10 Using Functions

In this chapter:

10 Using Functions

One way Informed helps you design intelligent forms is by letting you create cells with calculated values. Cells with calculated values are filled in automatically when you fill out a form with Informed Filler or the test mode of Informed Designer (we'll refer to these common capabilities as features of Informed). A cell's value can be calculated based on the values of other cells. In addition, the cells you fill in can be checked for errors, and the tabbing order of a form can be based on conditions.

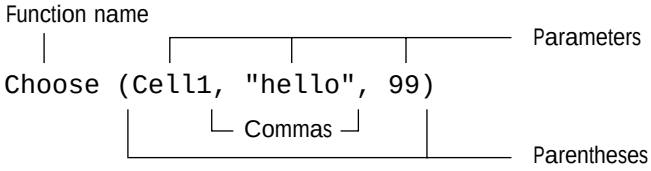
Overview

You calculate or check a cell's value, or specify a cell's next tab position with a formula. Formulas use operators to combine constant values, cell values, and functions to give a single value as a result. Operators are special symbols that produce a new value from other values. The other values (called operands) can be constant values that you enter directly, cell values, or the result of a function. For a complete discussion of formulas and how they work, see Chapter 9.

A *function* performs a predefined calculation using a given set of values, called *parameters*. Functions return a single value called a *result*. For example, you can use the 'Max' function to find the maximum number in a group of numbers.

```
Max (Cell1, Cell2, Cell3)
```

Without the 'Max' function, you'd have to create a formula that compares the numbers in Cell1, Cell2, and Cell3 individually. You can use a function, such as 'Max,' in a formula anywhere you can use a constant or a cell value, provided the type of the function's result is appropriate.



In this chapter you'll learn about functions, function parameters, and function results. A summary of Informed's functions, grouped into general categories, is followed by a detailed description of each function. For information about entering functions in formulas, see "Calculations," "Check Formulas," and "Conditional Tabbing" in Chapter 1.

Function Parameters

When you use a function, you must supply it the values that it needs to calculate its result. These values are called *parameters*. A function's parameters always appear immediately after the function's name. You enclose parameters in parentheses and separate them with commas. For example, the following function calculates the sum of three parameters: 2, 4, and 8.

```
Sum (2, 4, 8)
```

The number of parameters that you supply to a function depends on the particular function. Some functions don't require any parameters. Other functions use a specific number of parameters. For example, the 'Today' function, which returns the current date, doesn't have any parameters, whereas the 'AddDays' function requires two parameters.

```
Today
AddDays (PurchaseDate, 7)
```

If you use a function that doesn't have any parameters, you enter only the function's name. The parentheses are not required.

Some functions allow optional parameters. For example, the 'PMT' function uses an optional parameter to indicate when payments are made on an investment (either at the beginning or end of each period). If you don't supply an optional parameter, the function will calculate its result using a default parameter value.

In the formula below, the parameters to the 'Sum" function are all constant, or unchanging, values.

```
Sum (2, 4, 8)
```

You can use other kinds of parameters, such as cell names, other functions, or complete formulas with operators and operands. Consider the function shown below.

```
Sum (Cell1, 99 + Cell2, MakeList(Cell3), Sqrt(Cell4))
```

This function adds the following parameters:

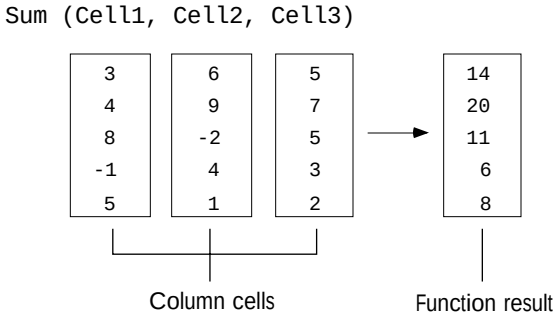
- the value in 'Cell1'
- a formula that adds 99 to the value in 'Cell2'
- the values in the column cell, 'Cell3'
- the result of the 'SQRT' function.

When you use a cell, formula, or function as a parameter, Informed Filler first calculates its value and then uses that result as the parameter value.

The type of a parameter value should match the type expected by the function. For example, all parameters to the ‘Sum’ function should be numbers. If the type of a parameter’s value is not appropriate, Informed Filler will try to convert the value to the correct type. If the value can’t be converted, the empty value is used instead. For more information about type compatibility, see “Type Conversion” in Chapter 9.

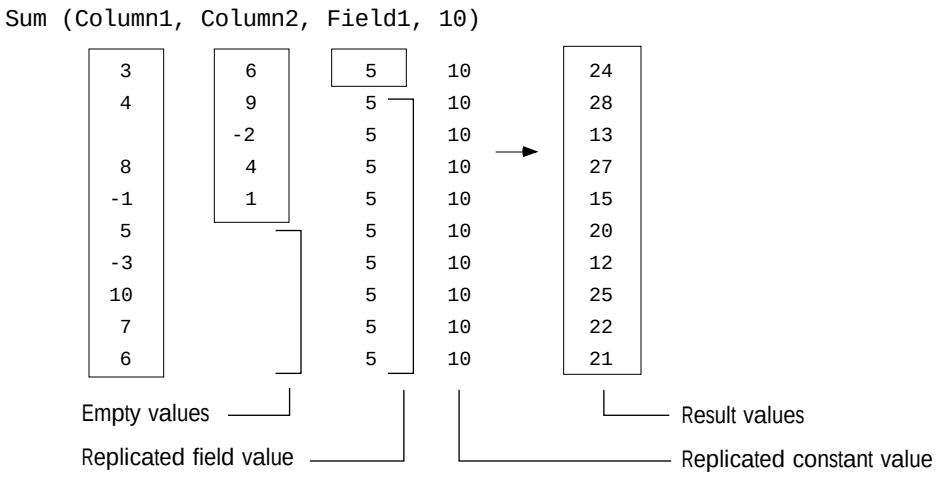
Column cell parameters

When you use a column cell as a function parameter, the function’s result will also be a column. That is, the result will consist of multiple values. For example, if the SUM function were used to add four column cells, each containing five rows, the function’s result would also be a column consisting of seven rows. Each row would contain the sum of the corresponding rows in the four column cells.



If the column cells in the above example were of different heights, the function’s result would contain as many values as the longest column. Informed would insert empty values in the missing rows of the shorter column cells.

If you use column cells along with field cells or constants as parameters, the field cell and constant values are applied to each row of the column cells. The following figure illustrates how the ‘Sum’ function would add two column cells with a field cell and a constant.



If you want the values in the rows of a column cell to be treated as though they were supplied as individual parameters, use the ‘MakeList’ function. This function separates the values in each row of a column cell into separate parameter values. You can use ‘MakeList’ only with functions that allow any number of parameters. For example, to calculate the sum of the rows in a column cell called ‘Extension’ plus the shipping charge in ‘Shipping,’ you could use the following formula.

```
Sum (MakeList (Extension), Shipping)
```

If ‘Extension’ contains four rows with the values 4.99, 5.95, 2.99, and 10.00, and ‘Shipping’ contains the value 5.00, the above formula is equivalent to:

```
Sum (4.99, 5.95, 2.99, 10.00, 5.00)
```

If you didn’t use the ‘MakeList’ function, the result of the ‘Sum’ function would be a column with four rows, each containing the sum of the corresponding row in ‘Extension’ plus the value of ‘Shipping.’ For more information about the ‘MakeList’ function, see the function reference later in this chapter.

If in a place where a list of values is normally expected, a single column cell is given instead, Informed will automatically apply the ‘MakeList’ function to turn the column cell into a list of individual values. The functions below each accept a list of values. In both cases, ‘Column1’ is a column cell.

```
Sum (Column1)
Choose (Cell1, Column1)
```

The Choose function accepts two or more parameters. The first is a single value cell. This parameter is followed by one or more parameters forming a variable length list. If you include a column cell as one of these parameters, Informed applies the ‘MakeList’ function. Informed interprets these functions as though you had typed:

```
Sum (MakeList (Column1))
Choose (Cell1, MakeList (Column1))
```

Function Results

Each Informed function returns a result of a particular type. You can use a function in a formula anywhere you can use a constant or cell value with the same type. When you use a function in a formula, the function’s result is calculated and then used as an operand when the formula is evaluated.

You can use a function as the single operand in a formula. The function’s result is the result of the formula. For example, the formula below extracts the year from a date.

```
YearOf (ToDate ("March 3, 1991"))
```

You can also use a function as one of many operands in a formula, or as a parameter to another function. The formula below uses the ‘SQRT’ function as an operand to the multiplication (*) operator.

```
(5 * Sqrt (Cell1))/100
```

The ‘SQRT’ function is evaluated and its result is then multiplied by 5.

In the following example, the ‘SQRT’ function is used to calculate the values of both parameters to the ‘Sum’ function.

```
Sum (Sqrt (Cell1), Sqrt (Cell2))
```

When you use a function in a formula or as a parameter to another function, the type of the function’s result value must match the type expected by the formula or function parameter. For example, since the multiplication operator multiplies numbers, it expects number values as its operands. If the types don’t match, Informed will try to convert the parameter or operand value to the correct type. For example, if you use the text constant ‘5’ with the multiplication operator, Informed will convert the text value to the numeric value 5 before multiplying. For more information regarding type compatibility, see “Type conversion” in Chapter 9.

Function Reference

This section summarizes the Informed functions and gives a detailed description of each function. Each function reference includes:

- the name of the function
- the function's parameters
- a description of the function's parameters and the calculation that the function performs
- example uses of the function
- any related functions.

In the function parameter list, optional parameters are enclosed in square brackets ([ ]). The square brackets are there only to tell you that the parameter is optional; don't type them. If a function allows any number of parameters, the parameter list will end with an ellipsis (…). All of the parameters that you supply must be separated with commas and they must be the appropriate type.

Each function description gives several examples of the function's use. The examples use different kinds of parameters: constant values, cell references, other function references, and formulas. When a cell reference is used in an example, the example gives the assumed value of the cell. The result of each example appears to the right of an arrow (→). Each result is shown in a common format that's appropriate for the function's result type.

The following lists show a summary of Informed's functions grouped by category. A detailed description of each function follows.

Mathematical Functions

Fact (number)
Sign (number)
SQRT (number)
Inv (number)
Int (number)
Frac (number)
Round (number, decimals)
Ceiling (number)
Floor (number)
Abs (number)
Markup (cost, selling)
Margin (cost, selling)
Convert (number, fromUnit, toUnit)
ConvertTo (value, toUnit)
Trunc (number, decimals)
Sum (number1, number2, …)

Mathematical Functions (continued)

Min (number1, number2, …)
Max (number1, number2, …)
RunningTotal (startValue, columnCell)
Random (min, max)
NumForm (number, format, currency)

Statistical Functions

Count (value1, value2, …)
Mean (number1, number2, …)
Median (number1, number2, …)
GMean (number1, number2, …)
HMean (number1, number2, …)
Range (number1, number2, …)
Var (number1, number2, …)
PVar (number1, number2, …)
StDev (number1, number2, …)
PStDev (number1, number2, …)
SumSq (number1, number2, …)

Trigonometry Functions

Sin (number)
Cos (number)
Tan (number)
ASin (number)
ACos (number)
ATan (number)

Logarithm Functions

Log (number)
ALog (number)
LogN (number, base)
ALogN (number, base)
Ln (number)
Ln1 (number)
Exp (number)
Exp1 (number)

Boolean Functions

AnyOf (boolean1, boolean2, …)
AllOf (boolean1, boolean2, …)
OneOf (boolean1, boolean2, …)

Boolean Functions (continued)

Choose (index, value1, value2, …)
Member (target, value1, value2, …)
WhichMember (target, value1, value2, ...)
IsEmpty (cell)
Between (value, startValue, endValue)
Within (value, startValue, endValue)
IFT (boolean, trueValue)
IFTE (boolean, trueValue, falseValue)

Text Functions

Length (text)
Concat (text1, text2, ...)
Mid (text, start, length)
Left (text, length)
Right (text, length)
Insert (text, start, subText)
Delete (text, start, count)
Replace (text, start, count, subText)
Pos (subText, text)
BeginsWith (text, subText)
EndsWith (text, subText)
Contains (text, subText)
Repeat (text, count)
Upper (text)
Lower (text)
Trim (text)
UpperFirst (text)
UpperWords (text)
Tokenize (text, delimiter)
TransLiterate (text, source, destination)
ASCIIChar (number)
ASCIICode (character)
CharForm (text, format, fromLeft, default)

Date Functions

Today
DayOf (date)
MonthOf (date)
YearOf (date)
MonthName (monthNumber)
MonthAbbrev (monthNumber)
DayOfWeek (date)
DayOfYear (date)

Date Functions (continued)

WeekOfYear (date)
DayName (dayNumber)
DayAbbrev (dayNumber)
AddDays (date, number)
AddMonths (date, number)
AddYears (date, number)
MakeDate (day, month, year)
LastDayOfMonth (date)
WorkDays (date1, date2, mask)
DateForm (date, format)

Time Functions

Now
HourOf (time)
MinuteOf (time)
SecondOf (time)
AddSeconds (time, number)
AddMinutes (time, number)
AddHours (time, number)
MakeTime (hour, minute, second)
TimeSpan (date, time)
TimeForm (time, format)

Name Functions

PrefixCount (name)
MiddleCount (name)
MiddleInitialsOf (name)
SuffixCount (name)
PrefixOf (name, number)
FirstOf (name)
FirstInitialOf (name)
MiddleOf (name, number)
LastOf (name)
LastInitialOf (name)
SuffixOf (name, number)
NameForm (name, format)

Spell Functions

NumberTH (number)
SpellNumber (number)
SpellNumberTH (number)
SpellCurrency (number, decimals)

Amortization Functions

PV (fv, pmt, rate, term[, BEGIN or END])
FV (pv, pmt, rate, term[, BEGIN or END])
PMT (pv, fv, rate, term[, BEGIN or END])
Rate (pv, fv, pmt, term[, BEGIN or END])
Term (pv, fv, pmt, rate[, BEGIN or END])
PPMT (pv, pmt, rate, term, per[, BEGIN or END])
IPMT (pv, pmt, rate, term, per[, BEGIN or END])
Principal (pv, pmt, rate, term, per[, BEGIN or END])

Bond Functions

BondPrice (face, rate, ytm, pmts, yield)
BondYield (price, face, rate, ytm, pmts)
PBondPrice (face, rate, yield)
PBondYield (price, face, rate)

Depreciation Functions

SLD (cost, salvage, life)
SLDValue (cost, salvage, life, term)
SOYD (cost, salvage, life, term)
SOYDValue (cost, salvage, life, term)
DBal (cost, salvage, life, term[, factor])
DBalValue (cost, salvage, life, term[, factor])
MDBal (cost, salvage, life, term[, factor])
MDBalValue (cost, salvage, life, term[, factor])

Cash Flow Functions

NPV (rate, columnCell[, BEGIN or END])

Choice Functions

Choices (cell)
ValidChoice (value, cell)

Page Functions

Page
PageCount
Part
PartLabel (label1, label2, label3, ...)
PartCount

Table Functions

Row
RowCount (columnCell)
MakeList (value1, value2, ...)
CollapseList (value1, value2, ...)
MakeColumn (value1, value2, ...)
CollapseColumn (value1, value2, ...)
Column

Type Conversion Functions

ToDate (value)
ToBoolean (value)
ToPicture (value)
ToSignature (value)
ToTime (value)
ToName (value)
ToNumber (value)
ToText (value)

Record Functions

Attachments
CreationTime
CreationDate
ModifyTime
ModifyDate
SendTime
SendDate
PrintTime
PrintDate

Template Functions

TemplateName
TemplateStatus
TemplateID
TemplateRevision
AuthorName
AuthorOrg

General Functions

RegisteredCompany
RegisteredName
UserName

General Functions (continued)

Application
External (externalName, value1, value2, ...)
Platform

ABS (number)

The ‘Abs’ function returns the absolute value of *number.*

Examples

```
Abs (-4) → 4
Abs (0) → 0
Abs (101) → 101
```

Related Functions

None.

ACOS (number)

The ‘Acos’ function returns the arccosine of *number.* The arccosine is the angle whose cosine is equal to *number. Number* must be in the range -1 to 1. ACOS returns the angle in radians between 0 and π.

Examples

```
Acos (-0.75) → -0.722734 radians
Acos (1) → 0 radians
```

Related Functions

‘Acos’ is the inverse of the ‘Cos’ function. ‘Asin’ and ‘Atan’ return the arcsine and arctangent, respectively, of a number.

