Function Reference

Formula One™ for Java®

Powerful spreadsheet application for end users.
Robust development tool for Java developers and Webmasters.
Version 7.0

Tidestone Technologies, Inc.
# Contents

<table>
<thead>
<tr>
<th>Preface</th>
<th>Overview</th>
<th>v</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>About The Function Reference</td>
<td>v</td>
</tr>
<tr>
<td></td>
<td>Using This Manual</td>
<td>vi</td>
</tr>
<tr>
<td></td>
<td>Technical Support</td>
<td>vi</td>
</tr>
<tr>
<td></td>
<td>Documentation Conventions</td>
<td>vi</td>
</tr>
<tr>
<td></td>
<td>New Functions in Version 7.0</td>
<td>vii</td>
</tr>
</tbody>
</table>

## Chapter 1

### Introduction to Functions

<table>
<thead>
<tr>
<th>What is a Function?</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function Basics</td>
<td>2</td>
</tr>
<tr>
<td>Function Arguments</td>
<td>3</td>
</tr>
<tr>
<td>Using Names</td>
<td>9</td>
</tr>
<tr>
<td>Identifying Functions on Worksheets</td>
<td>10</td>
</tr>
<tr>
<td>Nesting Functions</td>
<td>11</td>
</tr>
<tr>
<td>Creating Formulas with Functions</td>
<td>11</td>
</tr>
<tr>
<td>Array Functions</td>
<td>13</td>
</tr>
<tr>
<td>Database Functions</td>
<td>14</td>
</tr>
<tr>
<td>Financials Functions</td>
<td>18</td>
</tr>
<tr>
<td>Understanding Errors</td>
<td>19</td>
</tr>
</tbody>
</table>

## Chapter 2

### Functions by Category

<table>
<thead>
<tr>
<th>Database Functions</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date and Time Functions</td>
<td>22</td>
</tr>
<tr>
<td>Engineering Functions</td>
<td>23</td>
</tr>
<tr>
<td>Financial Functions</td>
<td>25</td>
</tr>
<tr>
<td>Information Functions</td>
<td>27</td>
</tr>
<tr>
<td>Logical Functions</td>
<td>28</td>
</tr>
<tr>
<td>Lookup and Reference Functions</td>
<td>28</td>
</tr>
<tr>
<td>Math and Trigonometry Functions</td>
<td>29</td>
</tr>
<tr>
<td>Statistical Functions</td>
<td>31</td>
</tr>
<tr>
<td>Text Functions</td>
<td>34</td>
</tr>
</tbody>
</table>

## Chapter 3

### A-Z Function Reference

| 37 |

## Index

| 271 |
Tidestone
PREFACE

Overview

Computer spreadsheets — tools that help us understand, model, and manipulate relationships between sets of numerical data — further empower us by providing the ability to manipulate, extrapolate, interpret, graph, and display numeric data with the speed and convenience of the computer. Formula One for Java unites all of the powerful utility and familiar interface of the computer spreadsheet with the speed, power, and the universal presence of the Internet.

Designed for use in Java development environments, Formula One for Java provides the tools you need to design, create, and distribute custom spreadsheets over the Internet as part of your application, applet, or JavaBean. Using Formula One for Java, you may also embed high-quality spreadsheets in a Web page with just a few lines of HTML code.

Formula One for Java also works as a standalone spreadsheet application that operates with the speed and functionality of the best desktop spreadsheet.

About The Function Reference

This Guide, the Formula One for Java Function Reference, provides:

- a general introduction to using functions in spreadsheets
- lists and short descriptions of functions categorized by types
- a complete reference record for each function.

The complete documentation for Formula One for Java consists of three separate publications:

1. Formula One for Java User’s Guide
   A basic guide to using electronic spreadsheets in general and using Formula One for Java on a desktop or over a network as an end user.

2. Formula One for Java Technical Guide
   The Technical Guide includes more advanced concepts and uses.

3. Formula One for Java Function Reference
Using This Manual

For the most part, this manual assumes you are using a Windows operating environment. Keyboard keys and some graphical user interface elements might vary depending upon your operating environment.

This document along with other useful information about Formula One for Java is also available online at www.tidestone.com.

Technical Support

The Tidestone technical support staff can help you with any problem you encounter installing or using Formula One for Java.

For specific information about support plans for Formula One for Java, visit www.tidestone.com and follow the Support links.

Documentation Conventions

Throughout this documentation, typographic conventions are used to define elements and references to Formula One for Java items.

Recognizing these conventions will help you understand and use the documentation.

<table>
<thead>
<tr>
<th>Convention example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>➤ <strong>To enter a function:</strong></td>
<td>A series of numbered instructions is preceded by an introductory line. The introductory line begins with an arrowhead.</td>
</tr>
<tr>
<td>1. Type =SUM(A4:B6)</td>
<td>Numbered instructions provide step-by-step directions for performing tasks. Perform the instructions in the order they are presented. In numbered steps, items you are to enter are shown in Letter Gothic font.</td>
</tr>
<tr>
<td>Workbook</td>
<td>In general sections, italic text is used for the first occurrence of a new term.</td>
</tr>
<tr>
<td>Fontname</td>
<td>In reference sections, italic text indicates variable or argument information you must supply.</td>
</tr>
<tr>
<td>=(3+4)*5</td>
<td>Letter Gothic font is used for examples of cell entries.</td>
</tr>
<tr>
<td>F1J7Swing.jar</td>
<td>File names are presented in bold type.</td>
</tr>
<tr>
<td>Format &gt; Sheet &gt; Properties</td>
<td>Choose the Properties option on the Sheet submenu of the Format menu.</td>
</tr>
<tr>
<td>CTRL + L</td>
<td>Type L while holding down the CTRL key.</td>
</tr>
</tbody>
</table>
New Functions in Version 7.0

The following functions have been added in the 7.0 version of Formula One for Java.

<table>
<thead>
<tr>
<th>Function</th>
<th>New Function</th>
<th>New Function</th>
<th>New Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCRINT</td>
<td>ACCRINTM</td>
<td>AMORDEGRC</td>
<td>AMORLINC</td>
</tr>
<tr>
<td>AREAS</td>
<td>AVEDEV</td>
<td>AVERAGEA</td>
<td>BESELI</td>
</tr>
<tr>
<td>BESSEJ</td>
<td>BESSELK</td>
<td>BESELY</td>
<td>BETADIST</td>
</tr>
<tr>
<td>BETAINV</td>
<td>BIN2DEC</td>
<td>BIN2HEX</td>
<td>BIN2OCT</td>
</tr>
<tr>
<td>BINOMDIST</td>
<td>CALL</td>
<td>CELL</td>
<td>CHIDIST</td>
</tr>
<tr>
<td>CHIINV</td>
<td>CHITEST</td>
<td>COMBIN</td>
<td>COMPLEX</td>
</tr>
<tr>
<td>CONFIDENCE</td>
<td>CONVERT</td>
<td>CORREL</td>
<td>COUNTBLANK</td>
</tr>
<tr>
<td>COUPDAYBS</td>
<td>COUPDAYS</td>
<td>COUPDAYSNC</td>
<td>COUPNCD</td>
</tr>
<tr>
<td>COUPNUM</td>
<td>COUPPCD</td>
<td>COVAR</td>
<td>CRITBINOM</td>
</tr>
<tr>
<td>CUMIPMT</td>
<td>CUMPRINC</td>
<td>DAVEGERAGE</td>
<td>DCOUNT</td>
</tr>
<tr>
<td>DCOUNTA</td>
<td>DEC2BIN</td>
<td>DEC2HEX</td>
<td>DEC2OCT</td>
</tr>
<tr>
<td>DEGREES</td>
<td>DELTA</td>
<td>DEVSQ</td>
<td>DGET</td>
</tr>
<tr>
<td>DISC</td>
<td>DMAX</td>
<td>DMIN</td>
<td>DOLLARDE</td>
</tr>
<tr>
<td>DOLLARFR</td>
<td>DPRODUCT</td>
<td>DSTDEV</td>
<td>DSTDEVP</td>
</tr>
<tr>
<td>DSUM</td>
<td>DURATION</td>
<td>DVAR</td>
<td>DVARP</td>
</tr>
<tr>
<td>EDATE</td>
<td>EFFECT</td>
<td>EOMONTH</td>
<td>ERF</td>
</tr>
<tr>
<td>ERFC</td>
<td>EXPONDIST</td>
<td>FACTDOUBLE</td>
<td>FDIST</td>
</tr>
<tr>
<td>FINV</td>
<td>FISHER</td>
<td>FISHERINV</td>
<td>FORECAST</td>
</tr>
<tr>
<td>FREQUENCY</td>
<td>FTEST</td>
<td>FVSCHEDULE</td>
<td>GAMMADIST</td>
</tr>
<tr>
<td>GAMMAINV</td>
<td>GAMMALN</td>
<td>GCD</td>
<td>GEOMEAN</td>
</tr>
<tr>
<td>GESTEP</td>
<td>GETPIVOTDATA</td>
<td>GROWTH</td>
<td>HARMEAN</td>
</tr>
<tr>
<td>HEX2BIN</td>
<td>HEX2DEC</td>
<td>HEX2OCT</td>
<td>HYPERLINK</td>
</tr>
<tr>
<td>HYPERGOMDIST</td>
<td>IMABS</td>
<td>IMAGINARY</td>
<td>IMARGUMENT</td>
</tr>
<tr>
<td>IMCONJUGATE</td>
<td>IMCOS</td>
<td>IMDIV</td>
<td>IMEXP</td>
</tr>
<tr>
<td>IMLN</td>
<td>IMLOG10</td>
<td>IMLOG2</td>
<td>IMPOWER</td>
</tr>
<tr>
<td>IMPRODUCT</td>
<td>IMREAL</td>
<td>IMSIN</td>
<td>IMSQRT</td>
</tr>
<tr>
<td>IMSUB</td>
<td>IMSUM</td>
<td>INFO</td>
<td>INTERCEPT</td>
</tr>
<tr>
<td>INTRATE</td>
<td>ISEVEN</td>
<td>ISODD</td>
<td>KURT</td>
</tr>
<tr>
<td>LARGE</td>
<td>LCM</td>
<td>LINEST</td>
<td>LOGEST</td>
</tr>
<tr>
<td>LOGINV</td>
<td>LOGNORMDIST</td>
<td>MAXA</td>
<td>MDETERM</td>
</tr>
<tr>
<td>MDURATION</td>
<td>MEDIAN</td>
<td>MINA</td>
<td>MINVERSE</td>
</tr>
<tr>
<td>MULT</td>
<td>MODE</td>
<td>MROUND</td>
<td>MULTINOMIAL</td>
</tr>
<tr>
<td>NEGBINOMDIST</td>
<td>NETWORKDAYS</td>
<td>NOMINAL</td>
<td>NORMDIST</td>
</tr>
<tr>
<td>NORMINV</td>
<td>NORMSDIST</td>
<td>NORMSINV</td>
<td>OCT2BIN</td>
</tr>
<tr>
<td>OCT2DEC</td>
<td>OCT2HEX</td>
<td>ODDFPRICE</td>
<td>ODDFYIELD</td>
</tr>
<tr>
<td>Function</td>
<td>Function</td>
<td>Function</td>
<td>Function</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------</td>
<td>------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>ODDLPRI CE</td>
<td>ODDLYIELD</td>
<td>PEARSON</td>
<td>PERCENTILE</td>
</tr>
<tr>
<td>PERCENTRANK</td>
<td>PERMUT</td>
<td>POISSON</td>
<td>POWER</td>
</tr>
<tr>
<td>PRICE</td>
<td>PRICEDISC</td>
<td>PRICEMAT</td>
<td>PROB</td>
</tr>
<tr>
<td>QUARTILE</td>
<td>QUOTIENT</td>
<td>RADIANS</td>
<td>RANDBETWEEN</td>
</tr>
<tr>
<td>RANK</td>
<td>RECEIVED</td>
<td>REGISTER.ID</td>
<td>ROMAN</td>
</tr>
<tr>
<td>RSQ</td>
<td>SERIESUM</td>
<td>SKEW</td>
<td>SLOPE</td>
</tr>
<tr>
<td>SMALL</td>
<td>SQLREQUEST</td>
<td>SQRTPI</td>
<td>STANDARDIZE</td>
</tr>
<tr>
<td>STDEVA</td>
<td>STDEVPA</td>
<td>STEYX</td>
<td>SUBTOTAL</td>
</tr>
<tr>
<td>SUMPRODUCT</td>
<td>SUMX2MY2</td>
<td>SUMX2PY2</td>
<td>SUMXMY2</td>
</tr>
<tr>
<td>TBILLEQ</td>
<td>TBILLPRICE</td>
<td>TBILLYIELD</td>
<td>TDIST</td>
</tr>
<tr>
<td>TINV</td>
<td>TRANSPOSE</td>
<td>TREND</td>
<td>TRIMMEAN</td>
</tr>
<tr>
<td>TTEST</td>
<td>VARA</td>
<td>VARPA</td>
<td>WEEKNUM</td>
</tr>
<tr>
<td>WEIBULL</td>
<td>WORKDAY</td>
<td>XIRR</td>
<td>XNPV</td>
</tr>
<tr>
<td>YEARFRAC</td>
<td>YIELD</td>
<td>YIELDDISC</td>
<td>YIELDMAT</td>
</tr>
<tr>
<td>ZTEST</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CHAPTER 1

Introduction to Functions

Formula One for Java’s functions allow you to use complex equations within spreadsheets without having to enter the equations themselves. This chapter provides an overview of how to enter and use functions. The following topics are covered.

- “What is a Function?” on page 2
- “Function Basics” on page 2
- “Function Arguments” on page 3
- “Using Names” on page 9
- “Identifying Functions on Worksheets” on page 10
- “Nesting Functions” on page 11
- “Creating Formulas with Functions” on page 11
- “Array Functions” on page 13
- “Database Functions” on page 14
- “Financials Functions” on page 18
- “Understanding Errors” on page 19
What is a Function?

A function is a string containing numbers, operators, cell references, and names. Functions calculate and evaluate data and return the result to the cell in which they are entered. Functions are a special kind of formula and follow the rules for entering and working with formulas.

Each function performs a specific calculation. For example, the SQRT function calculates the square root of a number. The following example calculates the square root of 118:

=SQRT(118)

Function Basics

Functions are composed of three elements: an equal sign, a keyword, and arguments.

- The equal sign tells Formula One for Java that this cell contains a formula or function.
- The keyword identifies the function so that the worksheet knows what type of calculation or evaluation to perform. Each function keyword is unique.
- Arguments provide the data for the function to calculate or evaluate. The arguments for a function immediately follow the function keyword and are enclosed in parentheses. Separate multiple arguments with commas.

Every function requires an equal sign and a keyword, but not all functions require arguments.

Functions can contain as many as 1024 characters.

To enter a function:

1. Select the cell where you want to display the function’s result.
2. Type the equals sign.
3. Type the function keyword in lowercase or uppercase characters.
   After the function is entered, the worksheet records the function keyword in uppercase characters, regardless of how it was entered.
4. Type any arguments required by the function, enclosed in parentheses and separated by commas.
   Functions that do not require arguments still require a set of empty parentheses () after the function keyword.
   Some functions contain optional arguments. If you omit an optional argument, a default value is assumed for the argument.
5. Press ENTER.

The result of the function calculation will appear in the cell.
If Formula One for Java finds a syntax error in the function you entered, it will display an error message. You must correct the error or delete the function before proceeding. For a list of syntax errors, see “Understanding Errors” on page 19.

**Function Arguments**

A function is a special kind of formula, so a function’s arguments must follow the rules for entering formulas. The arguments for a function can be:

- Numbers
- Text strings
- Logical values
- Error values
- Cell or range references
- Array constants

A function may require its first argument to be a number, the second argument a text string, and the third argument a cell reference. Some arguments can be of more than one type; for example, the SUM function’s argument can be a set of numbers, range references, or array constants. Refer to the documentation on the individual function to determine the type of data required for the function you are entering.

**Numbers in Arguments**

Usually you can enter a number in an argument by simply typing the number. Do not use commas to separate thousands, because the comma will be interpreted as separating two arguments.

**Negative numbers** are entered using the negative sign (-). Do not use parentheses: the function will interpret the value as a different argument.

**Fractions** can be entered by using a slash for the fraction separator (3/4 is evaluated as .75).

**Percentages** can be entered using the percent sign (34% is evaluated as .34).

**Scientific notation** can be used for very large or very small numbers (34E+09 is evaluated as 34,000,000,000).

**Exponents** can be entered using the caret sign (3^4 is evaluated as three to the fourth power, or 81).

**Dates and times** may be entered in two ways: as serial numbers, or in one of the conventional formats (e.g., mm/dd/yy) enclosed in quotation marks. Dates and times entered in this way are considered text, but Formula One for Java recognizes them as dates and internally converts them to their serial number values. For example, "10/10/94" - "10/1/94" is interpreted as 9.
Text Strings in Arguments

Text strings in an argument list must be enclosed in quotation marks. Without the quotation marks, the function will return the #NAME? error. When entering multiple text arguments, be sure the closing quotation mark comes before the comma that separates two arguments.

When the argument is a cell reference pointing to a cell containing a text string, the text in the cell does not have to be enclosed in quotes.

If a number is encountered when text is expected, the number is converted to text. "The number is " & 3 is interpreted as The number is 3. If text is encountered when a number is expected, the text is converted to a number (1 + "3" is interpreted as 4). If the text cannot be converted to a valid number (e.g., 1 + "Text"), the function will return the #VALUE! error.

Concatenation

You may use the ampersand (&) character to concatenate text strings when a text argument from two different sources is required. For example, in the spreadsheet below, cell A3 uses concatenation to create the text first quarter.

 Kagame uses the ampersand to concatenate text in cell A1 with other text.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>first</td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>=A1 &amp; &quot; quarter&quot;</td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

Logical Values in Arguments

The logical value TRUE converts to 1, while FALSE converts to 0.

If a number is encountered when a logical value is expected, 0 is evaluated as FALSE and all other numbers are evaluated as TRUE. If text is encountered when a logical value is expected, "TRUE" is evaluated as TRUE, "FALSE" is evaluated as FALSE, and all other text returns the #VALUE! error.

Cell References in Arguments

For most arguments, you can substitute a cell or range reference for the data required by an argument. A reference is a cell’s address. It identifies a cell or range of cells by referring to the column letter and row number of the cell(s). For example, A1 refers to the cell at the intersection of column A and row 1.

The reference tells Formula One for Java to use the contents of the referenced cell(s) as the function’s argument. For example, if an argument requires a number, you can substitute a reference to a cell that contains a number. The number in the referenced cell is used in the calculation of the function. The referenced cell must contain the appropriate data for the argument that uses it.
You specify a range of cells by placing a colon (:) between two cell references. For example, \( A1:C3 \) refers to the range anchored by cells A1 and C3. The range includes all cells in columns A, B, and C of rows 1, 2, and 3.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td>8</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>5</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>6</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>9</td>
<td>8</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

The range reference \( A1:C3 \) refers to all the grayed cells in this worksheet.

Some functions may take more than one cell as an argument. For example, when the AVERAGE function has a range reference as its argument, the function averages the data in all the cells in the range.

**Empty Cells**

Most functions ignore any empty cells found in a range referenced in an argument. However, cells that appear empty and are not may affect the results of the function. For example, cells containing empty text or text consisting only of spaces may be treated as text. Cells that are formatted not to display zero values may contain hidden zeroes that will be treated as numeric values. If your function displays unexpected results, check for empty cells and cells that appear empty.

**Entering Cell References**

You can enter cell references in arguments in three ways:

- Type in the cell or range address.
- Type in the name of a named cell or range. For information on defining names, see “Using Names” on page 9.
- Use the mouse to click and drag on cells and ranges. Formula One for Java automatically enters a relative reference identifying the cell(s) you select.

**Absolute and Relative References**

There are two types of cell references: relative and absolute.

- **Relative references** point to a cell based on its position relative to the current cell. When the cell containing the reference is moved or copied, the reference is adjusted to point to a new cell with the same relative offset as the originally referenced cell.

  For example, suppose the function \( \text{SUM}(B1:B3) \) is located in cell A1. When you copy the function and paste it down two rows into cell A3, the function will be adjusted down two rows, to \( \text{SUM}(B3:B5) \).

  Relative references will be adjusted whenever you cut or copy and paste a function or when you use the Edit > Fill commands to fill a range with a copy of a function.
Absolute references point to a cell at an exact location. When a cell containing a formula with absolute references is moved or copied, the reference does not change. Absolute references have a dollar sign ($) in front of the row number and/or column letter.

References that are part absolute and part relative are called mixed references. The following table lists the reference types.

<table>
<thead>
<tr>
<th>Reference</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$1</td>
<td>Absolute reference pointing to cell A1.</td>
</tr>
<tr>
<td>$A$1</td>
<td>Absolute column reference, relative row reference pointing to cell A1.</td>
</tr>
</tbody>
</table>

Using Relative and Absolute References

You can copy and paste absolute, relative, and mixed references to create worksheets that are easy to update and that are smaller than worksheets where each formula is created separately.

For example, in the following worksheet, the values in column A need to be multiplied by the percentages in row 1.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td>10%</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>8000</td>
<td></td>
<td></td>
<td>16%</td>
</tr>
<tr>
<td>4</td>
<td>10000</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

To do these calculations, you could enter the function `PRODUCT(A2,B1)` in cell B2, `PRODUCT(A3,B1)` in cell B3, `PRODUCT(A4,B1)` in cell B4, etc. Besides the fact that this would be a lot of typing, this solution would require Formula One for Java to keep nine separate formulas in memory.

A better way to do it would be to enter the function `PRODUCT($A2,$B1)` in cell B2, and use the worksheet's Edit > Fill command to fill cells B2 through D4 with copies of that function. When the function is copied in this manner, its relative references change, but the absolute references stay the same.
Chapter 1  Introduction to Functions

The resulting worksheet calculates all the figures using multiple copies of that one function. Only one function must be kept in memory.

References to Other Worksheets

You can reference cells in other worksheets in the same workbook by placing an exclamation mark between the sheet name and the reference. The sheet name is the name found on the worksheet tab. For example, Data!A1 refers to the top left cell in a sheet called Data.

Sheet names with spaces. If the sheet name contains spaces, you must enclose the name in single quotes: ‘1994 sales’!B17.

Cells on two worksheets. You can make a reference to cells on two or more different worksheets by placing a colon between the two sheet names. For example, Sheet1:Sheet2!A1 refers to two cells: cell A1 in Sheet1 and cell A1 in Sheet2.

Order of sheet names. References to more than one worksheet must list the worksheets in the order in which they appear in the workbook.
References to Other Workbooks

References that point to cells on worksheets in other workbooks are called external references.

An external reference is created by placing the workbook name in brackets, followed by the worksheet name, an exclamation point, and finally a cell or range reference.

```
[Sales]1998!A1
```

External references will work only if both workbooks are open in the workbook designer. If the referenced workbook is not open when you create the external reference, an Invalid Formula Syntax error message will appear.

Developers who want to use external references when they are not using the Workbook Designer control must create a group of workbooks using the setGroup method in the View class. All workbooks that refer to each other must be added to the same group in order for the external references to work.

The following are examples of external references using absolute, relative, and mixed references.

<table>
<thead>
<tr>
<th>Reference</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Q1]Sheet1:Sheet2!$A$1</td>
<td>Absolute column reference, relative row reference pointing to cell A1 in the first and second worksheets of a workbook titled Q1.</td>
</tr>
<tr>
<td>[Store1]Sheet1:Sheet4!A1:F1</td>
<td>Relative row and column reference pointing to the range A1 to F1 in the first four worksheets of a workbook titled Store1.</td>
</tr>
</tbody>
</table>
Paths in External References

After you enter an external reference, Formula One for Java will change the format of the reference to show the absolute path to the workbook you referenced. For example, say you entered this reference to a workbook named September in the Payroll directory on your C drive:

[September]Payroll!C2:C420

After you enter that reference, if you return to the cell where the reference was entered, you will note that Formula One for Java has changed it to:

'[C:\Payroll\September.vts]Payroll'!C2:C420

This absolute path is recorded in the worksheet. If you later move the September workbook, the external reference should still work, as long as you open September.vts in the Workbook Designer at the same time as the workbook that references it.

Array Constants in Arguments

For many arguments, you can substitute an array constant for the data required. An array constant is a list of numbers enclosed in curly brackets { }. The function treats each item in an array constant as an individual bit of data, just as it treats the data in each cell in a range reference argument individually.

You can enter an array constant that functions like a range reference, with cells and columns. Use commas to separate individual cells on the same row; use semicolons to separate rows.

For example, the array constant {2,4,6;8;10,12,14,16;18,20,22,24} is equivalent to this range reference:

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>12</td>
<td>14</td>
<td>16</td>
</tr>
<tr>
<td>18</td>
<td>20</td>
<td>22</td>
<td>24</td>
</tr>
</tbody>
</table>

You cannot use array constants as arguments in database functions.

Using Names

User-defined names are an easy way to identify a cell, a range of cells, a value, or a formula. For example, the formula =Sales-Expenses is much clearer than =A10-A6. When you create a name for a range of cells, you can use that name (without quotation marks) in formulas.

You can also use names to identify constants and formula expressions. For example, you might define the name LtSp as 186000. You could then use the name LtSp in all your formulas.
To define names:
1. If you are naming a range, select it.
2. Select Insert > Name. The Defined Name dialog, shown below, will appear.
3. When you are finished, click OK.
Do not use the names Print_Area or Print_Titles. Formula One for Java reserves those names for its own use.

Identifying Functions on Worksheets
Formula One for Java gives you two tools to help you identify cells that contain function formulas: type markers and view options.

Type Markers
Type markers are colored borders around cells that identify the type of data in the cell. By default, type markers are not displayed.

<table>
<thead>
<tr>
<th>Type marker color</th>
<th>Data in cell</th>
</tr>
</thead>
<tbody>
<tr>
<td>Green</td>
<td>Values</td>
</tr>
<tr>
<td>Red</td>
<td>Formulas and functions</td>
</tr>
<tr>
<td>Blue</td>
<td>Empty, formatted cells</td>
</tr>
</tbody>
</table>

To display type markers:
1. Choose Tools > Options and click the General tab, if necessary.
2. Check the Type Markers box, then click OK.

View Options
By default, the workbook designer displays the results of function calculations in the cells where functions are entered. You may choose to display the function formula instead.
➤ **To display function formulas instead of the results of the functions:**

1. Select the worksheet(s) on which you want to see function formulas.
2. Choose Format > Sheet > Properties and click the View tab, if necessary.
3. Check the Formulas box, then click OK.

When you choose to view function formulas, Formula One for Java automatically doubles the column widths of the worksheet(s) to provide space for the formulas. It also left-aligns all cells on that worksheet and displays all numbers in the General format. Returning to normal view will shrink the columns back to their original size and return the alignment and formatting to their original settings.

**Nesting Functions**

When you use a function as an argument for another function, you are *nesting* functions. The nested function must return the appropriate type of data for the function in which it is nested. You must also provide the necessary arguments for the nested function.

In the following example, the AVERAGE function is used as an argument for the SUM function. In this case, AVERAGE is nested in SUM.

```
=SUM(5.23, 6.82, AVERAGE(2.45, 5.62, 7.74), 8.95, 9.01)
```

**Creating Formulas with Functions**

You may use functions with numbers, cell references, names, and mathematical operators to create complex formulas.

The following operators let you specify the type of calculation or evaluation to be performed in the formula.

<table>
<thead>
<tr>
<th>Operator Type</th>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arithmetic</td>
<td>+</td>
<td>Addition</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>Subtraction</td>
</tr>
<tr>
<td></td>
<td>/</td>
<td>Division</td>
</tr>
<tr>
<td></td>
<td>*</td>
<td>Multiplication</td>
</tr>
<tr>
<td></td>
<td>%</td>
<td>Percentage</td>
</tr>
<tr>
<td></td>
<td>^</td>
<td>Exponentiation</td>
</tr>
<tr>
<td>Text</td>
<td>&amp;</td>
<td>Concatenation</td>
</tr>
</tbody>
</table>
When more than one of the operators appears in a formula, Formula One for Java uses a specific order of precedence to calculate the formula. The operators listed first in the following table are evaluated before the operators below them.

<table>
<thead>
<tr>
<th>Operator Type</th>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comparison</td>
<td>=</td>
<td>Equal to</td>
</tr>
<tr>
<td></td>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td></td>
<td>&lt;</td>
<td>Less than</td>
</tr>
<tr>
<td></td>
<td>&gt;=</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td></td>
<td>&lt;=</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td></td>
<td>&lt;&gt;</td>
<td>Not equal to</td>
</tr>
<tr>
<td>Reference</td>
<td>;</td>
<td>In function arguments: Decimal.</td>
</tr>
<tr>
<td></td>
<td>.., .</td>
<td>In range references: Range. Produces a reference that includes all the cells between the two references (e.g., A1:A5 includes cells A1, A2, A3, A4, and A5).</td>
</tr>
<tr>
<td></td>
<td>,</td>
<td>In function arguments: Argument separator.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>In range references: Union. Produces one reference that includes the two references (e.g., A1:A10,C1:C10).</td>
</tr>
</tbody>
</table>

When more than one of the operators appears in a formula, Formula One for Java evaluates them from left to right.

Use parentheses to change the order of evaluation. The following example illustrates how the result of a formula can be altered by adding parentheses to change the order of precedence.
Changing Formula Evaluation Rules

In general, Formula One for Java evaluates formulas the way Microsoft Excel does. The Lotus 1-2-3 spreadsheet program evaluates formulas in a slightly different way. Lotus 1-2-3’s rules always interpret the logical value TRUE as 1, FALSE as 0. Also, text in cells referred to by formulas and in function arguments is always evaluated as 0 (zero).

You can use Lotus 1-2-3’s evaluation rules instead of Microsoft Excel’s on any worksheet. You can even have different worksheets in the same workbook use different evaluation rules.

➤ To use Lotus 1-2-3 evaluation rules:

1. Select the worksheet(s) whose formulas you want to evaluate using Lotus 1-2-3 rules.
2. Choose Format > Sheet > Properties and click the General tab.
3. Click the Lotus Style Formula Evaluation check box.
4. Click OK.

Array Functions

Most functions return a calculated value in the same cell in which the function was entered. Array functions are a special kind of function that return data to a range of cells instead of to just one cell.

For example, the TRANSPOSE function, which takes a range of cells as its argument, returns a new range of cells in which the rows from the original range are now columns and the original columns are now rows.

When discussing array functions, we distinguish between two different types of ranges: the argument range is the range of cells that appear between the parentheses of an array function. The results range is the range of cells in which the data computed by the function will be displayed.

The following functions are array functions:

<table>
<thead>
<tr>
<th>Function</th>
<th>Formula</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>=1+2*37</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td>=(1+2)*37</td>
<td>111</td>
<td></td>
</tr>
</tbody>
</table>
Entering Array Functions

To enter an array function:
1. Select a results range. Some functions require the results range to contain a certain number of columns or rows. A too-small results range will not show all of the function results; a too-large results range will display error messages in some cells. Check the documentation on the specific function for information about the required size and shape of the results range.
2. In the results range you selected, type in the equals sign, the array function keyword, and the argument(s) or argument range(s), in parentheses.
3. To enter the function, press ENTER while holding down CONTROL and SHIFT. (If you simply press ENTER, the array function will return the #VALUE! error.)

Most normal functions can be used as array functions to return the same calculation to a range of cells. To do this, select a range, type in the function, and hold down CONTROL and SHIFT while pressing ENTER.

About Results Ranges

All the cells in the results range of an array function will contain the calculated array function. When you select one of these cells, the function displayed in the formula bar will be enclosed in curly brackets { }. Do not enter these brackets yourself.

Note Results ranges for array functions must be edited as a unit. After you enter an array function, you can change the format of individual cells in the results range, but you cannot edit the data in individual cells. If you try, Formula One for Java will display an error message. To change any of the data in a results range, you must select all the cells and change them all at once.

Database Functions

The 12 database functions let you find and perform calculations on specific pieces of data in a database. All of the database functions search a specified database for records that match specified criteria. Some of the database functions then perform calculations on data in a specified field of the matching records.

The following are the database functions.

<table>
<thead>
<tr>
<th>Database Function</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAVERAGE</td>
<td>DGET</td>
</tr>
<tr>
<td>DCOUNT</td>
<td>DPRODUCT</td>
</tr>
<tr>
<td>DCOUNTA</td>
<td>DSTDEV</td>
</tr>
<tr>
<td>DMAX</td>
<td>DSUM</td>
</tr>
<tr>
<td>DMIN</td>
<td>DVAR</td>
</tr>
<tr>
<td>DAVERAGE</td>
<td>DPRODUCT</td>
</tr>
<tr>
<td>DCOUNT</td>
<td>DSTDEV</td>
</tr>
<tr>
<td>DCOUNTA</td>
<td>DVARP</td>
</tr>
</tbody>
</table>

Database Function Example

The following worksheet showing salary, commission, and sales information for a company’s salespeople can be used as a database.
You can use the DAVERAGE database function to find the average commissions earned in 1996 by all salespeople who made less than $29,000 in salary that year. The arguments for DAVERAGE — database, field, and criteria — are shown in the illustration above. The function searches database for records that comply with criteria, then finds the average of field in the records that pass the test.

This function returns 24,497, the average commissions of the three salespeople (Brad, Denise, and James) whose salaries are less than $29,000:

\[ \text{DAVERAGE (A1:D8,"Commissions 96",B10:B11)} \]

All database functions have the same three arguments: database, field, and criteria.

**The database Argument**

*Database* is a reference to a range of cells containing data that the database function searches. Each row of data in the database is called a *record*.

The top row must contain unique text labels identifying the data in the columns. To use numbers or dates as labels, identify them as text by entering a leading apostrophe (‘). Since the labels must match the labels in *field* and *criteria*, it’s best to make them simple and descriptive, with no extra spaces. In general, names should be free of punctuation marks and other symbols.

**The field Argument**

*Field* is the column within *database* containing the data on which you want the function to perform the calculation. You may enter the column’s label enclosed in quotation marks or a number corresponding to the column’s position within the database: the first column is 1, the second column is 2, etc. You may also enter a reference to a cell that contains the column’s label or number.
The criteria Argument

*Criteria* is a reference to a range of cells, usually separate from *database*, containing search criteria. The top row contains text labels that match the labels in *database*. The cells beneath the labels contain criteria that data in the appropriate column of *database* must pass in order for that record to be selected for the function's calculation.

The criteria can be numbers, text, or blank cells. You can enter the criteria directly in the cell, or you can enter formulas and functions that calculate the criteria. You may also enter cell references.

**Note** Microsoft Excel does not support formulas or functions in database criteria. If your worksheets must be compatible with Microsoft Excel, do not use formulas or functions in your criteria.

Numbers

Numbers will be compared to the numbers in *database*. Only exact matches pass the test. An = before the number is optional.

To test for relationships between the criteria and the numbers in *database*, precede the criteria with the operators <, >, <=, >=, and <> (not equal). For example, <500 will find all numbers less than 500.

Text

Text will be compared to text in database. Only exact matches pass the test. The function does not consider the case of the letters, so the criteria DENISE will find Denise and denise. Do not use quotation marks (unless you want the database function to find matches containing the quotation marks).

To test for words close to a particular word, use the wildcard characters ? and *. The ? stands in for exactly one character; * stands for zero or more characters. You may use more than one wildcard character in a cell. Any characters after the * wildcard must be located at the end of the cell text. For example, the criteria JA*S will find JAMES and JARS.

To test for text that is alphabetized before or after the criteria, use the operators <, >, <=, and >=. For example, the criteria >BRAD will find all cells that start with text that is alphabetized after the word BRAD. To test for text that is not the same as the criteria, use the operator <> (not equal). For example, the criteria <>FRED will find all cells that do not contain just the word FRED.

Blank cells

A blank cell indicates that no tests should be performed. A blank row in the criteria range indicates all records should be selected.
Using Functions in Criteria

Functions in criteria let you create sophisticated search strategies that are easy to manipulate. For example, to find all dates that are earlier than today, you can enter the following function as criteria:

\(<\text{TODAY()}\>

➤ **Avoid using the following types of functions in criteria ranges:**

- Functions such as \(\text{ROW}\) and \(\text{COLUMN}\) that return information about the current cell. That information will change as the function steps through the records in the database.
- Functions such as \(\text{RAND}\) that do not return consistent results.
- Complex functions, if the user plans to use SQL to link to databases outside of Formula One for Java. (This capability is not currently part of Formula One for Java, but it may be added later as an enhancement.)

Using Cell References in Criteria

When creating criteria, you can use cell references. For example, to find all dates that match the date in cell H14, enter the following criteria:

\(=\text{H14}\)

You can also create complex criteria that refer to cells in the database. For example, say you have a database that contains customer names in column A and merchandise order dates in column B:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer</td>
<td>Order date</td>
<td>Ship</td>
</tr>
<tr>
<td>Henry, Max</td>
<td>3/11/97</td>
<td>3/2</td>
</tr>
<tr>
<td>O’Connell, Joseph</td>
<td>4/2/97</td>
<td>4/1</td>
</tr>
<tr>
<td>Ortiz, Helen</td>
<td>2/29/97</td>
<td>3/1</td>
</tr>
</tbody>
</table>

To search for all records with order dates within the last month, and to ensure that the formula you enter will always be updated to the current date, you can enter the following criteria:

\(=\text{AND(B2<\text{TODAY()}, B2>=\text{TODAY()}-30)}\)

The function will perform the test on cell B2, the topmost data cell in the order dates field. It will then perform the test on B3, B4, and so on.

You can use multiple nested functions and refer to one or more database column to create complex selection criteria.

➤ **When your criteria contains cell references, follow these guidelines:**

- Criteria that tests against the data in one particular cell (generally outside the database) should use an absolute cell reference.
Criteria that contains formulas that test against each record in the database should use a relative reference to the first row of data in its field.

Since criteria formulas that test against each record in the database may contain relative references to more than one cell in the database, the label above the formula’s cell can match any of the database labels.

Multiple Criteria

A criteria range that contains more than one row or column has multiple criteria.

When more than one cell in the first criteria row contains criteria, the function finds only the records that pass all the tests. You may use labels more than once in a criteria range to establish multiple criteria on a single field. For example, the following criteria range would find database records in which the 1996 salaries are between $29,000 and $33,000:

<table>
<thead>
<tr>
<th>Salary 96</th>
<th>Salary 96</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;29000</td>
<td>&lt;33000</td>
</tr>
</tbody>
</table>

You may also enter more than one row of criteria. If a record passes a test for one criteria row, it is picked regardless of whether it passes the tests in the other criteria rows. For example, the following criteria range would find all database records in which the 1996 salaries are either below $29,000 or above $33,000:

<table>
<thead>
<tr>
<th>Salary 96</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;29000</td>
</tr>
<tr>
<td>&gt;33000</td>
</tr>
</tbody>
</table>

Financials Functions

Many of the Financials functions have the same arguments. The following information expands on the information in the field descriptions of the functions’ documentation.

The frequency Argument

The frequency argument indicates the number of interest payments per year. The options are:

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Annual payments.</td>
</tr>
<tr>
<td>2</td>
<td>Semi-annual payments.</td>
</tr>
<tr>
<td>4</td>
<td>Quarterly payments.</td>
</tr>
</tbody>
</table>

If you enter any values other than 1, 2, or 4, the function will return the #NUM! error. Decimal values will be truncated to integers.
The **calendar_type** Argument

The `calendar_type` argument lets you pick from five different methods of counting days, usually for computing interest. The options are:

<table>
<thead>
<tr>
<th>Calendar type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>30-day months, 360-day years, American method.</td>
</tr>
<tr>
<td>1</td>
<td>Actual months, actual years.</td>
</tr>
<tr>
<td>2</td>
<td>Actual months, 360-day years.</td>
</tr>
<tr>
<td>3</td>
<td>Actual months, 365-day years.</td>
</tr>
<tr>
<td>4</td>
<td>30-day months, 360-day years, European method.</td>
</tr>
</tbody>
</table>

When `calendar_type` is 0 or 4, the DAYS360 function is used to calculate number of days. It has special rules for when the start and/or end dates fall on the 30th and/or 31st of the month. See “DAYS360” on page 86 for more information.

If you omit the `calendar_type` argument, 0 is used. If you enter any values other than 0, 1, 2, 3, or 4, the function will return the #NUM! error. Decimal values will be truncated to integers.

**Understanding Errors**

When a formula cannot be properly calculated, an error is returned in the cell. The following table lists the errors that can be generated and their causes.

