**@Function Help for 1-2-3 Release 5**

**Note** You are now in @Function Help.

**To get Help for an individual @function, choose Search and specify the name of the @function, without the @.**

For example, to get Help for @SUM, type **sum**.

To return to main Help for 1-2-3 Release 5, click the Contents button, or choose Back or History to return to another Help topic.

@Functions

**@@**

@@(*location*) lets you indirectly obtain the contents of the cell specified in *location*.

**Arguments**

*location* is the address or name of a cell that contains a cell address or name, or a formula that returns the address or name of a cell.

*location* points to another cell, whose contents @@ displays in the cell that contains @@. If *location* is not a valid cell address or range name, or is a multiple-cell range, @@ evaluates to ERR.

**Notes**

@@ is useful in building conditional formulas because its indirect reference can automatically alter its own value. For example, the formula

@IF(D2="Y";"D8";@IF(D2="N";"D9";@ERR))

in cell A10, and the formula @@(A10) in cell E2, return the contents of cell D8 or D9, or ERR, in E2, depending on whether D2 contains Y or N, or something else.

When *location* refers to a cell that contains a formula, press F9 (CALC) to update the @@ formula after automatic recalculation. If you do not press F9 (CALC), the @@ formula evaluates to 0.

**Examples**

@@

**Similar @functions**

@HLOOKUP and @VLOOKUP return the contents of a specified cell in a horizontal or vertical lookup table. @CHOOSE returns a value or label from *list*, and @INDEX returns the contents of a cell located at the intersection of a specified column, row, and worksheet.

---

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**Example: @@**

In this simple sales commission chart, cell A10 contains the formula

@IF(C3="W";"C7";@IF(C3="G";"C8";@ERR))

that results in one of two cell addresses (C7 or C8), depending on which product code (W or G) you enter in C3. @@(A10) entered in C4 returns the contents of the cell whose address is returned by the formula in A10.

If you enter anything in C3 other than one of the two product codes, both the @IF and @@ functions will evaluate to ERR.

```
A ----------- A ------------- B ---------- C -----
1   SALES COMMISSION CHART
2
3   Enter Product Code:            W
4   Commission rate:         5%
5
6   Product                        Code          Rate
7   Widgets                        W             5%
8   Grommets                 G              3%
9
10 C7
```

**@ABS**

@ABS(*x*) calculates the absolute (positive) value of *x*.

**Arguments**

*x* is any value.

**Notes**

Use -@ABS to force the result of the @function to be negative.

**Examples**

@ABS(A5) = 25 if cell A5 contains the value 25, -25, or a formula that results in 25 or -25.

-@ABS(A5) = -25 if cell A5 contains the value 25, -25, or a formula that results in 25 or -25.

@ABS(START-END) = 5.6, when START and END contain any combination of positive or negative numbers that differ by 5.6 or -5.6.

---

**See also**

**Help**

    @Functions

***User's Guide***

    Chapter 11, "Calculating with @Functions"

## @ACCRUED

@ACCRUED(*settlement*;*issue*;*first-interest*;*coupon*;[*par*];[*frequency*];[*basis*]) calculates the accrued interest for securities with periodic interest payments. @ACCRUED supports short, standard, and log coupon periods.

### Arguments

*settlement* is the security's settlement date. *settlement* is a <u>date number</u>. If *settlement* is less than *issue*, @ACCRUED returns <u>ERR</u>.

*issue* is the security's issue or dated date. *issue* is a date number.

*first-interest* is the security's first interest date. *first-interest* is a date number. If *first-interest* is less than or equal to *issue*, @ACCRUED returns ERR.

*coupon* is the security's annual coupon rate. *coupon* is any positive value or 0.

*par* is an optional argument that specifies the security's par value, that is, the principal to be paid at maturity. *par* is a positive value. If you do not include the *par* argument, 1-2-3 uses 100.

*frequency* is an optional argument that specifies the number of coupon payments per year. *frequency* is a value from the following table:

| *frequency* | Frequency of coupon payments |
|---|---|
| 1 | Annual |
| 2 | Semiannual; default if you omit the argument |
| 4 | Quarterly |
| 12 | Monthly |

*basis* is an optional argument that specifies the type of <u>day-count basis</u> to use. *basis* is a value from the following table:

| *basis* | Day count basis |
|---|---|
| 0 | 30/360; default if you omit the argument |
| 1 | Actual/actual |
| 2 | Actual/360 |
| 3 | Actual/365 |

You cannot use an optional argument without using the ones that precede it.

### Examples

A bond has a July 1, 1993, settlement date, a December 1, 1992 issue date, and a June 1, 1993, first interest date. The semiannual coupon rate is 5.50%. The bond has a $100 par value, and a 30/360 day-count basis.

To determine the bond's accrued interest:

@ACCRUED(@DATE(93;7;1);@DATE(92;12;1);@DATE(93;6;1);0.055;100;2;0) = 0.458333

### Similar @functions

<u>@PRICE</u> calculates the price per $100 face value for a bond. <u>@YIELD</u> calculates the yield for securities that pay periodic interest. <u>@DURATION</u> calculates the annual duration and <u>@MDURATION</u> calculates

the annual modified duration for securities that pay periodic interest.

___

**See also**

**Help**

[@Functions](#)

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@ACOS**

@ACOS(*x*) calculates the arc (inverse) cosine using the cosine *x* of an angle. The result of @ACOS is an angle, in <u>radians</u>, from 0 through $\pi$. This represents an angle between 0 and 180 degrees.

**Arguments**

*x* is the cosine of an angle and is a value from -1 through 1.

**Examples**

The cosine of an angle is 0.5. To determine the size of the angle, use @ACOS(0.5), which returns 1.0472 radians. To convert this to degrees, use @RADTODEG(1.0472). The result is 60 degrees.

**Similar @functions**

<u>@COS</u> calculates the cosine of an angle. <u>@ACOSH</u> calculates the arc hyperbolic cosine of a value.

**See also**

**Help**

<u>@Functions</u>
<u>@RADTODEG</u> to convert radians to degrees

*User's Guide*

Chapter 11, "Calculating with @Functions"

## @ACOSH

@ACOSH(*x*) calculates the arc (inverse) hyperbolic cosine using the hyperbolic cosine *x* of an angle.

### Arguments

*x* is the hyperbolic cosine of an angle and is a value greater than or equal to 1.

### Examples

@ACOSH(2) = 1.316958

### Similar @functions

@ACOS calculates the arc cosine of a value. @COSH calculates the hyperbolic cosine of an angle.

---

### See also

### Help

@Functions
@RADTODEG to convert radians to degrees

### *User's Guide*

Chapter 11, "Calculating with @Functions"

## @ACOT

@ACOT(*x*) calculates the arc (inverse) cotangent using the cotangent *x* of an angle. The result of @ACOT is an angle, in <u>radians</u>, from 0 through $\pi$. This represents an angle between 0 and 180 degrees.

### Arguments

*x* is the cotangent of an angle and can be any value.

### Examples

The cotangent of angle *x* is 1.732051. To determine the size of angle *x*, use @ACOT(1.732051), which returns 0.523599 radians. To convert this to degrees, use @RADTODEG(0.523599). The result is 30 degrees.

### Similar @functions

<u>@COT</u> calculates the cotangent of an angle. <u>@ATAN</u> calculates the arc tangent of a value.

---

### See also

### Help

<u>@Functions</u>
<u>@RADTODEG</u> to convert radians to degrees

### *User's Guide*

Chapter 11, "Calculating with @Functions"

**@ACOTH**

@ACOTH($x$) calculates the arc (inverse) hyperbolic cotangent using the hyperbolic cotangent $x$ of an angle.

**Arguments**

$x$ is the hyperbolic cotangent of an angle and is any value less than -1 or greater than 1.

**Examples**

@ACOTH(2) = 0.549306

**Similar @functions**

@ACOT calculates the arc cotangent of a value. @COTH calculates the hyperbolic cotangent of an angle.

---

**See also**

**Help**

@Functions
@RADTODEG to convert radians to degrees

***User's Guide***

Chapter 11, "Calculating with @Functions"

## @ACSC

@ACSC(*x*) calculates the arc (inverse) cosecant using the cosecant *x* of an angle. The result of @ACSC is an angle, in <u>radians</u>, from -$\pi$/2 through $\pi$/2. This represents an angle between -90 and 90 degrees.

### Arguments

*x* is the cosecant of an angle and is a value greater than or equal to 1, or less than or equal to -1.

### Examples

The cosecant of angle *x* is 1.743447. To determine the size of angle *x*, use @ACSC(1.743447), which returns 0.610865 radians. To convert this to degrees, use @RADTODEG(0.610865). The result is 35 degrees.

### Similar @functions

<u>@ACSCH</u> calculates the arc hyperbolic cosecant of a value. <u>@CSC</u> calculates the cosecant of an angle.

### See also

### Help

<u>@Functions</u>
<u>@RADTODEG</u> to convert radians to degrees

### *User's Guide*

Chapter 11, "Calculating with @Functions"

## @ACSCH

@ACSCH(*x*) calculates the arc (inverse) hyperbolic cosecant using the hyperbolic cosecant *x* of an angle.

### Arguments

*x* is the hyperbolic cosecant of an angle and is a value other than 0.

### Examples

@ACSCH(1.54) = 0.61068

### Similar @functions

@ACSC calculates the arc cosecant of a value. @CSCH calculates the hyperbolic cosecant of an angle.

### See also

### Help

@Functions
@RADTODEG to convert radians to degrees

### *User's Guide*

Chapter 11, "Calculating with @Functions"

## @ASEC

@ASEC(*x*) calculates the arc (inverse) secant using the secant *x* of an angle. The result of @ASEC is an angle, in <u>radians</u>, from 0 through $\pi$. This represents an angle between 0 and 180 degrees.

### Arguments

*x* is the secant of an angle and is a value less than or equal to -1 or greater than or equal to 1.

### Examples

In a right triangle, the secant of angle *x* is 2. To determine the size of angle *x*, use @ASEC(2), which returns 1.047198 radians. To convert this to degrees, use @RADTODEG(1.047198). The result is 60 degrees.

### Similar @functions

<u>@SEC</u> calculates the secant of an angle. <u>@ASECH</u> calculates the arc hyperbolic secant of a value.

---

### See also

### Help

<u>@Functions</u>
<u>@RADTODEG</u> to convert radians to degrees

### *User's Guide*

Chapter 11, "Calculating with @Functions"

## @ASECH

@ASECH(*x*) calculates the arc (inverse) hyperbolic secant using the hyperbolic secant *x* of an angle.

## Arguments

*x* is the hyperbolic secant of an angle and is a value greater than 0 and less than or equal to 1.

## Examples

@ASECH(0.5) = 1.316958

## Similar @functions

@ASEC calculates the arc secant of an angle. @SECH calculates the hyperbolic secant of an angle.

_____

## See also

## Help

@Functions
@RADTODEG to convert radians to degrees

## *User's Guide*

Chapter 11, "Calculating with @Functions"

## @ASIN

@ASIN(*x*) calculates the arc (inverse) sine using the sine *x* of an angle. The result of @ASIN is an angle, in <u>radians</u>, from -$\pi$ /2 through $\pi$ /2. This represents an angle between -90 and 90 degrees.

### Arguments

*x* is the sine of an angle and is a value from -1 through 1.

### Examples

The sine of angle *x* is 0.66. To determine the size of angle *x*, use @ASIN(0.66), which returns 0.72082 radians. To convert this to degrees, use @RADTODEG(0.72082). The result is 41.3 degrees.

### Similar @functions

<u>@SIN</u> calculates the sine of an angle. <u>@ASINH</u> calculates the arc hyperbolic sine of a value.

---

### See also

### Help

<u>@Functions</u>
<u>@RADTODEG</u> to convert radians to degrees

### *User's Guide*

Chapter 11, "Calculating with @Functions"

**@ASINH**

@ASINH(*x*) calculates the arc (inverse) hyperbolic sine using the hyperbolic sine *x* of an angle.

**Arguments**

*x* is the hyperbolic sine of an angle and is any value.

**Examples**

@ASINH(2) = 1.443635

**Similar @functions**

@SINH calculates the hyperbolic sine of an angle. @ASIN calculates the arc sine of a value.

---

**See also**

**Help**

@Functions
@RADTODEG to convert radians to degrees.

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@ATAN**

@ATAN(*x*) calculates the arc (inverse) tangent using the tangent *x* of an angle. The result of @ATAN is an angle, in <u>radians</u>, from -π /2 through π /2. This represents an angle between -90 and 90 degrees.

**Arguments**

*x* is the tangent of an angle and is any value.

**Examples**

The tangent of angle *x* is 2/1, or 2. To determine the size of angle *x*, use @ATAN(2), which returns 1.10715 radians. To convert this to degrees, use @RADTODEG(1.10715). The result is 63.4 degrees.

**Similar @functions**

<u>@ATANH</u> calculates the arc hyperbolic tangent of a value. <u>@ATAN2</u> calculates the size of an angle whose tangent is *y/x*. <u>@TAN</u> calculates the tangent of an angle.

---

**See also**

**Help**

    <u>@Functions</u>
    <u>@RADTODEG</u> to convert radians to degrees

***User's Guide***

    Chapter 11, "Calculating with @Functions"

## @ATAN2

@ATAN2(*x;y*) calculates the arc tangent using the tangent *y/x* of an angle. The result of @ATAN2 is an angle, in <u>radians</u>, from -$\pi$ through $\pi$. This represents an angle between -180 and 180 degrees, depending on the sign of *x* and *y* (see the list below).

### Arguments

*x* and *y* are values. If *y* is 0, @ATAN2 returns 0; if both *x* and *y* are 0, @ATAN2 returns <u>ERR</u>.

### Notes

The list below gives the value ranges for @ATAN2.

- If *x* is positive and *y* is positive, then the result can be from 0 to $\pi$ /2 (quadrant I).

- If *x* is negative and *y* is positive, then the result can be from $\pi$ /2 to $\pi$ (quadrant II).

- If *x* is negative and *y* is negative, then the result can be from -$\pi$  to -$\pi$ /2 (quadrant III).

- If *x* is positive and *y* is negative, then the result can be from -$\pi$ /2 to 0 (quadrant IV).

When *x* and *y* are both positive (quadrant I), and when *x* is positive and *y* is negative (quadrant IV), the results are the same as for @ATAN.

### Examples

In a right triangle, the two sides that form the right angle measure 1 and 2. To determine the size of the larger of the two acute angles, use @ATAN2(1;2), which returns 1.10715 radians. To convert this to degrees, use @RADTODEG(1.10715). The result is 63.4 degrees.

### Similar @functions

<u>@ATAN</u> calculates the arc tangent using the tangent *x* of an angle. <u>@TAN</u> calculates the tangent of an angle.

---

### See also

### Help

<u>@Functions</u>
<u>@RADTODEG</u> to convert radians to degrees

### *User's Guide*

Chapter 11, "Calculating with @Functions"

**@ATANH**

@ATANH(*x*) calculates the arc (inverse) hyperbolic tangent using the hyperbolic tangent *x*.

**Arguments**

*x* is the hyperbolic tangent of an angle and is a value between -1 and 1.

**Examples**

@ATANH(0.544736) = 0.610865

**Similar @functions**

@ATAN calculates the arc tangent of a value. @ATAN2 calculates the size of an angle whose tangent is *y/x*. @TANH calculates the hyperbolic tangent of an angle.

**See also**

**Help**

@Functions
@RADTODEG to convert radians to degrees

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@AVEDEV**

@AVEDEV(*list*) calculates the average of the absolute deviations of the values in *list*.

**Arguments**

*list* can contain any of the following, in any combination: numbers, numeric formulas, and addresses or names of ranges that contain numbers or numeric formulas. Separate elements of *list* with argument separators.

**Examples**

@AVEDEV

**Similar @functions**

@STD and @PURESTD calculate the standard deviation of the values in a list. @DEVSQ calculates the sum of squared deviations of the values in a list.

---

**See also**

**Help**

@Functions
Statistical @Function Arguments

***User's Guide***

Chapter 11, "Calculating with @Functions"

**Example: @AVEDEV**

This table lists house sales for the month of April. The ages (in years) of the houses are listed in a range named AGE_LIST (C2..C8). You want to determine the average deviation of the ages of the houses in the list:

@AVEDEV(AGE_LIST) = 15.18

```
A ---- A ------------- B --- C ---- D ----
1    ADDRESS                LOT    AGE    COST
2    12 Bartholomew Sq 0.25  48    $290,000
3    40 Prospect St    0.40  22    $105,000
4    103 Cranberry La  0.50  21    $135,000
5    27 Kilburn St           1.00  70    $128,000
6    468 Henshaw St    0.50  52    $174,000
7    9 Pleasant St           0.25  42    $195,000
8    80 Beach St             0.25  23    $118,000
```

**@AVG**
**@PUREAVG**

@AVG(*list*) calculates the average of a *list* of values.

@PUREAVG(*list*) calculates the average of a *list* of values, ignoring all cells that contain labels.

### Arguments

*list* can contain any of the following, in any combination: numbers, numeric formulas, and addresses or names of ranges that contain numbers or numeric formulas. Separate elements of *list* with argument separators.

### Examples

@AVG(A2..A5) = 252.75, when A2..A5 contains the values 160, 227, 397, and 227.

@AVG(A1..A5) = 202.20, when A1..A5 contains the values 160, 227, 397, and 227, and the label January. (@AVG counts the label as zero and uses it in the calculation.)

@PUREAVG(A1..A5) = 252.75, because @PUREAVG ignores the label January.

### Similar @functions

@DAVG finds the average of values in a field of a database table that meet criteria you specify. @GEOMEAN calculates the geometric mean and @HARMEAN calculates the harmonic mean of the values in a list. @MEDIAN finds the median value in a list of values.

---

### See also

**Help**

@Functions
Statistical @Function Arguments

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@BESSELI**
**@BESSELJ**
**@BESSELK**
**@BESSELY**

@BESSELI(*x*;*n*) calculates the modified Bessel function of integer order In(*x*).

@BESSELJ(*x*;*n*) calculates the Bessel function of integer order *Jn*(*x*).

@BESSELK(*x*;*n*) calculates the modified Bessel function of integer order *Kn*(*x*).

@BESSELY(*x*;*n*) calculates the Bessel function of integer order *Yn*(*x*), also known as the Neumann function.

## Arguments

*x* is the value at which to evaluate the function and is any value.

*n* is the order of the function and is any positive integer. For @BESSELI and @BESSELJ, *n* can also be 0.

## Notes

@BESSELI, @BESSELJ, @BESSELK, and @BESSELY approximate their respective functions to within $\pm 5*10^{-8}$.

Bessel functions are often used in problems with cylindrical symmetry, in connection with wave propagation, fluid motion, elasticity, and diffusion.

## Examples

@BESSELI(2;2) = 0.688948

@BESSELJ(1;0) = 0.765198

@BESSELK(3;0) = 0.03474

@BESSELY(1;1) = -0.781213

---

## See also

**Help**

@Functions

*User's Guide*

Chapter 11, "Calculating with @Functions"

**@BETA**

@BETA(*z*;*w*) calculates the beta function.

**Arguments**

*z* and *w* can be any values.

**Notes**

The result of @BETA is accurate to within at least six significant digits.

**Examples**

@BETA(0.5;0.5) = 3.141593

**Similar @functions**

@BETAI calculates the incomplete beta function. @GAMMA calculates the gamma function.

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@BETAI**

@BETAI(*a*;*b*;*x*) returns the incomplete beta function.

**Arguments**

*a* and *b* can be any values.

*x* is a value from 0 through 1.

**Notes**

The result of @BETAI is accurate to within at least six significant digits.

**Examples**

@BETAI(5;0.5;0.668271) = 0.050012

**Similar @functions**

@BETA calculates the beta function.

_____

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@BINOMIAL**

@BINOMIAL(*trials*;*successes*;*probability*;[*type*]) calculates the binomial probability mass function or the cumulative binomial distribution.

**Arguments**

*trials* is the number of independent trials. *trials* is any positive integer.

*successes* is the number of successes in *trials* and is any positive integer or 0 and must be less than or equal to *trials*.

If *trials* and *successes* are not integers, 1-2-3 truncates them to integers.

*probability* is the probability of success on each trial and is any value from 0 through 1.

*type* is an optional argument that specifies whether 1-2-3 calculates the probability mass function or the cumulative binomial distribution.

| *type* | 1-2-3 calculates |
| --- | --- |
| 0 | The probability of exactly *successes* number of successes; default if you omit the argument |
| 1 | The probability of at most *successes* number of successes |
| 2 | The probability of at least *successes* number of successes |

**Notes**

@BINOMIAL approximates the cumulative binomial distribution to within ± 3*10^-7.

**Examples**

You randomly select ten cola drinkers to participate in a blind taste test. You give each subject a glass of cola A and a glass of cola B. The glasses are identical in appearance, except for a code on the bottom to identify the cola. Assuming there is no tendency among cola drinkers to prefer one brand of cola to another, the probability that a test participant prefers cola A is 50%.

To determine the probability that exactly 7 out of 10 test participants prefer cola A:

@BINOMIAL(10;7;0.5) = 0.117188, or 11.72%.

To determine the probability that at least 7 out of 10 test participants prefer cola A:

@BINOMIAL(10;7;0.5;2) = 0.171875, or 17.19%.

**Similar @functions**

@CRITBINOMIAL calculates the largest value for which the cumulative binomial distribution is less than or equal to a specific criterion. @COMBIN calculates the number of combinations for a specified number of values. @PERMUT calculates the number of permutations for a list of values.

---

**See also**

**Help**

   @Functions

***User's Guide***

   Chapter 11, "Calculating with @Functions"

**@CELL**
**@CELLPOINTER**

@CELL(*attribute;location*) returns information about the first cell in *location*.

@CELLPOINTER(*attribute*) returns information about the current cell.

**Arguments**

*location* is the address or name of a range.

*attribute* is any of the items listed below, enclosed in " " (quotation marks), or the address or name of a cell that contains one of the items.

| attribute | Returns |
|---|---|
| across | 1 if data in the cell is aligned across columns |
| | 0 if data in the cell is not aligned across columns |
| address | The absolute address in abbreviated form (column letter and row number only) |
| backgroundcolor | The background color, as an integer from 0 through 255 that specifies a color in the color palette |
| bold | 1 if the cell is formatted as bold |
| | 0 if the cell is not formatted as bold |
| bottomborder | An integer from 0 (no border) through 8 that specifies a line style from the list of line styles in the <u>Style Lines & Color</u> dialog box |
| bottombordercolor | An integer from 0 through 15 that specifies a line color from the list of line colors in the Style Lines & Color dialog box |
| col | The column letter, as an integer from 1 through 256 (1 for column A; 2 for column B; and so on) |
| color | 1 if the cell is formatted for negative numbers in color |
| | 0 if the cell is not formatted for negative numbers in color |
| contents | The contents of the cell |
| coord | The absolute cell address in full form (worksheet letter; column letter; and row number) |
| datatype | The type of data in the cell: |
| | b if the cell is blank |
| | v if the cell contains a number or a numeric formula |
| | l if the cell contains a label or text formula |
| | e if the cell contains the value <u>ERR</u> |
| | n if the cell contains the value <u>NA</u> |
| filedate | A value that corresponds to the date and time the file that contains the cell was last saved. This includes both a <u>date number</u> (integer portion) and a <u>time number</u> (decimal portion). |
| filename | The name of the file that contains the cell, including the path |
| fontface | The typeface of the data in the cell |
| fontsize | The point size of the data in the cell |

| format | The cell format: |
|---|---|
| | C0 to C15 if Currency, 0 through 15 decimal places |
| | F0 to F15 if Fixed, 0 through 15 decimal places |
| | G if General, a label, or a blank cell |
| | P0 to P15 if Percent, 0 through 15 decimal places |
| | S0 to S15 if Scientific, 0 through 15 decimal places |
| | ,0 to ,15 if , (Comma), 0 through 15 decimal places |
| | + if +/- format |
| | D1 if Date format 31-Dec-92 |
| | D2 if Date format 31-Dec |
| | D3 if Date format Dec-92 |
| | D4 if Date format 12/31/92 |
| | D5 if Date format 12/31 |
| | D6 if Time format 11:59:59 AM |
| | D7 if Time format 11:59 AM |
| | D8 if Time format 23:59:59 |
| | D9 if Time format 59:59 |
| | T if Text |
| | H if Hidden |
| | L if Label |
| | A if Automatic |
| | -- if Color for negative numbers |
| | () if Parentheses |
| formulatype | The type of formula in the cell: |
| | b if the cell is blank |
| | v if the cell contains a number |
| | l if the cell contains a label |
| | fv if the cell contains a numeric formula |
| | fl if the cell contains a text formula |
| | fe if the cell contains a formula that evaluates to ERR |
| | fn if the cell contains a formula that evaluates to NA |
| halign | The horizontal alignment of data in the cell: |
| | 0 if General (labels left-aligned; values right-aligned) |
| | 1 if Left |
| | 2 if Center |
| | 3 if Right |
| | 4 if Evenly spaced |

| | |
|---|---|
| height | The row height, in points |
| italic | 1 if the cell is formatted as italics |
| | 0 if the cell is not formatted as italics |
| leftborder | An integer from 0 (no border) through 8 that specifies a line style from the list of line styles in the Style Lines & Color dialog box |
| leftbordercolor | An integer from 0 through 15 that specifies a line color from the list of line colors in the Style Lines & Color dialog box |
| orientation | The orientation for rotated text, as an integer from 0 through 4 that specifies an orientation from the Style Alignment dialog box |
| parentheses | 1 if the cell is formatted for parentheses |
| | 0 if the cell is not formatted for parentheses |
| pattern | An integer from 0 through 63 that specifies a pattern from the list of patterns in the Style Lines & Color dialog box |
| patterncolor | The pattern color, as an integer from 0 through 255 that specifies a color in the color palette |
| prefix | ' for a left-aligned label |
| | " for a right-aligned label |
| | ^ for a centered label |
| | \ for a repeating label |
| | \| for a nonprinting label |
| | Blank (no label prefix) if the cell is blank or contains a value |
| protect | 1 if the cell is protected |
| | 0 if the cell is not protected |
| rightborder | An integer from 0 (no border) through 8 that specifies a line style from the list of line styles in the Style Lines & Color dialog box |
| rightbordercolor | An integer from 0 through 15 that specifies a line color from the list of line colors in the Style Lines & Color dialog box |
| rotation | The rotation angle for rotated text, as an integer from 0 through 90 degrees |
| row | The row number, from 1 through 8192 |
| sheet | The worksheet letter, as an integer from 1 through 256 (1 for worksheet A; 2 for worksheet B; and so on) |
| sheetname | The worksheet name, or, if the worksheet is not named, the worksheet letter |
| textcolor | The color of the data in the cell, as an integer from 0 through 255 that specifies a color in the color palette |
| topborder | An integer from 0 (no border) through 8 that specifies a line style from the list of line styles in the Style Lines & Color dialog box |
| topbordercolor | An integer from 0 through 15 that specifies a line color from the list of line colors in the Style Lines & Color dialog box |
| type | The type of data in the cell: |
| | b if the cell is blank |
| | v if the cell contains a numeric value, a numeric formula, or a text formula |

|         | l if the cell contains a label |
|---------|--------------------------------|
| underline | The style of underline, as an integer from 0 (no underline) through 3 that specifies an underline style from the Style Font & Attributes dialog box |
| valign | The vertical alignment of data in the cell: |
|        | 0 if Bottom |
|        | 1 if Center |
|        | 2 if Top |
| width | The column width |
| wrap | 1 if data is wrapped within the cell |
|      | 0 if data is not wrapped within the cell |

## Notes

Press F9 (CALC) to recalculate your work before you use @CELL or @CELLPOINTER to be sure the results are correct.

@CELL and @CELLPOINTER are useful in macros and in combination with @IF. Use @CELL to check input during a macro to guard against certain types of entries, and to direct macro execution using subroutines based on a user's entry. @CELL can also allow an automated application to change cell attributes based on a user's entries.

Use @CELLPOINTER to find the current location of the cell pointer or to evaluate a formula based on the contents of the current cell. You can then direct processing depending on the cell's contents or type.

You can substitute the attribute name type2 for datatype and the attribute name type3 for formulatype.

## Examples

The following example uses @CELL with @IF and @ERR to return an error (ERR) if the user does not type a value in the cell named AMT, and to return the contents of AMT (a value) if the user types a value.

@IF(@CELL("type";AMT)="v",AMT,@ERR)

The following example uses @CELLPOINTER in a macro that tests for a blank cell in a list of items. In the following example, if 1-2-3 encounters a blank cell, it beeps and branches to a subroutine.

{IF @CELLPOINTER("type")="b"}{BEEP}{BRANCH Step2}

## Similar @functions

@INFO returns information about the current 1-2-3 session.

---

## See also

## Help

@Functions

## *User's Guide*

Chapter 11, "Calculating with @Functions"

## @CHAR

@CHAR(*x*) returns the character of the Lotus Multibyte Character Set (LMBCS) that corresponds to the number *x*.

### Arguments

*x* is an integer. Values that do not correspond to LMBCS character codes return ERR. If *x* is not an integer, 1-2-3 truncates it to an integer.

### Notes

If your monitor cannot display the character that corresponds to *x*, 1-2-3 displays a character that resembles the desired character, when possible. If no character approximates the character, 1-2-3 displays a solid rectangle, which represents an undisplayable character. Make sure your printer can print the characters you enter.

### Examples

@CHAR(156) = £ (British pound sign).

@CHAR(D9) = A, if cell D9 contains the value 65.

### Similar @functions

@CODE returns the LMBCS code that corresponds to a character.

---

### See also

### Help

@Functions

### *User's Guide*

Chapter 11, "Calculating with @Functions"

## @CHIDIST

@CHIDIST(*x*;*degrees-freedom*;[*type*]) calculates the chi-square distribution.

### Arguments

*x* is the value at which to evaluate the chi-square distribution. The value you enter for *x* depends on the value you enter for *type*.

| If *type* is | *x* is |
|---|---|
| 0 | The critical value or upper bound for the value of the chi-square cumulative distribution random variable and is a value greater than or equal to 0; default if you omit the argument |
| 1 | A probability (significance level) and is a value from 0 through 1 |

*degrees-freedom* is the number of degrees of freedom for the sample. *degrees-freedom* is a positive integer. If *degrees-freedom* is not an integer, 1-2-3 truncates it to an integer.

*type* is an optional argument that specifies how 1-2-3 calculates @CHIDIST.

| *type* | 1-2-3 calculates |
|---|---|
| 0 | The significance level corresponding to *x*; default if you omit the argument |
| 1 | The critical value that corresponds to the significance level *x* |

### Notes

@CHIDIST approximates the chi-square cumulative distribution to within ± $3*10^{-7}$. If @CHIDIST cannot approximate the result to within 0.0000001 after 100 calculation iterations, the result is ERR.

The chi-square distribution is a continuous, single-parameter distribution derived as a special case of the gamma distribution.

Use @CHIDIST to test the validity of a hypothesis by comparing the values you observe with those you expect.