ADDDAYS (date, number)
ADDMONTHS (date, number)
ADDYEARS (date, number)

These functions calculate a new date based on an existing *date* and a desired increment or decrement. The’AddDays’ function returns the date calculated by adding *number* days to *date*. The ‘AddMonths’ function returns the date calculated by adding *number* months to *date*. The ‘AddYears’ function returns the date calculated by adding *number* years to *date*. For an explanation of dates and date constants, see *Date* and *Name, date, and time constants.*

Examples

```
AddDays ("September 1, 1939", -10) → August 22, 1939
AddMonths ("07/23/55", 12) → 07/23/56
AddDays ("Dec 25, 1999", 7) → January 1, 2000
AddYears ("Thurs, May 30, 1991", -31) → Monday, May 30, 1960"
```

If the current date is September 2, 1989 and *years*, *months*, and *days* contain the values 25, 4, and 8, respectively, then

```
AddYears (AddMonths (AddDays (Today, -days), -months), -years) → Apr 25, 1964
```

Related Functions

'DayOf,' 'MonthOf,' and 'YearOf' return the numeric value for the day, month, or year of a date. 'DayOfWeek,' 'DayOfYear,' and 'WeekOfYear' return the numeric value for the weekday, day of year, and week of year of a date.

ADDHOURS (time, number)
ADDMINUTES (time, number)
ADDSECONDS (time, number)

These functions calculate a new time based on an existing *time* and a desired increment or decrement. The 'AddHours' function returns the time calculated by adding *number* hours to *time*. The 'AddMinutes' function returns the time calculated by adding *number* minutes to *time*. The 'AddSeconds' function returns the time calculated by adding *number* seconds to *time*.

Examples

```
AddSeconds ("6:45:01 PM", 45) → 6:45:46 PM
AddMinutes ("11:30", -100) → 09:50
AddHours ("23:00:00", 50.5) → 1:30:00
```

If *start* contains the value 8:30 AM and *hoursWorked* contains the value 8.5, then

```
AddHours (start, hoursWorked) → 5:00 PM
```

Related Functions

'HourOf,' 'MinuteOf,' and 'SecondOf' return the hour, minute, or second of a time value.

ALLOF (boolean1, boolean2, ...)

The 'AllOf' function returns the boolean value True if all of its boolean parameters are True. If one or more of its parameters are False, 'AllOf' returns False. The parameters can be any formula that returns a boolean value.

Examples

```
AllOf (True) → True
AllOf (1 + 2 = 3, True, 3 > 2) → True
AllOf (True, 0 > 1) → False
```

If *x* contains the value 1 or if *Cell1* is empty, then

```
AllOf (x ≠ 1, Not IsEmpty (Cell1)) → False
```

Related Functions

'AnyOf' returns True if any of its parameters are True. 'OneOf' returns True if exactly one of its parameters is True.

ALOG (number)

The 'ALog' function returns the base 10 antilogarithm of *number*. The antilogarithm is the number whose logarithm is *number*. A base 10 antilogarithm is equivalent to 10 raised to the power *number*.

Examples

```
Alog (2) → 100
Alog (0) → 1
Alog (-2) → 0.01
Alog (Log (5)) → 5
```

Related Functions

'ALog' is the inverse of the 'Log' function. AlogN' returns a number's antilogarithm for a given base.

ALOGN (number, base)

The 'ALogN' function returns the antilogarithm of *number* using the positive base *base*. The antilogarithm is the number whose logarithm is *number*. A base *base* antilogarithm is equivalent to *base* raised to the power *number*.

Examples

```
AlogN (2,10) → 100
AlogN (0,42) → 1
AlogN (-2,10) → 0.01
```

Related Functions

'ALogN' is the inverse of the 'LogN' function. 'ALog' returns the base 10 antilogarithm of a number.

ANYOF (boolean1, boolean2, ...)

The ‘AnyOf’ function returns the boolean value True if one or more of its boolean parameters are True. If all its parameters are False, ‘AnyOf’ returns False. The parameters can be any formula that returns a boolean value.

Examples

```
AnyOf (False, False, False, True) → True
AnyOf (100 / 10 = 9, 4 * 2 = 7, 0 = 1) → False
AnyOf (1 = 1, False) → True
```

If *x* contains the value 1 or if *Cell1* is empty, then

```
AnyOf (x = 0, AllOf (x = 1, False), IsEmpty (Cell1)) → True
```

Related Functions

‘AllOf’ returns True if all of its parameters are True. ‘OneOf’ returns True if exactly one of its parameters is True.

APPLICATION

The ‘Application’ function returns either “Informed Designer” or “Informed Filler.”

ASCIICHAR (number)

The ‘ASCIIChar’ function returns the ASCII character represented by the value *number*. *Number* must be numeric in the range 0 to 255.

Examples

```
ASCIIChar (65) → "A"
```

Related Functions

‘ASCIICode’ returns the number corresponding to an ASCII character.

ASCIICODE (character)

The ‘ASCIICode’ function returns the number corresponding to the ASCII character given in *character*. If *character* evaluates to a text value with more than one character, ‘ASCIICode’ uses the first character.

Examples

```
ASCIICode ("A") → 65
```

Related Functions

‘ASCIIChar’ returns the ASCII character of a specified value.

ASIN (number)

The ‘ASin’ function returns the arcsine of *number.* The arcsine is the angle whose sine is equal to *number. Number* must be in the range -1 to 1. ‘ASin’ returns the angle in radians between -π/2 and π/2.

Examples

```
Asin (-0.25) → -0.2527
Asin (0.84147) → 1
```

Related Functions

‘ASin’ is the inverse of the ‘Sin’ function. ‘ACos’ and “ATan’ return the arccosine and arctangent, respectively, of a number.

ATAN (number)

The ‘ATan’ function returns the arctangent of *number.* The arctangent is the angle whose tangent is equal to *number.* ‘ATan’ returns the angle in radians between -π/2 and π/2.

Examples

```
Atan (-0.025) → -0.02499
Atan (100) → 1.5608
Atan (0) → 0
```

Related Functions

‘ATan’ is the inverse of the ‘Tan’ function. ‘ACos’ and ‘ASin’ return the arccosine and arcsine, respectively, of a number.

ATTACHMENTS

The ‘Attachments’ function returns a column value with the names of every attachment for the current record.

AUTHORNAME

The ‘AuthorName’ function returns the author name from the Template Information dialog box for the curent template.

AUTHORORG

The ‘AuthorOrg’ function returns the author organization from the Template Information dialog box for the current template.

BEGINSWITH (text, subText)

The ‘BeginsWith’ function is a boolean function. It returns the value True if *text* begins with the group of characters in *subText*. The beginning characters of *text* must match the characters in *subText* exactly. If *text* does not begin with *subText*, ‘BeginsWith’ returns the value False. If *subtext* contains more characters than *text*, ‘BeginsWith’ returns False.

Examples

```
BeginsWith ("AAA Pizza", "A") → True
BeginsWith ("XYZ Vacuums", "XYz") → False
BeginsWith ("tele", "television") → False
BeginsWith ("television", "tele") → True
```

If *title* contains the value "Dr." and *name* contains the value "Dr John Smith", then

```
BeginsWith (name, title) → False
```

Related Functions

‘EndsWith’ returns True if a text value ends with a specified group of characters. ‘Pos’ returns the starting position of a group of characters in a text value. ‘Contains’ returns True if a text value contains a specified group of characters.

BETWEEN (value, startValue, endValue)

The ‘Between’ function is a boolean function. It returns the value True if *value* is between *startValue* and *endValue*. All three parameters to the ‘Between’ function must be the same type. They can be numbers, dates, times, text, or boolean values. If *value* is less than or equal to *startValue*, or if *value* is greater than or equal to *endValue*, then ‘Between’ returns False; otherwise, ‘Between’ returns True.

Informed uses the standard comparison operators to compare *value* with *startValue* and *endValue*. See *Comparison operators* for information about how each data type is compared.

Examples

```
Between (5, -1, 99) → True
Between (ToDate ("Oct. 3, 1989"), ToDate ("01/01/90"),
ToDate ("Jan. 1, 1990")) → False
Between (ToTime ("1:30 PM"), ToTime ("15:15:00"),
ToTime ("2:30 PM")) → False
Between ("actuary", "gymnast", "sensible") → False
Between ("actuary", "Gymnast", "sensible") → False
```

Related Functions

'Within' returns True if a value is between two other values, inclusive.

BONDPRICE (face, rate, ytm, pmts, yield)
BONDYIELD (price, face, rate, ytm, pmts)
PBONDPRICE (face, rate, yield)
PBONDYIELD (price, face, rate)

Bond prices (*price*) and yields to maturity (*yield*) can be equated using the face value of the bond (*face*), the interest rate shown on the bond (*rate*), the number of years to maturity (*ytm*), and the number of interest payments per year (*pmts*). The equation is as follows:

$$Price = \frac{\frac{rate \times face}{pmts}}{\left(1 + \frac{yield}{pmts}\right)} + \frac{\frac{rate \times face}{pmts}}{\left(1 + \frac{yield}{pmts}\right)^2} + \ldots + \frac{\frac{rate \times face}{pmts}}{\left(1 + \frac{yield}{pmts}\right)^{ytm \times pmts}} + \frac{face}{\left(1 + \frac{yield}{pmts}\right)^{ytm \times pmts}}$$

Given *face*, *rate*, *ytm*, *pmts*, and *yield*, the 'BondPrice' function returns the market *price* of a bond. Given *price*, *face*, *rate*, *ytm*, and *pmts*, the 'BondYield' function returns the effective *yield* to maturity of a bond.

The 'PBondPrice' and 'PBondYield' functions return the *price* and the *yield*, respectively, for perpetual bonds. A perpetual bond has no maturity date; payments are made forever. The *ytm* factor in the above equation is effectively infinite and the equation reduces to:

$$Price = \frac{rate \times face}{yield}$$

Examples

Suppose a 7% bond with a $1,000 face value, ten years to maturity, two interest payments per year, and a current market price of $1040 was sold. The 6.45% yield on the bond is calculated by using the 'BondYield' function as follows:

```
BondYield (1040.00, 1000.00, 0.07, 10, 2) → 0.0645
```

The selling price required to attain a desired yield of 8% for the same bond can be calculated using the 'BondPrice' function:

```
BondPrice (1000.00, 0.07, 10, 2, 0.08) → 932.05
```

The market price for a 5.0% perpetual bond with a face value of $1,000 and a desired yield of 6.0% is:

```
PBondPrice (1000.00, 0.05, 0.06) → 833.33
```

If the same perpetual bond is offered at a price of $900.00, the effective yield is:

```
PBondYield (900.00, 1000.00, 0.05) → 0.0556
```

Related Functions

None.

CEILING (number)

The ‘Ceiling’ function returns the next integer greater than or equal to *number*.

Examples

```
Ceiling (0.0000001) → 1
Ceiling (14.4) → 15
Ceiling (-42.0001) → -42
Ceiling (87.000) → 87
```

Related Functions

‘Floor’ returns the next integer less than or equal to a number.

CHARFORM (text, format, fromLeft, default)

The ‘CharForm’ function formats the text value *text* using the character format specified in the text parameter *format*. The parameters format, fromLeft, and default correspond directly to the settings on the Cell dialog for character cells. For information about the meaning and use of these parameters, see *Character*.

Examples

```
CharForm ("1234567", "(###) ###-####", False, "(415)000-0000") → "(415) 123-4567"
CharForm ("1234567", "(###) ###-####", True, "(415)000-0000") → "(123) 456-7000"
CharForm ("k735", "AA###", False, "FN000") → "FK735"
```

Related Functions

‘NumForm’ formats a number, ‘DateForm’ formats a date, ‘NameForm’ formats a name, and ‘TimeForm’ formats a time value.

CHOICES (cell)

The ‘Choices’ function returns a column value containing the values of the choices for the cell called *cell*. *Cell* must be the name of a cell. You enter the choices for a cell using the Choices command. See “Choices” in Chapter 1 for more information.

Examples

If the cell called ‘Ship Method’ has the choices “Federal Express”, “UPS”, and “US Mail”

```
Choices (Ship Method) → "Federal Express", "UPS", "US Mail"
Member ("UPS", MakeList (Choices (Ship Method))) → True
```

Related Functions

‘ValidChoices’ returns True if a specified value is a valid choice.

CHOOSE (index, value1, value2, ...)

The ‘Choose’ function returns one of its parameters. The parameter returned is selected based on the value of *index*. If *index* is 1, *value1* is returned; if *index* is 2, *value2* is returned, and so on. Any number of values can follow *index* and the values can be of any type. For a value to be returned, *index* must be an integer between 1 and the number of values. If *index* is not between 1 and the number of values, ‘Choose’ returns the empty value.

Examples

```
Choose (3, "one", "two", "three", "four") → "three"
Choose (4, "five", 5.0, 3+2, 15/3, "six - one") → 5
Choose (1, True, False, 1, 0, "on", "off") → True
```

Related Functions

‘Member’ returns True if a value is a member of a specified list of values.

CODE 39

The ‘Code 39’ function is an external function that reads the information in a text cell and returns a PICT of the corresponding “Code 39” barcode into a picture cell as shown below.

The “Code 39” barcode handles any number of digits or uppercase letters as well as the percent sign (%), asterisk (*), plus sign (+), and minus sign (-).

This function is only available if the Informed BarCode plug-in is installed in your Plug-ins folder.

Examples

```
External("Code 39",PartNumber)
```

The above example tells Informed to look in the cell named “PartNumber” and return the value of that cell as a PICT of the corresponding “Code 39” barcode.

Related Functions

The ‘UPC A’ function returns a PICT of the “UPC A” barcode.

COLLAPSECOLUMN (value1, value2,...)

This function accepts any number of input parameters (either field or column cells), and outputs a column which includes all of the input values. Empty values are stripped from the output and the other values move up to fill empty rows. Blank text values are considered equivalent to empty values.

Examples

If the column cell ‘Prices’ has the values 10.00, 3.50, 11.25, an empty row, and 2.50

`CollapseColumn (Prices)` → a column with values `10.00, 3.50, 11.25, 2.50`

Related Functions

The ‘MakeColumn’ function.

COLLAPSELIST (value1, value2, ...)

This function accepts any number of input parameters(either field or column cells), and outputs a list which includes all of the input values. Empty values are stripped from the output. Blank text values are considered equivalent to empty values. The output list is appropriate for functions that accept variable numbers of inputs, such as SUM or MEAN.

Examples

If the column cell ‘Prices’ has the values 10.00, 3.50, 11.25, an empty row, and 2.50

`CollapseList (Prices)` → a list with values `10.00, 3.50, 11.25, 2.50`

Related Functions

The ‘MakeList’ function.

COLUMN

The ‘Column’ function returns the index of the current cell in its owning table. If the current cell is a field cell, ‘Column’ returns null.

Examples

If ‘Column’ is the calculation formula for the second column in a table

```
Column → 2
```

Related Functions

The ‘Row’ function returns the corresponding list of row numbers for the column cell that uses this function.

CONCAT (text1, text2, ...)

The ‘Concat’ function returns the concatenation of its parameters. Any number of text parameters can be included. The return value is one text value created by joining the text values of all the parameters.

Examples

```
Concat (3*5, " equals fifteen is ", 3*5 = 15) → "15 equals fifteen is True"
Concat ("'Mac' has ", Length ("Mac"), " chars.") → "'Mac' has 3 chars."
Concat ("(", True, " or ", False, ")") → "(True or False)"
```

If *firstName* contains the value “Barbara”, *initial* contains the value “J”, *lastName* contains the value “Pearce” and today’s date is May 1, 1991, then

```
Concat (firstName, " ", initial, " ", lastName) → "Barbara J Pearce"
Concat ("Today’s date: ", Today) → "Today’s date: 5/1/91"
```

Related Functions

The concatenation operator (&) concatenates its operands; "A" & "B" is equivalent to ‘Concat’ ("A", "B").

CONTAINS (text, subText)

The ‘Contains’ function is a boolean function. It returns the value True if *text* contains the group of characters in *subText*. The subset of characters in *text* must match the characters in *subText* exactly. If they do not, ‘Contains’ returns the value False. If *subtext* contains more characters than *text*, ‘Contains’ returns False.

Examples

```
Contains ("abcdefghijklmnopqrstuvwxyz", "M") → False
```

```
Contains ("one two three four", "four") → True
Contains ("one two three four", " four ") → False
Contains ("1 2 3 4", 2*2) → True
```

Related Functions

'Pos' returns the starting position of a specified group of characters in a text value. 'EndsWith' returns True if a text value ends with a specified group of characters. 'BeginsWith' returns True if a text value begins with a specified group of characters.

