Formula Reference

FarPoint Spread™ for .NET
Version 4.0

FarPoint

Legal Notices

Unless otherwise noted, all names of companies, products, street addresses, and persons contained herein are part of a completely fictitious scenario or scenarios and are designed solely to document the use of a FarPoint Technologies, Inc., product.

 4.0.c

Distribution Restrictions

If you are using the trial or evaluation version of this product, then you may not distribute any of the files provided with the trial or evaluation version. The control DLLs and OCXs distributed with the retail product may be distributed by the user royalty free. Distribution by the user of any designers or design-time assistants (EXEs or DLLs), executables, source code, or license files (LICs) distributed by FarPoint as part of this product is prohibited. You shall not develop applications that provide an application programmable interface to the software or the software as modified. Redistribution by your users of FarPoint DLLs and OCXs, or your modified or wrapped version of our OCXs without the appropriate redistribution license from FarPoint is prohibited.

Table of Contents

Preface

Dedication

This documentation was written as a collaborative effort by employees at FarPoint. Developers, technical support personnel, testers, and technical writers all added their knowledge of how the product works and can best be applied. This documentation is a diverse collection that follows a tradition of extensive examples and code snippets to clarify the use and flexibility of our product. Because the world of software development and the computer-based economy are changing so rapidly, developers like you need a tool that is flexible and extendable, and you do not have a lot of time to learn how best to apply it. You cannot succeed without a well-supported and well-documented product.

Recognizing that there are various levels of readers, from beginners with .NET and those new to using components, to experienced Spread users and those who customize the tool greatly, we have tried our best to provide essential procedural documentation and tutorials to help you get started as well as detailed class library (API reference) documentation to let you customize the use of the product to the fullest extent.

We dedicate this documentation to the developers who can think beyond the traditional boundaries of spreadsheets and use this product to extend their creative and innovative ideas.

Getting Technical Support

If you have a technical question about this product, consult the following sources:

- Online help and other documentation files installed with the product.

  For instructions for accessing the online help and other documentation files, see "Finding Documentation".
- Product forum at http://www.clubfarpoint.com/Forums/

  If you are not a member of the forum, follow these instructions to become a member:

  Go to the forum at http://www.clubfarpoint.com/Forums/.

  At the top of the page, click the Register link.

  Read the Registration Agreement Terms and if you agree to the terms, click the appropriate link at the bottom of the page.

  Complete the Registration Information (required), Profile Information (optional), and Preferences (optional) sections, then click Submit.
- Knowledge base available on the FarPoint Web site.
- Code samples installed with the product.

If you cannot find the answer using these sources, please contact FarPoint Technical Support using one of these methods depending on your technical support package:

Web site	www.fpoint.com
E-mail	fpsupport@fpoint.com
Fax	(919) 460-7606
FTP site	ftp://ftp.fpoint.com
Phone	(919) 460-1887

FarPoint Technical Support is available between the hours of 9:00 a.m. and 5:30 p.m. Eastern time, Monday through Friday

For information about the our technical support packages, visit our Web site's Technical Support area (http://www.fpoint.com/support/supportplan.html), or contact our Sales department at (800) 645-5913 or fpsales@fpoint.com.

Contacting FarPoint

If you would like to find out more about FarPoint and our products, contact our Sales department using one of the following methods:

Web site	http://www.fpoint.com
E-mail	fpsales@fpoint.com
Phone	(800) 645-5913
Fax	(919) 460-7606
Mail	FarPoint Technologies, Inc. 808 Aviation Parkway Suite 1300 Morrisville, NC 27560 USA

Chapter 1 Formula Overview

Formulas in FarPoint Spread .NET include operators and functions that follow certain syntax rules and allow you to perform a range of calculations. These topics introduce the concepts you need to make full use of the built-in functions and extensive capability of formulas:

- What is a Formula?
- Cell References in a Formula
- Sheet References in a Formula
- Operators in a Formula
- Functions in a Formula
- Arrays in a Formula
- Data Types Using Formulas
- Custom Functions in Formulas
- Custom Names in Formulas
- Resultant Error Values

For a complete reference of all the built-in functions, refer to *Formula Functions* on page 19.

What is a Formula?

Formulas can consist of values, operators, and functions. Data can be from other cells, a combination of data in another cell and hard-coded data (for example, A1 + 2), or simply hard-coded data (for example, SUM(4,5)). Formulas can perform mathematical operations, such as addition and multiplication, on values in other cells or they can compare values in other cells. Formulas can refer to cells in the same sheet by their absolute cell location or relative to the cell with the formula in it; they can refer to individual cells or a range of contiguous cells. If the values in the referenced cells change, then the value of the formula cell changes.

Formulas can be made up of:

- cell references and cell ranges (notation indicating address of cell or cells)
- operators (that act on one or two values)
- built-in functions (predefined formulas) or user-defined functions
- user-defined names (for functions, constants, or cell references)
- constants or array of constants (values you enter that do not change)

Sample Formula

Use the SetFormula method in the Column, Row, or Cell class for specifying the formula for a column, row, or individual cell respectively. Returning the value of the Formula property for these classes provides a string containing the written expression of the formula, for example, SUM(A1:B1).

In code the setting of a formula would look something like this in Visual Basic .NET (for illustration purposes only):

```
FpSpread1.ActiveSheet.Cells(2, 0).Formula = "SUM(A1:A10)"
```

or something like this in C#:

```
fpSpread1.ActiveSheet.Cells[2, 0].Formula = "SUM(A1:A10)";
```

and if added in the cell by the end user:

```
=SUM(A1:A10)
```

In this documentation, where examples are shown, the formula appears as:

```
SUM(A1:A10)
```

or

`SUM(3,4,5)` gives the result `12`

to express that the result of the formula would display the value of 12 in the cell.

Keep these ways of expressing a formula in mind when looking at the examples in this documentation. Refer to the specific product Assembly Reference for more details on the Formula property for that product and the exact code syntax to use. Refer to the Developers Guide for that product to find more examples and discussion of formulas.

Cell References in a Formula

A formula can refer to constant values or cell references. If a value in any of the referenced cells changes, the result of the formula changes. If you use constant values in the formula instead of references to the cells, the result changes only if you modify the formula (or values in the formula).

If a new row is added right before or after a cell range in a formula then the range does not include the new row.

For more information on setting the reference style for a cell, refer to the ReferenceStyle enumeration in the product's Assembly Reference (or online help) and the ReferenceStyle property for the specific sheet (SheetView object).

Note: Remember that although most of Spread uses zero-based references to rows and columns, in the creation of formulas you must use one-based references. The column and row numbers start at one (1), not zero (0).

For more information on cell references that include sheet names, refer to *Sheet References in a Formula* on page 4.

A1 (Letter-Number) Notation

Each cell can be referenced by a combination of its column letter (A through Z, then AA to ZZ, AAA to ZZZ, etc.) and row number (1 and beyond) for a total of 2,147,483,648 rows and columns. For example, D50 refers to the cell at the intersection of column D and row 50. To refer to a range of cells, enter the reference for the cell in the upper-left corner of the range, a colon (:), and then the reference to the cell in the lower-right corner of the range.

R1C1 (Number-Number) Notation

Each cell can be referenced by its row and column number by preceding each by the letter "R" for row and the letter "C" for column. For example R1C3 is the cell in the first row and third column.

A1 Cell Ref.	R1C1 Cell Ref.	Description
B12	R12C2	Cell in the second column (column B) and twelfth row (row 12)
D14:D48	R14C4:R48C4	The range of cells in the fourth column (column D) and in rows 14 through 48
E16:H16	R16C5:R16C8	The range of cells in the sixteenth row (row 16) in the fifth through the eighth column (columns E through H)
A25:E70	R25C1:R70C5	The range of cells in the first five columns (column A through E) and rows 25 through 70

Relative and Absolute

A relative cell reference is a reference to a cell relative to the position of the cell with the formula. An absolute reference is a cell reference that always refers to a cell by its exact location in the sheet and not with reference to the present cell.

Relative references automatically adjust when you copy them and absolute references do not. The FpSpread control can use absolute or relative cell references. You can define the cell reference style for each sheet by using the ReferenceStyle property. The formula cannot contain both absolute and relative row or column references. The following table contains examples of valid relative cell references in formulas

Function	Description
SUM(A1:A10)	Sums rows 1 through 10 in the first column
PI( )*C6	Multiplies pi times the value in cell C6
(A1 + B1) * C1	Adds the values in the first two cells and multiplies the result by the value in the third cell
IF(A1>5, A1*2, A1*3)	Checks if the contents of cell A1 are greater than 5, and if so, multiplies the contents of cell A1 by 2, or else multiplies the contents of cell A1 by 3

For A1 (Letter-Number) Notation, use a dollar sign ($) preceding the row or column (or both) to indicate an absolute reference. For example

$A$1	absolute first column, absolute first row
$A1	absolute first column, relative row plus one
A$1	relative column plus one, absolute first row
A1	relative column plus one, relative row plus one

For R1C1 (Number-Number) Notation, use brackets [ ] around the row or column number (or both) to indicate a relative reference. For example

R1C1	absolute first row, absolute first column
R1C[1]	absolute first row, relative column plus one
R[1]C1	relative row plus one, absolute first column
R[1]C[1]	relative row plus one, relative column plus one
R[-1]C[-1]	relative row minus one, relative column minus one

In this notation, the number inside the brackets is an offset from the current cell. This number may be a negative or positive integer or zero. Leaving off the offset entirely is short hand way of indicating a zero offset. So,

RC2 is equivalent to R[0]C2

R[3]C is equivalent to R[3]C[0]

Scope of Cell References

References to cells within a sheet are handled as described in this documentation. When a cell is referenced that is beyond the dimensions of the sheet, the cell is still evaluated, but the result is a #REF! error value. For example, if the sheet has less than 20 columns and rows, then the function DDB(B20,1000,10,1) evaluates to DDB(#REF!,1000,10,1), which evaluates to #REF!

Spread.NET does not support Excel's reference operators (for example range, intersection, union) in formulas. However, Spread .NET does support the #NULL! constant in formulas. It does support reading the #NULL! value from Excel files. For more information about what is supported on importing from and exporting to Excel, refer to the Import and Export Reference for the particular Spread product you are using.

Sheet References in a Formula

A formula can have references to cells on the same sheet or to cells on other sheets, as well as ranges of cells on sheets.

In the examples shown below, we use A1 (Letter-Number) notation for the cell reference, but the same would be valid for R1C1 (Number-Number) notation. Simply precede the cell reference, regardless of the style, with the sheet name as described here.

Cross-Sheet Referencing

When a reference to a cell includes a reference to a cell on another sheet, this is called cross-sheet referencing.

An example of cross-sheet referencing in a formula that uses the addition operator would be:

```
(FirstRoundData!A2 + SecondRoundData!A2)
```

where the name of one sheet is "FirstRoundData" and the name of another sheet is "SecondRoundData". Sheet names precede the cell reference with the name of the sheet followed by an exclamation point (!). This formula could be on any sheet in the Spread since it explicitly names the sheets of each of the cells as operands. This example adds the values in the cell A2 on two different sheets. By making the sheet name explicit there is no confusion as to which cell A2 is meant. If you do not include the sheet name, the current sheet (in which the formula exists) is used. If the formula in the above example was on the SecondRoundData page, then the formula could be written as:

```
(FirstRoundData!A2 + A2)
```

It might be less confusing to put the cell on the current page first, as in:

```
(A2 + FirstRoundData!A2)
```

Sheet Naming

As long as the sheet name conforms to normal variable name rules (with the first character being a letter or an underscore and the remaining characters being letters, digits, or underscores) then the formula can use just the sheet name followed by the exclamation point. Otherwise, the sheet name needs to be enclosed in single quotes. If the sheet name itself contains a single quote, then use two single quotes in the formula. For example, if the name of the sheet includes a single quote (or apostrophe) as in these names for sales of a given month, then a reference to the sheet would look like this in a formula:

```
('November''s Sales'!A2 + 'December''s Sales'!A2)
```

with two single quotes (or apostrophes) before the s. If the sheet name has a space, use single quotes around the sheet name. In the following example the sheet name is East Coast Sales.

```
('East Coast Sales'!A2 + 'West Coast Sales'!A1)
```

If you have a quote in the name of the sheet, you need to add the delimiter that is required for that language. For instance, in C#, if the sheet name is "Zippy" Sales, where the quotes are part of the sheet name, a formula that includes a reference to this sheet might look like this:

```
('/"Zippy/" Sales'!A2 + 'West Coast Sales'!A1)
```

where a single quotes surrounds the entire sheet name and the backslash (/) delimiter precedes the quotes. For Visual Basic, you would use two double quote characters as in:

```
('""Zippy"" Sales'!A2 + 'West Coast Sales'!A1)
```

Using Ranges in Sheet References

For cross-sheet referencing of a range of cells in another page, precede the range with the sheet name. For example:

```
SUM(SecondRoundData!A2:A10)
```

This adds the values in cells A2 to A10 of the sheet named SecondRoundData. There is no reason to include the sheet name in the second half of the range reference since the cells are on the same sheet. You cannot specify two different sheets in a range; a range of cells is only on a particular sheet, not between sheets.

Operators in a Formula

The following table lists the available operators. For each operator, an example is given of the syntax of using a literal value as well as a cell reference. The type of value returned is given for each type of operator.

Type of Operator		**Example Syntax**		**Result**
Operator	Description	Literal & Literal	Cell Ref & Literal	Type Returned
Binary Operators				
+	Add	5 + 3	A1 + 3	double
–	Subtract	5 – 3	A1 – 3	double
*	Multiply	5 * 3	A1 * 3	double
/	Divide	5 / 3	A1 / 3	double
^	Exponent	5 ^ 3	A1 ^ 3	double
&	Concatenate	"F" & "p"	A1 & "p"	string
=	Equal		A1 <> 3	boolean
< >	Not Equal		A1 = 3	boolean
<	Less Than		A1 <3	boolean
>	Greater Than		A1 > 3	boolean
<=	Less Than Or Equal		A1 <= 3	boolean
>=	Greater Than Or Equal		A1 >= 3	boolean
Unary Operators				
-	Negate	-(5/3)	-(A1/3)	double
+	Plus	+(5/3)	+(A1/3)	double
%	Percent	(5/3)%	(A1/3)%	double

Operators specify the type of calculation that you want to perform on the elements of a formula. Most of the operators return double-precision floating point values for mathematical operations and boolean (or logical) values for comparison operators.

In Spread, all arithmetic operators (including the unary +) check their arguments and return a #VALUE error if any of the arguments are strings that can not be converted to a number. This is mathematically correct behavior and can not be overridden. For example, the three formulas +B5 and 0+B5 and --B5 should all produce the same result and, in Spread, they do.

Because more than one operator may be used in a formula, so be sure you understand the *Order of Precedence* on page 7.

The mathematical operators and unary operators may also be used with date-time and time-span values, as summarized in *Using Operators with Dates and Times* on page 7.

Order of Precedence

When there are several operators in a formula, the formula performs the operations in a specific order. The formula is parsed from left to right, according to a specific order for each operator or function in the formula. You can prioritize the order of operations by using parentheses in the formula.

If you combine several operators in a single formula, the operations are performed in the order shown in the following table. Unary operations precede binary operations. If a formula contains operators with the same precedence, the operators are evaluated from left to right. To change the order of evaluation, enclose the part of the formula to be calculated first in parentheses; this has the highest precedence. Where the order of precedence is the same for two operators, the formula is evaluated from left to right.

Order of Precedence from Highest to Lowest

Operator	**Description**
left to right	Direction
( )	Parentheses (for grouping)
–	Negate
+	Plus
%	Percent
^	Exponent
* and /	Multiply and Divide
+ and -	Add and Subtract
&	Concatenate
=, <, >, <=, >=, <>	Compare

Using Operators with Dates and Times

You can use several of the operators with dates and times as summarized here:

Operator	Type of Operation	Result
Plus	+ TimeSpan	TimeSpan
Negate	- TimeSpan	TimeSpan
Add	DateTime + TimeSpan	DateTime
Add	TimeSpan + DateTime	DateTime

Add	TimeSpan + TimeSpan	TimeSpan
Subtract	DateTime - DateTime	TimeSpan
Subtract	DateTime - TimeSpan	DateTime
Subtract	TimeSpan - TimeSpan	DateTime

The same order of precedence applies, including use of parentheses, as described in *Order of Precedence* on page 7. For more information about functions that use and return DateTime and TimeSpan objects, refer to *Date and Time Functions* on page 10.

If a DateTime or TimeSpan calculation results in an exception (for example, an OverflowException), the operator returns the #NUM! error.

Functions in a Formula

Functions are code segments that perform calculations by using specific values, called arguments, in a particular order that can be used in formulas. For example, the SUM function adds values or ranges of cells and the PMT function calculates the loan payments based on an interest rate, the length of the loan, and the principal amount of the loan. Functions may be either built-in functions that come with Spread or user-defined functions that you create.

Arguments can be numbers, text, logical values, arrays, cell ranges, cell references, or error values. The value you use for an argument must be valid for the given function. Arguments can also be constants, formulas, or other functions. Using a function as an argument for another function is known as nesting a function. Some arguments are optional; this reference displays "[Optional]" before the description of the argument for those arguments that are not required. These are described below .

The structure of a function begins with the function name, followed by an opening parenthesis, the arguments for the function separated by commas, and a closing parenthesis. If you are entering the function into a cell directly, type an equal sign (=) before the function name. The following topics describe the formula functions available. Each includes an example. Examples that provide results give decimal values for 10 decimal places.

Optional Arguments

Some functions have a variable number of arguments with some (typically the last) arguments being optional. These are displayed in this reference with the word Optional in brackets "[Optional]" before the argument in the table of arguments. For example, consider the payment function (PMT) which has five arguments with the last two being optional. In Spread, you can make any of the following calls:

```
PMT(rate,nper,pv,fv,type)
PMT(rate,nper,pv,fv)
PMT(rate,nper,pv,fv,)
PMT(rate,nper,pv,,type)
PMT(rate,nper,pv,,)
PMT(rate,nper,pv)
```

The optional arguments may be omitted. Any missing optional argument is handled with the default value being passed. For example

```
FIXED(1234.5678,,FALSE)
```

evaluates the same as

```
FIXED(1234.5678,2,FALSE)
```

since the default value for the number of decimal places is 2.

Missing Arguments

Missing arguments are intended to be used with functions that have optional arguments . If a missing argument is passed in for a required argument then the function will evaluate to the #N/A error.

Volatile Functions

Formulas that contain volatile functions are recalculated every time any other function is recalculated. If the EnableCrossSheetReferences property for the control is false, then the functions are recalculated only on the sheet where the changes occur.

Categories of Functions

These functions are categorized into one of these function types:

- Database Functions
- Date and Time Functions
- Engineering Functions
- Financial Functions
- Information Functions
- Logical Functions
- Lookup Functions
- Math and Trigonometry Functions
- Statistical Functions
- Text Functions
- Volatile Functions

Database Functions

The functions that relate to database and list management are:

DAVERAGE	DCOUNT	DCOUNTA	DGET
DMAX	DMIN	DPRODUCT	DSTDEV
DSTDEVP	DSUM	DVAR	DVARP

These functions apply a mathematical or statistical operation to a subset of values in a range of cells treated as a database. The database table can be thought of as a two-dimensional array organized into rows and columns. Or it can be thought of as a one-dimensional array of records where each record is a structure that has one or more fields. In the context of database tables, the terms "row" and "record" mean the same thing and the terms "column" and "field" mean the same thing. Database refers to a range of cells where the first row in the range represents field labels. The remaining rows in the range represent records. The columns in the range represent fields.

Date and Time Functions

The functions that relate to date-time values and time-span values are:

DATE	DATEDIF	DATEVALUE	DAY
DAYS360	EDATE	EOMONTH	HOUR
MINUTE	MONTH	NETWORKDAYS	NOW
SECOND	TIME	TIMEVALUE	TODAY
WEEKDAY	WEEKNUM	WORKDAY	YEAR
YEARFRAC			

For most of these functions you can specify the date argument as a DateTime object, as in the result of a function such as DATE(2003,7,4), or a TimeSpan object, as in the result of a function such as TIME(12,0,0). For compatibility with Excel, it also allows dates to be specified as a number (as in 37806.5) or as a string (as in "7/4/2003 12:00"). The numbers and strings are converted to instances of the DateTime class.

Dates as numeric values are in the form x.y, where x is the "number of days since December 30, 1899" and y is the fraction of day. Numbers to the left represent the date. Times as numeric values are decimal fractions ranging from 0 to 0.99999999, representing the times from 0:00:00 (12:00:00 A.M.) to 23:59:59 (11:59:59 P.M.).

The following three formulas produce the same result:

```
YEAR(DATE(2004,8,9))
YEAR(38208)
YEAR("8/9/2004")
```

In Excel, dates can range from 01/01/1900 to 12/31/9999, and in the .NET framework, instances of the DateTime class can range from 01/01/0001 to 12/31/9999. In Spread, we generally support the larger range found in the .NET framework. For Excel compatibility there are a few cases where the function allows only the smaller range (for example, the DATE function can only be used to enter dates since 01/01/1900).

You may see some differences in values if exporting to or importing from Excel. Both Excel and OLE automation use doubles to represent dates and times, with the integer portion of the double representing the number of days from a base date. In Excel, the base date that is used is 01/01/1900 and the year 1900 is treated as a leap year. In OLE automation, Microsoft corrected this by using the base date of 12/31/1899. As OLE automation does, our spreadsheets treat 1900 as a non-leap year and thus use the base date of 12/31/1899.

Engineering Functions

The functions that relate to engineering calculations are:

BESSELI	BESSELJ	BESSELK	BESSELY
BIN2DEC	BIN2HEX	BIN2OCT	COMPLEX
CONVERT	DEC2BIN	DEC2HEX	DEC2OCT
DELTA	ERF	ERFC	GESTEP
HEX2BIN	HEX2DEC	HEX2OCT	IMABS
IMAGINARY	IMARGUMENT	IMCONJUGATE	IMCOS
IMDIV	IMEXP	IMLN	IMLOG10
IMLOG2	IMPOWER	IMPRODUCT	IMREAL
IMSIN	IMSQRT	IMSUB	IMSUM
OCT2BIN	OCT2DEC	OCT2HEX	

For more information on the engineering functions that involve complex numbers, refer to the following section.

Complex Numbers in Engineering Functions

Many of the engineering functions involve complex numbers. A complex number consists of two parts, a real part and an imaginary part. You can think of a complex number as being a point (x,y) in a plane. You can think of a real number as being a point (x,0) on the x-axis of the plane. Note that real numbers are a subset of complex numbers with zero for the coefficient of the imaginary part.

There is not a complex number data type. Instead, complex numbers are represented using strings of the form "x+yi" where x and y are real numbers and x is the real part and yi is the imaginary part. For example:

```
"2+3i"

"1.23E4+5.67E8i"
```

Note that if either the real part or the imaginary part is zero then the zero part can be optionally omitted from the text representation. For example:

`"3"` is equivalent to `"3+0i"`

`"4i"` is equivalent to `"0+4i"`

Since real numbers are a subset of complex numbers, a real number can be used in place of a string of the form "x+yi". For example:

`3` is equivalent to `"3+0i"`

The functions that return a complex number return a string of the form "x+yi". For example:

`COMPLEX(3,5)` returns `"3+5i"`

The functions that accept a complex number can accept either a number or a string of the form "x+yi". For example:

`IMSUM("1+2i", "3+4i")` returns `"4+6i"`

`IMSUM(1, 3)` returns `"4"`

When a string cannot be converted to a number Spread returns a #VALUE error. For example:

`COS("abc")` returns #VALUE!

`IMCOS("abc")` returns #VALUE!

Spread allows either suffix "j" or the suffix "i" to denote the imaginary part. For example:

`"3+4j"` is equivalent to `"3+4i"`

Spread allows mixed suffixes in the a given formula and always returns the "i" suffix. For example:

`IMSUM("1+2i","3+4i")` returns `"4+6i"`

`IMSUM("1+2j","3+4j")` returns `"4+6i"`

`IMSUM("1+2i","3+4j")` returns `"4+6i"`

Spread does not allow spaces before the real part or before the imaginary part. For example:

`IMABS("3+4i")` returns `5`

`IMABS("    3+4i")` returns `#VALUE!`

`IMABS("3    +4i")` returns `#VALUE!`

`IMABS("3+4i    ")` returns `#VALUE!`

Financial Functions

The functions that relate to financial calculations such as interest calculations are:

ACCRINT	ACCRINTM	AMORDEGRC	AMORLINC
COUPDAYS	COUPDAYBS	COUPDAYSNC	COUPNCD
COUPNUM	COUPPCD	CUMIPMT	CUMPRINC
DB	DDB	DISC	DOLLAR
DOLLARDE	DOLLARFR	DURATION	EFFECT
EURO	EUROCONVERT	FV	FVSCHEDULE
INTRATE	IPMT	IRR	ISPMT
MDURATION	MIRR	NOMINAL	NPER
NPV	ODDFPRICE	ODDFYIELD	ODDLPRICE

ODDLYIELD	PMT	PPMT	PRICE
PRICEDISC	PRICEMAT	PV	RATE
RECEIVED	SLN	SYD	TBILLEQ
TBILLPRICE	TBILLYIELD	VDB	XIRR
XNPV	YIELD	YIELDDISC	YIELDMAT

For the arguments of some of these functions and for the results of some of these functions, money paid out is represented by negative numbers and money you receive is represented by positive numbers. How the currency values are displayed depends upon how you set up the cell type and the format settings.

Day Count Basis

For many of the financial functions, the day count basis is used:

Basis Number	Day Count Basis
0 (or omitted)	United States of America (NASD) 30/360
1	Actual/Actual
2	Actual/360
3	Actual/365
4	European 30/360

[NASD is the National Association of Securities Dealers.]

Information Functions

The functions that relate to information about a cell or the value in a cell are:

COUNTBLANK	ERRORTYPE	ISBLANK	ISERR
ISERROR	ISEVEN	ISLOGICAL	ISNA
ISNONTEXT	ISNUMBER	ISODD	ISREF
ISTEXT	N	NA	TYPE

Logical Functions

The functions that relate to logical operations are:

AND	FALSE	IF	NOT
OR	TRUE		

Lookup Functions

The functions that relate to referencing and finding other parts of the spreadsheet are:

ADDRESS	CHOOSE	COLUMN	COLUMNS
HLOOKUP	INDEX	LOOKUP	OFFSET
ROW	ROWS	TRANSPOSE	VLOOKUP

Math and Trigonometry Functions

The functions that relate to mathematical calculations are:

ABS	ACOS	ACOSH	ASIN
ASINH	ATAN	ATAN2	ATANH
CEILING	COMBIN	COS	COSH
COUNTIF	DEGREES	EVEN	EXP
FACT	FACTDOUBLE	FLOOR	GCD
INT	LCM	LN	LOG
LOG10	MDETERM	MINVERSE	MMULT
MOD	MROUND	MULTINOMIAL	ODD
PI	POWER	PRODUCT	QUOTIENT
RADIANS	RAND	RANDBETWEEN	ROMAN
ROUND	ROUNDDOWN	ROUNDUP	SERIESSUM
SIGN	SIN	SINH	SQRT
SQRTPI	SUBTOTAL	SUM	SUMIF
SUMPRODUCT	SUMSQ	SUMX2MY2	SUMX2PY2
SUMXMY2	TAN	TANH	TRUNC

Statistical Functions

The functions that relate to statistical operations are:

AVEDEV	AVERAGE	AVERAGEA	BETADIST
BETAINV	BINOMDIST	CHIDIST	CHIINV
CHITEST	CONFIDENCE	CORREL	COUNT
COUNTA	COVAR	CRITBINOM	DEVSQ
EXPONDIST	FDIST	FINV	FISHER
FISHERINV	FORECAST	FREQUENCY	FTEST
GAMMADIST	GAMMAINV	GAMMALN	GEOMEAN
GROWTH	HARMEAN	HYPGEOMDIST	INTERCEPT
KURT	LARGE	LINEST	LOGEST
LOGINV	LOGNORMDIST	MAX	MAXA
MEDIAN	MIN	MINA	MODE
NEGBINOMDIST	NORMDIST	NORMINV	NORMSDIST
NORMSINV	PEARSON	PERCENTILE	PERCENTRANK
PERMUT	POISSON	PROB	QUARTILE

RANK	RSQ	SKEW	SLOPE
SMALL	STANDARDIZE	STDEV	STDEVA
STDEVP	STDEVPA	STEYX	TDIST
TINV	TREND	TRIMMEAN	TTEST
VAR	VARA	VARP	VARPA
WEIBULL	ZTEST		

Text Functions

The functions that relate to handling text are:

CHAR	CLEAN	CODE	CONCATENATE
DOLLAR	EXACT	FIND	FIXED
LEFT	LEN	LOWER	MID
PROPER	REPLACE	REPT	RIGHT
SUBSTITUTE	T	TRIM	UPPER
VALUE			

Arrays in a Formula

Formulas may include functions that operate on arrays. Spread supports array constants in formulas. Use curly brackets { } to enclose the array elements. Use a comma to separate elements within a row. Use a semicolon to separate rows within the array. Individual elements can be number values, text values, logical values, or error values. Some examples of arrays are:

```
CORREL({5,10,15,20,25},{4,8,16,32,64})

CORREL({73000,45000,40360},{42,70,40})

ROWS({1,2,3;4,5,6})
```

Data Types Using Formulas

When you assign cell data using the Text property, the spreadsheet uses the cell type to parse the assigned string into the needed data type. For example, a NumberCellType parses a string into a double data type. When you assign the cell data using the Value property, the spreadsheet accepts the assigned object as is and no parsing occurs, so if you set it with a string, it remains a string. Some numeric functions (for example, SUM) ignore non-numeric values in a cell range. For example, if the cell range A1:A3 contains the values {1, "2", 3}, then the formula SUM(A1:A3) evaluates to 4 because the SUM function ignores the string "2". Be sure that you set the value correctly for any cells used in the calculation of a formula and that you set them with the correct data type.

Custom Functions in Formulas

Formulas may include custom, user-defined functions. If you have functions that you use on a regular basis that are not in the built-in functions or you wish to combine some of the built-in functions into a single function, you can do so by defining your own custom functions. They can be called as you would call any of the built-in functions. Custom functions can have up to 255 arguments.

Duplicate Function Names

A custom function can have the same name as a built-in function. The custom function takes priority over the built-in function. Custom functions are dynamically linked at evaluation time. Thus, the application can redefine an existing custom function.

Excel Function Names

In Spread, the HYPERLINK function is treated as a custom function since we do not have that as a built-in function. A custom function in Spread gets exported as a custom function in Excel. However, there is no way to export the implementation of the custom function. Thus, Excel sees the exported custom function as an unimplemented custom function which evaluates to the #NAME? error. When you enter or leave edit mode via the formula bar, Excel reparses the formula and recognizes the function as the built-in HYPERLINK function. Unfortunately, there is no way to tell the Spread control that a given custom functon in Spread should be exported as a built-in function in Excel.

Suppose the application needs an Excel function that Spread does not support. The application would have to supply a custom function to mimic the Excel function. Spread's implementation of a custom function can not be exported to an Excel file, so the custom function gets exported as an undefined custom function. In Excel, an undefined custom function will evaluate to the #VALUE! error. When you enter edit mode and then leave edit mode in Excel, Excel will reparse the formula (even if you made no changes to the formula). During reparsing, Excel will treat the function as the built-in function (instead of a custom function). The formula will then evaluate to the expected value (instead of the #VALUE! error). Your example of a problem formula does not appear to fall into the above described scenario because the formula only uses the MAX and SUM functions. However, it is still possible that the formula could be referencing a cell that uses a custom function which would get you back into the above described scenario. Editing the referenced cell would get rid of the #VALUE! error in the referenced cell. The recaluclations would cascade back the cell in question.

See Also

Refer to the product Developer’s Guide for more details on how to create a custom function.

Refer to the product Assembly Reference for more details on the methods that add or get custom functions.

Custom Names in Formulas

Formulas may include custom, user-defined names. Custom names are identifiers to represent information in the spreadsheet. A custom name can refer to a cell, a range of cells, a computed value, or a formula. (Methods that deal with custom names provide the same functionality as the Name in Excel.) A custom name can contain up to 255 characters and can include letters, numbers, or underscores. The first character must be a letter or an underscore.

The name's value can be assigned or retrieved as either a string object or as an expression object. Refer to the Assembly Reference for more details on the methods that add or get custom names.

From the example in C#:

```
dataModel.AddCustomName("Total", new
FarPoint.CalcEngine.CellExpression(3, 2));
```

a name called Total is created that represents the cell at absolute location 3,2. Assuming A1 notation (ReferenceStyle = A1), then this would be equivalent to:

```
dataModel.AddCustomName("Total", "$D$3", 0, 0);
```

In Excel, this would be equivalent to:

```
Name: Total
Refers To: =$D$3
```

Once the name is defined, the name can be used in formulas. When the formula is evaluated, the name's value is referenced and evaluated. Given the above definition, the following two formula assignments would produce the same result:

```
spread.ActiveSheet.SetFormula(0, 0, "Total");
spread.ActiveSheet.SetFormula(0, 0, "$D$3");
```

Note that the string versions of the AddCustomName and GetCustomName methods take the base row or base column arguments. These arguments are used to parse or unparse relative addresses in A1 notation. These arguments are ignored when using absolute addresses or when using R1C1 notation. A1 notation requires a base location from which the relative offset is computed.

For example:

```
dataModel.AddCustomName("Beta", "D3", 0, 0); // same as "R[2]C[3]"
dataModel.AddCustomName("Gamma", "D3", 4, 4); // same as "R[-2]C[-1]"
```

In other words, cell D3 is +3/+2 from cell A1 but -1/-2 from cell E5. In Excel, the Insert > Name > Define dialog uses the active cell as the base location.

Refer to the product Developer’s Guide for more details on how to create a custom name.

Refer to the Assembly Reference for more details on the methods that add or get custom names.

Resultant Error Values

The values that can be displayed in a cell as a result of an invalid entry or invalid formula are as follows:

Value	Description
#DIV/0!	This displays when a formula includes a division by zero or when a formula uses, in the divisor, a cell reference to a blank cell or to a cell that contains zero.
#N/A	This displays when a value is not available to a function or formula or when an argument in an array formula is not the same size as the range that contains the array formula.
#NAME?	This displays when text in a formula is not recognized or when the name of a function is misspelled, or when including text without using double quotation marks. This can also happen when you omit a colon (:) in a cell range reference.
#NULL!	This displays when you specify an intersection of two areas that do not intersect. Possible causes include a mistyped reference operator or a mistyped cell reference.
#NUM!	This displays when a number in a formula or function can not be calculated, when a formula produces a number that is too large or too small to represent, or when using an unacceptable argument in a function that requires a number. If you are using a function that iterates, such as IRR or RATE, and the function cannot find a result, this value is displayed.
#REF!	This displays when a cell reference is not valid or when you deleted cells referred to by a formula.
#VALUE!	This displays when the wrong type of argument or operand is used, such as using text when the formula requires a number or a logical value, or using a range instead of a single value.

Chapter 2 Formula Functions

Spread for .NET provides these built-in functions, listed alphabetically.

For a list of functions by type, refer to *Categories of Functions* on page 9.

ABS	ACCRINT	ACCRINTM	ACOS
ACOSH	ADDRESS	AMORDEGRC	AMORLINC
AND	ASIN	ASINH	ATAN
ATAN2	ATANH	AVEDEV	AVERAGE
AVERAGEA	BESSELI	BESSELJ	BESSELK
BESSELY	BETADIST	BETAINV	BIN2DEC
BIN2HEX	BIN2OCT	BINOMDIST	CEILING
CHAR	CHIDIST	CHIINV	CHITEST
CHOOSE	CLEAN	CODE	COLUMN
COLUMNS	COMBIN	COMPLEX	CONCATENATE
CONFIDENCE	CONVERT	CORREL	COS
COSH	COUNT	COUNTA	COUNTBLANK
COUNTIF	COUPDAYS	COUPDAYBS	COUPDAYSNC
COUPNCD	COUPNUM	COUPPCD	COVAR
CRITBINOM	CUMIPMT	CUMPRINC	

DATE	DATEDIF	DATEVALUE	DAVERAGE
DAY	DAYS360	DB	DCOUNT
DCOUNTA	DDB	DEC2BIN	DEC2HEX
DEC2OCT	DEGREES	DELTA	DEVSQ
DGET	DISC	DMAX	DMIN
DOLLAR	DOLLARDE	DOLLARFR	DPRODUCT
DSTDEV	DSTDEVP	DSUM	DURATION
DVAR	DVARP	EDATE	EFFECT
EOMONTH	ERF	ERFC	ERRORTYPE
EURO	EUROCONVERT	EVEN	EXACT
EXP	EXPONDIST	FACT	FACTDOUBLE
FALSE	FDIST	FIND	FINV
FINV	FISHERINV	FIXED	FLOOR
FORECAST	FREQUENCY	FTEST	FV
FVSCHEDULE	GAMMADIST	GAMMAINV	GAMMALN
GCD	GEOMEAN	GESTEP	GROWTH

HARMEAN	HEX2BIN	HEX2DEC	HEX2OCT
HLOOKUP	HOUR	HYPGEOMDIST	IF
IMABS	IMAGINARY	IMARGUMENT	IMCONJUGATE
IMCOS	IMDIV	IMEXP	IMLN
IMLOG10	IMLOG2	IMPOWER	IMPRODUCT
IMREAL	IMSIN	IMSQRT	IMSUB
IMSUM	INDEX	INT	INTERCEPT
INTRATE	IPMT	IRR	ISBLANK
ISERR	ISERROR	ISEVEN	ISLOGICAL
ISNA	ISNONTEXT	ISNUMBER	ISODD
ISPMT	ISREF	ISTEXT	KURT
LARGE	LCM	LEFT	LEN
LINEST	LN	LOG	LOG10
LOGEST	LOGINV	LOGNORMDIST	LOOKUP
LOWER			
MAX	MAXA	MDETERM	MDURATION
MEDIAN	MID	MIN	MINA
MINUTE	MINVERSE	MIRR	MMULT
MOD	MODE	MONTH	MROUND
MULTINOMIAL	N	NA	NEGBINOMDIST
NETWORKDAYS	NOMINAL	NORMDIST	NORMINV
NORMSDIST	NORMSINV	NOT	NOW
NPER	NPV	OCT2BIN	OCT2DEC
OCT2HEX	ODD	ODDFPRICE	ODDFYIELD
ODDLPRICE	ODDLYIELD	OFFSET	OR
PEARSON	PERCENTILE	PERCENTRANK	PERMUT
PI	PMT	POISSON	POWER
PPMT	PRICE	PRICEDISC	PRICEMAT
PROB	PRODUCT	PROPER	PV
QUARTILE	QUOTIENT		
RADIANS	RAND	RANDBETWEEN	RANK
RATE	RECEIVED	REPLACE	REPT
RIGHT	ROMAN	ROUND	ROUNDDOWN

ROUNDUP	ROW	ROWS	RSQ
SECOND	SERIESSUM	SIGN	SIN
SINH	SKEW	SLN	SLOPE
SMALL	SQRT	SQRTPI	STANDARDIZE
STDEV	STDEVA	STDEVP	STDEVPA
STEYX	SUBSTITUTE	SUBTOTAL	SUM
SUMIF	SUMPRODUCT	SUMSQ	SUMX2MY2
SUMX2PY2	SUMXMY2	SYD	
T	TAN	TANH	TBILLEQ
TBILLPRICE	TBILLYIELD	TDIST	TIME
TIMEVALUE	TINV	TODAY	TRANSPOSE
TREND	TRIM	TRIMMEAN	TRUE
TRUNC	TTEST	TYPE	UPPER
VALUE	VAR	VARA	VARP
VARPA	VDB	VLOOKUP	WEEKDAY
WEEKNUM	WEIBULL	WORKDAY	XIRR
XNPV	YEAR	YEARFRAC	YIELD
YIELDDISC	YIELDMAT	ZTEST	

For more information on how to use these functions, refer to the *Formula Overview* on page 1, and to the chapter on Managing Formulas in the product Developers Guide.