### Examples

@CHIDIST(12.592;6) = 0.05

@CHIDIST(0.05;6;1) = 12.59159

### Similar @functions

@CHITEST calculates the probability associated with a chi-square test. @FDIST calculates the *F*-distribution. @TDIST calculates the Student's *t*-distribution.

### See also

### Help

@Functions

### *User's Guide*

Chapter 11, "Calculating with @Functions"

**@CHOOSE**

@CHOOSE(*x;list*) returns a value or label, represented by *x*, from *list*.

**Arguments**

*x* is a value. *x* represents the offset number of an item's position in *list*.

*list* is a group of values or the addresses or names of cells that contain values and labels, separated by argument separators. 1-2-3 numbers each entry in *list*, and then chooses the entry that corresponds to the value of *x*.

**Examples**

A worksheet contains a list of labels in the range A1..A4 and a list of their offset numbers (0; 1; 2; 3) in the range B1..B4.

@CHOOSE(B3;A1;A2;A3;A4) returns the label in A3, which is the item whose offset number is 2 (2 is the value in B3) in *list*.

**Similar @functions**

@HLOOKUP and @VLOOKUP find entries in horizontal or vertical lookup tables. @INDEX and @XINDEX return the contents of a cell located at the intersection of a specified column, row, and worksheet. @MATCH returns the position of the cell in a range whose contents match data you specify. @MAXLOOKUP returns an absolute reference to the cell that contains the largest value in a list of ranges. @MINLOOKUP returns an absolute reference to the cell that contains the smallest value in a list of ranges.

---

**See also**

**Help**

 @Functions

***User's Guide***

 Chapter 11, "Calculating with @Functions"

## @CLEAN

@CLEAN(*text*) removes nonprinting characters from *text*.

## Arguments

*text* is text enclosed in " " (quotation marks), the address or name of a cell that contains a label, or a formula or @function that results in text.

## Examples

You imported data into 1-2-3 from a word processing program. Cell A45 contains the label

®Second, we must act soon.¬

@CLEAN(A45) = Second, we must act soon.

## Similar @functions

@CHAR returns the LMBCS character that corresponds to a code number. @TRIM removes leading, trailing, and consecutive spaces from text.

_____

## See also

**Help**

@Functions

## *User's Guide*

Chapter 11, "Calculating with @Functions"

**@CODE**

@CODE(*text*) returns the Lotus Multibyte Character Set (LMBCS) code that corresponds to the first character in *text*.

**Arguments**

*text* is text enclosed in " " (quotation marks), the address or name of a cell that contains a label, or a formula or @function that results in text.

**Examples**

@CODE("A") = 65

@CODE(C5) = 77, if C5 contains the label Ms. Jones, because 77 is the LMBCS code for M.

**Similar @functions**

@CHAR returns the LMBCS character that corresponds to a code number.

**See also**

**Help**

   @Functions

*User's Guide*

   Chapter 11, "Calculating with @Functions"

**@COLS**

@COLS(*range*) counts the number of columns in *range*.

**Arguments**

*range* is a range address or range name.

**Notes**

Use @COLS with {FOR} in a macro that repeats the same action on a series of columns to determine when the macro should stop.

**Examples**

@COLS(D9..J25) = 7, because *range* contains columns D through J (seven columns).

@COLS(SCORES) = 2, if SCORES is the name of the range B3..C45.

**Similar @functions**

@REFCONVERT converts the 1-2-3 column or worksheet letters A through IV to numbers from 1 through 256. @ROWS counts the rows, and @SHEETS counts the worksheets, in a range.

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@COMBIN**

@COMBIN(*n;r*) calculates the binomial coefficient for *n* and *r*. The binomial coefficient is the number of ways that *r* can be selected from *n*, without regard for order.

**Arguments**

*n* is the number of values and is any positive integer or 0.

*r* is the number of values in each combination and is any positive integer or 0. *r* must be less than or equal to *n*.

If *n* and *r* are not integers, 1-2-3 truncates them to integers.

**Notes**

@COMBIN approximates the binomial coefficient to within ± 3*10^-7.

**Examples**

A jar contains five marbles, each one a distinct color. You take out three marbles at random. The number of combinations of colors you could have is

@COMBIN(5;3) = 10

**Similar @functions**

@BINOMIAL calculates the binomial probability mass function or the cumulative binomial distribution. @CRITBINOMIAL calculates the smallest value for which the cumulative binomial distribution is less than or equal to a specific criterion. @PERMUT calculates the number of permutations for a list of values.

---

**See also**

**Help**

   @Functions

***User's Guide***

   Chapter 11, "Calculating with @Functions"

**@COORD**

@COORD(*worksheet;column;row;absolute*) creates a cell reference from values that correspond to *worksheet*, *column*, and *row*.

**Arguments**

*worksheet* and *column* are any integers from 1 through 256. *worksheet* and *column* correspond to the worksheet and column letters (1 for worksheet or column A; 2 for worksheet or column B; and so on).

*row* is any integer from 1 through 8192. *row* corresponds to the row number.

*absolute* is any integer from 1 through 8.

If *worksheet*, *column*, *row*, and *absolute* are not integers, 1-2-3 truncates them to integers.

**Notes**

@COORD creates a cell reference that is <u>relative</u>, <u>absolute</u>, or <u>mixed</u>, according to the value of *absolute*. The following table shows the possible values of *absolute* and their effect on the cell address A1 in worksheet A.

| *absolute* | Value of **@COORD(1;1;1;***absolute***)** |
|---|---|
| 1 | $A:$A$1 |
| 2 | $A:A$1 |
| 3 | $A:$A1 |
| 4 | $A:A1 |
| 5 | A:$A$1 |
| 6 | A:A$1 |
| 7 | A:$A1 |
| 8 | A:A1 |

Use @COORD with <u>@INDEX</u>, <u>@VLOOKUP</u>, or <u>@HLOOKUP</u> to create cell addresses from tables of values in the current file. Use @COORD with <u>@@</u> to return the value in the cell address created by @COORD.

**Examples**

@COORD(3;7;25;8) returns the relative cell address C:G25.

@@(@COORD(C1;D1;E4;8)) returns the value in cell A:A4 (C1 contains 1; D1 contains 1; and E4 contains 4).

**Similar @functions**

<u>@REFCONVERT</u> converts the 1-2-3 column or worksheet letters A through IV to numbers from 1 through 256.

**See also**

**Help**

    <u>@Functions</u>

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@CORREL**

@CORREL(*range1*;*range2*) calculates the correlation coefficient of values in *range1* and *range2*.

**Arguments**

*range1* and *range2* are range names or addresses. *range1* and *range2* must be the same size. If *range1* and *range 2* are not the same size, @CORREL returns ERR.

1-2-3 pairs cells in the two ranges by their order in the range. Ranges are ordered from top to bottom, left to right, first sheet through last.

**Notes**

Correlation and covariance both measure the relationship between two sets of data. However, the correlation statistic is independent of the unit of measure, while the covariance statistic is dependent on the unit of measure.

**Examples**

@CORREL

**Similar @functions**

@COV calculates the covariance of the values in two ranges.

_____

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**Example: @CORREL**

You want to determine if there is a relationship between height and weight among ten randomly selected test subjects. You record the subjects' heights and weights in a worksheet.

@CORREL(A2..A11;B2..B11) = 0.384947

```
A --------- A --------- B --
1      HEIGHT (cm)    WEIGHT (kg)
2         190.50        72.73
3         187.96        86.36
4         175.26        68.18
5         175.26        76.37
6         180.34        77.27
7         180.34        72.73
8         187.96        75.00
9         172.72        68.18
10        177.80        70.46
11        179.07        86.36
```

## @COS

@COS(*x*) calculates the cosine of angle *x*. The cosine is the ratio of the side adjacent an acute angle of a right triangle to the hypotenuse. The result of @COS is a value from -1 through 1.

### Arguments

*x* is an angle measured in <u>radians</u>. *x* is any value from -2^32*π through 2^32*π.

### Examples

@COS(@DEGTORAD(30)) = 0.866, the cosine of a 30-degree angle.

### Similar @functions

<u>@ACOS</u> calculates the arc cosine of a value. <u>@COSH</u> calculates the hyperbolic cosine of a value.

### See also

### Help

> <u>@DEGTORAD</u> to convert degrees to radians
> <u>@Functions</u>

### *User's Guide*

> Chapter 11, "Calculating with @Functions"

**@COSH**

@COSH(*x*) calculates the hyperbolic cosine of *x*. The result of @COSH is a value greater than or equal to 1.

**Arguments**

*x* is any value from approximately 11355.1371 through 11355.1371.

**Examples**

@COSH(@DEGTORAD(30)) = 1.140238, the hyperbolic cosine of a 30-degree angle.

**Similar @functions**

@ACOS calculates the arc (inverse) cosine of a value. @COS calculates the cosine of an angle.

---

**See also**

**Help**

   @DEGTORAD to convert degrees to radians
   @Functions

***User's Guide***

   Chapter 11, "Calculating with @Functions"

**@COT**

@COT(*x*) calculates the cotangent of angle *x*. The cotangent is the ratio of the side adjacent an acute angle of a right triangle to the opposite side.

**Arguments**

*x* is an angle measured in <u>radians</u>. *x* is any value from -2^32*$\pi$ through 2^32*$\pi$.

**Examples**

@COT(@DEGTORAD(30)) = 1.73205, the cotangent of a 30-degree angle.

**Similar @functions**

<u>@ACOT</u> calculates the arc cotangent of a value. <u>@COTH</u> calculates the hyperbolic cotangent of an angle. <u>@TAN</u> calculates the tangent of an angle.

---

**See also**

**Help**

<u>@DEGTORAD</u> to convert degrees to radians
<u>@Functions</u>

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@COTH**

@COTH(*x*) calculates the hyperbolic cotangent of *x*.

**Arguments**

*x* is any value from approximately 11355.1371 through 11355.1371 except 0.

**Examples**

@COTH(@DEGTORAD(30)) = 2.081283, the hyperbolic cotangent of a 30-degree angle.

**Similar @functions**

@ACOTH calculates the arc hyperbolic cotangent of a value. @COT calculates the cotangent of an angle. @TANH calculates the hyperbolic tangent of an angle.

---

**See also**

**Help**

@DEGTORAD to convert degrees to radians
@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@COUNT**
**@PURECOUNT**

@COUNT(*list*) counts the nonblank cells in a *list* of ranges.

@PURECOUNT(*list*) counts the cells in a *list* of ranges, excluding cells that contain labels.

**Arguments**

*list* is any combination of range addresses or names, separated by argument separators.

**Notes**

@COUNT counts every cell in *list* that contains an entry of any kind, including a label, a label-prefix character, or the values ERR and NA.

@COUNT and @PURECOUNT are useful to stop (or divert) a macro that performs a task on a series of ranges when the cell pointer reaches a range that has no entries.

**Examples**

@COUNT(A2..A3;A5) = 1, if A2..A3 is blank and whether or not A5 is blank, because A5 is a single-cell address.

{IF @PURECOUNT(SEPTEMBER)=0}{BRANCH YTD} branches to a macro called YTD if the range named SEPTEMBER is blank or contains a label, label prefix, or text formula.

**Similar @functions**

@DCOUNT and @DPURECOUNT count the nonblank cells in a field of a database table that meet criteria you specify.

---

**See also**

**Help**

    @Functions
    Statistical @Function Arguments

***User's Guide***

    Chapter 11, "Calculating with @Functions"

**@COV**

@COV(*range1*;*range2*;[*type*]) calculates either the population or sample covariance of the values in *range1* and *range2*.

**Arguments**

*range1* and *range2* are the names or addresses of ranges. *range1* and *range2* must be the same size. If *range1* or *range 2* are not the same size, @COV returns ERR.

1-2-3 pairs cells in the two ranges by their same order in the range. Ranges are ordered from top to bottom, left to right, first sheet through last.

*type* is an optional argument that specifies whether to calculate the population or sample covariance.

| *type* | 1-2-3 calculates |
| --- | --- |
| 0 | Population covariance; default if you omit the argument |
| 1 | Sample covariance |

**Notes**

Covariance is the average of the products of deviations of corresponding values in lists.

Correlation and covariance both measure the relationship between two sets of data.The correlation statistic, however, is independent of the unit of measure, and the covariance statistic is dependent on the unit of measure.

**Examples**

@COV

**Similar @functions**

@VAR and @PUREVAR calculate the population variance, and @VARS and @PUREVARS calculate the sample variance of values in a list. @CORREL calculates the correlation coefficient of corresponding values in two ranges.

**See also**

**Help**

@Functions

*User's Guide*

Chapter 11, "Calculating with @Functions"

**Example: @COV**

You want to determine if there is a relationship between height and weight among ten randomly selected test subjects. You record the subjects' heights and weights in a worksheet.

@COV(A2..A11;B2..B11) = 13.8872

```
A --------- A --------- B --
1      HEIGHT (cm)    WEIGHT (kg)
2         190.50       72.73
3         187.96       86.36
4         175.26       68.18
5         175.26       76.37
6         180.34       77.27
7         180.34       72.73
8         187.96       75.00
9         172.72       68.18
10        177.80       70.46
11        179.07       86.36
```

## @CSC

@CSC(*x*) calculates the cosecant of angle *x*. 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.

### Arguments

*x* is an angle measured in <u>radians</u>. *x* is any value from $-2^{32}*\pi$ through $2^{32}*\pi$ except 0 and multiples of $\pi$.

### Examples

@CSC(@DEGTORAD(30)) = 2, the cosecant of a 30-degree angle.

### Similar @Functions

<u>@ACSC</u> calculates the arc cosecant of a value. <u>@CSCH</u> calculates the hyperbolic cosecant of an angle. <u>@SIN</u> calculates the sine of an angle.

---

### See also

### Help

<u>@DEGTORAD</u> to convert from degrees to radians
<u>@Functions</u>

### *User's Guide*

Chapter 11, "Calculating with @Functions"

## @CSCH

@CSCH(*x*) calculates the hyperbolic cosecant of angle *x*. The hyperbolic cosecant is the reciprocal of the hyperbolic sine.

### Arguments

*x* is any value from approximately 11355.1371 through 11355.1371 except 0.

### Examples

@CSCH(@DEGTORAD(30)) = 1.825306, the hyperbolic cosecant of a 30-degree angle.

### Similar @Functions

@ACSCH calculates the arc hyperbolic cosecant of a value. @CSC calculates the cosecant of an angle. @SINH calculates the hyperbolic sine of an angle.

---

### See also

### Help

@DEGTORAD to convert from degrees to radians
@Functions

### *User's Guide*

Chapter 11, "Calculating with @Functions"

**@CTERM**

@CTERM(*interest;future-value;present-value*) calculates the number of compounding periods required for an investment (*present-value*) to grow to a *future-value*, earning a fixed *interest* rate per compounding period.

**Arguments**

*interest* is any value greater than -1 except 0.

*future-value* and *present-value* are any values. Both *future-value* and *present-value* must be either positive or negative.

**Examples**

You just deposited $10,000 in an account that pays an annual interest rate of 10% (.10), compounded monthly. You want to determine how many years it will take to double your investment.

@CTERM(.10/12;20000;10000)/12 = 6.960312

In other words, it will take about seven years to double the original investment of $10,000.

Because @CTERM calculates the total number of compounding periods, you may need to include the number of periods for which the *interest* rate is compounded in order to express the term and interest rate in the same unit of time. In the example above, the annual interest rate of 10%, compounded monthly, is entered as .10/12 (*interest* divided by the number of compounding periods).

**Similar @functions**

@TERM and @NPER determine the number of periods required for an investment of equal periodic payments to reach a specified value.

_____

**See also**

**Help**

  @Functions

***User's Guide***

  Chapter 11, "Calculating with @Functions"

**@DATE**

@DATE(*year;month;day*) calculates the <u>date number</u> for the specified *year*, *month*, and *day*.

**Arguments**

*year* is an integer from 0 (the year 1900) through 199 (the year 2099).

*month* is an integer from 1 through 12.

*day* is an integer from 1 through 31. The value you use for *day* must be a valid day for the *month*. For example, you cannot use 31 as the *day* if you use 4 (April) as the *month*.

**Notes**

Even though February 29,1900, did not exist (it was not a leap year), 1-2-3 assigns a date number to this day. This does not invalidate any of your date calculations unless you use dates from January 1, 1900, through March 1, 1900. If you are using dates within that period, subtract 1 from any results within the period.

If you want the results of an @DATE calculation to appear as an actual date, format the cell that contains the @DATE function with one of the <u>Style Number Format</u> date formats.

**Examples**

@DATE(92;2;21) returns 33655, or 21-Feb-92, in a cell formatted as day-month-year.

@DATE(91;2;29) returns ERR, because 1991 was not a leap year.

**Similar @functions**

<u>@DATEVALUE</u> calculates the date number for a date entered as a label. <u>@TIME</u> calculates the time number for a specified time. <u>@NOW</u> calculates the date-and-time number for the current date and time.

---

**See also**

**Help**

   <u>@Functions</u>

***User's Guide***

   Chapter 11, "Calculating with @Functions"

## @DATEDIF

@DATEDIF(*start-date*;*end-date*;*format*) calculates the number of years, months, or days between two date numbers.

### Arguments

*start-date* and *end-date* are date numbers.

*format* is a code from the following table, entered as text, that specifies the format you want the result of @DATEDIF to have.

| *format* | Returns the number of |
|----------|------------------------|
| y | Years |
| m | Months |
| d | Days |
| md | Days, ignoring months and years |
| ym | Months, ignoring years |
| yd | Days, ignoring years |

### Examples

The following examples use the dates February 15, 1990, and September 15, 1993.

@DATEDIF(@DATE(90;2;15),@DATE(93;9;15),"m") returns 43, the number of months between February 15, 1990, and September 15, 1993.

@DATEDIF(@DATE(90;2;15),@DATE(93;9;15),"md") returns 0, because the day of the month for both *start-date* and *end-date* is the 15th.

@DATEDIF(@DATE(90;2;15),@DATE(93;9;15),"ym") returns 7, the number of months between February and September.

### Similar @functions

@D360 and @DAYS360 calculate the number of days between two dates, based on a 360-day year (12 months; each with 30 days). @DAYS calculates the number of days between two dates, based on a day-count basis you specify.

### See also

### Help

@Functions

### *User's Guide*

Chapter 11, "Calculating with @Functions"

**@DATEINFO**

@DATEINFO(*date*;*attribute*) returns information about a <u>date number</u>.

**Arguments**

*date* is a date number.

*attribute* is any one of the integers listed in the following table:

| *attribute* | Returns |
|---|---|
| 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 |

**Examples**

@DATEINFO(23063;7) = 28, the number of days in February, 1963.

@DATEINFO(@DATE(92;10;5),10) = 4, because October is in the fourth quarter.

**See also**

**Help**

   <u>@Functions</u>

***User's Guide***

   Chapter 11, "Calculating with @Functions"

**@DATEVALUE**

@DATEVALUE(*text*) calculates the <u>date number</u> for the date specified in *text*.

**Arguments**

*text* is text enclosed in " " (quotation marks), a formula that results in text, or the address or name of a cell that contains a <u>label</u> or a formula that results in a label. *text* must be in one of the five 1-2-3 date formats.

**Notes**

If you want the results of an @DATEVALUE calculation to appear as an actual date, format the cell that contains the @DATEVALUE function with one of the <u>Style Number Format</u> date formats.

@DATEVALUE is useful with data imported from another program, such as a word processing program.

**Examples**

@DATEVALUE("21-Feb-91") returns the date number 33290.

@DATEVALUE(BIRTHDAY) returns the date number 20723, if the cell named BIRTHDAY contains the label 25-Sep-56.

**Similar @functions**

<u>@DATESTRING</u> converts a date number to its equivalent date and displays it as a label. <u>@DATE</u> calculates the date number for a specified date.

---

**See also**

**Help**

    <u>@Functions</u>

***User's Guide***

    Chapter 11, "Calculating with @Functions"

**@DAY**

@DAY(*date-number*) extracts the day of the month, a value from 1 through 31, from *date-number*.

**Arguments**

*date-number* is a value from 1 (January 1; 1900) through 73050 (December 31; 2099).

**Notes**

You can use one of the other date @functions to supply the value for *date-number*.

@DAY can supply the *day* argument for other date @functions that build on previously calculated dates.

**Examples**

@DAY(@NOW) = the current day of the month.

@DAY(D9) = 12, if cell D9 contains the date number 33250 (the date 12-Jan-91).

**Similar @functions**

@MONTH calculates the month, @YEAR calculates the year, and @WEEKDAY calculates the day of the week, using a date number.

**See also**

**Help**

   @Functions

***User's Guide***

   Chapter 11, "Calculating with @Functions"

## @DAYS

@DAYS(*start-date*;*end-date*;[*basis*]) calculates the number of days between two dates using a specified day-count *basis*.

### Arguments

*start-date* and *end-date* are <u>date numbers</u>. If *start-date* is earlier than *end-date*, the result of @DAYS is positive. If *start-date* is later than *end-date*, the result of @DAYS is negative. If *start-date* and *end-date* are the same, the result of @DAYS is 0.

*basis* is an optional argument that specifies the type of day-count basis to use. *basis* is a value from the following table:

| *basis* | Day-count basis |
|---------|-----------------|
| 0 | 30/360; default if you omit the argument |
| 1 | Actual/actual |
| 2 | Actual/360 |
| 3 | Actual/365 |

### Examples

@DAYS(@DATE(93;4;16),@DATE(93;9;25)) = 159, the number of days between April 16, 1993, and September 25, 1993, based on a 360-day year of twelve months, each with 30 days.

@DAYS(@DATE(93;4;16),@DATE(93;9;25),1) = 162,   the number of days between April 16, 1993, and September 25, 1993, based on the actual number of days in the months April through September.

### Similar @functions

<u>@DATEDIF</u> calculates the number of years, months, or days between two dates. <u>@D360</u> and <u>@DAYS360</u> calculate the number of days between two dates, based on a 360-day year. <u>@NETWORKDAYS</u> calculates the number of days between two dates, excluding weekends and holidays.

### See also

### Help

> <u>@Functions</u>

### *User's Guide*

> Chapter 11, "Calculating with @Functions"

**@D360**
**@DAYS360**

@D360(*start-date;end-date*) calculates the number of days between two dates, based on a 360-day year (12 months; each with 30 days).

@DAYS360(*start-date;end-date*) calculates the number of days between two dates, based on a 360-day year, according to the standards of the U.S. securities industry.

## Arguments

*start-date* and *end-date* are date numbers.

## Notes

The formula used to calculate @DAYS360 conforms to the 1990 modifications to the Securities Industry Association's 1986 edition of *Standard Security Calculation Methods*.

@DAYS360 and @D360 typically return different answers for the same data when either *start-date* or *end-date* is the last day of the month.

## Examples

@DAYS360(@DATE(89;4;16),@DATE(89;9;25)) = 159

@D360(33290;33524) = 232, the number of days between February 21, 1991, and October 13, 1991, based on a 360-day year.

## Similar @functions

@DATEDIF calculates the number of years, months, or days between two dates. @DAYS calculates the number of days between two dates, using a specified day-count basis. @NETWORKDAYS calculates the number of days between two dates, excluding weekends and holidays. @WORKDAY calculates the date that is a specified number of days before or after a specified date, excluding weekends and holidays. @NEXTMONTH calculates the date that is a specified number of months before or after a specified date.

---

**See also**

**Help**

   @Functions

***User's Guide***

   Chapter 11, "Calculating with @Functions"

**@DB**

@DB(*cost*;*salvage*;*life*;*period*) calculates the depreciation allowance of an asset with an initial value of *cost*, an expected useful *life*, and a final *salvage* value for a specified *period* of time, using the fixed-declining balance method.

**Arguments**

*cost* is the amount paid for the asset. *cost* is any positive value or 0. If *cost* is 0, the result of @DB is 0.

*salvage* is the estimated value of the asset at the end of its useful life. *salvage* is any positive value or 0. If *salvage* is greater than *cost*, the result of @DB is negative.

*life* is the number of periods the asset takes to depreciate to its *salvage* value. *life* is any value greater than or equal to 1 and less than or equal to *life*.

*period* is the time period for which you want to find the depreciation allowance. *period* is any value greater than or equal to 1.

You must express *life* and *period* in the same units, typically years.

**Notes**

The fixed-declining balance method slows the rate of depreciation in comparison to the double-declining balance method, so more depreciation expense occurs (and can be written off) in later periods. Depreciation stops when the book value of the asset -- that is, the total cost of the asset minus its total depreciation over all prior periods -- reaches the salvage value.

**Examples**

You just purchased an office machine for $10,000. The useful life of this machine is eight years, and the salvage value after eight years is $1,200. You want to calculate the depreciation expense for the fifth year:

@DB(10000;1200;8;5) = $806.51

**Similar @functions**

@VDB calculates depreciation using the variable-rate declining balance method. @DDB uses the double-declining balance method, @SLN uses the straight-line method, and @SYD uses the sum-of-the-years'-digits method.

---

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@DDB**

@DDB(*cost;salvage;life;period*) calculates the depreciation allowance of an asset with an initial value of *cost*, an expected useful *life*, and a final *salvage* value for a specified *period* of time, using the double-declining balance method.

**Arguments**

*cost* is the amount paid for the asset. *cost* is any value greater than or equal to *salvage*.

*salvage* is the estimated value of the asset at the end of its useful life. *salvage* is any value.

*life* is the number of periods the asset takes to depreciate to its *salvage* value. *life* is any value greater than 2.

*period* is the time period for which you want to find the depreciation allowance. *period* is any value greater than or equal to 1.

You must express *life* and *period* in the same units, typically years.

**Notes**

The double-declining balance method accelerates the rate of depreciation so that more depreciation expense occurs (and can be written off) in earlier periods than in later ones. Depreciation stops when the book value of the asset -- that is, the total cost of the asset minus its total depreciation over all prior periods -- reaches the salvage value.

If the salvage value of an asset is relatively low, @DDB may not fully depreciate the asset by the end of the estimated useful life. You may want to use @VDB, which always fully depreciates the asset within the estimated life.

**Examples**

You just purchased an office machine for $10,000. The useful life of this machine is eight years, and the salvage value after eight years is $1,200. You want to calculate the depreciation expense for the fifth year, using the double-declining balance method:

@DDB(10000;1200;8;5) = $791.02

**Similar @functions**

@VDB calculates depreciation using the variable-rate declining balance method. @DB uses the fixed-declining balance method, @SLN uses the straight-line method, and @SYD uses the sum-of-the-year's-digits method.

---

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@DECIMAL**

@DECIMAL(*hexadecimal*) converts a hexadecimal value to its signed decimal equivalent.

**Arguments**

*hexadecimal* is a value from 00000000 through FFFFFFFF, entered as text. *hexadecimal* can be up to eight characters long and can contain only numbers from 0 through 9 and letters from A through F. The letters can be either uppercase or lowercase.

**Notes**

Hexadecimal values from 00000000 through 7FFFFFFF correspond to 0 and positive decimal values.

Hexadecimal values from 80000000 through FFFFFFFF correspond to negative decimal values.

**Examples**

@DECIMAL("1A") = 26

@DECIMAL("FFFFFFFE") = -2

**Similar @functions**

@HEX converts decimal numbers to hexadecimal.

---

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@DEGTORAD**

@DEGTORAD(*degrees*) converts *degrees* to <u>radians</u>.

**Arguments**

*degrees* is a value.

**Examples**

@DEGTORAD(30) = 0.523599 radians

@COS(@DEGTORAD(45)) = 0.707107, the cosine of a 45-degree angle.

**Similar @functions**

<u>@RADTODEG</u> converts radians to degrees.

_____

**See also**

**Help**

   <u>@Functions</u>

***User's Guide***

   Chapter 11, "Calculating with @Functions"

**@DEVSQ**

@DEVSQ(*list*) calculates the sum of squared deviations of the values in *list* from their mean.

**Arguments**

*list* can contain any of the following, in any combination: numbers, numeric formulas, and addresses or names of ranges that contain numbers or numeric formulas. Separate elements of *list* with argument separators.

**Examples**

@DEVSQ(2;3;9;8;15;2;1) = 159.4286

**Similar @functions**

@STD and @PURESTD calculate the standard deviation of the values in a list.

---

**See also**

**Help**

@Functions
Statistical @Function Arguments

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@ERF**

@ERF(*lower-limit*;[*upper-limit*]) calculates the error function integrated between *lower-limit* and *upper-limit*.

**Arguments**

*lower-limit* is the lower bound for integrating @ERF and can be any value.

*upper-limit* is an optional argument that specifies the upper bound for integrating @ERF. If you omit the *upper-limit* argument, @ERF integrates between 0 and *lower-limit*.

**Notes**

@ERF approximates the error function to within $\pm$ 1.2 x 10^-7.

**Examples**

@ERF(0.7) = 0.677801

@ERF(0.8) = 0.742101

@ERF(0.7;0.8) = 0.0643, the difference between the previous examples.

**Similar @functions**

@ERFC calculates the complementary error function. @ERFD calculates the derivative of the error function.

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@ERFC**

@ERFC($x$) calculates the complementary error function, integrated between $x$ and $\infty$.

**Arguments**

*x* can be any value.

**Notes**

@ERFC($x$) is equal to 1-@ERF($x$).

@ERFC approximates the complementary error function to within $\pm$ 3*10^-7.

**Examples**

@ERFC(0.7) = 0.322199

**Similar @functions**

@ERF calculates the error function integrated between specified upper and lower limits. @ERFD calculates the derivative of the error function.

**See also**

**Help**

   @Functions

***User's Guide***

   Chapter 11, "Calculating with @Functions"

**@ERFD**

@ERFD(*x*) calculates the derivative of the error function.

**Arguments**

*x* is a value from approximately -106.560 to approximately 106.560.

**Notes**

If *x* is larger than approximately 106.560 or smaller than approximately -106.560, the calculation is too large for 1-2-3 to store, and @ERFD returns ERR. If *x* is smaller than approximately -15.102 or larger than approximately 15.102, 1-2-3 can calculate and store the value of @ERFD, but cannot display it. (The cell displays a series of asterisks.)

**Examples**

@ERFD(0.7) = 0.691275

**Similar @functions**

@ERF calculates the error function integrated between specified upper and lower limits.

---

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@ERR**

@ERR returns the value <u>ERR</u>.

**Notes**

@ERR is useful in flagging errors in calculations. It is seldom used by itself. For example, @ERR used as an argument with @IF produces the value ERR when certain conditions exist, such as when a formula results in an unacceptable value (such as a negative monthly payment).

**Examples**

@IF(B14>3;@ERR;B14) = ERR, if the value in cell B14 is greater than 3.

**Similar @functions**

<u>@NA</u> returns the value NA (not available). <u>@ISERR</u> tests for the value ERR.

_____

**See also**

**Help**

   <u>@Functions</u>

***User's Guide***

   Chapter 11, "Calculating with @Functions"

## @EVEN

@EVEN(*x*) rounds the value *x* away from 0 to the nearest even integer.

### Arguments

*x* is any value. If *x* is an even integer, @EVEN returns *x*.