<table>
<thead>
<tr>
<th>Error</th>
<th>Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>#DIV/0!</td>
<td>Divide by zero. May be caused by a reference to a blank cell or a cell containing zero.</td>
</tr>
<tr>
<td>#FORMULA!</td>
<td>Formula cannot be calculated. May be caused when opening a workbook from another file format.</td>
</tr>
<tr>
<td>#N/A</td>
<td>No value is available. May be caused by inappropriate values in the formula or a reference to a cell containing the #N/A value.</td>
</tr>
<tr>
<td>#NAME?</td>
<td>Name is not recognized. May be caused because a user-defined name is not defined, a function name is misspelled, or you are using an add-in function whose JAR file is not in your class path.</td>
</tr>
<tr>
<td>#NULL!</td>
<td>Null intersection. An intersection of two ranges was defined that does not intersect.</td>
</tr>
<tr>
<td>#NUM!</td>
<td>Number problem. May be caused by inappropriate numbers in functions, an iteration that cannot solve for a value, or a formula that results in a number too large or too small to represent.</td>
</tr>
<tr>
<td>#REF!</td>
<td>Reference error. May be caused by referring to a cell that was deleted.</td>
</tr>
<tr>
<td>#VALUE!</td>
<td>Wrong argument type. May be caused by entering text where a number was expected, or supplying a range to an operator or function that was expecting a single value.</td>
</tr>
</tbody>
</table>
Tidestone
This chapter lists the Formula One for Java functions by category and gives a short description of each function. The categories are:

- “Database Functions” on page 22
- “Date and Time Functions” on page 22
- “Engineering Functions” on page 23
- “Financial Functions” on page 25
- “Information Functions” on page 27
- “Logical Functions” on page 28
- “Lookup and Reference Functions” on page 28
- “Math and Trigonometry Functions” on page 29
- “Statistical Functions” on page 31
- “Text Functions” on page 34
# Database Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAVERAGE</td>
<td>Uses specified criteria to select records from a database, then computes the average of the numeric values in a specified field of the selected records.</td>
</tr>
<tr>
<td>DCOUNT</td>
<td>Uses specified criteria to select records from a database, then counts the number of selected records.</td>
</tr>
<tr>
<td>DCOUNTA</td>
<td>Uses specified criteria to select records from a database, then counts the number of selected records.</td>
</tr>
<tr>
<td>DGET</td>
<td>Uses specified criteria to select a single record from a database, then displays the data found in a specified field of the selected record.</td>
</tr>
<tr>
<td>DMAX</td>
<td>Uses specified criteria to select records from a database, then displays the highest number value found in a specified field of the selected records.</td>
</tr>
<tr>
<td>DMIN</td>
<td>Uses specified criteria to select records from a database, then displays the lowest number value found in a specified field of the selected records.</td>
</tr>
<tr>
<td>DPRODUCT</td>
<td>Uses specified criteria to select records from a database, then multiplies the numeric values in a specified field of the selected records.</td>
</tr>
<tr>
<td>DSTDEV</td>
<td>Uses specified criteria to select records from a database, then computes the sample standard deviation of the numeric values in a specified field of the selected records.</td>
</tr>
<tr>
<td>DSTDEVP</td>
<td>Uses specified criteria to select records from a database, then computes the population standard deviation of the numeric values in a specified field of the selected records.</td>
</tr>
<tr>
<td>DSUM</td>
<td>Uses specified criteria to select records from a database, then adds the numeric values in a specified field of the selected records.</td>
</tr>
<tr>
<td>DVAR</td>
<td>Uses specified criteria to select records from a database, then computes the sample variance of the numeric values in a specified field of the selected records.</td>
</tr>
<tr>
<td>DVARP</td>
<td>Uses specified criteria to select records from a database, then computes the population variance of the numeric values in a specified field of the selected records.</td>
</tr>
</tbody>
</table>

# Date and Time Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATE</td>
<td>Returns the serial number of the supplied date.</td>
</tr>
<tr>
<td>DATEDIF</td>
<td>Calculates the number of days, months, or years between two specified dates.</td>
</tr>
<tr>
<td>DATEVALUE</td>
<td>Returns the serial number of a date supplied as a text string.</td>
</tr>
<tr>
<td>DAY</td>
<td>Returns the day of the month that corresponds to the date represented by the supplied number.</td>
</tr>
</tbody>
</table>
Engineering Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAYS360</td>
<td>Returns the number of days between two dates, based on a 360-day year (twelve 30-day months).</td>
</tr>
<tr>
<td>EDATE</td>
<td>Finds a date a specified number of months before or after a given date.</td>
</tr>
<tr>
<td>EOMONTH</td>
<td>Finds the last date in the month that is a specified number of months before or after a given date.</td>
</tr>
<tr>
<td>HOUR</td>
<td>Returns the hour component of the specified time, in 24-hour format.</td>
</tr>
<tr>
<td>MINUTE</td>
<td>Returns the minute that corresponds to the supplied date.</td>
</tr>
<tr>
<td>MONTH</td>
<td>Returns the month that corresponds to the supplied date.</td>
</tr>
<tr>
<td>NETWORKDAYS</td>
<td>Computes the number of whole working days between the specified start date and the specified end date.</td>
</tr>
<tr>
<td>NOW</td>
<td>Returns the current date and time as a serial number.</td>
</tr>
<tr>
<td>SECOND</td>
<td>Returns the second that corresponds to the supplied date.</td>
</tr>
<tr>
<td>TIME</td>
<td>Returns a serial number for the supplied time.</td>
</tr>
<tr>
<td>TIMEVALUE</td>
<td>Returns a serial number for the supplied text representation of time.</td>
</tr>
<tr>
<td>TODAY</td>
<td>Returns the current date as a serial number.</td>
</tr>
<tr>
<td>WEEKDAY</td>
<td>Returns the day of the week that corresponds to the supplied date.</td>
</tr>
<tr>
<td>WORKDAY</td>
<td>Returns the date that is a specified number of days before or after a specified date, not counting weekends and specified holidays.</td>
</tr>
<tr>
<td>YEAR</td>
<td>Returns the year that corresponds to the supplied date.</td>
</tr>
<tr>
<td>YEARFRAC</td>
<td>Computes the fraction of a year between two specified dates.</td>
</tr>
</tbody>
</table>

Engineering Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BESSELI</td>
<td>Computes the value of the nth order modified Bessel function of the first kind evaluated at x.</td>
</tr>
<tr>
<td>BESSELJ</td>
<td>Computes the value of the nth order Bessel function of the first kind evaluated at x.</td>
</tr>
<tr>
<td>BESSELK</td>
<td>Computes the value of the nth order Bessel function of the second kind evaluated at x.</td>
</tr>
<tr>
<td>BESSELY</td>
<td>Computes the value of the nth order modified Bessel function of the second kind evaluated at x.</td>
</tr>
<tr>
<td>BIN2DEC</td>
<td>Converts a binary number (base 2) to a decimal number (base 10).</td>
</tr>
<tr>
<td>BIN2HEX</td>
<td>Converts a binary number (base 2) to a hexadecimal number (base 16).</td>
</tr>
<tr>
<td>BIN2OCT</td>
<td>Converts a binary number (base 2) to a octal number (base 8).</td>
</tr>
<tr>
<td>COMPLEX</td>
<td>Creates a complex number.</td>
</tr>
<tr>
<td>CONVERT</td>
<td>Converts a numeric value from one set of units to another.</td>
</tr>
<tr>
<td>DEC2BIN</td>
<td>Converts a decimal number (base 10) to a binary number (base 2).</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DEC2HEX</td>
<td>Converts a decimal number (base 10) to an hexadecimal number (base 16).</td>
</tr>
<tr>
<td>DEC2OCT</td>
<td>Converts a decimal number (base 10) to an octal number (base 8).</td>
</tr>
<tr>
<td>DELTA</td>
<td>Compares two specified values and returns 1 if they are equal, 0 if they are not.</td>
</tr>
<tr>
<td>ERF</td>
<td>Computes the “error” function.</td>
</tr>
<tr>
<td>ERFC</td>
<td>Computes the complementary “error” function.</td>
</tr>
<tr>
<td>GESTEP</td>
<td>Compares two specified values and returns 1 if the first value is greater than or equal to the second value, 0 if the first value is less than the second value.</td>
</tr>
<tr>
<td>HEX2BIN</td>
<td>Converts a hexadecimal number (base 16) to a binary number (base 2).</td>
</tr>
<tr>
<td>HEX2DEC</td>
<td>Converts a hexadecimal number (base 16) to a decimal number (base 10).</td>
</tr>
<tr>
<td>HEX2OCT</td>
<td>Converts a hexadecimal number (base 16) to an octal number (base 8).</td>
</tr>
<tr>
<td>IMABS</td>
<td>Computes the absolute value of a complex number.</td>
</tr>
<tr>
<td>IMAGINARY</td>
<td>Returns the coefficient of the imaginary portion of a complex number as a real number.</td>
</tr>
<tr>
<td>IMARGUMENT</td>
<td>Computes the argument $\theta$ as an angle expressed in radians.</td>
</tr>
<tr>
<td>IMCONJUGATE</td>
<td>Computes the complex conjugate of the argument.</td>
</tr>
<tr>
<td>IMCOS</td>
<td>Computes the complex cosine of the argument.</td>
</tr>
<tr>
<td>IMDIV</td>
<td>Divides one complex number by another.</td>
</tr>
<tr>
<td>IMEXP</td>
<td>Computes the complex exponential of the specified complex number.</td>
</tr>
<tr>
<td>IMLN</td>
<td>Computes the complex natural logarithm of the specified complex number.</td>
</tr>
<tr>
<td>IMLOG10</td>
<td>Computes the complex common logarithm of the specified complex number.</td>
</tr>
<tr>
<td>IMLOG2</td>
<td>Computes the complex logarithm base-2 of the specified complex number.</td>
</tr>
<tr>
<td>IMPower</td>
<td>Computes the real power of a complex number.</td>
</tr>
<tr>
<td>IMPRODUCT</td>
<td>Computes the product of up to 29 complex numbers.</td>
</tr>
<tr>
<td>IMREAL</td>
<td>Returns the coefficient of the real portion of a specified complex number.</td>
</tr>
<tr>
<td>IMSIN</td>
<td>Computes the complex sine of the specified complex number.</td>
</tr>
<tr>
<td>IMSQRT</td>
<td>Computes the complex square root of the specified complex number.</td>
</tr>
<tr>
<td>IMSUB</td>
<td>Subtracts one complex number from another.</td>
</tr>
<tr>
<td>IMSUM</td>
<td>Computes the sum of up to 29 complex numbers.</td>
</tr>
<tr>
<td>OCT2BIN</td>
<td>Converts an octal number (base 8) to a binary number (base 2).</td>
</tr>
<tr>
<td>OCT2DEC</td>
<td>Converts an octal number (base 8) to a decimal number (base 10).</td>
</tr>
<tr>
<td>OCT2HEX</td>
<td>Converts an octal number (base 8) to a hexadecimal number (base 16).</td>
</tr>
<tr>
<td>SQRTPI</td>
<td>Calculates the square root of the product of the specified number and $\pi$.</td>
</tr>
</tbody>
</table>
## Financial Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCRINT</td>
<td>Computes the accrued interest for a security that pays periodic interest, from the last coupon date to the settlement date.</td>
</tr>
<tr>
<td>ACCRINTM</td>
<td>Computes the accrued interest for a security that pays interest at maturity.</td>
</tr>
<tr>
<td>AMORDEGRC</td>
<td>Computes depreciation for the French accounting system, making use of a special French tax rule that allows over-depreciation of assets during the early years of ownership.</td>
</tr>
<tr>
<td>AMORLINC</td>
<td>Computes depreciation for the French accounting system.</td>
</tr>
<tr>
<td>COUPDAYBS</td>
<td>Computes the number of days from the coupon date previous to the settlement date and the settlement date.</td>
</tr>
<tr>
<td>COUPDAYS</td>
<td>Computes the number of days in the coupon period containing the settlement date.</td>
</tr>
<tr>
<td>COUPDAYSNC</td>
<td>Computes the number of days from the settlement date to the next coupon date.</td>
</tr>
<tr>
<td>COUPNCD</td>
<td>Computes the coupon date that follows the settlement date.</td>
</tr>
<tr>
<td>COUPNUM</td>
<td>Computes the number of coupons between settlement and maturity.</td>
</tr>
<tr>
<td>COUPPCD</td>
<td>Computes the last coupon date before the settlement date.</td>
</tr>
<tr>
<td>CUMIPMT</td>
<td>Computes the cumulative interest for the specified period.</td>
</tr>
<tr>
<td>CUMPRINC</td>
<td>Computes the cumulative principal paid for the specified period.</td>
</tr>
<tr>
<td>DB</td>
<td>Returns the real depreciation of an asset for a specific period of time using the fixed-declining balance method.</td>
</tr>
<tr>
<td>DDB</td>
<td>Returns the depreciation of an asset for a specific period of time using the double-declining balance method or a declining balance factor you supply.</td>
</tr>
<tr>
<td>DISC</td>
<td>Computes the discounted rate for a security.</td>
</tr>
<tr>
<td>DOLLARDE</td>
<td>Converts a dollar figure from fractional form to decimal form.</td>
</tr>
<tr>
<td>DOLLARFR</td>
<td>Converts a dollar figure from decimal form to fractional form.</td>
</tr>
<tr>
<td>DURATION</td>
<td>Computes the Macaulay duration for a security, in years.</td>
</tr>
<tr>
<td>EFFECT</td>
<td>Computes the effective annual interest rate, which adjusts the nominal rate to show the effect of compounding.</td>
</tr>
<tr>
<td>FV</td>
<td>Returns the future value of an annuity based on regular payments and a fixed interest rate.</td>
</tr>
<tr>
<td>FVSCHEDULE</td>
<td>Computes the future value of an investment after applying a series of compounded interest rates.</td>
</tr>
<tr>
<td>INTRATE</td>
<td>Computes the interest rate of a fully invested security, given a specified investment dollar amount and redemption value.</td>
</tr>
<tr>
<td>IPMT</td>
<td>Returns the interest payment of an annuity for a given period, based on regular payments and a fixed periodic interest rate.</td>
</tr>
<tr>
<td>IRR</td>
<td>Returns internal rate of return for a series of periodic cash flows.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MIRR</td>
<td>Returns the modified internal rate of return for a series of periodic cash flows.</td>
</tr>
<tr>
<td>MDURATION</td>
<td>Computes the modified duration for a security.</td>
</tr>
<tr>
<td>MINVERSE</td>
<td>Returns the modified internal rate of return for a series of periodic cash flows.</td>
</tr>
<tr>
<td>NOMINAL</td>
<td>Computes the nominal annual interest rate, given the effective interest rate.</td>
</tr>
<tr>
<td>NPER</td>
<td>Returns the number of periods of an investment based on regular periodic payments and a fixed interest rate.</td>
</tr>
<tr>
<td>NPV</td>
<td>Returns the net present value of an investment based on a series of periodic payments and a discount rate.</td>
</tr>
<tr>
<td>ODDFPRICE</td>
<td>Computes the price of a security purchased during a first coupon period that is shorter or longer than the other coupon periods.</td>
</tr>
<tr>
<td>ODDFYIELD</td>
<td>Computes the yield of a security per $100 face value when the first coupon period is shorter or longer than the other coupon periods.</td>
</tr>
<tr>
<td>ODDLPRICE</td>
<td>Computes the price of a security purchased during a last coupon period that is shorter or longer than the other coupon periods.</td>
</tr>
<tr>
<td>ODDLYIELD</td>
<td>Computes the yield of a security per $100 face value when the last coupon period is shorter or longer than the other coupon periods.</td>
</tr>
<tr>
<td>PMT</td>
<td>Returns the periodic payment of an annuity, based on regular payments and a fixed periodic interest rate.</td>
</tr>
<tr>
<td>PPMT</td>
<td>Returns the principle paid on an annuity for a given period.</td>
</tr>
<tr>
<td>PRICE</td>
<td>Computes the price of a security with specified rate, redemption value, and yield.</td>
</tr>
<tr>
<td>PRICEDISC</td>
<td>Computes the price of a discounted security per $100 face value, given a specified discount rate and redemption value.</td>
</tr>
<tr>
<td>PRICEMAT</td>
<td>Computes the price of a security that pays interest only at maturity, given specified dates, interest rate, and yield.</td>
</tr>
<tr>
<td>PV</td>
<td>Returns the present value of an annuity, considering a series of constant payments made over a regular payment period.</td>
</tr>
<tr>
<td>RATE</td>
<td>Returns the interest rate per period of an annuity, given a series of constant cash payments over a regular payment period.</td>
</tr>
<tr>
<td>RECEIVED</td>
<td>Computes the amount received at maturity for a fully invested security.</td>
</tr>
<tr>
<td>SLN</td>
<td>Returns the depreciation of an asset for a specific period of time using the straight-line balance method.</td>
</tr>
<tr>
<td>SYD</td>
<td>Returns the depreciation of an asset for a specified period using the sum-of-years method.</td>
</tr>
<tr>
<td>TBILLEQ</td>
<td>Computes the bond-equivalent yield for a treasury bill.</td>
</tr>
<tr>
<td>TBILLPRICE</td>
<td>Computes the price per $100 face value for a treasury bill.</td>
</tr>
<tr>
<td>TBILLYIELD</td>
<td>Computes the yield for a treasury bill.</td>
</tr>
</tbody>
</table>
Information Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CELL</td>
<td>Returns the specified type of information about a specified cell.</td>
</tr>
<tr>
<td>COUNTBLANK</td>
<td>Counts the number of empty cells in a specified range.</td>
</tr>
<tr>
<td>ERROR.TYPE</td>
<td>Returns a number corresponding to an error.</td>
</tr>
<tr>
<td>INFO</td>
<td>Returns the specified type of information about the current workbook and</td>
</tr>
<tr>
<td></td>
<td>system.</td>
</tr>
<tr>
<td>ISBLANK</td>
<td>Determines if the specified cell is blank.</td>
</tr>
<tr>
<td>ISERR</td>
<td>Determines if the specified expression returns an error value.</td>
</tr>
<tr>
<td>ISERROR</td>
<td>Determines if the specified expression returns an error value.</td>
</tr>
<tr>
<td>ISEVEN</td>
<td>Determines whether the specified expression returns an even value.</td>
</tr>
<tr>
<td>ISLOGICAL</td>
<td>Determines if the specified expression returns a logical value.</td>
</tr>
<tr>
<td>ISNA</td>
<td>Determines if the specified expression returns the value not available</td>
</tr>
<tr>
<td></td>
<td>error.</td>
</tr>
<tr>
<td>ISNONTEXT</td>
<td>Determines if the specified expression is not text.</td>
</tr>
<tr>
<td>ISNUMBER</td>
<td>Determines if the specified expression is a number.</td>
</tr>
<tr>
<td>ISODD</td>
<td>Determines whether the specified expression returns an odd value.</td>
</tr>
<tr>
<td>ISREF</td>
<td>Determines if the specified expression is a range reference.</td>
</tr>
<tr>
<td>ITEXT</td>
<td>Determines if the specified expression is text.</td>
</tr>
<tr>
<td>N</td>
<td>Tests the supplied value and returns the value if it is a number.</td>
</tr>
<tr>
<td>NA</td>
<td>Returns the error value #N/A, which represents “not available.”</td>
</tr>
<tr>
<td>TYPE</td>
<td>Returns the argument type of the given expression.</td>
</tr>
</tbody>
</table>
## Logical Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND</td>
<td>Returns True if all arguments are true; returns False if at least one argument is false.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Returns the logical value False.</td>
</tr>
<tr>
<td>IF</td>
<td>Tests the condition and returns the specified value.</td>
</tr>
<tr>
<td>NOT</td>
<td>Returns a logical value that is the opposite of its value.</td>
</tr>
<tr>
<td>OR</td>
<td>Returns True if at least one of a series of logical arguments is true.</td>
</tr>
<tr>
<td>TRUE</td>
<td>Returns the logical value True.</td>
</tr>
</tbody>
</table>

## Lookup and Reference Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDRESS</td>
<td>Creates a cell address as text.</td>
</tr>
<tr>
<td>AREAS</td>
<td>Counts the number of areas in a range reference.</td>
</tr>
<tr>
<td>CHOOSE</td>
<td>Returns a value from a list of numbers based on the index number supplied.</td>
</tr>
<tr>
<td>COLUMN</td>
<td>Returns the column number of the supplied reference.</td>
</tr>
<tr>
<td>COLUMNS</td>
<td>Returns the number of columns in a range reference.</td>
</tr>
<tr>
<td>HLOOKUP</td>
<td>Searches the top row of a table for a value and returns the contents of a cell in that table that corresponds to the location of the search value.</td>
</tr>
<tr>
<td>HYPERLINK</td>
<td>Sets up a hyperlink to a file or a web page.</td>
</tr>
<tr>
<td>INDEX (array type)</td>
<td>Returns a specified value from an array.</td>
</tr>
<tr>
<td>INDEX (non-array type)</td>
<td>Returns the contents of a cell from a specified range.</td>
</tr>
<tr>
<td>INDIRECT</td>
<td>Returns the contents of the cell referenced by the specified cell.</td>
</tr>
<tr>
<td>LOOKUP</td>
<td>Searches for a value in one range and returns the contents of the corresponding position in a second range.</td>
</tr>
<tr>
<td>MATCH</td>
<td>Compares a specified value against values in a range and returns the position of the matching value in the search range.</td>
</tr>
<tr>
<td>OFFSET</td>
<td>Returns the contents of a range that is offset from a starting point in the spreadsheet.</td>
</tr>
<tr>
<td>ROW</td>
<td>Returns the row number of the supplied reference.</td>
</tr>
<tr>
<td>ROWS</td>
<td>Returns the number of rows in a range reference.</td>
</tr>
<tr>
<td>TRANSPOSE</td>
<td>Places a copy of the contents of a range into a new range in which the original rows are now columns and the original columns are now rows.</td>
</tr>
<tr>
<td>VLOOKUP</td>
<td>Searches the first column of a table for a value and returns the contents of a cell in that table that corresponds to the location of the search value.</td>
</tr>
</tbody>
</table>
# Math and Trigonometry Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABS</td>
<td>Returns the absolute value of a number.</td>
</tr>
<tr>
<td>ACOS</td>
<td>Returns the arc cosine of a number.</td>
</tr>
<tr>
<td>ACOSH</td>
<td>Returns the inverse hyperbolic cosine of a number.</td>
</tr>
<tr>
<td>ASIN</td>
<td>Returns the arcsine of a number.</td>
</tr>
<tr>
<td>ASINH</td>
<td>Returns the inverse hyperbolic sine of a number.</td>
</tr>
<tr>
<td>ATAN</td>
<td>Returns the arctangent of a number.</td>
</tr>
<tr>
<td>ATAN2</td>
<td>Returns the arctangent of the specified coordinates.</td>
</tr>
<tr>
<td>ATANH</td>
<td>Returns the inverse hyperbolic tangent of a number.</td>
</tr>
<tr>
<td>CEILING</td>
<td>Rounds a number up to the nearest multiple of a specified significance.</td>
</tr>
<tr>
<td>COS</td>
<td>Returns the cosine of an angle.</td>
</tr>
<tr>
<td>COSH</td>
<td>Returns the hyperbolic cosine of a number.</td>
</tr>
<tr>
<td>COUNTIF</td>
<td>Returns the number of cells within a range that meet the given criteria.</td>
</tr>
<tr>
<td>DEGREES</td>
<td>Converts a value in radians to degrees.</td>
</tr>
<tr>
<td>EVEN</td>
<td>Rounds the specified number up to the nearest even integer.</td>
</tr>
<tr>
<td>EXP</td>
<td>Returns e raised to the specified power.</td>
</tr>
<tr>
<td>FACT</td>
<td>Returns the factorial of a specified number.</td>
</tr>
<tr>
<td>FACTDOUBLE</td>
<td>Returns the double-factorial of a specified number.</td>
</tr>
<tr>
<td>FLOOR</td>
<td>Rounds a number down to the nearest multiple of a specified significance.</td>
</tr>
<tr>
<td>GCD</td>
<td>Computes the greatest common divisor of a specified set of values.</td>
</tr>
<tr>
<td>INT</td>
<td>Rounds the supplied number down to the nearest integer.</td>
</tr>
<tr>
<td>LCM</td>
<td>Computes the least common multiple of a specified set of values.</td>
</tr>
<tr>
<td>LN</td>
<td>Returns the natural logarithm (based on the constant e) of a number.</td>
</tr>
<tr>
<td>LOG</td>
<td>Returns the logarithm of a number to the specified base.</td>
</tr>
<tr>
<td>LOG10</td>
<td>Returns the base-10 logarithm of a number.</td>
</tr>
<tr>
<td>MDETERM</td>
<td>Computes the determinant of a square matrix.</td>
</tr>
<tr>
<td>MINVERSE</td>
<td>Computes the inverse of a square matrix.</td>
</tr>
<tr>
<td>MMULT</td>
<td>Performs matrix multiplication on two arrays.</td>
</tr>
<tr>
<td>MOD</td>
<td>Returns the remainder after dividing a number by a specified divisor.</td>
</tr>
<tr>
<td>MROUND</td>
<td>Rounds a specified number to an even integral multiple of a specified factor.</td>
</tr>
<tr>
<td>MULTINOMIAL</td>
<td>Computes the multinomial of up to 29 numbers.</td>
</tr>
<tr>
<td>ODD</td>
<td>Rounds the specified number up to the nearest odd integer.</td>
</tr>
<tr>
<td>PI</td>
<td>Returns the value of pi.</td>
</tr>
<tr>
<td>POWER</td>
<td>Raises the specified base number to the specified power.</td>
</tr>
<tr>
<td>PRODUCT</td>
<td>Multiplies a list of numbers and returns the result.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>QUOTIENT</td>
<td>Computes the quotient of two numbers, truncated to an integer (toward 0).</td>
</tr>
<tr>
<td>RADIANS</td>
<td>Converts a value in degrees to radians.</td>
</tr>
<tr>
<td>RAND</td>
<td>Returns a number selected randomly from a uniform distribution greater than or equal to 0 and less than 1.</td>
</tr>
<tr>
<td>RANDBETWEEN</td>
<td>Returns an integer selected randomly from between two specified limits, inclusive, each time the sheet is recalculated.</td>
</tr>
<tr>
<td>ROMAN</td>
<td>Converts an arabic number to a Roman numeral.</td>
</tr>
<tr>
<td>ROUND</td>
<td>Rounds the given number to the supplied number of decimal places.</td>
</tr>
<tr>
<td>ROUNDDOWN</td>
<td>Rounds the given number down.</td>
</tr>
<tr>
<td>ROUNDUP</td>
<td>Rounds the given number up to the specified number of decimal places.</td>
</tr>
<tr>
<td>SERIESUM</td>
<td>Returns the sum of a power series.</td>
</tr>
<tr>
<td>SIGN</td>
<td>Determines the sign of the specified number.</td>
</tr>
<tr>
<td>SIN</td>
<td>Returns the sine of the supplied angle.</td>
</tr>
<tr>
<td>SINH</td>
<td>Returns the hyperbolic sine of the specified number.</td>
</tr>
<tr>
<td>SQRT</td>
<td>Returns the square root of the specified number.</td>
</tr>
<tr>
<td>SQRTPI</td>
<td>Calculates the square root of the product of the specified number and π.</td>
</tr>
<tr>
<td>SUM</td>
<td>Returns the sum of the supplied numbers.</td>
</tr>
<tr>
<td>SUMIF</td>
<td>Returns the sum of the specified cells based on the given criteria.</td>
</tr>
<tr>
<td>SUMPRODUCT</td>
<td>Multiplies the corresponding cells in the ranges, then returns the sum of those products.</td>
</tr>
<tr>
<td>SUMSQ</td>
<td>Squares each of the supplied numbers and returns the sum of the squares.</td>
</tr>
<tr>
<td>SUMX2MY2</td>
<td>Computes $x^2 - y^2$ for two sets of numbers, $arrayX$ and $arrayY$, paired up one-to-one.</td>
</tr>
<tr>
<td>SUMX2PY2</td>
<td>Computes $x^2 + y^2$ for two sets of numbers, $arrayX$ and $arrayY$, paired up one-to-one.</td>
</tr>
<tr>
<td>SUMXMY2</td>
<td>Computes $x^2 - y^2$ for two sets of numbers, $arrayX$ and $arrayY$, paired up one-to-one.</td>
</tr>
<tr>
<td>TAN</td>
<td>Returns the tangent of the specified angle.</td>
</tr>
<tr>
<td>TANH</td>
<td>Returns the hyperbolic tangent of a number.</td>
</tr>
<tr>
<td>TRUNC</td>
<td>Truncates the given number to an integer.</td>
</tr>
</tbody>
</table>
## Statistical Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVEDEV</td>
<td>Computes the average deviation of a list of numbers.</td>
</tr>
<tr>
<td>AVERAGE</td>
<td>Returns the average of the supplied numbers. The result of AVERAGE is also known as the arithmetic mean.</td>
</tr>
<tr>
<td>AVERAGEA</td>
<td>Computes the average of a list of numbers, considering textual values.</td>
</tr>
<tr>
<td>BETADIST</td>
<td>Computes the cumulative beta distribution.</td>
</tr>
<tr>
<td>BETAINV</td>
<td>Computes the inverse of the BETADIST function.</td>
</tr>
<tr>
<td>BINOMDIST</td>
<td>Computes the binomial distribution, which is used to determine probabilities on repeated tests where each test is independent of every other test.</td>
</tr>
<tr>
<td>CHIDIST</td>
<td>Computes the “complementary” version of the chi-square distribution.</td>
</tr>
<tr>
<td>CHINV</td>
<td>Computes the inverse of the “complementary” chi-square distribution.</td>
</tr>
<tr>
<td>CHITEST</td>
<td>Computes the probability of the distribution against expected values.</td>
</tr>
<tr>
<td>CONFIDENCE</td>
<td>Computes the confidence interval, assuming a normal distribution.</td>
</tr>
<tr>
<td>CORREL</td>
<td>Computes the correlation coefficient for two sets of numbers, paired up one-to-one.</td>
</tr>
<tr>
<td>COUNT</td>
<td>Returns the number of values in the supplied list.</td>
</tr>
<tr>
<td>COUNTA</td>
<td>Returns the number of nonblank values in the supplied list.</td>
</tr>
<tr>
<td>COVAR</td>
<td>Computes the covariance for pairwise numbers in two arrays.</td>
</tr>
<tr>
<td>CRITBINOM</td>
<td>Computes the smallest number of successes that will result in the BINOMDISTCUM function giving an answer larger than a specified criteria value.</td>
</tr>
<tr>
<td>DEVSQ</td>
<td>Computes the square of the deviation of the numbers.</td>
</tr>
<tr>
<td>EXPONDIST</td>
<td>Computes the exponential distribution.</td>
</tr>
<tr>
<td>FDIST</td>
<td>Computes the F-distribution, which is used to compare data from two different populations.</td>
</tr>
<tr>
<td>FINV</td>
<td>Computes the inverse of the complementary F-distribution (that is, the inverse of the FDIST function).</td>
</tr>
<tr>
<td>FISHER</td>
<td>Computes Fisher’s z-transformation.</td>
</tr>
<tr>
<td>FISHERINV</td>
<td>Computes the inverse of Fisher’s z-transformation.</td>
</tr>
<tr>
<td>FORECAST</td>
<td>Constructs the least squares regression line through the data given, then computes the predicted y value for the requested x.</td>
</tr>
<tr>
<td>FREQUENCY</td>
<td>Distributes the specified array of numbers, sorted in ascending order, into specified bins.</td>
</tr>
<tr>
<td>FTEST</td>
<td>Computes an F-test probability for two specified distributions.</td>
</tr>
<tr>
<td>GAMMADIST</td>
<td>Computes the gamma distribution for a given value.</td>
</tr>
<tr>
<td>GAMMAINV</td>
<td>Computes the inverse of the cumulative gamma distribution.</td>
</tr>
<tr>
<td>GAMMALN</td>
<td>Computes the logarithm of the gamma function.</td>
</tr>
<tr>
<td>GEOMEAN</td>
<td>Computes the geometric mean of a list of positive numbers.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GROWTH</td>
<td>Computes result values from the fitted curve of a multiple exponential regression for a group of specified observations relative to a group of specified independent variables.</td>
</tr>
<tr>
<td>HARMMEAN</td>
<td>Computes the harmonic mean of a list of positive numbers.</td>
</tr>
<tr>
<td>HYPGEOMDIST</td>
<td>Computes the hypergeometric distribution.</td>
</tr>
<tr>
<td>INTERCEPT</td>
<td>Computes the y-intercept of the least squares linear regression line through the given data.</td>
</tr>
<tr>
<td>KURT</td>
<td>Computes the kurtosis or of a list of numbers.</td>
</tr>
<tr>
<td>LARGE</td>
<td>Computes the Xth largest value in a list.</td>
</tr>
<tr>
<td>LINEST</td>
<td>Computes multiple linear regression for a group of observations relative to a number of independent variables.</td>
</tr>
<tr>
<td>LOGEST</td>
<td>Computes multiple exponential regression for a group of observations relative to a number of independent variables.</td>
</tr>
<tr>
<td>LOGINV</td>
<td>Computes the inverse of the cumulative lognormal distribution of x, where the logarithm of x is normally distributed.</td>
</tr>
<tr>
<td>LOGNORMDIST</td>
<td>Computes the cumulative lognormal distribution for x, where the logarithm of x is normally distributed.</td>
</tr>
<tr>
<td>MAX</td>
<td>Returns the largest value in the specified list of numbers.</td>
</tr>
<tr>
<td>MAXA</td>
<td>Returns the largest value in the specified list of numbers, considering textual values.</td>
</tr>
<tr>
<td>MEDIAN</td>
<td>Computes the median or middle value of a list of numbers arranged in ascending order.</td>
</tr>
<tr>
<td>MIN</td>
<td>Returns the smallest value in the specified list of numbers.</td>
</tr>
<tr>
<td>MINA</td>
<td>Returns the smallest value in the specified list of numbers, considering textual values.</td>
</tr>
<tr>
<td>MODE</td>
<td>Computes the most frequent value in a list of numbers (the number that appears most often in the list).</td>
</tr>
<tr>
<td>NEGBINOMDIST</td>
<td>Computes the negative binomial distribution.</td>
</tr>
<tr>
<td>NORMDIST</td>
<td>Computes the normal distribution.</td>
</tr>
<tr>
<td>NORMINV</td>
<td>Computes the inverse of the cumulative normal distribution.</td>
</tr>
<tr>
<td>NORMSDIST</td>
<td>Computes the standard normal cumulative distribution (often called “the bell curve”).</td>
</tr>
<tr>
<td>NORMSINV</td>
<td>Computes the inverse of the standard normal distribution.</td>
</tr>
<tr>
<td>PEARSON</td>
<td>Computes the correlation coefficient for two sets of numbers, paired up one-to-one.</td>
</tr>
<tr>
<td>PERCENTILE</td>
<td>Computes the value corresponding to the specified percentile of a specified range of numbers.</td>
</tr>
<tr>
<td>PERCENTRANK</td>
<td>Computes the percent rank of a specified value in a specified array of numbers.</td>
</tr>
<tr>
<td>PERMUT</td>
<td>Computes the number of combinations possible by taking k items at a time from a pool of n, where order in the sample taken is important.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>POISSON</td>
<td>Computes the Poisson distribution, which is usually used to determine the</td>
</tr>
<tr>
<td></td>
<td>probability of a certain number of repeated events taking place over time.</td>
</tr>
<tr>
<td>PROB</td>
<td>Given a set of numbers and a probability associated with each number,</td>
</tr>
<tr>
<td></td>
<td>computes the probability corresponding to a specified number.</td>
</tr>
<tr>
<td>QUARTILE</td>
<td>Computes the value corresponding to the specified quartile in a specified</td>
</tr>
<tr>
<td></td>
<td>array of numbers.</td>
</tr>
<tr>
<td>RANK</td>
<td>Computes the rank of a specified value in a specified array of numbers.</td>
</tr>
<tr>
<td>RSQ</td>
<td>Computes the square of the correlation coefficient for two sets of numbers,</td>
</tr>
<tr>
<td></td>
<td>paired up one-to-one.</td>
</tr>
<tr>
<td>SKEW</td>
<td>Computes the skewness of a set of numbers.</td>
</tr>
<tr>
<td>SLOPE</td>
<td>Computes the slope of the least squares linear regression line through the</td>
</tr>
<tr>
<td></td>
<td>given data.</td>
</tr>
<tr>
<td>SMALL</td>
<td>Computes the Xth smallest value in a list.</td>
</tr>
<tr>
<td>STANDARDIZE</td>
<td>Computes the proper argument for the function that calculates the standard</td>
</tr>
<tr>
<td></td>
<td>normal cumulative distribution, given specified values for x, mean, and</td>
</tr>
<tr>
<td></td>
<td>standard deviation.</td>
</tr>
<tr>
<td>STDEV</td>
<td>Returns the standard deviation of a population based on a sample of</td>
</tr>
<tr>
<td></td>
<td>supplied values.</td>
</tr>
<tr>
<td>STDEVA</td>
<td>Returns the standard deviation of a population based on a sample of</td>
</tr>
<tr>
<td></td>
<td>supplied values, considering textual values.</td>
</tr>
<tr>
<td>STDEVP</td>
<td>Returns the standard deviation of a population based on an entire</td>
</tr>
<tr>
<td></td>
<td>population of values.</td>
</tr>
<tr>
<td>STDEVP A</td>
<td>Returns the standard deviation of a population based on an entire</td>
</tr>
<tr>
<td></td>
<td>population of values, considering textual values.</td>
</tr>
<tr>
<td>STEYX</td>
<td>Computes the standard error of the predicted y-value for each x in the</td>
</tr>
<tr>
<td></td>
<td>regression.</td>
</tr>
<tr>
<td>TDIST</td>
<td>Computes the complementary student’s T-distribution.</td>
</tr>
<tr>
<td>TINV</td>
<td>Computes the x value corresponding to a specified probability value,</td>
</tr>
<tr>
<td></td>
<td>given a specified number of degrees of freedom.</td>
</tr>
<tr>
<td>TREND</td>
<td>Computes result values from the fitted curve of a multiple linear regression</td>
</tr>
<tr>
<td></td>
<td>for a group of specified observations relative to a group of specified</td>
</tr>
<tr>
<td></td>
<td>independent variables.</td>
</tr>
<tr>
<td>TRIMMEAN</td>
<td>Computes the mean of a list of numbers after reducing a specified percent</td>
</tr>
<tr>
<td></td>
<td>of the members of the list.</td>
</tr>
<tr>
<td>TTEST</td>
<td>Computes the student’s T-distribution from the data in two specified arrays</td>
</tr>
<tr>
<td></td>
<td>, then computes the probability.</td>
</tr>
<tr>
<td>VAR</td>
<td>Returns the variance of a population based on a sample of values.</td>
</tr>
<tr>
<td>VARA</td>
<td>Returns the variance of a population based on a sample of values,</td>
</tr>
<tr>
<td></td>
<td>considering textual values.</td>
</tr>
<tr>
<td>VARP</td>
<td>Returns the variance of a population based on an entire population of</td>
</tr>
<tr>
<td></td>
<td>values.</td>
</tr>
</tbody>
</table>
### Text Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAR</td>
<td>Returns a character that corresponds to the supplied Unicode value.</td>
</tr>
<tr>
<td>CLEAN</td>
<td>Removes all non-printable characters from the supplied text.</td>
</tr>
<tr>
<td>CODE</td>
<td>Returns the Unicode value of the first character of the supplied string.</td>
</tr>
<tr>
<td>CONCATENATE</td>
<td>Joins several text items into one item.</td>
</tr>
<tr>
<td>DOLLAR</td>
<td>Returns the specified number as text, using the local currency format and the supplied precision.</td>
</tr>
<tr>
<td>EXACT</td>
<td>Compares two expressions for identical, case-sensitive matches and returns True if identical, False if not.</td>
</tr>
<tr>
<td>FIND</td>
<td>Searches for a string of text within another text string and returns the character position at which the search string first occurs.</td>
</tr>
<tr>
<td>FIXED</td>
<td>Rounds a number to the supplied precision, formats the number in decimal format, and returns the result as text.</td>
</tr>
<tr>
<td>LEFT</td>
<td>Returns the leftmost character from the specified text string.</td>
</tr>
<tr>
<td>LEN</td>
<td>Returns the number of characters in the supplied text string.</td>
</tr>
<tr>
<td>LOWER</td>
<td>Changes the characters in the specified string to lowercase characters.</td>
</tr>
<tr>
<td>MID</td>
<td>Returns a specified number of characters from a text string, beginning with the specified starting position.</td>
</tr>
<tr>
<td>PROPER</td>
<td>Returns the specified string in proper-case format.</td>
</tr>
<tr>
<td>REPLACE</td>
<td>Replaces part of a text string with another text string.</td>
</tr>
<tr>
<td>REPT</td>
<td>Repeats a text string the specified number of times.</td>
</tr>
<tr>
<td>RIGHT</td>
<td>Returns the rightmost character(s) from the given text string.</td>
</tr>
<tr>
<td>SEARCH</td>
<td>Locates the position of the first character of a specified text string within another string.</td>
</tr>
<tr>
<td>SUBSTITUTE</td>
<td>Replaces a specified part of a text string with another text string.</td>
</tr>
<tr>
<td>T</td>
<td>Tests the supplied value and returns the value if it is text.</td>
</tr>
<tr>
<td>TEXT</td>
<td>Returns the given number as text, using the specified formatting.</td>
</tr>
<tr>
<td>TRIM</td>
<td>Removes all spaces from text except single spaces between words.</td>
</tr>
</tbody>
</table>
Function | Description
---|---
UPPER | Changes the characters in the specified string to uppercase characters.
USDOLLAR | Returns the specified number as text using the US dollar format and the supplied precision.
VALUE | Returns the specified text as a number.
This chapter provides a complete alphabetical reference for the Formula One for Java worksheet functions.

### ABS

**Description**

Returns the absolute value of a number.

**Syntax**

```plaintext
ABS (number)
```

**Remarks**

An absolute value does not display a positive or negative sign.

**Examples**

These functions both return 1:

- `ABS(-1)`
- `ABS(1)`

**See Also**

SIGN

### ACCRINT

**Description**

Computes the accrued interest for a security that pays periodic interest, from the last coupon date to the settlement date.

**Syntax**

```plaintext
ACCRINT (issue, first_interest, settlement, rate [, par], frequency [, calendar_type])
```
### Remarks
Use ACCRINT to compute the portion of the security’s price that is interest at settlement.

### Equation
\[
par \times \frac{rate}{frequency} \times \sum_{j=1}^{NC} \left( \frac{Aj}{NLj} \right)
\]

...where \( j \) is the number of pseudo-periods in the odd period. The other codes are explained in the following table.

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>( Aj )</td>
<td>Number of accrued days for the ( j )th pseudo-period within the odd period.</td>
</tr>
<tr>
<td>( NC )</td>
<td>Number of pseudo-periods that fit in the odd period (fractional values are raised to integers).</td>
</tr>
<tr>
<td>( NLj )</td>
<td>Number of days in the ( j )th pseudo-period within the odd period.</td>
</tr>
</tbody>
</table>

### Example
This function returns 51.50685:

\[
\text{ACCRINT(“9/12/92”,“9/30/92”,“12/15/92”,0.2,1000,1,1)}
\]

### See Also
PRICE, PRICEDISC, PRICEMAT, YIELD, YIELDDISC, YIELDMAT
ACCRINTM

Description
Computes the accrued interest for a security that pays interest at maturity. Note that this function does not calculate compounded interest.

Syntax
ACCRINTM (issue_date, maturity_date, rate [, par_value] [, calendar_type])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>issue_date</td>
<td>The date the security begins earning interest. Dates in the argument list must be in the form of a serial number or text. Decimal values will be truncated to integers.</td>
</tr>
<tr>
<td>maturity_date</td>
<td>The date the security is repaid and interest quits accumulating. It must be later than issue_date. Dates in the argument list must be in the form of a serial number or text. Decimal values will be truncated to integers.</td>
</tr>
<tr>
<td>rate</td>
<td>The security's annual interest rate.</td>
</tr>
<tr>
<td>[par_value]</td>
<td>Optional. The face value of the security. If this argument is omitted, $1000 is used.</td>
</tr>
<tr>
<td>[calendar_type]</td>
<td>Optional. One of five methods of counting days for computing interest. See “The calendar_type Argument” on page 19 for more information.</td>
</tr>
</tbody>
</table>

Equation
par value × rate × accrued days
days in year

Examples
This function returns 91.667
ACCRINTM("3/1/97","2/1/98",.1)

This function returns 183.333
ACCRINTM("3/1/97","2/1/98",.2)

See Also
ACCRINT

ACOS

Description
Returns the arc cosine of a number.

Syntax
ACOS (number)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>The cosine of the angle. The cosine can range from 1 to –1.</td>
</tr>
</tbody>
</table>
Remarks
The resulting angle is returned in radians (from 0 to PI). To convert the resulting radians to degrees, multiply the radians by 180/PI().

Examples
This function returns 1.05:
ACOS(.5)

This function returns 1.77:
ACOS(-.2)

See Also
COS

ACOSH
Description
Returns the inverse hyperbolic cosine of a number.
Syntax
ACOSH (number)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>Any number equal to or greater than 1.</td>
</tr>
</tbody>
</table>

Examples
This function returns .62:
ACOSH(1.2)

This function returns 1.76:
ACOSH(3)

See Also
ASINH, ATANH, COSH

ADDRESS
Description
Creates a cell address as text.
Syntax
ADDRESS ( row, column, ref_type [ , a1 ] [ , sheet ])
AMORDEGRC

Description
Computes depreciation for the French accounting system, making use of a special French tax rule that allows over-depreciation of assets during the early years of ownership.

Syntax
AMORDEGRC (cost, date, first_period, salvage, period, rate [, calendar_type])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cost</td>
<td>The cost of the asset.</td>
</tr>
<tr>
<td>date</td>
<td>The date the asset was purchased. Dates in the argument list must be entered as text.</td>
</tr>
<tr>
<td>first_period</td>
<td>The date marking the end of the first period of depreciation. Dates in the argument list must be entered as text.</td>
</tr>
<tr>
<td>salvage</td>
<td>The salvage value of the asset at the end of depreciation.</td>
</tr>
<tr>
<td>period</td>
<td>The number of the period you want to compute depreciation for. Enter 0 to compute depreciation for the period in which the asset was purchased.</td>
</tr>
</tbody>
</table>

Examples
This function returns $F5:
ADDRESS(5, 6, 1)

This function returns SALES!F5:
ADDRESS(5, 6, 4, TRUE(), SALES.)

See Also
COLUMN, OFFSET, ROW
Remarks
To calculate normal depreciation under the French system, use AMORLINC.

Equations
The equations require a value for depreciation coefficient, which is different depending on the number of years in the life of the asset being depreciated. The life of asset value is the inverse of rate.

<table>
<thead>
<tr>
<th>rate</th>
<th>Life of asset</th>
<th>Depreciation coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25 - 0.333333</td>
<td>3 - 4</td>
<td>1.5</td>
</tr>
<tr>
<td>0.2 - 0.166666</td>
<td>5 - 6</td>
<td>2.0</td>
</tr>
<tr>
<td>&lt;0.166666</td>
<td>&gt;6</td>
<td>2.5</td>
</tr>
</tbody>
</table>

For period 0:
\[ \text{cost} \times \text{rate} \times \text{depreciation coefficient} \times \text{YEARFRAC(date, first_period, calendar_type)} \]

For periods 1 through life of asset - 1:
\[ \frac{[\text{cost} - \text{value from period 0}] / 2 \times \text{rate} \times \text{depreciation coefficient}}{\text{rate} \times \text{depreciation coefficient}} \]

For the last period:
\[ [\text{cost} - \text{value from period 0}] \times \text{rate} \times \text{depreciation coefficient} \]

Examples
This function returns 58:
\[
\text{AMORDEGRC}(1000, "5/5/95", "7/1/95", .100, .0, .15)
\]

See Also
AMORLINC, DURATION, MDURATION
AMORLINC

Computes depreciation for the French accounting system.