ABS

Summary

This function calculates the absolute value of the specified value.

Syntax

`ABS(`*`value`*`)`
`ABS(`*`expression`*`)`

Arguments

This function can take either a value or an expression as an argument.

Remarks

This function turns negative values into positive values.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`ABS(R3C2)`
`ABS(B3)`
`ABS(-4)` gives the result `4`
`ABS(14-24)` gives the result `10`
`ABS(4)` gives the result `4`

Version Available

This function is available in product version 1.0 or later.

See Also

SIGN | Math and Trigonometry Functions

ACCRINT

Summary

This function calculates the accrued interest for a security that pays periodic interest.

Syntax

`ACCRINT(issue,first,settle,rate,par,frequency,basis)`

Arguments

This function has these arguments:

Argument	Description
issue	Date that the security is issued
first	First date for calculating the interest for the security
settle	Settlement date for the security
rate	Annual interest rate for the security
par	[Optional] Par value for the security; if omitted, the calculation uses a value of $1,000
frequency	Frequency of payment, number of payments per year
basis	[Optional] Integer representing the basis for day count (Refer to *Day Count Basis* on page 13.)

Data Types

Accepts numeric and DateTime object data. Returns numeric data.

Examples

`ACCRINT(A1,A2,A3,B4,D9,E9,0)`

Version Available

This function is available in product version 1.0 or later.

See Also

ACCRINTM | INTRATE | Financial Functions

ACCRINTM

Summary

This function calculates the accrued interest at maturity for a security that pays periodic interest.

Syntax

`ACCRINTM(issue,maturity,rate,par,basis)`

Arguments

This function has these arguments:

Argument	Description
issue	Date that the security is issued
maturity	Maturity date for security
rate	Annual interest rate for the security
par	[Optional] Par value for the security; if omitted, the calculation uses a value of $1,000
basis	[Optional] Integer representing the basis for day count (Refer to *Day Count Basis* on page 13.)

Data Types

Accepts numeric and DateTime object data. Returns numeric data.

Examples

```
ACCRINTM(A2,A3,B4,D9,3)
ACCRINTM(R1C1,R2C3,R4C2,R9C4,3)
```

Version Available

This function is available in product version 1.0 or later.

See Also

ACCRINT | INTRATE | Financial Functions

ACOS

Summary

This function calculates the arccosine, that is, the angle whose cosine is the specified value.

Syntax

`ACOS(value)`

Arguments

For the argument, you can specify the cosine of the angle you want to return, which must be a value between –1 and 1.

Remarks

The result angle is in radians between 0 (zero) and PI (pi). If you want to convert the result to degrees, multiply the result by 180/PI.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`ACOS(B3)`
`ACOS(R3C2)`
`ACOS(0.5)` gives the result `1.0471975512`

Version Available

This function is available in product version 1.0 or later.

See Also

ACOSH | ASIN | COS | COSH | Math and Trigonometry Functions

ACOSH

Summary

This function calculates the inverse hyperbolic cosine of the specified value.

Syntax

`ACOSH(`*`value`*`)`

Arguments

For the argument, you can specify any real number greater than or equal to 1.

Remarks

This function is the inverse of the hyperbolic cosine, so ACOSH(COSH(*n*)) gives the result *n*.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`ACOSH(B3)`
`ACOSH(R3C2)`
`ACOSH(1)` gives the result `0`
`ACOSH(10)` gives the result `2.9932228461`
`ACOS(R3C2)`

Version Available

This function is available in product version 1.0 or later.

See Also

ACOS | ASINH | Math and Trigonometry Functions

ADDRESS

Summary

This function uses the row and column numbers to create a cell address in text.

Syntax

`ADDRESS(row,column,absnum,a1style,sheettext)`

Arguments

This function has these arguments:

Argument	Description
row	Row number in the cell reference
*column*8	Column number in the cell reference
absnum	[Optional] Type of reference to return; can be any of: Value - Type of Cell Reference Returned 1 or omitted - Absolute 2 - Absolute row, relative column 3 - Relative row, absolute column 4 - Relative
a1style	[Optional] Logical value that indicates whether the reference style is A1; if TRUE or omitted, the style is A1; if FALSE, then the style is R1C1
sheettext	[Optional] Name of the sheet to use as an external reference; if omitted, no sheet name is used

Data Types

Accepts numeric and string data. Returns string data.

Examples

`ADDRESS(2,4,2,FALSE)`

Version Available

This function is available in product version 2.0 or later.

See Also

COLUMNS | ROWS | INDEX | Lookup Functions

AMORDEGRC

Summary

This function returns the depreciation for an accounting period, taking into consideration prorated depreciation, and applies a depreciation coefficient in the calculation based on the life of the assets.

Syntax

`AMORDEGRC(cost,datepurchased,firstperiod,salvage,period,drate,basis)`

Arguments

This function has these arguments:

Argument	Description
cost	Cost of the asset
datepurchased	Purchase date of the asset
firstperiod	End date of the first period
salvage	Salvage value at the end of the life of the asset
period	Accounting period
drate	Rate of depreciation
basis	[Optional] Integer representing the basis for day count (Refer to *Day Count Basis* on page 13)

Remarks

This function returns the depreciation until the last period of the asset life or until the total value of depreciation is greater than the cost of the assets minus the salvage value. The depreciation coefficients are:

Life of assets	Depreciation Coefficient
Between 3 and 4 years	1.5
Between 5 and 6 years	2
More than 6 years	2.5

This function differs from AMORLINC, which does not apply a depreciation coefficient in the calculation depending on the life of the assets.

Data Types

Accepts numeric and DateTime object data. Returns numeric data.

Examples

`AMORDEGRC(B1,B2,B3,B4,B5,B6,B7)`
`AMORDEGRC(2800,DATE(2003,9,4),DATE(2006,12,31),200,1,0.02,1)` gives the result `117`

Version Available

This function is available in product version 2.0 or later.

See Also

AMORLINC | Financial Functions

AMORLINC

Summary

This function calculates the depreciation for an accounting period, taking into account prorated depreciation.

Syntax

`AMORLINC(`*`cost`*`,`*`datepurchased`*`,`*`firstperiod`*`,`*`salvage`*`,`*`period`*`,`*`drate`*`,`*`basis`*`)`

Arguments

This function has these arguments:

Argument	Description
cost	Cost of the asset
datepurchased	Purchase date of the asset
firstperiod	End date of the first period
salvage	Salvage value at the end of the life of the asset
period	Accounting period
drate	Rate of depreciation
basis	[Optional] Integer representing the basis for day count (Refer to *Day Count Basis* on page 13)

Remarks

This function differs from AMORDEGRC, which applies a depreciation coefficient in the calculation depending on the life of the assets.

Data Types

Accepts numeric and DateTime object data. Returns numeric data.

Examples

`AMORLINC(B1,B2,B3,B4,B5,B6,B7)`

Version Available

This function is available in product version 2.0 or later.

See Also

AMORDEGRC | Financial Functions

AND

Summary

This function calculates logical AND.

Syntax

```
AND(bool1,bool2,...)
AND(array)
AND(array1,array2,...)
AND(expression)
AND(expression1,expression2,...)
```

Arguments

For the arguments of this function, provide numeric (0 or 1) or logical values (TRUE or FALSE) for up to 255 arguments. You can also specify a single array instead of listing the values separately, or up to 255 arrays. You can also specify the logical argument as an expression.

Remarks

This function returns TRUE if all its arguments are true; otherwise, returns FALSE if at least one argument is false.

Data Types

Accepts logical data (Boolean values of TRUE or FALSE) or numerical values (0 or 1). Returns logical data (Boolean values of TRUE or FALSE).

Examples

`AND(D12,E12)`
`AND(R12C42,R12C5,R12C1)`
`AND(D2:D12)`
`AND(R12C1:R12C9)`
`AND(true,true,true)` gives the result `TRUE`
`AND(TRUE(),FALSE())` gives the result `FALSE`
`AND(5+3=8,5+1=6)` gives the result `TRUE`

Version Available

This function is available in product version 1.0 or later.

See Also

NOT | OR | Logical Functions

ASIN

Summary

This function calculates the arcsine, that is, the angle whose sine is the specified value.

Syntax

`ASIN(value)`

Arguments

For the argument, specify the sine of the angle you want to return, which must be a value between –1 and 1.

Remarks

The result angle is in radians between –PI/2 and PI/2. If you want to convert the result to degrees, multiply the result by 180/PI.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`ASIN(B3)`
`ASIN(R3C2)`
`ASIN(0.5)` gives the result `0.5235987756`

Version Available

This function is available in product version 1.0 or later.

See Also

ACOS | SIN | SINH | Math and Trigonometry Functions

ASINH

Summary

This function calculates the inverse hyperbolic sine of a number.

Syntax

`ASINH(value)`

Arguments

For the argument, you can specify any real number.

Remarks

This function is the inverse of the hyperbolic sine, so ASINH(SINH(*n*)) gives the result *n*.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`ASINH(E4)`
`ASINH(R4C5)`
`ASINH(-5.5)` gives the result `-2.40606`
`ASINH(100)` gives the result `5.2983423656`

Version Available

This function is available in product version 1.0 or later.

See Also

ACOSH | ASIN | SIN | Math and Trigonometry Functions

ATAN

Summary

This function calculates the arctangent, that is, the angle whose tangent is the specified value.

Syntax

`ATAN(`*`value`*`)`

Arguments

For the argument, specify the tangent of the angle you want to return, which must be a value between –1 and 1.

Remarks

The result angle is in radians between –PI/2 and PI/2. If you want to convert the result to degrees, multiply the result by 180/PI.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`ATAN(B3)`
`ATAN(R3C2)`
`ATAN(1)` gives the result `0.7853981634`

Version Available

This function is available in product version 1.0 or later.

See Also

ACOS | ASIN | TAN | Math and Trigonometry Functions

ATAN2

Summary

This function calculates the arctangent of the specified x- and y-coordinates.

Syntax

`ATAN2(x,y)`

Arguments

This function can take real numbers as arguments.

Remarks

The arctangent is the angle from the x-axis to a line containing the origin (0, 0) and a point with coordinates (*x*, y).

The result is given in radians between –PI and PI, excluding –PI. If you want to convert the result to degrees, multiply the result by 180/PI.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`ATAN2(A1,E3)`
`ATAN2(R1C1,R3C5)`
`ATAN2(1,1)` gives the result `0.7853981634`

Version Available

This function is available in product version 1.0 or later.

See Also

ACOS | ASIN | ATAN | TAN | Math and Trigonometry Functions

ATANH

Summary

This function calculates the inverse hyperbolic tangent of a number.

Syntax

`ATANH(value)`

Arguments

For the argument, you can specify any real number between 1 and –1, excluding –1 and 1.

Remarks

This function is the inverse of the hyperbolic tangent, so ATANH(TANH(*n*)) gives the result *n*.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`ATANH(B5)`
`ATANH(R5C2)`
`ATANH(0.55)` gives the result`lt 0.6183813136`
`ATANH(-0.2)` gives the result `-0.2027325541`

Version Available

This function is available in product version 1.0 or later.

See Also

ACOSH | ASINH | ATAN | TAN | Math and Trigonometry Functions

AVEDEV

Summary

This function calculates the average of the absolute deviations of the specified values from their mean.

Syntax

AVEDEV(*value1*,*value2*,...)
AVEDEV(*array*)
AVEDEV(*array1*,*array2*,...)

Arguments

Each argument can be a double-precision floating-point value, an integer value, or an array (cell range) of these. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Remarks

This is a measure of the variability in a data set.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

AVEDEV(B5,L32,N25,D17)
AVEDEV(B1:B5)
AVEDEV(B1:B17,L1:L17,N2:N8)
AVEDEV(R5C2,R32C12,R25C15)
AVEDEV(R1C2:R1C7)
AVEDEV(98,79,85) gives the result 7.1111111111

Version Available

This function is available in product version 1.0 or later.

See Also

AVERAGE | DEVSQ | Statistical Functions

AVERAGE

Summary

This function calculates the average of the specified numeric values.

Syntax

AVERAGE(*value1*,*value2*,...)
AVERAGE(*array*)
AVERAGE(*array1*,*array2*,...)

Arguments

Each argument can be a double-precision floating-point value, an integer value, or an array (cell range) of these. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Remarks

This is a measure of the variability in a data set.

This function differs from AVERAGEA, which accepts text or logical values as well as numeric values.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

AVERAGE(A1,B3,D5,E9,L8,L9)
AVERAGE(R1C1,R3C2)
AVERAGE(A1:A9)
AVERAGE(A1:A9,B1:B9,D5:D8)
AVERAGE(98,72,85) gives the result 85

Version Available

This function is available in product version 1.0 or later.

See Also

AVEDEV | AVERAGEA | CONFIDENCE | DEVSQ | MEDIAN | VAR | Statistical Functions

AVERAGEA

Summary

This function calculates the average of the specified values, including text or logical values as well as numeric values.

Syntax

```
AVERAGEA(value1,value2,...)
AVERAGEA(array)
AVERAGEA(array1,array2,...)
```

Arguments

Each argument can be a double-precision floating-point value, an integer value, or an array (cell range). Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Remarks

This is a measure of the variability in a data set.

This function differs from AVERAGE because it allows text or logical values as well as numeric values.

Data Types

Accepts numeric, logical, or text data for all arguments. Returns numeric data.

Examples

`AVERAGEA(A1,B3,D5,E9,L8,L9)`
`AVERAGEA(R1C1,R3C2)`
`AVERAGEA(A1:A9)`
`AVERAGEA(A1:A9,B1:B9,D5:D8)`
`AVERAGEA(98,72,85)` gives the result `85`

Version Available

This function is available in product version 2.0 or later.

See Also

AVEDEV | DEVSQ | MEDIAN | VAR | AVERAGE | Statistical Functions

BESSELI

Summary

This function calculates the modified Bessel function of the first kind evaluated for purely imaginary arguments.

Syntax

`BESSELI(`*`value`*`,`*`order`*`)`

Arguments

This function has these arguments:

Argument	Description
value	Value at which to evaluate the function
order	Number representing the order of the function; if it is not an integer, it is truncated

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`BESSELI(A4,D5)`
`BESSELI(R4C1,R5C4)`
`BESSELI(1.8,2)` gives the result `0.5260402117`

Version Available

This function is available in product version 1.0 or later.

See Also

BESSELJ | BESSELY | Engineering Functions

BESSELJ

Summary

This function calculates the Bessel function of the first kind.

Syntax

`BESSELJ(`*`value`*`,`*`order`*`)`

Arguments

This function has these arguments:

Argument	Description
value	Value at which to evaluate the function
order	Number representing the order of the function; if it is not an integer, it is truncated

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`BESSELJ(A4,D5)`
`BESSELJ(R4C1,R5C4)`
`BESSELJ(1.85,2)` gives the result `0.31812827879`

Version Available

This function is available in product version 1.0 or later.

See Also

BESSELI | BESSELK | Engineering Functions

BESSELK

Summary

This function calculates the modified Bessel function of the second kind evaluated for purely imaginary arguments.

Syntax

`BESSELK(`*`value`*`,`*`order`*`)`

Arguments

This function has these arguments:

Argument	Description
value	Value at which to evaluate the function
order	Number representing the order of the function; if it is not an integer, it is truncated

Remarks

This function is also called the Neumann function

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`BESSELK(A4,D5)`
`BESSELK(R4C1,R5C4)`
`BESSELK(1.85,2)` gives the result `0.32165379`

Version Available

This function is available in product version 1.0 or later.

See Also

BESSELJ | BESSELY | Engineering Functions

BESSELY

Summary

This function calculates the Bessel function of the second kind.

Syntax

`BESSELY(`*`value`*`,`*`order`*`)`

Arguments

This function has these arguments:

Argument	Description
value	Value at which to evaluate the function
order	Number representing the order of the function; if it is not an integer, it is truncated

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`BESSELY(A4,D5)`
`BESSELY(R4C1,R5C4)`
`BESSELY(2.85,1)` gives the result `0.2801918953`

Version Available

This function is available in product version 1.0 or later.

See Also

BESSELJ | BESSELK | Engineering Functions

BETADIST

Summary

This function calculates the cumulative beta distribution function.

Syntax

`BETADIST(`*`x`*`,`*`alpha`*`,`*`beta`*`,`*`lower`*`,`*`upper`*`)`

Arguments

This function has these arguments:

Argument	Description
x	Value at which to evaluate the function, between the values of lower and upper
alpha	Alpha parameter of the distribution
beta	Beta parameter of the distribution
lower	[Optional] Lower bound of the interval for x; 0 if omitted
upper	[Optional] Upper bound of the interval for x; 1 if omitted

Remarks

If you omit values for *upper* and *lower*, the calculation uses the standard cumulative beta distribution, so that *lower* is zero and *upper* is one.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`BETADIST(3,B3,C3,2,4)`
`BETADIST(3,R3C2,R3C3,2,4)`
`BETADIST(3,6,9,2,4)` gives the result `0.7880249023`

Version Available

This function is available in product version 1.0 or later.

See Also

BETAINV | Statistical Functions

BETAINV

Summary

This function calculates the inverse of the cumulative beta distribution function.

Syntax

`BETAINV(prob,alpha,beta,lower,upper)`

Arguments

This function has these arguments:

Argument	Description
prob	Probability of the distribution
alpha	Alpha parameter of the distribution
beta	Beta parameter of the distribution
lower	[Optional] Lower bound of the interval for x; 0 if omitted
upper	[Optional] Upper bound of the interval for x; 1 if omitted

Remarks

If you omit values for *upper* and *lower*, the calculation uses the standard cumulative beta distribution, so that *lower* is zero and *upper* is one.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`BETAINV(0.75,B3,C3,2,4)`
`BETAINV(0.75,R3C2,R3C3,2,4)`
`BETAINV(0.75,9,12,2,4)` gives the result `3.0011968805`

Version Available

This function is available in product version 1.0 or later.

See Also

BETADIST | Statistical Functions

BIN2DEC

Summary

This function converts a binary number to a decimal number.

Syntax

`BIN2DEC(`*`number`*`)`

Arguments

For the argument of this function, specify the binary numeric value to convert.

Remarks

An error value is returned if the number contains more than 10 digits or is invalid.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`BIN2DEC(1111111)`

Version Available

This function is available in product version 2.0 or later.

See Also

BIN2HEX | BIN2OCT | DEC2BIN | OCT2DEC | Engineering Functions

BIN2HEX

Summary

This function converts a binary number to a hexadecimal number.

Syntax

`BIN2HEX(number,places)`

Arguments

This function has these arguments:

Argument	Description
number	Binary numeric value to convert
places	[Optional] Number of characters to return; if not an integer, the number is truncated

Remarks

An error value is returned if the *number* contains more than 10 digits or is invalid, or if the value of *places* is non-numeric or negative. If *places* is omitted, the calculation uses the minimum number of characters necessary. This argument is useful for adding leading zeros to the result.

Data Types

Accepts numeric data. Returns numeric data in hexadecimal format.

Examples

`BIN2HEX(1110)`

Version Available

This function is available in product version 2.0 or later.

See Also

BIN2DEC | BIN2OCT | DEC2HEX | OCT2HEX | Engineering Functions

BIN2OCT

Summary

This function converts a binary number to an octal number.

Syntax

`BIN2OCT(`*`number`*`,`*`places`*`)`

Arguments

This function has these arguments:

Argument	Description
number	Binary numeric value to convert
places	[Optional] Number of characters to return; if not an integer, the number is truncated

Remarks

An error value is returned if the *number* contains more than 10 digits or is invalid, or if the value of *places* is non-numeric or negative. If *places* is omitted, the calculation uses the minimum number of characters necessary. This argument is useful for adding leading zeros to the result.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`BIN2OCT(1001,2)`

Version Available

This function is available in product version 2.0 or later.

See Also

BIN2DEC | BIN2HEX | OCT2BIN | DEC2OCT | Engineering Functions

BINOMDIST

Summary

This function calculates the individual term binomial distribution probability.

Syntax

`BINOMDIST(x,n,p,cumulative)`

Arguments

This function has these arguments:

Argument	Description
x	Number representing the number of successes in trials; if not an integer, the number is truncated
n	Number representing the number of independent trials; if not an integer, the number is truncated
p	Probability of success on each trial; number between 0 and 1
cumulative	Logical value that determines the form of the function; if TRUE, then this function returns the cumulative distribution function, which is the probability that there are at most x successes; if FALSE, it returns the probability mass function, which is the probability that there are x successes

Remarks

Use this function in problems with a fixed number of tests or trials, when there are two mutually exclusive possible outcomes (a "success" and a "failure"), when trials are independent, and when the probability of one outcome is constant throughout the experiment. This function can, for example, calculate the probability that two of the next three babies born are male.

The binomial probability mass function is calculated as follows:

$$BINOMDIST(x,\ n,\ p,\ FALSE) = \frac{n!}{x!(n-x)!}p^x(1-p)^{n-x}$$

where x is the number of successes, n is the number of trials, and p is the probability of success on any one trial.

The cumulative binomial distribution is calculated as follows:

$$BINOMDIST(x,\ n,\ p,\ TRUE) = \sum_{y=x}^{n} BINOMDIST(y,\ n,\ p,\ FALSE)$$

where *n* is the number of trials, *x* is the number of successes, and *p* is the possibility of success on any one trial.

Data Types

Accepts numeric data for all arguments, except cumulative, which accepts logical data. Returns numeric data.

Example

A baby can be either male or female; for the sake of this example, assume the odds are 50/50 that a baby is either male or female. If female equals TRUE, we can use the following to determine the probability of the next 5 babies in 10 born being female. The probability of the first baby being female is 0.5, and the probability of exactly 5 of 10 babies born being female is:

`BINOMDIST(5,10,0.5,FALSE)` gives the result `0.2460937500`

Version Available

This function is available in product version 1.0 or later.

See Also

BETADIST | CRITBINOM | EXPONDIST | GAMMADIST | NEGBINOMDIST | WEIBULL | Statistical Functions

CEILING

Summary

This function rounds a number up to the nearest multiple of a specified value.

Syntax

`CEILING(value,signif)`

Arguments

This function has these arguments:

Argument	Description
value	Number to round
signif	Number representing the rounding factor

Use either both positive or both negative numbers for the arguments. Regardless of the sign of the numbers, the value is rounded away from zero.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`CEILING(C4,B2)`
`CEILING(B3,0.05)`
`CEILING(R4C3,1)`
`CEILING(4.65,2)` gives the result `6`
`CEILING(-2.78,-1)` gives the result `-3`

Version Available

This function is available in product version 1.0 or later.

See Also

FLOOR | EVEN | ODD | TRUNC | Math and Trigonometry Functions

CHAR

Summary

This function returns the character specified by a number.

Syntax

`CHAR(value)`

Arguments

For the argument, use a number between 1 and 255 specifying which character you want from the Windows character set (ANSI).

Data Types

Accepts numeric data. Returns string data.

Examples

`CHAR(B2)`
`CHAR(R2C2)`
`CHAR(66)` gives the result `B`
`CHAR(218)` gives the result `Ú`

Version Available

This function is available in product version 1.0 or later.

See Also

CODE | CONCATENATE | LOWER | PROPER | UPPER | Text Functions

CHIDIST

Summary

This function calculates the one-tailed probability of the chi-squared distribution.

Syntax

`CHIDIST(value,deg)`

Arguments

This function has these arguments:

Argument	Description
value	Value at which to evaluate the function
deg	Number of degrees of freedom; if not an integer, the number is truncated

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`CHIDIST(B5,D7)`
`CHIDIST(R5C2,R7C4)`
`CHIDIST(6.7,4)` gives the result `0.1526169403`

Version Available

This function is available in product version 1.0 or later.

See Also

CHIINV | CHITEST | Statistical Functions

CHIINV

Summary

This function calculates the inverse of the one-tailed probability of the chi-squared distribution.

Syntax

`CHIINV(`*`prob`*`,`*`deg`*`)`

Arguments

This function has these arguments:

Argument	Description
prob	Probability of the chi-squared distribution
deg	Number of degrees of freedom; if not an integer, the number is truncated

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`CHIINV(B5,D7)`
`CHIINV(R5C2,R7C4)`
`CHIINV(0.75,4)` gives the result `1.9225575262`

Version Available

This function is available in product version 1.0 or later.

See Also

CHIDIST | CHITEST | Statistical Functions

CHITEST

Summary

This function calculates the test for independence from the chi-squared distribution.

Syntax

`CHITEST(obs_array,exp_array)`

Arguments

This function has these arguments:

Argument	Description
obs_array	Array of observed values to test against expected values
exp_array	Array of expected values against which to test observed values

The arrays in the arguments must be of the same size.

Data Types

Accepts arrays of numeric data for both arguments. Returns numeric data.

Examples

```
CHITEST(B1:C8,B12:C19)
CHITEST(R1C2:R8C3,R12C2:R19C3)
```

Version Available

This function is available in product version 1.0 or later.

See Also

CHIDIST | CHIINV | AVERAGE | Statistical Functions

CHOOSE

Summary

This function returns a value from a list of values.

Syntax

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

Arguments

This function has these arguments:

Argument	Description
index	Index of the specified values to return; an integer value between 1 and 255
value1, etc.	Values from which to choose; can have up to 255 values; can be numbers, cell references, cell ranges, defined names, formulas, functions, or text

The value arguments can be range references as well as single values. For example, the formula:

```
SUM(CHOOSE(2,A1:A25,B1:B10,C1:C5))
```

evaluates to:

```
SUM(B1:B10)
```

which then returns a value based on the values in the range B1:B10.

Remarks

This function is evaluated first, returning the reference B1:B10. The SUM function is then evaluated using B1:B10.

Data Types

The *index* argument accepts numeric data. The *value* arguments accept any data. Returns the type of data of the specified value.

Examples

`CHOOSE(3,A1,B1,C1,D1,E1)` gives the result `C1`
`CHOOSE(3,R1C1,R1C2,R1C3,R1C4,R1C5)` gives the result `R1C3`
`CHOOSE(2,"dogs","birds","fish","cats","mice")` gives the result `birds`

Version Available

This function is available in product version 1.0 or later.

See Also

INDEX | SUM | Lookup Functions

CLEAN

Summary

This function removes all non-printable characters from text.

Syntax

`CLEAN(`*`text`*`)`

Arguments

The text argument is any data from which you want to remove non-printable characters.

Remarks

Use this function to remove text that contains characters that might not print with your operating system. For example, you can use this function to remove some low-level computer code, which is frequently at the beginning and end of data files and cannot be printed

Data Types

Accepts string data. Returns string data.

Example

In this example, CHR(7) returns a non-printable character

`CLEAN(CHAR(7)&"text"&CHAR(7))` gives the result `text`

Version Available

This function is available in product version 1.0 or later.

See Also

TRIM | SUBSTITUTE | Text Functions

CODE

Summary

This function returns a numeric code to represent the first character in a text string. The returned code corresponds to the Windows character set (ANSI).

Syntax

`CODE(`*`text`*`)`

Arguments

The argument is the text from which you want to determine the code of the first character.

Data Types

Accepts string data. Returns string data.

Examples

`CODE(H6)`
`CODE(R6C8)`
`CODE(""B"")` gives the result`t 66`
`CODE(""Buffalo"")` gives the result `66`

Version Available

This function is available in product version 1.0 or later.

See Also

CHAR | Text Functions

COLUMN

Summary

This function returns the column number of a reference.

Syntax

`COLUMN(`*`reference`*`)`

Arguments

The argument is a cell or a single area.

Remarks

If the reference is omitted, the reference of the cell that the function is in is used.

Data Types

Accepts cell references. Returns numeric data.

Examples

`COLUMN(A9)` gives the result `1`
`COLUMN(A1:A5)` gives the result `1`

Version Available

This function is available in product version 3.0 or later.

See Also

ROWS | INDEX | Lookup Functions

COLUMNS

Summary

This function returns the number of columns in an array.

Syntax

`COLUMNS(`*`array`*`)`

Arguments

The argument is an array, an array formula, or a range of cells.

Data Types

Accepts cell references or array. Returns numeric data.

Examples

`COLUMNS(B6:D12)` gives the result `3`
`COLUMNS(R6C2:R12C4)` gives the result `3`
`COLUMNS($B$8:$H$8)` gives the result `7`
`COLUMNS(R[2]C[1]:R[3]C[8])` gives the result `8`

Version Available

This function is available in product version 1.0 or later.

See Also

ROWS | INDEX | Lookup Functions

COMBIN

Summary

This function calculates the number of possible combinations for a specified number of items.

Syntax

`COMBIN(k,n)`

Arguments

This function has these arguments:

Argument	Description
k	Number representing the number of items; if not an integer, the number is truncated; must be positive and greater than or equal to *n*
n	Number of items in each possible permutation; if not an integer, the number is truncated; must be positive

Remarks

A combination is any set or subset of items, regardless of the internal order of the items. Contrast with permutation (the PERMUT function).

The number of combinations is calculated as follows:

$$COMBIN(k,\ n) = \binom{n}{k} = \frac{PERMUT(k,\ n)}{k!} = \frac{n!}{k!(n-k)!}$$

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`COMBIN(C4,B2)`
`COMBIN(B3,5)`
`COMBIN(R1C2,2)`
`COMBIN(8,2)` gives the result `28`
`COMBIN(100,3)` gives the result `161700`

Version Available

This function is available in product version 1.0 or later.

See Also

PERMUT | Math and Trigonometry Functions

COMPLEX

Summary

This function converts real and imaginary coefficients into a complex number.

Syntax

`COMPLEX(realcoeff,imagcoeff,suffix)`

Arguments

This function has these arguments:

Argument	Description
realcoeff	Coefficient of the real part of the complex number
imagcoeff	Coefficient of the imaginary part of the complex number
suffix	(Optional) Suffix of the imaginary part of the complex number, may be either "i "or "j". If omitted, "i" is used.

Remarks

For the suffix, use lowercase for "i" and "j" to prevent errors.

An error is returned if the real or imaginary coefficients are non-numeric.

For more information, refer to *Complex Numbers in Engineering Functions* on page 11.

Data Types

Accepts number and string data. Returns string data.

Examples

```
COMPLEX(3,5)
COMPLEX(3,5,"j")
```

Version Available

This function is available in product version 2.0 or later.

See Also

IMAGINARY | IMREAL | Engineering Functions | Complex Numbers in Engineering Functions

CONCATENATE

Summary

This function combines multiple text strings or numbers into one text string.

Syntax

`CONCATENATE(`*`text1`*`,`*`text2`*`,...)`

Arguments

The arguments can be strings, formulas that return a string, or references to cells containing a string. Up to 255 arguments may be included.

Data Types

Accepts string data for both arguments. Returns string data.

Examples

`CONCATENATE(B4,D5)`
`CONCATENATE(R4C2,R5C4)`
`CONCATENATE(""Gold "", ""Medal"")` gives the result `Gold Medal`

Version Available

This function is available in product version 1.0 or later.

See Also

CHAR | EXACT | Text Functions

CONFIDENCE

Summary

This function returns confidence interval for a population mean.

Syntax

`CONFIDENCE(alpha,stdev,size)`

Arguments

This function has these arguments:

Argument	Description
alpha	Alpha, significance level used in calculating confidence level, where confidence level is 100 times (1-*alpha*)%
stdev	Population standard deviation for the range
size	Number representing the size of the sample; if not an integer, the number is truncated

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`CONFIDENCE(0.5,B4,D5)`
`CONFIDENCE(0.5,R4C2,R5C4)`
`CONFIDENCE(0.05,3.5,150)` gives the result `0.560106363`

Version Available

This function is available in product version 1.0 or later.

See Also

AVERAGE | CHITEST | Statistical Functions

CONVERT

Summary

This function converts a number from one measurement system to its equivalent in another measurement system.

Syntax

`CONVERT(number,from-unit,to-unit)`

Arguments

This function has these arguments:

Argument	Description
number	Numeric value to convert
from-unit	Convertible units (see table below) of numeric value to convert
to-unit	Convertible units (see table below) of desired result

Remarks

In this context a measurement system is a set of units for different types of measurements. This function converts a number with one set of units to a number in different set of units.

An error value is returned if the convertible units (*from-unit* and *to-unit*) are invalid or are from different categories of unit types (different tables below).

The following tables list the convertible units by their unit type:

Weight and Mass

Unit Type	Convertible Units
Gram	"g"
Slug	"sg"
Pound Mass	"lbm"
U	"u"
Ounce Mass	"ozm"

Distance

Unit Type	Convertible Units
Meter	"m"
Statute mile	"mi"
Nautical mile	"Nmi"
Inch	"in"
Foot	"ft"
Yard	"yd"
Angstrom	"ang"
Pica (1/72 in.)	"Pica"

Time

Unit Type	Convertible Units
Year	"yr"
Day	"day"
Hour	"hr"
Minute	"mn"
Second	"sec"

Pressure

Unit Type	Convertible Units
Pascal	"Pa"
Atmosphere	"atm"
mm of Mercury	"mmHg"

Force

Unit Type	Convertible Units
Newton	"N"
Dyne	"dyn"
Pound force	"lbf"

Energy

Unit Type	Convertible Units
Joule	"J"
Erg	"e"
Thermodynamic calorie	"c"
IT calorie	"cal"
Electron volt	"eV"
Horsepower-hour	"Hph"
Watt-hour	"Wh"
Foot-pound	"flb"
BTU	"BTU"

Power

Unit Type	Convertible Units
Horsepower	"HP"
Watt	"W"

Magnetism

Unit Type	Convertible Units
Tesla	"T"
Gauss	"ga"

Temperature

Unit Type	Convertible Units
Degree Celsius	"C"
Degree Fahrenheit	"F"
Degree Kelvin	"K"

Liquid Measure Unit Type	Convertible Units
Teaspoon	"tsp"
Tablespoon	"tbs"
Fluid ounce	"oz"
Cup	"cup"
U.S. pint	"pt"
U.K. pint	"uk_pt"
Quart	"qt"
Gallon	"gal"
Liter	"l"

Data Types

Accepts numeric and string data. Returns numeric data.

Examples

```
CONVERT(68,"F","C")
```

Version Available

This function is available in product version 2.0 or later.

See Also

OCT2BIN | HEX2DEC | DEC2OCT | Engineering Functions

CORREL

Summary

This function returns the correlation coefficient of the two sets of data.

Syntax

`CORREL(`*`array1`*`,`*`array2`*`)`

Arguments

The two arrays of data in the arguments of this function should meet these criteria:

- The data should contain numbers, names, ranges, or references that are numeric. If some cells do not contain numeric data, they are ignored.
- The arrays should be the same size, with the same number of data points.
- The arrays should not be empty, nor should the standard deviation of their values equal zero.

Data Types

Accepts arrays of numeric data for both arguments. Returns numeric data.

Examples

`CORREL(C1:C10,D1:D10)`
`CORREL(R1C3:R10C3,R1C4:R10C4)`
`CORREL({5,10,15,20,25},{4,8,16,32,64})` gives the result `0.9332565253`
`CORREL({73000,45000,40360},{42,70,40})` gives the result `-0.3261046660`

Version Available

This function is available in product version 1.0 or later.

See Also

COVAR | Statistical Functions

COS

Summary

This function returns the cosine of the specified angle.

Syntax

`COS(angle)`

Arguments

This function can take any real number as an argument. The *angle* argument is the angle in radians for which you want the cosine.

Remarks

If the angle is in degrees, multiply it by PI/180 to convert it to radians.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`COS(B2)`
`COS(R1C3)`
`COS(45*PI()/180)` gives the result `0.7071067812`
`COS(RADIANS(30))`

Version Available

This function is available in product version 1.0 or later.

See Also

ACOS | ACOSH | COSH | Math and Trigonometry Functions

COSH

Summary

This function returns the hyperbolic cosine of the specified value.

Syntax

`COSH(value)`

Arguments

This function can take any real number as an argument.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`COSH(B3)`
`COSH(R1C2)`
`COSH(4)` gives the result `27.3082328360`

Version Available

This function is available in product version 1.0 or later.

See Also

ACOSH | COS | Math and Trigonometry Functions

COUNT

Summary

This function returns the number of cells that contain numbers.

Syntax

```
COUNT(value1,value2,...)
COUNT(array)
```

Arguments

The arguments may be separate values or an array of values. Up to 255 arguments of individual cells may be included.

Remarks

This function counts the number of cells that contain numbers in the specified cell range.

This function differs from COUNTA which also includes text or logical values as well as numbers.

Data Types

Accepts cell references. Returns numeric data.

Examples

```
COUNT(B2,B5,B8,D5,D8)
COUNT(A1:G5)
COUNT(R6C3:R9C4,2)
```

Version Available

This function is available in product version 2.0 or later.

See Also

COUNTA | Statistical Functions

COUNTA

Summary

This function returns the number of number of cells that contain numbers, text, or logical values.

Syntax

```
COUNTA(value1,value2,...)
COUNTA(array)
```

Arguments

The arguments may be separate values or an array of values. Up to 255 arguments of individual cells may be included.

Remarks

This function counts the number of non-empty cells in the specified cell range.

This function differs from COUNT because it includes text or logical values as well as numbers.

Data Types

Accepts cell references. Returns numeric data.

Examples

```
COUNTA(B2,D2,E4,E5,E6)
COUNTA(A1:G5)
COUNTA(R6C3:R9C4)
```

Version Available

This function is available in product version 2.0 or later.

See Also

COUNT | Statistical Functions

COUNTBLANK

Summary

This function returns the number of empty (or blank) cells in a range of cells on a sheet.

Syntax

COUNTBLANK(*cellrange*)

Arguments

This function takes a cell range reference as an argument.

Remarks

This function counts the number of empty or blank cells in the specified cell range on one sheet. This function does not count cells containing an empty string "". A cell is empty if the cell's Value is null (Nothing in VB). Note that there is a difference being a cell's Value being null and a cell's Value being the empty string "". For example, consider the following Spread code in C#:

```
spread.Sheets[0].Cells[0,0].Value = null;  // empty
spread.Sheets[0].Cells[1,0].Value = ""; // string
spread.Sheets[0].Cells[2,0].Value = "abc"; // string
spread.Sheets[0].Cells[3,0].Value = 123.0; // number
spread.Sheets[0].Cells[4,0].Formula = "COUNTBLANK(A1:A4)";
```

The formula in cell A5 evaluates to 1 because cell A1 is the only cell in the range A1:A4 that is empty.

Note: Spread's implementation of functions generally tries to follow the behavior found in popular spreadsheet applications. However, not all these applications agree whether the empty string "" should be treated the same as an empty cell. In Spread, both the COUNTBLANK and ISBLANK functions consistently treat the empty string "" differently than an empty cell.

Data Types

Accepts cell range reference. Returns numeric data.

Examples

COUNTBLANK(A1:G5)
COUNTBLANK(R6C3:R9C4)

Version Available

This function is available in product version 1.0 or later.

See Also

COUNTIF | ISBLANK | TYPE | Information Functions

COUNTIF

Summary

This function returns the number of cells that meet a certain condition.