### Examples

@EVEN(2.25) = 4

@EVEN(2) = 2

@EVEN(-2.25) = -4

### Similar @functions

@ODD rounds a value away from 0 to the nearest odd integer. @ROUND, @ROUNDDOWN, and @ROUNDUP round a value to a specified number of decimal places. @ROUNDM rounds a value to a specified multiple. @INT truncates a value, discarding the decimal portions. @TRUNC truncates a value to a specified decimal place.

---

### See also

### Help

@Functions

### *User's Guide*

Chapter 11, "Calculating with @Functions"

## @EXACT

@EXACT(*text1;text2*) compares two sets of characters. If the two sets match exactly, @EXACT returns 1 (true); if the two sets are not exactly the same, @EXACT returns 0 (false).

### Arguments

*text1* and *text2* are text enclosed in " " (quotation marks), formulas that result in text, or the addresses or names of cells that contain labels or formulas that result in labels.

### Notes

@EXACT is more precise than = (the equal operator) in a formula. Unlike = (the equal operator), @EXACT distinguishes between uppercase and lowercase letters and between letters with and without accent marks.

You can use @EXACT to set passwords for macros by comparing what a user enters with a required entry before continuing the macro.

### Examples

@EXACT("ATHENS";"Athens") = 0 (false).

@EXACT("Overdue";B2) = 1 (true), if cell B2 contains the label Overdue.

@EXACT("400";400) = ERR, because *text2* is a value.

---

### See also

### Help

   @Functions

### *User's Guide*

   Chapter 11, "Calculating with @Functions"

**@EXP**

@EXP(*x*) calculates the value of the constant *e* (approximately 2.718282) raised to the power *x*.

**Arguments**

*x* is a value from approximately 11355.1371 to approximately 11356.5234.

**Notes**

If *x* is larger than approximately 11356.5234 or smaller than approximately -11355.1371, the calculation is too large for 1-2-3 to store, and @EXP returns ERR. If *x* is smaller than approximately -227.956 or larger than approximately 230.259, 1-2-3 can calculate and store the value of @EXP, but cannot display it. (The cell displays a series of asterisks.)

**Examples**

@EXP(0.7) = 2.013753

**Similar @functions**

@EXP2 calculates the value of *e* raised to the power -*x*^2. @LN is the inverse of @EXP.

---

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@EXP2**

@EXP2(*x*) calculates the value of the constant *e* (approximately 2.718282) raised to the power (-*x*^2).

**Arguments**

*x* is a value from approximately -106.570 to approximately 106.570.

**Notes**

If *x* is larger than approximately 106.570 or smaller than approximately -106.570, the calculation is too large for 1-2-3 to store, and @EXP2 returns ERR. If *x* is smaller than approximately 15.098 or larger than approximately 15.098, 1-2-3 can calculate and store the value of @EXP2, but cannot display it. (The cell displays a series of asterisks.)

**Examples**

@EXP2(0.7) = 0.612626

**Similar @functions**

@EXP calculates *e* raised to a specified power.

---

**See also**

**Help**

@Functions

*User's Guide*

Chapter 11, "Calculating with @Functions"

**@FACT**

@FACT(*n*) calculates the factorial of *n*.

**Arguments**

*n* is any positive integer or 0.

**Notes**

The factorial of *n* is equal to the product of all positive integers from 1 to *n*.

If *n* is greater than or equal to 1755, the calculation is too large for 1-2-3 to store, and @FACT returns ERR. If *n* is greater than or equal to 70, 1-2-3 can calculate and store the value of @FACT, but cannot display it. (The cell displays a series of asterisks.)

**Examples**

@FACT(0) = 1

@FACT(5) = 120, the result of 1*2*3*4*5.

**Similar @functions**

@FACTLN calculates the natural logarithm of the factorial of *n*. @PRODUCT multiplies the values in a list.

---

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

## @FACTLN

@FACTLN(*n*) calculates the natural logarithm of the factorial of *n*.

### Arguments

*n* is any positive integer or 0.

### Notes

The factorial of *n* is equal to the product of all positive integers from 1 to *n*.

### Examples

@FACTLN(0) = 0, the result of @LN(1).

@FACTLN(5) = 4.787492, the result of @LN(1*2*3*4*5).

### Similar @functions

@FACT calculates the of the factorial of *n*. @LN calculates the natural logarithm of a value.

---

### See also

**Help**

> @Functions

***User's Guide***

> Chapter 11, "Calculating with @Functions"

**@FALSE**

@FALSE returns the logical value 0 (false).

**Notes**

If a logical statement such as A1=B1 is true, its logical value is 1. If it is false, its logical value is 0.

Using @FALSE is the same as using the value 0 in formulas that evaluate logical conditions, but @FALSE makes the formula easier to understand.

Use @FALSE with macros or @functions such as @IF and @CHOOSE that require a logical value of 0 (false). @FALSE is useful as the *y* argument for @IF, which is the value returned if the condition is not met.

**Examples**

@IF(A6>500;@TRUE;@FALSE) = 0 when cell A6 contains a value less than or equal to 500.

**Similar @functions**

@TRUE returns the logical value 1.

_____

**See also**

**Help**

@Functions

*User's Guide*

Chapter 11, "Calculating with @Functions"

**@FDIST**

@FDIST(*x*;*degrees-freedom1*;*degrees-freedom2*;[*type*]) calculates the *F*-distribution.

**Arguments**

*x* is the value at which you want to evaluate the *F*-distribution. The value you enter for *x* depends on the value you enter for *type*.

| If *type* is | *x* is |
|---|---|
| 0 | The critical value or upper bound for the value of the cumulative *F*-distribution and is a value greater than or equal to 0; default if you omit the argument |
| 1 | A probability and is a value from 0 to 1 |

*degrees-freedom1* and *degrees-freedom2* are the numbers of degrees of freedom for the first and second samples, respectively. *degrees-freedom1* and *degrees-freedom2* are positive integers.

*type* is an optional argument that specifies how 1-2-3 calculates @FDIST.

| *type* | 1-2-3 calculates |
|---|---|
| 0 | The significance level that corresponds to the critical value *x*; default if you omit the argument |
| 1 | The critical value that corresponds to the significance level *x* |

**Notes**

@FDIST approximates the cumulative *F*-distribution to within $\pm$ 3*10^-7. If @FDIST cannot approximate the result to within 0.0000001 after 100 calculation iterations, the result is ERR.

The *F*-distribution is a continuous distribution obtained from the ratio of two chi-square distributions, each divided by its number of degrees of freedom.

Use @FDIST to determine the degree to which two samples vary.

**Examples**

@FDIST(3.07;8;10) = 0.05

@FDIST(0.05;8;10) = 3.07

**Similar @functions**

@FTEST calculates the probability associated with an *F* test. @CHIDIST calculates the chi-square distribution. @TDIST calculates the Student's *t*-distribution.

---

**See also**

**Help**

@Functions

*User's Guide*

Chapter 11, "Calculating with @Functions"

**@FIND**

@FIND(*search-text;text;start-number*) calculates the position in *text* at which 1-2-3 finds the first occurrence of *search-text*, beginning at the position indicated by *start-number*.

**Arguments**

*search-text* and *text* are text enclosed in " " (quotation marks), a formula that results in text or the address or name of a cell that contains text or a formula that results in a label.

*start-number* is an offset number.

**Notes**

If 1-2-3 does not find *search-text* in *text*, @FIND returns ERR. @FIND also returns ERR if *start-number* is greater than the number of characters in *text*, or if *start-number* is negative.

@FIND is case-sensitive and accent-sensitive; for example, @FIND will not find the *search-text* "e" in the *text* "CAMBRIDGE."

@FIND is also useful when combined with @MID or @REPLACE to locate and extract or replace text.

**Examples**

@FIND("P";"Accounts Payable";0) = 9 because *search-text* P is at position 9 in *text* Accounts Payable.

**See also**

**Help**

@Functions

*User's Guide*

Chapter 11, "Calculating with @Functions"

**@FV**
**@FVAL**

@FV(*payments;interest;term*) calculates the future value of an investment, based on a series of equal *payments*, earning a periodic *interest* rate, over the number of payment periods in *term*.

@FVAL(*payments;interest;term*;[*type*];[*present-value*]) calculates the future value of an investment with a specified *present-value*, based on a series of equal *payments*, earning a periodic *interest* rate, over the number of payment periods in *term*. @FVAL calculates for either an ordinary annuity or an annuity due, depending on the value you specify for *type*.

## Arguments

*payments* and *term* are values.

For @FVAL, *term* must be a positive value.

*interest* is a value greater than -1.

*type* is an optional argument that specifies whether to calculate for an ordinary annuity or for an annuity due.

| *type* | 1-2-3 calculates for |
| --- | --- |
| 0 | Ordinary annuity (payments due at the end of a period); default if you omit the argument |
| 1 | Annuity due (payment due at the beginning of a period) |

*present-value* is an optional argument that specifies the present value of the series of future payments. *present-value* can be any value. If you omit the *present-value* argument, 1-2-3 uses 0.

You cannot use an optional argument without using the ones that precede it.

## Notes

The period used to calculate *interest* must be the same period used for *term*; for example, if you are calculating a monthly payment, enter the interest and term in monthly increments. Usually, this means you must divide the interest rate by 12 and multiply the number of years in *term* by 12.

## Examples

You plan to deposit $2,000 each year for the next 20 years into an account to save for retirement. The account pays 7.5% interest, compounded annually; interest is paid on the last day of each year. You want to calculate the value of your account in 20 years. You make each year's contribution on the last day of the year.

@FV(2000;0.075;20) = $86,609.36, the value of your account at the end of 20 years.

If you make each year's contribution on the first day of the year:

@FVAL(2000;0.075;20;1) = $93,105.06

## Similar @functions

@PV and @PVAL determine the present value of an investment. @NPV computes the net present value of an investment, discounting the future value to present value.

---

## See also

## Help

   @Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@GAMMA**

@GAMMA(*x*) calculates the gamma function.

**Arguments**

*x* is any value except 0 and negative integers.

**Notes**

@GAMMA approximates the gamma distribution accurately to within six significant figures.

**Examples**

@GAMMA(0.5) = 1.772454

@GAMMA(5) = 24

**Similar @functions**

@BETA calculates the beta function. @GAMMAI calculates the incomplete gamma function. @GAMMALN calculates the natural logarithm of the gamma function.

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@GAMMAI**

@GAMMAI(*a*;*x*;[*complement*]) calculates the incomplete gamma function.

**Arguments**

*a* is a positive value.

*x* is a positive value or 0.

*complement* is an optional argument that specifies how 1-2-3 calculates @GAMMAI.

| *complement* | **1-2-3 calculates** |
|---|---|
| 0 | P(*a*;*x*); default if you omit the argument |
| 1 | Q(*a*;*x*); this is equal to 1-P(*a*;*x*) |

**Notes**

@GAMMAI approximates the incomplete gamma function accurately to within six significant figures.

**Examples**

@GAMMAI(7.5;12.497;1) = 0.050024

**Similar @functions**

@GAMMA calculates the gamma distribution function. @GAMMALN calculates the natural logarithm of the gamma function.

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

## @GAMMALN

@GAMMALN(*x*) calculates the natural logarithm of the gamma function.

### Arguments

*x* is any value other than 0 and negative integers.

### Notes

@GAMMALN approximates the natural logarithm of the gamma function accurately to within six significant figures.

### Examples

@GAMMALN(0.5) = 0.572365

@GAMMALN(5) = 3.178054

### Similar @functions

@GAMMA calculates the gamma distribution function. @GAMMAI calculates the incomplete gamma function.

### See also

### Help

@Functions

### *User's Guide*

Chapter 11, "Calculating with @Functions"

**@GEOMEAN**

@GEOMEAN(*list*) calculates the geometric mean of the values in *list*.

**Arguments**

*list* can contain any of the following, in any combination: numbers, numeric formulas that evaluate to numbers, and addresses or names of ranges that contain numbers or numeric formulas. Separate elements of *list* with underlined argument separators.

All values in *list* must be greater than 0.

**Notes**

The geometric mean of a *list* with *n* values is the *n*th root of the product of the values in *list*.

For the same *list*, the result of @GEOMEAN is less than the result of @AVG unless all values in *list* are equal. If all values in list are equal, the results of @GEOMEAN and @AVG are equal.

**Examples**

@GEOMEAN(A1..A4) = 239.1886, when A1..A4 contains the values 160, 227, 397, and 227.

**Similar @functions**

@HARMEAN calculates the harmonic mean of the values in a list. @AVG and @PUREAVG calculate the average of the values in a list. @MEDIAN calculates the median value in a list of values.

**See also**

**Help**

@Functions
Statistical @Function Arguments

*User's Guide*

Chapter 11, "Calculating with @Functions"

**@HARMEAN**

@HARMEAN(*list*) calculates the harmonic mean of the values in *list*.

**Arguments**

*list* can contain any of the following, in any combination: numbers, numeric formulas that evaluate to numbers, and addresses or names of ranges that contain numbers or numeric formulas. Separate elements of *list* with argument separators.

All values in *list* must be greater than 0.

**Notes**

The harmonic mean of the values in *list* is the reciprocal of the arithmetic mean of the reciprocals of the values in *list*. For example, the average of 1/2, 1/3, and 1/4 is 13/36; therefore, the harmonic mean of 2, 3, and 4 is 36/13.

For the same *list*, the result of @HARMEAN is always less than the result of @GEOMEAN.

**Examples**

@HARMEAN(25;50;75) = 40.90909

**Similar @functions**

@GEOMEAN calculates the geometric mean of the values in a list. @AVG and @PUREAVG calculate the average of the values in a list. @MEDIAN calculates the median value in a list of values.

**See also**

**Help**

@Functions
Statistical @Functions Arguments

*User's Guide*

Chapter 11, "Calculating with @Functions"

**@HEX**

@HEX($x$) converts a decimal number to its hexadecimal equivalent.

**Arguments**

$x$ is an integer from -2,147,483,648 through 2,147,483,647. If $x$ is not an integer, 1-2-3 truncates it to an integer.

**Notes**

Hexadecimal values from 00000000 through 7FFFFFFF correspond to 0 and positive decimal values.

Hexadecimal values from 80000000 through FFFFFFFF correspond to negative decimal values.

**Examples**

@HEX(162) = A2

**Similar @functions**

@DECIMAL converts hexadecimal numbers to decimal numbers.

---

**See also**

**Help**

   @Functions

*User's Guide*

   Chapter 11, "Calculating with @Functions"

**@HLOOKUP**

@HLOOKUP(*x;range;row-offset*) finds the contents of the cell in a specified row of a horizontal lookup table, a range with either value information in ascending order or labels in the first row.

**Arguments**

*x* is either a value or text, depending on the contents of the first row of the horizontal lookup table.

| If the first row contains | Then *x* can be |
| --- | --- |
| Values | Any value greater than or equal to the first value in *range*. If *x* is smaller than the first value in *range*, @HLOOKUP returns ERR. If *x* is larger than the last value in the first row of *range,* @HLOOKUP stops at the last cell in the row specified by *row-offset* and returns the contents of that cell as the answer. |
| Labels | Text enclosed in " " (quotation marks), a formula that results in text, or the address or name of a cell that contains a label or a formula that results in a label. If *x* does not exactly match the contents of a cell in the first row of *range*, @HLOOKUP returns ERR. |

*range* represents the location of the horizontal lookup table. *range* is any range address or range name. If *range* is a 3D range, 1-2-3 uses only the first worksheet in *range*.

*row-offset* represents an offset number corresponding to the position the row occupies in *range*.

**Notes**

@HLOOKUP compares *x* to each cell in the first row of the table. When 1-2-3 locates a cell in the first row that contains *x* (or if *x* is a value; the value closest to; but not larger than; *x*), it moves down that column the number of rows specified by *row-offset* and returns the contents of that cell as the answer.

**Examples**

@HLOOKUP

**Similar @functions**

@VLOOKUP finds the contents of a cell in a vertical lookup table. @INDEX finds the contents of a cell when you specify offset numbers for both the column and row. @CHOOSE replaces a lookup table that requires only one row. @MATCH finds the relative position of a cell with specified contents. @XINDEX finds the contents of a cell specified by column, row, and worksheet headings.  @MAXLOOKUP returns an absolute reference to the cell that contains the largest value in a list of ranges. @MINLOOKUP returns an absolute reference to the cell that contains the smallest value in a list of ranges.

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**Example: @HLOOKUP**

A horizontal lookup table named RATES (A2..E7) lists rates for sending a parcel to several cities.

@HLOOKUP("Frankfurt";RATES;3), entered in a cell formatted as Currency with two decimal places, returns $24.00, the rate for sending a type 3 parcel to Frankfurt.

```
A ---- A ---------- B -------- C ----- D -------- E ---
1                    --------- PARCEL DESTINATION ---------
2 Parcel type London Paris   Frankfurt     New York
3             1      $18.36 $19.33  $20.12   $9.29
4             2      $20.32 $21.66  $22.03  $11.25
5             3      $22.44 $23.88  $24.00  $13.25
6             4      $24.14 $25.26  $25.75  $16.85
7             5      $28.32 $29.00  $29.80  $19.54
```

**@HOUR**

@HOUR(*time-number*) extracts the hour, a value between 0 (midnight) and 23 (23:00 or 11:00 PM), from *time-number*.

**Arguments**

*time-number* is a value from .000000 (midnight) through .999988 (11:59:59 PM). Usually, another time @function supplies *time-number*.

**Notes**

The hour portion is useful in calculations that involve whole hours, such as calculating hourly wages or hours elapsed since you began working on a project, or time-stamping a worksheet.

**Examples**

@HOUR(.51565) = 12 because .51565 is the time number for 12:22:32 PM.

@HOUR(@TIME(13;45;18)) = 13 (1:00 PM), because 13 is the *hour* argument for @TIME(13;45;18).

**Similar @functions**

@MINUTE extracts the minutes, and @SECOND extracts the seconds, from a time number.

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

## @IF

@IF(*condition;x;y*) evaluates *condition* and returns one of two values, depending on the result of the evaluation. If *condition* is true, @IF returns *x*; if *condition* is false, @IF returns *y*.

### Arguments

*condition* is usually a logical formula. However, you can use any formula, number, text enclosed in " " (quotation marks), or name or address of a cell as *condition*. 1-2-3 evaluates any *condition* that does not equal zero as true and any *condition* that does equal zero as false. Blank cells, text, and ERR and NA values all equal zero when used as *condition*.

*x* and *y* are values, text enclosed in " " (quotation marks), or the addresses or names of cells that contain values or labels.

### Notes

@IF is useful when combined with @ERR and @NA to document errors or missing data in formulas. It is also useful in preventing ERR, NA, and calculation errors in situations where data may be missing or inaccurate, for example, to prevent division by zero.

You can nest @IF functions within one another to create a complex condition. For example,

@IF(TOT>10000;TOT*0.15;@IF(TOT>5000;TOT*0.10;TOT*0.02))

nests two @IF functions to determine a commission rate based on three levels of sales: total sales greater than $10,000, total sales greater than $5,000, and total sales less than or equal to $5,000.

### Examples

@IF(BALANCE>=0;BALANCE;"Overdrawn") returns the value in the cell named BALANCE when the value in BALANCE is 0 or positive; or returns the label Overdrawn when the value in BALANCE is negative.

---

### See also

### Help

@Functions

### *User's Guide*

Chapter 11, "Calculating with @Functions"

**@INDEX**

@INDEX(*range;column;row*;[*worksheet*]) returns the contents of a cell located at the intersection of a specified *column*, *row*, and (optionally) *worksheet* of a range.

**Arguments**

*range* is a range address or range name.

*column* is the offset number of the column that @INDEX uses.

*row* is the offset number of the row that @INDEX uses, or the address or name of a cell that contains 0 or a positive integer.

*worksheet* is an optional argument that is the offset number of the worksheet that @INDEX uses. If you do not specify *worksheet*, @INDEX uses only the first worksheet in *range*.

**Examples**

@INDEX

**Similar @functions**

@HLOOKUP and @VLOOKUP find entries in horizontal and vertical lookup tables. @CHOOSE finds an entry in a list. @MATCH finds the relative position of a cell with specified contents. @XINDEX finds the contents of a cell specified by column, row, and worksheet headings.   @MAXLOOKUP returns an absolute reference to the cell that contains the largest value in a list of ranges. @MINLOOKUP returns an absolute reference to the cell that contains the smallest value in a list of ranges.

_____

**See also**

**Help**

@Functions

*User's Guide*

Chapter 11, "Calculating with @Functions"

**Example: @INDEX**

A table named INCREASE (A3..E8) shows salary increases based on employee performance ratings.

@INDEX(INCREASE;2;3), entered in a cell formatted as percent, returns 5%, the salary increase for an employee who has a rating of 3 and has a salary level of 2.

@INDEX(INCREASE;1;2), entered in a cell formatted as Percent, returns 7%, the salary increase for an employee who has a rating of 2 and a salary level of 1.

```
A ----- A ----------- B --- C --- D --- E --
1
2                        --- SALARY LEVEL ---
3     Rating        1       2       3       4
4          1       10%      9%      8%      7%
5          2        7%      6%      5%      4%
6          3        6%      5%      4%      3%
7          4        3%      2%      1%      0%
8          5        0%      0%      0%      0%
```

**@INFO**

@INFO(*attribute*) returns information for the current 1-2-3 session.

**Arguments**

*attribute* is one of the following items, entered as <u>text</u>.

| *attribute* | Returns |
| --- | --- |
| author | The user name of the person who first saved the current file |
| creation-date | A <u>date number</u> that corresponds to the date the current file was first saved |
| editing-time | A <u>time number</u> that corresponds to the total number of hours and minutes the current file has been open |
| dbreturncode | The most recent error code returned by the DataLens driver |
| dbdrivermessage | The most recent DataLens message |
| dbrecordcount | The number of records extracted, modified, or inserted from the last query (in the worksheet or in an external database) |
| directory | The current <u>path</u>, including the drive letter |
| last-revision-by | The user name of the person who last saved the current file |
| last-revision-date | A date number that corresponds to the date the current file was last saved |
| macro-step | Yes if Step mode is on; No if Step mode is off |
| macro-trace | Yes if the Macro Trace window is open; No if it is not open |
| memavail | The amount of available memory |
| mode | The current mode: |
| | 0  Wait |
| | 1  Ready |
| | 2  Label |
| | 3  Menu |
| | 4  Value |
| | 5  Point |
| | 6  Edit |
| | 7  Error |
| | 8  Find |
| | 9  Files |
| | 10 Help |
| | 11 Stat |
| | 13 Names |
| | 99 All other modes (for example, user-defined with <u>{INDICATE}</u>) |
| numfile | The number of currently open files |
| origin | The absolute address of the top left cell in the current worksheet |
| osreturncode | The value returned by the most recent operating system command |

| | |
|---|---|
| osversion | The current operating system version |
| recalc | The current recalculation mode as one of the two labels, automatic or manual |
| release | The release number for the 1-2-3 for Windows product being used, consisting of three parts: major release number, upgrade level, and version number |
| setup-user-name | Your email or network user name |
| screen-height | The height of the screen, in pixels |
| screen-width | The width of the screen, in pixels |
| selection | The address of the currently selected range, or the name of the currently selected chart, drawn object, or query table |
| selection-part | The name of the selected part of a range or object, for example, Title line 1 for the first line of a title of a chart |
| selection-type | The current selection type: Range, Draw, Query, or Chart |
| system | The name of the operating system |
| totmem | The total memory available (both the amount currently available and the amount being used) |
| windir | The path to the directory that contains Windows, including the drive letter |
| worksheet-number | The number of worksheets in the current file |
| worksheet-size | The size of the current file, in Kilobytes (KB) |

**Notes**

In addition to the attributes listed here, *attribute* can be any of the Info components.

Recalculate your work before you use @INFO to be sure the results are correct.

@INFO is useful in macros when you need to provide information about the status of 1-2-3 to the user or the macro (for example, to tell the user the current path in a macro that automates saving files; or to warn that memory is low).

Use @INFO with @IF to check the status of 1-2-3 and to tell a macro what to do in certain conditions, such as to change the path if necessary or to delete unnecessary data or close active files if memory is low.

**Examples**

@INFO("numfile") = 2, if two files are active.

@INFO(B4) = 3, if B4 contains the label "mode" and 1-2-3 is in Menu mode.

**Similar @functions**

@CELL returns information about the first cell in a range. @CELLPOINTER returns information about the current cell.

---

**See also**

**Help**

   @Functions

***User's Guide***

   Chapter 11, "Calculating with @Functions"

**@INT**

@INT(*x*) returns the integer portion of *x*.

**Arguments**

*x* is a value.

**Notes**

Select Fixed from Style Number Format or Style Worksheet Number Defaults Format to display values with a specified number of decimal places if you want 1-2-3 to calculate the values to their full precision; do not use @INT.

**Examples**

@INT(35.67) = 35

@INT(@NOW) = the date number for the current date and time, because the time portion is a decimal value.

**Similar @functions**

@ROUND, @ROUNDUP, and @ROUNDDOWN round a value to the closest multiple of the specified power of 10. @ROUNDM rounds a value to a specified multiple. @EVEN rounds a value away from 0 to the nearest even integer. @ODD rounds a value away from 0 to the nearest odd integer. @TRUNC truncates a value to a specified number of decimal places.

---

**See also**

**Help**

@Functions

*User's Guide*

Chapter 11, "Calculating with @Functions"

**@IPAYMT**
**@PPAYMT**

@IPAYMT(*principal*;*interest*;*term*;*start-period*;[*end-period*];[*type*];[*future-value*]) calculates the cumulative interest portion of the periodic payment on a loan (*principal*) at a given *interest* rate for a specified number of payment periods (*term*).

@PPAYMT(*principal*;*interest*;*term*;*start-period*;[*end-period*];[*type*];[*future-value*]) calculates the principal portion of the periodic payment on a loan (*principal*) at a given *interest* rate for a specified number of payment periods (*term*).

### Arguments

*principal* and *term* are values. *term* can be any value except 0.

*interest* is a decimal or percentage value greater than -1.

*start-period* is the point in the loan's term when you want to begin calculating interest or principal. *start-period* can be any value greater than or equal to 1, but cannot be greater than *term*.

*end-period* is the point in the loan's term when you want to stop calculating interest or principal. *end-period* can be any value greater than *start-period*. If you omit the *end-period* argument, *end-period* equals *start-period*.

*type* is an optional argument that specifies whether to calculate for an ordinary annuity or for an annuity due.

| *type* | 1-2-3 calculates for |
| --- | --- |
| 0 | Ordinary annuity (payments due at the end of a period); default if you omit the argument |
| 1 | Annuity due (payment due at the beginning of a period) |

*future-value* is an optional argument that specifies the future value of the series of payments. *future-value* can be any value. If you omit the *future-value* argument, 1-2-3 uses 0.

You cannot use an optional argument without using the ones that precede it.

### Notes

The period used to calculate *interest* must be the same period used for *term*; for example, if you are calculating a monthly payment, enter the interest and term in monthly increments. Usually, this means you must divide the interest rate by 12 and multiply the number of years in *term* by 12.

### Examples

You took out an $8,000 loan for 3 years at an annual interest rate of 10.5%, compounded monthly. Your monthly payments are $260.02. To determine the interest portion of the last year's payments:

@IPAYMT(8000;0.105/12;36;25;36) = $170.45

To determine the principal portion of the last year's payments:

@PPAYMT(8000;0.105/12;36;25;36) = $2,949.79

### Similar @Functions

@PMT calculates the periodic payment for a loan.

---

**See also**

**Help**

@Functions

**User's Guide**

Chapter 11, "Calculating with @Functions"

## @IRATE

@IRATE(*term*;*payment*;*present-value*;[*type*];[*future-value*];[*guess*]) calculates the periodic interest rate necessary for an annuity (*present-value*) to grow to a *future-value* over the number of compounding periods in *term*.

### Arguments

*term* is a positive integer.

*payment* and *present-value* are values.

*type* is an optional argument that specifies whether to calculate for an ordinary annuity or for an annuity due.

| *type* | 1-2-3 calculates for |
| --- | --- |
| 0 | Ordinary annuity (payments due at the end of a period); default if you omit the argument |
| 1 | Annuity due (payment due at the beginning of a period) |

*future-value* is an optional argument that specifies the future value of the series of payments. *future-value* can be any value. If you omit the *future-value* argument, 1-2-3 uses 0.

*guess* is an optional argument that represents your estimate of the interest rate. *guess* is a value from 0 through 1. If you omit the *guess* argument, 1-2-3 uses .10 (10%).

You cannot use an optional argument without using the ones that precede it.

### Notes

@IRATE uses a series of approximations, starting with your *guess* value, to calculate the interest rate. Start with a *guess* that you feel is reasonable for the interest rate. More than one solution may be possible, so try another *guess* if the result is less than 0 or greater than 1.

If @IRATE cannot approximate the result to within 0.0000001 after 30 calculation iterations, the result is ERR. If your guesses continue to return ERR, use @NPV to determine a better guess. If @NPV returns a positive value, your guess is too low. If @NPV returns a negative value, your guess is too high. @NPV returns 0 if your guess is accurate.

The period used to calculate *guess* must be the same period used for *term*; for example, if you are calculating a monthly payment, enter the interest and term in monthly increments. Usually, this means you must divide the interest rate and the number of years in *term* by 12.

### Examples

You deposited $6,000 in an account and want to withdraw $100 per month for eight years. To determine the interest you need to earn in order to make the withdrawals:

@IRATE(96;100;6000;0;0;0.01) = 0.010623, or 1.06% compounded monthly.

### Similar @functions

@NPV calculates the net present value of a series of future cash flows. @PV and @PVAL calculate the present value of an annuity based on a series of equal payments. @FV and @FVAL calculate the future value of an annuity. @RATE returns the periodic interest rate necessary for an investment to grow to a future value.

---

### See also

### Help

@Functions

*User's Guide*

Chapter 11, "Calculating with @Functions"

**@IRR**

@IRR(*guess;range*) calculates the internal rate of return (profit) for a series of cash-flow values generated by an investment. The internal rate of return is the percentage rate that equates the present value of an expected future series of cash flows to the initial investment.

**Arguments**

*guess* is a decimal or percentage value that represents your estimate of the internal rate of return. In most cases, *guess* should be a percentage between 0 (0%) and 1 (100%). With very large cash flows, make *guess* as accurate as possible.

*range* is the address or name of a range that contains the cash flows. 1-2-3 considers negative numbers as cash outflows and positive numbers as cash inflows. Normally, the first cash-flow amount in the range is a negative number (a cash outflow) that represents the investment. 1-2-3 assigns the value 0 to all blank cells and labels in *range* and includes them in the calculation.

**Notes**

Use @IRR to determine the profitability of an investment. Combine @IRR with other financial @functions, such as @NPV, to assess an investment.

1-2-3 assumes the cash flows are received at regular, equal intervals.

@IRR uses a series of approximations, starting with your *guess* value, to calculate the internal rate of return. Start with a *guess* that you feel is reasonable for the internal rate of return. More than one solution may be possible, so try another *guess* if the result is less than 0 or greater than 1.