**Syntax**

```
AMORLINC(cost, date, first_period, salvage, period, rate [, calendar_type])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cost</td>
<td>The cost of the asset.</td>
</tr>
<tr>
<td>date</td>
<td>The date the asset was purchased. Dates in the argument list must be entered as text.</td>
</tr>
<tr>
<td>first_period</td>
<td>The date marking the end of the first period of depreciation. Dates in the argument list must be entered as text.</td>
</tr>
<tr>
<td>salvage</td>
<td>The salvage value of the asset at the end of depreciation.</td>
</tr>
<tr>
<td>period</td>
<td>The number of the period you want to compute depreciation for. Enter 0 to compute depreciation for the period in which the asset was purchased.</td>
</tr>
<tr>
<td>rate</td>
<td>The rate of depreciation.</td>
</tr>
<tr>
<td>[calendar_type]</td>
<td>Optional. One of four methods of counting days for computing interest. The options are:</td>
</tr>
<tr>
<td></td>
<td>0 30-day months, 360-day years, American method.</td>
</tr>
<tr>
<td></td>
<td>1 Actual months, actual years.</td>
</tr>
<tr>
<td></td>
<td>3 Actual months, 365-day years.</td>
</tr>
<tr>
<td></td>
<td>4 30-day months, 360-day years, European method.</td>
</tr>
</tbody>
</table>

When `calendar_type` is 0 or 4, the DAYS360 function is used to calculate number of days. It has special rules for when the start and/or end dates fall on the 30th and/or 31st of the month. See “DAYS360” on page 86 for more information.

If you omit this argument, 0 is used. If you enter any values other than 0, 1, 3, or 4, the function will return the #NUM! error. Decimal values will be truncated to integers.

Note that while other functions that use the `calendar_type` argument allow an option 2, this one does not.

**Remarks**

To make use of a special French tax rule that allows over-depreciation of assets during the early years of ownership, use AMORDEGRC.

**Equations**

For period 0: \( \text{cost} \times \text{rate} \times \text{YEARFRAC}(date, first\_period, calendar\_type) \)

For the remaining periods: \( \text{cost} \times \text{rate} \)

In the final period(s), `cost` is adjusted so that it never goes below `salvage`. 
**Examples**

This function returns 23.333:

AMORLINC(1000, "5/5/95", "7/1/95", 100, 0, 0.15)

**See Also**

AMORDEGRC, DURATION, MDURATION

---

### AND

**Description**

Returns True if all arguments are true; returns False if at least one argument is false.

**Syntax**

AND ( logical_list )

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>logical_list</td>
<td>A list of conditions separated by commas. You can include as many as 30 conditions in the list. The list can contain logical values or a reference to a range containing logical values. Text and empty cells are ignored. If there are no logical values in the list, the error #VALUE! is returned.</td>
</tr>
</tbody>
</table>

**Examples**

This function returns True because both arguments are true:

AND(1+1=2, 5+5=10)

This function returns False:

AND(TRUE(), FALSE())

**See Also**

NOT, OR, ROW

---

### AREAS

**Description**

Counts the number of areas in a range reference.

**Syntax**

AREAS ( range )

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>range</td>
<td>One or more range references or the name(s) of named range references. If you enter more than one range reference, enclose them in an extra set of parentheses. Use commas or spaces to separate different ranges: spaces will find the number of areas in which all of the ranges intersect, while commas will find the number of ranges entered, regardless of the intersections. You can use multiple and nested parentheses. Areas for ranges in the innermost parentheses are determined first.</td>
</tr>
</tbody>
</table>
Examples

This function returns 1:
AREAS(B2:C3)

This function returns 2:
AREAS((B2:D3,C3))

This function returns 1:
AREAS((B2:D3 C3))

ASIN

Description
Returns the arcsine of a number.

Syntax
ASIN ( number)

Argument | Description
---|---
number | The sine of the resulting angle, ranging from –1 to 1.

Remarks
The resulting angle is returned in radians (ranging from -PI/2 to PI/2). To convert the resulting radians to degrees, multiply the radians by 180/PI().

Examples
This function returns 1.57:
ASIN(1)

This function returns .41:
ASIN(.4)

See Also
ASINH, PI, SIN

ASINH

Description
Returns the inverse hyperbolic sine of a number.

Syntax
ASINH ( number)

Argument | Description
---|---
number | Any number.

Examples
This function returns 2.37:
ASINH(5.3)
This function returns ~2.09:

\[ \text{ASINH}(-4) \]

**See Also**

ACOSH, ASIN, ATANH, SINH

## ATAN

**Description**
Returns the arctangent of a number.

**Syntax**

\[
\text{ATAN}( \text{number} )
\]

**Argument** | **Description**
--- | ---

\text{number} | The tangent of the angle.

**Remarks**
The resulting angle is returned in radians, ranging from -PI/2 to PI/2. To convert the resulting radians to degrees, multiply the radians by 180/PI().

**Examples**

This function returns 1.29:

\[ \text{ATAN}(3.5) \]

This function returns -1.33:

\[ \text{ATAN}(4) \]

**See Also**

ATAN2, ATANH, PI, TAN

## ATAN2

**Description**
Returns the arctangent of the specified coordinates.

**Syntax**

\[
\text{ATAN2}( x, y )
\]

**Argument** | **Description**
--- | ---

\( x \) | The x coordinate.

\( y \) | The y coordinate.

**Remarks**
The arctangent is the angle from the x axis to a line with end points at the origin (0, 0) and a point with the given coordinates (x, y). The angle is returned in radians, ranging from -PI to PI, excluding -PI.

**Examples**

This function returns 1.11:

\[ \text{ATAN2}(3, 6) \]
This function returns 3.04:

\[
\text{ATAN2}(-1, .1)
\]

**See Also**

ATAN, ATANH, PI, TAN

---

**ATANH**

**Description**

Returns the inverse hyperbolic tangent of a number.

**Syntax**

\[
\text{ATANH}( \text{number} )
\]

**Examples**

This function returns .55:

\[
\text{ATANH}( .5 )
\]

This function returns −.26:

\[
\text{ATANH}( -.25 )
\]

**See Also**

ACOS, ASINH, TANH

---

**AVEDEV**

**Description**

Computes the average deviation of a list of numbers.

**Syntax**

\[
\text{AVEDEV}( \text{number_list} )
\]

**Equation**

\[
\frac{1}{N} \sum_{j=1}^{N} |x_j - \bar{x}|
\]
Examples
This function returns 0.5:
\[
\text{AVEDEV}(1, 2)
\]

See Also
AVERAGE, COUNT, DEVSQ, KURT, SKEW

AVERAGE

Description
Returns the average of the supplied numbers. The result of AVERAGE is also known
as the arithmetic mean.

Syntax
\[
\text{AVERAGE}(\text{number_list})
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number_list</td>
<td>A list of up to 30 numbers separated by commas. The list may contain numeric values, cell references, range references, or array constants. Text and logical values in range references and array constants are ignored. Text entered in the argument list will be evaluated as a number, if possible; otherwise it will cause a #VALUE! error. Logical values entered into the argument list are evaluated as numeric 1 if TRUE and numeric 0 if FALSE.</td>
</tr>
</tbody>
</table>

Examples
This function returns 8.25:
\[
\text{AVERAGE}(5, 6, 8, 14)
\]

This function returns 134, the average of the values in the range C15:C17:
\[
\text{AVERAGE(C15:C17)}
\]

See Also
AVERAGEA, MIN, MAX

AVERAGEA

Description
Computes the average of a list of numbers in a Lotus-compatible fashion. This function is equivalent to the AVERAGE function, but its implementation treats text and logical values in cell and range references differently.

Syntax
\[
\text{AVERAGEA}(\text{number_list})
\]
Chapter 3  A-Z Function Reference 49

Examples

This function returns 1.5:

```
AVERAGEA(1,2)
```

See Also

AVERAGE, COUNTA

---

**BESSELI**

**Description**  Computes the value of the $n$th order modified Bessel function of the first kind evaluated at $x$.

**Syntax**  

```
BESSELI ( number, order )
```

**Argument** | **Description**
---|---
number | Any real number $x$ at which to evaluate the function.
order | Any positive whole number or 0. Values are truncated to integers.

**Equation**  

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

**Examples**  

This function returns 3.953370217:

```
BESSELI(3,1)
```

**See Also**  

BESSELJ, BESSELK, BESSELY

---

**BESSELJ**

**Description**  Computes the value of the $n$th order Bessel function of the first kind evaluated at $x$.

**Syntax**  

```
BESSELJ ( number, order )
```
**BESSELK**

**Description**
Computes the value of the *n*th order Bessel function of the second kind evaluated at *x*.

**Syntax**
```
BESSELK ( number, order )
```

**Equation**

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

**Examples**
This function returns 9.758562803:
```
BESSELK(5,10)
```

This function returns 180713288.5:
```
BESSELK(1,10)
```

**See Also**
BESSELI, BESSELJ, BESSELY

---

**BESSELJ**

**Description**
Computes the value of the *n*th order Bessel function of the first kind evaluated at *x*.

**Syntax**
```
BESSELJ ( number, order )
```

**Equation**

\[
J_n(z) = \left(\frac{1}{2} \right)^n \sum_{k = 0}^{\infty} \frac{\left(\frac{1}{4} \right)^k}{k! \Gamma(n + k + 1)}
\]

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

**Examples**
This function returns 0.497094103:
```
BESSELJ(2.5,1)
```

**See Also**
BESSELI, BESSELK, BESSELY

---

**BESSELI**

**Description**
Computes the value of the *n*th order modified Bessel function of the first kind evaluated at *x*.

**Syntax**
```
BESSELI ( number, order )
```

**Equation**

\[
I_n(x) = \sum_{k = 0}^{\infty} \frac{\left(\frac{1}{2} \right)^k}{k! \Gamma(n + k + 1)}
\]

**Examples**
This function returns 0.497094103:
```
BESSELI(2.5,1)
```

**See Also**
BESSELJ, BESSELK, BESSELY

---

**BESSELK**

**Description**
Computes the value of the *n*th order Bessel function of the second kind evaluated at *x*.

**Syntax**
```
BESSELK ( number, order )
```

**Equation**

\[
J_n(z) = \left(\frac{1}{2} \right)^n \sum_{k = 0}^{\infty} \frac{\left(\frac{1}{4} \right)^k}{k! \Gamma(n + k + 1)}
\]

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

**Examples**
This function returns 0.497094103:
```
BESSELJ(2.5,1)
```

**See Also**
BESSELI, BESSELK, BESSELY
### BESSELY

**Description**
Computes the value of the \( n \)th order modified Bessel function of the second kind evaluated at \( x \).

**Syntax**

```plaintext
BESSELY ( number, order)
```

**Equation**

\[
Y_v(x) = \frac{J_v(x)\cos(v\pi) - J_{-v}(x)}{\sin(v\pi)}
\]

**Examples**

This function returns 0.376850015:

```
BESSELY(3,0)
```

This function returns -25.12910983:

```
BESSELY(5,10)
```

**See Also**

BESSELI, BESSELJ, BESSELK

### BETADIST

**Description**
Computes the cumulative beta distribution.

**Syntax**

```plaintext
BETADIST (x, alpha, beta [, A] [, B])
```

**Argument**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x )</td>
<td>The value at which the function will be evaluated. It must be a number larger than ( A ) and smaller than ( B ).</td>
</tr>
<tr>
<td>( alpha )</td>
<td>The parameter to the beta function. It must be a number greater than zero.</td>
</tr>
<tr>
<td>( beta )</td>
<td>A second parameter to the beta function. It must be a number greater than zero.</td>
</tr>
<tr>
<td>( [A] )</td>
<td>Optional. The lower bound to the interval of ( x ). It must be a number smaller than ( x ) and ( B ). If this argument is omitted, 0 is used.</td>
</tr>
<tr>
<td>( [B] )</td>
<td>Optional. The upper bound to the interval of ( x ). It must be a number larger than ( x ) and ( A ). If this argument is omitted, 1 is used.</td>
</tr>
</tbody>
</table>
**Formula One for Java Function Reference**

**Equation**

\[
\frac{1}{B(a, b)} \int_{0}^{x} t^{a-1}(1-t)^{b-1}dt \quad \text{where} \quad B(a, b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}
\]

**Examples**

This function returns 0.999023437:

\[\text{BETADIST}(0.5, 1, 10)\]

This function returns 0.000976563:

\[\text{BETADIST}(0.5, 10, 1)\]

This function returns 1:

\[\text{BETADIST}(0.5, 1000, 2000)\]

This function returns 0.685470581:

\[\text{BETADIST}(2, 8, 10, 1, 3)\]

**See Also**

BINOMDIST, CHIDIST, COMBIN, CRITBINOM, EXPONDIST, GAMMADIST, NORMDIST, POISSON

---

**BETAINV**

**Description**

Computes the inverse of the BETADIST function.

**Syntax**

\[
\text{BETAINV} \left( \text{probability}, \alpha, \beta \ [, \ A] \ [, \ B] \right)
\]

**Argument** | **Description**
--- | ---
\( \text{probability} \) | The probability, as obtained from the BETADIST function. It must be a number between zero and 1 that represents a probability.
\( \alpha \) | The parameter to the beta function. It must be a number greater than zero.
\( \beta \) | A second parameter to the beta function. It must be a number greater than zero.
\( [A] \) | Optional. The lower bound to the interval of \( x \). It must be a number smaller than \( x \) and \( B \). If this argument is omitted, 0 is used.
\( [B] \) | Optional. The upper bound to the interval of \( x \). It must be a number larger than \( x \) and \( A \). If this argument is omitted, 1 is used.

**Examples**

This function returns 2:

\[\text{BETAINV}(0.685470581, 8, 10, 1, 3)\]
This function returns 0.5:
BETAINV(0.685470581, 8, 10)

This function returns 0.5:
BETAINV(0.5, 1, 1)

See Also
BETADIST

---

**BIN2DEC**

**Description**
Converts a binary number (base 2) to a decimal number (base 10).

**Syntax**
BIN2DEC (binary_number)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>binary_number</td>
<td>Any number expressed in the binary number system (represented by the digits 1 or 0).</td>
</tr>
<tr>
<td></td>
<td>Data may be entered in text or numerical form.</td>
</tr>
<tr>
<td></td>
<td>A maximum of 9 value bits (or digits) is allowed in representing positive values.</td>
</tr>
<tr>
<td></td>
<td>To represent negative values, enter exactly 10 digits—the first digit must be a 1.</td>
</tr>
<tr>
<td></td>
<td>Positive values may be zero-padded on the left up to a maximum of 10 digits.</td>
</tr>
</tbody>
</table>

**Remarks**
Note that the result is a numeric value.

**Examples**
This function returns 1:
BIN2DEC(0000000001)

This function returns 128:
BIN2DEC(10000000)

**See Also**
BIN2HEX, BIN2OCT, DEC2BIN, DEC2HEX, DEC2OCT, HEX2BIN, HEX2DEC, HEX2OCT, OCT2BIN, OCT2DEC, OCT2HEX
**BIN2HEX**

**Description**  
Converts a binary number (base 2) to a hexadecimal number (base 16).

**Syntax**  
BIN2HEX ( *binary_number* [, *places]*)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
</table>
| *binary_number* | Any number expressed in the binary number system (represented by the digits 1 or 0).  
Data may be entered in text or numerical form.  
A maximum of 9 value bits (or digits) is allowed in representing positive values.  
To represent negative values, enter exactly 10 digits—the first digit must be a 1.  
Positive values may be zero-padded on the left up to a maximum of 10 digits. |
| *places* | Optional number of characters to use in the output. If *places* is omitted, the function returns the number of spaces required to display the result.  
The *places* argument can be used to 0-pad a positive number.  
If the number is negative, *places* is ignored and 10 characters are returned. |

**Remarks**  
Note that the result of this function is a text string.

**Examples**  
This function returns 3F:

BIN2HEX(11111)

This function returns FF:

BIN2HEX(11111111)

**See Also**  
BIN2DEC, BIN2OCT, DEC2BIN, DEC2HEX, DEC2OCT, HEX2BIN, HEX2DEC, HEX2OCT, OCT2BIN, OCT2DEC, OCT2HEX

---

**BIN2OCT**

**Description**  
Converts a binary number (base 2) to an octal number (base 8).

**Syntax**  
BIN2OCT ( *binary_number* [, *places]*)
Chapter 3  A-Z Function Reference

**Remarks**
Note that the result of this function is a text string.

**Examples**
This function returns 164:

\[
\text{BIN2OCT}(1110100)
\]

This function returns 00777:

\[
\text{BIN2OCT}(111111111,5)
\]

**See Also**
BIN2DEC, BIN2HEX, DEC2BIN, DEC2HEX, DEC2OCT, HEX2BIN, HEX2DEC, HEX2OCT, OCT2BIN, OCT2DEC, OCT2HEX

---

**BINOMDIST**

**Description**
Computes the binomial distribution, which is used to determine probabilities on repeated tests where each test is independent of every other test (that is, the probability is the same for each test).

**Syntax**

\[
\text{BINOMDIST} (successes, trials, probability, cumulative)
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>successes</td>
<td>An integer describing the number of successes in the test. It must be a number larger than or equal to 0 and smaller than trials.</td>
</tr>
</tbody>
</table>
BINOMDIST is similar to NEGBINOMDIST, only in BINOMDIST the number of successes is variable and the number of trials is fixed, while in NEGBINOMDIST the number of successes is fixed and the number of trials is variable.

Equations

When cumulative is FALSE: \( f(x) = \binom{n}{x} p^x q^{n-x} \)

When cumulative is TRUE: \( \sum_{j=k}^{n} \binom{n}{j} p^j (1-p)^{n-j} = I_p(k, n-k+1) \)

Examples

The following examples return the probabilities of getting a 6 on a die roll.

This function calculates the possibility of getting a 6 exactly once in 6 trials. It returns 0.401878:

\[
\text{BINOMDIST}(1,6,0.166666667,\text{FALSE})
\]

This function calculates the possibility of getting a 6 zero or 1 time in 6 trials. It returns 0.736776:

\[
\text{BINOMDIST}(1,6,0.166666667,\text{TRUE})
\]

See Also

BETADIST, CHIDIST, COMBIN, CRITBINOM, EXPONDIST, GAMMADIST, NEGBINOMDIST, NORMSDIST, POISSON

Argument | Description
--- | ---
trials | An integer describing the total number of attempts in the test. It must be a number larger than successes and smaller than the largest positive integer the machine can handle.
probability | A value between 0 and 1 that represents the chance of success on an individual attempt. For example, the probability of heads in a coin toss is considered to be 0.5; the probability of getting “snake eyes” (two ones) when throwing a pair of dice is 1/36 or 0.02778.
cumulative | A boolean value indicating the type of calculation you want. Enter FALSE to calculate the possibility of the success case (for example, getting exactly 3 heads in 4 coin tosses). Enter TRUE to calculate the sum of all cases from 0 to the success case (for example, for the probability of getting 0 or 1 or 2 or 3 heads in 4 coin tosses).

Remarks

Coin tosses or die rolls are examples of uses for BINOMDIST. For cases of sampling without replacement, use HYPGEOMDIST.
CALL

Description
Returns the #N/A error message.

Syntax
CALL (arguments)

Remarks
This function was included in Formula One for Java only for compatibility with Microsoft Excel. In Excel, CALL is used to call procedures in dynamic linked libraries or code resources, neither of which is supported in Java.

This function is intended only for users or developers who want to import Excel worksheets that contain Excel’s CALL function. To make those worksheets work properly, developers should convert these calls to add-in functions, which they can write in Java and set up to be automatically loaded.

For more information, see the chapter on creating add-in functions in the Formula One for Java Technical Guide.

CELL

Description
Returns the specified type of information about a specified cell. If no cell is specified, the function will return information about the cell the function was entered in.

Syntax
CELL (info_type [, cell])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>info_type</td>
<td>The type of information you want about cell. It must be text, surrounded by quotation marks. The following are the valid entries and the types of information they return, listed in alphabetical order:</td>
</tr>
<tr>
<td>&quot;across&quot;</td>
<td>Returns 1 if the cell is formatted for long text centered across several cells. Otherwise, it returns 0.</td>
</tr>
<tr>
<td>&quot;address&quot;</td>
<td>Returns an absolute reference to cell.</td>
</tr>
<tr>
<td>&quot;backgroundcolor&quot;</td>
<td>Returns the index number of the fill color of cell.</td>
</tr>
<tr>
<td>&quot;bold&quot;</td>
<td>Returns 1 if the font in cell is bold. Otherwise, it returns 0.</td>
</tr>
<tr>
<td>&quot;bottomborder&quot;</td>
<td>Returns a code corresponding to the type of the bottom border of cell. The codes are: 0 for none, 1 for thin, 2 for medium, 3 for dashed, 4 for dotted, 5 for thick, 6 for double, and 7 for hairline.</td>
</tr>
<tr>
<td>Argument</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>info_type</td>
<td>Returns the index number of the color of cell’s bottom border.</td>
</tr>
<tr>
<td>“bottombordercolor”</td>
<td>Returns the index number of the color of cell’s bottom border.</td>
</tr>
<tr>
<td>“col”</td>
<td>Returns the column number of cell. The result is the same as the result of the COLUMN function.</td>
</tr>
<tr>
<td>“color”</td>
<td>Returns #N/A.</td>
</tr>
<tr>
<td>“contents”</td>
<td>Returns the contents of cell.</td>
</tr>
<tr>
<td>“coord”</td>
<td>Returns an absolute reference to cell, including the book and sheet names if they are different from the cell containing this function.</td>
</tr>
<tr>
<td>“datatype”</td>
<td>Returns a code corresponding to the type of data found in cell. The codes are: b for a blank cell; v for a number, a formula that returns a number, a formula that returns a logical value, or a formula that returns a date/time value; l for text or a formula that returns text; e for any error value except #N/A; and n for the #N/A error value.</td>
</tr>
<tr>
<td>“filedate”</td>
<td>Returns #N/A.</td>
</tr>
<tr>
<td>“filename”</td>
<td>Returns #N/A.</td>
</tr>
<tr>
<td>“fontface”</td>
<td>Returns the name of the font in cell.</td>
</tr>
<tr>
<td>“fontsize”</td>
<td>Returns the size of the font in cell, in points.</td>
</tr>
<tr>
<td>“format”</td>
<td>Returns #N/A.</td>
</tr>
<tr>
<td>“formulatype”</td>
<td>Returns a code corresponding to the type of data and/or formula found in cell. The codes are: b for a blank cell, v for a number, fv for a formula that returns a number, a formula that returns text, e for any error value except #N/A, fe for a formula that returns any error value except #N/A, n for the #N/A error value, and fn for a formula that returns the #N/A error value.</td>
</tr>
<tr>
<td>“halign”</td>
<td>Returns a code corresponding to cell’s horizontal alignment type. The codes are: 0 for general, 1 for left, 2 for centered, 3 for right, 4 for fill (the fill string is repeated to fill the width of cell), 5 for justified, and 6 for centered across cells.</td>
</tr>
<tr>
<td>“height”</td>
<td>Returns the row height of cell’s row, in points.</td>
</tr>
<tr>
<td>“italic”</td>
<td>Returns 1 if the font in cell is italic. Otherwise, it returns 0.</td>
</tr>
<tr>
<td>“leftborder”</td>
<td>Returns a code corresponding to the type of the left border of cell. The codes are: 0 for none, 1 for thin, 2 for medium, 3 for dashed, 4 for dotted, 5 for thick, 6 for double, and 7 for hairline.</td>
</tr>
<tr>
<td>“leftbordercolor”</td>
<td>Returns the index number of the color of cell’s left border.</td>
</tr>
<tr>
<td>“orientation”</td>
<td>Returns #N/A.</td>
</tr>
<tr>
<td>“parentheses”</td>
<td>Returns #N/A.</td>
</tr>
</tbody>
</table>
“info_type”

Returns the index number of cell’s fill pattern, or 0 if cell has no fill pattern.

“patterncolor”

Returns the index number of the foreground color used in cell’s fill pattern.

“prefix”

Returns a code corresponding to cell’s alignment. The codes are: single tic (') for left-justified text, quotation marks (") for right-justified text, caret (^) for centered text, and backslash (\) for fill-aligned text. Otherwise it returns empty text ('"').

“protect”

Returns #N/A.

“rightborder”

Returns a code corresponding to the type of the right border of cell. The codes are: 0 for none, 1 for thin, 2 for medium, 3 for dashed, 4 for dotted, 5 for thick, 6 for double, and 7 for hairline.

“rightbordercolor”

Returns the index number of the color of cell’s right border.

“rotation”

Returns #N/A.

“row”

Returns the row number of cell.

“sheet”

Returns a letter corresponding to the position of cell’s worksheet: A is the leftmost worksheet, B is the next worksheet, etc.

“sheetname”

Returns the name of cell’s worksheet, if it is named; otherwise, returns a letter corresponding to the position of cell’s worksheet: A is the leftmost worksheet, B is the next worksheet, etc.

“textcolor”

Returns the index number of cell’s text color. Note that number formats that display text in specific colors may override this setting.

“topborder”

Returns a code corresponding to the type of the top border of cell. The codes are: 0 for none, 1 for thin, 2 for medium, 3 for dashed, 4 for dotted, 5 for thick, 6 for double, and 7 for hairline.

“topbordercolor”

Returns the index number of the color of cell’s top border.

“type”

Returns b if cell is empty, 1 if cell contains text, and v if cell contains anything other than text.

“underline”

Returns 1 if the font in cell is underlined. Otherwise, it returns 0.

“valign”

Returns a code corresponding to cell’s vertical alignment. The codes are: 0 for top, 1 for center, 2 for bottom.

“width”

Returns the column width of cell, truncated to an integer. Each unit of column width is equal to the width of one character in the default font size.
Remarks
This function is included in Formula One for Java in order to be compatible with Excel, which in turn included it in order to be compatible with other worksheet formats, notably Lotus 1-2-3.
Developers will find it more efficient to extract this type of information using Java direct calls.

Example
This function returns $D$57:
CELL("address", D57)

See Also
INFO

CEILING

Description
Rounds a number up to the nearest multiple of a specified significance.

Syntax
CEILING ( number, significance )

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>The value to round.</td>
</tr>
<tr>
<td>significance</td>
<td>The multiple to which to round.</td>
</tr>
</tbody>
</table>

Remarks
Regardless of the sign of the number, the value is rounded up, away from zero. If number is an exact multiple of significance, no rounding occurs. If number or significance is non-numeric, the error #VALUE! is returned. When the arguments have opposite signs, the error #NUM! is returned.

Examples
This function returns 1.25:
CEILING(1.23459, .05)

This function returns 150:
CEILING(148.24, 2)

See Also
EVEN, FLOOR, INT, ODD, ROUND, TRUNC
**CHAR**

**Description**
Returns a character that corresponds to the supplied Unicode value.

**Syntax**
CHAR (number)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>A value between 1 and 65535 that specifies an Unicode character.</td>
</tr>
</tbody>
</table>

**Remarks**
Numeric code and associated text are defined according to the operating system’s native character set. Since the Java platform is 100% Unicode, the characters returned by CHAR come from the Unicode character set regardless of the underlying platform.

In non-Java applications such as Formula One ActiveX and Microsoft Excel, CHAR often returns different results from those returned by Formula One for Java. Java applications consistently return results from Unicode, its native character set; non-Java applications may access and return results from character sets other than Unicode.

When CHAR is called with strings containing what is commonly called “plain ASCII” text (characters with values from 32 to 126), the results are the same in Formula One for Java, Formula One ActiveX, and Microsoft Excel.

---

**CHIDIST**

**Description**
Computes the “complementary” version of the chi-square distribution.

**Syntax**
CHIDIST (x, df)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>The value at which the function will be evaluated. It must be larger than zero and smaller than 1e10.</td>
</tr>
<tr>
<td>df</td>
<td>An integer indicating the number of degrees of freedom. It must be larger than zero. Decimal values will be rounded down to the nearest integer.</td>
</tr>
</tbody>
</table>

**Equation**

\[
\frac{1}{2^{v/2} \Gamma(v/2)} \int_0^\infty t^{(v/2)-1} e^{-t/2} dt = P(v/2, x^2/2)
\]

**Examples**
This function returns 0.476258754:

CHIDIST(9.6,10)

This function returns 0.951229425:

CHIDIST(0.1, 2)
CHIINV

Description
Computes the inverse of the “complementary” chi-square distribution.

Syntax
CHIINV (probability, df)

Argument | Description
---|---
probability | The probability, as obtained from the CHIDIST function. It must be a number between zero and 1 that represents a probability.
df | An integer indicating the number of degrees of freedom. It must be larger than zero and smaller than 1e10.

Examples
This function returns 11.0705:

CHIINV(0.05,5)

See Also
CHIDIST, CHITEST

CHITEST

Description
Computes the probability of the distribution against expected values.

Syntax
CHITEST (actual, expected)

Argument | Description
---|---
actual | The actual test values, in the form of a range reference or array constant.
expected | The expected test values, in the form of a range reference or array constant.

Actual and expected must contain the same potential numbers of values. Text and logical values are ignored, along with the number they are paired up with. That is, when a number from actual is paired up with text from expected, the entire pair is ignored.

Remarks
Before calculating the probability, CHITEST must determine the number of degrees of freedom (df), which is used to calculate the probability outcome. To calculate df, CHITEST uses the actual argument:

- When actual is an array constant, df is one less than the number of elements in the array.
- When actual is a range reference, df is the product of (rows - 1)(columns - 1).
You can see that empty cells (or cells with text or logical values) in the actual argument’s range reference will affect df, which in turn affects the outcome of CHITEST.

**Equation**

\[ \chi^2 = \sum_i \frac{(N_i - n_i)^2}{n_i} \]

where \( N_i \) is the actual value and \( n_i \) is the expected value.

**Examples**

The following examples use this worksheet.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>14</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>78</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>82</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>8</td>
<td>84</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>9</td>
<td>88</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>11</td>
<td>91</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>12</td>
<td>99</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>14</td>
<td>103</td>
<td>1</td>
</tr>
</tbody>
</table>

This function returns 0.71:

\[ \text{CHITEST(B1:B10, A1:A10)} \]

This function returns 0:

\[ \text{CHITEST(C1:C10, A1:A10)} \]

**See Also**

FTEST, TTEST, ZTEST

---

**CHOOSE**

**Description**

Returns a value from a list of numbers based on the index number supplied.

**Syntax**

\[ \text{CHOOSE} \left( \text{index, item_list} \right) \]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>index</td>
<td>A number that refers to an item in item_list.</td>
</tr>
<tr>
<td>item_list</td>
<td>A list of numbers, formulas, or text separated by commas. This argument can also be a range reference. You can specify as many as 29 items in the list.</td>
</tr>
</tbody>
</table>
Remarks

Index can be a cell reference; index can also be a formula that returns any value from 1 to 29. If index is less than 1 or greater than the number of items in item_list, #VALUE! is returned. If index is a fractional number, it is truncated to an integer.

Examples

This function returns Q2:

CHOOSE(2, "Q1", "Q2", "Q3", "Q4")

This function returns the average of the contents of range A1:A10:

AVERAGE(CHOOSE(1, A1:A10, B1:B10, C1:C10))

See Also
INDEX (non-array type)

---

**CLEAN**

**Description**
Removes all nonprintable characters from the supplied text.

**Syntax**
CLEAN ( text )

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>Any worksheet information.</td>
</tr>
</tbody>
</table>

**Remarks**

Text that is imported from another environment may require this function.

**Examples**

This function returns Payments Due because the character returned by CHAR (8) is nonprintable:

CLEAN("Payments " & CHAR(8) & "Due")

**See Also**
CHAR, TRIM

---

**CODE**

**Description**
Returns a Unicode value of the first character of the supplied string.

**Syntax**
CODE ( text )

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>Any string.</td>
</tr>
</tbody>
</table>

**Remarks**

Numeric code and associated text are defined according to the operating system’s native character set. Since the Java platform is 100% Unicode, the numbers returned by CODE come from the Unicode character set regardless of the underlying platform.
In non-Java applications such as Formula One ActiveX and Microsoft Excel, CODE will often return different results from those returned by Formula One for Java. Java applications consistently return results from Unicode, its native character set; non-Java applications may access and return results from character sets other than Unicode.

When CODE is called with strings containing what is commonly called “plain ASCII” text (characters with values from 32 to 126), the results will be the same in Formula One for Java, Formula One ActiveX, and Microsoft Excel.

### COLUMN

**Description**

Returns the column number of the supplied reference.

**Syntax**

\[
\text{COLUMN ( reference )}
\]

**Table:**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>reference</td>
<td>A reference to a cell or range. Omitting the argument returns the number of the column in which COLUMN is placed.</td>
</tr>
</tbody>
</table>

**Examples**

This function returns 2:

\[
\text{COLUMN(B3)}
\]

This function returns 4 if the function is entered in cell D2:

\[
\text{COLUMN()}
\]

**See Also**

COLUMNS, ROW

### COLUMNS

**Description**

Returns the number of columns in a range reference.

**Syntax**

\[
\text{COLUMNS ( range )}
\]

**Table:**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>range</td>
<td>A reference to a range of cells.</td>
</tr>
</tbody>
</table>

**Example**

This function returns 4:

\[
\text{COLUMNS(A1:D5)}
\]

**See Also**

COLUMN, ROWS
COMBIN

Description
Computes the number of combinations possible by taking \( k \) items at a time from a pool of \( n \), when order in the sample taken is not important.

Syntax
\[
\text{COMBIN} \left( \text{number, chosen} \right)
\]

Remarks
For an example of the use of this function, say we have a box containing four different items. The number of combinations possible taking out 2 items at a time is 6; that is, the six combinations of items 1-2, 1-3, 1-4, 2-3, 2-4, and 3-4.

This function is similar to the PERMUT function, except that PERMUT requires the samples to be ordered, while COMBIN takes samples in any order.

Equation
\[
\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{n(n-1)\ldots(n-k+1)}{1\times2\ldots k}
\]
where \( n \) is the total number of items and \( k \) is the number of items taken at a time.

Examples
This function returns 6:
\[
\text{COMBIN}(4, 2)
\]
This function returns 252:
\[
\text{COMBIN}(10, 5)
\]

See Also
GAMMALN, PERMUT
COMPLEX

Description
Creates a complex number. The result is text in the form “a+bi” or “a+bj”) where “a” and “b” are from the first two arguments.

Syntax
```
COMPLEX ( real_number, imaginary_number [, suffix])
```

Examples
This function returns 2+3i:
```
COMPLEX(2,3)
```
This function returns 2+3j:
```
COMPLEX(2,3,"j")
```

See Also
IMABS, IMAGINARY, IMARGUMENT, IMCONJUGATE, IMCOS, IMDIV, IMEXP, IMLN, IMLOG10, IMLOG2, IMPOWER, IMPRODUCT, IMREAL, IMSIN, IMSQRT, IMSUB, IMSUM

CONCATENATE

Description
Joins several text items into one item.

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

Examples

Remarks
The “&” operator can be used instead of CONCATENATE to join text items.
Examples

The following example returns “Sale Price.” It is the same as typing “Sale” & “Price”:

```
CONCATENATE ("Sale ", "Price")
```

Suppose in an inventory worksheet, C2 contains “extruder1”, C5 contains “gaskets”, and C8 contains the number 15. The following example returns “Inventory currently holds 15 gaskets for extruder1.”:

```
CONCATENATE ("Inventory currently holds ", C8, " ", C5," for ", C2)
```

See Also

COLUMN, ROWS

CONFIDENCE

Description

Computes the confidence interval, assuming a normal distribution. The confidence interval is the portion of a value that appears after the +/-, for example, 10 +/- 2.1.

Syntax

```
CONFIDENCE (α , σ , n)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>A number between zero and one indicating confidence level. For example, 0.05 represents a 95% confidence level. The relation of confidence level to α is: Confidence level (as a percent) = 100 * (1 - α).</td>
</tr>
<tr>
<td>σ</td>
<td>A number representing the population standard deviation for the range. It must be larger than zero.</td>
</tr>
<tr>
<td>n</td>
<td>An integer larger than 1 indicating the sample size.</td>
</tr>
</tbody>
</table>

Equation

```
K = c × \frac{σ}{\sqrt{n}}
```

where

```
c = NORMSINV\left(1 - \frac{α}{2}\right)
```

Examples

This function returns 0.69295089:

```
CONFIDENCE(0.05, 2.5, 50)
```

See Also

NORMDIST, NORMINV, NORMSDIST, NORMSINV, STANDARDIZE

CONVERT

Description

Converts a numeric value from one set of units to another.

Syntax

```
CONVERT ( number, from_units, to_units)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>The value to be converted, in terms of from_units.</td>
</tr>
</tbody>
</table>

For the US method of distance measurement, the “international foot” is used. One “international foot” = 0.999998 “survey foot.”

Length measurements are based on the “international” system where 1 inch = 2.540 centimeters. The international system replaces the older “survey” system where 1.0 meter = 39.370 inches. The difference amounts to about 3 millimeters per mile.

Microsoft Excel uses the term “pica” as a unit of length where “computer point” is actually meant. “Computer point” also differs from a “printer’s point,” which is approximately 1/72 inch. According to international standards, a “pica” is a unit of length that equals 1/6 inch.

Year is based on nominal average year of 365.25 days (1 leap year every 4), not the true solar year (which is slightly shorter). To maintain Excel compatibility, related conversions involving years are factored based on the nominal average year.

These tables show the abbreviations you can use in the from_units and to_units arguments.

### Weight and mass abbreviations

<table>
<thead>
<tr>
<th>Unit</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gram</td>
<td>g</td>
</tr>
<tr>
<td>Slug</td>
<td>sg</td>
</tr>
<tr>
<td>Pound mass (avoirdupois)</td>
<td>lbm</td>
</tr>
<tr>
<td>U (atomic mass unit)</td>
<td>u</td>
</tr>
<tr>
<td>Ounce mass (avoirdupois)</td>
<td>ozm</td>
</tr>
</tbody>
</table>

### Distance abbreviations

<table>
<thead>
<tr>
<th>Unit</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meter</td>
<td>m</td>
</tr>
<tr>
<td>Statute mile</td>
<td>mi</td>
</tr>
<tr>
<td>Nautical Mile</td>
<td>Nmi</td>
</tr>
<tr>
<td>Inch</td>
<td>in</td>
</tr>
<tr>
<td>Foot</td>
<td>ft</td>
</tr>
<tr>
<td>Yard</td>
<td>yd</td>
</tr>
<tr>
<td>Angstrom</td>
<td>ang</td>
</tr>
<tr>
<td>Pica (1/72 in.)</td>
<td>Pica</td>
</tr>
</tbody>
</table>
### Time abbreviations

<table>
<thead>
<tr>
<th>Unit</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Year</td>
<td>yr</td>
</tr>
<tr>
<td>Day</td>
<td>day</td>
</tr>
<tr>
<td>Hour</td>
<td>hr</td>
</tr>
<tr>
<td>Minute</td>
<td>mn</td>
</tr>
<tr>
<td>Second</td>
<td>sec</td>
</tr>
</tbody>
</table>

### Pressure abbreviations

<table>
<thead>
<tr>
<th>Unit</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pascal</td>
<td>Pa</td>
</tr>
<tr>
<td>Atmosphere</td>
<td>atm</td>
</tr>
<tr>
<td>mm of Mercury</td>
<td>mmHg</td>
</tr>
</tbody>
</table>

### Force abbreviations

<table>
<thead>
<tr>
<th>Unit</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Newton</td>
<td>N</td>
</tr>
<tr>
<td>Dyne</td>
<td>dyn</td>
</tr>
<tr>
<td>Pound force</td>
<td>lbf</td>
</tr>
</tbody>
</table>

### Energy abbreviations

<table>
<thead>
<tr>
<th>Unit</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joule</td>
<td>J</td>
</tr>
<tr>
<td>Erg</td>
<td>e</td>
</tr>
<tr>
<td>Thermodynamic calorie</td>
<td>c</td>
</tr>
<tr>
<td>IT calorie</td>
<td>cal</td>
</tr>
<tr>
<td>Electron volt</td>
<td>eV</td>
</tr>
<tr>
<td>Horsepower-hour</td>
<td>HPh</td>
</tr>
<tr>
<td>Watt-hour</td>
<td>Wh</td>
</tr>
<tr>
<td>Foot-pound</td>
<td>flb</td>
</tr>
<tr>
<td>BTU</td>
<td>BTU</td>
</tr>
</tbody>
</table>

### Power abbreviations

<table>
<thead>
<tr>
<th>Unit</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horsepower</td>
<td>HP</td>
</tr>
<tr>
<td>Watt</td>
<td>W</td>
</tr>
</tbody>
</table>

### Magnetism abbreviations

<table>
<thead>
<tr>
<th>Unit</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tesla</td>
<td>T</td>
</tr>
<tr>
<td>Gauss</td>
<td>ga</td>
</tr>
<tr>
<td>Temperature abbreviations</td>
<td></td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Degree Celsius</td>
<td>C</td>
</tr>
<tr>
<td>Degree Fahrenheit</td>
<td>F</td>
</tr>
<tr>
<td>Degree Kelvin</td>
<td>K</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Liquid measure abbreviations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teaspoon</td>
</tr>
<tr>
<td>Tablespoon</td>
</tr>
<tr>
<td>Fluid ounce</td>
</tr>
<tr>
<td>Cup</td>
</tr>
<tr>
<td>Pint</td>
</tr>
<tr>
<td>Quart</td>
</tr>
<tr>
<td>Gallon</td>
</tr>
<tr>
<td>Liter</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Prefix character abbreviations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exa-</td>
</tr>
<tr>
<td>Peta-</td>
</tr>
<tr>
<td>Tera-</td>
</tr>
<tr>
<td>Giga-</td>
</tr>
<tr>
<td>Mega-</td>
</tr>
<tr>
<td>Kilo-</td>
</tr>
<tr>
<td>Hecto-</td>
</tr>
<tr>
<td>Dekao-</td>
</tr>
<tr>
<td>Deci-</td>
</tr>
<tr>
<td>Centi-</td>
</tr>
<tr>
<td>Milli-</td>
</tr>
<tr>
<td>Micro-</td>
</tr>
<tr>
<td>Nano-</td>
</tr>
<tr>
<td>Pico-</td>
</tr>
<tr>
<td>Femto-</td>
</tr>
<tr>
<td>Atto-</td>
</tr>
</tbody>
</table>
Examples

This function returns 1.093513298:

\[
\text{CONVERT(1, "m", "yd")}
\]

This function returns 0.621371192237334:

\[
\text{CONVERT(1, "km", "mi")}
\]

CORREL

Description

Computes the correlation coefficient for two sets of numbers, paired up one-to-one.

Note

The CORREL function is exactly the same as the PEARSON function. We provide both in order to be compatible with all the Microsoft Excel functions.

Syntax

\[
\text{CORREL (array1, array2)}
\]

Remarks

The correlation coefficient is a number between -1 and 1 (inclusive) that measures the “relatedness” of the numbers in the samples. A coefficient of 1 indicates a direct relationship in which all points are linearly related on a line with positive slope. A coefficient of -1 indicates an inverse relationship in which a large value in the first argument pairs with a small value in the second argument. A coefficient of 0 indicates no relationship between the pairs of values, or complete randomness.

This function pairs up the numbers in the two ranges by moving left-to-right through each sequential row.

Equation

\[
r = \frac{\sum_{i} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i} (x_i - \bar{x})^2} \sqrt{\sum_{i} (y_i - \bar{y})^2}}
\]

where \( \bar{x} \) is the mean of the \( x_i \)’s and \( \bar{y} \) is the mean of the \( y_i \)’s.
Examples

The following examples use this worksheet.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>10</td>
<td>11</td>
</tr>
</tbody>
</table>

This function returns 1:

\[
\text{CORREL}(A1:A3,B1:B3)
\]

This function returns 0.991117:

\[
\text{CORREL}(A1:B2,C1:C4)
\]

See Also

COVAR, PEARSON

---

**COS**

Description

Returns the cosine of an angle.

Syntax

\[
\text{COS}(\text{number})
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>The angle in radians. If the angle is in degrees, convert the angle to radians by multiplying the angle by PI()/180.</td>
</tr>
</tbody>
</table>

Examples

This function returns .126:

\[
\text{COS}(1.444)
\]

This function returns .28:

\[
\text{COS}(5)
\]

See Also

ACOS, ASINH, ATANH, COSH, PI
**COSH**

**Description**
Returns the hyperbolic cosine of a number.

**Syntax**
\[ \text{COSH ( number )} \]

**Examples**
- This function returns 4.14:
  \[ \text{COSH(2.10)} \]
- This function returns 1.03:
  \[ \text{COSH(.24)} \]

**See Also**
ASINH, ATANH, COS

---

**COUNT**

**Description**
Returns the number of values in the supplied list.

**Syntax**
\[ \text{COUNT ( value_list )} \]

**Remarks**
COUNT only numerates numbers or numerical values such as logical values, dates, or text representations of dates. If you supply a range, only numbers and numerical values in the range are counted. Empty cells, logical values, text, and error values in the range are ignored.

**Examples**
- This function returns 2:
  \[ \text{COUNT(5, 6, "Q2")} \]
- This function returns 3:
  \[ \text{COUNT("03/06/94", "06/21/94", "10/19/94")} \]

**See Also**
AVERAGE, COUNTA, SUM
COUNTA

Description
Returns the number of nonblank values in the supplied list.

Syntax
COUNTA ( expression_list )

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>expression_list</td>
<td>A list of expressions. As many as 30 expressions can be included in the list.</td>
</tr>
</tbody>
</table>

Remarks
COUNTA returns the number of cells that contain data in a range. Null values (""") are counted, but references to empty cells are ignored.

Examples
This function returns 4:

\[
\text{COUNTA}(32, 45, "Earnings", ")
\]

This function returns 0 when the specified range contains empty cells:

\[
\text{COUNTA(C38:C40)}
\]

See Also
AVERAGE, COUNT, PRODUCT, SUM

COUNTBLANK

Description
Counts the number of empty cells in a specified range.

Syntax
COUNTBLANK (range)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>range</td>
<td>A range reference or the name of a named range reference.</td>
</tr>
</tbody>
</table>

Remarks
COUNTBLANK will not count cells containing hidden zero values, spaces, and other entries that make the cell appear empty. This function counts all cells that COUNTA does not count.

Examples
The following examples use the worksheet below.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This function returns 2:

\[
\text{COUNTBLANK(A1:D2)}
\]

See Also
COUNT, COUNTA
COUNTIF

Description
Returns the number of cells within a range that meet the given criteria.

Syntax
COUNTIF ( range, criteria)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>range</td>
<td>Range of cells you want to count.</td>
</tr>
<tr>
<td>criteria</td>
<td>Number, expression, or text that defines which cells are counted.</td>
</tr>
</tbody>
</table>

Example
The following example uses this worksheet.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>3</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>7</td>
<td>8</td>
<td>3</td>
</tr>
</tbody>
</table>

This function returns 4:
COUNTIF(A1:D3, 3)

See Also
AVERAGE, COUNTA, SUM, SUMIF

COUPDAYBS

Description
Computes the number of days from the coupon date previous to the settlement date and the settlement date.

Syntax
COUPDAYBS ( settlement, maturity, frequency [, calendar_type])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>settlement</td>
<td>The date when the security is traded to the buyer. Dates in the argument list must be in the form of a serial number or text.</td>
</tr>
<tr>
<td>maturity</td>
<td>The date the security expires and the remaining amount is paid to the investor. It must be later than settlement. Dates in the argument list must be in the form of a serial number or text.</td>
</tr>
<tr>
<td>frequency</td>
<td>The number of interest payments per year. See “The frequency Argument” on page 18 for more information.</td>
</tr>
<tr>
<td>[calendar_type]</td>
<td>Optional. One of five methods of counting days for computing interest. See “The calendar_type Argument” on page 19 for more information.</td>
</tr>
</tbody>
</table>

Remarks
This function truncates all arguments to integers.
### COUPDAYS

**Description**
Computes the number of days in the coupon period containing the settlement date.

**Syntax**
```
COUPDAYS (settlement, maturity, frequency [, calendar_type])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>settlement</td>
<td>The date when the security is traded to the buyer. Dates in the argument list must be in the form of a serial number or text.</td>
</tr>
<tr>
<td>maturity</td>
<td>The date the security expires and the remaining amount is paid to the investor. It must be later than settlement. Dates in the argument list must be in the form of a serial number or text.</td>
</tr>
<tr>
<td>frequency</td>
<td>The number of interest payments per year. See “The frequency Argument” on page 18 for more information.</td>
</tr>
<tr>
<td>[calendar_type]</td>
<td>Optional. One of five methods of counting days for computing interest. See “The calendar_type Argument” on page 19 for more information.</td>
</tr>
</tbody>
</table>

**Remarks**
This function truncates all arguments to integers.

**Examples**
This function returns 180:
```
COUPDAYS("1/25/93","8/31/94",2,0)
```

This function returns 181:
```
COUPDAYS("1/25/93","8/31/94",2,1)
```

**See Also**
COUPDAYBS, COUPDAYSNC, COUPNCD, COUPNUM, COUPPCD
COUPDAYSNC

Description
Computes the number of days from the settlement date to the next coupon date.

Syntax
COUPDAYSNC (settlement, maturity, frequency [, calendar_type])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>settlement</td>
<td>The date when the security is traded to the buyer. Dates in the argument list must be in the form of a serial number or text.</td>
</tr>
<tr>
<td>maturity</td>
<td>The date the security expires and the remaining amount is paid to the investor. It must be later than settlement. Dates in the argument list must be in the form of a serial number or text.</td>
</tr>
<tr>
<td>frequency</td>
<td>The number of interest payments per year. See “The frequency Argument” on page 18 for more information.</td>
</tr>
<tr>
<td>[calendar_type]</td>
<td>Optional. One of five methods of counting days for computing interest. See “The calendar_type Argument” on page 19 for more information.</td>
</tr>
</tbody>
</table>

Remarks
This function truncates all arguments to integers.

Examples
This function returns 35:
COUPDAYSNC("1/25/93","8/31/94",2,0)

This function returns 34:
COUPDAYSNC("1/25/93","8/31/94",2,1)

See Also
COUPDAYBS, COUPDAYS, COUPNCD, COUPNUM, COUPPCD

COUPNCD

Description
Computes the coupon date that follows the settlement date. If the coupon date falls on the settlement date, this function will display the next coupon date.

Syntax
COUPNCD (settlement, maturity, frequency [, calendar_type])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>settlement</td>
<td>The date when the security is traded to the buyer. Dates in the argument list must be in the form of a serial number or text.</td>
</tr>
<tr>
<td>maturity</td>
<td>The date the security expires and the remaining amount is paid to the investor. It must be later than settlement. Dates in the argument list must be in the form of a serial number or text.</td>
</tr>
<tr>
<td>frequency</td>
<td>The number of interest payments per year. See “The frequency Argument” on page 18 for more information.</td>
</tr>
</tbody>
</table>
Remarks
This function displays the date as a serial number unless the cell is formatted to
display dates. This function truncates all arguments to integers.