Syntax

`COUNTIF(`*`cellrange`*`,`*`condition`*`)`

Arguments

This function has these arguments:

Argument	Description
cellrange	Range of cells to count; cell range reference
condition	Condition that determines which cells are counted, as a text, number, or expression (where expressions use the relational operators detailed in *Operators in a Formula* on page 6)

Data Types

Accepts cell range reference. Returns numeric data.

Examples

```
COUNTIF(A1:G5,"test")
COUNTIF(R6C3:R9C4,"<2")
```

Version Available

This function is available in product version 2.0 or later.

See Also

COUNT | COUNTA | COUNTBLANK | SUMIF | Math and Trigonometry Functions

COUPDAYS

Summary

This function returns the number of days in the coupon period that contains the settlement date.

Syntax

`COUPDAYS(`*`settlement`*`,`*`maturity`*`,`*`frequency`*`,`*`basis`*`)`

Arguments

This function has these arguments:

Argument	Description
settlement	Settlement date for the security
maturity	Maturity date for the security
frequency	Frequency of payment, number of coupon payments per year; must be 1, 2, or 4
basis	[Optional] Integer representing the basis for day count (Refer to *Day Count Basis* on page 13.)

Remarks

This function returns an error if *settlement* or *maturity* is invalid, or if *frequency* is a number other than 1, 2, or 4.

Data Types

Accepts numeric and DateTime object data. Returns numeric data.

Examples

`COUPDAYS(A1,A2,A3,A4)`

Version Available

This function is available in product version 2.0 or later.

See Also

COUPDAYBS | DURATION | Financial Functions

COUPDAYBS

Summary

This function calculates the number of days from the beginning of the coupon period to the settlement date.

Syntax

`COUPDAYBS(settlement,maturity,frequency,basis)`

Arguments

This function has these arguments:

Argument	Description
settlement	Settlement date for the security
maturity	Maturity date for the security
frequency	Frequency of payment, number of coupon payments per year; must be 1, 2, or 4
basis	[Optional] Integer representing the basis for day count (Refer to *Day Count Basis* on page 13.)

Remarks

This function returns an error if *settlement* or *maturity* is invalid, or if *frequency* is a number other than 1, 2, or 4.

Data Types

Accepts numeric and DateTime object data. Returns numeric data.

Examples

`COUPDAYBS(A1,A2,A3,A4)`

Version Available

This function is available in product version 2.0 or later.

See Also

COUPDAYS | Financial Functions

COUPDAYSNC

Summary

This function calculates the number of days from the settlement date to the next coupon date.

Syntax

COUPDAYSNC(*settlement*,*maturity*,*frequency*,*basis*)

Arguments

This function has these arguments:

Argument	Description
settlement	Settlement date for the security
maturity	Maturity date for the security
frequency	Frequency of payment, number of coupon payments per year; must be 1, 2, or 4
basis	[Optional] Integer representing the basis for day count (Refer to *Day Count Basis* on page 13.)

Remarks

This function returns an error if *settlement* or *maturity* is invalid, or if *frequency* is a number other than 1, 2, or 4.

Data Types

Accepts numeric and DateTime object data. Returns numeric data.

Examples

```
COUPDAYSNC(A1,A2,A3,A4)
```

Version Available

This function is available in product version 2.0 or later.

See Also

COUPDAYS | COUPDAYBS | Financial Functions

COUPNCD

Summary

This function returns a date number of the next coupon date after the settlement date.

Syntax

`COUPNCD(settlement,maturity,frequency,basis)`

Arguments

This function has these arguments:

Argument	Description
settlement	Settlement date for the security
maturity	Maturity date for the security
frequency	Frequency of payment, number of coupon payments per year; must be 1, 2, or 4
basis	[Optional] Integer representing the basis for day count (Refer to *Day Count Basis* on page 13.)

Remarks

This function returns an error if *settlement* or *maturity* is invalid, or if *frequency* is a number other than 1, 2, or 4.

Data Types

Accepts numeric and DateTime object data. Returns numeric data.

Examples

```
COUPNCD(A1,A2,A3,A4)
COUPNCD(A1,A2,A3,A4)
```

Version Available

This function is available in product version 2.0 or later.

See Also

COUPPCD | Financial Functions

COUPNUM

Summary

This function returns the number of coupons due between the settlement date and maturity date.

Syntax

`COUPNUM(settlement,maturity,frequency,basis)`

Arguments

This function has these arguments:

Argument	Description
settlement	Settlement date for the security
maturity	Maturity date for the security
frequency	Frequency of payment, number of coupon payments per year; must be 1, 2, or 4
basis	[Optional] Integer representing the basis for day count (Refer to *Day Count Basis* on page 13.)

Remarks

This function returns an error if *settlement* or *maturity* is invalid, or if *frequency* is a number other than 1, 2, or 4.

Data Types

Accepts numeric and DateTime object data. Returns numeric data.

Examples

```
COUPNUM(A1,A2,A3,A4)
COUPNUM(R6C3:R9C4)
```

Version Available

This function is available in product version 2.0 or later.

See Also

COUPDAYS | Financial Functions

COUPPCD

Summary

This function returns a date number of the previous coupon date before the settlement date.

Syntax

`COUPPCD(`*`settlement`*`,`*`maturity`*`,`*`frequency`*`,`*`basis`*`)`

Arguments

This function has these arguments:

Argument	Description
settlement	Settlement date for the security
maturity	Maturity date for the security
frequency	Frequency of payment, number of coupon payments per year; must be 1, 2, or 4
basis	[Optional] Integer representing the basis for day count (Refer to *Day Count Basis* on page 13.)

Remarks

This function returns an error if *settlement* or *maturity* is invalid, or if *frequency* is a number other than 1, 2, or 4.

Data Types

Accepts numeric and DateTime object data. Returns numeric data.

Examples

```
COUPPCD(B1,B2,B3,B4)
COUPPCD(R6C3,R9C4,R1C1,R2C2)
```

Version Available

This function is available in product version 2.0 or later.

See Also

COUPNCD | Financial Functions

COVAR

Summary

This function returns the covariance, which is the average of the products of deviations for each data point pair in two sets of numbers.

Syntax

`COVAR(array1,array2)`

Arguments

The two arrays of data in the arguments of this function should meet these criteria:

- The data should contain numbers, names, arrays, or references that are numeric. If some cells do not contain numeric data, they are ignored.
- The data sets should be the same size, with the same number of data points.
- The data sets should not be empty, nor should the standard deviation of their values equal zero.

Remarks

Use this covariance function to determine the relationship between two sets of data. For example, you can examine whether greater income accompanies greater levels of education in a population.

The covariance is calculated as follows, where *n* is the size of the arrays and mu is the mean.

$$COVAR(X,\ Y)\ =\ \frac{\left(\sum_{1}^{n}(x-\mu_x)(y-\mu_y)\right)}{(n-1)}$$

Data Types

Accepts arrays of numeric data for both arguments. Returns numeric data.

Examples

`COVAR(J2:J5,L2:L5)`
`COVAR(R2C12:R15C12,R2C14:R15C14)`
`COVAR({7,5,6},{7,4,4})` gives the result `1`
`COVAR({5,10,15,20,25},{4,8,16,32,64})` gives the result `144`

Version Available

This function is available in product version 1.0 or later.

See Also

CORREL | VAR | Statistical Functions

CRITBINOM

Summary

This function returns the criterion binomial, the smallest value for which the cumulative binomial distribution is greater than or equal to a criterion value.

Syntax

`CRITBINOM(n,p,alpha)`

Arguments

This function has these arguments:

Argument	Description
n	Number of trials; if not an integer, the number is truncated
p	Probability of success on each trial; number between 0 and 1
alpha	Alpha, value for the criterion

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`CRITBINOM(B5,0.75,0.92)`
`CRITBINOM(R5C2,R8C14,0.75)`
`CRITBINOM(14,0.75,0.85)` gives the result `12`

Version Available

This function is available in product version 1.0 or later.

See Also

BINOMDIST | Statistical Functions

CUMIPMT

Summary

This function returns the cumulative interest paid on a loan between the starting and ending periods.

Syntax

`CUMIPMT(rate,nper,pval,startperiod,endperiod,paytype)`

Arguments

This function has these arguments:

Argument	Description
rate	Interest rate
nper	Total number of payment periods
pval	Present value
startperiod	Starting period
endperiod	Ending period
paytype	Type of payment timing; can be any of: 0 - Payment at end of the period 1 - Payment at beginning of the period

Remarks

This functions returns an error when *rate*, *nper*, or *pval* is negative or zero.

Data Types

Accepts numeric and DateTime object data. Returns numeric data.

Examples

```
CUMIPMT(B2/12,B4*12,C4,14,20,0)
CUMIPMT(B2/12,B4*12,C4,14,20,0)
```

Version Available

This function is available in product version 2.0 or later.

See Also

CUMPRINC | INTRATE | Financial Functions

CUMPRINC

Summary

This function returns the cumulative principal paid on a loan between the start and end periods.

Syntax

`CUMPRINC(rate,nper,pval,startperiod,endperiod,paytype)`

Arguments

This function has these arguments:

Argument	Description
rate	Interest rate
nper	Total number of payment periods
pval	Present value
startperiod	Starting period
endperiod	Ending period
paytype	Type of payment timing; can be any of: 0 - Payment at end of the period 1 - Payment at beginning of the period

Remarks

This functions returns an error when *rate*, *nper*, or *pval* is negative or zero.

Data Types

Accepts numeric and DateTime object data. Returns numeric data.

Examples

```
CUMPRINC(B2/12,B4*12,C4,14,20,0)
CUMPRINC(B2/12,B4*12,C4,14,20,0)
```

Version Available

This function is available in product version 2.0 or later.

See Also

CUMIPMT | IPMT | Financial Functions

DATE

Summary

This function returns the DateTime object for a particular date, specified by the year, month, and day.

Syntax

`DATE(`*`year`*`,`*`month`*`,`*`day`*`)`

Arguments

This function has these arguments:

Argument	Description
year	Number representing the year, from 1 to 9999, using four digits; if not integer, number is truncated
month	Number representing the month of the year; if not integer, number is truncated
day	Number representing the day of the month; if not integer, number is truncated

If month is greater than 12, then month increments by the number of months over 12 and the year advances, if needed. For example, `DATE(2003,16,2)` returns the DateTime object representing April 2, 2004.

If day is greater than the number of days in the specified month, then day increments that number of days from the first day of the next month. For example, `DATE(2004,1,35)` returns the DateTime object representing February 4, 2004.

If values for the arguments are not integers, any decimal places are truncated. Negative values for months are taken from the year into previous years. Negative values for days are taken from the month into previous months.

Data Types

Accepts numeric data. Returns a DateTime object.

Examples

`DATE(A1,B1,C1)`
`DATE(R1C1,R1C2,R1C3)`
`DATE(2003,1,1)` gives the result `January 1, 2003`
`DATE(2004,2,10)` gives the result `February 10, 2004`

Version Available

This function is available in product version 1.0 or later.

See Also

DATEVALUE | TIME | Date and Time Functions

DATEDIF

Summary

This function returns the number of days, months, or years between two dates.

Syntax

`DATEDIF(date1,date2,outputcode)`

Arguments

The first two arguments are any dates, as strings, numeric values, or DateTime objects.

The output codes are:

Code	Description
"D"	The number of days between date1 and date2
"M"	The number of complete months between date1 and date2
"Y"	The number of complete years between date1 and date2
"YD"	The number of days between date1 and date2 as if they were in the same year
"YM"	The number of months between date1 and date2 as if they were in the same year
"MD"	The number of days between date1 and date2 as if they were in the same month and year

Data Types

Accepts strings, numeric values, and DateTime objects. Strings and numbers are converted to DateTime objects.

Examples

```
DATEDIF(A1,B1,C1)
DATEDIF(R1C1,R1C2,R1C3)
DATEDIF("2001/1/1","2003/1/1","Y")
```

Version Available

This function is available in product version 2.0 or later.

See Also

DATEVALUE | TIME | Date and Time Functions

DATEVALUE

Summary

This function returns a DateTime object of the specified date.

Syntax

`DATEVALUE(`*`date_string`*`)`

Arguments

The argument for this function is a date as a string.

Remarks

Use this function to convert a date represented by text to a DateTime object in standard format.

Data Types

Accepts string data. Returns a DateTime object.

Examples

`DATEVALUE(B18)`
`DATEVALUE(R18C2)`
`DATEVALUE("2004/10/6")` gives the result `10/6/2004 12:00:00 AM`

Version Available

This function is available in product version 1.0 or later.

See Also

DATE | TIMEVALUE | Date and Time Functions

DAVERAGE

Summary

This function calculates the average of values in a column of a list or database that match the specified conditions.

Syntax

`DAVERAGE(`*`database, field, criteria`*`)`

Arguments

This function has these arguments:

Argument	Description
database	Range of cells that make up the database; cell range reference
field	Column in the database, referred to by label or index
criteria	Range of cells that specify which rows in the database are used; cell range reference

The *database* argument is a range of cells that make up the database. Each column represents a field. The first row represents the field labels. Each remaining row represents a record of data.

The *field* argument determines which column in the database to use. The *field* argument can be a string (field label) or a number (field index).

The *criteria* argument is a range of cells that specify which rows in the database contain the conditions that select a subset of the data in the database. The first row represents field labels. The remaining rows represent conditions. Conditions in the same row are combined using an AND operation. Conditions in different rows are combined using an OR operation. Each condition can be a number or a string. The string can include a comparison operator (=, <>, <, >, <=, >=). If no operator is included then the equal operator (=) is assumed.

Wild card characters are not supported in the *criteria* argument.

Remarks

This is one of several database or list functions that treat a range of cells as if they were a database. For more details on this type of function, refer to *Database Functions* on page 9.

Data Types

Accepts cell ranges for database and criteria. Accepts a string or a number for field. Returns numeric data.

Examples

```
DAVERAGE(A4:E10,3,A4:E10)
DAVERAGE(A1:A9,"Income",D5:D8)
```

Version Available

This function is available in product version 2.5 or later.

See Also

DVAR | DVARP | AVERAGE | VAR | VARP | Database Functions

DAY

Summary

This function returns the day number of the month (integer 1 to 31) that corresponds to the specified date.

Syntax

`DAY(`*`date`*`)`

Arguments

Specify the date argument as a number (as in 37806.5) a string (as in "7/4/2003 12:00"), or a DateTime object, as in DATE(2003,7,4). For more details on the date inputs, refer to the discussion in *Date and Time Functions* on page 10.

Data Types

Accepts numeric, string, or DateTime object data. Returns numeric data.

Examples

`DAY(A2)`
`DAY(R2C1)`
`DAY(366778)` gives the result `14`
`DAY(33239)` gives the result `1` (because 33239 is the value for January 1, 1991)
`DAY("7/4/2003 12:00")`
`DAY(DATE(2003,7,4))`

Version Available

This function is available in product version 1.0 or later.

See Also

DATE | DATEVALUE | WEEKDAY | MONTH | Date and Time Functions

DAYS360

Summary

This function returns the number of days between two dates based on a 360-day year.

Syntax

`DAYS360(startdate,enddate,method)`

Arguments

This function has these arguments:

Argument	Description
startdate	Date from which to calculate days
enddate	Date to which to calculate days
method	[Optional] Method for calculating days; if FALSE or omitted, uses U.S. (NASD) method; if TRUE, uses European method.

Specify the date argument as a number (as in 37806.5) a string (as in "7/4/2003 12:00"), or a DateTime object, as in DATE(2003,7,4). For more details on the date inputs, refer to the discussion in *Date and Time Functions* on page 10

The methods for calculating the number of days can vary. The U.S. or NASD method works as follows:

- If the starting date is the 31st of a month, it becomes equal to the 30th of the same month.
- If the ending date is the 31st of a month and the starting date is earlier than the 30th of a month, the ending date becomes equal to the 1st of the next month.
- If the ending date is the 31st of a month and the starting date is the 30th or 31st of a month, the ending date becomes equal to the 30th of the ending date month.

The European method considers starting dates or ending dates that occur on the 31st of a month to be equal to the 30th of the same month.

Remarks

Use this function to help compute payments if your accounting system is based on a 360-day year (twelve 30-day months).

Data Types

Accepts numeric, string, or DateTime object data for the two date arguments and boolean for the *method* argument. Returns numeric data.

Examples

`DAYS360(B8,C8)`
`DAYS360(R8C2,R8C3)`
`DAYS360("7/15/2004","12/25/2004")` gives the result `160`

Version Available

This function is available in product version 1.0 or later.

See Also

DAY | DATEVALUE | Date and Time Functions

DB

Summary

This function calculates the depreciation of an asset for a specified period using the fixed-declining balance method.

Syntax

`DB(cost,salvage,life,period,month)`

Arguments

This functions has these arguments:

Argument	Description
cost	Initial cost of the asset
salvage	Value at the end of the depreciation period
life	Number of periods over which the asset is being depreciated
period	Period for which you want to calculate the depreciation; use the same units as the life argument
month	[Optional] Number of months in the first year; if omitted, the calculation uses 12 months

Remarks

The fixed-declining balance method computes depreciation at a fixed rate. This function uses the following equation to calculate depreciation for a period:

```
(cost - total depreciation from prior periods) x rate
```

where:

```
rate = 1 - ((salvage/cost)^(1/life)), rounded to three decimal places
```

Depreciation for the first and last periods is a special case. For the first period, the function uses this equation:

```
dep = cost x rate x month/12
```

For the last period, the function uses this equation:

```
dep = ((cost - total dep. from prior periods) x rate x (12 - month))/
12.
```

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`DB(B1,1000,10,1)`
`DB(R1C2,10000,10,1)`
`DB(500000,5000,5,1,10)` gives the result `$25,0833.3333333333`

Version Available

This function is available in product version 1.0 or later.

See Also

DB | DDB | SLN | SYD | Financial Functions

DCOUNT

Summary

This function counts the cells that contain numbers in a column of a list or database that match the specified conditions.

Syntax

`DCOUNT(database, field, criteria)`

Arguments

This function has these arguments:

Argument	Description
database	Range of cells that make up the database; cell range reference
field	[Optional] Column in the database, referred to by label or index
criteria	Range of cells that specify which rows in the database are used; cell range reference

The *database* argument is a range of cells that make up the database. Each column represents a field. The first row represents the field labels. Each remaining row represents a record of data.

The *field* argument determines which column in the database to use. The *field* argument can be a string (field label) or a number (field index). The *field* argument is optional. If omitted the function counts all the records that meet the criteria.

The *criteria* argument is a range of cells that specify which rows in the database contain the conditions that select a subset of the data in the database. The first row represents field labels. The remaining rows represent conditions. Conditions in the same row are combined using an AND operation. Conditions in different rows are combined using an OR operation. Each condition can be a number or a string. The string can include a comparison operator (=, <>, <, >, <=, >=). If no operator is included then the equal operator (=) is assumed.

Wild card characters are not supported in the *criteria* argument.

Remarks

This is one of several database or list functions that treat a range of cells as if they were a database. For more details on this type of function, refer to *Database Functions* on page 9.

Data Types

Accepts cell ranges for database and criteria. Accepts a string or a number for field. Returns numeric data.

Examples

```
DCOUNT(A4:E10,"Type",A4:E10)
DCOUNT(A1:A9,3,D5:D8)
```

Version Available

This function is available in product version 2.5 or later.

See Also

DCOUNTA | COUNT | COUNTA | Database Functions

DCOUNTA

Summary

This function counts the non-blank cells in a column of a list or database that match the specified conditions.

Syntax

`DCOUNTA(database, field, criteria)`

Arguments

This function has these arguments:

Argument	Description
database	Range of cells that make up the database; cell range reference
field	[Optional] Column in the database, referred to by label or index
criteria	Range of cells that specify which rows in the database are used; cell range reference

The *database* argument is a range of cells that make up the database. Each column represents a field. The first row represents the field labels. Each remaining row represents a record of data.

The *field* argument determines which column in the database to use. The *field* argument can be a string (field label) or a number (field index). The *field* argument is optional. If omitted the function counts all the records that meet the criteria.

The *criteria* argument is a range of cells that specify which rows in the database contain the conditions that select a subset of the data in the database. The first row represents field labels. The remaining rows represent conditions. Conditions in the same row are combined using an AND operation. Conditions in different rows are combined using an OR operation. Each condition can be a number or a string. The string can include a comparison operator (=, <>, <, >, <=, >=). If no operator is included then the equal operator (=) is assumed.

Wild card characters are not supported in the *criteria* argument.

Remarks

This is one of several database or list functions that treat a range of cells as if they were a database. For more details on this type of function, refer to *Database Functions* on page 9.

Data Types

Accepts cell ranges for database and criteria. Accepts a string or a number for field. Returns numeric data.

Examples

```
DCOUNTA(A4:E10,"Type",A4:E10)
DCOUNTA(A1:A9,3,D5:D8)
```

Version Available

This function is available in product version 2.5 or later.

See Also

DCOUNT | COUNT | COUNTA | DAVERAGE | Database Functions

DDB

Summary

This function calculates the depreciation of an asset for a specified period using the double-declining balance method or another method you specify.

Syntax

`DDB(cost,salvage,life,period,factor)`

Arguments

This function has these arguments:

Argument	Description
cost	Initial cost of the asset
salvage	Value at the end of depreciation
life	Number of periods over which the asset is being depreciated
period	Period for which you want to calculate the depreciation in the same units as the *life* argument
factor	[Optional] Rate at which the value declines; if omitted, the calculation uses 2 (double-declining method)

All arguments must be positive numbers.

This function uses the following calculation for depreciation for a period:

```
cost - salvage(total depreciation from prior periods) x factor/life
```

Remarks

This function uses the following calculation for depreciation for a period:

```
cost - salvage(total depreciation from prior periods) x factor/life
```

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`DDB(B1,1000,10,1)`
`DDB(R1C2,10000,10,1)`
`DDB(500000,5000,5,1,4)` gives the result `$40,0000`

Version Available

This function is available in product version 1.0 or later.

See Also

DB | DDB | SYD | Financial Functions

DEC2BIN

Summary

This function converts a decimal number to a binary number.

Syntax

`DEC2BIN(`*`number`*`,`*`places`*`)`

Arguments

This function has these arguments:

Argument	Description
number	Decimal numeric value to convert in the range of -512 to 511
places	[Optional] Number of characters to return; if not an integer, the number is truncated

If *places* argument is omitted, the calculation uses the minimum number of characters necessary. This argument is useful for adding leading zeros to the result.

Remarks

An error value is returned if the number is non-numeric or outside the range, or if the *places* value is non-numeric, negative, or too small.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`DEC2BIN(3,3)`

Version Available

This function is available in product version 2.0 or later.

See Also

DEC2HEX | DEC2OCT | BIN2DEC | OCT2BIN | Engineering Functions

DEC2HEX

Summary

This function converts a decimal number to a hexadecimal number.

Syntax

`DEC2HEX(number,places)`

Arguments

This function has these arguments:

Argument	Description
number	Decimal numeric value to convert in the range of -549,755,813,888 to 549,755,813,887
places	[Optional] Number of characters to return; if not an integer, the number is truncated

If *places* argument is omitted, the calculation uses the minimum number of characters necessary. This argument is useful for adding leading zeros to the result.

Remarks

An error value is returned if the number is non-numeric or outside the range, or if the *places* value is non-numeric, negative, or too small.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`DEC2HEX(103,4)`

Version Available

This function is available in product version 2.0 or later.

See Also

DEC2BIN | DEC2OCT | BIN2HEX | OCT2HEX | Engineering Functions

DEC2OCT

Summary

This function converts a decimal number to an octal number.

Syntax

`DEC2OCT(number,places)`

Arguments

This function has these arguments:

Argument	Description
number	Decimal numeric value to convert in the range of -536,870,912 and 536,870,911
places	[Optional] Number of characters to return; if not an integer, the number is truncated

If *places* argument is omitted, the calculation uses the minimum number of characters necessary. This argument is useful for adding leading zeros to the result.

Remarks

An error value is returned if the number is non-numeric or outside the range, or if the *places* value is non-numeric, negative, or too small.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`DEC2OCT(-99)`

Version Available

This function is available in product version 2.0 or later.

See Also

DEC2BIN | DEC2HEX | BIN2OCT | OCT2BIN | Engineering Functions

DEGREES

Summary

This function converts the specified value from radians to degrees.

Syntax

`DEGREES(`*`angle`*`)`

Arguments

This function takes any real number angle value as the argument.

Remarks

This function converts angle in radians to angle in degrees.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`DEGREES(B3)`
`DEGREES(R1C2)`
`DEGREES(PI())` gives the result `180`

Version Available

This function is available in product version 1.0 or later.

See Also

RADIANS | PI | Math and Trigonometry Functions

DELTA

Summary

This function identifies whether two values are equal. Returns 1 if they are equal; returns 0 otherwise.

Syntax

`DELTA(`*`value1`*`,`*`value2`*`)`

Arguments

This function takes two values as arguments.

Remarks

Also called the Kronecker Delta function. This is a discrete version of the Dirac delta function.

Data Types

Accepts numeric data. Returns numeric data (0 or 1).

Examples

`DELTA(A1,5)`
`DELTA(R1C4,R2C5)`
`DELTA(3,3)` gives the result `1`
`DELTA(3,2)` gives the result `0`
`DELTA(3,2.99999)` gives the result `0`
`DELTA(3,QUOTIENT(6,2))` gives the result `1`

Version Available

This function is available in product version 1.0 or later.

See Also

GESTEP | Engineering Functions

DEVSQ

Summary

This function calculates the sum of the squares of deviations of data points (or of an array of data points) from their sample mean.

Syntax

```
DEVSQ(value1,value2, ...)
DEVSQ(array)
DEVSQ(array1,array2,...)
```

Arguments

Each argument can be a double-precision floating-point value, an integer value, or an array (cell range) of these. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Remarks

This is a measure of the variability in a data set.

The sum of squared deviations is calculated as follows, where n is the number of values.

$$DEVSQ(x_1,x_2,\ldots x_n) = \sum_1^n (x - \bar{x})$$

If an array or cell reference argument contains text, logical values, or empty cells, the function ignores those values; however, the function includes in calculations cells with the value zero.

Data Types

Accepts numeric data for all arguments or array of numeric data. Returns numeric data.

Examples

`DEVSQ(B3,B5,B9,B10)`
`DEVSQ(B3:B14)`
`DEVSQ(R3C2,R5C2,R9C2)`
`DEVSQ(R3C2:R3C12)`
`DEVSQ(35,31,47,51,37,31,58,39)` gives the result `680.875`

Version Available

This function is available in product version 1.0 or later.

See Also

AVEDEV | AVERAGE | Statistical Functions

DGET

Summary

This function extracts a single value from a column of a list or database that matches the specified conditions.

Syntax

`DGET(database, field, criteria)`

Arguments

Argument	Description
database	Range of cells that make up the database; cell range reference
field	Column in the database, referred to by label or index
criteria	Range of cells that specify which rows in the database are used; cell range reference

The *database* argument is a range of cells that make up the database. Each column represents a field. The first row represents the field labels. Each remaining row represents a record of data.

The *field* argument determines which column in the database to use. The *field* argument can be a string (field label) or a number (field index).

The *criteria* argument is a range of cells that specify which rows in the database contain the conditions that select a subset of the data in the database. The first row represents field labels. The remaining rows represent conditions. Conditions in the same row are combined using an AND operation. Conditions in different rows are combined using an OR operation. Each condition can be a number or a string. The string can include a comparison operator (=, <>, <, >, <=, >=). If no operator is included then the equal operator (=) is assumed.

Wild card characters are not supported in the *criteria* argument.

Remarks

If no value matches the *criteria* argument, a #VALUE! error is returned. A #NUM! error is returned if more than one match is found.

This is one of several database or list functions that treat a range of cells as if they were a database. For more details on this type of function, refer to *Database Functions* on page 9.

Data Types

Accepts cell ranges for database and criteria. Accepts a string or a number for field. Returns numeric data.

Examples

```
DGET(A4:E10,"Type",A4:E10)
DGET(A1:A9,3,D5:D8)
```

Version Available

This function is available in product version 2.5 or later.

See Also

DAVERAGE | DCOUNT | Database Functions

DISC

Summary

This function calculates the discount rate for a security.

Syntax

`DISC(`*`settle`*`,`*`mature`*`,`*`pricep`*`,`*`redeem`*`,`*`basis`*`)`

Arguments

This function has these arguments:

Argument	Description
settle	Settlement date for the security
mature	Maturity date for the security
pricep	Amount invested in the security
redeem	Amount to be received at maturity
basis	[Optional] Integer representing the basis for day count (Refer to *Day Count Basis* on page 13.)

Data Types

Accepts numeric and DateTime object data. Returns numeric data.

Examples

`DISC(A1,B1,C4,100,2)`
`DISC("3/15/2003","5/15/2003",R3C4,R5C5,4)`
`DISC("5/15/2004","9/1/2004",98.2,100,3)` gives the result `0.0602752294`

Version Available

This function is available in product version 1.0 or later.

See Also

RATE | INTRATE | PRICEDISC | Financial Functions

DMAX

Summary

This function returns the largest number in a column of a list or database that matches the specified conditions.

Syntax

`DMAX(database, field, criteria)`

Arguments

Argument	Description
database	Range of cells that make up the database; cell range reference
field	Column in the database, referred to by label or index
criteria	Range of cells that specify which rows in the database are used; cell range reference

The *database* argument is a range of cells that make up the database. Each column represents a field. The first row represents the field labels. Each remaining row represents a record of data.

The *field* argument determines which column in the database to use. The *field* argument can be a string (field label) or a number (field index).

The *criteria* argument is a range of cells that specify which rows in the database contain the conditions that select a subset of the data in the database. The first row represents field labels. The remaining rows represent conditions. Conditions in the same row are combined using an AND operation. Conditions in different rows are combined using an OR operation. Each condition can be a number or a string. The string can include a comparison operator (=, <>, <, >, <=, >=). If no operator is included then the equal operator (=) is assumed.

Wild card characters are not supported in the *criteria* argument.

Remarks

This is one of several database or list functions that treat a range of cells as if they were a database. For more details on this type of function, refer to *Database Functions* on page 9.

Data Types

Accepts cell ranges for database and criteria. Accepts a string or a number for field. Returns numeric data.

Examples

```
DMAX(A4:E10,"Type",A4:E10)
DMAX(A1:A9,3,D5:D8)
```

Version Available

This function is available in product version 2.5 or later.

See Also

DAVERAGE | DCOUNT | DMIN | MAX | MIN | Database Functions

DMIN

Summary

This function returns the smallest number in a column of a list or database that matches the specified conditions.

Syntax

`DMIN(database, field, criteria)`

Arguments

Argument	Description
database	Range of cells that make up the database; cell range reference
field	Column in the database, referred to by label or index
criteria	Range of cells that specify which rows in the database are used; cell range reference

The *database* argument is a range of cells that make up the database. Each column represents a field. The first row represents the field labels. Each remaining row represents a record of data.

The *field* argument determines which column in the database to use. The *field* argument can be a string (field label) or a number (field index).

The *criteria* argument is a range of cells that specify which rows in the database contain the conditions that select a subset of the data in the database. The first row represents field labels. The remaining rows represent conditions. Conditions in the same row are combined using an AND operation. Conditions in different rows are combined using an OR operation. Each condition can be a number or a string. The string can include a comparison operator (=, <>, <, >, <=, >=). If no operator is included then the equal operator (=) is assumed.

Wild card characters are not supported in the *criteria* argument.

Remarks

This is one of several database or list functions that treat a range of cells as if they were a database. For more details on this type of function, refer to *Database Functions* on page 9.

Data Types

Accepts cell ranges for database and criteria. Accepts a string or a number for field. Returns numeric data.

Examples

```
DMIN(A4:E10,"Type",A4:E10)
DMIN(A1:A9,3,D5:D8)
```

Version Available

This function is available in product version 2.5 or later.

See Also

DAVERAGE | DCOUNT | DMAX | MAX | MIN | Database Functions

DOLLAR

Summary

This function converts a number to text using currency format, with the decimals rounded to the specified place.

Syntax

`DOLLAR(value,digits)`

Arguments

This function has these arguments:

Argument	Description
value	Numeric value to convert to text using the currency format
digits	[Optional] Number of decimal places to maintain; if negative, the value is rounded to the left of the decimal point; if omitted, the function rounds to two decimal places

Remarks

This function uses the current regional Windows settings to determine the format of the returned string.

Data Types

Accepts numeric data for both arguments. Returns string data.

Examples

`DOLLAR(B5,D2)`
`DOLLAR(R5C2,R2C4)`
`DOLLAR(1234.5678,3)` gives the result `$1,234.568`
`DOLLAR(123.45,1)` gives the result `$123.5`

Version Available

This function is available in product version 1.0 or later.

See Also

DOLLARDE | DOLLARFR | FIXED | Text Functions

DOLLARDE

Summary

This function converts a fraction dollar price to a decimal dollar price.

Syntax

`DOLLARDE(`*`fractionaldollar`*`,`*`fraction`*`)`

Arguments

This function has these arguments:

Argument	Description
fractionaldollar	Numeric value expressed as a fraction
fraction	Denominator of the fraction; if not an integer, the number is truncated

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

```
DOLLARDE(1.10,17)
DOLLARDE(R5C2,R2C4)
```

Version Available

This function is available in product version 2.0 or later.

See Also

DOLLAR | DOLLARFR | Financial Functions

DOLLARFR

Summary

This function converts a decimal number dollar price to a fraction dollar price.

Syntax

`DOLLARFR(decimaldollar,fraction)`

Arguments

This function has these arguments:

Argument	Description
decimaldollar	Decimal number
fraction	Denominator of the fraction; if not an integer, the number is truncated

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`DOLLARFR(B5,D2)`
`DOLLARFR(R5C2,R2C4)`
`DOLLARFR(1.125,16)` gives the result `1.02`

Version Available

This function is available in product version 2.0 or later.

See Also

DOLLAR | DOLLARDE | Financial Functions

DPRODUCT

Summary

This function multiplies the values in a column of a list or database that match the specified conditions.

Syntax

`DPRODUCT(`*`database, field, criteria`*`)`

Arguments

Argument	Description
database	Range of cells that make up the database; cell range reference
field	Column in the database, referred to by label or index
criteria	Range of cells that specify which rows in the database are used; cell range reference

The *database* argument is a range of cells that make up the database. Each column represents a field. The first row represents the field labels. Each remaining row represents a record of data.

The *field* argument determines which column in the database to use. The *field* argument can be a string (field label) or a number (field index).

The *criteria* argument is a range of cells that specify which rows in the database contain the conditions that select a subset of the data in the database. The first row represents field labels. The remaining rows represent conditions. Conditions in the same row are combined using an AND operation. Conditions in different rows are combined using an OR operation. Each condition can be a number or a string. The string can include a comparison operator (=, <>, <, >, <=, >=). If no operator is included then the equal operator (=) is assumed.

Wild card characters are not supported in the *criteria* argument.

Remarks

This is one of several database or list functions that treat a range of cells as if they were a database. For more details on this type of function, refer to *Database Functions* on page 9.

Data Types

Accepts cell ranges for database and criteria. Accepts a string or a number for field. Returns numeric data.

Examples

```
DPRODUCT(A4:E10,"Type",A4:E10)
DPRODUCT(A1:A9,3,D5:D8)
```

Version Available

This function is available in product version 2.5 or later.

See Also

DSUM | DCOUNT | PRODUCT | SUM | Database Functions

DSTDEV

Summary

This function estimates the standard deviation of a population based on a sample by using the numbers in a column of a list or database that match the specified conditions.

Syntax

`DSTDEV(database, field, criteria)`

Arguments

Argument	Description
database	Range of cells that make up the database; cell range reference
field	Column in the database, referred to by label or index
criteria	Range of cells that specify which rows in the database are used; cell range reference

The *database* argument is a range of cells that make up the database. Each column represents a field. The first row represents the field labels. Each remaining row represents a record of data.

The *field* argument determines which column in the database to use. The *field* argument can be a string (field label) or a number (field index).

The *criteria* argument is a range of cells that specify which rows in the database contain the conditions that select a subset of the data in the database. The first row represents field labels. The remaining rows represent conditions. Conditions in the same row are combined using an AND operation. Conditions in different rows are combined using an OR operation. Each condition can be a number or a string. The string can include a comparison operator (=, <>, <, >, <=, >=). If no operator is included then the equal operator (=) is assumed.

Wild card characters are not supported in the *criteria* argument.

Remarks

This is one of several database or list functions that treat a range of cells as if they were a database. For more details on this type of function, refer to *Database Functions* on page 9.

Data Types

Accepts cell ranges for database and criteria. Accepts a string or a number for field. Returns numeric data.

Examples

```
DSTDEV(A4:E10,"Type",A4:E10)
DSTDEV(A1:A9,3,D5:D8)
```

Version Available

This function is available in product version 2.5 or later.

See Also

DSTDEVP | DAVERAGE | STDEV | Database Functions

DSTDEVP

Summary

This function calculates the standard deviation of a population based on the entire population using the numbers in a column of a list or database that match the specified conditions.

Syntax

`DSTDEVP(database, field, criteria)`

Arguments

Argument	Description
database	Range of cells that make up the database; cell range reference
field	Column in the database, referred to by label or index
criteria	Range of cells that specify which rows in the database are used; cell range reference

The *database* argument is a range of cells that make up the database. Each column represents a field. The first row represents the field labels. Each remaining row represents a record of data.

The *field* argument determines which column in the database to use. The *field* argument can be a string (field label) or a number (field index).

The *criteria* argument is a range of cells that specify which rows in the database contain the conditions that select a subset of the data in the database. The first row represents field labels. The remaining rows represent conditions. Conditions in the same row are combined using an AND operation. Conditions in different rows are combined using an OR operation. Each condition can be a number or a string. The string can include a comparison operator (=, <>, <, >, <=, >=). If no operator is included then the equal operator (=) is assumed.

Wild card characters are not supported in the *criteria* argument.

Remarks

This is one of several database or list functions that treat a range of cells as if they were a database. For more details on this type of function, refer to *Database Functions* on page 9.

Data Types

Accepts cell ranges for database and criteria. Accepts a string or a number for field. Returns numeric data.

Examples

```
DSTDEVP(A4:E10,"Type",A4:E10)
DSTDEVP(A1:A9,3,D5:D8)
```

Version Available

This function is available in product version 2.5 or later.

See Also

DSTDEV | DAVERAGE | STDEV | Database Functions

DSUM

Summary

This function adds the numbers in a column of a list or database that match the specified conditions.

Syntax

`DSUM(database, field, criteria)`

Arguments

Argument	Description
database	Range of cells that make up the database; cell range reference
field	Column in the database, referred to by label or index
criteria	Range of cells that specify which rows in the database are used; cell range reference

The *database* argument is a range of cells that make up the database. Each column represents a field. The first row represents the field labels. Each remaining row represents a record of data.

The *field* argument determines which column in the database to use. The *field* argument can be a string (field label) or a number (field index).

The *criteria* argument is a range of cells that specify which rows in the database contain the conditions that select a subset of the data in the database. The first row represents field labels. The remaining rows represent conditions. Conditions in the same row are combined using an AND operation. Conditions in different rows are combined using an OR operation. Each condition can be a number or a string. The string can include a comparison operator (=, <>, <, >, <=, >=). If no operator is included then the equal operator (=) is assumed.

Wild card characters are not supported in the *criteria* argument.

Remarks

This is one of several database or list functions that treat a range of cells as if they were a database. For more details on this type of function, refer to *Database Functions* on page 9.

Data Types

Accepts cell ranges for database and criteria. Accepts a string or a number for field. Returns numeric data.

Examples

```
DSUM(A4:E10,"Type",A4:E10)
DSUM(A1:A9,3,D5:D8)
```

Version Available

This function is available in product version 2.5 or later.