If @IRR cannot approximate the result to within 0.0000001 after 30 calculation iterations, the result is ERR. If your guesses continue to return ERR, use @NPV to determine a better guess. If @NPV returns a positive value, your guess is too low. If @NPV returns a negative value, your guess is too high. @NPV returns 0 if your guess is accurate.

Use @AVG to determine the internal rate of return if you calculate several rates.

**Examples**

A schedule calculates the internal rate of return of an initial investment of $10,000 that is followed by 12 monthly payments of $1,500. *guess* (12.00%) is entered in GUESS and the payments are listed in a range named CASHFLOWS.

@IRR(GUESS;CASHFLOWS) = returns 10.45%, the internal rate of return.

**Similar @functions**

@NPV calculates the net present value of a series of future cash flows. @PV and @PVAL calculate the present value of an annuity based on a series of equal payments. @FV and @FVAL calculate the future value of an annuity. @RATE returns the periodic interest rate necessary for an investment to grow to a future value. @MIRR calculates the modified internal rate of return.

---

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@ISAAF**
**@ISAPP**
**@ISMACRO**

@ISAAF(*name*) tests *name* for a defined add-in @function. If *name* is a defined add-in @function, @ISAAF returns 1 (true); if *name* is not a defined add-in @function, @ISAAF returns 0 (false).

@ISAPP(*name*) tests *name* for an add-in application that is currently in memory. If *name* is an add-in application that is currently in memory, @ISAPP returns 1 (true); if *name* is not an add-in application that is currently in memory, @ISAPP returns 0 (false).

@ISMACRO(*name*) tests *name* for a defined add-in macro command. If *name* is a defined add-in macro command, @ISMACRO returns 1 (true); if *name* is not a defined add-in macro command, @ISMACRO returns 0 (false).

### Arguments

*name* is the name of the add-in @function, application, or macro command you want to test, entered as text.

If *name* is the name of an add-in @function, do not include the initial @ symbol.

If *name* is the name of an add-in macro command, do not include the { } (braces).

### Notes

@ISAPP returns 1 (true) only for any add-in applications you load using Tools Add-in Load. For add-ins that only define add-in @functions or macro commands, or any add-in installed in your DCF, @ISAPP returns 0 (false).

### Examples

@ISAAF("degrees") = 1 if @DEGREES is a defined add-in @function.

@ISAAF("dsum") = 0, because @DSUM is a 1-2-3 @function, not an add-in @function.

@ISAPP("finance") = 1 if an add-in application called FINANCE is currently in memory.

@ISMACRO("payroll") = 1 if {PAYROLL} is a defined add-in macro command.

---

### See also

**Help**

   @Functions

### *User's Guide*

   Chapter 11, "Calculating with @Functions"

## @ISERR

@ISERR(*x*) tests *x* for the value <u>ERR</u>. If *x* is the value ERR, @ISERR returns 1 (true); if *x* is not the value ERR, @ISERR returns 0 (false) .

### Arguments

*x* is any value, location, text, or condition.

### Notes

Use @ISERR to block errors that arise from division by zero. For example, the formula @IF(@ISERR(A1/A2),0,A1/A2) tests the result of the division A1/A2 (the contents of cell A1 divided by the contents of cell A2). If the result is the value ERR, the formula returns 0. If the result is any other value, the formula returns that result.

### Examples

The subroutine CHKQTY consists of three short subroutines that check entries in the cells named QTY and PRICE. CHKQTY tests whether the entry in QTY is a value; if it is, processing transfers to the subroutine CKERRNA. If QTY does not contain a value, NEWQTY requests a new entry and then transfers to CHKQTY.

CKERRNA uses @ISERR to determine whether QTY contains the value ERR; if @ISERR returns 1 (true), it requests a new value. If QTY does not contain ERR and PRICE does not contain NA, the subroutine multiplies the values in the two cells and enters the result in the cell named TOTAL.

```
...
CHKQTY          {IF @ISNUMBER(QTY)}{BRANCH CKERRNA}
NEWQTY          {GETNUMBER "Enter Quantity number: ";QTY}
                {BRANCH CHKQTY}
CKERRNA         {IF @ISERR(QTY)}{BRANCH NEWQTY}
                {IF @ISNA(PRICE)}{GETNUMBER "Enter new price: ";PRICE}{BRANCH CHKQTY}
                {GOTO}TOTAL~+QTY*Price~
...
```

### Similar @functions

<u>@ISNA</u> tests for the value NA.

---

### See also

### Help

@Functions

### *User's Guide*

Chapter 11, "Calculating with @Functions"

## @ISNA

@ISNA(*x*) tests *x* for the value <u>NA</u>. If *x* is the value NA, @ISNA returns 1 (true); if *x* is not the value NA, @ISNA returns 0 (false).

### Arguments

*x* is any value, location, text, or condition.

### Examples

The subroutine CHKQTY consists of three short subroutines that check entries in the cells named QTY and PRICE. CHKQTY tests whether the entry in QTY is a value; if it is, processing transfers to the subroutine CKERRNA. If QTY does not contain a value, NEWQTY requests a new entry and then transfers to CHKQTY.

CKERRNA uses @ISNA to determine whether PRICE contains the value NA; if @ISNA returns 1 (true), it requests a new value. If PRICE does not contain NA and QTY does not contain ERR, the subroutine multiplies the values in the two cells and enters the result in the cell named TOTAL.

```
...
CHKQTY          {IF @ISNUMBER(QTY)}{BRANCH CKERRNA}
NEWQTY          {GETNUMBER "Enter Quantity number: ",QTY}
                {BRANCH CHKQTY}
CKERRNA         {IF @ISERR(QTY)}{BRANCH NEWQTY}
                {IF @ISNA(PRICE)}{GETNUMBER "Enter new price: ",PRICE}{BRANCH CHKQTY}
                {GOTO}TOTAL~+QTY*Price~
...
```

### Similar @functions

<u>@ISERR</u> tests for the value ERR.

---

### See also

**Help**

    <u>@Functions</u>

### *User's Guide*

    Chapter 11, "Calculating with @Functions"

## @ISNUMBER

@ISNUMBER(*x*) tests *x* for a value. If *x* is a value, <u>NA</u>, <u>ERR</u>, or a <u>blank cell</u>, @ISNUMBER returns 1 (true); if *x* is text or a cell that contains a label or a formula that results in a label, @ISNUMBER returns 0 (false).

## Arguments

*x* is any value, location, text, or condition. If *x* is a multiple-cell range, @ISNUMBER returns 0 (false), even if the first cell of the range contains a value.

## Examples

The subroutine CHKQTY consists of three short subroutines that check entries in the cells named QTY and PRICE.

CHKQTY uses @ISNUMBER to determine whether the entry in QTY is a value; if it is, processing transfers to the CKERRNA subroutine. If QTY does not contain a value, NEWQTY requests a new entry.

```
...
CHKQTY          {IF @ISNUMBER(QTY)}{BRANCH CKERRNA}
NEWQTY          {GETNUMBER "Enter Quantity number: ",QTY}
                {BRANCH CHKQTY}
CKERRNA         {IF @ISERR(QTY)}{BRANCH NEWQTY}
                {IF @ISNA(PRICE)}{GETNUMBER "Enter new price: ",PRICE}{BRANCH CHKQTY}
                {GOTO}TOTAL~+QTY*Price~
...
```

## Similar @functions

<u>@ISSTRING</u> tests for a label. <u>@CELL</u> and <u>@CELLPOINTER</u>can also determine whether a cell contains a value or a label.

---

## See also

**Help**

   <u>@Functions</u>

## *User's Guide*

   Chapter 11, "Calculating with @Functions"

## @ISRANGE

@ISRANGE(*range*) tests *range* for a <u>defined range name</u> or valid range address (a range address with worksheet and column letters from A through IV and row numbers from 1 through 8192). If *range* is a defined range name or valid range address, @ISRANGE returns 1 (true); if *range* is not a defined range name or valid range address, @ISRANGE returns 0 (false).

### Arguments

*range* is any text or range address.

### Notes

@ISRANGE is useful with @IF to determine if an entry is a valid range name for subroutine calls and branching with <u>{DISPATCH}</u>.

You can use @ISRANGE only with files in memory.

### Examples

@ISRANGE(A1) = 1 (true).

@ISRANGE(+A1) = 0 (false).

@ISRANGE(A1..C3) = 1(true).

@ISRANGE(SALES) = 1 (true), if SALES is a defined range name.

@ISRANGE(PRICE) = 0 (false), if PRICE is an undefined range name.

@ISRANGE(3) = 0 (false).

@ISRANGE("COMMISSION") = 0 (false) because the range name is enclosed in " " (quotation marks).

---

### See also

**Help**

   <u>@Functions</u>

### *User's Guide*

   Chapter 11, "Calculating with @Functions"

## @ISSTRING

@ISSTRING(*x*) tests *x* for text or a label. If *x* is text or a cell that contains a label or a formula that results in a label, @ISSTRING returns 1 (true); if *x* is a value, <u>ERR</u>, <u>NA</u>, or <u>blank cell</u>, @ISSTRING returns 0 (false).

### Arguments

*x* is any value, location, text, or condition.

### Examples

The subroutine CHKSTR checks the contents of the cell named CUSTOMER. If CUSTOMER contains a label (@ISSTRING(CUSTOMER) = 1), the subroutine branches to a new subroutine named FILEORDER. If CUSTOMER does not contain a label, the subroutine requests a new entry.

```
...
CHKSTR        {IF @ISSTRING(CUSTOMER)}{BRANCH FILEORDER}
              {GETLABEL "Enter CUSTOMER name: ",CUSTOMER}
              {CHKSTR}
...
```

### Similar @functions

<u>@ISNUMBER</u> tests for a value. <u>@CELL</u> and <u>@CELLPOINTER</u>can also determine whether a cell contains a value or a label.

---

### See also

### Help

   <u>@Functions</u>

### *User's Guide*

   Chapter 11, "Calculating with @Functions"

**@KURTOSIS**

@KURTOSIS(*range*;[*type*]) calculates the kurtosis of the values in *range*.

**Arguments**

*range* is the name or address of a range that contains values. If *range* contains fewer than four values, @KURTOSIS returns the value ERR.

*type* is an optional argument that specifies whether to calculate for a population or a sample.

| *type* | **1-2-3 calculates** |
| --- | --- |
| 0 | Population kurtosis; default if you omit the argument |
| 1 | Sample kurtosis |

**Notes**

Kurtosis is a measure of the concentration of a distribution about its mean. Positive kurtosis indicates a relatively peaked distribution; negative kurtosis indicates a relatively flat distribution.

**Examples**

The range DATA contains these values: 2, 5, 5, 9, 1, 2, 4.

@KURTOSIS(DATA;1) = 1.021488

@KURTOSIS(DATA) = -0.32438

**Similar @functions**

@SKEWNESS calculates the skewness of the values in a range.

**See also**

**Help**

@Functions
Statistical @Function Arguments

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@LARGE**

@LARGE(*range*;*n*) finds the *n*th largest value in *range*.

**Arguments**

*range* is the name or address of a range that contains values.

*n* is any positive integer. If *n* is larger than the number of values in *range*, @LARGE returns <u>ERR</u>.

**Examples**

A range named SCORES contains these test scores: 87, 85, 90, 80, 82, 92, 79, 85, 95, 86.

@LARGE(SCORES;3) returns 90, the third-highest score.

**Similar @functions**

<u>@SMALL</u> finds the *n*th smallest value in a range. <u>@MAX</u> and <u>@PUREMAX</u> find the largest value in a range. <u>@MIN</u> and <u>@PUREMIN</u> find the smallest value in a range.

**See also**

**Help**

    <u>@Functions</u>

***User's Guide***

    Chapter 11, "Calculating with @Functions"

## @LEFT

@LEFT(*text;n*) returns the first *n* characters in *text*.

### Arguments

*text* is text enclosed in " " (quotation marks), a formula that results in text, or the address or name of a cell that contains a label or a formula that results in a label.

*n* can be a positive integer or 0. If *n* is 0, the result is an empty string. If *n* is greater than the length of *text*, @RIGHT returns all of *text*.

### Notes

@LEFT counts punctuation and spaces as characters.

@LEFT is useful for copying only part of a label into another cell, starting at the beginning of the label (for example, for separating titles such as Dr. and Ms. from names).

In a macro, @LEFT can extract parts of labels the user enters to store them in a database, for subroutine calls, or to alter the macro itself.

Use @LEFT with @FIND when you do not know the exact value for *n*, or when *n* may vary.

### Examples

@LEFT(PHONE;3) = the area code for the telephone number in the cell PHONE.

@LEFT(A1;@FIND("*";A1;0)) = the first name in cell A1 (for example, Richard if cell A1 contains the name Richard Smith). The * (asterisk) represents one space.

### Similar @functions

@MID returns characters from within *text*. @RIGHT returns the last *n* characters in *text*.

---

### See also

### Help

@Functions

### *User's Guide*

Chapter 11, "Calculating with @Functions"

**@LENGTH**

@LENGTH(*text*) counts the characters in *text*.

**Arguments**

*text* is text enclosed in " " (quotation marks), a formula that results in text, or the address or name of a cell that contains a label or a formula that results in a label.

**Notes**

@LENGTH counts punctuation and spaces as characters.

Use @LENGTH with @TRIM to find the length of *text* without including leading, trailing, or consecutive spaces.

@LENGTH is also useful in any application in which labels should be a certain length, such as ZIP codes and purchase order numbers.

**Examples**

@LENGTH("fiscal") = 6.

@LENGTH(A5&G12) = the total number of characters in cells A5 and G12.

@LENGTH(@TRIM("     Mr.      Jones")) = 9.

---

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@LN**

@LN($x$) calculates the natural logarithm (base $e$) of $x$.

**Arguments**

$x$ is a value greater than 0.

**Notes**

A natural logarithm is one that uses the number $e$ (approximately 2.718282) as a base.

**Examples**

@LN(2) = 0.693147

@LN(@EXP(1)) = 1, because @EXP(1) = 2.718282.

**Similar @functions**

@EXP is the inverse of @LN. @LOG calculates the common logarithm (base 10) of $x$.

_____

**See also**

**Help**

  @Functions

***User's Guide***

  Chapter 11, "Calculating with @Functions"

**@LOG**

@LOG(*x*) calculates the common logarithm (base 10) of *x* .

**Arguments**

*x* is a value greater than 0.

**Examples**

10^(@LOG(8)/3) = 2, the cube root of 8.

@LOG(B3) = 0.60206, if cell B3 contains the value 4.

**Similar @functions**

@LN calculates the natural logarithm (base *e*) of a value.

**See also**

**Help**

@Functions

*User's Guide*

Chapter 11, "Calculating with @Functions"

**@LOWER**

@LOWER(*text*) converts all the letters in *text* to lowercase.

**Arguments**

*text* is text enclosed in " " (quotation marks), a formula that results in text, or the address or name of a cell that contains a label or a formula that results in a label.

**Notes**

If you selected ASCII sort order during Install, capitalization affects the sort order of labels when you use Range Sort or Query Sort; two otherwise identical labels may not appear together if their capitalization is different.

**Examples**

@LOWER("Sales Forecast") = sales forecast.

**Similar @functions**

@UPPER converts all letters in *text* to uppercase. @PROPER converts the first letter of each word in *text* to uppercase and converts the rest of the letters to lowercase.

---

**See also**

**Help**

   @Functions

***User's Guide***

   Chapter 11, "Calculating with @Functions"

**@MAX**
**@PUREMAX**

@MAX(*list*) finds the largest value in *list*.

@PUREMAX(*list*) finds the largest value in *list*, ignoring cells that contain labels.

## Arguments

*list* can contain any of the following, in any combination: numbers, numeric formulas, and addresses or names of ranges that contain numbers or numeric formulas. Separate elements of *list* with argument separators.

## Examples

A range named TEST contains the following data: -5, -7, -9, -11, January.

@MAX(TEST) returns 0, the value of the label January, as the largest value in TEST.

@PUREMAX(TEST) ignores the label January and returns -5 as the largest value in TEST.

## Similar @functions

@MIN and @PUREMIN find the smallest value in a list. @DMAX finds the largest value in the field of a database table that meets criteria you specify. @LARGE returns the *n*th largest value in a list.

---

## See also

## Help

@Functions
Statistical @Function Arguments

## *User's Guide*

Chapter 11, "Calculating with @Functions"

**@MEDIAN**

@MEDIAN(*list*) returns the median value in *list*.

**Arguments**

*list* can contain any of the following, in any combination: numbers, numeric formulas, and addresses or names of ranges that contain numbers or numeric formulas. Separate elements of *list* with <u>argument separators</u>.

**Notes**

If *list* contains an odd number of values, @MEDIAN returns the middle value. If *list* contains an even number of values, @MEDIAN returns the arithmetic average of the two middle values.

**Examples**

@MEDIAN(5;12;65;82;9) = 12

@MEDIAN(5;12;65;82;9;78) = 38.50

**Similar @functions**

<u>@GEOMEAN</u> calculates the geometric mean of the values in a list. <u>@HARMEAN</u> calculates the harmonic mean of the values in a list. <u>@AVG</u> and <u>@PUREAVG</u> calculate the average of the values in a list.

---

**See also**

**Help**

<u>@Functions</u>
<u>Statistical @Function Arguments</u>

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@MID**

@MID(*text;start-number;n*) copies *n* characters from *text*, beginning with the character at *start-number*.

**Arguments**

*text* is text enclosed in " " (quotation marks), a formula that results in text, or the address or name of a cell that contains a label or a formula that results in a label.

*start-number* is an <u>offset number</u>. If *start-number* is larger than the length of *text*, the result of @MID is an <u>empty string</u>.

*n* is any positive integer or 0. If *n* is 0, the result of @MID is an <u>empty string</u>. If *n* is larger than the length of *text*, 1-2-3 returns all the characters from *start-number* to the end of *text*.

**Notes**

@MID counts punctuation and spaces as characters.

Use a large number for *n* if you do not know the length of *text*; 1-2-3 ignores the extra spaces and returns all of *text* beginning with *start-number*.

To extract part of a label when you do not know its *start-number*, use @MID with <u>@FIND</u>.

@MID is useful in macros to store parts of labels the user enters, to create subroutine calls, or to alter the macro itself.

**Examples**

@MID("Daily Account Balance";6;7) = Account.

**Similar @functions**

<u>@LEFT</u> returns the first *n* characters of *text*, and <u>@RIGHT</u> returns the last *n* characters in *text*.

---

**See also**

**Help**

   <u>@Functions</u>

***User's Guide***

   Chapter 11, "Calculating with @Functions"

**@MIN**
**@PUREMIN**

@MIN(*list*) finds the smallest value in *list*.

@PUREMIN(*list*) finds the smallest value in *list*, ignoring all cells that contain labels.

**Arguments**

*list* can contain any of the following, in any combination: numbers, numeric formulas, and addresses or names of ranges that contain numbers or numeric formulas. Separate elements of *list* with <u>argument separators</u>.

**Examples**

A range named TEST contains the following entries: 5, 7, 9, 11, January.

@MIN(TEST) returns 0, the value of the label January, as the smallest value in TEST.

@PUREMIN(TEST) ignores the label January and returns 5 as the smallest value in TEST.

**Similar @functions**

<u>@MAX</u> and <u>@PUREMAX</u> find the largest value in *list*. <u>@DMIN</u> finds the smallest value in the field of a database table that meets criteria you specify.

---

**See also**

**Help**

<u>@Functions</u>
<u>Statistical @Function Arguments</u>

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@MINUTE**

@MINUTE(*time-number*) extracts the minutes, a value from 0 through 59, from *time-number*.

**Arguments**

*time-number* is a value from .000000 (midnight) through .999988 (11:59:59 PM). Usually, another time @function supplies *time-number*.

**Notes**

The minutes portion is useful in calculations that involve only minutes, such as the time that has elapsed since the start of an application.

**Examples**

@MINUTE(0.333) = 59 because 0.333 is the time number for 7:59:31.

@MINUTE(@TIME(11;15;45)) = 15 because 15 is the *minutes* argument for @TIME(11;15;45).

**Similar @functions**

@HOUR extracts the hour, and @SECOND extracts the seconds, from a time number.

**See also**

**Help**

   @Functions

*User's Guide*

   Chapter 11, "Calculating with @Functions"

## @MIRR

@MIRR(*range*;*finance-rate*;*reinvest-rate*;[*type*]) calculates the modified internal rate of return (profit) for a series of cash-flow values generated by an investment.

The internal rate of return is the percentage rate that equates the present value of an expected future series of cash flows to the initial investment.

### Arguments

*range* is the address or name of a range that contains the cash flows. 1-2-3 considers negative numbers as cash outflows and positive numbers as cash inflows. *range* must contain at least one positive value and one negative value.

Normally, the first cash-flow amount in the range is a negative number (a cash outflow) that represents the investment. 1-2-3 assigns the value 0 to all blank cells and labels in *range* and includes them in the calculation.

*finance-rate* is the interest rate paid on money used in cash flows.

*reinvest-rate* is the interest rate you receive on cash flows as you reinvest them.

*finance-rate* and *reinvest-rate* can be any values.

*type* specifies the timing of the cash flows. *type* is an integer from the following table:

| *type* | Cash flows occur |
|--------|------------------|
| 0 | At the end of each period; default if you omit the argument |
| 1 | At the beginning of each period |

### Notes

Use @MIRR to determine the profitability of an investment. Combine @MIRR with other financial @functions, such as @NPV, to assess an investment.

1-2-3 assumes the cash flows are received at regular, equal intervals.

Only Release 5 supports the optional *type* argument for @MIRR. If you save an @MIRR formula that contains a type argument and then open the file in Release 4 or Release 4.01, the formula evaluates to ERR.

### Examples

You own an apartment building. Five and six years ago, you borrowed $100,000 at 9.5% annual interest to purchase the building. The following list, stored in the range INCOME, contains your initial investments and your subsequent rental income:

$-100,000
$-100,000
$   45,500
$   47,000
$   48,500
$   50,000
$   50,000

During these years, your reinvested profits earned 11% annually.

@MIRR(INCOME;0.095;0.11) = 7.96%

The rate of return is slightly less if you made the investments at the beginning of the year.

@MIRR(INCOME;0.095;0.11;1) = 7.70%

**Similar @functions**

@IRR calculates the internal rate of return.

_____

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@MOD**
**@MODULO**

@MOD(*x;y*) and @MODULO(*x;y*) calculate the remainder (modulus) of *x*/*y*.

## Arguments

*x* is a value. If *x* is 0, @MOD and @MODULO return 0.

*y* is a value other than 0.

## Notes

The result of @MOD is x - y * @INT(x/y). The sign of the result (+ or -) is always the same as the sign of *x*.

The result of @MODULO is x - y * @ROUNDDOWN(x/y). The sign of the result (+ or -) is always the same as the sign of *y*.

## Examples

@MOD(9;4) = 1

@MODULO(9;4) = 1

@MOD(-14;3) = -2

@MODULO(-14;3) = 1

## Similar @functions

@QUOTIENT calculates the result of *x*/*y*, truncated to an integer.

## See also

**Help**

@Functions
@INT
@ROUNDDOWN

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@MONTH**

@MONTH(*date-number*) extracts the month, a value from 1 through 12, from *date-number*.

**Arguments**

*date-number* is a value from 1 (January 1; 1900) through 73050 (December 31; 2099).

**Notes**

You can use one of the other <u>date @functions</u> to supply the value for *date-number*.

**Examples**

@MONTH(@DATE(91;3;27)) = 3 because 3 is the *month* argument for @DATE(91;3;27).

@MONTH(20181) = 4 because the date number 20181 is the date 02-Apr-55.

@MONTH(@NOW) = the current month.

**Similar @functions**

<u>@DAY</u> calculates the day, using a date number. <u>@YEAR</u> calculates the year, using a date number.

---

**See also**

**Help**

   <u>@Functions</u>

***User's Guide***

   Chapter 11, "Calculating with @Functions"

**@N**

@N(*range*) returns the entry in the first cell of *range* as a <u>value</u>. If the cell contains a <u>label</u>, @N returns the value 0.

**Arguments**

*range* is a cell or range address, or a range name.

**Notes**

@N is useful with any @function or formula when a cell may contain a label and the entry must be a value. Use @N to prevent formulas from resulting in <u>ERR</u>.

@N is also useful in macros to check user entries.

**Examples**

+100+@N(B5..F5) = 885, if cell B5 contains the value 785.

@N(A5)+@N(B5) returns 785, if A5 contains a label and B5 contains the value 785.

**Similar @Functions**

<u>@S</u> returns the entry in the first cell of a range as a label. <u>@ISNUMBER</u> can determine whether a cell contains a value.

_____

**See also**

**Help**

    <u>@Functions</u>

***User's Guide***

    Chapter 11, "Calculating with @Functions"

**@NA**

@NA returns the value <u>NA</u> (not available).

**Notes**

@NA is useful when you are building a worksheet that will contain data that you have not yet determined. Use @NA to flag cells where you will enter the data; formulas that refer to those cells result in the value NA until you supply the correct data.

@NA is also useful to determine which formulas depend on a particular cell.

**Examples**

@IF(@CELL("type";B14)="b",@NA,B14) returns the value NA when B14 is blank.

**Similar @Functions**

<u>@ERR</u> returns the value ERR. <u>@ISNA</u> tests for the value NA.

**See also**

**Help**

    <u>@Functions</u>

*User's Guide*

    Chapter 11, "Calculating with @Functions"

**@NOW**

@NOW calculates the number that corresponds to the current date and time on your computer's clock. This includes both a <u>date number</u> (integer portion) and a <u>time number</u> (decimal portion).

**Notes**

Use @NOW with F2 (EDIT) and F9 (CALC) to create a fixed record of a date and time for time-stamping worksheets or in calculations of elapsed time.

Format the value of @NOW as any of the Date or Time formats. If you format @NOW as a date, 1-2-3 displays only the date (integer) portion of the date and time number. If you format @NOW as time, 1-2-3 displays only the time (decimal) portion of the date and time number. In both cases, 1-2-3 stores and calculates with the entire date and time number.

1-2-3 recalculates @NOW each time you recalculate your work. If you use Tools User Setup Recalculation Automatic, 1-2-3 recalculates @NOW whenever it recalculates another value.

**Examples**

@NOW = 31050.5 at noon on January 3, 1985.

@NOW = 33418.395 at 9:28 A.M. on June 29, 1991.

**Similar @Functions**

<u>@TODAY</u> calculates the date number that corresponds to the current date.

_____

**See also**

**Help**

    <u>@Functions</u>
    <u>Style Number Format</u>

***User's Guide***

    Chapter 11, "Calculating with @Functions"

**@NPV**

@NPV(*interest;range*;[*type*]) calculates the net present value of a series of future cash-flow values (*range*), discounted at a fixed periodic *interest* rate.

**Arguments**

*interest* is a decimal or percentage value greater than -1.

*range* is the range that contains the cash flows.

*type* specifies the timing of the cash flows. *type* is an integer from the following table:

| *type* | Cash flows occur |
| --- | --- |
| 0 | At the end of each period; default if you omit the argument |
| 1 | At the beginning of each period |

**Notes**

Use @NPV to evaluate an investment or to compare one investment with others. @NPV calculates the initial investment necessary to achieve a certain cash outflow at a certain rate.

@NPV returns ERR if *range* contains more than one row or more than one column. For example, @NPV returns ERR if *range* is A1..D25, but does not return ERR if *range* is A1..D1 (a single row) or A1..A25 (a single column).

Only Release 5 supports the optional *type* argument for @NPV. If you save an @NPV formula that contains a *type* argument and then open the file in a previous release of 1-2-3, the formula evaluates to ERR.

**Examples**

This example uses @NPV to discount to today's dollars a series of irregular distributions invested at 11.5% annual percentage rate.

*range* is a list of cash flows, one a month for 12 months, in a range named DISTRIBUTIONS:

```
$       0.00
$       0.00
$    2,500.00
$    2,500.00
$    3,000.00
$    5,000.00
$    6,000.00
$    9,000.00
$    3,000.00
$    2,500.00
$       0.00
$    7,500.00
```

To provide @NPV with the correct number of periods, months in which no distribution is made must be included in *range*. The distributions are monthly, so @NPV requires *interest* (the discount rate), in a cell named DISCOUNT, to be expressed as a monthly percentage:

0.115/12 = 0.96

@NPV(DISCOUNT;DISTRIBUTIONS) = $38,084.13

The result is different if the cash outflows occured at the beginning of each period.

@NPV(DISCOUNT;DISTRIBUTIONS;1) = $38,449.10

**Similar @Functions**

@PV calculates the present value of an annuity based on a series of equal payments. @FV calculates the future value of an annuity.

---

**See also**

**Help**

@Functions

*User's Guide*

Chapter 11, "Calculating with @Functions"

**@ODD**

@ODD(*x*) rounds the value *x* away from 0 to the nearest odd integer.

**Arguments**

*x* is any value. If *x* is an odd integer, @ODD returns *x*.

**Examples**

@ODD(3.25) = 5

@ODD(3) = 3

@ODD(-3.25) = -5

**Similar @functions**

@EVEN rounds a value away from 0 to the nearest even integer. @ROUND, @ROUNDDOWN, and @ROUNDUP round a value to a specified number of decimal places. @ROUNDM rounds a value to a specified multiple. @INT truncates a value, discarding the decimal portion. @TRUNC truncates a value to a specified decimal place.

**See also**

**Help**

@Functions

*User's Guide*

Chapter 11, "Calculating with @Functions"

**@PERCENTILE**

@PERCENTILE(*x*;*range*) calculates the *x*th sample percentile among the values in *range*.

**Arguments**

*x* is the percentile you want to find. *x* is a value from 0 to 1.

*range* is the name or address of the range that contains values.

1-2-3 assigns the value 0 to all labels in *range* and includes them in the @PERCENTILE calculation. 1-2-3 ignores blank cells in *range*.

**Examples**

A range named SCORES contains these test scores: 87, 85, 90, 80, 82, 92, 79, 85, 95, 86. You want to find out the score at the 90th percentile.

@PERCENTILE(0.9;SCORES) = 92.3

**Similar @functions**

@PRANK finds the percentile in a range associated with a value.

---

**See also**

**Help**

   @Functions

***User's Guide***

   Chapter 11, "Calculating with @Functions"

## @PERMUT

@PERMUT(*n*;*r*) calculates the number of ordered sequences (permutations) of *r* objects that can be selected from a total of *n* objects.

### Arguments

*n* is any positive integer or 0.

*r* is any positive integer or 0. *r* cannot be greater than *n*.

If *n* and *r* are not integers, 1-2-3 truncates them to integers.

### Examples

Tests scheduled for 9:00, 10:00, and 11:00 AM will be monitored by three of the five department members. The following formula calculates the number of possible ways of assigning monitors.

@PERMUT(5;3) = 60

### Similar @functions

@COMBIN calculates the number of ways that *r* can be selected from *n*, without regard for order.