Examples
This function returns 2/28/93:

\[ \text{COUPNCD("1/25/93","8/31/94",2,0)} \]

This function returns 8/31/93:

\[ \text{COUPNCD("1/25/93","8/31/94",1,0)} \]

See Also
COUPDAYBS, COUPDAYS, COUPDAYSNC, COUPNUM, COUPPCD

COUPNUM

Description
Computes the number of coupons between settlement and maturity. Maturity counts
as a coupon date.

Syntax
COUPNUM ( settlement, maturity, frequency [, calendar_type])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>settlement</td>
<td>The date when the security is traded to the buyer. Dates in the argument list must be in the form of a serial number or text.</td>
</tr>
<tr>
<td>maturity</td>
<td>The date the security expires and the remaining amount is paid to the investor. It must be later than settlement. Dates in the argument list must be in the form of a serial number or text.</td>
</tr>
<tr>
<td>frequency</td>
<td>The number of interest payments per year. See “The frequency Argument” on page 18 for more information.</td>
</tr>
<tr>
<td>[calendar_type]</td>
<td>Optional. One of five methods of counting days for computing interest. See “The calendar_type Argument” on page 19 for more information.</td>
</tr>
</tbody>
</table>

Remarks
This function rounds results up to the nearest integer, so if it finds 3 coupon periods plus 10 days, the result will be 4. This function truncates all arguments to integers.

Examples
This function returns 4:

\[ \text{COUPNUM("1/25/93","8/31/94",2,0)} \]
This function returns 7:

\[
\text{COUPNUM("1/25/93","8/31/94",4,0)}
\]

See Also

COUPDAYBS, COUPDAYS, COUPDAYSNC, COUPNCD, COUPPCD

COUPPCD

Description

Computes the last coupon date before the settlement date. If the coupon date falls on the settlement date, this function will display the settlement date.

Syntax

\[
\text{COUPPCD ( settlement, maturity, frequency [, calendar_type])}
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>settlement</td>
<td>The date when the security is traded to the buyer. Dates in the argument list must be in the form of a serial number or text.</td>
</tr>
<tr>
<td>maturity</td>
<td>The date the security expires and the remaining amount is paid to the investor. It must be later than settlement. Dates in the argument list must be in the form of a serial number or text.</td>
</tr>
<tr>
<td>frequency</td>
<td>The number of interest payments per year. See “The frequency Argument” on page 18 for more information.</td>
</tr>
<tr>
<td>[calendar_type]</td>
<td>Optional. One of five methods of counting days for computing interest. See “The calendar_type Argument” on page 19 for more information.</td>
</tr>
</tbody>
</table>

Remarks

This function displays the date as a serial number unless the cell is formatted to display dates. This function truncates all arguments to integers.

Examples

This function returns 8/31/92:

\[
\text{COUPPCD("1/25/93","8/31/94",2,0)}
\]

This function returns 11/30/92:

\[
\text{COUPPCD("1/25/93","8/31/94",4,0)}
\]

See Also

COUPDAYBS, COUPDAYS, COUPDAYSNC, COUPNCD, COUPNUM

COVAR

Description

Computes the covariance for pairwise numbers in two arrays.

Syntax

\[
\text{COVAR (arrayX, arrayY)}
\]
Chapter 3   A-Z Function Reference  81

Remarks

This function pairs up the numbers in the two ranges by moving left-to-right through each sequential row.

Equation

\[
\text{COVAR}(x,y) = \frac{1}{n} \sum_{j=1}^{n} (x_j - \overline{x})(y_j - \overline{y})
\]

Please note that different statistics methods divide by \( n \) or by \( n - 1 \). We chose to divide by \( n \) in order to be consistent with Microsoft Excel.

Examples

This function returns 4:

\[
\text{COVAR}({1,2,3,4,5},{2,4,6,8,10})
\]

See Also

CORREL

---

**CRITBINOM**

Description

Computes the smallest number of successes that will result in the cumulative BINOMDIST function giving an answer larger than a specified criteria value.

Syntax

\[
\text{CRITBINOM}(\text{trials}, \text{probability}, \text{alpha})
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>trials</strong></td>
<td>An integer larger than 0 describing the total number of attempts in the test.</td>
</tr>
<tr>
<td><strong>probability</strong></td>
<td>A value between 0 and 1 that represents the chance of success on an individual attempt. For example, the probability of heads in a coin toss is considered to be 0.6; the probability of getting &quot;snake eyes&quot; (two ones) when throwing a pair of dice is 1/36.</td>
</tr>
<tr>
<td><strong>alpha</strong></td>
<td>A value between 0 and 1 that represents the overall criteria value. When the result of CRITBINOM is put into BINOMDIST, the value obtained from BINOMDIST will be equal to or larger than alpha.</td>
</tr>
</tbody>
</table>

Examples

This function returns 492:

\[
\text{CRITBINOM}(1000,0.5,0.3)
\]
This function returns 500:

```java
CRITBINOM(1000, 0.5, 0.5)
```

**See Also**

BINOMDIST, COMBIN

---

**CUMIPMT**

**Description**

Computes the cumulative interest for the specified period.

**Syntax**

```java
CUMIPMT(rate, n_periods, PV, start_period, end_period, type)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>rate</code></td>
<td>The interest rate. The <code>rate</code> should be adjusted to reflect the period length. For example, if you have a mortgage with an annual rate of 9%, but pay monthly, the appropriate rate is .0075 (.09/12 months).</td>
</tr>
<tr>
<td><code>n_periods</code></td>
<td>The total number of payment periods. Decimal values will be truncated to integers.</td>
</tr>
<tr>
<td><code>PV</code></td>
<td>The present value or principal amount of the loan.</td>
</tr>
<tr>
<td><code>start_period</code></td>
<td>The first period for accumulating interest; payment periods start with 1. Decimal values will be truncated to integers.</td>
</tr>
<tr>
<td><code>end_period</code></td>
<td>The last period of the calculation. If the calculation is based on a 30 year mortgage paid monthly, the maximum number would be 360. Decimal values will be truncated to integers.</td>
</tr>
<tr>
<td><code>type</code></td>
<td>Indicates whether payment is due at the beginning or end of the period. Enter 0 for the end of the period (typical for most consumer loans and mortgages) or 1 for the beginning of the period.</td>
</tr>
</tbody>
</table>

**Remarks**

This function is useful for calculating the anticipated interest on a home mortgage in a calendar year for tax purposes.

The result (payment to interest) displays as a negative number because it reflects “cash flow out” in Present Value nomenclature.

**Examples**

This function returns -5177.27:

```java
CUMIPMT(0.009166667, 60, 17000, 1, 60, 0)
```

This function returns -309.71:

```java
CUMIPMT(0.009166667, 60, 17000, 1, 2, 0)
```

**See Also**

FV, CUMPRINC, IPMT, NPER, PMT, PPMT, PV, RATE
CUMPRINC

Description: Computes the cumulative principal paid for the specified period.

Syntax: CUMPRINC (rate, n_periods, PV, start_period, end_period, type)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rate</td>
<td>The interest rate. The rate should be adjusted to reflect the period length. For example, if you have a mortgage with an annual rate of 9%, but pay monthly, the appropriate rate is .0075 (.09/12 months).</td>
</tr>
<tr>
<td>n_periods</td>
<td>The total number of payment periods. Decimal values will be truncated to integers</td>
</tr>
<tr>
<td>PV</td>
<td>The present value or principal amount of the loan.</td>
</tr>
<tr>
<td>start_period</td>
<td>The first period for accumulating interest; payment periods start with 1. Decimal values will be truncated to integers.</td>
</tr>
<tr>
<td>end_period</td>
<td>The last period of the calculation. If the calculation is based on a 30 year mortgage paid monthly, the maximum number would be 360. Decimal values will be truncated to integers.</td>
</tr>
<tr>
<td>type</td>
<td>Indicates whether payment is due at the beginning or end of the period. Enter 0 for the end of the period (typical for most consumer loans and mortgages) or 1 for the beginning of the period.</td>
</tr>
</tbody>
</table>

Remarks: The result (payment for principal) displays as a negative number because it reflects “cash flow out” in Present Value nomenclature.

Examples: This function returns -8483.764676:

CUMIPMT(0.009166667,60,17000,1,34,0)

See Also: FV, CUMIPMT, IPMT, NPER, PMT, PPMT, PV, RATE

DATE

Description: Returns the serial number of the supplied date.

Syntax: DATE (year, month, day)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>year</td>
<td>A number from 1900 to 2078. If year is between 1920 to 2019, you can specify two digits to represent the year; otherwise specify all four digits.</td>
</tr>
</tbody>
</table>
### DATEDIF

**Description**
Calculates the number of days, months, or years between two specified dates.

**Syntax**
```
DATEDIF ( start, end, unit )
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>The start date. Dates in the argument list must be entered as text, enclosed in quotation marks.</td>
</tr>
<tr>
<td>end</td>
<td>The end date. It must be after start. Dates in the argument list must be entered as text, enclosed in quotation marks.</td>
</tr>
<tr>
<td>unit</td>
<td>A letter representing the time units you want the returned value in. The options are:</td>
</tr>
<tr>
<td></td>
<td>Y The number of complete years between start and end.</td>
</tr>
<tr>
<td></td>
<td>M The number of complete months between start and end.</td>
</tr>
<tr>
<td></td>
<td>D The number of days between start and end.</td>
</tr>
<tr>
<td></td>
<td>MD The difference between the days in start and end, ignoring the months and years of the dates.</td>
</tr>
<tr>
<td></td>
<td>YM The difference between the months in start and end, ignoring the days and years of the dates.</td>
</tr>
<tr>
<td></td>
<td>YD The difference between the years in start and end, ignoring the days and months of the dates.</td>
</tr>
</tbody>
</table>

**Examples**
This function returns 247:
```
DATEDIF("3/12/96","11/14/96","d")
```
This function returns 2:
DATEDIF("3/12/96","11/14/96","md")

See Also  DAY, DAYS360, EDATE, EOMONTH, MONTH, NETWORKDAYS, WORKDAY, YEAR

DATEVALUE

Description  Returns the serial number of a date supplied as a text string.

Syntax  DATEVALUE (text)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>A date in text format between January 1, 1900, and December 31, 2078. If you omit the year, the current year is used.</td>
</tr>
</tbody>
</table>

Examples  This function returns 34399:
DATEVALUE("3/6/94")

This function returns 35058:
DATEVALUE("12/25/95")

See Also  NOW, TIMEVALUE, TODAY

DAVERAGE

Description  Uses specified criteria to select records from a database, then computes the average of the numeric values in a specified field of the selected records.

Note  This is a database function, which has specific requirements that are different from most functions. For information and expanded argument descriptions, see “Database Functions” on page 14.

Syntax  DAVERAGE (database, field, criteria)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>database</td>
<td>A reference to a range containing data that the function searches.</td>
</tr>
<tr>
<td>field</td>
<td>The column within database that contains the data you want to average.</td>
</tr>
<tr>
<td>criteria</td>
<td>A reference to a range containing search criteria.</td>
</tr>
</tbody>
</table>
Examples
This example uses the example database and criteria ranges shown in “Database Functions” on page 14. This function returns 20.33:


See Also
DCOUNT, DCOUNTA, DGET, DMAX, DMIN, DPRODUCT, DSTDEV, DSTDEVP, DSUM, DVAR, DVARP

---

**DAY**

**Description**
Returns the day of the month that corresponds to the date represented by the supplied number.

**Syntax**

DAY ( serial_number )

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serial_number</td>
<td>A date represented as a serial number or as text (for example, 06-21-94 or 21-Jun-94).</td>
</tr>
</tbody>
</table>

**Examples**

This function returns 6:

DAY(34399)

This function returns 21:

DAY("06-21-94")

See Also
NOW, HOUR, MINUTE, MONTH, SECOND, TODAY, WEEKDAY, YEAR

---

**DAYS360**

**Description**
Returns the number of days between two dates based on a 360-day year (twelve 30-day months). Use this function to help compute payments if your accounting system is based on twelve 30-day months.

**Syntax**

DAYS360 ( start_date, end_date [, method ] )

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>start_date, end_date</td>
<td>The two dates between which you want to know the number of days.</td>
</tr>
</tbody>
</table>
Remarks

Start_date and end_date can be text strings using numbers to represent the month, day, and year (for example, “1/30/93” or “1-30-93”), or they can be serial numbers representing the dates.

If start_date occurs after end_date, DAYS360 returns a negative number.

If method is set to False and start_date is the 31st of a month, it becomes equal to the 30th of the same month. If end_date is the 31st of a month and start_date is less than the 30th of a month, the ending date becomes equal to the 1st of the next month, otherwise the ending date becomes equal to the 30th of the same month.

If method is set to True, start_dates or end_dates which occur on the 31st of a month become equal to the 30th of the same month.

Note To determine the number of days between two dates in a normal year, you can use normal subtraction. For example, “12/31/93” - “1/1/93” equals 364.

Example

This function returns 1:

DAYS360("1/30/93", "2/1/93")

DB

Description

Returns the real depreciation of an asset for a specific period of time using the fixed-declining balance method.

Syntax

DB (cost, salvage, life, period [, months])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cost</td>
<td>The initial cost of the asset.</td>
</tr>
<tr>
<td>salvage</td>
<td>The salvage value of the asset.</td>
</tr>
<tr>
<td>life</td>
<td>The number of periods in the useful life of the asset.</td>
</tr>
<tr>
<td>period</td>
<td>The period for which to calculate the depreciation. The time units used to determine period and life must match.</td>
</tr>
<tr>
<td>[months]</td>
<td>Optional. The number of months in the first year of the item’s life. If this argument is omitted, 12 is used.</td>
</tr>
</tbody>
</table>
DCOUNT

Description

Uses specified criteria to select records from a database, then counts the number of selected records. Can also count the number of selected records that have a number in the specified field.

Note

This is a database function, which has specific requirements that are different from most functions. For information and expanded argument descriptions, see “Database Functions” on page 14.

Syntax

DCOUNT( database, [field], criteria)

Argument Description

database A reference to a range containing data that the function searches.

[field] Optional. The column within database that contains the data you want to count. The function will only count the selected records that have a number in field. If this argument is omitted, the function will count all selected records.

criteria A reference to a range containing search criteria.

Examples

This example uses the example database and criteria ranges shown in “Database Functions” on page 14. This function returns 3:


This function returns 0:


See Also

DAVERAGE, DCOUNTA, DGET, DMAX,DMIN, DPRODUCT, DSTDEV, DSTDEVP, DSUM, DVAR, DVARP
**DCOUNTA**

**Description**
Uses specified criteria to select records from a database, then counts the number of selected records. Can also count the number of selected records for which a specified field is not blank.

**Note**
This is a database function, which has specific requirements that are different from most functions. For information and expanded argument descriptions, see “Database Functions” on page 14.

**Syntax**
DCOUNTA (database, [field,] criteria)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>database</td>
<td>A reference to a range containing data that the function searches.</td>
</tr>
<tr>
<td>[field]</td>
<td>Optional. The column within database that contains the data you want to count. The function will only count the selected records that have a number in field. If this argument is omitted, the function will count all selected records.</td>
</tr>
<tr>
<td>criteria</td>
<td>A reference to a range containing search criteria.</td>
</tr>
</tbody>
</table>

**Examples**
This example uses the example database and criteria ranges shown in “Database Functions” on page 14. This function returns 3:


**See Also**
DAVERAGE, DCOUNT, DGET, DMAX, DMIN, DPRODUCT, DSTDEV, DSTDEVP, DSUM, DVAR, DVARP

**DDB**

**Description**
Returns the depreciation of an asset for a specific period of time using the double-declining balance method or a declining balance factor you supply.

**Syntax**
DDB (cost, salvage, life, period [, factor])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cost</td>
<td>The initial cost of the asset.</td>
</tr>
<tr>
<td>salvage</td>
<td>The salvage value of the asset.</td>
</tr>
<tr>
<td>life</td>
<td>The number of periods in the useful life of the asset.</td>
</tr>
<tr>
<td>period</td>
<td>The period for which to calculate the depreciation. The time units used to determine period and life must match.</td>
</tr>
</tbody>
</table>
### Remarks
The double-declining balance method uses an accelerated rate where the highest depreciation occurs in the first period, decreasing in successive periods.

All arguments for this function must be positive numbers.

### Example
This function returns 1457.73:

$$DDB(10000, 1000, 7, 3)$$

### See Also
DB, SLN, SYD, VDB

### DEC2BIN

**Description**
Converts a decimal number (base 10) to a binary number (base 2).

**Syntax**
DEC2BIN ( integer [, places])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>integer</td>
<td>Any negative or positive whole number or zero.</td>
</tr>
<tr>
<td></td>
<td>For the purposes of this function, the integer entered must be small enough to convert within the limits of the target. Any integer between -512 and 511 may be used.</td>
</tr>
<tr>
<td>places</td>
<td>Optional number of characters to use in the output. If places is omitted, the function returns the number of spaces required to display the result.</td>
</tr>
<tr>
<td></td>
<td>The places argument can be used to 0-pad a positive number.</td>
</tr>
<tr>
<td></td>
<td>If the number is negative, places is ignored and 10 characters are returned.</td>
</tr>
</tbody>
</table>

**Remarks**
The result of this function is a text string.

**Examples**
This function returns 100000000:

```java
DEC2BIN(256)
```

This function returns 1000000000:

```java
DEC2BIN(-512)
```
This function returns 01010

DEC2BIN(10,5)

**See Also**
BIN2DEC, BIN2HEX, BIN2OCT, DEC2HEX, DEC2OCT, HEX2BIN, HEX2DEC, HEX2OCT, OCT2BIN, OCT2DEC, OCT2HEX

---

**DEC2HEX**

**Description**
Converts a decimal number (base 10) to an hexadecimal number (base 16).

**Syntax**

```
DEC2HEX ( integer [, places])
```

**Argument** | **Description**
--- | ---
integer | Any negative or positive whole number or zero.

For the purposes of this function, the integer entered must be small enough to convert within the limits of the target. Any integer between -549,755,813,888 to 549,755,813,887 may be used.

places | Optional number of characters to use in the output. If places is omitted, the function returns the number of spaces required to display the result.

The places argument can be used to 0-pad a positive number.

If the number is negative, places is ignored and 10 characters are returned.

**Remarks**
The result of this function is a text string.

**Examples**
This function returns 0000A:

```
DEC2HEX(10,5)
```

This function returns FFFFFFFFF6:

```
DEC2HEX(-10,5)
```

**See Also**
BIN2DEC, BIN2HEX, BIN2OCT, DEC2BIN, DEC2OCT, HEX2BIN, HEX2DEC, HEX2OCT, OCT2BIN, OCT2DEC, OCT2HEX

---

**DEC2OCT**

**Description**
Converts a decimal number (base 10) to an octal number (base 8).

**Syntax**

```
DEC2OCT ( integer [, places])
```
### DEGREES

**Description**
Converts a value in radians to degrees.

**Syntax**
```
DEGREES (radians)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>radians</td>
<td>A number in radians that you want to convert to degrees.</td>
</tr>
</tbody>
</table>

**Equation**
```
\[
\text{radians} \times \left( \frac{180}{\pi} \right)
\]
```

**Examples**
This function returns 360:
```
DEGREES(6.283185307)
```

This function returns 90:
```
DEGREES(1.570796327)
```

**See Also**
PI, RADIANS
DELTA

Description
Compares two specified values and returns 1 if they are equal, 0 if they are not.

Syntax
DELTA (number1 [, number2])

Remarks
This function can be used in place of the equals (=) operator when a numeric rather than logical result is needed (for example, when using aggregate functions such as SUM and COUNT).

This function is sometimes referred to as the Kronecker Delta function.

Examples
This function returns 0:
DELTA(6,7)

This function returns 1:
DELTA(0)

See Also
IF, GESTEP

DEVSQ

Description
Computes the square of the deviation of the numbers.

Syntax
DEVSQ (number_list)

Remarks
A list of up to 30 numbers separated by commas. The list may contain numeric values, cell references, range references, or array constants.

Text and logical values in cell references and arrays are ignored. Text entered in the argument list will be evaluated as a number, if possible; otherwise it will cause a #VALUE! error. Logical values entered into the argument list are evaluated as numeric 1 if TRUE and numeric 0 if FALSE.

Equation
\[ \sum_{i} (x_i - \bar{x})^2 \] where \( \bar{x} \) is the mean of the \( x_i \)’s.
DGET

Description
Uses specified criteria to select a single record from a database, then displays the data found in a specified field of the selected record.

Note
This is a database function, which has specific requirements that are different from most functions. For information and expanded argument descriptions, see “Database Functions” on page 14.

Syntax
DGET (database, field, criteria)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>database</td>
<td>A reference to a range containing data that the function searches.</td>
</tr>
<tr>
<td>field</td>
<td>The column within database that contains the data you want to display.</td>
</tr>
<tr>
<td>criteria</td>
<td>A reference to a range containing search criteria.</td>
</tr>
</tbody>
</table>

Remarks
If the function finds 0 records that match the criteria, it will display the #VALUE! error. If the function finds more than one record that match the criteria, it will display the #NUM! error.

Examples
This example uses the example database shown in “Database Functions” on page 14 and the criteria range below.

<table>
<thead>
<tr>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Salary 96</td>
</tr>
<tr>
<td>&gt;33000</td>
</tr>
</tbody>
</table>

This function returns 18940:

DGET(A1:D8,"Commissions 96",E1:E2)

See Also
DAVERAGE, DCOUNT, DCOUNTA, DMAX,DMIN, DPRODUCT, DSTDEV, DSTDEVP, DSUM, DVAR, DVARP
DISC

Description
Computes the discounted rate for a security.

Syntax
DISC (settlement, maturity, price, redemption [, calendar_type])

Argument | Description
--- | ---
settlement | The date when the security is traded to the buyer. Dates in the argument list must be in the form of a serial number or text. Decimal values are truncated to integers.
maturity | The date the security expires and the remaining amount is paid to the investor. It must be later than settlement. Dates in the argument list must be in the form of a serial number or text. Decimal values are truncated to integers.
price | The amount paid per $100 face value.
redemption | The security’s redemption value per $100 face value. This is the amount paid at maturity that is not part of any final coupon payment.
[calendar_type] | Optional. One of five methods of counting days for computing interest. See “The calendar_type Argument” on page 19 for more information.

Remarks
DISC understates the equivalent yield normally quoted on CDs and coupon bonds. See YIELDDISC for the yield rate comparison.

DISC, not YIELDDISC, is the complementary function to PRICEDISC.

Equation
\[
\frac{(\text{redemption} - \text{price})}{\text{redemption} \cdot \text{YEARFRAC}(\text{settlement}, \text{maturity}, \text{calendar_type})}
\]

Examples
This function returns 0.02024:

\[
\text{DISC("7/15/92","12/30/95",93,100)}
\]

See Also
PRICE, PRICEDISC, PRICEMAT, YIELD, YIELDDISC, YIELDMAT

DMAX

Description
Uses specified criteria to select records from a database, then displays the highest number value found in a specified field of the selected records.

Note
This is a database function, which has specific requirements that are different from most functions. For information and expanded argument descriptions, see “Database Functions” on page 14.
Syntax

DMAX (database, field, criteria)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>database</td>
<td>A reference to a range containing data that the function searches.</td>
</tr>
<tr>
<td>field</td>
<td>The column within database that contains the maximum value you want to find.</td>
</tr>
<tr>
<td>criteria</td>
<td>A reference to a range containing search criteria.</td>
</tr>
</tbody>
</table>

Remarks

If there are no numeric values in field in the selected records, DMAX will display 0.

Examples

This example uses the example database and criteria ranges shown in “Database Functions” on page 14. This function returns 32460:


See Also

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

DMIN

Description

Uses specified criteria to select records from a database, then displays the lowest number value found in a specified field of the selected records.

Note

This is a database function, which has specific requirements that are different from most functions. For information and expanded argument descriptions, see “Database Functions” on page 14.

Syntax

DMIN (database, field, criteria)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>database</td>
<td>A reference to a range containing data that the function searches.</td>
</tr>
<tr>
<td>field</td>
<td>The column within database that contains the minimum value you want to find.</td>
</tr>
<tr>
<td>criteria</td>
<td>A reference to a range containing search criteria.</td>
</tr>
</tbody>
</table>

Remarks

If there are no numeric values in field in the selected records, DMIN will display 0.

Examples

This example uses the example database and criteria ranges shown in “Database Functions” on page 14. This function returns 10320:


See Also

DAVERAGE, DCOUNT, DCOUNTA, DGET, DMIN, DPRODUCT, DSTDEV, DSTDEVP, DSUM, DVAR, DVARP
DOLLAR

Description
Returns the specified number as text, using the local currency format and the supplied precision.

Syntax
DOLLAR ( number [, precision])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>A number, a formula that evaluates to a number, or a reference to a cell that contains a number.</td>
</tr>
<tr>
<td>[precision]</td>
<td>Optional. A value representing the number of decimal places to the right of the decimal point. If this argument is omitted, 2 is used.</td>
</tr>
</tbody>
</table>

Note
“Local” currency refers to the currency format for the current system.

Remarks
Dollar will return the specified number format as text using currency format for the current system. If you wish to always convert to the US Dollar format, regardless of the language of your system, then use the USDOLLAR worksheet function.

US Example
When using a US setting in Windows, this function returns $1023.79:
DOLLAR(1023.789)
This function returns $500:
DOLLAR(495.301, -2)

UK Example
When using a British setting in Windows, this function returns £1023.8:
DOLLAR(1023.789)
This function returns £500:
DOLLAR(495.301, -2)

German Example
When using a German setting in Windows, this function returns 1023,8 DM
DOLLAR(1023.789)
This function returns 500 DM:
DOLLAR(495.301, -2)

See Also
FIXED, TEXT, VALUE, USDOLLAR
DOLLARDE

Description
Converts a dollar figure from fractional form to decimal form.

Syntax
DOLLARDE (dollar, denominator)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dollar</td>
<td>A dollar amount expressed as a fraction in the following manner:</td>
</tr>
<tr>
<td></td>
<td>The digits to the left of the decimal represent the number of</td>
</tr>
<tr>
<td></td>
<td>whole dollars. The digits to the right of the decimal point</td>
</tr>
<tr>
<td></td>
<td>represent the numerator of the fractional portion of the dollar</td>
</tr>
<tr>
<td>denominator</td>
<td>The denominator of the fractional portion of the dollar.</td>
</tr>
</tbody>
</table>

Remarks
This function is useful for converting dollar amounts involved in securities dealings, where dollar amounts are often expressed in fractions.

Examples
This function returns 25.75:
DOLLARDE(25.3,4)

See Also
DOLLARFR

DOLLARFR

Description
Converts a dollar figure from decimal form to fractional form.

Syntax
DOLLARFR (dollar, denominator)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dollar</td>
<td>A dollar amount expressed in the normal decimal form.</td>
</tr>
<tr>
<td>denominator</td>
<td>The denominator you want for the fractional portion of the dollar.</td>
</tr>
<tr>
<td></td>
<td>Decimal values are truncated to integers.</td>
</tr>
</tbody>
</table>

Remarks
This function is useful for converting dollar amounts involved in securities dealings, where dollar amounts are often expressed in fractions.

Examples
This function returns 25.1:
DOLLARFR(25.25,4)

See Also
DOLLARDE
DPRODUCT

**Description**
Uses specified criteria to select records from a database, then multiplies the numeric values in a specified field of the selected records.

**Note** This is a database function, which has specific requirements that are different from most functions. For information and expanded argument descriptions, see “Database Functions” on page 14.

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>database</td>
<td>A reference to a range containing data that the function searches.</td>
</tr>
<tr>
<td>field</td>
<td>The column within database that contains the data you want to multiply.</td>
</tr>
<tr>
<td>criteria</td>
<td>A reference to a range containing search criteria.</td>
</tr>
</tbody>
</table>

**Remarks**
If there are no numeric values in field in the selected records, DPRODUCT will display 0.

**Examples**
This example uses the example database and criteria ranges shown in “Database Functions” on page 14. This function returns 8160:


**See Also**
DAVERAGE, DCOUNT, DCOUNTA, DGET, DMAX,DMIN, DSTDEV, DSTDEVP, DSUM, DVAR, DVARP

DSTDEV

**Description**
Uses specified criteria to select records from a database, then computes the sample standard deviation of the numeric values in a specified field of the selected records.

**Note** This is a database function, which has specific requirements that are different from most functions. For information and expanded argument descriptions, see “Database Functions” on page 14.

**Syntax**
DSTDEV (database, field, criteria)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>database</td>
<td>A reference to a range containing data that the function searches.</td>
</tr>
</tbody>
</table>
**Remarks**

At least two values are required to compute sample standard deviation, so if there are fewer than two numeric values in *field* in the selected records, DSTDEV will display the #DIV/0! error.

**Examples**

This example uses the example database and criteria ranges shown in “Database Functions” on page 14. This function returns 3.5119:

```
```

**See Also**

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

---

**DSTDEVP**

**Description**

Uses specified criteria to select records from a database, then computes the population standard deviation of the numeric values in a specified field of the selected records.

**Note** This is a database function, which has specific requirements that are different from most functions. For information and expanded argument descriptions, see “Database Functions” on page 14.

**Syntax**

```
DSTDEVP ( database, field, criteria)
```

**Remarks**

At least one value is required to compute population standard deviation, so if there are fewer than one numeric values in *field* in the selected records, DSTDEVP will display the #DIV/0! error.

**Examples**

This example uses the example database and criteria ranges shown in “Database Functions” on page 14. This function returns 2.8674:

```
```
DSUM

Description
Uses specified criteria to select records from a database, then adds the numeric values in a specified field of the selected records.

Note This is a database function, which has specific requirements that are different from most functions. For information and expanded argument descriptions, see “Database Functions” on page 14.

Syntax
DSUM (database, field, criteria)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>database</td>
<td>A reference to a range containing data that the function searches.</td>
</tr>
<tr>
<td>field</td>
<td>The column within database that contains the data you want to add.</td>
</tr>
<tr>
<td>criteria</td>
<td>A reference to a range containing search criteria.</td>
</tr>
</tbody>
</table>

Remarks
If there are no numeric values in field in the selected records, DSUM will display 0.

Examples
This example uses the example database and criteria ranges shown in “Database Functions” on page 14. This function returns 73490:


See Also
DAVERAGE, DCOUNT, DCOUNTA, DGET, DMAX, DMIN, DPRODUCT, DSTDEV, DVAR, DVARP

DURATION

Description
Computes the Macaulay duration for a security, in years.

Syntax
DURATION (settlement, maturity, rate, yield, frequency [, calendar_type])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>settlement</td>
<td>The date when the security is traded to the buyer. Dates in the argument list must be in the form of a serial number or text. Numbers will be truncated to integers.</td>
</tr>
</tbody>
</table>
Equation

\[
\frac{(N - 1 + \frac{DSC}{E}) \times 100}{(1 + \frac{\text{yield}}{\text{freq}})^{N - 1 + \frac{DSC}{E}}} + \sum_{k=1}^{N} \left( \frac{100 \times \text{rate}}{\text{freq} \times \left(1 + \frac{\text{yield}}{\text{freq}}\right)^{k - 1 + \frac{DSC}{E}}} \times \left(1 + \frac{DSC}{E}\right) \right) \times \frac{1}{\text{freq}}
\]

...where the codes correspond to values you can compute using other functions, as shown in the following table.

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSC</td>
<td>Number of days from settlement to the next coupon period.</td>
<td>COUPDAYSNC</td>
</tr>
<tr>
<td>E</td>
<td>Number of days in the coupon period.</td>
<td>COUPDAYS</td>
</tr>
<tr>
<td>N</td>
<td>Number of coupons payable between settlement and maturity.</td>
<td>COUPNUM</td>
</tr>
</tbody>
</table>

Examples
This function returns 7.24649:
\[\text{DURATION}("3/17/89","3/17/99",0.07,0.08,2,1)\]

See Also
MDURATION

DVAR

Description
Uses specified criteria to select records from a database, then computes the sample variance of the numeric values in a specified field of the selected records.
Note This is a database function, which has specific requirements that are different from most functions. For information and expanded argument descriptions, see “Database Functions” on page 14.

**Syntax**

DVAR ( database, field, criteria)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>database</td>
<td>A reference to a range containing data that the function searches.</td>
</tr>
<tr>
<td>field</td>
<td>The column within database that contains the data you want to find the variance of.</td>
</tr>
<tr>
<td>criteria</td>
<td>A reference to a range containing search criteria.</td>
</tr>
</tbody>
</table>

**Remarks**

At least two values are required to compute sample variance, so if there are fewer than two numeric values in field in the selected records, DVAR will display the #DIV/0! error.

**Examples**

This example uses the example database and criteria ranges shown in “Database Functions” on page 14. This function returns 12.333:


**See Also**

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

**DVARP**

**Description**

Uses specified criteria to select records from a database, then computes the population variance of the numeric values in a specified field of the selected records.

Note This is a database function, which has specific requirements that are different from most functions. For information and expanded argument descriptions, see “Database Functions” on page 14.

**Syntax**

DVARP ( database, field, criteria)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>database</td>
<td>A reference to a range containing data that the function searches.</td>
</tr>
<tr>
<td>field</td>
<td>The column within database that contains the data you want to find the variance of.</td>
</tr>
<tr>
<td>criteria</td>
<td>A reference to a range containing search criteria.</td>
</tr>
</tbody>
</table>
Remarks
At least one value is required to compute population variance, so if there are fewer than one numeric values in field in the selected records, DVARP will display the #DIV/0! error.

Examples
This example uses the example database and criteria ranges shown in “Database Functions” on page 14. This function returns 8.2222:

\[
\text{DVARP(A1:D8,"Sales 96",B10:B11)}
\]

See Also
DAVERAGE, DCOUNT, DCOUNTA, DGET, DMAX, DMIN, DPRODUCT, DSTDEV, DSTDEVP, DSUM, DVAR

---

**EDATE**

Description
Finds a date a specified number of months before or after a given date. This is useful for computing dates associated with securities.

Syntax
EDA TE ( start_date, months )

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>start_date</td>
<td>Any date. Dates in the argument list must be in the form of a serial number or text.</td>
</tr>
<tr>
<td>months</td>
<td>The number of months after or before start_date. Enter a negative number for months before, a positive number for months after. Decimal values are truncated to integers.</td>
</tr>
</tbody>
</table>

Examples
This function returns 4/15/99:

\[
\text{EDATE("2/15/99",2)}
\]

See Also
EOMONTH

---

**EFFECT**

Description
Computes the effective annual interest rate, which adjusts the nominal rate to show the effect of compounding.

Syntax
EFFECT ( nominal_rate, periods )

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nominal_rate</td>
<td>The nominally quoted interest rate.</td>
</tr>
<tr>
<td>periods</td>
<td>The number of compounding periods per year. Decimal values are truncated to integers.</td>
</tr>
</tbody>
</table>
Equation

\[
\left(\frac{1 + \text{nominal\_rate}}{\text{periods}}\right)^{\text{periods}} - 1
\]

Examples

This function returns 0.0696:

\[\text{EFFECT(0.0675, 12)}\]

This function returns 0.0686:

\[\text{EFFECT(0.0675, 2)}\]

See Also

NOMINAL

---

**EOMONTH**

Description

Finds the last date in the month that is a specified number of months before or after a given date. This is useful for computing dates associated with securities.

Syntax

\[\text{EOMONTH (start\_date, months)}\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>start_date</td>
<td>Any date. Dates in the argument list must be in the form of a serial number or text.</td>
</tr>
<tr>
<td>months</td>
<td>The number of months after or before start_date. Enter a negative number for months before, a positive number for months after. Decimal values are truncated to integers.</td>
</tr>
</tbody>
</table>

Examples

This function returns 4/30/99:

\[\text{EOMONTH("2/15/99",2)}\]

See Also

EDATE

---

**ERF**

Description

Computes the “error” function.

Syntax

\[\text{ERF (lower\_limit [, upper\_limit])}\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lower_limit</td>
<td>The lower limit of the values across which you want to evaluate the function. Must be a real number.</td>
</tr>
<tr>
<td>upper_limit</td>
<td>The optional upper limit of the values across which you want to evaluate the function. Must be a real number. Omitting the upper limit calculates the error function between 0 and the lower limit.</td>
</tr>
</tbody>
</table>
Remarks
Called the “error” function because it follows the standard distribution of “errors” around the mean (the “bell-shaped curve”).
This function forms the basis of “normal” distribution functions.

Equation
\[ \text{ERF}(a, b) = \frac{2}{\pi} \int_a^b e^{-t^2} dt \]

Examples
This function returns 0.995322265:
\[ \text{ERF}(2) \]
This function returns 0.004655645:
\[ \text{ERF}(2, 3) \]

See Also
ERFC, NORMDIST, NORMSDIST

ERFC

Description
Computes the complementary “error” function.

Syntax
ERFC ( x )

Argument | Description
--- | ---
x | The value at which the function will be evaluated.

Remarks
Used to compute errors in larger cases of x, where the ERF function would lose digits while approaching 1 as a limit.

Equation
\[ \text{ERFC}(a) = \frac{2}{\pi} \int_a^\infty e^{-t^2} dt = 1 - \text{ERF}(a) \]

Examples
This function returns 0.157272:
\[ \text{ERFC}(1) \]

See Also
ERF, NORMDIST, NORMSDIST

ERROR.TYPE

Description
Returns a number corresponding to an error.

Syntax
ERROR.TYPE ( error_ref )
**Remarks**
The following error text or numbers can be returned by this function.

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>#NULL!</td>
</tr>
<tr>
<td>2</td>
<td>#DIV/0!</td>
</tr>
<tr>
<td>3</td>
<td>#VALUE!</td>
</tr>
<tr>
<td>4</td>
<td>#REF!</td>
</tr>
<tr>
<td>5</td>
<td>#NAME?</td>
</tr>
<tr>
<td>6</td>
<td>#NUM!</td>
</tr>
<tr>
<td>7</td>
<td>#N/A</td>
</tr>
<tr>
<td>#N/A</td>
<td>Other</td>
</tr>
</tbody>
</table>

**Example**
This function returns 2 if the formula in cell A1 attempts to divide by zero:
```
ERROR.TYPE(A1)
```

**See Also**
ISERR, ISERROR

---

**EVEN**

**Description**
Rounds the specified number up to the nearest even integer.

**Syntax**
```
EVEN ( number )
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>Any number, a formula that evaluates to a number, or a reference to a cell that contains a number.</td>
</tr>
</tbody>
</table>

**Examples**
This function returns 4:
```
EVEN(2.5)
```

This function returns 2032:
```
EVEN(2030.45)
```

**See Also**
CEILING, FLOOR, INT, ODD, ROUND, TRUNC
EXACT

Description
Compares two expressions for identical, case-sensitive matches. True is returned if the expressions are identical; False is returned if they are not.

Syntax
EXACT (expression1, expression2)

Examples
This function returns True:
EXACT("Match", "Match")

This function returns False:
EXACT("Match", "match")

See Also
LEN, SEARCH

EXP

Description
Returns e raised to the specified power. The constant e is 2.71828182845904 (the base of the natural logarithm).

Syntax
EXP (number)

Examples
This function returns 12.18:
EXP(2.5)

This function returns 20.09:
EXP(3)

See Also
LINEST, LOG
### EXPONDIST

**Description**
Computes the exponential distribution, which can be used to determine the amount of time between random events.

**Syntax**
EXPONDIST (x, lambda, cumulative)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>The value at which the function will be evaluated. It must be larger than 0.</td>
</tr>
<tr>
<td>lambda</td>
<td>A parameter value for distribution that determines the shape of the curve. The value must be larger than 0.</td>
</tr>
<tr>
<td>cumulative</td>
<td>A logical value that determines whether EXPONDIST returns a singular or cumulative value. Use False to calculate the distribution for the x value only. Use True to calculate the cumulative distribution.</td>
</tr>
</tbody>
</table>

**Equation**
When cumulative is FALSE: $\lambda e^{-\lambda x}$

When cumulative is TRUE: $1 - e^{-\lambda x}$

**Examples**
This function returns 0.6065306597:
EXPONDIST(0.5,1,FALSE)

This function returns 0.3934693403:
EXPONDIST(0.5,1,TRUE)

**See Also**
BETADIST, BINOMDIST, CHIDIST, COMBIN, CRITBINOM, GAMMADIST, NORMDIST, POISSON

---

### FACT

**Description**
Returns the factorial of a specified number.

**Syntax**
FACT (number)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>Any non-negative integer. If you supply a real number, FACT truncates the number to an integer before calculation.</td>
</tr>
</tbody>
</table>

**Examples**
This function returns 2:
FACT(2.5)

This function returns 720:
FACT(6)

**See Also**
FACT, FACTDOUBLE, PRODUCT
FACTDOUBLE

Description
Returns the double-factorial of a specified number.

Syntax
FACTDOUBLE ( number)

Remarks
Double-factorials differ from factorials in that the decrement value (the amount of
decrease in the variable) is 2 instead of one, denoted by the double exclamation
point: 6!! = 6 • 4 • 2 = 48 and 5!! = 5 • 3 • 1 = 15.

Examples
This function returns 2027025:
FACTDOUBLE(15)

This function returns 3840:
FACTDOUBLE(10)

See Also
FACT, FACTDOUBLE, PRODUCT

FALSE

Description
Returns the logical value False. This function always requires the trailing
parentheses.

Syntax
FALSE ( )

See Also
TRUE

FDIST

Description
Computes the complementary F-distribution, which is used to compare data from two
different populations.

Syntax
FDIST (X, df1, df2)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
</table>
| X        | The value at which the function will be evaluated. It must be larger than
          | or equal to 0. |
Equation

\[ I_w(d_1/2, d_2/2) \quad \text{where} \quad w = \frac{d_1 X}{d_2 + d_1 X} \quad \text{and} \quad I \text{ refers to the beta distribution.} \]

Examples

This function returns 0.256387198:

\[ \text{FDIST}(2, 3, 4) \]

This function returns 0.047619048:

\[ \text{FDIST}(20, 2, 2) \]

See Also

BETADIST, BINOMDIST, CHIDIST, COMBIN, CRITBINOM, FINV, FTEST, GAMMADIST, NORMDIST, POISSON

---

FIND

Description

Searches for a string of text within another text string and returns the character position at which the search string first occurs.

Syntax

\[ \text{FIND} \left( \text{search_text}, \text{text} \left[ , \text{start_position} \right] \right) \]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>search_text</td>
<td>The text to find. If you specify an empty string (&quot;&quot;), FIND matches the first character in text.</td>
</tr>
<tr>
<td>text</td>
<td>The text to be searched.</td>
</tr>
<tr>
<td>[start_position]</td>
<td>Optional. The character position in text where the search begins. The first character in text is character number 1. If this argument is omitted, 1 is used.</td>
</tr>
</tbody>
</table>

Remarks

FIND is case-sensitive. You cannot use wildcard characters in the search_text.

Examples

This function returns 12:

\[ \text{FIND("time", "There's no time like the present")} \]

This function returns 19:

\[ \text{FIND("4", "Aisle 4, Part 123-4-11", 9)} \]

See Also

EXACT, LEN, MID, SEARCH
FINV

Description
Computes the inverse of the complementary F-distribution (that is, the inverse of the FDIST function.) It is used when comparing data from two different populations.

Syntax
FINV (probability, df1, df2)

Remarks
This function can be used to compute the critical value needed to test the “null hypothesis” in many statistical experiments. For example, if there is to be less than a 1% chance that results are due to random effects for two populations, you can run FINV with 0.01 as the probability argument and the number of degrees of freedom you choose. Values achieved in the experiment must be higher than the resultant critical value in order to be significant.

Examples
This function returns 7.71:
FINV(0.05, 1, 4)
This function returns 9.01:
FINV(0.05, 5, 3)

See Also
FDIST, FTEST, TDIST, TINV, TTEST

FISHER

Description
Computes Fisher’s z-transformation. Note that this is a two-tailed test.

Syntax
FISHER (x)

Equation
\[ z = \frac{1}{2} \ln \left( \frac{1 + x}{1 - x} \right)\]
Examples
This function returns 0.549306144:
\[ \text{FISHER}(0.5) \]
This function returns -0.255412812:
\[ \text{FISHER}(-0.25) \]

See Also
CORREL, FISHERINV, FTEST, PEARSON

FISHERINV

Description
Computes the inverse of Fisher's z-transformation.

Syntax
\[ \text{FISHERINV}(z) \]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( z )</td>
<td>A value from FISHER for which you want the inverse transformation.</td>
</tr>
</tbody>
</table>

Examples
This function returns \(.999999\):
\[ \text{FISHERINV}(10) \]
This function returns \(0.244918662\):
\[ \text{FISHERINV}(0.25) \]

See Also
CORREL, FISHER, FTEST, PEARSON

FIXED

Description
Rounds a number to the supplied precision, formats the number in decimal format, and returns the result as text.

Syntax
\[ \text{FIXED}(\ number [\ precision][, no_commas]) \]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( number )</td>
<td>Any number.</td>
</tr>
<tr>
<td>( [precision] )</td>
<td>Optional. The number of digits that appear to the right of the decimal place. If you specify negative precision, ( number ) is rounded to the left of the decimal point. You can specify a precision as great as 127 digits. If this argument is omitted, 2 is used.</td>
</tr>
<tr>
<td>( [no_commas] )</td>
<td>Optional. Determines if thousands separators (commas) are used in the result. Use 1 to exclude commas in the result. Use 0 to include thousands separators (for example, 1,000.00). If the argument is omitted, 0 is used.</td>
</tr>
</tbody>
</table>
FLOOR

Description
Rounds a number down to the nearest multiple of a specified significance.

Syntax
FLOOR (number, significance)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>The value to round.</td>
</tr>
<tr>
<td>significance</td>
<td>The multiple to which to round.</td>
</tr>
</tbody>
</table>

Remarks
Regardless of the sign of the number, the value is rounded towards zero. If number is an exact multiple of significance, no rounding occurs.

If number or significance is non-numeric, #NAME? is returned. When the arguments have opposite signs, #NUM! is returned.

Examples
This function returns 1.2:
FLOOR(1.23459, .05)

This function returns –148:
FLOOR(-148.24, –2)

See Also
CEILING, EVEN, INT, ODD, ROUND, TRUNC

FORECAST

Description
Constructs the least squares regression line through the data given, then computes the predicted y value for the requested x.

