIN - Sine

Syntax

@SIN(X)

X	A numeric value.

@SIN returns the sine of the angle X. X must be given in radians, not degrees. To convert degrees to radians, use @RADIANS.

Examples

@SIN(@RADIANS(30)) = 0.5

@SIN(@RADIANS(75)) = 0.965926

@SIN(@RADIANS(45)) = 0.707107

@SIN(@PI/6) = 0.5

Related topics

@SINH - Hyperbolic Sine

Syntax

@SINH(X)

X	A value from approximately -710.47558 to approximately 710.47558.

@SINH returns the hyperbolic sine of X, in radians. X must be specified in radians, not degrees. To convert degrees to radians, use @RADIANS.

Examples

@SINH(@RADIANS(30)) = 0.547853

@SINH(@RADIANS(75)) = 1.716184

@SINH(@RADIANS(45)) = 0.868671

@SINH(@PI/6) = 0.547853

Related topics

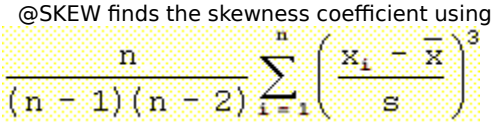
@SKEW - Skewness of a Distribution

Syntax

@SKEW(List)

List	One or more numeric or cell values.

@SKEW returns the skewness of a distribution. Skewness characterizes the degree of asymmetry of a distribution around its mean value. Use @SKEW when you want a non-dimensional quantity to measure the "shape" of a distribution rather than its moment, which is a measure in the same units as the elements of the distribution. @SKEW finds the skewness coefficient using this formula:



$$\frac{n}{(n-1)(n-2)}\sum_{i=1}^{n}\left(\frac{x_i-\bar{x}}{s}\right)^3$$

A positive result means that the distribution is skewed to the right (the median is less than the mean). A negative result means that the distribution is skewed to the left (the median is greater than the mean). @SKEW returns 0 when the distribution is symmetrical around its mean.

If there are less than three data points in List, or if the standard deviation is zero, @SKEW returns ERR.

Example

@SKEW(4,5,8,5,7,12,6,9,2,5) = 0.685055

Related topics

@SLOPE - Slope of the Linear Regression

Syntax

@SLOPE(KnownY, KnownX)

KnownY	Dependent range of values.
KnownX	Independent range of values.

@SLOPE returns the slope of the linear regression line through data points in x and y. The slope (the rate of change along the regression line) is the distance between the y values of two points, divided by the distance between their respective x values.

@SLOPE uses this formula:

$$b = \frac{n\sum xy - \left(\sum x\right)\left(\sum y\right)}{n\sum x^2 - \left(\sum x\right)^2}$$

Example

@SLOPE({1,2,3,4,5,6},{4,8,12,16,20,24}) = 0.25

Related topics

@SLN - Straight-Line Depreciation

Syntax

@SLN(Cost, Salvage, Life)

Cost	A numeric value representing the amount paid for an asset.
Salvage	A numeric value representing the value of an asset at the end of its useful life.
Life	A numeric value representing the expected useful life of an asset (in years).

@SLN calculates the straight-line depreciation allowance for an asset over one period of its life. It uses this formula:

(Cost - Salvage) divided by Life

To compute accelerated depreciation with the sum-of-the-years'-digits method (allowing higher depreciation values in the first years of the asset's life), use @SYD. To calculate depreciation using the double-declining balance method, use @DDB.

Examples

Assume you just bought a new $4000 computer. The dealer says you can sell it back to the store for $350 after eight years, but that no one would want to buy it after that. In other words, the Salvage value of that computer is $350 and its Life is 8. To determine the depreciation allowance of the computer for each year of its life, enter this formula:

@SLN(4000,350,8) = 456.25

Other examples:

@SLN(15000,3000,10) = $1,200

@SLN(5000,500,5) = $900

@SLN(1800,0,3) = $600

Related topics

@SMALLEST - *N*th Smallest number

Syntax

@SMALLEST(Array, N)

Array	A numeric array or cells of values.
N	Number that indicates the rank in size from the data set Array; must be greater than 0 and less than or equal to the number of values in Array.

@SMALLEST finds the Nth smallest number in Array. Use @SMALLEST to determine a value's rank in a data set from the bottom of that set.

If there are duplicates in Array, @SMALLEST treats them as separate numbers.

Example

@SMALLEST({3,45,8,4,7,6,13,2,87,23,58,14,17,21},5) = 7

Related topics

@SPLINE - Piecewise Polynomial Fit

Syntax

@SPLINE(KnownX's, KnownY's, OutputBlock)

KnownX's	Independent selection or array of values.
KnownY's	Dependent selection or array of values.
OutputBlock	The cell address where the result of the @function is to be displayed..

@SPLINE returns a polynomial fitted piecewise to pass through a specified set of points.

- KnownX's and KnownY's must contain the same number of values.
- Values in KnownX's must be unique: The same x-value cannot have more than one y-value. If an x-value has more than one y-value, @SPLINE returns NA.

@SPLINE takes the 1-dimenisional arrays of KnownX's and KnownY's and produces a set of coefficients in the OutputBlock. The coefficients produced are those for a linear spline.

Note that OutputBlock must be large enough to accomodate the number of interpolation coefficients to be returned, otherwise a value of ERR will be returned. There will be (k-1) coefficients where k is the number of data points in the series.

A spline is an elastic ruler used by engineers and shipbuilders that bends to pass through a specified set of points.

Example

Given the following data,

	A	B	C	D	E
1					
2			1	2	3
3			4	2	5
4					

@SPLINE(C2..E2,C3..E3,C4..D4) = 1

As well, cells C4 and D4 will be populated with the values -2 and 3 respectively.

Related topics

@SQRT - Square Root

Syntax

@SQRT(X)

X	A numeric value equal to or greater than 0.

@SQRT returns the square root of X. If X is a negative value, the result of @SQRT is ERR. If X is a string or reference to a cell containing a label, the @function returns 0.

Examples

@SQRT(9) = 3

@SQRT(2) = 1.414213562

@SQRT(144) = 12

@SQRT(@SQRT(16)) = 2

@SQRT(-4) = ERR

Related topics

@SQRTPI - Square Root of pi*X

Syntax

@SQRTPI(X)

X	Value ≥ 0 to multiply by pi.

@SQRTPI returns the square root of (@PI * X). If X is negative, @SQRTPI returns ERR.

Example

@SQRTPI(2) = 2.506628

Related topics

@STANDARDIZE - Normalize Values from a Distribution

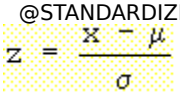
Syntax

@STANDARDIZE(X, Mean, SDev)

X	Number to normalize.
Mean	Arithmetic mean of a distribution.
SDev	Standard deviation of a distribution.

@STANDARDIZE normalizes the values from a distribution. A standard normal cumulative distribution assumes a mean of 0 and a standard deviation of 1. Use @STANDARDIZE to normalize values for use with other statistical @functions that require normally distributed variables (such as @ZTEST).

@STANDARDIZE uses this formula:



Example

@STANDARDIZE(2.6,1.6,0.5) = 2

Related topics

@STD - Population Standard Deviation

Syntax

@STD(List)

List	One or more numeric values, cell addresses, and cell references or names, separated by commas.

@STD returns the population standard deviation (the square root of the population variance) of all values in List. @STDS computes the standard deviation of sample data.

List can be any combination of single cell references, cell selections, and numeric values. When more than one component is used, all components must be separated by commas. @STD treats any labels within a cell selection as zero and ignores any blank cells. If the List contains only blank cells, however, @STD returns ERR.

@STD determines how much individual values in List differ from the average (mean) of all values in List. It can be used to verify the reliability of the average; the lower the value returned by @STD, the less individual values vary from the average.

Examples

	A	B	C	D
1		January	February	March
2	John	$652	$833	$599
3	Mary	$456	$305	$522
4	Ralph	$68	$59	$73
5	Anna	$80	$80	$80

@STD(B4..D4) = $5.79

@STD(C2..C5,260) = $279.97

@STD(B2..D5) = $270.20

@STD(A15..D20) = ERR (because the entire selection is blank)

@STDS(B4..D4) = $7.09

@STDS(B2..D5) = $282.22

Related topics

@STDS - Sample Standard Deviation

Syntax

@STDS(List)

List	One or more numeric values, cell addresses, and cell references or names, separated by commas.

@STDS returns the sample standard deviation (the square root of the sample variance) of all values in List. @STD computes population standard deviation.

Examples

	A	B	C	D
1		January	February	March
2	John	$652	$833	$599
3	Mary	$456	$305	$522
4	Ralph	$68	$59	$73
5	Anna	$80	$80	$80

@STD(B4..D4) = $5.79

@STD(C2..C5,260) = $279.97

@STD(B2..D5) = $270.20

@STD(A15..D20) = ERR (because the entire selection is blank)

@STDS(B4..D4) = $7.09

@STDS(B2..D5) = $282.22

Related topics

@STEC - Standard Error of Regression Coefficients

Syntax

@STEC(KnownY, KnownX)

KnownY	Dependent range of 3 or more values.
KnownX	Independent range of 3 or more values.

@STEC computes the standard error of the regression coefficient.

@STEC is equal to StdError divided by (Std(x) times square root of n)

$$\text{@STEC(KnownY, KnownX)} = \frac{\text{@STEYX(KnownY, KnownX)}}{\text{@STD(KnownX)} * \text{@SQRT(n)}}$$

@STEC returns ERR if KnownY and KnownX do not have the same number of values, or if KnownY and KnownX have less than 3 values each.

Example

This formula calculates the standard error of the regression coefficient for range y (3,7,4,5) in cells A1..A4 and range x (3.4,5.3,6,8) in cells B1..B4:

@STEC(A1..A4,B1..B4) = 0.593769

Related topics

@STEYX - Standard Error of Linear Regression

Syntax

@STEYX(KnownY, KnownX)

KnownY	Dependent range of 3 or more values.
KnownX	Independent range of 3 or more values.

@STEYX returns the standard error of a linear regression. The standard error is the deviation of the observed y values from the linear combinations. @STEYX uses this formula:

$$S_{yx} = \sqrt{\left[\frac{1}{n(n-2)}\right]\left[n\sum y^2 - \left(\sum y\right)^2 - \frac{\left[n\sum xy - \left(\sum x\right)\left(\sum y\right)\right]^2}{n\sum x^2 - \left(\sum x\right)^2}\right]}$$

If KnownY or KnownX have a different number of data points, or if the variance of KnownX = 0, @STEYX returns ERR.

Example

@STEYX({4,6,7,9,8},{7,9,12,9,5}) = 2.215697

Related topics

@STKOPT - Stock Options

Syntax

@STKOPT(OptCode, OptPrem, UndStkValue, Date, Load, CmdString)

OptCode	Option code string with expiration month, strike-price, and put or call symbol enclosed in quotation marks (for example, "MAY 22.5 C"); the strike price can be a decimal or fractional number, but not both (for example, it can be 11/32 or 1.625, but not 2 3/8); valid month codes are JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, and DEC; you can add spaces between month, strike-price, and Put or Call symbol; the total string (not including quotation marks) must be 20 characters or less.
OptPrem	Option premium or price
UndStkValue	Value or price of the underlying stock.
Date	Serial date number between 2 (January 1, 1900) and 73050 (December 31, 2099) representing the date on which to evaluate the stock option.
Load	Load or commission involved in sale or purchase.
CmdString	Command string enclosed in quotation marks specifying the operations to perform; cannot exceed 20 characters (not including quotes); see the table below for a list of valid CmdString characters.

@STKOPT calculates values useful in evaluating stock options. It takes as input the basic information for the option, such as the option code (expiration month, strike price, and call or put indicator), the premium, the underlying value, the date at which the option is evaluated, and the associated "load" or fee. It then calculates a value such as the expiration date of the option, the number of calendar days remaining until the option expires, or the intrinsic value of the option.

If the expiration month specified in OptCode is earlier in the year than the month specified by Date, the expiration date of the option occurs in the following year.

Load should be in the same units as OptPrem. You can use the Load argument as a general purpose value to incorporate any factor in the calculated result. For example, if you prefer a different annualization, such as 360/D, set Load to 360 and replace "A" in the CmdString formula with "L/D".

@STKOPT calculates output values according to a formula that you specify in the CmdString argument. The formula may include the following characters.

@STKOPT is valid only for options whose duration is 1 year or less

Character	**Description**
A	Annualization ratio (365.25/D); can be used to adjust percentages to equivalent yearly percentages
D	Days from DateTimeNum until expiration of option specified
E	Expiration date code of option specified
I	Intrinsic value of option
L	Load or fee to purchase or sell the option; can also be a general-purpose numeric constant
P	Premium or market value of the option
S	Strike or exercise price of the option
T	Time value of the option
U	Underlying value or price of the stock
+	Addition operator

-	Subtraction operator
*	Multiplication operator
/	Division operator

@STKOPT evaluates the command string formula from left to right with no operator precedence or associativity. The command string can be upper- or lowercase. The next table shows some examples of command strings.

String	**Result**
"D"	Outputs the number of days until option expiration
"I"	Outputs the intrinsic value
"L/P"	Outputs the ratio of the Load divided by the Premium
"T/U*A"	Time Value / Underlying value * Annualization ratio
"P-L/U*A"	((Premium - Load) / Underlying value) * Annualization ratio

For a call, the intrinsic value is the underlying value minus the strike price or zero, whichever is greater. For a put, the intrinsic value is the strike price minus the underlying value or zero, whichever is greater. The time value is equal to the premium minus the intrinsic value. The load is an expense for either the buyer or the seller. A positive time value is considered to be an expense for the buyer and income for the seller of the option.

Examples

A November put-option on stock of company XYZ has a strike price of $65 and a premium of $2.625, with an average load of $.12 per share. XYZ's stock is currently trading at $67 per share, and the date of evaluation is June 8, 1993.

Expiration date:

@STKOPT("NOV65P",2.625,67,@DATE(93,6,8),.12,"E") = 34293 (November 20, 1993)

Number of days to expiration:

@STKOPT("NOV65P",2.625,67,@DATE(93,6,8),.12,"D") = 165

Time value/stock price:

@STKOPT("NOV65P",2.625,67,@DATE(93,6,8),.12,"T/U") = 0.039179

Percentage load:

@STKOPT("NOV65P",2.625,67,@DATE(93,6,8),.12,"L/P") = 0.045714

Related topics

@STRCMPNORM - Compare Half/Full-Width Normalized Strings

Syntax

@STRCMPNORM(String1, String2)

String1	The first string to be compared
String2	The second string to be compared

@STRCMPNORM compares two strings. @STRCMPNORM also lets you compare single-byte and double-byte character strings used in software localized to most Far Eastern languages (for example, Japanese, Chinese, and Korean). The localized machine will display a toolbar that lets you select various single and double-byte character sets from within Quattro Pro.

The result returned by @STRCMPNORM is:

- 1 for equal (indicates both the strings are the same)
- 0 for not equal (indicates that the strings are different)
- -1 for an error (indicates a problem with the function or with the string or cells selected)

Example

If you have strings in cells A1 and B1, type @STRCMPNORM(A1, B1) in cell C1.

Related topics

@STRING - Convert Number to String

Syntax

@STRING(X, DecPlaces)

X	A numeric value, a formula that evaluates to a numeric value, or a reference to a cell containing a numeric value.
DecPlaces	A numeric value from 0 through 15.

@STRING converts X to a string, rounding X to the decimal precision indicated by DecPlaces.

Once a number or date has been converted to a label using @STRING, no display formatting can be done with it. To format strings derived from numbers as anything other than General format, you must build a macro that uses the {CONTENTS} keyword.

Examples

@STRING(3.59,0) = 4

@STRING(98.6,2) = 98.60

@STRING(0.3902,0) = 0

@STRING("Harry",0) = 0

@STRING(A1,2) = 10.00, if A1 contains the value 10

Related topics

@SUBB - Subtract Binary Numbers

Syntax

@SUBB(Binary1, Binary2, <BitIn>, <Bits>)

Binary1	First binary number.
Binary2	Second binary number.
BitIn	Input borrow bit (either 0 or 1); the default is 0.
Bits	Number of binary bits used for both input and output; if omitted, Bits = number of bits in Binary1 or Binary2, whichever is greater; must be in the range 0 <n ≤ 64.

@SUBB returns the difference of two binary numbers.

Use two's complement notation (see Quattro Pro glossary) to represent negative numbers. If BitIn is 1, @SUBB subtracts one extra bit from the result.

Examples

@SUBB(100,100,0,1) = 0

@SUBB(1000,100,1,3) = 011

@SUBB(1100,1,1,5) = 01010

Related topics

@SUBBO - Overflow of Binary Subtraction

Syntax

@SUBBO(Binary1, Binary2, <BitIn>, <Bits>)

Binary1	First binary number.
Binary2	Second binary number.
BitIn	Input borrow bit (either 0 or 1); the default is 0.
Bits	Number of input binary digits used for subtraction; if omitted, Bits = the number of bits in Binary1 or Binary2, whichever is greater; must be in the range 0 <n ≤ 64.

@SUBBO returns the overflow bit (either 0 or 1) of the difference of two binary numbers. An overflow occurs when a bit is borrowed from outside the word size specified by Bits. For example, if Binary1 = 10 and Binary2 = 110, the difference of the two numbers is 100 with 1 borrow bit in the fourth place not shown.

Use two's complement notation (see Quattro Pro glossary) to represent negative numbers. If BitIn is 1, @SUBBO subtracts one extra bit from the difference of the two numbers before returning the overflow.

Examples

@SUBBO(1000,1111) = 1

@SUBBO(1000,111,1,5) = 0

@SUBBO(1100,1101,1,4) = 1

Related topics

@SUBH - Subtract Hexadecimal Numbers

Syntax

@SUBH(Hex1, Hex2, <BitIn>, <Bits>)

Hex1	First hexadecimal number.
Hex2	Second hexadecimal number.
BitIn	Input borrow bit (either 0 or 1); the default is 0.
Bits	Number of binary bits used for both input and output; if omitted, Bits = number of bits in Hex1 or Hex2, whichever is greater; 4 binary digits = 1 hexadecimal digit; must be in the range 0 <n ≤ 64.

@SUBH returns the difference of two hexadecimal numbers.

Use two's complement notation (see Quattro Pro glossary) to represent negative numbers. If BitIn is 1, @SUBH subtracts one extra bit from the result.

Examples

@SUBH("1000","100",1) = 0EFF

@SUBH("1000","100",0) = 0F00

@SUBH("C","1",1,8) = 0A

Related topics

@SUBHO - Overflow of Hexadecimal Subtraction

Syntax

@SUBHO(Hex1, Hex2, <BitIn>, <Bits>)

Hex1	First hexadecimal number.
Hex2	Second hexadecimal number.
BitIn	Input borrow bit (either 0 or 1); the default is 0.
Bits	Number of binary bits used for both input and output; if omitted, Bits = number of bits in Hex1 or Hex2, whichever is greater; 4 binary digits = 1 hexadecimal digit; must be in the range 0 <n ≤ 64.

@SUBHO returns the overflow bit (either 0 or 1) of the difference of two hexadecimal numbers. An overflow occurs when a bit is borrowed from outside the word size specified by Bits. For example, if the binary equivalent for Hex1 and Hex2 are 10 and 110, respectively, the difference of the two numbers is 100 with 1 borrow bit in the fourth place not shown.

Use two's complement notation (see Quattro Pro glossary) to represent negative numbers. If BitIn is 1, @SUBHO subtracts one extra bit from the difference of the two numbers before returning the overflow.

Examples

@SUBHO(8,"F") = 1

@SUBHO(8,7,1,5) = 0

@SUBHO("C","D",1,4) = 1

Related topics

@SUBSTITUTE - Substitutes Text

Syntax

@SUBSTITUTE(Text, OldText, NewText, <InstanceNum>)

Text	Text or reference to single cell containing OldText.
OldText	Text to be changed.
NewText	Text to substitute for OldText.
InstanceNum	Which occurrence of OldText to change. If you specify InstanceNum, @SUBSTITUTE changes only that instance. Otherwise, @SUBSTITUTE changes all occurrences.

@SUBSTITUTE returns a copy of a specified text string, substituting new text for old text. Use @SUBSTITUTE to replace specific text in a text string; to replace any text in a specific location in a text string, use @REPLACE.

Example

To copy "11/1/96" but change it to "11/8/96", enter @SUBSTITUTE("11/1/96","1","8",3)

Related topics

@SUBTOTAL - Subtotal

Syntax

@SUBTOTAL(FunctionNum, Ref)

FunctionNum	Number from 1 to 11, specifying which function to use in calculating subtotals.
Ref	List of selections or cell names to subtotal.

@SUBTOTAL returns a subtotal in a list or database. You can create a list using @SUBTOTAL, then modify it by editing the @SUBTOTAL formula.

- @SUBTOTAL ignores any nested subtotals within Ref.
- @SUBTOTAL ignores any hidden rows in filtered lists, so you can subtotal only visible data.

FunctionNum	Quattro Pro Function
1	PUREAVG
2	PURECOUNT
3	COUNT
4	PUREMAX
5	PUREMIN
6	MULT
7	PURESTDS
8	PURESTD
9	SUM
10	PUREVARS
11	PUREVAR

Examples

In the following, Cells A3 and C3 contain @SUBTOTAL(9,A1..A2) and @SUBTOTAL(9,C1..C2), respectively. Cell A4 contains the formula @TOTAL(A1..C3) and sums all the values in the cells except those generated by @SUBTOTAL or @SUBTOTAL123. To omit possible subtotals in Column B, you could also write @SUM(A1..A2,B1..B2)

	A	B	C	D
1	$30		$18	
2	$65		$22	
3	$95		$40	@SUBTOTAL in A3 and C3, FunctionNum=9
4	$135			@TOTAL in A4

In the next selection, rows 9 through 12 contain @SUBTOTAL functions, illustrating the way they are ignored in subsequent @SUBTOTAL calculations.

	A	B
5	28	37
6	31	35
7	29	36
8	32	40
9	120	148
10	268	

11	33.5
12	4.174754

A9 and B9 contain @SUBTOTAL functions summing rows 5 through 8 of their respective columns.

A10 contains @SUBTOTAL(9,A5..B9) = 268, the sum of the 8 values in A5.. B8, ignoring the @SUBTOTAL functions in row 9.

A11 contains @SUBTOTAL(1, A5..B10) = 33.5, the average of the 8 values in A5.. B8, ignoring all other @SUBTOTAL functions in the referenced cells.

A12 contains @SUBTOTAL(7, A5..B11) = 4.174754, the sample standard deviation of the 8 values in A5.. B8, ignoring all other @SUBTOTAL functions in the cells.

Related topics

@SUBTOTAL123 - Subtotal

Syntax

@SUBTOTAL123(List)

List	One or more numeric values, cell addresses, and cell references or names, separated by commas.

@SUBTOTAL123 adds the values in a list or cell reference. Use @SUBTOTAL123 to indicate which cells @GRANDTOTAL123 should sum.

Example

In the following, Cells A3 and C3 contain @SUBTOTAL123(A1..A2) and @SUBTOTAL123(C1..C2), respectively. Cell A4 contains the formula @GRANDTOTAL123(A1..C3) and sums only the subtotals in the cells. To omit possible subtotals in Column B, you could also write: @SUM(A1..A2,B1..B2).

	A	B	C	D
1	$30		$18	
2	$65		$22	
3	$95		$40	@SUBTOTAL123 in A3 and C3
4	$135			@GRANDTOTAL123 in A4

Related topics

@SUM - Sum

Syntax

@SUM(List)

List	One or more numeric values, cell addresses, and cell references or names, separated by commas.

@SUM returns the total of all numeric values in List. List can be any combination of single cell references, cell selections, and numeric values. When more than one component is used, they must be separated by commas. Any labels or blank cells within a cell selection are ignored by @SUM.

Any dates in the cells will be converted to serial numbers and included in the calculation. Since this will throw off your sum, avoid including dates in the @SUM argument cells.

If you use a mouse, the QuickSum button  on the Main Toolbar offers a convenient way to total columns, rows, or both. It can total rows and columns in the selected cells, but you do not need to enter a formula.

Examples

	A	B	C	D
1		January	February	March
2	John	$652	$833	$599
3	Mary	$456	$305	$522
4	Ralph	$68	$59	$73
5	Anna	$80	$80	$80

@SUM(B4..D4) = $200

@SUM(C2..C5,260) = $1,537

@SUM(A5,534) = $534

@SUM(B2..B5,D2..D5) = $2,530

@SUM(B2..D5) = $3,807

Related topics

@SUMIF - Sum Matching Cells

Syntax

@SUMIF(Block, Criteria, <Sum Range>)

Block	Overall range of one or more cell addresses, a cell reference, or name.
Criteria	Numeric or string values that determine whether a cell within the Block is added.
<Sum Range>	Cell addresses within the Block to be included in the sum. Cell values must meet Criteria in order to be included in the sum. (optional)

@SUMIF adds those cells in Block that meet Criteria. An optional Sum Range may be specified to limit Criteria consideration and sum inclusion to particular cells within the Block.

Examples

	A	B	C	D
1	Customer	QTY	Price	Total
2	Adams Electric	5	$1	$5
3	Frys Service	2	$1	$2
4	Major Hardware	4	$1	$4
5	Adams Electric	5	$1	$5
6	Adams Electric	5	$1	$5

@SUMIF(A2..D6,"Adams Electric",D2..D6) = $15

The second argument (criteria) must be on the left-hand side of the third argument (sum range). As well, the columns containing these arguments must be directly beside each other. If these conditions are not met the function will return ERR.

Related topics

@SUMNEGATIVE - Adds Negative Values Only

Syntax

@SUMNEGATIVE(List)

List	One or more numeric values or formulas, cell addresses, and cell references or names, separated by commas.

@SUMNEGATIVE sums only negative values in cells or list. It ignores blank cells and labels.

Examples

	A
1	Profit/ Loss
2	($300)
3	$2,500
4	($70)
5	$500

@SUMNEGATIVE(A1..A6) = ($370), ignoring the label and the empty cell

@SUMNEGATIVE(-300,2500,-70,500) = ($370)

Related topics

@SUMPOSITIVE - Adds Positive Values Only

Syntax

@SUMPOSITIVE(List)

List	One or more numeric values or formulas, cell addresses, and cell references or names, separated by commas.

@SUMPOSITIVE sums only positive values in cells or a list. It ignores blank cells and labels.

Examples

	A
1	Profit/ Loss
2	($300)
3	$2,500
4	($70)
5	$500

@SUMPOSITIVE(A1..A6) = $3,000, ignoring the label and the empty cell

@SUMPOSITIVE(-300,2500,-70,500) = $3,000

Related topics

@SUMPRODUCT - Sum of Products of Corresponding Coordinates

Syntax

@SUMPRODUCT(Block1, Block2)

Block1	A cell reference or name.
Block2	A cell reference or name.

@SUMPRODUCT(Block1, Block2) returns the dot product of the vectors corresponding to the selections. Quattro Pro multiplies each corresponding cell from Block1 and Block2 and then totals those results. The selections must be the same size (same number of rows and same number of columns), or else the selections must both be one-dimensional (either a row or a column) and they must have the same length. If the selections do not match, @SUMPRODUCT returns ERR.

This @function is not compatible with 1-2-3. If your notebook must be compatible with 1-2-3, do not use @SUMPRODUCT.

Examples

Assume the following values for these cells:

A1 = 1, B1 = 5, A2 = 2, B2 = 6, A3 = 3, B3 = 7, A4 = 4, B4 = 8

@SUMPRODUCT(A1..A2,B1..B2) = 17 (because 1*5 + 2*6 = 5+12 = 17)

@SUMPRODUCT(A1..A4,B1..B4) = 70

@SUMPRODUCT(A1..A4,B1..B5) = ERR (selections are not the same size)

Related topics

@SUMSQ - Sum of the Squares

Syntax

@SUMSQ(List)

List	One or more numeric or cell values.

@SUMSQ returns the sum of the squares of the values in List.

Example

@SUMSQ(2,3) = 2^2 + 3
^{2} = 13

Related topics

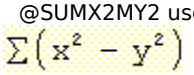
@SUMX2MY2 - Sum of the Difference of the Squares

Syntax

@SUMX2MY2(Array1, Array2)

Array1	First array of numeric values.
Array2	Second array of numeric values.

@SUMX2MY2 returns the sum of the difference of the squares of the corresponding values in Array1 and Array2.
@SUMX2MY2 uses this formula:



If Array1 and Array2 have a different number of values, @SUMX2MY2 ignores extra values in the larger array.

Example

@SUMX2MY2({3,4,5},{2,3,4}) = (9 - 4) + (16 - 9) + (25 - 16) = 21

Related topics

@SUMX2PY2 - Sum of the Sum of Squares

Syntax

@SUMX2PY2(Array1, Array2)

Array1	First array of numeric values.
Array2	Second array of numeric values.

@SUMX2PY2 returns the sum of the sum of squares of the corresponding values in Array1 and Array2.
@SUMX2PY2 uses this formula:

If Array1 and Array2 have a different number of values, @SUMX2PY2 ignores extra values in the larger array.

Example

@SUMX2PY2({3,4,5},{2,3,4}) = (9 + 4) + (16 + 9) + (25 + 16) = 79

Related topics

@SUMXMY2 - Sum of the Squares of the Differences

Syntax

@SUMXMY2(Array1, Array2)

Array1	First array of numeric values.
Array2	Second array of numeric values.

@SUMXMY2 returns the sum of the squares of the differences of corresponding values in Array1 and Array2.
@SUMXMY2 uses this formula:

$$\sum(x - y)^2$$

If Array1 and Array2 have a different number of values, @SUMXMY2 ignores extra values in the larger array.

Example

@SUMXMY2({3,4,5},{2,3,4}) = (3 -2)^2 + (4 - 3)^2 + (5 - 4)^2 = 3

Related topics

@SUMXPY2 - Sum of the Squares of the Sums

Syntax

@SUMXPY2(Array1, Array2)

Array1	First array of numeric values.
Array2	Second array of numeric values.

@SUMXPY2 returns the sum of the squares of the sums of the corresponding values in Array1 and Array2.

@SUMXPY2 uses this formula:

$$\sum(x + y)^2$$

If Array1 and Array2 have a different number of values, @SUMXPY2 ignores extra values in the larger array.

Example

@SUMXPY2({3,4,5},{2,3,4}) = $(3 + 2)^2$ + $(4 + 3)^2$ + $(5 + 4)^2$ = 155

Related topics

@SUMXY - Sum of the Products

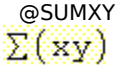
Syntax

@SUMXY(Array1, Array2)

Array1	First array of numeric values.
Array2	Second array of numeric values.

@SUMXY returns the sum of the products of the corresponding numbers in Array1 and Array2.

@SUMXY uses this formula:



If Array1 and Array2 have a different number of values, @SUMXY ignores extra values in the larger array.

Example

@SUMXY({3,4,5},{2,3,4}) = (3 * 2) + (4 * 3) + (5 * 4) = 38

Related topics

@SUMXY2 - Sum of the Squares of the Products

Syntax

@SUMXY2(Array1, Array2)

Array1	First array of numeric values.
Array2	Second array of numeric values.

@SUMXY2 returns the sum of the squares of the products of the corresponding values in Array1 and Array2.
@SUMXY2 uses this formula:

$$\sum(xy)^2$$

If Array1 and Array2 have a different number of values, @SUMXY2 ignores extra values in the larger array.

Example

@SUMXY2({3,4,5},{2,3,4}) = $(3 * 2)^2$ + $(4 * 3)^2$ + $(5 * 4)^2$ = 580

Related topics

@SUUJI - Convert Number to Kanji

Syntax

@SUUJI(Number, Style)

Number	The number
Style	1 Long form
	2 Long form: accounting
	3 Short form

@SUUJI converts an arabic number into its kanji representation. The Style parameter lets you specify how you want the representation to be displayed.

Related topics

@SYD - Sum-of-the-years'-digits Depreciation

Syntax

@SYD(Cost, Salvage, Life, Period)

Cost	A numeric value representing the amount paid for an asset.
Salvage	A numeric value representing the value of an asset at the end of its useful life.
Life	A numeric value representing the expected useful life of an asset (in years).
Period	A numeric value representing the time period for which you want to calculate depreciation.

@SYD calculates depreciation amounts for an asset using an accelerated depreciation method. This allows higher depreciation in the earlier years of the asset's life. @SYD uses this formula to compute depreciation:

((Cost - Salvage)(Life - Period + 1)) divided by (Life(Life + 1)/2)

Cost must be equal to or greater than Salvage; both must be equal to or greater than 0. Life must be equal to or greater than Period; both must be equal to or greater than 1.

@DDB and @SLN offer other methods of calculating depreciation.

Examples

Assume you just bought a new $4000 computer. The dealer says you can sell it back to the store for $350 after eight years, but that no one would want to buy it after that. The Salvage value of that computer is $350 and its Life is 8. To see what the depreciation allowance of this computer will be by the second year (using this method of depreciation), enter this formula:

@SYD(4000,350,8,2) = 709.72

These examples show depreciation values for the first five years of an asset's life. These can be compared to those calculated with @DDB,FUNCTION_DDB which distributes more of the depreciation in the first year of life.

@SYD(12000,1000,5,1) = $3,667

@SYD(12000,1000,5,2) = $2,933

@SYD(12000,1000,5,3) = $2,200

@SYD(12000,1000,5,4) = $1,467

@SYD(12000,1000,5,5) = $733

@DDB(12000,1000,5,1) = $4,800

@DDB(12000,1000,5,2) = $2,880

@DDB(12000,1000,5,3) = $1,728

@DDB(12000,1000,5,4) = $1,037

@DDB(12000,1000,5,5) = $555

Related topics

@QUARTILE - Quartile

Syntax

@QUARTILE(Array, X)

Array	A numeric array or cells of values.
X	Number signifying what quartile value to return: 0 = minimum value in Array 1 = 25th percentile 2 = 50th percentile (median) 3 = 75th percentile 4 = maximum value in Array

@QUARTILE returns a number from Array at the quartile indicated by X. You create quartiles of a data set by partitioning the values into four groups containing an equal number of values.

If the quartile falls between two discrete values in the list, a fractional value is determined using linear interpolation.

Examples

@QUARTILE({4,5,7,9,10,12,13,16},0) = 4

@QUARTILE({4,5,7,9,10,12,13,16},1) = 6.5

@QUARTILE({4,5,7,9,10,12,13,16},2) = 9.5

@QUARTILE({4,5,7,9,10,12,13,16},3) = 12.25

@QUARTILE({4,5,7,9,10,12,13,16},4) = 16

Related topics

@QUOTIENT - Integer Portion of Quotient

Syntax

@QUOTIENT(X, Y)

X	Value to divide.
Y	Nonzero value to divide x by.

@QUOTIENT is similar to @INT; it returns the integer portion of X/Y. If Y is zero, @QUOTIENT returns ERR.

Example

@QUOTIENT(7,2) = 3

Related topics

@RADIANS - Convert Degrees to Radians

Syntax

@RADIANS(X)

X	A numeric value representing degrees. Choose a numeric value in the range between -1.698E+308 to 1.085E+308 (approximately).

@RADIANS converts the given number of degrees to radians. It uses this formula:

pi times X divided by 180

One degree is equal to approximately 0.017 radians.

Examples

@RADIANS(1) = 0.017453

@RADIANS(57) = 0.994838

@RADIANS(@DEGREES(3.5)) = 3.5

@RADIANS(A4) = 0.994838 (where cell A4 contains the value 57)

Related topics

@RAND - Fractional Random Number

Syntax

@RAND

@RAND returns a fractional random number between 0 and 1. This offers a sampling of figures, useful for generating sample data for simulated situations.

To generate random numbers in another range, multiply @RAND by the difference between the new high and low ends, then add the new low end number. The formula is @RAND * (high number - low number) + low number.

For example, to indicate a range of 10 to 100, enter @RAND*90+10. This extends the upper limit to 100 and the lower limit to 10.

@RAND generates a new random number with each recalculation.

Examples

@RAND = a random number between 0 and 1

@RAND*9+1 = a random number between 1 and 10

@RAND*1000 = a random number between 0 and 1000

@RAND+5 = a random number between 5 and 6

-@INT(@RAND*90+10) = a random integer between -10 and -100

Related topics

@RANDBETWEEN - Random Number Between *N* and *M*

Syntax

@RANDBETWEEN(N, M)

N	Integer value that random number must be greater than or equal to.
M	Integer value that random number must be less than or equal to.

@RANDBETWEEN returns a random number between N and M using a uniform distribution. @RANDBETWEEN returns a new random number each time you recalculate a notebook.

Related topics

@RANK - Rank of Number in List

Syntax

@RANK(Number, Array, Order)

Number	A number from Array.
Array	One or more numeric or cell values.
Order	Flag indicating how to sort the list of numbers: any nonzero value = ascending order; 0 = descending order.

@RANK returns the rank of Number in Array in either ascending or descending order.

Examples

Theses examples refer to the next figure.

@RANK(B6,B2..B10,0) = 1

@RANK(B2,B2..B10,1) = 3

@RANK(B9,B2..B10,1) = 1

	A	B
1		
2	HJ	117
3	MW	122
4	KM	125
5	WC	109
6	AD	137
7	MS	119
8	DP	125
9	MS	106
10	DM	121

Related topics

@RATE - Interest Rate per Period

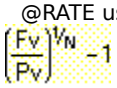
Syntax

@RATE(Fv, Pv, Nper)

Fv	A numeric value representing the future value of an investment (the value the investment will reach at some point).
Pv	A numeric value representing the current value of an investment (the present value).
Nper	A numeric value > 0, representing the number of periods of the loan (the number of payments to be made) or investment (the number of compounding periods).

@RATE calculates the interest rate required for an investment of Pv to be worth Fv within Nper compounding periods. If Nper represents years, an annual interest rate results; if Nper represents months, a monthly interest rate results, and so on.

@RATE uses this formula to calculate interest rate:



where

Fv	future value
Pv	present value
N	number of periods

An equivalent for this formula using @IRATE is

@IRATE(Nper, 0, - Pv, Fv)

@RATE assumes the investment is an ordinary annuity. The related @function @IRATE lets you use an optional argument, Type, to indicate whether the investment is an ordinary annuity or an annuity due.

Examples

This formula determines what yearly interest rate will double an initial investment of $2000 at the end of 10 years:

@RATE(4000,2000,10) = 7.18%

Other examples:

@RATE(10000,7000,6*12) = 0.50% (monthly)

@RATE(1200,1000,3) = 6.27% (yearly)

@RATE(500,100,25) = 6.65% (yearly)

Related topics

@RECEIVED - Redemption Value of a Bill

Syntax

@RECEIVED(Settle, Maturity, Investment, Discount, <Calendar>)

Settle	Number representing the settlement date; must be < Maturity.
Maturity	Number representing the maturity date; must be > Settle.
Investment	Amount invested; must be m 0.
Discount	Rate of discount; 0 ≤ Discount ≤ 1.
Calendar	Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

@RECEIVED returns the redemption value (the amount received after maturity) of a discount security. @RECEIVED uses this formula:

$$R = \frac{I}{\left(1 - \frac{D(M - S)}{t_b}\right)}$$

R	redemption
I	investment
D	discount
M	maturity
S	settle
b	basis

tb is the number of days over which the discount rate quote applies (360 or 365).

Example

This formula calculates the redemption value of a bill with the following terms: Settle is January 17, 1995, Maturity is June 15, 1995, Investment is $1,000,000, Discount is 7.922%, and Calendar is 2 (actual/360).

@RECEIVED(@DATE(95,1,17),@DATE(95,6,15),1000000,0.07922,2) = $1,033,899.79

Related topics

@REGRESSION - Multiple Linear Regression

Syntax

@REGRESSION(XBlock, YBlock, Attribute, <Compute>)

XBlock	Name or address of cell containing independent variables; can be up to 75 columns and 8,192 rows.
YBlock	Name or address of cells containing dependent variable; must be a single-column selection with the same number of rows as XBlock.
Attribute	Specifies the type of regression value returned. Valid attributes are 1, 2, 3, 4, 5, 101 to 175, or 201 to 275; see the table for the type of regression value returned for each attribute value.
Compute	Logical value (optional) that tells @REGRESSION whether to force the Y intercept to equal 0: 0 = make the Y intercept 0 1 = calculate the Y intercept (default if you omit the argument)

@REGRESSION performs a multiple linear regression, returning the specified statistic. Choose Attribute arguments from this table:

Attribute	**Regression Value Returned**
1	Constant
2	Standard error of Y estimate
3	R squared
4	Number of observations
5	Degrees of freedom
101 to 175	Slope, or X coefficient for the independent variable specified by Attribute
201 to 275	Standard error of the coefficient for the independent variable specified by Attribute

For the last two attributes, Quattro Pro numbers the independent variables in XBlock, starting with 1, from top to bottom in a column and from left to right. For example, if XBlock is B2..D7, use Attribute = 102 to find the X-coefficient for the independent variable in column C. Use Attribute = 201 to find the standard error of coefficient for the independent variable in column B.

Example

In trying to lower employee absenteeism, you suspect sick days correlate directly with outdoor temperatures and deadline pressure. You assemble data for the first Monday of the month over a six-month period:

	A	**B**	**C**	**D**	**E**
1	% absent	Date	Due date	Days left	9 a.m. temp.
2	14%	12/04/95	04/01/96	119	17
3	22%	01/02/96	04/01/96	90	10
4	32%	02/05/96	04/01/96	56	20
5	48%	03/04/96	04/01/96	28	35
6	2%	04/01/96	11/03/96	216	52
7	18%	05/06/96	11/03/96	181	55

*This January, "Monday" comes on Tuesday, the day after New Year's Day.

To find values for R squared:

@REGRESSION(D2..D7, A2..A7, 3) = 0.811862, fairly close to 1, indicating a fair correlation between absenteeism and deadlines.

@REGRESSION(E2..E7, A2..A7, 3) = 0.075683, much closer to 0, indicating little relation between absenteeism and weather.

Related topics

@REPEAT - Repeat Copies of a String

Syntax

@REPEAT(String, Num)

String	A string value.
Num	A numeric value equal to or greater than 0.

@REPEAT returns Num copies of String as one continuous label. This @function is similar to the repeating label prefix (\) in that it repeats one or more characters. The difference is that you can specify exactly how many times you want the string to be repeated. The \ label prefix adjusts the display to fill the column, even when the width is changed. @REPEAT displays a fixed number of copies of String and does not change.

When you specify a text string with @REPEAT, it must be enclosed by double quotes.

Examples

@REPEAT("-",20) = ----------

@REPEAT("good day!",3) = good day!good day!good day!

@REPEAT(A5,5) = the contents of A5 repeated 5 times

@REPEAT("-",@CELL("width",A1..A1)) = ---------- if column A is 12 characters wide. If you change the column width, you can press F9 to adjust the repeat string to fill the cell.

Related topics

@REPLACE - Replace Characters in a String

Syntax

@REPLACE(Original String, Starting Position, Chars to Replace, New String)

Original String	A valid string value, representing the text to operate on.
Starting Position	A numeric value equal to or greater than 0, representing the character position to begin with.
Chars to Replace	A numeric value equal to or greater than 0, representing the number of characters to delete.
New String	A string value, representing the characters to insert at position Num.

@REPLACE lets you replace characters in text with a new text string. It searches through the Original String from left to right beginning with the first character (character 0) until it reaches character position specified by Starting Position. Then it removes the number of characters from the string specified by Chars to Replace, replacing them with the New String.

Both the Original String and the New String can be either cell references or text strings. If text strings, they must be enclosed by double quotes.

To replace one string with another, specify 0 as the Starting Position. For Chars to Replace, enter a number equal to or greater than the number of characters in the Original String.

To insert one string into another string, specify 0 as the Chars to Replace.

To add one string to the end of another, specify as Starting Position a number one greater than the number of characters in the Original String.