See Also

DPRODUCT | DCOUNT | SUM | PRODUCT | Database Functions

DURATION

Summary

This function returns the Macauley duration for an assumed par value of $100.

Syntax

`DURATION(settlement,maturity,coupon,yield,frequency,basis)`

Arguments

This function has these arguments:

Argument	Description
settlement	Settlement date for the security
maturity	Maturity date for the security
coupon	Annual coupon rate
yield	Annual yield for the security
frequency	Frequency of payment, number of coupon payments per year; must be 1, 2, or 4
basis	[Optional] Integer representing the basis for day count (Refer to *Day Count Basis* on page 13.)

Remarks

This functions returns an error when *settlement* or *maturity* is invalid or *frequency* is a number other than 1, 2, or 4.

Data Types

Accepts numeric and DateTime object data. Returns numeric data.

Examples

```
DURATION(C1,C2,C3,C4,C5,C6)
DURATION(R5C2,R2C4,R3C1,R4C1,R5C1)
```

Version Available

This function is available in product version 2.0 or later.

See Also

COUPDAYS | MDURATION | Financial Functions

DVAR

Summary

This function estimates the variance of a population based on a sample by using the numbers in a column of a list or database that match the specified conditions.

Syntax

`DVAR(database, field, criteria)`

Arguments

Argument	Description
database	Range of cells that make up the database; cell range reference
field	Column in the database, referred to by label or index
criteria	Range of cells that specify which rows in the database are used; cell range reference

The *database* argument is a range of cells that make up the database. Each column represents a field. The first row represents the field labels. Each remaining row represents a record of data.

The *field* argument determines which column in the database to use. The *field* argument can be a string (field label) or a number (field index).

The *criteria* argument is a range of cells that specify which rows in the database contain the conditions that select a subset of the data in the database. The first row represents field labels. The remaining rows represent conditions. Conditions in the same row are combined using an AND operation. Conditions in different rows are combined using an OR operation. Each condition can be a number or a string. The string can include a comparison operator (=, <>, <, >, <=, >=). If no operator is included then the equal operator (=) is assumed.

Wild card characters are not supported in the *criteria* argument.

Remarks

This is one of several database or list functions that treat a range of cells as if they were a database. For more details on this type of function, refer to *Database Functions* on page 9.

Data Types

Accepts cell ranges for database and criteria. Accepts a string or a number for field. Returns numeric data.

Examples

```
DVAR(A4:E10,"Type",A4:E10)
DVAR(A1:A9,3,D5:D8)
```

Version Available

This function is available in product version 2.5 or later.

See Also

DSTDEV | DSTDEVP | DVARP | DAVERAGE | DMIN | DMAX | Database Functions

DVARP

Summary

This function calculates the variance of a population based on the entire population by using the numbers in a column of a list or database that match the specified conditions.

Syntax

`DVARP(database, field, criteria)`

Arguments

Argument	Description
database	Range of cells that make up the database; cell range reference
field	Column in the database, referred to by label or index
criteria	Range of cells that specify which rows in the database are used; cell range reference

The *database* argument is a range of cells that make up the database. Each column represents a field. The first row represents the field labels. Each remaining row represents a record of data.

The *field* argument determines which column in the database to use. The *field* argument can be a string (field label) or a number (field index).

The *criteria* argument is a range of cells that specify which rows in the database contain the conditions that select a subset of the data in the database. The first row represents field labels. The remaining rows represent conditions. Conditions in the same row are combined using an AND operation. Conditions in different rows are combined using an OR operation. Each condition can be a number or a string. The string can include a comparison operator (=, <>, <, >, <=, >=). If no operator is included then the equal operator (=) is assumed.

Wild card characters are not supported in the *criteria* argument.

Remarks

This is one of several database or list functions that treat a range of cells as if they were a database. For more details on this type of function, refer to *Database Functions* on page 9.

Data Types

Accepts cell ranges for database and criteria. Accepts a string or a number for field. Returns numeric data.

Examples

```
DVARP(A4:E10,"Type",A4:E10)
DVARP(A1:A9,3,D5:D8)
```

Version Available

This function is available in product version 2.5 or later.

See Also

DSTDEV | DSTDEVP | DVAR | DAVERAGE | DMIN | DMAX | Database Functions

EDATE

Summary

This function calculates the date that is the indicated number of months before or after a specified date.

Syntax

`EDATE(startdate,months)`

Arguments

This function has these arguments:

Argument	Description
startdate	Starting date
months	Number of months before (negative) or after (positive) the starting date; if not an integer, the number is truncated

Remarks

Use this function to calculate maturity dates or due dates that fall on the same day of the month as the date of issue.

Data Types

Accepts numeric, string, or DateTime object data for the *startdate* argument and numeric data for the *months* argument. Returns a DateTime object.

Examples

`EDATE(A1,-6)`
`EDATE(R1C1,4)`
`EDATE("2004/01/09",2)` gives the result `3/9/2004 12:00:00 AM`

Version Available

This function is available in product version 1.0 or later.

See Also

DATE | EOMONTH | Date and Time Functions

EFFECT

Summary

This function calculates the effective annual interest rate for a given nominal annual interest rate and the number of compounding periods per year.

Syntax

`EFFECT(`*`nomrate`*`,`*`comper`*`)`

Arguments

This function has these arguments:

Argument	Description
nomrate	Nominal interest rate
comper	Number of compounding periods; if not an integer, the number is truncated

Remarks

The #VALUE! error is returned if either argument is nonnumeric. The #NUM error is returned if *nomrate* is less than or equal to zero or if *comper* is less than one.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`EFFECT(J12,B3)`
`EFFECT(R12C10,R3C2)`
`EFFECT(6.5%,8)` gives the result `0.66878782`

Version Available

This function is available in product version 1.0 or later.

See Also

INTRATE | NOMINAL | Financial Functions

EOMONTH

Summary

This function calculates the date for the last day of the month (end of month) that is the indicated number of months before or after the starting date.

Syntax

`EOMONTH(startdate,months)`

Arguments

This function has these arguments:

Argument	Description
startdate	Starting date
months	Number of months before (negative) or after (positive) the starting date; if not an integer, the number is truncated

Specify the date argument as a number (as in 37806.5) a string (as in "7/4/2003 12:00"), or a DateTime object, as in DATE(2003,7,4).

Data Types

Accepts numeric, string, or DateTime object data for the *startdate* argument and numeric data for the *months* argument. Returns a DateTime object.

Examples

`EOMONTH(A3,6)`
`EOMONTH(R3C1,-4)`
`EOMONTH("2004/01/09",2)` gives the result `3/31/2004 12:00:00 AM`

Version Available

This function is available in product version 1.0 or later.

See Also

EDATE | MONTH | Date and Time Functions

ERF

Summary

This function calculates the error function integrated between a lower and an upper limit.

Syntax

`ERF(limit,upperlimit)`

Arguments

This function has these arguments:

Argument	Description
limit	Either this is the lower limit, if the upper limit is supplied, or it is the upper limit (with 0 as the lower limit) if the second argument is not supplied
upperlimit	[Optional] Upper limit for integrating the function

Remarks

If *upperlimit* is supplied, the function is integrated from *limit* to *upperlimit*. If not supplied, the function is integrated from 0 to *limit*.

If there *upperlimit* is not supplied, the function calculates:

$$ERF(x) = \frac{2}{\pi}\int_{0}^{x}\left(e^{-t^2}\right)dt$$

where x is the *limit* argument.

If there *upperlimit* is supplied, the function calculates:

$$ERF(lo,\ hi) = \frac{2}{\pi}\int_{lo}^{hi}\left(e^{-t^2}\right)dt$$

where lo is the *limit* argument and hi is the *upperlimit* argument.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`ERF(K16)`
`ERF(R16C11,R16,C12)`
`ERF(0.49)` gives the result `0.51166826`
`ERF(0.25,0.85)` gives the result `0.494341544`

Version Available

This function is available in product version 1.0 or later.

See Also

ERFC | STEYX | Engineering Functions

ERFC

Summary

This function calculates the complementary error function integrated between a lower limit and infinity.

Syntax

`ERFC(lowerlimit)`

Arguments

The argument is the lower limit from which to integrate to infinity when calculating this function.

Remarks

This function calculates the complementary error function as follows:

$$ERFC(x) = \frac{2}{\pi}\int_{x}^{\infty}\left(e^{-t^2}\right)dt$$

where x is the lower limit specified in the argument.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`ERFC(K16)`
`ERFC(R16C11)`
`ERFC(0.49)` gives the result 0.48833174

Version Available

This function is available in product version 1.0 or later.

See Also

ERF | STEYX | Engineering Functions

ERRORTYPE

Summary

This function returns a number corresponding to one of the error values.

Syntax

`ERRORTYPE(`*`errorvalue`*`)`

Arguments

The valid error values that can be used in the arguments and their corresponding returned values are summarized here:

Error Value	Function Returns
#NULL!	1
#DIV/0!	2
#VALUE!	3
#REF!	4
#NAME?	5
#NUM!	6
#N/A	7

Remarks

You can use this function in an IF-THEN structure to test for the error value and return a text string, such as a message, instead of the error value.

Data Types

Accepts error value as data. Returns numeric data.

Examples

`ERRORTYPE(B13)`
`ERRORTYPE(R13C2)`
`ERRORTYPE(#REF!)` gives the result `4`

Version Available

This function is available in product version 1.0 or later.

See Also

ISERROR | Information Functions

EURO

Summary

This function returns the equivalent of one Euro based on the ISO currency code.

Syntax

`EURO(code)`

Arguments

The argument is the ISO currency code of certain countries. This function does not convert all currencies; only those Euro member currencies listed here.

Country/Region	ISO Currency Code
Belgium	BEF
Luxembourg	LUF
Germany	DEM
Spain	ESP
France	FRF
Ireland	IEP
Italy	ITL
Netherlands	NLG
Austria	ATS
Portugal	PTE
Finland	FIM
Euro member state	EUR

Remarks

ISO Currency Codes are from ISO 4217, the international standard describing three-letter codes to define the names of currencies. ISO is the nickname for the International Organization for Standardization.The first two letters of the code are the two-letter country codes (ISO 3166) and the third is usually the initial of the currency itself. So BEF is Belgium Franc.

Data Types

Accepts string data for the code. Returns numeric data.

Examples

`EURO(""BEF"")`

Version Available

This function is available in product version 2.0 or later.

See Also

EUROCONVERT | Financial Functions

EUROCONVERT

Summary

This function converts currency from a Euro member currency (including Euros) to another Euro member currency (including Euros).

Syntax

EUROCONVERT(*currency*,*source*,*target*,*fullprecision*,*triangulation*)

Arguments

This function has these arguments:

Argument	Description
currency	Number to convert
source	ISO currency code for the number to convert (see table below)
target	ISO currency code for the result of the conversion (see table below)
fullprecision	[Optional] Logical value representing whether to display the value in full precision or not; if omitted, the value is not displayed in full precision
triangulation	[Optional] Integer greater than or equal to 3 that specifies the number of significant digits to be used for the intermediate Euro value when converting between two Euro member currencies

If *triangulation* is omitted, the calculation does not round the intermediate Euro value. If it is included when converting from a Euro member currency to the Euro, the calculation finds the intermediate Euro value that could then be converted to a Euro member currency.

Remarks

This function does not convert all currencies; only those Euro member currencies listed in this table.

Country/Region	ISO Currency Code
Belgium	BEF
Luxembourg	LUF
Germany	DEM
Spain	ESP
France	FRF
Ireland	IEP
Italy	ITL
Netherlands	NLG
Austria	ATS
Portugal	PTE
Finland	FIM
Euro member state	EUR

ISO Currency Codes are from ISO 4217, the international standard describing three-letter codes to define the names of currencies. ISO is the nickname for the International Organization for Standardization.The first two letters of the code are the two-letter country codes (ISO 3166) and the third is usually the initial of the currency itself. So BEF is Belgium Franc.

Data Types

Accepts numeric and string data for most arguments; the *fullprecision* argument is a logical value. Returns numeric data.

Examples

```
EUROCONVERT(B5,"DEM","EUR")
EUROCONVERT(R5C2,"DEM","EUR",TRUE, 3)
```

Version Available

This function is available in product version 2.0 or later.

See Also

ROUND | Financial Functions

EVEN

Summary

This function rounds the specified value up to the nearest even integer.

Syntax

`EVEN(value)`

Arguments

The argument can be any numeric value.

Remarks

Regardless of the sign of the number specified by the argument, the number is rounded away from zero.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`EVEN(A3)`
`EVEN(R1C2)`
`EVEN(5)` gives the result `6`
`EVEN(-2.5)` gives the result `-4`

Version Available

This function is available in product version 1.0 or later.

See Also

CEILING | FLOOR | ODD | ISEVEN | Math and Trigonometry Functions

EXACT

Summary

This function returns true if two strings are the same; otherwise, false.

Syntax

`EXACT(`*`text1`*`,`*`text2`*`)`

Arguments

The arguments are text strings.

Remarks

This function compares the string in the first argument to the string in the second argument.

Although this function is case-sensitive, it ignores formatting differences.

Data Types

Accepts string data for both arguments. Returns boolean data (true or false).

Examples

`EXACT(A3,A5)`
`EXACT(R3C1,R5C1)`
`EXACT(""SPREAD"",""spread"")` gives the result `FALSE`

Version Available

This function is available in product version 1.0 or later.

See Also

CONCATENATE | Text Functions

EXP

Summary

This function returns e raised to the power of the specified value.

Syntax

`EXP(`*`value`*`)`

Arguments

The argument for this function is any numeric value.

Remarks

Mathematically, this function is (e^x).

This function is the inverse of LN, so EXP (LN(x)) results in x.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`EXP(B3)`
`EXP(R1C2)`
`EXP(1)` gives the result `2.7182818285`

Version Available

This function is available in product version 1.0 or later.

See Also

LN | LOG | POWER | Math and Trigonometry Functions

EXPONDIST

Summary

This function returns the exponential distribution or the probability density.

Syntax

`EXPONDIST(value,lambda,cumulative)`

Arguments

This function has these arguments:

Argument	Description
value	Value of the function; must be positive or zero
lambda	Parameter value; must be greater than zero
cumulative	Logical value indicating whether to return the cumulative distribution; set to TRUE to return the cumulative distribution; set to FALSE to return the probability density

Remarks

Use this function to model the time between events, such as how long an automated bank teller takes to deliver cash. For example, you can use this function to determine the probability that the process takes at most one minute.

The cumulative distribution is calculated as follows:

$$EXPONDIST(x,\ \lambda,\ FALSE)\ =\ \lambda e^{(-\lambda x)}$$

where x is the *value* argument, lambda is the *lambda* argument.

The probability density is calculated as follows:

$$EXPONDIST(x,\ \lambda,\ TRUE)\ =\ 1 - e^{(-\lambda x)}$$

where x is the *value* argument, lambda is the *lambda* argument.

Data Types

Accepts numeric data, except the third argument, which accepts logical data. Returns numeric data.

Examples

`EXPONDIST(C12,10,TRUE)`
`EXPONDIST(R12C3,8,FALSE)`
`EXPONDIST(0.2,10,TRUE)` gives the result `0.8646647168`

Version Available

This function is available in product version 1.0 or later.

See Also

BINOMDIST | Statistical Functions

FACT

Summary

This function calculates the factorial of the specified number.

Syntax

`FACT(`*`number`*`)`

Arguments

The argument can be any numeric value.

Remarks

The factorial is the product of the positive integers less than or equal to a number and is calculated as 1 x 2 x 3 x . . . x *number*, and is typically written as n! for n being the number. For example, 4! is 1 x 2 x 3 x 4, which is 24. The argument must be a non-negative number. If you provide a number that is not an integer for the argument, the decimal portion of the number is ignored.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`FACT(B3)`
`FACT(R1C2)`
`FACT(10)` gives the result `3628800`

Version Available

This function is available in product version 1.0 or later.

See Also

FACTDOUBLE | PRODUCT | Math and Trigonometry Functions

FACTDOUBLE

Summary

This function calculates the double factorial of the specified number.

Syntax

`FACTDOUBLE(`*`number`*`)`

Arguments

The argument can be any non-negative numeric value.

Remarks

The *number* argument must be a non-negative number. If you provide a number that is not an integer for the *number* argument, the decimal portion of the number is ignored. The double factorial is calculated as follows for even numbers:

```
n!! = n(n-2)(n-4) ... (4)(2)
```

The double factorial is calculated as follows for odd numbers:

```
n!! = n(n-2)(n-4) ... (3)(1)
```

Data Types

Accepts numeric data. Returns numeric data.

Examples

`FACTDOUBLE(E3)`
`FACTDOUBLE(R3C5)`
`FACTDOUBLE(6)` gives the result `48`

Version Available

This function is available in product version 1.0 or later.

See Also

FACT | PRODUCT | Math and Trigonometry Functions

FALSE

Summary

This function returns the value for logical FALSE.

Syntax

`FALSE()`

Remarks

This function does not accept arguments.

Data Types

Does not accept data. Returns numeric (boolean) data.

Example

`FALSE()` gives the result `0 (FALSE)`

Version Available

This function is available in product version 1.0 or later.

See Also

IF | TRUE | Logical Functions

FDIST

Summary

This function calculates the F probability distribution, to see degrees of diversity between two sets of data.

Syntax

`FDIST(value,degnum,degden)`

Arguments

This function has these arguments:

Argument	Description
value	Value at which to evaluate the function
degnum	Number of degrees of freedom for the numerator; if not an integer, the number is truncated
degden	Number of degrees of freedom for the denominator; if not an integer, the number is truncated

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`FDIST(A1,2,2)`
`FDIST(R1C1,2,1)`
`FDIST(16.83975,5,3)` gives the result `0.021`

Version Available

This function is available in product version 1.0 or later.

See Also

FINV | Statistical Functions

FIND

Summary

This function finds one text value within another and returns the text value's position in the text you searched.

Syntax

`FIND(findtext,intext,start)`

Arguments

This function has these arguments:

Argument	Description
findtext	Text you are trying to find; if empty (""), the function matches the first character in the search string (that is, the character numbered start or 1); cannot contain wildcard characters
intext	Text through which you are searching
start	[Optional] Number representing character at which to start the search; the first character of *intext* is 1; if omitted, the calculation starts at 1; if not an integer, the number is truncated

Remarks

This function performs a case-specific search (for example, to specify a capital letter and not lower case letters).

Data Types

Accepts string data for the findtext argument, string data for the intext argument, and numeric data for the start argument. Returns numeric data.

Examples

`FIND(""G"",A2,1)`
`FIND(""G"",R2C1,1)`
`FIND(""P"",""FarPoint Technologies"")` gives the result `4`
`FIND(""n"",""FarPoint Technologies"",8)` gives the result `4`

Version Available

This function is available in product version 1.0 or later.

See Also

REPLACE | SUBSTITUTE | Text Functions

FINV

Summary

This function returns the inverse of the F probability distribution.

Syntax

`FINV(`*p*`,`*degnum*`,`*degden*`)`

Arguments

This function has these arguments:

Argument	Description
p	Probability associated with the F cumulative distribution
degnum	Number of degrees of freedom for the numerator; if not an integer, the number is truncated
degden	Number of degrees of freedom for the denominator; if not an integer, the number is truncated

If either *degnum* or *degden* is not an integer, it is truncated.

Remarks

This function calculates the inverse of the F probability distribution, so if p = FDIST(x,...), then FINV(p,...) = x.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`FINV(A1,2,2)`
`FINV(R1C1,2,1)`
`FINV(0.021,5,3)` gives the result `16.83975`

Version Available

This function is available in product version 1.0 or later.

See Also

FDIST | Statistical Functions

FISHER

Summary

This function returns the Fisher transformation for a specified value.

Syntax

`FISHER(value)`

Arguments

Provide a numeric value that is less than 1 and greater than –1 for which you want the transformation.

Remarks

This transformation produces an approximately normal distribution. Use this function to perform hypothesis testing on the correlation coefficient.

The Fisher transformation is calculated as follows:

$$FISHER(x) = \frac{1}{2}\ln\frac{(1+x)}{(1-x)}$$

where x is the *value* argument.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`FISHER(A43)`
`FISHER(R4C12)`
`FISHER(-0.65)` gives the result `-0.7752987062`

Version Available

This function is available in product version 1.0 or later.

See Also

FISHERINV | Statistical Functions

FISHERINV

Summary

This function returns the inverse of the Fisher transformation for a specified value.

Syntax

`FISHERINV(`*`value`*`)`

Arguments

The argument is the specified numeric value.

Remarks

Use this transformation when analyzing correlations between ranges or arrays of data.

This function calculates the inverse of the Fisher transformation, so if y = FISHER(x), then FISHERINV(y) = x.

The inverse Fisher transformation is calculated as follows:

$$FISHERINV(y) = \frac{e^{2y} - 1}{e^{2y} + 1}$$

where y is the *value* argument.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`FISHERINV(A43)`
`FISHERINV(R4C12)`
`FISHERINV(0.56)` gives the result `0.5079774329`

Version Available

This function is available in product version 1.0 or later.

See Also

FISHER | Statistical Functions

FIXED

Summary

This function rounds a number to the specified number of decimal places, formats the number in decimal format using a period and commas (if so specified), and returns the result as text.

Syntax

`FIXED(`*`num`*`,`*`digits`*`,`*`notcomma`*`)`

Arguments

This function has these arguments:

Argument	Description
num	Number to round and convert to text
digits	[Optional] Number of decimal places; if omitted, uses two places
notcomma	[Optional] Logical value whether not to use commas; if omitted or FALSE, returns with commas

Data Types

Accepts numeric data for first two arguments; accepts logical value for the third argument. Returns string (text) data.

Examples

```
FIXED(B3)
FIXED(R3C2,2,FALSE)
FIXED(4.2365,3)
```

Version Available

This function is available in product version 1.0 or later.

See Also

DOLLAR | Text Functions

FLOOR

Summary

This function rounds a number down to the nearest multiple of a specified value.

Syntax

`FLOOR(value,signif)`

Arguments

This function has these arguments:

Argument	Description
value	Number to round
signif	Number representing the rounding factor

Use either both positive or both negative numbers for the arguments. Regardless of the sign of the numbers, the value is rounded toward zero.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`FLOOR(C4,B2)`
`FLOOR(B3,0.05)`
`FLOOR(R1C2,1)`
`FLOOR(4.65,2)` gives the result `4`
`FLOOR(-2.78,-1)` gives the result `-2`

Version Available

This function is available in product version 1.0 or later.

See Also

CEILING | EVEN | ODD | TRUNC | Math and Trigonometry Functions

FORECAST

Summary

This function calculates a future value using existing values.

Syntax

`FORECAST(`*`value`*`,`*`Yarray`*`,`*`Xarray`*`)`

Arguments

This function has these arguments:

Argument	Description
value	Value for which to predict the future dependent value
Yarray	An array of known dependent values (y's)
Xarray	An array of known independent values (x's)

Remarks

The predicted value is a y value for a given x value. The known values are existing x values and y values, and the new value is predicted by using linear regression. You can use this function to predict future sales, inventory requirements, or consumer trends.

This function is calculated as follows:

$$FORECAST(v,\ Y,\ X) = \bar{Y} - \left[\frac{n\sum xy - \sum x\sum y}{n\sum x - (\sum x)^2}\right]\bar{X} + \left[\frac{n\sum xy - \sum x\sum y}{n\sum x - (\sum x)^2}\right]v$$

where v is the *value* argument, Y is the *Yarray* argument, X is the *Xarray* argument, and n is the size of the arrays.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`FORECAST(30,G1:G9,F1:F9)`
`FORECAST(30,R1C7:R9C7,R1C6:R9C6)`
`FORECAST(45,{53000,57000,58000,69000,74500,55620,80000,`
`68700},{35,31,47,51,37,31,58,39})` gives the result `67060.8665320360`

Version Available

This function is available in product version 1.0 or later.

See Also

INTERCEPT | Statistical Functions

FREQUENCY

Summary

This function calculates how often values occur within a range of values. This function returns a vertical array of numbers.

Syntax

`FREQUENCY(`*`dataarray`*`,`*`binarray`*`)`

Arguments

This function has these arguments:

Argument	Description
dataarray	Array of values or a reference to a set of values for which to count frequencies
binarray	Array of intervals or a reference to intervals into which to group the values of *dataarray*

Remarks

The number of elements in the returned array is one greater than the number of elements in *binarray*. The extra element in the returned array is the count of values in *dataarray* that is above the highest value in *binarray*.

Use the INDEX function to get individual elements from the returned arrays.

Data Types

Accepts an array. Returns an array.

Examples

`FREQUENCY(A1:A7,C2:C5)`

Version Available

This function is available in product version 2.0 or later.

See Also

AVEDEV | AVERAGEA | CONFIDENCE | DEVSQ | MEDIAN | VAR | Statistical Functions

FTEST

Summary

This function returns the result of an F-test, which returns the one-tailed probability that the variances in two arrays are not significantly different.

Syntax

`FTEST(`*`array1`*`,`*`array2`*`)`

Arguments

The arguments may be arrays of values.

Data Types

Accepts arrays of numeric data for both arguments. Returns numeric data.

Examples

```
FTEST(A1:D34,A35:D68)
FTEST(R1C1:R34C4,R35C1:R68C4)
```

Version Available

This function is available in product version 1.0 or later.

See Also

ZTEST | TTEST | Statistical Functions

FV

Summary

This function returns the future value of an investment based on a present value, periodic payments, and a specified interest rate.

Syntax

`FV(rate,numper,paymt,pval,type)`

Arguments

This function has these arguments:

Argument	Description
rate	Interest rate expressed as percentage (per period)
numper	Total number of payment periods
paymt	Payment made each period
pval	[Optional] Present value; if omitted, uses zero and the calculation is based on the *paymt* argument.
type	[Optional] Indicates when payments are due; at the end (0) or beginning (1) of the period; if omitted, the calculation uses the end (0)

Remarks

Use consistent units for specifying the rate and number of periods arguments. If you make monthly payments on a five-year loan at 8 percent annual interest, use 0.08/12 for the rate argument and 5*12 for the number of periods argument. If you make annual payments on the same loan, use 0.08 for rate and 5 for number of periods.

For the arguments, money paid out (such as deposits in an investment) is represented by negative numbers; money you receive (such as dividend checks) is represented by positive numbers.

See the PV function for the equations for calculating financial values.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`FV(A1/12,48,B1,1000,0)`
`FV(R1C1/12,48,R1C2,1000,0)`
`FV(0.005,60,-100,100,1)` gives the result `$6877.00`

Version Available

This function is available in product version 1.0 or later.

See Also

FVSCHEDULE | NPER | PMT | PV | Financial Functions

FVSCHEDULE

Summary

This function returns the future value of an initial principal after applying a series of compound interest rates. Calculate future value of an investment with a variable or adjustable rate.

Syntax

`FVSCHEDULE(`*`principal`*`,`*`schedule`*`)`

Arguments

This function has these arguments:

Argument	Description
principal	Present value of the principal
schedule	Schedule, array of interest rates to apply

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`FVSCHEDULE(4,A1:C1)`
`FVSCHEDULE(45,R1C1:R7C1)`
`FVSCHEDULE(1000,{0.8,0.6,0.7})` gives the result `4896`

Version Available

This function is available in product version 1.0 or later.

See Also

FV | Financial Functions

GAMMADIST

Summary

This function returns the gamma distribution.

Syntax

`GAMMADIST(x,alpha,beta,cumulative)`

Arguments

This function has these arguments:

Argument	Description
x	Value at which to evaluate the distribution
alpha	Alpha parameter of the distribution
beta	Beta parameter of the distribution
cumulative	Logical value that determines the form of the function If cumulative is TRUE, then this function returns the cumulative distribution function; if FALSE, it returns the probability mass function.

Remarks

The equation for this function is:

$$GAMMADIST(x,\ \alpha,\ \beta,\ TRUE) = \frac{1}{\beta^{\alpha}\Gamma(\alpha)}x^{\alpha-1}e^{-x/\ \beta}$$

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`GAMMADIST(A5,1,3,FALSE)`
`GAMMADIST(R5C1,2,1,TRUE)`
`GAMMADIST(4,3,2,TRUE)` gives the result `0.3233235838`
`GAMMADIST(4,3,2,FALSE)` gives the result `0.1353352832`

Version Available

This function is available in product version 1.0 or later.

See Also

BETADIST | GAMMAINV | GAMMALN | KURT | POISSON | Statistical Functions

GAMMAINV

Summary

This function returns the inverse of the gamma cumulative distribution.

Syntax

`GAMMAINV(p,alpha,beta)`

Arguments

This function has these arguments:

Argument	Description
p	Probability
alpha	Alpha parameter of the distribution
beta	Beta parameter of the distribution

Remarks

This function calculates the inverse of the F probability distribution, so if p = GAMMADIST(x,...), then GAMMAINV(p,...) = x.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`GAMMAINV(A3,3,4)`
`GAMMAINV(0.8902,R3C8,R3C9)`
`GAMMAINV(0.75,2,3)` gives the result `8.0779035867`

Version Available

This function is available in product version 1.0 or later.

See Also

GAMMADIST | GAMMALN | Statistical Functions

GAMMALN

Summary

This function returns the natural logarithm of the Gamma function, G(x).

Syntax

`GAMMALN(`*`value`*`)`

Arguments

The argument is any numeric value.

Remarks

This function is calculated as the natural logarithm (LN) of the Gamma function.

The equation for this function is:

$$GAMMALN(x) = LN\left(\int_0^{\infty} e^{-u} u^{x-1} du\right)$$

where x is the *value* argument.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`GAMMALN(A4)`
`GAMMALN(R4C1)`
`GAMMALN(12)` gives the result `17.5023078459`

Version Available

This function is available in product version 1.0 or later.

See Also

GAMMADIST | GAMMAINV | LN | Statistical Functions

GCD

Summary

This function returns the greatest common divisor of two numbers.

Syntax

`GCD(`*`number1`*`,`*`number2`*`)`

Arguments

The arguments are two numeric values. If the arguments are not integers, they are truncated to integers. This function can have up to 255 arguments.

Remarks

The greatest common divisor is the largest integer that divides both numbers without a remainder.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`GCD(B5,G7)`
`GCD(R5C2,R7C7)`
`GCD(3348,972)` gives the result `108`
`GCD(12.8,16.3)` gives the result `4`

Version Available

This function is available in product version 1.0 or later.

See Also

LCM | Math and Trigonometry Functions

GEOMEAN

Summary

This function returns the geometric mean of a set of positive data.

Syntax

GEOMEAN(*value1*,*value2*,...)
GEOMEAN(*array*)
GEOMEAN(*array1*,*array2*,...)

Arguments

You can specify a set of numeric values. You can also use a single array or a reference to an array instead of arguments separated by commas. If an array or reference argument contains text, logical values, or empty cells, the function ignores those values; however, the function includes in calculations cells with the value zero. This function can have up to 255 arguments.

Data should be provided so that the value arguments should be greater than zero.

Remarks

You can use this function to calculate average growth rate given compound interest with variable rates.

The equation for this function is:

$$GEOMEAN(x_1,\ x_2,\ \ldots,\ x_n) = \sqrt[n]{x_1 x_2 \ldots x_n}$$

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

GEOMEAN(F1:F9)
GEOMEAN(R1C6:R9C6)
GEOMEAN(35,31,47,51,37,31,58,39) gives the result 40.1461796637

Version Available

This function is available in product version 1.0 or later.

See Also

HARMEAN | Statistical Functions

GESTEP

Summary

This function, greater than or equal to step, returns an indication of whether a number is equal to a threshold.

Syntax

`GESTEP(number,step)`

Arguments

This function has these arguments:

Argument	Description
number	Value to test against the step (which is either step or zero)
step	[Optional] Value of the threshold against which to test; if omitted, uses zero

Remarks

If the *number* is greater than or equal to the *step*, this function returns one. Otherwise it returns zero.

Data Types

Accepts numeric data for all arguments. Returns numeric (0 or 1) data.

Examples

`GESTEP(B5,7)`
`GESTEP(43)` gives the result `1`

Version Available

This function is available in product version 1.0 or later.

See Also

DELTA | Engineering Functions

GROWTH

Summary

This function calculates predicted exponential growth. This function returns the y values for a series of new x values that are specified by using existing x and y values.

Syntax

`GROWTH(y,x,newx,constant)`

Remarks

This function has these arguments:

Argument	Description
y	Set of y values that are known in the relationship y=b*m^x
x	(Optional) X is an optional set of x values that may be known in the relationship y=b*m^x
newx	New x values for which this functions returns the corresponding y values
constant	Logical value that specifies whether to force the constant b to equal 1

If *constant* is true or omitted then b is calculated normally. If constant is false then b is equal to 0 and the m values are adjusted so that y=m^x.

If *x* is omitted then *x* defaults to the array {1,2,3...}, that has the same dimensions as *y*.

If *newx* is omitted then it defaults to x.

Remarks

Use the INDEX function to get individual elements from the returned array.

Data Types

Accepts an array. Returns an array.

Examples

`GROWTH(A2:A7,C2:C7,A9:A10)`

Version Available

This function is available in product version 2.0 or later.

See Also

AVEDEV | AVERAGEA | FREQUENCY | DEVSQ | MEDIAN | TREND | VAR | Statistical Functions

HARMEAN

Summary

This function returns the harmonic mean of a data set.

Syntax

`HARMEAN(value1,value2,...)`
`HARMEAN(array)`
`HARMEAN(array1,array2,...)`

Arguments

You can specify a set of numeric values. You can also use a single array or a reference to an array instead of arguments separated by commas. If an array or reference argument contains text, logical values, or empty cells, the function ignores those values; however, the function includes in calculations cells with the value zero. This function can have up to 255 arguments.

Data should be provided so that the value arguments should be greater than zero.

Remarks

The harmonic mean is always less than the geometric mean, which is always less than the arithmetic mean

The equation for this function is:

$$HARMEAN(x_n) = \frac{1}{\frac{1}{n}\sum\frac{1}{X}}$$

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`HARMEAN(F1:F9)`
`HARMEAN(R1C6:R9C6)`
`HARMEAN(35,31,47,51,37,31,58,39)` gives the result `39.2384929823`

Version Available

This function is available in product version 1.0 or later.

See Also

GEOMEAN | Statistical Functions

HEX2BIN

Summary

This function converts a hexadecimal number to a binary number.

Syntax

`HEX2BIN(number, places)`

Arguments

This function has these arguments:

Argument	Description
number	Hexadecimal numeric value to convert, must be between FFFFFFFE00 and 1FF
places	[Optional] Number of characters to return; if not an integer, the number is truncated

Remarks

This functions returns an error when the *number* is not a valid hexadecimal value or if the value for *places* is non-numeric or negative. If *places* is omitted, the calculation uses the minimum number of characters necessary. This argument is useful for adding leading zeros to the result.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`HEX2BIN("F",5)`

Version Available

This function is available in product version 2.0 or later.

See Also

HEX2DEC | HEX2OCT | BIN2HEX | OCT2HEX | Engineering Functions

HEX2DEC

Summary

This function converts a hexadecimal number to a decimal number.

Syntax

`HEX2DEC(number)`

Arguments

Specify the number to convert, which is limited to a maximum of 10 characters.

Remarks

An error value is returned if the *number* is invalid or more than 10 characters.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`HEX2DEC("FF")`

Version Available

This function is available in product version 2.0 or later.

See Also

HEX2BIN | HEX2OCT | BIN2DEC | OCT2DEC | Engineering Functions

HEX2OCT

Summary

This function converts a hexadecimal number to an octal number.

Syntax

`HEX2OCT(number, places)`

Arguments

This function has these arguments:

Argument	Description
number	Hexadecimal numeric value to convert, must be between FFE0000000 and 1FFFFFFF
places	[Optional] Number of characters to return; if not an integer, the number is truncated

Remarks

This functions returns an error when the *number* is not a valid hexadecimal number or if the value for *places* is non-numeric or negative. If *places* is omitted, the calculation uses the minimum number of characters necessary. This argument is useful for adding leading zeros to the result.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`HEX2OCT("2B")`

Version Available

This function is available in product version 2.0 or later.

See Also

HEX2BIN | HEX2DEC | BIN2OCT | DEC2OCT | Engineering Functions

HLOOKUP

Summary

This function searches for a value in the top row and then returns a value in the same column from a specified row.

Syntax

`HLOOKUP(value,array,row,approx)`

Arguments

This function has these arguments:

Argument	Description
value	Value to be found in the first row
array	Array or range that contains the data to search
row	Row number in the array from which the matching value will be returned
approx	[Optional] Logical value indicating whether to find an approximate match; if omitted, uses TRUE and finds an approximate match

Remarks

If *approx* is FALSE, it finds an exact match, not an approximate match. If it cannot find one, it returns an #N/A error value.

If *approx* is TRUE or omitted, and the *value* cannot be found, then the largest value that is less than the *value* is used.

This function is similar to VLOOKUP except that it searches by row (horizontally), instead of vertically (by column).

Data Types

Accepts numeric or string data. Returns numeric data.

Examples

`HLOOKUP("Test",A1:D5,3,TRUE)`

Version Available

This function is available in product version 2.0 or later.

See Also

VLOOKUP | LOOKUP | Lookup Functions

HOUR

Summary

This function returns the hour that corresponds to a specified time.

Syntax

`HOUR(`*`time`*`)`

Arguments

Specify the *time* argument as a number (as in 37806.5) a string (as in "7/4/2003 12:00"), a DateTime object, as in DATE(2003,7,4), or a TimeSpan object, as in TIME(12,0,0). For more details on the date and time inputs, refer to the discussion in *Date and Time Functions* on page 10

Dates as numeric values are in the form x.y, where x is the "number of days since December 30, 1899" and y is the fraction of day. Numbers to the left represent the date. Times as numeric values are decimal fractions ranging from 0 to 0.99999999, representing the times from 0:00:00 (12:00:00 A.M.) to 23:59:59 (11:59:59 P.M.).

Remarks

The hour is returned as an integer, ranging from 0 (12:00 A.M.) to 23 (11:00 P.M.).

Data Types

Accepts numeric, string, DateTime object, or TimeSpan object data. Returns numeric data.

Examples

`HOUR(A2)`
`HOUR(R2C1)`
`HOUR(0.25)` gives the result `6`
`HOUR(347.25)` gives the result `6`
`HOUR("2:22 PM")` gives the result `14`
`HOUR("2:22 AM")` gives the result `2`
`HOUR(TIME(12,0,0))`

Version Available

This function is available in product version 1.0 or later.

See Also

MINUTE | SECOND | Date and Time Functions

HYPGEOMDIST

Summary

This function returns the hypergeometric distribution.

Syntax

HYPGEOMDIST(*x*,*n*,*M*,*N*)

Arguments

The arguments are as follows, and are truncated if not integers:

Argument	Description
x	An integer representing the number of successes in the sample
n	An integer representing the size of the sample
M	An integer representing the number of successes in the population
N	An integer representing the size of the population

Remarks

The equation for this function is:

$$HYPGEOMDIST(x,\ n,\ M,\ N) = \frac{\binom{M}{x}\binom{N-M}{n-x}}{\binom{N}{n}}$$

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

HYPGEOMDIST(A22,B23,62,1000)
HYPGEOMDIST(R22C11,R22C12,R34C14,R35C15)
HYPGEOMDIST(2,37,6,100) gives the result 0.3327981975

Version Available

This function is available in product version 1.0 or later.

See Also

BINOMDIST | GAMMADIST | Statistical Functions

IF

Summary

This function performs a comparison and returns one of two provided values based on that comparison.