Syntax
FORECAST (x, arrayY, arrayX)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>The x value at which you are requesting the corresponding y value for this regression.</td>
</tr>
</tbody>
</table>
Chapter 3    A-Z Function Reference 115

### Remarks
This function pairs up the numbers in the two ranges by moving left-to-right through each sequential row.

### Equations
FORECAST uses the following formulas for slope and intercept to draw the line used to generate the forecast of $y$ at a given $x$.

\[
\text{slope } b = \frac{\sum xy - \left(\sum x\right)\left(\sum y\right)}{n \sum x^2 - \left(\sum x\right)^2}
\]

\[
\text{intercept } a = \frac{\sum y}{n} - b\left(\frac{\sum y}{n}\right)
\]

where $n$ is the number of members in $arrayY$ and $arrayX$.

### Examples
This function returns 1:

\[
\text{FORECAST(0.5, \{1, 2, 4, 6, 7, 9\}, \{0, 2, 4, 5, 7, 8\})}
\]

### See Also
INTERCEPT, SLOPE

---

### FREQUENCY

#### Description
Distributes the specified array of numbers, sorted in ascending order, into specified bins.

**Note** This is an array function. For information, see “Array Functions” on page 13.

#### Syntax
FREQUENCY (array, bins)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>A range reference or the name of a named range reference.</td>
</tr>
</tbody>
</table>
Results range

Before entering this function, select a results range one column wide with one more row than the number of entries in bins. If you select too few cells, some cells will be truncated. If you select too many cells, the extra cells will contain the #N/A error.

Remarks

When the values in bins are in ascending order, FREQUENCY sorts array in ascending order. It then calculates the number of members of array that are smaller than or equal to the smallest value in bins and places that value in the first cell of the results range. It then calculates the number of members of array that are larger than the smallest value in bins but smaller than or equal to the next larger member of bins, and places that value in the second cell of the results range. The last cell of the results range will contain the number of members of array that are larger than all the values in bins.

When the values in bins are not in ascending order, FREQUENCY determines the number to appear in each bin in the same manner. However, it displays the numbers in the order that their bins were entered. The last cell of the results range will always contain the number of members of array that are larger than all the values in bins. See examples.

Examples

The examples use the following worksheet.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
FREQUENCY(A1:A10,B1:B3) returns the following range:

<table>
<thead>
<tr>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

FREQUENCY(A1:A10,C1:C3) returns the following range:

<table>
<thead>
<tr>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

FREQUENCY computed the same results in the two examples, but displayed them in a different order in the results range. Also note that the last value is the same regardless of the order of the values in bins.

See Also PROB

**FTEST**

**Description**
Computes an F-test probability for two specified distributions.

**Syntax**
FTEST (array1, array2)

**Argument** | **Description**
--- | ---
array1 | A set of at least 2 values. It must be a range reference or array constant containing numeric values. Text, logical values, and blank cells referenced by this function are ignored.
array2 | A set of at least 2 values. It must be a range reference or array constant containing numeric values. Text, logical values, and blank cells referenced by this function are ignored.

The number of values in array2 may be different than in array1.

**Remarks**
To arrive at a result, this function first computes the variances of the two distributions, then an F ratio based on the two variances. It also computes the degrees of freedom by adding the counts of each sample and subtracting 1 from each count. Finally, it uses the FDIST function internally, with F ratio and degrees of freedom as arguments, to calculate the end result.

If the variance of array1 or array2 is 0, this function will return the #DIV/0! error.
Examples

This function returns 0.167:

\[ FTEST((51,45,41,27),(91,37,89,82)) \]

See Also

CHITEST, TTEST, ZTEST

---

**FV**

Description

Returns the future value of an annuity based on regular payments and a fixed interest rate.

Syntax

\[ FV(\text{interest}, \text{nper}, \text{payment}[, \text{pv}] [, \text{type}]) \]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>interest</td>
<td>The fixed interest rate.</td>
</tr>
<tr>
<td>nper</td>
<td>The number of payments in an annuity.</td>
</tr>
<tr>
<td>payment</td>
<td>The fixed payment made each period.</td>
</tr>
<tr>
<td>[pv]</td>
<td>Optional. The present value, or the lump sum amount, the annuity is currently worth. If this argument is omitted, 0 is used.</td>
</tr>
<tr>
<td>[type]</td>
<td>Optional. Indicates when payments are due. Use 0 if payments are due at the end of the period or 1 if payments are due at the beginning of the period. If you omit this argument, 0 is used.</td>
</tr>
</tbody>
</table>

Remarks

The units used for \text{interest} must match those used for \text{nper}. For example, if the annuity has an 8 percent annual interest rate over a period of 5 years, specify 8 percent/12 for \text{interest} and 5*12 for \text{nper}.

Cash paid out, such as a payment, is shown as a negative number. Cash received, such as a dividend check, is shown as a positive number.

Examples

This function returns 4,774.55:

\[ FV(5\%, 8, -500) \]

This function returns 531,550.86:

\[ FV(10\%/12, 240, -700, 1) \]

See Also

CUMIPMT, CUMPRINC, FVSCHEDULE, IPMT, NPER, PMT, PPMT, PV, RATE
FVSCHEDULE

**Description**
Computes the future value of an investment after applying a series of compounded interest rates.

**Syntax**
FVSCHEDULE (value, schedule)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>The starting value of the investment (the principal).</td>
</tr>
<tr>
<td>schedule</td>
<td>A list of interest rates for the compounding periods. This may be a range reference or an array constant. The number of values in schedule is the number of periods for compounding interest. An empty cell or 0 value will indicate no interest earned in that period.</td>
</tr>
</tbody>
</table>

**Examples**
This function returns 1367.52:

FVSCHEDULE(1000, {0.2, 0.21, 0.22})

**See Also**
FV
GAMMADIST

Description
Computes the gamma distribution for a given value.

Syntax
GAMMADIST (x, alpha, beta, cumulative)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>The value at which you want to evaluate the function. It must be greater than or equal to 0.</td>
</tr>
<tr>
<td>alpha</td>
<td>The alpha parameter. It must be greater than 0.</td>
</tr>
<tr>
<td>beta</td>
<td>The beta parameter. It must be greater than 0.</td>
</tr>
<tr>
<td>cumulative</td>
<td>A logical value indicating the type of calculation.</td>
</tr>
<tr>
<td></td>
<td>TRUE Computes the cumulative area from 0 to x.</td>
</tr>
<tr>
<td></td>
<td>FALSE Computes the value at x.</td>
</tr>
</tbody>
</table>

Equations
\[ f(x; \alpha, \beta) = \frac{1}{\beta \Gamma(\alpha)} \int_0^x x^{\alpha-1} e^{-x/\beta} \, dx \] ...when cumulative is TRUE.

\[ f(x; \alpha, \beta) = \frac{1}{\beta \Gamma(\alpha)} x^{\alpha-1} e^{-x/\beta} \] ...when cumulative is FALSE.

Examples
This function returns 0.247:
GAMMADIST (12, 3, 7, TRUE)

This function returns 0.038:
GAMMADIST (12, 3, 7, FALSE)

See Also
BETADIST, BINOMDIST, CHIDIST, COMBIN, CRITBINOM, FDIST, GAMMAINV, NORMDIST, POISSON

GAMMAINV

Description
Computes the inverse of the cumulative gamma distribution. This is the inverse of the GAMMADIST function when cumulative is TRUE.

Syntax
GAMMAINV (probability, alpha, beta)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>probability</td>
<td>A number between 0 and 1 indicating the probability.</td>
</tr>
<tr>
<td>alpha</td>
<td>The alpha parameter. It must be greater than 0.</td>
</tr>
<tr>
<td>beta</td>
<td>The beta parameter. It must be greater than 0.</td>
</tr>
</tbody>
</table>
Examples

This function returns 5.812:
GAMMAINV (0.01, 8, 2)

This function returns 7.962:
GAMMAINV (0.05, 8, 2)

See Also
GAMMADIST

GAMMALN

Description
Computes the logarithm of the gamma function.

Syntax
GAMMALN (number)

Remarks
The gamma function is a generalization of the factorial function. For positive integers, the factorial of \( n \) is the same as the gamma function of \( n + 1 \). The gamma function is a building block used in worksheet functions like COMBIN and PERMUT.

The inverse of the GAMMALN function is the EXP function with the GAMMALN function as the argument.

Equation
\[
\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt
\]

where \( z \) is the argument.

As noted above, when \( z \) is an integer, the equation can be simplified to
\[
n! = \Gamma(n + 1) \]

Examples
This function returns 3.17805383
GAMMALN(5)

See Also
COMBIN, FACT, PERMUT

GCD

Description
Computes the greatest common divisor of a specified set of values. This number will divide all the numbers in the set evenly, with no fraction or remainder.

Syntax
GCD (number_list)
GEOMEAN

Description
Computes the geometric mean of a list of positive numbers.

Syntax
GEOMEAN (number_list)

Argument | Description
--- | ---
number_list | A list of between 1 and 30 arguments consisting of numbers, cell references, range references, and/or array constants. The arguments must evaluate to positive numbers or 0.

Text and logical values in arrays or cells referenced by this function is ignored. Empty cells are ignored. Text entered in the argument list will be evaluated as a number, if possible; otherwise it will cause a #VALUE! error. Logical values entered into the argument list are evaluated as numeric 1 if TRUE and numeric 0 if FALSE.

Remarks
The geometric mean is an alternative form of calculating a mean for a sample with strong positive skewedness and no negative values. It is obtained from a list of \( n \) numbers by taking the \( n \)th root of the product of the numbers.

Example
This function returns 6.931:

\[
\text{GEOMEAN}(1, 3, 5, 16, 42, 11)
\]

This function returns 12:

\[
\text{GEOMEAN}(24, 6)
\]

See Also
AVERAGE, COUNT, HARMEAN, SKEW
GESTEP

Description
Compared two specified values and returns 1 if the first value is greater than or equal
to the second value, 0 if the first value is less than the second value.

Syntax
DELTA (number1 [, number2])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number1</td>
<td>A number you want to compare.</td>
</tr>
<tr>
<td>[number2]</td>
<td>Optional. This is the number you want to compare to number1. If you omit this argument, the function will compare number1 to 0.</td>
</tr>
</tbody>
</table>

Remarks
This function can be used in place of the equals (=) operator when a numeric rather
than logical result is needed (for example, when using aggregate functions such as
SUM and COUNT).

GE stands for “greater than or equal to.”

Examples
This function returns 0:
DELTA(6, 7)

This function returns 1:
DELTA(6)

See Also
DELTA, IF

GETPIVOTDATA

Description
Returns the #N/A error message.

Syntax
GETPIVOTDATA (pivot_table, name)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pivot_table</td>
<td>The name of the pivot table.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the cell.</td>
</tr>
</tbody>
</table>

Remarks
This function was included only for compatibility with Microsoft Excel, which uses
GETPIVOTDATA to return data from a pivot table. Pivot tables are not supported in
Formula One for Java.

This function is intended only for users or developers who want to import Excel
worksheets that contain Excel’s GETPIVOTDATA function. Such worksheets can be
opened and saved in Formula One for Java, but the GETPIVOTDATA functionality
will not be included.
### GROWTH

**Description**
Computes result values from the fitted curve of a multiple exponential regression for a group of specified observations relative to a group of specified independent variables.

This type of regression is often applied to exponential growth situations. Because of the type of formula and the often limited amount of data the formula is based on, using these formulas to estimate values outside of the range of data is dangerous, since they are likely to give exponentially wrong results.

Another problematic aspect of this function is that it assumes the range of data in question fits an exponential model. While this assumption is appropriate in some situations (for example, radioactive decay), in many other situations it is misleading, particularly in regards to economic data.

**Note**
If you want GROWTH to return more than one result value, you must enter it as an array function. For information, see “Array Functions” on page 13.

**Syntax**

```plaintext
GROWTH (known_y’s [, known_x’s] [, new_x’s] [, constant])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>known_y’s</td>
<td>A list of observed values for the dependent variable. This must be a range reference or an array constant containing all numeric values and no empty cells.</td>
</tr>
</tbody>
</table>

For a regression with a **single independent variable**, enter any type of range. For a regression with **multiple independent variables**, enter a single row or column of values.

If you are using array constants, see “Array Constants in Arguments” on page 9 for information on how to enter the array.
Remarks

This function uses LOGEST internally to compute the regression’s coefficients. The resulting formula is then used to compute the requested values.

Results range

Before entering this function, select a results range according to the following criteria:

- If there is a single independent variable, the results range should match the size and shape of the new_x’s argument.
If there are multiple independent variables, the results range depends on the number of data sets in new_x’s. If there is only one data set, the results range should be a single cell. For multiple data sets, the results range should be a column if known_y’s is a column, a row if known_y’s is a row, with the same number of cells as data sets entered in new_x’s. If you omitted the new_x’s argument, the results range must match the size and shape of the known_y’s argument.

Equation

The regression computed by GROWTH attempts to fit the following formula:

\[ y = C_1 x_1 \times C_2 x_2 \times \ldots \times C_n x_n \times b \]

where \( C \) is the coefficient determined internally by the LOGEST function, \( n \) is the number of independent variables, and \( b \) is the constant.

This formula can be converted to a linear form if logarithms are taken of both sides. The formula then becomes:

\[ y = lC_1 x_1 + lC_2 x_2 + \ldots + lC_n x_n + lb \]

This is the same formula fitted by the LINEST function. LOGEST use LINEST internally to fit the linearized form of the equation. LOGEST then takes the LINEST coefficients \( (lC_1, lC_2, \ldots, lC_n) \) and converts them to the LOGEST coefficients \( (C_1, C_2, \ldots, C_n) \) by applying the EXP function: \( C_1 = \exp(lC_1) \). The same conversion is done with the constant, \( b \).

Example

This example of exponential regression uses the following data on the growth of the national debt of a small country.

<table>
<thead>
<tr>
<th>Year</th>
<th>National debt, in millions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1971</td>
<td>0.5</td>
</tr>
<tr>
<td>1972</td>
<td>1.1</td>
</tr>
<tr>
<td>1973</td>
<td>1.4</td>
</tr>
<tr>
<td>1974</td>
<td>2.0</td>
</tr>
<tr>
<td>1975</td>
<td>4.4</td>
</tr>
<tr>
<td>1976</td>
<td>20.0</td>
</tr>
<tr>
<td>1977</td>
<td>445.0</td>
</tr>
</tbody>
</table>

In this case, the debt figures are the known_y’s and we can use default values for the known_x’s, since the years increment by one.

We want to use this data to find what the national debt would have been in June of 1973. If the national debt figures fill range A2:A8, this function returns 2.958273:

\( \text{GROWTH}(A2:A8, 3.5, \text{TRUE}) \)

See Also

INTERCEPT, LINEST, LOGEST, SLOPE, TREND
HARMEAN

Description
Computes the harmonic mean of a list of positive numbers.

Syntax
HARMEAN (number_list)

Remarks
This is an alternative form of calculating a mean for a sample with strong positive skewedness and no negative values. It is obtained by taking the reciprocal of the arithmetic mean of the reciprocals of a list of numbers.

The result of HARMEAN is smaller than the geometric mean calculated by GEOMEAN.

Example
This function returns 9.731:

HARMEAN(5, 4, 25, 60, 14, 26)

See Also
AVERAGE, COUNT, GEOMEAN, SKEW

HLOOKUP

Description
Searches the top row of a table for a value and returns the contents of a cell in that table that corresponds to the location of the search value.

Syntax
HLOOKUP (search_item, search_range, row_index)

Argument Description

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>search_item</td>
<td>A value, text string, or reference to a cell containing a value that is matched against data in the top row of search_range.</td>
</tr>
<tr>
<td>search_range</td>
<td>A reference to the range (table) to be searched. The cells in the first row of search_range can contain numbers, text, or logical values. The contents of the first row must be in ascending order (for example, -2, -1, 0, 2 ... A through Z, False, True). Text searches are not case-sensitive.</td>
</tr>
</tbody>
</table>
**Remarks**

HLOOKUP compares the information in the top row of `search_range` to the supplied `search_item`. When a match is found, information located in the same column and supplied row (`row_index`) is returned.

If `search_item` cannot be found in the top row of `search_range`, the largest value that is less than `search_item` is used. When `search_item` is less than the smallest value in the first row of the `search_range`, the error #REF! is returned.

**Examples**

The following examples use this worksheet.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>Midwest</td>
<td>Northeast</td>
<td>Pacific</td>
<td>South</td>
</tr>
<tr>
<td>2</td>
<td>Q1</td>
<td>40.23</td>
<td>278.21</td>
<td>61.97</td>
<td>164.60</td>
</tr>
<tr>
<td>3</td>
<td>Q2</td>
<td>163.83</td>
<td>22.63</td>
<td>161.73</td>
<td>183.96</td>
</tr>
<tr>
<td>4</td>
<td>Q3</td>
<td>43.96</td>
<td>233.56</td>
<td>278.16</td>
<td>171.98</td>
</tr>
<tr>
<td>5</td>
<td>Q4</td>
<td>245.69</td>
<td>167.09</td>
<td>245.23</td>
<td>163.00</td>
</tr>
</tbody>
</table>

This function returns 22.63:

HLOOKUP("Northeast", B1:E5, 3)

This function returns #REF!:

HLOOKUP("Pacific", B1:E5, 7)

**See Also**

INDEX (non-array type), LOOKUP, MATCH, VLOOKUP

---

**HEX2BIN**

**Description**

Converts a hexadecimal number (base 16) to a binary number (base 2).

**Syntax**

`HEX2BIN ( integer [, places] )`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>integer</code></td>
<td>Any negative or positive whole number or zero.</td>
</tr>
</tbody>
</table>

For the purposes of this function, the integer entered must be small enough to convert within the limits of the target. Any integer between FFFFFFFFE00 to 1FF may be used.
HEX2BIN

Description
Converts a hexadecimal number (base 16) to a binary number (base 2).

Syntax
HEX2BIN (integer)

Remarks
Note that the result of this function is a binary string.

Examples
This function returns 10000:
HEX2BIN(10)

This function returns 1111110000:
HEX2BIN("FFFFFFFFF0")

This function returns 000001000:
HEX2BIN(10,10)

See Also
BIN2DEC, BIN2HEX, BIN2OCT, DEC2BIN, DEC2HEX, DEC2OCT, HEX2DEC,
HEX2OCT, OCT2BIN, OCT2DEC, OCT2HEX

HEX2DEC

Description
Converts a hexadecimal number (base 16) to a decimal number (base 10).

Syntax
HEX2DEC (integer)

Remarks
Note that the result of this function is a number.

Examples
This function returns 16:
HEX2DEC(10)

This function returns 5.49756E + 11:
HEX2DEC("7FFFFFFF")

See Also
BIN2DEC, BIN2HEX, BIN2OCT, DEC2BIN, DEC2HEX, DEC2OCT, HEX2BIN,
HEX2OCT, OCT2BIN, OCT2DEC, OCT2HEX
HEX2OCT

Description
Converts a hexadecimal number (base 16) to an octal number (base 8).

Syntax
HEX2OCT(integer [, places])

Argument | Description
--- | ---
integer | Any negative or positive whole number or zero. For the purposes of this function, the integer entered must be small enough to convert within the limits of the target. Any integer between FFE0000000 to 1FFFFFF may be used.
places | Optional number of characters to use in the output. If places is omitted, the function returns the number of spaces required to display the result. The places argument can be used to 0-pad a positive number. If the number is negative, places is ignored and 10 characters are returned.

Remarks
Note that the result of this function is a text string.

Examples
This function returns 400:
HEX2OCT(100)

This function returns 007:
HEX2OCT(7,3)

See Also
BIN2DEC, BIN2HEX, BIN2OCT, DEC2BIN, DEC2HEX, DEC2OCT, HEX2BIN, HEX2DEC, OCT2BIN, OCT2DEC, OCT2HEX

HOUR

Description
Returns the hour component of the specified time in 24-hour format.

Syntax
HOUR(serial_number)

Argument | Description
--- | ---
serial_number | The time as a serial number. The decimal portion of the number represents time as a fraction of the day.

Remarks
The result is an integer ranging from 0 (12:00 AM) to 23 (11:00 PM).

Examples
This function returns 9:
HOUR(34259.4)
This function returns 23:

\[
\text{HOUR}(34619.976)
\]

See Also

DAY, MINUTE, MONTH, NOW, SECOND, WEEKDAY, YEAR

---

**HYPERLINK**

**Description**

Returns the #N/A error message.

**Syntax**

\[
\text{HYPERLINK} \left( \text{link\_location} \left[ , \text{cell\_content} \right] \right)
\]

**Argument** | **Description**
--- | ---
link\_location | The path and filename to the document to be opened as text in a browser.
[cell\_content] | Optional argument. Text to display in cell.

**Remarks**

This function was included only for compatibility with Microsoft Excel. In Excel, HYPERLINK sets up a hyperlink to a file or web page, which is not supported in Java.

This function is intended only for users or developers who want to import Excel worksheets that contain Excel’s HYPERLINK function. To make those worksheets work properly, developers should convert these functions to add-in functions, which they can write in Java and set up to be automatically loaded.

For more information, see the chapter on creating add-in functions in the *Formula One for Java Technical Guide*.

---

**HYPGEOMDIST**

**Description**

Computes the hypergeometric distribution. This is used in cases of sampling without replacement, and is especially important for determining the likelihood of finding defective parts in a production run.

For cases of sampling with replacement, use BINOMDIST.

**Syntax**

\[
\text{HYPGEOMDIST} \left( x, n, M, N \right)
\]

**Argument** | **Description**
--- | ---
x | A positive integer indicating the number of successes in the sample. It must be smaller than \( n \) and \( M \). Decimal values will be truncated to integers.
n | A positive integer indicating the size of the sample. It must be smaller than \( N \). Decimal values will be truncated to integers.
Equation

\[ f(x) = \frac{\binom{M}{x} \binom{N - M}{n - x}}{\binom{N}{n}} \]

Examples

Say 10 sweaters on a store rack have been inspected, and 2 of the sweaters have flaws. This function calculates the probability that, if you buy four sweaters, none of them will have flaws.

This function returns 0.33:

\[ \text{HYPGEOMDIST}(0, 4, 2, 10) \]

See Also

BETADIST, BINOMDIST, CHIDIST, COMBIN, CRITBINOM, FDIST, NORMDIST, POISSON

IF

Description

Tests the condition and returns the specified value.

Syntax

\[ \text{IF}( \text{condition, true_value, false_value} ) \]

Example

This function returns Greater if the contents of A1 is greater than 10 and Less if the contents of A1 is less than 10:

\[ \text{IF}(A1>10, \text{"Greater"}, \text{"Less"}) \]

See Also

AND, FALSE, NOT, OR, TRUE
**IMABS**

**Description**
Computes the absolute value of a complex number.

**Syntax**

```
IMABS (complex_number)
```

**Argument** | **Description**
--- | ---
complex_number | A complex number in the form \((a + bi)\). You may enter either or both components of the complex number. See COMPLEX for details.

If the complex number or imaginary component of a complex number is entered within the function’s argument list, it must be enclosed within quotation marks (e.g., \(IMABS("1+1i")\) or \(IMABS("i")\)).

**Remarks**
The result can be thought of as the distance from the origin of a graph to the point where the X-axis is the value of the real number and the Y-axis is the value of the imaginary number.

**Example**
This function returns 5:

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

**See Also**
COMPLEX, IMAGINARY, IMARGUMENT, IMCONJUGATE, IMCOS, IMDIV, IMEXP, IMLN, IMLOG10, IMLOG2, IMPower, IMPRODUCT, IMREAL, IMSIN, IMSQRT, IMSUB, IMSUM

---

**IMAGINARY**

**Description**
Returns the coefficient of the imaginary portion of a complex number as a real number.

**Syntax**

```
IMAGINARY (complex_number)
```

**Argument** | **Description**
--- | ---
complex_number | A complex number in the form \((a + bi)\). You may enter either or both components of the complex number. See COMPLEX for details.

If the complex number or imaginary component of a complex number is entered within the function’s argument list, it must be enclosed within quotation marks (e.g., \(IMABS("1+1i")\) or \(IMABS("i")\)).

**Example**
This function returns 3:

```
IMAGINARY("2+3i")
```
IMARGUMENT

Description
Computes the argument θ as an angle expressed in radians.

Syntax
IMARGUMENT (complex_number)

Argument Description

complex_number A complex number in the form (a + bi). You may enter either or both components of the complex number. See COMPLEX for details.

If the complex number or imaginary component of a complex number is entered within the function’s argument list, it must be enclosed within quotation marks (e.g., IMABS("1+1i") or IMABS("i")).

Equation
ATAN(b/a)

Remarks
This is the complementary function to IMABS, where the degree of angle (θ) begins at the positive X-axis, and rotates counterclockwise in polar coordinates.

An alternate form of complex notation is r(cos θ + i sin θ) where “r” is the result of IMABS.

Note
The range of θ is -π < θ ≤ π.

Example
This function returns 0.785398163:
IMARGUMENT("1+1i")

This function returns 3.14159265, the value π:
IMARGUMENT(-1)

See Also
COMPLEX, IMABS, IMARGUMENT, IMCONJUGATE, IMCOS, IMDIV, IMEXP, IMLN, IMLOG10, IMLOG2, IMPOWER, IMPRODUCT, IMREAL, IMSIN, IMSQRT, IMSUB, IMSUM
## IMCONJUGATE

**Description**
Computes the complex conjugate of the argument.

**Syntax**
```
IMCONJUGATE ( complex_number )
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>complex_number</td>
<td>A complex number in the form ((a + bi)). You may enter either or both components of the complex number. See COMPLEX for details. If the complex number or imaginary component of a complex number is entered within the function’s argument list, it must be enclosed within quotation marks (e.g., <code>IMABS(&quot;1+1i&quot;)</code> or <code>IMABS(&quot;i&quot;)</code>).</td>
</tr>
</tbody>
</table>

**Remarks**
This is simply the same number with the sign changed on the imaginary component.

**Example**
This function returns \(2-3i\):
```
IMCONJUGATE("2+3i")
```
This function returns \(2+3i\):
```
IMCONJUGATE("2-3i")
```

**See Also**
COMPLEX, IMABS, IMAGINARY, IMARGUMENT, IMCOS, IMDIV, IMEXP, IMLN, IMLOG10, IMLOG2, IMPOWER, IMPRODUCT, IMREAL, IMSIN, IMSQRT, IMSUB, IMSUM

## IMCOS

**Description**
Computes the complex cosine of the argument.

**Syntax**
```
IMCOS ( complex_number )
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>complex_number</td>
<td>A complex number in the form ((a + bi)). You may enter either or both components of the complex number. See COMPLEX for details. If the complex number or imaginary component of a complex number is entered within the function’s argument list, it must be enclosed within quotation marks (e.g., <code>IMABS(&quot;1+1i&quot;)</code> or <code>IMABS(&quot;i&quot;)</code>).</td>
</tr>
</tbody>
</table>

**Equation**
\[
\cos(a + bi) = \cos(a) \cosh(b) - i \sin(a) \sinh(b)
\]

**Example**
This function returns \(-0.64214812471552 - 1.06860742138278i\):
```
IMCOS("2+i")
```
This function returns -0.9899992496600445:

\[
\text{IMCOS}(3)
\]

See Also

COMPLEX, IMABS, IMAGINARY, IMARGUMENT, IMCONJUGATE, IMDIV, IMEXP, IMLN, IMLOG10, IMLOG2, IMPOWER, IMPRODUCT, IMREAL, IMSIN, IMSQRT, IMSUB, IMSUM

---

**IMDIV**

**Description**
Divides one complex number by another.

**Syntax**

IMDIV ( complex_number, complex_number2)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>complex_number</td>
<td>A complex number in the form (a + bi). You may enter either or both components of the complex number. See COMPLEX for details.</td>
</tr>
<tr>
<td>complex_number2</td>
<td>Another complex number (a + bi).</td>
</tr>
</tbody>
</table>

If the complex number or imaginary component of a complex number is entered within the function’s argument list, it must be enclosed within quotation marks (e.g., IMABS("1+1i") or IMABS("i").

Any function that accepts two or more complex arguments must have the same suffix (i or j) on all arguments to that function, or a #VALUE error will result.

\[
\frac{z_1}{z_2} = \frac{a + bi}{c + di} + \frac{-a + bi}{c + di} \quad \text{...where } z \text{ is a complex number.}
\]

**Example**
This function returns 2 + 6i:

IMDIV("-10+10i","1+2i")

See Also

COMPLEX, IMABS, IMAGINARY, IMARGUMENT, IMCONJUGATE, IMCOS, IMEXP, IMLN, IMLOG10, IMLOG2, IMPOWER, IMPRODUCT, IMREAL, IMSIN, IMSQRT, IMSUB, IMSUM

---

**IMEXP**

**Description**
Computes the complex exponential of the specified complex number. (The constant, \(e\), is raised to the power equal to the specified complex number.)

**Syntax**

IMEXP ( complex_number)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>complex_number</td>
<td>A complex number in the form (a + bi). You may enter either or both components of the complex number. See COMPLEX for details.</td>
</tr>
</tbody>
</table>

\[
\frac{z_1}{z_2} = \frac{a + bi}{c + di} + \frac{-a + bi}{c + di} \quad \text{...where } z \text{ is a complex number.}
\]

**Example**
This function returns 2 + 6i:

IMDIV("-10+10i","1+2i")

See Also

COMPLEX, IMABS, IMAGINARY, IMARGUMENT, IMCONJUGATE, IMCOS, IMEXP, IMLN, IMLOG10, IMLOG2, IMPOWER, IMPRODUCT, IMREAL, IMSIN, IMSQRT, IMSUB, IMSUM
**Equation**

\[ e^z = e^{(a+bi)} = e^a (\cos(b) + i\sin(b)) \] ...where \( z \) is a complex number.

**Example**

This function returns -7.3151100949011 + 1.0427436562359i:

\[ \text{IMEXP("2+3i")} \]

**See Also**

COMPLEX, IMABS, IMAGINARY, IMARGUMENT, IMCONJUGATE, IMCOS, IMDIV, IMLN, IMLOG10, IMLOG2, IMPOWER, IMPRODUCT, IMREAL, IMSIN, IMSQRT, IMSUB, IMSUM

---

**IMLN**

**Description**

Computes the complex natural logarithm of the specified complex number.

**Syntax**

\[ \text{IMLN (complex_number)} \]

**Argument**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>complex_number</td>
<td>A complex number in the form ((a + bi)). You may enter either or both components of the complex number. See COMPLEX for details.</td>
</tr>
</tbody>
</table>

If the complex number or imaginary component of a complex number is entered within the function's argument list, it must be enclosed within quotation marks (e.g., \text{IMABS("1+1i")} or \text{IMABS("1")}).

**Equation**

\[ \text{LN}z = \text{LN}(|z| + \text{IMARG}z) = \text{LN}\sqrt{a^2 + b^2} + \text{IMARG}(a + bi) \] ...where \( z \) is a complex number.

**Example**

This function returns 1.28247467873077 + 0.982793723247329i:

\[ \text{IMLN("2+3i")} \]

**See Also**

COMPLEX, IMABS, IMAGINARY, IMARGUMENT, IMCONJUGATE, IMCOS, IMDIV, IMEXP, IMLOG10, IMLOG2, IMPOWER, IMPRODUCT, IMREAL, IMSIN, IMSQRT, IMSUB, IMSUM
**IMLOG10**

**Description**
Computes the complex common logarithm of the specified complex number.

**Syntax**
`IMLOG10 ( complex_number )`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>complex_number</td>
<td>A complex number in the form (a + bi). You may enter either or both components of the complex number. See COMPLEX for details. If the complex number or imaginary component of a complex number is entered within the function’s argument list, it must be enclosed within quotation marks (e.g., <code>IMABS(&quot;1+1i&quot;)</code> or <code>IMABS(&quot;1&quot;)</code>).</td>
</tr>
</tbody>
</table>

**Equation**
\[ \log_{10} e \cdot \ln(a + bi) \]

**Example**
This function returns 0.556971676153418 + 0.426821890855467i:
`IMLOG10("2+3i")`

**See Also**
COMPLEX, IMABS, IMAGINARY, IMARGUMENT, IMCONJUGATE, IMCOS, IMDIV, IMEXP, IMLN, IMLOG2, IMPOWER, IMPRODUCT, IMREAL, IMSIN, IMSQRT, IMSUB, IMSUM

---

**IMLOG2**

**Description**
Computes the complex logarithm base-2 of the specified complex number.

**Syntax**
`IMLOG2 ( complex_number )`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>complex_number</td>
<td>A complex number in the form (a + bi). You may enter either or both components of the complex number. See COMPLEX for details. If the complex number or imaginary component of a complex number is entered within the function’s argument list, it must be enclosed within quotation marks (e.g., <code>IMABS(&quot;1+1i&quot;)</code> or <code>IMABS(&quot;1&quot;)</code>).</td>
</tr>
</tbody>
</table>

**Equation**
\[ \log_2 e \cdot \ln(a + bi) \]

**Example**
This function returns 1.85021985921295 + 1.41787163085485i:
`IMLOG2("2+3i")`
**IMPOWER**

**Description**
Computes the real power of a complex number (raises a complex number to a specified power).

**Syntax**
\[ \text{IMPOWER (} \text{complex\_number} , \text{number} \) \]

**Argument** | **Description**
--- | ---
\text{complex\_number} | A complex number in the form \((a + bi)\), or a cell or range reference containing complex numbers. You may enter either or both components of the complex number. See COMPLEX for details. If the complex number or imaginary component of a complex number is entered within the function’s argument list, it must be enclosed within quotation marks (e.g., \text{IMABS("1+1i") or IMABS("i")}).
\text{number} | Any real number

**Equation**
\[ z^n = r^n [\cos(n\theta) + i\sin(n\theta)] \]

...where \( z \) is a complex number, \( n \) is an integer greater than 0, \( r = \text{imabs}(z) \), and \( \theta = \text{imargument}(z) \).

**Example**
This function returns \(-3 + 4i\):
\[ \text{IMPOWER("1+2i", 2)} \]

**See Also**
COMPLEX, IMABS, IMAGINARY, IMARGUMENT, IMCONJUGATE, IMCOS, IMDIV, IMEXP, IMLN, IMLOG10, IMPOWER, IMPRODUCT, IMREAL, IMSIN, IMSQRT, IMSUB, IMSUM

---

**IMPRODUCT**

**Description**
Computes the product of up to 29 complex numbers.

**Syntax**
\[ \text{IMPRODUCT (} \text{complex\_number}_{1}, \text{complex\_number}_{2} \) \]

**See Also**
COMPLEX, IMABS, IMAGINARY, IMARGUMENT, IMCONJUGATE, IMCOS, IMDIV, IMEXP, IMLN, IMLOG10, IMLOG2, IMPRODUCT, IMREAL, IMSIN, IMSQRT, IMSUB, IMSUM
IMREAL

Description
Returns the coefficient of the real portion of a specified complex number.

Syntax
IMREAL (complex_number)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>complex_number</td>
<td>A complex number in the form ( a + bi ). You may enter either or both components of the complex number. See COMPLEX for details. If the complex number or imaginary component of a complex number is entered within the function’s argument list, it must be enclosed within quotation marks (e.g., IMABS(&quot;1+1i&quot;) or IMABS(&quot;i&quot;).)</td>
</tr>
</tbody>
</table>

Equation
\[ z_1z_2 = (a_1 + ib_1) \cdot (a_2 + ib_2) = (a_1a_2 - b_1b_2) + i(a_1b_2 + a_2b_1) \]
...where \( z \) is a complex number.

Examples
This function returns \( 3 + 29i \):
IMPRODUCT("3+4i","5+3i")

This function returns \( 30 + 60i \):
IMPRODUCT("1+2i",30)

This function returns \( -1650 + 1050i \):
IMPRODUCT("3+4i","5+31i","1+21i",30)

See Also
COMPLEX, IMABS, IMAGINARY, IMARGUMENT, IMCONJUGATE, IMCOS, IMDIV, IMEXP, IMLN, IMLOG10, IMLOG2, IMPOWER, IMREAL, IMSIN, IMSQRT, IMSUB, IMSUM
Example

This function returns 2:

IMREAL("2+3i")

See Also

COMPLEX, IMABS, IMAGINARY, IMARGUMENT, IMCONJUGATE, IMCOS, IMDIV, IMEXP, IMLN, IMLOG10, IMLOG2, IMPOWER, IMPRODUCT, IMSIN, IMSQRT, IMSUB, IMSUM

---

**IMSIN**

Description

Computes the complex sine of the specified complex number.

Syntax

IMSIN ( complex_number)

**Argument** | **Description**
--- | ---
complex_number | A complex number in the form (a + bi). You may enter either or both components of the complex number. See COMPLEX for details.

   If the complex number or imaginary component of a complex number is entered within the function’s argument list, it must be enclosed within quotation marks (e.g., IMABS("1+i") or IMABS("i").

Equation

\[ \sin(a + bi) = \sin(a) \cosh(b) + i \cos(a) \sinh(b) \]

Example

This function returns: 9.15449914691143 - 4.16890695996656i:

IMSIN("2+3i")

See Also

COMPLEX, IMABS, IMARGUMENT, IMCONJUGATE, IMCOS, IMDIV, IMEXP, IMLN, IMLOG10, IMLOG2, IMPOWER, IMPRODUCT, IMSIN, IMSQRT, IMSUB, IMSUM

---

**IMSQRT**

Description

Computes the complex square root of the specified complex number.

Syntax

IMSQRT ( complex_number)

**Argument** | **Description**
--- | ---
complex_number | A complex number in the form (a + bi). You may enter either or both components of the complex number. See COMPLEX for details.

   If the complex number or imaginary component of a complex number is entered within the function’s argument list, it must be enclosed within quotation marks (e.g., IMABS("1+i") or IMABS("i").

Equation

\[ \sqrt{a + bi} = \sqrt{\frac{a^2 + b^2}{2}} \pm \frac{b}{2 \sqrt{\frac{a^2 + b^2}{2}}} \cdot i \]

Example

This function returns: 2.50662827463109 + 1.26909504598167i:

IMSQRT("2+3i")

See Also

COMPLEX, IMABS, IMARGUMENT, IMCONJUGATE, IMCOS, IMDIV, IMEXP, IMLN, IMLOG10, IMLOG2, IMPower, IMPRODUCT, IMSIN, IMSQRT, IMSUB, IMSUM
Equation

\[ z = r \left( \cos \frac{\theta}{2} + i \sin \frac{\theta}{2} \right) \]

where \( z \) is a complex number, \( r = \text{IMABS}(z) \), and \( \theta = \text{IMARGUMENT}(z) \).

Example

This function returns 1.67414922803554 + 0.895977476129838i:

\[ \text{IMSQRT}("2+3i") \]

See Also

COMPLEX, IMABS, IMAGINARY, IMARGUMENT, IMCONJUGATE, IMCOS, IMDIV, IMEXP, IMLN, IMLOG10, IMLOG2, IMPower, IMPRODUCT, IMREAL, IMSIN, IMSUB, IMSUM

---

**IMSUB**

Description

Subtracts one complex number from another.

Syntax

IMSUB ( complex_number, complex_number2 )

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>complex_number</td>
<td>A complex number in the form ((a + bi)). You may enter either or both components of the complex number. See COMPLEX for details. If the complex number or imaginary component of a complex number is entered within the function’s argument list, it must be enclosed within quotation marks (e.g., (\text{IMABS}(&quot;1+1i&quot;)) or (\text{IMABS}(&quot;i&quot;))).</td>
</tr>
<tr>
<td>complex_number2</td>
<td>Another complex number</td>
</tr>
</tbody>
</table>

Equation

\[ z_1 - z_2 = (a_1 - a_2) + i(b_1 - b_2) \]

...where \( z \) is a complex number.

Example

This function returns \(-1 - i\):

\[ \text{IMSUB}("2+3i", "3+4i") \]

See Also

COMPLEX, IMABS, IMAGINARY, IMARGUMENT, IMCONJUGATE, IMCOS, IMDIV, IMEXP, IMLN, IMLOG10, IMLOG2, IMPower, IMPRODUCT, IMREAL, IMSIN, IMSUB, IMSUM
IMSUM

Description
Computes the sum of up to 29 complex numbers.

Syntax
IMSUM (complex_number, complex_number2)

Argument | Description
---|---
complex_number | A complex number in the form (a + bi), or a cell or range reference containing complex numbers. You may enter either or both components of the complex number. See COMPLEX for details.

If the complex number or imaginary component of a complex number is entered within the function’s argument list, it must be enclosed within quotation marks (e.g., IMABS("1+1i") or IMABS("i").

complex_number2 | Another complex number

Any function that accepts two or more complex arguments must have the same suffix (i or j) on all arguments to that function, or a #VALUE error will result.

Equation
\[ z_1 + z_2 = (a_1 + a_2) + i(b_1 + b_2) \]

...where z is a complex number.

Examples
This function returns 8 + 7i:
IMSUM("2+3i","3+4i")

This function returns 36 + 9i:
IMSUM("1+2i","2+3i","3+4i",30)

See Also
COMPLEX, IMABS, IMAGINARY, IMARGUMENT, IMCONJUGATE, IMCOS, IMDIV, IMEXP, IMLN, IMLOG10, IMLOG2, IMPRODUCT, IMREAL, IMSIN, IMSQRT, IMSUB

INDEX (non-array type)

Description
Returns the contents of a cell from a specified range.

Syntax
INDEX (reference [ , row] [ , column] [ , range_number])

Argument | Description
---|---
reference | A reference to one or more ranges. If reference specifies more than one range, separate each reference with a comma and enclose reference in parentheses. For example, INDEX((A1:C6,B7:E14,F4),3,2,2).
Remarks

If each range in reference contains only one row or column, you can omit the row or column argument. For example, if reference is A1:A15, you can omit the column argument like so: INDEX(A1:A15,3,,1).

If row, column, and range_number do not point to a cell within reference, #REF! is returned. If row and column are omitted, INDEX returns the range in reference specified by range_number.

Examples

The following examples use this worksheet.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sales</td>
<td>Group 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Adams</td>
<td>$1,225.14</td>
<td>Cash</td>
<td>$1,819.47</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Baker</td>
<td>$1,415.35</td>
<td>Johnson</td>
<td>$1,733.67</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Martinez</td>
<td>$1,573.57</td>
<td>Nelson</td>
<td>$1,138.23</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Smith</td>
<td>$1,469.78</td>
<td>Randall</td>
<td>$1,634.58</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>White</td>
<td>$1,390.89</td>
<td>Schultz</td>
<td>$1,093.82</td>
<td></td>
</tr>
</tbody>
</table>

This function returns $1415.35:
INDEX(A2:B6,2,2)

This function returns $1634.58:
INDEX((A2:B6,D2:E6),4,2,2)

See Also

CHOOSE, HLOOKUP, LOOKUP, MATCH, VLOOKUP

INDEX (array type)

Description

Returns a specified value from an array.

Note

This is an array function. For information, see “Array Functions” on page 13.

Syntax

INDEX (array [, row] [, column])
 Remarks
You can omit either the row or column argument. If you omit one or the other, the function returns all the values in the specified row or column. See examples.

If row, column, and range_number do not point to a value within array, #REF! is returned.

Examples
When entered in a single cell, this function returns 7:
INDEX({1,4,7;2,5,8;3,6,9},,,3)

When entered as an array function, the same function returns the following range:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

See Also
CHOOSE, HLOOKUP, LOOKUP, MATCH, VLOOKUP

INDIRECT

Description
Returns the contents of the cell referenced by the specified cell.

Syntax
INDIRECT ( ref_text [, a1])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ref_text</td>
<td>A reference to a cell that references a third cell. If ref_text is not a valid reference, the error #REF! is returned.</td>
</tr>
<tr>
<td>[a1]</td>
<td>Optional. The reference format. This argument must be TRUE() to represent an A1 reference format; Formula One does not support the R1C1 reference format.</td>
</tr>
</tbody>
</table>

Example
This function returns the contents of the cell that C1 references. If C1 contains “D1,” then the contents of D1 is returned:

INDIRECT(C1)

See Also
OFFSET
## INFO

**Description**

Returns the specified type of information about the current workbook and system.

**Syntax**

INFO (info_type)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>info_type</td>
<td>The type of information you want. It must be text, surrounded by quotation marks. The following are the valid entries and the types of information they return, in alphabetical order:</td>
</tr>
<tr>
<td>“author”</td>
<td>Returns #N/A.</td>
</tr>
<tr>
<td>“creation-date”</td>
<td>Returns #N/A.</td>
</tr>
<tr>
<td>“editing-time”</td>
<td>Returns #N/A.</td>
</tr>
<tr>
<td>“dbreturncode”</td>
<td>Returns #N/A.</td>
</tr>
<tr>
<td>“dbdrivermessage”</td>
<td>Returns #N/A.</td>
</tr>
<tr>
<td>“dbrecordcount”</td>
<td>Returns #N/A.</td>
</tr>
<tr>
<td>“directory”</td>
<td>Returns the path of the current directory or folder.</td>
</tr>
<tr>
<td>“last-revision-by”</td>
<td>Returns #N/A.</td>
</tr>
<tr>
<td>“last-revision-date”</td>
<td>Returns #N/A.</td>
</tr>
<tr>
<td>“macro-step”</td>
<td>Returns No.</td>
</tr>
<tr>
<td>“macro-trace”</td>
<td>Returns No.</td>
</tr>
<tr>
<td>“memavail”</td>
<td>Returns the amount of memory available, in bytes.</td>
</tr>
<tr>
<td>“memused”</td>
<td>Returns the amount of memory being used for data.</td>
</tr>
<tr>
<td>“mode”</td>
<td>Returns 1.</td>
</tr>
<tr>
<td>“numfile”</td>
<td>Returns #N/A.</td>
</tr>
<tr>
<td>“origin”</td>
<td>Returns #N/A.</td>
</tr>
<tr>
<td>“ostreturncode”</td>
<td>Returns 0.</td>
</tr>
<tr>
<td>“osversion”</td>
<td>Returns the current operating system version as text.</td>
</tr>
<tr>
<td>“recalc”</td>
<td>Returns the current recalculation mode: either Automatic or Manual.</td>
</tr>
<tr>
<td>“release”</td>
<td>Returns the release number of the version of Formula One for Java you are using, as text.</td>
</tr>
<tr>
<td>“setup-user-name”</td>
<td>Returns #N/A.</td>
</tr>
<tr>
<td>“screen-height”</td>
<td>Returns #N/A.</td>
</tr>
<tr>
<td>“screen-width”</td>
<td>Returns #N/A.</td>
</tr>
<tr>
<td>“selection”</td>
<td>Returns #N/A.</td>
</tr>
<tr>
<td>“selection-part”</td>
<td>Returns #N/A.</td>
</tr>
<tr>
<td>“selection-type”</td>
<td>Returns #N/A.</td>
</tr>
<tr>
<td>“system”</td>
<td>Returns the name of the operating system as text.</td>
</tr>
</tbody>
</table>
Remarks

This function is included in Formula One for Java in order to be compatible with Excel, which in turn included it in order to be compatible with other worksheet formats, notably Lotus 1-2-3.

Developers will find it more efficient to extract this type of information using Java direct calls.

Example

This function returns 7.0:

INFO("release")

See Also

CELL

**INT**

Description

Rounds the supplied number down to the nearest integer.

Syntax

`INT ( number )`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>number</code></td>
<td>Any real number.</td>
</tr>
</tbody>
</table>

Examples

This function returns 10:

`INT(10.99)`

This function returns −11:

`INT(-10.99)`

See Also

CEILING, FLOOR, MOD, ROUND, TRUNC
INTERCEPT

Description
Computes the y-intercept of the least squares linear regression line through the given data.

Syntax
INTERCEPT (arrayY, arrayX)

Remarks
This function pairs up the numbers in the two ranges by moving left-to-right through each sequential row.

Equation
\[ \text{intercept} = a = \frac{\sum y}{n} - b \left( \frac{\sum x}{n} \right) \]

where \( b \) is the slope and \( n \) is the number of members in \( arrayY \) and \( arrayX \).

Examples
This function returns 2.142857143:

\[ \text{INTERCEPT}((1,5,8,9,12,13),(0,1,2,3,4,5)) \]

See Also
FORECAST, SLOPE

INTRATE

Description
Computes the interest rate of a fully invested security, given a specified investment dollar amount and redemption value.

Syntax
INTRATE ( settlement, maturity, investment, redemption [, calendar_type])

Arguments
<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>settlement</td>
<td>The date when the security is traded to the buyer. Decimal values will be truncated to integers.</td>
</tr>
<tr>
<td>maturity</td>
<td>The date the security expires and the remaining amount is paid to the investor. It must be the same day or later than settlement. Decimal values will be truncated to integers.</td>
</tr>
<tr>
<td>investment</td>
<td>The dollar amount invested in the security.</td>
</tr>
<tr>
<td>redemption</td>
<td>The amount paid at maturity.</td>
</tr>
</tbody>
</table>
**IPMT**

**Description**

Returns the interest payment of an annuity for a given period, based on regular payments and a fixed periodic interest rate.

**Syntax**

\[
\text{IPMT} \left( \text{interest}, \text{per}, \text{nper}, \text{pv}, [\text{fv}], [\text{type}] \right)
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>interest</td>
<td>The fixed periodic interest rate.</td>
</tr>
<tr>
<td>per</td>
<td>The period for which to return the interest payment. This number must be between 1 and nper.</td>
</tr>
<tr>
<td>nper</td>
<td>The number of payments.</td>
</tr>
<tr>
<td>pv</td>
<td>The present value, or the lump sum amount the annuity is currently worth.</td>
</tr>
<tr>
<td>[fv]</td>
<td>Optional. The future value, or the value after all payments are made. If this argument is omitted, 0 is used.</td>
</tr>
<tr>
<td>[type]</td>
<td>Optional. Indicates when payments are due. Use 0 if payments are due at the end of the period or 1 if payments are due at the beginning of the period. If this argument is omitted, 0 is used.</td>
</tr>
</tbody>
</table>

**Remarks**

The units used for interest must match those used for nper. For example, if the annuity has an 8 percent annual interest rate over a period of 5 years, specify 8 percent/12 for interest and 5*12 for nper.

Cash paid out, such as a payment, is shown as a negative number. Cash received, such as a dividend check, is shown as a positive number.

**Examples**

This function returns −117.87:

\[
\text{IPMT}\left(\frac{8\%}{12}, 2, 48, 18000\right)
\]
This function returns –117.09:

\[ \text{IPMT}(8\%/12, 2, 48, 18000, 0, 1) \]

See Also

FV, CUMIPMT, CUMPRINC, NPER, PMT, PPMT, PV, RATE

### IRR

**Description**

Returns the internal rate of return for a series of periodic cash flows.

**Syntax**

\[ \text{IRR}( \text{cash}\_\text{flow} [, \text{guess}] ) \]

**Argument** | **Description**
---|---
\text{cash}\_\text{flow} | A reference to a range that contains values for which to calculate the internal rate of return. The values must contain at least one positive and one negative value. During calculation, IRR uses the order in which the values appear to determine the order of the cash flow. Text, logical values, and empty cells in the range are ignored.

\[ \text{guess} \] | Optional. The estimate of the internal rate of return. If this argument is omitted, 10 is used.

**Remarks**

The internal rate of return is the interest rate received for an investment consisting of payments (specified by negative numbers) and investments (specified by positive numbers).

IRR is calculated iteratively, cycling through the calculation until the result is accurate to .00001 percent. If the result cannot be found after 20 iterations, #NUM! is returned. When this occurs, supply a different value for \text{guess}.

**Examples**

The following examples use this worksheet.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Investment</td>
<td>($60,000.00)</td>
</tr>
<tr>
<td>2 1989 income</td>
<td>9,590.00</td>
</tr>
<tr>
<td>3 1990 income</td>
<td>10,560.00</td>
</tr>
<tr>
<td>4 1991 income</td>
<td>12,790.00</td>
</tr>
<tr>
<td>5 1992 income</td>
<td>15,830.00</td>
</tr>
<tr>
<td>6 1993 income</td>
<td>18,930.00</td>
</tr>
</tbody>
</table>

This function returns 3.72 percent:

\[ \text{IRR}(B1:B6) \]
This function returns –49.26 percent:

`IRR(B1:B3, -20%)`

**See Also**  
MINVERSE, NPV, RATE

### ISBLANK

**Description**  
Determines if the specified cell is blank.

**Syntax**  
`ISBLANK (reference)`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>reference</td>
<td>A reference to any cell.</td>
</tr>
</tbody>
</table>

**Remarks**  
If the referenced cell is blank, True is returned. False is returned if the cell is not blank.

**Example**  
This function returns True if A1 is a blank cell:

`ISBLANK(A1)`

**See Also**  
ISERR, ISERROR, ISLOGICAL, ISNA, ISNONTEXT, ISNUMBER, ISREF, ISTEXT

### ISERR

**Description**  
Determines if the specified expression returns an error value.

**Syntax**  
`ISERR (expression)`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>expression</td>
<td>Any expression.</td>
</tr>
</tbody>
</table>

**Remarks**  
If the expression returns any error except #N/A!, True is returned. Otherwise, False is returned.

**Example**  
This function returns True if A1 contains a formula that returns an error such as #NUM!:

`ISERR(A1)`

**See Also**  
ISBLANK, ISERROR, ISLOGICAL, ISNA, ISNONTEXT, ISNUMBER, ISREF, ISTEXT
### ISERROR

**Description**
Determines if the specified expression returns an error value.

**Syntax**
`ISERROR ( expression )`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>expression</code></td>
<td>Any expression.</td>
</tr>
</tbody>
</table>

**Remarks**
If `expression` returns any error value, such as #N/A!, #VALUE!, #REF!, #DIV/0!, #NUM!, #NAME?, or #NULL!, True is returned. Otherwise, False is returned.

**Examples**
This function returns True:

```
ISERROR(4/0)
```

This function returns False if A1 contains a formula that does not return an error.

```
ISERROR(A1)
```

**See Also**
ISBLANK, ISERR, ISLOGICAL, ISNA, ISNONTEXT, ISNUMBER, ISREF, ISTEXT

### ISEVEN

**Description**
Determines whether the specified number is even.

**Syntax**
`ISEVEN ( number )`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>number</code></td>
<td>Number value to be analyzed. Decimal values will be truncated to integers.</td>
</tr>
</tbody>
</table>

**Examples**
This function returns FALSE:

```
ISEVEN(5.8)
```

This function returns TRUE:

```
ISEVEN(4.5)
```

**See Also**
ISODD
ISLOGICAL

Description
Determines if the specified expression returns a logical value.

Syntax
ISLOGICAL (expression)

Remarks
If expression returns a logical value, True is returned. Otherwise, False is returned.

Example
This function returns True because ISBLANK returns a logical value:
ISLOGICAL(ISBLANK(A1))

See Also
ISBLANK, ISERR, ISERROR, ISNA, ISNONTEXT, ISNUMBER, ISREF, ISTEXT

ISNA

Description
Determines if the specified expression returns the value not available error.

Syntax
ISNA (expression)

Remarks
If expression returns the #N/A! error, True is returned. Otherwise, False is returned.

Example
This function returns True if cell A1 contains the NA ( ) function or returns the error value #N/A!:
ISNA(A1)

See Also
ISBLANK, ISERR, ISERROR, ISLOGICAL, ISNONTEXT, ISNUMBER, ISREF, ISTEXT

ISNONTEXT

Description
Determines if the specified expression is not text.

Syntax
ISNONTEXT (expression)

Table:

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>expression</td>
<td>Any expression.</td>
</tr>
</tbody>
</table>

Remarks
If expression returns the #N/A! error, True is returned. Otherwise, False is returned.

Example
This function returns True if cell A1 contains the NA ( ) function or returns the error value #N/A!:
ISNA(A1)

See Also
ISBLANK, ISERR, ISERROR, ISLOGICAL, ISNONTEXT, ISNUMBER, ISREF, ISTEXT
Remarks
If expression returns any value that is not text, True is returned. Otherwise, False is returned.

Examples
This function returns True if cell F3 contains a number or is a blank cell:
```
ISNONTEXT(F3)
```
This function returns False:
```
ISNONTEXT("text")
```

See Also
ISBLANK, ISERR, ISERROR, ISLOGICAL, ISNA, ISNUMBER, ISREF, ITEXT

---

**ISNUMBER**

Description
Determines if the specified expression is a number.

Syntax
```
ISNUMBER ( expression )
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>expression</td>
<td>Any expression.</td>
</tr>
</tbody>
</table>

Remarks
If expression returns a number, True is returned. Otherwise, False is returned. If expression returns a number represented as text (for example, “12”), False is returned.

Examples
This function returns True:
```
ISNUMBER(123.45)
```
This function returns False:
```
ISNUMBER("123")
```

See Also
ISBLANK, ISERR, ISERROR, ISLOGICAL, ISNA, ISNONTEXT, ISREF, ITEXT

---

**ISODD**

Description
Determines whether the specified number is odd.

Syntax
```
ISODD ( number )
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>Number value to be analyzed. Decimal values will be truncated to integers.</td>
</tr>
</tbody>
</table>
Examples

This function returns TRUE:

\text{ISODD}(5.8)

This function returns False:

\text{ISODD}(4.5)

See Also

ISEVEN

\textbf{ISREF}

\textbf{Description}

Determines if the specified expression is a range reference.

\textbf{Syntax}

\text{ISREF} ( \textit{expression} )

\begin{tabular}{ll}
\textbf{Argument} & \textbf{Description} \\
\textit{expression} & Any expression. \\
\end{tabular}

\textbf{Remarks}

If \textit{expression} returns a range reference, True is returned. Otherwise, False is returned.

\textbf{Example}

This function returns True:

\text{ISREF}(A3)

See Also

ISBLANK, ISERR, ISERROR, ISLOGICAL, ISNA, ISNONTEXT, ISNUMBER, ISTEXT

\textbf{ISTEXT}

\textbf{Description}

Determines if the specified expression is text.

\textbf{Syntax}

\text{ISTEXT} ( \textit{expression} )

\begin{tabular}{ll}
\textbf{Argument} & \textbf{Description} \\
\textit{expression} & Any expression. \\
\end{tabular}

\textbf{Remarks}

If \textit{expression} returns text, True is returned. Otherwise, False is returned.

\textbf{Example}

This function returns True:

\text{ISTEXT}("2nd Quarter")

See Also

ISBLANK, ISERR, ISERROR, ISLOGICAL, ISNA, ISNONTEXT, ISNUMBER, ISREF
KURT

Description Computes the kurtosis of a list of numbers.

Syntax KURT (number_list)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number_list</td>
<td>A list of between 1 and 30 arguments consisting of values, cell references, range references, and/or array constants. The argument(s) must contain at least 4 numbers. All numeric values, including 0, are used. Text and logical values in array constants and cells referenced by this function are ignored. Text entered in the argument list will be evaluated as a number, if possible; otherwise it will cause a #VALUE! error. Logical values entered into the argument list are evaluated as numeric 1 if TRUE and numeric 0 if FALSE.</td>
</tr>
</tbody>
</table>

Remarks Kurtosis measures the peakedness of the curve around its mean. A kurtosis of zero is approximately the same as the normal distribution; a larger kurtosis indicates a more pronounced central peak, while a smaller kurtosis indicates a flattened distribution around the peak.

Kurtosis is also known as the fourth movement, the first three being mean, standard deviation, and skewness.

Equation \[
\frac{n(n+1)}{(n-1)(n-2)(n-3)} \sum \left( \frac{x_j - \bar{x}}{s} \right)^4 - \frac{3(n-1)^2}{(n-2)(n-3)}
\]

where \( s \) is the sample standard deviation.

Example This function returns -1.2:

KURT(1,2,3,4,5,6,7)

See Also AVEDEV, AVERAGE, COUNT, DEVSQ, SKEW

LARGE

Description Computes the \( x \)th largest value in a list. If you set \( x \) at 5, LARGE will sort the values in ascending order and return the 5th largest value.

Syntax LARGE (numbers, \( x \))
**Example**

This function returns 7:

```
LARGE({1,2,3,4,5,6,7,8,9,15},4)
```

**See Also**

MAX, MIN, SMALL

---

**LCM**

**Description**

Computes the least common multiple of a specified set of values. This number is the smallest that can be divided evenly by all the numbers in the set, with no fraction or remainder.

**Syntax**

```
LCM (number_list)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number_list</td>
<td>A list of as many as 29 arguments in the form of numbers, cell references, range references, and array constants. The values in the arguments must be positive integers. Decimal values will be truncated to integers. Non-numeric arguments will return errors.</td>
</tr>
</tbody>
</table>

**Examples**

This function returns 16:

```
LCM(2,4,8,16)
```

This function returns 60

```
LCM(15,20)
```

**See Also**

GCD
**LEFT**

**Description**
Returns the leftmost characters from the specified text string.

**Syntax**
`LEFT ( text [, num_chars] )`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>text</code></td>
<td>Any text string.</td>
</tr>
<tr>
<td><code>[num_chars]</code></td>
<td>Optional. The number of characters to return. This value must be greater than or equal to zero. If <code>num_chars</code> is greater than the number of characters in <code>text</code>, the entire string is returned. If this argument is omitted, 1 is used.</td>
</tr>
</tbody>
</table>

**Examples**
This function returns 2:
`LEFT("2nd Quarter")`

This function returns 2nd:
`LEFT("2nd Quarter", 3)`

**See Also**
MID, RIGHT

**LEN**

**Description**
Returns the number of characters in the supplied text string.

**Syntax**
`LEN ( text )`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>text</code></td>
<td>Any text string. Spaces in the string are counted as characters.</td>
</tr>
</tbody>
</table>

**Examples**
This function returns 11:
`LEN("3rd Quarter")`

This function returns 3:
`LEN("1-3")`

**See Also**
EXACT, SEARCH
**LINEST**

**Description**
Computes multiple linear regression for a group of observations relative to a number of independent variables. Also optionally computes statistics related to the regression.

**Note**
This is an array function. For information, see “Array Functions” on page 13.

**Syntax**
LINEST (known_y’s [, known_x’s] [, constant] [, statistics])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>known_y’s</td>
<td>A list of observed values for the dependent variable. This must be a range reference or an array constant containing all numeric values and no empty cells.</td>
</tr>
<tr>
<td></td>
<td>For a regression with a single independent variable, enter any type of range. For a regression with multiple independent variables, enter a single row or column of values.</td>
</tr>
<tr>
<td></td>
<td>If you are using array constants, see “Array Constants in Arguments” on page 9 for information on how to enter the array.</td>
</tr>
<tr>
<td>[known_x’s]</td>
<td>Optional. A list of observed values for the independent variable(s). This must be a range reference or an array constant containing all numeric values and no empty cells.</td>
</tr>
<tr>
<td></td>
<td>For a regression with a single independent variable, when known_y’s is a single row or column, enter a range that exactly matches the size and shape of the known_y’s range. When known_y’s is more than one row and column, known_x’s must be a range containing the same number of values as the known_y’s range, although the two ranges may be different shapes.</td>
</tr>
<tr>
<td></td>
<td>For a regression with multiple independent variables, when known_y’s is a single row, enter a contiguous row of values for each independent variable. The number of columns must match the number of columns in known_y’s. Similarly, when known_y’s is a single column, enter a contiguous column of values for each independent variable, ensuring that the number of rows matches known_y’s.</td>
</tr>
<tr>
<td></td>
<td>If you are using array constants, see “Array Constants in Arguments” on page 9 for information on how to enter the array.</td>
</tr>
<tr>
<td></td>
<td>To fit the regression to a formula that uses polynomials, see “Polynomials,” below.</td>
</tr>
<tr>
<td></td>
<td>When you omit this argument, the function assumes a single independent variable of the sequence {1, 2, 3, ...}.</td>
</tr>
</tbody>
</table>
## Results range

Before entering this function, select a results range of the following size:

- It must have the same number of columns as your regression has independent variables, plus one. (If your regression has 3 independent variables, your results range must have 4 columns.)
- It must have 1 row if `statistics` is False, 5 rows if `statistics` is true.

## Results map

This map shows how LINEST fills the results range with data and statistics.

<table>
<thead>
<tr>
<th>Coefficient for ( x_2 )</th>
<th>Coefficient for ( x_1 )</th>
<th>Constant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard error for ( x_2 )</td>
<td>Standard error for ( x_1 )</td>
<td>Standard error for constant</td>
</tr>
<tr>
<td>Coefficient of determination</td>
<td>Standard error for the Y estimate</td>
<td>#N/A</td>
</tr>
<tr>
<td>F-test statistic</td>
<td>Degrees of freedom</td>
<td>#N/A</td>
</tr>
<tr>
<td>Sum of squares attributed to the regression</td>
<td>Residual sum of squares</td>
<td>#N/A</td>
</tr>
</tbody>
</table>

In the last three statistics rows, for results ranges wider than three columns, \#N/A will appear in the third and subsequent columns. This is correct.

## Coefficients

The coefficients computed by LINEST are displayed in the first row of the results range. For a multiple regression, the coefficients are displayed in the reverse order of the independent variables. (If your regression has 3 independent variables, the coefficient associated with the 1st variable is displayed in the 3rd column, the coefficient associated with the 2nd variable in the 2nd column, and the coefficient associated with the 3rd variable in the 1st column.)
The last cell of the first row of the results range displays the constant value, which will be 0 if \textit{constant} is False.

If there is only one independent variable and \textit{constant} is True, then the two coefficients will equal the results of the SLOPE and INTERCEPT functions when calculated on the same data.

Coefficients of 0 indicate either that there are not enough data points for the number of independent variables, or that some of the independent variables are too closely related.

**Equation**

The regression computed by LINEST attempts to fit the following formula:

$$y = C_1x_1 + C_2x_2 + \ldots + C_nx_n + b$$

where \( C \) is the coefficient, \( n \) is the number of independent variables, and \( b \) is the constant.

**Statistics**

If \textit{statistics} is True, the results range displays the following statistics in the following locations.

<table>
<thead>
<tr>
<th>Statistics</th>
<th>Row/Col</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard Error Values</td>
<td>row 2</td>
<td>The error values correspond to the coefficients displayed directly above them. This is computed by dividing up the standard error for the Y estimate (see below) into components for each independent variable. It is an estimate of how much the errors in individual independent variable measurements contribute to the overall error.</td>
</tr>
<tr>
<td>Coefficient of determination</td>
<td>row 3, col 1</td>
<td>This statistic is similar in concept to the correlation coefficient computed by the RSQ function. It is a number between 0 and 1 that measures goodness of fit: low numbers indicate a poor fit, high numbers a good fit.</td>
</tr>
<tr>
<td>Standard error for the Y estimate</td>
<td>row 3, col 2</td>
<td>This is computed by taking the square root of the residual sum of squares (see below) divided by the degrees of freedom (see below).</td>
</tr>
<tr>
<td>F-test statistic</td>
<td>row 4, col 1</td>
<td>This statistic can be used with FINV to do null hypothesis testing on the overall goodness of fit of the regression.</td>
</tr>
</tbody>
</table>
Using Statistics

A user who wishes to determine if the formula returned by LINEST is a good fit for the data in the regression should perform the following two tests.

- **Overall goodness of fit.** To determine if the overall regression is a good fit, you can do null hypothesis testing using the F-test statistic returned by LINEST. Compare the F-test statistic to the F-rejection value calculated by running FINV with the probability of your choice (usually 0.01 or 0.05), and a denominator value of the degrees of freedom statistic returned by LINEST.

  If the F-test statistic is larger than the F-rejection value, then the null hypothesis is rejected and this is a good fit.

- **Goodness of fit of each independent variable.** After determining that the overall regression is a good fit, test each independent variable to determine whether it adds to the soundness of the regression as a whole. Do this by finding the T-distribution of the T-observed statistic. The T-observed statistic is computed by dividing the coefficient of the variable you want to test by its standard error.

  Find the T-distribution by running TDIST using the absolute value of T-observed, the degrees of freedom statistic returned by LINEST, and a one-tailed distribution.

  If the probability returned by TDIST is larger than the rejection criteria (usually 0.01 or 0.05), then the variable you tested may not be contributing to the regression.

Polynomials

To create a polynomial regression, the data for the independent variable must be set up carefully. Create a worksheet in which column A represents \( x \) and contains the appropriate data or formulas. For column B, which will contain \( x^2 \), fill the cells with a formula such as \( =A1^\text{COLUMN()} \), which will raise the data in column A to the 2nd power. Fill column C with the same formula to raise the data in column A to the 3rd power.
Example
The following example of multiple linear regression attempts to correlate physiological data on preadolescent boys with their maximal oxygen uptake. The data are given in the table below.

<table>
<thead>
<tr>
<th>Maximal O2 Uptake (y)</th>
<th>Age (years)</th>
<th>Height (cm)</th>
<th>Weight (kilos)</th>
<th>Chest Depth (cm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.54</td>
<td>8.4</td>
<td>132.0</td>
<td>29.1</td>
<td>14.4</td>
</tr>
<tr>
<td>1.74</td>
<td>8.7</td>
<td>135.5</td>
<td>29.7</td>
<td>14.5</td>
</tr>
<tr>
<td>1.32</td>
<td>8.9</td>
<td>127.7</td>
<td>28.4</td>
<td>14.0</td>
</tr>
<tr>
<td>1.50</td>
<td>9.9</td>
<td>131.1</td>
<td>28.8</td>
<td>14.2</td>
</tr>
<tr>
<td>1.46</td>
<td>9.0</td>
<td>130.0</td>
<td>25.9</td>
<td>13.6</td>
</tr>
<tr>
<td>1.35</td>
<td>7.7</td>
<td>127.6</td>
<td>27.6</td>
<td>13.9</td>
</tr>
<tr>
<td>1.53</td>
<td>7.3</td>
<td>129.9</td>
<td>29.0</td>
<td>14.0</td>
</tr>
<tr>
<td>1.71</td>
<td>9.9</td>
<td>138.1</td>
<td>33.6</td>
<td>14.6</td>
</tr>
<tr>
<td>1.27</td>
<td>9.3</td>
<td>126.6</td>
<td>27.7</td>
<td>13.9</td>
</tr>
<tr>
<td>1.50</td>
<td>8.1</td>
<td>131.8</td>
<td>30.8</td>
<td>14.5</td>
</tr>
</tbody>
</table>

The data were entered in a worksheet, the LINEST function was run, and the following results range was returned.

| 0.0345 | -0.0234 | 0.0516 | -0.0352 | -4.7747 |
| 0.0852 | 0.0134  | 0.0662 | 0.0154  | 0.6628  |
| 0.9675 | 0.0372  | #N/A   | #N/A    | #N/A    |
| 37.2037| 5       | #N/A   | #N/A    | #N/A    |
| 0.2060 | 0.0069  | #N/A   | #N/A    | #N/A    |

See Also
GROWTH, INTERCEPT, LOGEST, SLOPE, TREND

LN

Description
Returns the natural logarithm (based on the constant e) of a number.

Syntax
LN ( number )

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>Any positive real number.</td>
</tr>
</tbody>
</table>

Remarks
LN is the inverse of the EXP function.
LOG

Description
Returns the logarithm of a number to the specified base.

Syntax
LOG ( number [, base])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>Any positive real number.</td>
</tr>
<tr>
<td>[base]</td>
<td>Optional. The base of the logarithm. If this argument is omitted, 10 is used.</td>
</tr>
</tbody>
</table>

Examples
This function returns 0:
LOG(1)

This function returns 1:
LOG(10)

See Also
EXP, LN, LOG10

LOG10

Description
Returns the base-10 logarithm of a number.

Syntax
LOG10 ( number)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>Any positive real number.</td>
</tr>
</tbody>
</table>

Examples
This function returns 2.41:
LOG10(260)
This function returns 2:
\[ \log_{10}(100) \]

See Also
EXP, LN, LOG

---

**LOGEST**

**Description**
Computes multiple exponential regression for a group of observations relative to a number of independent variables. Also optionally computes statistics related to the regression.

This type of regression is often applied to growth situations. Because of the type of formula and the often limited amount of data the formula is based on, using these formulas to estimate values outside of the range of data is dangerous, since they are likely to give exponentially wrong results.

Another problematic aspect of this function is that it assumes the range of data in question fits an exponential model. While this assumption is appropriate in some situations (for example, radioactive decay), in many other situations it is misleading, particularly in regards to economic data.

**Note**
This is an array function. For information, see “Array Functions” on page 13.

**Syntax**

\[
\text{LOGEST} \left( \text{known}_y'\text{s}, \text{known}_x'\text{s}, \text{constant}, \text{statistics} \right)
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
</table>
| known_y's    | A list of observed values for the dependent variable. This must be a range reference or an array constant containing all numeric values and no empty cells.

For a regression with a **single independent variable**, enter any type of range. For a regression with **multiple independent variables**, enter a single row or column of values.

If you are using array constants, see “Array Constants in Arguments” on page 9 for information on how to enter the array.
Results range

Before entering this function, select a results range of the following size:

- It must have the same number of columns as your regression has independent variables, plus one. (If your regression has 3 independent variables, your results range must have 4 columns.)
- It must have 1 row if `statistics` is False, 5 rows if `statistics` is true.

**Argument Description**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>[known_x's]</code></td>
<td>Optional. A list of observed values for the independent variable(s). This must be a range reference or an array constant containing all numeric values and no empty cells. For a regression with a single independent variable, when <code>known_y's</code> is a single row or column, enter a range that exactly matches the size and shape of the <code>known_y's</code> range. When <code>known_y's</code> is more than one row and column, <code>known_x's</code> must be a range containing the same number of values as the <code>known_y's</code> range, although the two ranges may be different shapes. For a regression with multiple independent variables, when <code>known_y's</code> is a single row, enter a contiguous row of values for each independent variable. The number of columns must match the number of columns in <code>known_y's</code>. Similarly, when <code>known_y's</code> is a single column, enter a contiguous column of values for each independent variable, ensuring that the number of rows matches <code>known_y's</code>. If you are using array constants, see “Array Constants in Arguments” on page 9 for information on how to enter the array. To fit the regression to a formula that uses polynomials, see “Polynomials,” below. When you omit this argument, the function assumes a single independent variable of the sequence {1, 2, 3, ...}.</td>
</tr>
<tr>
<td><code>[new_x's]</code></td>
<td>Optional. A list of values for the independent variable(s) that will compute the results of this function. This must be a range reference or an array constant containing all numeric values and no empty cells. You may enter any number of values in <code>new_x's</code>. The shape of the range only matters if this is a regression with multiple independent variables, in which case it must have the same number of rows or columns of independent variables as <code>known_x's</code>. If you omit this argument, the function uses the specified or default values in <code>known_x's</code>.</td>
</tr>
<tr>
<td><code>[constant]</code></td>
<td>Optional. A logical value that determines whether or not a constant is to be included. The constant allows the regression line to intercept the Y axis at a point other than (0,0). Use True (the default value if this argument is omitted) to include the constant. Use False to force the line to intercept the Y axis at 0.</td>
</tr>
</tbody>
</table>
Results map

This map shows how LOGEST fills the results range with data and statistics.

<table>
<thead>
<tr>
<th>Coefficient for x₂</th>
<th>Coefficient for x₁</th>
<th>Constant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard error for x₂</td>
<td>Standard error for x₁</td>
<td>Standard error for constant</td>
</tr>
<tr>
<td>Coefficient of determination</td>
<td>Standard error for the Y estimate</td>
<td>#N/A</td>
</tr>
<tr>
<td>F-test statistic</td>
<td>Degrees of freedom</td>
<td>#N/A</td>
</tr>
<tr>
<td>Sum of squares attributed to the regression</td>
<td>Residual sum of squares</td>
<td>#N/A</td>
</tr>
</tbody>
</table>

In the last three statistics rows, for results ranges wider than three columns, #N/A will appear in the third and subsequent columns. This is correct.

Coefficients

The coefficients computed by LOGEST are displayed in the first row of the results range. For a multiple regression, the coefficients are displayed in the reverse order of the independent variables. (If your regression has 3 independent variables, the coefficient associated with the 1st variable is displayed in the 3rd column, the coefficient associated with the 2nd variable in the 2nd column, and the coefficient associated with the 3rd variable in the 1st column.)

The last cell of the first row of the results range displays the constant value, which will be 1 if constant is False.

Coefficients of 1 indicate either that there are not enough data points for the number of independent variables, or that some of the independent variables are too closely related.

Equation

The regression computed by LOGEST attempts to fit the following formula:

\[ y = C_1 x_1 + C_2 x_2 + \ldots + C_n x_n + b \]

where \( C \) is the coefficient, \( n \) is the number of independent variables, and \( b \) is the constant.

This formula can be converted to a linear form if logarithms are taken of both sides. The formula then becomes:

\[ y = lC_1 x_1 + lC_2 x_2 + \ldots + lC_n x_n + lb \]

This is the same formula fitted by the LINEST function. LOGEST uses LINEST internally to fit the linearized form of the equation. It then takes the LINEST coefficients \( lC_1, lC_2, \ldots, lC_n \) and converts them to the LOGEST coefficients \( C_1, C_2, \ldots, C_n \) by applying the EXP function: \( C_j = \text{EXP}(lC_j) \). The same conversion is done with the constant, \( b \).
If `statistics` is True, the results range displays the following statistics in the following locations. Please note that these statistics apply to the linearized form of the equation, as explained above.

**Statistics**

- **Standard Error Values**
  - row 2
  - The error values correspond to the coefficients displayed directly above them. This is computed by dividing up the standard error for the Y estimate (see below) into components for each independent variable. It is an estimate of how much the errors in individual independent variable measurements contribute to the overall error.

- **Coefficient of determination**
  - row 3, col 1
  - This statistic is similar in concept to the correlation coefficient computed by the RSQ function. It is a number between 0 and 1 that measures goodness of fit: low numbers indicate a poor fit, high numbers a good fit.

- **Standard error for the Y estimate**
  - row 3, col 2
  - This is computed by taking the square root of the residual sum of squares (see below) divided by the degrees of freedom (see below).

- **F-test statistic**
  - row 4, col 1
  - This statistic can be used to do null hypothesis testing on the overall goodness of fit of the regression.

- **Degrees of Freedom**
  - row 4, col 2
  - This is determined by subtracting the number of independent variables from the number of values in `known_y`’s, less 1 if `constant` is True.

- **Sum of squares attributed to the regression**
  - row 5, col 1
  - This is determined by subtracting the residual sum of squares (below) from the total sum of squares of the differences between the values in `known_y`’s and `Y`. This statistic helps determine how well the formula fits the data.

- **Residual sum of squares**
  - row 5, col 2
  - This is determined by summing the squares of the differences between the values in `known_y`’s and the Y values computed by the formula you are trying to fit. It is a measure of the goodness of fit.

**Example**

This example of exponential regression uses the following data on the growth of the national debt of a small country.

<table>
<thead>
<tr>
<th>Year</th>
<th>National debt, in millions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1971</td>
<td>0.5</td>
</tr>
<tr>
<td>1972</td>
<td>1.1</td>
</tr>
<tr>
<td>1973</td>
<td>1.4</td>
</tr>
<tr>
<td>1974</td>
<td>2.0</td>
</tr>
<tr>
<td>1975</td>
<td>4.4</td>
</tr>
<tr>
<td>1976</td>
<td>20.0</td>
</tr>
<tr>
<td>1977</td>
<td>445.0</td>
</tr>
</tbody>
</table>
In this case, the debt figures are the known_y’s and we can use default values for the known_x’s, since the years increment by one.

If the national debt figures fill range A2:A8, the function

\[
\text{LOGEST(A2:A8,,TRUE,TRUE)}
\]

returns the following results range:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2.653283</td>
<td>0.057262</td>
</tr>
<tr>
<td>0.197756</td>
<td>0.064799</td>
</tr>
<tr>
<td>0.029699</td>
<td>1.046435</td>
</tr>
<tr>
<td>24.34261</td>
<td>5</td>
</tr>
<tr>
<td>26.65681</td>
<td>5.475134</td>
</tr>
</tbody>
</table>

See Also

GROWTH, INTERCEPT, LINEST, SLOPE, TREND

**LOGINV**

**Description**

Computes the inverse of the cumulative lognormal distribution of \(x\), where the logarithm of \(x\) is normally distributed.

**Syntax**

\[
\text{LOGINV(prob, mean, st_dev)}
\]

**Argument** | **Description**
--- | ---
prob | The probability, as computed by the LOGNORMDIST function.
mean | The arithmetic mean of the logarithm of \(x\).
st_dev | The standard deviation of the logarithm of \(x\).

**Remarks**

LOGINV assumes the logarithm of \(x\) is normally distributed, meaning that, if you graph the distribution with a logarithmic X axis, the bell-shaped curve will appear.

**Equation**

\[
\text{LOGINV(prob, } \mu, \sigma) = \text{EXP(} \mu + \sigma \times \text{NORMSINV(prob)})
\]

**Example**

This function returns 16.0002:

\[
\text{LOGINV(0.223218,18,20)}
\]

See Also

LOGNORMDIST

**LOGNORMDIST**

**Description**

Computes the cumulative lognormal distribution for \(x\), where the logarithm of \(x\) is normally distributed.

**Syntax**

\[
\text{LOGNORMDIST(x, mean, st_dev)}
\]
LOGNORMDIST assumes the logarithm of $x$ is normally distributed, meaning that, if you graph the distribution with a logarithmic X axis, the bell-shaped curve will appear.

**Equation**

$$\text{LOGNORMDIST} (x, \mu, \sigma) = \left( \frac{\ln(x) - \mu}{\sigma} \right)$$

**Example**

This function returns 0.2232:

```
LOGNORMDIST(16,18,20)
```
This function returns Detroit:

LOOKUP("North", A2:A7, B2:B7)

This function returns #N/A:


**See Also**

HLOOKUP, INDEX (non-array type), VLOOKUP

---

### LOWER

**Description**

Changes the characters in the specified string to lowercase characters. Numeric characters in the string are not changed.

**Syntax**

LOWERR (text)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>Any string.</td>
</tr>
</tbody>
</table>

**Examples**

This function returns 3rd quarter:

LOWERR("3rd Quarter")

This function returns john doe:

LOWERR("JOHN DOE")

**See Also**

PROPER, UPPER
MATCH

Description  A specified value is compared against values in a range. The position of the matching value in the search range is returned.

Syntax  MATCH (lookup_value, lookup_range, comparison)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lookup_value</td>
<td>The value against which to compare. It can be a number, text, or logical value or a reference to a cell that contains one of those values.</td>
</tr>
<tr>
<td>lookup_range</td>
<td>The range to search and contains only one row or one column. The range can contain numbers, text, or logical values.</td>
</tr>
<tr>
<td>comparison</td>
<td>A number that represents the type of comparison to be made between lookup_value and the values in lookup_range. When you omit this argument, comparison method 1 is assumed. When comparison is 1, the largest value that is less than or equal to lookup_value is matched. When using this comparison method, the values in lookup_range must be in ascending order (for example, ...–2, –1, 0, 1, 2..., A through Z, False, True). When comparison is 0, the first value that is equal to lookup_value is matched. When using this comparison method, the values in lookup_range can be in any order. When comparison is –1, the smallest value that is greater than or equal to lookup_value is matched. When using this comparison method, the values in lookup_range must be in descending order (for example, True, False, Z through A, ...2, 1, 0, –1, –2...).</td>
</tr>
</tbody>
</table>

Remarks  When using comparison method 0 and lookup_value is text, lookup_value can contain wildcard characters. The wildcard characters are * (asterisk), which matches any sequence of characters, and ? (question mark), which matches any single character. When no match is found for lookup_value, #N/A is returned.

Examples  The following examples use this worksheet.
This function returns 5:
MATCH(7600, B2:B7, 1)

This function returns 2:
MATCH("D*", A2:A7, 0)

**See Also**
HLOOKUP, INDEX (non-array type), LOOKUP, VLOOKUP

### MAX

**Description**
Returns the largest value in the specified list of numbers.

**Syntax**
MAX ( number_list )

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number_list</td>
<td>A list of up to 30 numbers separated by commas. The list may contain numeric values, cell references, range references, or array constants. Text in and logical values in cell references and arrays are ignored. Text entered in the argument list will be evaluated as a number, if possible; otherwise it will cause a #VALUE! error. Logical values entered into the argument list are evaluated as numeric 1 if TRUE and numeric 0 if FALSE.</td>
</tr>
</tbody>
</table>

**Examples**
This function returns 500:
MAX(50, 100, 150, 500, 200)

This function returns the largest value in the range:
MAX(A1:F12)

**See Also**
AVERAGE, MIN

### MAXA

**Description**
Returns the largest value in the specified list of numbers.

This function is equivalent to the MAX function, but its implementation treats text and logical values in cell and range references differently.

**Syntax**
MAXA ( number_list )
**MDETERM**

**Description**
Computes the determinant of a square matrix.

**Syntax**

\[
\text{MDTERM}(\text{array})
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>An array of numbers or a range of cells.</td>
</tr>
</tbody>
</table>

**Remarks**

The determinant can be used as a measurement of the singularity of a matrix. If the matrix represents the coefficients of a set of simultaneous equations, and the determinant is relatively small, this may indicate that one or more pairs of equations (i.e., rows) are closely related and the solution of the simultaneous equations may not be very accurate (if it is computed at all). A value of 0 indicates that the matrix is singular and not solvable (see MINVERSE).

**Note**
In contrast to the closely related functions MINVERSE and MMULT, the result of MDETERM is a single value. MDETERM is not an array function.

**Examples**
The first example uses this worksheet.
This function returns 33:

\[
\text{MDETERM}(A1:D4)
\]

This function returns 33:

\[
\text{MDETERM}([5,3,2,4;0,3,5,1;1,2,3,0;1,3,5,7])
\]

See Also
MINVERSE, MMULT

**MDURATION**

Description
Computes the modified duration for a security.

Syntax
MDURATION (settlement, maturity, rate, yield, frequency [, calendar_type])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>settlement</td>
<td>The date when the security is traded to the buyer. Dates in the argument list must be in the form of a serial number or text. Numbers will be truncated to integers.</td>
</tr>
<tr>
<td>maturity</td>
<td>The date the security expires and the remaining amount is paid to the investor. It must be later than settlement. Dates in the argument list must be in the form of a serial number or text. Numbers will be truncated to integers.</td>
</tr>
<tr>
<td>rate</td>
<td>The security’s annual coupon rate. The coupons pay at this rate divided by frequency.</td>
</tr>
<tr>
<td>yield</td>
<td>The security’s annual yield.</td>
</tr>
<tr>
<td>frequency</td>
<td>The number of interest payments per year. See “The frequency Argument” on page 18 for more information.</td>
</tr>
<tr>
<td>[calendar_type]</td>
<td>Optional. One of five methods of counting days for computing interest. See “The calendar_type Argument” on page 19 for more information.</td>
</tr>
</tbody>
</table>

Remarks
Modified duration is a direct measure of the sensitivity of a bond’s price to a change in its yield.

Equation
\[
\text{DURATION} (settlement, maturity, rate, yield, frequency, calendar_type) = \frac{1 + \frac{\text{yield}}{\text{frequency}}}{1 + \frac{\text{yield}}{\text{frequency}}}
\]
Examples

This function returns 7.2465:

\[
\text{MDURATION("3/17/89","3/17/99",0.07,0.08,2,1)}
\]

See Also

DURATION

---

**MEDIAN**

**Description**

Computes the median or middle value of a list of numbers arranged in ascending order.

**Syntax**

\[
\text{MEDIAN (number_list)}
\]

**Argument** | **Description**
--- | ---
number_list & A list of between 1 and 30 arguments consisting of numbers, cell references, range references, and/or array constants.

All numeric values, including 0, are used. Text and logical values in arrays and cells referenced by this function are ignored. Text entered in the argument list will be evaluated as a number, if possible; otherwise it will cause a #VALUE! error. Logical values entered into the argument list are evaluated as numeric 1 if TRUE and numeric 0 if FALSE.

**Remarks**

If there is an odd number of values in number_list, the middle value is returned. If there is an even number of values, the average of the two middle values is returned.

**Example**

This function returns 3:

\[
\text{MEDIAN(1,2,3,4,5)}
\]

This function returns 12:

\[
\text{MEDIAN(24,15,9,6,2,76,11,13)}
\]

**See Also**

AVERAGE, MODE

---

**MID**

**Description**

Returns the specified number of characters from a text string, beginning with the specified starting position.

**Syntax**

\[
\text{MID (text, start_position, num_chars)}
\]

**Argument** | **Description**
--- | ---

\[
text
\]

The string from which to return characters.
**Remarks**

If `start_position` plus the number of characters in `num_chars` exceeds the length of `text`, the characters from `start_position` to the end of `text` are returned.

**Examples**

This function returns Expenses:

```
MID("Travel Expenses", 8, 8)
```

This function returns 45:

```
MID("Part #45-7234", 7, 2)
```

**See Also**

CODE, FIND, LEFT, RIGHT, SEARCH

---

**MIN**

**Description**

Returns the smallest value in the specified list of numbers.

**Syntax**

```
MIN ( number_list )
```

**Examples**

This function returns 50:

```
MIN(50, 100, 150, 500, 200)
```

This function returns the smallest value in the range:

```
MIN(A1:F12)
```

**See Also**

AVERAGE, MAX
MINA

Description
Returns the smallest value in the specified list of numbers.

This function is equivalent to the MIN function, but its implementation treats text and logical values in cell and range references differently.

Syntax
MINA (number_list)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number_list</td>
<td>A list of between 1 and 30 arguments consisting of numbers, cell references, range references, and/or array constants. Text in cells referenced by this function is treated as the number 0 (this includes zero-length text). Text entered in the argument list will be evaluated as a number, if possible; otherwise it will cause a #VALUE! error. Text and logical values in arrays are ignored. Logical values referenced in cells or entered into the argument list are evaluated as numeric 1 if TRUE and numeric 0 if FALSE.</td>
</tr>
</tbody>
</table>

Examples
This function returns 50:
MINA(50,100,150,"500",200)

This function returns 0:
MINA(TRUE,FALSE)

See Also
AVERAGE, AVERAGEA, MAX, MAXA, MIN

MINUTE

Description
Returns the minute that corresponds to the supplied date.

Syntax
MINUTE (serial_number)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serial_number</td>
<td>The time as a serial number. The decimal portion of the number represents time as a fraction of the day.</td>
</tr>
</tbody>
</table>

Remarks
The result is an integer ranging from 0 to 59.

Examples
This function returns 36:
MINUTE(34506.4)
This function returns 48:
MINUTE(34399.825)

See Also
DAY, HOUR, MONTH, NOW, SECOND, WEEKDAY, YEAR

MINVERSE

Description
Computes the inverse of a square matrix.

Note
This is an array function. For information, see “Array Functions” on page 13.

Syntax
MINVERSE (array)

Argument      Description
array          An array of numbers or range of cells.

Remarks
The inverse is often used to solve series of simultaneous equations. The inverse has the property that when it is multiplied (using MMULT) by the original matrix, the identity matrix is the result--1’s on the diagonal, 0’s (or very small numbers because of round-off errors in computers) in all other locations.

Examples
The first example uses this worksheet.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8</td>
<td>-1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>-4</td>
<td>-3</td>
<td>-9</td>
</tr>
<tr>
<td>3</td>
<td>-5</td>
<td>5</td>
<td>7</td>
</tr>
</tbody>
</table>

MINVERSE(A1:C3) returns the following results range:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.4897959184</td>
<td>0.3469387755</td>
<td>0.306122449</td>
</tr>
<tr>
<td>1.4897959184</td>
<td>1.346938776</td>
<td>1.306122449</td>
</tr>
<tr>
<td>-0.7142857143</td>
<td>-0.7142857143</td>
<td>-0.57142857144</td>
</tr>
</tbody>
</table>
Example Problem

Given: \( x + y = 3 \) and \( x - y = 2 \), solve for \( x \) and \( y \):

To solve using MINVERSE and MMULT:

1. Enter the coefficient and constant array matrices.
2. Select a four-cell area (A4:B5 in the example) to hold the results of the formula.
3. Type in the formula \( \text{MINVERSE(A1:B2)} \).
4. Press CTRL + SHIFT + ENTER to enter the array formula so that it applies to all four cells in the selected area.
5. Select a two-cell column area where you want the answer to appear.
6. Multiply the inverse coefficient array matrix and the constant array matrix by typing \( \text{MMULT(A4:B5,D4:D5)} \), then pressing CTRL + SHIFT + ENTER.

See Also

MDETERM, MMULT

**MIRR**

Description

Returns the modified internal rate of return for a series of periodic cash flows.

Syntax

\[
\text{MIRR} \ (\text{cash\_flows}, \text{finance\_rate}, \text{reinvest\_rate})
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cash_flow</td>
<td>A reference to a range that contains values for which to calculate the modified internal rate of return. The values must contain at least one positive and one negative value. During calculation, MIRR uses the order in which the values appear to determine the order of cash flow. Values that represent cash received should be positive; negative values represent cash paid. Text, logical values, and empty cells in the range are ignored.</td>
</tr>
</tbody>
</table>
Remarks

The modified internal rate of return considers the cost of the investment and the interest received on the reinvestment of cash.