To delete part or all of a string, specify "" as the New String.

Examples

@REPLACE("McDougal Corp.",2,6,"Douglas") = McDouglas Corp.

@REPLACE("Leslie J. Cooper",7,3,"") = Leslie Cooper

@REPLACE("Sales Salaries",6,0, "Reps' ") = Sales Reps' Salaries (There must be a space between Reps and the final quotation mark.)

@REPLACE("355 Howard",11,0," St.") = 355 Howard St. (There must be a space between " and St.)

You can use @REPLACE with other string functions. For instance, to replace one word with another within a sentence, you can use @FIND and @LENGTH to simplify the search-and-replace operation. For example,

@REPLACE(A7,@FIND("man",A7,0),@LENGTH("man"),"person")

searches through A7 for man, then replaces man with person.

Related topics

@RIGHT - Rightmost Characters

Syntax

@RIGHT(String, Num)

String	A string value.
Num	A numeric value ≥ 0.

@RIGHT returns Num characters of String counting from right to left. It extracts a specified number of characters from the right side of a string or label.

If String is a not a valid string, @RIGHT returns ERR. If Num is 0, the result is "", or an empty string. If Num is greater than or equal to the number of characters in String, the entire string is returned.

Examples

@RIGHT("Jennifer Meyer",5) = Meyer

@RIGHT("Jennifer Meyer",25) = Jennifer Meyer

@RIGHT("Jennifer ",6) = fer (including 3 subsequent spaces)

@RIGHT("155",1) = 5

@RIGHT(123,1) = ERR (123 is a value)

@RIGHT(A10,5) = the last five characters of A10

@RIGHT(A16,@LENGTH(A16) - @FIND("Roosevelt",A16,0)) = Roosevelt (if A16 = Theodore Roosevelt)

Related topics

@ROMAN - Arabic Numeral to Roman Numeral

Syntax

@ROMAN(Number, <Form>)

Number	Arabic numeral.
Form	Style of Roman numeral, ranging from full classic to brief, becoming more concise as the value of Form increases: FALSE (0) = Classic; default, if omitted TRUE (1) = More concise 2 = More concise 3 = More concise 4 = Most concise

@ROMAN returns the Roman numeral corresponding to a specified Arabic numeral, displaying it as text.

@ROMAN returns ERR if Number is negative or greater than 3999.

Examples

@ROMAN(1999) = MCMXCIX (classic)

@ROMAN(1999,FALSE) = MCMXCIX

@ROMAN(1999,0) = MCMXCIX

@ROMAN(1999,TRUE) = MLMVLIV

@ROMAN(1999,1) = MLMVLIV

@ROMAN(1999,2) = MXMIX

@ROMAN(1999,3) = MVMIV

@ROMAN(1999,4) = MIM

Related topics

@ROOTN - Nth Root of a Number

Syntax

@ROOTN(Number, Root)

Number	Number; can be positive or negative.
Root	Number, not zero.

@ROOTN calculates the nth root of a specified number. @ROOTN returns ERR if N=0.

Examples

@ROOTN(27,3) = 3
@ROOTN(64,2) = 8

Related topics

@ROUND - Round Number

Syntax

@ROUND(X, Num)

X	A numeric value.
Num	A numeric value between -15 and 15.

@ROUND adjusts the precision of X to Num decimal places. Num specifies the power of 10 to which X is rounded. If Num is positive, X is rounded Num digits to the right of the decimal point. If Num is negative, X is rounded Num digits to the left of the decimal point. For example, if Num is -3, X is rounded to the nearest thousand.

If Num is 0, X is rounded to an integer. If Num is not an integer, it is truncated to an integer.

Examples

@ROUND(12345.54321,0) = 12346

@ROUND(12345.54321,2) = 12345.54

@ROUND(12345.54321,-2) = 12300

Related topics

@ROUNDDOWN - Round Number Down

Syntax

@ROUNDDOWN(X, <Digits>, <Direction>)

X	Number to round down.
Digits	Number of digits (optional) to which you want to round X.
Direction	Argument (optional) specifying how to round negative values: 0 = round negative values down; default if omitted 1 = round negative values up

@ROUNDDOWN rounds a positive number down with a specified precision and rounds a negative number the direction you specify. @ROUNDDOWN behaves like @ROUND, except that it always rounds a number down.

- If Digits is positive, X is rounded down to Digits decimal places.
- If Digits is negative, X is rounded down to the nearest multiple of the power of 10 specified by Digits.
- If Digits is 0 or omitted, X is rounded down to the nearest integer.
- If X is positive, Direction has no effect.

Use the Fixed numeric format to display values with a specified number of decimal places if you want to calculate the values to their full precision; do not use @ROUNDDOWN. Quattro Pro stores a maximum of 15 digits.

Examples

@ROUNDDOWN(1234.5678) = 1234

@ROUNDDOWN(1234.5678,2) = 1234.56

@ROUNDDOWN(1234.5678,-2) = 1200

@ROUNDDOWN(-1234.5678) = -1235

@ROUNDDOWN(-1234.5678,0,1) = -1234

Related topics

@ROUNDDOWNXL - Round Number Down

Syntax

@ROUNDDOWNXL(X, <Digits>)

X	Number to round down.
Digits	Number of digits to which you want to round X.

@ROUNDDOWNXL rounds positive and negative numbers toward zero.

- If Digits is positive, X is rounded down to Digits decimal places.
- If Digits is negative, X is rounded down to the nearest multiple of the power of 10 specified by Digits.
- If Digits is 0 or omitted, then X is rounded down to the nearest integer.
- Rounding is always to a lower absolute value, regardless of sign.

Examples

@ROUNDDOWNXL(1234.5678,0) = 1234
@ROUNDDOWNXL(1234.5678,2) = 1234.56
@ROUNDDOWNXL(1234.5678,-2) = 1200
@ROUNDDOWNXL(-1234.5678,0) = -1234
By contrast, @ROUNDDOWN rounds negative numbers down, if you omit its Direction argument:
@ROUNDDOWN(-1234.5678,0) = -1235

Related topics

@ROUNDUP - Round Number Up

Syntax

@ROUNDUP(X, <Digits>, <Direction>)

X	Number to round up.
Digits	Number of digits (optional) to which you want to round X.
Direction	Argument (optional) specifying how to round negative values: 0 = round negative values up; default if omitted 1 = round negative values down

@ROUNDUP rounds a positive number up with a specified precision and rounds a negative number the direction you specify. @ROUNDUP behaves like @ROUND, except that it always rounds a number up.

- If Digits is positive, X is rounded up to Digits decimal places.
- If Digits is negative, X is rounded up to the nearest multiple of the power of 10 specified by Digits.
- If Digits is 0 or omitted, X is rounded up to the nearest integer.
- If X is positive, Direction has no effect.

Use the Fixed numeric format to display values with a specified number of decimal places if you want to calculate the values to their full precision; do not use @ROUNDUP. Quattro Pro stores a maximum of 15 digits.

Examples

@ROUNDUP(1234.5678) = 1235

@ROUNDUP(1234.5678,2) = 1234.57

@ROUNDUP(1234.5678,-2) = 1300

@ROUNDUP(-1234.5678) = -1234

@ROUNDUP(-1234.5678,0,1) = -1235

Related topics

@ROUNDUPXL - Round Number Up

Syntax

@ROUNDUPXL(X, <Digits>)

X	Number to round up.
Digits	Number of digits to which you want to round X.

@ROUNDUPXL rounds a positive number upward and rounds a negative number toward zero.

- If Digits is positive, X is rounded up to Digits decimal places.
- If Digits is negative, X is rounded up to the nearest multiple of the power of 10 specified by Digits.
- If Digits is 0 or omitted, then X is rounded up to the nearest integer.
- Rounding is always to a higher absolute value, regardless of sign.

Examples

@ROUNDUPXL(1234.5678,0) = 1235

@ROUNDUPXL(1234.5678,2) = 1234.57

@ROUNDUPXL(1234.5678,-2) = 1300

@ROUNDUPXL(-1234.5678,0) = -1235

By contrast, @ROUNDUP rounds negative numbers up, if you omit its Direction argument:

@ROUNDUP(-1234.5678,0) = -1234

Related topics

@ROW - Row Number

Syntax

@ROW(<Block>)

Block	The cell or cells for which you want the row number(s).

@ROW returns the row number(s) for a cell or cells.

- Block can be a cell name.
- If you omit Block, Quattro Pro assumes you want the row number of the cell where you entered @ROW.
- Block cannot refer to non-contiguous areas.

Examples

@ROW(A5..F5) = 5

@ROW(K1..M5) = {1| 2| 3| 4| 5}

If A12.. F12 is a selection named GIFTS, @ROW(GIFTS) = 12

Entered in C4 without an argument, @ROW = 4

Related topics

@ROWS - Number of Rows

Syntax

@ROWS(Block)

Block	Cell reference or name.

@ROWS returns the number of rows within the specified cells.

Examples

@ROWS(A1..A1) = 1
@ROWS(A1..C15) = 15
@ROWS(B100..B8192) = 8093
@ROWS(NAME) = 30 (if the cell NAME contains 30 rows)

Related topics

@RSQ - *r*-Squared Value of Linear Regression

Syntax

@RSQ(KnownX, KnownY)

KnownX	Independent range of values.
KnownY	Dependent range of values.

@RSQ returns *r* squared, the square of the Pearson product moment correlation coefficient. KnownX and KnownY must have the same number of values. Use @RSQ to test the linear relationship of KnownX and KnownY and to show the proportion of the variance of KnownY that can be attributed to a variance in KnownX. The r^2 statistic measures the fraction of the variance explained by the regression equation.

Example

@RSQ({7,8,5,9,7,2},{10,5,4,9,6,7}) = 0.063962

Related topics

@TABLELINK - Link to Database Table

Syntax

@TABLELINK(TableName,<Columns>,<Rows>)

TableName	Path and filename of the database table.
Columns	Number of columns (fields) to show from the linked table.
Rows	Number of rows (records) to show from the linked table.

@TABLELINK establishes a link to an external database table and displays the table in a Quattro Pro notebook. *TableName* is the filename of the database table to link to the notebook. By default, @TABLELINK links the entire table; to link selected columns and rows, enter values for the optional *Columns* and *Rows* arguments.

You can use Insert External Data Table Link to create a formula using @TABLELINK.

Examples

The following formula creates a link to a database table file named CUSTOMER.DB and displays seven columns and five rows of the table:

@TABLELINK("C:\COREL\SUITE8\SAMPLES\CUSTOMER.DB",7,5)

The next figure shows the resulting table. Cell A1 contains the @TABLELINK formula.

	A	B	C	D	E	F	G
1	1386	Aberdeen	F	45 Utah Street	Washington	DC	20032
2	1388	Svenvald	I	Government House	Reykjavik		
3	1784	McDougal	L	4950 Pullman Ave NE	Seattle	WA	98105
4	2177	Bonnefemme	S	128 University Drive	Stanford	CA	94323
5	2579	Chavez	L	Cypress Drive	Palm Springs	FL	32938

Related topics

@TAN - Tangent

Syntax

@TAN(X)

X	A numeric value.

@TAN returns the tangent of the angle X. X must be specified in radians, not degrees. To convert degrees to radians, use @RADIANS.

Examples

@TAN(4) = 1.157821

@TAN(@PI/4) = 1

@TAN(@RADIANS(45)) = 1

Related topics

@TANH - Hyperbolic Tangent

Syntax

@TANH(X)

X	A value from approximately -1.789E+308 to approximately 1.789E+308.

@TANH calculates the hyperbolic tangent of X. X must be specified in radians, not degrees. To convert degrees to radians, use @RADIANS.

The hyperbolic tangent is the ratio of hyperbolic sine to the hyperbolic cosine. @TANH returns a value from -1 through 1.

Examples

@TANH(4) = 0.999329

@TANH(@PI/4) = 0.655794

@TANH(RADIANS(45)) = 0.655794

Related topics

@TBILLEQ - Bond Equivalent Yield of a Treasury Bill

Syntax

@TBILLEQ(Settle, Maturity, Discount)

Settle	Number representing the settlement date; must be < Maturity. Also, Maturity cannot be more than one year after the settlement date.
Maturity	Number representing the maturity date.
Discount	Rate of discount expressed as a decimal fraction; must be > 0 and ≤ 1.

@TBILLEQ returns the bond equivalent yield (also called coupon equivalent yield) of a Treasury bill. Use @TBILLEQ to compare the yields of Treasury bills and bonds.

Example

This formula calculates the bond equivalent yield of a bill quoted at 9.35% with a maturity date of October 16, 1996 for settlement on April 9, 1996:

@TBILLEQ(@DATE(96,4,9),@DATE(96,10,16),0.0935) = 0.099524

Related topics

@TBILLPRICE - Price of a Treasury Bill

Syntax

@TBILLPRICE(Settle, Maturity, Discount)

Settle	Number representing the settlement date; must be < Maturity.
Maturity	Number representing the maturity date.
Discount	Rate of discount expressed as a decimal fraction; must be ≥ 0 and ≤ 1.

@TBILLPRICE calculates the price per 100 face value of a Treasury bill. The dollar price of a Treasury bill is its face value less the applicable dollar discount. @TBILLPRICE uses this formula:

$$P = 100 * \left(1 - D * \left(\frac{M - S}{360}\right)\right)$$

P	price
D	discount
M	Maturity
S	Settle

Example

This formula calculates the price per 100 face value for a bill maturing August 1, 1996 and trading at 9.14% for January 3, 1996 settlement:

@TBILLPRICE(@DATE(96,1,3),@DATE(96,8,1),0.0914) = 94.64294

Related topics

@TBILLYIELD - Yield of a Treasury Bill

Syntax

@TBILLYIELD(Settle, Maturity, Price)

Settle	Number representing the settlement date; must be < Maturity.
Maturity	Number representing the maturity date.
Price	The Treasury bill's price per 100 face value.

@TBILLYIELD returns the yield of a Treasury bill.

Example

This formula calculates the yield of a bill maturing August 1, 1996 and trading at a price of 94.64294 for January 3, 1996 settlement:

@TBILLYIELD(@DATE(96,1,3),@DATE(96,8,1),94.64294) = 0.096574

Related topics

@TDIST - Student's t-Distribution

Syntax

@TDIST(X, DegFreedom, Tails)

X	Value at which to evaluate the distribution.
DegFreedom	Integer number of degrees of freedom; must be ≥ 1.
Tails	1 to return a one-tailed distribution; 2 to return a two-tailed distribution.

@TDIST returns the one-tailed or two-tailed Student's t-distribution, which is the probability that a specified realization will be greater than X. Use the t-distribution when comparing the means of small samples. The single-tailed t-distribution yields a probability of deviation in a single direction from the mean; a two-tailed t-distribution yields a probability of deviation in two directions.

If DegFreedom is not an integer, @TDIST rounds it to the nearest integer.

Example

@TDIST(2.228139,10,2) = 0.05

Related topics

@TERM - Number of Payment Periods for an Annuity

Syntax

@TERM(Pmt, Rate, Fv)

Pmt	A numeric value representing the amount of the periodic payment.
Rate	A numeric value > -1, representing the periodic interest rate (the fixed interest rate per compounding period).
Fv	A numeric value representing the future value of an investment (the value the investment will reach at some point).

@TERM computes the number of payment periods required to accumulate an investment of Fv, making regular payments of Pmt and accruing interest at the rate of Rate.

@TERM uses this formula:

$$\frac{\ln(1 + Fv/Pmt*Rate)}{\ln(1 + Rate)}$$

An equivalent for this formula using @NPER is
@NPER(Rate,-Pmt,0,Fv)
@TERM assumes the investment is an ordinary annuity. The related @function @NPER uses an optional argument, Type, to indicate whether the investment is an ordinary annuity or an annuity due.

Examples

To determine how long it will take to accrue $50,000 by depositing $2000 at the end of each year into a savings account that earns 11% annually, enter this formula:

@TERM(2000,11%,50000) = 12.67

Quattro Pro determines that it will take 12.67 years to accumulate $50,000 in your account. (Depending upon how your bank pays interest, your balance might not exceed $50,000 until the end of the 13th year.)

If, on the other hand, the money is not coming in to you but is being paid out by you, you can enter the future value as a negative number.

You can also use @NPER to calculate this example:

@NPER(11%,-2000,0,50000,0) = 12.67

Other examples:

@TERM(300,6%,5000) = 11.9 years

@TERM(500,7%,1000) = 1.94 years

@TERM(500,.07,1000) = 1.94 years

@TERM(1000,10%,50000) = 18.8 years

@TERM(100,5%,1000) = 8.3 years

Related topics

@TIME - Date serial number for Hr:Min:Sec

Syntax

@TIME(Hr, Min, Sec)

Hr	A number between 0 and 23, representing Hour.
Min	A number between 0 and 59, representing Minute.
Sec	A number between 0 and 59, representing Second.

@TIME returns the date/time serial number represented by Hr:Min:Sec. Any fractional portions of Hr, Min, and Sec are rounded. You can display the resulting time string values in standard time formats by right-clicking the cell, clicking Cell Properties, then clicking Numeric Format.

See "Using dates and times in Quattro Pro."

Examples

@TIME(3,0,0) = 0.125 (3:00 am)

@TIME(3,30,15) = 0.14600694444 (3:30:15 am)

@TIME(18,15,59) = 0.76109953704 (6:15:59 pm)

@TIME(B15,23,45) = 0.099826388889 (when the value in B15 is 2)

@TIME(@HOUR(C3),A4,B10) = 0.5751388889 (1:48:12 pm) (when C3 = 01:23:13 pm (formatted date/time serial number), A4 = 48, and B10 =12)

Related topics

@TIMEVALUE - Value Corresponding to TimeString

Syntax

@TIMEVALUE(TimeString)

TimeString	A numeric value or a string value in any valid time format, enclosed by quotation marks.

@TIMEVALUE returns a serial time value that corresponds to the value in TimeString. If the value in TimeString is not in the correct format, or is not enclosed in quotes (if entered as a literal string), an ERR value is returned.

You can display resulting time string values in standard time formats by right-clicking the cell, clicking Cell Properties, then clicking Numeric Format.

There are four valid formats for TimeString:

- HH:MM:SS AM/PM (03:45:30 PM)
- HH:MM AM/PM (03:45 PM)
- The Long International time format chosen as a system default, one of which is HH:MM:SS (15:45:30)
- The Short International time format chosen as a system default, one of which is HH:MM (15:45)

See "Using dates and times in Quattro Pro."

Examples

@TIMEVALUE("03:30:15 AM") = 0.1460069444

@TIMEVALUE("03:00") = 0.125

@TIMEVALUE("18:15:59") = 0.76109953704

@TIMEVALUE("3.45") = ERR

@TIMEVALUE(@TIME(12,30,45)) = 0.521354

@TIMEVALUE(A1) = 0.125 if A1 contains the label '03:00

Related topics

@TINV - Inverse of Student's t-Distribution

Syntax

@TINV(Prob, DegFreedom)

Prob	Cumulative probability value; $0 \leq Prob \leq 1$.
DegFreedom	Number of degrees of freedom.

@TINV returns the inverse of the t-distribution. Use the t-distribution when comparing the means of small samples.

Example

@TINV(0.05,10) = 2.228139

Related topics

@TODAY - Today's Date

Syntax

@TODAY

@TODAY enters the numeric value of the system's date. It is equal to the expression @INT(@NOW).

See "Using dates and times in Quattro Pro."

Related topics

@TOTAL - Sum, Excluding Subtotals

Syntax

@TOTAL(List)

List	One or more numeric values, cell addresses, and cell references or names, separated by commas.

@TOTAL returns the total of all numeric values in a list or reference, excluding any subtotals. List can be any combination of single cell references, cells, and numeric values. When more than one component is used, they must be separated by commas. Any labels, blank cells, or cells containing @SUBTOTAL or @SUBTOTAL123 are ignored by @TOTAL.

Any dates in the cells will be converted to serial numbers and included in the calculation. Since this will throw off your sum, avoid including dates in the @TOTAL argument cells.

Example

In the following, Cells A3 and C3 contain @SUBTOTAL(A1..A2) and @SUBTOTAL(C1..C2), respectively. Cell A4 contains the formula @TOTAL(A1..C3) and sums all the values in the cells except those generated by @SUBTOTAL or @SUBTOTAL123. To omit possible subtotals in Column B, you could also write: @SUM(A1..A2,B1..B2).

	A	B	C	D
1	$30		$18	
2	$65		$22	
3	$95		$40	@SUBTOTAL in A3 and C3
4	$135			@TOTAL in A4

Related topics

@TRANSPOSE - Transpose of Cells

Syntax

@TRANSPOSE(Block)

Block	Cells or array to transpose; you can use a 2-D cell reference or cell name or an array constant like {1,2|3,4}.

@TRANSPOSE returns the transpose of cells or an array. In the transpose of cells, each row of the cells becomes the corresponding column of the output array.

Use @TRANSPOSE to shift the vertical and horizontal orientation of cells or an array.

Formatting is not transposed with the values. Cells in the output cells retain the formatting they had before you entered the transposed cell values.

You can also copy an array to the Clipboard and then use Edit|PasteSpecial or the {PasteSpecial} macro to transpose the array and paste only the values it contains.

Examples

@TRANSPOSE({1,2|3,4}) = {1,3|2,4}

Suppose you want to display the following columns as rows:

	A	B
1	Quarter	Sales
2	1	$80,000
3	2	$90,000
4	3	$95,000
5	4	$105,000

Enter the formula

@TRANSPOSE(A1..B5)

The result is displayed as follows:

	A	B	C	D	E
6	Quarter	1	2	3	4
7	Sales	80000	90000	95000	105000

Related topics

@TREND - Fits Straight Line to Data

Syntax

@TREND(KnownYs, <KnownXs>, <NewXs>, <Const>)

KnownYs	Array of known y-values for the line y = mx + b.
KnownXs	Array of known x-values (optional).
NewXs	Array of new x-values for which you want the corresponding y-values (optional).
Const	Logical value (optional) that tells @TREND whether to force the constant b = 0: If Const is TRUE or omitted, @TREND uses the actual value of b. If Const is FALSE, @TREND sets b = 0, then adjusts the m-values so that y = mx.

@TREND fits a straight line to data, using the "least squares" method, then predicts further y-values on that line for a specified array of x-values.

- If known y-values are in one column, @TREND takes each column of known x-values to be a separate variable. If known y-values are in one row, @TREND takes each row of known x-values to be a separate variable.
- The argument KnownXs can include more than one set of variables. If you use only one variable, KnownYs and KnownXs can be selections of any shape, but must have the same dimensions. If you use more than one variable, KnownYs must be a single-column or single-row selection. Use commas to separate x-values in the same row and pipes (|) to separate rows.
- The argument NewXs must follow the pattern of KnownXs: It must include a row or column for each independent variable. If you omit the argument NewXs, @TREND assumes it is the same as KnownXs. If you omit both KnownXs and NewXs, @TREND assumes they are the array {1,2,3,...} of a size equal to KnownYs.

You can use @TREND for polynomial curve fitting by regressing against the same variable raised to different powers. For example, suppose column A contained y-values and column B contained x-values. You could enter x^2 in column C, x^3 in column D, and so on, and then regress columns B through D one at a time against column A.

Example

Sales for your company in its first four quarters are entered in a selection named Sales:

	A	B
1	Quarter	Sales
2	1	$80,000
3	2	$90,000
4	3	$95,000
5	4	$105,000

To predict sales for the following year, @TREND(Sales,A2..A5,A6..A9) = {$112,500, $120,500, $128,500, $136,500}

Related topics

@TRIM - Remove Extra Spaces

Syntax

@TRIM(String)

String	A string value.

@TRIM removes any extra spaces from String; that is, spaces following the last nonspace character or preceding the first nonspace character, and duplicate spaces between words. Strings with no extra spaces are not affected. If String is empty or contains a numeric value, it returns ERR.

Examples

@TRIM(" too many spaces ") = too many spaces

@TRIM("no extra spaces") = no extra spaces

@TRIM(125) = ERR

Related topics

@TRIMMEAN - Mean, with Fraction Excluded

Syntax

@TRIMMEAN(Array, Fraction)

Array	Numeric array or cells of values.
Fraction	Decimal fraction of data points to exclude; $0 \leq$ Fraction < 1.

@TRIMMEAN computes the mean of a data set, with a fraction of the points excluded. @TRIMMEAN is helpful when you want to exclude outlying values from your analysis. Because the function excludes an equal number of data points from the top and bottom of the data set, it rounds the number of excluded data points down to the nearest multiple of 2.

Example

@TRIMMEAN({3,6,7,8,10,11,11,14,15,20},0.2) = 10.25

Related topics

@TRUE - Logical Value 1

Syntax

@TRUE

@TRUE returns the logical value 1 and is usually used in @IF formulas. The 1 it returns is the same as the regular numeral 1, but @TRUE makes the formula easier to read.

Examples

@TRUE = 1

@IF(C3=100,@TRUE,10) = 1 (if C3 = 100) or 10 (if C3 is not equal to100)

@IF(C3=100,@TRUE,@FALSE) = 1 (if C3 = 100) or 0 (if C3 is not equal to 100)

Related topics

@TRUNC - Truncate Number

Syntax

@TRUNC(X, <Digits>)

X	Number to truncate.
Digits	Numeric value specifying precision (optional); can be from -100 through 100.

@TRUNC truncates a number to the precision you specify. The default precision is without decimal places, if you omit the Digits argument.

- If Digits is positive, X is truncated to Digits decimal places.
- If Digits is negative, X is truncated to the nearest multiple of the power of 10 specified by Digits.
- If Digits is 0, X is truncated to the nearest integer.
- Though precision can be from -100 through 100, only 15 digits can be displayed.

Use the Fixed numeric format to display values with a specified number of decimal places if you want to calculate the values to their full precision; do not use @TRUNC.

Examples

@TRUNC(1234.5678) = 1234

@TRUNC(1234.5678,3) = 1234.567

@TRUNC(1234.5678,-2) = 1200

@TRUNC(-1234.5678,-2) = -1200

Related topics

@TTEST - Probability of Student's t-Test

Syntax

@TTEST(Array1, Array2, Tails, Type)

Array1	First array of numeric values.
Array2	Second array of numeric values.
Tails	1 to return a one-tailed test; 2 to return a two-tailed test.
Type	A discrete variable specifying the type of test to conduct; 1 = a paired test; 2 = a two-sample equal variance test; 3 = a two-sample unequal variance test.

@TTEST returns the probability associated with the Student's t-Test. Use @TTEST to test the means of two small samples.

If Array1 and Array2 have a different number of data points, @TTEST returns ERR.

Example

@TTEST({62,77,73,69,54,67,59,76},{64,80,72,53,69,63,76,74},2,2) = 0.68502

Related topics

@TYPE - Value Type

Syntax

@TYPE(Value)

Value	Numeric, text, logical, formula, or error value, or reference to a cell containing such a value.

@TYPE returns a code indicating the type of a specified value. Value types and their codes are:

Type	**@TYPE returns**
Numeric or logical value, or empty cell	1
Text	2
Error	16
Array	64

Some functions accept and return several types of data, some accept only one type. Use @TYPE when you need to find out what type of data was returned by a function.

Examples

@TYPE(457) = 1

@TYPE("number") = 2

@TYPE(TRUE) = 1

If D3 contains @ROMAN(-5), @TYPE(D3) = 16, because @ROMAN of a negative number returns ERR

If D3 is a blank cell, @TYPE(D3) = 1

If D3 contains @ARRAY(A1..A5), @TYPE(D3) = 64

Related topics

@UPPER - String in Uppercase

Syntax

@UPPER(String)

String	A string value.

@UPPER returns String in uppercase characters. Numbers and symbols within a string are unaffected. If String is blank, or contains a numeric or date value, the result is ERR.

Examples

@UPPER(4839) = ERR

@UPPER(@LEFT("johnson",1)) = J

@UPPER("upper") = UPPER

@UPPER("Hello, world.") = HELLO, WORLD.

@UPPER("145 Bancroft Lane") = 145 BANCROFT LANE

Related topics

@USESPLINE - Use Piecewise Polynomial Fit

Syntax

@USESPLINE(KnownX's, KnownY's, Coefficients, x)

KnownX's	Independent cells or array of values.
KnownY's	Dependent cells or array of values.
Coefficients	Coefficient array produced by @SPLINE.
x	Value for which you want the corresponding y-value.

@USESPLINE returns, for any specified x value, the corresponding y value, given the original x and y arrays and the coefficient array produced by @SPLINE.

If x is larger than the largest value contained in the interval over which the spline is performed, then UseSpline will return the y co-ordinate associated with that largest value as the interpolated value. Similarly, if x is smaller than the smallest value contained in the interval over which the spline is performed, then UseSpline will return the y co-ordinate associated with that smallest value as the interpolated value.

Example

Given the following data:

	A	B	C	D	E
1					
2			1	2	3
3			4	2	5
4			-2	3	
5	0				
6	0.2				
7	0.3				
8	1				
9	1.25				
10	1.333				
11	1.5				
12	1.9				
13	2				
14	2.5				
15	2.6				
16	2.96				
17	3				
18	3.1				
19	3.3				
20	3.6				
21	3.777				
22	4				

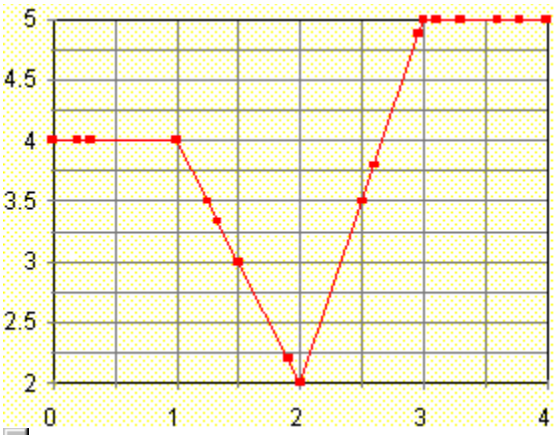
@USESPLINE(C$2..E$2,C$3..E$3,C$4..D$4,A5) = 4

Copying this function into successive rows in Column B gives the corresponding y-values for x-value in column A:

5	0	4
6	0.2	4

7	0.3	4
8	1	4
9	1.25	3.5
10	1.333	3.334
11	1.5	3
12	1.9	2.2
13	2	2
14	2.5	3.5
15	2.6	3.8
16	2.96	4.88
17	3	5
18	3.1	5
19	3.3	5
20	3.6	5
21	3.777	5
22	4	5

This is the graph using the x-values in Column A and the corresponding y-values generated in Column B by @USESPLINE:



Related topics

@VALUE - Value of a String

Syntax

@VALUE(String)

String	A string value.

@VALUE converts String into a numeric value. String can contain arithmetic operators (but do not place arithmetic operators within quotes). String must not contain embedded spaces. Dollar signs, commas, and leading and trailing spaces are ignored.

This @function is useful for converting imported data that has not already been converted into values.

Examples

@VALUE(" 3.59") = 3.59 (leading spaces are stripped)

@VALUE(" 98.6 ") = 98.6 (leading and trailing spaces are stripped)

@VALUE("98.6 4") = ERR (an embedded space is not allowed)

@VALUE(3+4) = 7

@VALUE("3+4") = ERR (arithmetic operators within quotes are not allowed)

@VALUE(" 88.039") = 88.039

@VALUE(A10) = 56.34 (where cell A10 = '$56.34)

@VALUE("34,200") = 34200

@VALUE(A1) = ERR (where cell A1 = '1800 Green Hills Road)

Related topics

@VAR - Population Variance

Syntax

@VAR(List)

List	One or more numeric or string values, cell addresses, and cell references or names, separated by commas.

@VAR calculates the population variance of all nonblank, numeric cells in List, using the n method (biased). Use @VARS to compute the variance of a data sample.

If List contains text, a reference to a single cell containing a label (for example, @VAR(B1) where B1 = Adam), or label cells within references to multiple cell selections (such as @VAR(B1..B5)), @VAR treats the string as having a value of 0. @VAR ignores blank cells within a referenced selection of cells, but returns ERR if every cell in the selection is blank.

Examples

@VAR(23,24,25) = 0.666666667

@VAR("Adam",53) = 702.25 (same as for @VAR(0,53)

@VAR(B1..B4) = 54.6875 (if B1=10, B2=15, B3="Susan", B4=20; the string in B3 is treated as if it were 0)

@VARS(23,24,25) = 1

@VARS("Adam",53) = 1404.5 (same as for @VARS(0,53)

@VARS(B1..B4) = 72.9167 (if B1=10, B2=15, B3="Susan", B4=20; the string in B3 is treated as if it were 0)

Related topics

@VARS - Sample Population Variance

Syntax

@VARS(List)

List	One or more numeric or string values, cell addresses, and cell references or names, separated by commas.

@VARS calculates the sample variance of all nonblank, numeric cells in List, using the n-1 method (unbiased). @VAR computes population variance.

Examples

@VARS(23,24,25) = 1

@VARS("Adam",53) = 1404.5 (same as for @VARS(0,53)

@VARS(B1..B4) = 72.9167 (if B1=10, B2=15, B3="Susan", B4=20; the string in B3 is treated as if it were 0)

@VAR(23,24,25) = 0.666666667

@VAR("Adam",53) = 702.25 (same as for @VAR(0,53)

@VAR(B1..B4) = 54.6875 (if B1=10, B2=15, B3="Susan", B4=20; the string in B3 is treated as if it were 0)

Related topics

@VDB - Variable-rate Declining Balance Depreciation

Syntax

@VDB(Cost, Salvage, Life, StartPeriod, EndPeriod, <Factor>, <Switch>)

Cost	Cost of asset; must be greater than Salvage.
Salvage	Salvage value at end of asset life; can be any value.
Life	Number of periods for asset to depreciate to salvage value; must greater than 0.
StartPeriod	Starting period to begin depreciation, in same units as Life; can be any positive value or 0, but not greater than Life.
EndPeriod	Ending period for depreciation, in same units as Life; can be any value greater than StartPeriod, but not greater than Life.
Factor	Percentage of straight-line depreciation to use as the depreciation rate (optional); 200% (double-declining balance rate) if omitted. Factor can be any value greater than or equal to 0; commonly used rates are 1.25, 1.50, 1.75, and 2.
Switch	Tells @VDB whether to switch to straight-line depreciation for the remaining useful life (optional): 0 = automatically switch to straight-line depreciation when that is greater than declining-balance depreciation (default if you omit the argument) 1 = never switch to straight-line depreciation

@VDB calculates depreciation allowance using the variable-rate declining balance method. The calculation is based on initial cost, expected useful life, and final salvage value for a specified period.

The variable-rate declining balance method uses a fixed depreciation rate until the asset's salvage value is less than the value of the following expression:

BV * ((1- (R / L)) L)

where

BV	book value = cost - salvage - prior depreciation
R	rate
L	life

At this point, Quattro Pro switches to straight-line depreciation for the balance of the life of the asset so that there is no excess salvage value. By switching to straight-line depreciation, Quattro Pro adjusts the result of @VDB when necessary to ensure that total depreciation taken over the life of the asset equals the asset's cost minus its salvage value.

When you use any optional argument, you must also use the ones before it.

Examples

You purchased a delivery truck for your business for $45,000 at the end of the first quarter of your fiscal year. Its useful life is projected to be 5 years, with a salvage value of $1100. Using the variable-rate declining balance method, with a depreciation rate of 150%, your depreciation expense for each year will be:

@VDB(45000,1100,5,0,0.75,1.5) = $10,125.00

@VDB(45000,1100,5,0.75,1.75,1.5) = $10,462.50

@VDB(45000,1100,5,1.75,2.75,1.5) = $7,323.75

@VDB(45000,1100,5,2.75,3.75,1.5) = $7,106.11

@VDB(45000,1100,5,3.75,4.75,1.5) = $7,106.11

@VDB(45000,1100,5,4.75,5,1.5) = $1,776.53

Total depreciation (cost minus salvage) = $43,900.00

The switch to straight-line depreciation begins automatically in the third year: If Switch = 1, the calculation for that year would be:

@VDB(45000,1100,5,2.75,3.75, 1.5,1) = $5,126.63

Related topics

@VERSION -Quattro Pro Version Number

Syntax

@VERSION

@VERSION returns the version number of Quattro Pro.

Related topics

@VLOOKUP - Vertical Lookup

Syntax

@VLOOKUP (X, Block, Column, <Type>)

X	The numeric or string value you want to search for.
Block	The range of cells.
Column	The number of the referenced column. The columns are referenced from 0 to the number of columns in Block minus 1.
	The first column (index column) in Block = 0
	The second column in Block = 1, ...
Type <optional>	Lets you specify whether or not the match must be exact.
	0 Must be an exact match
	1 Does not need to be an exact match (default)

@VLOOKUP searches vertically through the index column of Block for the value X. When @VLOOKUP finds the value X, it returns the value displayed Column columns beneath it.

All values in the index column must be sorted in ascending order for the function to work correctly. Otherwise, ERR or an incorrect answer may be returned.

@VLOOKUP returns 0 if the referenced cell is blank. ERR is returned if:

- Column is less than 0 or greater than the number of columns minus 1 in Block.
- X is less than the smallest value in the first column of Block.
- X is a string and the index column is not found.

If X is a string, @VLOOKUP looks for an exact case-sensitive match. If X is a number and @VLOOKUP cannot find an equal number, it locates the highest number, not more than X, in the column.

The Block must have its index values in the leftmost column.

There must be no blank cells in the index column. Blanks in the table to the right of the index column are treated as a 0.

Example

In the following example, @VLOOKUP searches down the index column (A) of the Block (A1..D4) looking for the largest number equal to or less than X (17). It stops at cell A3, then moves across Column columns (3). It stops at cell D3 and returns the value 22.

	A	B	C	D
1	5	52	84	43
2	10	32	67	45
3	15	42	18	22
4	20	83	76	47

@VLOOKUP(17, A1..D4, 3)

Returns: 22

Related topics

@VHLOOKUP - Vertical and Horizontal Lookup

Syntax

@VHLOOKUP (V_Val, H_Val, Block, <Type>)

V_Val	The value of the index row.
H_Val	The value of the index column.
Block	The range of cells.
Type <optional>	Lets you specify whether or not the match must be exact. 0 Does not need to be an exact match 1 Must be an exact match

@VHLOOKUP returns the value at the intersection of the row and column specified by V_Val and H_Val.

Example

In the following example, @VHLOOKUP searches across the index row ("FEBRUARY") and down the index column ("TWO") of the Block (A1..D4). @VHLOOKUP returns the value at the intersection of the index row and the index column.

	A	B	C	D
1		ONE	TWO	THREE
2	JANUARY	5	52	84
3	FEBRUARY	10	32	67
4	MARCH	15	42	18

@VHLOOKUP("FEBRUARY", "TWO", A1..D4)

Returns: 32

Related topics

@WEEKDAY - Number of the Weekday

Syntax

@WEEKDAY(Date, Date Type)

Date	Number representing a date to check. See "Using dates and times in Quattro Pro."
Date Type	One of three numbering systems representing days of the week.

Date Type	What It Means
1 or blank	1=Sun, 2=Mon, 3=Tues, 4=Wed, 5=Thur, 6=Fri, 7=Sat
2	1=Mon, 2=Tues, 3=Wed, 4=Thur, 5=Fri, 6=Sat, 7=Sun
3	0=Mon, 1=Tues, 2=Wed, 3=Thur, 4=Fri, 5=Sat, 6=Sun

@WEEKDAY returns a number representing the day Date falls on.

Example

@WEEKDAY("1/22/97",1) = 4 (January 22, 1997 falls on Wednesday)

@WEEKDAY(35452,1) = 4 (Wednesday)

@WEEKDAY(35452,3) = 2 (Wednesday, using Date Type 3)

@WEEKDAY(@DATE(96,1,22)) = 2 (Monday)

Related topics

@WEEKNUM - Week of the Year

Syntax

@WEEKNUM(DateNum, WeekBeg)

DateNum	The date expressed as a serial date number. See "Using dates and times in Quattro Pro."
WeekBeg	A number that determines on what day the week begins. 1 = week begins on Sunday; default if omitted 2 = week begins on Monday 3 = week begins on Saturday

@WEEKNUM calculates in which week of the year a specified date falls. Make sure to set Cell Properties to General, not Date, to display to display the result of @WEEKNUM.

Examples

February 4, 1996, is a Sunday.

@WEEKNUM(@DATE(96,2,4),1) = 6

@WEEKNUM(@DATE(96,2,4),2) = 5

Related topics

@WEIBULL - Weibull Distribution (Mean Time to Failure)

Syntax

@WEIBULL(X, Alpha, Beta, Cum)

X	Function parameter to evaluate.
Alpha	Parameter to the distribution; must be > 0.
Beta	Parameter to the distribution; must be > 0.
Cum	A numeric value (0 or 1) indicating whether to use the cumulative distribution function (1) or the probability density function (0).

@WEIBULL returns the Weibull distribution, which is used to calculate the mean time to failure of a device. If Alpha = 1, @WEIBULL returns the same value as @EXPONDIST with Lambda = 1/Beta.

Examples

@WEIBULL(20,2,15,1) = 0.830987

@WEIBULL(20,2,15,0) = 0.030047

Related topics

@WEIGHTAVG - Weighted Average

Syntax

@WEIGHTAVG(DataBlock, WeightsBlock, <Type>)

DataBlock	Cell reference or name where values to be averaged are stored.
WeightsBlock	Cell reference or name where data affecting the weighting are stored; WeightsBlock must have the same dimensions as DataBlock.
Type	Optional value that tells Quattro Pro how to calculate the weighted average: 0 = divide by sum of values in WeightsBlock; default if you omit the argument 1 = divide by number of values in DataBlock

@WEIGHTAVG returns a weighted average of the values in the cells.

@WEIGHTAVG returns ERR if DataBlock and WeightsBlock are not the same dimensions.

Examples

Your son's school places heavy emphasis on math and science courses, and weights them accordingly in comparison with humanities courses. His fall semester grades are in the following table, along with the school's weighting of the courses:

	A	B	C
1	Course	Weight	Grade
2	English	2	70.0%
3	Geography	2	65.0%
4	Math	3	95.0%
5	Chemistry	3	91.0%

@WEIGHTAVG(C2..C5,B2..B5) = 82.8%

@AVG(C2..C5) = 80.3%

Related topics

@WKDAY - Number of the Weekday

Syntax

@WKDAY(Date)

Date	Number representing a date to check. See "Using dates and times in Quattro Pro."

@WKDAY returns a number (from 1 for Saturday to 7 for Friday) representing the day Date falls on.

Example

@WKDAY(@DATE(95,6,22)) = 6 (Thursday), since June 22, 1995 falls on a Thursday.

Related topics

@WORKDAY - Date a Specified Number of Days Away

Syntax

@WORKDAY(StartDate,Days,<Holidays>,<Weekends>)

StartDate	Serial number for the date you're starting from. See "Using dates and times in Quattro Pro."
Days	Number of days after StartDate (if Days is positive) or before (if Days is negative).
Holidays	Optional cell name or reference containing serial date numbers of holidays to exclude from the calculation.
Weekends	Optional argument, in quotation marks, to tell @WORKDAY which days are weekend days. Use 0 through 6 (Monday through Sunday); for example, "45" means Friday and Saturday. The default, if you omit Weekends, is Saturday and Sunday. To specify no weekends, use "7".

@WORKDAY returns the serial number for a date that is a specified number of days before or after a specified date, optionally excluding weekends and/or holidays. To see the date as text, format the cell by choosing Cell Properties and Date.

You cannot use any optional argument without using all the ones preceding it. To specify weekends but not holidays, refer to a blank cell for holidays.

Examples

You want the date 10 working days after Monday, December 23, 1996. Holidays are stored in a selection named Holidays, and your weekend days are Saturday and Sunday.