Syntax

`IF(`*`valueTest`*`,`*`valueTrue`*`,`*`valueFalse`*`)`

Arguments

This function has these arguments:

Argument	Description
valueTest	Value or expression to evaluate
valueTrue	Value to return if the test evaluates to true
valueFalse	Value to return if the test evaluates to false

Remarks

The value of *valueTest* is evaluated. If it is non-zero (or TRUE), then *valueTrue* is returned. If it is zero (or FALSE), then *valueFalse* is returned. The value of *valueTest* must be or evaluate to numeric data, where non-zero values indicate TRUE, and a value of zero indicates FALSE. It may contain one of the relational operators: greater than (>), less than (<), equal to (=), or not equal to (<>).

Data Types

Accepts numeric (boolean) data. Returns any data type.

Example

`IF(A3<>2000,1900,2000)`
`IF(R1C2>65,1000,2000)`
`IF(C4,B2,B4)`
`IF(1>2,5,10)` gives the result `10`
`IF(1<2,""dogs"",""cats"")` gives the result `dogs`

Version Available

This function is available in product version 1.0 or later.

See Also

AND | FALSE | Logical Functions

IMABS

Summary

This function returns the absolute value or modulus of a complex number.

Syntax

`IMABS(complexnum)`

Arguments

The *complexnum* argument is a complex number for which to return the absolute value.

Remarks

An error is returned if the *complexnum* argument is not in the form "x+yi" or "x+yj". For more information, refer to *Complex Numbers in Engineering Functions* on page 11.

Data Types

Accepts number and string data. Returns number data.

Examples

`IMABS("3+5j")`

Version Available

This function is available in product version 2.0 or later.

See Also

COMPLEX | Engineering Functions | Complex Numbers in Engineering Functions

IMAGINARY

Summary

This function returns the imaginary coefficient of a complex number.

Syntax

`IMAGINARY(`*`complexnum`*`)`

Arguments

The *complexnum* argument is a complex number for which to return the imaginary coefficient.

Remarks

An error is returned if the *complexnum* argument is not in the form "x+yi" or "x+yj". For more information, refer to *Complex Numbers in Engineering Functions* on page 11.

Data Types

Accepts number and string data. Returns number data.

Examples

`IMAGINARY("3+5j")`

Version Available

This function is available in product version 2.0 or later.

See Also

COMPLEX | IMREAL | Engineering Functions | Complex Numbers in Engineering Functions

IMARGUMENT

Summary

This function returns the argument theta, which is an angle expressed in radians.

Syntax

`IMARGUMENT(`*`complexnum`*`)`

Arguments

The *complexnum* argument is a complex number for which to return the argument theta.

Remarks

An error is returned if the *complexnum* argument is not in the form "x+yi" or "x+yj". For more information, refer to *Complex Numbers in Engineering Functions* on page 11.

Remarks

The *complexnum* argument is a complex number for which to return the argument theta.

An error is returned if number is not in the form "x+yi" or "x+yj". For more information, refer to *Complex Numbers in Engineering Functions* on page 11.

Data Types

Accepts number and string data. Returns number data.

Examples

`IMARGUMENT("3+5j")`

Version Available

This function is available in product version 2.0 or later.

See Also

COMPLEX | IMCOS | IMSIN | Engineering Functions | Complex Numbers in Engineering Functions

IMCONJUGATE

Summary

This function returns the complex conjugate of a complex number.

Syntax

`IMCONJUGATE(complexnum)`

Arguments

The *complexnum* argument is a complex number for which to return the conjugate.

Remarks

An error is returned if the *complexnum* argument is not in the form "x+yi" or "x+yj". For more information, refer to *Complex Numbers in Engineering Functions* on page 11.

Data Types

Accepts number and string data. Returns string data.

Examples

`IMCONJUGATE("3+5j")`

Version Available

This function is available in product version 2.0 or later.

See Also

COMPLEX | IMABS | Engineering Functions | Complex Numbers in Engineering Functions

IMCOS

Summary

This function returns the cosine of a complex number.

Syntax

`IMCOS(`*`complexnum`*`)`

Arguments

The *complexnum* argument is a complex number for which to return the cosine.

Remarks

An error is returned if the *complexnum* argument is not in the form "x+yi" or "x+yj". For more information, refer to *Complex Numbers in Engineering Functions* on page 11.

Data Types

Accepts number and string data. Returns string data.

Examples

`IMCOS("3+5j")`

Version Available

This function is available in product version 2.0 or later.

See Also

COMPLEX | IMSIN | IMARGUMENT | Engineering Functions | Complex Numbers in Engineering Functions

IMDIV

Summary

This function returns the quotient of two complex numbers.

Syntax

`IMCOS(`*`complexnum`*`,`*`complexdenom`*`)`

Arguments

This function has these arguments:

Argument	Description
complexnum	Complex numerator or dividend
complexdenom	Complex denominator or divisor

Remarks

An error is returned if the arguments are not in the form "x+yi" or "x+yj". For more information, refer to *Complex Numbers in Engineering Functions* on page 11.

Data Types

Accepts number and string data. Returns string data.

Examples

`IMDIV("3+5j","10+20i")`

Version Available

This function is available in product version 2.0 or later.

See Also

IMPRODUCT | IMSQRT | Engineering Functions | Complex Numbers in Engineering Functions

IMEXP

Summary

This function returns the exponential of a complex number.

Syntax

`IMEXP(`*`complexnum`*`)`

Arguments

The *complexnum* argument is a complex number for which to return the exponential.

Remarks

An error is returned if the *complexnum* argument is not in the form "x+yi" or "x+yj". For more information, refer to *Complex Numbers in Engineering Functions* on page 11.

Data Types

Accepts number and string data. Returns string data.

Examples

`IMEXP("2+5j")`

Version Available

This function is available in product version 2.0 or later.

See Also

IMLN | IMLOG10 | IMLOG2 | IMPOWER | Engineering Functions | Complex Numbers in Engineering Functions

IMLN

Summary

This function returns the natural logarithm of a complex number.

Syntax

`IMLN(complexnum)`

Arguments

The *complexnum* argument is a complex number for which to return the natural logarithm.

Remarks

An error is returned if the *complexnum* argument is not in the form "x+yi" or "x+yj". For more information, refer to *Complex Numbers in Engineering Functions* on page 11.

Data Types

Accepts number and string data. Returns string data.

Examples

`IMLN("2+5j")`

Version Available

This function is available in product version 2.0 or later.

See Also

IMEXP | IMLOG10 | IMLOG2 | Engineering Functions | Complex Numbers in Engineering Functions

IMLOG10

Summary

This function returns the common logarithm of a complex number.

Syntax

`IMLOG10(complexnum)`

Arguments

The *complexnum* argument is a complex number for which to return the common logarithm.

Remarks

An error is returned if the *complexnum* argument is not in the form "x+yi" or "x+yj". For more information, refer to *Complex Numbers in Engineering Functions* on page 11.

Data Types

Accepts number and string data. Returns string data.

Examples

`IMLOG10("2+5j")`

Version Available

This function is available in product version 2.0 or later.

See Also

IMEXP | IMLN | IMLOG2 | Engineering Functions | Complex Numbers in Engineering Functions

IMLOG2

Summary

This function returns the base-2 logarithm of a complex number.

Syntax

`IMLOG2(complexnum)`

Arguments

The *complexnum* argument is a complex number for which to return the base-2 logarithm.

Remarks

An error is returned if the *complexnum* argument is not in the form "x+yi" or "x+yj". For more information, refer to *Complex Numbers in Engineering Functions* on page 11.

Data Types

Accepts number and string data. Returns string data.

Examples

`IMLOG2("2+5j")`

Version Available

This function is available in product version 2.0 or later.

See Also

IMEXP | IMLN | IMLOG10 | Engineering Functions | Complex Numbers in Engineering Functions

IMPOWER

Summary

This function returns a complex number raised to a power.

Syntax

`IMPOWER(`*`complexnum`*`,`*`powernum`*`)`

Arguments

This function has these arguments:

Argument	Description
complexnum	Complex number to raise to a power
powernum	Power to which to raise the complex number

The power (*powernum* argument) can be an integer, negative, or fractional.

Remarks

An error is returned if *complexnum* is not in the form "x+yi" or "x+yj" or if *powernum* is non-numeric. For more information, refer to *Complex Numbers in Engineering Functions* on page 11.

Data Types

Accepts number and string data. Returns string data.

Examples

`IMPOWER("2+5j",4)`

Version Available

This function is available in product version 2.0 or later.

See Also

IMEXP | IMPRODUCT | Engineering Functions | Complex Numbers in Engineering Functions

IMPRODUCT

Summary

This function returns the product of up to 29 complex numbers in the "x+yi" or "x+yj" text format.

Syntax

`IMPRODUCT(`*`complexnum1`*`,`*`complexnum2`*`, ...)`

Arguments

The arguments are the complex numbers to multiply. There can be up to 29 of them.

Arrays in the x+yi format or range references are allowed.

Remarks

An error is returned if the arguments are not in the form "x+yi" or "x+yj". For more information, refer to *Complex Numbers in Engineering Functions* on page 11.

Data Types

Accepts number and string data. Returns string data.

Examples

```
IMPRODUCT("2+5j",4)
IMPRODUCT({"1+2i","3+4i"})
```

Version Available

This function is available in product version 2.0 or later.

See Also

IMDIV | IMPOWER | Engineering Functions | Complex Numbers in Engineering Functions

IMREAL

Summary

This function returns the real coefficient of a complex number in the x+yi or x+yj text format.

Syntax

`IMREAL(`*`complexnum`*`)`

Arguments

The *complexnum* argument is a complex number for which to return the real coefficient.

Remarks

An error is returned if the *complexnum* argument is not in the form "x+yi" or "x+yj". For more information, refer to *Complex Numbers in Engineering Functions* on page 11.

Data Types

Accepts number and string data. Returns number data.

Examples

`IMEXP("2-5j")`

Version Available

This function is available in product version 2.0 or later.

See Also

COMPLEX | IMAGINARY | Engineering Functions | Complex Numbers in Engineering Functions

IMSIN

Summary

This function returns the sine of a complex number in the x+yi or x+yj text format.

Syntax

`IMSIN(`*`complexnum`*`)`

Arguments

The *complexnum* argument is a complex number for which to return the sine.

Remarks

An error is returned if the *complexnum* argument is not in the form "x+yi" or "x+yj". For more information, refer to *Complex Numbers in Engineering Functions* on page 11.

Data Types

Accepts number and string data. Returns string data.

Examples

`IMSIN("2-5j")`

Version Available

This function is available in product version 2.0 or later.

See Also

IMCOS | IMARGUMENT | Engineering Functions | Complex Numbers in Engineering Functions

IMSQRT

Summary

This function returns the square root of a complex number in the x+yi or x+yj text format.

Syntax

`IMSQRT(`*`complexnum`*`)`

Arguments

The *complexnum* argument is a complex number for which to return the square root.

Remarks

An error is returned if the *complexnum* argument is not in the form "x+yi" or "x+yj". For more information, refer to *Complex Numbers in Engineering Functions* on page 11.

Data Types

Accepts number and string data. Returns string data.

Examples

`IMSQRT("2-5j")`

Version Available

This function is available in product version 2.0 or later.

See Also

IMDIV | IMPRODUCT | Engineering Functions | Complex Numbers in Engineering Functions

IMSUB

Summary

This function returns the difference of two complex numbers in the x+yi or x+yj text format.

Syntax

`IMSUB(`*`complexnum1`*`,`*`complexnum2`*`)`

Arguments

The *complexnum1* is a complex number from which to subtract the other complex number *complexnum2*.

Remarks

An error is returned if the arguments are not in the form "x+yi" or "x+yj". For more information, refer to *Complex Numbers in Engineering Functions* on page 11.

Data Types

Accepts number and string data. Returns string data.

Examples

`IMSUB("2+5j","5+3i")`

Version Available

This function is available in product version 2.0 or later.

See Also

COMPLEX | IMSUM | Engineering Functions | Complex Numbers in Engineering Functions

IMSUM

Summary

This function returns the sum of two or more complex numbers in the x+yi or x+yj text format.

Syntax

`IMSUM(`*`complexnum1`*`,`*`complexnum2, ...`*`)`

Arguments

The arguments are the complex numbers to multiply. There can be up to 29 of them.

Arrays in the "x+yi" or "x+yj" format or range references are allowed.

Remarks

An error is returned if the arguments are not in the form "x+yi" or "x+yj". For more information, refer to *Complex Numbers in Engineering Functions* on page 11.

Data Types

Accepts number and string data. Returns string data.

Examples

```
IMSUM("2+5j","5+3i")
IMSUM(A1:B5)
IMSUM({"2+5j","5+3i"})
```

Version Available

This function is available in product version 2.0 or later.

See Also

COMPLEX | IMSUB | Engineering Functions | Complex Numbers in Engineering Functions

INDEX

Summary

This function returns a value or the reference to a value from within an array or range.

Syntax

`INDEX(`*`return`*`,`*`row`*`,`*`col`*`,`*`area`*`)`

Arguments

The arguments are as follows, and are truncated if not integers:

Argument	Description
return	Returns a value or a reference of a cell or range of cells
row	Row number in the range
col	Column number in the range
area	[If *return* is a cell range reference] Area of the range

Data Types

Accepts numeric data. Returns numeric data.

Examples

```
INDEX(B4,5,3)
INDEX(R4C2,5,3)
```

Version Available

This function is available in product version 1.0 or later.

See Also

CHOOSE | Lookup Functions

INT

Summary

This function rounds a specified number down to the nearest integer.

Syntax

`INT(`*`value`*`)`

Arguments

Use any numeric value for the argument.

Remarks

You can use this function to return the decimal portion of the value in a cell by subtracting the value of this function for the cell from the value in the cell, as illustrated in the first example.

The TRUNC and INT functions are similar in that both return integers. Use the TRUNC function to remove the decimal portion of the number; the TRUNC function does not round up or down. Use the INT function to round numbers down to the nearest integer-based decimal portion of the number.

These functions differ also when using negative numbers: TRUNC(–4.2) returns –4, but INT(–4.2) returns –5 because –5 is the lower number.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`INT(A3)`
`R1C2-INT(R1C2)`
`INT(2.85)` gives the result `2`
`INT(-2.85)` gives the result `-3`

Version Available

This function is available in product version 1.0 or later.

See Also

CEILING | EVEN | FLOOR | TRUNC | Math and Trigonometry Functions

INTERCEPT

Summary

This function returns the coordinates of a point at which a line intersects the y-axis, by using existing x values and y values.

Syntax

`INTERCEPT(`*`dependent`*`,`*`independent`*`)`

Arguments

This function has these arguments:

Argument	Description
dependent	An array of known dependent values (y's)
independent	An array of known independent values (x's)

You can use numbers, arrays, or references for the arguments.

Remarks

The intercept point is based on a best-fit regression line plotted through the known x-values and known y-values. Use the intercept when you want to determine the value of the dependent variable when the independent variable is 0 (zero). For example, you can use this function to predict a metal's electrical resistance at 0°C when your data points were taken at room temperature and higher.

If an array or reference argument contains text, logical values, or empty cells, the function ignores those values; however, the function includes in calculations cells with the value zero.

The number of dependent data points must be equal to the number of independent data points.

The equation for this function is:

$$INTERCEPT(Y,\ X) = \bar{Y} - \left[\frac{n\sum xy - \sum x\sum y}{n\sum x - (\sum x)^2}\right]\bar{X}$$

where Y is the array of dependent variables, X is the array of independent variables, and n is the size of the arrays.

Data Types

Accepts arrays of numeric data for both arguments. Returns numeric data.

Examples

`INTERCEPT(G1:G9,F1:F9)`
`INTERCEPT(R1C7:R9C7,R1C6:R9C6)`
`INTERCEPT({53000,57000,58000,69000,74500,55620,80000,`
`68700},{35,31,47,51,37,31,58,39})` gives the result `37060.4809987149`

Version Available

This function is available in product version 1.0 or later.

See Also

FORECAST | Statistical Functions

INTRATE

Summary

This function calculates the interest rate for a fully invested security.

Syntax

`INTRATE(settle,mature,invest,redeem,basis)`

Arguments

This function has these arguments:

Argument	Description
settle	Settlement date for the security.
mature	Maturity date for the security.
invest	Amount invested in the security.
redeem	Amount to be received at maturity.
basis	[Optional] Integer representing the basis for day count (Refer to *Day Count Basis* on page 13.)

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`INTRATE(A1,B3,70000,72000,3)`
`INTRATE(R1C1,R4C4,82000,86500,2)`
`INTRATE("3/1/2003","5/31/2003",65000,70000,2)` gives the result `0.304311074`

Version Available

This function is available in product version 1.0 or later.

See Also

ACCRINT | EFFECT | RATE | RECEIVED | Financial Functions

IPMT

Summary

This function calculates the payment of interest on a loan.

Syntax

`IPMT(`*`rate`*`,`*`per`*`,`*`nper`*`,`*`pval`*`,`*`fval`*`,`*`type`*`)`

Arguments

This function has these arguments:

Argument	Description
rate	Value of interest rate per period.
per	Number of the period for which to find the interest, between 1 and *nper*
nper	Total number of payment periods in an annuity.
pval	Present value, worth now
fval	[Optional] Future value, cash value after the last payment; if omitted, the calculation uses zero
type	[Optional] Indicates when payments are due; at the end (0) or beginning (1) of the period; if omitted, the calculation uses the end (0)

Remarks

The result is represented by a negative number because it is money paid out by you.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`IPMT(0.65,A1,B3,C42)`
`IPMT(R1C1,R12C12,R13C13,R32C1)`
`IPMT(0.45, 2, 30, 6000)` gives the result `-$2,699.98`

Version Available

This function is available in product version 1.0 or later.

See Also

PMT | PPMT | RATE | Financial Functions

IRR

Summary

This function returns the internal rate of return for a series of cash flows represented by the numbers in an array.

Syntax

`IRR(arrayvals,estimate)`

Remarks

This function has these arguments:

Argument	Description
arrayvals	An array of numbers for which you want to estimate the internal rate of return representing payments and income occurring at regular intervals (and use positive for income and negative for payment)
estimate	[Optional] An estimate of the internal rate of return; if omitted, the calculation uses 0.1 (10 percent)

Values must contain at least one positive value (some income) and one negative value (a payment) to calculate the internal rate of return.

Remarks

This function uses the order of values to interpret the order of payments and income. Be sure to enter your payment and income values in the sequence you want with correct signs.
The payments and income must occur at regular time intervals, such as monthly or annually.

If an array or reference argument contains text, logical values, or empty cells, the function ignores those values; however, the function includes in calculations cells with the value zero.

The function is calculated using an iterative technique. Starting with the estimate, this function cycles through the calculation until the result is accurate within 0.00001 (0.001 percent). If this function cannot find a result that works after 50 iterations, it returns an error.

If the function returns an error or if the result is not close to what you expected, try again with a different value for the estimate.

This function is closely related to NPV, the net present value function. The rate of return calculated by IRR is the interest rate corresponding to a 0 (zero) net present value.

For a schedule of cash flows that is non-periodic, use XIRR.

Data Types

Accepts numeric data for both arguments, the first being an array. Returns numeric data.

Examples

```
IRR(D1:D6)
IRR(R1C4:R6C4, -.02)
```

Version Available

This function is available in product version 1.0 or later.

See Also

MIRR | NPV | XIRR | Financial Functions

ISBLANK

Summary

This function tests whether a value, an expression, or contents of a referenced cell is empty.

Syntax

`ISBLANK(`*`cellreference`*`)`
`ISBLANK(`*`value`*`)`
`ISBLANK(`*`expression`*`)`

Arguments

Specify a cell reference, a numeric or text value, or an expression for the argument.

Remarks

Use this function to test the contents of a cell, a numeric or text value directly, or a function or expression.

This function returns TRUE if the value refers to an empty cell or to no data.

Note: Spread's implementation of functions generally tries to follow the behavior found in popular spreadsheet applications. However, not all these applications agree whether the empty string "" should be treated the same as an empty cell. In Spread, both the COUNTBLANK and ISBLANK functions consistently treat the empty string "" differently than an empty cell.

Data Types

Accepts any data type for an argument. Returns Boolean (TRUE or FALSE) data.

Examples

`ISBLANK(B1)`
`ISBLANK(A4)`
`ISBLANK(A4-52)`
`ISBLANK(4)` gives the result `FALSE`

Version Available

This function is available in product version 1.0 or later.

See Also

COUNTBLANK | ISERROR | ISREF | ISTEXT | Information Functions

ISERR

Summary

This function, Is Error Other Than Not Available, tests whether a value, an expression, or contents of a referenced cell has an error other than not available (#N/A).

Syntax

`ISERR(`*`cellreference`*`)`
`ISERR(`*`value`*`)`
`ISERR(`*`expression`*`)`

Arguments

Specify a cell reference, a numeric or text value, or an expression for the argument.

Remarks

Use this function to test the contents of a cell, a numeric or text value directly, or a function or expression.

This function returns TRUE if the value refers to an empty cell or to no data.

Data Types

Accepts any data type for an argument. Returns Boolean (TRUE or FALSE) data.

Examples

`ISERR(B12)`
`ISERR(R12C2)`
`ISERR(#N/A)` gives the result `FALSE`
`ISERR(#REF!)` gives the result `TRUE`
`ISERR(C14)` gives the result `TRUE` if C14 contains a #NUM! error.

Version Available

This function is available in product version 1.0 or later.

See Also

ERRORTYPE | ISERROR | ISNA | Information Functions

ISERROR

Summary

This function, Is Error of Any Kind, tests whether a value, an expression, or contents of a referenced cell has an error of any kind.

Syntax

`ISERROR(`*`cellreference`*`)`
`ISERROR(`*`value`*`)`
`ISERROR(`*`expression`*`)`

Arguments

Specify a cell reference, a numeric or text value, or an expression for the argument.

Remarks

Use this function to test the contents of a cell, a numeric or text value directly, or a function or expression.

This function returns TRUE if the value refers to an empty cell or to no data.

Data Types

Accepts any data type for an argument. Returns Boolean (TRUE or FALSE) data.

Examples

`ISERROR(B12)`
`ISERROR(R12C2)`
`ISERROR(#N/A)` gives the result `TRUE`
`ISERROR(#REF!)` gives the result `TRUE`
`ISERR(C14)` gives the result `TRUE` if C14 contains a #NUM! error.

Version Available

This function is available in product version 1.0 or later.

See Also

ERRORTYPE | ISERR | ISNA | Information Functions

ISEVEN

Summary

This function, Is Number Even, tests whether a value, an expression, or contents of a referenced cell is even.

Syntax

`ISEVEN(cellreference)`
`ISEVEN(value)`
`ISEVEN(expression)`

Arguments

Specify a cell reference, a numeric or text value, or an expression for the argument.

Remarks

If the number specified by the argument is even, the function returns TRUE. If the number specified by the argument is odd, the function returns FALSE. If the number specified by the argument is zero, the function returns TRUE. If the number specified by the argument refers to an empty cell or to no data, the function returns TRUE.

Use this function to test the contents of a cell, a numeric or text value directly, or a function or expression.

Data Types

Accepts numeric data. Returns Boolean (TRUE or FALSE) data.

Examples

`ISEVEN(B3)`
`ISEVEN(R1C2)`
`ISEVEN(574)` gives the result `TRUE`
`ISEVEN(9)` gives the result `FALSE`
`ISEVEN(2.4)` gives the result `TRUE`
`ISEVEN(3.6)` gives the result `FALSE`
`ISEVEN(ROUND(3.6))` gives the result `TRUE`

Version Available

This function is available in product version 1.0 or later.

See Also

ISODD | EVEN | Information Functions

ISLOGICAL

Summary

This function tests whether a value, an expression, or contents of a referenced cell is a logical (Boolean) value.

Syntax

```
ISLOGICAL(cellreference)
ISLOGICAL(value)
ISLOGICAL(expression)
```

Arguments

Specify a cell reference, a numeric or text value, or an expression for the argument.

Remarks

This function returns FALSE if the value refers to an empty cell or to no data.

Use this function to test the contents of a cell, a numeric or text value directly, or a function or expression.

Data Types

Accepts any data type for an argument. Returns Boolean (TRUE or FALSE) data.

Examples

`ISLOGICAL(B7)`
`ISLOGICAL(R4C8)`
`ISLOGICAL(true)` gives a result TRUE
`ISLOGICAL(OR(B7,B8))` gives a result TRUE

Version Available

This function is available in product version 1.0 or later.

See Also

ISNONTEXT | ISNUMBER | ISTEXT | Information Functions

ISNA

Summary

This function, Is Not Available, tests whether a value, an expression, or contents of a referenced cell has the not available (#N/A) error value.

Syntax

`ISNA(`*`cellreference`*`)`
`ISNA(`*`value`*`)`
`ISNA(`*`expression`*`)`

Arguments

Specify a cell reference, a numeric or text value, or an expression for the argument.

Remarks

Use this function to test the contents of a cell, a numeric or text value directly, or a function or expression.

This function returns TRUE if the value is or refers to the Not Available error value, and returns FALSE if the value is or refers to a cell with no data.

Data Types

Accepts any data type for an argument. Returns Boolean (TRUE or FALSE) data.

Examples

`ISNA(B12)`
`ISNA(R12C2)`
`ISNA(#N/A)` gives the result TRUE
`ISNA(NA())` gives the result TRUE
`ISNA(#REF)` gives the result FALSE

Version Available

This function is available in product version 1.0 or later.

See Also

ERRORTYPE | ISERR | ISERROR | NA | Information Functions

ISNONTEXT

Summary

This function tests whether a value, an expression, or contents of a referenced cell has any data type other than text.

Syntax

ISNONTEXT(*cellreference*)
ISNONTEXT(*value*)
ISNONTEXT(*expression*)

Arguments

Specify a cell reference, a numeric or text value, or an expression for the argument.

Remarks

Use this function to test the contents of a cell, a numeric or text value directly, or a function or expression.

This function returns TRUE if the value refers to a blank cell.

Data Types

Accepts any data type for an argument. Returns Boolean (TRUE or FALSE) data.

Examples

ISNONTEXT(A3)
ISNONTEXT(R3C1)
ISNONTEXT(12) gives the result TRUE
ISNONTEXT("Total") gives the result FALSE

Version Available

This function is available in product version 1.0 or later.

See Also

ISLOGICAL | ISNUMBER | ISTEXT | Information Functions

ISNUMBER

Summary

This function tests whether a value, an expression, or contents of a referenced cell has numeric data.

Syntax

```
ISNUMBER(cellreference)
ISNUMBER(value)
ISNUMBER(expression)
```

Arguments

Specify a cell reference, a numeric or text value, or an expression for the argument.

Remarks

Use this function to test the contents of a cell, a numeric or text value directly, or a function or expression.

This function returns TRUE if the argument is or refers to a number, and returns FALSE if the argument is or refers to a value that is not a number. This function returns FALSE if the value is or refers to a cell with no data.

You might want to use this function to test whether cells contain numeric data before you perform mathematical operations on them, such as averaging the contents of a range of cells.

Data Types

Accepts any data type for an argument. Returns Boolean (TRUE or FALSE) data.

Examples

`ISNUMBER(B3)`
`ISNUMBER(R1C2)`
`ISNUMBER(12)` gives the result `TRUE`

Version Available

This function is available in product version 1.0 or later.

See Also

ISLOGICAL | ISNONTEXT | ISREF | ISTEXT | N | Information Functions

ISODD

Summary

This function, Is Number Odd, tests whether a value, an expression, or contents of a referenced cell has numeric data.

Syntax

`ISODD(`*`cellreference`*`)`
`ISODD(`*`value`*`)`
`ISODD(`*`expression`*`)`

Arguments

Specify a cell reference, a numeric or text value, or an expression for the argument.

Remarks

Use this function to test the contents of a cell, a numeric or text value directly, or a function or expression.

If the number specified by the argument is odd, the function returns TRUE. If the number specified by the argument is even, the function returns FALSE. If the number specified by the argument is zero, the function returns FALSE. If the number specified by the argument refers to an empty cell or to no data, the function returns TRUE.

Data Types

Accepts any data type for an argument. Returns Boolean (TRUE or FALSE) data.

Examples

`ISODD(B3)`
`ISODD(R1C2)`
`ISODD(12)` gives the result `FALSE`
`ISODD(2.5)` gives the result `FALSE`
`ISODD(3.6)` gives the result `TRUE`
`ISODD(ROUND(3.6))` gives the result `FALSE`

Version Available

This function is available in product version 1.0 or later.

See Also

ISEVEN | ODD | Information Functions

ISPMT

Summary

This function calculates the interest paid during a specific period of an investment.

Syntax

`ISPMT(rate,per,nper,pv)`

Remarks

This function has these arguments:

Argument	Description
rate	Interest rate for the investment.
per	Number of the period for which to find the interest, between 1 and *nper*.
nper	Total number of payment periods for the investment.
pv	Present value of the investment.

Remarks

Be consistent with the units for *rate* and *nper*.

The cash you pay out is represented by negative numbers and the cash you receive by positive numbers.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

```
ISPMT(B1,C4,C5,1)
ISPMT(R1C2,R4C3,R6C3,R7C3)
```

Version Available

This function is available in product version 2.0 or later.

See Also

IPMT | PMT | PV | Financial Functions

ISREF

Summary

This function, Is Reference, tests whether a value, an expression, or contents of a referenced cell is a reference to another cell.

Syntax

`ISREF(`*`cellreference`*`)`
`ISREF(`*`value`*`)`
`ISREF(`*`expression`*`)`

Arguments

Specify a cell reference, a numeric or text value, or an expression for the argument.

Remarks

Use this function to test the contents of a cell, a numeric or text value directly, or a function or expression.

If the argument is a reference, this function returns TRUE. If the argument is not a reference, this function returns FALSE.

Data Types

Accepts any data type for an argument. Returns Boolean (TRUE or FALSE) data.

Examples

`ISREF(B3)` gives the result `TRUE`
`ISREF(R1C2)` gives the result `TRUE`
`ISREF(12)` gives the result `FALSE`

Version Available

This function is available in product version 1.0 or later.

See Also

ISBLANK | Information Functions

ISTEXT

Summary

This function tests whether a value, an expression, or contents of a referenced cell has text data.

Syntax

`ISTEXT(`*`cellreference`*`)`
`ISTEXT(`*`value`*`)`
`ISTEXT(`*`expression`*`)`

Arguments

Specify a cell reference, a numeric or text value, or an expression for the argument.

Remarks

Use this function to test the contents of a cell, a numeric or text value directly, or a function or expression.

If the data type of the argument is text, this function returns TRUE. If the data type of the argument is not text, this function returns FALSE. If the argument refers to an empty cell, this function returns FALSE.

Data Types

Accepts any data type for an argument. Returns Boolean (TRUE or FALSE) data.

Examples

`ISTEXT(B3)`
`ISTEXT(R1C2)`
`ISTEXT("Total")` gives the result `TRUE`
`ISTEXT(12)` gives the result `FALSE`

Version Available

This function is available in product version 1.0 or later.

See Also

ISLOGICAL | ISNONTEXT | ISNUMBER | T | Information Functions

KURT

Summary

This function returns the kurtosis of a data set.

Syntax

```
KURT(value1,value2,value3,value4,...)
KURT(array)
KURT(array1,array2,...)
```

Arguments

For the arguments, you can use numbers, arrays, or references. If an array or reference argument contains text, logical values, or empty cells, the function ignores those values; however, the function includes cells with the value zero in its calculations.

You must provide four or more value arguments. You may provide up to 255 arguments.

Remarks

Kurtosis describes how peaked or flat a distribution is compared with the normal distribution. Positive kurtosis indicates a relatively peaked distribution. Negative kurtosis indicates a relatively flat distribution.

If the standard deviation of the values is zero, this function returns the #DIV/0! error value.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`KURT(F1:F8)`
`KURT(R1C6:R8C6)`
`KURT(F1:F8,G1:G8)`
`KURT(35,31,47,51,37,31,58,39)` gives the result `-0.7496238078`

Version Available

This function is available in product version 1.0 or later.

See Also

GAMMADIST | Statistical Functions

LARGE

Summary

This function returns the *n*th largest value in a data set, where *n* is specified.

Syntax

```
LARGE(array,n)
```

Arguments

This function has these arguments:

Argument	Description
array	Array from which to return the nth largest value
n	The position (from the largest value) for which to return the value (for example, 5 to return the fifth largest value). Must be equal to or less than the number of items in the array.

Remarks

Use this function to select a value based on its relative standing. For example, you can use it to return the third-place score in a competition.

Data Types

Accepts array and numeric data for all arguments. Returns numeric data.

Examples

`LARGE(F1:F8,2)`
`LARGE(R1C6:R8C6,5)`
`LARGE({35,31,47,51,37,31,58,39},3)` gives the result `47.0000000000`

Version Available

This function is available in product version 1.0 or later.

See Also

SMALL | Statistical Functions

LCM

Summary

This function returns the least common multiple of two numbers.

Syntax

`LCM(number1,number2)`

Arguments

For the arguments, use numeric values. If the arguments are not integers, they are truncated to integers. This function can have up to 255 arguments.

Remarks

The least common multiple is the smallest positive integer that is a multiple of all integers given.

Use this function to add fractions with different denominators by calculating the least common multiple of both denominators first.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`LCM(B12,C22)`
`LCM(R12C2,R22C3)`
`LCM(300,500)` gives the result `1500`
`LCM(12.3,16.99)` gives the result `48`

Version Available

This function is available in product version 1.0 or later.

See Also

GCD | Math and Trigonometry Functions

LEFT

Summary

This function returns the specified leftmost characters from a text value.

Syntax

LEFT(*mytext*,*num_chars*)

Arguments

This function has these arguments:

Argument	Description
mytext	Text string that contains the characters you want to extract.
num_chars	[Optional] Number of characters to extract; if omitted, uses one; if not an integer, the number is truncated

The *mytext* argument can be a string, a formula that returns a string, or a reference to a cell containing a string.

The *num_chars* argument has these rules:

- It must be greater than or equal to zero.
- If it is greater than the length of text, this function returns all the text.

Data Types

Accepts string data for the first argument and numeric data the second argument. Returns string data.

Examples

`LEFT(A2,LEN(A2)-1)`
`LEFT(R2C1,LEN(R2C1)-1)`
`LEFT(""TotalPrice"")` gives the result `T`
`LEFT(""Total Price"", 5)` gives the result `Total`

Version Available

This function is available in product version 1.0 or later.

See Also

MID | RIGHT | Text Functions

LEN

Summary

This function returns the length of, the number of characters in, a text string.

Syntax

`LEN(value)`

Arguments

The argument is the text whose length you want to find. Spaces count as characters.

The argument must be a string or a cell reference to a string value.

Data Types

Accepts string data. Returns numeric data.

Examples

`LEFT(A2,LEN(A2)-1)`
`LEN(""FarPoint Technologies, NC"")` gives the result `25`

Version Available

This function is available in product version 1.0 or later.

See Also

CHAR | TRIM | Text Functions

LINEST

Summary

This function calculates the statistics for a line.

Syntax

`LINEST(y,x,constant,stats)`

Arguments

The equation for the line is y=mx+b or y=m1x1+m2x2+...+b.

This function has these arguments:

Argument	Description
y	Set of y values that are known in the relationship y=mx+b
x	(Optional) X is an optional set of x values that may be known in the relationship y=mx+b
constant	Logical value that specifies whether to force the constant b to equal 0. If true or omitted then b is calculated normally; if false then b is equal to 0 and the m values are adjusted so that y=mx.
stats	Logical value that specifies whether to return additional regression statistics. If true, then the additional regression statistics are returned if false or omitted then only the m-coefficients and b are returned.

If x is omitted then x defaults to the array {1,2,3...}, that has the same dimensions as y.

Remarks

Use the INDEX function to get individual elements from the returned array.

Data Types

Accepts an array. Returns an array.

Examples

`LINEST(A2:A7,C2:C7,,FALSE)`

Version Available

This function is available in product version 2.0 or later.

See Also

GROWTH | TREND | LOGEST | DEVSQ | MEDIAN | VAR | Statistical Functions

LN

Summary

This function returns the natural logarithm of the specified number.

Syntax

`LN(value)`

Arguments

For the argument, specify a positive numeric value.

Remarks

This function is the inverse of EXP, so LN(EXP(x)) is x.

Data Types

Accepts numeric data. Returns numeric data.

Examples

```
LN(B3)
LN(R1C2)
LN(10) gives the result 2.3025850930
```
`LN(exp(1))` gives the result `1`

Version Available

This function is available in product version 1.0 or later.

See Also

EXP | LOG | LOGINV | Math and Trigonometry Functions

LOG

Summary

This function returns the logarithm base Y of a number X.

Syntax

`LOG(`*`number`*`,`*`base`*`)`

Arguments

This function has these arguments:

Argument	Description
number	Number for which to find a logarithm. This must be a positive real number
base	[Optional] Base of the logarithm; if omitted, the calculation uses 10 as the base (See LOG10.)

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`LOG(B3,C5)`
`LOG(R1C2,R4C4)`
`LOG(255,16)` gives the result `1.9985883592`

Version Available

This function is available in product version 1.0 or later.

See Also

LN | LOG10 | Math and Trigonometry Functions

LOG10

Summary

This function returns the logarithm base 10 of the number given.

Syntax

`LOG10(`*`value`*`)`

Arguments

The number specified by the argument must be a positive real number.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`LOG10(B3)`
`LOG10(R1C2)`
`LOG10(115)` gives the result `2.0606978404`

Version Available

This function is available in product version 1.0 or later.

See Also

LN | LOG | Math and Trigonometry Functions

LOGEST

Summary

This function calculates an exponential curve that fits the data and returns an array of values that describes the curve.

Syntax

`LOGEST(y,x,constant,stats)`

Arguments

The equation for the curve is y=b*m^x or y=(b*(m1^x1)*(m2^x2)*_).

This function has these arguments:

Argument	Description
y	Set of y values that are known in the relationship y=b*m^x
x	(Optional) X is an optional set of x values that may be known in the relationship y=mx+b
constant	Logical value that specifies whether to force the constant b to equal 0. If true or omitted then b is calculated normally; if false then b is equal to 0 and the m values are adjusted so that y=m^x.
stats	Logical value that specifies whether to return additional regression statistics. If true, then the additional regression statistics are returned if false or omitted then only the m-coefficients and b are returned.

If x is omitted then x defaults to the array {1,2,3...}, that has the same dimensions as y.

Remarks

Use the INDEX function to get individual elements from the returned array.

Data Types

Accepts an array. Returns an array.

Examples

`LOGEST(A2:A7,C2:C7,TRUE,FALSE)`

Version Available

This function is available in product version 2.0 or later.

See Also

GROWTH | TREND | LINEST | DEVSQ | MEDIAN | VAR | Statistical Functions

LOGINV

Summary

This function returns the inverse of the lognormal cumulative distribution function of *x*, where LN(*x*) is normally distributed with the specified mean and standard deviation.

Syntax

`LOGINV(`*`prob`*`,`*`mean`*`,`*`stdev`*`)`

Arguments

This function has these arguments:

Argument	Description
prob	Value at which to evaluate the function
mean	Value of mean of natural logarithm of x, LN(x)
stdev	Value representing the standard deviation of LN(x)

Remarks

This function calculates the inverse of the lognormal cumulative distribution functions, so if *p* = LOGNORMDIST(*x*,...) then LOGINV(*p*,...) = *x*.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`LOGINV(0.92,B8,G22)`
`LOGINV(0.88,2,1.2)` gives the result `30.26479297`

Version Available

This function is available in product version 1.0 or later.