Examples

The following examples use this worksheet.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Investment</td>
</tr>
<tr>
<td>2</td>
<td>1993 income</td>
</tr>
<tr>
<td>3</td>
<td>1994 income</td>
</tr>
<tr>
<td>4</td>
<td>1995 income</td>
</tr>
<tr>
<td>5</td>
<td>1996 income</td>
</tr>
<tr>
<td>6</td>
<td>1997 income</td>
</tr>
</tbody>
</table>

This function returns 5.20 percent:

\[
\text{MIRR(B1:B6, 12\%, 8\%)}
\]

This function returns –40.93 percent:

\[
\text{MIRR(B1:B3, 12\%, 8\%)}
\]

See Also

IRR, NPV, RATE

**MMULT**

Description

Performs matrix multiplication on two arrays.

**Note** This is an array function. For information, see “Array Functions” on page 13.

Syntax

\[
\text{MMULT (array1, array2)}
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array1 and array2</td>
<td>Two range references. All cells must contain numeric values or the function will return the #VALUE! error. The number of rows in array2 must be the same as the number of columns in array1.</td>
</tr>
</tbody>
</table>

Results range

Before entering this function, select a results range with the same number of rows as array1 and the same number of columns as array2. If you select too few cells, some cells will be truncated. If you select too many cells, the extra cells will contain the #N/A error.
Remarks
To fill the results range, MMULT computes SUMPRODUCT (see page 238) with the first row of array1 and first column of array2 as the arguments. The result goes in the first row and column of the results range. MMULT then computes SUMPRODUCT on the second row of array1 and the first column of array2, and puts the result in the second row, first column of the results range. The function proceeds in this manner until the results range is filled.

Examples
The following example uses this worksheet.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

MMULT(A1:C2,A4:B6) returns the following results range:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>54</td>
<td>70</td>
</tr>
<tr>
<td>22</td>
<td>62</td>
</tr>
</tbody>
</table>

MOD
Description
Returns the remainder after dividing a number by a specified divisor.

Syntax
MOD (number, divisor)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>Any number.</td>
</tr>
<tr>
<td>divisor</td>
<td>Any nonzero number. If divisor is 0, #DIV/0! is returned.</td>
</tr>
</tbody>
</table>

Examples
This function returns 1:

MOD(-23, 3)

This function returns -2:

MOD(-23, -3)

See Also
INT, ROUND, TRUNC

MODE
Description
Computes the most frequent value in a list of numbers (the number that appears most often in the list).

Syntax
MODE (number_list)
Remarks: If no number is repeated in \textit{number\_list}, the error value \#N/A is returned.

Example: This function returns 3:

\[ \text{MODE}(1,2,3,3,4) \]

See Also: AVERAGE, MEDIAN

\begin{itemize}
  \item \textbf{MONTH}
      \begin{description}
        \item[Description] Returns the month that corresponds to the supplied date.
        \item[Syntax] \texttt{MONTH ( serial\_number) }
        \item[Argument Description] \begin{tabular}{ll}
          \textit{serial\_number} & The date as a serial number or as text (for example, 06-21-94 or 21-Jun-94). \\
        \end{tabular}
        \item[Remarks] MONTH returns a number ranging from 1 (January) to 12 (December).
        \item[Examples] This function returns 6:
          \begin{tabular}{l}
            \texttt{MONTH("06-21-94")} \\
          \end{tabular}
          This function returns 10:
          \begin{tabular}{l}
            \texttt{MONTH(34626)} \\
          \end{tabular}
        \item[See Also] DAY, HOUR, MINUTE, NOW, SECOND, TODAY, WEEKDAY, YEAR
      \end{description}
\end{itemize}

\begin{itemize}
  \item \textbf{MROUND}
      \begin{description}
        \item[Description] Rounds a specified number to an even integral multiple of a specified factor.
        \item[Syntax] \texttt{MROUND ( number, factor) }
      \end{description}
\end{itemize}
### MROUND

**Description**
Computes the multiple of up to 29 numbers.

**Syntax**
MROUND (number)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>The value to be rounded. Number must have the same sign as factor.</td>
</tr>
<tr>
<td>factor</td>
<td>Lowest factor of desired result after rounding. Factor must have the same sign as number.</td>
</tr>
</tbody>
</table>

**Remarks**
If the remainder is exactly half of factor, the result is rounded away from 0. (See examples.)

**Examples**
This function returns 14:

MROUND(13,2)

This function returns -3.3:

MROUND(-3.8,-1.1)

**See Also**
ODD, EVEN, ROUND, ROUNDUP, ROUNDDOWN

---

### MULTINOMIAL

**Description**
Computes the multinomial of up to 29 numbers.

**Syntax**
MULTINOMIAL (number)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>A list of as many as 29 arguments. The list can contain numbers, ranges containing values, or array constants. Decimal values are truncated to integers.</td>
</tr>
</tbody>
</table>

**Equation**
MULTINOMIAL(a + b + c) = \( \frac{(a + b + c)!}{(a! + b! + c!)} \)

**Examples**
This function returns 12600:

MULTINOMIAL(1,2,3,4)

This function returns 12600:

MULTINOMIAL(1.9,2.9,3.9,4.9)

**See Also**
FACT
N

Description Tests the supplied value and returns the value if it is a number.

Syntax N ( value)

Remarks Numbers are returned as numbers, serial numbers formatted as dates are returned as serial numbers, and the logical function TRUE() is returned as 1. All other expressions return 0.

Examples This function returns 32467:
N(32467)

This function returns 1 if A4 contains the logical function TRUE:
N(A4)

See Also T, VALUE

NA

Description Returns the error value #N/A, which represents “not available.”

Syntax NA ( )

Remarks Use NA to mark cells that lack data without leaving them empty. Empty cells may not be correctly represented in some calculations.

Although NA does not use arguments, you must supply the empty parentheses to correctly reference the function.

See Also ISNA

NEGBINOMDIST

Description Computes the negative binomial distribution, which is used to determine probabilities on repeated tests where each test is independent of every other test (that is, the probability is the same for each test).

Syntax NEGBINOMDIST (failures, threshold, probability)
### NEGBINOMDIST

#### Argument Description

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>failures</td>
<td>The number of failures you expect in the trial. It must be a positive integer or 0.</td>
</tr>
<tr>
<td>threshold</td>
<td>The number of successes the trial requires. It must be a positive integer. It may not be 0.</td>
</tr>
<tr>
<td>probability</td>
<td>The chance of success on an individual attempt in the trial. It must be a number larger than 0 and smaller than 1. For example, the probability of throwing heads on a coin toss is 0.5.</td>
</tr>
</tbody>
</table>

#### Remarks

Coin tosses or die rolls are examples of uses for NEGBINOMDIST. The probability returned measures the likelihood that there will be a certain number of failures before a certain number of successes occurs. This can be used to determine how many tests to run to achieve the desired results.

NEGBINOMDIST is similar to BINOMDIST, only in NEGBINOMDIST the number of successes is fixed and the number of trials is variable, while in BINOMDIST the number of successes is variable and the number of trials is fixed.

#### Equation

\[
\binom{f + t - 1}{t - 1} p^t (1 - p)^f
\]

...where \( f \) is failures, \( t \) is threshold, and \( p \) is probability.

#### Example

This function returns 0.1875:

\[
\text{NEGBINOMDIST(2, 2, 0.5)}
\]

#### See Also

BETADIST, BINOMDIST, CHIDIST, COMBIN, CRITBINOM, EXPONDIST, GAMMADIST, NORMDIST, POISSON

### NETWORKDAYS

#### Description

Computes the number of whole working days between the specified start date and the specified end date.

#### Syntax

\[
\text{NETWORKDAYS (start_date, end_date, [, holidays])}
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>start_date</td>
<td>First date of the period.</td>
</tr>
<tr>
<td>end_date</td>
<td>Last date of the period.</td>
</tr>
<tr>
<td>holidays</td>
<td>Optional. A list containing numbers, a range of cells, or an array constant representing the dates of holidays.</td>
</tr>
</tbody>
</table>

#### Remarks

Counts both the first and last days if appropriate.

Excludes weekends and any days in the list of holidays.
Useful for computations that are dependent on the actual hours or days worked in a time period.

**Examples**

This function returns 260:

```
NETWORKDAYS(1,365)
```

**See Also**

WEEKDAY

### NOMINAL

**Description**

Computes the nominal annual interest rate, given the effective interest rate. This will remove the effect of yearly compounding on the specified interest rate.

**Syntax**

```
NOMINAL ( effective_rate, periods)
```

**Argument Description**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>effective_rate</td>
<td>The interest rate that reflects the effect of compounding.</td>
</tr>
<tr>
<td>periods</td>
<td>The number of compounding periods per year. Decimal values are truncated to integers.</td>
</tr>
</tbody>
</table>

**Examples**

This function returns 0.0675:

```
NOMINAL(0.0696,12)
```

This function returns 0.0684:

```
NOMINAL(0.0696,2)
```

**See Also**

EFFECT

### NORMDIST

**Description**

Computes the normal distribution.

**Syntax**

```
NORMDIST ( x, mean, st_dev, cumulative)
```

**Argument Description**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>The value at which you want to evaluate the function.</td>
</tr>
<tr>
<td>mean</td>
<td>The arithmetic mean of the distribution.</td>
</tr>
<tr>
<td>st_dev</td>
<td>The standard deviation of the distribution. It must be a number greater than 0.</td>
</tr>
</tbody>
</table>


**Remarks**

You may use the STANDARDIZE function to “translate” the first three arguments of this function into a form that can be used in the NORMSDIST function.

**Equation**

\[ f(x) = \frac{1}{\sqrt{2\pi} \sigma} e^{-\frac{(x - \mu)^2}{2\sigma^2}} \]

**Examples**

This function returns 0.5:

\[
\text{NORMDIST (50, 50, 4, TRUE)}
\]

**See Also**

BETADIST, BINOMDIST, CHIDIST, COMBIN, CRITBINOM, FDIST, GAMMADIST, GAMMAINV, NORMINV, NORMDIST, POISSON

---

## NORMINV

**Description**

Computes the inverse of the cumulative normal distribution. This is the inverse of the NORMDIST function when the last argument is TRUE.

**Syntax**

NORMINV (probability, mean, st_dev)

**Table**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>probability</td>
<td>A number between 0 and 1 that indicates the probability associated with the cumulative normal distribution.</td>
</tr>
<tr>
<td>mean</td>
<td>The arithmetic mean of the distribution.</td>
</tr>
<tr>
<td>st_dev</td>
<td>The standard deviation of the distribution. It must be a number greater than 0.</td>
</tr>
</tbody>
</table>

**Examples**

This function returns 21.64:

\[
\text{NORMINV (0.95, 20, 1)}
\]

**See Also**

NORMDIST
### NORMSDIST

**Description**
Computes the standard normal cumulative distribution (often called “the bell curve”).

**Syntax**

\[
\text{NORMSDIST} \ (x)
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x)</td>
<td>The value at which you want to evaluate the function.</td>
</tr>
</tbody>
</table>

**Remarks**
This is a version of the NORMDIST function that assumes the mean of the distribution is 0 and the standard deviation is 1. You can adjust values from a distribution with a different mean and/or standard deviation by plugging those values into the STANDARD function. The result of the STANDARD function is the argument for NORMSDIST.

**Equation**

\[
f(x) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{1}{2} \left( \frac{x - \mu}{\sigma} \right)^2}
\]

**Examples**
This function returns 0.89:

\[
\text{NORMSDIST} \ (1.25)
\]

**See Also**
BETADIST, BINOMDIST, CHIDIST, COMBIN, CRITBINOM, FDIST, LOGNORMDIST, NORMDIST, NORMSINV, POISSON

### NORMSINV

**Description**
Computes the inverse of the standard normal distribution. This is the inverse of the NORMDIST function.

**Syntax**

\[
\text{NORMSINV} \ (probability)
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(probability)</td>
<td>A number between 0 and 1 that indicates the probability associated with the standard normal distribution.</td>
</tr>
</tbody>
</table>

**Remarks**
This

**Examples**
This function returns 0.25:

\[
\text{NORMSINV} \ (0.599)
\]

**See Also**
NORMDIST
**NOT**

**Description**
Returns a logical value that is the opposite of its value.

**Syntax**
\[ \text{NOT}( \text{logical} ) \]

**Remarks**
If \( \text{logical} \) is false, \( \text{NOT} \) returns True. Conversely, if \( \text{logical} \) is true, \( \text{NOT} \) returns False.

**Examples**
- This function returns False:
  \[ \text{NOT}(\text{TRUE}()) \]
- This function returns False:
  \[ \text{NOT}(\text{MONTH("12/25/94") = 12}) \]

**See Also**
AND, IF, OR

---

**NOW**

**Description**
Returns the current date and time as a serial number.

**Syntax**
\[ \text{NOW}() \]

**Remarks**
In a serial number, numbers to the left of the decimal point represent the date; numbers to the right of the decimal point represent the time. The result of this function changes only when a recalculation of the worksheet occurs.

**See Also**
DATE, DAY, HOUR, MINUTE, MONTH, SECOND, TODAY, WEEKDAY, YEAR

---

**NPER**

**Description**
Returns the number of periods of an investment based on regular periodic payments and a fixed interest rate.

**Syntax**
\[ \text{NPER}( \text{interest}, \text{pmt}, \text{pf} [, \text{fv}] [, \text{type})} \]

**Argument**  **Description**
---
\( \text{interest} \)  The fixed interest rate.
Examples

This function returns 36.67:
\[ \text{NPER}(12\%/12, -350, -300, 16000, 1) \]

This function returns 36.98:
\[ \text{NPER}(1\%, -350, -300, 16000) \]

See Also

FV, CUMIPMT, CUMPRINC, IPMT, PMT, PPMT, PV, RATE

NPV

Description

Returns the net present value of an investment based on a series of periodic payments and a discount rate.

Syntax

\[
\text{NPV} \left( \text{discount\_rate}, \text{value\_list} \right)
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>discount_rate</td>
<td>The rate of discount for one period. A list of as many as 29 arguments or a reference to a range that contains values that represent payments and income. During calculation, NPV uses the order in which the values appear to determine the order of cash flow. Numbers, empty cells, and text representations of numbers are included in the calculation. Errors and text that cannot be translated into numbers are ignored. If value_list is a range reference, only numeric data in the range is included in the calculation. Other types of data in the range, such as empty cells, logical values, text, and error values, are ignored.</td>
</tr>
</tbody>
</table>
Remarks
The time span NPV uses for calculation begins one period before the first cash flow date and ends when the last cash flow payment is made. This function is based on future cash flows. When your first cash flow occurs at the beginning of the first period, the first value must be added to the NPV result, not supplied as a value in `value_list`.

Example
This function returns 811.57:
\[
\text{NPV}(8\%, -12000, 3000, 3000, 3000, 7000)
\]

See Also
FV, IRR, PV

---

**OCT2BIN**

Description
Converts an octal number (base 8) to a binary number (base 2).

Syntax

\[
\text{OCT2BIN}(\text{integer}[,\text{places}])
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>integer</code></td>
<td>Any negative or positive whole number or zero. For the purposes of this function, the integer entered must be small enough to convert within the limits of the target. Any integer between 0 and 777 may be used.</td>
</tr>
<tr>
<td><code>places</code></td>
<td>Optional number of characters to use in the output. If <code>places</code> is omitted, the function returns the number of spaces required to display the result. The <code>places</code> argument can be used to 0-pad a positive number. If the number is negative, <code>places</code> is ignored and 10 characters are returned.</td>
</tr>
</tbody>
</table>

Remarks
Note that the result of this function is a text string.

Examples
This function returns 1000:
\[
\text{OCT2BIN}(10)
\]

This function returns 11111111:
\[
\text{OCT2BIN}(777)
\]

See Also
BIN2DEC, BIN2HEX, BIN2OCT, DEC2BIN, DEC2HEX, DEC2OCT, HEX2BIN, HEX2DEC, HEX2OCT, OCT2DEC, OCT2HEX
**OCT2DEC**

**Description**  
Converts an octal number (base 8) to a decimal number (base 10).

**Syntax**  
OCT2DEC (integer)

**Remarks**  
Note that the result of this function is a number.

**Examples**  
This function returns 8:

```
OCT2DEC(10)
```

This function returns 536870911:

```
OCT2DEC(3777777777)
```

**See Also**  
BIN2DEC, BIN2HEX, BIN2OCT, DEC2BIN, DEC2HEX, DEC2OCT, HEX2BIN, HEX2DEC, HEX2OCT, OCT2BIN, OCT2HEX

---

**OCT2HEX**

**Description**  
Converts an octal number (base 8) to a hexadecimal number (base 16).

**Syntax**  
OCT2HEX (integer [, places])

**Remarks**  
The result of this function is a text string.

**Examples**  
This function returns 8:

```
OCT2HEX(10)
```
This function returns 1FFFFFFF:

\[ \text{OCT2HEX}(3777777777) \]

**See Also**
BIN2DEC, BIN2HEX, BIN2OCT, DEC2BIN, DEC2HEX, DEC2OCT, HEX2BIN, HEX2DEC, HEX2OCT, OCT2BIN, OCT2DEC

---

## ODD

**Description**
Rounds the specified number up to the nearest odd integer.

**Syntax**

\[
\text{ODD} ( \text{number})
\]

**Examples**
This function returns 5:
\[ \text{ODD}(3.5) \]
This function returns 7:
\[ \text{ODD}(6) \]

**See Also**
CEILING, EVEN, FLOOR, INT, ROUND, TRUNC

---

## ODDFPRICE

**Description**
Computes the price of a security purchased during a first coupon period that is shorter or longer than the other coupon periods. In addition to the dates, you must specify the rate, redemption value, and yield expected.

**Syntax**

\[
\text{ODDFPRICE} ( \text{settlement, maturity, issue, first_coup, rate, yield, redemption, frequency [, calendar_type]})
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>settlement</td>
<td>The date when the security is traded to the buyer. It must be before first_coup. Decimal values will be truncated to integers.</td>
</tr>
<tr>
<td>maturity</td>
<td>The date the security expires and the remaining amount is paid to the investor. Decimal values will be truncated to integers.</td>
</tr>
<tr>
<td>issue</td>
<td>The date the security becomes effective.</td>
</tr>
<tr>
<td>first_coup</td>
<td>The date of the first interest payment.</td>
</tr>
</tbody>
</table>
Chapter 3  A-Z Function Reference  195

Remarks
This function is useful in cases where the security’s issuer does not pay interest for the first few months or years after issuing the security.

This function should be used only when the security’s first coupon date is a different length than the other coupon dates AND the security was purchased during this odd coupon period.

If all coupon periods are of equal length, use PRICE. If the last coupon period is shorter or longer than the rest, use ODDLPRICE.

Equation
\[
\text{ODDFPRICE} = \frac{\text{redemption}}{1 + \frac{\text{yield}}{\text{frequency}}} + \sum_{k=1}^{N} \frac{100 \times \frac{\text{rate}}{\text{frequency}}}{1 + \frac{\text{yield}}{\text{frequency}}} \times \left(\frac{(N-1) + NQ + FQC}{(k-1) + NQ + FQC}\right)
\]

...where the codes correspond to values computed using other functions, as shown in the following table.

Note The software divides the odd first period into non-paying coupon periods in order to correctly calculate the price. These non-paying periods are referred to as pseudo-periods. The variable \( i \) represents the number of pseudo-periods. The first pseudo-period may be shorter than the rest of the pseudo-periods.

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>Number of coupons payable between first_coup and settlement. COUPNUM</td>
<td></td>
</tr>
<tr>
<td>NQ</td>
<td>Number of whole coupon periods in the pseudo-period. Those having a settlement date on a pseudo-period date are not considered whole, but the fraction (FQC) is 1.0. COUPDAYSNC</td>
<td></td>
</tr>
<tr>
<td>FQC</td>
<td>Fraction of the coupon period between settlement and next pseudo-period. COUPNUM</td>
<td></td>
</tr>
</tbody>
</table>

Examples
This function returns 108.0131:

\[\text{ODDFPRICE}({"1/15/93", "1/1/98", "1/1/93", "3/1/93", 0.07, 0.06, 100, 2})\]

See Also
ODDFYIELD, ODDLPRICE, ODDLYIELD, PRICE, YIELD
ODDFYIELD

Description
Computes the yield of a security (per $100 face value) purchased during a first coupon period that is shorter or longer than the other coupon periods. In addition to the dates, you must specify the rate, redemption value, and price.

Syntax
ODDFYIELD (settlement, maturity, issue, first_coup, rate, price, redemption, frequency [, calendar_type])

Remarks
This function is useful in cases where the security’s issuer does not pay interest for the first few months or years after issuing the security.

This function should be used only when the security’s first coupon date is a different length than the other coupon dates AND the security was purchased during this odd coupon period.

If all coupon periods are of equal length, use YIELD. If the last coupon period is shorter or longer than the rest, use ODDLPRICE.

Examples
This function returns 0.075014:

ODDFYIELD("1/15/93","1/1/98","1/1/93","3/1/93",0.07,98,100,2)

See Also
ODDFPRICE, ODDLPRICE, ODDLPRICE, PRICE, YIELD
ODDLPRICE

Description
Computes the price of a security purchased during a last coupon period that is shorter or longer than the other coupon periods. In addition to the dates, you must specify the rate, redemption value, and yield expected.

Syntax
ODDLPRICE (settlement, maturity, last_coupon, rate, yield, redemption, frequency [, calendar_type])

Remarks
This function should be used only when the security's last coupon date is a different length than the other coupon dates AND the security was purchased during this odd coupon period.

If all coupon periods are of equal length, use PRICE. If the first coupon period is shorter or longer than the rest, use ODDFPRICE.

Equation
\[
\frac{\text{redemption} + \frac{\text{fracDCi} \times 100 \times \text{rate}}{\text{frequency}}}{\left(1 + \frac{\text{fracDCi} \times \text{yield}}{\text{frequency}}\right) - \left(\frac{\text{fracAi} \times 100 \times \text{rate}}{\text{frequency}}\right)}
\]

...where the codes correspond to values you can compute using the coupon functions, as shown in the following table.
Note The software divides the odd last period into non-paying coupon periods in order to correctly calculate the price. These non-paying periods are referred to as pseudo-periods. The variable \( i \) represents the pseudo-periods.

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>fracDCi</td>
<td>Number of coupon periods from ( \text{last_coupon} ) to ( \text{maturity} ). This number may be greater or less than 1.</td>
<td>COUPNUM</td>
</tr>
<tr>
<td>fracDCi</td>
<td>Number of coupon periods from ( \text{settlement} ) to ( \text{maturity} ). This number may be greater or less than 1.</td>
<td>COUPNUM</td>
</tr>
<tr>
<td>fracAi</td>
<td>Number of coupon periods from ( \text{last_coupon} ) to ( \text{settlement} ). This number may be greater or less than 1.</td>
<td>COUPNUM</td>
</tr>
</tbody>
</table>

Examples
This function returns 1.416517:

ODDLPRICE("4/19/97","11/25/97","4/1/97",0.05,99.8,100,4)

See Also  ODDFPRICE, ODDFYIELD, ODDLYIELD, PRICE, YIELD

ODDLYIELD

Description Computes the yield of a security (per $100 face value) purchased during a last coupon period that is shorter or longer than the other coupon periods. In addition to the dates, you must specify the rate, redemption value, and price.

Syntax  ODDLYIELD (\( \text{settlement}, \text{maturity}, \text{last\_coupon}, \text{rate}, \text{price}, \text{redemption}, \text{frequency} \) [\( \text{calendar\_type} \)])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{settlement} )</td>
<td>The date when the security is traded to the buyer. It must be after ( \text{last_coupon} ). Decimal values will be truncated to integers.</td>
</tr>
<tr>
<td>( \text{maturity} )</td>
<td>The date the security expires and the remaining amount is paid to the investor. Decimal values will be truncated to integers.</td>
</tr>
<tr>
<td>( \text{last_coupon} )</td>
<td>The date of the last interest payment.</td>
</tr>
<tr>
<td>( \text{rate} )</td>
<td>The security’s annual coupon rate.</td>
</tr>
<tr>
<td>( \text{price} )</td>
<td>The price paid by the buyer at ( \text{settlement} ).</td>
</tr>
<tr>
<td>( \text{redemption} )</td>
<td>The security’s redemption value per $100 face value. This is the amount paid at ( \text{maturity} ) that is not part of any final coupon payment.</td>
</tr>
<tr>
<td>( \text{frequency} )</td>
<td>The number of interest payments per year. See “The frequency Argument” on page 18 for more information.</td>
</tr>
<tr>
<td>[( \text{calendar_type} )]</td>
<td>Optional. One of five methods of counting days for computing interest. See “The calendar_type Argument” on page 19 for more information.</td>
</tr>
</tbody>
</table>
Remarks

This function should be used only when the security’s last coupon date is a different length than the other coupon dates AND the security was purchased during this odd coupon period.

If all coupon periods are of equal length, use YIELD. If the first coupon period is shorter or longer than the rest, use ODDFYIELD.

Equation

\[
\left( \frac{\text{redemption} + \frac{\text{fracDCi}}{\text{frequency}} \times \text{rate}}{\text{frequency}} \right) - \left( \frac{\text{value} + \frac{\text{fracAi}}{\text{frequency}} \times \text{rate}}{\text{frequency}} \right)
\]

\[
\times \frac{\text{frequency}}{\frac{\text{fracDSCi}}{\text{frequency}}}
\]

...where the codes correspond to values you can compute using the coupon functions, as shown in the following table.

**Note** The software divides the odd last period into non-paying coupon periods in order to correctly calculate the price. These non-paying periods are referred to as pseudo-periods. The variable \( i \) represents the pseudo-periods.

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>fracDCi</td>
<td>Number of coupon periods from last_coupon to maturity. This number may be greater or less than 1.</td>
<td>COUPNUM</td>
</tr>
<tr>
<td>fracAi</td>
<td>Number of coupon periods from last_coupon to settlement. This number may be greater or less than 1.</td>
<td>COUPNUM</td>
</tr>
<tr>
<td>fracDSCi</td>
<td>Number of coupon periods from settlement to maturity. This number may be greater or less than 1.</td>
<td>COUPNUM</td>
</tr>
</tbody>
</table>

Examples

This function returns 0.079064:

\[
\text{ODDLYIELD}("6/1/96","1/1/98","1/1/96",0.07,98.4,100,2)
\]

See Also

ODDFPRICE, ODDFYIELD, ODDLPRICE, PRICE, YIELD

**OFFSET**

Description

Returns the contents of a range that is offset from a starting point in the spreadsheet.

Syntax

OFFSET ( reference, rows, columns [, height] [, width])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>reference</td>
<td>A reference to a cell from which the offset reference is based. If you specify a range reference, #VALUE! is returned.</td>
</tr>
</tbody>
</table>
Remarks
OFFSET does not change the current selection in the worksheet. Because it returns a reference, OFFSET can be used in any function that requires or uses a cell or range reference as an argument.

Examples
This function returns the contents of cell D4:

OFFSET(B1, 3, 2, 1, 1)

This function returns the sum of the values in the range E3:F5:

SUM(OFFSET(A1, 2, 4, 3, 2))

---

**OR**

Description
Returns True if at least one of a series of logical arguments is true.

Syntax
OR ( logical_list )

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>logical_list</td>
<td>A list of conditions separated by commas. You can include as many as 30 conditions in the list. The list can contain logical values or a reference to a range containing logical values. Text and empty cells are ignored. If there are no logical values in the list, the error value #VALUE! is returned.</td>
</tr>
</tbody>
</table>

Example
This function returns True because one of the arguments is true:

OR(1 + 1 = 1, 5 + 5 = 10)

See Also
AND, IF, NOT
**PEARSON**

**Description**
Computes the correlation coefficient for two sets of numbers, paired up one-to-one.

The correlation coefficient is a number between -1 and 1 (inclusive) that measures the “relatedness” of the numbers in the samples. A coefficient of 1 indicates a direct relationship in which all points are linearly related on a line with positive slope. A coefficient of -1 indicates an inverse relationship in which a large value in the first argument pairs with a small value in the second argument. A coefficient of 0 indicates no relationship between the pairs of values, or complete randomness.

**Note** The PEARSON function is exactly the same as the CORREL function. We provide both in order to be compatible with all the Microsoft Excel functions.

**Syntax**

```
PEARSON (array1, array2)
```

**Argument**

- `array1` and `array2`: A range reference or array constant containing numeric values. Array1 and array2 must contain the same potential number of values.

The function will return the error value #DIV/0! if `array1` or `array2` contains non-numeric data (text, logical values, or blank cells).

**Remarks**

This function pairs up the numbers in the two ranges by moving left-to-right through each sequential row.

**Equation**

\[
 r = \frac{\sum_{i} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i} (x_i - \bar{x})^2 \sum_{i} (y_i - \bar{y})^2}}
\]

where \( \bar{x} \) is the mean of the \( x_i \)’s and \( \bar{y} \) is the mean of the \( y_i \)’s.

**Examples**

The following examples use this worksheet.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>11</td>
</tr>
</tbody>
</table>

This function returns 1:

```
PEARSON(A1:A3, B1:B3)
```
This function returns 0.991117:

```
PEARSON(A1:B2,C1:C4)
```

**See Also**

CORREL, COVAR

---

**PERCENTILE**

**Description**

Computes the value corresponding to the specified percentile of a specified range of numbers.

**Syntax**

```
PERCENTILE (array, percentile)
```

**Argument**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>array</code></td>
<td>A list of numbers, in the form of a range reference or array constant. Text and logical values are ignored.</td>
</tr>
<tr>
<td><code>percentile</code></td>
<td>The percentile for which you want to find the numeric value. It must be a number between 0 and 1. Text in the argument list that converts to a number will be evaluated as a number; otherwise the function will return the #VALUE! error. Text referenced in cells will return the #VALUE! error. Logical values are evaluated as 1 if TRUE and 0 if FALSE.</td>
</tr>
</tbody>
</table>

**Remarks**

To determine the value, the function sorts `array` in ascending order. Then the function assigns a percentile to each number in `array`: the lowest number is the 0th percentile, the highest number is the 100th percentile, and the middle number is 50th percentile.

If `percentile` falls on a value in `array`, the function returns that value.

If `percentile` does not fall on a value in `array`, the function calculates the value that would fall at `percentile`, given the range values above and below that percentile.

This function is the inverse of the PERCENTRANK function.

**Examples**

This function returns 2:

```
PERCENTILE((1,2,3,4,5), .25)
```

This function returns 2.32:

```
PERCENTILE((1,2,3,4,5), .33)
```

**See Also**

PERCENTRANK, QUARTILE, RANK
PERCENTRANK

Description
Computes the percent rank of a specified value in a specified array of numbers.

Syntax
RANK (array, number [ , digits ])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>A list of at least 2 numbers, in the form of a range reference or array constant. Text and logical values are ignored.</td>
</tr>
<tr>
<td>number</td>
<td>The number you want ranked. It must be within the range of values in array. Text in the argument list that converts to a number will be evaluated as a number; otherwise the function will return the #VALUE! error. Text referenced in cells will return the #VALUE! error. Logical values are evaluated as 1 if TRUE and 0 if FALSE.</td>
</tr>
<tr>
<td>[digits]</td>
<td>Optional. The number of decimal digits you want the answer to be rounded to. It must be a numeric value between 1 and 308. If this argument is omitted, 3 is used.</td>
</tr>
</tbody>
</table>

Remarks
To determine percent rank, the function sorts array in ascending order. Then the function assigns a percent rank to each number in array: the lowest number is 0% rank, the highest number is 100% rank, and the middle number is 50% rank.

If number is in array, the function returns the percent rank assigned to number.

If number is not in array, the function calculates the percent rank that falls proportionately between the percent ranks above and below number.

This function is the inverse of the PERCENTILE function.

Examples
This function returns .5:
PERCENTRANK ({1, 2, 3, 4, 5}, 3)

This function returns .375:
PERCENTRANK ({1, 2, 4, 5, 6}, 3)

See Also
PERCENTILE, QUARTILE, RANK

PERMUT

Description
Computes the number of combinations possible by taking k items at a time from a pool of n, where order in the sample taken is important.
For example, in a box of four different items, the number of combinations possible taking out 2 items at a time is 12; that is, the twelve combinations of items 1-2, 1-3, 1-4, 2-1, 2-3, 2-4, 3-1, 3-2, 3-4, 4-1, 4-2, and 4-3.

Syntax

PERMUT (number, chosen)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>A positive integer representing the total number of items in the pool of items. Decimal numbers are truncated to integers.</td>
</tr>
<tr>
<td>chosen</td>
<td>A positive integer representing the number of items taken from the pool at a time. It must be less than or equal to number. Decimal values are truncated to integers.</td>
</tr>
</tbody>
</table>

Remarks

This function is similar to the COMBIN function, except that PERMUT requires the samples to be ordered, while COMBIN takes samples in any order.

Equation

\[ n(n - 1)(n - 2)\ldots(n - k + 1) = \frac{n!}{(n-k)!} \]

where \( n \) is the total number of items and \( k \) is the number of items taken at a time.

Examples

This function returns 12:

PERMUT(4, 2)

This function returns 30240:

PERMUT(10, 5)

See Also

COMBIN, GAMMALN

PI

Description

Returns the value of pi, which is approximately 3.14159265358979 when calculated to 15 significant digits.

Syntax

PI ( )

Remarks

Although PI does not use arguments, you must supply the empty parentheses to correctly reference the function.

See Also

COS, SIN, TAN
**PMT**

**Description**
Returns the periodic payment of an annuity, based on regular payments and a fixed periodic interest rate.

**Syntax**
PMT( interest, nper, pv [,fv] [,type])

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>interest</td>
<td>The fixed periodic interest rate.</td>
</tr>
<tr>
<td>nper</td>
<td>The number of periods in the annuity.</td>
</tr>
<tr>
<td>pv</td>
<td>The present value, or the amount the annuity is currently worth.</td>
</tr>
<tr>
<td>[fv]</td>
<td>The future value, or the amount the annuity will be worth. If this argument is omitted, 0 is used.</td>
</tr>
<tr>
<td>[type]</td>
<td>Indicates when payments are due. Use 0 if payments are due at the end of the period or 1 if payments are due at the beginning of the period. If this argument is omitted, 0 is used.</td>
</tr>
</tbody>
</table>

**Remarks**
PMT returns only the principal and interest payment, it does not include taxes or other fees.

The units used for *interest* must match those used for *nper*. For example, if the annuity has an 8 percent annual interest rate over a period of 5 years, specify 8 percent/12 for *interest* and 5*12 for *nper*.

Cash paid out, such as a payment, is shown as a negative number. Cash received, such as a dividend check, is shown as a positive number.

**Examples**
This function returns −439.43:
PMT(8%/12, 48, 18000)

This function returns −436.52:
PMT(8%/12, 48, 18000, 0, 1)

**See Also**
FV, CUMIPMT, CUMPRINC, IPMT, NPER, PPMT, PV, RATE

---

**POISSON**

**Description**
Computes the Poisson distribution, which is usually used to determine the probability of a certain number of repeated events taking place over time.

For example, you could use POISSON to calculate the probability that more than the average number of calls will pass through a telephone switching station over an hour’s period.
### POISSON (x, mean, cumulative)

**Syntax**

```
POISSON(x, mean, cumulative)
```

**Argument**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>A positive integer indicating the number of events you want to test the probability of happening over a given period. Decimal values will be rounded down to the nearest integer. Logical values are interpreted as 1 for TRUE and 0 for FALSE.</td>
</tr>
<tr>
<td>mean</td>
<td>A positive number indicating the average or expected number of events for the given test period. Logical values are interpreted as 1 for TRUE and 0 for FALSE.</td>
</tr>
<tr>
<td>cumulative</td>
<td>A logical value. True or 1 indicates you want to find the probability of at least x events happening. False or 0 indicates you want to find the probability of exactly x events happening.</td>
</tr>
</tbody>
</table>

**Equations**

When cumulative is FALSE:

\[ f(x) = \frac{\mu^x}{x!}e^{-\mu} \]

When cumulative is TRUE:

\[ \sum_{j=0}^{x} e^{-\mu} \frac{\mu^j}{j!} \]

**Examples**

This function returns .57, or a 57% probability:

```
POISSON(15,15,1)
```

**See Also**

BETADIST, BINOMDIST, CHIDIST, COMBIN, CRITBINOM, FDIST, GAMMADIST, NORMDIST

---

### POWER

**Description**

Raises the specified base number to the specified power.

**Syntax**

```
POWER(base, exponent)
```

**Argument**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>base</td>
<td>Any number. Nonnumeric values will return the #VALUE! error.</td>
</tr>
<tr>
<td>exponent</td>
<td>Any number. Nonnumeric values will return the #VALUE! error. If base is negative, exponent must be an integer, except in the case when exponent is larger than -1, smaller than 1, and corresponds to an odd root (for example, (\frac{1}{3}) and (\frac{1}{5})). If base is 0, exponent must be larger than 0. If exponent is 0, base may not be 0.</td>
</tr>
</tbody>
</table>
Examples

This function returns 9:

\[ \text{POWER}(3, 2) \]

This function returns 0.001:

\[ \text{POWER}(10, -3) \]

PPMT

Description

Returns the principle paid on an annuity for a given period.

Syntax

\[ \text{PPMT} \left( \text{interest}, \text{per}, \text{nper}, \text{pv}, \left[ \text{fv} \right], \left[ \text{type} \right] \right) \]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>interest</td>
<td>The fixed periodic interest rate.</td>
</tr>
<tr>
<td>per</td>
<td>The period for which to return the principal.</td>
</tr>
<tr>
<td>nper</td>
<td>The number of periods in the annuity.</td>
</tr>
<tr>
<td>pv</td>
<td>The present value, or the amount the annuity is currently worth.</td>
</tr>
<tr>
<td>[fv]</td>
<td>The future value, or the amount the annuity will be worth. If this argument is omitted, 0 is used.</td>
</tr>
<tr>
<td>[type]</td>
<td>Indicates when payments are due. Use 0 if payments are due at the end of the period or 1 if payments are due at the beginning of the period. If this argument is omitted, 0 is used.</td>
</tr>
</tbody>
</table>

Remarks

The units used for \( \text{interest} \) must match those used for \( \text{nper} \). For example, if the annuity has an 8 percent annual interest rate over a period of 5 years, specify 8 percent/12 for \( \text{interest} \) and 5*12 for \( \text{nper} \).

Examples

This function returns \(-321.56\):

\[ \text{PPMT}(8\%/12, 2, 48, 18000) \]

This function returns \(-319.43\):

\[ \text{PPMT}(8\%/12, 2, 48, 18000, 0, 1) \]

See Also

\( \text{FV}, \text{CUMIPMT}, \text{CUMPRINC}, \text{IPMT}, \text{NPER}, \text{PMT}, \text{PV}, \text{RATE} \)
PRICE

Description

Computes the price of a security with specified rate, redemption value, and yield.

Syntax

PRICE (settlement, maturity, rate, yield, redemption, frequency [, calendar_type])

Remarks

Use this function when all coupon periods between settlement and maturity are of equal length. If the first coupon period is shorter or longer than the rest, use ODDFPRICE. If the last coupon period is shorter or longer than the rest, use ODDLPRICE.

Equation

\[
\text{redemption} \left(1 + \frac{\text{yield}}{\text{frequency}}\right)^{\text{NCP} - 1 + \frac{\text{DCS}}{\text{DIC}}} + \sum_{\text{DCS}=1}^{\text{NCP}} \left(100 \times \frac{\text{rate}}{\text{frequency}}\right)^{\text{DCS} - 1 + \frac{\text{DCS}}{\text{DIC}}} - \left(100 \times \frac{\text{rate}}{\text{frequency}}\right)^{\frac{\text{DCS}}{\text{DIC}}}
\]

where the three-letter codes correspond to values you can compute using the coupon functions, as shown in the following table.

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCS</td>
<td>Number of days from beginning of the coupon period to settlement. COUPDAYBS</td>
<td></td>
</tr>
<tr>
<td>DIC</td>
<td>Number of days in coupon period in which settlement falls. COUPDAYS</td>
<td></td>
</tr>
</tbody>
</table>
PRICEDISC

Description
Computes the price of a discounted security per $100 face value, given a specified discount rate and redemption value.

Syntax
PRICEDISC (settlement, maturity, discount, redemption[, calendar_type])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>settlement</td>
<td>The date when the security is traded to the buyer. Decimal values will be truncated to integers.</td>
</tr>
<tr>
<td>maturity</td>
<td>The date the security expires and the remaining amount is paid to the investor. It must be later than settlement. Decimal values will be truncated to integers.</td>
</tr>
<tr>
<td>discount</td>
<td>The security’s annual discount rate.</td>
</tr>
<tr>
<td>redemption</td>
<td>The security’s redemption value per $100 face value. This is the amount paid at maturity.</td>
</tr>
<tr>
<td>[calendar_type]</td>
<td>Optional. One of five methods of counting days for computing interest. See “The calendar_type Argument” on page 19 for more information.</td>
</tr>
</tbody>
</table>

Equation
redemption - discount \times redemption \times \text{YEARFRAC}(settlement, maturity, calendar_type)

Examples
This function returns 56.01:
PRICEDISC("5/7/92", "12/31/99", 0.0575, 100)

See Also
DISC, PRICE, PRICEMAT, YIELD, YIELDDISC, YIELDMAT
PRICEMAT

Description
Computes the price of a security that pays interest only at maturity, given specified
dates, interest rate, and yield. The price computed is per $100 face value.

Syntax
PRICEMAT (settlement, maturity, issue, rate, yield [, calendar_type])

Argument | Description
--- | ---
settlement | The date when the security is traded to the buyer. Decimal values will be truncated to integers.
maturity | The date the security expires and the remaining amount is paid to the investor. It must be the same day or later than settlement. Decimal values will be truncated to integers.
issue | The date the security was originally issued and began earning interest. Decimal values will be truncated to integers.
rate | The security's annual interest rate at date of issue.
yield | The annual income produced by the security. This can be considered the effective interest rate for the buyer.
[calendar_type] | Optional. One of five methods of counting days for computing interest. See “The calendar_type Argument” on page 19 for more information.

Equation
\[
100 + \frac{\left(\text{DIM} \times \text{rate} \times 100\right)}{1 + \left(\text{DSM} \times \text{yield} \times 100\right)} - \frac{\left(\text{DIS} \times \text{rate} \times 100\right)}{}
\]

...where the three-letter codes correspond to values you can compute using the
YEARFRAC function, as shown in the following table.

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
<th>Function and arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIM/DIY</td>
<td>Days from issue to maturity.</td>
<td>YEARFRAC (issue, maturity, calendar_type)</td>
</tr>
<tr>
<td>DIS/DIY</td>
<td>Days from issue to settlement.</td>
<td>YEARFRAC (issue, settlement, calendar_type)</td>
</tr>
<tr>
<td>DSM/DIY</td>
<td>Days from settlement to maturity.</td>
<td>YEARFRAC (settlement, maturity, calendar_type)</td>
</tr>
</tbody>
</table>

Examples
This function returns 82.11:

PRICEMAT("4/8/93","2/14/99", "2/14/89", 0.045, 0.075)

See Also
PRICE, PRICEDISC, YIELD, YIELDDISC, YIELDMAT
PROB

Description
Given a set of numbers and a probability associated with each number, computes the probability corresponding to a specified number. Also computes all the probabilities corresponding to a specified range of numbers.

Syntax
PROB (arrayX, probabilities, test [, upper])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arrayX</td>
<td>A range reference or array constant containing numeric values. There must be the same number of values in probabilities as in arrayX. The function will return the error value #NUM! if arrayX contains non-numeric data (text, logical values, or blank cells).</td>
</tr>
<tr>
<td>probabilities</td>
<td>A range reference or array constant containing the probabilities associated with the numbers in arrayX. Each probability must be a number between 0 and 1. All values in probabilities must add up to 1. There must be the same number of values in probabilities as in arrayX. The function will return the error value #NUM! if probabilities contains non-numeric data (text, logical values, or blank cells).</td>
</tr>
<tr>
<td>test</td>
<td>The value from arrayX whose probability you want to find. If test is not found in arrayX and no value is entered for upper, the function returns 0.</td>
</tr>
<tr>
<td>[upper]</td>
<td>Optional. Specifies the upper limit of a range of numbers from arrayX whose cumulative probabilities you want to know. Test is the lower limit, upper the upper limit, inclusive. PROB will return the sum of all values in probabilities that correspond to the numbers in arrayX that fall between or include test and upper.</td>
</tr>
</tbody>
</table>

Remarks
When arrayX and probabilities are range references, the function pairs up the numbers by moving left-to-right through each sequential row.

Examples
This function returns 0.05:

PROB ((1,2,3,4,5), {0.05, 0.1, 0.2, 0.3, 0.35}, 1)

This function returns 0.3:

PROB ((1,2,3,4,5), {0.05, 0.1, 0.2, 0.3, 0.35}, 2, 3)

This function returns 0.1:

PROB ((1,2,3,4,5), {0.05, 0.1, 0.2, 0.3, 0.35}, 1.5, 2.5)

See Also
COVAR, PEARSON
PRODUCT

Description
Multiplies a list of numbers and returns the result.

Syntax
PRODUCT ( number_list )

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number_list</td>
<td>A list of as many as 30 numbers, separated by commas. The list can contain numbers, logical values, text representations of numbers, or a reference to a range containing those values. Error values or text that cannot be translated into numbers return errors. If a range reference is included in the list, text, logical expressions, and empty cells in the range are ignored. All numeric values, including 0, are used in the calculation.</td>
</tr>
</tbody>
</table>

Example
This function returns 24:
PRODUCT(1, 2, 3, 4)

See Also
FACT, FACTDOUBLE, PRODUCT, SUM

PROPER

Description
Returns the specified string in proper-case format.

Syntax
PROPER ( text )

Argument Description

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>Any string.</td>
</tr>
</tbody>
</table>

Remarks
In proper-case format, the first alphabetic character in a word is capitalized. If an alphabetic character follows a number, punctuation mark, or space, it is capitalized. All other alphabetic characters are lowercase. Numbers are not changed by PROPER.

Examples
This function returns 3Rd Quarter:
PROPER("3rd Quarter")

This function returns John Doe:
PROPER("JOHN DOE")

See Also
LOWER, UPPER
PV

Description
Returns the present value of an annuity, considering a series of constant payments made over a regular payment period.

Syntax
```
PV( interest, nper, pmt [, fv] [, type])
```

Remarks
The units used for `interest` must match those used for `nper`. For example, if the annuity has an 8 percent annual interest rate over a period of 5 years, specify 8 percent/12 for `interest` and 5*12 for `nper`.