HOLIDAYS	**A**
1	35397
2	35398
3	35424
4	35425
5	35426
6	35431
7	35432
8	35433

@WORKDAY(@DATE(96,12,23),10,Holidays) = 35444, or 14-Jan-97

If your weekend days are Friday and Saturday,

@WORKDAY(@DATE(96,12,23),10,Holidays,"45") = 35442, or 12-Jan-97

Say you're planning ton work on Saturdays and Sundays, but take the holidays:

@WORKDAY(@DATE(96,12,23),10,Holidays,"7") = 35438, or 08-Jan-97

If you decide to work holidays but not Snaturdays or Sundays during that period, you would not specify holidays, and Saturdays and Sundays would be assumed:

@WORKDAY(@DATE(96,12,23),10) = 35436, or 06-Jan-97

By working straight through, taking neither holidays nor weekends off, you could finish four days earlier:

@WORKDAY(@DATE(96,12,23),10, B3,"7") = 35432, or 02-Jan-97. Cell B3 is empty, to specify no holidays.

The date 10 working days before Monday, December 23, 1996, is

@WORKDAY(@DATE(96,12,23),-10, Holidays) = 35408, or 9-Dec-96

Related topics

@XCOUNT - Counts Numeric Cells

Syntax

@XCOUNT(LIST)

List	One or more numeric, cell addresses, and references or names, separated by commas.

@XCOUNT returns the number of cells in List containing a numeric value. If there is no content in the list to return, @XCOUNT returns 0.

Example

@XCOUNT(I1..I34) = 13 (if there are 13 cells containing numeric values)

Related topics

@XINDEX - Return Value at Column and Row

Syntax

@XINDEX(Block,ColHead,RowHead,<PageName>)

Block	Cell name or reference.
ColHead	Column to look in; must be the contents of a cell in the first row of the cells.
RowHead	Row to look in; must be the contents of a cell in the first column of the cells.
PageName	Name of notebook sheet (optional).

@XINDEX returns the contents of a cell located at the intersection of a specified column, row, and (optionally) notebook sheet.

ColHead, RowHead, and PageName can be numeric values or text.

@XINDEX looks first for the sheet name, if specified. Then it scans the leftmost column for the value RowHead and the top row for the value ColHead, returning the value in the cell at the intersection.

Examples

You record home sales prices and dates in a table called HOUSES, with a notebook sheet for each area. The sheet for Long Island looks like this:

	A	B	C	D	E	F
1	Cust#	Offered	Date sold	Asked	Sold for	Months on mkt
2	286	05-May-95	27-Jul-95	$325,000	$315,000	2
3	183	02-Feb-93	7-Aug-95	$295,000	$250,000	30
4	173	04-Apr-95	11-Sep-95	$150,000	$135,000	5
5	218	25-May-94	12-Oct-95	$495,000	$425,000	16
6	104	15-Mar-94	17-Oct-95	$195,000	$150,000	19

The amount Long Island customer 183 paid is

@XINDEX(HOUSES,"Sold for", 183, "Long Island") = $250,000

The date Customer 218 closed is

@XINDEX(HOUSES,"Date sold", 218, "Long Island") = 12-Oct-95

At 5% commission, the agent who sold Customer 104 a house received

+0.05*@XINDEX(HOUSES, "Sold for", 104, "Long Island") = $7,500

Related topics

@XIRR - Internal Rate of Return

Syntax

@XIRR(Values, Dates, <Guess>)

Values	Series of cash flows; first payment is the one occurring at the beginning of the investment; succeeding payments are discounted based on a 365-day year.
Dates	Payment corresponding to cash flow payments; first date must be the earliest date, but all other dates can be in any order.
Guess	Numeric value (optional) that estimates the internal rate of return on an investment; assumed to be 10% if omitted.

@XIRR returns the internal rate of return on an investment when cash flow is not necessarily periodic.

- Numbers in Dates are truncated to integers.
- In most cases you do not need to provide Guess for the @XIRR calculation. Use the Guess argument to speed up calculation and ensure a correct solution.

@XIRR returns ERR if:

- An argument is non-numeric.
- It does not find at least one positive cash flow and one negative cash flow.
- A number in Dates is not a valid date number.
- A number in Dates precedes the starting date.
- Values and Dates contain a different number of values.

@XIRR is closely related to XNPV, the net present value function. The rate of return calculated by @XIRR is the interest rate corresponding to XNPV = 0.

Quattro Pro uses calculating @XIRR by iteration. Starting with Guess, it uses a changing rate and repeats the calculation until the result is accurate to 0.000001%. If no result is found after 100 cycles, @XIRR returns ERR. The criterion for solution is:

$$0 = \sum_{i=1}^{N} \frac{P_i}{(1+rate)^{\frac{(d_i - d_1)}{365}}}$$

where

di	the ith payment date
d1	the 0th payment date
Pi	the ith payment

Examples

Suppose an investment of $20,000 made on February 1, 1995, has paid off as follows:

	A	B
1	Cash Flow	Dates
2	($20,000)	1-Feb-95
3	$5,000	1-Aug-95
4	$8,300	15-Nov-95
5	$6,100	15-Feb-96
6	$5,100	1-May-96

The internal rate of return is

@XIRR(A2..A6,B2..B6) = 0.260097 or 26.0097%

You can also enter the function as

@XIRR({-20000,5000,8300,6100, 5100},{34731,34912,35018, 35110,35186})

Related topics

@XNPV - Net Present Value

Syntax

@XNPV(Rate, Values, Dates)

Rate	Discount rate to apply to cash flows.
Values	Series of cash flows; first payment is the one occurring at the beginning of the investment; succeeding payments are discounted based on a 365-day year.
Dates	Payment corresponding to cash flow payments; first date must be the earliest date, but all other dates can be in any order.

@XNPV calculates the net present value of an investment when cash flow is not necessarily periodic. Numbers in Dates are truncated to integers.

@XNPV returns ERR if:

- An argument is non-numeric.
- A number in Dates is not a valid date number.
- A number in Dates precedes the starting date.
- Values and Dates contain a different number of values.

@XNPV is calculated as follows:

$$XNPV = \sum_{i=1}^{N} \frac{P_i}{(1+rate)^{\frac{(d_i - d_1)}{365}}}$$

where

di	the ith payment date
d1	the 0th payment date
Pi	the ith payment

Examples

Suppose an investment of $20,000 made on February 1, 1995, has paid off as follows, with cash flows discounted at 9.5%:

	A	B
1	Cash Flow	Dates
2	($20,000)	1-Feb-95
3	$5,000	1-Aug-95
4	$8,300	15-Nov-95
5	$6,100	15-Feb-96
6	$5,100	1-May-96

The net present value is

@XNPV(0.095,A2..A6,B2..B6) = $2,614.20

You can also enter the function as

@XNPV(0.095,{-20000,5000,8300, 6100,5100},{34731,34912,35018, 35110,35186})

Related topics

@XORB - Binary Exclusive OR

Syntax

@XORB(Binary1, <Binary2>, <Bits>)

Binary1	First binary number.
Binary2	Second binary number.
Bits	Number of binary bits used for both input and output; if omitted, Bits = number of bits in Binary1 or Binary2, whichever is greater; must be in the range 0 <n ≤ 64.

@XORB performs a bit-by-bit logical exclusive OR of each bit in Binary1 and Binary2. Use @XORB to set bits to 1 if the bits being compared in Binary1 and Binary2 are different.

If only one number is specified, then @XORB performs an exclusive OR operation on the bits in Binary1; this parity test, or XOR reduction, returns 1 or 0 based on successive bit comparisons.

Examples

@XORB(10,1) = 11

@XORB(11,10) = 01

@XORB(11) = 0

@XORB(1100,110,5) = 01010

Related topics

@XORH - Hexadecimal Exclusive OR

Syntax

@XORH(Hex1, <Hex2>, <Bits>)

Hex1	First hexadecimal number.
Hex2	Second hexadecimal number.
Bits	Number of binary bits used for both input and output; if omitted, Bits = number of bits in Hex1 or Hex2, whichever is greater; 4 binary digits = 1 hexadecimal digit; must be in the range 0 <n ≤ 64.

@XORH performs a bit-by-bit logical exclusive OR of each bit in Hex1 and Hex2. Use @XORH to set bits to 1 if the bits being compared in Hex1 and Hex2 are different.

If only one number is specified, then @XORH performs the exclusive OR operation on the bits in Hex1; this parity test, or XOR reduction, returns 1 or 0 based on successive bit comparisons.

Examples

@XORH("A","A") = 0

@XORH("E") = 1

@XORH("C","6",8) = 0A

Related topics

@YDAYS - Calendar Days in Year

Syntax

@YDAYS(Year)

Year	Number from 0 (1900) to 199 (2099) or a standard year like 1993.

@YDAYS returns the number of calendar days in a specified year.

The valid date calculation range for this function is 01/01/1900 through 12/31/2099.

Example

@YDAYS(1996) = 366, the number of days in 1996, since it is a leap year.

Related topics

@YDIV - Beginning of Year Division

Syntax

@YDIV(Date, <Numdiv>, <Months>, <Anchor>, <Endmnth>)

Date	Number representing the date about which to calculate year division. See "Using dates and times in Quattro Pro."
NumDiv	Integer representing number of divisions away from beginning of division in which date falls; can be negative.
Months	Number of months per division; does not have to be a divisor of 12; can be longer than 1 year; must be an integer >0 (the default is 3).
Anchor	Date to anchor division boundaries on (the default is January 1, 1900).
EndMnth	1 to indicate adherence to ends of months; 0 to indicate that ends of months are ignored (the default is 1).

@YDIV returns the date of the beginning of the year division (quarter, half-year, or some other fixed period of months, specified by Months) in which Date falls.

If NumDiv is used, @YDIV returns the date that is NumDiv divisions away from the beginning of the division in which Date falls. For example, @YDIV can calculate the beginning of the second quarter after the quarter containing March 21, 1993.

By default, divisions start at January 1, 1900; all other boundaries are some number of divisions away from January 1, 1900. For example, the default quarter boundaries are January 1, April 1, July 1, and October 1 of each year. Use Anchor to specify an alternate start date.

Division cycles can be changed to coincide with any date specified by Anchor. For example, you can change the quarter boundaries to February 23, May 23, August 23, and November 23 by setting Anchor to one of these dates in any year. When the division length is an integral divisor of 12, the year component of Anchor is ignored since the boundary dates repeat each year.

Examples

@YDIV(@DATE(94,8,19)) = 34516 (July 1, 1994), the date of the beginning of the quarter in which August 19, 1994, falls (assuming that quarters begin January 1, April 1, July 1 and October 1).

A bond pays interest every six months until the bond's maturity, at which time it pays the final interest. The next formula calculates the date of the next coupon payment after September 12, 1994 if the bond matures July 15, 1999:

@YDIV(@DATE(94,9,12),1,6,@DATE(99,7,15)) = 34714 (January 15, 1995)

The value 1 (NumDiv), passed as the second argument, designates the beginning of the next division (next coupon payment date). The value 6 (Months), passed as the third argument, designates that divisions are six months long. The fourth argument (Anchor) designates that year divisions must coincide with that date; in this case, that coupon dates coincide with the maturity date.

Related topics

@YEAR - Year Portion of Date Serial Number

Syntax

@YEAR(DateTimeNumber

DateTimeNumber	A numeric value between -109571 (January 1, 1600) and 474816.9999999 (December 31, 3199).

@YEAR returns the year portion of DateTimeNumber. To display the actual year, just add 1900 to the result of @YEAR. If you want to extract the year portion of a string that is in date format, use @DATEVALUE with @YEAR to convert the string into a serial number.

See "Using dates and times in Quattro Pro."

Examples

@YEAR(22222) = 60 (1960)

@YEAR(A6)+1900 = 19nn, where nn is the year value in A6

@YEAR(@DATEVALUE("12-Oct-54")) = 54

@YEAR(-75000) = -206 (1694)

Related topics

@YEARFRAC - Fraction of Year

Syntax

@YEARFRAC(StartDate, EndDate, <Calendar>)

StartDate	Number representing the start date. See "Using dates and times in Quattro Pro."
EndDate	Number representing the end date.
Calendar	Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

@YEARFRAC returns the fraction of the year from a specified starting and ending date covered between StartDate and EndDate. The value returned includes both the starting and ending dates.

Example

This formula calculates the fraction of a year between August 15, 1993 and November 4, 1993:

@YEARFRAC(@DATE(93,8,15),@DATE(93,11,4)) = 0.222222

Related topics

@YIELD - Yield of a Bond

Syntax

@YIELD(Settle, Maturity, Issue, Coupon, Price, <Calendar>)

Settle	Number representing the settlement date.
Maturity	Number representing the maturity date.
Issue	Number representing the issue date.
Coupon	Coupon rate; must be ≥ 0.
Price	Price; must be > 0.
Calendar	Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

@YIELD returns the annual yield of a security that pays periodic interest.

Dates for @YIELD must follow this pattern:

Issue < Settle < Maturity

Example

This formula calculates the annual yield of an 11.75% bond that was issued January 15, 1993 and matures November 15, 2014. The bond was priced at 126.1875 for settlement on January 23, 1993.

@YIELD(@DATE(93,1,23),@DATE(114,11,15),@DATE(93,1,15),0.1175,126.1875) = 0.089877

Related topics

@YIELDDISC - Yield of a Bill

Syntax

@YIELDDISC(Settle, Maturity, Price, <Redemption>, <Calendar>)

Settle	Number representing the settlement date; must be < Maturity.
Maturity	Number representing the maturity date.
Price	Settlement price; must be > 0.
Redemption	Redemption value per 100 face value (must be > 0; the default is 100).
Calendar	Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

@YIELDDISC returns the annualized yield for a discount security. @YIELDDISC uses this formula:

$$Y = \left(\frac{R}{P} - 1\right) * \left(\frac{t_b}{M - S}\right)$$

Y	yield
R	redemption
P	price
b	basis
M	maturity
S	settle

tb is the number of days over which the discount rate applies (360 or 365).

Example

This formula calculates the yield on a bill maturing October 15, 1993 and trading at a price of 97.8 for July 15, 1993 settlement, using an actual/360 calendar:

@YIELDDISC(@DATE(93,7,15),@DATE(93,10,15),97.8,100,2) = 0.088023

Related topics

@YIELDMAT - Yield of a CD

Syntax

@YIELDMAT(Settle, Maturity, Issue, Coupon, Price, <Calendar>)

Settle	Number representing the settlement date; must be < Maturity.
Maturity	Number representing the maturity date.
Issue	Number representing the issue date; must be < Settle.
Coupon	Coupon rate; 0 ≤ Coupon ≤ 1.
Price	Price per 100 face value.
Calendar	Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

@YIELDMAT returns the annual yield of a security that pays interest at maturity.

Example

This formula calculates the annual yield of a security with the following terms: Settle is March 7, 1993, Maturity is November 3, 1993, Issue is November 8, 1992, Coupon is 6.247%, Price is 100, and Calendar is 3 (actual/365).

@YIELDMAT(@DATE(93,3,7),@DATE(93,11,3),@DATE(92,11,8),0.06247,100,3) = 0.06122

Related topics

@YIELDPER - Yield for Securities Paying Periodic Interest

Syntax

@YIELDPER(Settle, Maturity, Coupon, Price, <Redemption>, <Freq>, <Basis>)

Settle	Date number representing the settlement date.
Maturity	Date number representing the maturity date; must be greater than Settle.
Coupon	Coupon rate; must be ≥ 0.
Price	Price per $100 face value; must be > 0.
Redemption	Redemption value per $100 face value (optional); must be ≥ 0; the default is 100.
Freq	Frequency of coupon payments (optional) in number of payments per year; can be 1, 2, 3, 4, or 12; the default is 2.
Basis	Flag specifying which calendar to observe: 0 = US (NASD) 30/360; default if you omit the argument 1 = Actual/actual 2 = Actual/360 3 = Actual/365 4 = European 30/360

@YIELDPER returns the yield for securities paying periodic interest. @YIELDPER uses the following formula:

$$Y = \frac{\left(\frac{R}{100} + \frac{r}{f}\right) - \left(\frac{par}{100} + \left(\frac{A}{E} \times \frac{r}{f}\right)\right)}{\frac{par}{100} + \left(\frac{A}{E} \times \frac{r}{f}\right)} \times \frac{f \times E}{D}$$

where

Y	yield
R	redemption value
r	coupon rate
f	number of coupon payments per year
par	price per $100 face value
A	accrued days from beginning of coupon period to settlement date
E	number of days in coupon period
D	number of days from settlement date to redemption date

You cannot use any optional argument without using all the ones preceding it.

Example

A security with a settlement date of September 1, 1995, and maturity date of March 1, 2001, pays quarterly at a 4.75% coupon rate. The price is $95.00 and the redemption value is $100. Basis is a 30/360 calendar.

@YIELDPER(@DATE(95,9,1), @DATE(101,3,1), 0.0475,95,100,4) = 5.81885 %

Related topics

@YLD2YLD - Convert Yield

Syntax

@YLD2YLD(Y1, F1, F2, <Q1>, <Q2>)

Y1	Specified yield to be converted to another compounding frequency.
F1	Number of times specified yield is compounded per year.
F2	Number of times target yield is compounded per year.
Q1	Number of periods for quoted yields (F1 is the default).
Q2	Number of periods for quoted yields (F2 is the default).

@YLD2YLD converts a yield expressed in one compounding frequency to another. For example, since bond yields are compounded twice per year while mortgage yields are compounded 12 times per year, some conversion is necessary to compare the two different yield figures.

To use @YLD2YLD, you supply Y1, F1, F2 and the optional arguments Q1 and Q2. The result is Y2. @YLD2YLD solves this equation for Y2:

$$\left(1 + \frac{Y1}{Q1}\right)^{F1} = \left(1 + \frac{Y2}{Q2}\right)^{F2}$$

Examples

This formula converts an annual yield of 6.5% into a semiannual yield:

@YLD2YLD(0.065,1,2) = 0.063977

Related topics

@ZTEST - Two-tailed Probability of a z-Test

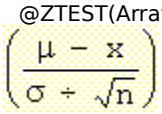
Syntax

@ZTEST(Array, X, <S>)

Array	A numeric array or cells of values.
X	A value to test against the mean of the values in Array.
S	Population standard deviation; if omitted, @ZTEST uses the sample standard deviation.

@ZTEST returns the two-tailed probability of a z-test. @ZTEST calculates a z-score, which is the distance between X and the mean for Array, and then returns the two-tailed probability of the z-score for a normal distribution. Use @ZTEST to test whether a value is drawn from a large sample population.

@ZTEST(Array,x) is equal to 1 minus @NORMDIST times



Example

@ZTEST({10,12,14,17,19,21,22,25},15) = 0.087352

Related topics

Calculating Dates and Times

For calculation purposes, Quattro Pro stores all dates as serial integers beginning with 0 for December 30, 1899. The minimum, -109,571, equals January 1, 1600; the maximum, 474,816, equals December 31, 3199.

Quattro Pro stores times as decimal fractions; 0.000 represents 00:000:00, and 0.99999 represents 23:59:59. To format time expressions in your notebook, right-click the cell you want to format, click **Selection properties**, then click **Numeric format**.

Related topics

Setting Holidays for Date & Time @Functions

Business date functions have three arguments to specify which dates are holidays: Saturday, Sunday, and Holidays. By default, Saturday and Sunday are holidays. Setting the argument Saturday to 1 specifies that Saturday is a business day; setting Sunday to 1 specifies that Sunday is a business day. Use the argument Holidays to specify holidays that do not fall on weekends (unless weekends are used as business days). You can set Holidays to cells containing holiday dates, the date of a single holiday, or 0 to specify no special holidays.

Related topics

Entering Number Conversion @Functions

Input numbers that include non-numeric characters (such as hexadecimal or ASCII values) must be enclosed in quotation marks (for example, "1AF3C"). Numbers that exceed 14 digits, except decimal numbers, must also be enclosed in quotation marks.

For the 64-bit number conversion @functions, numbers must be in the following ranges for each base:

Base	Range
Signed decimal	-9223372036854775808 to +9223372036854775807
Unsigned decimal	0 to 18446744073709551615
Hexadecimal	0000000000000000 to FFFFFFFFFFFFFFFF
Octal	0000000000000000000000 to 1777777777777777777777

If a 64-bit number conversion @function results in a string value greater than 2^{64} - 1, the @function returns ERR.

Related topics

Entering Boolean @Functions

The shift @functions, @SHLB, @SHRB, @SHLH, and @SHRH, shift the bits in a number to the left or right. You can use the shift @functions to perform quick multiplication of integers. Each binary shift to the left is equivalent to multiplying the number by 2. Each binary shift to the right is equivalent to dividing the number by 2.

Binary	Decimal
00011	3
00110	6
01100	12
11000	24

As you shift bits off one end of a number, bits are added to the other end. Shift, however, is not equal to rotate. For example, the binary number 1001 shifted left is 0010. The same number rotated left is 0011. To perform this rotation, you can use a nested bit test @function to set the BitIn argument.

The addition @functions, @ADDB and @ADDH, return the sum of two numbers. The subtraction @functions, @SUBB and @SUBH, return the difference of two numbers.

The overflow @functions, @SHLBO, @SHRBO, @SHLHO, @SHRHO, @ADDBO, @ADDHO, @SUBBO, @SUBHO, return the overflow bit (either 0 or 1) of a shift, addition, or subtraction operation. For example, if you shift the binary number 1001 to the left and maintain four places, the result is 0010. The overflow bit, that is, the bit that has shifted into the fifth place not shown, is 1.

The AND @functions, @ANDB and @ANDH, combine the zeros of the input words. Any bit that is 0 in either number sets the corresponding output bit to 0.

The OR @functions, @ORB and @ORH, combine the ones of the input words. Any bit that is 1 in either number sets the corresponding output bit to 1.

The exclusive OR @functions, @XORB and @XORH, set an output bit to 1 when two corresponding input bits are not equal.

The following table shows the results of AND, OR, and exclusive OR operations of Bit A and Bit B.

Bit A	Bit B	A AND B	A OR B	A XOR B
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

The invert @functions, @INVB and @INVH, invert individual bits of a number; that is, all bits with a value of 1 change to 0, and all bits with a value of 0 change to 1.

The bit manipulation @functions, @BITSB, @BITRB, @BITTB, @BITSH, @BITRH, and @BITTH, let you set or reset bits, or test the value of a bit in a number.

The concatenation @functions, @CATB, @CATH, @CATNB, and @CATNH, link numbers together in a chain. For example, the concatenation of the two binary numbers 11111 and 10101 is 1111110101.

The argument Bits defines the word size (in number of binary bits) for both input and output. The default for Bits is the number of bits in the largest input number. If you perform an operation with two numbers of different word size, the smaller number is left-padded with zeros. If Bits is less than the length of an input value, then the excess most significant digits are truncated. For Boolean @functions using hexadecimal numbers, note that each hexadecimal digit equals 4 bits.

The argument BitIn represents either the binary bit inserted during a shift, the carry bit for addition, or the borrow bit for subtraction; it can be either 0 (the default) or 1.

Tip

To add or subtract negative numbers with the Boolean @functions, use two's complement notation. Two's complement notation is a code used to give meaning to a binary string. The sign bit of two's complement notation is the leftmost bit of the word. A number is positive if the sign bit is 0 and negative if it is 1. Negative integers are converted to two's complement by inverting each bit (that is, changing each 1 to a 0 and each 0 to a 1), and then adding 1.

Related topics

Calendar Conventions

Financial @functions support four different calendar conventions to count the difference in days between two dates. The optional Calendar argument lets you specify which calendar convention to use.

Calendar	Description
30/360	The 30/360 calendar convention assumes all months have 30 days and every year has 360 days. Using the 30/360 calendar, the number of years, months and days between two dates are counted separately. Then, the number of days between two dates is the sum of three quantities: the number of years times 360, the number of months time 30, and the number of days.
Actual/Actual	The Actual/Actual calendar convention considers the actual number of days between two dates and the actual number of days in the year. For example, February 28, 1994 and August 31, 1994 are 184 days apart. February 28, 1994 and March 1, 1994 are 1 day apart.
Actual/360	The Actual/360 calendar convention considers the actual number of days in each month, but assumes 360 days in the year.
Actual/365	The Actual/365 calendar convention considers the actual number of days in each month, but assumes 365 days in the year, thus making no provision for leap year.

Related topics

Annuity @Function Arguments

Argument	Description
Adv	Number of cash flows (payments, deposits) made before the annuity begins.
Fv	Future value.
Int	Interest charged on the loan per period (not per year; many loans quote annual interest rates that must be divided by the number of payments per year).
Nper	Number of periods of the loan or investment (should be an integer greater than 0).
Odd	Number ≥ 0 that specifies the number of periods between the start of a loan and the first payment (for example, if the loan is made two and a half months before the first monthly payment is due, use 2.5)
Payment	Cash flow made each period.
Per	A specified loan or investment period, 1 through Nper.
Pmt	Payment.
Principal	Amount of money loaned or the initial deposit on an annuity that increases principal (like depositing $2,500 to open a savings account)
Pv	Present value.
Rate	Interest rate (should be greater than -1).
Residual	Remaining principal and interest at the end of a loan that the annuity did not take care of
ResOff	Number of periods after the annuity ends before the residual must be paid; express it as a fraction of a period (for example, in a monthly loan, 1.5 means 1.5 months before the residual is due)
Simp	Specifies how the interest is calculated: 0 for compounded interest, 1 for simple interest
Term	The total number of cash flows (payments or deposits) to make
Type	0 if payments are at the end of each period, 1 if at the beginning. This optional argument lets you use financial @functions to compute either an ordinary annuity, where periodic payments are made at the end of each period, or an annuity due, where payments are made at the beginning of each period. Quattro Pro assumes that Type = 0 unless you indicate otherwise.

Non-integer values are allowed for Nper, and the @functions give results that are consistent with other spreadsheet programs, but which are actually not very meaningful. If you borrow money from a bank for, say, 15.2 months with interest paid monthly, giving Nper a value of 15.2 in the financial @functions will only be a rough indicator of what the bank will tell you to pay. In order to compute the figures the way the bank would, you have to consider two transactions, one for 15 months and one for 0.2 months.

The functions assume that there is no residual unless a nonzero value is specified for the optional argument Residual. When a residual is specified, the functions assume that it is paid along with the last payment. When it is not, a positive value should be specified for the optional argument ResOff. For example, if the residual is paid three months after the last monthly payment, ResOff = 3. Compound interest is used during any fractional component of ResOff unless Simp = 1.

Advance payments, specified by Adv, are made on or before the first day of the loan period. They are included in the total payment count. The functions assume zero advance payments unless a nonzero value is specified for the optional argument Odd.

Odd specifies the time period between the beginning of a loan (or issue of an annuity) and the date of the first periodic payment, and does not necessarily constitute exactly one normal payment period. For example, if a loan begins on March 19, 1993 and monthly payments are due the first of every month beginning April 5, 1993, the first payment period is 17 days long. Since the implied normal first payment period, March 5 to April 5, is 31 days

long, Odd = 17/31.

Tip

- In the following @functions, as well as @NPV and @IRR, amounts with positive signs represent money received, and amounts with negative signs represent money paid: @FVAL, @IRATE, @IPAYMT, @NPER, @PAYMT, @PPAYMT, and @PVAL. This convention applies to arguments and to the results of the @functions. In 1-2-3-compatible @functions (such as @PV, @PMT, @FV, @RATE, @TERM, and @CTERM) the amounts are usually all positive regardless of which way the money changes hands.

Related topics

Entering Cash Flow @Functions

Cash flow analysis is a process of listing a stream of cash gains and losses (positive and negative cash flows), modifying them using a percentage or percentages (discount rate(s)), and determining their future value, present value, or rate of growth (or decline; both are called the internal rate of return).

Unlike an annuity, this stream of cash flows does not always occur periodically, and does not have a fixed interest rate for each cash flow.

You can use cash flow functions to estimate the net present value of a cash flow stream, project the future value of the stream, compute the gains the stream is making as a percentage, or compute how the discount rates must change to achieve a specific future value.

For details on using cash flow @function arguments, choose one of the following topics:

Using the Discrate Argument

Using the Filter, Start, and End Arguments

Using the Flows Argument

Using the Odd and Periods Arguments

Using the Simp and PathDep Arguments

Related topics

Using the Flows Argument

In Quattro Pro, a stream of cash flows is specified by a column (or row) of values. The Flows argument of a cash flow function is set to this selection. Positive values add cash to the stream; negative values subtract from it. For example, if your savings account had two deposits of $50, one withdrawal of $25, and one deposit of $75 (in that order), you could use A2..A5 or B2..E2 of the next figure to represent it in a cash flow function:

If the stream contains a series of equal cash flows, you can add an additional column (or row) to specify how many times a specified cash flow repeats. For example, you can replace A2..A5 of the previous figure with A2..B4 of the next figure:

The first column (or row) of the selection specifies how many times each cash flow occurs. For example, in the previous figure the value 2 (in A2) specifies that two cash flows of $50 occur in the stream, not one.

Tip

- Quattro Pro uses the size of the cash flow cells to determine whether you are specifying a column of cash flows or a row of cash flows. It assumes that selections with more than two rows contain cash flows in the second column; selections with more than two columns contain cash flows in the second row. In the case of a two-column, two-row selection, Quattro Pro assumes that the cash flows are in the second row.

Related topics

Using the Filter, Start, and End Arguments

You can use the argument, Filter, Start, and End to make Quattro Pro automatically exclude cash flows that do not fall in a certain range, such as all deposits, or any withdrawals less than $20. Excluded cash flows are not included in the function calculations. Use Filter to specify the rules for exclusion, as shown in the next table.

Filter	Cash flows are excluded when
0	No filtering (the default)
1	Cash flow < Start
2	Cash flow ≤ Start
3	Cash flow > Start
4	Cash flow ≥ Start
5	Start < Cash flow < End
6	Start ≤ Cash flow ≤ End

As shown, Start and End are used differently, depending on the setting of Filter. They always bind the cash flows in some way; Start and End could be a range of cash flows values to use (Filter set to 5) or an upper limit for values (Filter set to 1, Start set to the upper limit).

Related topics

Using the Discrate Argument

The Discrate argument of a cash flow function specifies how the cash flows are discounted to achieve their future or net present value. It can be a single percentage (like 0.05 for 5%) that applies to all the cash flows, or a column (or row) of discount rates, one for each cash flow in the Flows cells (see the previous section). Positive discount rates decrease the cash flow; negative ones increase it. The next figure shows a stream of cash flows (in A2..B4) and their corresponding discount rates (in C2..C4).

The first two cash flows (specified by A2..B2) are discounted by 5% (as specified by C2). The third is discounted by -2.5% (an increase, as specified by the negative percentage in C3), and the final cash flow is discounted by 7.5% (as specified by C4).

Related topics

Using the Simp and PathDep Arguments

You can use the Simp argument to specify how Quattro Pro applies discount rates to cash flows. The next table shows the discounting methods available.

Simp	Discounting
0	Compounded
1	Mixture of compounded and simple
2	Simple

In addition to Simp, PathDep, which is used only when Discrate is a selection, specifies whether path-dependent discounting is used. When path-dependent discounting is used (PathDep is set to 1), the set of discount rates are chained together to determine future or net present value. If the order of discount rates changes, the future or net present value can be affected.

When path-dependent discounting is not used (PathDep is set to 0, the default), each cash flow is affected by its associated discount rate; other discount rates in Discrate do not affect it.

Related topics

Using the Odd and Periods Arguments

By default, cash flow functions assume that each cash flow occurs periodically (every month, every year, and so on). The arguments Odd and Periods let you specify irregular periods. You normally use one or the other, so these arguments appear in the function descriptions as Odd|Periods.

If the length of time of the first period is odd, specify a number for Odd. For example, if a series of cash flows are monthly, and the first period is half a month long, set Odd to 0.5; if the first period is one and a half months, set Odd to 1.5.

If several cash flows are unevenly spaced, specify cells for Periods. Periods is a column (or row) of numbers that specify the duration of each cash flow in the Flows cells. Like Odd, each value in the cells is expressed as a fraction of the regular period. For example, the next figure shows a cash flow cells in B2..B4, and a Period selection in A2..A4.

The value 1 in A2 specifies that the first cash flow ($50) occurs at a regular period. You decide what this period is; it could be a week, a month, or a year. Assuming the regular period is a month, the value 3.5 (in A3) specifies that the second cash flow ($75) occurs three and a half months after the first. The final value, 2, specifies that two months elapse between the second cash flow and the third.

Like Flows, the Periods selection can have an additional column (or row) added to specify how many times a specified period length repeats. Periods does not have to be the same size as Flows. For example, in the next figure, the cash flow stream is A2..B5.

Periods is C2..C4, and specifies that the first cash flow is 0.56745 periods away, the next 11 cash flows occur one period apart, and the last four cash flows are 1.5 periods apart.

Tip

- In @FUTV, Odd specifies the length of the last period.

Related topics

Bond @Function Arguments

Argument	Description
Settle	Settlement date for the trade
Maturity	Redemption date for the bond
Coupon	Annual coupon rate expressed as a decimal
Issue	Issue date, that is, the date at which the bond is first offered for sale and begins accruing interest
FirstCpn	Date on which the first coupon period ends; if the first coupon period is longer or shorter than the other periods, the first coupon payment date is explicitly specified at issue; the size of the first coupon payment is linearly prorated in accordance with the length of the first coupon period
Redemption	Redemption value per par of 100
Freq	Frequency of coupon payments in the number of payments per year; the default is 2 (semiannual)
Calendar	Calendar to observe; the default is 0 (30/360)
Yield	Internal rate of return expressed as a decimal
Price	Quoted price of the bond assuming a par value of 100 and not including accrued interest
LastCpn	Date on which the last coupon period ends

Related topics

Quattro Pro Functions List

A B C D E F G H I J K L M N
O P Q R S T U V W X Y Z Lists

@

A

ABDAYS
ABS
ACCRINT
ACCRINTM
ACCRINTXL
ACCRUED
ACDAYS
ACOS
ACOSH
ACOT
ACOTH
ACSC
ACSCH
ADDB
ADDBO
ADDH
ADDHO
ADDRESS
AMAINT
AMINT
AMNTHS
AMPMT
AMPMTI
AMPRN
AMRES
AMRPRN
AMTERM
AND
ANDB
ANDH
ARRAY
ASCTOHEX
ASEC
ASECH
ASIN
ASINH
ATAN
ATAN2
ATANH
AVEDEV
AVG

B

BASE
BDAYS
BESSELI
BESSELJ
BESSELK
BESSELY
BETA
BETADIST
BETAI
BETAINV
BINOMDIST
BINTOHEX
BINTOHEX64
BINTONUM
BINTONUM64
BINTOOCT
BINTOOCT64
BITRB
BITRH
BITSB
BITSH
BITTB
BITTH
BLOCKNAME
BLOCKNAME2
BLOCKNAMES
BLOCKNAMES2
BUSDAY

C

CATB
CATH
CATNB
CATNH
CDAYS
CEILING
CELL
CELLINDEX
CELLPOINTER
CHAR
CHIDIST
CHIINV
CHITEST
CHOOSE
CLEAN
CODE
COLS

COLUMN
COMB
COMMAND
COMPLEX
CONCATENATE
CONFIDENCE
CONVERT
CORREL
COS
COSH
COT
COTH
COUNT
COUNTBLANK
COUNTIF
COUPDAYBS
COUPDAYS
COUPDAYSNC
COUPNCD
COUPNUM
COUPPCD
COVAR
CRITBINOM
CSC
CSCH
CTERM
CUMIPMT
CUMPRINC
CURVALUE

D

D360
DATE
DATEDIF
DATEINFO
DATEVALUE
DAVG
DAY
DAYS360
DB
DCOUNT
DDB
DDELINK
DEGREES
DELTA
DEVSQ
DFRAC
DGET
DISC

DMAX
DMIN
DOLLAR
DOLLARDE
DOLLARFR
DOLLARTEXT
DPRODUCT
DPURECOUNT
DSTD
DSTDS
DSUM
DURAT
DURATION
DVAR
DVARS

E

EFFECT
EMNTH
EOMONTH
ERF
ERFC
ERFD
ERR
EVEN
EXACT
EXP
EXP2
EXPONDIST

F

FACT
FACTDOUBLE
FACTLN
FALSE
FBDAY
FDIST
FEETBL
FIB
FIELD
FILEEXISTS
FIND
FINV
FIRSTBLANKPAGE
FIRSTINGROUP
FISHER
FISHERINV
FIXED
FLOOR

FORECAST
FRACD
FRACTION
FREQDIST
FTEST
FULLP
FUTV
FV
FVAL

G

GAMMA
GAMMADIST
GAMMAINV
GAMMALN
GAMMAP
GAMMAQ
GCD
GEOMEAN
GEOSUM
GESTEP
GETGROUP
GETREGISTRYKEY
GRANDTOTAL123
GROWTH

H

HALFP
HARMEAN
HEXTOASC
HEXTOBIN
HEXTOBIN64
HEXTONUM
HEXTONUM64
HEXTOOCT
HEXTOOCT64
HLOOKUP
HOLS
HOUR
HYPGEOMDIST

I

IF
IMABS
IMAGINARY
IMARGUMENT
IMCONJUGATE
IMCOS

IMDIV
IMEXP
IMLN
IMLOG2
IMLOG10
IMPOWER
IMPRODUCT
IMREAL
IMSIN
IMSQRT
IMSUB
IMSUM
INDEX
INDEXTOLETTER
INT
INTERCEPT
INTRATE
INTXL
INVB
INVH
IPAYMT
IRATE
IRR
ISBDAY
ISBLANK
ISBLOCK
ISERR
ISEVEN
ISLEGALPAGENAME
ISLOGICAL
ISNA
ISNONTEXT
ISNUMBER
ISODD
ISSTRING

J

No Functions

K

KANSUUJI
KURT

L

LARGEST
LASTBLANKPAGE
LASTCELLVALUE
LASTINGROUP

LBDAY
LCM
LEFT
LENGTH
LETTERTOINDEX
LINEST
LINTERP
LLDEC
LN
LOG
LOGBASE
LOGCONV
LOGEST
LOGINV
LOGNORMDIST
LOOKUP
LOWER
LWKDAY

M

MATCH
MAX
MAXLOOKUP
MDAYS
MDET
MDURATION
MEDIAN
MEMAVAIL
MEMEMSAVAIL
MID
MIN
MINLOOKUP
MINUTE
MINVERSE
MIRR
MMULT
MNTHS
MOD
MODE
MODULO
MONTH
MROUND
MTGACC
MULT
MULTINOMIAL

N

N
NA

NBDAY
NEGBINOMDIST
NENGO
NETPV
NETWORKDAYS
NOMINAL
NORMDIST
NORMINV
NORMSDIST
NORMSINV
NOT
NOW
NPER
NPV
NUMTOBIN
NUMTOBIN64
NUMTOHEX
NUMTOHEX64
NUMTOOCT
NUMTOOCT64
NWKDAY

O

OCTTOBIN
OCTTOHEX
OCTTONUM
ODD
ODDFPRICE
ODDFYIELD
ODDLPRICE
ODDLYIELD
OFFSET
OR
ORB
ORH

P

PAGEINDEX
PAGEINDEX2
PAGENAME
PAGENAME2
PAGENAMES
PAGENAMES2
PAYMT
PBDAY
PEARSON
PERCENTILE
PERCENTRANK
PERMUT

PI
PIRATE
PMT
PMTC
POISSON
POWER
PPAYMT
PRICE
PRICEDISC
PRICEMAT
PROB
PROPER
PROPERTY
PUREAVG
PURECOUNT
PUREMAX
PUREMIN
PURESTD
PURESTDS
PUREVAR
PUREVARS
PV
PVAL

Q

QUARTILE
QUOTIENT

R

RADIANS
RAND
RANDBETWEEN
RANK
RATE
RECEIVED
REGRESSION
REPEAT
REPLACE
RIGHT
ROMAN
ROOTN
ROUND
ROUNDDOWN
ROUNDDOWNXL
ROUNDUP
ROUNDUPXL
ROW
ROWS
RSQ

S

S
SCMARG
SEC
SECH
SECOND
SEMEAN
SERIESSUM
SETSTRING
SHEETS
SHLB
SHLBO
SHLH
SHLHO
SHRB
SHRBO
SHRH
SHRHO
SIGN
SIN
SINH
SKEW
SLN
SLOPE
SMALLEST
SPLINE
SQRT
SQRTPI
STANDARDIZE
STD
STDS
STEC
STEYX
STKOPT
STRCMPNORM
STRING
SUBB
SUBBO
SUBH
SUBHO
SUBSTITUTE
SUBTOTAL
SUBTOTAL123
SUM
SUMIF
SUMNEGATIVE
SUMPOSITIVE
SUMPRODUCT

SUMSQ
SUMX2MY2
SUMX2PY2
SUMXMY2
SUMXPY2
SUMXY
SUMXY2
SUUJI
SYD

T

TABLELINK
TAN
TANH
TBILLEQ
TBILLPRICE
TBILLYIELD
TDIST
TERM
TIME
TIMEVALUE
TINV
TODAY
TOTAL
TRANSPOSE
TREND
TRIM
TRIMMEAN
TRUE
TRUNC
TTEST
TYPE

U

UPPER
USESPLINE

V

VALUE
VAR
VARS
VDB
VERSION
VHLOOKUP
VLOOKUP

W

WEEKDAY

WEEKNUM
WEIBULL
WEIGHTAVG
WKDAY
WORKDAY

X

XCOUNT
XINDEX
XIRR
XNPV
XORB
XORH

Y

YDAYS
YDIV
YEAR
YEARFRAC
YIELD
YIELDDISC
YIELDMAT
YIELDPER
YLD2YLD

Z

ZTEST

Quattro Pro Functions Categories

Lists

Database functions

Date and Time functions

Engineering functions

Financial functions

Logical functions

Mathematical functions

Miscellaneous functions

Statistical functions

String functions

Database spreadsheet functions

The database spreadsheet functions are similar to the statistical spreadsheet functions. Instead of taking a simple list of values, these spreadsheet functions all take three arguments, as in @DMAX(*Block*,*Column*,*Criteria*).

@DAVG	The average (mean) of all numeric values in specified cells.
@DCOUNT	The number of nonblank cells in specified cells.
@DGET	A value or label from cells.
@DMAX	The largest numeric or latest date value in specified cells.
@DMIN	The smallest numeric or earliest date value in specified cells.
@DPRODUCT	The product of values in specified cells.
@DPURECOUNT	A count of all numeric entries in specified cells.
@DSTD	The population standard deviation of all values in specified cells.
@DSTDS	The sample standard deviation of all values in specified cells.
@DSUM	The total of all numeric values in specified cells.
@DVAR	The population variance of all values in specified cells.
@DVARS	The sample variance of all values in specified cells.

Related topics

Date and Time spreadsheet functions

The date and time spreadsheet functions calculate dates and times, perform calculations involving business days, or convert location coordinates.