See Also

LN | LOGNORMDIST | Statistical Functions

LOGNORMDIST

Summary

This function returns the cumulative natural log normal distribution of x, where LN(x) is normally distributed with the specified mean and standard deviation. Analyze data that has been logarithmically transformed with this function.

Syntax

`LOGNORMDIST(x,mean,stdev)`

Arguments

This function has these arguments:

Argument	Description
x	Value at which to evaluate the function
mean	Value of mean of natural logarithm of x, LN(x)
stdev	Value representing the standard deviation of LN(x)

Remarks

If *p* = LOGNORMDIST(*x*,...) then LOGINV(*p*,...) = *x*.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`LOGNORMDIST(0.92,B8,G22)`
`LOGNORMDIST(42,2,1.2)` gives the result `0.926199546`

Version Available

This function is available in product version 1.0 or later.

See Also

LN | LOGINV | Statistical Functions

LOOKUP

Summary

This function searches for a value and returns a value from the same location in a second area.

Syntax

`LOOKUP(lookupvalue,lookupvector,resultvector)`
`LOOKUP(lookupvalue,lookuparray)`

Arguments

Vector Form

The arguments for the vector form are:

Argument	Description
lookupvalue	Value for which to search; can be number, text, logical value, or name or reference that refers to a value
lookupvector	Cell range that contains one row or one column; can be text, numbers, or a logical value; values need to be in ascending order
resultvector	Cell range that contains one row or column; must be the same size as *lookupvector*

Array Form

The arguments for the array form are:

Argument	Description
lookupvalue	Value for which to search; can be number, text, logical value, or name or reference that refers to a value
lookuparray	Range of cells that contains text, numbers, or logical values; values must be ascending order

Remarks

Vector Form

The vector form of this function searches for a value from a range with a single row or column and returns a value from the same location in a second one row or one column range.

In the vector form, if *lookupvalue* can not be found, it matches the largest value in *lookupvector* that is less than or equal to *lookupvalue*.

Array Form

The array form of this function searches in the first row or column of an array for the specified value and returns a value from the same location in the last row or column of the array.

In the array form, if *lookuparray* has more columns than rows then the first row is searched. If *lookuparray* has more rows than columns then the first column is searched. The values in *lookuparray* must be in ascending order.

Data Types

Accepts numeric or string data. Returns numeric or string data.

Examples

```
LOOKUP(30,A1:A5,B1:B5)
LOOKUP("A",{"a","b","c","d";1,2,3,5})
```

Version Available

This function is available in product version 2.0 or later.

See Also

HLOOKUP | VLOOKUP | Lookup Functions

LOWER

Summary

This function converts text to lower case letters.

Syntax

`LOWER(`*`string`*`)`

Arguments

The argument is the text you want to convert to lower case. This function does not change characters in value that are not letters.

The argument may be a string, a reference to a cell containing a string, or a formula that returns a string.

Data Types

Accepts string data. Returns string data.

Examples

`LOWER(A4)`
`LOWER(R4C1)`
`LOWER(""Road Race 2"")` gives the result `road race 2`
`LOWER(CONCATENATE(A1,A5))`

Version Available

This function is available in product version 1.0 or later.

See Also

UPPER | T | Text Functions

MAX

Summary

This function returns the maximum value, the greatest value, of all the values in the arguments.

Syntax

```
MAX(value1,value2,...)
MAX(array)
MAX(array1,array2,...)
```

Arguments

Each argument can be a double-precision floating point value, an integer value, or an array of these. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Remarks

If an array or reference argument contains text, logical values, or empty cells, the function ignores those values; however, the function includes in calculations cells with the value zero.

This function differs from MAXA, which allows text and logical values as well as numeric values.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`MAX(A1,B2,C3,D4,E5)`
`MAX(A1:A9)`
`MAX(R1C2:R1C15,R2C2:R2C15)`
`MAX(2,15,12,3,7,19,4)` gives the result `19`

Version Available

This function is available in product version 1.0 or later.

See Also

MIN | MAXA | Statistical Functions

MAXA

Summary

This function returns the largest value in a list of arguments, including text and logical values.

Syntax

`MAXA(`*`value1`*`,`*`value2`*`,...)`
`MAXA(`*`array`*`)`
`MAXA(`*`array1`*`,`*`array2`*`,...)`

Arguments

Each argument can be a double-precision floating point value, an integer value, text, or logical values. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Remarks

This function differs from MAX because it allows text and logical values as well as numeric values.

Data Types

Accepts numeric, text, or logical data for all arguments. Returns numeric data.

Examples

`MAXA(A1,B2,C3,D4,E5)`
`MAXA(A1:A9)`
`MAXA(R1C2:R1C15,R2C2:R2C15)`
`MAXA(2,15,12,3,7,19,4)` gives the result `19`

Version Available

This function is available in product version 2.0 or later.

See Also

MINA | MAX | Statistical Functions

MDETERM

Summary

This function returns the matrix determinant of an array.

Syntax

`MDETERM(`*`array`*`)`

Arguments

The array is a numeric array that has an equal number of columns and rows.

Arrays can be a cell range. If any of the array cells are empty or contain text then an error is returned.

Data Types

Accepts an array. Returns numeric data.

Examples

`MDETERM(A3:E7)`

Version Available

This function is available in product version 2.0 or later.

See Also

MINVERSE | MMULT | Math and Trigonometry Functions

MDURATION

Summary

This function calculates the modified Macauley duration of a security with an assumed par value of $100.

Syntax

`MDURATION(`*`settlement`*`,`*`maturity`*`,`*`coupon`*`,`*`yield`*`,`*`frequency`*`,`*`basis`*`)`

Arguments

This function has these arguments:

Argument	Description
settlement	Settlement date for the security
maturity	Maturity date for the security
coupon	Annual coupon rate
yield	Annual yield for the security
frequency	Frequency of payment, number of coupon payments per year; must be 1, 2, or 4
basis	[Optional] Integer representing the basis for day count (Refer to *Day Count Basis* on page 13.)

Remarks

This functions returns an error when *settlement* or *maturity* is invalid or *frequency* is a number other than 1, 2, or 4.

Data Types

Accepts numeric and DateTime object data. Returns numeric data.

Examples

`MDURATION(A1,B2,C3,D4,E5,F6)`

Version Available

This function is available in product version 2.0 or later.

See Also

DURATION | Financial Functions

MEDIAN

Summary

This function returns the median, the number in the middle of the provided set of numbers; that is, half the numbers have values that are greater than the median, and half have values that are less than the median.

Syntax

```
MEDIAN(value1,value2,...)
MEDIAN(array)
MEDIAN(array1,array2,...)
```

Arguments

Each argument can be a double-precision floating-point value, an integer value, or an array (cell range) of these. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

If there are an even number of arguments, the function calculates the average of the two numbers in the middle.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`MEDIAN(A3,B5,C1,D4,E7)`
`MEDIAN(A1:A9)`
`MEDIAN(R1C2,R3C5,R4C7,R6C7)`
`MEDIAN(89,95,76,88,92)` gives the result `89`

Version Available

This function is available in product version 1.0 or later.

See Also

AVERAGE | MODE | Statistical Functions

MID

Summary

This function returns the requested number of characters from a text string starting at the position you specify.

Syntax

`MID(`*`text`*`,`*`start_num`*`,`*`num_chars`*`)`

Arguments

This function has these arguments:

Argument	Description
text	Text string containing the characters you want to extract
start_num	Number representing the first character you want to extract in text, with the first character in the text having a value of one (1); if not an integer, the number is truncated
num_chars	Number of characters to return from text; if not an integer, the number is truncated

The *text* argument can be a string, a formula that returns a string, or a reference to a cell containing a string.

The *start_num* argument has these rules

- If *start_num* is greater than the length of *text*, this function returns "" (empty text).
- If *start_num* is less than the length of *text*, but *start_num* plus *num_chars* exceeds the length of *text*, this function returns the characters up to the end of text.

Data Types

Accepts string data for the *text* argument, numeric data for the *start_num* argument, and numeric data for the *num_chars* argument. Returns string data.

Examples

`MID(B17,5,8)`
`MID(""wind surfing"", 6, 20)` gives the result `surfing`

Version Available

This function is available in product version 1.0 or later.

See Also

LEFT | RIGHT | Text Functions

MIN

Summary

This function returns the minimum value, the least value, of all the values in the arguments.

Syntax

```
MIN(value1,value2,...)
MIN(array)
MIN(array1,array2,...)
```

Arguments

Each argument can be a double-precision floating point value, an integer value, or an array of these. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Remarks

If an array or reference argument contains text, logical values, or empty cells, the function ignores those values; however, the function includes in calculations cells with the value zero.

This function differs from MINA, which includes text and logical values as well as numeric values.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`MIN(A3,B5,C1,D4,E7)`
`MIN(A1:A9)`
`MIN(R1C2,R3C5,R4C7,R6C7)`
`MIN(2,15,12,3,7,19,4)` gives the result `2`

Version Available

This function is available in product version 1.0 or later.

See Also

MAX | MINA | Statistical Functions

MINA

Summary

This function returns the minimum value in a list of arguments, including text and logical values.

Syntax

`MINA(`*`value1`*`,`*`value2`*`,...)`
`MINA(`*`array`*`)`
`MINA(`*`array1`*`,`*`array2`*`,...)`

Arguments

Each argument can be a double-precision floating point value, an integer value, text, logical value, or an array of these. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Remarks

This function differs from MIN because it includes text and logical values as well as numeric values.

Data Types

Accepts numeric, text, or logical data for all arguments. Returns numeric data.

Examples

`MINA(A3,B5,C1,D4,E7)`
`MINA(A1:A9)`
`MINA(R1C2,R3C5,R4C7,R6C7)`
`MINA(A1,B1)` gives the result `0 if A1 is 10 and B1 is FALSE`

Version Available

This function is available in product version 2.0 or later.

See Also

MIN | MAXA | Statistical Functions

MINUTE

Summary

This function returns the minute corresponding to a specified time.

Syntax

`MINUTE(`*`time`*`)`

Arguments

Specify the time argument as a number (as in 37806.5) a string (as in "7/4/2003 12:00"), a DateTime object, as in DATE(2003,7,4), or a TimeSpan object, as in TIME(12,0,0). For more details on the date and time inputs, refer to the discussion in *Date and Time Functions* on page 10

Dates as numeric values are in the form x.y, where x is the "number of days since December 30, 1899" and y is the fraction of day. Numbers to the left represent the date. Times as numeric values are decimal fractions ranging from 0 to 0.99999999, representing the times from 0:00:00 (12:00:00 A.M.) to 23:59:59 (11:59:59 P.M.).

Remarks

The minute is returned as an integer, ranging from 0 to 59.

Data Types

Accepts numeric, string, DateTime object, or TimeSpan object data. Returns numeric data.

Examples

`MINUTE(D1)`
`MINUTE(R1C4)`
`MINUTE(0.7)` gives the result `48`
`MINUTE("12:17")` gives the result `17`
`MINUTE(TIME(12,0,0))`

Version Available

This function is available in product version 1.0 or later.

See Also

HOUR | SECOND | Date and Time Functions

MINVERSE

Summary

This function returns the inverse matrix for the matrix stored in an array.

Syntax

`MINVERSE(`*`array`*`)`

Arguments

The array is a numeric array that has an equal number of columns and rows.

Arrays can be a cell range. If any of the array cells are empty or contain text then an error is returned.

Remarks

Use the INDEX function to get individual elements from the returned array.

Data Types

Accepts an array. Returns an array.

Examples

`MINVERSE(A3:E7)`

Version Available

This function is available in product version 2.0 or later.

See Also

MDETERM | MMULT | Math and Trigonometry Functions

MIRR

Summary

This function returns the modified internal rate of return for a series of periodic cash flows.

Syntax

MIRR(*arrayvals*,*payment_int*,*income_int*)

Arguments

This function has these arguments:

Argument	Description
arrayvals	An array of numbers for which you want to estimate the internal rate of return representing payments and income occurring at regular intervals (and use positive for income and negative for payment)
payment_int	Interest rate on money in cash flows
income_int	Interest rate on money invested from cash flows

Values must contain at least one positive value (some income) and one negative value (a payment) to calculate the internal rate of return. The payments and income must occur at regular time intervals, such as monthly or annually.

Remarks

If an array or reference argument contains text, logical values, or empty cells, the function ignores those values; however, the function includes in calculations cells with the value zero.

This function uses the order of values to interpret the order of payments and income. Be sure to enter your payment and income values in the sequence you want with correct signs. The payments and income must occur at regular time intervals, such as monthly or annually.

Data Types

Accepts numeric data for all arguments, the first being an array. Returns numeric data.

Examples

MIRR(D1:D6, D10, D12)
MIRR(R1C4:R6C4, R10C4, R12C4)
MIRR({7300,-15000,4036,3050},6.5%,8%) gives the result 0.0564050548577524

Version Available

This function is available in product version 1.0 or later.

See Also

IRR | XIRR | Financial Functions

MMULT

Summary

This function returns the matrix product for two arrays.

Syntax

`MMULT(`*`array1`*`,`*`array2`*`)`

Arguments

The arrays are numeric arrays where the columns in array1 match the rows in array2.

Arrays can be a cell range. If any of the array cells are empty or contain text then an error is returned.

Remarks

Use the INDEX function to get individual elements from the returned array.

Data Types

Accepts an array for all arguments. Returns an array.

Examples

`MMULT(A2:B3,D5:E6)`

Version Available

This function is available in product version 2.0 or later.

See Also

MDETERM | MINVERSE | Math and Trigonometry Functions

MOD

Summary

This function returns the remainder of a division operation.

Syntax

`MOD(`*`dividend`*`,`*`divisor`*`)`

Arguments

This function has these arguments:

Argument	Description
dividend	Number for which you want to find the remainder by dividing the divisor into it
divisor	Number by which you want to divide the dividend argument

Remarks

The remainder has the same sign as the divisor.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`MOD(B3,10)`
`MOD(C4,B2)`
`MOD(R1C2,12)`
`MOD(255,16)` gives the result `15`
`MOD(-3,2)` gives the result `1`

Version Available

This function is available in product version 1.0 or later.

See Also

PRODUCT | QUOTIENT | Math and Trigonometry Functions

MODE

Summary

This function returns the most frequently occurring value in a set of data.

Syntax

MODE(*value1*,*value2*,...)
MODE(*array*)
MODE(*array1*,*array2*,...)

Arguments

Each argument can be a double-precision floating-point value, an integer value, or an array (cell range) of these. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Remarks

If no value occurs more than once, the function does not return a value. If more than one value occurs the same number of times, the function returns the first value that repeats that same number of times.

If an array or reference argument contains text, logical values, or empty cells, the function ignores those values; however, the function includes in calculations cells with the value zero.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

MODE(A3,B3,C3,D3)
MODE(A1:A9)
MODE(R1C2,12,10,R2C3)
MODE(A2:A9,B2:B9,B12:35)
MODE(89,95,88,97,88,74) gives the result 88
MODE(1,2,2,3,4,5,5) gives the result 2

Version Available

This function is available in product version 1.0 or later.

See Also

AVERAGE | MEDIAN | Statistical Functions

MONTH

Summary

This function returns the month corresponding to the specified date value.

Syntax

`MONTH(`*`date`*`)`

Arguments

Specify the date argument as a number (as in 37806.5) a string (as in "7/4/2003 12:00"), or a DateTime object, as in DATE(2003,7,4). For more details on the date inputs, refer to the discussion in *Date and Time Functions* on page 10.

Remarks

The month is returned as an integer, ranging from 1 (January) to 12 (December).

Data Types

Accepts numeric, string, or DateTime object data. Returns numeric data.

Examples

`MONTH(L4)`
`MONTH(R4C12)`
`MONTH(366)` gives the result `12`
`MONTH("12/17/2004")` gives the result `12`

Version Available

This function is available in product version 1.0 or later.

See Also

DAY | EOMONTH | YEAR | Date and Time Functions

MROUND

Summary

This function returns a number rounded to the desired multiple.

Syntax

`MROUND(number,multiple)`

Arguments

This function has these arguments:

Argument	Description
number	Numeric value to round
multiple	Numeric value representing the rounded result

Remarks

This function rounds to the nearest multiple (either up or down). For even numbers where there may be two choices (one rounding up and one rounding down), the result is the number farther from zero. For example, MROUND(18,4) returns 20 even though 16 is as near since 20 is farther from zero. For MROUND(-18,-4) returns -20 since that value is farther from zero.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`MROUND(B14,3)`
`MROUND(R14C2,5)`
`MROUND(100,8)` gives the result `104`
`MROUND (11,8)` gives the result `8`
`MROUND (12,8)` gives the result `16`
`MROUND (13,8)` gives the result `16`
`MROUND (-12,-8)` gives the result `-16`
`MROUND (50,8)` gives the result `48`
`MROUND (-50,-8)` gives the result `-48`

Version Available

This function is available in product version 1.0 or later.

See Also

ROUND | Math and Trigonometry Functions

MULTINOMIAL

Summary

This function calculates the ratio of the factorial of a sum of values to the product of factorials.

Syntax

MULTINOMIAL(*value1*,*value2*,...)
MULTINOMIAL(*array*)
MULTINOMIAL(*array1*,*array2*,...)

Arguments

The arguments are the values to calculate in the multinomial. Each argument can be a double-precision floating point value, an integer value, or an array of these. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Remarks

If an array or reference argument contains text, logical values, or empty cells, the function ignores those values; however, the function includes in calculations cells with the value zero.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

MULTINOMIAL(D5,D6,D7,D8)
MULTINOMIAL(R5C4,R6C4,R7C4,R8C4)
MULTINOMIAL(1,2,3) gives the result 60

Version Available

This function is available in product version 1.0 or later.

See Also

MODE | Math and Trigonometry Functions

N

Summary

This function returns a value converted to a number.

Syntax

`N(value)`

Arguments

Use any value as the argument.

Remarks

It is not always necessary to use this function, because Spread automatically converts values as necessary in many cases.

Data Types

Accepts many types of data. Returns numeric data.

Examples

`N(G12)`
`N(R12C7)`
`N(2.53)` gives the result `2.53`
`N(TRUE)` gives the result `1`

Version Available

This function is available in product version 1.0 or later.

See Also

ISNUMBER | Information Functions

NA

Summary

This function returns the error value #N/A that means "not available."

Syntax

`NA( )`

Arguments

This function does not require an argument.

Remarks

It is necessary to include empty parentheses with this function.

Data Types

Returns an error value.

Examples

`NA()`
`NA(R12C7)`
`ISNA(NA())` gives the result TRUE

Version Available

This function is available in product version 1.0 or later.

See Also

ISNA | ISNUMBER | Information Functions

NEGBINOMDIST

Summary

This function returns the negative binomial distribution.

Syntax

`NEGBINOMDIST(`*x*`,`*r*`,`*p*`)`

Arguments

This function has these arguments:

Argument	Description
x	An integer representing the number of failures in trials
r	An integer representing the threshold number of successes
p	Probability of success on each trial A number between 0 and 1.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`NEGBINOMDIST(B1,C15,0.335)`
`NEGBINOMDIST(R1C2,R15C3,0.75)`
`NEGBINOMDIST(4,13,0.85)` gives the result `0.111399299`

Version Available

This function is available in product version 1.0 or later.

See Also

BINOMDIST | HYPGEOMDIST | Statistical Functions

NETWORKDAYS

Summary

This function returns the total number of complete working days between the start and end dates.

Syntax

`NETWORKDAYS(`*`startdate`*`,`*`enddate`*`,`*`holidays`*`)`

Arguments

This function has these arguments:

Argument	Description
startdate	Date that is the starting date; a number (as in 37806.5), or a DateTime object, as in DATE(2003,7,4)
enddate	Date that is the ending date; a number (as in 37806.5), or a DateTime object, as in DATE(2003,7,4)
holidays	[Optional] Range of dates to exclude from the calculation; if omitted, the calculation assumes no holidays and all weekdays are workdays

Data Types

Accepts numeric, string, or DateTime object data. Returns numeric data.

Examples

```
NETWORKDAYS(L4,L5)
NETWORKDAYS(R4C12,R1C1,R2C2)
```

Version Available

This function is available in product version 2.0 or later.

See Also

WORKDAY | NOW | Date and Time Functions

NOMINAL

Summary

This function returns the nominal annual interest rate for a given effective rate and number of compounding periods per year.

Syntax

`NOMINAL(effrate,comper)`

Arguments

This function has these arguments:

Argument	Description
effrate	Value representing the effective interest rate
comper	Number of compounding periods per year; if not an integer, the number is truncated

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`NOMINAL(A4,A5)`
`NOMINAL(R4C1,3)`
`NOMINAL(6.2336%,2)` gives the result `0.061393703`
`NOMINAL(6.2336%,6)` gives the result `0.060776004`

Version Available

This function is available in product version 1.0 or later.

See Also

EFFECT | INTRATE | Financial Functions

NORMDIST

Summary

This function returns the normal cumulative distribution for the specified mean and standard deviation.

Syntax

`NORMDIST(x,mean,stdev,cumulative)`

Arguments

This function has these arguments:

Argument	Description
x	Value for which to find the distribution
mean	Arithmetic mean of the distribution
stdev	Standard deviation of the distribution Must be greater than zero.
cumulative	Set to TRUE to return the cumulative distribution function. Set to FALSE to return the probability mass function.

Remarks

If *mean* = 0 and *stdev* = 1, this function returns the standard normal distribution, NORMSDIST.

Data Types

The *x*, *mean*, and *stdev* arguments accept numeric data. The *cumulative* argument accepts logical data. Returns numeric data.

Examples

`NORMDIST(10,A3,B17,FALSE)`
`NORMDIST(10,R3C1,R17C2,FALSE)`
`NORMDIST(37,41.125,9.86,TRUE)` gives the result `0.3378810361`

Version Available

This function is available in product version 1.0 or later.

See Also

NORMINV | NORMSDIST | Statistical Functions

NORMINV

Summary

This function returns the inverse of the normal cumulative distribution for the given mean and standard deviation.

Syntax

`NORMINV(`*`prob`*`,`*`mean`*`,`*`stdev`*`)`

Arguments

This function has these arguments:

Argument	Description
prob	Probability of the normal distribution
mean	Arithmetic mean of the distribution
stde*stdev*	Standard deviation of the distribution Must be greater than zero.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`NORMINV(B3,C12,D14)`
`NORMINV(R3C2,R12C3,R14C4)`
`NORMINV(0.978,32,0.252)` gives the result `32.50755174`

Version Available

This function is available in product version 1.0 or later.

See Also

NORMDIST | NORMSINV | Statistical Functions

NORMSDIST

Summary

This function returns the standard normal cumulative distribution function.

Syntax

`NORMSDIST(value)`

Arguments

The argument can be any numeric value.

Remarks

The distribution has a mean of zero and a standard deviation of one.

Use this function in place of a table of standard normal curve areas.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`NORMSDIST(F1)`
`NORMSDIST(R1C6)`
`NORMSDIST(1.288)` gives the result `0.901127`

Version Available

This function is available in product version 1.0 or later.

See Also

NORMDIST | NORMSINV | Statistical Functions

NORMSINV

Summary

This function returns the inverse of the standard normal cumulative distribution. The distribution has a mean of zero and a standard deviation of one.

Syntax

`NORMSINV(`*`prob`*`)`

Arguments

The argument is the probability for the normal distribution.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`NORMSINV(A3)`
`NORMSINV(R1C2)`
`NORMSINV(0.9244)` gives the result `1.43530571453713`

Version Available

This function is available in product version 1.0 or later.

See Also

NORMINV | NORMSDIST | Statistical Functions

NOT

Summary

This function reverses the logical value of its argument.

Syntax

`NOT(value)`

Arguments

Provide a numeric or logical value for the argument.

Remarks

If the specified value is zero, then the function returns TRUE. If the specified value is a value other than zero, then the function returns FALSE.

Data Types

Accepts boolean data (TRUE or FALSE). Returns boolean data (TRUE or FALSE).

Examples

`NOT(A3)`
`NOT(R1C2)`
`NOT(D5>100)`
`NOT(0)` gives the result `TRUE`
`NOT(TRUE)` gives the result `FALSE`
`NOT(12)` gives the result `FALSE`

Version Available

This function is available in product version 1.0 or later.

See Also

AND | OR | Logical Functions

NOW

Summary

This function returns the current date and time.

Syntax

`NOW()`

Arguments

This function does not accept arguments.

Remarks

This function is updated only when the spreadsheet or cell containing the function is recalculated. This is a volatile function with version 2.5 or later.

Data Types

Does not accept data. Returns a DateTime object.

Examples

If it is 05:10:00 P.M., November 11, 2004, then:

`NOW()` gives the result `November 11, 2004, 5:10pm`

Version Available

This function is available in product version 1.0 or later. This function is a volatile function in version 2.5 or later.

See Also

DATEVALUE | TIME | Date and Time Functions

NPER

Summary

This function returns the number of periods for an investment based on a present value, future value, periodic payments, and a specified interest rate.

Syntax

`NPER(`*`rate`*`,`*`paymt`*`,`*`pval`*`,`*`fval`*`,`*`type`*`)`

Arguments

This function has these arguments:

Argument	Description
rate	Interest rate expressed as percentage (per period)
paymt	Payment made each period; cannot change over life of the annuity
pval	Present value
fval	[Optional] Future value; if omitted, the calculation uses zero (0)
type	[Optional] Indicates when payments are due; at the end (0) or beginning (1) of the period; if omitted, the calculation uses the end (0)

For the arguments, money paid out (such as deposits in an investment) is represented by negative numbers; money you receive (such as dividend checks) is represented by positive numbers.

Remarks

Be sure to express the interest rate as per period. For example, if you make monthly payments on a loan at 8 percent interest, use 0.08/12 for the rate argument.

See the PV function for the equations for calculating financial values.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`NPER(A1/12,50,1000,0,1)`
`NPER(R1C1/12,50,1000,0,1)`
`NPER(0.005,-790,90000,0,1)` gives the result `167.7227522114`

Version Available

This function is available in product version 1.0 or later.

See Also

FV | PMT | PV | Financial Functions

NPV

Summary

This function calculates the net present value of an investment by using a discount rate and a series of future payments and income.

Syntax

`NPV(discount,value1,value2,...)`

Arguments

This function has these arguments:

Argument	Description
discount	Rate of discount for one period
value1,...	Values for money paid out (as for a payment) are negative numbers; values for money you receive (as for income) are positive numbers

The function includes in calculations arguments that are numbers, empty cells, logical values, or text representations of numbers; the function ignores arguments that are error values or text that cannot be translated into numbers. If an argument is an array or reference, only numbers in that array or reference are counted. Empty cells, logical values, text, or error values in the array or reference are ignored. This function can have up to 255 arguments.

Remarks

The payments and income must be equally spaced in time and occur at the end of each period. The function uses the order of the values to interpret the order of cash flows. Be sure to enter your payment and income values in the correct sequence.

The investment begins one period before the date of the value1 cash flow and ends with the last cash flow in the list. The calculation is based on future cash flows. If your first cash flow occurs at the beginning of the first period, the first value must be added to the result, not included in the value arguments.

This function is similar to the PV function (present value). Use PV to work with cash flows that begin at the beginning or the end of the period; this function allows cash flows only at the end of the period. Unlike the variable cash flow values of this function, PV cash flows must be constant throughout the investment.

This is also related to the IRR function (internal rate of return). IRR is equivalent to this function when the rate argument for net present value equals zero: NPV(IRR(...),...) = 0.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`NPV(0.065,D12:D19)`
`NPV(R1C1,R12C4:R19C4)`
`NPV(6.5%, -10000, 3000, 3400, 7700)` gives the result `$2,055.38`

Version Available

This function is available in product version 1.0 or later.

See Also

IRR | PV | Financial Functions

OCT2BIN

Summary

This function converts an octal number to a binary number.

Syntax

`OCT2BIN(number,places)`

Arguments

This function has these arguments:

Argument	Description
number	Octal numeric value to convert, must be 10 characters or less, and must be between 7777777000 and 777
places	[Optional] Number of characters to return; if not an integer, the number is truncated

Remarks

An error value is returned if the *number* is invalid or if *places* is non-numeric or negative. If *places* is omitted, the calculation uses the minimum number of characters necessary. This argument is useful for adding leading zeros to the result.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`OCT2BIN(77770000)`

Version Available

This function is available in product version 2.0 or later.

See Also

OCT2DEC | OCT2HEX | HEX2BIN | DEC2BIN | Engineering Functions

OCT2DEC

Summary

This function converts an octal number to a decimal number.

Syntax

`OCT2DEC(`*`number`*`)`

Arguments

Specify the octal number to convert. The number should not contain more than 10 octal characters. An error value is returned if the number is invalid.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`OCT2DEC(7777)`

Version Available

This function is available in product version 2.0 or later.

See Also

OCT2BIN | OCT2HEX | HEX2DEC | DEC2OCT | Engineering Functions

OCT2HEX

Summary

This function converts an octal number to a hexadecimal number.

Syntax

`OCT2HEX(number,places)`

Arguments

This function has these arguments:

Argument	Description
number	Octal numeric value to convert, must be 10 characters or less
places	[Optional] Number of characters to return; if not an integer, the number is truncated

Remarks

An error value is returned if the *number* is invalid or if *places* is non-numeric or negative. If *places* is omitted, the calculation uses the minimum number of characters necessary. This argument is useful for adding leading zeros to the result.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`OCT2HEX(7777)`

Version Available

This function is available in product version 2.0 or later.

See Also

OCT2BIN | OCT2DEC | HEX2OCT | DEC2OCT | Engineering Functions

ODD

Summary

This function rounds the specified value up to the nearest odd integer.

Syntax

`ODD(`*`value`*`)`

Arguments

The argument can be any numeric value.

Remarks

Regardless of the sign of the number specified by the argument, the number is rounded away from zero.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`ODD(A3)`
`ODD(R1C2)`
`ODD(4)` gives the result `5`
`ODD(-2.5)` gives the result `-3`

Version Available

This function is available in product version 1.0 or later.

See Also

CEILING | EVEN | FLOOR | ISODD | Math and Trigonometry Functions

ODDFPRICE

Summary

This function calculates the price per $100 face value of a security with an odd first period.

Syntax

`ODDFPRICE(`*`settle`*`,`*`maturity`*`,`*`issue`*`,`*`first`*`,`*`rate`*`,`*`yield`*`,`*`redeem`*`,`*`freq`*`,`*`basis`*`)`

Arguments

This function has these arguments:

Argument	Description
settle	Settlement date for the security
maturity	Maturity date for the security
issue	Issue date for the security
first	First coupon date
rate	Annual interest rate
yield	Annual yield for the security
redeem	Redemption value per $100 face value for the security
freq	Frequency of payment, number of payments per year
basis	[Optional] Integer representing the basis for day count (Refer to *Day Count Basis* on page 13.)

Remarks

This functions returns an error when *settle*, *maturity*, *issue*, or *first* is invalid.

Data Types

Accepts numeric data or dates. Returns numeric data.

Examples

`ODDFPRICE(A1,A2,A3,A4,A5,A6,A7,A8,A9)`

Version Available

This function is available in product version 2.0 or later.

See Also

ODDLPRICE | PRICE | ODDFYIELD | ODDLYIELD | Financial Functions

ODDFYIELD

Summary

This function calculates the yield of a security with an odd first period.

Syntax

`ODDFYIELD(settle,maturity,issue,first,rate,price,redeem,freq,basis)`

Arguments

This function has these arguments:

Argument	Description
settle	Settlement date for the security
maturity	Maturity date for the security
issue	Issue date for the security
first	First coupon date
price	Price of the security
redeem	Redemption value per $100 face value for the security
freq	Frequency of payment, number of payments per year
basis	[Optional] Integer representing the basis for day count (Refer to *Day Count Basis* on page 13.)

Remarks

This functions returns an error when *settle*, *maturity, issue*, or *first* is invalid.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`ODDFYIELD(B1,B2,B3,B4,B5,B6,B7,B8,B9)`

Version Available

This function is available in product version 2.0 or later.

See Also

PRICE | ODDLYIELD | ODDFPRICE | ODDLPRICE | Financial Functions

ODDLPRICE

Summary

This function calculates the price per $100 face value of a security with an odd last coupon period.

Syntax

`ODDLPRICE(`*`settle`*`,`*`maturity`*`,`*`last`*`,`*`rate`*`,`*`yield`*`,`*`redeem`*`,`*`freq`*`,`*`basis`*`)`

Arguments

This function has these arguments:

Argument	Description
settle	Settlement date for the security
maturity	Maturity date for the security
last	Last coupon date
rate	Annual interest rate
yield	Annual yield for the security
redeem	Redemption value per $100 face value for the security
freq	Frequency of payment, number of payments per year
basis	[Optional] Integer representing the basis for day count (Refer to *Day Count Basis* on page 13.)

Remarks

This functions returns an error when *settle*, *maturity,* or *last* is invalid.

Data Types

Accepts numeric data and dates. Returns numeric data.

Examples

`ODDLPRICE(C1,C2,A3,C4,C5,C6,C7,C8)`

Version Available

This function is available in product version 2.0 or later.

See Also

PRICE | ODDFPRICE | ODDFYIELD | ODDLYIELD | Financial Functions

ODDLYIELD

Summary

This function calculates the yield of a security with an odd last period.

Syntax

`ODDLYIELD(settle,maturity,last,rate,price,redeem,freq,basis)`

Arguments

This function has these arguments:

Argument	Description
settle	Settlement date for the security
maturity	Maturity date for the security
last	Last coupon date
rate	Annual interest rate
price	Price of the security
redeem	Redemption value per $100 face value for the security
freq	Frequency of payment, number of payments per year
basis	[Optional] Integer representing the basis for day count (Refer to *Day Count Basis* on page 13.)

Remarks

This functions returns an error when *settle*, *maturity,* or *last* is invalid.

Data Types

Accepts numeric data or dates. Returns numeric data.

Examples

`ODDLYIELD(G1,G2,G3,G4,G5,G6,G7,G8)`

Version Available

This function is available in product version 2.0 or later.

See Also

PRICE | ODDFPRICE | ODDFYIELD | ODDLPRICE | Financial Functions

OFFSET

Summary

This function returns a reference to a range. The range is a specified number of rows and columns from a cell or range of cells. The function returns a single cell or a range of cells.

Syntax

`OFFSET(`*`reference,rows,cols,height,width`*`)`

Remarks

This function has these arguments:.

Argument	Description
reference	The location from which to base the offset
rows	Number of rows to which the upper left cell refers
cols	Number of columns to which the upper left cell refers
height	[Optional] Number of returned rows; if omitted, same as *reference*
width	[Optional] Number of returned columns; if omitted, same as *reference*

The *cols* can be positive (right of the reference) or negative (left). If height or width is omitted, it is the same as the reference.

Remarks

This is a volatile function.

Data Types

Accepts a cell range for reference. Accepts numbers for rows, cols, height, and width. Returns a cell range.

Examples

```
OFFSET(D3,2,3,1,1)
OFFSET(D3:E5,2,3,1,1)
```

Version Available

This function is available in product version 2.5 or later.

See Also

HLOOKUP | LOOKUP | Lookup Functions

OR

Summary

This function calculates logical OR. It returns TRUE if any of its arguments are true; otherwise, returns FALSE if all arguments are false.

Syntax

`OR(`*`bool1`*`,`*`bool2`*`,...)`
`OR(`*`array`*`)`
`OR(`*`array1`*`,`*`array2`*`,...)`
`OR(`*`expression`*`)`
`OR(`*`expression1`*`,`*`expression2`*`,...)`

Arguments

Provide numeric (1 or 0) or logical values (TRUE or FALSE) for up to 255 arguments. You can also specify a single array instead of listing the values separately, or up to 255 arrays. Similarly, you can specify an expression or up to 255 expressions.

Data Types

Accepts logical data (Boolean values of TRUE or FALSE) or numerical values (0 or 1). Returns logical data (Boolean values of TRUE or FALSE).

Examples

`OR(B3,B6,B9)`
`OR(R1C2,R1C3,R1C4,R1C5)`
`OR(D2:D12)`
`OR(R12C1:R12C9)`
`OR(TRUE,FALSE,FALSE)` gives the result `TRUE`
`OR(TRUE())` gives the result `TRUE`
`OR(FALSE(),FALSE())` gives the result `FALSE`
`OR(1+1=1,2+2=5)` gives the result `FALSE`
`OR(5+3=8,5+4=12)` gives the result `TRUE`

Version Available

This function is available in product version 1.0 or later.

See Also

AND | NOT | Logical Functions

PEARSON

Summary

This function returns the Pearson product moment correlation coefficient, a dimensionless index between -1.0 to 1.0 inclusive indicative of the linear relationship of two data sets.

Syntax

`PEARSON(array_ind,array_dep)`

Arguments

This function has these arguments:

Argument	Description
array_ind	Array of independent values (x's)
array_dep	Array of dependent values (y's)

The arrays must be the same size.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`PEARSON(B4:G7,B8:G11)`
`PEARSON(R4C2:R7C7,R8C2:R11C7)`
`PEARSON({2,8,4,16,10,12},{8,2,15,14,18,11})` gives the result `0.262017`

Version Available

This function is available in product version 1.0 or later.

See Also

RSQ | STEYX | Statistical Functions

PERCENTILE

Summary

This function returns the *n*th percentile of values in a range.

Syntax

`PERCENTILE(array,n)`

Arguments

This function has these arguments:

Argument	Description
array	Array of values representing the data
n	Value representing the percentile value between 0 and 1

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`PERCENTILE(A1:A12,0.95)`
`PERCENTILE(R1C1:R1C45,0.866)`
`PERCENTILE({5,15,25,50,65},0.45)` gives the result `23`

Version Available

This function is available in product version 1.0 or later.

See Also

PERCENTRANK | QUARTILE | Statistical Functions

PERCENTRANK

Summary

This function returns the rank of a value in a data set as a percentage of the data set.

Syntax

`PERCENTRANK(`*`array`*`,`*`n`*`,`*`sigdig`*`)`

Arguments

This function has these arguments:

Argument	Description
array	Array of data with numeric values that defines the relative ranking
n	Value for which you want to find the rank in percentage
sigdig	[Optional] Number of significant digits for the ranked percentage value; if omitted, the calculation used three significant digits; if not an integer, number is truncated

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`PERCENTRANK(A1:A12,0.95)`
`PERCENTRANK(R1C1:R1C45,0.866)`
`PERCENTRANK(A1:A17,23,3)`
`PERCENTRANK(R1C1:R43:C1,255.4,2)`
`PERCENTRANK({10,12,13,14,14,14.5,16,17.5,17.75,20,22},18,4)` gives the result `0.8111`
`PERCENTRANK({10,12,13,14,14,14.5,16,17.5,17.75,20,22},18,1)` gives the result `0.8`

Version Available

This function is available in product version 1.0 or later.

See Also

PERCENTILE | Statistical Functions

PERMUT

Summary

This function returns the number of possible permutations for a specified number of items.

Syntax

`PERMUT(k,n)`

Arguments

This function has these arguments:

Argument	Description
k	Number of items; must be greater than 0; if not an integer, the number is truncated
n	Number of items in each possible permutation; must be positive or 0; if not an integer, the number is truncated

Remarks

A permutation is any set or subset of items where internal order is significant. Contrast with combinations (the COMBIN function).

The equation for this function is:

$$PERMUT(k,\ n) = P_{k,\ n} = \frac{n!}{(n-k)!}$$

where k and n are defined in the arguments.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`PERMUT(B3,5)`
`PERMUT(C4,B2)`
`PERMUT(R1C2,2)`
`PERMUT(8,2)` gives the result `56`
`PERMUT(100,3)` gives the result `970200`

Version Available

This function is available in product version 1.0 or later.