CONVERT (number, fromUnit, toUnit)

The 'Convert' function converts *number* from one unit of measure to another. *Number* is converted from the unit described in the text value *fromUnit* to the unit described in the text value *toUnit*. *FromUnit* and *toUnit* must be similarly derived from compatible unit classes. The following table lists the units recognized by Informed.

Units of Measure

Unit Class	Name	Description
mass	ct	Carat
	g	Gram
	grain	Grain
	lb	Avoirdupois pound
	lbt	Troy pound
	oz	Ounce
	ozt	Troy ounce
	slug	Slug
	t	Metric ton
	ton	Short ton
	tonUK	Long ton
length	chain	Chain
	fath	Fathom
	ft	International foot
	ftUS	Survey foot
	in	Inch
	m	Meter

	mi	International mile
	mil	Mil
	miUS	US statute mile
	nmi	Nautical mile
	pt	Point (1/72 in)
	rd	Rod
	yd	Yard
	µ	Micron
area (length*length	a	Are
	acre	Acre
volume (length*length*length	bbl	Barrel
	bu	Bushel
	cu	US cup
	fbm	Board foot
	gal	US gallon
	galC	Canadian gallon
	galUK	UK gallon
	l	litre
	ozFl	US fluid ounce
	ozUK	UK fluid ounce
	pk	Peck
	pnt	Pint
	qt	Quart
	tbsp	Tablespoon
	tsp	Teaspoon
pressure (mass/length*time*time)	atm	Atmosphere
	bar	Bar
	inHg	Inches of mercury

	inH2O	Inches of water
	mmHg	Millimeters of mercury
	Pa	Pascal
	psi	Pounds per square inch
	torr	Torr
time	d	Day
	h	Hour
	min	Minute
	s	Second
	yr	Year
speed (length/time	knot	Knot
	kph	Kilometers per hour
	mph	Miles per hour
angle	deg or °	Degrees
	rad or r	Radians
	grad	Grades
temperature	°C	Degree Celsius
	°F	Degree Fahrenheit
	°K	Degree Kelvin
	°R	Degree Rankine

You can convert a value of measure only between different units of the same class. However, in addition to converting between the units of each class listed in the previous table, you can also convert measurements between units that are derived by combining units from other classes. For example, not only can you convert a value from quarts to liters, you can also convert a value from cubic inches to cubic centimeters.

```
Convert (15, 'quart', 'liter')
Convert (15 'in * in * in', 'cm * cm * cm')
```

Since volume is a measurement of cubic length, you can convert any value between any units derived by multiplying three units of length.

Examples

```
Convert (-1, "ft/s", "in/s") → -12
Convert (5.5, "ft*ft*ft/s", "gal/min") → 2468.5714
Convert (2.3, "atm", "mmHg") → 1748
```

Related Functions

‘ConvertTo’ also converts a measurement value from one unit to another.

CONVERTTO (value, toUnit)

The ‘ConvertTo’ function converts the number/unit pair *value* to another unit of measure. *Value* is a text value that combines a number and its unit. The number is converted to the unit described in the text value *toUnit*. The number’s original unit and *toUnit* must be similarly derived from compatible base units. See ‘Convert’ for more information.

Examples

```
ConvertTo ("-1 ft/s", "in/s") → -12
ConvertTo ("100 knot", "mi/h") → 115.1
ConvertTo ("5 slug", "lb") → 160.87
```

Related Functions

‘Convert’ also converts a measurement value from one unit to another.

COS (number)

The ‘Cos’ function returns the cosine of *number*. *Number* must be an angle in radians. The return value is in the range -1 to 1.

Examples

```
Cos (0) → 1
Cos (Pi) → -1
Cos (9* π/2) → 0
```

Related Functions

‘ACos’ is the inverse of the ‘Cos’ function. ‘Sin’ and ‘Tan’ return the sine and tangent, respectively, of a number.

COUNT (value1, value2, ...)

The ‘Count’ function returns the number of non-empty values in its parameter list. Any number of values can appear in the parameter list.

Examples

If *salary* is empty and *time* contains the time value 8:50, then

```
Count ("", time) → 2
Count (True, 4*5, salary, time) → 3
Count (salary) → 0
Count (time) → 1
```

If *attendees* is a column cell with 10 rows, 7 of which have been calculated or filled in, then

```
Count (attendees) → 7
```

Related Functions

None.

CREATIONDATE

The ‘CreationDate’ function returns the creation date for the current record.

CREATIONTIME

The ‘CreationTime’ function returns the creation time for the current record.

DATEFORM (date, format)

The ‘DateForm’ function formats the date *date* using the format specified in the text parameter *format*. The resulting text value is returned. For an explanation of date formats, see “Date” in Chapter 1.

Examples

```
DateForm ("01/01/89", "MONTH DD") → "JANUARY 01"
DateForm ("Thursday, November 2, 1989", "MM-DD-YY")
→ "11-02-89"
DateForm ("Feb 27, 1990", "Dy, Month D, YYYY")
→ "Tue, February 27, 1990"
DateForm ("01/08/89", "M/D/YYYY") → "1/8/1989"
```

Related Functions

‘CharForm’ formats a text value, ‘NameForm’ formats a name, ‘NumForm’ formats a number, and ‘TimeForm’ formats a time value.

DAYABBREV (dayNumber)
DAYNAME (dayNumber)

The ‘DayAbbrev’ and ‘DayName’ functions return the name of the weekday identified by *dayNumber*. *DayNumber* should be an integer between 1 and 7. The first day of the week is Sunday, the second is Monday, and so on. The ‘DayName’ function returns the full name of the day with the first letter capitalized. The ‘DayAbbrev’ function returns an abbreviated name consisting of the first three characters of the day name with the first letter capitalized. ‘DayName’ and ‘DayAbbrev’ are often used with the ‘DayOfWeek’ function to obtain the name of the day in a date.

Examples

```
DayName (1) → "Sunday"
DayAbbrev (1) → "Sun"
DayName (1+3) → "Wednesday"
DayAbbrev (5) → "Thu"
```

If *birthDate* is a date containing the value May 30, 1960, then

```
DayName (DayOfWeek (birthDate)) → "Monday"
```

Related Functions

‘MonthAbbrev’ and ‘MonthName’ return the names of months. ‘DayOfWeek’ returns the number of the weekday in a date.

DAYOF (date)

The ‘DayOf’ function returns the number of the day represented in *date*. The number returned is an integer in the range 1 to 31.

Examples

```
DayOf (ToDate ("Saturday, April 25, 1964")) → 25
DayOf (ToDate ("09/12/89")) → 12
DayOf (ToDate ("12/26/89")) → 26
DayOf (ToDate ("26/12/89")) → 26
```

If *startDate* contains the date value Aug 20, 1999 and *duration* contains the value 20, then

```
DayOf (AddDays (startDate, duration)) → 9
```

Related Functions

‘MonthOf’ returns the number of the month in a date. ‘YearOf’ returns the number of the year in a date.

DAYOFWEEK (date)

The ‘DayOfWeek’ function returns an integer identifying the weekday represented in *date*. The integer returned is in the range 1 to 7. Sunday is the first day of the week, Monday is the second, and so on. ‘DayOfWeek’ is often used with the ‘DayName’ and ‘DayAbbrev’ functions to calculate the name of the weekday in a date.

Examples

```
DayOfWeek (ToDate ("June 17, 1992")) → 4
DayOfWeek (ToDate ("Mon Apr 20, 1987")) → 2
DayOfWeek (MakeDate (5, 6, 1889)) → 4
```

If the current year is 1990, then

```
DayOfWeek ("Nov 10") → 7
```

Related Functions

‘DayOfYear’ returns the day of the year in a date. ‘DayName’ and ‘DayAbbrev’ return the name of a weekday.

DAYOFYEAR (date)

The ‘DayOfYear’ function returns an integer identifying the day of year represented in *date*. The integer returned is in the range 1 to 366.

Examples

```
DayOfYear (ToDate ("01/01/90")) → 1
DayOfYear (ToDate ("Dec 31, 1984")) → 366
DayOfYear (ToDate ("Dec 31, 1985")) → 365
DayOfYear (MakeDate (8, 6, 1989)) → 159
```

If the current year is 1989, then

```
DayOfYear (ToDate ("Apr 25")) → 115
```

Related Functions

‘DayOfWeek’ returns the weekday in a date.

DBAL (cost, salvage, life, term[, factor])
DBALVALUE (cost, salvage, life, term[, factor])
MDBAL (cost, salvage, life, term[, factor])
MDBALVALUE (cost, salvage, life, term[, factor])

These functions calculate an asset’s depreciation amount and value. The ‘DBal’ and ‘DBalValue’ functions use the declining balance depreciation method. Under the declining balance method, the

depreciation amount for each period is a constant percentage of the depreciated value of the asset. Since the depreciated value of the asset declines over time, this creates larger depreciation amounts during the early years of the useful life of the asset. The percentage used to depreciate the asset is *factor* times the straight line depreciation rate. See SLD.

If *factor* is omitted, a value of 2.0 is used. *Cost* is the initial cost of the asset, *salvage* is the estimated value of the asset at the end of its useful life, and *life* is the number of time periods in the depreciation lifespan of the asset. *Term* identifies a particular depreciation period.

The 'DBal' function returns the depreciation amount for the period identified by *term*.

The 'DBalValue' function returns the depreciated value of the asset at the end of the period identified by *term*. *Term* should be between 0 and *life*. If *term* is equal to 0, 'DBalValue' returns *cost*. If *term* is equal to *life*, 'DBalValue' returns a value greater than *salvage*.

The 'MDBal' and 'MDBalValue' functions use the modified declining balance depreciation method. This method is a combination of the declining balance and the straight line depreciation methods. See also SLD. Depreciation is first calculated using the declining balance method. When the straight line depreciation amount on the remaining depreciable cost is less than the declining balance amount, a switch is made to the straight line depreciation method.

The 'MDBal' function returns the depreciation amount for the period identified by *term*.

The 'MDBalValue' function returns the depreciated value of the asset at the end of the period identified by *term*. *Term* should be between 0 and *life*. If *term* is equal to 0, 'MDBalValue' returns *cost*. If *term* is equal to *life*, 'MDBalValue' returns *salvage*.

Examples

If a machine costs $36,000 and is expected to last 12 years with a $1,500 salvage value, then the double rate (*factor* equal to 2) declining balance depreciation amounts and values for years 3 and 9 are:

```
DBal (36000, 1500, 12, 3) → 4166.66
DBal (36000, 1500, 12, 9, 2.0) → 1395.408
DBalValue (36000, 1500, 12, 3, 2.0) → 20833.33
DBalValue (36000, 1500, 12, 9) → 6977.04
```

and the corresponding modified declining balance depreciation amounts and values are:

```
MDBal (36000, 1500, 12, 3, 2.0) → 4167
MDBal (36000, 1500, 12, 9) → 1709
MDBalValue (36000, 1500, 12, 3, 2.0) → 20833
MDBalValue (36000, 1500, 12, 9) → 6629
```

Related Functions

'SLD' and 'SLDValue' return an asset's depreciation amount and depreciated value using the straight line method. 'SOYD' and 'SOYDValue' return an asset's depreciation amount and depreciated value using the sum-of-years method.

DELETE (text, start, count)

The ‘Delete’ function removes *count* characters from *text* and returns the resulting text value. Characters are removed starting at *start* characters from the beginning of *text*. If *count* is greater than the number of characters from *start* to the end of *text*, all characters from *start* onward are deleted.

Examples

```
Delete ("Hello there", 2, 5) → "Hthere"
Delete ("Goods and services", 6, 99) → "Goods"
Delete ("(413) 555-1234", 1, 6) → "555-1234"
```

Related Functions

‘Insert’ inserts a group of characters into a text value. ‘Replace’ replaces a group of characters in a text value with another group of characters.

ENDSWITH (text, subText)

The ‘EndsWith’ function is a boolean function. It returns the value True if *text* ends with the group of characters in *subText*. The ending characters of *text* must match the characters in *subText* exactly. If *text* does not end with *subText*, ‘EndsWith’ returns the value False. If *subtext* contains more characters than *text*, ‘EndsWith’ returns False.

Examples

```
EndsWith ("Have a nice day.", "day") → False
EndsWith ("tele", "television") → False
EndsWith ("television", "vision") → True
EndsWith ("Number of participants:  15", 3*5) → True
```

Related Functions

‘BeginsWith’ returns True if a text value begins with a specified group of characters. ‘Pos’ returns the starting position of a group of characters in a text value. ‘Contains’ returns True if a text value contains a specified group of characters.

EXP (number)
EXP1 (number)

The ‘Exp’ function returns the value of e raised to the power *number*. e is 2.7182818...; it is the base of the natural logarithm. Like ‘Exp,’ the ‘Exp1’ function returns the exponential of *number*; but ‘Exp1’ returns a more accurate result when *number* is close to zero.

Examples

```
Exp (1) → 2.7182818
Exp (-9.87) → 0.0000517
Exp (Ln (12.34)) → 12.34
```

```
Exp1 (0.0245) → 0.024803
Exp1 (-0.1) → -0.95162582
```

Related Functions

'Exp' and 'Exp1' are the inverses of the natural logarithm functions, 'Ln' and 'Ln1.' The exponentiation (^) operator raises an arbitrary number to a given power.

EXTERNAL (externalName, value1, value2, ...)

The 'External' function calls an external function. The name of the external function is given as the first parameter. The remaining parameters (any number) are passed to the external function.

Examples

The example below calls the external function named 'MyFunction' and passes the 'Now' function as the single parameter. The external function is evaluated and, in this example, returns the numeric value 15093.

```
External ("MyFunction", Now) → 15093
```

Related Functions

None.

FACT (number)

The 'Fact' functions returns the factorial of *number*. *Number* must be an integer greater than or equal to zero. The factorial of a number, *n*, is calculated as follows:

$\text{Fact}(n) = n \times (n-1) \times (n-2) \times \ldots \times 1 \quad \text{and} \quad \text{Fact}(0) = 1$

Examples

```
Fact (0) → 1
Fact (11) → 39916800
Fact (6) → 720
```

Related Functions

None.

FIRSTINITIALOF (name)

The 'FirstInitialOf' function returns a text value containing the initial from the first name in *name*. If *name* contains a first name, the capitalized first letter of the first name is returned. If *name* does not contain a first name, the empty value is returned.

Examples

```
FirstInitialOf (ToName ("Mr. Anthony Marner")) → "A."
FirstInitialOf (ToName ("John Jacob Joseph Jones")) → "J."
FirstInitialOf (ToName ("e e cummings")) → "E."
```

If *theName* contains the name value Dr. Anderson, MD, then

```
FirstInitialOf (theName)
```

returns the empty value.

Related Functions

‘LastInitialOf’ returns the last initial of a name. ‘MiddleInitialsOf’ returns all middle initials of a name. ‘FirstOf’ and ‘LastOf’ return the first name and last name, respectively, of a name. ‘MiddleOf’ returns a given middle name of a name. ‘PrefixOf’ and ‘SuffixOf’ return a given prefix or suffix of a name.

FIRSTOF (name)

The ‘FirstOf’ function returns a text value containing the first name from *name*. If *name* does not contain a first name, ‘FirstOf’ returns the empty value.

Examples

```
FirstOf (ToName ("Dr. Susan Applehoff")) → "Susan"
FirstOf (ToName ("J S Bach")) → "J"
```

If *herName* contains the name value Ms. Pearce, then

```
FirstOf (herName)
```

returns the empty value.

Related Functions

‘LastOf’ returns the last name of a name. ‘MiddleOf’ returns a given middle name of a name. ‘PrefixOf’ and ‘SuffixOf’ return a given prefix or suffix of a name. ‘FirstInitialOf’ and ‘LastInitialOf’ return the initial of the first or last name, respectively, of a name. ‘MiddleInitialsOf’ returns the middle initials of a name.

FLOOR (number)

The ‘Floor’ function returns the next integer less than or equal to *number.*

Examples

```
Floor (12.999999) → 12
Floor (-42.01) → -43
Floor (0.001) → 0
```

```
Floor (-2.00) → -2
```

Related Functions

'Ceiling' returns the next integer greater than or equal to a number.

FRAC (number)

The 'Frac' function returns the fractional part of *number*.

Examples

```
Frac (-12.25) → -0.25
Frac (42) → 0
Frac (-0.0001) → -0.0001
Frac (3/4) → 0.75
```

Related Functions

'Int' returns the integer part of a number.

FV (pv, pmt, rate, term[, Begin or End])

The 'FV' function returns the future value of an investment for a given present value, payment amount, interest rate, and term. See 'PV.'