@ABDAYS	Adds (or subtracts) a specified number of business days to a specified date.
@ACDAYS	Adds a specified number of calendar days to a specified date.
@AMNTHS	Adds a specified number of months to a specified date.
@BDAYS	Returns the number of business days between two dates, inclusive of the second date.
@BUSDAY	Returns a specified date if it is a business day, or the closest business day before (or after) the date.
@CDAYS	Returns the number of calendar days between two dates, inclusive of the second date.
@D360	Returns the number of days between two dates, based on a 360-day year (twelve 30-day months).
@DATE	Returns the date number for a specified year, month, and day; 0 = December 30, 1899.
@DATEDIF	Calculates the number of years, months, or days between two dates.
@DATEINFO	Returns information (for example, day of week or month of year) about a date number.
@DATEVALUE	Returns the date number for a specified formatted date string.
@DAY	Returns the day of the month (1-31) represented by a specified date/time serial number.
@DAYS360	Returns the number of days between two dates based on a 360-day year (twelve 30-day months).
@EMNTH	Returns the date of the last day of the month in which a specified date falls.
@EOMONTH	Returns the serial date number for the last day of the month a specified number of months before or after a start date.
@FBDAY	Returns the date of the first business day of a month in which a specified date falls.
@HOLS	Returns the number of holidays between two dates, excluding holidays that fall on weekends.
@HOUR	Returns the number of hours past midnight (0-23) represented by a specified date/time serial number.
@ISBDAY	Returns 1 if the specified date is a business day, or 0 if it is not.
@LBDAY	Returns the date of the last business day of a month in which a specified date falls.
@LWKDAY	Returns the date of the last specified weekday in a specified month.
@MDAYS	Returns the number of calendar days in a specified month of a specified year.
@MINUTE	Returns the number of minutes past the hour (0-59) represented by a specified date/time serial number.
@MNTHS	Returns the number of whole months between two dates.
@MONTH	Returns the month in number form (1-12) represented by a specified date/time serial number.
@NBDAY	Returns the date of the first valid business day after a specified date.
@NENGO	Converts a date to its kanji representation
@NETWORKDAYS	Returns the number of days from a start date through an end date, excluding weekends and holidays.
@NOW	Returns the date and time serial number for the current system date and time.
@NWKDAY	Returns the date of the nth occurence of a specified weekday in a specified month.
@PBDAY	Returns the date of the first valid business day before a specified date.
@SECOND	Returns the number of seconds past the minute (0-59) represented by a specified date/time serial number.
@TIME	Returns the time number for a specified hour, minute, and second. The hour is a numeric value between 0 and 23.
@TIMEVALUE	Returns the time number for a specified formatted time string.
@TODAY	Returns the date number for the current system date.
@WEEKDAY	Returns a number (from 1 for Saturday to 7 for Friday) representing the day Date falls on
@WEEKNUM	Calculates in which week of the year a specified date falls.
@WKDAY	Returns a number (from 1 for Sunday to 7 for Saturday) representing the day Date falls on.

@WORKDAY	Returns the serial number for a date that is a specified number of days before or after a specified date, optionally excluding weekends and/or holidays.
@YDAYS	Returns the number of calendar days in a specified year.
@YDIV	Returns the date of the beginning of the year division in which a specified date or specified number of divisions always falls.
@YEAR	Returns the year number (-300 to 1299; 0 = 1900) represented by a specified date/time serial number.
@YEARFRAC	Returns the year fraction representing the number of whole days between a specified starting and ending date.

Related topics

Engineering spreadsheet functions

Bessel

These spreadsheet functions return values that satisfy the Bessel equation or the modified Bessel equation. Bessel functions have various applications in physics and engineering.

Boolean

These spreadsheet functions handle applications of digital logic and bitwise operations, which involve the testing, setting, or shifting of actual bits in a number.

Complex number

These spreadsheet functions convert or modify a complex number (a number whose square is a negative real number).

Miscellaneous

These spreadsheet functions have various applications in engineering, including converting values to different measures, testing the relationship of two numeric values, and returning error functions.

Number conversion

These spreadsheet functions convert a value from one number system to another.

Related topics

Bessel Engineering spreadsheet functions

@BESSELI	Returns the modified Bessel function In(x).
@BESSELJ	Returns the Bessel function Jn(x).
@BESSELK	Returns the modified Bessel function Kn(x).
@BESSELY	Returns the Bessel function Yn(x).

Related topics

Boolean Engineering spreadsheet functions

@ADDB	Returns the sum of two binary numbers.
@ADDBO	Returns the overflow bit of the sum of two binary numbers.
@ADDH	Returns the sum of two hexadecimal numbers.
@ADDHO	Returns the overflow bit of the sum of two hexadecimal numbers.
@ANDB	Returns the AND operation of two binary numbers.
@ANDH	Returns the AND operation of two hexadecimal numbers.
@BITRB	Returns a binary number with a specified bit reset to 0.
@BITRH	Returns a hexadecimal number with a specified bit reset to 0.
@BITSB	Returns a binary number with a specified bit set to 1.
@BITSH	Returns a hexadecimal number with a specified bit set to 1.
@BITTB	Returns the value of a specified bit in a binary number.
@BITTH	Returns the value of a specified bit in a hexadecimal number.
@CATB	Returns the concatenation of two binary numbers.
@CATH	Returns the concatenation of two hexadecimal numbers.
@CATNB	Returns the concatenation of *n* binary numbers.
@CATNH	Returns the concatenation of *n* hexadecimal numbers.
@INVB	Returns the inverse of a binary number.
@INVH	Returns the inverse of a hexadecimal number.
@ORB	Returns the OR operation of binary numbers.
@ORH	Returns the OR operation of hexadecimal numbers.
@SHLB	Returns a binary number shifted left by a specified number of bits.
@SHLBO	Returns the overflow bit of a binary number after it has been shifted left one bit.
@SHLH	Returns a hexadecimal number shifted left by a specified number of bits.
@SHLHO	Returns the overflow bit of a hexadecimal number after it has been shifted left one bit.
@SHRB	Returns a binary number shifted right by a specified number of bits.
@SHRBO	Returns the overflow bit of a binary number after it has been shifted right one bit.
@SHRH	Returns a hexadecimal number shifted right by a specified number of bits.
@SHRHO	Returns the overflow bit of a hexadecimal number after it has been shifted right one bit.
@SUBB	Returns the difference of two binary numbers.
@SUBBO	Returns the overflow bit of the subtraction of one binary number from another.
@SUBH	Returns the difference of two binary numbers.
@SUBHO	Returns the overflow bit of the subtraction of one hexadecimal number from another.
@XORB	Returns the exclusive OR operation of binary numbers.
@XORH	Returns the exclusive OR operation of hexadecimal numbers.

Related topics

Complex Number Engineering spreadsheet functions

@COMPLEX	Converts real & imaginary coefficients into a complex number.
@IMABS	Returns the distance from the origin on a complex plane for a complex number.
@IMAGINARY	Returns the imaginary coefficient of a complex number.
@IMARGUMENT	Returns the argument theta, an angle expressed in radians.
@IMCONJUGATE	Returns the complex conjugate of a complex number.
@IMCOS	Returns the cosine of a complex number.
@IMDIV	Returns the quotient of two complex numbers.
@IMEXP	Returns the exponential of a complex number.
@IMLN	Returns the natural logarithm of a complex number.
@IMLOG10	Returns the base-10 logarithm of a complex number.
@IMLOG2	Returns the base-2 logarithm of a complex number.
@IMPOWER	Returns a complex number raised to a complex power.
@IMPRODUCT	Returns the product of two complex numbers.
@IMREAL	Returns the real coefficient of a complex number.
@IMSIN	Returns the sine of a complex number.
@IMSQRT	Returns the square root of a complex number.
@IMSUB	Returns the difference of two complex numbers.
@IMSUM	Returns the sum of complex numbers.

Related topics

Miscellaneous Engineering spreadsheet functions

@DELTA	Tests whether two numbers are equal.
@ERF	Returns the error function.
@ERFC	Returns the complementary function.
@ERFD	Returns the derivative of the error function.
@GAMMA	Calculates the gamma distribution function.
@GESTEP	Tests whether a number is greater than a threshold value.
@SPLINE	Returns a polynomial fitted piecewise to pass through a specified set of points.
@USESPLINE	Returns, for any specified x value, the corresponding y value, specified the original t and y arrays and the coefficient array produced by @SPLINE.

Related topics

Number Conversion Engineering spreadsheet functions

@ASCTOHEX	Returns the hexadecimal string equivalent of an ASCII value
@BASE	Converts a base-10 number into another base.
@BINTOHEX	Converts a binary number to hexadecimal.
@BINTOHEX64	Converts a binary number (up to 64 bits) to hexadecimal.
@BINTONUM	Converts a binary number to decimal.
@BINTONUM64	Converts a binary number (up to 64 bits) to decimal.
@BINTOOCT	Converts a binary number to octal.
@BINTOOCT64	Converts a binary number (up to 64 bits) to octal.
@CONVERT	Converts a number from one measurement system to another.
@HEXTOASC	Converts a hexadecimal number to ASCII.
@HEXTOBIN	Converts a hexadecimal number to binary.
@HEXTOBIN64	Converts a hexadecimal number (up to 64 bits) to binary.
@HEXTONUM	Converts a hexadecimal number to decimal.
@HEXTONUM64	Converts a hexadecimal number (up to 64 bits) to decimal.
@HEXTOOCT	Converts a hexadecimal number to octal.
@HEXTOOCT64	Converts a hexadecimal number (up to 64 bits) to octal.
@NUMTOBIN	Converts a decimal number to binary.
@NUMTOBIN64	Converts a decimal number (up to 64 bits) to binary.
@NUMTOHEX	Converts a decimal number to hexadecimal.
@NUMTOHEX64	Converts a decimal number (up to 64 bits) to hexadecimal.
@NUMTOOCT	Converts a decimal number to octal.
@NUMTOOCT64	Converts a decimal number (up to 64 bits) to octal.
@OCTTOBIN	Coverts an octal number to binary.
@OCTTOHEX	Converts an octal number to hexadecimal.
@OCTTONUM	Converts an octal number to decimal.

Related topics

Financial spreadsheet functions

Annuity

The investment spreadsheet functions involve a series of periodic payments over a term measured in the number of payment periods. This set of spreadsheet functions allows you to compute one value, knowing three of the other values.

Bill

These spreadsheet functions compute values for Treasury billls.

Bond

These spreadsheet functions compute values for bonds.

Cash Flow

These spreadsheet functions operate on tables of data that record income and expenditures.

CD

These spreadsheet functions compute values for certificates of deposit.

Depreciation

These spreadsheet functions compute depreciation over time.

Stock

These spreadsheet functions compute values for common stock.

Related topics

Annuity Financial spreadsheet functions

@AMAINT	Calculates the accumulated interest paid on an amortized loan after n payments.
@AMINT	Calculates the periodic interest rate for an amortized loan.
@AMPMT	Calculates the periodic payment for an amortized loan.
@AMPMTI	Calculates the interest portion of the nth periodic payment of an amortized loan.
@AMPRN	Calculates the initial principal of an amortized loan.
@AMRES	Calculates the end value of an amortized loan or the future value of an annuity.
@AMRPRN	Calculates the remaining balance of an amortized loan after n payments.
@AMTERM	Calculates the length of an amortized loan, expressed as number of payments.
@CTERM	Returns the number of compounding time periods required to achieve a specified future value, specified the present value and interest rate.
@CUMIPMT	Returns the cumulative interest paid on a loan between specified periods or in a single period.
@CUMPRINC	Returns the cumulative principal paid on a loan between specified periods or in a single period.
@EFFECT	Calculates the effective annual interest rate for a specified nominal rate and number of compounding periods a year.
@FV	Returns the future value of an investment at the end of the term, specified the payment, interest rate, and number of payments.
@FVAL	Returns the future value of an investment, specified the interest rate, number of payments, periodic payment rate, and optional present value and type.
@IPAYMT	Returns the interest portion of a single payment specified the interest rate, period, number of periods, present value, and optional future value and type.
@IRATE	Returns the interest rate specified the number of payments, payment amount, present value, and optional future value and type.
@MTGACC	Returns the new loan term, the payoff-date, or interest saved by paying extra monthly principal for a home loan.
@NOMINAL	Calculates the nominal annual interest rate for a specified effective rate and number of compounding periods a year.
@NPER	Returns the number of periods specified the interest rate, payment amount, present value, and optional future value and type.
@PAYMT	Returns periodic payments specified the rate, number of payments, present value, and optional future value and type.
@PMT	Returns the periodic payment required to fully amortize the principal during the term, specified present value, interest rate, and number of payments.
@PMTC	Calculates monthly payments based on semi-annual interest compounding commonly used in Canada.
@PPAYMT	Returns the principal portion of a single payment.
@PV	Returns the present value of an investment, specified periodic payment, interest rate, and number of periods.
@PVAL	Returns the present value of an investment, specified the interest rate, number of payments, periodic payment rate, and optional future value and type.
@RATE	Returns the interest rate required to achieve a specified future value, specified the future value, present value, and number of payments.
@TERM	Returns the number of periodic payments required to achieve a specified future value, specified the periodic payment amount and interest rate.
@YLD2YLD	Converts a yield expressed in one compounding frequency and time length to that in another frequency and/or time length.

Related topics

Bill Financial spreadsheet functions

@DISC	Returns the discount rate for a security.
@INTRATE	Returns the simple annualized yield for a fully invested security.
@PRICEDISC	Returns the price per 100 face value of a security that pays periodic interest.
@RECEIVED	Returns the amount received at maturity for a fully invested security.
@TBILLEQ	Returns the bond equivalent yield for a Treasury bill.
@TBILLPRICE	Returns the price per 100 face value for a Treasury bill.
@TBILLYIELD	Returns the yield for a Treasury bill.
@YIELDDISC	Returns the annual yield for a discounted security.

Related topics

Bond Financial spreadsheet functions

@ACCRINT	Returns the accrued interest for a bond.
@ACCRINTXL	Returns the accrued interest for a bond. (Excel version)
@ACCRUED	Returns the accrued interest for a bond. (Lotus-123 version)
@COUPDAYBS	Returns the number of days from the beginning of the coupon period to the settlement date.
@COUPDAYS	Returns the number of days in the coupon period that contains the settlement date.
@COUPDAYSNC	Returns the number of days from the settlement date to the next coupon date.
@COUPNCD	Returns the next coupon date after the settlement date.
@COUPNUM	Returns the number of coupons payable between the settlement date and maturity date.
@COUPPCD	Returns the previous coupon date before the settlement date.
@DURATION	Returns the Macaulay duration of a security with par value of 100.
@MDURATION	Returns the modified Macauley duration for a security with an assumed par value of 100.
@ODDFPRICE	Returns the price per 100 face value of a security with an odd first period.
@ODDFYIELD	Returns the yield of a security with an odd first period.
@ODDLPRICE	Returns the price per 100 face value of a security with an odd last period.
@ODDLYIELD	Returns the yield of a security with an odd last period.
@PRICE	Returns the price per 100 face value of a security that pays periodic interest.
@YIELD	Returns the yield on a security that pays periodic interest.
@YIELDPER	Returns the yield for securities paying periodic interest.

Related topics

Cash Flow Financial spreadsheet functions

@DURAT	Calculates the Macaulay duration of a cash flow stream.
@FUTV	Calculates the future value of a cash flow stream.
@IRR	Returns the internal rate of return of an investment.
@MIRR	Calculates the modified internal rate of return on an investment consisting of payments made at regular intervals.
@NETPV	Calculates net present value of a stream of cash flows.
@NPV	Returns the net present value of a future cash flow.
@PIRATE	Calculates the internal rate of return for a stream of cash flows.
@SCMARG	Calculates the discount scenario margin, the margin to add to each discount rate in order to arrive at a specified net present value.
@XIRR	Returns the internal rate of return on an investment when cash flow is not necessarily periodic.
@XNPV	Calculates the net present value of an investment when cash flow is not necessarily periodic.

Related topics

CD Financial spreadsheet functions

@ACCRINTM	Returns the accrued interest for a security that pays interest at maturity.
@PRICEMAT	Returns the price per 100 face value of a security that pays interest at maturity.
@YIELDMAT	Returns the annual yield of a security that pays interest at maturity.

Related topics

Depreciation Financial spreadsheet functions

@DB	Calculates the depreciation of an asset over a specified period using the fixed-declining balance method.
@DDB	Returns the double-declining balance depreciation of an asset during the specified period.
@SLN	Returns the straight-line depreciation of an asset over each period in its specified useful life.
@SYD	Returns the sum-of-the-years'-digits depreciation of an asset during the specified period.
@VDB	Calculates depreciation allowance using the variable-rate declining balance method.

Related topics

Stock Financial spreadsheet functions

@DOLLARDE	Converts a fractional price into dollars.
@DOLLARFR	Converts a dollar price into a fractional price.
@FEETBL	Returns fee calculations for stock transactions.
@STKOPT	Returns the time value and earnings value of a stock option.

Related topics

Logical spreadsheet functions

@AND	Returns 1 (true) if all arguments are true, 0 (false) if even one argument is false.
@FALSE	Always returns the logical value 0.
@FILEEXISTS	Checks to see whether the named file exists. Returns a logical true (1) or false (0) value.
@IF	Evaluates a specified condition, and returns the specified expression if it is true, or another specified expression if it is false.
@ISBLANK	Tests a specified cell to see if it is empty.
@ISBLOCK	Tests input to see if it is a defined cell name or valid cell coordinates.
@ISERR	Returns 1 if a specified cell contains ERR (error indicator), otherwise 0.
@ISEVEN	Returns 1 (true) if a specified number is even, 0 (false) if it is odd.
@ISLOGICAL	Returns 1 (true) if its argument refers to a 1 or 0; it returns 0 (false) if its argument refers to any other number.
@ISNA	Returns 1 if a specified cell is NA (not available), otherwise 0.
@ISNONTEXT	Returns 1 (true) if its argument refers to any item that is not text. @ISNONTEXT also returns 1 if Value refers to an empty cell.
@ISNUMBER	Returns 1 if a specified cell contains a number, otherwise 0.
@ISODD	Returns 1 (true) if a specified number is odd, 0 (false) if it is even.
@ISSTRING	Returns 1 if a specified cell contains a label or text string, otherwise 0.
@NOT	Reverses the value of its argument; for example if the expression is FALSE, @NOT returns TRUE; if the expression is TRUE, @NOT returns FALSE.
@OR	Returns 1 (true) if any argument is true, 0 (false) only if all arguments are false.
@TRUE	Always returns the logical value 1.

Related topics

Mathematical spreadsheet functions

Mathematical spreadsheet functions take one or more numeric values as arguments, and return a numeric value. You must enter all angles in radians for @COS, @SIN, and @TAN. Accordingly, @ASIN, @ATAN, and @ATAN2 return all angles in radians. To convert radians to degrees, use @DEGREES; to convert degrees to radians, use @RADIANS.

@ABS	Returns the absolute value of a specified number.
@ACOS	Returns the angle whose cosine is a specified value.
@ACOSH	Returns the arc, or inverse, hyperbolic cosine of a number.
@ACOT	Calculates the arc, or inverse, cotangent using the cotangent X of an angle.
@ACOTH	Calculates the arc, or inverse, hyperbolic cotangent using the hyperbolic cotangent X of an angle.
@ACSC	Calculates the arc, or inverse, cosecant using the cosecant X of an angle.
@ACSCH	Calculates the arc, or inverse, hyperbolic cosecant using the hyperbolic cosecant X of an angle.
@ASEC	Calculates the arc, or inverse, secant using the secant X of an angle.
@ASECH	Calculates the arc, or inverse, hyperbolic secant using the hyperbolic secant X of an angle.
@ASIN	Returns the angle whose sine is a specified value.
@ASINH	Calculates the arc, or inverse, hyperbolic sine using the hyperbolic sine X of an angle.
@ATAN	Returns the angle whose tangent is a specified value.
@ATAN2	Returns the angle represented by a specified pair of coordinates.
@ATANH	Calculates the arc, or inverse, hyperbolic tangent using the hyperbolic tangent X.
@CEILING	Rounds a number up to the nearest integer.
@COS	Returns the cosine of a specified angle.
@COSH	Calculates the hyperbolic cosine of the angle X.
@COT	Calculates the cotangent of angle X.
@COTH	Calculates the hyperbolic cotangent of X.
@CSC	Returns the cosecant of angle X, in radians.
@CSCH	Calculates the hyperbolic cosecant of X.
@DEGREES	Converts a specified value from radians to degrees.
@DFRAC	Converts a decimal number to a whole number and fractional component.
@EVEN	Rounds a number up to the nearest even integer.
@EXP	Returns the exponential of a specified value, which is the value of *e* (the mathematical constant) raised to the power of the specified value. The value must be less than or equal to 709.
@EXP2	Calculates the value of the constant e raised to the power (-X^2).
@FACT	Calculates the factorial of a number.
@FACTDOUBLE	Returns the double factorial of a number.
@FACTLN	Returns the natural logarithm of the factorial of n.
@FIB	Calculates the nth term of a Fibonacci sequence.
@FLOOR	Rounds a number down, toward zero.
@FRACD	Converts a number with a fractional component to a decimal.
@GCD	Calculates the greatest comman divisor of x and y.
@GEOSUM	Calculates the geometric series that is sum of the terms of a geometric sequence of a number of terms (n) based on the first term and common ratio.
@INT	Returns the integer portion of of a specified value. In this @function, the number is simply truncated, not rounded off.
@INTXL	Rounds the number x down to an integer value.
@LCM	Calculates the least common multiple of x and y.
@LINTERP	Performs linear interpolation between sets of xy pairs.
@LN	Returns the natural logarithm of a specified value. This is the logarithm of the number to the base *e*; @LN is the inverse of @EXP (the value must be greater than 0).
@LOG	Returns the logarithm of a specified value to base 10 (the value must be greater than 0).

@LOGBASE	Calculates the logarithm of a specified number to the specified base.
@LOGCONV	Converts a specified logarithm from one specified base to another.
@MDET	Calculates the determinant of a matrix.
@MINVERSE	Returns the inverse matrix for a matrix stored in a square array.
@MMULT	Calculates the matrix product of two arrays.
@MOD	Returns the modulus of a specified value with respect to another. (Modulus is the remainder when the first is divided by the second.)
@MODULO	Returns the remainder, or modulus, of x/y.
@MROUND	Returns a number rounded to the desired multiple.
@MULT	Calculates cumulative product of a set of numbers.
@MULTINOMIAL	Returns the multinomial of a set of numbers.
@ODD	Rounds a number up to the nearest odd integer.
@PI	Returns the value of pi.
@POWER	Calculates the result of a specified number raised to a power.
@QUOTIENT	Returns the integer portion of a division.
@RADIANS	Converts a specified value from degrees to radians.
@RAND	Returns a fractional random number between 0 and 1.
@RANDBETWEEN	Returns a random number between the numbers you specify.
@ROMAN	Returns the Roman numeral corresponding to a specified Arabic numeral, displaying it as text.
@ROOTN	Calculates the nth root of a specified number.
@ROUND	Rounds off a specified value (with up to 15 digits) to a specified number of decimal places.
@ROUNDDOWN	Rounds a positive number down with a specified precision and rounds a negative number the direction you specify.
@ROUNDDOWNXL	Rounds positive and negative numbers toward zero.
@ROUNDUP	Rounds a positive number up with a specified precision and rounds a negative number the direction you specify.
@ROUNDUPXL	Rounds positive and negative numbers toward zero.
@SEC	Returns the secant of angle X, in radians.
@SECH	Calculates the hyperbolic secant of X.
@SERIESSUM	Returns the sum of a power series based on a formula.
@SIGN	Returns 1 if X is positive, 0 if X is zero, and -1 if X is negative.
@SIN	Returns the sine of a specified angle.
@SINH	Returns the hyperbolic sine of X, in radians.
@SQRT	Returns the square root of a specified value (the value must be greater than or equal to 0).
@SQRTPI	Returns the square root of a number multiplied by pi.
@TAN	Returns the tangent of a specified angle.
@TANH	Calculates the hyperbolic tangent of X.
@TRANSPOSE	Returns the transpose of cells.
@TRUNC	Truncates a number to the precision you specify.

Related topics

Miscellaneous spreadsheet functions

Attribute

Each cell has a number of attributes, including its address, contents, type, protection status, format, etc. These spreadsheet functions return the attributes of a specified cell. If cells are specified, they use the top left cell in the cells.

Cell and Table

These spreadsheet functions generally return simple information from a cell or cells.

Status

Return the current setting for commands, properties, and other elements.

Table Lookup

These spreadsheet functions are used to search for a value in cells that has been specified as a data table.

Related topics

Cell Attribute spreadsheet functions

The attribute spreadsheet functions return the requested attribute of a cell (or of the upper left cell in specified cells).

@CELL	Returns the requested attribute of the upper left cell in specified cells.
@CELLINDEX	Returns the requested attribute of the cell at the intersection of a specified column and row in cells.
@CELLPOINTER	Returns the requested attribute of the active (currently selected) cell.

Related topics

Miscellaneous Cell and Table spreadsheet functions

@@	Returns the contents of a specified cell.
@ADDRESS	Returns, as text, a cell reference for which you specify row and column numbers.
@ARRAY	Returns the result of an expression (a formula or @function) using array syntax.
@BLOCKNAME	Returns the name of specified cells.
@BLOCKNAME2	Returns the cell name created in a specified notebook for specified cells.
@BLOCKNAMES	Returns a two-column table showing the cell names that intersect with specified cells.
@BLOCKNAMES2	Returns a two-column table showing the cell names created in a specified notebook that intersect with specified cells.
@CHOOSE	Returns the element from a specified list in the specified position.
@COLS	Returns the number of columns in specified cells.
@COLUMN	Returns the column number(s) for a cell or cells.
@COUNTBLANK	Counts blank cells in specified cells.
@DDELINK	Creates a "live" data link from another Windows application that supports DDE (Dynamic Data Exchange).
@ERR	Always returns ERR (the error indicator).
@FIRSTBLANKPAGE	Returns the sheet letters for the first unnamed blank sheet in a notebook that is not part of a group.
@FIRSTINGROUP	Returns the sheet letters for the first sheet in the specified group.
@GETGROUP	Returns the name of the group that contains the sheet for specified cells, or the name of the group in the specified cells that contains a specified sheet name.
@GETREGISTRYKEY	Returns the contents of the specified key in the Windows Registry.
@INDEXTOLETTER	Returns a one- or two-character string for the index number of a sheet or column.
@ISLEGALPAGENAME	Returns 1 if the specified sheet name is valid (whether it exists or not); otherwise, returns 0.
@LASTBLANKPAGE	Returns the sheet letters for the last unnamed blank sheet in a notebook that is not part of a group.
@LASTCELLVALUE	Returns the contents of the last cell that is not blank in the specified cells.
@LASTINGROUP	Returns the sheet letters for the last sheet in the specified group.
@LETTERTOINDEX	Returns the index number for column letters or sheet letters.
@LLDEC	Converts latitude and longitude coordinates to decimals.
@NA	Always returns NA (not available).
@PAGEINDEX	Returns the index number for a notebook sheet with a specified name.
@PAGEINDEX2	Returns the index number for a notebook sheet with a specified name in a specified notebook.
@PAGENAME	Returns the name of a notebook sheet with a specified index number.
@PAGENAME2	Returns the name of a notebook sheet with a specified index number in a specified notebook.
@PAGENAMES	Returns a two-column table showing the sheet letters and corresponding sheet names for the active notebook.
@PAGENAMES2	Returns a two-column table showing the sheet letters and corresponding sheet names for a specified notebook.
@ROW	Returns the row number(s) for a cell or cells.
@ROWS	Returns the number of rows in specified cells.
@SHEETS	Returns the number of sheets in specified cells.
@TYPE	Returns a code indicating the type (for example, number or formula) of a specified value.

Related topics

Status spreadsheet functions

@COMMAND	Returns current settings for the specified command equivalent.
@CURVALUE	Returns the current value of a specified menu command setting.
@MEMAVAIL	Returns the number of bytes of available memory.
@MEMEMSAVAIL	Under DOS, returns the number of bytes of available expanded memory; under Windows, returns NA (included for compatibility with Quattro Pro for DOS).
@PROPERTY	Returns current property settings for the specified object and property.
@VERSION	Returns the version number of Quattro Pro.

Related topics

Table Lookup spreadsheet functions

@FIELD	Returns the nth substring in a delimited list of strings.
@HLOOKUP	Searches for a specified value in the first row of specified cells. Returns the value a specified number of rows from the first.
@INDEX	Returns the value at the intersection of a specified column and row in specified cells.
@LOOKUP	Looks up values in a specified row or column.
@MATCH	Returns the position of the cell whose contents match specified cell contents within cells.
@MAXLOOKUP	Returns the address of the cell containing the largest value in specified cells or list of cells.
@MINLOOKUP	Returns the address of the cell containing the smallest value in specified cells or list of cells.
@OFFSET	returns a reference that is offset from another reference by a specified number of rows and columns; dimensions of the offset can also be specified.
@TABLELINK	Establishes a link to an external database table and displays the table in a Quattro Pro notebook.
@VHLOOKUP	Searches for a specified value in the first row of specified cells. Returns the value from a specified number of rows from the first
@VLOOKUP	Searches for a specified value in the first column of specified cells. Returns the value from a specified number of columns from the first.
@XINDEX	Returns the contents of a cell located at the intersection of a specified column, row, and (optionally) notebook sheet.

Related topics

Statistical spreadsheet functions

The statistical spreadsheet functions perform aggregation, counting, and analysis operations on a group of values expressed as a list (or lists) of one or more arguments. These arguments can be numeric values or cell values.

Descriptive

These spreadsheet functions return a value that helps you summarize and describe a group of values.

Inferential

These spreadsheet functions return a value (or values) that helps you draw conclusions about a group (or groups) of values.

Related topics

Descriptive Statistical spreadsheet functions

@AVG	Returns the average (mean) of all numeric values in a list.
@COUNT	Returns the number of nonblank cells in a list.
@COUNTIF	Count Matching Cells
@FREQDIST	Calculates a frequency distribution, displaying it as a vertical array.
@GEOMEAN	Returns the geometric mean of all numeric values in a list.
@GRANDTOTAL123	Sums all cells in a designated area that contain @SUBTOTAL123 in their formulas.
@GROWTH	Fits an exponential curve to data, then predicts further y-values on that curve for a specified array of x-values.
@HARMEAN	Returns the harmonic mean of all numeric values in a list.
@KURT	Returns the kurtosis (peakedness or flatness) of a data set.
@LARGEST	Returns the k-th largest value in a data set.
@LINEST	Uses the "least squares" method to calculate a straight line that best fits your data and returns an array to describe the line.
@LOGEST	Calculates an exponential curve that fits your data and returns an array to describe the curve.
@MAX	Returns the largest numeric or last date value in a list.
@MEDIAN	Returns the median of a data set.
@MIN	Returns the smallest numeric or earliest date value in a list.
@MODE	Returns the most common value in a data set.
@PERCENTILE	Returns the value from a group of values at a specified percentile.
@PERCENTRANK	Returns the percentage rank of a value in a data set.
@PUREAVG	Calculates the average of values in a list, ignoring blank cells and labels.
@PURECOUNT	Returns the number of entries and cells in a list, excluding blank cells and labels.
@PUREMAX	Returns the largest numeric value in a list, ignoring blank cells and labels.
@PUREMIN	Returns the smallest numeric value in a list, ignoring blank cells and labels.
@PURESTD	Returns the population standard deviation (square root of the population variance) of numeric values in a list, ignoring blank cells and labels.
@PURESTDS	Returns the sample standard deviation (square root of the sample variance) of numeric values in a list, ignoring blank cells and labels.
@PUREVAR	Calculates the population variance of numeric values in a list, ignoring blank cells and labels.
@PUREVARS	Calculates the sample population variance of numeric values in a list, ignoring blank cells and labels.
@QUARTILE	Returns the quartile of a data set.
@RANK	Returns the rank of a number in a list of numbers.
@REGRESSION	Rerforms a multiple linear regression, returning the specified statistic.
@SEMEAN	Returns the standard error of the sample mean for values in specified cells.
@SKEW	Returns the skewness of a distribution.
@SMALLEST	Returns the k-th smallest value in a data set.
@STANDARDIZE	Returns a normalized value.
@STD	Returns the population standard deviation of all values in a list.
@STDS	Returns the sample standard deviation of all values in a list.
@SUBTOTAL	Returns a subtotal in a list or database.
@SUBTOTAL123	Adds the values in a list or cell reference.
@SUM	Returns the total of all numeric values in a list.
@SUMIF	Adds those cells that meet specified criteria. An optional Sum Range may be specified to limit criteria consideration and sum inclusion to particular cells within the cells.
@SUMNEGATIVE	Sums only negative values in a block or list. It ignores blank cells and labels.
@SUMPOSITIVE	Sums only positive values in cells or list. It ignores blank cells and labels.
@TOTAL	Returns the total of all numeric values in a list or reference, excluding any subtotals.

@TREND	Fits a straight line to data, using the "least squares" method, then predicts further y-values on that line for a specified array of x-values.
@TRIMMEAN	Returns the mean of all numeric values in a list with a fraction of values excluded.
@VAR	Returns the population variance of all values in a list.
@VARS	Returns the sample variance of all values in a list.
@WEIGHTAVG	Returns a weighted average of the values in specified cells.
@XCOUNT	Returns the number of non-blank cells in a list.

Related topics

Inferential Statistical spreadsheet functions

@AVEDEV	Performs the average of the absolute deviations of data points from their means.
@BETA	Returns the beta function.
@BETADIST	Returns the cumulative beta probability density function.
@BETAI	Returns the incomplete beta function.
@BETAINV	Returns the inverse of the cumulative beta probability density function.
@BINOMDIST	Returns the binomial probability mass function.
@CHIDIST	Returns the cumulative chi-square distribution.
@CHIINV	Returns the inverse of the cumulative chi-square distribution.
@CHITEST	Computes the probability that the actual and expected frequencies are similar by chance (chi-square test).
@COMB	Calculates the number of unordered subgroups of specified size in a group.
@CONFIDENCE	Returns the confidence interval for a population mean.
@CORREL	Returns the correlation coefficient of two data sets.
@COVAR	Returns the covariance of two data sets.
@CRITBINOM	Returns the smallest value for which the cumulative binomial distribution is less than or equal to a criterion value.
@DEVSQ	Returns the sum of the squares of the deviations.
@EXPONDIST	Returns the exponential distribution.
@FDIST	Returns the F distribution function.
@FINV	Returns the inverse of the cumulative F distribution function.
@FISHER	Returns the Fisher transformation.
@FISHERINV	Returns the inverse of the Fisher transformation.
@FORECAST	Returns a value along a linear trend.
@FTEST	Returns the result of the F-test.
@GAMMADIST	Returns the gamma distribution function.
@GAMMAINV	Computes the inverse of the cumulative Gamma distribution function.
@GAMMALN	Returns the natural logarithm of the gamma function.
@GAMMAP	Returns the incomplete gamma function.
@GAMMAQ	Returns the complement of the incomplete gamma function.
@HYPGEOMDIST	Returns the hypergeometric distribution.
@INTERCEPT	Returns the intercept of the linear regression line.
@LOGINV	Returns the inverse of the lognormal distribution.
@LOGNORMDIST	Returns the lognormal distribution.
@NEGBINOMDIST	Returns the negative binomial distribution.
@NORMDIST	Returns the normal cumulative distribution.
@NORMINV	Computes the inverse of the cumulative normal distribution function.
@NORMSDIST	Computes the standard normal cumulative distribution.
@NORMSINV	Returns the inverse of the standard normal cumulative distribution.
@PEARSON	Returns the Pearson product moment correlation coefficient.
@PERMUT	Calculates the number of ordered subgroups of specified size in a group (permutations).
@POISSON	Returns the Poisson probability distribution.
@PROB	Returns the probability that values in a range are between two limits.
@RSQ	Returns the square of the coefficient of correlation of the linear regression line through data points in known xs and known ys.
@SLOPE	Returns the slope of the linear regression line.
@STEC	Returns the standard error of the regression coefficient.
@STEYX	Standard error of the predicted y-value for each x.
@SUMPRODUCT	The dot (scalar) product of the vectors corresponding to cells.
@SUMSQ	Returns the sum of the squares of the arguments.

@SUMX2MY2	Returns the sum of the differences of the squares of the corresponding values in two arrays.
@SUMX2PY2	Returns the sum of the sum of the squares of corresponding values in two arrays.
@SUMXMY2	Returns the sum of squares of differences of corresponding values in two arrays.
@SUMXPY2	Returns the sum of the squares of corresponding values in two arrays.
@SUMXY	Sum of the products of the corresponding numbers in two arrays.
@SUMXY2	Sum of the product of values and the squares of the corresponding numbers in two arrays.
@TDIST	Returns the Student's t-distribution.
@TINV	Returns the inverse of the Student's t-distribution.
@TTEST	Returns the probability associated with the Student's t-test.
@WEIBULL	Returns the Weibull distribution.
@ZTEST	Returns the two-tailed probability value of a z-test.

Related topics

String spreadsheet functions

String spreadsheet functions work on strings of characters, or text. They include one or more strings as arguments, and return either a string or a numerical value.

@CHAR	Returns the ANSI character that corresponds to the decimal code specified as the argument.
@CLEAN	Returns a specified string with any non-printable ASCII codes removed.
@CODE	Returns the ANSI code of the first character in a specified string.
@CONCATENATE	Links several items together.
@DOLLAR	Converts a numeric value to text, using currency format.
@DOLLARTEXT	Converts a numeric value to text, using cardinal number format.
@EXACT	Returns 1 if two specified strings are identical (including capitalization), otherwise returns 0.
@FIND	Looks for a specified substring in a specified string, beginning with the character in the specified position. Returns the position of the first matched character in the string.
@FIXED	Rounds a number to a specified number of decimals, formats it, and displays the result as text.
@FRACTION	Converts a number with a decimal component to a fraction.
@FULLP	Converts a half-width character string to a full-width character string.
@HALFP	Converts a full-width character string to a half-width character string.
@KANSUUJI	Converts a kanji number to its arabic representation.
@LEFT	Returns a specified number of characters from the beginning of a specified string.
@LENGTH	Returns the length of a specified string, including spaces.
@LOWER	Returns a specified string with all the alphabetic characters converted to lowercase (small letters).
@MID	Returns a specified number of characters from a specified string, starting with the character in the specified position.
@N	Returns the numeric value of the top left cell of cells.
@PROPER	Returns a specified string with the first letter of each word capitalized, and with all other letters lowercase.
@REPEAT	Returns a string made up of a specified number of repetitions of a specified string.
@REPLACE	Deletes a specified number of characters in a specified string, beginning with a specified position, and replaces them with a specified string. Returns the modified string.
@RIGHT	Returns a specified number of characters from the end of a specified string.
@S	Returns the string value of the top left cell of cells.
@SETSTRING	Returns a label as long as the number of characters you specify.
@STRCOMPNORM	Compares two strings using half-width/full-width normalization.
@STRING	Converts a specified numeric value into a string, rounding to a specified number of decimal places.
@SUBSTITUTE	Returns a copy of a specified text string, substituting new text for old text.
@SUUJI	Converts an arabic number to its kanji representation based on given style.
@TRIM	Returns a specified string without leading or trailing spaces, and without multiple spaces.
@UPPER	Returns a specified string with all of the alphabetic characters converted to uppercase (capital letters).
@VALUE	Returns the numeric value of a specified string. (Returns ERR if the argument is not a simple number string.)

Related topics

Number representing the date to which a number of business days should be added.

Integer representing number of business days to add; can be negative.

Cells containing dates that are holidays or the date of a single holiday or 0 to indicate no holidays (the default is 0).

0 to specify that Saturday is not a business day; 1 to specify that Saturday is a business day (the default is 0).

0 to specify that Sunday is not a business day; 1 to specify that Sunday is a business day (the default is 0).

A numeric value.

Number representing the settlement date.

Number representing the maturity date.

Coupon rate; $0 \leq \text{Coupon} \leq 1$.

Number representing the issue date.

Number representing the first coupon date.

Par value (the default is 1000).

Frequency of coupon payments in the number of payments per year (can be 1, 2, 3, 4, 6, or 12; the default is 2).

Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

Number representing the issue date; must be < Settle.

Number representing the settlement date.

Coupon rate; $0 \leq \text{Coupon} \leq 1$.

Par value (the default is 1000).

Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

Number representing the issue date. (Issue, FirstInt, Settle, Freq, and Basis are truncated to integers.)

Number representing the first interest date

Number representing the settlement date.

Interest rate; $0 \leq$ Coupon.

Par value (the default is 1000).

Number of coupon payments per year. For annual paymnets, frequency = 1; for semiannual, frequency = 2; for quarterly, freguency = 4.

Specifies which calendar to use as a basis (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365, 4 = European 30/360; the default is 0).

Number representing the settlement date; must be greater than Issue.

Number representing the issue date.

Number representing the first interest date; must be greater than Issue.

Coupon rate; can be any positive value, including 0.

Par value, or the principal to be paid at maturity (optional); the default is 100.

Frequency of coupon payments (optional) in number of payments per year; can be 1, 2, 4, or 12; the default is 2.

Flag specifying which calendar to observe:
0 = US (NASD) 30/360; default if you omit the argument
1 = Actual/actual
2 = Actual/360
3 = Actual/365
4 = European 30/360

Number representing the date to add days to.

Integer representing number of days to add; can be negative.

Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

1 to indicate adherence to ends of months; 0 to indicate that ends of months are ignored (the default is 1).

A numeric value between -1 and 1.

The hyperbolic cosine of an angle. X must be greater than or equal to 1 but less than approximately 1.34078E+154.

The cotangent of an angle. X can be any value from approximately -1.789E+308 through 1.789E+308.

The hyperbolic cotangent of an angle. X can be any value between approximately -1.79E+308 and less than -1 and between greater than 1 and approximately 1.79E+308.

The cosecant of an angle. X can be any value between approximately -1.79E+308 and -1 and between 1 and approximately 1.79E+308.

The hyperbolic cosecant of an angle. X can be any value between approximately -1.34078E+154 and 1.34078E+154, but not 0.

First binary number.

Second binary number.

Input carry bit; can be either 0 (the default) or 1.

Number of binary bits used for both input and output; if omitted, Bits = number of bits in Binary1 or Binary2, whichever is greater; must be ≤ 64.

First binary number.

Second binary number.

Input carry bit; can be either 0 (the default) or 1.

Number of binary bits used for input; if omitted, Bits = number of bits in Binary1 or Binary2, whichever is greater; must be ≤ 64.

First hexadecimal number.

Second hexadecimal number.

Input carry bit; can be either 0 (the default) or 1.

Number of binary bits used for both input and output; if omitted, Bits = number of bits in Hex1 or Hex2, whichever is greater; must be ≤ 64.

First hexadecimal number.

Second hexadecimal number.

Input carry bit; can be either 0 (the default) or 1.

Number of binary bits used for input; if omitted, Bits = number of bits in Hex1 or Hex2, whichever is greater; 4 binary digits = 1 hexadecimal digit; must be ≤ 64.

Row number of the cell for which you want the reference.

Column number.

Type of cell reference to return (optional).

Logical value indicating A1 or R1C1 reference style (optional):
TRUE= A1 (default, if omitted)
FALSE= R1C1

Name of notebook sheet (optional).

Initial loan principal.

0 to specify compounded interest or 1 to specify simple interest (the default is 0).

Periodic interest rate.

Term of the loan, expressed in number of total payments.

Number of payments made; must be an integer from 0 to Term.

Part of (n+1)th period passed (must be from 0 to 1; the default is 0).

Remaining balance on loan at end of loan term (the default is 0).

Number of periods after last periodic payment that residual is to be paid; can have fractional component (the default is 0).

Number of advance payments made at loan inception (the default is 0); n - Adv must be an integer.

Number of periods between loan inception and date of first payment (not including advance payments); can have fractional component (the default is 1).

Initial loan principal.

Term of loan, expressed in number of total payments.

Periodic payment (for example, if Term is expressed in months, Payment must be a monthly payment).

Remaining balance on loan at end of loan term (the default is 0).

Number of periods after last periodic payment that Residual is to be paid; can have fractional component (the default is 0).

Number of advance payments made at loan inception (the default is 0).

Number of periods between loan inception and date of first payment (not including advance payments); can have fractional component (the default is 1).

0 to specify compounded interest or 1 to specify simple interest (the default is 0).

Required precision of result (the default is 0.000001); must be ≥ 0.

Number representing the date to add number of months to.

Integer representing number of months to add; can be negative.

1 to indicate adherence to ends of months; 0 to indicate that ends of months are ignored (the default is 1).

Initial loan principal.

Periodic interest rate (for example, if Term is expressed in months, Int must be a monthly rate).