### See also

### Help

@Functions

### *User's Guide*

Chapter 11, "Calculating with @Functions"

**@PI**

@PI produces the value π (calculated at 3.14159265358979324). The value π is the ratio of the circumference of a circle to its diameter.

**Examples**

@PI = 3.1415926536

@PI*4^2 = 50.26548, the area of a circle with a radius of 4.

---

**See also**

**Help**

 @Functions

***User's Guide***

 Chapter 11, "Calculating with @Functions"

**@PMT**
**@PAYMT**
**@PMTC**

@PMT(*principal;interest;term*) calculates the payment on a loan (*principal*) at a given *interest* rate for a specified number of payment periods (*term*).

@PAYMT(*principal;interest;term*;[*type*];[*future-value*]) calculates the payment on a loan (*principal*) at a given *interest* rate for a specified number of payment periods (*term*). @PAYMT calculates for either an ordinary annuity or an annuity due, depending on the value you specify for *type*.

@PMTC(*principal;interest;term*) is a special form of @PMT that supports Canadian mortgage conventions.

## Arguments

*principal* and *term* are values.

*interest* is a decimal or percentage value greater than -1.

*type* is an optional argument that specifies whether to calculate for an ordinary annuity or for an annuity due.

| *type* | 1-2-3 calculates for |
| --- | --- |
| 0 | Ordinary annuity (payments due at the end of a period); default if you omit the argument |
| 1 | Annuity due (payment due at the beginning of a period) |

*future-value* is an optional argument that specifies the future value of the series of payments. *future-value* can be any value. If you omit the *future-value* argument, 1-2-3 uses 0.

You cannot use an optional argument without using the ones that precede it.

## Notes

For @PMT and @PAYMT, the period used to calculate *interest* must be the same period used for *term*; for example, if you are calculating a monthly payment, enter the interest and term in monthly increments. Usually, this means you must divide the interest rate by 12 and multiply the number of years in *term* by 12.

For @PMTC, the period used to calculate *interest* is years while the period for *term* is months.

## Examples

You are considering taking out an $8,000 loan for 3 years at an annual interest rate of 10.5%, compounded monthly. Payments are due on the last day of each month. You want to determine your monthly payment:

@PMT(8000;0.105/12;36) = $260.02

If payments are due on the first day of each month:

@PAYMT(8000;0.105/12;36;1;-2500) = $198.90

If you calculate the monthly payment using @PMTC:

@PMTC(8000;0.105;36) = $259.18

## Similar @functions

@IPAYMT calculates the cumulative interest portion of the periodic payment for an investment. @PPAYMT calculates the principal portion of the periodic payment for an investment. @TERM calculates

the number of payment periods of an investment.

---

**See also**

**Help**

    <u>@Functions</u>

***User's Guide***

    Chapter 11, "Calculating with @Functions"

## @POISSON

@POISSON(*x*;*mean*;[*cumulative*]) calculates the Poisson distribution.

### Arguments

*x* is the number of observed events and is a positive integer or 0.

*mean* is the expected number of events and is a positive integer.

If *x* and *mean* are not integers, 1-2-3 truncates them to integers.

*cumulative* is an optional argument that specifies how 1-2-3 calculates @POISSON.

| *cumulative* | 1-2-3 calculates |
|---|---|
| 0 | The probability of exactly *x* events; default if you omit the argument |
| 1 | The probability of, at most, *x* events |

### Notes

@POISSON approximates the Poisson distribution to within ± 3*10^-7.

@POISSON is useful for predicting the number of events that occur during a specified period of time, for example, the number of visitors who pass through the gates of an amusement park in one hour.

### Examples

You expect six cars to pass through a toll booth in one hour. To determine the probability that at most four cars will pass through the toll booth in one hour:

@POISSON(4;6;1) = 0.285057, or 28.51%

To determine the probability that exactly four cars will pass through the toll booth in one hour:

@POISSON(4;6) = 0.133853, or 13.39%

### See also

### Help

   @Functions

### *User's Guide*

   Chapter 11, "Calculating with @Functions"

**@PRANK**

@PRANK(*x*;*range*;[*places*]) finds the percentile of *x* among the values in *range*.

**Arguments**

*x* is any value.

*range* is the name or address of a range that contains values.

*places* is an optional argument that specifies the number of decimal places to round the result of @PRANK. *places* is a value from 0 to 100. If you omit the *places* argument, 1-2-3 uses 2.

**Notes**

If *x* is not a value in *range*, 1-2-3 assigns the 0th percentile position to the lowest value in *range* and assigns the 100th percentile position to the highest value in *range* and interpolates.

**Examples**

A range named SCORES contains these test scores: 87, 85, 90, 80, 82, 92, 79, 85, 95, 86. You want to determine the percentile for a score of 90.

@PRANK(90;SCORES) = 0.78, or 78%.

**Similar @functions**

@PERCENTILE calculates a sample percentile for the values in a list of values.

---

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

## @PRICE

@PRICE(*settlement*;*maturity*;*coupon*;*yield*;[*redemption*];[*frequency*];[*basis*]) calculates the price per $100 face value for securities that pay periodic interest.

### Arguments

*settlement* is the security's settlement date. *settlement* is a date number.

*maturity* is the security's maturity date. *maturity* is a date number. If *maturity* is less than or equal to *settlement*, @PRICE returns ERR.

*coupon* is the security's annual coupon rate. *coupon* is any positive value or 0.

*yield* is the annual yield. *yield* is any positive value.

*redemption* is an optional argument that specifies the security's redemption value per $100 face value. *redemption* is any positive value or 0. If you omit the *redemption* argument, 1-2-3 uses 100.

*frequency* is an optional argument that specifies the number of coupon payments per year. *frequency* is a value from the following table:

| *frequency* | Frequency of coupon payments |
|---|---|
| 1 | Annual |
| 2 | Semiannual; default if you omit the argument |
| 4 | Quarterly |
| 12 | Monthly |

*basis* is an optional argument that specifies the type of day-count basis to use. *basis* is a value from the following table:

| *basis* | Day count basis |
|---|---|
| 0 | 30/360; default if you omit the argument |
| 1 | Actual/actual |
| 2 | Actual/360 |
| 3 | Actual/365 |

You cannot use an optional argument without using the ones that precede it.

### Examples

A bond has a July 1, 1993, settlement date and a December 1, 1998, maturity date. The semiannual coupon rate is 5.50% and the annual yield is 5.61%. The bond has a 30/360 day-count basis.

To determine the bond's price:

@PRICE(@DATE(93;7;1),@DATE(98;12;1),0.055,0.0561,100,2,0) = $99.49

### Similar @functions

@ACCRUED calculates the accrued interest for securities that pay periodic interest. @YIELD calculates the yield for securities that pay periodic interest. @DURATION calculates the annual duration and @MDURATION calculates the annual modified duration for securities that pay periodic interest.

___

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@PRODUCT**

@PRODUCT(*list*) multiplies the values in *list*.

**Arguments**

*list* can contain any of the following, in any combination: numbers, numeric formulas, and addresses or names of ranges that contain numbers or numeric formulas. Separate elements of *list* with argument separators.

**Examples**

@PRODUCT(2; 4; 6; 8) = 384

**Similar @functions**

@FACT calculates the factorial of a value. @SUM adds the values in a list. @SUMPRODUCT calculates the sum of the products of corresponding values in multiple ranges.

---

**See also**

**Help**

@Functions
Statistical @Function Arguments

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@PROPER**

@PROPER(*text*) capitalizes the first letter of each word in *text* and converts the remaining letters to lowercase.

**Arguments**

*text* can be text enclosed in " " (quotation marks), a formula that results in text, or the address or name of a cell that contains a label or a formula that results in a label.

**Notes**

@PROPER is useful when you combine data from several sources and want labels to be consistent throughout the file. Use @PROPER in a database to ensure consistent capitalization of names before sorting the names or before using the names to create mailing labels.

If you selected ASCII sort order in Install, capitalization affects the <u>sort</u> order of labels when you use <u>Range Sort</u> or <u>Query Sort</u>; two otherwise identical labels may not appear together if their capitalization is different.

**Examples**

@PROPER(A7&"; "&G7) returns Morton Smith; Athens, Georgia if A7 contains the label MORTON SMITH, and G7 contains the label athens, georgia. Note that the ; (semicolon) is in quotation marks and is therefore treated as a literal text instead of an argument separator.

**Similar @Functions**

<u>@LOWER</u> converts all letters in *text* to lowercase. <u>@UPPER</u> converts all letters in *text* to uppercase.

---

**See also**

**Help**

    <u>@Functions</u>

***User's Guide***

    Chapter 11, "Calculating with @Functions"

**@PV**
**@PVAL**

@PV(*payments;interest;term*) calculates the present value of an investment, based on a series of equal *payments*, discounted at a periodic *interest* rate over the number of periods in *term*.

@PVAL(*payments;interest;term*;[*type*];[*future-value*]) calculates the present value of an investment with a specified *future-value*, based on a series of equal *payments*, discounted at a periodic *interest* rate over the number of periods in *term*. @PVAL calculates for either an ordinary annuity or an annuity due, depending on the value you specify for *type*.

**Arguments**

*payments* and *term* are values.

*interest* is a decimal or percentage value greater than -1.

*type* is an optional argument that specifies whether to calculate for an ordinary annuity or for an annuity due.

| *type* | **1-2-3 calculates for** |
| --- | --- |
| 0 | Ordinary annuity (payments due at the end of a period); default if you omit the argument |
| 1 | Annuity due (payment due at the beginning of a period) |

*future-value* is an optional argument that specifies the future value of the series of payments. *future-value* can be any value. If you omit the *future-value* argument, 1-2-3 uses 0.

You cannot use an optional argument without using the ones that precede it.

**Notes**

The period used to calculate *interest* must be the same period used for *term*; for example, if you are calculating a monthly payment, enter the interest and term in monthly increments. Usually, this means you must divide the interest rate and the number of years in *term* by 12.

Use @PV to evaluate an investment or to compare one investment with others. @PV is useful in comparing different types of investments, for example, comparing a single-payment investment from a pension fund with a series of periodic payments. Use @PV with @PMT to create an amortization table.

@PV complements @PMT:@PV tells you how large a loan you can take out, given the constraint of the size of the monthly payment you can afford. Conversely, @PMT tells you how large your monthly payment will be, given the constraint of the size of the loan you want to take out.

**Examples**

You won $1,000,000. You can receive either 20 annual payments of $50,000 at the end of each year or a single payment of $400,000 instead of the $1,000,000 annuity. You want to find out which option is worth more in today's dollars.

If you were to accept the annual payments of $50,000, you assume that you would invest the money at a rate of 8%, compounded annually.

@PV(50000;0.08;20) returns $490,907, which tells you that the $1,000,000 paid over 20 years is worth $490,907 in today's dollars.

If you receive the payments at the beginning of each year:

@PVAL(50000;0.08;20;1) = $530,180

**Similar @Functions**

@FV and @FVAL calculate the future value of an investment based on a series of equal payments. @NPV computes the net present value of an investment, discounting future value to present value. @PMT and @PAYMT calculate the payment on a loan at a given interest rate for a specified number of payment periods.

---

**See also**

**Help**

　　@Functions

*User's Guide*

　　Chapter 11, "Calculating with @Functions"

**@QUOTIENT**

@QUOTIENT(*x*;*y*) calculates the result of *x*/*y*, truncated to an integer.

**Arguments**

*x* is a value. If *x* is 0, @QUOTIENT returns 0.

*y* is a value other than 0.

**Examples**

@QUOTIENT(7;3) = 2

@QUOTIENT(12.25;3.5) = 3

@QUOTIENT(-7;3) = -2

**Similar @functions**

@MOD calculates the remainder (modulus) of *x*/*y*.

**See also**

**Help**

@Functions

*User's Guide*

Chapter 11, "Calculating with @Functions"

**@RADTODEG**

@RADTODEG(*radians*) converts *radians* to degrees.

**Arguments**

*radians* is a value.

**Examples**

@RADTODEG(0.523599) = 30 degrees

**Similar @Functions**

@DEGTORAD converts degrees to radians.

---

**See also**

**Help**

  @Functions

***User's Guide***

  Chapter 11, "Calculating with @Functions"

**@RAND**

@RAND generates a random value between 0 and 1. 1-2-3 calculates @RAND to 17 decimal places. Each time 1-2-3 recalculates your work, @RAND generates a new random value.

**Notes**

To convert the value generated by @RAND to a fixed value, press F2 (EDIT) and then F9 (CALC).

To generate random values in different numeric intervals, multiply @RAND by the size of the interval. Use @ROUND or @INT with the result to create random whole numbers.

**Examples**

@RAND = 0.419501, or any value between 0 and 1.

@RAND*10 = 6.933674, or any value between 0 and 10.

@INT(@RAND*50)+1 = 49, or any integer between 1 and 50.

---

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

## @RANK

@RANK(*item*;*range*;[*order*]) calculates the relative size or position of a value in a range relative to other values in the range.

## Arguments

*item* is the value whose rank you want to determine.

*range* is the address or name of a range that contains values. *range* must include *item*.

*order* is an optional argument that specifies how to rank *item*. *order* is one of the following values:

| order | 1-2-3 treats values in *range* as if they are sorted in |
| --- | --- |
| 0 | Descending order (9 to 1) before ranking *item*; default if you omit the argument |
| 1 | Ascending order (1 to 9) before ranking *item* |

## Notes

1-2-3 assigns duplicate numbers in *range* the same rank. Duplicate numbers affect the rank of subsequent numbers in *range*. For example, for the values 2, 4, 6, 8, 8, 10, 12, the number 8 appears twice and has an ascending rank of 4 The number 10 has an ascending rank of 6; none of the numbers has a rank of 5.

## Examples

The range named SALES (A1..A5) contains the following values:

$5,000
$4,900
$5,150
$4,800
$4,900

@RANK(4900;SALES) = 3; $4,900 is the third highest value in the range SALES. No value would have the rank of 4.

@RANK(4900;SALES;1) = 2; because SALES is sorted in ascending order, $4,900 is the second lowest value in the range SALES. No value would have the rank of 3.

## See also

## Help

@Functions

## *User's Guide*

Chapter 11, "Calculating with @Functions"

**@RATE**

@RATE(*future-value;present-value;term*) calculates the periodic interest rate necessary for an investment (*present-value*) to grow to a *future-value* over the number of compounding periods in *term*.

**Arguments**

*future-value*, *present-value*, and *term* are values.

**Examples**

You invested $10,000 in a bond that matures in five years and has a maturity value of $18,000. Interest is compounded monthly. You want to determine the periodic interest rate for this investment.

@RATE(18000;10000;60) returns 0.984%, the periodic (monthly) interest rate. To determine the annual interest rate, use the formula ((1+@RATE(18000;10000;60))^12)-1. This yields an annual interest rate of 12.47%.

**Similar @functions**

@IRATE calculates the periodic interest rate necessary for an annuity to grow to a future value.

_____

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@REFCONVERT**

@REFCONVERT(*reference*) converts the 1-2-3 column or worksheet letters A through IV to numbers from 1 through 256, and numbers from 1 through 256 to their corresponding column or worksheet letters.

**Arguments**

*reference* specifies a 1-2-3 column or worksheet and can be either a letter from A through IV entered as text, or an integer from 1 through 256.

@REFCONVERT is not case-sensitive; you can enter *reference* as either uppercase or lowercase letters.

**Examples**

@REFCONVERT(10) = J

@REFCONVERT("J") = 10

**Similar @functions**

@COLS counts the columns in a range and @SHEETS counts the worksheets in a range. @COORD creates a cell address from values you specify.

---

**See also**

**Help**

    @Functions

***User's Guide***

    Chapter 11, "Calculating with @Functions"

**@REGRESSION**

@REGRESSION(*x-range*;*y-range*;*attribute*;[*compute*]) performs multiple linear regression and returns the specified statistic.

**Arguments**

*x-range* contains the independent variables. *x-range* is the name or address of a range that can contain up to 75 columns and 8,192 rows.

*y-range* contains the set of values for the dependent variable. *y-range* is the name or address of a single-column range with the same number of rows as *x-range*.

*attribute* specifies which regression output value to calculate. *attribute* is one of the following values:

| *attribute* | **1-2-3 calculates** |
| --- | --- |
| 1 | Constant |
| 2 | Standard error of Y estimate |
| 3 | R squared |
| 4 | Number of observations |
| 5 | Degrees of freedom |
| 101 to175 | X coefficient (slope) for the independent variable specified by *attribute* |
| 201 to 275 | Standard error of coefficient for the independent variable specified by *attribute* |

For the last two attributes, 1-2-3 numbers the independent variables in *x-range*, starting with the number 1, from top to bottom in a column and from left to right.

For example, if *x-range* is B2..D7, use the attribute 201 to find the standard error of coefficient for the independent variable in column B; use the attribute 102 to find the X-coefficient for the independent variable in column C.

*compute* is an optional argument that specifies the Y intercept.

| *compute* | **1-2-3** |
| --- | --- |
| 0 | Uses 0 as the Y intercept |
| 1 | Calculates the Y intercept; default if you omit the argument |

**Notes**

For the same data, @REGRESSION and Range Analyze Regression return the same result.

**Examples**

@REGRESSION

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**Example: @REGRESSION**

You run an ice cream stand at a tourist location, and you want to predict roughly how many quarts of ice cream you'll sell in the next day. You think your sales depend on three key factors: the number of hours of sunshine, the midday temperature, and the number of buses in a nearby parking lot. You want to determine the correlation between these factors and your sales. You collect data for a six-day period and record your observations in a worksheet.

```
A ---------- A --------- B ---- C ------- D ----
1     Ice Cream Sales       Sun    Temp   Buses in Lot
2                 250        3      84         10
3                 545        5      91          7
4                 550        5      89          8
5                 450        6      85         10
6                 605        6      90         11
7                 615        7      88          9
```

@REGRESSION(B2..D7;A2..A7;3) = 0.977225

Because R Squared is very close to 1, you know that a strong correlation exists between ice cream sales, the weather, and the number of buses.

**@REPEAT**

@REPEAT(*text;n*) duplicates *text* the number of times specified by *n*.

**Arguments**

*text* can be text enclosed in " " (quotation marks), a formula that results in text, or the address or name of a cell that contains a label or a formula that results in a label.

*n* can be any positive integer.

**Notes**

@REPEAT duplicates the text as many times as you specify; it is not limited by the current column width. This differs from using the repeating label-prefix character \ (backslash), which repeats a label only as many times as will fill the current column.

**Examples**

@REPEAT("Hello ";3) returns Hello Hello Hello.

@REPEAT("-";10) returns ----------.

_____

**See also**

**Help**

   @Functions

*User's Guide*

   Chapter 11, "Calculating with @Functions"

**@REPLACE**

@REPLACE(*original-text;start-number;n;new-text*) replaces *n* characters in *original-text* with *new-text*, beginning at *start-number*.

**Arguments**

*original-text* and *new-text* can be text enclosed in " " (quotation marks), formulas that result in text, or the addresses or names of cells that contain labels or formulas that result in labels.

*start-number* is the offset number of a character in *original-text*. It can be any positive value or 0. If *start-number* is greater than the length of *original-text*, @REPLACE appends *new-text* to *original-text*.

*n* can be any positive integer or 0. If *n* is 0, @REPLACE inserts *new-text* at *start-number* without deleting any characters in *original-text*.

**Notes**

@REPLACE counts punctuation and spaces as characters. If you use @REPLACE to append or insert text, remember to include the necessary spaces.

Use @FIND with @REPLACE to search for and replace a label or to calculate an unknown *start-number*.

@REPLACE is useful when you need to replace one set of characters with another, for example, to change the area code in a database of telephone numbers.

**Examples**

@REPLACE(CELL;@FIND("-";CELL;0),1,"/") copies the label in Cell, 4-24, as 4/24.

---

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@RIGHT**

@RIGHT(*text;n*) returns the last *n* characters in *text*.

**Arguments**

*text* can be text enclosed in " " (quotation marks), a formula that results in text, or the address or name of a cell that contains a label or a formula that results in a label.

*n* can be a positive integer or 0. If *n* is 0, the result is an <u>empty string</u>. If *n* is greater than the length of *text*, @RIGHT returns all of *text*.

**Notes**

@RIGHT counts punctuation and spaces as characters.

@RIGHT is useful for copying only part of a label into another cell (for example, for extracting last names from labels that include both first and last names).

In a macro, @RIGHT can extract parts of labels the user enters to store them in a database, for subroutine calls, or to alter the macro itself.

Use @RIGHT with <u>@FIND</u> when you do not know the exact value for *n,* or when *n* may vary.

**Examples**

@RIGHT(B3;5) = Sales, if B3 contains the label January Sales.

**Similar @Functions**

<u>@LEFT</u> returns the first *n* characters in *text*. <u>@MID</u> returns characters from within *text*.

---

**See also**

**Help**

   <u>@Functions</u>

***User's Guide***

   Chapter 11, "Calculating with @Functions"

**@ROUND**
**@ROUNDDOWN**
**@ROUNDUP**

@ROUND(*x;n*) rounds the value *x* to the nearest multiple of the power of 10 specified by *n*.

@ROUNDDOWN(*x;*[*n*]*;*[*direction*]) rounds the value *x* down to the nearest multiple of the power of 10 specified by *n*.

@ROUNDUP(*x;*[*n*]*;*[*direction*]) rounds the value *x* up to the nearest multiple of the power of 10 specified by *n*.

### Arguments

*x* is a value.

*n* is a value from -100 through 100. For @ROUNDDOWN and @ROUNDUP, if you omit the *n* argument, 1-2-3 uses 0.

| If *n* is | @ROUND |
|---|---|
| Positive | Affects the decimal portion of the number (moving right from the decimal point). For example, if *n* is 2, 1-2-3 rounds *x* to the nearest hundredth. |
| Negative | Affects the integer portion of the number (moving left from the decimal point). For example, if *n* is -2, 1-2-3 rounds *x* to the nearest hundred. |
| 0 | Rounds to the nearest integer. |

*direction* is an optional argument that specifies how to round negative values. *direction* can be 0 or 1.

- For @ROUNDUP: If *direction* is 0, 1-2-3 rounds negative values up; if *direction* is 1, 1-2-3 rounds negative values down.

- For @ROUNDDOWN: If *direction* is 0, 1-2-3 rounds negative values down; if *direction* is 1, 1-2-3 rounds negative values up.

If you omit *direction*, 1-2-3 uses 0. If *x* is positive, *direction* has no effect.

### Notes

Select Fixed from Style Number Format or Style Worksheet Defaults Number Format to display values with a specified number of decimal places if you want 1-2-3 to calculate the values to their full precision; do not use @ROUND.

### Examples

@ROUND(134.578;2) = 134.58
@ROUND(134.578;0) = 135
@ROUND(134.578;-2) = 100

@ROUNDDOWN(134.578;2) = 134.57
@ROUNDDOWN(134.578;0) = 134
@ROUNDDOWN(134.578;-2) = 100

@ROUNDUP(134.578;2) = 134.58
@ROUNDUP(134.578;0) = 135
@ROUNDUP(134.578;-2) = 200

### Similar @Functions

@ROUNDM rounds a value to a specified multiple. @EVEN rounds a value away from 0 to the nearest

even integer. @ODD rounds a value away from 0 to the nearest odd integer. @INT truncates a value, discarding the decimal portion. @TRUNC truncates a value to a specified decimal place.

**See also**

**Help**

@Functions

*User's Guide*

Chapter 11, "Calculating with @Functions"

**@ROUNDM**

@ROUNDM(*x*;*multiple*;[*direction*]) rounds the value *x* to the nearest *multiple*.

**Arguments**

*x* and *multiple* are any values that have the same sign.

*direction* is an optional argument that specifies whether to round *x* up or down.

| *direction* | **1-2-3 rounds *x*** |
| --- | --- |
| 1 | Up |
| 0 | To the nearest *multiple*; default if you omit the argument |
| -1 | Down |

**Examples**

@ROUNDM(25.37;0.05,1) = 25.40

@ROUNDM(25.37,.05,-1) = 25.35

**Similar @functions**

@INT truncates a value, discarding the decimal portion. @ROUND, @ROUNDDOWN, and @ROUNDUP round a value to a specified number of decimal places. @EVEN rounds a value away from 0 to the nearest even integer. @ODD rounds a value away from 0 to the nearest odd integer. @TRUNC truncates a value to a specified decimal place.

**See also**

**Help**

   @Functions

***User's Guide***

   Chapter 11, "Calculating with @Functions"

**@ROWS**

@ROWS(*range*) counts the number of rows in *range*.

**Arguments**

*range* is a range address or range name.

**Notes**

Use @ROWS with {FOR} in a macro that repeats the same action on a series of rows to determine when the macro should stop.

**Examples**

@ROWS(A3..B7) = 5 (rows 3 through 7).

@ROWS(SCORES) = 43, if SCORES is the range B3..B45.

**Similar @Functions**

@COLS counts the columns, and @SHEETS counts the worksheets, in a range.

---

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@S**

@S(*range*) returns the entry in the first cell in *range* as a label.

**Arguments**

*range* is a cell address or range name.

**Notes**

@S is useful with any text @function or <u>text formula</u> when a cell may contain a value and the entry must be a label (for example, a cell that contains a ZIP code). Use @S to prevent text formulas from resulting in <u>ERR</u>, for example, +A1&A2 returns ERR if either cell contains a value.

**Examples**

In the macro instructions

{IF @S(B6)=""}{BEEP}{INDICATE "ENTRY MUST BE A LABEL"}

@S returns an <u>empty string</u> if B6 contains a value or is a blank cell; 123 then beeps and changes the mode indicator to ENTRY MUST BE A LABEL.

**Similar @functions**

<u>@N</u> returns the entry in the first cell of *range* as a value. <u>@ISSTRING</u> determines whether a cell contains a label.

---

**See also**

**Help**

> <u>@Functions</u>

***User's Guide***

> Chapter 11, "Calculating with @Functions"

**@SEC**

@SEC(*x*) calculates the secant of angle *x*. The secant is the ratio of the hypotenuse to the side adjacent to an acute angle of a right triangle. Secant is the reciprocal of cosine.

**Arguments**

*x* is an angle measured in <u>radians</u>. *x* is any value from $-2^{32}*\pi$ through $2^{32}*\pi$.

**Examples**

@SEC(@DEGTORAD(30)) = 1.154701, the secant of a 30-degree angle.

**Similar @functions**

<u>@ASEC</u> calculates the arc secant of a value. <u>@ASECH</u> calculates the arc hyperbolic secant of a value. <u>@COS</u> calculates the cosine of an angle. <u>@SECH</u> calculates the hyperbolic secant of a value.

---

**See also**

**Help**

<u>@DEGTORAD</u> to convert from degrees to radians
<u>@Functions</u>

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@SECH**

@SECH(*x*) calculates the hyperbolic secant of angle *x*. The hyperbolic secant is the reciprocal of the hyperbolic cosine. The result of @SECH is a value greater than 0 or less than or equal to 1.

**Arguments**

*x* is a value from approximately -1135.571 to approximately 1135.571.

**Examples**

@SECH(@DEGTORAD(30)) = 0.87701

**Similar @functions**

@ASECH calculates the arc hyperbolic secant of a value. @SEC calculates the secant of a value.

**See also**

**Help**

@DEGTORAD to convert from degrees to radians
@Functions

*User's Guide*

Chapter 11, "Calculating with @Functions"

**@SECOND**

@SECOND(*time-number*) extracts the seconds, an integer from 0 through 59, from *time-number*.

**Arguments**

*time-number* is a value from .000000 (midnight) through .999988 (11:59:59 PM).

**Examples**

@SECOND(0.333) = 31

@SECOND(@TIME(11;15;45)) = 45, because 45 is the *seconds* argument for @TIME(11;15;45).

**Similar @functions**

@HOUR extracts the hour, and @MINUTE extracts the minutes, from a time number.

_____

**See also**

**Help**

   @Functions

*User's Guide*

   Chapter 11, "Calculating with @Functions"

**@SERIESSUM**

@SERIESSUM(*x*;*n*;*m*;*coefficients*) calculates the sum of a power series.

**Arguments**

*x* is the power series' input value.

*n* is the initial power to which to raise *x*.

*m* is the increment by which to increase *n* for each term in the series.

*x*, *n*, and *m* are values.

*coefficients* is a range that contains the coefficients by which 123 multiplies each successive power of *x*. The number of cells in *coefficients* determines the number of terms in the series. For example, if *coefficients* contains ten cells, the power series contains ten terms.

**Examples**

The range DATA contains these coefficients: 0.2, 0.7, 1.3.

@SERIESSUM(3.5;2;1;DATA) = 227.5438

**See also**

**Help**

@Functions

*User's Guide*

Chapter 11, "Calculating with @Functions"

**@SHEETS**

@SHEETS(*range*) counts the number of worksheets in *range*.

**Arguments**

*range* is a range address or range name.

**Notes**

Use @SHEETS with {FOR} in a macro that repeats the same action in a series of worksheets to determine when the macro should stop.

**Examples**

@SHEETS(Q_2) = 4 if Q_2 is the range B:B3..E:C45 (worksheets B; C; D; and E).

**Similar @functions**

@COLS counts the columns, and @ROWS counts the rows, in a range. @REFCONVERT converts the 123 column or worksheet letters A through IV to numbers from 1 through 256.

---

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@SIGN**

@SIGN(*x*) returns 1 if *x* is a positive value, 0 if *x* is 0, and -1 if *x* is a negative value.

**Arguments**

*x* is any value.

**Examples**

@SIGN(15) = 1

@SIGN(15*0) = 0

@SIGN(-15) = -1

---

**See also**

**Help**

@Functions

*User's Guide*

Chapter 11, "Calculating with @Functions"

**@SIN**

@SIN(*x*) calculates the sine of angle *x*. The sine is the ratio of the side opposite an acute angle of a right triangle to the hypotenuse.

**Arguments**

*x* is an angle measured in <u>radians</u>. *x* can be any value from -2^32*π through 2^32*π.

**Examples**

@SIN(@DEGTORAD(30)) = 0.5, the sine of a 30-degree angle.

**Similar @functions**

<u>@ASIN</u> calculates the arc sine of a value. <u>@SINH</u> calculates the hyperbolic sine of an angle.

---

**See also**

**Help**

    <u>@DEGTORAD</u> to convert degrees to radians
    <u>@Functions</u>

***User's Guide***

    Chapter 11, "Calculating with @Functions"

**@SINH**

@SINH(*x*) calculates the hyperbolic sine of angle *x*.

**Arguments**

*x* is a value from approximately -1135.571 to approximately 1135.571.

**Examples**

@SINH(@DEGTORAD(30)) = 0.547853

**Similar @functions**

@ASINH calculates the arc hyperbolic sine of a value. @SIN calculates the sine of an angle.

_____

**See also**

**Help**

@DEGTORAD to convert degrees to radians
@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@SKEWNESS**

@SKEWNESS(*range*;[*type*]) calculates the skewness of the values in *range*.

**Arguments**

*range* is the name or address of a range that contains values. If *range* contains fewer than three values, @SKEWNESS returns ERR.