GMEAN (number1, number2, ...)

The 'GMean' function returns the geometric mean of *number1*, *number2*, and so on. Any number of parameters can be specified. The geometric mean of *N* numbers is the product of the numbers to the $1/N^{th}$ power.

Numbers that contain the value 0 are ignored.

Examples

```
GMean (2) → 2
GMean (2, 4, 8) → 4
GMean (30, 33, 33.66, 41.74) → 34.34199
```

If *entries* is a column cell with three non-empty rows that contain the values 47.8, 0.0001, and 9998, then

```
GMean (entries) → 3.62894
```

Related Functions

'Mean' and 'Median' return the arithmetic mean and median, respectively, of a group of numbers. 'HMean' returns the harmonic mean of a group of numbers.

GRABTEXT

The ‘GrabText’ function is an external function that looks for a label in a text file and returns the text that follows that label. In order to use this function, you must have the Informed GrabText plug-in installed in your Plug-ins folder.

Examples

```
External("GrabText",":Report.txt","Comments")
```

The above example tells Informed to look in a text file called “Report.txt” and return the text that follows the label “Comments.” The default location for an external text file is in the Informed Preferences folder. This is referenced by the colon (":") preceding the file name. If the text file is located elsewhere you must specify the full path to its location.

Note The GrabText plug-in can also be used to calculate dynamic choice lists. See “Dynamic Choice Lists” in Chapter 1 of this manual for more details.

Related Functions

None.

HMEAN (number1, number2, ...)

The ‘HMean’ function returns the harmonic mean of *number1*, *number2*, and so on. Any number of parameters can be specified. The harmonic mean of *N* numbers is *N* divided by the sum of the reciprocals of the numbers.

$$HMean = \frac{\mathrm{N}}{\frac{1}{number\ 1} + \frac{1}{number\ 2} + \ldots + \frac{1}{number\ \mathrm{N}}}$$

Numbers that contain the value 0 are ignored.

Examples

```
HMean (2) → 2
HMean (30, 50) → 37.5
HMean (0.01, 100, 55.345) → 0.02999
```

If *entries* is a column cell with four non-empty rows that contain the values 2.0, 2.0, 3.0, and 0.5, then

```
HMean (entries) → 1.2
```

Related Functions

‘Mean’ and ‘Median’ return the arithmetic mean and median, respectively, of a group of numbers. ‘GMean’ returns the geometric mean of a group of numbers.

HOUROF (time)

The ‘HourOf’ function returns the number of the hour represented in *time*. The value returned by ‘HourOf’ is an integer between 0 and 23.

Examples

```
HourOf ("6:45:01 AM") → 6
HourOf ("23:00:01") → 23
HourOf ("00:59:59") → 0
HourOf ("1:00:00") → 1
HourOf ("10:34 PM") → 22
```

Related Functions

‘MinuteOf’ returns the number of the minute in a time value. ‘SecondOf’ returns the number of the second in a time value.

IFT (expr1, expr2)

The ‘IFT’ function is a shortcut to using the ‘If’ statement and returns different results based on different conditions. If *expr1* is True, then *expr2* is returned, otherwise the empty value is returned. The ‘IFT’ function can also be used as an operand in a formula. For more information on the ‘If’ statement, see “The If Statement” in Chapter 9, “Using Formulas.”

Examples

```
IFT (12>10, "Hello There") → "Hello There"
```

Related Functions

The ‘IFTE’ function.

IFTE (expr1, expr2, expr3)

The ‘IFTE’ function is a shortcut to using the ‘If-Then-Else’ statement and returns different results based on different conditions. If *expr1* is True, *expr2* is returned, otherwise *expr3* is returned. The ‘IFTE’ function can also be used as an operand in a formula. For more information on the ‘If-Then-Else’ statement, see “The If Statement” in Chapter 9, “Using Formulas.”

Examples

```
IFTE (12>10, "Hello There", "GoodBye") → "Hello There"
```

Related Functions

The ‘IFT’ function.

INSERT (text, start, subText)

The ‘Insert’ function inserts characters from *subText* into *text* and returns the resulting text value. Characters are inserted immediately after the character that is *start* characters from the beginning of *text*. If *start* is greater than the number of characters in *text*, ‘Insert’ appends *subText* to the end of *text* and returns the resulting text value.

Examples

```
Insert ("Received:", 1, "Omitted:") → "ROmitted:eceived:"
Insert ("pencils", 7, " and pens") → "pencils and pens"
Insert ("555-6945", 0, "(413) ") → "(413) 555-6945"
```

If *description* contains the value “The car is ” and *color* contains the value “blue”, then

```
Insert (description, Length (description) + 1, color)
→ "The car is blue"
```

Related Functions

‘Delete’ deletes a group of characters from a text value. ‘Replace’ replaces a group of characters in a text value with another group of characters.

INT (number)

The ‘Int’ function returns the integer part of *number*. No rounding is done; *number* is truncated to obtain its integer part.

Examples

```
Int (-12.75) → -12
Int (42) → 42
Int (-0.0001) → 0
Int (7/4) → 1
```

Related Functions

‘Frac’ returns the fractional part of a number.

INV (number)

The ‘Inv’ function returns the inverse of *number*. The inverse of a number, *n*, is 1/*n*. *Number* can be any number except 0. If *number* is 0, the empty value is returned.

Examples

```
Inv (1.0) → 1.0
Inv (23) → 0.04348
Inv (-0.003) → -333.333
Inv (3/4) → 1.333
```

Related Functions

None.

IPMT (pv, pmt, rate, term, per[, Begin or End])

The ‘IPMT’ function returns the interest payment amount on an investment for a given present value, payment amount, interest rate, and term. See ‘PV.’

ISEMPTY (value)

The ‘IsEmpty’ function returns a boolean value. *Value* can be any cell or formula. If *value* is empty, ‘IsEmpty’ returns True. When you fill out a form, a cell is empty until its value is entered or calculated.

Examples

If the text cell *destination* is blank and hasn’t been filled in or calculated, then

```
IsEmpty (destination) → True
```

Related Functions

None.

LASTDAYOFMONTH (date1)

The ‘LastDayOfMonth’ function returns a date value which is the last day of the month represented in the date value ‘date1.’

Examples

```
LastDayOfMonth (ToDate("Dec 7, 1996")) → December 31, 1996
```

LASTINITIALOF (name)

The ‘LastInitialOf’ function returns a text value containing the initial of the last name in *name*. If *name* contains a last name, the capitalized first letter of the last name is returned. If *name* does not contain a last name, the empty value is returned.

Examples

```
LastInitialOf (ToName ("Mr. Anthony Marner, MD")) → "M."
LastInitialOf (ToName ("John Jacob Joseph Jones")) → "J."
LastInitialOf (ToName ("e e cummings")) → "C."
```

If *theName* contains the name value Bill Smith, then

```
LastInitialOf (theName) → "S."
```

Related Functions

'FirstInitialOf' returns the first initial of a name. 'MiddleInitialsOf' returns all middle initials of a name. 'FirstOf' and 'LastOf' return the first name and last name, respectively, of a name. 'MiddleOf' returns a given middle name of a name. 'PrefixOf' and 'SuffixOf' return a given prefix or suffix of a name.

LASTOF (name)

The 'LastOf' function returns a text value containing the last name of *name*. If *name* doesn't contain a last name, 'LastOf' returns the empty value.

Examples

```
LastOf (ToName ("Dr. Susan Applehoff")) → "Applehoff"
LastOf (ToName ("J S Bach")) → "Bach
```

If *hisName* contains the name value Bill and *herName* contains the name value Brenda Wright, R.N., then

```
LastOf (herName) → "Wright"
LastOf (hisName) → "Bill"
```

Related Functions

'FirstOf' returns the first name of a name. 'MiddleOf' returns a given middle name of a name. 'PrefixOf' and 'SuffixOf' return a given prefix or suffix of a name.

LEFT (text, length)

The 'Left' function returns the first *length* characters in *text*. *Length* must be greater than or equal to 0. If *length* is greater than or equal to the number of characters in *text*, 'Left' returns *text*. If *length* is zero, a text value with no characters ("") is returned.

Examples

```
Left ("Macintosh", 3) → "Mac"
Left ("orange", 10) → "orange"
Left ("Forms Design, A Primer", 0) → ""
Left (2 = 2.0, 1) → "T"
```

If *theName* is a name value containing Cecil Featherstone-Haugh and *maxLength* contains the value 15, then

```
Left (LastOf (theName), maxLength) → "Featherstone-Ha"
```

Related Functions

'Right' returns the last characters in a text value. 'Mid' returns a group of characters from the middle of a text value.

LENGTH (text)

The ‘Length’ function returns the number of characters in *text*.

Examples

```
Length ("John Q. Public") → 14
Length ("") → 0
Length (True) → 4
Length (3*4) → 2
```

If *theName* is a name value containing “Cecil Featherstone-Haugh” and *maxLength* contains the value 15, then

```
Length (LastOf (theName)) <= maxLength → False
```

Related Functions

‘Trim’ removes extra spaces from the start and end of a text value.

LN (number)
LN1 (number)

The ‘Ln’ function returns the natural logarithm of *number.* The natural logarithm of a number is the base e logarithm of the number (e is 2.7182818...). *Number* must be greater than 0. Like ‘Ln,’ the ‘Ln1’ function returns the natural logarithm of *number*; but ‘Ln1’ returns a more accurate result when *number* is close to zero.

Examples

```
Ln (e^2) → 2
Ln1 (0.034) → 0.033434776
Ln (8976.46) → 9.1024
Ln1 (0.6) → 0.470003629
Ln (Exp (-3.4)) → -3.4
```

Related Functions

‘Ln’ and ‘Ln1’ are the inverses of the ‘Exp’ and ‘Exp1’ functions. ‘Log’ returns the base 10 logarithm of a number. ‘LogN’ returns a number’s logarithm for a given base.

LOG (number)

The ‘Log’ function returns the base 10 logarithm of *number. Number* must be greater than 0. If *number* is less than or equal to 0, the empty value is returned.

Examples

```
Log (1) → 0
Log (100) → 2
```

```
Log (0.0432) → -1.3645
Log (ALog (3.5)) → 3.5
```

Related Functions

'Log' is the inverse of the 'ALog' function. 'LogN' returns a number's logarithm for a given base. 'Ln' and 'Ln1' return the natural logarithm of a number.

LOGN (number, base)

The 'LogN' function returns the base *base* logarithm of *number*. Both *number* and *base* must be greater than 0. If *number* or *base* is less than or equal to 0, the empty value is returned.

Examples

```
LogN (1, 10) → 0
LogN (64, 4) → 3
LogN (0.0657, 6) → -1.51954
LogN (ALogN (3.5, 23.3), 23.3) → 3.5
```

Related Functions

'Log' is the inverse of the 'ALog' function. 'Log' returns the base 10 logarithm of a number. 'Ln' and 'Ln1' return the natural logarithm of a number.

LOWER (text)

The 'Lower' function converts all letters in *text* to lower case. The resulting text value is returned.

Examples

```
Lower ("Serial #: 146A889X") → "serial #: 146a889x"
Lower ("") → ""
Lower (True) → "true"
Lower ("tHiS iS hArD tO rEaD!!") → "this is hard to read!!"
```

Related Functions

'Upper' converts a text value to upper case. 'UpperFirst' converts the first letter of the first word of a text value to upper case. 'UpperWords' converts the first letter of all words of a text value to upper case.

MAKECOLUMN (value1, value2, ...)

The ‘MakeColumn’ function returns a column value that includes all inputs.

Examples

If ‘Names’ is a column cell with the values “Tom”, “Sally”, and “Glenn”

```
MakeColumn (Names, "Scott", "Joan")  →
```
a column with the values `"Tom", "Sally", "Glenn", "Scott", "Joan"`

Related Functions

The ‘CollapseColumn’ function.

MAKEDATE (day, month, year)

The ‘MakeDate’ function returns the date represented by the integers *day*, *month*, and *year*.

Examples

```
MakeDate (1, 1, 1981) → January 1, 1981
MakeDate (2*5, 5, 1651) → May 10, 1651
MakeDate (29, 2, 1988) → February 29, 1988
```

If *startDay* contains the value 3, *startMonth* contains the value 10, *startYear* contains the value 1980, and *term* contains the value 5, then

```
MakeDate (startDay, startMonth, startYear + term)
→ October 3, 1985
```

Related Functions

‘AddDays,’ ‘AddMonths,’ and ‘AddYears’ create a new date by adding a specified number of days, month, or years to a date. ‘ToDate’ converts a text value to a date.

Note When using the ‘MakeDate’ function be sure to add the complete year. For example, type 1, 1, 1993 not 1, 1, 93.

MAKELIST (value1, value2, ...)

This function accepts any number of input parameters (either field or column cells) and outputs a list that includes all of the input values. Empty values are preserved. The output list is appropriate for functions that accept a variable number of inputs, such as SUM or MEAN.

Examples

If 'Names' is a column cell with the values "Tom", "Sally", and "Glenn"

```
MakeList (Names, "Scott", "Joan")  →
    a list with the values "Tom", "Sally", "Glenn", "Scott", "Joan"
```

Related Functions

The 'CollapseList' function.

MAKETIME (hour, minute, second)

The 'MakeTime' function returns the time represented by the integers *hour*, *minute*, and *second*.

Examples

```
MakeTime (23, 59, 59) → 23:59:59
MakeTime (0, 0, 0) → 00:00:00
MakeTime (8, 34, 0) → 08:34:00
MakeTime (1, 1, 1) → 01:01:01
```

Related Functions

'AddHours,' 'AddMinutes,' and 'AddSeconds' create a new time by adding a specified number of hours, minutes, or seconds to a time.

MARGIN (cost, selling)

The 'Margin' function calculates the profit margin for an item. *Cost* is the original cost of the item and *selling* is the final selling price of the item. The selling price must not be 0. If *selling* price is 0, the empty value is returned. The profit margin for an item is calculated as follows:

$$\text{margin} = \frac{selling - cost}{selling}$$

Multiplying the return value of MARGIN by 100 gives the percentage profit margin for the item.

Examples

```
Margin (80.00, 100.00) → 0.20
Margin (3.45, 3.00) → -0.15
```

```
Margin (0, 2*2.5) → 1
Margin (2345.45, 2345.45) → 0
```

If *finalPrice* contains the value 5.00 and *cost* contains the value 5.50, then

```
Margin (cost, finalPrice) < 0 → True
```

Related Functions

'Markup' returns the markup for an item, given its cost and selling price.

MARKUP (cost, selling)

The 'Markup' function calculates the markup for an item. *Cost* is the original cost of the item and *selling* is the final selling price of the item. The cost must not be 0. If *cost* is 0, the empty value is returned. The markup for an item is calculated as follows:

Multiplying the return value of 'Markup' by 100 gives the percentage markup for the item.

Examples

```
Markup (80.00, 100.00) → 0.25
Markup (3.45, 3.00) → -0.13
Markup (2*2.5, 0) → -1
Markup (2345.45, 2345.45) → 0
```

If *finalPrice* contains the value 5.00, *cost* contains the value 4.00, and *minMarkup* contains the value 0.10, then

```
Markup (cost, finalPrice) > minMarkup → True
```

Related Functions

'Margin' returns the profit margin for an item, given its cost and selling price.

MAX (number1, number2, ...)

The 'Max' function returns the largest number in its list of parameters. Any number of parameters can be specified.

Examples

```
Max (-12, 0, 34.5, -23.43, 99) → 99
Max (1, 1.0, 2/2, -5+6) → 1
Max (-100, -99.5) → -99.5
```

Related Functions

'Min' returns the smallest number in a group of numbers. 'Median' returns the median number of a group of numbers.

MDBAL (cost, salvage, life, term[, factor])

The ‘MDBal’ function uses the modified declining balance method to calculate an asset’s depreciation allowance for a particular period. See ‘DBal.’

MDBALVALUE (cost, salvage, life, term[, factor])

The ‘MDBalValue’ function uses the modified declining balance method to calculate the book value of an asset after a specified number of depreciation periods. See ‘DBal.’

MEAN (number1, number2, ...)

The ‘Mean’ function returns the arithmetic mean of *number1*, *number2*, and so on. Any number of parameters can be specified. The arithmetic mean of *n* numbers is the sum of the numbers divided by *n*.

Examples

```
Mean (2) → 2
Mean (30, 50) → 40
Mean (0.01, 100, 55.345) → 51.785
Mean (2.0, -2.0, 0, -0.5) → -0.125
```

Related Functions

‘HMean’ and ‘Median’ return the harmonic mean and median, respectively, of a group of numbers. ‘GMean’ returns the geometric mean of a group of numbers.