Term of loan, expressed in number of total payments.

Remaining balance on loan at end of loan term (the default is 0).

Number of periods after last periodic payment that residual is to be paid; can have fractional component (the default is 0).

Number of advance payments made at loan inception (the default is 0).

Number of periods between loan inception and date of first payment (not including advance payments); can have fractional component (the default is 1).

0 to specify compounded interest or 1 to specify simple interest (the default is 0).

Initial loan principal.

Periodic interest rate (for example, if Term is expressed in months, then Int must be a monthly rate).

Term of loan, expressed in number of total payments.

Number of payments made; must be an integer from 0 to Term.

Remaining balance on loan at end of loan term (the default is 0).

Number of periods after last periodic payment that residual is to be paid; can have fractional component (the default is 0).

Number of advance payments made at loan inception (the default is 0).

Number of periods between loan inception and date of first payment (not including advance payments); can have fractional component (the default is 1).

0 to specify compounded interest or 1 to specify simple interest (the default is 0).

Periodic interest rate (for example, if Term is expressed in half-years, Int must be a semiannual rate).

Term of loan, expressed in number of total payments.

Periodic payment.

Remaining balance on loan at end of loan term (the default is 0).

Number of periods after last periodic payment that residual is to be paid; can have fractional component (the default is 0).

Number of advance payments made at loan inception (the default is 0).

Number of periods between loan inception and date of first payment (not including advance payments); can have fractional component (the default is 1).

0 to specify compounded interest or 1 to specify simple interest (the default is 0).

Initial loan principal.

Periodic interest rate (for example, if Term is expressed in months, then Int must be a monthly rate).

Term of loan, expressed in number of total payments.

Periodic payment.

Number of periods after last periodic payment that Residual is to be paid; can have fractional component (the default is 0).

Number of advance payments made at loan inception (the default is 0).

Number of periods between loan inception and date of first payment (not including advance payments); can have fractional component (the default is 1).

0 to specify compounded interest or 1 to specify simple interest (the default is 0).

Initial loan principal.

0 to specify compounded interest or 1 to specify simple interest (the default is 0).

Periodic interest rate (for example, if Term is expressed in years, then Int must be a yearly rate).

Term of loan, expressed in number of total payments.

Number of payments made; must be an integer from 0 to Term.

Part of (n+1)th period passed; $0 \leq \text{Part} \leq 1$ (the default is 0).

Remaining balance on loan at end of loan term (the default is 0).

Number of periods after last periodic payment that residual is to be paid; can have fractional component (the default is 0).

Number of advance payments made at loan inception (the default is 0).

Number of periods between loan inception and date of first payment (not including advance payments); can have fractional component (the default is 1).

Initial loan principal.

Periodic interest rate (for example, if term is expressed in quarters, Int must be a quarterly rate).

Periodic payment.

Remaining balance on loan at end of loan term (the default is 0).

Number of periods after last periodic payment that residual is to be paid; can have fractional component (the default is 0).

Number of advance payments made at loan inception (the default is 0).

Number of periods between loan inception and date of first payment (not including advance payments); can have fractional component (the default is 1).

0 to specify compounded interest or 1 to specify simple interest (the default is 0).

True-or-false conditions to test.

First binary number.

Second binary number.

Number of binary bits used for both input and output; if omitted, Bits = number of bits in Binary1 or Binary2, whichever is greater; must be ≤ 64.

First hexadecimal number.

Second hexadecimal number.

Number of binary bits used for both input and output; if omitted, Bits = number of bits in Hex1 or Hex2, whichever is greater; 4 binary digits = 1 hexadecimal digit; must be ≤ 64.

Formula or @function using array syntax; @functions can be nested, that is, you can have more than one @function in a single statement.

Number of columns in the output range, including the column of the current cell (the default Columns depends on dimensions of input array(s) in Expression).

Number of rows in the output range, including the row of the current cell (the default Rows depends on dimensions of input array(s) in Expression).

ASCII character string to convert; can be up to 20 ASCII characters.

Number of characters to return; can be from 1 to 40 characters.

The secant of an angle. X can be any value from approximately -1.789E+308 through -1 and from 1 through approximately 1.789E+308.

The hyperbolic secant of an angle. X must be greater than 0 and less than or equal to 1.

A numeric value between -1 and 1.

The hyperbolic sine of an angle. X can be any value from approximately -1.34078E+154 to 1.34078E+154.

A numeric value.

A numeric value.

A numeric value.

The hyperbolic tangent of an angle. X must be greater than -1 and less than 1.

A single cell address that contains another cell address or cell name that is written as a label.

One or more numeric or cell values.

One or more numeric or cell values.

Any decimal value to convert.

Indicates the target base in which to express Decimal; can be any integer from 2 to 36, inclusive (the default is 16).

Indicates the number of desired digits after the decimal point; can be any integer from 0 to 15, inclusive (the default is 0).

Number representing the start date.

Number representing the end date.

Cells containing dates that are holidays or the date of a single holiday or 0 to indicate no holidays (the default is 0).

0 to specify that Saturday is not a business day; 1 to specify that Saturday is a business day (the default is 0).

0 to specify that Sunday is not a business day; 1 to specify that Sunday is a business day (the default is 0).

Numeric value at which to evaluate the function.

Number ≥ 0 representing the order of the Bessel function; if n is not an integer, it is truncated to an integer.

Numeric value at which to evaluate the function.

Number ≥ 0 representing the order of the Bessel function; if n is not an integer, it is truncated to an integer.

Numeric value at which to evaluate the function; must be > 0.

Integer ≥ 0 representing the order of the Bessel function; if n is not an integer, it is truncated to an integer.

Nonnegative numeric value at which to evaluate the function.

Integer ≥ 0 representing the order of the Bessel function; if n is not an integer, it is truncated to an integer.

α parameter to the function; must be > 0.

β parameter to the function; must be > 0.

Value at which to evaluate the function over the interval $A \leq X \leq B$.

α distribution parameter; if W = 0, Z > 0.

β distribution parameter; if Z = 0, W > 0.

Optional lower bound to the interval of X (the default is 0); A cannot equal B and must be $\leq$ X.

Optional upper bound to the interval of X (the default is 1); B cannot equal A and must be ≥ X.

α parameter to the function; if W = 0, Z > 0.

β parameter to the function; if Z = 0, W > 0.

Value at which to evaluate the function; cannot exceed 1.

Cumulative probability value; $0 \leq \text{Prob} \leq 1$.

α parameter to the Beta distribution; must be > 0.

β parameter to the Beta distribution; must be > 0.

Optional lower bound to the interval of X (the default is 0); A cannot equal B and must be $\leq$ X.

Optional upper bound to the interval of X (the default is 1); B cannot equal A and must be ≥ X.

Number of successes in number of trial runs; must be ≥ 0.

Number of independent trial runs in sample; must be > Successes.

Probability of a success on each trial run; must be ≥ 0 and ≤ 1.

1 to return the cumulative distribution function; 0 to return the probability that there are exactly Successes successes.

Binary number to convert; denote negative numbers using a minus sign.

Binary number to convert; must be positive.

Number of characters to return; must be ≤ 16.

Binary number to convert; denote negative numbers using a minus sign.

Binary number to convert.

1 if the most significant bit of Binary is a sign bit; 0 (the default) if Binary is positive.

Binary number to convert; denote negative numbers using a minus sign.

Binary number to convert; must be positive.

Number of characters to return; must be ≤ 22.

Binary number.

Bit position; must be ≥ 0 and ≤ number of bits in Binary - 1.

Hexadecimal number.

Bit position; must be ≥ 0 and ≤ number of bits in Hex - 1.

Binary number.

Bit position; must be ≥ 0 and ≤ number of bits in Binary - 1.

Hexadecimal number.

Bit position; must be ≥ 0 and ≤ number of bits in Hex - 1.

Binary number.

Bit position; must be ≥ 0 and ≤ number of bits in Binary - 1.

Hexadecimal number.

Bit position; must be ≥ 0 and ≤ number of bits in Hex - 1.

Cell or cell reference (for example, A1 or B1..B5).

A reference to a sheet, cell, or cells in another notebook (for example, [BUDGET]A:A1).

Cell or cell reference (for example, A1 or B1..B5).

Cell or cell reference (for example, A1 or B1..B5).

A reference to a sheet, cell, or cells in another notebook (for example, [BUDGET]A:A1).

Cell or cell reference (for example, A1 or B1..B5).

Number representing a date.

Flag specifying direction of adjustment; 0 = forward; 1 = backward; 2 = forward if in same month as Date, otherwise backward (the default is 0).

Cells containing dates that are holidays or the date of a single holiday or 0 to indicate no holidays (the default is 0).

0 to specify that Saturday is not a business day; 1 to specify that Saturday is a business day (the default is 0).

0 to specify that Sunday is not a business day; 1 to specify that Sunday is a business day (the default is 0).

First binary number.

Highest bit of the first number to use for concatenation; the default is the most significant bit.

Lowest bit of the first number to use for concatenation; the default is 0.

Second binary number.

Highest bit of the second number to use for concatenation; the default is the most significant bit.

Lowest bit of the second number to use for concatenation; the default is 0.

Number of binary digits to return; must be ≤ 64.

First hexadecimal number.

Highest bit of the first number to use for concatenation; the default is the most significant bit.

Lowest bit of the first number to use for concatenation; the default is 0.

Second hexadecimal number.

Highest bit of the second number to use for concatenation; the default is the most significant bit.

Lowest bit of the second number to use for concatenation; the default is 0.

Number of equivalent binary digits to return; 4 binary digits = 1 hexadecimal digit; must be ≤ 64.

Number of binary numbers being concatenated; $n \leq 64$.

First binary number.

Second through the nth binary numbers.

Number of binary digits to return; must be ≤ 64.

Number of hexadecimal numbers being concatenated; $n \leq 16$.

First hexadecimal number.

Second through the nth hexadecimal numbers.

Number of equivalent binary digits to return; 4 binary digits = 1 hexadecimal digit; must be ≤ 64.

Number representing the start date.

Number representing the end date.

Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

0 to use 30-day treatment of February for 30/360 calendar; 1 to use the actual-day treatment (the default is 0).

Value to round.

Value to make rounded x evenly divisible by.

Any one of the attributes listed for @CELL.

A cell reference or name.

Any one of the attributes listed for @CELL.

A cell reference or name.

The number of the referenced column, from 0 to 255 (the first column in Block = 0, the second = 1, and so on).

The number of the referenced row; if an offset, the first row in Block = 0, the second = 1, and so on.

The number of the referenced sheet, from 0 to 255 (the first sheet in Block = 0, the second = 1, and so on).

Any one of the attributes listed for @CELL.

A numeric value between 1 and 255.

Value at which to evaluate the function; must be ≥ 0.

Integer number of degrees of freedom in the distribution; must be ≥ 1.

Cumulative probability value; must be ≥ 0 and ≤ 1.

Integer number of degrees of freedom; must be ≥ 1.

Cells containing actual values.

Cells containing expected values.

A positive integer equal to or less than the number of items in List - 1.

One or more numeric or string values, cell addresses, and cell references or names, separated by commas.

A string value.

A string value.

A cell reference or name.

The cell or cells for which you want the column number(s).

Number of elements in each subgroup selected from group N; must be ≥ 0.

Number of elements in the group; must be ≥ 0.

A Quattro Pro command equivalent; to display a list, press Shift+F3 and choose Command Equivalents.

Numeric value representing real coefficient of complex number.

Numeric value representing imaginary coefficient of complex number.

One or more values to link together; can be labels, numbers, or cell references.

Significance level; the percentage of the normal curve that is outside the confidence interval (1 - Alpha); for example, if the confidence interval is 95%, Alpha = 5%; must be > 0 and < 1.

Population standard deviation; must be > 0.

Sample size; must be ≥ 1.

Numeric value in FromUnit to convert, in the units specified by FromUnit.

Unit type of the value X (must be on the list of supported unit names).

Units to convert the value X into; must be on the list of supported unit names.

First array of numeric values.

Second array of numeric values.

A numeric value.

Any value from approximately -710.47558 to approximately 710.47558.

An angle measured in radians. X can be any value from approximately -9.00719E+15 through 9.00719E+15.

Any value from approximately -708.39599 through 708.39599, but not 0.

One or more numeric or string values, cell addresses, and cell references or names, separated by commas.

The cells where you want to count blank cells.

Range of one or more cell addresses, a cell reference, or name to include in the count .

Numeric or string values that determine whether a cell is counted.

Number representing the settlement date; must be < Maturity.

Number representing the maturity date.

Frequency of coupon payments in number of payments per year (can be 1, 2, 3, 4, 6, or 12; the default is 2).

Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

Number representing the settlement date; must be < Maturity.

Number representing the maturity date.

Frequency of coupon payments in number of payments per year (can be 1, 2, 3, 4, 6, or 12; the default is 2).

Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

Number representing the settlement date; must be < Maturity.

Number representing the maturity date.

Frequency of coupon payments in number of payments per year (can be 1, 2, 3, 4, 6, or 12; the default is 2).

Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

Number representing the settlement date; must be < Maturity.

Number representing the maturity date.

Frequency of coupon payments in number of payments per year (can be 1, 2, 3, 4, 6, or 12; the default is 2).

Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

Number representing the settlement date; must be < Maturity.

Number representing the maturity date.

Frequency of coupon payments in number of payments per year (can be 1, 2, 3, 4, 6, or 12; the default is 2).

Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

Number representing the settlement date; must be < Maturity.

Number representing the maturity date.

Frequency of coupon payments in number of payments per year (can be 1, 2, 3, 4, 6, or 12; the default is 2).

Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

First array of numeric values.

Second array of numeric values.

Integer number of Bernoulli trials; must be ≥ 0.

Probability of success per trial; must be ≥ 0 and ≤ 1.

Critical probability to test; must be ≥ 0 and ≤ 1.

An angle measured in radians. X can be any value from approximately -9.00719E+15 through 9.00719E+15, excluding 0.

Any value from approximately -708.39599 through 708.39599, but not 0.

A numeric value > -1, representing the periodic interest rate (the fixed interest rate per compounding period).

A numeric value representing the future value of an investment (the value the investment will reach at some point).

A numeric value representing the current value of an investment (the present value).

Interest rate.

Total number of payment periods.

Present value.

First period in the calculation.

Last period in the calculation.

Timing of payment:
0 = payment at the end of the period
1 = payment at the beginning of the period

Interest rate.

Total number of payment periods.

Present value.

First period in the calculation.

Last period in the calculation.

Timing of the payment:
0 = payment at the end of the period
1 = payment at the beginning of the period

A general menu category.

A menu item that requires setting.

Date number.

Date number.

A numeric value between -300 and 1299; -300 = 1600, 0 = 1900, 1299 = 3199.

A numeric value between 1 and 12.

A numeric value between 1 and 31.

Date number representing start date.

Date number representing end date.

Code, entered as text, specifying format of the result:
y= Years
m= Months
d= Days
md= Days, disregarding months and years
ym= Months, disregarding years
yd= Days, disregarding years

Date number.

Code for the type of information you want:

1= Day of the week as a label, in short format (Mon)
2= Day of the week as a label, long format (Monday)
3= Day of the week as an integer from 0 (Monday) through 6 (Sunday)
4= Week of the year as an integer from 1 to 53
5= Month of the year as a label, in short format (Jan)
6= Month of the year as a label, in long format (January)
7= Number of days in the month specified by date
8= Number of days left in the month specified by date
9= Last day of the month specified by date
10= The Quarter date is in, as an integer from 1 (Q1) through 4 (Q4)
11= 1 if the year specified by date is a leap year; 0 if the year is not a leap year
12= Day of the year specified by date, as a number from 1 to 366
13= Days left in the year specified by date, as a number

A numeric or string value in any valid date format, enclosed by quotation marks (or cell coordinates or a cell name for a cells that contain a date string).

Cells (reference or name) containing the database, including field names.

The number of the column containing the field you want to average (the first column in Block is 0, the second is 1, and so on).

Cells containing search criteria; the first row must be field names.

Cells (reference or name) containing the database, including field names.

The number of the column containing the field you want to average (the first column in Block is 0, the second is 1, and so on).

Cells containing search criteria; the first row must be field names.

Cells (reference or name) containing the database, including field names.

The number of the column containing the field you want to average (the first column in Block is 0, the second is 1, and so on).

Cells containing search criteria; the first row must be field names.

Cells (reference or name) containing the database, including field names.

The number of the column containing the field you want to average (the first column in Block is 0, the second is 1, and so on).

Cells containing search criteria; the first row must be field names.

A numeric value between -109571 and 474816.9999999, representing a date/time serial number: -109571 = January 1, 1600; 0 = December 31, 1899; 474816 = December 31, 3199; the decimal = time (24 hr).

Date number.

Date number.

Optional logical value that specifies US or European method:
FALSE or 0= US (NASD); the default if you do not specify a method
TRUE or 1= European method

Amount originally paid for an asset.

Estimated value at end of asset life.

Number of periods the asset takes to depreciate to its salvage value.

Length of time for which you want to know the depreciation allowance.

Number of months in the first year (optional); If Month is omitted, @DB uses 12.

Cell (reference or name) containing the database, including field names.

The number of the column containing the field you want to count (the first column in Block is 0, the second is 1, and so on).

Cells containing search criteria; the first row must be field names.

A numeric value representing the amount paid for an asset.

A numeric value representing the value of an asset at the end of its useful life.

A numeric value representing the expected useful life of an asset (in years).

A numeric value representing the time period for which you want to calculate depreciation.

The DDE-server application to contact.
The table, spreadsheet, document, or other file in the DDE-server application from which to retrieve data.
The field, cells, bookmark, or other information to receive from the application (DDE Item string).

The number of columns in the data cells (optional).

The number of rows in the data cells (optional).

The number of sheets in the data cells (optional).

A numeric value representing radians.

Numeric value to check.

Numeric value that X must equal for the function to return 1 (if omitted, assumed to be zero).

One or more numeric or cell values.

Number to be converted, expressed as a decimal.

Denominator; must be an integer.

Number representing the settlement date; must be < Maturity.

Number representing the maturity date; must be > Settle.

Settlement price per 100 face value; must be ≥ 0 and ≤ 100.

Redemption value per 100 face value (must be > 0; the default is 100).

Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

Cells (reference or name) containing the database, including field names.

The number of the column containing the field for which you want to find the maximum value (the first column in Block is 0, the second is 1, and so on).

Cells containing search criteria; the first row must be field names.

Cells (reference or name) containing the database, including field names.

The number of the column containing the field for which you want to find the minimum value (the first column in Block is 0, the second is 1, and so on).

Cells containing search criteria; the first row must be field names.

Numeric value, reference to a cell that contains a numeric value, or formula that returns a numeric value.

Numeric value, reference to a cell that contains a numeric value, or formula that returns a numeric value.

1= Displays dollar value in text; ignores decimal values.
2= Displays dollar value in text, followed by "Dollars".
3= Displays dollar value in text, followed by cent value in numbers.
The decimal is rounded to two decimal places.
4= Displays dollar value in text, followed by cent value in numbers,
followed by "Dollars". This is the default if no Format is specified.
5= Displays dollar value in text, followed by "Dollars",
followed by cent value in text, followed by "Cents".

How many digits you want to display to the right of the decimal point. If Dec is negative, @DOLLAR rounds off Num to the left of the decimal point. If you omit Dec, @DOLLAR rounds off to 2 decimal places.

Number and numerator of the fraction, expressed as number.numerator.

Denominator of the fraction:

- Denom must be a numeric value >0

- If Denom is not an integer, @DOLLARDE truncates it

Dollar price.

Denominator of the fraction:

- Denom must be a numeric value >0

- If Denom is not an integer, @DOLLARDE truncates it

Cells (reference or name) containing the database, including field names.

The number of the column containing the field for which you want to find the standard deviation (the first column in Block is 0, the second is 1, and so on).

Cells containing search criteria; the first row must be field names.

Cells (reference or name) containing the database, including field names.

The number of the column containing the field for which you want to find the standard deviation (the first column in Block is 0, the second is 1, and so on).

Cells containing search criteria; the first row must be field names.

Cells (reference or name) containing the database, including field names.

The number of the column containing the field you want to total (the first column in Block is 0, the second is 1, and so on).

Cells containing search criteria; the first row must be field names.

Discount rate or cells containing discount rates that correspond to cash flows stored in Flows.

Cells containing cash flows associated with the discount rates in Discrate.

Initial cash flow (the default is 0).

Delay between initial and first cash flow, in number of periods (the default is 1) or cells containing lengths of periods between cash flows (the default is 1).

Flag specifying how to discount:
0 = compounded discounting (default)
1 = mixed compounded and simple discounting
2 = simple discounting

Flag specifying whether to apply path-dependent compounding to each flow; 0 = no path (default); 1 = path.

Flag specifying filter type: 0 = no filter (default); 1 = cashflow < Start; 2 = cashflow ≤ Start; 3 = cashflow > Start; 4 = cashflow ≥ Start; 5 = Start < cashflow < End; 6 = Start ≤ cashflow ≤ End.

A starting cash flow amount to compare against individual flows.

An ending cash flow amount to compare against individual flows.

Number representing the settlement date; must be < Maturity.

Number representing the maturity date; must be > Settle.

Coupon rate; must be ≥ 0.

Annual yield; must be > 0 and ≤ 1.

Frequency of coupon payments in the number of payments per year (can be 1, 2, 3, 4, 6, or 12; the default is 2).

Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

Cells (reference or name) containing the database, including field names.

The number of the column containing the field for which you want to compute variance (the first column in Block is 0, the second is 1, and so on).

Cells containing search criteria; the first row must be field names.

Cells (reference or name) containing the database, including field names.

The number of the column containing the field for which you want to compute variance (the first column in Block is 0, the second is 1, and so on).

Cells containing search criteria; the first row must be field names.

Nominal interest rate.

Number of compounding periods per year, truncated to an integer.

Number representing a date.

Serial number of start date.

Number of months before or after StartDate:
If Months is positive, @EOMONTHS returns a date after StartDate.
If Months is negative, @EOMONTHS returns a date before StartDate.
If Months is not an integer, @EOMONTHS truncates it.

Lower bound for integrating @ERF; must be ≥ 0.

Upper bound for integrating @ERF; if omitted, @ERF integrates the error function between 0 and Lower.

Lower bound for integrating @ERF; must be ≥ 0.

A value from -26.6417 to 26.6417.

Value to round.

A valid string value.

A valid string value.

A numerical value equal to or less than 709.

A value from -26.6417 to 26.6417.

Value at which to evaluate the function; must be ≥ 0.

Value to indicate; Lambda = 1/Mean; must be > 0.

1 to perform cumulative distribution function; 0 to perform the probability density function.

Integer ≥ 0 specifying the factorial to calculate.

Value ≥ 0 to calculate factorial of.

Integer from 0 through 170.

Number representing a date.

Cells containing dates that are holidays or the date of a single holiday or 0 to indicate no holidays (the default is 0).

0 to specify that Saturday is not a business day; 1 to specify that Saturday is a business day (the default is 0).

0 to specify that Sunday is not a business day; 1 to specify that Sunday is a business day (the default is 0).

Positive value at which to evaluate the function.

Numerator degrees of freedom; must be ≥ 1.

Denominator degrees of freedom; must be ≥ 1.

Total units; if Tu is negative, @FEETBL uses its absolute value.

Price per unit.

Fee table or a single value that defines the standard fee calculation.

Fee table or a single value that defines the minimum fee calculation (if omitted, MinTbl equals StdTbl).

Fee table or a single value that defines the maximum fee calculation (if omitted, MaxTbl equals StdTbl).

Number of places to which the final result is rounded; can be from 0 to 10 places (the default is no rounding).

Integer ≥ 0 specifying the desired term of a Fibonacci sequence.

A string value containing two or more delimited substrings, or a cell reference to a delimited string value.

Number of the substring you want to find (the first substring is numbered 1, the second 2, and so on).

Optional delimiter character; if you do not specify a delimiter, Quattro* Pro uses the delimiter character specified in the Application International property.

Any file name.

A valid string value, representing the value to search for.

A valid string value, representing the value to search through.

A numeric value ≥ 0, representing the character position to begin searching with; 0 = the first character.

Cumulative probability value; must be ≥ 0 and ≤ 1.

Numerator degrees of freedom; must be ≥ 1.

Denominator degrees of freedom; must be ≥ 1.

A cell or cell reference; can be a link to another opened notebook (for example, [BUDGET]A:A1).

A cell or cells of the notebook to check.

A string value representing a group name.

Numeric value; -1 < X < 1.

Numeric value ≤ 354 for which you want the inverse of the Fisher transformation.

Number to be rounded and converted to text.

Number of decimal places to be displayed. If Dec is negative, @FIXED rounds off Num to the left of the decimal point. If you omit Dec, @FIXED rounds off to 2 decimal places.

A logical value:
1 = do not display thousands separators
0 = display thousands separators (the default, if omitted, or if NoCommas ≠ 1)

Value to round.

Value to make rounded X evenly divisible by.

Numeric value at which to evaluate the function.

Dependent range of values.

Independent range of values.

Number to be converted.

Denominator; must be an integer.

Decimal value to be converted.

Denominator; must be an integer. If you specify a denominator that doesn't allow an exact fraction, the numerator is rounded to the nearest whole number.

When ForceDenom is not specified, the fraction is displayed in its lowest common denominator form; use 1 to display the denominator value specified in <Denom>.

Cells of values for which you want to count frequencies.

Array of or reference to intervals into which you want to group the values.

First array of numeric values.

Second array of numeric values.

Interest rate or cells containing interest (discount) rates.

Cells containing cash flows.

Delay after last cash flow in number of periods (the default is 0) or cells containing lengths of periods between cash flows (the default is 1).

Flag specifying how to discount:
0 = compounded discounting (default)
1 = mixed compounded and simple discounting
2 = simple discounting

Flag specifying whether to apply path-dependent compounding to each flow; 0 = no path (default); 1 = path.

Flag specifying filter type: 0 = no filter (default); 1 = cashflow < Start; 2 = cashflow ≤ Start; 3 = cashflow > Start; 4 = cashflow ≥ Start; 5 = Start < cashflow < End; 6 = Start ≤ cashflow ≤ End.

A starting cash flow amount to compare against individual flows.

An ending cash flow amount to compare against individual flows.

A numeric value representing the amount of the periodic payment.

A numeric value > -1, representing the periodic interest rate (the fixed interest rate per compounding period).

Number of periods, which should be an integer ≥ 2.

A numeric value > -1, representing the periodic interest rate (the fixed interest rate per compounding period).

Number of periods, which should be an integer > 0.

A numeric value representing the amount of the periodic payment.

A numeric value representing the current value of an investment (the present value).

An optional numeric value that indicates whether payments or cash flows occur at the beginning (1) or the end (0) of the period; default = 0.

Any positive number, not 0.

Value at which to evaluate the function; must be ≥ 0.

Parameter to the gamma distribution; must be > 0.

Parameter to the gamma distribution; must be > 0.

1 to return the cumulative gamma distribution function; 0 to return the probability density function.

Probability associated with the gamma cumulative function; must be ≥ 0 and ≤ 1.

A parameter to the gamma distribution; must be > 0.

A parameter to the gamma distribution; must be > 0.

Value for which you want to calculate @GAMMALN; must be > 0.

Parameter to the function; must be > 0.

Value at which to evaluate the function; must be ≥ 0.

Parameter to the function; must be > 0.

Value at which to evaluate the function; must be ≥ 0.

Positive integer to find greatest common divisor of.

Positive integer to find greatest common divisor of.

One or more numeric or cell values; values in List must be positive.

First term of the series.

Number of terms in the series.

Common ratio of the series.

Numeric value to check.

Numeric value that X must exceed for function to return 1 (if omitted, assumed to be 0).

A cell or cells of the notebook to check.

A string value representing a sheet name or an address specifying the sheet name to check (optional).

A string value representing the path in the registry.

A string value representing the stored value in the registry, at the specified path.

Any combination of cellss; separated by valid argument separators.

Array of known y-values for the curve y = b*m^x.

Array of known x-values (optional).

Array of new x-values for which you want the corresponding y-values (optional).

Logical value (optional) that tells @GROWTH whether to force the constant b = 1:

- If Const is TRUE or omitted, @GROWTH uses the actual value of b.
- If Const is FALSE, @GROWTH sets b = 1, then adjusts the m-values so that y = m^x.

One or more numeric or cell values; none of the values in List can equal 0.

Hexadecimal number to convert; can be up to 40 hexadecimal digits.

Hexadecimal number to convert.

Hexadecimal number to convert.

Number of characters to return; must be ≤ 64.

A hexadecimal number enclosed by double quotes.

Hexadecimal number to convert.

1 if the most significant bit of Hex is a sign bit; 0 if Hex is positive (the default is 0).

Hexadecimal number to convert.

Hexadecimal number to convert.

Number of characters to return; must be ≤ 22.

A numeric or string value.

Cell value.

The number of the referenced row, from 0 to the number of rows in Block - 1 (the first row in Block = 0, the second = 1, and so on).

Number representing the start date.

Number representing the end date.

Cells containing dates that are holidays; to indicate no holidays, enter an empty cell or cells.

0 to specify that Saturday is not a business day; 1 to specify that Saturday is a business day (the default is 0).

0 to specify that Sunday is not a business day; 1 to specify that Sunday is a business day (the default is 0).

A numeric value between -109571 and 474816.9999999, representing a date/time serial number: -109571 = January 1, 1600; 0 = December 31, 1899; 474816 = December 31, 3199; the decimal = time (24 hr).

Successes in the sample; must be ≥ 0.

Sample size; must be ≥ 0 and $\leq$ PopSize.

Successes in the population; must be ≥ 0 and $\leq$ PopSize.

Population size; must be ≥ 0.

A logical expression representing the condition to be tested.

A numeric or string value representing the value to use if Cond is true.

A numeric or string value representing the value to use if Cond is false.

Complex number in the format $x + y$i, $x + iy$, $x + y$j, or $x + jy$ for which you want the absolute (modulus) value.

Complex number in the format $x + y$i, $x + iy$, $x + y$j, or $x + jy$ from which you want to extract the imaginary coefficient.

Complex number in the format $x + y$i, $x + iy$, $x + y$j, or $x + jy$ for which you want to calculate the angle in the complex plane.

Complex number in the format $x + y$i, $x + iy$, $x + y$j, or $x + jy$ for which you want to calculate the complex conjugate.

Complex number in the format $x + y$i, $x + iy$, $x + y$j, or $x + jy$ for which you want to calculate the cosine.

Complex numerator or dividend in the format $x + y$i, $x + $ iy, $x + y$j, or $x + $ jy.

Complex denominator or divisor in the format $x + y\mathrm{i}$, $x + \mathrm{i}y$, $x + y\mathrm{j}$, or $x + \mathrm{j}y$.

Complex number in the format $x + y$i, $x + iy$, $x + y$j, or $x + jy$ for which you want to calculate the exponential.

Complex number in the format $x + y$i, $x + iy$, $x + y$j, or $x + jy$ for which you want to calculate the natural logarithm.

Complex number in the format $x + y$i, $x + iy$, $x + y$j, or $x + jy$ for which you want to calculate the base 10 log.

Complex number in the format $x + y$i, $x + iy$, $x + y$j, or $x + jy$ for which you want to calculate the base 2 log.

Complex number in the format $x + y$i, $x + iy$, $x + y$j, or $x + jy$.

The power to which you want to raise Complex; can be a complex number in the format *x* + *y*i, *x* + i*y*, *x* + *y*j, or *x* + j*y*.

Complex number in the format $x + y$i, $x + iy$, $x + y$j, or $x + jy$.

Complex number in the format $x + y$i, $x + iy$, $x + y$j, or $x + jy$.

Complex number in the format $x + y$i, $x + iy$, $x + y$j, or $x + jy$ from which you want to extract the real coefficients.

Complex number in the format $x + y$i, $x + iy$, $x + y$j, or $x + jy$ for which you want to calculate the sine.

Complex number in the format $x + y$i, $x + iy$, $x + y$j, or $x + jy$ to calculate square root of.

Complex number in the format $x + y$i, $x +$ iy, $x + y$j, or $x +$ jy from which to subtract Complex2.

Complex number in the format $x + y$i, $x + iy$, $x + y$j, or $x + jy$ to subtract from Complex1.

One or more complex numbers in the format $x + yi$, $x + iy$, $x + yj$, or $x + jy$, separated by commas.

A cell reference or name.

The number of the referenced column, from 0 to 255 (the first column in Block = 0, the second = 1, and so on).

The number of the referenced row; if an offset, the first row in Block = 0, the second = 1, and so on.

The number of the referenced sheet, from 0 to 255 (the first sheet in Block = 0, the second = 1, and so on).

An integer number from 0 to 255 inclusive.

A numeric value.

A numeric value.

Dependent range of values.

Independent range of values.

Number representing the settlement date; must be < Maturity.

Number representing the maturity date; must be > Settle.

Amount invested; must be > 0.

Redemption value; must be > 0.

Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

Binary number.

Number of binary bits used for both input and output; if omitted, Bits = number of bits in Binary; must be ≤ 64.

Hexadecimal number.

Number of binary bits used for both input and output; if omitted, Bits = number of bits in Hex; 4 binary digits = 1 hexadecimal digit; must be ≤ 64.

A numeric value > -1, representing the periodic interest rate (the fixed interest rate per compounding period).

The number of the loan period for which the interest is desired (where Nper is the total number of periods).

A numeric value > 0, representing the number of periods of the loan (the number of payments to be made) or investment (the number of compounding periods).

A numeric value representing the amount borrowed (the principal).

A numeric value representing the future value of an investment (the value the investment will reach at some point).

An optional numeric value that indicates whether payments or cash flows occur at the beginning (1) or the end (0) of the period; default = 0.

A numeric value > 0, representing the number of periods of the loan (the number of payments to be made) or investment (the number of compounding periods).

A numeric value representing the amount of the periodic payment.

A numeric value representing the current value of an investment (the present value).

A numeric value representing the future value of an investment (the value the investment will reach at some point).

An optional numeric value that indicates whether payments or cash flows occur at the beginning (1) or the end (0) of the period; default = 0.

A numeric value that estimates the internal rate of return on an investment.

Cell (reference or name) containing cash flow information for the investment.

Number representing a date.

Cells containing dates that are holidays or the date of a single holiday or 0 to indicate no holidays (the default is 0).

0 to specify that Saturday is not a business day; 1 to specify that Saturday is a business day (the default is 0).

0 to specify that Sunday is not a business day; 1 to specify that Sunday is a business day (the default is 0).

Name or address of a cell.

Cell address or presumed cell name.

A cell address or expression.

Value to test.

A string value.

Empty cell, logical value, text, number, ERR, cell reference, or cell name to test.

A cell address or expression.

Empty cell, logical value, text, number, ERR, cell reference, or cell name to test.

A cell address or expression.

Value to test.

A cell address or expression.

One or more numeric or cell values.

A numeric array or cells of values.

Number that indicates the rank in size from the data set Array; must be greater than 0 and less than or equal to the number of values in Array.

A cell or reference; can be a link to another opened notebook (for example, [BUDGET]A:A1).

A cell or reference.

Number 1 (column) or 2 (row) ; the default type is 1.

A cell or cells of the notebook to check.

A string value representing a group name.

Number representing a date.

Cells containing dates that are holidays or the date of a single holiday or 0 to indicate no holidays (the default is 0).

0 to specify that Saturday is not a business day; 1 to specify that Saturday is a business day (the default is 0).

0 to specify that Sunday is not a business day; 1 to specify that Sunday is a business day (the default is 0).

Positive integer to find least common multiple of.

Positive integer to find least common multiple of.

A string value.

A numeric value equal to or greater than 0.

A string value.

A one- or two-character string enclosed in quotation marks; column and sheet letters run in sequence from A to Z, and continue from AA to AZ, up to IV.

Array of known y-values for the line y = mx + b.

Array of known x-values (optional).

Logical value (optional) that tells @LINEST whether to force the constant b = 0:
If Const is TRUE or omitted, @LINEST uses the actual value of b.
If Const is FALSE, @LINEST sets b = 0, then adjusts the m-values so that y = mx.

Logical value (optional) that tells @LINEST whether to return more regression statistics.
If Stats is TRUE, @LINEST returns the array

If Stats is FALSE or omitted, @LINEST returns only the m-coefficients and b.

One-dimensional cells containing X values in increasing order.

One-dimensional cells containing Y values corresponding to the X values in KnownX.

Number for which the corresponding Y value is desired.

Degrees of Latitude or Longitude.

Minutes of Latitude or Longitude.

Seconds of Latitude or Longitude.

For Latitude, North (1) or South (2) of the equator; for Longitude, East (3) or West (4) of the prime meridian at Greenwich, England.

A numeric value > 0.

A numeric value > 0.

Positive real number.

A numeric value greater than 0

Value to be converted = the log of a number m to the base b.

Positive integer greater than 1.

Positive integer greater than 1.

Array of known y-values for the curve y = b*m^x.

Array of known x-values.

Logical value (optional) that tells @LOGEST whether to force the constant b = 1:
If Const is TRUE or omitted, @LOGEST uses the actual value of b.
If Const is FALSE, @LOGEST sets b = 1, then adjusts the m-values so that y = m^x.

Logical value (optional) that tells @LOGEST whether to return more regression statistics.
If Stats is TRUE, @LOGEST returns the array

If Stats is FALSE or omitted, @LOGEST returns only the m-coefficients and b.

Probability associated with the cumulative lognormal distribution function; $0 \leq \text{Prob} < 1$.

Mean of ln(x).

Standard deviation of ln(x); must be > 0.

Value to evaluate the function; must be > 0.

Mean of ln(x).

Standard deviation of ln(x); must be > 0.

Value to look for in LookupVector; can be a number, text, logical value, or reference to a value.

Cells containing only one row or column.

Cells of the same dimensions as LookupVector and containing corresponding values.

A string value.

Number from 1 (Saturday) to 7 (Friday).

Number from 1 (January) to 12 (December).

Number from 0 (1900) to 199 (2099) or a standard year like 1993.

Auxiliary day of the week that must fall in the same week as Wkday; 0 for no auxiliary day or a number from 1 (Saturday) to 7 (Friday) indicating the auxiliary day (the default is 0).

Numeric or string value to be matched.

Cells, contiguous selections, an array, or array reference.

-1, 0, or 1. Match Type specifies which cell positions are returned:
-1 = smallest, 1 = largest, 0 = first found.

One or more numeric values, cell addresses, and cell references or names, separated by commas.

cells or list of selections containing numeric values.

Number from 1 (January) to 12 (December).

Number from 0 (1900) to 199 (2099) or a standard year like 1993.

A numeric array or cells of values specifying a square matrix; must have an equal number of rows and columns, and cannot contain blank cells.

Number representing the settlement date; must be < Maturity.

Number representing the maturity date; must be > Settle.

Coupon rate; $0 \leq \text{Coupon} \leq 1$.

Annual yield; $0 < \text{Yield} \leq 1$.

Frequency of coupon payments in number of payments per year (can be 1, 2, 3, 4, 6, or 12; the default is 2).

Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

One or more numeric or cell values.

A string value.

A numeric value equal to or greater than 0.

A numeric value equal to or greater than 0.

One or more numeric values, cell addresses, and cell references or names, separated by commas.

Cells or list of selections containing numeric values.

A numeric value between -109571 and 474816.9999999, representing a date/time serial number: -109571 = January 1, 1600; 0 = December 31, 1899; 474816 = December 31, 3199; the decimal = time (24 hr).

Square numeric array (same number of rows as columns); you can use a cell reference or cell name or an array constant like {1,2|3,4}.

Cells containing cash flows; negative = outflow, positive = inflow.

Interest rate paid for funds used in cash flows.

Interest rate received on reinvested funds used in cash flows.

Timing of the cash flows (optional):
0 = end of each period (default)
1 = beginning of each period

Array to be multiplied.

Array to be multiplied.

Number representing the start date.

Number representing the end date.

1 to indicate adherence to ends of months; 0 to indicate that ends of months are ignored; the default is 1.

A numeric value.

A numeric value not equal to 0.

One or more numeric or cell values.

Numeric value.

Numeric value, but not 0.

A numeric value between -109571 and 474816.9999999, representing a date/time serial number: -109571 = January 1, 1600; 0 = December 31, 1899; 474816 = December 31, 3199; the decimal = time (24 hr).

Value to round.

Value to make rounded X divisible by.

Number ≥ 0 representing the periodic interest rate.

Total periods in the loan from start to finish, or the total periods remaining from the chosen starting period forward.

Original loan balance; also can be any starting point in the loan.

Remaining balance on loan at end of loan term; enter 0 if the loan will be paid in full.

Extra principal amount to be paid each period (must be positive).

Number of the first period, relative to the starting point, in which extra principal is paid; the default is 1 (the first period).

Number of the last period, relative to the starting point, in which extra principal is paid; the default is until the end of the loan; you can set Lper to any number greater than or equal to the last period number when extra principal payments last the life of the loan (for example, Lper can be 400 for a loan which lasts 360 periods).

Period for which the loan status is reported; the default is at loan end (any number greater than the end of the loan defaults to loan end); Rper does not affect the value @MTGACC returns if Option is 0 or 10.

Specifies the output value type (the default is 0):
0 = number of periods to loan end, when balance equals Residual
1 = balance of loan at the Rper
2 = cumulative interest paid at Rper
3 = cumulative principal paid at Rper
10 = number of fewer periods in loan life, due to payment of extra principal
11 = balance reduction at Rper due to payment of extra principal
12 = reduction in cumulative interest paid at Rper due to payment of extra principal
13 = increase in cumulative principal paid at Rper due to payment of extra principal

One or more numbers or selections of numbers, separated by commas.

One or more numbers to calculate multinomial of; each number in List must be ≥ 0.

A cell reference or name.

Number representing a date.

Cells containing dates that are holidays or the date of a single holiday or 0 to indicate no holidays (the default is 0).

0 to specify that Saturday is not a business day; 1 to specify that Saturday is a business day (the default is 0).

0 to specify that Sunday is not a business day; 1 to specify that Sunday is a business day (the default is 0).

Number of failures.

Threshold of successes.

Probability of a success; $0 \leq \text{Prob} \leq 1$.

Discount rate or cells containing discount rates corresponding to cells of cash flows.

Cells containing cash flows.

Initial cash flow (the default is 0).

Delay between initial and first cash flow in number of periods (the default is 1) or cells containing lengths of periods between cash flows (the default is 1).

Flag specifying how to discount:
0 = compounded discounting (default)
1 = mixed compounded and simple discounting
2 = simple discounting

Flag specifying whether to apply path-dependent compounding to each flow; 0 = no path (default); 1 = path.

Flag specifying filter type: 0 = no filter (default); 1 = cashflow < Start; 2 = cashflow ≤ Start; 3 = cashflow > Start; 4 = cashflow ≥ Start; 5 = Start < cashflow < End; 6 = Start ≤ cashflow ≤ End.

A starting cash flow amount to compare against individual flows.

An ending cash flow amount to compare against individual flows.

Date number representing start date.

Date number representing end date.

Optional cell name or reference containing serial date numbers of holidays to exclude from the calculation.

Optional argument, in quotation marks, to tell @NETWORKDAYS which days are weekend days. Use 0 through 6 (Monday through Sunday); for example, "45" means Friday and Saturday. The default, if you omit Weekends, is Saturday and Sunday. To specify no weekends, use "7".

Effective interest rate.

Number of compounding periods per year, truncated to an integer.

Value at which to evaluate function.

Mean of the normal distribution.

Standard deviation of the normal distribution; must be > 0.

1 to return the cumulative normal distribution function; 0 (the default) to return the probability density function.

Probability corresponding to the normal distribution; 0 < Prob < 1.

Mean of the normal distribution.

Standard deviation of the normal distribution; must be > 0.

Value at which to evaluate the function.

Probability corresponding to the normal distribution; must be > 0 and < 1.