*type* is an optional argument that specifies whether to calculate the population or sample skewness.

| *type* | **1-2-3 calculates** |
| --- | --- |
| 0 | Population skewness; default if you omit the argument |
| 1 | Sample skewness |

**Notes**

Skewness measures the symmetry of a distribution around its mean. Positive skewness indicates a drawn-out tail to the left; negative skewness indicates a drawn-out tail to the right.

**Examples**

The range DATA contains these values: 2, 5, 6, 9, 1, 2, 4.

@SKEWNESS(DATA) = 0.584816

**Similar @functions**

@KURTOSIS calculates the kurtosis of the values in a list. @STD and @PURESTD calculate population standard deviation. @VAR and @PUREVAR calculate population variance.

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@SLN**

@SLN(*cost;salvage;life*) calculates the straight-line depreciation allowance of an asset with an initial value of *cost*, an expected useful *life*, and a final value of *salvage*, for one period.

**Arguments**

*cost* is the amount paid for the asset. *cost* can be any value.

*salvage* is the value of the asset at the end of its life. *salvage* can be any value.

*life* is the number of periods the asset takes to depreciate to its salvage value. *life* can be any value except 0.

**Notes**

Straight-line depreciation divides the depreciable cost (the actual cost minus the salvage value) equally into each period of the useful life of the asset. The useful life is the number of periods (typically years) over which the asset is depreciated.

**Examples**

You have an office machine that cost $10,000. The useful life of this machine is 10 years, and the salvage value in 10 years will be $1,200. You want to calculate yearly depreciation expense, using the straight-line method.

@SLN(10000;1200;10) = $880

**Similar @functions**

@DB calculates depreciation using the declining balance method. @DDB calculates depreciation using the double-declining balance method, @VDB calculates depreciation using the variable-rate declining balance method, and @SYD calculates depreciation using the sum-of-the-years'-digits method.

---

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@SMALL**

@SMALL(*range*;n) finds the *n*th smallest value in *range*.

**Arguments**

*range* is the name or address of a range that contains values.

*n* is any positive integer. If *n* is larger than the number of values in *range*, @SMALL returns <u>NA</u>.

**Examples**

A range named SCORES contains these test scores: 87, 85, 90, 80, 82, 92, 79, 85, 95, 86.

@SMALL(SCORES;3) returns 82, the third-lowest score.

**Similar @functions**

<u>@LARGE</u> finds the *n*th largest value in a range.    <u>@MAX</u> and <u>@PUREMAX</u> find the largest value in a range. <u>@MIN</u> and <u>@PUREMIN</u> find the smallest value in a range.

**See also**

**Help**

<u>@Functions</u>

*User's Guide*

Chapter 11, "Calculating with @Functions"

**@SOLVER**

@SOLVER(*attribute*) returns a value that gives information about the status of Solver.

**Arguments**

*attribute* is any of the words from the table below, as text.

| attribute | Returns |
|---|---|
| consistent | 1 if all constraints are satisfied |
| | 2 if at least one constraint is not satisfied |
| | ERR if Solver is not active or there is no answer in the worksheet |
| done | 1 if Solver has finished solving |
| | 2 if Solver is currently solving |
| | 3 if Solver is active, but hasn't begun solving |
| | ERR if Solver is not active |
| moreanswers | 1 if Solver has found all the answers |
| | 2 if Solver might be able to find more answers |
| | ERR if Solver is not active |
| needguess | 1 if Solver needs no guesses to find an answer |
| | 2 if Solver needs guesses to find an answer |
| | ERR if Solver is not active or there is no answer in the worksheet |
| numanswers | *n*, where *n* is the number of answers or attempts Solver found |
| | ERR if Solver is not active or has not solved the problem yet |
| optimal | 1 for the optimal answer |
| | 2 for the best answer found |
| | 3 if the problem is unbounded |
| | 4 if no optimization requested or no answer found |
| | ERR if Solver is not active |
| progress | *n*, where *n* is the percentage of solving completed |
| | ERR if Solver is not active or solving has not begun |
| result | 1 if Solver found one or more answers |
| | 2 if no answers found but representative attempts are available |
| | ERR if Solver is not active or has not solved the problem yet |

**Notes**

@SOLVER is not recalculated whenever 1-2-3 performs a recalculation. Use {RECALC} or {RECALCCOL} to recalculate cells containing @SOLVER.

**Examples**

The following lines are an excerpt from a longer macro that runs Solver. The macro uses @SOLVER to determine if Solver has found all answers.

If it has found all answers, the macro enters the first answer in the worksheet and ends processing. If it hasn't found all answers, the macro presses the Solve button to continue solving and then repeats {IF} to see if more answers can be found.

The loop continues until Solver finishes finding all answers. When Solver has found all answers, subroutine {A} puts the first answer in the worksheet, and {RETURN} ends the macro.

```
CHKSLV          {IF @SOLVER("moreanswers")=1}{A}{RETURN}
                {SOLVER-ANSWER "solve"}
                {BRANCH CHKSLV}
```

---

**See also**

**Help**

    @Functions

***User's Guide***

    Chapter 11, "Calculating with @Functions"

**@SQRT**

@SQRT(*x*) returns the positive square root of *x*.

**Arguments**

*x* is a positive value.

**Examples**

@SQRT(B3) = 10, if B3 contains the value 100.

@SQRT(-2) = ERR, because *x* is negative.

**See also**

**Help**

@Functions

*User's Guide*

Chapter 11, "Calculating with @Functions"

**@SQRTPI**

@SQRTPI($x$) calculates the square root of $x*\pi$.

**Arguments**

$x$ is any positive value or 0.

**Examples**

@SQRTPI(0.5) = 1.253314

@SQRTPI(2) = 2.506628

**Similar @functions**

@SQRT calculates the positive square root of a value. @PI produces the value $\pi$.

---

**See also**

**Help**

@Functions

*User's Guide*

Chapter 11, "Calculating with @Functions"

**@STD**
**@STDS**
**@PURESTD**
**@PURESTDS**

@STD(*list*) calculates the population standard deviation of the values in *list*.

@STDS(*list*) calculates the sample standard deviation of the values in *list*.

@PURESTD(*list*) calculates the population standard deviation of the values in *list*, ignoring cells that contain labels.

@PURESTDS(*list*) calculates the sample standard deviation of the values in *list*, ignoring cells that contain labels.

## Notes

@STD and @PURESTD use the n, or population, method to calculate standard deviation of population data. The n method assumes that the selected values are the entire population. If the values are only a sample of the population, the standard deviation is biased because of errors introduced in taking the sample.

@STDS and @PURESTDS use the n-1, or sample, method to calculate standard deviation of sample population data. The n-1 method makes the standard deviation slightly larger than the n method to compensate for errors in the sample. A larger standard deviation is unbiased by sampling errors, and thus tends to be more accurate.

Standard deviation is the square root of the variance of all individual values from the mean.

## Arguments

*list* can contain any of the following, in any combination: numbers, numeric formulas, and range addresses or range names that contain numbers or formulas. Separate elements of *list* with argument separators.

## Examples

@STD and @STDS

## Similar @functions

@DSTD and @DSTDS calculate the standard deviation of the values in a field of a database table. @VAR and @PUREVAR calculate the population variance of values in a list. @VARS and @PUREVARS calculate the sample variance of values in a list.

---

**See also**

**Help**

@Functions
Statistical @Function Arguments

***User's Guide***

Chapter 11, "Calculating with @Functions"

**Example: @STD and @STDS**

This table lists the heights and weights of ten randomly selected test subjects. You want to determine the standard deviation of their heights.

@STD(A2..A11) = 5.793483

Assume the subjects represent a randomly selected sample of a larger group of test subjects.

@STDS(A2..A11) = 6.106868

```
A --------- A --------- B --
1       HEIGHT (cm)    WEIGHT (kg)
2          190.50        72.73
3          187.96        86.36
4          175.26        68.18
5          175.26        76.37
6          180.34        77.27
7          180.34        72.73
8          187.96        75.00
9          172.72        68.18
10         177.80        70.46
11         179.07        86.36
```

**@STRING**

@STRING(*x;n*) converts the <u>value</u> *x* to a <u>label</u> using the format specified by *n*.

**Arguments**

*x* is a value.

*n* is an integer from the following table:

| If *n* is | @STRING returns a label formatted as |
|---|---|
| 0 through 116 | <u>Fixed</u>, with *n* decimal places |
| 1000 through 1116 | <u>Comma</u>, with *n*-1000 decimal places |
| -18 through -1 | <u>Scientific</u>, with @ABS(*n*) digits |
| 10001 through 10512 | <u>General</u>, up to *n*-10000 characters |

**Notes**

@STRING ignores any formatting characters 1-2-3 uses to display the value *x*. This includes all currency and other numeric formatting symbols, whether you enter them or 1-2-3 creates them after you select a format from <u>Style Number Format</u> or <u>Style Worksheet Defaults Number Format</u>. For example, if cell A7 contains the formatted value $45.23, @STRING(A7;2) returns the label 45.23.

**Examples**

@STRING(203;3) = the label 203.000

@STRING(1.23587;0) = the label 1

@STRING(20500;1002) = the label 20,500.00

@STRING(@PI;-5) = the label 3.1416E+00

@STRING(123456.789;10008) = the label 1234567.8

**Similar @functions**

<u>@VALUE</u> converts a number entered as text to its corresponding value.

---

**See also**

**Help**

   <u>@Functions</u>

***User's Guide***

   Chapter 11, "Calculating with @Functions"

## @SUBTOTAL

@SUBTOTAL(*list*) adds the values in *list*. Use @SUBTOTAL to indicate which cells @GRANDTOTAL should sum.

### Arguments

*list* can contain any of the following, in any combination: numbers, numeric formulas, and range addresses or range names that contain numbers or formulas. Separate elements of *list* with argument separators.

### Examples

@SUBTOTAL(B5..B9) returns the sum of the values in B5..B9.

@SUBTOTAL(SALES;M25..R25) returns the sum of the values in the range SALES and the range M25..R25.

### Similar @functions

@SUM adds the values in a list. @SUMNEGATIVE sums only the negative values in a list. @SUMPOSITIVE sums only the positive values in a list.

### See also

### Help

@Functions
Statistical @Function Arguments

### *User's Guide*

Chapter 11, "Calculating with @Functions"

**@SUM**

@SUM(*list*) adds the values in *list* .

**Arguments**

*list* can contain any of the following, in any combination: numbers, numeric formulas, and range addresses or range names that contain numbers or formulas. Separate elements of *list* with argument separators.

**Examples**

@SUM(B5..B9) returns the sum of the values in B5..B9.

@SUM(SALES;M25..R25) returns the sum of the values in the range SALES and the range M25..R25.

**Related SmartIcons**

Sums values in the highlighted or adjacent range, if you include empty cells below or to the right of the range.

**Similar @functions**

@DSUM calculates the sum of values that meet criteria you specify. @SUBTOTAL adds the values in a list and indicates which values @GRANDTOTAL should sum.   @SUMNEGATIVE sums only the negative values in a list. @SUMPOSITIVE sums only the positive values in a list.

---

**See also**

**Help**

> @Functions
> Statistical @Function Arguments

***User's Guide***

> Chapter 11, "Calculating with @Functions"

**@SUMPRODUCT**

@SUMPRODUCT(*list*) multiplies the values in corresponding cells in multiple ranges and then sums the products.

**Arguments**

*list* can be any combination of ranges that contain values and are the same size and shape. If the ranges in *list* are not the same size and shape, @SUMPRODUCT returns ERR.

**Notes**

If the ranges in *list* are columns, @SUMPRODUCT multiplies by rows. If the ranges in *list* are rows, @SUMPRODUCT multiplies by columns. If each range in *list* spans more than one column, @SUMPRODUCT multiplies by rows.

**Examples**

This example, taken from a real estate database table, uses @SUMPRODUCT to calculate the total commissions due to agents on house sales in February.

*list* contains two ranges: SOLD (D4..D6) contains the prices paid for three houses, and COMM (E4..E6) contains the agent's commission percentage of the sale price:

| SOLD | COMM |
| --- | --- |
| $25,000 | 0.04 |
| $34,580 | 0.05 |
| $77,325 | 0.04 |

@SUMPRODUCT(SOLD;COMM) = $5,822, the total commissions ($1;000 + $1;729 + $3;093) due to agents on the sale of three houses.

**Similar @functions**

@SUMSQ calculates the sum of the squares of the values in a list. SUMXMY2 calculates the sum of the squared difference of values in corresponding cells in two ranges.

---

**See also**

**Help**

> @Functions
> Statistical @Function Arguments

***User's Guide***

> Chapter 11, "Calculating with @Functions"

**@SUMSQ**

@SUMSQ(*list*) calculates the sum of the squares of the values in *list*.

**Arguments**

*list* can contain any of the following, in any combination: numbers, numeric formulas, and range addresses or range names that contain numbers or formulas. Separate elements of *list* with argument separators.

**Examples**

@SUMSQ(2;4;6) = 56

**Similar @functions**

@SUM adds the values in a list. @SUMPRODUCT multiplies the values in corresponding cells in multiple ranges and then sums the products.

---

**See also**

**Help**

@Functions
Statistical @Function Arguments

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@SUMXMY2**

@SUMXMY2(*range1*;*range2*) subtracts the values in *range2* from the corresponding cells in *range1*, squares the differences, and then sums the results.

**Arguments**

*range1* and *range2* are ranges that contain values and are the same size and shape. If *range1* and *range2* are not the same size and shape, @SUMX2MY2 returns ERR.

**Notes**

If *range1* and *range2* are single-column ranges, 1-2-3 subtracts by row. If *range1* and *range2* are multi-column ranges, 1-2-3 subtracts by columns.

**Examples**

In the following example, *range1* is named TUES and *range2* is named WED:

| TUES | WED |
|------|-----|
| 5 | 3 |
| 4 | 4 |
| 7 | 8 |

@SUMXMY2(TUES;WED) = 5

**Similar @functions**

@SUMPRODUCT calculates the sum of the products of the values in corresponding cells in multiple ranges. @SUMSQ calculates the sum of the squares of the values in a list.

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@SYD**

@SYD(*cost;salvage;life;period*) calculates the sum-of-the-years'-digits depreciation allowance of an asset with an initial value of *cost*, an expected useful *life*, and a final value of *salvage*, for a specified *period*.

**Arguments**

*cost* is the amount paid for the asset. *cost* can be any value.

*salvage* is the value of the asset at the end of its life. *salvage* can be any value.

*life* is the number of periods the asset takes to depreciate to its salvage value. *life* can be any value greater than or equal to 1.

*period* is the time for which you want to find the depreciation allowance. *period* can be any value greater than or equal to 1.

**Notes**

The sum-of-the-years'-digits method accelerates the rate of depreciation so that more depreciation expense occurs in earlier periods than in later ones (although not so much as when you use the double-declining balance method). The depreciable cost is the actual cost minus the salvage value.

Use @SYD when you need a higher depreciation expense early in the life of an asset, such as in preparing tax returns.

**Examples**

You have an office machine that cost $10,000. The useful life of the machine is 10 years, and the salvage value in 10 years will be $1,200. You want to calculate depreciation expense for the fifth year, using the sum-of-the-years'-digits method:

@SYD(10000;1200;10;5) = $960

**Similar @functions**

@DDB calculates depreciation using the double-declining balance method. @VDB uses the variable-rate declining balance method, @DB uses the fixed-declining balance method, and @SLN uses the straight-line method.

---

**See also**

**Help**

   @Functions

***User's Guide***

   Chapter 11, "Calculating with @Functions"

**@TAN**

@TAN(*x*) calculates the tangent of angle *x*. The tangent is the ratio of the side opposite an acute angle of a right triangle to the side adjacent the same acute angle.

**Arguments**

*x* is an angle measured in <u>radians</u>. *x* is a value from $-2\text{\textasciicircum}32*\pi$ to $2\text{\textasciicircum}32*\pi$.

**Examples**

@TAN(@DEGTORAD(35)) = 0.700208, the tangent of a 35-degree angle.

**Similar @functions**

<u>@ATAN</u> calculates the arc tangent of a value. <u>@TANH</u> calculates the hyperbolic tangent of an angle.

---

**See also**

**Help**

<u>@DEGTORAD</u> to convert from degrees to radians
<u>@Functions</u>

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@TANH**

@TANH(*x*) calculates the hyperbolic tangent of angle *x*. The hyperbolic tangent is the ratio of hyperbolic sine to the hyperbolic cosine. The result of @TANH is a value from -1 through 1.

**Arguments**

*x* is a value from approximately -11355.1371 to approximately 11355.1371.

**Examples**

@TANH(@DEGTORAD(30)) = 0.480473

**Similar @functions**

@ATANH calculates the arc hyperbolic tangent of a value. @TAN calculates the tangent of an angle.

**See also**

**Help**

@DEGTORAD to convert from degrees to radians
@Functions

*User's Guide*

Chapter 11, "Calculating with @Functions"

**@TDIST**

@TDIST(*x*;*degrees-freedom*;[*type*];[*tails*]) calculates the Student's *t*-distribution.

**Arguments**

The value you enter for *x* depends on the value you enter for *type*.

| If *type* is | *x* is |
| --- | --- |
| 0 | The critical value or upper bound for the value of the cumulative *t*-distribution random variable and is any value; default if you omit the argument |
| 1 | A probability and is a value from 0 to 1 |

*degrees-freedom* is the number of degrees of freedom for the sample. *degrees-freedom* is a positive integer.

*type* is an optional argument that specifies how 1-2-3 calculates @TDIST.

| *type* | 1-2-3 calculates |
| --- | --- |
| 0 | The significance level that corresponds to the critical value, *x*; default if you omit the argument |
| 1 | The critical value that corresponds to the significance level, *x* |

*tails* is an optional argument that specifies the direction of the *t*-test.

| *tails* | 1-2-3 performs |
| --- | --- |
| 1 | A one-tailed *t*-test |
| 2 | A two-tailed *t*-test; default if you omit the argument |

You cannot use an optional argument without using the ones that precede it.

**Notes**

@TDIST approximates the cumulative *t*-distribution to within $\pm$ 3*10^-7. If @TDIST cannot approximate the result to within 0.0000001 after 100 calculation iterations, the result is ERR.

The Student's *t*-distribution is the distribution of the ratio of a standardized normal distribution to the square root of the quotient of a chi-square distribution by the number of its degrees of freedom.

**Examples**

@TDIST(2.228;10) = 0.05

@TDIST(0.05;10;1) = 2.228

**Similar @functions**

@CHIDIST calculates the chi-square distribution. @FDIST calculates the *F*-distribution. @TTEST calculates the probability associated with a Student's *t*-test.

---

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

## @TERM
## @NPER

@TERM(*payments;interest;future-value*) calculates the number of periods required for a series of equal *payments* to accumulate a *future-value* at a periodic *interest* rate.

@NPER(*payments;interest;future-value*;[*type*];[*present-value*]) calculates the number of periods required for a series of equal *payments* with a specified *present-value* to accumulate a *future-value* at a periodic *interest* rate. @NPER calculates for either an ordinary annuity or an annuity due, depending on the value you specify for *type*.

### Arguments

*payments* is the value of the equal investments. *payments* can be any value except 0.

*interest* is the periodic interest rate. *interest* can be any value greater than -1.

*future-value* is the amount you want to accumulate. *future-value* can be any value.

*type* is an optional argument that specifies whether to calculate for an ordinary annuity or for an annuity due.

| *type* | 1-2-3 calculates for |
|--------|----------------------|
| 0 | Ordinary annuity (payments due at the end of a period); default if you omit the argument |
| 1 | Annuity due (payment due at the beginning of a period) |

*present-value* is an optional argument that specifies the present value of the series of future payments. *present-value* can be any value. If you omit the *present-value* argument, 1-2-3 uses 0.

You cannot use an optional argument without using the ones that precede it.

### Notes

You can calculate the term necessary to pay back a loan by using @TERM with a negative *future-value*. For example, you want to know how long it will take to pay back a $10,000 loan at 10% yearly interest, making payments of $1,174 per year. @ABS(@TERM(1174.6;0.1;-10000)) calculates 20 years to pay back the loan.

### Examples

You deposit $2,000 at the end of each year into a savings account. Your account earns 7.5% a year, compounded annually. You want to determine how long it will take to accumulate $100,000:

@TERM(2000;0.075;100000) = 21.5 years

If you make payments at the beginning of each year:

@NPER(2000;0.075;100000;1) = 20.76 years

### Similar @functions

@CTERM calculates the number of compounding periods for a single-deposit investment.

---

### See also

### Help

   @Functions

### *User's Guide*

Chapter 11, "Calculating with @Functions"

**@TIME**

@TIME(*hour;minutes;seconds*) calculates the <u>time number</u> for the specified *hour*, *minutes*, and *seconds*.

**Arguments**

*hour* is an integer from 0 (midnight) through 23 (11:00 PM).

*minutes* is an integer from 0 through 59.

*seconds* is an integer from 0 through 59.

**Notes**

<u>Style Number Format</u> and <u>Style Worksheet Defaults Number Format</u> make the time number appear as the time it represents.

**Examples**

The formula @TIME(13;0;0)-@TIME(9;15;0)*95*24 calculates the amount due to a consultant on a given day by subtracting the start time from the stop time and multiplying the result by an hourly rate of $95.00.

**Similar @functions**

<u>@TIMEVALUE</u> converts labels to time numbers.

---

**See also**

**Help**

   <u>@Functions</u>

***User's Guide***

   Chapter 11, "Calculating with @Functions"

**@TIMEVALUE**

@TIMEVALUE(*text*) calculates the <u>time number</u> for the time specified in *text*.

**Arguments**

*text* can be text enclosed in " " (quotation marks), a formula that results in text, or the address or name of a cell that contains a label or a formula that results in a label. *text* must be in one of the four 1-2-3 <u>Time formats</u>.

**Notes**

@TIMEVALUE is useful when you need to convert times entered as labels into time numbers for use in calculations. @TIMEVALUE is especially useful with data that has been imported from another program, such as a word processing program.

Use <u>Style Number Format</u> and <u>Style Worksheet Defaults Number Format</u> to make the time number appear as the time it represents.

**Examples**

@TIMEVALUE("08:19:27 AM") = 0.34684

@TIMEVALUE("08:19:27 AM") = 0.34684, formatted as 08:19:27 AM, if the cell is formatted as 11:59:59 AM/PM.

**Similar @functions**

<u>@TIME</u> calculates the time number when you specify the hour, minutes, and seconds.

---

**See also**

**Help**

<u>@Functions</u>

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@TODAY**

@TODAY calculates the <u>date number</u> that corresponds to the current date on your computer.

**Notes**

1-2-3 recalculates @TODAY each time you recalculate your work. If you use <u>Tools User Setup Recalculation Automatic</u>, 1-2-3 recalculates @TODAY whenever it recalculates another value.

Use <u>Style Number Format</u> and <u>Style Worksheet Defaults Number Format</u> to make the time number appear as the time it represents.

**Examples**

@TODAY = 31048 on January 01, 1985.

@TODAY = 33418 on June 29, 1991.

_____

**See also**

**Help**

    <u>@Functions</u>

***User's Guide***

    Chapter 11, "Calculating with @Functions"

**@TRIM**

@TRIM(*text*) removes leading, trailing, and consecutive space characters from *text*.

**Arguments**

*text* can be text enclosed in " " (quotation marks), a formula that results in text, or the address or name of a cell that contains a label or a formula that results in a label.

**Notes**

Use @TRIM to ensure that database entries do not contain unnecessary spaces that would affect sort order when you use Range Sort or Query Sort.

**Examples**

@TRIM(" 45   3/8") = 45 3/8, removing the leading space before 45 and one of the two spaces between 45 and 3/8.

@TRIM(" 500      South    St.") = 500 South St., removing the leading space before 500, two of the three spaces between 500 and South, and one of the two spaces between South and St.

**Similar @functions**

@SETSTRING returns text aligned within a specified number of spaces.

---

**See also**

**Help**

   @Functions

***User's Guide***

   Chapter 11, "Calculating with @Functions"

**@TRUE**

@TRUE returns the logical value 1 (true).

**Notes**

If a logical statement such as A1=B1 is true, its logical value is 1. If it is false, its logical value is 0.

Using @TRUE is the same as using the value 1 in formulas that evaluate logical conditions, but @TRUE makes the formula easier to understand.

**Examples**

@IF(A6>500;@TRUE;@FALSE) = 1 when cell A6 contains a value greater than 500.

**Similar @functions**

@FALSE returns the logical value 0.

---

**See also**

**Help**

@Functions

*User's Guide*

Chapter 11, "Calculating with @Functions"

## @TRUNC

@TRUNC(*x*;[*n*]) truncates *x* to the number of decimal places specified by *n*.

**Arguments**

*x* is a value.

*n* is an optional argument and is a value from -100 through 100.

| If *n* is | @TRUNC |
|---|---|
| Positive | Affects the decimal portion of the number (moving right from the decimal point). For example, if *n* is 2, 1-2-3 truncates *x* to the nearest hundredth. |
| Negative | Affects the integer portion of the number (moving left from the decimal point). For example, if *n* is -2, 1-2-3 truncates *x* to the nearest hundred. |
| 0 | Truncates *x* to the nearest integer; default if you omit the argument |

**Notes**

Select Fixed from Style Number Format or Style Worksheet Defaults Number Format to display values with a specified number of decimal places if you want 1-2-3 to calculate the values to their full precision; do not use @TRUNC.

**Examples**

@TRUNC(123.45) = 123

@TRUNC(-123.45) = -123

@TRUNC(123.45;-2) = 100

@TRUNC(123.45;1) = 123.4

@TRUNC(-123.45;-2) = -100

@TRUNC(-123.45;1) = -123.4

**Similar @functions**

@ROUND, @ROUNDDOWN, and @ROUNDUP round a value to a specified number of decimal places. @ROUNDM rounds a value to a specified multiple. @EVEN rounds a value away from 0 to the nearest even integer. @ODD rounds a value away from 0 to the nearest odd integer. @INT truncates a value, discarding the decimal portion.

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@UPPER**

@UPPER(*text*) converts all the letters in *text* to uppercase.

**Arguments**

*text* can be text enclosed in " " (quotation marks), a formula that results in text, or the address or name of a cell that contains a label or a formula that results in a label.

**Notes**

Capitalization affects the <u>sort</u> order of labels when you use <u>Range Sort</u> or <u>Query Sort</u>. Two otherwise identical labels may not appear together if their capitalization is different.

**Examples**

@UPPER("Account Number") = ACCOUNT NUMBER

@UPPER(B2) = WARNING, if B2 contains the label warning.

**Similar @functions**

<u>@LOWER</u> converts all letters in *text* to lowercase. <u>@PROPER</u> capitalizes only the first letter of each word in *text*.

---

**See also**

**Help**

<u>@Functions</u>

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@VALUE**

@VALUE(*text*) converts a number entered as a *text* to its corresponding value.

**Arguments**

*text* can be text in " " (quotation marks) or a label that contains only numbers. *text* can resemble a standard number (456.7), a number in scientific format (4.567E2), a mixed number (45 7/8), or a formatted number ($45.67).

**Notes**

@VALUE ignores leading and trailing spaces; however, @VALUE returns ERR when *text* contains spaces that separate symbols from the numbers (such as $ 32.85 or £   56.20).

@VALUE results in 0 when *text* is a blank cell or an empty string, and returns ERR when *text* contains non-numeric characters.

Press F2 (EDIT) and then press F9 (CALC) to replace @VALUE with its value.

You cannot calculate within a *text* argument in @VALUE, but you can create a formula with several @VALUE functions. For example, @VALUE("22"+"20") = 0, but @VALUE("22")+@VALUE("20") = 42.

**Examples**

@VALUE("543") = the value 543.

@VALUE(B3) = the value 49.75, if cell B3 contains the label 49 3/4.

@VALUE("85%") = the value .85.

**Similar @functions**

@STRING converts a value to a label.

---

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@VAR**
**@VARS**
**@PUREVAR**
**@PUREVARS**

@VAR(*list*) calculates the population variance in a *list* of values.

@VARS(*list*) calculates the sample population variance in a *list* of values.

@PUREVAR(*list*) calculates the population variance in a *list* of values, ignoring cells that contain labels.

@PUREVARS(*list*) calculates the sample population variance in a *list* of values, ignoring cells that contain labels.

## Arguments

*list* can contain any of the following, in any combination: numbers, numeric formulas, and range addresses or range names that contain numbers or formulas. Separate elements of *list* with argument separators.

## Notes

The variance @functions are useful when you need to carry out ANOVA (analysis of variance) statistical tests.

@VAR and @PUREVAR use the n, or population, method to calculate variance. The n method assumes the selected values are the entire population. If the values are only a sample of the population, the variance is biased because of errors introduced in taking a sample.

@VARS and @PUREVARS use the n-1, or sample, method to calculate variance. The n-1 method produces a variance that is slightly larger than the n method to compensate for errors in the sample. A larger variance is unbiased by sampling errors and thus tends to be more accurate.

## Examples

@VAR and @VARS

## Similar @functions

@DVAR and @DVARS calculate the population variance of values that meet criteria you specify.

---

## See also

## Help

@Functions
Statistical @Function Arguments

## *User's Guide*

Chapter 11, "Calculating with @Functions"

**Example: @VAR and @VARS**

This table lists the heights and weights of ten randomly selected test subjects. You want to determine the variation of their weights.

@VAR(B2..B11) = 38.77462

Assume the subjects represent a randomly selected sample of a larger group of test subjects.

@VARS(B2..B11) = 43.08292

```
A --------- A --------- B --
1       HEIGHT (cm)    WEIGHT (kg)
2          190.50        72.73
3          187.96        86.36
4          175.26        68.18
5          175.26        76.37
6          180.34        77.27
7          180.34        72.73
8          187.96        75.00
9          172.72        68.18
10         177.80        70.46
11         179.07        86.36
```

**@VDB**

@VDB(*cost;salvage;life;start-period;end-period;*[*depreciation-factor*];[*switch*]) calculates the depreciation allowance of an asset with an initial value of *cost*, an expected useful *life*, and a final value of *salvage* for a period specified by *start-period* and *end-period*, using the variable-rate declining balance method.

**Arguments**

*cost* is the amount paid for the asset. *cost* can be any value greater than *salvage*.

*salvage* is the value of the asset at the end of its life. *salvage* can be any value.

*life* is the number of periods the asset takes to depreciate to its salvage value. *life* can be any value greater than 0.

*start-period* is the point in the asset's life when you want to begin calculating depreciation. *start-period* can be any value greater than or equal to 0, but cannot be greater than *life*.

*end-period* is the point in the asset's life when you want to stop calculating depreciation. *end-period* can be any value greater than *start-period*.

*start-period* and *end-period* correspond to the asset's life, relative to the fiscal period. For example, if you want to find the first year's depreciation of an asset purchased at the beginning of the second quarter of a fiscal year, *start-period* would be 0 and *end-period* would be 0.75 (1 minus 0.25 of a year). You can use @VDB for multiple-period depreciation calculations.