MEDIAN (number1, number2, ...)

The ‘Median’ function returns the median from *number1*, *number2*, etc. Any number of parameters can be specified. The median of a group of numbers is the value in the group that has as many numbers less than it as it has numbers greater than it. If the number of values in the group is even, the median is the arithmetic mean of the two middle values.

Examples

```
Median (2, 3, -2) → 2
Median (3, -12, -1, -23, -87, 4, 5) → -1
Median (2, 18, 34, 20) → 19
Median (0, 10) → 5
```

Related Functions

‘HMean’ and ‘Mean’ return the harmonic mean and arithmetic mean, respectively, of a group of numbers. ‘GMean’ returns the geometric mean of a group of numbers.

MEMBER (target, value1, value2, ...)

The ‘Member’ function returns a boolean value. If *target* equals any of *value1*, *value2*, and so on, ‘Member’ returns True. Any number of values can be specified.

Examples

```
Member (True, 1=4, 5>8, 9<3) → False
Member ("5", 1*5, "6", "8") → True
Member (Pi, 3.1543, π, 4.3522, 3.1234) → True
```

If *answer* is 7, then

```
Member (answer, 1, 2, 3, 5, 7, 11, 13, 17) → True
```

Related Functions

‘Choose’ selects a value from a specified list of values.

MID (text, start, length)

The ‘Mid’ function returns a group of *length* characters from *text* starting at *start* characters from the beginning of *text*. *Length* must be greater than or equal to 0. If *length* is greater than or equal to the length of *text* minus *start*, ‘Mid’ returns all characters from *start* onward. If *length* is zero, a text value with no characters ("") is returned. If *start* is less than or equal to 0, or greater than the length of *text*, the empty value is returned.

Examples

```
Mid ("Macintosh", 1, 3) → "Mac"
Mid ("orange", 2, 10) → "range"
Mid ("Forms Design, A Primer", 12, 0) → ""
Mid (2 = 2.1, 1, 4) → "False"
```

If *phone* contains the value “(408) 555-1616”, then

```
Mid (Phone, Length (Phone) - 7, 3) → "555"
```

Related Functions

‘Right’ returns the last characters in a text value. ‘Left’ returns the first characters in a text value.

MIDDLECOUNT (name)

The ‘MiddleCount’ function returns the number of middle names in *name*. If *name* contains no middle names, ‘MiddleCount’ returns 0.

Examples

```
MiddleCount (ToName ("Harry S Truman")) → 1
MiddleCount (ToName ("J. S. Bach")) → 1
MiddleCount (ToName ("Dr. Wendy George")) → 0
MiddleCount (ToName ("Philip Arthur Charles George Windsor,
B.A."))  → 3
```

Related Functions

‘PrefixCount’ returns the number of prefixes in a name. ‘SuffixCount’ returns the number of suffixes in a name. ‘MiddleOf’ returns a given middle name of a name.

MIDDLEINITIALSOF (name)

The ‘MiddleInitialsOf’ function returns a text value containing the initials from the middle names in *name*. The value returned contains the capitalized first letters of all middle names. The returned initials are separated by spaces. If *name* doesn’t contain any middle names, the empty value is returned.

Examples

```
MiddleInitialsOf (ToName ("Mr. Anthony Frank Marner")) → "F."
MiddleInitialsOf (ToName ("John Jacob George Jones")) → "J. G."
MiddleInitialsOf (ToName ("e e cummings")) → "E."
```

If *theName* contains the name value Dr. Roger Anderson, MD, then

```
MiddleInitialsOf (theName)
```

returns the empty value.

Related Functions

‘LastInitialOf’ returns the last initial of a name. ‘FirstInitialOf’ returns the first initial of a name. ‘FirstOf’ and ‘LastOf’ return the first name and last name, respectively, of a name. ‘MiddleOf’ returns a given middle name of a name. ‘PrefixOf’ and ‘SuffixOf’ return a given prefix or suffix of a name.

MIDDLEOF (name, number)

The ‘MiddleOf’ function returns a text value containing a middle name of *name*. The value of *number* determines which middle name is returned. If *number* is 1, the first middle name is returned; if *number* is 2, the second middle name is returned, and so on. If *number* is 0, then all middle names,

separated by spaces, are returned. If *name* does not contain the middle name identified by *number*, ‘MiddleOf’ returns the empty value.

Examples

```
MiddleOf (ToName ("Ms. Susan Patricia Anderson"), 1) → "Patricia"
MiddleOf (ToName ("J S M Bach", 2) → "M"
MiddleOf (ToName ("Philip Arthur Charles George Windsor, B.Sc."), 3) → "George"
MiddleOf (ToName ("Philip Arthur Charles George Windsor"), 0)
    → "Arthur Charles George"
```

If *herName* contains the name value Sandra Susan Joan Wright, R.N., then

```
MiddleOf (herName, MiddleCount (herName)) → "Joan"
```

Related Functions

‘LastOf’ returns the last name of a name. ‘FirstOf’ returns the first name of a name. ‘PrefixOf’ and ‘SuffixOf’ return a given prefix or suffix of a name.

MIN (number1, number2, ...)

The ‘Min’ function returns the smallest number in its list of parameters. Any number of parameters can be specified.

Examples

```
Min (-12, 0, 34.5, -23.43, 99) → -23.43
Min (1, 1.0, 2/2, -5+6) → 1
Min (-100, -99.5) → -100
```

Related Functions

‘Max’ returns the largest number in a group of numbers. ‘Median’ returns the median number from a group of numbers.

MINUTEOF (time)

The ‘MinuteOf’ function returns the number of the minute represented in *time*. The number returned is an integer in the range 0 to 59.

Examples

```
MinuteOf ("6:45:01 AM") → 45
MinuteOf ("23:00:01") → 0
MinuteOf ("00:59:59") → 59
MinuteOf ("10:01 PM") → 1
```

Related Functions

'HourOf' returns the number of the hour in a time value. 'SecondOf' returns the number of the second in a time value.

MODIFYDATE

The 'ModifyDate' function returns the date the current record was last modified.

MODIFYTIME

The 'ModifyTime' function returns the time the current record was last modified.

MONTHABBREV (monthNumber)
MONTHNAME (monthNumber)

The 'MonthAbbrev' the 'MonthName' functions return the name of the month identified by *monthNumber*. *MonthNumber* must be an integer between 1 and 12. The 'MonthName' function returns the full name of the month with the first letter capitalized. The 'MonthAbbrev' function returns an abbreviated name consisting of the first three characters of the month name with the first letter capitalized. 'MonthName' and 'MonthAbbrev' are often used with the 'MonthOf' function to obtain the name of the month in a date.

Examples

```
MonthName (1) → "January"
MonthAbbrev (1) → "Jan"
MonthName (1+3) → "April"
MonthAbbrev (5) → "May"
```

If *birthDate* is a date containing the value December 30, 1965, then

```
MonthAbbrev (MonthOf (birthDate)) → "Dec"
```

Related Functions

'DayName' and 'DayAbbrev' return the name of a day. 'MonthOf' returns the number of the month in a date.

MONTHOF (date)

The 'MonthOf' function returns the number of the month represented in *date*. The number returned is an integer in the range 1 to 12.

Examples

```
MonthOf (ToDate ("Saturday, April 25, 1964")) → 4
MonthOf (ToDate ("09/12/89")) → 9
```

```
MonthOf (ToDate ("12/26/89")) → 12
MonthOf (ToDate ("26/01/89")) → 1
```

If *startDate* contains the date value Aug 20, 1999, then

```
MonthOf (startDate) → 8
```

Related Functions

'DayOf' returns the number of the day in a date. 'YearOf' returns the number of the year in a date.

NAMEFORM (name, format)

The 'NameForm' function formats the name *name* using the format specified in the text value *format*. The resulting text value is returned. The value of *format* describes the format of the name returned. The letters 'P,' 'F,' 'M,' 'L' and 'S' represent the name parts *prefix*, *first*, *middle*, *last*, and *suffix* in uppercase form. The corresponding lowercase letters represent the same parts in their abbreviated form. If *format* contains one of these letters, the corresponding name part will be included in the function result. The position of 'L' or 'l' in *format* determines if the surname comes first or last. If this letter is the first character of *format*, the surname comes first, otherwise the surname comes last. For an explanation of name formats, see *Name*.

Examples

```
NameForm ("Dr. Pat A. Smith", "PFL") → "Doctor Pat Smith"
NameForm ("Mr. William Jones, Esq.", "LpFMs") → "Jones, Mr. William, Esq."
NameForm ("Susan Anderson", "PfMLS") → "S. Anderson"
NameForm ("A. F. Houston", "FL") → "A. Houston"
```

Related Functions

'CharForm' formats a text value. 'DateForm' formats a date. 'NumForm' formats a number. 'TimeForm' formats a time value.

NOW

The 'Now' function returns the current time. The current time is returned whenever a cell calculation, default formula, or check formula containing the 'Now' function is calculated.

Examples

If the current time is 1:45:01 PM, then

```
Now → 13:45:01
HourOf (Now) → 13
MinuteOf (Now) → 45
```

Related Functions

The 'Today' function returns the current date.

NPV (rate, columnCell[, BEGIN or END])

The 'NPV' function calculates the net present value of future cash flows generated from an investment project. The present value of cash to be received in the future is the amount that, if it was invested today, would grow to equal the future sum at the future date. Net present value is the sum of the present values of all cash flows generated (or expended) during a project.

Rate is the interest rate that's used to calculate the present value of the future cash flows. The rows in the column cell called *columnCell* contain the projected cash flows for the project. The first row contains the initial cash flow (usually an expenditure), the second row contains the cash flow for the first period, and so on. Net outflow of cash during a period is represented by a negative number in the corresponding row and net inflow is represented by a positive number. There can be any number of rows in *columnCell*. Rows that contain empty values are not ignored; they're assumed to contribute no net cash flow for the corresponding period.

'NPV' calculates the net present value using the following formula, where *N* is the number of rows in *columnCell* and each *row* represents the value contained in the corresponding row of *columnCell*:

$$NPV = \frac{row\ 1}{(1 + rate)^{0}} + \frac{row\ 2}{(1 + rate)^{1}} + \ldots + \frac{row\ \mathrm{N}}{(1 + rate)^{\mathrm{N}-1}}$$

If the word BEGIN (or begin) appears as the last parameter to a function, payments are assumed to occur at the beginning of each period. If END (or end) appears as the last parameter, payments are assumed to occur at the end of each period. If neither BEGIN nor END appear, payments are assumed to occur at the end of each period.

Examples

If *discountRate* contains the value 0.10 (i.e. 10%) and *cashFlows* is a column cell with rows that contain the values -10000, 4000, 5000, and 6000, then

```
NPV (discountRate, cashFlows) → 2069.53
```

NUMBERTH (number)

The 'NumberTH' function returns a text value that consists of *number* followed by the appropriate ordinal suffix. *Number* should be an integer.

Examples

```
NumberTH (11) → "11th"
NumberTH (-32) → "-32nd"
NumberTH (21) → "21st"
```

Related Functions

'SpellNumber' spells out a number. 'SpellNumberTH' spells out the ordinal value of a number.

NUMFORM (number, format, currency)

The 'NumForm' function formats the number *number* using the format specified in the text value *format*. *Currency* is a boolean parameter. If it's value is True, 'NumForm' includes the currency symbol in the formatted number. The resulting text value is returned. The number is formatted according to the formatting rules and options for the number type. For more information, see "Number" in Chapter 1.

Examples

```
NumForm (128.89, "0", False) → "129"
NumForm (2754.7, "#,##0.00", False) → "2,754.70"
NumForm (2754.7, "#,##0.00", True) → "$2,754.70"
NumForm (-.56, "##,##0.00 Cr;##,##0.00 Dr", False) → "0.56 Dr"
NumForm (4381.99, "The price is #,##0.00", True) → "The price is $4,381.99"
```

Related Functions

'CharForm' formats a text value, 'DateForm' formats a date, 'NameForm' formats a name, and 'TimeForm' formats a time value.

ONEOF (boolean1, boolean2, ...)

The 'OneOf' function returns the boolean value True if exactly one of its boolean parameters is True. If no parameter is True or if more than one parameter is True, 'OneOf' returns False. The parameters can be any formula that returns a boolean value.

Examples

```
OneOf (True) → True
OneOf (1 + 2 = 3, False, 3 > 2) → False
OneOf (True, 0 > 1) → True
OneOf (5 < 3, 1 + 1 = 6, 8 - 3 = 4) → False
```

If *x* contains the value 1 and if *Cell1* is empty, then

```
OneOf (x ≠ 1, IsEmpty (Cell1)) → True
```

Related Functions

'AnyOf' returns True if any of its parameters are True. 'AllOf' returns True if all of its parameters are True.

PAGE

When used in a cell's formula, the 'Page' function returns the number of the page on which the cell appears. The number returned is in the range 1 to 99. Page numbering starts at the first numbered page in a form. The work and master pages are not included. 'Page' always returns the correct page number, even when pages are added and removed.

Examples

If Cell5 appears on the third page of a form, and is calculated as

```
Concat ("Page ", Page)
```

then the value of Cell5 will be "Page 3".

Related Functions

'PageCount' returns the total number of pages in a form. 'Part' returns the part number of a printed page. 'PartLabel' returns a text label according the part of a page currently printing. 'PartCount' returns the total number of parts for a page on which a cell appears.

PAGECOUNT

The 'PageCount' function returns the total number of pages in the form. The number returned is in the range 1 to 99. The work and master pages are not included in the count.

Examples

If a form contains 5 numbered pages, then for any cell on any page,

```
Concat ("Total pages: ", PageCount) → "Total pages: 5"
```

Related Functions

When used in a cell's formula, the 'Page' function returns the number of the page on which the cell appears. 'Part' returns the part number of a printed page. 'PartLabel' returns a text label according the part of a page currently printing. 'PartCount' returns the total number of parts for a page on which a cell appears.

PART

The 'Part' function returns the part number of the page currently being printed. When you print a form, Informed changes the value of 'Part' each time a different part of a page is printed. 'Part' is used to calculate a cell's value based on the part of a page. The number returned is in the range 1 to 99. For a description of the Multipart command and multipart pages, see *Multipart pages*.

Examples

If a page of a form contains three parts: the original, a customer copy, and an accounting copy, you can calculate a cell using the ‘Part’ function as shown below.

```
Choose (Part, "Original", "Customer Copy", "Accounting Copy")
```

Related Functions

‘PartLabel’ returns a text label according the part of a page currently printing. ‘PartCount’ returns the total number of parts for a page on which a cell appears. When used in a cell’s formula, the ‘Page’ function returns the number of the page on which the cell appears. ‘PageCount’ returns the total number of pages in a form.

PARTCOUNT

When used in a cell’s formula, the ‘PartCount’ function returns the total number of parts for the page containing the cell. The number returned is in the range 1 to 99. For a description of the Multipart command and multipart pages, see *Multipart pages*.

Examples

If the following formula is used to calculate a cell on a page that contains three parts, and the second part is currently printing, then

```
Concat ("Part ", Part, " of ", PartCount) → "Part 2 of 3"
```

Related Functions

‘Part’ returns the part number of a printed page. ‘PartLabel’ returns a text label according the part of a page currently printing. When used in a cell’s formula, the ‘Page’ function returns the number of the page on which the cell appears. ‘PageCount’ returns the total number of pages in a form.

PARTLABEL (label1, label2, label3, ...)

When used in a cell’s formula, the ‘PartLabel’ function returns the text value which corresponds to the current part that’s printing. If part 1 is printing, *label1* is returned; if part 2 is printing, *label2* is returned, and so on. ‘PartLabel’ is used to calculate a cell’s value based on the part of a page. For a description of the Multipart command and multipart pages, see *Multipart pages*.

Examples

If a page of a form contains three parts: the original, a customer copy, and an accounting copy, you can calculate a cell using the ‘PartLabel’ function as shown below.

```
PartLabel ("Original", "Customer Copy", "Accounting Copy")
```

Related Functions

'Part' returns the part number of a printed page. 'PartLabel' returns a text label according the part of a page currently printing. 'PartCount' returns the total number of parts for a page on which a cell appears. When used in a cell's formula, the 'Page' function returns the number of the page on which the cell appears. 'PageCount' returns the total number of pages in a form.

PBONDPRICE (face, rate, yield)

The 'PBondPrice' function returns the price of a perpetual bond with face value *face*, interest rate *rate*, and yield *yield*. See 'BondPrice.'