See Also

COMBIN | Math and Trigonometry Functions

PI

Summary

This function returns PI as 3.1415926536.

Syntax

`PI()`

Arguments

This function does not accept arguments.

Data Types

Does not accept data. Returns numeric data.

Examples

`PI( )`

`DEGREES(PI())` gives the result `180`

Version Available

This function is available in product version 1.0 or later.

See Also

DEGREES | RADIANS | Math and Trigonometry Functions

PMT

Summary

This function returns the payment amount for a loan given the present value, specified interest rate, and number of terms.

Syntax

`PMT(rate,nper,pval,fval,type)`

Arguments

This function has these arguments:

Argument	Description
rate	Value of interest rate per period
nper	Total number of payment periods
pval	Present value, worth now
fval	[Optional] Future value, cash value after the last payment; if omitted, the calculation uses zero
type	[Optional] Indicates when payments are due; at the end (0) or beginning (1) of the period; if omitted, the calculation uses the end (0)

Remarks

Be sure that the interest rate and the number of payment periods correspond to the same units. If payment periods are monthly, then the interest rate should be calculated per month. If the interest rate is 6 percent annually, you can use 6% or (6/100) or 0.06 for the rate argument if the payment period is a year, but for monthly pay periods, divide the 6% by 12.

The result is represented by a negative number because it is money paid out by you.

See the PV function for the equation for calculating financial values.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`PMT(B1,C4,C5,C6,1)`
`PMT(R1C2,8,16,4)`
`PMT(6%/12, 15, 5000)` gives the result `-$346.82`
`PMT(0.005, 15, 5000, 0, 1)` gives the result `-$345.10`

Version Available

This function is available in product version 1.0 or later.

See Also

IPMT | PPMT | PV | Financial Functions

POISSON

Summary

This function returns the Poisson distribution.

Syntax

`POISSON(nevents,mean,cumulative)`

Remarks

This function has these arguments:

Argument	Description
nevents	Number of events Provide an integer, or the value is truncated. The number must be greater than zero.
mean	Expected numeric value The number must be greater than zero.
cumulative	Set to TRUE to return the cumulative Poisson probability that the number of random events occurring is between zero and *nevents* inclusive. Set to FALSE to return the Poisson probability mass function that the number of events occurring is exactly *nevents*.

Remarks

The cumulative Poisson probability is calculated as follows:

$$POISSON(x,\ \mu,\ TRUE) = \sum_{j=0}^{x} \frac{e^{-\lambda}\lambda^{j}}{j!}$$

The Poisson probability mass function is calculated as follows:

$$POISSON(x,\ \mu,\ FALSE) = \sum_{j=0}^{x} \frac{e^{-\lambda}\lambda^{j}}{j!}$$

where x is the number of events (*nevents* argument), mu is the mean (*mean* argument).

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`POISSON(A3,B4,TRUE)`
`POISSON(R1C2,3,FALSE)`
`POISSON(7,4,TRUE)` gives the result `0.948866384`
`POISSON(7,4,FALSE)` gives the result `0.059540363`

Version Available

This function is available in product version 1.0 or later.

See Also

BINOMDIST | GAMMADIST | HYPGEOMDIST | Statistical Functions

POWER

Summary

This function raises the specified number to the specified power.

Syntax

`POWER(number,power)`

Arguments

This function has these arguments:

Argument	Description
number	Number to raise to the power given in *power*
power	Power to which to raise the number given in *number*

Specify the number to raise using the first argument and specify the power to raise it to using the second argument.

Remarks

You can use the exponent operator (^) instead of this function to raise a number to a power; for example, 16^3.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`POWER(A3,B4)`
`POWER(R1C2,3)`
`POWER(16,3)` gives the result `4096`

Version Available

This function is available in product version 1.0 or later.

See Also

EXP | SQRT | Math and Trigonometry Functions

PPMT

Summary

This function returns the amount of payment of principal for a loan given the present value, specified interest rate, and number of terms.

Syntax

`PPMT(`*`rate`*`,`*`per`*`,`*`nper`*`,`*`pval`*`,`*`fval`*`,`*`type`*`)`

Arguments

This function has these arguments:

Argument	Description
rate	Value of interest rate per period.
per	Number of the period for which to find the interest, between 1 and *nper*
nper	Total number of payment periods in an annuity.
pval	Present value, worth now
fval	[Optional] Future value, cash value after the last payment; if omitted, the calculation uses zero
type	[Optional] Indicates when payments are due; at the end (0) or beginning (1) of the period; if omitted, the calculation uses the end (0)

Remarks

Be sure to express the interest rate as per annum. For example, if the interest rate is 8 percent, use 8 for the rate argument.

The result is represented by a negative number because it is money paid out by you.

See the PV function for the equation for calculating financial values.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`PPMT(B1,C4,C5,C6,C7,1)`
`PPMT(R1C2,R4C3,R6C3,R7C3,0)`
`PPMT(0.45, 22, 30, 6000, 7000)` gives the result `-$206.47`

Version Available

This function is available in product version 1.0 or later.

See Also

IPMT | PMT | PV | Financial Functions

PRICE

Summary

This function calculates the price per $100 face value of a periodic interest security.

Syntax

`PRICE(settlement,maturity,rate,yield,redeem,frequency,basis)`

Arguments

This function has these arguments:

Argument	Description
settle	Settlement date for the security
maturity	Maturity date for the security
rate	Annual coupon rate
yield	Annual yield for the security
redeem	Redemption value per $100 face value for the security
frequency	Frequency of payment, number of payments per year; must be 1, 2, or 4
basis	[Optional] Integer representing the basis for day count (Refer to *Day Count Basis* on page 13.)

Remarks

This functions returns an error when *settle*, or *maturity* is invalid or when *frequency* is a number other than 1, 2, or 4.

Data Types

Accepts numeric data and dates. Returns numeric data.

Examples

`PRICE(A3,A4,A5,A6,A7,A8,A9)`

Version Available

This function is available in product version 2.0 or later.

See Also

PRICEMAT | PRICEDISC | ODDFPRICE | ODDLPRICE | Financial Functions

PRICEDISC

Summary

This function returns the price per $100 face value of a discounted security.

Syntax

`PRICEDISC(`*`settle`*`,`*`mature`*`,`*`discount`*`,`*`redeem`*`,`*`basis`*`)`

Arguments

This function has these arguments:

Argument	Description
settle	Settlement date for the security.
mature	Maturity date for the security.
discount	Amount invested in the security.
redeem	Amount to be received at maturity.
basis	[Optional] Integer representing the basis for day count (Refer to *Day Count Basis* on page 13.)

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`PRICEDISC(A1,A2,A5,A7,1)`
`PRICEDISC(R1C1,R2C1,R5C5,R5C7,2)`
`PRICEDISC("5/15/2004","9/1/2004",0.06,100,3)` gives the result `98.20822`

Version Available

This function is available in product version 1.0 or later.

See Also

DISC | PRICEMAT | Financial Functions

PRICEMAT

Summary

This function returns the price at maturity per $100 face value of a security that pays interest.

Syntax

`PRICEMAT(`*`settle`*`,`*`mature`*`,`*`issue`*`,`*`rate`*`,`*`yield`*`,`*`basis`*`)`

Arguments

This function has these arguments:

Argument	Description
settle	Settlement date for the security
mature	Maturity date for the security
issue	Issue date for the security
rate	Interest rate for the security at the issue date
yield	Annual yield for the security
basis	[Optional] Integer representing the basis for day count (Refer to *Day Count Basis* on page 13.)

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`PRICEMAT(A1,A2,A5,A7,A7,1)`
`PRICEMAT(R1C1,R2C1,R5C5,R5C7,R5C9,2)`
`PRICEMAT("5/15/2004","9/1/2004","5/15/2003",0.06,0.07,3)` gives the result `99.5842915904314`

Version Available

This function is available in product version 1.0 or later.

See Also

DISC | PRICEDISC | Financial Functions

PROB

Summary

This function returns the probability that values in a range are between two limits.

Syntax

`PROB(array,probs,lower,upper)`

Arguments

This function has these arguments:

Argument	Description
array	Array of numeric values, which has corresponding probs
probs	Probabilities associated with the numeric values in array
lower	Lower limit on the numeric value for which you want a probability
upper	[Optional] Upper limit on the numeric value for which you want a probability; if omitted, returns the probability of result equal to lower limit

Remarks

If the *upper* argument is not provided, the function uses the value for the *lower* argument only, and returns the probability that the values are equal to the *lower* argument.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`PROB({B1:B6},{E1:E6},10,100)`
`PROB({B2,B4,B5,B7},{0.4,0.25,0.1,.025),10,100)`
`PROB({R1C2:R6C2},{R1C5:R6C5},1,50)`
`PROB({0,1,2,3},{0.2,0.3,0.1,0.4},2)` gives the result `0.1`

Version Available

This function is available in product version 1.0 or later.

See Also

BINOMDIST | CRITBINOM | Statistical Functions

PRODUCT

Summary

This function multiplies all the arguments and returns the product.

Syntax

```
PRODUCT(value1,value2,...)
PRODUCT(array)
PRODUCT(array1,array2,...)
```

Arguments

Each argument can be a double-precision floating-point value, an integer value, or an array (cell range) of these. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Remarks

If an array or reference argument contains text, logical values, or empty cells, the function ignores those values; however, the function includes in calculations cells with the value zero.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`PRODUCT(B3,B7,12)`
`PRODUCT(C4,B2,B4,C5)`
`PRODUCT(A1:A9)`
`PRODUCT(R1C2,2,10)`
`PRODUCT(A1:A8,B1:B8,C2:C18)`
`PRODUCT(1,2,3,5,7,11,13)` gives the result `30030`

Version Available

This function is available in product version 1.0 or later.

See Also

FACT | QUOTIENT | SUMPRODUCT | Statistical Functions

PROPER

Summary

This function capitalizes the first letter in each word of a text string.

Syntax

`PROPER(`*`text`*`)`

Arguments

The text argument can be a string, a formula that returns a string, or a reference to a cell containing a string.

Remarks

This function capitalizes letters that follow any character other than a letter, for example, a space. This function converts all other letters to lowercase letters.

Data Types

Accepts string data. Returns string data.

Examples

`PROPER(D2)`
`PROPER(""INTRO to SPREAD"")` gives the result `Intro To Spread`
`PROPER(""Tom's one-time order"")` gives the result `Tom’S One-Time Order`

Version Available

This function is available in product version 1.0 or later.

See Also

CHAR | UPPER | Text Functions

PV

Summary

This function returns the present value of an investment based on the interest rate, number and amount of periodic payments, and future value. The present value is the total amount that a series of future payments is worth now.

Syntax

`PV(rate,numper,paymt,fval,type)`

Arguments

This function has these arguments:

Argument	Description
rate	Interest rate expressed as percentage (per period)
numper	Total number of payment periods
paymt	Payment made each period; cannot change over the life of the annuity
fval	[Optional] Future value; if omitted, the calculation is based on the payments
type	[Optional] Indicates when payments are due; at the end (0) or beginning (1) of the period; if omitted, the calculation uses the end (0)

For the arguments, money paid out (such as deposits in an investment) is represented by negative numbers; money you receive (such as dividend checks) is represented by positive numbers.

Remarks

Use consistent units for specifying the rate and number of periods arguments. If you make monthly payments on a five-year loan at 8 percent annual interest, use 0.08/12 for the rate argument and 5*12 for the number of periods argument. If you make annual payments on the same loan, use 0.08 for rate and 5 for number of periods.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`PV(B1/12,N24,-75,0,1)`
`PV(R1C1/12,48,R1C2,0,0)`
`PV(0.005,60,-100,0,1)` gives the result `$5,198.42`

Version Available

This function is available in product version 1.0 or later.

See Also

FV | NPER | PMT | Financial Functions

QUARTILE

Summary

This function returns which quartile (which quarter or 25 percent) of a data set a value is.

Syntax

`QUARTILE(`*`array`*`,`*`quart`*`)`

Arguments

This function has these arguments:

Argument	Description
array	Array or cell range of numeric values for which you want the quartile value
quart	Quartile value for the array (see the table below for returned values)

Remarks

A quarter is 25 percent. So the quartile number is an integer between 0 (the minimum value in the data set) and 4 (the maximum value in the data set) and determines the value to return as listed in the table below.

If the number is...	Then this function returns the...
0	Minimum value
1	First quartile (25th percentile)
2	Median value (50th percentile)
3	Third quartile (75th percentile)
4	Maximum value

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`QUARTILE(A1:A17,2)`
`QUARTILE(R1C1:R17C1,3)`
`QUARTILE({11,21,42,27,18,29,32,52},1)` gives the result `20.25`

Version Available

This function is available in product version 1.0 or later.

See Also

PERCENTILE | PERCENTRANK | Statistical Functions

QUOTIENT

Summary

This function returns the integer portion of a division. Use this to ignore the remainder of a division.

Syntax

`QUOTIENT(numerator,denominator)`

Arguments

This function has these arguments:

Argument	Description
numerator	Numerator or dividend
denominator	Denominator or divisor

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`QUOTIENT(B8,B10)`
`QUOTIENT(R8B2,R10B2)`
`QUOTIENT(14,4)` gives the result `3`

Version Available

This function is available in product version 1.0 or later.

See Also

MOD | PRODUCT | Math and Trigonometry Functions

RADIANS

Summary

This function converts the specified number from degrees to radians.

Syntax

`RADIANS(value)`

Arguments

This function takes any real number angle value as the argument.

Remarks

Converts angle in degrees to angle in radians.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`RADIANS(B3)`
`RADIANS(R1C2)`
`RADIANS(45)` gives the result `0.7853981634` (which is p/4)

Version Available

This function is available in product version 1.0 or later.

See Also

DEGREES | PI | Math and Trigonometry Functions

RAND

Summary

This function returns an evenly distributed random number between 0 and 1.

Syntax

`RAND()`

Arguments

This function does not accept arguments.

Remarks

This function returns a new random number.

To generate a random real number between *x* and *y*, with *y* greater than *x*, use the following expression:

`RAND()*(y-x)+x`

To generate a random integer between x and y, with y greater than x, use the following expression:

`INT((y-x+1)*RAND()+x)`

This is a volatile function with version 2.5 or later. For more information, refer to *Volatile Functions* on page 9.

Data Types

Does not accept data. Returns numeric data.

Examples

```
RAND()
RAND()*100
INT(RAND()*100)
```

Version Available

This function is available in product version 1.0 or later. This function is a volatile function in version 2.5 or later.

See Also

RANDBETWEEN | INT | Math and Trigonometry Functions

RANDBETWEEN

Summary

This function returns a random number between the numbers you specify.

Syntax

`RANDBETWEEN(lower,upper)`

Arguments

This function has these arguments:

Argument	Description
lower	Lower number of two numbers between which a random number is chosen; this number must be less than *upper*
upper	Upper number of two numbers between which a random number is chosen

Remarks

This function returns a new random number every time the sheet is calculated.

This functions returns an integer value. The first argument must be less than the second argument.

This is a volatile function with version 2.5 or later. For more information, refer to *Volatile Functions* on page 9.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

```
RANDBETWEEN(A1,B2)
RANDBETWEEN(10,20)
RANDBETWEEN(10,40)*100
INT(RANDBETWEEN(1,10)*100)
```

Version Available

This function is available in product version 1.0 or later. This function is a volatile function in version 2.5 or later.

See Also

RAND | Math and Trigonometry Functions

RANK

Summary

This function returns the rank of a number in a set of numbers. If you were to sort the set, the rank of the number would be its position in the list.

Syntax

`RANK(number,array,order)`

Arguments

This function has these arguments:

Argument	Description
number	Number whose rank you want to return
array	Reference to the set of numbers
order	[Optional] How the number is ranked, either in descending order (0 or omitted) or ascending order (non-zero value)

Remarks

This function gives duplicate numbers the same rank. The presence of duplicate numbers affects the ranks of subsequent numbers. For example, in a list of integers, if the number 12 appears twice and has a rank of 4, then 13 would have a rank of 6 (no number would have a rank of 5).

Data Types

Accepts numeric data for the *number* argument, a reference for the *array* argument, and numeric data for the *order* argument. Returns numeric data.

Examples

```
RANK(B3,B1:B8,1)
RANK(R3C2,R1C2:R8C2,1)
RANK(16,{2,4,8,16,32},1) gives the result 4
```

Version Available

This function is available in product version 1.0 or later.

See Also

MEDIAN | MODE | Statistical Functions

RATE

Summary

This function returns the interest rate per period of an annuity.

Syntax

`RATE(nper,pmt,pval,fval,type)`

Arguments

This function has these arguments:

Argument	Description
nper	Total number of payment periods in an annuity
pmt	Value representing the payment made each period
pval	Present value, worth now
fval	Future value, cash value after the last payment
type	[Optional] Indicates when payments are due; at the end (0) or beginning (1) of the period; if omitted, the calculation uses the end (0)

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`RATE(A1,B2,C3,C4,1)`
`RATE(360, -600, 80000)` gives the result `0.686%`

Version Available

This function is available in product version 1.0 or later.

See Also

IPMT | PMT | PPMT | Financial Functions

RECEIVED

Summary

This function returns the amount received at maturity for a fully invested security.

Syntax

RECEIVED(*settle*,*mature*,*invest*,*discount*,*basis*)

Arguments

This function has these arguments:

Argument	Description
settle	Settlement date for the security
mature	Maturity date for the security
invest	Amount invested in the security
discount	Discount rate for the security
basis	[Optional] Integer representing the basis for day count (Refer to *Day Count Basis* on page 13.)

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

```
RECEIVED(A1,B2,C3,C4,1)
RECEIVED("3/01/2004","6/01/2004",600000,0.03,2) gives $604,635.50
```

Version Available

This function is available in product version 1.0 or later.

See Also

INTRATE | Financial Functions

REPLACE

Summary

This function replaces part of a text string with a different text string.

Syntax

`REPLACE(`*`old_text`*`,`*`start_char`*`,`*`num_chars`*`,`*`new_text`*`)`

Arguments

This function has these arguments:

Argument	Description
old_text	Original text in which you want to replace characters
start_char	Starting position in the original text to begin the replacement
num_chars	Number of characters in the original text that you want to replace with characters from the new text; if not an integer, the number is truncated
new_text	New text that replaces characters in the original text

Remarks

Use this function to replace a specified number of characters in a specified location with other characters. Use the SUBSTITUTE function to replace specific text with other text.

Data Types

Accepts string data for the *old_text* argument, numeric data for the *start_char* argument, numeric data for the *num_chars* argument, and string data for the *new_text* argument. Returns string data.

Examples

This example replaces three characters with one character, starting with the sixth character in the provided text:

`REPLACE(""abcdefghijk"", 6, 3, ""%"")` gives the result `abcde%ijk`

Version Available

This function is available in product version 1.0 or later.

See Also

FIND | SUBSTITUTE | Text Functions

REPT

Summary

This function repeats text a specified number of times.

Syntax

`REPT(`*`text`*`,`*`number`*`)`

Arguments

This function has these arguments:

Argument	Description
text	Text you want to repeat
number	Number of times you want to repeat the text; if not an integer, the number is truncated; if zero (0), returns empty (" ")

Remarks

The result of this function must be less than or equal to 255 characters.

Data Types

Accepts string data for the *text* argument and numeric data for the *number* argument. Returns string data.

Examples

`REPT(D9, 2)`
`REPT(R9C4, 2)`
`REPT(""*4"", 3)` gives the result `*4*4*4`

Version Available

This function is available in product version 1.0 or later.

See Also

CONCATENATE | Text Functions

RIGHT

Summary

This function returns the specified rightmost characters from a text value.

Syntax

`RIGHT(text,num_chars)`

Arguments

This function has these arguments:

Argument	Description
text	Text string from which you want to return characters
num_chars	[Optional] Number of characters to return; if omitted, calculation uses one (1); if not an integer, the number is truncated

The *text* argument can be a string, a formula that returns a string, or a reference to a cell containing a string.

The *num_chars* argument has these rules:

- The *num_chars* argument must be greater than or equal to zero.
- If the *num_chars* argument is greater than the length of text, this function returns all text.

Data Types

Accepts string data for the *text* argument and numeric data for the *num_chars* argument. Returns string data.

Examples

`RIGHT(""Total Sales"",5)` gives the result `Sales`
`RIGHT(""Collie dog"")` gives the result `g`

Version Available

This function is available in product version 1.0 or later.

See Also

LEFT | MID | Text Functions

ROMAN

Summary

This function converts an arabic numeral to a roman numeral text equivalent.

Syntax

`ROMAN(number,style)`

Arguments

This function has these arguments:

Argument	Description
number	Arabic number to convert
style	Type of roman numeral

Remarks

The style of roman numeral is set by the numeric value of the style argument:

Style **value**	**Roman numeral style**
0 or omitted	Classic
1	More concise
2	More concise
3	More concise
4	Simplified
TRUE	Classic
FALSE	Simplified

An error is returned if the *number* argument is negative.

Data Types

Accepts numeric data. Returns string data.

Examples

`ROMAN(100,3)`

Version Available

This function is available in product version 2.0 or later.

See Also

ABS | Math and Trigonometry Functions

ROUND

Summary

This function rounds the specified value to the nearest number, using the specified number of decimal places.

Syntax

`ROUND(`*`value`*`,`*`places`*`)`

Arguments

Use the *value* argument to specify the number to round. Use the *places* argument to specify the number of decimal places. The *places* argument has these rules:

- Set *places* to a value greater than zero to round to the specified number of decimal places.
- Set *places* to zero to round to the nearest whole number.
- Set *places* to a value less than zero to round the value left of the decimal to the nearest ten, hundred, etc.

Remarks

The result may be rounded up or rounded down.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`ROUND(A3,-2)`
`ROUND(C4,B2)`
`ROUND(R1C2,2)`
`ROUND(PI(),5)` gives the result `3.14159`
`ROUND(29.2,-2)` gives the result `0` because 29.2 is closer to 0 than to 100.
`ROUND(-1.963,0)` gives the result `-2`

Version Available

This function is available in product version 1.0 or later.

See Also

ROUNDDOWN | ROUNDUP | CEILING | FLOOR | MROUND | Math and Trigonometry Functions

ROUNDDOWN

Summary

This function rounds the specified number down to the nearest number, using the specified number of decimal places.

Syntax

`ROUNDDOWN(value,places)`

Arguments

Use the *value* argument to specify the number to round. Use the *places* argument to specify the number of decimal places. The *places* argument has these rules:

- Set *places* to a value greater than zero to round to the specified number of decimal places.
- Set *places* to zero to round to the nearest whole number.
- Set *places* to a value less than zero to round the value left of the decimal to the nearest ten, hundred, etc.

Regardless of the sign of the number specified by the *value* argument, the number is rounded away from zero.

Remarks

The result is always rounded down.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`ROUNDDOWN(3.2,0)` gives the result `3`
`ROUNDDOWN(D14,3)`
`ROUNDOWN(R14C4,10)`
`ROUNDDOWN(3.14159,3)` gives the result `3.141`
`ROUNDDOWN(-3.14159,1)` gives the result `-3.1`
`ROUNDDOWN(31415.92654,-2)` gives the result `31400`

Version Available

This function is available in product version 1.0 or later.

See Also

ROUND | ROUNDUP | CEILING | FLOOR | Math and Trigonometry Functions

ROUNDUP

Summary

This function rounds the specified number up to the nearest number, using the specified number of decimal places.

Syntax

`ROUNDUP(`*`value`*`,`*`places`*`)`

Arguments

Use the *value* argument to specify the number to round. Use the *places* argument to specify the number of decimal places. The *places* argument has these rules:

- Set *places* to a value greater than zero to round to the specified number of decimal places.
- Set *places* to zero to round to the nearest whole number.
- Set *places* to a value less than zero to round the value left of the decimal to the nearest ten, hundred, etc.

Remarks

Regardless of the sign of the number specified by the *value* argument, the number is rounded away from zero.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`ROUNDUP(A3,-2)`
`ROUNDUP(C4,B2)`
`ROUNDUP(R1C2, 2)`
`ROUNDUP(PI(),5)` gives the result `3.14160`
`ROUNDUP(29.2,-2)` gives the result `100`
`ROUNDUP(-1.963,0)` gives the result `-2`

Version Available

This function is available in product version 1.0 or later.

See Also

ROUND | ROUNDDOWN | CEILING | FLOOR | Math and Trigonometry Functions

ROW

Summary

This function returns the number of a row from a reference.

Syntax

`ROW(`*`reference`*`)`

Arguments

The argument is a cell or a single area.

Remarks

If the reference is omitted, the reference of the cell that the function is in is used.

Data Types

Accepts a cell or a single area. Returns numeric data.

Examples

`ROW(B2)` gives the result `2`
`ROW(B1:B5)` gives the result `1`

Version Available

This function is available in product version 3.0 or later.

See Also

COLUMNS | INDEX | Lookup Functions

ROWS

Summary

This function returns the number of rows in an array.

Syntax

`ROWS(`*`array`*`)`

Arguments

The argument is an array, an array formula, or a range of cells.

Data Types

Accepts array. Returns numeric data.

Examples

`ROWS(B2:B14)` gives the result `13`
`ROWS(R2C6:R4C12)` gives the result `3`
`ROWS($H$2:$H$8)` gives the result `7`
`ROWS(R[2]C[3]:R[8]C[3])` gives the result `7`
`ROWS(R3C2:R17C2)` gives the result `15`

Version Available

This function is available in product version 1.0 or later.

See Also

COLUMNS | INDEX | Lookup Functions

RSQ

Summary

This function returns the square of the Pearson product moment correlation coefficient (R-squared) through data points in known y's and known x's.

Syntax

`RSQ(array_dep,array_ind)`

Arguments

This function has these arguments:

Argument	Description
array_dep	Array of dependent values (y's)
array_ind	Array of independent values (x's)

The arrays must be the same size.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`RSQ(B2:B14,H2:H14)`
`RSQ(R2C2:R14C2,R2C8:R14C8)`
`RSQ({2,4,6},{10,15,25})` gives the result `0.964286`

Version Available

This function is available in product version 1.0 or later.

See Also

PEARSON | Statistical Functions

SECOND

Summary

This function returns the seconds (0 to 59) value for a specified time.

Syntax

`SECOND(`*`time`*`)`

Arguments

Specify the time argument as a number (as in 37806.5) a string (as in "7/4/2003 12:00"), a DateTime object, as in DATE(2003,7,4), or a TimeSpan object, as in TIME(12,0,0). For more details on the date and time inputs, refer to the discussion in *Date and Time Functions* on page 10

Dates as numeric values are in the form x.y, where x is the "number of days since December 30, 1899" and y is the fraction of day. Numbers to the left represent the date. Times as numeric values are decimal fractions ranging from 0 to 0.99999999, representing the times from 0:00:00 (12:00:00 A.M.) to 23:59:59 (11:59:59 P.M.).

Remarks

The second is returned as an integer, ranging from 0 to 59

Data Types

Accepts numeric, string, DateTime object, or TimeSpan object data. Returns numeric data.

Examples

`SECOND(A2)`
`SECOND(R2C1)`
`SECOND(0.01)` gives the result `24`
`SECOND(TIME(12,0,0))`

Version Available

This function is available in product version 1.0 or later.

See Also

HOUR | MINUTE | Date and Time Functions

SERIESSUM

Summary

This function returns the sum of a power series.

Syntax

`SERIESSUM(x,n,m,coeff)`

Arguments

This function has these arguments:

Argument	Description
x	Value to evaluate in the power series
n	Power to which to raise x
m	Step by which to increase n for each term in the series
coeff	Set of coefficients for the series (the values of a1, a2, ... ai)

Remarks

The power series formula is:

$$SERIESSUM(\mathrm{x, n, m, a}) \approx a_1 x^n + a_2 x^{(n+m)} + a_3 x^{(n+2m)} + \ldots + a_i x^{(n+(i-1))}$$

where x, n, and m are the similarly named arguments and a is the *coeff* argument.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

```
SERIESSUM(34,3,2,A1:A6)
SERIESSUM(12,3,1,B2:B24)
```

Version Available

This function is available in product version 1.0 or later.

See Also

SUM | Math and Trigonometry Functions

SIGN

Summary

This function returns the sign of a number or expression.

Syntax

`SIGN(`*`cellreference`*`)`
`SIGN(`*`value`*`)`
`SIGN(`*`expression`*`)`

Arguments

Specify a cell reference, a numeric or text value, or an expression for the argument.

Remarks

Returns 1 if the number is positive, 0 if the number is 0, and –1 if the number is negative.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`SIGN(B3)`
`SIGN(R1C2)`
`SIGN(-5)` gives the result `-1`
`SIGN(15-8)` gives the result `1`

Version Available

This function is available in product version 1.0 or later.

See Also

ABS | Math and Trigonometry Functions

SIN

Summary

This function returns the sine of the specified angle.

Syntax

`SIN(`*`angle`*`)`

Arguments

This function can take any real number as an argument. The *angle* argument is the angle in radians for which you want the sine.

Remarks

If the angle is in degrees, multiply it by PI/180 to convert it to radians.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`SIN(B4)`
`SIN(R1C2)`
`SIN(30*PI()/180)` gives the result `0.5`
`SIN(RADIANS(45))`

Version Available

This function is available in product version 1.0 or later.

See Also

ACOS | ASIN | COS | SINH | Math and Trigonometry Functions

SINH

Summary

This function returns the hyperbolic sine of the specified number.

Syntax

`SINH(value)`

Arguments

You can use any real number for the *value* argument.

Remarks

The equation for calculating the hyperbolic sine is:

$$SINH(z) = \frac{e^{z} - e^{-z}}{2}$$

where z is the *value* argument.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`SINH(B4)`
`SINH(R1C2)`
`SINH(1)` gives the result `1.1752011936`

Version Available

This function is available in product version 1.0 or later.

See Also

ACOSH | ASINH | SIN | COSH | Math and Trigonometry Functions

SKEW

Summary

This function returns the skewness of a distribution.

Syntax

`SKEW(`*`number1`*`,`*`number2`*`,...)`

Arguments

The arguments are numeric values. Only the first argument is required. Up to 255 arguments may be included.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`SKEW(A1,B2,B3,C1,C4)`

Version Available

This function is available in product version 1.0 or later.

See Also

KURT | Statistical Functions

SLN

Summary

This function returns the straight-line depreciation of an asset for one period.

Syntax

`SLN(cost,salvage,life)`

Arguments

This function has these arguments:

Argument	Description
cost	Initial cost of the asset
salvage	Value at the end of the depreciation
life	Number of periods over which the asset is being depreciated

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`SLN(B1,1000,10)`
`SLN(R1C2,1000,10)`
`SLN(500000,20000,5)` gives the result `$96,000`

Version Available

This function is available in product version 1.0 or later.

See Also

DB | DDB | SYD | Financial Functions

SLOPE

Summary

This function calculates the slope of a linear regression.

Syntax

`SLOPE(`*`array_dep`*`,`*`array_ind`*`)`

Arguments

This function has these arguments:

Argument	Description
array_dep	Array of dependent values (y's)
array_ind	Array of independent values (x's)

The arrays must be the same size.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`SLOPE(A1:A4,B1:B4)`

Version Available

This function is available in product version 1.0 or later.

See Also

SERIESSUM | Math and Trigonometry Functions

SMALL

Summary

This function returns the *n*th smallest value in a data set, where *n* is specified.

Syntax

`SMALL(array,n)`

Arguments

This function has these arguments:

Argument	Description
array	Array from which to return the nth largest value
n	The position (from the largest value) for which to return the value (for example, 5 to return the fifth largest value). Must be equal to or less than the number of items in the array.

Remarks

Use this function to select a value based on its relative standing.

Data Types

Accepts array and numeric data for all arguments. Returns numeric data.

Examples

`SMALL(B4:B8,2)`
`SMALL(R4C2:R8C2,2)`
`SMALL({15, 20, 10, 5}, 2)` gives the result `10`

Version Available

This function is available in product version 1.0 or later.

See Also

LARGE | Statistical Functions

SQRT

Summary

This function returns the positive square root of the specified number.

Syntax

`SQRT(`*`value`*`)`

Arguments

The argument may be any positive numeric value. You must provide a positive number for the argument.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`SQRT(B4)`
`SQRT(R4C2)`
`SQRT(256)` gives the result `16`

Version Available

This function is available in product version 1.0 or later.

See Also

POWER | EXP | Math and Trigonometry Functions

SQRTPI

Summary

This function returns the positive square root of a multiple of pi (p).

Syntax

`SQRTPI(multiple)`

Arguments

Specify the number of multiples of pi (p) of which to calculate the square root.

Remarks

This function calculates the square root of a multiple of pi.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`SQRTPI(A3)`
`SQRTPI(1)` is the same as `SQRT(PI())`
`SQRTPI(5)` gives the result `3.963327`

Version Available

This function is available in product version 1.0 or later.

See Also

PI | SQRT | Statistical Functions

STANDARDIZE

Summary

This function returns a normalized value from a distribution characterized by mean and standard deviation.

Syntax

`STANDARDIZE(`*`x`*`,`*`mean`*`,`*`stdev`*`)`

Arguments

This function has these arguments:

Argument	Description
x	Value to normalize
mean	Arithmetic mean of the distribution
stdev	Standard deviation of the distribution Must be greater than zero.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`STANDARDIZE(15.6,A4,B2)`
`STANDARDIZE(88,48,1.6)` gives the result `25`

Version Available

This function is available in product version 1.0 or later.

See Also

NORMDIST | NORMSDIST | Statistical Functions

STDEV

Summary

This function returns the standard deviation for a set of numbers.

Syntax

`STDEV(value1,value2,...)`

Arguments

Each argument can be a cell, a cell range, a float value, or an integer value. This function can have up to 255 arguments.

Remarks

The standard deviation is a measure of how widely values are dispersed from the average value.

The standard deviation is calculated using the "non-biased" or "n–1" method.

The equation for calculating the standard deviation is:

$$STDEV(x_n) = \sqrt{\frac{n\sum x^2 - (\sum x)^2}{n(n-1)}}$$

where x is the value and n is the number of values.

This function assumes that its arguments are a sample of the population. If your data represents the entire population, then compute the standard deviation using the STDEVP function.

This function differs from the STDEVA, which allows text or logical values as well as numeric values.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`STDEV(A1,B2,C3,D4,E5,F6)`
`STDEV(A1:A9)`
`STDEV(R1C2,R3C4,R4C5,R7C2)`
`STDEV(95,89,73,87,85,76,100,96,96)` gives the result `9.3422576382`

Version Available

This function is available in product version 1.0 or later.

See Also

AVEDEV | AVERAGE | Statistical Functions

STDEVA

Summary

This function returns the standard deviation for a set of numbers, text, or logical values.

Syntax

`STDEVA(value1,value2,...)`

Arguments

Each argument can be a cell, a cell range, a float value, an integer value, text, or a logical value. There can be up to 255 arguments. TRUE evaluates to 1 and FALSE or text evaluates to 0.

Remarks

The standard deviation is a measure of how widely values are dispersed from the average value.

The standard deviation is calculated using the "non-biased" or "n–1" method.

The equation for calculating the standard deviation is the same as for STDEV:

$$STDEVA(x_n) = \sqrt{\frac{n\sum x^2 - (\sum x)^2}{n(n-1)}}$$

where x is the value and n is the number of values

This function assumes that its arguments are a sample of the population.

This function differs from STDEV because it accepts text or logical values as well as numeric values.

Data Types

Accepts numeric, text, and logical data for all arguments. Returns numeric data.

Examples

`STDEVA(A1,B2,C3,D4,E5,F6)`
`STDEVA(A1:A9)`
`STDEVA(R1C2,R3C4,R4C5,R7C2)`
`STDEVA(95,89,73,87,85,76,100,96,96)` gives the result `9.3422576382`

Version Available

This function is available in product version 2.0 or later.

See Also

AVEDEV | AVERAGE | STDEV | STDEVPA | Statistical Functions

STDEVP

Summary

This function returns the standard deviation for an entire specified population (of numeric values).

Syntax

`STDEVP(value1,value2,...)`

Arguments

Each argument can be a cell, a cell range, a float value, or an integer value. This function can have up to 255 arguments.

Remarks

The standard deviation is a measure of how widely values are dispersed from the average value.

The standard deviation is calculated using the "biased" or "n" method.

The equation for calculating the standard deviation for a population is:

$$STDEVP(x_n) = \sqrt{\frac{n\sum x^2 - (\sum x)^2}{n^2}}$$

where x is the value and n is the number of values.

This function assumes that its arguments are the entire population. If your data represents a sample of the population, then compute the standard deviation using the STDEV function.

This function differs from STDEVPA, which accepts text or logical values as well as numeric values.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`STDEVP(A1,B2,C3,D4,E5,F6)`
`STDEVP(A1:A9)`
`STDEVP(R1C2,R3C4,R4C5,R7C2)`
`STDEVP(95,89,73,87,85,76,100,96,96)` gives the result `8.8079649700`

Version Available

This function is available in product version 1.0 or later.

See Also

AVERAGE | STDEV | STDEVPA | Statistical Functions

STDEVPA

Summary

This function returns the standard deviation for an entire specified population, including text or logical values as well as numeric values.

Syntax

`STDEVPA(value1,value2,...)`

Arguments

Each argument can be a cell, a cell range, a float value, text, a logical value, or an integer value. There can be up to 255 arguments. TRUE evaluates as 1. Text or FALSE evaluates as 0.

Remarks

The standard deviation is a measure of how widely values are dispersed from the average value.

The standard deviation is calculated using the "biased" or "n" method.

The equation for calculating the standard deviation for a population is:

$$STDEVPA(x_n) = \sqrt{\frac{n\sum x^2 - (\sum x)^2}{n^2}}$$

where x is the value and n is the number of values.

This function assumes that its arguments are the entire population. If your data represents a sample of the population, then compute the standard deviation using the STDEVA function.

This function differs from STDEVP because it accepts text or logical values as well as numeric values.

Data Types

Accepts numeric, text, and logical data for all arguments. Returns numeric data.

Examples

```
STDEVPA(A1,B2,C3,D4,E5,F6)
STDEVPA(A1:A9)
STDEVPA(R1C2,R3C4,R4C5,R7C2)
```

`STDEVPA(95,89,73,87,85,76,100,96,96)` gives the result `8.8079649700`

Version Available

This function is available in product version 2.0 or later.

See Also

AVERAGE | STDEVP | STDEVA | Statistical Functions

STEYX

Summary

This function returns the standard error of the predicted y value for each x. The standard error is a measure of the amount of error in the prediction of y for a value of x.

Syntax

`STEYX(`*`array_dep`*`,`*`array_ind`*`)`

Arguments

This function has these arguments:

Argument	Description
array_dep	Array of dependent values (y’s)
array_ind	Array of independent values (x’s)

The arrays must be the same size.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`STEYX(A1:A17,B1:B17)`
`STEYX({22,33,49,21,32,37,43},{31,28,29,42,35,37,34])` gives the result `10.14406`

Version Available

This function is available in product version 1.0 or later.

See Also

ERF | PEARSON | Statistical Functions

SUBSTITUTE

Summary

This function substitutes a new string for specified characters in an existing string.

Syntax

`SUBSTITUTE(`*`text`*`,`*`old_piece`*`,`*`new_piece`*`,`*`instance`*`)`

Arguments

This function has these arguments:

Argument	Description
text	String or reference to a cell containing the string in which you want to replace characters
old_piece	String to be replaced
new_piece	New string to use instead of existing string
instance	[Optional] Which occurrence of the existing string to replace; otherwise every occurrence is replaced

Remarks

Use this function to replace specific text with other text. Use the REPLACE function to replace a specific number of characters in a specific location with other characters.