Logical value or expression that can be evaluated to TRUE or FALSE.

A numeric value > -1, representing the periodic interest rate (the fixed interest rate per compounding period).

A numeric value representing the amount of the periodic payment.

A numeric value representing the current value of an investment (the present value).

A numeric value representing the future value of an investment (the value the investment will reach at some point).

An optional numeric value that indicates whether payments or cash flows occur at the beginning (1) or the end (0) of the period; default = 0.

A numeric value > -1, representing the periodic interest rate (the fixed interest rate per compounding period).

Cells (reference or name) containing cash flow information for the investment.

An optional numeric value that indicates whether payments or cash flows occur at the beginning (1) or the end (0) of the period; default = 0.

Decimal number to convert.

Decimal number to convert.

Number of characters to return; must be ≤ 64.

Decimal number to convert.

Decimal number to convert.

Number of characters to return; must be ≤ 16.

Decimal number to convert.

Decimal number to convert.

Number of characters to return; must be ≤ 22.

Number from 1 to 5.

Number from 1 (Saturday) to 7 (Friday).

Number from 1 (January) to 12 (December).

Number from 0 (1900) to 199 (2099) or a standard year like 1993.

Auxiliary day of the week that must fall in the same week as Wkday; 0 for no auxiliary day or a number from 1 (Saturday) to 7 (Friday) indicating the auxiliary day (the default is 0).

Octal number to convert; denote negative numbers using a minus sign.

Octal number to convert; denote negative numbers using a minus sign.

Octal number to convert; denote negative numbers using a minus sign.

Value to round.

Number representing the settlement date.

Number representing the maturity date.

Number representing the issue date.

Number representing the first coupon date.

Coupon rate; must be ≥ 0.

Annual yield; $0 < \text{Yield} \leq 1$.

Redemption value per 100 face value (must be > 0; the default is 100).

Frequency of coupon payments in number of payments per year (can be 1, 2, 3, 4, 6, or 12; the default is 2).

Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

Number representing the settlement date.

Number representing the maturity date.

Number representing the issue date.

Number representing the first coupon date.

Coupon rate; must be ≥ 0.

Price of the security; must be > 0.

Redemption value per 100 face value (must be > 0; the default is 100).

Frequency of coupon payments in number of payments per year (can be 1, 2, 3, 4, 6, or 12; the default is 2).

Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

Number representing the settlement date; must be < Maturity.

Number representing the maturity date.

Number representing the last coupon date; must be < Maturity.

Coupon rate; must be ≥ 0.

Annual yield; $0 < \text{Yield} \leq 1$.

Redemption value per 100 face value (must be > 0; the default is 100).

Frequency of coupon payments in number of payments per year (can be 1, 2, 3, 4, 6, or 12; the default is 2).

Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

Number representing the settlement date; must be < Maturity.

Number representing the maturity date.

Number representing the last coupon date; must be < Maturity.

Coupon rate; must be ≥ 0.

Price; must be > 0.

Redemption value per 100 face value (must be > 0; the default is 100).

Frequency of coupon payments in number of payments per year (can be 1, 2, 3, 4, 6, or 12; the default is 2).

Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

Reference on which you want to base the offset; cannot refer to non-contiguous areas.

Number of rows from Reference you want the offset to refer to. If Rows = 5, the upper left cell in the offset is five rows below the upper left cell in Reference. Negative Rows are above, positive are below.

Number of columns from Reference you want the offset to refer to. If Cols = 5, the upper left cell in the offset is five columns to the right of the upper left cell in Reference. Negative Cols are to the left, positive are to the right.

Height (optional) of the returned offset, in rows. Height must be a positive number. If omitted, Height is the same height as Reference.

Width (optional) of the returned offset, in columns. Width must be a positive number. If omitted, Width is the same width as Reference.

True-or-false conditions to test.

First binary number.

Second binary number.

Number of binary bits used for both input and output; if omitted, Bits = number of bits in Binary1 or Binary2, whichever is greater; must be ≤ 64.

First hexadecimal number.

Second hexadecimal number.

Number of binary bits used for both input and output; if omitted, Bits = number of bits in Hex1 or Hex2, whichever is greater; 4 binary digits = 1 hexadecimal digit; must be ≤ 64.

A string corresponding to the name of a sheet; must be enclosed in quotation marks.

A reference to a sheet, cell, or cells in another notebook (for example, [BUDGET]A:A1).

A string corresponding to the name of a sheet; must be enclosed in quotation marks.

A number from 0 to 255 inclusive.

A reference to a sheet, cell, or cells in another notebook (for example, [BUDGET]A:A1).

A number from 0 to 255 inclusive.

A optional argument specifying sheet names to exclude from the table; enclose each sheet name in quotation marks, and separate sheet names with a comma.

A reference to a sheet, cell, or cells in another notebook (for example, [BUDGET]A:A1).

A optional argument specifying sheet names to exclude from the table; enclose each sheet name in quotation marks, and separate sheet names with a comma.

A numeric value > -1, representing the periodic interest rate (the fixed interest rate per compounding period).

A numeric value > 0, representing the number of periods of the loan (the number of payments to be made) or investment (the number of compounding periods).

A numeric value representing the amount borrowed (the principal).

A numeric value representing the future value of an investment (the value the investment will reach at some point).

An optional numeric value that indicates whether payments or cash flows occur at the beginning (1) or the end (0) of the period; default = 0.

Number representing a date.

Cells containing dates that are holidays or the date of a single holiday or 0 to indicate no holidays (the default is 0).

0 to specify that Saturday is not a business day; 1 to specify that Saturday is a business day (the default is 0).

0 to specify that Sunday is not a business day; 1 to specify that Sunday is a business day (the default is 0).

Array of independent values.

Array of dependent values.

A numeric array or cells of values.

A percentile value between 0 and 1, inclusive.

A numeric array or cells of values.

Number to rank in Array; if X does not match a value in Array, @PERCENTRANK interpolates to return a percentage rank.

Number of significant digits for returned percentage value; must be ≥ 1 (the default is 3).

Number of different objects; $n \geq 0$.

Number of objects taken at a time; $R \leq N$.

Net present value.

Flag specifying filter type: 0 = no filter (default); 1 = cashflow < Start; 2 = cashflow ≤ Start; 3 = cashflow > Start; 4 = cashflow ≥ Start; 5 = Start < cashflow < End; 6 = Start ≤ cashflow ≤ End.

A starting cash flow amount to compare against individual flows.

An ending cash flow amount to compare against individual flows.

Cells containing cash flows.

Initial cash flow (the default is 0).

Delay between initial and first cash flow, in number of periods (the default is 1) or cells containing lengths of periods between cash flows (the default is 1).

Flag specifying how to discount:
0 = compounded discounting (default)
1 = mixed compounded and simple discounting
2 = simple discounting

Flag specifying whether to apply path-dependent compounding to each flow; 0 = no path (default); 1 = path.

IRR guess for numerical search (useful for locating multiple roots); must be > -100%; the default is 0.10.

Minimum required precision; Precision > 0; the default is 0.000001.

Maximum number of iterations for search; Maxiter > 0; the default is 50.

A numeric value representing the amount borrowed (the principal).

A numeric value > -1, representing the periodic interest rate (the fixed interest rate per compounding period).

A numeric value > 0, representing the number of periods of the loan (the number of payments to be made) or investment (the number of compounding periods).

A numeric value representing the amount borrowed (the principal).

A numeric value > -1, representing the yearly interest rate (the fixed interest rate per compounding period).

A numeric value > 0, representing the number of months of the loan (the number of payments to be made).

Number of events; must be ≥ 0.

Expected numeric value for the mean over the distribution; must be > 0.

1 to return the cumulative Poisson probability distribution that the number of random events will be in the range from zero to N; 0 to return the Poisson probability mass function that the number of events will be N.

Number (base) to be raised to a power; can be any real number.

Power (exponent), to which Num is to be raised.

A numeric value > -1, representing the periodic interest rate (the fixed interest rate per compounding period).

The number of the loan period for which the principal is desired (where Nper is the total number of periods).

A numeric value > 0, representing the number of periods of the loan (the number of payments to be made) or investment (the number of compounding periods).

A numeric value representing the amount borrowed (the principal).

A numeric value representing the future value of an investment (the value the investment will reach at some point).

An optional numeric value that indicates whether payments or cash flows occur at the beginning (1) or the end (0) of the period; default = 0.

Number representing the settlement date; must be < Maturity.

Number representing the maturity date; must be > Settle.

Coupon rate; must be ≥ 0.

Annual yield; must be > 0 and ≤ 1.

Redemption value per 100 face value; must be > 0; the default is 100.

Frequency of coupon payments in number of payments per year (can be 1, 2, 3, 4, 6, or 12; the default is 2).

Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

Number representing the settlement date; must be < Maturity.

Number representing the maturity date; must be > Settle.

Rate of discount; $0 \leq \text{Discount} \leq 1$.

Redemption value per 100 face value (must be > 0; the default is 100).

Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

Number representing the settlement date; must be < Maturity.

Number representing the maturity date; must be > Settle.

Number representing the issue date; must be < Settle.

Coupon rate; $0 \leq \text{Coupon} \leq 1$.

Annual yield; $0 \leq \text{Yield} \leq 1$.

Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

Values of X associated with the probabilities.

cells or array of probability values associated with XData; each value in ProbRange must be ≥ 0 and ≤ 1; the sum of ProbRange values must equal 1.

Lower limit on the value for the desired probability.

Upper limit on the value for the desired probability.

A string value.

The name of the object whose property settings you are requesting.

The property whose settings you are requesting.

One or more numeric or cell values.

One or more numeric or string values, cell addresses, and cell references or names, separated by commas.

One or more numeric, cell addresses, and cell references or names, separated by commas.

One or more numeric, cell addresses, and cell references or names, separated by commas.

One or more numeric, cell addresses, and cell references or names, separated by commas.

One or more numeric, cell addresses, and cell references or names, separated by commas.

One or more numeric, cell addresses, and cell references or names, separated by commas.

One or more numeric, cell addresses, and cell references or names, separated by commas.

A numeric value representing the amount of the periodic payment.

A numeric value > -1, representing the periodic interest rate (the fixed interest rate per compounding period).

A numeric value > 0, representing the number of periods of the loan (the number of payments to be made) or investment (the number of compounding periods).

A numeric value > -1, representing the periodic interest rate (the fixed interest rate per compounding period).

A numeric value > 0, representing the number of periods of the loan (the number of payments to be made) or investment (the number of compounding periods).

A numeric value representing the amount of the periodic payment.

A numeric value representing the future value of an investment (the value the investment will reach at some point).

An optional numeric value that indicates whether payments or cash flows occur at the beginning (1) or the end (0) of the period; default = 0.

A numeric array or cells of values.

Number signifying what quartile value to return:
0 = minimum value in Array
1 = 25th percentile
2 = 50th percentile (median)
3 = 75th percentile
4 = maximum value in Array

Value to divide.

Nonzero value to divide x by.

A numeric value representing degrees (0 to 360).

Integer value that random number must be greater than or equal to.

Integer value that random number must be less than or equal to.

A number from Array.

One or more numeric or cell values.

Flag indicating how to sort the list of numbers: any nonzero value = ascending order; 0 = descending order.

A numeric value representing the future value of an investment (the value the investment will reach at some point).

A numeric value representing the current value of an investment (the present value).

A numeric value > 0, representing the number of periods of the loan (the number of payments to be made) or investment (the number of compounding periods).

Number representing the settlement date; must be < Maturity.

Number representing the maturity date; must be > Settle.

Amount invested; must be > 0.

Rate of discount; $0 \leq \text{Discount} \leq 1$.

Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

Name or address of cells containing independent variables; can be up to 75 columns and 8,192 rows.

Name or address of cells containing dependent variable; must be a single-column selection with the same number of rows as XBlock.

Specifies the type of regression value returned. Valid attributes are 1, 2, 3, 4, 5, 101 to 175, or 201 to 275; see the table for the type of regression value returned for each attribute value.

Logical value (optional) that tells @REGRESSION whether to force the Y intercept to equal 0:
0 = make the Y intercept 0
1 = calculate the Y intercept (default if you omit the argument)

A string value.

A numeric value equal to or greater than 0.

A valid string value, representing the text to operate on.

A numeric value equal to or greater than 0, representing the character position to begin with.

A numeric value equal to or greater than 0, representing the number of characters to delete.

A string value, representing the characters to insert at position Num.

A string value.

A numeric value ≥ 0.

Arabic numeral.

Style of Roman numeral, ranging from full classic to brief, becoming more concise as the value of Form increases:
FALSE (0) = Classic; default, if omitted
TRUE (1) = More concise
2 = More concise
3 = More concise
4 = Most concise

Number; can be positive or negative.

Number, not zero.

A numeric value.

A numeric value between -15 and 15.

Number to round down.

Number of digits (optional) to which you want to round X.

Argument (optional) specifying how to round negative values:
0 = round negative values down; default if omitted
1 = round negative values up

Number to round up.

Number of digits (optional) to which you want to round X.

The cell or cells for which you want the row number(s).

A cell reference or name.

Independent range of values.

Dependent range of values.

A cell reference or name.

Net present value.

Maximum number of iterations for search (must be > 0; the default is 50).

Flag specifying filter type: 0 = no filter (default); 1 = cashflow < Start; 2 = cashflow ≤ Start; 3 = cashflow > Start; 4 = cashflow ≥ Start; 5 = Start < cashflow < End; 6 = Start ≤ cashflow ≤ End.

A starting cash flow amount to compare against individual flows.

An ending cash flow amount to compare against individual flows.

Discount rate or cells containing discount rates corresponding to cells of cash flows.

Cells containing cash flows.

Initial cash flow (the default is 0).

Delay between initial and first cash flow, in number of periods (the default is 1) or cells containing lengths of periods between cash flows (the default is 1).

Flag specifying how to discount:
0 = compounded discounting (default)
1 = mixed compounded and simple discounting
2 = simple discounting

Flag specifying whether to apply path-dependent compounding to each flow; 0 = no path (default); 1 = path.

Initial margin for numerical search (useful for locating multiple roots) (must be > -100%; the default is 0).

Minimum required precision (must be > 0; the default is 0.000001).

An angle measured in radians. X can be any value from approximately -9.00719E+15 through 9.00719E+15.

A value from approximately -708.39599 to approximately 708.39599.

A numeric value between -109571 and 474816.9999999, representing a date/time serial number: -109571 = January 1, 1600; 0 = December 31, 1899; 474816 = December 31, 3199; the decimal = time (24 hr).

Cell reference or name.

Value in the power series.

Initial power to raise X to.

Increment of the power N for each successive term in the power series.

Cells or array of one or more numeric values by which each power of X is multiplied; the number of values in Coefficients sets the number of terms in the power series.

Label text, in quotation marks.

Integer from 1 through 1022 specifying label length.

Optional argument specifying text alignment:
0 = align text left; default if you omit the argument
1 = center text
2 = align text right

A cell reference or name.

Binary number.

Number of bits to shift; $0 \leq \text{ShiftBits} \leq 64$; the default is 1.

Binary bit inserted during the shift (can be 0, 1, "S" or "I"; "S" = same as the least significant bit before shifting; "I" = inverse of the least significant bit before shifting; the default is 0).

Number of binary bits used for both input and output; if omitted, Bits = number of bits in Binary; must be ≤ 64.

Binary number.

Number of input binary digits used during the shift operation; if omitted, Bits = the number of bits in Binary; must be ≤ 64.

Hexadecimal number.

Number of bits to shift; $0 \leq \text{ShiftBits} \leq 64$; the default is 1.

Binary bit inserted during the shift (can be 0, 1, "S" or "I"; "S" = same as the least significant bit before shifting; "I" = inverse of the least significant bit before shifting; the default is 0).

Number of binary bits used for both input and output; if omitted, Bits = number of bits in Hex; 4 binary digits = 1 hexadecimal digit; must be ≤ 64.

Hexadecimal number.

Number of equivalent input binary digits used during the shift operation; if omitted, Bits = the number of bits in Hex; 4 binary digits = 1 hexadecimal digit; must be $\leq$ 64.

Binary number.

Number of bits to shift; $0 \leq \text{ShiftBits} \leq 64$; the default is 1.

Binary bit inserted during the shift (can be 0, 1, "S" or "I"; "S" = same as the most significant bit before shifting; "I" = inverse of the most significant bit before shifting; the default is "S").

Number of binary bits used for both input and output; if omitted, Bits = number of bits in Binary; must be ≤ 64.

Binary number.

Number of input binary digits used during the shift operation; if omitted, Bits = the number of bits in Binary; must be ≤ 64.

Hexadecimal number.

Number of bits to shift; $0 \leq \text{ShiftBits} \leq 64$; the default is 1.

Binary bit inserted during the shift (can be 0, 1, "S" or "I"; "S" = same as the most significant bit before shifting; "I" = inverse of the most significant bit before shifting; the default is "S").

Number of binary bits used for both input and output; if omitted, Bits = number of bits in Hex; 4 binary digits = 1 hexadecimal digit; must be ≤ 64.

Hexadecimal number.

Number of equivalent input binary digits used during the shift operation; if omitted, Bits = the number of bits in Hex; 4 binary digits = 1 hexadecimal digit; must be $\leq$ 64.

Value.

A numeric value.

A value from approximately -710.47558 to approximately 710.47558.

One or more numeric or cell values.

A numeric value representing the amount paid for an asset.

A numeric value representing the value of an asset at the end of its useful life.

A numeric value representing the expected useful life of an asset (in years).

Dependent range of values.

Independent range of values.

A numeric array or cells of values.

Number that indicates the rank in size from the data set Array; must be greater than 0 and less than or equal to the number of values in Array.

Independent cells or array of values.

Dependent cells or array of values.

The cell address where the result of the @function is to be displayed.

A numeric value equal to or greater than 0.

Value ≥ 0 to multiply by pi.

Number to normalize.

Arithmetic mean of a distribution.

Standard deviation of a distribution.

One or more numeric values, cell addresses, and cell references or names, separated by commas.

One or more numeric values, cell addresses, and cell references or names, separated by commas.

Dependent range of 3 or more values.

Independent range of 3 or more values.

Dependent range of 3 or more values.

Independent range of 3 or more values.

Option code string with expiration month, strike-price, and put or call symbol enclosed in quotation marks (for example, "MAY 22.5 C"); the strike price can be a decimal or fractional number, but not both (for example, it can be 11/32 or 1.625, but not 2 3/8); valid month codes are JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, and DEC; you can add spaces between month, strike-price, and Put or Call symbol; the total string (not including quotation marks) must be 20 characters or less.

Option premium or price.

Value or price of the underlying stock.

Serial date number between 2 (January 1, 1900) and 73050 (December 31, 2099) representing the date on which to evaluate the stock option.

Load or commission involved in sale or purchase.

Command string enclosed in quotation marks specifying the operations to perform; cannot exceed 20 characters (not including quotes).

A numeric value, a formula that evaluates to a numeric value, or a reference to a cell containing a numeric value.

A numeric value from 0 through 15.

First binary number.

Second binary number.

Input borrow bit (either 0 or 1); the default is 0.

Number of binary bits used for both input and output; if omitted, Bits = number of bits in Binary1 or Binary2, whichever is greater; must be ≤ 64.

First binary number.

Second binary number.

Input borrow bit (either 0 or 1); the default is 0.

Number of input binary digits used for subtraction; if omitted, Bits = the number of bits in Binary1 or Binary2, whichever is greater; must be ≤ 64.

First hexadecimal number.

Second hexadecimal number.

Input borrow bit (either 0 or 1); the default is 0.

Number of binary bits used for both input and output; if omitted, Bits = number of bits in Hex1 or Hex2, whichever is greater; 4 binary digits = 1 hexadecimal digit; must be ≤ 64.

First hexadecimal number.

Second hexadecimal number.

Input borrow bit (either 0 or 1); the default is 0.

Number of input binary digits used for subtraction; if omitted, Bits = the number of bits in Hex1 or Hex2, whichever is greater; 4 binary digits = 1 hexadecimal digit; must be ≤ 64.

Text or reference to single cell containing OldText.

Text to be changed.

Text to substitute for OldText.

Which occurrence of OldText to change. If you specify InstanceNum, @SUBSTITUTE changes only that instance. Otherwise, @SUBSTITUTE changes all occurrences.

Number from 1 to 11, specifying which function to use in calculating subtotals.

List of selections or cell names to subtotal.

One or more numeric values, cell addresses, and cell references or names, separated by commas.

One or more numeric values, cell addresses, and cell references or names, separated by commas.

Overall range of one or more cell addresses, a cell reference. or name.

Numeric or string values that determine whether a cell within the Cells is added.

Cell addresses within the cells to be included in the sum. Cell values must meet Criteria in order to be included in the sum.

One or more numeric values or formulas, cell addresses, and cell references or names, separated by commas.

One or more numeric values or formulas, cell addresses, and cell references or names, separated by commas.

A cell reference or name.

A cell reference or name.

One or more numeric or cell values.

First array of numeric values.

Second array of numeric values.

First array of numeric values.

Second array of numeric values.

First array of numeric values.

Second array of numeric values.

First array of numeric values.

Second array of numeric values.

First array of numeric values.

Second array of numeric values.

First array of numeric values.

Second array of numeric values.

A numeric value representing the amount paid for an asset.

A numeric value representing the value of an asset at the end of its useful life.

A numeric value representing the expected useful life of an asset (in years).

A numeric value representing the time period for which you want to calculate depreciation.

Path and filename of the database table.

Number of columns (fields) to show from the linked table.

Number of rows (records) to show from the linked table.

A numeric value.

A value from approximately -1.789E+308 to approximately 1.789E+308.

Number representing the settlement date; must be < Maturity.

Number representing the maturity date.

Rate of discount expressed as a decimal fraction; must be ≥ 0 and ≤ 1.

Number representing the settlement date; must be < Maturity.

Number representing the maturity date.

Rate of discount expressed as a decimal fraction; must be ≥ 0 and ≤ 1.

Number representing the settlement date; must be < Maturity.

Number representing the maturity date.

The Treasury bill's price per 100 face value.

Value at which to evaluate the distribution.

Integer number of degrees of freedom; must be ≥ 1.

1 to return a one-tailed distribution; 2 to return a two-tailed distribution.

A numeric value representing the amount of the periodic payment.

A numeric value > -1, representing the periodic interest rate (the fixed interest rate per compounding period).

A numeric value representing the future value of an investment (the value the investment will reach at some point).

A number between 0 and 23, representing Hour.

A number between 0 and 59, representing Minute.

A number between 0 and 59, representing Second.

A numeric value or a string value in any valid time format, enclosed by quotation marks.

Cumulative probability value; $0 \leq \text{Prob} \leq 1$.

Number of degrees of freedom.

One or more numeric values, cell addresses, and cell references or names, separated by commas.

Cells or array to transpose; you can use a 2-D cell reference or cell name or an array constant like {1,2|3,4}.

Array of known y-values for the line y = mx + b.

Array of known x-values (optional).

Array of new x-values for which you want the corresponding y-values (optional).

Logical value (optional) that tells @TREND whether to force the constant b = 0:
If Const is TRUE or omitted, @TREND uses the actual value of b.
If Const is FALSE, @TREND sets b = 0, then adjusts the m-values so that y = mx.

A string value.

Numeric array or cells of values.

Decimal fraction of data points to exclude; $0 \leq \text{Fraction} < 1$.

Number to truncate.

Numeric value specifying precision (optional); can be from -100 through 100.

First array of numeric values.

Second array of numeric values.

1 to return a one-tailed test; 2 to return a two-tailed test.

A discrete variable specifying the type of test to conduct; 1 = a paired test; 2 = a two-sample equal variance test; 3 = a two-sample unequal variance test.

Numeric, text, logical, formula, or error value, or reference to a cell containing such a value.

A string value.

Independent cells or array of values.

Dependent cells or array of values.

Coefficient array produced by @SPLINE.

Value for which you want the corresponding y-value.

A string value.

One or more numeric or string values, cell addresses, and cell references or names, separated by commas.

One or more numeric or string values, cell addresses, and cell references or names, separated by commas.

Cost of asset; must be greater than Salvage.

Salvage value at end of asset life; can be any value.

Number of periods for asset to depreciate to salvage value; must be greater than 0.

Starting period to begin depreciation, in same units as Life; can be any positive value or 0, but not greater than Life.

Ending period for depreciation, in same units as Life; can be any value greater than StartPeriod, but not greater than Life.

Percentage of straight-line depreciation to use as the depreciation rate (optional); 200% (double-declining balance rate) if omitted. Factor can be any value greater than or equal to 0; commonly used rates are 1.25, 1.50, 1.75, and 2.

Tells @VDB whether to switch to straight-line depreciation for the remaining useful life (optional):
0 = automatically switch to straight-line depreciation when that is greater than declining-balance depreciation (default if you omit the argument)
1 = never switch to straight-line depreciation

A numeric or string value.

A 2-D cell value.

The number of the referenced column, from 0 to the number of columns in Block - 1 (the first column in Block = 0, the second = 1, and so on).

The date expressed as a serial date number.

A number that determines on what day the week begins.
1 = week begins on Sunday; default if omitted
2 = week begins on Monday
3 = week begins on Saturday

Function parameter to evaluate.

Parameter to the distribution; must be > 0.

Parameter to the distribution; must be > 0.

A numeric value (0 or 1) indicating whether to use the cumulative distribution function (1) or the probability density function (0).

Cell reference or name where values to be averaged are stored.

Cell reference or name where data affecting the weighting are stored; WeightsBlock must have the same dimensions as DataBlock.

Optional value that tells Quattro Pro how to calculate the weighted average:
0 = divide by sum of values in WeightsBlock; default if you omit the argument
1 = divide by number of values in DataBlock

Number representing a date to check.

Number representing a date to check.

Serial number for the date you're starting from.

Number of days after StartDate (if Days is positive) or before (if Days is negative).

Optional cell name or reference containing serial date numbers of holidays to exclude from the calculation.

Optional argument, in quotation marks, to tell @WORKDAY which days are weekend days. Use 0 through 6 (Monday through Sunday); for example, "45" means Friday and Saturday. The default, if you omit Weekends, is Saturday and Sunday. To specify no weekends, use "7".

One or more numeric, cell addresses, and cell references or names, separated by commas.

Cell name or reference.

Column to look in; must be the contents of a cell in the first row of Block.

Row to look in; must be the contents of a cell in the first column of Block.

Name of notebook sheet (optional).

Series of cash flows; first payment is the one occurring at the beginning of the investment; succeeding payments are discounted based on a 365-day year.

Payment corresponding to cash flow payments; first date must be the earliest date, but all other dates can be in any order.

Numeric value (optional) that estimates the internal rate of return on an investment; assumed to be 10% if omitted.

Discount rate to apply to cash flows.

Series of cash flows; first payment is the one occurring at the beginning of the investment; succeeding payments are discounted based on a 365-day year.

Payment corresponding to cash flow payments; first date must be the earliest date, but all other dates can be in any order.

First binary number.

Second binary number.

Number of binary bits used for both input and output; if omitted, Bits = number of bits in Binary1 or Binary2, whichever is greater; must be ≤ 64.

First hexadecimal number.

Second hexadecimal number.

Number of binary bits used for both input and output; if omitted, Bits = number of bits in Hex1 or Hex2, whichever is greater; 4 binary digits = 1 hexadecimal digit; must be ≤ 64.

Number to round down.

Number of digits to which you want to round X.

Number to round up.

Number of digits to which you want to round X.

Number from 0 (1900) to 199 (2099) or a standard year like 1993.

Number representing the date about which to calculate year division.

Integer representing number of divisions away from beginning of division in which date falls; can be negative.

Number of months per division; does not have to be a divisor of 12; can be longer than 1 year; must be an integer ≥ 0 (the default is 3).

Date to anchor division boundaries on (the default is January 1, 1900).

1 to indicate adherence to ends of months; 0 to indicate that ends of months are ignored (the default is 1).

A numeric value between -109571 (January 1, 1600) and 474816.9999999 (December 31, 3199).

Number representing the start date.

Number representing the end date.

Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

Number representing the settlement date.

Number representing the maturity date.

Number representing the issue date.

Coupon rate; must be ≥ 0.

Price; must be > 0.

Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

Number representing the settlement date; must be < Maturity.

Number representing the maturity date.

Settlement price; must be > 0.

Redemption value per 100 face value (must be > 0; the default is 100).

Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

Number representing the settlement date; must be < Maturity.

Number representing the maturity date.

Number representing the issue date; must be < Settle.

Coupon rate; $0 \leq \text{Coupon} \leq 1$.

Price per 100 face value.

Flag specifying which calendar to observe (0 = 30/360, 1 = actual/actual, 2 = actual/360, 3 = actual/365; the default is 0).

Date number representing the settlement date.

Date number representing the maturity date; must be greater than Settle.

Coupon rate; must be ≥ 0.

Price per $100 face value; must be > 0.

Redemption value per $100 face value (optional); must be ≥ 0; the default is 100.

Frequency of coupon payments (optional) in number of payments per year; can be 1, 2, 4, or 12; the default is 2.

Flag specifying which calendar to observe:
0 = US (NASD) 30/360; default if you omit the argument
1 = Actual/actual
2 = Actual/360
3 = Actual/365
4 = European 30/360

Given yield to be converted to another compounding frequency.

Number of times given yield is compounded per year.

Number of times target yield is compounded per year.

Number of periods for quoted yields (F1 is the default).

Number of periods for quoted yields (F2 is the default).

The date (Imperial Year, Month, Day).

The double-byte (full-width) character string.

The single-byte (half-width) character string.

The first string to be compared.

The second string to be compared.

The kanji number.

The arabic number.

1 - Long form, 2 - Long form (accounting), 3 - Short form.

A numeric array or cells of values.

A value to test against the mean of the values in Array.

Population standard deviation; if omitted, @ZTEST uses the sample standard deviation.

Using properties

Most objects in Quattro Pro have properties. For example, you can apply the Bold property to a cell, or the Name property to a sheet. You can use the @PROPERTY function or a Property macro command to automate the setting or viewing of properties. When you use the @PROPERTY function or a Property macro command, you need to use the correct syntax to view or change the values of the properties of an object. For further information, see Understanding syntax, Object precedence, and Identifying objects.

To display or change the property settings of an object:

- @PROPERTY(Object.Property)
- {GETOBJECTPROPERTY Cell, Object.Property}—see Quattro Pro Macros Help
- {SETOBJECTPROPERTY Object.Property, Value}—see Quattro Pro Macros Help
- {GETPROPERTY Cell, Property}—see Quattro Pro Macros Help
- {SETPROPERTY Property, Value}—see Quattro Pro Macros Help

To display the syntax for specific objects:

- Active Object properties
- Common Chart Object properties
- Drawn Chart Object properties
- Fixed Chart Object properties
- Common properties
- Dialog Control properties
- Menu Item properties
- Notebook Object properties
- Objects Sheet Icon properties

Related topics

Understanding syntax

The tables in this section can consist of the following columns:

Property column — lists the objects and their properties.
Argument column — shows the correct name of the property to use in the @PROPERTY function or a Property macro command.
Syntax column — shows the syntax of the property settings. Italicized items in the Syntax column describe the type of data returned; items in normal type are entered (when setting a property). If vertical bars (|) separate items, then only those items are returned or allowed in the property. For example, "Both | Window | Panel | None" means that the property setting is either Both, Window, Panel, or None; you can enter only one of these items to set the property. Another example is Precision | Type, which indicates that either the type or precision setting is listed in that position. Items in angle brackets (<>) are optional.

Related topics

Object precedence

If a situation arises where a property command could affect multiple objects, the object highest on the following list is identified:

1 Dialog box

2 Chart

3 Floating object

4 Named cell

Notes

- A property command could affect multiple objects if the objects have the same name.
- When the Objects sheet is active, you cannot identify a dialog box or chart. The icon representing the dialog box or chart is identified instead.

Related topics

Identifying objects

To identify an object, use the following Syntax:

Object	Syntax	Example	Note
Cell or Cells	*[NBName]BlockAddress*	A:A23	*[NBName]* is optional
Chart	*[NBName]ChartName*	Chart1	*[NBName]* is optional.
Chart object	*[NBName]ChartName:ObjName*		*[NBName]* is optional. ChartName: is not needed if the chart window containing the object is active.
Chart icon	*ChartName*		Chart icons can only be identified when the Objects sheet is selected.
Dialog box	*[NBName]DialogName*	Dialog1	*[NBName]* is optional.
Dialog control	*[NBName]DialogName:ObjName*	Dialog1:Bitmap1	*[NBName]* is optional. DialogName: is not needed if the dialog box containing the control is active.
Dialog icon	*DialogName*		Dialog icons can only be identified when the Objects sheet is selected.
Floating object	*[NBName]Sheet:ObjName*		*[NBName]* and Page: are optional.
Menu item	*Menupath*	/Edit/Paste special	To identify a menu item, enter its path separated by forward slashes (/)
Notebook	*[NBName]*		You can change properties of the active notebook using Active_Notebook.Property.
Sheet	*Active_Page*		You can only read or set properties of the active sheet using this syntax.
Toolbar	*[ToolbarName]*	Chart and Drawing Tools	
Toolbar control	*[ToolbarName]ObjName*		[ToolbarName] is not needed if the Toolbar containing the control is active.

Related topics

Cell Property

The following examples show how you can manipulate the property of a cell: the Numeric Format property of the cells A:A23.

Example 1 (the setting is stored in B:C32)

{GETOBJECTPROPERTY B:C32,"A:A23.Numeric_Format"}

Example 2 (the link command formats A:A23 as General)

ON Init SET General TO A:A23.Numeric_Format

@PROPERTY("A:A23.Numeric_Format")

Example 3 (formats A:A23 as Currency with 2 decimal places)

SETOBJECTPROPERTY "A:A23.Numeric_Format","Currency,2"}

Related topics

Control Property

The following examples show how you can manipulate the property of an object: the Disabled property of a bitmap button named Bitmap1 in a dialog box named Dialog1.

Example 1 (the setting is stored in B:C32)

{GETOBJECTPROPERTY B:C32,"Dialog1:Bitmap1.Disabled"}

Example 2 (the link command disables the button)

ON Init SET Yes TO Dialog1:Bitmap1.Disabled

Example 3 (enables the button.)

@PROPERTY("Dialog1:Bitmap1.Disabled")

{SETOBJECTPROPERTY "Dialog1:Bitmap1.Disabled","No"}

Note

- {SETPROPERTY}and {GETPROPERTY} work on the selected object, and just take the name of the property to manipulate.

Related topics

Common properties

Many objects contain the same property. The following are some of the more common properties:

Color

Dimension

Font

Related topics

Color property

Syntax

Red, Green, Blue

Description

Each component is an integer from 0 to 255; 0 indicates that none of the hue is present; 255 indicates maximum saturation for the hue. Black is 0,0,0 and white is 255,255,255.

You can also retrieve each of these color components individually. The following table lists the argument required to read or set an individual component of a Color property. Item is the word appearing in property dialogs and property tables that describes what color is being manipulated.

The following table lists properties and syntax for Color:

Property	**Syntax**	**Description**
Item_Color	*Item*_Color;Red, Green, Blue	The amount of red, green, and blue in the color
Red	*Item*_Color.Red	The amount of red in the color
Green	*Item*_Color.Green	The amount of green in the color
Blue	*Item*_Color.Blue	The amount of blue in the color

Related topics

Dimension property

Syntax

X, Y, Width, Height

Description

Lets you specify the precise size and position of an object relative to the window containing it.

You can read or set an individual option of the Dimension property by using the following arguments:

The following table lists properties and syntax for the Dimension:

Property	**Syntax**	**Description**
Dimension	Dimension *X, Y, Width, Height*	
X Pos	Dimension.X	The distance in pixels between the left edge of the object and the left side of the window
Y Pos	Dimension.Y	The distance in pixels between the top edge of the object and the bottom edge of the window's title bar
Width	Dimension.Width	The width of the object in pixels
Height	Dimension.Height	The height of the object in pixels

 **Note**

- For drawn objects, measure Y Pos from the top of the chart background.

 **Related topics**

Font property

Syntax

Typeface, PointSize, Bold, Italic, Underline, Strikeout

Description

Lets you specify the font attributes of the object. You can read or set an individual option of a Font property by using the following arguments.

The following table lists properties, arguments, and syntax for the Font:

Property	**Argument**	**Syntax**
Item Font	*Item*_Font	*Typeface, PointSize, Bold, Italic, Underline, Strikeout*
Typeface	*Item*_Font.Typeface	*Typeface*
Point Size	*Item*_Font.Point_Size	*PointSize*
Bold	*Item*_Font.Bold	Yes | No
Italic	*Item*_Font.Italic	Yes | No
Underline	*Item*_Font.Underline	Yes | No
Strikeout	*Item*_Font.Strikeout	Yes | No

Example

{Setproperty Cell_Font;"Arial;10;No;No;No;Yes"}

Related topics

Common Chart Object properties

Two properties are found in many chart objects:

Fill Settings

Text Settings

Related topics

Fill Settings property

Syntax

Fill_Settings;None | Solid | Pattern | Wash | Bitmap, *Type, Color1, Color2*

Description

Lets you specify the color and pattern of objects. If Fill Style is None, leave the *Type* field empty

The following table lists properties and syntax for the Fill Settings:

Property	**Syntax**
Fill Settings	Fill_Settings;None | Solid | Pattern | Wash | Bitmap, *Type, Color1, Color2*
None	Fill_Settings;None
Solid	Fill_Settings;Solid,Solid
Pattern	Fill_Settings;Pattern,Solid
	Fill_Settings;Pattern,"Tight Dots"
	Fill_Settings;Pattern,"Thick Stripes Down"
	Fill_Settings;Pattern,"Thick Stripes Up"
	Fill_Settings;Pattern,"Vertical Lines"
	Fill_Settings;Pattern,"Horizontal Lines"
	Fill_Settings;Pattern,"Horizontal Grid"
	Fill_Settings;Pattern,"Hatch 1"
	Fill_Settings;Pattern,"Diagonal 1"
	Fill_Settings;;Pattern,"Diagonal 2"
	Fill_Settings;Pattern,Vertical
	Fill_Settings;Pattern,Horizontal
	Fill_Settings;Pattern,"Loose Dots"
	Fill_Settings;Pattern,"Medium Dots"
	Fill_Settings;Pattern,Pepita
	Fill_Settings;Pattern,Scales
	Fill_Settings;Pattern,"Diagonal Grid"
	Fill_Settings;Pattern,"Hatch 2"
	Fill_Settings;Pattern,"Fuzzy Stripes Down"
	Fill_Settings;Pattern,Weave
	Fill_Settings;Pattern,"Zig Zag"
	Fill_Settings;Pattern,"Staggered Dashes"
	Fill_Settings;Pattern,Lattice
	Fill_Settings;Pattern,Bricks
Wash	Fill_Settings;Wash,"Left to right"
	Fill_Settings Wash,"Right to left"
	Fill_Settings;Wash,"Center to left and right"
	Fill_Settings;Wash,"Top to bottom"
	Fill_Settings;wash,"Bottom to top"
	Fill_Settings;Wash,"Center to top and bottom"
Bitmap	Fill_Settings;Bitmap,"Crop to fit" | "Shrink to fit" | "Tile to fit" | "3-D perspective",*BitmapName*
Color1	*Red, Blue, Green*
Color2	*Red, Blue, Green*

Example

{SETPROPERTY Fill_Settings;"Pattern;Solid;;0;0;128;255;255;255"}

Related topics

Text Settings property

Syntax

Text_Settings;Solid|Wash|Bitmap|3-D, *WashType*, *BitmapName*,*Shadow, Color1, Color2*

Description

Lets you specify whether chart text is solid, a washed tone, or a bitmap. If the style is Solid or Wash, leave the *BitmapName* field empty. *Shadow* indicates whether text should have a drop shadow (Yes) or no shadow (No).

The following table lists properties and syntax for the Text Settings:

Property	**Syntax**
Text Settings	Text_Settings;Solid|Wash|Bitmap|3-D, *WashType*, *BitmapName*,*Shadow, Color1, Color2*
Solid	Text_Settings;Solid,None,,*Shadow*
Wash	Text_Settings;Wash,"Left to right",,*Shadow*
	Text_Settings;Wash,"Right to left",,*Shadow*
	Text_Settings;Wash,"Center to left and right",,*Shadow*
	Text_Settings;Wash,"Top to bottom",, *Shadow*
	Text_Settings;Wash,"Bottom to top",,*Shadow*
	Text_Settings;Wash,"Center to top and bottom",,*Shadow*
Bitmap	Text_Settings;Bitmap,"Crop to fit"|"Shrink to fit",*BitmapName*,*Shadow*
3-D	Text_Settings;3-D, Above Left | Above Center | Above Right | Level Left | Level Center | Level Right | Bottom Left | Bottom Center | Bottom Right,,,*PerspectiveAmount*(0-100), *Shaded?*(Yes | No)
Color 1	*Red, Blue, Green*
Color 2	*Red, Blue, Green*

Related topics

Active Object properties

Syntax

If you do not know the exact name of the active object you can still view the property settings:

Active object	
The selected cells.	Active_Block
The active notebook window	Active_Notebook
The active notebook sheet	Active_Page

Description

The active object is the object that you are currently using.

The following table lists properties, arguments, and syntax for the Active Object object:

Property	**Syntax**
Active Selection	See Cell in topic Notebook Object Properties for a list of cell properties
Active Notebook	See Notebook in topic Notebook Object Properties for a list of notebook properties
Active Sheet	
Conditional Color	Conditional_Color;*Enable*, *SmallVal*, *GreatVal*, *BelColor*, *Normal*, *AboveCol*, *ERRCol*
Above Normal Color	Conditional_Color.Above_Normal_Color;0-15
Below Normal Color	Conditional_Color.Below_Normal_Color;0-15
Enable	Conditional_Color.Enable;Yes | No
ERR Color	Conditional_Color.ERR_Color;0-15
Greatest Normal Value	Conditional_Color.Greatest_Normal_Value;*GreatVa*
Normal Color	Conditional_Color.Normal_Color;0-15
Smallest Normal Value	Conditional_Color.Smallest_Normal_Value;*SmallVal*
Default Width	Default_Width;*WidthInTwips*
Display	Display;*DisplayZeros?*(Yes | No), *RowBorders?*(Yes | No), *ColBorders?*(Yes | No), *HorzGridLines?*(Yes | No), *VertGridLines?*(Yes | No)
Border Options	Display.Borders;*RowBorders?*(Yes | No), *ColBorders?*(Yes | No)
Column Borders	Display.Borders.Column_Borde rs;Yes | No
Display Zeros	Display.Display_Zeros;Yes | No
Grid Lines	Display.Grid_Lines;*HorzGridLines?*(Yes | No), *VertGridLines?*(Yes | No)
Horizontal Grid Lines	Display.Grid_Lines.Horizontal;Yes | No
Row Borders	Display.Borders.Row_Borders;Yes | No
Vertical Grid Lines	Display.Grid_Lines.Vertical;Yes | No
Name	Name;*Name*
Protection	Protection;*CellLocking?*(Yes | No), *ObjectLocking?*(Yes | No)
Enable Cell Locking	Protection.Cells;Yes | No
Enable Object Locking	Protection.Objects;Yes | No
Tab Color	Tab_Color

Zoom Factor	Zoom_Factor;10-400

Related topics

Menu Item properties

Syntax

ID Syntax	Description
/<-	The first item on the menu bar. You can include this name at the end of a menu path (for example, /File/<- identifies the first item on the File menu).
/->	The last item on the menu bar. You can include this name at the end of a menu path (for example, /Tools/Macro/-> identifies the last item on the Tools Macro menu).
/n	The nth item on the menu bar. You can include this name at the end of a menu path (for example, /Help/2 identifies the second item on the Help menu).

Description

You can manipulate the properties for each item on the menu bar. To identify an item on the menu bar, enter its path separated by forward slashes (/). Do not include ellipses (...).The following table lists some special ways to identify menu items.