PBONDYIELD (price, face, rate)

The 'PBondYield' function returns the yield from a perpetual bond with face value *face*, interest rate *rate*, and price *price*. See 'BondPrice.'

PLATFORM

The 'Platform' function returns either "Windows" or "Mac OS."

PMT (pv, fv, rate, term[, BEGIN or END])

The 'Pmt' function returns the payment amount on an investment for a given present value, future value, interest rate and term. See 'PV.'

POS (subText, text)

The 'Pos' function returns the starting position of the first occurrence of *subText* in *text*. If no portion of *text* exactly matches the group of characters in *subText*, 'Pos' returns 0. The first character in *text* numbered 1, the second is numbered 2, and so on. If *subText* contains no characters or if the length of *subText* is greater than the length of *text*, 'Pos' returns 0.

Examples

```
Pos ("nice", "Have a nice day.") → 8
Pos (3*5, "Product Lifespan: 15 years") → 19
Pos ("for", "Informed for forms.") → 3
Pos ("television", "tele") → 0
```

If *adj* contains the value "Valued " and *salut* contains the value "Dear Customer", then

```
Insert (salut, Pos ("Customer", salut)-1, adj) → "Dear Valued Customer"
```

Related Functions

‘Contains’ returns True if a text value contains a specified group of characters. ‘Insert’ inserts a group of characters into a text value. ‘Replace’ replaces one group of characters in a text value with another group of characters. ‘Delete’ deletes a group of characters from a text value.

PPMT (pv, pmt, rate, term, per[, BEGIN or END])

The ‘PPMT’ function returns the principal payment amount on an investment for a given present value, payment amount, interest rate, term and period. See ‘PV.’

PREFIXCOUNT (name)

The ‘PrefixCount’ function returns the number of prefixes in *name*. If *name* contains no prefixes, ‘PrefixCount’ returns 0.

Examples

```
PrefixCount (ToName ("Harry S Truman")) → 0
PrefixCount (ToName ("Hon. Mr. John S. Edgemont")) → 2
PrefixCount (ToName ("Dr. Wendy George")) → 1
PrefixCount (ToName ("Mr. Philip Arthur Charles George Windsor, B.Sc.")) → 1
```

Related Functions

‘MiddleCount’ returns the number of middle names in a name. ‘SuffixCount’ returns the number of suffixes of a name.

PREFIXOF (name, number)

The ‘PrefixOf’ function returns a text value containing a prefix from *name*. The value of *number* determines which prefix is returned. If *number* is 1, the first prefix is returned; if *number* is 2, the second prefix is returned, and so on. If *number* is 0, then all prefixes, separated by spaces, are returned. If *name* doesn’t contain the prefix identified by *number*, ‘PrefixOf’ returns the empty value.

Examples

```
PrefixOf (ToName ("Ms. Susan Patricia Anderson"), 1) → "Ms."
PrefixOf (ToName ("Hon. Mr. John S. Edgemont"), 2) → "Mr."
```

If *herName* contains the name value Ms. Sandra Susan Joan Wright, R.N., then

```
PrefixOf (herName, PrefixCount (herName)) → "Ms."
```

Related Functions

‘LastOf’ returns the last name of a name. ‘FirstOf’ returns the first name of a name. ‘MiddleOf’ returns a given middle name of a name. ‘SuffixOf’ returns a given suffix of a name.

PRINCIPAL (pv, pmt, rate, term, per[, BEGIN or END])

The 'Principal' function returns the principal amount remaining on an investment for a given present value, payment amount, rate, term, and period. See 'PV.'

PRINTDATE

The 'PrintDate' function returns the date the current record was last printed.

PRINTTIME

The 'PrintTime' function returns the time the current record was last printed.

PSTDEV (number1, number2, ...)

The 'PSTDev' function returns the population standard deviation, σ, of *number1*, *number2*, and so on. 'PSTDev' calculates the true standard deviation for data sets constituting entire populations of interest. The value returned is calculated as follows:

$$\sigma = \sqrt{\sigma^2}$$

It's the positive square root of the population variance. See 'PVar.'

Any number of parameters can be specified.

Examples

```
PStDev (3.74, 3.89, 4.00, 3.68, 3.69) → 0.125
PStDev (9.3455) → 0
PStDev (40, 30, 50, 15, 5) → 16.31
```

If *ages* is a column cell with rows that contain the values 28, 31, 27, 29, 45, and 24, then

```
PStDev (ages) → 6.749
```

Related Functions

'PVar' calculates the population variance of a group of numbers. 'Var' and 'STDev' calculate the sample variance and sample standard deviation, respectively, of a group of numbers. 'Range' calculates the range of a group of numbers.

PV (fv, pmt, rate, term[, BEGIN or END])
FV (pv, pmt, rate, term[, BEGIN or END])
PMT (pv, fv, rate, term[, BEGIN or END])
RATE (pv, fv, pmt, term[, BEGIN or END])
TERM (pv, fv, pmt, rate[, BEGIN or END])

PPMT (pv, pmt, rate, term, per[, BEGIN or END])
IPMT (pv, pmt, rate, term, per[, BEGIN or END])
PRINCIPAL (pv, pmt, rate, term, per[, BEGIN or END])

These functions calculate common financial parameters used in constant payment cash flow analysis. They calculate present value (PV), future value (FV), periodic payment amount (Pmt), interest rate per period (Rate), investment term (Term), principal payment amount (PPMT), interest payment amount (IPmt), and the principal amount remaining on an investment (Principal).

The parameters *pv*, *fv*, *pmt*, *rate*, and *term* correspond to the functions with the same names. *Per* is a number between 1 and *term* that specifies a particular period. If the word *BEGIN* (or *begin*) appears as the last parameter to a function, payments are assumed to occur at the beginning of each period. If *END* (or *end*) appears as the last parameter, payments are assumed to occur at the end of each period. If neither *BEGIN* nor *END* appear, payments are assumed to occur at the end of each period. All of the above functions assume that cash received is represented by a positive number and cash paid out is represented by a negative number.

The 'PV,' 'FV,' 'Rate,' 'Pmt,' and 'Term' functions use the following equations to calculate their return values:

$$pv = term \times pmt + fv$$

(if rate = 0)

$$pv = pmt\left[\frac{1-(1+rate)^{-term}}{rate}\right] + fv\,[1+rate]^{-term}$$

(if rate <> 0)

The 'Principal' function calculates the principal amount remaining after *per* periods by finding the future value of the investment after *per* periods. The IPMT function calculates the interest payment amount for period *per* by multiplying the interest rate per period *rate* by the principal amount remaining at the previous period. The 'PPmt' function calculates the principal payment for period *per* by subtracting the interest payment for the period from the total payment amount for the period, *pmt*.

Examples

If a savings account contains $5,000 and earns 12% annual interest (1% monthly interest) and, at the beginning of each month for the next 24 months, deposits of $200 are made, the amount of money in the account after 24 months is:

```
FV (-5000, -200, 0.01, 24, BEGIN) → 11797.31
```

If a loan of $60,000 is made at an annual interest rate of 18% (monthly interest rate of 1.5%), and it must be paid off in 4 years with monthly payments made at the end of each month, the monthly payment amount is:

```
PMT (60000, 0, 0.015, 48, END) → -1762.5
```

and the interest payment amount and principal payment amount for the 5th period, and principal amount remaining after the 5th period are:

```
IPMT (60000, -1762.5, 0.015, 48, 5, END) → -847.07
PPMT (60000, -1762.5, 0.015, 48, 5, END) → -915.43
Principal (60000, -1762.5, 0.015, 48, 5, END) → 55556.17
```

If an investor purchases a business for $500,000 with the expectation of selling it for $1,000,000 in 5 years, and the business makes a $100,000 profit annually, the annual rate of return on the investment is:

```
Rate (-500000, 1000000, 100000, 5, END) → 0.3087
```

If the same investor wants a fixed annual return of 30% on the same investment, the required term (in years) before selling the business is:

```
Term (-500000, 1000000, 100000, 0.3) → 5.28
```

If the same investor wants a fixed annual return of 20% for a 5 year term on the same investment, the required purchase price is:

```
PV (1000000, 100000, 0.2, 5, END) → -700938.79
```

Related Functions

None.

PVAR (number1, number2, ...)

The ‘PVar’ function returns the population variance, σ^2, of *number1*, *number2*, and so on. ‘PVar’ calculates the true variance for data sets constituting entire populations of interest. The population variance of *N* numbers is calculated as follows:

$$\sigma^2 = \frac{(number\ 1 - \mu)^2 + (number\ 2 - \mu)^2 + \ldots + (number\ \mathrm{N} - \mu)^2}{\mathrm{N}}$$

where μ is the arithmetic mean of the numbers:

$$\mu = \frac{number\ 1 + number\ 2 + \ldots + number\ \mathrm{N}}{\mathrm{N}}$$

Any number of parameters can be specified.

Examples

```
PVar (3.74, 3.89, 4.00, 3.68, 3.69) → 0.0156
PVar (9.3455) → 0
PVar (40, 30, 50, 15, 5) → 266.0
```

If *ages* is a column cell with rows that contain the values 28, 31, 27, 29, 45, and 24, then

```
PVar (ages) → 45.556
```

Related Functions

‘PSTDev’ calculates the population standard deviation of a group of numbers. ‘Var’ and ‘STDev’ calculate the sample variance and sample standard deviation, respectively, of a group of numbers. ‘Range’ calculates the range of a group of numbers.

RANDOM (min, max)

The ‘Random’ function returns a random number between ‘min’ and ‘max.’ The result is a random floating point number that is greater than or equal to ‘min’ and less than ‘max.’ If you want to use the ‘Random’ function to generate a random integer within a range of values (inclusive), use this function:

```
Int (Random (min, max+1))
```

Examples

```
Random (1, 10) → 6.36652
Int (Random (1, 10)) → 9
```

Related Functions

None.

RANGE (number1, number2, ...)

The ‘Range’ function returns the difference between the highest number and the lowest number in *number1*, *number2*, and so on. Any number of parameters can be specified.

Examples

```
Range (-5, 0, 1) → 6
Range (65) → 0
Range (1.01, 0.98, 1.38, 1.24, 1.05, 0.96, 0.99) → 0.42
```

Related Functions

‘Var’ and ‘PVar’ calculate the sample variance and population variance, respectively, of a group of numbers. ‘STDev’ and ‘PSTDev’ calculate the sample standard deviation and population standard deviation, respectively, of a group of numbers.

RATE (pv, fv, pmt, term[, BEGIN or END])

The ‘Rate’ function calculates the interest rate per period on an investment for a given present value, future value, payment amount, and term. See ‘PV.’

REGISTEREDCOMPANY
REGISTEREDNAME

The ‘RegisteredName’ and ‘RegisteredCompany’ functions return the names of the person and company, respectively, to which the application being used is registered. By using these functions in default formulas, you can have Informed Filler automatically fill in the user’s name or company name on the form.

Examples

Suppose that your form contains a cell called *Initiator* which is the name and company of the user filling out the form. You might use the default formula shown below to automatically fill in the person’s name and company. Assume that the software being used is registered to a person by the name of ‘John Smith’ of the company ‘ABC Company.’

```
Concat (RegisteredName, " of ", RegisteredCompany) → "John Smith of ABC Company"
```

By using the ‘RegisteredName’ and ‘RegisteredCompany’ functions (instead of typing the names directly in the default formula), the form can be filled out by different people using different applications provided that each person is using an application that was personally registered to him or her.

REPEAT (text, count)

The ‘Repeat’ function returns a text value created by repeating *text*, *count* times. *Count* should be greater than zero. If *count* is zero, ‘Repeat’ returns a text value with no characters (“”).

Examples

```
Repeat ("-- ", 10) → "-- -- -- -- -- -- -- -- -- -- "
Repeat ("", 23) → ""
Repeat ("x", 3) → "xxx"
```

REPLACE (text, start, count, subText)

The ‘Replace’ function replaces *count* characters from *text* with *subText*. Characters are replaced starting at *start* characters from the beginning of *text*. If *count* is greater than the number of characters from *start* to the end of *text*, all characters from *start* onward are replaced. If *start* is greater than the number of characters in *text*, ‘Replace’ appends *subText* to *text* and returns the resulting text value. If *count* is zero, *text* is returned.

Examples

```
Replace ("Intramural", 0, 2, "Ex") → "Extramural"
Replace ("January", 4, 3, "itor") → "Janitory"
Replace ("Hello", 0, 99, "Goodbye") → "Goodbye"
```

If *firstName* contains the value “Walter” and *greeting* contains the value “Hello name,” then

```
Replace (greeting, 7, 4, firstName) → "Hello Walter,"
```

Related Functions

‘Delete’ deletes a group of characters from a text value. ‘Insert’ inserts a group of characters into a text value. ‘Pos’ returns the position of a specified group of character in a text value.

RIGHT (text, length)

The ‘Right’ function returns the last *length* characters from *text*. *Length* must be greater than or equal to 0. If *length* is greater than or equal to the number of characters in *text*, ‘Right’ returns *text*. If *length* is zero, a text value with no characters (“”) is returned.

Examples

```
Right ("Macintosh", 3) → "osh"
Right ("orange", 10) → "orange"
Right ("Forms Design, A Primer", 0) → ""
Right (2 = 2.0, 2) → "ue"
```

If *phoneNumber* contains the value "(413) 555-6732" and *suffixLength* contains the value 4, then

```
Right (phoneNumber, suffixLength) → "6732"
```

Related Functions

‘Left’ returns the first characters in a text value. ‘Mid’ returns a group of characters from the middle of a text value.

ROUND (number, decimals)

The ‘Round’ function rounds *number* to *decimals* decimal places. *Decimals* should be greater than or equal to zero. If *Decimals* is greater than the number of digits in the fractional part of *number*, ‘Round’ returns *number.*

Examples

```
Round (12.234, 0) → 12
Round (-12.235, 2) → -12.24
Round (-12.234, 2) → -12.23
Round (-4.1355, 3) → -4.135
Round (-8.8, 0) → -9
Round (4.1355, 5) → 4.1355
```

Related Functions

‘Ceiling’ returns the next integer greater than or equal to a number. ‘Floor’ returns the next integer less than or equal to a number.

ROW

The ‘Row’ function returns the corresponding list of row numbers for the column cell that uses this function (in its calculation, default, or check formula). Use the ‘Row’ function to fill in the row numbers in a column cell.

Examples

If the column cell called *Item* contains 5 rows, the ‘Row’ function will return the numbers 1, 2, 3, 4, and 5 if it’s used in the calculation, default, or check formula of the cell.

Related Functions

‘RowCount’ returns the total number of rows in a column cell.

ROWCOUNT (columnCell)

The ‘RowCount’ function returns the total number of rows in the column cell called *columnCell*. *ColumnCell* must the be name of a column cell.

Examples

If the column cell called Item contains 10 rows, then

```
RowCount (Item) → 10
```

Related Functions

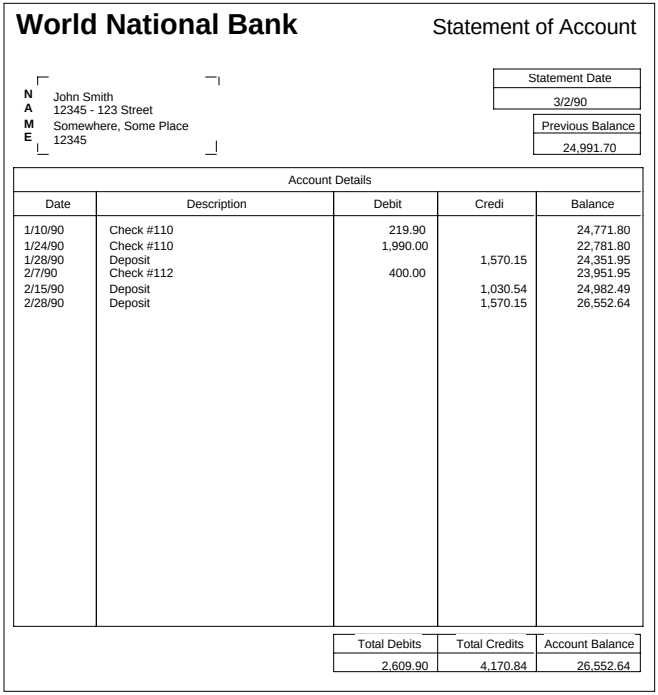
‘Row’ returns the row numbers of a column cell.

RUNNINGTOTAL (startValue, columnCell)

The ‘RunningTotal’ function automatically calculates the running total of a column cell on a form. The *columnCell* parameter must be a numeric column cell on your form. *StartValue* is the starting value of the running total. The result of ‘RunningTotal’ is a column of values which corresponds to the accumulative total of the column cell *columnCell* with an initial starting value of *startValue*.