Data Types

Accepts string data for the *text*, *old_piece*, and *new_piece* arguments, and numeric data for the *instance* argument. Returns string data.

Examples

`SUBSTITUTE(""Down Trend"",""Down"",""Up"")` gives the result `Up Trend`
`SUBSTITUTE(""Feb 1, 1991"",""1"",""2"", 3)` gives the result `Feb 1, 1992`

Version Available

This function is available in product version 1.0 or later.

See Also

FIND | REPLACE | TRIM | Text Functions

SUBTOTAL

Summary

This function calculates a subtotal of a list of numbers using a specified built-in function.

Syntax

```
SUBTOTAL(functioncode,value1,value2,...)
SUBTOTAL(functioncode,array)
```

Arguments

The *functioncode* argument is the number that represents the built-in function to use for the subtotal, as given in this table.

Built-In Function	Function Code
AVERAGE	1
COUNT	2
COUNTA	3
MAX	4
MIN	5
PRODUCT	6
STDEV	7
STDEVP	8
SUM	9
VAR	10
VARP	11

Each additional argument can be a double-precision floating-point value, an integer value, or an array (cell range) of these. Up to 255 arguments can be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Remarks

The SUBTOTAL function does not include other SUBTOTAL formula results that are in the same range.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

```
SUBTOTAL(8,A1:B7)
```

Version Available

This function is available in product version 2.0 or later.

See Also

SUMPRODUCT | SUM | Math and Trigonometry Functions

SUM

Summary

This function returns the sum of cells or range of cells.

Syntax

SUM(*value1*,*value2*,...)
SUM(*array*)
SUM(*array1*,*array2*,...)

Arguments

Each argument can be a double-precision floating-point value, an integer value, or an array (cell range) of these. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Remarks

Range references with mixed relativeness for column or row end points are not supported with the SUM function. R1C[1]:R2C[2] is okay but, R1C1:R2C[2] is not.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

SUM(A1,B7,C11)
SUM(A1:A9)
SUM(A2:A14,B2:B18,D12:D30)
SUM(R1C2,R3C5,R6C2,R1C7)
SUM(95,89,73,87,85,76,100,96,96) gives the result 797

Version Available

This function is available in product version 1.0 or later.

See Also

SUMPRODUCT | SERIESSUM | PRODUCT | Math and Trigonometry Functions

SUMIF

Summary

This function adds the cells using a given criteria.

Syntax

`SUM(`*`array`*`,`*`condition`*`,`*`sumrange`*`)`

Arguments

This function has these arguments:

Argument	Description
array	Range of cells to check; each cell in the array can be a double-precision floating-point value or an integer value
condition	Condition that determines which cells are added, as a text, number, or expression (where expressions use the relational operators detailed in *Operators in a Formula* on page 6)
sumrange	[Optional] Range of cells to add; if omitted, then all the cells in the array are added

Data Types

Accepts numeric data for *array* and *sumrange*. Accepts text, numeric or expression data for *condition*. Returns numeric data.

Examples

```
SUMIF(A1:B7,">150",C1:C11)
SUMIF(A1:A9,">150")
```

Version Available

This function is available in product version 2.0 or later.

See Also

SUMPRODUCT | SUM | COUNTIF | Math and Trigonometry Functions

SUMPRODUCT

Summary

This function returns the sum of products of cells. Multiplies corresponding components in the given arrays, and returns the sum of those products.

Syntax

`SUMPRODUCT(array1,array2,...)`

Arguments

There must be at least two arrays (*array1*, *array2*) and optionally up to 255 arrays (*array3*, ...) as arguments. The arrays must have the same dimension.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`SUMPRODUCT(A1:A17,B1:B17,C1:C17)`
`SUMPRODUCT({2,3,5,6,4,7},{5,6,4,4,7,2})` gives the result `114`

Version Available

This function is available in product version 1.0 or later.

See Also

PRODUCT | SUM | Math and Trigonometry Functions

SUMSQ

Summary

This function returns the sum of the squares of the arguments.

Syntax

`SUMSQ(`*`value1`*`,`*`value2`*`,...)`
`SUMSQ(`*`array`*`)`
`SUMSQ(`*`array1`*`,`*`array2`*`,...)`

Arguments

Each argument can be a double-precision floating-point value, an integer value, or an array (cell range) of these. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`SUMSQ(A1,B7,C11)`
`SUMSQ(A1:A9)`
`SUMSQ(R1C2,R3C5,R6C2,R1C7)`
`SUMSQ(95,89,73,87,85,76,100,96,96)` gives the result `71277`

Version Available

This function is available in product version 1.0 or later.

See Also

SUMPRODUCT | SUM | Math and Trigonometry Functions

SUMX2MY2

Summary

This function returns the sum of the difference of the squares of corresponding values in two arrays.

Syntax

`SUMX2MY2(array_x,array_y)`

Arguments

This function has these arguments:

Argument	Description
array_x	First array of values (x's)
array_y	Second array of values (y's)

The arrays must be the same size.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`SUMX2MY2(A1:A17,B1:B17)`

Version Available

This function is available in product version 1.0 or later.

See Also

SUMX2PY2 | SUMXMY2 | SUM | Math and Trigonometry Functions

SUMX2PY2

Summary

This function returns the sum of the sum of squares of corresponding values in two arrays.

Syntax

`SUMX2PY2(array_x,array_y)`

Arguments

This function has these arguments:

Argument	Description
array_x	First array of values (x's)
array_y	Second array of values (y's)

The arrays must be the same size.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`SUMX2PY2(A1:A17,B1:B17)`

Version Available

This function is available in product version 1.0 or later.

See Also

SUMX2MY2 | SUMXMY2 | SUM | Math and Trigonometry Functions

SUMXMY2

Summary

This function returns the sum of the square of the differences of corresponding values in two arrays.

Syntax

`SUMXMY2(array_x,array_y)`

Arguments

This function has these arguments:

Argument	Description
array_x	First array of values (x's)
array_y	Second array of values (y's)

The arrays must be the same size.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`SUMXMY2(A1:A17,B1:B17)`

Version Available

This function is available in product version 1.0 or later.

See Also

SUMX2PY2 | SUMX2MY2 | SUM | Math and Trigonometry Functions

SYD

Summary

This function returns the sum-of-years' digits depreciation of an asset for a specified period.

Syntax

`SYD(cost,salvage,life,period)`

Arguments

This function has these arguments:

Argument	Description
cost	Initial cost of the asset
salvage	Value at the end of the depreciation
life	Number of periods over which the asset is being depreciated
period	Period for depreciation; must use the same units as the *life* argument.

Remarks

This function calculates the digits depreciation as follows:

$$SYD = \frac{(cost - salvage) \times (life - period + 1) \times 2}{(life)(life + 1)}$$

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`SYD(B1,1000,10,1)`
`SYD(R1C2,1000,10,1)`
`SYD(100000,10000,5,2)` gives the result `$2,4000`

Version Available

This function is available in product version 1.0 or later.

See Also

DB | DDB | SLN | Financial Functions

T

Summary

This function returns the text in a specified cell.

Syntax

`T(value)`

Arguments

The argument is any cell reference.

Remarks

If the cell contains text, this function returns text. If the cell contains a number, this function returns an empty string.

Data Types

Accepts cell reference. Returns string data.

Examples

```
T(B3) If B3 contains "Test" then this function returns "Test".
T(R3C2)
T(A1)
```

Version Available

This function is available in product version 2.0 or later.

See Also

LEN | ISTEXT | CHAR | UPPER | LOWER | Text Functions

TAN

Summary

This function returns the tangent of the specified angle.

Syntax

`TAN(`*`angle`*`)`

Arguments

This function can take any real number as an argument. The *angle* argument is the angle in radians for which you want the tangent.

Remarks

If the angle is in degrees, multiply it by PI/180 to convert it to radians.

Data Types

Accepts numeric data. Returns numeric data.

Examples

`TAN(B3)`
`TAN(R3C2)`
`TAN(45*PI()/180)` gives the result `1`
`TAN(RADIANS(20))`

Version Available

This function is available in product version 1.0 or later.

See Also

ATAN | ATAN2 | COS | SIN | Math and Trigonometry Functions

TANH

Summary

This function returns the hyperbolic tangent of the specified number.

Syntax

`TANH(value)`

Remarks

You can use any real number for the *value* argument.

The equation for calculating the hyperbolic sine is:

$$TANH(z) = \frac{SINH(z)}{COSH(z)}$$

Data Types

Accepts numeric data. Returns numeric data.

Examples

`TANH(B3)`
`TANH(R1C2)`
`TANH(0.5)` gives the result `0.4621171573`

Version Available

This function is available in product version 1.0 or later.

See Also

ATAN | ATANH | COSH | SINH | TAN | Math and Trigonometry Functions

TBILLEQ

Summary

This function returns the equivalent yield for a Treasury bill (or T-bill).

Syntax

`TBILLEQ(`*`settle`*`,`*`mature`*`,`*`discount`*`)`

Arguments

This function has these arguments:

Argument	Description
settle	Settlement date for the Treasury bill
mature	Maturity date for the Treasury bill
discount	Discount rate for the Treasury bill

Data Types

Accepts numeric and DateTime object data for all arguments. Returns numeric data.

Examples

`TBILLEQ(A1,B2,C3)`
`TBILLEQ("3/31/2003","6/1/2003",0.0532)` gives the result `0.054437659` (or `5.44%`)

Version Available

This function is available in product version 1.0 or later.

See Also

TBILLPRICE | TBILLYIELD | Financial Functions

TBILLPRICE

Summary

This function returns the price per $100 face value for a Treasury bill (or T-bill).

Syntax

`TBILLPRICE(settle,mature,discount)`

Arguments

This function has these arguments:

Argument	Description
settle	Settlement date for the Treasury bill
mature	Maturity date for the Treasury bill
discount	Discount rate for the Treasury bill

Data Types

Accepts numeric and DateTime object data for all arguments. Returns numeric data.

Examples

`TBILLPRICE(A1,B2,C3)`
`TBILLPRICE("3/31/2003","6/1/2003",0.065)` gives the result `$98.88055556`

Version Available

This function is available in product version 1.0 or later.

See Also

TBILLEQ | TBILLYIELD | Financial Functions

TBILLYIELD

Summary

This function returns the yield for a Treasury bill (or T-bill).

Syntax

`TBILLYIELD(`*`settle`*`,`*`mature`*`,`*`priceper`*`)`

Arguments

This function has these arguments:

Argument	Description
settle	Settlement date for the Treasury bill
mature	Maturity date for the Treasury bill
priceper	Price per $100 face value for the Treasury bill

Data Types

Accepts numeric and DateTime object data for all arguments. Returns numeric data.

Examples

`TBILLYIELD(A1,B2,C3)`
`TBILLYIELD("3/31/2003","6/1/2003",98.65)` gives the result `0.0794598041299475` (or `5.80%`)

Version Available

This function is available in product version 1.0 or later.

See Also

TBILLEQ | TBILLPRICE | Financial Functions

TDIST

Summary

This function returns the probability for the t-distribution.

Syntax

`TDIST(x,deg,tails)`

Arguments

This function has these arguments:

Argument	Description
x	Probability of the two-tailed student's t-distribution
deg	Number of degrees of freedom to characterize the distribution; if not an integer, the number is truncated
tails	Number of tails to return; if not an integer, the number is truncated; for 1, returns one-tailed distribution; for 2, returns two-tailed distribution

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`TDIST(A1,B45,2)`
`TDIST(0.245,2,1)` gives the result `0.414651`

Version Available

This function is available in product version 1.0 or later.

See Also

FDIST | TINV | TTEST | Statistical Functions

TIME

Summary

This function returns the TimeSpan object for a specified time.

Syntax

`TIME(`*`hour`*`,`*`minutes`*`,`*`seconds`*`)`

Arguments

This function has these arguments:

Argument	Description
hour	Hour as a number from 0 to 23.
minutes	Minutes as a number from 0 to 59.
seconds	Seconds as a number from 0 to 59.

Data Types

Accepts numeric data for all arguments. Returns a TimeSpan object.

Examples

`TIME(A1,B1,C1)`
`TIME(R1C1,R1C2,R1C3)`
`TIME(12,0,0)` gives the result `12:00:00`
`TIME(16,48,10)` gives the result `16:48:10`

Version Available

This function is available in product version 1.0 or later.

See Also

HOUR | MINUTE | DAY | NOW | TODAY | Date and Time Functions

TIMEVALUE

Summary

This function returns the TimeSpan object of the time represented by a text string.

Syntax

`TIMEVALUE(`*`time_string`*`)`

Arguments

Specify a time as a text string.

Remarks

Use this function to convert a time represented by text to a TimeSpan object in standard format. The time span is an amount of days, hours, minutes, and seconds.

Data Types

Accepts string data. Returns a TimeSpan object.

Examples

`TIMEVALUE(B18)`
`TIMEVALUE(R18C2)`
`TIMEVALUE("5:29")` gives the result `05:29`
`TIMEVALUE("5:29 PM")` gives the result `17:29`
`TIMEVALUE("17:29")` gives the result `17:29`

Version Available

This function is available in product version 1.0 or later.

See Also

TIME | DATEVALUE | Date and Time Functions

TINV

Summary

This function returns the t-value of the student's t-distribution as a function of the probability and the degrees of freedom.

Syntax

`TINV(prog,deg)`

Arguments

This function has these arguments:

Argument	Description
prog	Probability of the two-tailed student's t-distribution
deg	Number of degrees of freedom to characterize the distribution; if not an integer, the number is truncated

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`TINV(A4,2)`
`TINV(0.68,4)` gives the result `0.444006`

Version Available

This function is available in product version 1.0 or later.

See Also

TDIST | TTEST | Statistical Functions

TODAY

Summary

This function returns the date and time of the current date.

Syntax

`TODAY()`

Arguments

This function does not accept arguments.

Remarks

If you use this function in a date-time cell (DateTimeCellType), the cell formats the value using the date format settings.

This function is updated only when the spreadsheet or cell containing the function is recalculated. This is a volatile function with version 2.5 or later.

Data Types

Does not accept data. Returns a DateTime object.

Examples

If today is the 14th of November in the year 2003, then

`TODAY()` gives the result `November 14, 2003 12:00:00AM`

Version Available

This function is available in product version 1.0 or later. This function is a volatile function in version 2.5 or later.

See Also

DATE | DAY | NOW | TIME | Date and Time Functions

TRANSPOSE

Summary

This function returns a vertical range of cells as a horizontal range or a horizontal range of cells as a vertical range.

Syntax

`TRANSPOSE(`*`array`*`)`

Arguments

The *array* argument is a range of cells or an array that you want to switch.

Remarks

This function uses the first row of the array as the first column of the new array and so on.

Use the INDEX function to get individual elements from the returned array.

Data Types

Accepts an array. Returns an array.

Examples

`TRANSPOSE(A2:A5)`

Version Available

This function is available in product version 2.0 or later.

See Also

HLOOKUP | INDEX | LOOKUP | VLOOKUP | Lookup Functions

TREND

Summary

This function returns values along a linear trend. This function fits a straight line to the arrays known x and y values. Trend returns the y values along that line for the array of specified new x values.

Syntax

`TREND(y,x,newx,constant)`

Arguments

This function has these arguments:

Argument	Description
y	Set of y values that are known in the relationship y=mx+b
x	(Optional) X is an optional set of x values that may be known in the relationship y=mx+b
newx	New x values for which this functions returns the corresponding y values
constant	Logical value that specifies whether to force the constant b to equal 0

Remarks

If *constant* is true or omitted then b is calculated normally. If constant is false then b is equal to 0 and the m values are adjusted so that y=mx.

If *x* is omitted then *x* defaults to the array {1,2,3...}, that has the same dimensions as *y*.

If *newx* is omitted then it defaults to x.

Use the INDEX function to get individual elements from the returned array.

Data Types

Accepts an array. Returns an array.

Examples

`TREND(A2:A7,C2:C7,A9:A10)`

Version Available

This function is available in product version 2.0 or later.

See Also

AVEDEV | AVERAGEA | FREQUENCY | DEVSQ | GROWTH | INDEX | MEDIAN | VAR | Statistical Functions

TRIM

Summary

This function removes extra spaces from a string and leaves single spaces between words.

Syntax

`TRIM(`*`text`*`)`

Arguments

The argument specifies the string containing the spaces you want to remove.

Data Types

Accepts string data. Returns string data.

Examples

`TRIM("" First     Quarter"")` gives the result `First Quarter`

Version Available

This function is available in product version 1.0 or later.

See Also

CLEAN | SUBSTITUTE | Text Functions

TRIMMEAN

Summary

This function returns the mean of a subset of data excluding the top and bottom data.

Syntax

`TRIMMEAN(array,percent)`

Arguments

This function has these arguments:

Argument	Description
array	Array of values to trim and find the mean
percent	Fractional amount of data in array to trim (to exclude from calculation)

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`TRIMMEAN(A1:A17,0.25)`

Version Available

This function is available in product version 1.0 or later.

See Also

GEOMEAN | HARMEAN | Statistical Functions

TRUE

Summary

This function returns the value for logical TRUE.

Syntax

`TRUE()`

Arguments

This function does not accept arguments.

Data Types

Does not accept data. Returns numeric (boolean) data.

Example

`TRUE()` gives the result `1 (TRUE)`

Version Available

This function is available in product version 1.0 or later.

See Also

FALSE | IF | Logical Functions

TRUNC

Summary

This function removes the specified fractional part of the specified number.

Syntax

`TRUNC(value,precision)`

Arguments

This function has these arguments:

Argument	Description
value	Number to truncate
precision	Integer representing the precision; if greater than zero, truncates to the specified number of decimal places; if zero (or not specified), truncate to the nearest whole number; if less than zero, rounds the value left of the decimal to the nearest order of tens

Remarks

The TRUNC and INT functions are similar in that both can return integers. Use the TRUNC function to remove the decimal portion of the number; the TRUNC function does not round up or down. Use the INT function to round numbers down to the nearest integer based decimal portion of the number.

These functions differ also when using negative numbers: TRUNC(–4.2, 0) returns –4, but INT(–4.2) returns –5 because –5 is the lower number.

Data Types

Accepts numeric data for both arguments. Returns numeric data.

Examples

`TRUNC(B16)`
`TRUNC(R16C2)`
`TRUNC(5.745)` gives the result `5`
`TRUNC(-5.745)` gives the result `-5`
`TRUNC(5.745,2)` gives the result `5.74`
`TRUNC(PI())` gives the result `3`

Version Available

This function is available in product version 1.0 or later.

See Also

CEILING | EVEN | FLOOR | INT | Math and Trigonometry Functions

TTEST

Summary

This function returns the probability associated with a t-test.

Syntax

`TTEST(`*`array1`*`,`*`array2`*`,`*`tails`*`,`*`type`*`)`

Arguments

This function has these arguments:

Argument	Description
array1	Array of values in first data set
array2	Array of values in second data set
tails	Number of tails
type	Type of t-test to perform (1, 2, or 3)

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`TTEST(A1:A17,B1:B17,4,3)`
`TTEST({2,2,2,3,4},{2,3,3,4,5},1,2)` gives the result `0.126036`

Version Available

This function is available in product version 1.0 or later.

See Also

FTEST | TDIST | TINV | ZTEST | Statistical Functions

TYPE

Summary

This function returns the type of value.

Syntax

`TYPE(`*`value`*`)`

Arguments

The argument is any value as summarized here:

Type of Value	Returned Number
Number	1
DateTime object	1
TimeSpan object	1
Text	2
Logical value	4
Error value	16
Array	64

Data Types

Accepts many types of data. Returns numeric data.

Examples

`TYPE(G15)`
`TYPE(R15C7)`
`TYPE(154)` gives the result `1`
`TYPE("String")` gives the result `2`
`TYPE(TRUE)` gives the result `4`

Version Available

This function is available in product version 1.0 or later.

See Also

ERRORTYPE | ISERROR | ISLOGICAL | ISNUMBER | ISTEXT | Information Functions

UPPER

Summary

This function converts text to uppercase letters.

Syntax

`UPPER(`*`string`*`)`

Arguments

The argument is the text you want to convert to uppercase. The argument may be a string, a reference to a cell containing a string, or a formula that returns a string.

Remarks

This function does not change characters in value that are not letters.

Data Types

Accepts string data. Returns string data.

Examples

`UPPER(G15)`
`UPPER(R15C7)`
`UPPER("Report")` gives the result `REPORT`
`UPPER(""summary"")` gives the result `"SUMMARY"`

Version Available

This function is available in product version 1.0 or later.

See Also

PROPER | LOWER | T | Text Functions

VALUE

Summary

This function converts a text string that is a number to a numeric value.

Syntax

`VALUE(`*`text`*`)`

Arguments

This function has these arguments:

Argument	Description
text	Number in quotation marks or a reference to a cell with the text.

Remarks

The text can be in number, date, or time format. If the text is not in the correct format, a #VALUE! error is returned.

Data Types

Accepts string data. Returns numeric data.

Examples

`VALUE("$9,000")` gives the result `9000`

Version Available

This function is available in product version 3.0 or later.

See Also

DOLLAR | DOLLARFR | FIXED | Text Functions

VAR

Summary

This function returns the variance based on a sample of a population, which uses only numeric values.

Syntax

```
VAR(value1,value2,...)
VAR(array)
VAR(array1,array2,...)
```

Arguments

Each argument can be a double-precision floating-point value, an integer value, or an array (cell range) of these. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Remarks

The variance returns how spread out a set of data is.

This function uses the following equation to calculate the variance, where *n* is the number of values.

$$VAR(x_n) = \frac{n\sum x^2 - (\sum x)^2}{n(n-1)}$$

where x is the value and n is the number of values.

This function assumes that its arguments are a sample of the population. If your data represents the entire population, then compute the variance using the VARP function.

This function differs from VARA, which accepts text and logical values as well as numeric values.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`VAR(B3,C4,B2,D10,E5)`
`VAR(A1:A9)`
`VAR(R1C2,100,R2C5,102)`
`VAR(R1C1:R9C1)`
`VAR(R1C1:R1C9)`
`VAR(98,85,76,87,92,89,90)` gives the result `45.8095238095`

Version Available

This function is available in product version 1.0 or later.

See Also

AVERAGE | COVAR |VARP | VARA | Statistical Functions

VARA

Summary

This function returns the variance based on a sample of a population, which includes numeric, logical, or text values.

Syntax

```
VARA(value1,value2,...)
VARA(array)
VARA(array1,array2,...)
```

Remarks

Each argument can be a double-precision floating-point value, an integer value, text, a logical value, or an array (cell range) of these. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Remarks

The variance returns how spread out a set of data is.

This function uses the following equation to calculate the variance, where *n* is the number of values.

$$VARA(x_n) = \frac{n\sum x^2 - (\sum x)^2}{n(n-1)}$$

where x is the value and n is the number of values.

This function assumes that its arguments are a sample of the population. If your data represents the entire population, then compute the variance using the VARPA function.

This function differs from VARA because it accepts text and logical values as well as numeric values.

Data Types

Accepts numeric, logical, and text data for all arguments. Returns numeric data.

Examples

```
VARA(B3,C4,B2,D10,E5)
VARA(A1:A9)
VARA(R1C2,100,R2C5,102)
VARA(R1C1:R9C1)
VARA(R1C1:R1C9)
```

`VARA(98,85,76,87,92,89,90)` gives the result `45.8095238095`

Version Available

This function is available in product version 2.0 or later.

See Also

AVERAGEA | VAR | VARP | Statistical Functions

VARP

Summary

This function returns variance based on the entire population, which uses only numeric values.

Syntax

```
VARP(value1,value2,...)
VARP(array)
VARP(array1,array2,...)
```

Arguments

Each argument can be a double-precision floating-point value, an integer value, or an array (cell range) of these. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Remarks

The variance returns how spread out a set of data is.

This function uses the following equation to calculate the variance, where *n* is the number of values.

$$VARP(x_n) = \frac{n\sum x^2 - (\sum x)^2}{n^2}$$

where x is the value and n is the number of values.

This function assumes that its arguments are the entire population. If your data represents only a sample of the population, then compute the variance using the VAR function.

This function differs from VARPA, which accepts logical or text values as well as numeric values.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`VARP(B3,C4,B2,D10,E5)`
`VARP(A1:A9)`
`VARP(R1C2,100,R2C5,102)`
`VARP(98,85,76,87,92,89,90)` gives the result `39.2653061224`

Version Available

This function is available in product version 1.0 or later.

See Also

AVERAGE | VAR | VARPA | Statistical Functions

VARPA

Summary

This function returns variance based on the entire population, which includes numeric, logical, or text values.

Syntax

```
VARPA(value1,value2,...)
VARPA(array)
VARPA(array1,array2,...)
```

Arguments

Each argument can be a double-precision floating-point value, an integer value, text, a logical value, or an array (cell range) of these. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

Remarks

The variance returns how spread out a set of data is.

Each argument can be a double-precision floating-point value, an integer value, text, a logical value, or an array (cell range) of these. Up to 255 arguments may be included. You can use a single array (cell range) instead of a list of values. You can use multiple arrays (cell ranges) as well.

This function uses the following equation to calculate the variance, where *n* is the number of values.

$$VARPA(x_n) = \frac{n\sum x^2 - (\sum x)^2}{n^2}$$

where x is the value and n is the number of values.

This function assumes that its arguments are the entire population. If your data represents only a sample of the population, then compute the variance using the VARA function.

This function differs from VARP because it accepts logical and text values as well as numeric values.

Data Types

Accepts numeric, logical, and text data for all arguments. Returns numeric data.

Examples

`VARPA(B3,C4,B2,D10,E5)`
`VARPA(A1:A9)`
`VARPA(R1C2,100,R2C5,102)`
`VARPA(98,85,76,87,92,89,90)` gives the result `39.2653061224`

Version Available

This function is available in product version 2.0 or later.

See Also

AVERAGEA | VARA | VARP | Statistical Functions

VDB

Summary

This function returns the depreciation of an asset for any period you specify using the variable declining balance method.

Syntax

`VDB(cost,salvage,life,start,end,factor,switchnot)`

Arguments

This function has these arguments:

Argument	Description
cost	Initial cost of the asset
salvage	Value at the end of the depreciation period
life	Number of periods over which the asset is being depreciated
start	Number representing the starting period for which to calculate the depreciation in the same units as *life*; if not an integer, the number is truncated
end	Number representing the ending period for which to calculate the depreciation in the same units as *life*; if not an integer, the number is truncated
factor	[Optional] Rate at which the balance declines; if omitted, uses two (2)
switchnot	[Optional] Logical value specifying whether to switch to straight-line depreciation when depreciation is greater than the declining balance calculation; if omitted uses FALSE

Remarks

If *factor* is omitted, the calculation uses two, which represents the double-declining balance method. For other methods, use a different value. For more information about the double-declining balance method, see DDB.

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`VDBD(B1,1000,10,1,8)`
`VDB(50000,500,1200,100,1000,1)` gives the result `$37,122.94`

Version Available

This function is available in product version 1.0 or later.

See Also

DB | DDB | SLN | SYD | Financial Functions

VLOOKUP

Summary

This function searches for a value in the leftmost column and returns a value in the same row from a column you specify.

Syntax

`VLOOKUP(`*`value`*`,`*`array`*`,`*`colindex`*`,`*`approx`*`)`

Arguments

This function has these arguments:

Argument	Description
value	Value for which to search
array	Array or cell range that contains the data to search
colindex	Column number in the array from which the matching value is returned
approx	[Optional] Logical value indicating whether to find an approximate match; if omitted, uses TRUE and finds an approximate match

Remarks

If *approx* is FALSE, it finds an exact match, not an approximate match. If it cannot find one, it returns an #N/A error value.

If *approx* is TRUE or omitted, and the *value* cannot be found, then the largest value that is less than the *value* is used.

This function is similar to HLOOKUP except that it searches vertically (by column), instead of by row (horizontally).

Data Types

Accepts numeric or string data. Returns numeric data.

Examples

```
VLOOKUP(2,A1:D10,3)
```

Version Available

This function is available in product version 2.0 or later.

See Also

HLOOKUP | LOOKUP | Lookup Functions

WEEKDAY

Summary

This function returns the number corresponding to the day of the week for a specified date.

Syntax

`WEEKDAY(date,type)`

Arguments

This function has these arguments:

Argument	Description	
date	Date for which you want to determine the day of the week provided	
type	[Optional] Number that represents the numbering scheme for the returned weekday value; can be any of:	
	Value	Number returned
	1 or omitted	Numbers 1 (Sunday) through 7 (Saturday)
	2	Numbers 1 (Monday) through 7 (Sunday)
	3	Numbers 0 (Monday) through 6 (Sunday)

Specify the date argument as a number (as in 37806.5) a string (as in "7/4/2003 12:00"), or a DateTime object, as in DATE(2003,7,4). For more details on the date inputs, refer to the discussion in *Date and Time Functions* on page 10.

Remarks

The returned day of the week is given as an integer, ranging from 0 to 6 or 1 to 7, depending on the setting of the *type* argument.

Data Types

Accepts numeric, string, or DateTime object for both arguments. Returns numeric data.

Examples

`WEEKDAY(A2)`
`WEEKDAY(R2C1)`
`WEEKDAY(36828)` gives the result `1` equivalent to Sunday
`WEEKDAY(46,2)` gives the result `3`

Version Available

This function is available in product version 1.0 or later.

See Also

DATE | DAY | MONTH | WEEKNUM | WORKDAY | Date and Time Functions

WEEKNUM

Summary

This function returns a number that indicates the week of the year numerically.

Syntax

`WEEKNUM(`*`date`*`,`*`weektype`*`)`

Arguments

This function has these arguments:

Argument	Description	
date	Date for which you want to determine the number of week	
weektype	Type of week determined by on which day the week starts	
	Value	Number returned
	1 (assumed if omitted)	Week starts on a Sunday
	2	Week starts on a Monday

Specify the date argument as a number (as in 37806.5) a string (as in "7/4/2003 12:00"), or a DateTime object, as in DATE(2003,7,4). For more details on the date inputs, refer to the discussion in *Date and Time Functions* on page 10.

Data Types

Accepts numeric, string, DateTime object, or TimeSpan object data. Returns numeric data.

Examples

`WEEKNUM(A2)`
`WEEKNUM(R2C1,2)`
`WEEKNUM(23,1)` gives the result `4`

Version Available

This function is available in product version 1.0 or later.

See Also

MONTH | WEEKDAY | Date and Time Functions

WEIBULL

Summary

This function returns the two-parameter Weibull distribution, often used in reliability analysis.

Syntax

`WEIBULL(x,alpha,beta,cumulative)`

Arguments

This function has these arguments:

Argument	Description
x	Value at which to evaluate the distribution
alpha	Scale parameter of the distribution, represented by alpha
beta	Shape parameter of the distribution, represented by beta
cumulative	Logical value that determines the form of the function If cumulative is TRUE, then this function returns the cumulative distribution function; if FALSE, it returns the probability mass function.

Data Types

Accepts numeric data for all arguments except cumulative, which is logical (boolean). Returns numeric data.

Examples

```
WEIBULL(3,D4,D5,FALSE)
WEIBULL(50,10,20,TRUE)
```

Version Available

This function is available in product version 1.0 or later.

See Also

BINOMDIST | Statistical Functions

WORKDAY

Summary

This function returns the number of working days before or after the starting date.

Syntax

`WORKDAY(`*`startdate`*`,`*`numdays`*`,`*`holidays`*`)`

Arguments

This function has these arguments:

Argument	Description
startdate	Date that is the starting date; a number (as in 37806.5), or a DateTime object, as in DATE(2003,7,4)
numdays	Number of non-weekend or non-holiday days before or after the starting date; days in the future are positive and days in the past are negative; if not an integer, the number is truncated
holidays	[Optional] Range of dates to exclude from the calculation; if omitted, the calculation assumes no holidays and all weekdays are workdays

Data Types

Accepts numeric, string, or DateTime object data. Returns numeric data.

Examples

```
WORKDAY(A2,A4)
WORKDAY(R2C1,R5C5)
WORKDAY(A1,A2,A5:A7)
```

Version Available

This function is available in product version 2.0 or later.

See Also

DATE | NETWORKDAYS | MONTH | Date and Time Functions

XIRR

Summary

This function calculates the internal rate of return for a schedule of cash flows that may not be periodic.

Syntax

`XIRR(`*`values`*`,`*`dates`*`,`*`guess`*`)`

Arguments

This function has these arguments:

Argument	Description
values	Series of cash flows that correspond to a schedule of payments in dates. The first payment is optional and corresponds to a cost or payment that occurs at the beginning of the investment
dates	Schedule of payment dates that corresponds to the cash flow payments in *values*
guess	[Optional] Estimate of the internal rate of return that you guess is close to the result of this function; if omitted, the calculation uses 0.1 (10 percent)

Remarks

For a schedule of cash flows that is periodic, use IRR.

Data Types

Accepts numeric data for *values* and *guess*, DateTime object data for *dates*.
Returns numeric data.

Examples

```
XIRR(B2:B6,C2:C6,0.2)
```

Version Available

This function is available in product version 2.0 or later.

See Also

IRR | XNPV | MIRR | Financial Functions

XNPV

Summary

This function calculates the net present value for a schedule of cash flows that may not be periodic.

Syntax

`XNPV(`*`rate`*`,`*`values`*`,`*`dates`*`)`

Arguments

This function has these arguments:

Argument	Description
rate	Discount rate to apply to the cash flows
values	Series of cash flows that correspond to a schedule of payments in dates. The first payment is optional and corresponds to a cost or payment that occurs at the beginning of the investment
dates	Schedule of payment dates that corresponds to the cash flow payments in *values*

Data Types

Accepts numeric data for *rate* and *values*, and DateTime object data for *dates*.
Returns numeric data.

Examples

```
XNPV(0.09,B2:B6,C2:C6)
```

Version Available

This function is available in product version 2.0 or later.

See Also

IRR | NPV | MIRR | XIRR | Financial Functions

YEAR

Summary

This function returns the year as an integer for a specified date.

Syntax

`YEAR(`*`date`*`)`

Arguments

Specify the date argument as a number (as in 37806.5) a string (as in "7/4/2003 12:00"), or a DateTime object, as in DATE(2003,7,4). For more details on the date inputs, refer to the discussion in *Date and Time Functions* on page 10.

Remarks

The Spread control correctly treats the year 1900 as a non-leap year and uses a base date of 12/31/1899.

Data Types

Accepts numeric, string, DateTime object, or TimeSpan object data. Returns numeric data.

Examples

`YEAR(A2)`
`YEAR(R2C1)`
`YEAR(0.007)` gives the result (which may be different from Excel) `1899`
`YEAR(DATE(2004,8,9))` gives the result `2004`
`YEAR(38208)` gives the result `2004`
`YEAR("8/9/2004")` gives the result `2004`

Version Available

This function is available in product version 1.0 or later.

See Also

DATE | MONTH | TODAY | YEARFRAC | Date and Time Functions

YEARFRAC

Summary

This function returns the fraction of the year represented by the number of whole days between the start and end dates.

Syntax

`YEARFRAC(startdate,enddate,basis)`

Arguments

This function has these arguments:

Argument	Description
startdate	Starting date (DateTime object)
enddate	Ending date (DateTime object)
basis	[Optional] Integer representing the basis for day count (Refer to *Day Count Basis* on page 13.)

Remarks

This functions returns an error when *start*, *end*, or *basis* is invalid.

Data Types

Accepts numeric, string, DateTime object data for the date arguments and numeric data for the optional argument. Returns numeric data.

Examples

`YEARFRAC(A1,A2,A3)`

Version Available

This function is available in product version 2.0 or later.

See Also

DATE | MONTH | TODAY | YEAR | Date and Time Functions

YIELD

Summary

This function calculates the yield on a security that pays periodic interest.

Syntax

`YIELD(`*`settle`*`,`*`maturity`*`,`*`rate`*`,`*`price`*`,`*`redeem`*`,`*`frequency`*`,`*`basis`*`)`

Arguments

This function has these arguments:

Argument	Description
settle	Settlement date for the security
maturity	Maturity date for the security
rate	Annual coupon rate
price	Price per $100 face value for the security
redeem	Redemption value per $100 face value
frequency	Frequency of payment, number of coupon payments per year; must be 1, 2, or 4
basis	[Optional] Integer representing the basis for day count (Refer to *Day Count Basis* on page 13.)

Remarks

This functions returns an error when *settle*, *maturity*, or *basis* is invalid or *frequency* is a number other than 1, 2, or 4.

Data Types

Accepts numeric data and dates. Returns numeric data.

Examples

`YIELD(A1,A2,A3,A4,A5,A6,A7)`

Version Available

This function is available in product version 2.0 or later.

See Also

YIELDDISC | YIELDMAT | ODDFYIELD | Financial Functions

YIELDDISC

Summary

This function calculates the annual yield for a discounted security.

Syntax

`YIELDDISC(`*`settle`*`,`*`maturity`*`,`*`price`*`,`*`redeem`*`,`*`basis`*`)`

Arguments

This function has these arguments:

Argument	Description
settle	Settlement date for the security
maturity	Maturity date for the security
price	Price per $100 face value for the security
redeem	Redemption value per $100 face value
basis	[Optional] Integer representing the basis for day count (Refer to *Day Count Basis* on page 13.)

Remarks

This functions returns an error when *settle*, *maturity*, or *basis* is invalid.

Data Types

Accepts numeric data and dates. Returns numeric data.

Examples

`YIEDDISC(B1,B2,B3,B4,B5)`

Version Available

This function is available in product version 2.0 or later.

See Also

YIELD | YIELDMAT | ODDLYIELD | Financial Functions

YIELDMAT

Summary

This function calculates the annual yield of a security that pays interest at maturity.

Syntax

`YIELDMAT(settle,maturity,issue,issrate,price,basis)`

Arguments

This function has these arguments:

Argument	Description
settle	Settlement date for the security
maturity	Maturity date for the security
issue	Issue date for the security
issrate	Interest rate for the security at the date of issue
price	Price per $100 face value for the security
basis	[Optional] Integer representing the basis for day count (Refer to *Day Count Basis* on page 13.)

Remarks

This functions returns an error when *settle*, *maturity, issue,* or *basis* is invalid.

Data Types

Accepts numeric and date data. Returns numeric data.

Examples

`YIELDMAT(C1,C2,C3,C4,C5,C6)`

Version Available

This function is available in product version 2.0 or later.

See Also

YIELD | YIELDDISC | PRICEMAT | Financial Functions

ZTEST

Summary

This function returns the significance value of a z-test. The z-test generates a standard score for x with respect to the set of data and returns the two-tailed probability for the normal distribution.

Syntax

`ZTEST(array,x,sigma)`

Arguments

This function has these arguments:

Argument	Description
array	Array of data to test
x	Value at which to test
sigma	[Optional] Known standard deviation for the population; if omitted, the calculation uses the sample standard deviation

Remarks

If sigma is not specified, the calculated standard deviation of the data in array is used.

The equation for calculating the z-test is as follows, where *n* is the number of data points.

$$ZTEST(array,\ x,\ \sigma) = 1 - NORMSDIST\left(\frac{\mu - x}{\sigma \div n}\right)$$

Data Types

Accepts numeric data for all arguments. Returns numeric data.

Examples

`ZTEST(A2:D12,40,0.877)`
`ZTEST(R2C1:R12C4,2)`
`ZTEST({5,10,15,12,11,8,16,7},10)` gives the result `0.355512703503418`
`ZTEST({5,10,15,12,11,8,16,7},10,3)` gives the result `0.318675944098237`

Version Available

This function is available in product version 1.0 or later.

See Also

FTEST | TTEST | Statistical Functions

Index

A

B

C

D

E

F

O

P

T