The following table lists properties and syntax for the Menu Item object:

Property	Syntax
Menu Item	Menupath.Property
Checked (H)	Checked;Yes | No
Depend On (H)	Depend_On;Desktop, NoteWin, ChartWin, DiaWin, EditWin, ObjectsPage
Disabled (H)	Disabled;Yes | No
Enabled (H)	Enabled;Yes | No
Grayed (H)	Grayed;Yes | No
Help Line (H)	Help_Line;HelpLine
Hidden (H)	Hidden;Yes | No
HotKey (H)	HotKey;HotKey
Show (H)	Show;Yes | No
Title (H)	Title;Title

Note

- If a property name is followed by (H), it is a hidden property and does not appear in the object's property dialog.

Related topics

Dialog Control properties

You can view or set properties for dialog controls. For a table of arguments and syntax for the entire dialog box as an object, see Dialog Box Properties.

To view properties, arguments, and syntax for the following dialogs, choose from the following list:

Bitmap Button

Button

Check Box

Color Control

Combo Box

Edit Field

Edit Integer

File Control

Group Box

Horizontal Scroller (HScrollBar)

Label

List Box

Radio Button

Rectangle

Spin Control

Tab Control

Tab Button Control

Time Control (TimeCtrl)

Vertical Scroller (ScrollBar)

Related topics

Dialog Box properties

Syntax

[NBName]DialogName:ObjName.Property

Description

To identify a dialog box, use its name followed by a colon. To specify the property settings of a dialog box outside of the active notebook, enter the notebook name in brackets before the dialog box name:

NBName is the name of the notebook containing the dialog box;it is optional.

The following table shows properties and syntax for the entire dialog box as an object:

Property	**Syntax**	**Description**
Dialog Box (Dialog)	*[NBName]DialogName:ObjName.Property*	NBName is the name of the active notebook DialogName is the name of the dialog box containing the object ObjName is either the object ID number or the name of the object Property is one of the strings listed in the Argument column of the dialog control property tables
Dimension	Dimension	see Dimension
Disabled	Disabled;Yes | No	
Grid Options	Grid_Options;Gridsize, GridShown, GridEnabled	
Name	Name;Name	
Position Adjust	Position_Adjust;Depend, LeftRel, TopRel, RightRel, BottomRel, CenterHor, CenterVer	
Title	Title;Title	
Value (H)	Value;the current settings of all dialog/Toolbar controls that have Process Value set to Yes	

Example

@PROPERTY("Dialog1:.Title")

Note

- If a property name is followed by (H), it is a hidden property and does not appear in the object's property dialog.

Related topics

Bitmap Button properties

Syntax

[NBName]DialogName:ObjName.Property

Description

Lets you specify the properties of the bitmap button.

The following table lists properties and syntax for the Bitmap Button object:

Property	Syntax	Description
Bitmap Button	*[NBName]DialogName:ObjName.Property*	*NBName* is the name of the active notebook *DialogName* is the name of the dialog box containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Bitmap	Bitmap;BitmapName	
Button Type	Button_Type;Push Button | Radio Button | Check Box | OK Exit Button | Cancel Exit Button	
Default Button (D)	Default_Button;Yes | No	
Depend On	Depend_On;Desktop, NoteWin, ChartWin, DiaWin, EditWin, ObjectsPage	
Dimension (D)	Dimension	see Dimension
Disabled	Disabled;Yes | No	
Draw Beveling	Draw_Beveling;Yes | No	
Enabled (H, D)	Enabled;Yes | No	
Grayed	Grayed;Yes | No	
Help Line	Help_Line;HelpLine	
Hidden (D)	Hidden;Yes | No	
Label Text	Label_Text;LabelText	
Name	Name;Name	
Object Help	Object_Help;Title, Text, Context	
Object ID (R)	Object_ID;ObjectID	
Position Adjust (D)	Position_Adjust;Depend, LeftRel, TopRel, RightRel, BottomRel, CenterHor, CenterVer	
Show (H, D)	Show;Yes | No	
Tab Stop (D)	Tab_Stop;Yes | No	
Text Draw Flags	Text_Draw_Flags;Apply, HorCenter, AlignRight(Yes)orLeft(No), VertCenter, AlignBottom(Yes)orTop(No), WordBreak, SingleLine	
Value (H, D)	Value;Yes | No	

Notes

- If a property name is followed by (D), it is available for Dialogs only. All other properties are available for Toolbars and Dialogs.
- If a property name is followed by (H), it is a hidden property and does not appear in the object's property dialog.
- If a property name is followed by (R), you cannot set it with a macro command or link command; it is a read

only property.

Related topics

Button properties

Syntax

[NBName]DialogName:ObjName.Property

Description

Lets you specify the button properties.

The following table lists properties and syntax for the Button object:

Property	Syntax	Description
Button	*[NBName]DialogName:ObjName.Property*	*NBName* is the name of the active notebook *DialogName* is the name of the dialog box containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Button Type	Button_Type;Push Button | Radio Button | Check Box | OK Exit Button | Cancel Exit Button	
Default Button (D)	Default_Button;Yes | No	
Depend On	Depend_On;*Desktop*, *NoteWin*, *ChartWin*, *DiaWin*, *EditWin*, *ObjectsPage*	
Dimension (D)	Dimension	see Dimension
Disabled	Disabled;Yes | No	
Enabled (H, D)	Enabled;Yes | No	
Grayed	Grayed;Yes | No	
Help Line	Help_Line;*HelpLine*	
Hidden (D)	Hidden;Yes | No	
Label Text	Label_Text;*LabelText*	
Name	Name;*Name*	
Object Help	Object_Help;*Title*, *Text*, *Context*	
Object ID (R)	Object_ID;*ObjectID*	
Position Adjust	Position_Adjust;*Depend*, *LeftRel*, *TopRel*, *RightRel*, *BottomRel*, *CenterHor*, *CenterVer*	
Show (H, D)	Show;Yes | No	
Tab Stop (D)	Tap_Stop;Yes | No	
Text Draw Flags	Text_Draw_Flags;*Apply*, *HorCenter*, *AlignRight*(Yes)*orLeft*(No), *VertCenter*, *AlignBottom*(Yes)*orTop*(No), *WordBreak*, *SingleLine*	
Value (H, D)	Value;{Page.Display "No;Yes;Yes;Yes;Yes"}Yes | No	

Example

@PROPERTY("Dialog1:Button1.Dimension")

Notes

- If a property name is followed by (D), it is available for Dialogs only. All other properties are available for Toolbars and Dialogs.
- If a property name is followed by (H), it is a hidden property and does not appear in the object's property

dialog.

- If a property name is followed by (R), you cannot set it with a macro command or link command; it is a read only property.

Related topics

Check Box properties

Syntax

[NBName]DialogName:ObjName.Property

Description

Lets you specify the check box properties.

The following table lists properties and syntax for the Check Box object:

Property	Syntax	Description
Check Box	*[NBName]DialogName:ObjName.Property*	*NBName* is the name of the active notebook *DialogName* is the name of the dialog box containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Button Type	Button_Type;Push Button | Radio Button | Check Box | OK Exit Button | Cancel Exit Button	
Depend On	Depend_On;Desktop, NoteWin, ChartWin, DiaWin, EditWin, ObjectsPage	
Dimension	Dimension	see Dimension
Disabled	Disabled;Yes | No	
Draw to Right	Draw_to_right;Yes | No	
Enabled (H)	Enabled;Yes | No	
Grayed	Grayed;Yes | No	
Help Line	Help_Line;HelpLine	
Hidden	Hidden;Yes | No	
Label Text	Label_Text;LabelText	
Name	Name;Name	
Object Help	Object_Help;Title, Text, Context	
Object ID (R)	Object_ID;ObjectID	
Position Adjust	Position_Adjust;Depend, LeftRel, TopRel, RightRel, BottomRel, CenterHor, CenterVer	
Process Value	Process_Value;Yes | No	
Show (H)	Show;Yes | No	
Tab Stop	Tab_Stop;Yes | No	
Text Draw Flags	Text_Draw_Flags;Apply, HorCenter, AlignRight(Yes)orLeft(No), VertCenter, AlignBottom(Yes)orTop(No), WordBreak, SingleLine	
Value (H)	Value;Yes | No	

Example

@PROPERTY("Dialog1:Checkbox1.Name")

Notes

- If a property name is followed by (H), it is a hidden property and does not appear in the object's property dialog.

- If a property name is followed by (R), you cannot set it with a macro command or link command; it is a read only property.

Related topics

Color Control properties

Syntax

[NBName]DialogName:ObjName.Property

Description

Lets you specify the color control properties.

The following table lists properties and syntax for the Color Control object:

Property	Syntax	Description
Color Control	*[NBName]DialogName:ObjName.Property*	*NBName* is the name of the active notebook *DialogName* is the name of the dialog box containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
56 Colors	56_Color;Yes | No	
Custom Colors	Custom_Colors;Yes | No	
Depend On	Depend_On;*Desktop*, *NoteWin*, *ChartWin*, *DiaWin*, *EditWin*, *ObjectsPage*	
Dimension (D)	Dimension;	see Dimension
Disabled	Disabled;Yes | No	
Enabled (H, D)	Enabled;Yes | No	
Grayed	Grayed;Yes | No	
Help Line	Help_Line;*HelpLine*	
Hidden (D)	Hidden;Yes | No	
Name	Name;*Name*	
Object Help	Object_Help;*Title*, *Text*, *Context*	
Object ID (R)	Object_ID;*ObjectID*	
Position Adjust (D)	Position_Adjust;*Depend*, *LeftRel*, *TopRel*, *RightRel*, *BottomRel*, *CenterHor*, *CenterVer*	
Process Value (D)	Process_Value;Yes | No	
Show (H, D)	Show;Yes | No	
Tab Stop (D)	Tab_Stop;Yes | No	
Value (H, D)	Value;*Red*,*Green*,*Blue*	

Example

@PROPERTY("Dialog1:ColorCtl2.56_Colors")

Notes

- If a property name is followed by (D), it is available for Dialogs only. All other properties are available for Toolbars and Dialogs.
- If a property name is followed by (H), it is a hidden property and does not appear in the object's property dialog.
- If a property name is followed by (R), you cannot set it with a macro command or link command; it is a read only property.

Related topics

Combo Box properties

Syntax

[NBName]DialogName:ObjName.Property

Description

Lets you specify the combo box properties.

The following table lists properties and syntax for the Combo Box object:

Property	Syntax	Description
Combo Box	*[NBName]DialogName:ObjName.Property*	*NBName* is the name of the active notebook *DialogName* is the name of the dialog box containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Add Down Button	Add_Down_Button;Yes | No	
Allow Point Mode	Allow_Point_Mode;Yes | No	
Depend On	Depend_On;*Desktop*, *NoteWin*, *ChartWin*, *DiaWin*, *EditWin*, *ObjectsPage*	
Dimension (D)	Dimension	see Dimension
Disabled	Disabled;Yes | No	
Edit Disabled	Edit_Disabled;Yes | No	
Edit Length	Edit_Length;*Length*	
Enabled (H, D)	Enabled;Yes | No	
Grayed	Grayed;Yes | No	
Help Line	Help_Line;*HelpLine*	
Hidden (D)	Hidden;Yes | No	
History List	History_List;Yes | No	
List	List;*List*	
List Length	List_Length;*Length*	
Name	Name;*Name*	
Object Help	Object_Help;*Title*, *Text*, *Context*	
Object ID (R)	Object_ID;*ObjectID*	
Ordered	Ordered;Yes | No	
Position Adjust (D)	Position_Adjust;*Depend*, *LeftRel*, *TopRel*, *RightRel*, *BottomRel*, *CenterHor*, *CenterVer*	
Process Value (D)	Process_Value;Yes | No	
Selected	Selected;*Selected*	
Show (H, D)	Show;Yes | No	
Tab Stop (D)	Tab_Stop;Yes | No	
Terminate Dialog (D)	Terminate_Dialog;Yes | No	
Value (H, D)	Value;the text of the selected item in the combo box	

Example

@PROPERTY("Dialog1:Combobox3.Allow_Point_Mode")

Notes

- If a property name is followed by (D), it is available for Dialogs only. All other properties are available for Toolbars and Dialogs.
- If a property name is followed by (H), it is a hidden property and does not appear in the object's property dialog.
- If a property name is followed by (R), you cannot set it with a macro command or link command; it is a read only property.

Related topics

Edit Field properties

Syntax

[NBName]DialogName:ObjName.Property

Description

Lets you specify the edit field properties.

The following table lists properties and syntax for the Edit Field object:

Property	Syntax	Description
Edit Field	*[NBName]DialogName:ObjName.Property*	*NBName* is the name of the active notebook *DialogName* is the name of the dialog box containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Allow Point Mode	Allow_Point_Mode;Yes | No	
Convert Text	Convert_Text;Yes | No	
Depend On	Depend_On;*Desktop*, *NoteWin*, *ChartWin*, *DiaWin*, *EditWin*, *ObjectsPage*	
Dimension (D)	Dimension	see Dimension
Disabled	Disabled;Yes | No	
Edit Length	Edit_Length;*Length*	
Enabled (H, D)	Enabled;Yes | No	
Field Type	Field_Type;Integer | String | Real | Range | Hidden	
Grayed	Grayed;Yes | No	
Help Line	Help_Line;*HelpLine*	
Hidden (D)	Hidden;Yes | No	
Name	Name;*Name*	
Object Help	Object_Help;*Title*, *Text*, *Context*	
Object ID (R)	Object_ID;*ObjectID*	
Position Adjust (D)	Position_Adjust;*Depend*, *LeftRel*, *TopRel*, *RightRel*, *BottomRel*, *CenterHor*, *CenterVer*	
Process Value (D)	Process_Value;Yes | No	
Show (H, D)	Show;Yes | No	
Show Frame	Show_Frame;Yes | No	
Tab Stop (D)	Tab_Stop;Yes | No	
Terminate Dialog (D)	Terminate_Dialog;Yes | No	
Value (H, D)	Value;the text of the edit field	

Example

```
@PROPERTY("Dialog1:EditField4.Edit_Length")
```

Notes

- If a property name is followed by (D), it is available for Dialogs only. All other properties are available for Toolbars and Dialogs.

- If a property name is followed by (H), it is a hidden property and does not appear in the object's property dialog.
- If a property name is followed by (R), you cannot set it with a macro command or link command; it is a read only property.

Related topics

Edit Integer properties

Syntax

[NBName]DialogName:ObjName.Property

Description

Lets you specify the edit integer properties.

The following table lists properties and syntax for the Edit Integer object:

Property	Syntax	Description
Edit Integer	*[NBName]DialogName:ObjName.Property*	*NBName* is the name of the active notebook *DialogName* is the name of the dialog box containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Default	Default;*DefaultValue*	
Depend On	Depend_On;*Desktop*, *NoteWin*, *ChartWin*, *DiaWin*, *EditWin*, *ObjectsPage*	
Dimension (D)	Dimension	see Dimension
Disabled	Disabled;Yes | No	
Edit Length	Edit_Length;*Length*	
Enabled (H, D)	Enabled;Yes | No	
Field Type	Field_Type;Integer | String | Real | Range | Hidden	
Grayed	Grayed;Yes | No	
Help Line	Help_Line;*HelpLine*	
Hidden (D)	Hidden;Yes | No	
Maximum	Maximum;*MaxValue*	
Minimum	Minimum;*MinValue*	
Name	Name;*Name*	
Object Help	Object_Help;*Title*, *Text*, *Context*	
Object ID (R)	Object_ID;*ObjectID*	
Position Adjust (D)	Position_Adjust;*Depend*, *LeftRel*, *TopRel*, *RightRel*, *BottomRel*, *CenterHor*, *CenterVer*	
Process Value (D)	Process_Value;Yes | No	
Show (H, D)	Show;Yes | No	
Show Frame	Show_Frame;Yes | No	
Tab Stop (D)	Tab_Stop;Yes | No	
Terminate Dialog (D)	Terminate_Dialog;Yes | No	
Value (H, D)	Value;the value of the edit field	

Example

@PROPERTY("Dialog1:EditInteger2.Field_Type")

Notes

- If a property name is followed by (D), it is available for Dialogs only. All other properties are available for

Toolbars and Dialogs.

- If a property name is followed by (H), it is a hidden property and does not appear in the object's Property dialog.
- If a property name is followed by (R), you cannot set it with a macro command or link command; it is a read only property.

Related topics

File Select Control properties

Syntax

[NBName]DialogName:ObjName.Property

Description

Lets you specify the file select control properties.

The following table lists properties and syntax for the File Select Control object:

Property	Syntax	Description
File Select Control	*[NBName]DialogName:ObjName.Property*	*NBName* is the name of the active notebook *DialogName* is the name of the dialog box containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Depend On	Depend_On;*Desktop*, *NoteWin*, *ChartWin*, *DiaWin*, *EditWin*, *ObjectsPage*	
Dialog Type	Dialog_Type;Open | Save	
Dimension (D)	Dimension	see Dimension
Disabled	Disabled;Yes | No	
Edit Disabled	Edit_Disabled;Yes | No	
Enabled (H, D)	Enabled;Yes | No	
File TypeList	FileTypeList;*FileTypeList*	
Grayed	Grayed;Yes | No	
Help Line	Help_Line;*HelpLine*	
Hidden (D)	Hidden;Yes | No	
Name	Name;*Name*	
Object Help	Object_Help;*Title, Text, Context*	
Object ID	ObjectID;*ObjectID*	
Position Adjust (D)	Position_Adjust;*Depend*, *LeftRel*, *TopRel*, *RightRel*, *BottomRel*, *CenterHor*, *CenterVer*	
Process Value (D)	Process_Value;Yes | No	
Show (H, D)	Show;Yes | No	
Tab Stop (D)	Tab_Stop;Yes | No	
Terminate Dialog (D)	Terminate_Dialog;Yes | No	
Title	Title;*Title*	
Value (H, D)	Value;The full path and filename	

Example

@PROPERTY("Dialog1:FileSelCtrl6.Dialog_Type")

Notes

- If a property name is followed by (D), it is available for Dialogs only. All other properties are available for Toolbars and Dialogs.
- If a property name is followed by (H), it is a hidden property and does not appear in the object's Property dialog.
- If a property name is followed by (R), you cannot set it with a macro command or link command; it is a read

only property.

Related topics

Group Box properties

Syntax

[NBName]DialogName:ObjName.Property

Description

Lets you specify the group box properties.

The following table lists properties and syntax for the Group Box object:

Property	Syntax	Description
Group Box	*[NBName]DialogName:ObjName.Property*	*NBName* is the name of the active notebook *DialogName* is the name of the dialog box containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Attach Child	Attach_Child;Yes | No	
Dimension	Dimension	see Dimension
Disabled	Disabled;Yes | No	
Enabled (H)	Enabled;Yes | No	
Group Text	Group_Text;*Text*	
Hidden	Hidden;Yes | No	
Name	Name;*Name*	
Object Help	Object_Help;*Title*, *Text*, *Context*	
Object ID (R)	Object_ID;*ObjectID*	
Position Adjust	Position_Adjust;*Depend*, *LeftRel*, *TopRel*, *RightRel*, *BottomRel*, *CenterHor*, *CenterVer*	
Process Value	Process_Value;Yes | No	
Selected	Selected;*Number*	
Show (H)	Show;Yes | No	
Value (H)	Value;The type of control selecte	

Example

```
@PROPERTY("Dialog1:GroupBox10.Group_Text")
```

Notes

- If a property name is followed by (H), it is a hidden property and does not appear in the object's Property dialog.
- If a property name is followed by (R), you cannot set it with a macro command or link command; it is a read only property.

Related topics

Horizontal Scroller properties

Syntax

[NBName]DialogName:ObjName.Property

Description

Lets you specify the horizontal scroller properties.

The following table lists properties and syntax for the Horizontal Scroller object:

Property	Syntax	Description
Horizontal Scroller (HScrollBar)	*[NBName]DialogName:ObjName.Property*	*NBName* is the name of the active notebook *DialogName* is the name of the dialog box containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Depend On	Depend_On;*Desktop*, *NoteWin*, *ChartWin*, *DiaWin*, *EditWin*, *ObjectsPage*	
Dimension (D)	Dimension	see Dimension
Disabled	Disabled;Yes | No	
Enabled (H, D)	Enabled;Yes | No	
Grayed	Grayed;Yes | No	
Help Line	Help_Line;*HelpLine*	
Hidden (D)	Hidden;Yes | No	
Name	Name;*Name*	
Object Help	Object_Help;*Title*, *Text*, *Context*	
Object ID (R)	Object_ID;*ObjectID*	
Parameters	Parameters;*Min*, *Max*, *Line*, *Page*, *Time*	
Position Adjust (D)	Position_Adjust;*Depend*, *LeftRel*, *TopRel*, *RightRel*, *BottomRel*, *CenterHor*, *CenterVer*	
Process Value (D)	Process_Value;Yes | No	
Show (H, D)	Show;Yes | No	
Tab Stop (D)	Tab_Stop;Yes | No	
Value (H, D)	Value;the numeric value for the position of the scroll thumb	

Example

@PROPERTY("Dialog1:Hscrollbar1.Show")

Notes

- If a property name is followed by (D), it is available for Dialogs only. All other properties are available for Toolbars and Dialogs.
- If a property name is followed by (H), it is a hidden property and does not appear in the object's Property dialog.
- If a property name is followed by (R), you cannot set it with a macro command or link command; it is a read only property.

Related topics

Label properties

Syntax

[NBName]DialogName:ObjName.Property

Description

Lets you specify the label properties.

The following table lists properties and syntax for the Label object:

Property	Syntax	Description
Label	*[NBName]DialogName:ObjName.Property*	*NBName* is the name of the active notebook *DialogName* is the name of the dialog box containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Depend On	Depend_On;*Desktop*, *NoteWin*, *ChartWin*, *DiaWin*, *EditWin*, *ObjectsPage*	
Dimension (D)	Dimension	see Dimension
Disabled	Disabled;Yes | No	
Enabled (H, D)	Enabled;Yes | No	
Grayed	Grayed;Yes | No	
Hidden (D)	Hidden;Yes | No	
Label Font	Label_Font;	
Label Text	Label_Text;*LabelText*	
Name	Name;*Name*	
Object ID (R)	Object_ID;*ObjectID*	
Position Adjust (D)	Position_Adjust;*Depend*, *LeftRel*, *TopRel*, *RightRel*, *BottomRel*, *CenterHor*, *CenterVer*	
Process Value (D)	Process_Value;Yes | No	
Show (H, D)	Show;Yes | No	
Text Draw Flags	Text_Draw_Flags;*Apply*, *HorCenter*, *AlignRight*(Yes) *or Left*(No), *VertCenter*, *AlignBottom*(Yes)*or Top*(No), *WordBreak*, *SingleLine*	
Value (H, D)	Value;*LabelText*	

Example

```
@PROPERTY("Dialog1:label3.label_font")
```

Notes

- If a property name is followed by (D), it is available for Dialogs only. All other properties are available for Toolbars and Dialogs.
- If a property name is followed by (H), it is a hidden property and does not appear in the object's Property dialog.
- If a property name is followed by (R), you cannot set it with a macro command or link command; it is a read only property.

Related topics

List Box properties

Syntax

[NBName]DialogName:ObjName.Property

Description

Lets you specify the list box properties.

The following table lists properties and syntax for the List Box object:

Property	Syntax	Description
List Box	*[NBName]DialogName:ObjName.Property*	*NBName* is the name of the active notebook *DialogName* is the name of the dialog box containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Attach Child	Attach_Child;Yes | No	
Depend On	Depend_On;*Desktop*, *NoteWin*, *ChartWin*, *DiaWin*, *EditWin*, *ObjectsPage*	
Dimension	Dimension	see Dimension
Disabled	Disabled;Yes | No	
Enabled (H)	Enabled;Yes | No	
Grayed	Grayed;Yes | No	
Help Line	Help_Line;*HelpLine*	
Hidden	Hidden;Yes | No	
List	List;*List*	
Name	Name;*Name*	
Number of Columns	Number_of_Columns;*Number*	
Object Help	Object_Help;*Title*, *Text*, *Context*	
Object ID (R)	Object_ID;*ObjectID*	
Ordered	Ordered;Yes | No	
Position Adjust	Position_Adjust;*Depend*, *LeftRel*, *TopRel*, *RightRel*, *BottomRel*, *CenterHor*, *CenterVer*	
Process Value	Process_Value;Yes | No	
Selected	Selected;*NumberSelected*	
Selection Text	Selection_Text;*SelectionText*	
Show (H)	Show;Yes | No	
Tab Stop	Tab_Stop;Yes | No	
Value (H)	Value;the selected item in the list box	

Example

@PROPERTY("Dialog1:listbox2.number_of_columns")

Notes

- If a property name is followed by (H), it is a hidden property and does not appear in the object's Property

dialog.

- If a property name is followed by (R), you cannot set it with a macro command or link command; it is a read only property.

Related topics

Radio Button properties

Syntax

[NBName]DialogName:ObjName.Property

Description

Lets you specify the radio button properties.

The following table lists properties and syntax for the Radio Button object:

Property	Syntax	Description
Radio Button	*[NBName]DialogName:ObjName.Property*	*NBName* is the name of the active notebook *DialogName* is the name of the dialog box containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Button Type	Button_Type;Push Button | Radio Button | Check Box | OK Exit Button | Cancel Exit Button	
Depend On	Depend_On;*Desktop*, *NoteWin*, *ChartWin*, *DiaWin*, *EditWin*, *ObjectsPage*	
Dimension	Dimension	see Dimension
Disabled	Disabled;Yes | No	
Draw to Right	Draw_to_right;Yes | No	
Enabled (H)	Enabled;Yes | No	
Grayed	Grayed;Yes | No	
Help Line	Help_Line;*HelpLine*	
Hidden	Hidden;Yes | No	
Label Text	Label_Text;*LabelText*	
Name	Name;*Name*	
Object Help	Object_Help;*Title*, *Text*, *Context*	
Object ID (R)	Object_ID;*ObjectID*	
Position Adjust	Position_Adjust;*Depend*, *LeftRel*, *TopRel*, *RightRel*, *BottomRel*, *CenterHor*, *CenterVer*	
Process Value	Process_Value;Yes | No	
Show (H)	Show;Yes | No	
Tab Stop	Tab_Stop;Yes | No	
Text Draw Flags	Text_Draw_Flags;*Apply*, *HorCenter*, *AlignRight*(Yes)*orLeft*(No), *VertCenter*, *AlignBottom*(Yes)*orTop*(No), *WordBreak*, *SingleLine*	
Value (H)	Value;Yes | No	

Example

@PROPERTY("Dialog1:radiobutton3.button_type")

Notes

- If a property name is followed by (H), it is a hidden property and does not appear in the object's Property dialog.

- If a property name is followed by (R), you cannot set it with a macro command or link command; it is a read only property.

Related topics

Rectangle properties

Syntax

[NBName]DialogName:ObjName.Property

Description

Lets you specify the rectangle properties.

The following table lists properties and syntax for the Rectangle object:

Property	Syntax	Description
Rectangle	*[NBName]DialogName:ObjName.Property*	*NBName* is the name of the active notebook *DialogName* is the name of the dialog box containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Attach Child (D)	Attach_Child;Yes | No	
Depend On	Depend_On;*Desktop*, *NoteWin*, *ChartWin*, *DiaWin*, *EditWin*, *ObjectsPage*	
Dimension (D)	Dimension	see Dimension
Disabled	Disabled;Yes | No	
Enabled (H)	Enabled;Yes | No	
Fill Color	Fill_Color	
Frame Color	Frame_Color	
Grayed	Grayed;Yes | No	
Hidden (D)	Hidden;Yes | No	
Name	Name;*Name*	
Object ID (R)	Object_ID;*ObjectID*	
Position Adjust (D)	Position_Adjust;*Depend*, *LeftRel*, *TopRel*, *RightRel*, *BottomRel*, *CenterHor*, *CenterVer*	
Rectangle Style	Rectangle_Style;Plain | Framed | Beveled Out | Beveled In | Transparent | Engraved	
Show (H, D)	Show;Yes | No	

Example

@PROPERTY("Dialog1:rectangle3.fill_color")

Notes

- If a property name is followed by (D), it is available for Dialogs only. All other properties are available for Toolbars and Dialogs.
- If a property name is followed by (H), it is a hidden property and does not appear in the object's Property dialog.
- If a property name is followed by (R), you cannot set it with a macro command or link command; it is a read only property.

Related topics

Spin Control properties

Syntax

[NBName]DialogName:ObjName.Property

Description

Lets you specify the spin control properties.

The following table lists properties and syntax for the Spin Control object:

Property	Syntax	Description
Spin Control	*[NBName]DialogName:ObjName.Property*	*NBName* is the name of the active notebook *DialogName* is the name of the dialog box containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Attach Child (D)	Attach_Child;Yes | No	
Default	Default;*DefaultValue*	
Depend On	Depend_On;*Desktop*, *NoteWin*, *ChartWin*, *DiaWin*, *EditWin*, *ObjectsPage*	
Dimension (D)	Dimension	see Dimension
Disabled	Disabled;Yes | No	
Edit Length	Edit_Length;*Length*	
Enabled (H, D)	Enabled;Yes | No	
Grayed	Grayed;Yes | No	
Help Line	Help_Line;*HelpLine*	
Hidden (D)	Hidden;Yes | No	
Maximum	Maximum;*MaxValue*	
Minimum	Minimum;*MinValue*	
Name	Name;*Name*	
Object Help	Object_Help;*Title*, *Text*, *Context*	
Object ID (R)	Object_ID;*ObjectID*	
Position Adjust (D)	Position_Adjust;*Depend*, *LeftRel*, *TopRel*, *RightRel*, *BottomRel*, *CenterHor*, *CenterVer*	
Process Value (D)	Process_Value;Yes | No	
Show (H, D)	Show;Yes | No	
Tab Stop (D)	Tab_Stop;Yes | No	
Terminate Dialog (D)	Terminate_Dialog;Yes | No	
Value (H, D)	Value;the current value of the spin control	

Example

@PROPERTY("Dialog1:spinctrl3.maximum")

Notes

- If a property name is followed by (D), it is available for Dialogs only. All other properties are available for

Toolbars and Dialogs.

- If a property name is followed by (H), it is a hidden property and does not appear in the object's Property dialog.
- If a property name is followed by (R), you cannot set it with a macro command or link command; it is a read only property.

Related topics

Tab Control properties

Syntax

[NBName]DialogName:ObjName.Property

Description

Lets you specify the tab control properties.

The following table lists properties and syntax for the Tab Control object:

Property	Syntax	Description
Tab Control	*[NBName]DialogName:ObjName.Property*	*NBName* is the name of the active notebook *DialogName* is the name of the dialog box containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Active Sheet	Active_Page;*ActivePage*	
Depend On	Depend_On;*Desktop*, *NoteWin*, *ChartWin*, *DiaWin*, *EditWin*, *ObjectsPage*	
Dimension	Dimension	see Dimension
Disabled	Disabled;Yes | No	
Enabled (H)	Enabled;Yes | No	
Grayed	Grayed;Yes | No	
Help Line	Help_Line;*HelpLine*	
Hidden	Hidden;Yes | No	
Name	Name;*Name*	
Object Help	Object_Help;*Title, Text, Context*	
Object ID	Object_ID;*ObjectID*	
Position Adjust	Position_Adjust;*Depend*, *LeftRel*, *TopRel*, *RightRel*, *BottomRel*, *CenterHor*, *CenterVer*	
Process Value	Process_Value;Yes | No	
Sheet List	Page_List;	
Show (H)	Show;Yes | No	
Tab Stop	Tab_Stop;Yes | No	
Value (H)	Value;the current value of the spin control	

Example

@PROPERTY("Dialog1:TabCtrl3.Active_Page")

Note

- If a property name is followed by (H), it is a hidden property and does not appear in the object's Property dialog.

Related topics

Tab Button Control properties

Syntax

[NBName]DialogName:ObjName.Property

Description

Lets you specify the tab button control properties.

The following table lists properties and syntax for the Tab Button Control object:

Property	Syntax	Description
Tab Button Control	*[NBName]DialogName:ObjName.Property*	*NBName* is the name of the active notebook *DialogName* is the name of the dialog box containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Grayed	Grayed;Yes | No	
Help Line	Help_Line;*HelpLine*	
Name	Name;*Name*	
Object Help	Object_Help;*Title, Text, Context*	
Object ID	Object_ID;*ObjectID*	
Sheet Name Text	PageName_Text;*PageNameText*	

Example

```
@PROPERTY("Dialog1:TabButton2.Object_ID")
```

Related topics

Time Control properties

Syntax

[NBName]DialogName:ObjName.Property

Description

Lets you specify the time control properties.

The following table lists properties and syntax for the Time Control object:

Property	Syntax	Description
Time Control (TimeCtrl)	*[NBName]DialogName:ObjName.Property*	*NBName* is the name of the active notebook *DialogName* is the name of the dialog box containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Alarm On	Alarm_On;Yes | No	
Alarm Time	Alarm_Time;*Hour*, *Minute*, *Second*	
Hour	Alarm_Time.Hour;*Hour*	
Minute	Alarm_Time.Minute;*Minute*	
Second	Alarm_Time.Second;*Second*	
Attach Child (D)	Attach_Child;Yes | No	
Current Time (R)	Current_Time;*Hour*, *Minute*, *Second*	
Hour	Current_Time.Hour;*Hour*	
Minute	Current_Time.Minute;*Minute*	
Second	Current_Time.Second;*Second*	
Depend On	Depend_On;*Desktop*, *NoteWin*, *ChartWin*, *DiaWin*, *EditWin*, *ObjectsPage*	
Dimension (D)	Dimension	see Dimension
Help Line	Help_Line;*HelpLine*	
Hidden (D)	Hidden;Yes | No	
Interval in Units	Interval_In_Units;*Number*	
Name	Name;*Name*	
Object Help	Object_Help;*Title*, *Text*, *Context*	
Object ID (R)	Object_ID;*ObjectID*	
Position Adjust (D)	Position_Adjust;*Depend*, *LeftRel*, *TopRel*, *RightRel*, *BottomRel*, *CenterHor*, *CenterVer*	
Process Value (D)	Process_Value;Yes | No	
Show (H, D)	Show;Yes | No	
Show Time	Show_Time;Yes | No	
Timer On	Timer_On;Yes | No	
Units in Milliseconds	Units_in_Milliseconds;*Number*	

Example

@PROPERTY("Dialog1:TimeCtrl4.alarm_on")

Notes

- If a property name is followed by (D), it is available for Dialogs only. All other properties are available for

Toolbars and Dialogs.

- If a property name is followed by (H), it is a hidden property and does not appear in the object's Property dialog.
- If a property name is followed by (R), you cannot set it with a macro command or link command; it is a read only property.

Related topics

Vertical Scroller properties

Syntax

[NBName]DialogName:ObjName.Property

Description

Lets you specify the vertical scroller properties.

The following table lists properties and syntax for the Vertical Scroller object:

Property	Syntax	Description
Vertical Scroller (ScrollBar)	*[NBName]DialogName:ObjName.Property*	*NBName* is the name of the active notebook *DialogName* is the name of the dialog box containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Depend On	Depend_On;*Desktop*, *NoteWin*, *ChartWin*, *DiaWin*, *EditWin*, *ObjectsPage*	
Dimension	Dimension	see Dimension
Disabled	Disabled;Yes | No	
Enabled (H)	Enabled;Yes | No	
Grayed	Grayed;Yes | No	
Help Line	Help_Line;*HelpLine*	
Hidden	Hidden;Yes | No	
Name	Name;*Name*	
Object Help	Object_Help;*Title*, *Text*, *Context*	
Object ID (R)	Object_ID;*ObjectID*	
Parameters	Parameters;*Min*, *Max*, *Line*, *Page*, *Time*	
Position Adjust	Position_Adjust;*Depend*, *LeftRel*, *TopRel*, *RightRel*, *BottomRel*, *CenterHor*, *CenterVer*	
Process Value	Process_Value;Yes | No	
Show (H)	Show;Yes | No	
Tab Stop	Tab_Stop;Yes | No	
Value (H)	Value;the numeric value for the position of the scroll thumb	

Example

@PROPERTY("Dialog1:scrollbar2.parameters")

Notes

- If a property name is followed by (H), it is a hidden property and does not appear in the object's Property dialog.
- If a property name is followed by (R), you cannot set it with a macro command or link command; it is a read only property.

Related topics

Drawn Chart Object properties

Drawn objects in a chart are created by the Chart tools; see Fixed Chart Object Properties for a list of fixed chart objects. Note that some properties of a drawn object only appear in a property dialog when Quattro Pro is loaded with the /D command line switch.

To view properties, arguments, and syntax for the following chart annotation objects, choose from the following list:

Arrow

Ellipse

Freehand Polygon

Freehand Polyline

Line

Polygon

Polyline

Rectangle

Rounded Rectangle

Selection

Text Box

Related topics

Arrow properties

Syntax

[*NBName*]*ChartName*:*ObjName.Property*

Description

Lets you specify the arrow properties.

The following table lists properties and syntax for the Arrow object:

Property	Syntax	Description
Arrow	[*NBName*]*ChartName*:*ObjName.Property*	*NBName* is the name of the active notebook *ChartName* is the name of the chart containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Border Settings	Border_Settings;S0W1 | S0W2 | S0W3 | S0W4 | S1W1 | S2W1 | S3W1 | S4W1 | S5W1, *Red, Blue, Green*	
Dimension (H)	Dimension	see Dimension
Fill Settings	Fill_Settings	see Common Chart Object properties
Name (H)	Name;*Name*	
Object ID (H, R)	Object_ID;*ObjectID*	

Example

{SetProperty "Object_Name";"Arrow2"}

Notes

- If a property name is followed by (H), it is a hidden property and does not appear in the object's Property dialog.
- If a property name is followed by (R), you cannot set it with a macro command or link command; it is a read only property.

Related topics

Cell properties

Description

Lets you specify the cell properties.

The following table lists properties and syntax for the Cell object:

Property	Syntax	Description
Cell	[*NBName*]*ChartName*:*ObjName.Property*	*NBName* is the name of the active notebook *ChartName* is the name of the chart containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Dimension (H)	Dimension	see Dimension
Display	Display;*RowBorders?*(Yes | No), *ColBorders?*(Yes | No), *HorzGridLines?* (Yes | No), *VertGridLines?*(Yes | No), *AspectRatio?*(Yes | No)	
Name (H)	Name;*Name*	
Object ID (H, R)	Object_ID;*ObjectID*	
Selection	Selection;Block	

Example

```
{GETOBJECTPROPERTY B:C32,"A:A23.Numeric_Format"}
```

Notes

- If a property name is followed by (H), it is a hidden property and does not appear in the object's property dialog.
- If a property name is followed by (R), you cannot set it with a macro command or link command; it is a read-only property.

Related topics

Ellipse properties

Syntax

[*NBName*]*ChartName*:*ObjName.Property*

Description

Lets you specify the ellipse properties.

The following table lists properties and syntax for the Ellipse object:

Property	Syntax	Description
Ellipse	[*NBName*]*ChartName*:*ObjName.Property*	*NBName* is the name of the active notebook *ChartName* is the name of the chart containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Border Settings	Border_Settings;S0W1 | S0W2 | S0W3 | S0W4 | S1W1 | S2W1 | S3W1 | S4W1 | S5W1, *Red, Blue, Green*	
Dimension (H)	Dimension	see Dimension
Fill Settings	Fill_Settings	see Common Chart Object properties
Name (H)	Name;*Name*	
Object ID (H, R)	Object_ID;*ObjectID*	
Object Name	Object_Name;*ObjName*	
Protection	Protection;Yes | No	

Example

{SETOBJECTPROPERTY "A:Ellipse1.Fill_Settings";"Bitmap,Shrink to fit, C:\Corel\Suite9\Graphics\Pictures\Business\World.bmp"}

Notes

- If a property name is followed by (H), it is a hidden property and does not appear in the object's Property dialog.
- If a property name is followed by (R), you cannot set it with a macro command or link command;it is a read only property.

Related topics

Freehand Polygon properties

Syntax

[*NBName*]*ChartName*:*ObjName.Property*

Description

Lets you specify the freehand polygon properties.

The following table lists properties and syntax for the Freehand Polygon object:

Property	Syntax	Description
Freehand Polygon	[*NBName*]*ChartName*:*ObjName.Property*	*NBName* is the name of the active notebook *ChartName* is the name of the chart containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Border Settings	Border_Settings;S0W1 | S0W2 | S0W3 | S0W4 | S1W1 | S2W1 | S3W1 | S4W1 | S5W1, *Red, Blue, Green*	
Dimension (H)	Dimension	see Dimension
Fill Settings	Fill_Settings	see Common Chart Object properties
Name (H)	Name;	
Object ID (H, R)	Object_ID;*ObjectID*	

Notes

- If a property name is followed by (H), it is a hidden property and does not appear in the object's Property dialog.
- If a property name is followed by (R), you cannot set it with a macro command or link command; it is a read only property.

Related topics

Freehand Polyline properties

Syntax

[*NBName*]*ChartName*:*ObjName.Property*

Description

Lets you specify the freehand polyline properties.

The following table lists properties and syntax for the Freehand Polyline object:

Property	Syntax	Description
Freehand Polyline	[*NBName*]*ChartName*:*ObjName.Property*	*NBName* is the name of the active notebook *ChartName* is the name of the chart containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Dimension (H)	Dimension	see Dimension
Line Setting	Line_Setting;S0W1 | S0W2 | S0W3 | S0W4 | S1W1 | S2W1 | S3W1 | S4W1 | S5W1, *Red, Blue, Green*	
Name (H)	Name;*Name*	
Object ID (H, R)	Object_ID;*ObjectID*	

Notes

- If a property name is followed by (H), it is a hidden property and does not appear in the object's Property dialog.
- If a property name is followed by (R), you cannot set it with a macro command or link command; it is a read only property.

Related topics

Line properties

Syntax

[*NBName*]*ChartName*:*ObjName.Property*

Description

Lets you specify the line properties.

The following table lists properties and syntax for the Line object:

Property	Syntax	Description
Line	[*NBName*]*ChartName*:*ObjName.Property*	*NBName* is the name of the active notebook *ChartName* is the name of the chart containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Dimension (H)	Dimension	see Dimension
Line Setting	Line_Setting;S0W1 | S0W2 | S0W3 | S0W4 | S1W1 | S2W1 | S3W1 | S4W1 | S5W1, *Red, Blue, Green*	
Name (H)	Name;*Name*	
Object ID (H, R)	Object_ID;*ObjectID*	

Example

{SetProperty "Object_Name";"Test"}

Notes

- If a property name is followed by (H), it is a hidden property and does not appear in the object's Property dialog.
- If a property name is followed by ®, you cannot set it with a macro command or link command; it is a read only property.

Related topics

Polygon Properties

Syntax

[*NBName*]*ChartName*:*ObjName.Property*

Description

Lets you specify the polygon properties.

The following table lists properties and syntax for the Polygon object:

Property	Syntax	Description
Polygon	[*NBName*]*ChartName*:*ObjName.Property*	*NBName* is the name of the active notebook *ChartName* is the name of the chart containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Border Settings	Border_Settings;S0W1 | S0W2 | S0W3 | S0W4 | S1W1 | S2W1 | S3W1 | S4W1 | S5W1, *Red, Blue, Green*	
Dimension (H)	Dimension	see Dimension
Fill Settings	Fill_Settings	see Common Chart Object properties
Name (H)	Name;*Name*	
Object ID (H, R)	Object_ID;*ObjectID*	

Notes

- If a property name is followed by (H), it is a hidden property and does not appear in the object's Property dialog.
- If a property name is followed by (R), you cannot set it with a macro command or link command; it is a read only property.