Examples

Suppose that you’re creating a bank statement form to keep track of your bank account balance. This form might look like the one shown in the following illustration.



World National Bank Statement of Account

NAME
John Smith
12345 - 123 Street
Somewhere, Some Place
12345

Statement Date
3/2/90

Previous Balance
24,991.70

Account Details

Date	Description	Debit	Credi	Balance
1/10/90	Check #110	219.90		24,771.80
1/24/90	Check #110	1,990.00		22,781.80
1/28/90	Deposit		1,570.15	24,351.95
2/7/90	Check #112	400.00		23,951.95
2/15/90	Deposit		1,030.54	24,982.49
2/28/90	Deposit		1,570.15	26,552.64

Total Debits	Total Credits	Account Balance
2,609.90	4,170.84	26,552.64

Each row in the balance column is calculated by subtracting the debits in the current and previous rows of the debit column from the credits in the current and previous rows of the credit column, then adding that result to the previous statement balance. Assuming that the cells on this form are named ‘Previous Balance,’ ‘Credits,’ and ‘Debits,’ respectively, the formula shown below would correctly calculate the balance column.

```
RunningTotal (Previous Balance, Credits-Debits)
```

The columnCell parameter (Credits-Debits) calculates the net change in balance for each row in the table. The running total is calculated based on this columnar result plus the Previous Balance amount.

SECONDOF (time)

The ‘SecondOf’ function returns the number of the second represented in *time*. The number returned is an integer in the range 0 to 59.

Examples

```
SecondOf (ToTime ("6:45:01 AM")) → 01
SecondOf (ToTime ("23:00:00")) → 0
SecondOf (ToTime ("00:59:59")) → 59
SecondOf (ToTime ("10:01 PM")) → 0
```

Related Functions

'HourOf' returns the number of the hour in a time value. 'MinuteOf' returns the number of the minute in a time value.

SENDDATE

The 'SendDate' function returns the date the current record was last sent (mailed).

SENDTIME

The 'SendTime' function returns the time the current record was last sent (mailed).

SIGN (number)

The 'Sign' function returns 1 if *number* is positive, 0 if *number* is zero, or -1 if *number* is negative.

Examples

```
Sign (2+3) → 1
Sign (5-10) → -1
Sign (-3+3) → 0
Sign (-5^2) → 1
```

Related Functions

None.

SIGNED DATE

The 'Signed Date' function is an external function that extracts the signing date from a signed signature cell. This feature is useful for situations where information about a digital signature is required by other cells on the form, or by another application.

Although you configure the calculation that calls the 'Signed Date' function with Informed Designer, the Informed Filler user must have the Informed 'SignInfo' plug-in installed in his or her Plug-ins folder for the 'Signed Date' function to work.

Examples

```
External("Signed Date",Signature)
```

The above example tells Informed to look in a cell named "Signature" and extract the date that the cell was signed.

Related Functions

‘Signer Name’ returns the name of the person who signed the signature cell. ‘Signed Time’ returns the time that the signature cell was signed.

SIGNED TIME

The ‘Signed Time’ function is an external function that extracts the signing time from a signed signature cell. This feature is useful for situations where information about a digital signature is required by other cells on the form, or by another application.

Although you configure the calculation that calls the ‘Signed Time’ function with Informed Designer, the Informed Filler user must have the Informed ‘SignInfo’ plug-in installed in his or her Plug-ins folder for the ‘Signed Time’ function to work.

Examples

```
External("Signed Time",Signature)
```

The above example tells Informed to look in a cell named “Signature” and extract the time that the cell was signed.

Related Functions

‘Signer Name’ returns the name of the person who signed the signature cell. ‘Signed Date’ returns the date that the signature cell was signed.

SIGNER NAME

The ‘Signed Name’ function is an external function that extracts the name of the person who signs a signature cell. This feature is useful for situations where information about a digital signature is required by other cells on the form, or by another application.

Although you configure the calculation that calls the ‘Signer Name’ function with Informed Designer, the Informed Filler user must have the Informed ‘SignInfo’ plug-in installed in his or her Plug-ins folder for the ‘Signer Name’ function to work.

Examples

```
External("Signer Name",Signature)
```

The above example tells Informed to look in a cell named “Signature” and extract the name of the person who signed the cell.

Related Functions

‘Signed Date’ returns the date that the signature cell was signed. ‘Signed Time’ returns the time that the signature cell was signed.

SIN (number)

The ‘Sin’ function returns the sine of *number*. *Number* must be an angle in radians. The return value is in the range -1 to 1.

Examples

```
Sin (0) → 0
Sin (- π/2) → -1
Sin (Pi) → 0
Sin (9* π/2) → 1
```

Related Functions

‘ASin’ is the inverse of the ‘Sin’ function. ‘Cos’ and ‘Tan’ return the cosine and tangent, respectively, of a number.

SLD (cost, salvage, life)
SLDVALUE (cost, salvage, life, term)

These functions calculate an asset’s depreciation amount and value. They use the straight line depreciation method. This method spreads depreciation evenly over the useful life of the asset. *Cost* is the initial cost of the asset, *salvage* is the estimated value of the asset at the end of its useful life, and *life* is the number of time periods in the depreciation lifespan of the asset. The rate of depreciation is 1 over *life*.

The ‘SLD’ function returns the depreciation amount for a single period. It’s calculated as the wearing value of the asset (*cost* minus *salvage*) divided by the useful life of the asset (*life*).

The ‘SLDValue’ function returns the depreciated value of the asset at the end of the period identified by *term*. *Term* should be between 0 and *life*. If *term* is equal to 0, ‘SLDValue’ returns *cost*. If *term* is equal to *life*, ‘SLDValue’ returns *salvage*.

Examples

If a machine costs $5,000 and has an estimated value of $500 at the end of 5 years, then

```
SLD (5000, 500, 5) → 900
```

and the depreciated value of the machine after 3 years is:

```
SLDValue (5000, 500, 5, 3) → 2300
```

Related Functions

‘SOYD’ and ‘SOYDValue’ return an asset’s depreciation amount and depreciated value using the sum-of-years method. ‘DBal’ and ‘DBalValue’ return an asset’s depreciation amount and depreciated value using the declining balance method. ‘MDBal’ and ‘MDBalValue’ return an asset’s depreciation amount and depreciated value using the modified declining balance method.

SOYD (cost, salvage, life, term)
SOYDVALUE (cost, salvage, life, term)

These functions calculate an asset's depreciation amount and value. They use the sum-of-years depreciation method. This method creates larger depreciation amounts during the early years of the useful life of the asset. *Cost* is the initial cost of the asset, *salvage* is the estimated value of the asset at the end of its useful life, and *life* is the number of time periods in the depreciation lifespan of the asset. *Term* identifies a particular depreciation period.

The 'SOYD' function returns the depreciation amount for the period identified by *term*. It's calculated as a fraction of the wearing value of the asset (*cost* minus *salvage*). The denominator of the fraction is obtained by numbering the periods in *life* and adding. For example, if *life* is 5, the denominator is 1 + 2 + 3 + 4 + 5 = 15. The numerator for the first period is *life*. For each subsequent period the numerator is reduced by 1. If *life* is 5, the fractions for the 5 periods are: 5/15, 4/15, 3/15, 2/15, and 1/15.

The 'SOYDValue' function returns the depreciated value of the asset at the end of the period identified by *term. Term* should be between 0 and *life*. If *term* is equal to 0, 'SOYDValue' returns *cost*. If *term* is equal to *life*, 'SOYDValue' returns *salvage*.

Examples

If a machine costs $5,000 and has an estimated value of $500 at the end of 5 years, then the wearing value of the machine is $4,500 and the sum-of-years fraction for the fourth period is 2/15. The following formulas calculate the depreciation amount and the depreciated value for the first, second and fourth terms.

```
SOYD (5000, 500, 5, 1) → 1500
SOYD (5000, 500, 5, 2) → 1200
SOYD (5000, 500, 5, 4) → 600
SOYDValue (5000, 500, 5, 1) → 3500
SOYDValue (5000, 500, 5, 2) → 2300
SOYDValue (5000, 500, 5, 4) → 800
```

Related Functions

'SLD' and 'SLDValue' return an asset's depreciation amount and depreciated value using the straight line method. 'DBal' and 'DBalValue' return an asset's depreciation amount and depreciated value using the declining balance method. 'MDBal' and 'MDBalValue' return an asset's depreciation amount and depreciated value using the modified declining balance method.

SPELLCURRENCY (number, decimals)

The 'SpellCurrency' function returns a text value describing *number. Number* is described in monetary terms, using "dollars" and "cents". The number of decimal places to use in the description is given in *decimals*.

Examples

```
SpellCurrency (100, 2) → "One Hundred Dollars and 00 Cents"
SpellCurrency (45.99, 2) → "Forty Five Dollars and 99 Cents"
SpellCurrency (-10.01, 2) → "Ten Dollars and 01 Cents"
SpellCurrency (1000, 4) → "One Thousand Dollars and 00.00 Cents"
SpellCurrency (0, 2) → "No Dollars and 00 Cents"
```

Related Functions

‘SpellNumber’ spells out a number. ‘SpellNumberTH’ spells out the ordinal value of a number.

SPELLNUMBER (number)

The ‘SpellNumber’ function returns a text value describing *number*.

Examples

```
SpellNumber (101) → "One Hundred One"
SpellNumber (-54) → "Fifty Four"
SpellNumber (34.987) → "Thirty Four"
SpellNumber (-4.00) → "Four"
SpellNumber (0) → "Zero"
```

Related Functions

‘SpellCurrency’ spells out a number using “dollars” and “cents”. ‘SpellNumberTH’ spells out the ordinal value of a number.

SPELLNUMBERTH (number)

The ‘SpellNumberTH’ function returns a text value describing the ordinal value of *number*. *Number* should be an integer value. If *number* is not an integer, the fractional part is ignored.

Examples

```
SpellNumberTH (100) → "One Hundredth"
SpellNumberTH (1) → "First"
SpellNumberTH (-32.123) → "Thirty Second"
SpellNumberTH (0) → "Zeroth"
SpellNumberTH (1000000) → "One Millionth"
```

Related Functions

‘SpellCurrency’ spells out a number using “dollars” and “cents”. ‘SpellNumber’ spells out a number.

SQRT (number)

The ‘SQRT’ function returns the positive square root of *number*. *Number* must be greater than or equal to zero.

Examples

```
Sqrt (16) → 4
Sqrt (0.1) → 0.316227766
Sqrt (2) → 1.414213562
Sqrt (0) → 0
```

Related Functions

The exponentiation operator (^) raises a number to a given exponent.

STDEV (number1, number2, ...)

The ‘STDev’ function returns the sample standard deviation, *s*, of *number1*, *number2*, and so on. ‘STDev’ calculates the standard deviation for data sets constituting samples taken from populations of interest. The value returned is calculated as follows:

$$s = \sqrt{s^2}$$

It’s the positive square root of the sample variance. See ‘Var.’

Any number of parameters can be specified.

Examples

```
StDev (3.74, 3.89, 4.00, 3.68, 3.69) → 0.1398
StDev (9.3455) → 0
StDev (40, 30, 50, 15, 5) → 18.235
```

If *ages* is a column cell with rows that contain the values 28, 31, 27, 29, 45, and 24, then

```
StDev (ages) → 7.393691004
```

Related Functions

‘Var’ calculates the sample variance of a group of numbers. ‘PVar’ and ‘PSTDev’ calculate the population variance and population standard deviation, respectively, of a group of numbers. ‘Range’ calculates the range of a group of numbers.

SUFFIXCOUNT (name)

The ‘SuffixCount’ function returns the number of suffixes in *name*. If *name* contains no suffixes, ‘SuffixCount’ returns 0.

Examples

```
SuffixCount (ToName ("Antonio Salieri, B.Mus.")) → 1
SuffixCount (ToName ("Marsha")) → 0
SuffixCount (ToName ("A. E. Forrester, O.C., Q.C, M.Ed., B.Ed."))
→ 4
```

Related Functions

‘MiddleCount’ returns the number of middle names in a name. ‘PrefixCount’ returns the number of prefixes in a name. ‘SuffixOf’ returns a given suffix of a name.

SUFFIXOF (name, number)

The ‘SuffixOf’ function returns a text value containing a suffix of *name*. The value of *number* determines which suffix is returned. If *number* is 1, the first suffix is returned; if *number* is 2, the second suffix is returned, and so on. If *number* is 0, then all suffixes, separated by spaces, are returned. If *name* doesn’t contain the suffix identified by *number*, ‘SuffixOf’ returns the empty value.

Examples

```
SuffixOf (ToName ("Ms. Susan Patricia Anderson, O.C., Ph.D."), 1)
→ "O.C."
SuffixOf (ToName ("Hon. Mr. Justice S. Edge, L.L.B., B.Sc."), 2)
→ "B.Sc."
```

If *herName* contains the name value Ms. Sandra Susan Joan Wright, R.N., then

```
SuffixOf (herName, SuffixCount (herName)) → "R.N."
```

Related Functions

‘LastOf’ returns the last name of a name. ‘FirstOf’ returns the first name of a name. ‘MiddleOf’ returns a given middle name of a name. ‘PrefixOf’ returns a given prefix of a name.

SUM (number1, number2, ...)

The ‘Sum’ function returns the sum of the numbers in its list of parameters. Any number of parameters can be specified.

Examples

```
Sum (45)  → 45
Sum (-3, 0, 4.5) → 1.5
Sum (-2, -5, -9) → -16
```

If *receipts* is a column cell with 5 rows that contain the values 998.00, 750.00, 515.50, 222.95, and 800.05, then

```
Sum (receipts) → 3286.50
```

Related Functions

‘SumSQ’ returns the sum of squares of a group of numbers.

SUMSQ (number1, number2, ...)

The ‘SumSQ’ function returns the sum of the squares of the numbers in its list of parameters. Any number of parameters can be specified.

Examples

```
SumSq (45) → 2025
SumSq (-3, 0, 4.5) → 29.25
SumSq (-2, -5, -9) → 110
```

If *rangeData* is a column cell with 5 rows containing the values 2.345, 1.239, 0.045, -1.852, and -0.099, then

```
SumSq (rangeData) → 10.4759
```

Related Functions

‘Sum’ returns the sum of a group of numbers.

TAN (number)

The ‘Tan’ function returns the tangent of *number*. *Number* must be an angle in radians.

Examples

```
Tan (0) → 0
Tan (Pi) → 0
Tan (3* π/4) → -1
```

Related Functions

‘ATan’ is the inverse of the ‘Tan’ function. ‘Sin’ and ‘Cos’ return the sine and cosine, respectively, of a number.

TEMPLATEID

The ‘TemplateID’ function returns the unique template ID from the Template Information dialog box for the current template.

TEMPLATENAME

The ‘TemplateName’ function returns the template name from the Template Information dialog box for the current template.

TEMPLATEREVISION

The ‘TemplateRevision’ function returns the revision number from the Template Information dialog box for the current template.

TEMPLATESTATUS

The ‘TemplateStatus’ function returns the current status from the the Revision Options dialog box for the current template. The Informed Filler user can view this information by using the Revision Status command.

TERM (pv, fv, pmt, rate[, BEGIN or END])

The ‘Term’ function returns the term of an investment given the present value, future value, payment amount, and interest rate. See ‘PV.’

TIMEFORM (time, format)

The ‘TimeForm’ function formats the time value *time* using the format specified in the text value *format*. The resulting text value is returned. For an explanation of time formats, see *Time*.

Examples

```
TimeForm (ToTime ("4:6:59"), "0H:0M:0S") → "04:06:59"
TimeForm (ToTime ("23:34:56"), "H:MM AM") → "11:34 PM"
TimeForm (ToTime ("2:07 PM"), "H24:MM") → "14:07"
TimeForm (ToTime ("5:06:09"), "H:M:S") → "5:6:9"
```

Related Functions

‘CharForm’ formats a text value, ‘DateForm’ formats a date, ‘NameForm’ formats a name, and ‘NumForm’ formats a number.

TIMESPAN (date1, time1, date2, time2)

The ‘TimeSpan’ function returns the number of seconds between two date/time values. The example below shows the number of seconds between the current time today, and the same time one day later.

Examples

```
TimeSpan (Today, Now, AddDays(Today, 1), Now) → 86400
```

TOBOOLEAN (value)

The ‘ToBoolean’ function converts a text or numeric value to a boolean value. The following table shows which text values are converted to which boolean values. Upper and lower case is ignored when a text value is compared with those in the table.