@VDB allows for the use of an initial-period option to calculate depreciation for the period the asset is placed in service. @VDB uses the fractional part of *start-period* and *end-period* to determine the initial-period option. If both *start-period* and *end-period* have fractional parts, then @VDB uses the fractional part of *start-period*.

*depreciation-factor* is an optional argument that specifies the percentage of straight-line depreciation you want to use as the depreciation rate. If you omit this argument, 1-2-3 uses 200%, which is the double-declining balance rate. *depreciation-factor* can be any value greater than or equal to 0; commonly used rates are 1.25, 1.50, 1.75, and 2.

*switch* is an optional argument that you include if you do not want @VDB to switch to straight-line depreciation for the remaining useful life. Normally, declining-balance switches to such a straight-line calculation when it is greater than the declining-balance calculation.

| If *switch* is | @VDB |
|---|---|
| 0 | Automatically switches to straight-line depreciation when that is greater than declining-balance depreciation; default if you omit the argument |
| 1 | Never switches to straight-line depreciation |

You cannot use an optional argument without using the ones that precede it.

**Notes**

The variable-rate declining balance method maintains a steady rate of depreciation until the salvage value of an asset drops to less than the value of the following equation:

(book value*((1-(rate/life))life))

where book value = cost - salvage - prior depreciation.

At this point, 1-2-3 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, 1-2-3 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.

**Examples**

This example calculates depreciation for an office machine, purchased in the middle of the first quarter of the fiscal year, that cost $10,000. The useful life of the machine is 10 years, and the salvage value after 10 years is $600. The following formulas calculate the depreciation expense for each of the 10 years, using the variable-rate declining balance method, with a depreciation rate of 150%. Notice that the switch to straight-line depreciation occurs in the sixth year.

@VDB(10000;600;10;0;0.875;1.5)      = $1,312.50

@VDB(10000;600;10;0.875;1.875;1.5)  = $1,303.13

@VDB(10000;600;10;1.875;2.875;1.5)  = $1,107.66

@VDB(10000;600;10;2.875;3.875;1.5)  = $    941.51

@VDB(10000;600;10;3.875;4.875;1.5)  = $    800.28

@VDB(10000;600;10;4.875;5.875;1.5)  = $    767.79

@VDB(10000;600;10;5.875;6.875;1.5)  = $    767.79

@VDB(10000;600;10;6.875;7.875;1.5)  = $    767.79

@VDB(10000;600;10;7.875;8.875;1.5)  = $    767.79

@VDB(10000;600;10;8.875;9.875;1.5)  = $    767.79

@VDB(10000;600;10;9.875;10;1.5)      = $      95.97

$9,400.00   Total depreciation (*cost* minus *salvage*)

**Similar @functions**

@DDB calculates depreciation using the double-declining balance method. @SLN uses the straight-line method, and @SYD uses the sum-of-the-years'-digits method.

---

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@VLOOKUP**

@VLOOKUP(*x;range;column-offset*) finds the contents of the cell in a specified column of a vertical lookup table.

**Arguments**

*x* can be either a value or text, depending of the contents of the first column of the vertical lookup table.

| If the first column contains | Then *x* can be |
|---|---|
| Values | Any value greater than or equal to the first value in *range*. If *x* is smaller than the first value in *range*, @VLOOKUP returns ERR. If *x* is larger than the last value in the first column of *range,* @VLOOKUP stops at the last cell in the column specified by *column-offset* and returns the contents of that cell as the answer. |
| Labels | text enclosed in " " (quotation marks), a formula that results in text, or the address or name of a cell that contains a label or a formula that results in a label. If *x* does not exactly match the contents of a cell in the first column of *range*, @VLOOKUP returns ERR. |

*range* represents the location of the vertical lookup table. *range* can be any range address or range name. If *range* is a 3D range, 1-2-3 uses only the first worksheet in *range*.

*column-offset* represents an offset number corresponding to the position the column occupies in *range*.

**Notes**

@VLOOKUP compares *x* to each cell in the first column of the table. When 1-2-3 locates a cell in the first column that contains *x* (or; if *x* is a value; the value closest to ; but not larger than; *x*), it moves across that row the number of columns specified by *column-offset* and returns the contents of that cell as the answer.

**Examples**

@VLOOKUP

**Similar @functions**

@HLOOKUP finds the contents of a cell in a horizontal lookup table. @INDEX finds the contents of a cell when you specify offset numbers for both the column and row. @CHOOSE replaces a lookup table that requires only one row. @MATCH finds the relative position of a cell with specified contents. @XINDEX finds the contents of a cell specified by column, row, and worksheet headings.   @MAXLOOKUP returns an absolute reference to the cell that contains the largest value in a list of ranges. @MINLOOKUP returns an absolute reference to the cell that contains the smallest value in a list of ranges.

**See also**

**Help**

@Functions

*User's Guide*

Chapter 11, "Calculating with @Functions"

**Example: @VLOOKUP**

A vertical lookup table named TAXTABLE (A3..E11) lists tax amounts based on income and filing status.

@VLOOKUP(35329;TAXTABLE;1), entered in a cell formatted as Currency with no decimal places, returns $9,351, the tax amount for the income figure that is closest to, but not greater than, $35,329.

```
A ----- A --------- B -------- C ---------- D -------- E --
1                    ------- F I L I N G   S T A T U S -------
2       Income >=    1               2          3              4
3       $35,000 $9,219  $7,265  $11,31  $8,531
4       $35,050 $9,241  $7,282  $11,340 $8,552
5       $35,100 $9,263  $7,299  $11,363 $8,573
6       $35,150 $9,285  $7,313  $11,386 $8,594
7       $35,200 $9,307  $7,330  $11,411 $8,615
8       $35,250 $9,329  $7,347  $11,436 $8,636
9       $35,300 $9,351  $7,361  $11,459 $8,657
10      $35,350 $9,373  $7,377  $11,483 $8,678
11      $35,400 $9,395  $7,393  $11,507 $8,699
```

**@WEEKDAY**

@WEEKDAY(*date-number*) extracts the day of the week from *date*, and displays it as an integer from 0 (Monday) through 6 (Sunday).

**Arguments**

*date* is a date number.

**Examples**

@WEEKDAY(@DATE(91;7;3)) = 2, Wednesday.

**Similar @functions**

@MONTH calculates the month, using a date number. @YEAR calculates the year, using a date number.

---

**See also**

**Help**

  @Functions

***User's Guide***

  Chapter 11, "Calculating with @Functions"

**@YEAR**

@YEAR(*date-number*) extracts the year, an integer from 0 (1900) through 199 (2099), from *date number*.

**Arguments**

*date-number* is an integer, or the address or name of a cell that contains an integer, from 1 (January 1; 1900) through 73050 (December 31; 2099).

**Notes**

Add 1900 to @YEAR to convert it into a four-digit year. For example, @YEAR(20181)+1900 creates the four-digit year 1955.

@YEAR can supply the *year* argument for other date @functions that build on previously calculated dates.

**Examples**

@YEAR(20181) = 55, because the date number 20181 is the date 02-Apr-55.

@YEAR(@NOW) = the current year

@YEAR(@DATEVALUE("14-Feb-92")) = 92

**Similar @functions**

@DAY extracts the day of the month (1 to 31), and @MONTH extracts the month (1 to 12), from a date number.

---

**See also**

**Help**

@Functions

*User's Guide*

Chapter 11, "Calculating with @Functions"

**@YIELD**

@YIELD(*settlement*;*maturity*;*coupon*;*price*;[*redemption*];[*frequency*];[*basis*]) returns the yield for securities that pay periodic interest.

**Arguments**

*settlement* is the security's settlement date. *settlement* is a date number.

*maturity* is the security's maturity date. *maturity* is a date number. If *maturity* is less than or equal to *settlement*, @YIELD evaluates to ERR.

*coupon* is the security's annual coupon rate. *coupon* is any positive value or 0.

*price* is the security's price per $100 face value. *price* is any positive value.

*redemption* is an optional argument that specifies the security's redemption value per $100 face value. *redemption* is any positive value or 0. If you omit the *redemption* argument, 1-2-3 uses 100.

*frequency* is an optional argument that specifies the number of coupon payments per year.

| *frequency* | Frequency of coupon payments |
|---|---|
| 1 | Annual |
| 2 | Semiannual; default if you omit the argument |
| 4 | Quarterly |
| 12 | Monthly |

*basis* is an optional argument that specifies the type of day-count basis to use.

| *basis* | Day-count basis |
|---|---|
| 0 | 30/360; default if you omit the argument |
| 1 | Actual/actual |
| 2 | Actual/360 |
| 3 | Actual/365 |

You cannot use an optional argument without using the ones that precede it.

**Examples**

A bond has a July 1, 1993, settlement date and a December 1, 1998, maturity date. The semiannual coupon rate is 5.50%. The bond costs $99.50, has a $100 redemption value, and a 30/360 day-count basis. You want to determine the bond's yield:

@YIELD(@DATE(93;7;1),@DATE(98;12;1),0.055,99.5,100,2,0) = 0.056072

**Similar @functions**

@ACCRUED calculates the accrued interest for securities that pay periodic interest. @PRICE calculates the price per $100 face value for securities that pay periodic interest. @DURATION calculates the annual duration for securities that pay periodic interest. @MDURATION calculates the annual modified duration for securities that pay periodic interest.

---

**See also**

**Help**

    @Functions

***User's Guide***

    Chapter 11, "Calculating with @Functions"

**@CHITEST**

@CHITEST(*range1*;[*range2*]) performs a chi-square test for independence on the data in *range1*, or a chi-square test for goodness of fit on the data in *range1* and *range2*.

**Arguments**

*range1* and *range2* are ranges of the same size. If *range1* and *range2* are not the same size, @CHITEST returns ERR.

**Notes**

@CHITEST approximates the probability associated with a chi-square test to within ± $3*10^{-7}$.

**Examples**

@CHITEST: Test for independence
@CHITEST: Test for goodness of fit

**Similar @functions**

@CHIDIST calculates the chi-square distribution. @FTEST performs an *F*-test, @TTEST performs a Student's *t*-test, and @ZTEST performs a *z*-test.

_____

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**Example: @CHITEST (Test for Goodness of Fit)**

@CHITEST(A2..A9;B2..B9) = 0.996882

```
A ------ A -------- B ------
1      Observed        Expected
2            25               23
3            34               35
4            87               91
5            91               88
6            34               34
7            23               22
8            56               60
9            70               68
```

**Example: @CHITEST (Test for Independence)**

@CHITEST(B3..C5) = 0.080809

```
A --- A ----- B ----- C --
1              Rough    Smooth
2
3  High         42         51
4  Medium       25         37
5  Low     85         68
```

**@CRITBINOMIAL**

@CRITBINOMIAL(*trials*;*probability*;*alpha*) returns the largest integer for which the cumulative binomial distribution is less than or equal to *alpha*.

**Arguments**

*trials* represents the number of Bernoulli trials and can be any positive integer or 0.

*probability* represents the probability of success for a single Bernoulli trial and is a value from 0 through 1.

*alpha* represents the criterion probability and is a value from 0 through 1.

**Notes**

@CRITBINOMIAL approximates the cumulative binomial distribution to within ± 3*10^-7.

**Examples**

You manage a small plant that manufactures oil filters. The filters are manufactured in lots of 100. There is an 85% chance that each filter is free from defects. You want to be 99% confident that at least a given number of filters are free from defects.

@CRITBINOMIAL(100;0.85;0.01) = 76, the number of filters free from defects

**Similar @functions**

@BINOMIAL calculates the binomial probability mass function or the cumulative binomial distribution. @COMBIN calculates the binomial coefficient. @PERMUT calculates the number of permutations for a list of values.

_____

**See also**

**Help**

   @Functions

***User's Guide***

   Chapter 11, "Calculating with @Functions"

**@DAVG**

@DAVG(*input*;*field*;[*criteria*]) calculates the average of the values in a *field* of a database table that meet specified *criteria*.

**Arguments**

See Database @Function Arguments.

**Examples**

@DAVG

**Similar @functions**

@AVG and @PUREAVG average the values in a list.

_____

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**Example: @DAVG**

A database table named SALES lists house sales in Arlington, Belmont, and Cambridge in April and May. The sale prices of the houses are listed in the field named SOLD. You want to determine the average price of a house sold in Cambridge:

@DAVG(SALES;"SOLD";CITY="Cambridge") = $365,667

```
A ---- A -------------- B -------- C ----
1   ADDRESS               CITY          SOLD
2   467 Brattle           Cambridge     720,000
3   183 Hillside          Arlington     318,000
4   64 N. Gate       Belmont 332,000
5   80 Mt. Auburn         Cambridge     278,000
6   14 Charles       Cambridge     160,000
7   1160 Memorial         Cambridge     227,000
8   130 Crescent          Arlington     397,000
9   12 Trenton       Arlington     303,000
10  36 Barnes        Cambridge     669,000
11  234 Third        Cambridge     140,000
```

**@DCOUNT**
**@DPURECOUNT**

@DCOUNT(*input*;*field*;[*criteria*]) counts the <u>nonblank</u> cells in a *field* of a database table that meet specified *criteria*.

@DPURECOUNT(*input*;*field*;*criteria*) counts the cells that contain values in a *field* of a database table that meet specified *criteria*.

**Arguments**

See <u>Database @Function Arguments</u>.

**Examples**

<u>@DCOUNT</u>

**Similar @functions**

<u>@COUNT</u> and <u>@PURECOUNT</u> count cells in a list of ranges.

---

**See also**

**Help**

<u>@Functions</u>

***User's Guide***

Chapter 11, "Calculating with @Functions"

**Example: @DCOUNT**

A database table named APR_SALES lists house sales for the month of April. The types of heating systems the houses have are listed in a field named HEAT. You want to find the number of houses heated with gas:

@DCOUNT(APR_SALES;"HEAT";HEAT="Gas") = 4

```
A ---- A ----------- B ------- C ------- D -
1   ADDRESS           BDRMS HEAT        COST
2   671 Washington    5     Gas    $290,000
3   131 Aslett        5     Oil    $105,000
4   46 Carlton        2     Gas    $135,000
5   76 Phillips             4      Elec    $128,000
6   479 Marlborough   2     Gas    $174,000
7   8844 Wonderland   3     Gas    $195,000
```

## @DGET

@DGET(*input*;*field*;[*criteria*]) retrieves a value or label from a *field* of a database table that meets specified *criteria*.

## Arguments

See Database @Function Arguments.

## Notes

If more than one entry meets the *criteria* you specify, @DGET returns ERR.

@DGET is useful when you need to retrieve a value from a single record that meets specific criteria: the employee number of a particular employee, for example. With @DGET, you can retrieve this kind of information automatically for use in a macro, as an argument in an @function, or as a variable in a formula.

## Examples

@DGET

## Similar @functions

@HLOOKUP and @VLOOKUP return the contents of a specified cell in a horizontal or vertical lookup table. @CHOOSE finds an entry in a list. @INDEX returns the contents of a cell in a table based on relative worksheet, column, and row locations. @XINDEX returns the contents of a cell in a table based on worksheet, column, and row headings. @@ indirectly returns the contents of a specified cell.

---

## See also

## Help

@Functions

## *User's Guide*

Chapter 11, "Calculating with @Functions"

**Example: @DGET**

A database table named SALES lists house sales in three towns in April and May. Brokers' commissions are listed in the field named COMM. You want to determine the broker's commission on the sale of the house at 12 Trenton Street:

@DGET(SALES;"COMM";ADDRESS="12 Trenton") = $12,120

```
A ---- A ---------- B -------- C --------
1   ADDRESS              BROKER          COMM
2   467 Brattle          Higle         28,800
3   183 Hillside         Levine        12,720
4   64 N. Gate           Higle         19,920
5   80 Mt. Auburn        Smith         11,120
6   14 Charles           Dunbar         9,600
7   1160 Memorial        Levine        13,620
8   130 Crescent         Dunbar        15,880
9   12 Trenton           Higle         12,120
```

**@DMAX**

@DMAX(*input*;*field*;[*criteria*]) finds the largest value in a *field* of a database table that meets specified *criteria*.

**Arguments**

See Database @Function Arguments.

**Notes**

You can use @DMAX to find the most recent date or time in a list of dates or times.

**Examples**

@DMAX

**Similar @functions**

@MAX and @PUREMAX find the largest value in a list.

---

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**Example: @DMAX**

A database table named SALES lists house sales in Arlington, Belmont, and Cambridge in April and May. The sale prices of the houses are listed in the field named SOLD. You want to determine the highest price paid for a house in Cambridge:

@DMAX(SALES;"SOLD";CITY="Cambridge") = $720,000

```
A ---- A -------------- B -------- C ----
1   ADDRESS             CITY           SOLD
2   467 Brattle         Cambridge    720,000
3   183 Hillside        Arlington    318,000
4   64 N. Gate          Belmont 332,000
5   80 Mt. Auburn       Cambridge    278,000
6   14 Charles          Cambridge    160,000
7   1160 Memorial       Cambridge    227,000
8   130 Crescent        Arlington    397,000
9   12 Trenton          Arlington    303,000
10  36 Barnes           Cambridge    669,000
11  234 Third           Cambridge    140,000
```

**@DMIN**

@DMIN(*input*;*field*;[*criteria*]) finds the smallest value in a *field* of a database table that meets specified *criteria*.

**Arguments**

See Database @Function Arguments.

**Notes**

You can use @DMIN to find the earliest date or time in a list of dates or times.

**Examples**

@DMIN

**Similar @functions**

@MIN and @PUREMIN find the smallest value in a list.

_____

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**Example: @DMIN**

A database table named SALES lists house sales in Arlington, Belmont, and Cambridge in April and May. The sale prices of the houses are listed in the field named SOLD. You want to determine the lowest price paid for a house in Cambridge:

@DMIN(SALES;"SOLD";CITY="Cambridge") = $140,000

```
A ---- A -------------- B -------- C ----
1    ADDRESS            CITY          SOLD
2    467 Brattle        Cambridge   720,000
3    183 Hillside       Arlington   318,000
4    64 N. Gate         Belmont 332,000
5    80 Mt. Auburn      Cambridge   278,000
6    14 Charles         Cambridge   160,000
7    1160 Memorial      Cambridge   227,000
8    130 Crescent       Arlington   397,000
9    12 Trenton         Arlington   303,000
10   36 Barnes          Cambridge   669,000
11   234 Third          Cambridge   140,000
```

**@DQUERY**

@DQUERY(*function*;[*ext-arguments*]) sends a command to an external database.

**Arguments**

*function* is the name of the external database command that you want to execute.

*ext-arguments* is an optional argument that lists the arguments the external command requires, separated by valid argument separators.

**Notes**

Use @DQUERY with another database @function to select specific records from an external database table.

You cannot use @DQUERY with a database @function whose input range contains more than one database table.

Consult the documentation that came with your external database driver to see if the driver supports @DQUERY.

**Examples**

An external database management program has a function called LITERS that converts gallons to liters. The function requires a single argument: the number of gallons to convert. To use the function in 1-2-3, you would enter

+QUANTITY=@DQUERY("LITERS";10)

in a criteria range of a database @function to extract entries from the field QUANTITY of an external table that match the quantity in liters equivalent to 10 gallons.

---

**See also**

**Help**

@Functions
Tools Database Connect to External

*User's Guide*

Chapter 11, "Calculating with @Functions"

**@DSTD**
**@DSTDS**

@DSTD(*input*;*field*;[*criteria*]) calculates the population standard deviation of the values in a *field* of a database table that meet specified *criteria*.

@DSTDS(*input*;*field*;[*criteria*]) calculates the sample standard deviation of sample values in a *field* of a database table that meet specified *criteria*.

**Arguments**

See Database @Function Arguments.

**Notes**

@DSTD uses the n, or population, method to calculate standard deviation of population data. The n method assumes that the selected values are the entire population. If the values are only a sample of the population, the standard deviation is biased because of errors introduced in taking the sample.

Standard deviation is the square root of the variance of all individual values from the mean.

**Examples**

@DSTD and @DSTDS

**Similar @functions**

@STD and @PURESTD calculate the standard deviation of the entire population of values in a range. @STDS and @PURESTDS calculated the standard deviation of sample values. @DVAR calculates the population variance of values that meet criteria you specify.

_____

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**Example: @DSTD and @DSTDS**

This table lists the heights and weights of ten randomly selected test subjects. You want to determine the standard deviation of the heights of subjects who weigh more than 75 kg.

@DSTD(A1..B11;"HEIGHT";WEIGHT>75) = 4.611954

Suppose the ten test subjects are a randomly selected sample of a larger group of test subjects.

@DSTDS(A1..B11;"HEIGHT";WEIGHT>75) = 5.325426

```
A --------- A --------- B --
1       HEIGHT     WEIGHT
2       190.50      72.73
3       187.96      86.36
4       175.26      68.18
5       175.26      76.37
6       180.34      77.27
7       180.34      72.73
8       187.96      75.00
9       172.72      68.18
10      177.80      70.46
11      179.07      86.36
```

**@DSUM**

@DSUM(*input*;*field*;[*criteria*]) calculates the sum of the values in a *field* of a database table that meet specified *criteria*.

**Arguments**

See Database @Function Arguments.

**Examples**

@DSUM

**Similar @functions**

@SUM calculates the sum of the values in a list. @SUMNEGATIVE sums only the negative values in a list. @SUMPOSITIVE sums only the positive values in a list.

**See also**

**Help**

@Functions

*User's Guide*

Chapter 11, "Calculating with @Functions"

**Example: @DSUM**

A database table named SALES lists house sales in Arlington, Belmont, and Cambridge in April and May. The brokers' commissions on the sales are listed in the field named COMM. You want to determine the total commission earned by the broker Dunbar:

@DSUM(SALES;"COMM";BROKER="Dunbar") = $25,480

```
A ---- A -------------- B -------- C ---
1   ADDRESS             BROKER        COMM
2   467 Brattle         Higle       28,800
3   183 Hillside        Levine      12,720
4   64 N. Gate          Higle       19,920
5   80 Mt. Auburn       Smith       11,120
6   14 Charles          Dunbar       9,600
7   1160 Memorial       Levine      13,620
8   130 Crescent        Dunbar      15,880
9   12 Trenton          Higle       12,120
```

**@DVAR**
**@DVARS**

@DVAR(*input*;*field*;[*criteria*]) calculates the population variance of the values in a *field* of a database table that meet specified *criteria*.

@DVARS(*input*;*field*;[*criteria*]) calculates the variance of sample values in a field of a database table that meet specified *criteria*.

## Arguments

See Database @Function Arguments.

## Notes

Variance measures the degree to which individual values in a list vary from the mean (average) of all the values in the list. The lower the variance, the less individual values vary from the mean, and the more reliable the mean. A variance of 0 indicates that all values in the list are equal. Variance is necessary in several ANOVA (analysis of variance) statistical tests.

@DVAR uses the n, or population, method to calculate variance. The n method assumes the selected values are the entire population. If the values are only a sample of the population, the variance is biased because of errors introduced in taking a sample.

Variance is the square of standard deviation.

## Examples

@DVAR and @DVARS

## Similar @functions

@VAR and @PUREVAR calculates the population variance of values in a list. @DSTD calculates the population standard deviation of values that meet criteria you specify.

---

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**Example: @DVAR and @DVARS**

This table lists the heights and weights of ten randomly selected test subjects. You want to determine the variation of the weights of subjects who are taller than 180 cm.

@DVAR(A1..B11;"WEIGHT";HEIGHT>180) = 25.59654

Suppose the subjects are a randomly selected sample of a larger group of test subjects.

@DVARS(A1..B11;"WEIGHT";HEIGHT>180) = 28.36016

```
A --------- A --------- B --
1       HEIGHT    WEIGHT
2       190.50     72.73
3       187.96     86.36
4       175.26     68.18
5       175.26     76.37
6       180.34     77.27
7       180.34     72.73
8       187.96     75.00
9       172.72     68.18
10      177.80     70.46
11      179.07     86.36
```

**@FTEST**

@FTEST(*range1*;*range2*) performs an *F*-test and returns the associated probability.

**Arguments**

*range1* and *range2* are ranges that contain the data you want to test. *range1* and *range2* do not have to be the same size.

**Notes**

@FTEST approximates the probability associated with an *F*-test to within ± 3*10^-7.

Use @FTEST to determine if two samples have different variances.

**Examples**

@FTEST

**Similar @functions**

@FDIST calculates the *F*-distribution. @CHITEST performs a chi-square test, @TTEST performs a Student's *t*-test, and @ZTEST performs a *z*-test.

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**Example: @FTEST**

@FTEST(A2..A13;B2..B15) = 0.157348

```
A ------ A -------- B ------
1      Sample1 Sample
2      84.5              1.65
3      80.7              4.58
4      34.5              42.6
5      54.6              4.37
6      50.5              30.8
7      33.7              97.7
8      46.8              87.2
9      47.6              40.7
10     22.8              38.4
11     15.5              10.6
12     60.6              56.3
13     80.5              70.5
14                       9.04
15                       97.3
```

**@GRANDTOTAL**

@GRANDTOTAL(*list*) calculates the sum of all cells in *list* that contain @SUBTOTAL in their formulas.

**Arguments**

*list* can be any combination of ranges. Separate elements of *list* with argument separators.

**Examples**

> @GRANDTOTAL

**Similar @functions**

@SUM adds a list of values. @DSUM adds values in a database table that meet certain criteria. @SUMNEGATIVE sums only the negative values in a list. @SUMPOSITIVE sums only the positive values in a list.

_____

**See also**

**Help**

> @Functions
> Statistical @Functions Arguments

***User's Guide***

> Chapter 11, "Calculating with @Functions"

**Example: @GRANDTOTAL**

The @GRANDTOTAL formula in cell A10 calculates the sum of all cells in A1..A8 that contain
@SUBTOTAL in their formulas (A4 and A8).

```
A --- A ----- B -------------
1     10
2     15
3
4     25     @SUBTOTAL(A1..A2)
5
6     20
7     25
8     45     @SUBTOTAL(A6..A7)
9
10    70     @GRANDTOTAL(A1..A8)
```

## @ISFILE

@ISFILE(*file-name*;[*type*]) tests *file-name* for a file in memory or on disk. If *file-name* is found, @ISFILE returns 1 (true); if *file-name* is not found, @ISFILE returns 0 (false).

## Arguments

*file-name* is the full name, including the extension, of the file you want to test for, entered as text. Unless you want 1-2-3 to look for the file in the current directory, you must also specify the path as part of *file-name*.

*type* specifies whether to look for *file-name* in memory or on disk. If *type* is 0, 1-2-3 looks for *file-name* in memory; if *type* is 1, 1-2-3 looks for *file-name* on disk. If you omit *type*, 1-2-3 uses 0.

## Examples

@ISFILE("C:\123W\BUDGET\COSTS_93.WK4";1) = 1, if the file COSTS_93.WK4 is stored in C:\123W\BUDGET.

---

## See also

**Help**

    @Functions

***User's Guide***

    Chapter 11, "Calculating with @Functions"

**@MATCH**

@MATCH(*cell-contents*;*range*;[*type*]) returns the position of the cell in *range* whose contents match *cell-contents*. @MATCH returns the cell's position as an offset number.

**Arguments**

*cell-contents* can be either a value or text. If *cell-contents* is text, you can include wildcard characters.

*range* is a range name or address.

*type* is an optional argument that specifies how 1-2-3 compares *cell-contents* with the contents of the cells in *range*.

| *type* | **@MATCH returns the relative position of** |
|--------|---------------------------------------------|
| 0 | The first cell whose contents match *cell-contents* |
| 1 | The cell that contains the largest value that is less than or equal to *cell-contents*; default if you omit the argument. Sort *range* in ascending order. |
| 2 | The cell that contains the smallest value that is greater than or equal to *cell-contents*. Sort *range* in descending order. |

**Notes**

1-2-3 searches *range* from top to bottom in a column and from left to right. If you specify a multi-sheet range, 1-2-3 searches the first worksheet in the range, continues on to the second worksheet, and so on until 1-2-3 reaches a match or the end of the range.

If 1-2-3 cannot find a match for *cell-contents* @MATCH returns ERR.

If *type* is 1 and the first cell in *range* contains a value that is greater than *cell-contents*, @MATCH returns ERR.

If *type* is 2 and the first cell in *range* contains a value that is less than *cell-contents*, @MATCH returns ERR.

**Examples**

@MATCH

**Similar @functions**

@HLOOKUP and @VLOOKUP find the contents of cells in horizontal and vertical lookup tables. @INDEX finds the contents of a cell when you specify offset numbers for both the column and row. @CHOOSE replaces a lookup table that requires only one row. @MAXLOOKUP returns an absolute reference to the cell that contains the largest value in a list of ranges. @MINLOOKUP returns an absolute reference to the cell that contains the smallest value in a list of ranges.

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**Example: @MATCH**

A medicine doseage is determined by body weight. A patient's weight, entered in a cell named PATIENT_WEIGHT, is 125 lbs.

@INDEX(A2..C7;2;@MATCH(PATIENT_WEIGHT;A2..C7;1)) = 2

```
A ---- A --------- B -------- C ------------
1      Pounds  Kilograms    Number of Pills
2          50      22.5                  1.5
3         100      45.5                  2.0
4         150      68.0                  2.5
5         200      90.5                  3.0
6         250     113.5                  3.5
7         300     136.0                  4.0
```

## @NORMAL

@NORMAL(*x*;[*mean*];[*std*];[*type*]) calculates the normal distribution function for *x*.

### Arguments

*x* is the upper bound for the value of the cumulative normal distribution. *x* is any value; if *x* is negative,1-2-3 converts it to its absolute (positive) value.

*mean* is an optional argument that specifies the mean of the distribution. *mean* is any positive value or 0. If you omit *mean*, 1-2-3 uses 0.

*std* is an optional argument that specifies the standard deviation of the distribution. *std* is any positive value or 0. If you omit *std*, 1-2-3 uses 1.

*type* is an optional argument that specifies what function you want @NORMAL to calculate.

| *type* | @NORMAL calculates |
|--------|--------------------|
| 0 | Cumulative distribution function; default if you omit the argument |
| 1 | Inverse cumulative distribution |
| 2 | Probability density function |

You cannot use an optional argument without using the ones that precede it.

### Notes

@NORMAL approximates the cumulative distribution function to within $\pm\ 7.5*10^{-8}$ and the inverse cumulative distribution to within $\pm\ 4.5*10^{-4}$

### Examples

@NORMAL(1.96) = 0.9750

@NORMAL(0.975;0;1;1) = 1.96

@NORMAL(1.96;0;1;2) = 0.58441

### Similar @functions

@CHIDIST calculates the chi-square distribution. @FDIST calculates the *F*-distribution. @POISSON calculates the Poisson distribution. @TDIST calculates the Student's *t*-distribution.

### See also

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@RANGENAME**

@RANGENAME(*cell*) returns the name of the range in which *cell* is located.

**Argument**

*cell* is a cell address or the name of a single-cell range.

**Notes**

If you specify a cell that is in several overlapping named ranges, 1-2-3 returns the first range name it finds.

If *cell* is not in a named range, @RANGENAME returns <u>ERR</u>.

You can use @RANGENAME only with files in memory.