Related topics

Polyline properties

Syntax

[*NBName*]*ChartName*:*ObjName.Property*

Description

Lets you specify the polyline properties.

The following table lists properties and syntax for the Polyline object:

Property	Syntax	Description
Polyline	[*NBName*]*ChartName*:*ObjName.Property*	*NBName* is the name of the active notebook *ChartName* is the name of the chart containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Dimension (H)	Dimension	see Dimension
Line Setting	Line_Setting;S0W1 | S0W2 | S0W3 | S0W4 | S1W1 | S2W1 | S3W1 | S4W1 | S5W1, *Red, Blue, Green*	
Name (H)	Name;*Name*	
Object ID (H, R)	Object_ID;*ObjectID*	

Notes

- If a property name is followed by (H), it is a hidden property and does not appear in the object's Property dialog.
- If a property name is followed by (R), you cannot set it with a macro command or link command; it is a read only property.

Related topics

Rectangle properties

Syntax

[*NBName*]*ChartName*:*ObjName.Property*

Description

Lets you specify the rectangle properties.

The following table lists properties and syntax for the Rectangle object:

Property	Syntax	Description
Rectangle	[*NBName*]*ChartName*:*ObjName.Property*	*NBName* is the name of the active notebook *ChartName* is the name of the chart containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Border Settings	Border_Settings;S0W1 | S0W2 | S0W3 | S0W4 | S1W1 | S2W1 | S3W1 | S4W1 | S5W1, *Red, Blue, Green*	
Dimension (H)	Dimension	see Dimension
Fill Settings	Fill_Settings	see Common Chart Object properties
Name (H)	Name;*Name*	
Object ID (H, R)	Object_ID;*ObjectID*	

Note

- If a property name is followed by (H), it is a hidden property and does not appear in the object's Property dialog.

Related topics

Rounded Rectangle properties

Syntax

[*NBName*]*ChartName*:*ObjName.Property*

Description

Lets you specify the rounded rectangle properties.

The following table lists properties and syntax for the Rounded Rectangle object:

Property	Syntax	Description
Rounded Rectangle	[*NBName*]*ChartName*:*ObjName.Property*	*NBName* is the name of the active notebook *ChartName* is the name of the chart containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Border Settings	Border_Settings;S0W1 | S0W2 | S0W3 | S0W4 | S1W1 | S2W1 | S3W1 | S4W1 | S5W1, *Red, Blue, Green*	
Dimension (H)	Dimension	see Dimension
Fill Settings	Fill_Settings	see Common Chart Object properties
Name (H)	Name;*Name*	
Object ID (H, R)	Object_ID;*ObjectID*	

Notes

- If a property name is followed by (H), it is a hidden property and does not appear in the object's Property dialog.
- If a property name is followed by (R), you cannot set it with a macro command or link command; it is a read only property.

Related topics

Text Box properties

Syntax

[*NBName*]*ChartName*:*ObjName.Property*

Description

Lets you specify the text box properties.

The following table lists properties and syntax for the Text Box object:

Property	Syntax	Description
Text Box	[*NBName*]*ChartName*:*ObjName.Property*	*NBName* is the name of the active notebook *ChartName* is the name of the chart containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Alignment	Alignment;Left | Right | Center, *WordWrap*, *TabStops*	
Box Settings	Box_Settings;No box | Single outline | Rounded corners | Double outline | Thick outline 1 | Thick rounded corners | Three dimensional | Thick outline 2 | Bevel out | Shadowed | Thick outline 3 | Bevel in, *Red, Blue, Green*	
Chart Button	Chart_Button;*GotoSlide?*, *SlideName*, *Effect*, *Duration*, Slow | Med | Fast, *Overlay?*, *RunMacro?*, *MacroText*	
Dimension (H)	Dimension	
Fill Settings	Fill_Settings	
Name (H)	Name;*Name*	
Object ID (H, R)	Object_ID;*ObjectID*	
Text Font	Text_Font	see Font
Text Settings	Text_Settings	see Common Chart Object properties
Value (H)	Value;Contents of the textbox	

Example

{SetProperty Box_Settings;"Thick outline 1;0;0;0"}

Notes

- If a property name is followed by (H), it is a hidden property and does not appear in the object's Property dialog.
- If a property name is followed by (R), you cannot set it with a macro command or link command; it is a read only property.

Related topics

Fixed Chart Object properties

A fixed object in a chart is an object not created with the Chart toolbar. See Chart Window Properties and Chart Background Properties for chart properties. For a list of drawn chart objects (such as arrows), see Chart Annotation Object Properties.

To view properties, arguments, and syntax for the following fixed chart objects, choose from the following list:

Area Fill

Area Series

Axis Title

Bar Series

Bullet Series

Chart Background

Chart Legend

Chart Pane

Chart Subtitle

Chart Title Box

Chart Window

Column Chart

Float Series

Line Series

Map Legend

Map Properties

Pie Chart

Series Label

X-Axis

Y-Axis

Related topics

Fixed Chart Object Names

The following table lists the names for the objects in a fixed chart:

Property	Description
G$Base	Base of a 3-D chart
G$BulletSeries[*n*]	*n*th level of bulleted text in a bullet chart;the first level is 1, the second is 2
G$Graph	Background of the chart window
G$LeftWall	Left wall of a 3-D chart grid
G$Legend	Chart legend
G$Series[x,y]	yth data point of the xth series in the chart
G$SeriesLabel	Series labels
G$Title	Title, subtitle, and title box of the chart
G$X1Axis	x-axis
G$X1Title	x-axis title
G$Y1Axis	Primary y-axis
G$Y1Title	Primary y-axis title
G$Y2Axis	Secondary y-axis
G$Y2Title	Secondary y-axis title

Related topics

Area Fill properties

Syntax

[*NBName*]*ChartName*:*ObjName.Property*

Description

Lets you specify the area fill properties.

The following table lists properties and syntax for the Area Fill object:

Property	**Syntax**	**Description**
Area Fill	[*NBName*]*ChartName*:*ObjName.Property*	*NBName* is the name of the active notebook *ChartName* is the name of the chart containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Border Settings	Border_Setting s;S0W1 | S0W2 | S0W3 | S0W4 | S1W1 | S2W1 | S3W1 | S4W1 | S5W1, *Red, Blue, Green*	
Fill Settings	Fill_Settings	see Common Chart Object properties

Example

{SelectObject "G$RightWall"}

{SetProperty Fill_Settings;"Pattern;Horizontal Grid;;172;87;86;255;255;255"}

Related topics

Area Series properties

Syntax

[*NBName*]*ChartName*:*ObjName.Property*

Description

Lets you specify the area series properties.

The following table lists properties and syntax for the Area Series object:

Property	Syntax	Description
Area Series	[*NBName*]*Sheet*:*ObjName.Property*	*NBName* is the name of the active notebook *ChartName* is the name of the chart containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Analyze	Analyze;None | Aggregration | Moving Average | Linear Fit | Exponential Fit,...	Note: If Aggregration, other arguments are:...<*Table*>, *Show in Legend?*(0 | 1), Days | Weeks | Months | Quarters | Years, Weeks | Months | Quarters | Years, SUM | AVG | STD | STDS | MIN | MAX | VAR | VARS Note: If Moving Average, other arguments are:...<*Table*>, *Show in Legend?*(0 | 1), *Period*, None | Standard Note: If Linear Fit or Exponential Fit, other arguments are:...<*Table*>, *Show in Legend?*(0 | 1)
Border Settings	Border_Settings;S0W1 | S0W2 | S0W3 | S0W4 | S1W1 | S2W1 | S3W1 | S4W1 | S5W1, *Red, Blue, Green*	
Fill Settings	Fill_Settings	see Common Chart Object properties
Series Options	Series_Options;*DataSeries*, *LabelSeries*, *Legend*	

Related topics

Axis Title properties

Syntax

[*NBName*]*ChartName*:*ObjName.Property*

Description

Lets you specify the axis title properties.

The following table lists properties and syntax for the Axis Title object:

Property	**Syntax**	**Description**
Axis Title	[*NBName*]*Sheet*:*ObjName.Property*	*NBName* is the name of the active notebook *ChartName* is the name of the chart containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Text Font	Text_Font	see Font
Text Settings	Text_Settings	see Common Chart Object properties
Title	Title;*Title*	

Example

{SelectObject G$Y1Axis}

{SetProperty Text_Font;"Arial;18;Yes;No;No;No"}

Related topics

Bar Series properties

Syntax

[*NBName*]*ChartName*:*ObjName.Property*

Description

Lets you specify the bar series properties.

The following table lists properties and syntax for the Bar Series object:

Property	Syntax	Description
Bar Series	[*NBName*]*Sheet*:*ObjName.Property*	*NBName* is the name of the active notebook *ChartName* is the name of the chart containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Analyze	None | Aggregration | Moving Average | Linear Fit | Exponential Fit, ...	Note: If Aggregration, other arguments are: ...<*Table*>, *Show in Legend?*(0 | 1), Days | Weeks | Months | Quarters | Years, Weeks | Months | Quarters | Years, SUM | AVG | STD | STDS | MIN | MAX | VAR | VARS Note: If Moving Average, other arguments are:...<*Table*>, *Show in Legend?*(0 | 1), *Period*, None|Standard Note: If Linear Fit or Exponential Fit, other arguments are:...<*Table*>, *Show in Legend?* (0 | 1)
Bar Options	*Width%*, *Margin%*, No | Partial | Full	
Border Settings	S0W1 | S0W2 | S0W3 | S0W4 | S1W1 | S2W1 | S3W1 | S4W1 | S5W1, *Red, Blue, Green*	
Fill Settings	see Common Chart Object properties	
Riser Style	*Style*(1 | 2 | 3 | 4), *NumberOfFaces*(0 | 4 | 6 | 8) Note: Riser Style is available only for 3-D bar charts	
Series Options	*DataSeries*, *LabelSeries*, *Legend*, Bar | Line | Area | Default, Primary | Secondary	

Example

```
{SelectObject "G$Series[3,6]"}
{SetProperty Riser_Style;"3;4"}
```

Related topics

Bullet Series properties

Syntax

[*NBName*]*ChartName*:*ObjName.Property*

Description

Lets you specify the bullet series properties.

The following table lists properties and syntax for the Bullet Series object:

Property	Syntax	Description
Bullet Series	[*NBName*]*Sheet*:*ObjName.Property*	*NBName* is the name of the active notebook *ChartName* is the name of the chart containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Bullet Color	see Color	
Bullet Indentation	*BeforeBullet* (0-500), *AfterBullet* (0-500), *HangingIndent?*(0|1)	
Bullet Style	*BulletStyle*, *BulletSize*(0-32)	
Line Spacing	*SpaceBefore*(0-500), *SpaceAfter*(0-500)	
Text Font	see Font	
Text Settings	see Common Chart Object properties	

Example

```
{SelectObject G$BulletSeries[1]}
{SetProperty Bullet_Style;"8;22"}
```

Related topics

Column Chart properties

Syntax

[*NBName*]*ChartName*:*ObjName.Property*

Description

Lets you specify the column chart properties.

The following table lists properties and syntax for the Column Chart object:

Property	Syntax	Description
Column Chart	[*NBName*]*Sheet*:*ObjName.Property*	*NBName* is the name of the active notebook *ChartName* is the name of the chart containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Border Settings	S0W1 | S0W2 | S0W3 | S0W4 | S1W1 | S2W1 | S3W1 | S4W1 | S5W1, *Red, Blue, Green*	
Fill Settings	see Common Chart Object properties	
Label Options	*LabelSeries*, Currency | Value | Percent | None, *ShowTick*	
Text Font	see Font	
Text Settings	see Common Chart Object properties	

Example

```
{SelectObject "G$Series[1,4]"}
{SetProperty Border_Settings;"S0W2;0;128;255"}
```

Related topics

Float Series properties

Syntax

[*NBName*]*ChartName*:*ObjName.Property*

Description

Lets you specify the float series properties.

The following table lists properties and syntax for the Float Series object:

Property	Syntax	Description
Float Series	[*NBName*]*Sheet*:*ObjName.Property*	*NBName* is the name of the active notebook *ChartName* is the name of the chart containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Analyze	None | Aggregration | Moving Average | Linear Fit | Exponential Fit,...	Note: If Aggregration, other arguments are:...<*Table*>, *Show in Legend?*(0 | 1), Days | Weeks | Months | Quarters | Years, Weeks | Months | Quarters | Years, SUM | AVG | STD | STDS | MIN | MAX | VAR | VARS Note: If Moving Average, other arguments are:...<*Table*>, *Show in Legend?*(0 | 1), *Period*, None|Standard Note: If Linear Fit or Exponential Fit, other arguments are:...<*Table*>, *Show in Legend?*(0 | 1)
Border Settings	S0W1 | S0W2 | S0W3 | S0W4 | S1W1 | S2W1 | S3W1 | S4W1 | S5W1, *Red, Blue, Green*	
Fill Settings	see Common Chart Object properties	
Float Style	*Style*(0 | 1 | 2 | 3 | 4), *NumberOfFaces*(0 | 4 | 6 | 8), *Size*(8-32)	
Series Options	*DataBlock*, *LabelBlock*, *Legend*, Bar | Line | Area | Default, Primary | Secondary	

Example

```
{SelectObject "G$[1,3]"}
{SetProperty Analyze;"Aggregation;;No;Days;Weeks;Sum"}
```

Related topics

Chart Pane properties

Syntax

[*NBName*]*ChartName*:*ObjName.Property*

Description

Lets you specify the chart pane properties.

The following table lists properties and syntax for the Chart Pane object:

Property	Syntax	Description
Chart Pane	[*NBName*]*Sheet*:*ObjName.Property*	*NBName* is the name of the active notebook *ChartName* is the name of the chart containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Border Options	*Left*, *Top*, *Right*, *Bottom*, *Grids on Top*	
Border Style	S0W1 | S0W2 | S0W3 | S0W4 | S1W1 | S2W1 | S3W1 | S4W1 | S5W1, *Red, Blue, Green*	
Dimension	see Dimension	
Fill Settings	see Common Chart Object properties	

Example

```
{SelectObject G$Pane}
{SetProperty Border_Position;"Yes;Yes;Yes;Yes;Yes"}
```

Related topics

Chart Background properties

Syntax

[*NBName*]*ChartName*:*ObjName.Property*

Description

Lets you specify the chart background properties.

The following table lists properties and syntax for the Chart Background object:

Property	Syntax	Description
Chart Background	[*NBName*]*Sheet*:*ObjName.Property*	*NBName* is the name of the active notebook *ChartName* is the name of the chart containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Box Settings	No box | Single outline | Rounded corners | Double outline | Thick outline 1 | Thick rounded corners | Three dimensional | Thick outline 2 | Bevel out | Shadowed | Thick outline 3 | Bevel in, *Red, Blue, Green*	
Chart Button	*GotoSlide?*, *SlideName*, *Effect*, *Duration*, Slow | Med | Fast, *Overlay?*, *RunMacro?*, *MacroText*	
Fill Settings	see Common Chart Object properties	
Name (H)	*Name*	

Example

{SelectObject}

{SetProperty Chart_Button;"No;;Cut;0;Fast;No;Yes;No;Yes;{A:E1}"}

Note

- If a property name is followed by (H), it is a hidden property and does not appear in the object's Property dialog.

Related topics

Chart Subtitle properties

Syntax

[*NBName*]*ChartName*:*ObjName.Property*

Description

Lets you specify the chart subtitle properties.

The following table lists properties and syntax for the Chart Subtitle object:

Property	**Syntax**	**Description**
Chart Subtitle	[*NBName*]*Sheet*:*ObjName.Property*	*NBName* is the name of the active notebook *ChartName* is the name of the chart containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Subtitle Font	see Font	
Subtitle Text Settings	see Common Chart Object properties	

Example

{SelectObject G$Title}

{SetProperty Subtitle_Font;"Arial;24;Yes;No;No;No"}

Related topics

Chart Title Box properties

Syntax

[*NBName*]*ChartName*:*ObjName.Property*

Description

Lets you specify the chart title box properties.

The following table lists properties and syntax for the Chart Title Box object:

Property	Syntax	Description
Chart Title Box	[*NBName*]*Sheet*:*ObjName.Property*	*NBName* is the name of the active notebook *ChartName* is the name of the chart containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Alignment	Left | Center | Right	
Border Color	see Color	
Box Settings	No box | Single outline | Rounded corners | Double outline | Thick outline 1 | Thick rounded corners | Three dimensional | Thick outline 2 | Bevel out | Shadowed | Thick outline 3 | Bevel in, *Red, Blue, Green*	
Dimension	see Dimension	
Fill Settings	see Common Chart Object properties	
Name (H)	*Name*	
Text Font	see Font	
Text Settings	see Common Chart Object properties	

Example

{SelectObject G$Title}

{SetProperty Alignment;Left"}

Note

- If a property name is followed by (H), it is a hidden property and does not appear in the object's Property dialog.

Related topics

Chart Window properties

Syntax

[*NBName*]*ChartName*:*ObjName.Property*

Description

Lets you specify the chart window properties.

The following table lists properties and syntax for the Chart Window object:

Property	**Syntax**	**Description**
Chart Window	[*NBName*]*Sheet*:*ObjName.Property*	*NBName* is the name of the active notebook *ChartName* is the name of the chart containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Aspect Ratio	Floating Chart | Screen Slide | 35mm Slide | Printer Preview | Full Extent	
Grid	*GridSize*, *DisplayGrid*, *SnapToGrid*	

Example

{SetProperty Aspect_Ratio;"35mm Slide"}

Related topics

Chart Legend properties

Syntax

[*NBName*]*ChartName*:*ObjName.Property*

Description

Lets you specify the chart legend properties.

The following table lists properties and syntax for the Chart Legend object:

Property	Syntax	Description
Legend	[*NBName*]*Sheet*:*ObjName.Property*	*NBName* is the name of the active notebook *ChartName* is the name of the chart containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Box Settings	No box | Single outline | Rounded corners | Double outline | Thick outline 1 | Thick rounded corners | Three dimensional | Thick outline 2 | Bevel out | Shadowed | Thick outline 3 | Bevel in, *Red, Blue, Green*	
Dimension (H)	see Dimension	
Fill Settings	see Common Chart Object properties	
Legend Position	None | Bottom | Right	
Text Font	see Font	
Text Settings	see Common Chart Object properties	

Example

```
{SelectObject G$Legend}
{SetProperty Legend_Position;Bottom"}
```

Note

- If a property name is followed by (H), it is a hidden property and does not appear in the object's Property dialog.

Related topics

Map Legend properties

Syntax

[*NBName*]*ChartName*:*ObjName.Property*

Description

Lets you specify the map legend properties.

The following table lists properties and syntax for the Map Legend object:

Property	Syntax	Description
Map Legend	[*NBName*]*Sheet*:*ObjName.Property*	*NBName* is the name of the active notebook *ChartName* is the name of the chart containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Border Settings	Border_Settings;S0W1 | S0W2 | S0W3 | S0W4 | S1W1 | S2W1 | S3W1 | S4W1 | S5W1, *Red, Blue, Green*	
Colors	Colors;*Ranges* (0-6), *ShowLegend?* (Yes | No), *ExactMatch?* (Yes | No), *SetValuesManually?* (Yes | No), *MaxValue1, Red, Blue, Green, MaxValue2, Red, Blue, Green, MaxValue3, Red, Blue, Green, MaxValue4, Red, Blue, Green, MaxValue5, Red, Blue, Green, MaxValue6, Red, Blue, Green*	
Dimension (H)	Dimension	see Dimension
Fill Settings	Fill_Settings	see Common Chart Object properties
Font	Font;*Font* (see Font), *Red, Blue, Green*	
Legend Position	Legend_Position;None | Bottom | Right	
Patterns	Patterns;*Ranges* (0-6), *ShowLegend?* (Yes | No), *ExactMatch?* (Yes | No), *SetValuesManually?* (Yes | No), *Value1, Pattern* (0-6), *Value2, Pattern* (0-6), *Value3, Pattern* (0-6), *Value4, Pattern* (0-6), *Value5, Pattern* (0-6), *Value6, Pattern* (0-6)	
Title	Title;*Title, DisplayTitle* (Yes | No), *Font* (see Font), *Red, Blue, Green*	

Example

{SetProperty Title;"No;Arial;18;No;No;No;No;0;0;0"}

Note

- If a property name is followed by (H), it is a hidden property and does not appear in the object's Property dialog.

Related topics

Line Series properties

Syntax

[*NBName*]*ChartName*:*ObjName.Property*

Description

Lets you specify the line series properties.

The following table lists properties and syntax for the Line Series object:

Property	Syntax	Description
Line Series	[*NBName*]*Sheet*:*ObjName.Property*	*NBName* is the name of the active notebook *ChartName* is the name of the chart containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Analyze	Analyze;None | Aggregration | Moving Average | Linear Fit | Exponential Fit,...	Note: If Aggregration, other arguments are:...<*Table*>, *Show in Legend?*(0 | 1), Days | Weeks | Months | Quarters | Years, Weeks | Months | Quarters | Years, SUM | AVG | STD | STDS | MIN | MAX | VAR | VARS Note: If Moving Average, other arguments are:...<*Table*>, *Show in Legend?*(0 | 1), *Period*, None|Standard Note: If Linear Fit or Exponential Fit, other arguments are:...<*Table*>, *Show in Legend?*(0 | 1)
Fill Settings	Fill_Settings;	
Line Settings	Line_Settings;S0W1 | S0W2 | S0W3 | S0W4 | S1W1 | S2W1 | S3W1 | S4W1 | S5W1, *Red, Blue, Green*	
Marker Style	Marker_Style;M00 | M01 | M02 | M03 | M04 | M05 | M06 | M07 | M08 | M09 | M10 | M11 | M12 | M13 | M14 | M15, *MarkerWeight*, *AutoSize?*(0 | 1)	
Series Options	Series_Options;*DataBlock*, *LabelBlock*, *Legend*, Bar | Line | Area | Default, Primary | Secondary	

Example

```
{SelectObject "G$Series[1,1]"}
{SetProperty Marker_Style;M02;5;1"}
```

Related topics

Map properties

Syntax

[*NBName*]*ChartName*:*ObjName.Property*

Description

Lets you specify the map properties.

The following table lists properties and syntax for the Map object:

Property	Syntax	Description
Map	[*NBName*]*Sheet*:*ObjName.Property*	*NBName* is the name of the active notebook *ChartName* is the name of the chart containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Font	Font;*Font* (see Font), *Red, Blue, Green*	
Pin Symbol	Pin_Symbol;*DisplayPinLabels?* (Yes | No), *DisplayPinSymbol?* (Yes | No), *Font* (see Font), *Red, Blue, Green, Symbol* (0-239)	
Redraw Options	Redraw_Options;Automatic | Manual	

Example

{SetProperty Redraw_Options;Manual}

Related topics

Pie Chart properties

Syntax

[*NBName*]*ChartName*:*ObjName.Property*

Description

Lets you specify the pie chart properties.

The following table lists properties and syntax for the Pie Chart object:

Property	Syntax	Description
Pie Chart	[*NBName*]*Sheet*:*ObjName.Property*	*NBName* is the name of the active notebook *ChartName* is the name of the chart containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Border Settings	S0W1 | S0W2 | S0W3 | S0W4 | S1W1 | S2W1 | S3W1 | S4W1 | S5W1, *Red, Blue, Green*	
Explode Slice	*Distance*, *Explode*	
Fill Settings	see Common Chart Object properties	
Label Options	*Series*, Currency | Value | Percent | None, *ShowTick*	
Text Font	see Font	
Text Settings	see Common Chart Object properties	

Example

```
{SelectObject "G$Series[1,1]"}
{SetProperty Explode_Slice;"25"}
```

Related topics

Series Label properties

Syntax

[*NBName*]*ChartName*:*ObjName.Property*

Description

Lets you specify the series label properties.

The following table lists properties and syntax for the Series Label object.

Property	Syntax	Description
Series Label	[*NBName*]*Sheet*:*ObjName.Property*	*NBName* is the name of the active notebook *ChartName* is the name of the chart containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Format	*Format*, *Precision* | *Type*	
Label Alignment	Above | Top | Middle | Below (for bar charts);Above | Center | Below | Left | Right (for area and line charts)	
Text Font	see Font	
Text Style	see Common Chart Object properties	

Example

{SelectObject G$SeriesLabel[2]}

{SetProperty Label_Alignment;Middle}

Related topics

X-Axis properties

Syntax

[*NBName*]*ChartName*:*ObjName.Property*

Description

Lets you specify the X-Axis properties.

The following table lists properties and syntax for the X-Axis object:

Property	**Syntax**	**Description**
X-Axis	[*NBName*]*Sheet*:*ObjName.Property*	*NBName* is the name of the active notebook *ChartName* is the name of the chart containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Major Grid Style	S0W1 | S0W2 | S0W3 | S0W4 | S1W1 | S2W1 | S3W1 | S4W1 | S5W1, Red, Green, Blue	
Minor Grid Style	S0W1 | S0W2 | S0W3 | S0W4 | S1W1 | S2W1 | S3W1 | S4W1 | S5W1, Red, Green, Blue	Note: This property only appears when inspecting an XY chart
Numeric Format	*Format*, *Precision* | *Type*	Note: This property only appears when inspecting an XY chart
Scale	Normal | Log, *Automatic*, *High*, *Low*, *Increment*, *#Minors*, *ShowUnits*	Note: This property only appears when inspecting an XY chart
Text Font	see Font	
Text Settings	see Common Chart Object properties	
Tick Options	None | Below | Above | Across, *DisplayLabels*, *NumRows*, *NoOverlap*, *#OfLabelsToSkip*, *Limit*	
X-Axis Series	*SeriesBlock*	

Example

```
{SelectObject G$X1Axis
{SetProperty X-Axis Series;"A:A8..A11"}
```

Related topics

Y-Axis properties

Syntax

[*NBName*]*ChartName*:*ObjName.Property*

Description

Lets you specify the Y-Axis properties.

The following table lists properties and syntax for the Y-Axis object:

Property	Syntax	Description
Y-Axis	[*NBName*]*Sheet*:*ObjName.Property*	*NBName* is the name of the active notebook *ChartName* is the name of the chart containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Major Grid Style	S0W1 | S0W2 | S0W3 | S0W4 | S1W1 | S2W1 | S3W1 | S4W1 | S5W1, *Red*, *Green*, *Blue*	
Minor Grid Style	S0W1 | S0W2 | S0W3 | S0W4 | S1W1 | S2W1 | S3W1 | S4W1 | S5W1, *Red*, *Green*, *Blue*	
Numeric Format	*Format*, *Precision* | *Type*	
Scale	Normal | Log, *Automatic*, *High*, *Low*, *Increment*, *#Minors*, *ShowUnits*	Note: An additional setting, *ZeroLine*, appears if the chart type is Variance.
Text Font	see Font	
Text Settings	see Common Chart Object properties	
Tick Options	None | Left | Right | Across, *DisplayLabels*, *LengthLimit*, *Limit*	

Example

```
{SelectObject G$Y1Axis}
{SetProperty Tick_Options;"Right;Yes;No;3"}
```

Related topics

Notebook Object properties

To view properties, arguments, and syntax for the following notebook objects, choose from the following list:

Arrow
Bitmap
Button
Chart
Ellipse
Line
Notebook
OLE
Picture
Rectangle
Rounded Rectangle
Selection
Text Box

Related topics

Arrow Properties

Syntax

[*NBName*]*Sheet*:*ObjName.Property*

Description

Lets you specify the arrow properties.

The following table lists properties and syntax for the Arrow object:

Property	**Syntax**	**Description**
Arrow	[*NBName*]*Sheet*:*ObjName.Property*	*NBName* is the name of the active notebook *Sheet* is the name of the sheet containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Border Settings	Border_Settings;S0W1 | S0W2 | S0W3 | S0W4 | S1W1 | S2W1 | S3W1 | S4W1 | S5W1, *Red, Blue, Green*	
Fill Settings	Fill_Settings	see Common Chart Object properties
Object Name	Object_Name;*ObjName*	
Protection	Protection;Yes | No	

Related topics

Bitmap properties

Syntax

[*NBName*]*Sheet*:*ObjName.Property*

Description

Lets you specify the bitmap properties.

The following table lists properties and syntax for the Bitmap object:

Property	Syntax	Description
Bitmap	[*NBName*]*Sheet*:*ObjName.Property*	*NBName* is the name of the active notebook *Sheet* is the name of the sheet containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Border Color	Border_Color	see Color
Box Type	Box_Type;None | Thin | Medium | Thick, *DropShadow?*(Yes | No), *Transparent?*(Yes | No)	
Box Type	Box_Type.Frame_Line_Style;None | Thin | Medium | Thick	
Drop Shadow	Box_Type.Drop_Shadow;Yes | No	
Transparent	Box_Type.Transparent;Yes | No	
Object Name	Object_Name;*ObjName*	

Related topics

Cell properties

Syntax

[*NBName*]*Sheet*:*ObjName.Property*

Description

Lets you specify the cell properties.

The following table lists properties and syntax for the Cell object:

Property	Syntax	Description
Cell	[*NBName*]*BlockAddress.Property*	*NBName* is the name of the active notebook *Sheet* is the name of the sheet containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Alignment	Alignment;General | Left | Right | Center | Center Across Block, Top | Center | Bottom, *WrapText?*(Yes | No), Horizontal | Vertical	
Horizontal	Alignment.Horizontal;General | Left | Right | Center | Center Across Block	
Orientation	Alignment.Orientation;Horizontal | Vertical	
Vertical	Alignment.Vertical;Top | Center | Bottom	
Wrap Text	Alignment.WrapText;Yes | No	
Column Width	Column_Width;*Operation*, *WidthInTwips*, *ColSpacing*	
Auto Width	Column_Width;Auto Width,, *ExtraSpace*	
Reset Width	Column_Width;Reset Width	
Set Width	Column_Width;Set Width, *NewWidthInTwips*	
Constraints	Constraints;Protect | Unprotect, General | Labels Only | Dates Only	
Data Entry Input	Constraints.Data_Entry_Input;General | Labels Only | Dates Only	
Protection	Constraints.Protection Protect| Unprotect	
Font	Font	see Font
Line Drawing	Line_Drawing;*Left*, *Top*, *Right*, *Bottom*, *Vert*, *Horiz*, *LeftColor*, *TopColor*, *RightColor*, *BottomColor*, *VertColor*, *HorizColor*	Note: The first six settings can take NoChange | Clear | Thin | Thick | Double;the last six settings can take 0-15;@PROPERTY and {GETPROPERTY} always return default Line Drawing property settings.
Number Value (H)	Number_Value;the value in the cell	
Numeric Format	Numeric_Format;*Format*, *Precision* | *Type*	
Reveal/Hide	Reveal/Hide;Row | Column, Reveal | Hide	
Row Height	Row_Height;*Operation*, *Size*	
Reset Height	Row_Height;Reset Height	

Set Height	Row_Height;Set Height, *NewSize*
Selection (H, R)	Selection;the coordinates of the selected cells
Shading	Shading;*Color1*, *Color2*, *Blend*
Color Blend 1	Shading.Color_1;0-15
Color Blend 2	Shading.Color_2;0-15
Select Color Blend	Shading.Blend;Blend1 | Blend2 | Blend3 | Blend4 | Blend5 | Blend6 | Blend7
String Value (H)	String_Value;the label in the cell
Style (H)	Style;the named style of the active cells
Text Color	Text_Color;0-15
Value (H)	Value;the contents of the cell (as they appear on the input line)

Example

{SETOBJECTPROPERTY "A:A23.Numeric_Format","Currency,2"}

Notes

- If a property name is followed by (H), it is a hidden property and does not appear in the object's Property dialog.
- If a property name is followed by (R), you cannot set it with a macro command or link command; it is a read only property.

Related topics

Button properties

Syntax

[*NBName*]*Sheet*:*ObjName.Property*

Description

Lets you specify the button properties.

The following table lists properties and syntax for the Button object:

Property	Syntax	Description
Button	[*NBName*]*Sheet*:*ObjName.Property*	*NBName* is the name of the active notebook *Sheet* is the name of the sheet containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Border Color	Border_Color	see Color
Box Type	Box_Type;None | Thin | Medium | Thick, *DropShadow?*(Yes | No)	
Box Type	Box_Type.Frame_Line_ Style;None | Thin | Medium | Thick	
Drop Shadow	Box_Type.Drop_Shadow;Yes | No	
Label Text	Label_Text;*LabelText*	
Macro	Macro;*Macro*	
Object Name	Object_Name;*ObjName*	

Example

{SetObjectProperty "A:Button1.border_color","0;128;255"}

Related topics

Ellipse properties

Syntax

[*NBName*]*Sheet*:*ObjName.Property*

Description

Lets you specify the ellipse properties.

The following table lists properties and syntax for the Ellipse object:

Property	Syntax	Description
Ellipse	[*NBName*]*Sheet*:*ObjName.Property*	*NBName* is the name of the active notebook *Sheet* is the name of the sheet containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Border Settings	Border_Settings;S0W1 | S0W2 | S0W3 | S0W4 | S1W1 | S2W1 | S3W1 | S4W1 | S5W1, *Red, Blue, Green*	
Fill Settings	Fill_Settings	see Common Chart Object properties
Object Name	Object_Name;*ObjName*	
Protection	Protection;Yes | No	

Related topics

Chart/Map properties

Syntax

[*NBName*]*Sheet*:*ObjName.Property*

Description

Lets you specify the chart and map properties.

The following table lists properties and syntax for the Chart object and the Map object:

Property	**Syntax**	**Description**
Chart	[*NBName*]*Sheet*:*ObjName.Property*	*NBName* is the name of the active notebook *Sheet* is the name of the sheet containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Border Color	Border_Color	see Color
Box Type	Box_Type;None | Thin | Medium | Thick, *DropShadow?*(Yes | No), *Transparent?*(Yes | No)	
Box Type	Box_Type.Frame_Line_ Style;None | Thin | Medium | Thick	
Drop Shadow	Box_Type.Drop_Shadow;Yes | No	
Transparent	Box_Type.Transparent;Yes | No	
Object Name	Object_Name;*ObjName*	
Protection	Protection;Yes | No	
Source Chart/Map	Source_Chart/ Source_Map;*Name*	

Related topics

Line properties

Syntax

[*NBName*]*Sheet*:*ObjName.Property*

Description

Lets you specify the line properties.

The following table lists properties and syntax for the Line object:

Property	**Syntax**	**Description**
Line	[*NBName*]*Sheet*:*ObjName.Property*	*NBName* is the name of the active notebook *Sheet* is the name of the sheet containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Line Settings	Line_Settings;S0W1 | S0W2 | S0W3 | S0W4 | S1W1 | S2W1 | S3W1 | S4W1 | S5W1, *Red, Blue, Green*	
Object Name	Object_Name;*ObjName*	
Protection	Protection;Yes | No	

Related topics

Notebook properties

Syntax

[*NBName*].*Property*

Description

Lets you specify the notebook properties.

The following table lists properties and syntax for the Notebook object:

Property	Syntax	Description
Notebook	*[NBName].Property*	*NBName* is the name of the active notebook *Property* is one of the strings listed in the Argument column of the dialog control property tables
Display	Display;*VertScroll?*(Yes | No), *HorzScroll?*(Yes | No), *Tabs?*(Yes | No), *Objects* (Show All | Show Outline | Hide)	
Horizontal Scroll Bar	Display.Show_HorizontalScroller;Yes | No	
Objects	Display.Objects;Show All | Show Outline | Hide	
Sheet Tabs	Display.Show_Tabs;Yes | No	
Vertical Scroll Bar	Display.Show_VerticalScroller;Yes | No	
Group Mode (H)	Group_Mode;On enables Group Mode;Off disables group mode	
Macro Library	Macro_Library;Yes | No	
Palette	Palette;*Color1*, *Color2*, ..., *Color16*	
Color 1	Palette.Color_1	see Color
Color 2	Palette.Color_2	
Color 3	Palette.Color_3	
Color 4	Palette.Color_4	
Color 5	Palette.Color_5	
Color 6	Palette.Color_6	
Color 7	Palette.Color_7	
Color 8	Palette.Color_8	
Color 9	Palette.Color_9	
Color 10	Palette.Color_10	
Color 11	Palette.Color_11	
Color 12	Palette.Color_12	
Color 13	Palette.Color_13	
Color 14	Palette.Color_14	
Color 15	Palette.Color_15	
Color 16	Palette.Color_16	
Password Level	Password_Level;None | Low | Medium | High	
Recalc Settings	Recalc_Settings;Automatic | Manual | Background, Natural | Column-wise | Row-wise, Iterations, <CompileFormulas?(0 | 1)>, <AuditErrors?(0 | 1)>	
Statistics (R)	Statistics;*Filename, Directory,*	

	Created(Date Time), *Last_Saved(Date Time),* *Last_Saved_By, Revision_Number*
Summary	Summary;*Title, Subject, Author,* *Keywords, Comments*
System	System;Yes | No
Zoom Factor	Zoom_Factor;10-400

Example

{Notebook.Zoom_Factor"200"}

Notes

- If a property name is followed by (H), it is a hidden property and does not appear in the object's Property dialog.
- If a property name is followed by (R), you cannot set it with a macro command or link command; it is a read only property.

Related topics

OLE properties

Syntax

[*NBName*]*Sheet*:*ObjName.Property*

Description

Lets you specify the OLE properties.

The following table lists properties and syntax for the OLE object:

Property	Syntax	Description
OLE		*NBName* is the name of the active notebook *Sheet* is the name of the sheet containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Border Color	Border_Color	see Color
Box Type	Box_Type;None | Thin | Medium | Thick, *DropShadow?*(Yes | No), *Transparent?*(Yes | No)	
Box Type	Box_Type.Frame_Line_Style;None | Thin | Medium | Thick	
Drop Shadow	Box_Type.Drop_Shadow;Yes | No	
Transparent	Box_Type.Transparent;Yes | No	
Object Name	Object_Name;*ObjName*	
OLE	OLE;*AutoResize?*(Yes | No), *AutoLinkUpdate?*(Yes | No)	
Protection	Protection;Yes | No	

Example

{Setobjectproperty "A:Embedded1.Box_Type";"Thick;Yes;Yes"}

Related topics

Picture properties

Syntax

[*NBName*]*Sheet*:*ObjName.Property*

Description

Lets you specify the picture properties.

The following table lists properties and syntax for the Picture object:

Property	Syntax	Description
Picture		*NBName* is the name of the active notebook *Sheet* is the name of the sheet containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Border Color	Border_Color	see Color
Box Type	Box_Type;None | Thin | Medium | Thick, *DropShadow?*(Yes | No), *Transparent?*(Yes | No)	
Box Type	Box_Type.Frame_Line_Style;None | Thin | Medium | Thick	
Drop Shadow	Box_Type.Drop_Shadow;Yes | No	
Transparent	Box_Type.Transparent;Yes | No	
Object Name	Object_Name;*ObjName*	
Protection	Protection;Yes | No	

Related topics

Rectangle properties

Syntax

[*NBName*]*Sheet*:*ObjName.Property*

Description

Lets you specify the rectangle properties.

The following table lists properties and syntax for the Rectangle object:

Property	**Syntax**	**Description**
Rectangle	[*NBName*]*Sheet*:*ObjName.Property*	*NBName* is the name of the active notebook *Sheet* is the name of the sheet containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Border Settings	Border_Settings;S0W1 | S0W2 | S0W3 | S0W4 | S1W1 | S2W1 | S3W1 | S4W1 | S5W1, *Red, Blue, Green*	
Fill Settings	Fill_Settings	see Common Chart Object properties
Object Name	Object_Name;*ObjName*	
Protection	Protection;Yes | No	

Related topics

Rounded Rectangle properties

Syntax

[*NBName*]*Sheet*:*ObjName.Property*

Description

Lets you specify the rounded rectangle properties.

The following table lists properties and syntax for the Rounded Rectangle object:

Property	Syntax	Description
Rounded Rectangle	[*NBName*]*Sheet*:*ObjName.Property*	*NBName* is the name of the active notebook *Sheet* is the name of the sheet containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Border Settings	Border_Settings;S0W1 | S0W2 | S0W3 | S0W4 | S1W1 | S2W1 | S3W1 | S4W1 | S5W1, *Red, Blue, Green*	
Fill Settings	Fill_Settings	see Common Chart Object properties
Object Name	Object_Name;*ObjName*	
Protection	Protection;Yes | No	

Related topics

Text Box properties

Syntax

[*NBName*]*Sheet*:*ObjName.Property*

Description

Lets you specify the text box properties.

The following table lists properties and syntax for the Text Box object:

Property	Syntax	Description
Text Box	[*NBName*]*Sheet*:*ObjName.Property*	*NBName* is the name of the active notebook *Sheet* is the name of the sheet containing the object *ObjName* is either the object ID number or the name of the object *Property* is one of the strings listed in the Argument column of the dialog control property tables
Alignment	Alignment;Left | Right | Center, *WordWrap*, *TabStops*	
Box Settings	Box_Settings;No box | Single outline | Rounded corners | Double outline | Thick outline 1 | Thick rounded corners | Three dimensional | Thick outline 2 | Bevel out | Shadowed | Thick outline 3 | Bevel in, *Red, Blue, Green*	
Fill Settings	Fill_Settings	see Common Chart Object properties
Object Name	Object_Name;*ObjName*	
Protection	Protection;Yes | No	
Text Font	Text_Font	see Font
Text Settings	Text_Settings	see Common Chart Object properties

Related topics

Objects Sheet Icon Properties

Lets you specify the objects sheet icon properties. When the Objects sheet is active you can change the properties of icons on the Objects sheet instead of the objects they represent.

The following table lists properties and syntax for the Objects Sheet Icon object:

Property	Syntax	Description
Chart Icon		
Name	Name;*Name*	
Dialog Icon		
Dimension	Dimension	see Dimension
Disabled	Disabled;Yes | No	
Grid Options	Grid_Options;*Gridsize*, *GridShown*, *GridEnabled*	
Name	Name;*Name*	
Position Adjust	Position_Adjust;*Depend*, *LeftRel*, *TopRel*, *RightRel*, *BottomRel*, *CenterHor*, *CenterVer*	
Title	Title;*Title*	
Value (H)	Value;the current settings of all dialog controls that have Process Value set to Yes	
Slide Show Icon		
Default Effect	Default_Effect;*Effect*, *DisplayTime*, Slow | Med | Fast, *Overlay?*(Yes | No), *UseMasterSlide?*(Yes | No), *SkipSlide?*(Yes | No)	
Master Slide	Master_Slide;*SlideName*	
Name	Name;*Name*	
Show Pointer	Show_Pointer;Yes | No	

Note

- If a property name is followed by (H), it is a hidden property and does not appear in the object's Property dialog.

Related topics

Using dates and times in Quattro Pro

Entering dates

To enter a date or time

To enter the current date

To enter a date using the spreadsheet Date function

Default date and time formats

To format dates and times

To change the default date format

Related topics

Quattro Pro/Excel Equivalent Functions

Most spreadsheet functions exist in both Microsoft Excel and Quattro Pro using the same name. However, some Excel functions have a corresponding Quattro Pro function with a different name. The following table outlines these equivalent functions.

Excel function	QP Equivalent
ACCRINT	ACCRINTXL
AVERAGE	AVG
COLUMNS	COLS
COMBIN	COMB
COUNTA	XCOUNT
FREQUENCY	FREQDIST
INT	INTXL
ISTEXT	ISSTRING
LEN	LENGTH
LOG	LOGBASE
LOG10	LOG
MDETERM	MDET
REPT	REPEAT
ROUNDDOWN	ROUNDDOWNXL
ROUNDUP	ROUNDUPXL
SMALL	SMALLEST
STDEV	STDS
STDEVP	STD
VAR	VARS
VARP	VAR