Converting Text Values to Boolean Values

Text Value	Converts To
“True”	True
“T”	True
“False”	False
“F”	False
“Yes”	True
“Y”	True
“No”	False
“N”	False
“On”	True
“Off”	False

When you try to convert a text value not listed above to a boolean value, Informed will return the empty value.

When you convert a numeric value to a boolean value, the resulting boolean value will be True if the numeric value is non-zero; False otherwise. For more information about type conversion, see *Type conversion*. For a description of the boolean cell type, see *Boolean*.

Examples

```
ToBoolean ("on") → True
ToBoolean (75.35) → True
ToBoolean ("F") → False
ToBoolean ("75.35") → Empty value
```

Related Functions

The ‘ToText,’ ‘ToNumber,’ ‘ToDate,’ ‘ToTime,’ ‘ToName,’ ‘ToPicture,’ and ‘ToSignature’ functions convert values to text, number, date, time, name, picture, and signature values respectively.

TODATE (value)

The ‘ToDate’ function converts a text value to a date value. The *value* parameter can be any text value that represents a valid date. The format of *value* can be any date format that Informed recognizes. If *value* doesn’t represent a valid date, ‘ToDate’ returns the empty value. For more information about type conversion, see “Type Conversion.” For a description of the date cell type, see “Date.”

Examples

```
ToDate ("1/1/90") → January 1, 1990
DayOf (ToDate ("May 23 80")) → 23
ToDate ("abcdefg") → Empty value
```

If the current year is 1991, then

```
ToDate ("Oct 3") → October 3, 1991
```

Related Functions

The ‘ToText,’ ‘ToNumber,’ ‘ToBoolean,’ ‘ToTime,’ ‘ToName,’ ‘ToPicture,’ and ‘ToSignature’ functions convert values to text, number, boolean, time, name, picture, and signature values respectively.

TODAY

The ‘Today’ function returns the current date. The current date is returned whenever the cell calculation, default formula, or check formula containing the ‘Today’ function is calculated.

Examples

If the current date is September 15, 1991, then

```
Today  → September 15, 1991
MonthOf (Today) → 9
DayOf (Today) → 15
```

Related Functions

The ‘Now’ function returns the current time.

TOKENIZE (text, delimiter)

This function searches for the *delimiter* in the *text*, and breaks the input up into individual phrases. The phrases are collected and returned as a column value.

Examples

```
Tokenize ("123-456-789", "-") → {"123", "456", "789"}
Tokenize ("These are words", " ") → {"These", "are", "words"}
```

Related Functions

None.

TONAME (value)

The ‘ToName’ function converts a text value to a name value. The *value* parameter can be any text value that represents a valid name. The format of *value* can be any name format that Informed recognizes. If the ‘ToName’ function is used to calculate the value of a name cell, the format of the displayed name depends on the format of the name cell. If *value* doesn’t represent a valid name, ‘ToName’ returns the empty value.

For more information about type conversion, see “Type Conversion.” For a description of the name cell type and name formatting options, see “Name.”

Examples

```
ToName ("Mr. John Harold Smith") → Mister John Harold Smith
ToName ("Smith, John Harold") → John Harold Smith
FirstOf (ToName ("Dr. Susan Applehoff")) → "Susan"
```

Related Functions

The ‘ToText,’ ‘ToNumber,’ ‘ToBoolean,’ ‘ToTime,’ ‘ToDate,’ ‘ToPicture,’ and ‘ToSignature’ functions convert values to text, number, boolean, time, date, picture, and signature values respectively.

TONUMBER (value)

The ‘ToNumber’ function converts a text or boolean value to a numeric value. The *value* parameter can be any text value that represents a valid number, or any boolean value. ‘ToNumber’ converts the boolean values True and False to the numeric values 1 and 0 respectively.

If the ‘ToNumber’ function is used to calculate the value of a number cell, the format of the displayed number depends on the format of the number cell. If *value* is a text value that doesn’t represent a valid number, ‘ToNumber’ returns the empty value.

For more information about type conversion, see *Type conversion*. For a description of the number cell type and number formatting options, see *Number.*

Examples

```
ToNumber ("514.234") → 514.234
SpellNumber (ToNumber ("101")) → "One Hundred One"
```

Related Functions

The ‘ToText,’ ‘ToDate,’ ‘ToBoolean,’ ‘ToTime,’ ‘ToName,’ ‘ToPicture,’ and ‘ToSignature’ functions convert values to text, date, boolean, time, name, picture, and signature values respectively.

TOPICTURE (value)

The ‘ToPicture’ function converts the text in *value* to a picture value. The text value must be formatted according to Informed’s textual representation for pictures.

TOSIGNATURE (value)

The ‘ToSignature’ function converts the text in *value* to a digital signature value. The text value must be formatted according to Informed’s textual representation for signatures.

TOTEXT (value)

The ‘ToText’ function converts a number, name, date, time, boolean, picture, or signature value to a text value. The *value* parameter can be any value of any type. The following table summarizes how ‘ToText’ converts a value of each type to a text value.

Converting Values to Text

Original Type	Convert to Text
number	using the General number format
name	using all name parts in full, in order with surname first
date	using the date format “M/D/YY”
time	using the time format “H:MM:SS AM”
boolean	using “True” for True and “False” for False
picture	using an Informed-specific text format
signature	using an Informed-specific text format

For information about each cell type and the available formatting options, see *Cell types*. For more information about type conversion, see *Type conversion*.

Examples

```
ToText (ToDate ("December 18, 1990")) → "12/18/90"
ToText (ToName ("Smith, John Harold")) → "John Harold Smith"
```

If *Insured* is a boolean cell containing the value False, then

```
ToText (Insured) → "False"
```

Related Functions

The ‘ToDate,’ ‘ToNumber,’ ‘ToBoolean,’ ‘ToTime,’ ‘ToName,’ ‘ToPicture,’ and ‘ToSignature’ functions convert values to date, number, boolean, time, name, picture, and signature values respectively.

TOTIME (value)

The ‘ToTime’ function converts a text value to a time value. The *value* parameter can be any text value that represents a valid time. The format of *value* can be any time format that Informed recognizes. If *value* doesn’t represent a valid time, ‘ToTime’ returns the empty value. For more information about type conversion, see *Type conversion*. For a description of the time cell type, see *Time*.

Examples

```
ToTime ("7 15") → 7:15:00
ToTime ("2:55:12 PM") → 14:55:12
ToTime ("abcdefg") → Empty value
```

Related Functions

The ‘ToText,’ ‘ToNumber,’ ‘ToBoolean,’ ‘ToDate,’ ‘ToName,’ ‘ToPicture,’ and ‘ToSignature’ functions convert values to text, number, boolean, date, name, picture, and signature values respectively.

TRANSLITERATE (text, source, destination)

The ‘TransLiterate’ function translates the input text in *text* on a character by character basis. Normally the source and the destination will be of identical length, and will form a mapping from the input text. If a character in *source* exists in *text*, the character in *text* will be replaced with the character in *destination* at the same character position as the character in *source*. If the *source* is longer than *destination*, then any extra characters in *source* are mapped to nothing (that is, if they exist in *text*, they are deleted).

Examples

```
Transliterate ("encode me", "abcdefg", "ABCDEFG") → "EnCoDE mE"
```

Related Functions

The ‘Replace’ function.

TRIM (text)

The ‘Trim’ function removes all leading and trailing blanks from *text* and returns the resulting text value. If all the characters in *text* are blanks, ‘Trim’ returns a text value with no characters (“”).

Examples

```
Trim ("   Greetings   from    us.   ") → "Greetings   from    us."
Trim ("  Time:") → "Time:"
Trim ("Invoiced amount:     ") → "Invoiced amount:"
Trim ("") → ""
Trim ("          ") → ""
```

Related Functions

'Delete' deletes a group of characters from a text value. 'Replace' replaces a group of characters in a text value with another group of characters.

TRUNC (number, decimals)

'Trunc' truncates a number by deleting any significant digits after a specified number of decimal places.

Examples

```
Trunc (Cell1,2) → 123.4567, 123.45
```

Related Functions

None.

UPC A

The 'UPC A' function is an external function that reads the information in a text cell and returns a PICT of the corresponding "UPC A" bar code into a picture cell as shown below.

The number that you enter in the text cell to generate the "UPC A" barcode must contain 11 digits. This includes the digit to the left of the bar code and the 10 digits below the bar code. The digit to the right of the bar code is the check digit and is generated automatically.

This function is only available if the Informed BarCode plug-in is installed in your Plug-ins folder.

Examples

```
External("UPC A",PartNumber)
```

The above example tells Informed to look in the cell named "PartNumber" and return the value of that cell as a PICT of the corresponding "UPC A" barcode.

Related Functions

The 'Code 39' function returns a PICT of the "Code 39" barcode.

UPPER (text)

The ‘Upper’ function converts all letters in *text* to upper case. The resulting text value is returned.

Examples

```
Upper ("Serial #: 146a889x") → "SERIAL #: 146A889X"
Upper ("") → ""
Upper (True) → "TRUE"
Upper ("tHiS iS hArD tO rEaD!!") → "THIS IS HARD TO READ!!"
```

Related Functions

‘Lower’ converts a text value to lower case. ‘UpperFirst’ converts the first character of the first word of a text value to upper case. ‘UpperWords’ converts the first character of all words in a text value to upper case.

UPPERFIRST (text)

The ‘UpperFirst’ function converts the first character of each sentence in *text* to upper case, where a sentence is a number of words terminated by a period (.). The resulting text value is returned.

Examples

```
UpperFirst ("goods RECEIVED") → "Goods RECEIVED"
UpperFirst ("2. Inventory") → "2. Inventory"
UpperFirst ("x") → "X"
```

Related Functions

‘Upper’ converts a text value to upper case. ‘Lower’ converts a text value to lower case. ‘UpperWords’ converts the first character of all words in a text value to upper case.

UPPERWORDS (text)

The ‘UpperWords’ function converts the first character of each word in *text* to upper case. A word is any sequence of characters that starts *text* or that follows a space. The resulting text value is returned.

Examples

```
UpperWords ("goods RECEIVED") → "Goods RECEIVED"
UpperWords ("2. inventory") → "2. Inventory"
UpperWords ("x") → "X"
UpperWords ("John’s car is a 240se") → "John’s Car Is A 240se"
```

Related Functions

‘Upper’ converts a text value to upper case. ‘Lower’ converts a text value to lower case. ‘UpperFirst’ converts the first character in a text value to upper case.

USERNAME

The ‘UserName’ function returns a text value containing the user name as set by the Chooser desk accessory. For information about the Chooser desk accessory and its use, see your *Macintosh Owner’s Guide*.

Examples

If the current name in the Chooser desk accessory is “John Smith”, then

```
UserName  → "John Smith"
```

Related Functions

None.

VALIDCHOICE (value, cell)

The ‘ValidChoice’ function returns a boolean result. If *value* matches a choice in the list of choices for the cell called *cell*, then ‘ValidChoice’ returns True; False otherwise. Informed ignores upper and lower case when it compares *value* with each choice in the choices list.

You enter a cell’s list of choices using the Choices command in the Settings menu. For a complete explanation of choices and the Choices command, see *Choices*.

Examples

If *Terms* is a cell that has the choices, “Cash”, “On account”, “Net 30 days”, and “Net 60 days”, then

```
ValidChoice ("cash", Terms) → True
ValidChoice ("free", Terms) → False
```

The most common use of ‘ValidChoice’ is to check if the value entered in a cell is in that cell’s choice list. To do this, pass the name of the cell in both *value* and *cell*.

```
ValidChoice (Terms, Terms)
```

The value in Terms is compared with the choices for the same cell. If a match is found, ‘Valid-Choice’ returns True; False otherwise.

Related Functions

‘Choices’ returns a columnar text value containing the choices for a particular cell, one in each row.

VAR (number1, number2, ...)

The 'Var' function returns the sample variance, s^2, of *number1*, *number2*, and so on. VAR calculates the variance for data sets constituting samples from populations of interest. The population variance of *N* samples is calculated as follows:

$$s^2 = \frac{(number\ 1 - \overline{X})^2 + (number\ 2 - \overline{X})^2 + \ldots + (number\ \mathrm{N} - \overline{X})^2}{\mathrm{N} - 1}$$

where $\overline{X}$ is the arithmetic mean of the numbers:

$$\overline{X} = \frac{number\ 1 + number\ 2 + \ldots + number\ \mathrm{N}}{\mathrm{N}}$$

Any number of parameters can be specified.

Examples

```
Var (3.74, 3.89, 4.00, 3.68, 3.69) → 0.01955
Var (9.3455) → 0
Var (40, 30, 50, 15, 5) → 332.5
```

If *ages* is a column cell with rows that contain the values 28, 31, 27, 29, 45, and 24, then

```
Var (ages) → 54.666666667
```

Related Functions

'STDev' calculates the sample standard deviation of a group of numbers. 'PVar' and 'PSTDev' calculate the population variance and population standard deviation, respectively, of a group of numbers. 'Range' calculates the range of a group of numbers.

WEEKOFYEAR (date)

The 'WeekOfYear' function returns an integer describing the week of the year represented in *date*. The integer returned is in the range 1 to 53.

Examples

```
WeekOfYear (ToDate ("01/01/90")) → 1
WeekOfYear (ToDate ("Dec 31, 1984")) → 53
WeekOfYear (ToDate ("Dec 31, 1985")) → 53
WeekOfYear (MakeDate (8, 6, 1989)) → 23
```

If the current year is 1989, then

```
WeekOfYear (ToDate ("Apr 25")) → 17
```

Related Functions

'DayOfYear' returns the day of year in a date. 'DayOfWeek' returns the weekday of a date. 'Last-DayofMonth' returns a date value which is the last day of the month represented in a date.

WHICHMEMBER (target, value1, value2, ...)

The 'WhichMember' function tries to match "target" against each of the other parameters. If a successful match is found, 'WhichMember' returns the index of the first matched value. If no match is found, 'WhichMember' returns null.

Examples

```
WhichMember (True, 1=4, 5>8, 9<3) → NULL
WhichMember ("5", 1*5, "6", "8") → 1
WhichMember (Pi, 3.1543,  π, 4.3522, 3.1234) → 2
```

Related Functions

'Member' returns True if target equals any of value1, value2, and so on.

WITHIN (value, startValue, endValue)

The 'Within' function is a boolean function. It returns the value True if *value* is between *startValue* and *endValue*, inclusive. All three parameters to the 'Within' function must be the same type. They can be numbers, dates, times, text, or boolean values. If *value* is less than *startValue*, or if *value* is greater than *endValue*, then 'Within' returns the value False; otherwise, 'Within' returns the value True.

Informed uses the standard comparison operators to compare *value* with *startValue* and *endValue*. See *Comparison operators* for information about how each type of value is compared.

Examples

```
Within (-1, 99, 105) → False
Within (ToDate ("Oct. 3, 1989"), ToDate ("01/01/90"),
ToDate ("Oct. 5, 1990")) → False
Within (ToTime ("1:30 PM"), ToTime ("13:30:00"),
ToTime ("2:30 PM")) → True
Within ("actuary", "Sensible", "Tourist") → False
Within ("actuary", "actually", "sensible") → True
```

Related Functions

'Between' returns True if a value is between two other values, non-inclusive.

WORKDAYS (date1, date2, mask)

The ‘WorkDays’ function returns the number of working days between two dates *date1* and *date2*. *Mask* indicates which of the days of the week are working days, and which are days off. *Mask* must consist of the characters “SMTWTFS”, in that order, in either upper or lower case. Starting at Sunday, and going through Saturday, each character in *mask* indicates a work day with upper case and a day off with lower case. Thus, a normal Monday through Friday work week would be given as “sMTWTFs.”

Examples

```
WorkDays (ToDate("May 6, 1997"), ToDate("Jun 3, 1997"), "sMTWTFs") → 21
```

YEAROF (date)

The ‘YearOf’ function returns the year represented in *date*.

Examples

```
YearOf (ToDate ("Saturday, April 25, 1964")) → 1964
YearOf (ToDate ("09/12/89")) → 1989
YearOf (ToDate ("12/26/01")) → 1901
YearOf (ToDate ("26/01/1654")) → 1654
```

If *startDate* contains the date value Aug 20, 1999, then

```
YearOf (startDate) → 1999
```

Related Functions

‘DayOf’ returns the number of the day in a date. ‘MonthOf’ returns the number of the month in a date. ‘WeekOfYear’ returns the number of the week of the year in a date. ‘LastDayofMonth’ returns a date value which is the last day of the month represented in a date.