**Examples**

@RANGENAME(A:A2) returns SALES if A:A2 is in the range named SALES.

---

**See also**

**Help**

<u>@Functions</u>
<u>Range Version</u>

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@SCENARIOINFO**
**@VERSIONINFO**

@SCENARIOINFO(*option;name;*[*creator*]) returns information about a <u>scenario</u>.

@VERSIONINFO(*option;version-range;name*[;*creator*]) returns information about a <u>version</u>.

**Arguments**

*option* is <u>text</u> that specifies what information you want 1-2-3 to return.

| *option* | 1-2-3 returns |
|----------|---------------|
| creator | The name of the person who created the version or scenario |
| modifier | The name of the person who last modified the version or scenario |
| created | The date and time the version or scenario was created, as a <u>date</u> and <u>time number</u> |
| modified | The date and time the version or scenario was last modified, as a date and time number |
| comment | The comment for the version or scenario; 1-2-3 truncates the comment if it is longer than 512 single-byte characters |
| hidden | 0 (false) if the version or scenario is not hidden or 1 (true) if it is hidden |
| protected | 0 (false) if the version or scenario is not protected or 1 (true) if it is protected |

*name* is text that specifies the name of the version or scenario. If more than one version or scenario has the same *name*, 1-2-3 uses the one most recently created.

*creator* is text that specifies the name of the user who created the version or scenario. 1-2-3 uses *creator* to help determine which version or scenario to use.

*version-range* is the name of the range that contains the version. *version-range* must be an existing named range.

**Examples**

@SCENARIOINFO("comment";"Best Case";"Kimberly Parker") returns the comment for the latest scenario named Best Case created by Kimberly Parker.

@SCENARIOINFO("creator";"Sales") returns the name of the user who created the latest scenario named Sales.

@VERSIONINFO("created";SALESRANGE;"Best Case";"Kimberly Parker") returns the date and time that Kimberly Parker created her latest Best Case version for SALESRANGE.

@VERSIONINFO("modified";SALESRANGE;"Widgets") returns the date and time that the version Widgets in SALESRANGE was last modified.

**See also**

**Help**

    <u>@Functions</u>
    <u>Range Version</u>

***User's Guide***

    Chapter 11, "Calculating with @Functions"

## @SCENARIOLAST

@SCENARIOLAST(*file-name*) returns the name of the last-displayed scenario in a file during the current 1-2-3 session.

### Arguments

*file-name* is the full name, including the extension, of the file you want to test for, entered as text. Unless you want 1-2-3 to look for the file in the current directory, you must also specify the path as part of *file-name*.

### Notes

If no scenarios have been displayed in *file-name* during the current 1-2-3 session, @SCENARIOLAST returns ERR.

### Examples

@SCENARIOLAST("d:\123r5\sheets\july.wk4") returns the name of the last-displayed scenario in the file JULY.WK4, which is stored in D:\123R5\SHEETS.

---

### See also

### Help

    @Functions
    Range Version

### *User's Guide*

    Chapter 11, "Calculating with @Functions"

**@SEMEAN**

@SEMEAN(*range*) calculates the standard error of the sample mean for the values in *range*.

**Arguments**

*range* is a range name or address.

**Examples**

Suppose the range TEST contains the values 2, 6, 8, 5, 3, 9, 1, and 2.

@SEMEAN(TEST) = 1.052209

**Similar @functions**

@GEOMEAN calculates the geometric mean of the values in a list. @HARMEAN calculates the harmonic mean of the values in a list. @STD and @PURESTD calculate the standard deviation of the values in a list.

---

**See also**

**Help**

   @Functions

***User's Guide***

   Chapter 11, "Calculating with @Functions"

## @TTEST

@TTEST(*range1*;*range2*;[*type*];[*tails*]) performs a Student's *t*-test on the data in *range1* and *range2* and returns the associated probability.

### Arguments

*range1* and *range2* are ranges that contain values.

*type* is an optional argument that specifies what type of *t*-test to perform.

| type | 1-2-3 performs |
|------|----------------|
| 0 | A *t*-test for samples drawn from populations with the same variance (homoscedastic); *range1* and *range2* do not have to contain the same number of cells; default if you omit the argument |
| 1 | A *t*-test for samples drawn from populations with unequal variances (heteroscedastic); *range1* and *range2* do not have to contain the same number of cells |
| 2 | A paired *t*-test; *range1* and *range2* must contain the same number of cells |

*tails* is an optional argument that specifies the direction of the *t*-test.

| tails | 1-2-3 performs |
|-------|----------------|
| 1 | A one-tailed *t*-test |
| 2 | A two-tailed *t*-test; default if you omit the argument |

You cannot use an optional argument without using the ones that it.

### Notes

@TTEST approximates the probability associated with a *t*-test to within $\pm 3*10^{\wedge}-7$.

### Examples

@TTEST

### Similar @functions

@TDIST calculates the Student's *t*-distribution. @CHITEST performs a chi-square test, @FTEST performs an *F*-test, and @ZTEST performs a *z*-test.

### See also

### Help

@Functions

### *User's Guide*

Chapter 11, "Calculating with @Functions"

**Example: @TTEST**

@TTEST(A2..A13;B2..B13) = 0.050022

```
A ------ A -------- B ------
1      Sample1 Sample
2      84.5              65.1
3      80.7              85.4
4      34.5              62.4
5      54.6              73.4
6      50.5              80.3
7      33.7              66.7
8      46.8              87.2
9      47.6              70.4
10     22.8              30.2
11     15.5              60.1
12     60.6              56.3
13     80.5              70.5
```

**@VERSIONCURRENT**

@VERSIONCURRENT(*range*) returns the name of the current <u>version</u> in *range*.

**Arguments**

*range* is the name or address of the range you want to find the version name for.

**Notes**

If no version is current, @VERSIONCURRENT returns <u>ERR</u>.

**Examples**

@VERSIONCURRENT(PROFITS) returns the name of the current version in the range PROFITS.

---

**See also**

**Help**

<u>@Functions</u>
<u>Range Version</u>

*User's Guide*

Chapter 11, "Calculating with @Functions"

**@VERSIONDATA**

@VERSIONDATA(*option*;*cell*;*version-range*;*name*;[*creator*]) returns the contents of a specified cell in a version.

**Arguments**

*option* is text that specifies how you want 1-2-3 to return the contents of *cell*.

| *option* | **1-2-3 returns** |
|----------|-------------------|
| formula | The formula in the cell, as a label, or ERR if the cell does not contain a formula |
| value | The result of the formula if the cell contains a formula |
| | The value or label if the cell contains a value or a label |
| | An empty string if the cell is blank |

*cell* is the name or address of the cell whose contents 1-2-3 returns. *cell* must be located in *version-range*.

*version-range* is the name of the range that contains the version. *version-range* must be an existing named range.

*name* is text that specifies the name of the version. If more than one version has the same *name*, 1-2-3 uses the one most recently created.

*creator* is text that specifies the name of the user who created the version. 1-2-3 uses *creator* to help determine which version to use or delete. If *creator* created multiple versions with the same name, 1-2-3 uses the most recently created of those versions.

**Examples**

@VERSIONDATA("formula";A:B12;SALES;"Best Case") returns the formula located in cell A:B12 of the most recently created version named Best Case in the range SALES.

@VERSIONDATA("value";A:B12;SALES;"Best Case";"Robin Levine") returns the value or label in cell A:B12 of the version named Best Case most recently created by Robin Levine in the range SALES.

**See also**

**Help**

   @Functions
   Range Version

*User's Guide*

   Chapter 11, "Calculating with @Functions"

## @WEIGHTAVG

@WEIGHTAVG(*data-range*;*weights-range*;[*type*]) calculates the weighted average of values in *data-range*.

### Arguments

*data-range* and *weights-range* are the names or addresses of ranges that contain values and are the same size and shape.

If *data-range* and *weights-range* are not the same size and shape, @WEIGHTAVG returns ERR.

*type* is a value that determines how 1-2-3 calculates the weighted average.

| *type* | 1-2-3 divides by |
| --- | --- |
| 0 | The sum of the values in *weights-range*; default if you omit the argument. |
| 1 | The number of values in *data-range*. |

### Examples

This example, taken from a real estate database table, uses @WEIGHTAVG to calculate the weighted average commission due to an agent on house sales in February.

SOLD contains the prices paid for three houses, and COMM contains the agent's commission percentage of the sale price:

```
SOLD        COMM
$25,000     0.04
$34,580     0.05
$77,325     0.04
```

@WEIGHTAVG(SOLD;COMM) = $44,784.62, the weighted average commission due to an agent on the sale of three houses.

### Similar @functions

@SUMPRODUCT calculates the sum of the products of the corresponding cells in multiple ranges.

### See also

### Help

@Functions
Statistical @Functions Arguments

### *User's Guide*

Chapter 11, "Calculating with @Functions"

**@WORKDAY**

@WORKDAY(*start-date*;*days*;[*holidays-range*];[*weekends*]) calculates the underline date number that corresponds to the date that is a specified number of *days* before or after *start-date*, excluding weekends and, optionally, holidays.

**Arguments**

*start-date* is a date number.

*days* is an integer. Use a positive integer to specify a number of days after *start-date* or a negative integer to specify a number of days before *start-date*.

*holidays-range* is an optional argument that specifies holidays to exclude from the @WORKDAY calculation. *holidays-range* is the name or address of a range that contains date numbers.

If you omit the *holidays-range* argument, 1-2-3 does not exclude any holidays from the @WORKDAY calculation.

*weekends* is an optional argument that specifies which days of the week are weekend days. *weekends* is text that uses the integers 0 (Monday) through 6 (Sunday) to represent the days you specify as weekend days.

For example, "45" indicates that Friday and Saturday are weekend days. If you omit *weekends*, 1-2-3 uses "56", which indicates that Saturday and Sunday are weekend days. To specify no weekends, use 7.

**Notes**

If you want to use *weekends* but don't want to use *holidays*, specify a underline blank cell for *holidays*.

**Examples**

You want to determine the date 30 working days after Tuesday, November 1, 1994. You want to specify November 24 and 25 as holidays so you enter date numbers for these dates in a range named HOLIDAYS. You want to specify Saturday and Sunday as weekend days, so you omit the *weekends* argument.

@WORKDAY(@DATE(94;11;1);30;HOLIDAYS) = 34683, or, Thursday, December 15, 1994

**Similar @functions**

@DAYS360 and @D360 calculate the number of days between two date numbers. @NETWORKDAYS calculates the number of days between two dates, excluding weekends and holidays. @NEXTMONTH calculates the date that is a specified number of months before or after a specified date.

---

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

## @XINDEX

@XINDEX(*range*;*column-heading*;*row-heading*;[*worksheet-heading*]) returns the contents of a cell located at the intersection specified by *column-heading*, *row-heading*, and (optionally) *worksheet-heading*.

## Arguments

*range* is a range address or range name.

*column-heading* is the contents of a cell in the first row of *range*.

*row-heading* is the contents of a cell in the first column of *range*.

*worksheet-heading* is an optional argument that is the contents of the first cell in *range*.

*column-heading*, *row-heading*, and *worksheet-heading* can be values or text.

## Examples

@XINDEX

## Similar @functions

@CHOOSE finds an entry in a list. @HLOOKUP and @VLOOKUP find entries in horizontal and vertical lookup tables. @MATCH returns the relative position of a cell in a range. @MAXLOOKUP returns an absolute reference to the cell that contains the largest value in a list of ranges. @MINLOOKUP returns an absolute reference to the cell that contains the smallest value in a list of ranges.

---

## See also

## Help

@Functions

## *User's Guide*

Chapter 11, "Calculating with @Functions"

**Example: @XINDEX**

A table named RATES (A2..E7) lists rates for sending a parcel to several cities.

@XINDEX(RATES;"New York";1) = $9.29, the rate for sending a type 1 parcel to New York.

@XINDEX(RATES;"Paris";5), = $29.00, the rate for sending a type 5 parcel to Paris.

```
A ---- A ---------- B -------- C ----- D --------- E --------
1                     ------------ PARCEL DESTINATION ------------
2 Parcel type London  Paris           Frankfurt     New York
3             1        $18.36 $19.33 $20.12  $ 9.29
4             2        $20.32 $21.66 $22.03  $11.25
5             3        $22.44 $23.88 $24.00  $13.25
6             4        $24.14 $25.26 $25.75  $16.85
7             5        $28.32 $29.00 $29.80  $19.54
```

**@ZTEST**

@ZTEST(*range1*;*mean1*;*std1*;[*tails*];[*range2*];[*mean2*];[*std2*]) performs a *z*-test on one or two populations and returns the associated probability.

**Arguments**

*range1* is a range that contains the first, or only, set of data to test.

*mean1* is the known population mean of *range1* and can be any value.

*std1* is the known population standard deviation of *range1*. *std1* is a value greater than 0.

*tails* is an optional argument that specifies the direction of the *z*-test.

| *tails* | 1-2-3 performs |
|---------|----------------|
| 1 | A one-tailed *z*-test |
| 2 | A two-tailed *z*-test; default if you omit the argument |

*range2* is a range that contains the second set of data to test.

*mean2* is the known population mean of *range2* and can be any value. If you omit *mean2*, 1-2-3 uses 0.

*std2* is the known population standard deviation of *range2*. *std2* is a value greater than 0. If you omit *std2*, 1-2-3 uses 1.

You cannot use an optional argument without using the ones that precede it.

**Notes**

@ZTEST approximates the probability associated with a *z*-test to within ± 7.5*10^-8.

**Examples**

The range A1..A8 contains the following values: 12, 19, 21, 22, 18, 16, 15, 17. If the population mean of these values is 16, and the population standard deviation is 3.041381, then z = 1.394972.

@ZTEST(A1..A8;16;3.041381;1) = 0.081512

**Similar @functions**

@CHITEST performs a chi-square test, @FTEST performs an *F*-test, and @TTEST performs a *t*-test.

**See also**

**Help**

@Functions

*User's Guide*

Chapter 11, "Calculating with @Functions"

**@DDELINK**

@DDELINK(*app-name*;*topic-name*;*item-name*;[*format*];[*max-rows*];[*max-cols*];[*max-sheets*]) creates a DDE link in the current cell.

You can change the link by changing the @DDELINK arguments.

**Arguments**

*app-name* is text that specifies the name of an open Windows application that supports DDE as a server.

*topic-name* is text that specifies the name of the application file to link to. Use "system" to link to the system topic.

*item-name* is text that specifies the name of the item in the server application to link to. This is the item in the server application file to which you want to transfer data through the link.

*format* is text that specifies one of the Clipboard formats. *format* can be Text, WK1, or WK3.

If you omit *format*, 1-2-3 uses the Text Clipboard format.

*max-rows*, *max-cols*, and *max-sheets* specify the maximum number of rows, columns, and worksheets for the destination range. If you omit *max-rows*, *max-cols*, or *max-sheets*, 1-2-3 uses as many rows, columns, or worksheets as the destination range requires.

**Notes**

When you create an on-sheet or DDE link with Edit Links Create, @DDELINK appears in the first cell of the link's destination range.

**Examples**

The following @DDELINK formula creates a DDE link named DDE_LINK3 to the Ami Pro file LOAN.SAM.

@DDELINK("AmiPro";"D:\AMIPRO\DOCS\LOAN.SAM";"DDE_LINK3")

---

**See also**

**Help**

> @Functions
> Overview of DDE and OLE in 1-2-3

***User's Guide***

> Chapter 11, "Calculating with @Functions"

**@DATESTRING**

@DATESTRING(*date*) converts a <u>date number</u> to its equivalent date and displays it as a label using the default International Date format.

**Arguments**

*date* is a date number.

**Notes**

You can change the default International Date format with <u>Tools User Setup International Format</u>.

**Examples**

If the default International Date format is mm/dd/yy, @DATESTRING(34635) returns the label 10/28/94.

**Similar @Functions**

<u>@DATEVALUE</u> calculates the date number for a date entered as a label. <u>@DATE</u> calculates the date number for a specified date.

---

**See also**

**Help**

   <u>@Functions</u>

***User's Guide***

   Chapter 11, "Calculating with @Functions"

**@NETWORKDAYS**

@NETWORKDAYS(*start-date*;*end-date*;[*holidays-range*];[*weekends*]) calculates the number of days from *start-date* through *end-date*, excluding weekends and holidays.

**Arguments**

*start-date* and *end-date* are date numbers.

*holidays-range* is an optional argument that specifies holidays to exclude from the @NETWORKDAYS calculation. *holidays-range* is the name or address of a range that contains date numbers.

*weekends* is an optional argument that specifies which days of the week are weekend days. *weekends* is text that uses the integers 0 (Monday) through 6 (Sunday) to represent the days you specify as weekend days.

For example, "45" indicates that Friday and Saturday are weekend days. If you omit *weekends*, 1-2-3 uses "56", which indicates that Saturday and Sunday are weekend days. To specify no weekends, use 7.

You cannot use an optional argument without using the ones that precede it.

**Notes**

@NETWORKDAYS includes both *start-date* and *end-date* in the result.

If you want to use *weekends* but don't want to use *holidays*, specify a blank cell for *holidays*.

**Examples**

You want to determine the number of working days between Tuesday, November 1, 1994, and Thursday, December 1, 1994. You want to specify November 24 and 25 as holidays, so you enter date numbers for these dates in a range named HOLIDAYS. You want to specify Saturday and Sunday as weekend days, so you omit the *weekends* argument.

@NETWORKDAYS(@DATE(94;11;1);@DATE(94;12;1);HOLIDAYS) = 21

**Similar @Functions**

@DAYS360 and @D360 calculate the number of days between two date numbers. @DAYS calculates the number of days between two dates, using a specified day-count basis. @WORKDAY calculates the date that is a specified number of days before or after a specified date, excluding weekends and holidays. @NEXTMONTH calculates the date that is a certain number of months before or after a specified date.

_____

**See also**

**Help**

@Functions

*User's Guide*

Chapter 11, "Calculating with @Functions"

**@SUMNEGATIVE**
**@SUMPOSITIVE**

@SUMNEGATIVE(*list*) sums only the negative values in *list*.

@SUMPOSITIVE(*list*) sums only the positive values in *list*.

## Arguments

*list* can contain any of the following, in any combination: numbers, numeric formulas, and addresses or names of ranges that contain numbers or numeric formulas. Separate elements of *list* with argument separators.

## Examples

@SUMNEGATIVE(-2;21;5;12;-2;-7) = -11

@SUMPOSITIVE(-2;21;5;12;-2;-7) = 38

## Similar @Functions

@SUM adds a list of values. @DSUM adds values in a database table that meet certain criteria. @SUBTOTAL adds the values in a list and tells @GRANDTOTAL which values to sum.

---

## See also

## Help

@Functions

## *User's Guide*

Chapter 11, "Calculating with @Functions"

**@MAXLOOKUP**
**@MINLOOKUP**

@MAXLOOKUP(*range-list*) returns an <u>absolute reference</u>, including the file name, to the cell that contains the largest value in a list of ranges.

@MINLOOKUP(*range-list*) returns an absolute reference, including the file name, to the cell that contains the smallest value in a list of ranges.

## Arguments

*range-list* can be any combination of ranges. Separate the range names or addresses in *range-list* with <u>argument separators</u>.

If you want to include a single-cell range in *range-list*, make sure you enter it so it looks like a range address. For example, do not use A1; instead, use A1..A1.

1-2-3 ignores labels and blank cells in *range-list*.

If none of the cells in *range-list* contain values, @MAXLOOKUP and @MINLOOKUP return <u>NA</u>.

## Examples

Suppose your 1-2-3 directory contains the files BID1.WK4, BID2.WK4, and BID3.WK4. Each file contains bid information from a different vendor. The files were all created from the same template, so in each file, the total bid figure is in a cell named TOTAL.

The following formula returns the location, including the file name, of the highest bid:

@MAXLOOKUP(<<BID1.WK4>>TOTAL;<<BID2.WK4>>TOTAL;<<BID3.WK4>>TOTAL)

The following formula returns the location, including the file name, of the lowest bid:

@MINLOOKUP(<<BID1.WK4>>TOTAL;<<BID2.WK4>>TOTAL;<<BID3.WK4>>TOTAL)

## Similar @Functions

<u>@HLOOKUP</u> and <u>@VLOOKUP</u> find entries in horizontal or vertical lookup tables. <u>@INDEX</u> finds the contents of a cell when you specify offset numbers for both the column and row. <u>@CHOOSE</u> replaces a lookup table that requires only one row. <u>@MATCH</u> finds the relative position of a cell with specified contents. <u>@XINDEX</u> finds the contents of a cell specified by column, row, and worksheet headings.

---

## See also

**Help**

   <u>@Functions</u>

*User's Guide*

   Chapter 11, "Calculating with @Functions"

**@NEXTMONTH**

@NEXTMONTH(*start-date*;*months*;[*day-of-month*];[*basis*]) calculates the date number for the date that is a specified number of *months* before or after *start-date*.

**Arguments**

*start-date* is a date number.

*months* is an integer. Use a positive integer to specify a number of months after *start-date* or a negative integer to specify a number of months before *start-date*.

*day-of-month* is an optional argument that specifies what day of the month you want the result of @NEXTMONTH to fall on. *day-of-month* is a value from the following table:

| *day-of-month* | @NEXTMONTH returns |
|---|---|
| 0 | A date that falls on the same day of the month as *start-date*. If *start-date* falls on a day of the month that does not exist for the new month (for example, if *start-date* is January 30, 1994 and the new month is February, which has 28 days), @NEXTMONTH returns a date that falls on the last day of the month. Default if you omit the argument. |
| 1 | A date that falls on the first day of the month. |
| 2 | A date that falls on the last day of the month. |

*basis* is an optional argument that specifies the type of day-count basis to use. *basis* is a value from the following table:

| *basis* | Day-count basis |
|---|---|
| 0 | 30/360 |
| 1 | Actual/actual; default if you omit the argument |
| 2 | Actual/360 |
| 3 | Actual/365 |

You cannot use an optional argument without using the ones that precede it.

**Examples**

You want to determine the date that falls on the last day of the month, one month after Thursday, April 7, 1994.

@NEXTMONTH(@DATE(94;4;7);1;2) = 34485, or Tuesday, May 31, 1994

**Similar @Functions**

@DAYS360 and @D360 calculate the number of days between two date numbers. @WORKDAY calculates the date a specified number of days before or after a specified date, excluding weekends and holidays. @NETWORKDAYS calculates the number of days between two dates, excluding weekends and holidays.

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@DURATION**
**@MDURATION**

@DURATION(*settlement*;*maturity*;*coupon*;*yield*;[*frequency*];[*basis*]) calculates the annual duration for a security that pays periodic interest.

@MDURATION(*settlement;maturity;coupon;yield;[frequency];[basis]*) calculates the modified annual duration for a security that pays periodic interest.

**Arguments**

*settlement* is the security's settlement date. *settlement* is a <u>date number</u>.

*maturity* is the security's maturity date. *maturity* is a date number. If *maturity* is less than or equal to *settlement*, @DURATION and @MDURATION evaluate to <u>ERR</u>.

*coupon* is the security's annual coupon rate. *coupon* is any positive value or 0.

*yield* is the annual yield. *yield* is any positive value or 0.

*frequency* is an optional argument that specifies the number of coupon payments per year. *frequency* is a value from the following table:

| *frequency* | Frequency of coupon payments |
| --- | --- |
| 1 | Annual |
| 2 | Semiannual; default if you omit the argument |
| 4 | Quarterly |
| 12 | Monthly |

*basis* is an optional argument that specifies the type of <u>day-count basis</u> to use. *basis* is a value from the following table:

| *basis* | Day-count basis |
| --- | --- |
| 0 | 30/360; default if you omit the argument |
| 1 | Actual/actual |
| 2 | Actual/360 |
| 3 | Actual/365 |

You cannot use an optional argument without using the ones that precede it.

**Notes**

Duration is the weighted average term to maturity of a security's cash flows. The weights are the present value of each cash flow as a fraction of the present value of all cash flows.

**Examples**

A security has a July 1, 1993, settlement date and a December 1, 1998, maturity date. The semiannual coupon rate is 5.50% and the annual yield is 5.61%. The bond has a 30/360 day-count basis.

To determine the security's annual duration:

@DURATION(@DATE(93;7;1);@DATE(98;12;1);0.055;0.0561;2;0) = 4.734591

To determine the security's modified annual duration:

@MDURATION(@DATE(93;7;1);@DATE(98;12;1);0.055;0.0561;2;0) = 4.60541

**Similar @Functions**

@ACCRUED calculates the accrued interest for securities that pay periodic interest. @PRICE calculates the price per $100 face value for securities that pay periodic interest. @YIELD calculates the yield for securities that pay periodic interest.

_____

**See also**

**Help**

@Functions

***User's Guide***

Chapter 11, "Calculating with @Functions"

**@ISEMPTY**

@ISEMPTY(*location*) tests *location* for a <u>blank cell</u>. If *location* is a blank cell, @ISEMPTY returns 1 (true); if *location* is not a blank cell, @ISEMPTY returns 0 (false).

**Arguments**

*location* is the name or address of a single cell. If you specify a range for *location*, @ISEMPTY returns 0 (false).

**Examples**

@ISEMPTY(A1) = 1 if cell A1 is a blank cell

@ISEMPTY(A1) = 0 if cell A1 contains the value 1,963

@ISEMPTY(A1) = 0 if cell A1 contains the label Revenues

@ISEMPTY(A1) = 0 if cell A1 contains a <u>label prefix character</u>

---

**See also**

**Help**

<u>@Functions</u>
<u>Logical @Functions</u> for a list of other @functions that calculate <u>logical formulas</u>.

***User's Guide***

Chapter 11, "Calculating with @Functions"

## @SETSTRING

@SETSTRING(*text*;*length*;[*alignment*]) returns a label that is *length* characters long. The label consists of *text* and sufficient blank spaces to align *text* as specified by *alignment*.

### Arguments

*text* can be any text.

*length* can be any integer from 1 through 512. If *length* is smaller than the number of characters in *text*, @SETSTRING returns *text*.

*alignment* is an optional argument that specifies how to align *text*. *alignment* is a value from the following table:

| *alignment* | 1-2-3 aligns *text* |
|---|---|
| 0 | To the left of the extra spaces in *length*; default if you omit the argument. |
| 1 | In the center of the extra spaces in *length*. If there is an odd number of extra spaces, 1-2-3 adds the one leftover space to the left of *text*. |
| 2 | To the right of the extra spaces in *length*. |

### Notes

Most of the fonts used with Windows are proportionally spaced fonts. Blank spaces generally use less space than letters in proportionally spaced fonts.

### Examples

In the following examples, each • represents a blank space.

@SETSTRING("Element Nine, Inc.";24) = Element Nine, Inc.••••••

@SETSTRING("Element Nine, Inc.";24;1) = •••Element Nine, Inc.•••

@SETSTRING("Element Nine, Inc.";24;2) = ••••••Element Nine, Inc.

### Similar @functions

@TRIM removes leading, trailing, and consecutive spaces from text.

---

### See also

### Help

@Functions

### *User's Guide*

Chapter 11, "Calculating with @Functions"

**Printing Sections of @Functions Help**

Scroll through the list below and then, to print a section of @Functions Help, just click it.

A book containing detailed descriptions of all 1-2-3 @functions and macro commands is now available from Lotus. Click "@Functions and Macros Book Coupon" to print the coupon for ordering this book.

 **@Functions and Macros Book Coupon**

 **Calendar @Functions**

 **Database @Functions**

 **Engineering @Functions**

 **Financial @Functions**

 **Information @Functions**

 **Logical @Functions**

 **Lookup @Functions**

 **Text @Functions**

**Mathematical @Functions:**

 **Conversion @Functions**

 **General @Functions**

 **Hyperbolic @Functions**

 **Rounding Values @Functions**

 **Trigonometric @Functions**

**Statistical @Functions:**

 **Forecasting @Functions**

**General @Functions**

 **Probability @Functions**

 **Ranking @Functions**

 **Significant Tests @Functions**

**Online Coupon**

**Want a One-Stop Shop for All the Information on @Functions and Macro Commands in 1-2-3?**

We've developed a comprehensive resource guide that contains detailed information about each @function and macro command that is available in 1-2-3 Release 5 for Windows. The information (which is also available on-line under the 1-2-3 Help menu) is consolidated into a single guide making it easy for you to find the information you need — in a hurry.

| Working with  @Functions | Developing Powerful Macros |
|---|---|
| The @functions section provides detailed information about each @function including arguments, notes, formula type and examples. | The macro language section provides detailed information about each macro command, including arguments, notes, and examples. |
| Using this section, you will find the descriptions for various formats, learn how to enter @functions and how   to customize the @functions menu. | This section contains information to help you plan, create, name, run, debug, and save your 1-2-3 macros. |

**Order your 1-2-3 Release 5 for Windows
@Functions and Macro Commands Reference Guide
Call 1-800-872-3387 extension 9635
In Canada: 1-800-GO-LOTUS**

*Detach this card and place it in a stamped envelope.  Make checks payable to Lotus Development Corp.*
*In Canada, make checks payable to Lotus Development Canada Limited*

**Please allow 4 - 6 weeks for delivery.**

PLEASE PRINT

Company:

Name:

Address:

City:
State/Province:

Zip Code/Postal Code:

Payment: AMEX___  VISA___  MC___C heck___            Card #:

Signature:
Exp. Date:

**UNITED STATES**:   To order the reference guide:   Please call 1-800-872-3387 ext. 9635.   Or simply complete this card, enclose $14.95 plus any applicable sales tax*, and mail to:

**Lotus Development Corporation
Box 100, 440 Lincoln Street
Worcester, MA 01653-0100
Attn:   Guide to @Functions and Macros in 1-2-3 Release 5**

*Include sales tax for the following states:   AZ, CA, CO, CT, FL, GA, IL, IN, KS, LA, MD, MA, MI, MN, MO, NJ, NY, NC, OH, PA, TN, TX, VA, WA

**CANADA**:   To order the reference guide:   Please call 1-800-GO-LOTUS.   Or simply complete this card, enclose C$19.95 plus applicable taxes, and mail to:

**Lotus Development Canada Limited
P.O Box 679**

**Scarborough, Ontario   M1K 5C5**
**Attn: Guide to @Functions and Macros in 1-2-3 Release 5**

**Marks the start of the @Function section of 123CRTF.H**

**Marks the end of the @Function section of 123CRTF.H**

**Marks the start of the Macro section of 123CRTF.H**

**Marks the end of the Macro section of 123CRTF.H**

**Marks the start of the Main Help section of 123CRTF.H**

**Marks the end of the Main Help section of 123CRTF.H**

**Brought to you by the 1-2-3 for Windows Doc Team and their friends**

Wendy Clarke, Stan Doherty, Maida Eisenberg, David Folk, Gisele Gauthier, Joseph Haungs, Kathy Howard, John Hunt, Mark Jacober, Mike Lowry, Arthur Manzi, Lisa Mosley, Henry Nigro, Peter Rodman, Laura Sohval, Debbie Stolper, Marie Tahir, Bob Voges, Darrell Williams, and of course, Tonya and Nancy.