Cash paid out, such as a payment, is shown as a negative number. Cash received, such as a dividend check, is shown as a positive number.

Examples
This function returns \(-17999.89\):

```
PV(8%/12, 48, 439.43)
```

This function returns \(17999.89\):

```
Pv(8%/12, 48, -439.43)
```

See Also
FV, CUMIPMT, CUMPRINC, IPMT, NPER, PMT, PPMT, RATE

QUARTILE

Description
Computes the value corresponding to the specified quartile in a specified array of numbers.

Syntax
```
QUARTILE(array, quartilerank)
```

Remarks
A list of numbers, in the form of a range reference or array constant. Text and logical values are ignored.
The function sorts the array in ascending order. Then the function divides the array into four parts or quartiles. The lowest value in the array is assigned the 0th quartile and the highest value is the 4th quartile. If the array contains an odd number of values, the middle value is assigned the 2nd quartile; otherwise, a value halfway between the two middle values is calculated and assigned the 2nd quartile. Values corresponding to the 1st and 3rd quartiles are either assigned or calculated in the same manner.

The function returns the value it assigned to or calculated for the `quartilerank` value you chose.

### Examples

This function returns 57:

```
QUARTILE({12,13,57,89,90}, 2)
```

This function returns 5:

```
QUARTILE({1,1,9,9}, 2)
```

### See Also

PERCENTILE, PERCENTRANK

---

## QUOTIENT

### Description

Computes the quotient of two numbers, truncated to an integer (toward 0).

### Syntax

`QUOTIENT ( dividend, divisor )`

### Argument Description

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>dividend</code></td>
<td>The number to be divided.</td>
</tr>
<tr>
<td><code>divisor</code></td>
<td>The number by which to divide. Any number other than zero.</td>
</tr>
</tbody>
</table>

### Examples

This function returns 1:

```
QUOTIENT(9,7)
```

### See Also

MOD, INT
RADIANS

**Description**

Converts a value in degrees to radians.

**Syntax**

\[ \text{RADIANS} (\text{degrees}) \]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>degrees</td>
<td>A number describing the angle in degrees that you want to convert to radians. Nonnumeric entries cause a #VALUE! error.</td>
</tr>
</tbody>
</table>

**Equation**

\[ \text{degrees} \left( \frac{\pi}{180} \right) \]

**Examples**

This function returns -3.141592654:

\[ \text{RADIANS}(-180) \]

This function returns 0:

\[ \text{RADIANS}(0) \]

**See Also**

PI, DEGREES

RAND

**Description**

Returns a number selected randomly from a uniform distribution greater than or equal to 0 and less than 1 each time the worksheet is recalculated.

**Syntax**

\[ \text{RAND} ( ) \]

**Remarks**

Although RAND does not use arguments, you must supply the empty parentheses to correctly reference the function.

Formula One for Java will regenerate this random number every time the worksheet is recalculated. Each recalculation will generate only a single random number that is used during all of a recalc. Multiple copies of this function will generate different random numbers.

**Example**

This function returns a random number greater than or equal to 0 and less than 10:

\[ \text{RAND()} \times 10 \]

**See Also**

RANDBETWEEN
RANDBETWEEN

Description
Returns an integer selected randomly from between two specified limits, inclusive, each time the sheet is recalculated.

Syntax
RANDBETWEEN (bottom, top)

Remarks
Formula One for Java will regenerate this random number every time the worksheet is recalculated.

Example
This function returns a random integer greater than or equal to 14 and less than or equal to 27:
RANDBETWEEN(14,27)

See Also
RAND

RANK

Description
Computes the rank of a specified value in a specified array of numbers. By default, the array is sorted in descending order, but you may choose to sort it in ascending order.

Syntax
RANK (number, array [,order])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>The member of array that you want ranked. If the function finds more than one of number in array, it will return the rank of the first instance it finds. If the function doesn't find number in array, it will return the #N/A! error. Text in the argument list that converts to a number will be evaluated as a number; otherwise the function will return the #VALUE! error. Text referenced in cells will return the #VALUE! error. Logical values are evaluated as 1 if TRUE and 0 if FALSE.</td>
</tr>
<tr>
<td>array</td>
<td>A list of from 2 to 8191 numbers, in the form of a range reference or array constant. Text and logical values are ignored.</td>
</tr>
</tbody>
</table>
Examples

This function returns 5:
\[
\text{RANK (3, \{5, 1, 8, 3, 0, 4, 8, 3, 2\})}
\]

This function returns 4:
\[
\text{RANK (3, \{5, 1, 8, 3, 0, 4, 8, 3, 2\},1)}
\]

See Also

PERCENTILE, PERCENTRANK, QUARTILE

RATE

Description

Returns the interest rate per period of an annuity, given a series of constant cash payments made over a regular payment period.

Syntax

\[
\text{RATE ( nper, pmt, pv \[, \text{fv} \] \[, \text{type} \] \[, \text{guess} \])}
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nper</td>
<td>The number of periods in the annuity.</td>
</tr>
<tr>
<td>pmt</td>
<td>The fixed payment made each period. Generally, ( pmt ) includes only principle and interest, not taxes or other fees.</td>
</tr>
<tr>
<td>pv</td>
<td>The present value of the annuity.</td>
</tr>
<tr>
<td>[fv]</td>
<td>Optional. The future value, or the amount the annuity will be worth. If this argument is omitted, 0 is used.</td>
</tr>
<tr>
<td>[type]</td>
<td>Optional. Indicates when payments are due. Use 0 if payments are due at the end of the period or 1 if payments are due at the beginning of the period. If this argument is omitted, 0 is used.</td>
</tr>
<tr>
<td>[guess]</td>
<td>Optional. Your estimate of the interest rate. If this argument is omitted, 0.1 (10 percent) is used.</td>
</tr>
</tbody>
</table>

Remarks

RATE is calculated iteratively, cycling through the calculation until the result is accurate to .00001 percent. If the result cannot be found after 20 iterations, #NUM! is returned. When this occurs, supply a different value for \( \text{guess} \).

Example

The following example returns the monthly interest rate of .0067; the annual interest rate (.0067 multiplied by 12) is 8 percent:

\[
\text{RATE(48, -439.43, 18000)}
\]

See Also

FV, CUMIPMT, CUMPRINC, IPMT, NPER, PMT, PPMT, PV
RECEIVED

Description
Computes the amount received at maturity for a fully invested security. Investment amount and discount rate must be specified.

Syntax
RECEIVED ( settlement, maturity, investment, discount [ , calendar_type])

Argument | Description
---|---
settlement | The date when the security is traded to the buyer. Decimal values will be truncated to integers.
maturity | The date the security expires and the remaining amount is paid to the investor. It must be later than settlement. Decimal values will be truncated to integers.
investment | The amount invested in the security.
discount | The security’s discount rate. Note that this is not the security’s yield.
[calendar_type] | Optional. One of five methods of counting days for computing interest. See “The calendar_type Argument” on page 19 for more information.

Equation
\[
\text{investment} \times \text{YEARFRAC}(\text{settlement}, \text{maturity}, \text{calendar_type}) = 1 - \text{discount} \\
\]

Examples
This function returns 63.745:

```
RECEIVED("1/1/94","10/1/97",50,0.0575)
```

See Also
INTRATE, PRICEDISC, YIELD

REGISTER.ID

Description
Returns the #N/A error message.

Syntax
REGISTER.ID (arguments)

Argument | Description
---|---
arguments | 2 to 3 arguments of varying types

Remarks
This function was included only for compatibility with Microsoft Excel. In Excel, REGISTER.ID extracts the identification number from a dynamic linked library or code resource, neither of which is supported in Java.

This function is intended only for users or developers who want to import Excel worksheets that contain Excel’s REGISTER.ID function. To make those worksheets work properly, developers should convert these functions to add-in functions, which they can write in Java and set up to be automatically loaded.
For more information, see the chapter on creating add-in functions in the *Formula One for Java Technical Guide*.

**REPLACE**

**Description**
Replaces part of a text string with another text string.

**Syntax**
REPLACE (orig_text, start_position, num_chars, repl_text)

**Examples**
This function returns “For the year: 1994”:
REPLACE("For the year: 1993", 18, 1, "4")

**See Also**
MID, SEARCH, TRIM

**REPT**

**Description**
Repeats a text string the specified number of times.

**Syntax**
REPT (text, number)

**Remarks**
The result of REPT cannot exceed 255 characters.

**Example**
This function returns error-error-error-:
REPT("error-", 3)
### RIGHT

**Description**
Returns the rightmost characters from the given text string.

**Syntax**
```plaintext
RIGHT ( text [, num_chars])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>Any text string.</td>
</tr>
<tr>
<td>[num_chars]</td>
<td>Optional. The number of characters to return. The value must be</td>
</tr>
<tr>
<td></td>
<td>greater than or equal to zero. If num_chars is greater than the number of</td>
</tr>
<tr>
<td></td>
<td>characters in text, the entire string is returned. If this argument is</td>
</tr>
<tr>
<td></td>
<td>omitted, 1 is used.</td>
</tr>
</tbody>
</table>

**Examples**
This function returns r:
```
RIGHT("2nd Quarter")
```
This function returns Quarter:
```
RIGHT("2nd Quarter", 7)
```

**See Also**
LEFT, MID

### ROMAN

**Description**
Converts an arabic number to a Roman numeral.

**Syntax**
```plaintext
ROMAN ( number [, type])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>A number between 0 and 3999. Any nonnumeric value will return the #VALUE!</td>
</tr>
<tr>
<td></td>
<td>error. Decimal values are truncated to integers.</td>
</tr>
<tr>
<td>[type]</td>
<td>Optional. One of five types or styles of roman numerals. The options are</td>
</tr>
<tr>
<td></td>
<td>0, 1, 2, 3, or 4. If this argument is omitted, 0 is used.</td>
</tr>
<tr>
<td></td>
<td>Type 0 is the classical type taught in schools. The differences between the</td>
</tr>
<tr>
<td></td>
<td>styles have to do with whether the function allows the letters I (1), V (5),</td>
</tr>
<tr>
<td></td>
<td>and X (10) to appear before the letters L (50), C (100), D (500), and M</td>
</tr>
<tr>
<td></td>
<td>(1000). See the examples below. The logical value TRUE is equivalent to type</td>
</tr>
<tr>
<td></td>
<td>0; the logical value FALSE is equivalent to type 4.</td>
</tr>
</tbody>
</table>
**Examples**

This function returns CDXCIX:

```sql
ROMAN(499)
```

This function returns LDVLIV:

```sql
ROMAN(499, 1)
```

This function returns XDIX:

```sql
ROMAN(499, 2)
```

This function returns VDIV:

```sql
ROMAN(499, 3)
```

This function returns ID:

```sql
ROMAN(499, 4)
```

**See Also**

BIN2DEC, BIN2HEX, BIN2OCT, OCT2BIN, OCT2DEC, OCT2HEX

---

**ROUND**

**Description**

Rounds the given number to the supplied number of decimal places.

**Syntax**

```
ROUND ( number, precision )
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>number</code></td>
<td>Any value.</td>
</tr>
<tr>
<td><code>precision</code></td>
<td>The number of decimal places to which <code>number</code> is rounded.</td>
</tr>
<tr>
<td></td>
<td>When a negative precision is used, the digits to the right of the decimal point are dropped and the absolute number of significant digits specified by <code>precision</code> are replaced with zeros.</td>
</tr>
<tr>
<td></td>
<td>If <code>precision</code> is 0, <code>number</code> is rounded to the nearest integer.</td>
</tr>
</tbody>
</table>

**Example**

This function returns 123.46:

```sql
ROUND(123.456, 2)
```

This function returns 9900:

```sql
ROUND(9899.435, -2)
```

**See Also**

CEILING, FLOOR, INT, MOD, ROUNDDOWN, ROUNDUP, TRUNC
ROUNDUP

Description
Rounds the given number up to the supplied number of decimal places.

Syntax
ROUNDUP ( number, number_of_digits )

Example
This function returns 77:
ROUNDUP(76.9, 0)

See Also
CEILING, FLOOR, INT, MOD, ROUND, ROUNDUP, TRUNC

ROUNDDOWN

Description
Rounds a number down.

Syntax
ROUNDDOWN ( number, number_of_digits )

Example
This function returns 31.141:
ROUNDDOWN(3.14159, 3)

This function returns 31.400:
ROUNDDOWN(31415.92654, -2)

See Also
CEILING, FLOOR, INT, MOD, ROUND, ROUNDUP, TRUNC
This function returns 31500:
ROUNDUP(31415.92654, -2)

See Also  
CEILING, FLOOR, INT, MOD, ROUND, ROUNDDOWN, TRUNC

---

**ROW**

**Description**  
Returns the row number of the supplied reference.

**Syntax**  
ROW (reference)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>reference</td>
<td>A cell or range reference. Omitting this argument returns the row number of the cell in which ROW is entered.</td>
</tr>
</tbody>
</table>

**Examples**  
This function returns 3:
ROW(B3)

See Also  
COLUMN, ROWS

---

**ROWS**

**Description**  
Returns the number of rows in a range reference.

**Syntax**  
ROWS (range)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>range</td>
<td>A reference to a range of cells.</td>
</tr>
</tbody>
</table>

**Examples**  
This function returns 5:
ROWS(A1:D5)

This function returns 6:
ROWS(C30:F35)

See Also  
COLUMNS, ROW
RSQ

Description
Computes the square of the correlation coefficient for two sets of numbers, paired up one-to-one.

The correlation coefficient is a number between -1 and 1 (inclusive) that measures the “relatedness” of the numbers in the samples. A coefficient of 1 indicates a direct relationship in which all points are linearly related on a line with positive slope. A coefficient of -1 indicates an inverse relationship in which a large value in the first argument pairs with a small value in the second argument. A coefficient of 0 indicates no relationship between the pairs of values, or complete randomness.

Note The RSQ function simply squares the value found by the PEARSON function. See “PEARSON” on page 201 for the equation and further examples.

Syntax
RSQ (array1, array2)

Remarks
This function pairs up the numbers in the two ranges by moving left-to-right through each sequential row.

Examples
This function returns 0.75:
RSQ({1,2,3,4,5}, {9,9,10,10,10})

See Also
CORREL, COVAR, PEARSON

SEARCH

Description
Locates the position of the first character of a specified text string within another text string.

Syntax
SEARCH ( search_text, text [, start_position])
### SEARCH

**Remarks**
Text is searched from left to right, starting at the position specified. The search is not case-sensitive. If `text` does not contain the search string, #VALUE! is returned.

**Examples**
- This function returns 6:
  
  `SEARCH("?5", "Bin b45")`

- This function returns 5:
  
  `SEARCH("b", "Bin b45", 4)`

**See Also**
FIND, MID, REPLACE, SUBSTITUTE

### SECOND

**Description**
Returns the second that corresponds to the supplied date.

**Syntax**
SECOND ( `serial_number` )

**Examples**
- This function returns 58:
  
  `SECOND(.259)`

- This function returns 46:
  
  `SECOND(34657.904)`

**See Also**
DAY, HOUR, MINUTE, MONTH, NOW, WEEKDAY, YEAR
SERIESSUM

Description

Returns the sum of a power series.

Syntax

SERIESSUM (x, n, m, a)

Argument Description

x  The value of the independent variable in the series.

n  The initial power of x in the series.

m  The increment of the power of x in the series.

a  A list of coefficients for the series. The list can be a range reference or array constant. The number of coefficients determines the number of terms in the series.

Equation

\[ a_1 \cdot x^n + a_2 \cdot x^{(n+m)} + a_3 \cdot x^{(n+2m)} + \ldots + a_i \cdot x^{(n+(i-1)m)} \]

...where \( i \) is the number of coefficients in \( a \).

Examples

This function returns 11368:

SERIESSUM(2,2,3,{2,3,4,5})

SIGN

Description

Determines the sign of the specified number.

Syntax

SIGN (number)

Argument Description

number  Any number.

Remarks

SIGN returns 1 if the specified number is positive, –1 if it is negative, and 0 if it is 0.

Examples

This function returns –1:

SIGN(-123)

This function returns 1:

SIGN(123)

See Also

ABS
**SIN**

**Description**
Returns the sine of the supplied angle.

**Syntax**
SIN (number)

**Examples**
This function returns .85:
SIN(45)

This function returns .89:
SIN(90)

**See Also**
ASIN, PI

**SINH**

**Description**
Returns the hyperbolic sine of the specified number.

**Syntax**
SINH (number)

**Examples**
This function returns 1.18:
SINH(1)

This function returns 10.02:
SINH(3)

**See Also**
ASINH, PI
SKEW

Description
Computes the skewness of a set of numbers.

Syntax
SKEW (number_list)

Remarks
Skewness, also called the third movement, is a measure of the asymmetry of a distribution around its mean. A large positive skewness indicates that more of the distribution falls closer to the positive numbers on the x axis; a negative skewness indicates more of the distribution is closer to the negative numbers on the x axis.

Equation
\[
\frac{n}{(n-1)(n-2)} \sum \left( \frac{x_j - \bar{x}}{s} \right)^3
\]

Examples
This function returns 0:
SKEW(1,2,3,4,5,6,7,8,9)

This function returns 0.148190368:
SKEW(1,1,1,3,4,5,6,7,8)

This function returns -0.245294714:
SKEW(-1,-1,-1,-4,-5,-6,-7,-8)

See Also
AVERAGE, AVEDEV, COUNT, DEVSQ, KURT

SLN

Description
Returns the depreciation of an asset for a specific period of time using the straight-line balance method.

Syntax
SLN (cost, salvage, life)
Example
This function returns 1285.71:
\[
\text{SLN}(10000, 1000, 7)
\]

See Also
DDB, SYD, VDB

### SLOPE

**Description**
Computes the slope of the least squares linear regression line through the given data.

**Syntax**
\[
\text{SLOPE} \left( \text{arrayY}, \text{arrayX} \right)
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arrayY</td>
<td>A range reference or an array constant containing numeric values. ArrayX</td>
</tr>
<tr>
<td>arrayX</td>
<td>and arrayY must contain the same potential number of values. Text, logical</td>
</tr>
<tr>
<td></td>
<td>values, and empty cells referenced by this function are ignored, along with</td>
</tr>
<tr>
<td></td>
<td>the number they are paired up with. That is, when a number from arrayX is</td>
</tr>
<tr>
<td></td>
<td>paired up with text from arrayY, the entire pair is ignored.</td>
</tr>
</tbody>
</table>

**Remarks**
This function pairs up the numbers in the two ranges by moving left-to-right through each sequential row.

**Equation**
\[
slope b = \frac{n \sum xy - (\sum x)(\sum y)}{n \sum x^2 - (\sum x)^2}
\]

where \( n \) is the number of members in \( \text{arrayY} \) and \( \text{arrayX} \).

**Examples**
This function returns 2.342857143:
\[
\text{SLOPE}((1.5, 8.9, 12.13), (0.1, 2.3, 4.5))
\]

**See Also**
FORECAST, INTERCEPT
SMALL

Description
Computes the xth smallest value in a list. If you set x at 5, SMALL will sort the values in descending order and return the 5th smallest value.

Syntax
SMALL (numbers, x)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>numbers</td>
<td>A range reference or array constant containing numbers. Text and logical values in cell references and arrays are ignored. Empty cells are ignored.</td>
</tr>
<tr>
<td>x</td>
<td>The index of the number to return after sorting the list in ascending order. X must be larger than zero and smaller than the number of values in numbers. The logical value TRUE is evaluated as 1, but FALSE will return the #NUM! error. Text will be evaluated as a number, if possible; otherwise the function will return the #VALUE! error.</td>
</tr>
</tbody>
</table>

Example
This function returns 4:
SMALL({1,2,3,4,5,6,7,8,9,15},4)

See Also
LARGE, MAX, MIN

SQLREQUEST

Description
Returns the #N/A error message.

Syntax
SQLREQUEST (arguments)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arguments</td>
<td>4 to 5 arguments.</td>
</tr>
</tbody>
</table>

Remarks
This function was included only for compatibility with Microsoft Excel. In Excel, SQLREQUEST connects the worksheet to an external data source, functionality which can be handled more efficiently in Formula One for Java using JDBC support.

This function is intended only for users or developers who want to import Excel worksheets that contain Excel’s SQLREQUEST function. To make those worksheets work properly, developers should either convert these functions to add-in functions, which they can write in Java and set up to be automatically loaded, or utilize Formula One for Java’s JDBC capabilities.

For more information on add-in functions and JDBC, see the Formula One for Java Technical Guide.
**SQRT**

**Description**
Returns the square root of the specified number.

**Syntax**

```
SQRT ( number )
```

**Examples**

This function returns 3:

```
SQRT(9)
```

This function returns 1.58:

```
SQRT(2.5)
```

**See Also**
SUMSQ

**SQRTPI**

**Description**
Calculates the square root of the product of the specified number and π.

**Syntax**

```
SQRTPI ( number )
```

**Examples**

This function returns 1.772453851:

```
SQRTPI(1)
```

This function returns 2.506628275:

```
SQRTPI(2)
```

**See Also**
SQRT, PI
STANDARDIZE

Description Computes the proper argument for the function that calculates the standard normal cumulative distribution, given specified values for x, mean, and standard deviation. This function is used to “translate” the three arguments required for the NORMDIST function into a form that can be used in NORMSDIST.

Syntax

STANDARDIZE (x, mean, st_dev)

Example

This function returns 11.25:

STANDARDIZE (95, 50, 4)

See Also

NORMDIST, NORMSDIST

STDEV

Description Returns the standard deviation of a population based on a sample of supplied values. The standard deviation of a population represents an average of deviations from the population mean within a list of values.

Syntax

STDEV (number_list)

Example

This function returns .56:

STDEV(4.0, 3.0, 3.0, 3.5, 2.5, 4.0, 3.5)

See Also

STDEVA, STDEVPA, STDEVPA, VAR, VARA, VARP, VARPA
STDEVA

Description
Returns the standard deviation of a population based on a sample of supplied values. The standard deviation of a population represents an average of deviations from the mean within a list of values.

This function is equivalent to the STDEV function, but its implementation treats text and logical values in cell and range references differently.

Syntax
STDEVA (number_list)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number_list</td>
<td>A list of up to 30 numbers. The list may contain numeric values, cell references, range references, or array constants. Text in cells referenced by this function is treated as the number 0 (this includes zero-length text). Text entered in the argument list will be evaluated as a number, if possible; otherwise it causes a #VALUE! error. Text and logical values in arrays are ignored. Logical values referenced in cells or entered into the argument list are evaluated as numeric 1 if TRUE and numeric 0 if FALSE.</td>
</tr>
</tbody>
</table>

Example
This function returns .55634864:

STDEV(4.0, 3.0, 3.0, 3.5, 2.5, 4.0, 3.5)

See Also
STDEV, STDEVP, STDEVPA, VAR, VARA, VARP, VARPA

STDEVP

Description
Returns the standard deviation of a population based on an entire population of values. The standard deviation of a population represents an average of deviations from the population mean within a list of values.

Syntax
STDEVP (number_list)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number_list</td>
<td>A list of up to 30 numbers. The list may contain numeric values, cell references, range references, or array constants. Text and logical values in range references and array constants are ignored. Text entered in the argument list will be evaluated as a number, if possible; otherwise it will cause a #VALUE! error. Logical values entered into the argument list are evaluated as numeric 1 if TRUE and numeric 0 if FALSE.</td>
</tr>
</tbody>
</table>
Example
This function returns .52:

```
STDEVP(4.0, 3.0, 3.0, 3.5, 2.5, 4.0, 3.5)
```

See Also
STDEV, VAR, VARP

---

### STDEVPA

**Description**
Returns the standard deviation of a population based on an entire population of values. The standard deviation of a population represents an average of deviations from the population mean within a list of values.

This function is equivalent to the STDEVP function, but its implementation treats text and logical values in cell and range references differently.

**Syntax**
```
STDEVPA ( number_list )
```

**Argument Description**
- `number_list`:
  - A list of up to 30 numbers. The list may contain numeric values, cell references, range references, or array constants.

  - Text in cells referenced by this function is treated as the number 0 (this includes zero-length text). Text entered in the argument list will be evaluated as a number, if possible; otherwise it causes a #VALUE! error. Text and logical values in arrays are ignored.

  - Logical values referenced in cells or entered into the argument list are evaluated as numeric 1 if TRUE and numeric 0 if FALSE.

Example
This function returns .52:

```
STDEVPA(4.0, 3.0, 3.0, 3.5, 2.5, 4.0, 3.5)
```

See Also
STDEV, STDEVA, STDEVP, VAR, VARA, VARP, VARPA

---

### STEYX

**Description**
Computes the standard error of the predicted y-value for each x in the regression.

**Syntax**
```
STEYX (arrayX, arrayY)
```
### REMARKS

This function pairs up the numbers in the two ranges by moving left-to-right through each sequential row.

### EQUATION

\[
\frac{1}{n(n-2)} \left( n \sum y^2 - \left( \sum y \right)^2 \right) - \left( \frac{n \sum xy - \left( \sum x \right) \left( \sum y \right)}{n \sum x^2 - \left( \sum x \right)^2} \right)
\]

### EXAMPLES

The examples are based on the following worksheet.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>0</td>
</tr>
</tbody>
</table>

This function returns 3.651484:

\[\text{STEFX(A1:A5, B1:B5)}\]

This function returns 2.459675:

\[\text{STEFX(A1:B2, D2:D5)}\]

### SUBSTITUTE

**Description**

Replaces a specified part of a text string with another text string.

**Syntax**

\[\text{SUBSTITUTE (text, old_text, new_text [, instance])}\]
**SUBSTITUTE**

**Description**
Performs one of 11 functions on the specified data ranges or other cell references.

**Syntax**
SUBTOTAL (function_index, references)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>function_index</td>
<td>The index number of the function performed on the cells in the specified references, as follows:</td>
</tr>
<tr>
<td>index_number</td>
<td>function</td>
</tr>
<tr>
<td>1</td>
<td>AVERAGE</td>
</tr>
<tr>
<td>2</td>
<td>COUNT</td>
</tr>
<tr>
<td>3</td>
<td>COUNTA</td>
</tr>
<tr>
<td>4</td>
<td>MAX</td>
</tr>
<tr>
<td>5</td>
<td>MIN</td>
</tr>
<tr>
<td>6</td>
<td>PRODUCT</td>
</tr>
<tr>
<td>7</td>
<td>STDEV</td>
</tr>
<tr>
<td>8</td>
<td>STDEVP</td>
</tr>
<tr>
<td>9</td>
<td>SUM</td>
</tr>
<tr>
<td>10</td>
<td>VAR</td>
</tr>
<tr>
<td>11</td>
<td>VARP</td>
</tr>
</tbody>
</table>

| references | 1-29 data ranges or cell references |

**Remarks**
SUBTOTAL ignores embedded SUBTOTAL functions. See Examples.

**Examples**
This function returns “Second Quarter Results”:

```
SUBSTITUTE("First Quarter Results", "First", "Second")
```

This function returns "Shipment 45, Bin 52”:

```
SUBSTITUTE("Shipment 45, Bin 45", "45", "52", 2)
```

**See Also**
REPLACE, TRIM
Examples

The examples are based on the following worksheets.

```
+---+---+---+
| A | B | C |
+---+---+---+
| 1 | 2 | 3 |
| 2 | 4 | 6 |
| 3 | 7 | 9 |
```

This function returns 5:

```plaintext
SUBTOTAL(1,A1:C3)
```

This function returns 45:

```plaintext
SUBTOTAL(9,A1:C3)
```

```
+---+---+---+---+---+
| D | E | F |
+---+---+---+
| 1 | 2 | 3 |
| 2 | 4 | 6 |
| 3 | 7 | 9 |
```

This function returns 4.875:

```plaintext
SUBTOTAL(1,A1:C3)
```

This function returns 39:

```plaintext
SUBTOTAL(9,A1:C3)
```

See Also

AVERAGE, COUNT, COUNTA, MAX, MIN, PRODUCT, STDEV, STDEVP, SUM, VAR, VARP

---

**SUM**

Description

Returns the sum of the supplied numbers.

Syntax

```plaintext
SUM( number_list )
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number_list</td>
<td>A list of as many as 30 numbers. The list can contain numbers, logical values, text representations of numbers, range references, or array constants. Error values or text that cannot be translated into numbers return errors. If a range reference is included in the list, text, logical expressions, and empty cells in the range are ignored.</td>
</tr>
</tbody>
</table>

Examples

This function returns 6000:

```plaintext
SUM(1000, 2000, 3000)
```
This function returns 4000 when each cell in the range contains 1000:

```
SUM(A10:D10)
```

**See Also**
AVERAGE, COUNT, COUNTA, PRODUCT, SUMSQ

---

## SUMIF

### Description
Returns the sum of the specified cells based on the given criteria.

### Syntax
```
SUMIF (range, criteria, sum_range)
```

### Argument Description

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>range</code></td>
<td>The range of cells you want evaluated.</td>
</tr>
<tr>
<td><code>criteria</code></td>
<td>A number, expression, or text that defines which cells are added. For example, <code>criteria</code> can be expressed as 15, “15”, “&gt;15”, “cars”.</td>
</tr>
<tr>
<td><code>sum_range</code></td>
<td>The actual cells to sum. These cells are only summed if their corresponding cells in <code>range</code> match the criteria. If this argument is omitted, the cells in <code>range</code> are summed.</td>
</tr>
</tbody>
</table>

**See Also** AVERAGE, COUNT, COUNTA, COUNTIF, PRODUCT, SUM

---

## SUMPRODUCT

### Description
Multiplies the corresponding cells in the given ranges, then returns the sum of those products.

### Syntax
```
SUMPRODUCT (range_list)
```

### Argument Description

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>range_list</code></td>
<td>Two or more range references that provide the sets of numbers you want to multiply. The values in the upper left cell in each range are multiplied together, then the values in the next cell, etc. All the products are then summed.</td>
</tr>
</tbody>
</table>

All the ranges in `range_list` must contain the same number of cells in the same arrangement. That is, if the first range is three rows deep and three columns wide, the second and subsequent ranges must also be three rows deep and three columns wide.

**Remarks** Excel documentation states that SUMPRODUCT takes “2 to 30” arguments. Both Excel and Formula One for Java support 1 to 30 arguments for this function.
Examples

The following examples use this worksheet.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>6</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>6</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SUM</td>
</tr>
</tbody>
</table>

This function returns 630:

```
SUMPRODUCT(A1:C1,A2:C2,A3:C3)
```

This function returns 50:

```
SUMPRODUCT(A1:A3,C1:C3)
```

See Also

SUM

SUMSQ

Description

Squares each of the supplied numbers and returns the sum of the squares.

Syntax

```
SUMSQ( number_list )
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number_list</td>
<td>A list of as many as 30 numbers. The list can contain numbers, logical values, text representations of numbers, range references, or array constants. Error values or text that cannot be translated into numbers return errors. If a range reference is included in the list, text, logical expressions, and empty cells in the range are ignored.</td>
</tr>
</tbody>
</table>

Example

This function returns 302:

```
SUMSQ(9, 10, 11)
```

See Also

SUM
**SUMX2MY2**

**Description**
Computes \( x^2 - y^2 \) for two sets of numbers, \( arrayX \) and \( arrayY \), paired up one-to-one. The result is part of many statistical computations.

**Syntax**
\[
\text{SUMX2MY2} \left( array1, array2 \right)
\]

**Remarks**
This function pairs up the numbers in the two ranges by moving left-to-right through each sequential row.

**Examples**
The examples are based on the following worksheet.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>0</td>
</tr>
</tbody>
</table>

This function returns 45:
\[
\text{SUMX2MY2} \left( A1:A5, B1:B5 \right)
\]

This function returns -42:
\[
\text{SUMX2MY2} \left( A1:A3, B2:D2 \right)
\]

**See Also**
SUMX2PY2, SUMXMY2

---

**SUMX2PY2**

**Description**
Computes \( x^2 + y^2 \) for two sets of numbers, \( arrayX \) and \( arrayY \), paired up one-to-one. The result is part of many statistical computations.

**Syntax**
\[
\text{SUMX2PY2} \left( array1, array2 \right)
\]
### Remarks
This function pairs up the numbers in the two ranges by moving left-to-right through each sequential row.

### Examples
The examples are based on the following worksheet.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>0</td>
<td>9</td>
<td>0</td>
</tr>
</tbody>
</table>

This function returns 285:

\[
\text{SUMX2PY2}(A1:A5, B1:B5)
\]

This function returns 256:

\[
\text{SUMX2PY2}(A1:A3, B2:D2)
\]

### See Also
SUMX2MY2, SUMXMY2

---

### SUMXMY2

**Description**
Computes \((x - y)^2\) for two sets of numbers, \(arrayX\) and \(arrayY\), paired up one-to-one. The result is part of many statistical computations.

**Syntax**

\[
\text{SUMX2MY2}(array1, array2)
\]

**Remarks**
This function pairs up the numbers in the two ranges by moving left-to-right through each sequential row.
242 Formula One for Java Function Reference

Examples

The examples are based on the following worksheet.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>0</td>
</tr>
</tbody>
</table>

This function returns 85:

SUMXMY2(A1:A5, B1:B5)

This function returns 30:

SUMXMY2(A1:A3, B2:D2)

See Also

SUMX2MY2, SUMX2PY2

SYD

Description

Returns the depreciation of an asset for a specified period using the sum-of-years method. This depreciation method uses an accelerated rate, where the greatest depreciation occurs early in the useful life of the asset.

Syntax

SYD (cost, salvage, life, period)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cost</td>
<td>The initial cost of the asset.</td>
</tr>
<tr>
<td>salvage</td>
<td>The salvage value of the asset.</td>
</tr>
<tr>
<td>life</td>
<td>The number of periods in the useful life of the asset.</td>
</tr>
<tr>
<td>period</td>
<td>The period for which to calculate the depreciation. The time units used to determine period and life must match.</td>
</tr>
</tbody>
</table>

Example

This function returns 1607.14:

SYD(10000, 1000, 7, 3)

See Also

DDB, SLN, VDB
**T**

**Description**
Tests the supplied value and returns the value if it is text.

**Syntax**
T (value)

**Remarks**
Empty text ("") is returned for any value that is not text.

**Examples**
This function returns Report:
T("Report")

This function returns empty text (" ") if A4 contains a number:
T(A4)

**See Also**
N, VALUE

---

**TAN**

**Description**
Returns the tangent of the specified angle.

**Syntax**
TAN (number)

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>The angle in radians. To convert a number expressed as degrees to radians, use the RADIANS function.</td>
</tr>
</tbody>
</table>

**Examples**
This function returns 0.752:
TAN(0.645)

This function returns 1:
TAN(45*PI()/180)

**See Also**
ATAN, ATAN2, PI, TANH
TANH

Description
Returns the hyperbolic tangent of a number.

Syntax
TANH ( number )

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>Any number.</td>
</tr>
</tbody>
</table>

Examples
This function returns −0.96:
TANH(-2)

This function returns 0.83:
TANH(1.2)

See Also
ATANH, COSH, SINH, TAN

TBILLEQ

Description
Computes the bond-equivalent yield for a treasury bill.

Syntax
TBILLEQ ( settlement, maturity, discount )

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>settlement</td>
<td>The date when the security is traded to the buyer. It must be later than the issue date.</td>
</tr>
<tr>
<td>maturity</td>
<td>The date the security expires and the remaining amount is paid. It must fall during the 1-year period following settlement.</td>
</tr>
<tr>
<td>discount</td>
<td>The T-bill’s discount rate, represented as a decimal. It must be larger than 0.</td>
</tr>
</tbody>
</table>

Equation
\[
\frac{(365 \times \text{discount})}{360 - (\text{discount} \times \text{DSM})}
\]

where DSM is the number of days from settlement to maturity, computed according to the 360-day year, the standard for bond interest computations.

Examples
This function returns 0.1026, or 10.26%:
TBILLEQ("6/28/91","10/23/91",0.098)

See Also
TBILLPRICE, TBILLYIELD
### TBILLPRICE

**Description**
Computes the price per $100 face value for a treasury bill.

**Syntax**
```
TBILLPRICE ( settlement, maturity, discount )
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>settlement</td>
<td>The date when the security is traded to the buyer. It must be later than the issue date.</td>
</tr>
<tr>
<td>maturity</td>
<td>The date the security expires and the remaining amount is paid. It must fall during the 1-year period following settlement.</td>
</tr>
<tr>
<td>discount</td>
<td>The T-bill’s discount rate, represented as a decimal. It must be larger than 0.</td>
</tr>
</tbody>
</table>

**Equation**

\[
100 \times \left( 1 - \frac{\text{discount} \times \text{DSM}}{360} \right)
\]

where DSM is the number of actual days from `settlement` to `maturity`.

**Examples**
This function returns 96.815:
```
TBILLPRICE("6/28/91", "10/23/91", 0.098)
```

**See Also**
TBILLEQ, TBILLYIELD

### TBILLYIELD

**Description**
Computes the yield for a treasury bill.

**Syntax**
```
TBILLYIELD ( settlement, maturity, price )
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>settlement</td>
<td>The date when the security is traded to the buyer. It must be later than the issue date.</td>
</tr>
<tr>
<td>maturity</td>
<td>The date the security expires and the remaining amount is paid. It must fall during the 1-year period following settlement.</td>
</tr>
<tr>
<td>price</td>
<td>The T-bill’s price per $100 face value. It must be larger than 0.</td>
</tr>
</tbody>
</table>

**Remarks**
The bond-equivalent yield, which is the discount rate quoted for most treasury bills, is calculated with the TBILLEQ function.
Equation
\[ \frac{100 - price}{price} \times \frac{360}{DSM} \]
where DSM is the number of actual days from settlement to maturity.

Examples
This function returns 0.0469:
TBILLYIELD("6/28/91", "10/23/91", 98.5)

See Also
TBILLEQ, TBILLPRICE

---

**TDIST**

**Description**
Computes the complementary student’s T-distribution, one of the methods of determining whether two samples (often the experimental sample and a control group) are statistically different.

This function assumes the value you are testing is part of a normal distribution. You may choose to compute a one-tailed or two-tailed distribution.

**Syntax**
TDIST (x, df, tails)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>The value at which you want to evaluate the distribution. It must be a positive number.</td>
</tr>
<tr>
<td>df</td>
<td>A positive integer indicating the number of degrees of freedom. Decimal values will be truncated to integers.</td>
</tr>
<tr>
<td>tails</td>
<td>Enter 1 for a one-tailed distribution, 2 for two-tailed.</td>
</tr>
</tbody>
</table>

**Remarks**
This function returns a probability value between zero and one. Values closer to one indicate a high probability that the samples are similar. Lower values indicate a lower probability that the samples are similar.

**Equation**
\[ \frac{\Gamma\left(\frac{k+1}{2}\right)}{\sqrt{k\pi}\Gamma\left(\frac{k}{2}\right)} \int_{-\infty}^{\infty} \left(1 + \frac{x^2}{k}\right)^{-\frac{k+1}{2}} \, dx \]

...where k is the degrees of freedom.

**Examples**
This function returns 0.089213:
TDIST(1.75, 3, 1)

This function returns 0.178425:
TDIST(1.75, 3, 2)

**See Also**
BETADIST, BINOMDIST, CHIDIST, COMBIN, CRITBINOM, FDIST, GAMMADIST, NORMSDIST, TINV
TEXT

Description
Returns the given number as text, using the specified formatting.

Syntax
TEXT ( number, format)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>Any value, a formula that evaluates to a number, or a reference to a cell that contains a value.</td>
</tr>
<tr>
<td>format</td>
<td>A string representing a number format. The string can be any valid format string including “General,” “M/DD/YY,” or “H:MM AM/PM.” The format must be surrounded by a set of double quotation marks. Asterisks cannot be included in format.</td>
</tr>
</tbody>
</table>

Examples
This function returns 123.620:
TEXT(123.62, "0.000")

This function returns 10/19/94:
TEXT(34626.2, "MM/DD/YY")

See Also
DOLLAR, FIXED, T, VALUE

TIME

Description
Returns a serial number for the supplied time.

Syntax
TIME ( hour, minute, second)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hour</td>
<td>A number from 0 to 23.</td>
</tr>
<tr>
<td>minute</td>
<td>A number from 0 to 59.</td>
</tr>
<tr>
<td>second</td>
<td>A number from 0 to 59.</td>
</tr>
</tbody>
</table>

Examples
This function returns .52:
TIME(12, 26, 24)

This function returns .07:
TIME(1, 43, 34)

See Also
HOUR, MINUTE, NOW, SECOND, TIMEVALUE
### TIMEVALUE

**Description**
Returns a serial number for the supplied text representation of time.

**Syntax**
TIMEVALUE( text )

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>A time in text format.</td>
</tr>
</tbody>
</table>

**Examples**
This function returns .07:
TIMEVALUE("1:43:43 am")

This function returns .59:
TIMEVALUE("14:10:07")

**See Also**
HOUR, MINUTE, NOW, SECOND, TIME

### TINV

**Description**
Computes the x value corresponding to a specified probability value, given a specified number of degrees of freedom. TINV assumes that this is a two-tailed test of a normal distribution curve.

TINV is the inverse of TDIST.

**Syntax**
TINV( prob, df )

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>prob</td>
<td>A number between 0 and 1 indicating the probability for which you want the corresponding x value.</td>
</tr>
<tr>
<td>df</td>
<td>A positive integer indicating the number of degrees of freedom. Decimal values will be truncated to integers.</td>
</tr>
</tbody>
</table>

**Remarks**
This is the complementary version of TINV.

**Examples**
This function returns 9.925:
TINV(0.01, 2)

This function returns 3.182:
TINV(0.05, 3)

**See Also**
BETADIST, BINOMDIST, CHIDIST, COMBIN, CRITBINOM, FDIST, GAMMADIST, NORMDIST, TDIST
TODAY
Description
Returns the current date as a serial number.
Syntax
TODAY ( )
Remarks
This function is updated only when the worksheet is recalculated.
See Also
DATE, DAY, NOW

TRANSPOSE
Description
Places a copy of the contents of a range into a new range in which the original rows are now columns and the original columns are now rows.

Note
This is an array function. For information, see “Array Functions” on page 13.

Syntax
TRANSPOSE ( range)

Argument Description
range A range reference or the name of a named range reference.

Results range
Before entering this function, select a results range with the same number of rows as the argument range has columns and the same number of columns as the argument range has rows. If you select too few cells, some cells will be truncated. If you select too many cells, the extra cells will contain the #N/A error.

Example
The following example uses this worksheet.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

TRANSPOSE(A1:B5) returns the following results range:

<table>
<thead>
<tr>
<th>4</th>
<th>4</th>
<th>4</th>
<th>4</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>17</td>
</tr>
</tbody>
</table>

See Also
SIGN
TREND

Description
Computes result values from the fitted curve of a multiple linear regression for a group of specified observations relative to a group of specified independent variables. This function is useful for extrapolating information based on a trend in existing data.

Note
If you want TREND to return more than one result value, you must enter it as an array function. For information, see “Array Functions” on page 13.

Syntax
TREND (known_y’s [, known_x’s] [, new_x’s] [, constant])

Argument             Description

known_y’s             A list of observed values for the dependent variable. This must be a range reference or an array constant containing all numeric values and no empty cells.

For a regression with a single independent variable, enter any type of range. For a regression with multiple independent variables, enter a single row or column of values.

If you are using array constants, see “Array Constants in Arguments” on page 9 for information on how to enter the array.

[known_x’s]           Optional. A list of observed values for the independent variable(s). This must be a range reference or an array constant containing all numeric values and no empty cells.

For a regression with a single independent variable, when known_y’s is a single row or column, enter a range that exactly matches the size and shape of the known_y’s range. When known_y’s is more than one row and column, known_x’s must be a range containing the same number of values as the known_y’s range, although the two ranges may be different shapes.

For a regression with multiple independent variables, when known_y’s is a single row, enter a contiguous row of values for each independent variable. The number of columns must match the number of columns in known_y’s. Similarly, when known_y’s is a single column, enter a contiguous column of values for each independent variable, ensuring that the number of rows matches known_y’s.

If you are using array constants, see “Array Constants in Arguments” on page 9 for information on how to enter the array.

To fit the regression to a formula that uses polynomials, see “Polynomials,” below.

When you omit this argument, the function assumes a single independent variable of the sequence {1, 2, 3, ...}. 
Remarks

This function uses LINEST internally to compute the regression’s coefficients. The resulting formula is then used to compute the requested values.

Results range

Before entering this function, select a results range according to the following criteria:

- If there is a single independent variable, the results range should match the size and shape of the new_x’s argument.
- If there are multiple independent variables, the results range depends on the number of data sets in new_x’s. If there is only one data set, the results range should be a single cell. For multiple data sets, the results range should be a column if known_y’s is a column, a row if known_y’s is a row, with the same number of cells as data sets entered in new_x’s. If you omitted the new_x’s argument, the results range must match the size and shape of the known_y’s argument.

Equation

The regression computed by TREND attempts to fit the following formula:

\[ y = C_1x_1 + C_2x_2 + \ldots + C_nx_n + b \]

where \( C \) is the coefficient determined internally by the LINEST function, \( n \) is the number of independent variables, and \( b \) is the constant.

Polynomials

To create a polynomial regression, the data for the independent variable must be set up carefully. Create a worksheet in which column A represents \( x \) and contains the appropriate data or formulas. For column B, which will contain \( x^2 \), fill the cells with a formula such as \( =A1^COLUMN() \), which will raise the data in column A to the 2nd power. Fill column C with the same formula to raise the data in column A to the 3rd power.

The data for the new_x’s variable should be set up in the same manner.
Example

The following example of multiple linear regression attempts to correlate physiological data on preadolescent boys with their maximal oxygen uptake. The data are given in the table below.

<table>
<thead>
<tr>
<th>Maximal O2 Uptake (y)</th>
<th>Age (years)</th>
<th>Height (cm)</th>
<th>Weight (kilos)</th>
<th>Chest Depth (cm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.54</td>
<td>8.4</td>
<td>132.0</td>
<td>29.1</td>
<td>14.4</td>
</tr>
<tr>
<td>1.74</td>
<td>8.7</td>
<td>135.5</td>
<td>29.7</td>
<td>14.5</td>
</tr>
<tr>
<td>1.32</td>
<td>8.9</td>
<td>127.7</td>
<td>28.4</td>
<td>14.0</td>
</tr>
<tr>
<td>1.50</td>
<td>9.9</td>
<td>131.1</td>
<td>28.8</td>
<td>14.2</td>
</tr>
<tr>
<td>1.46</td>
<td>9.0</td>
<td>130.0</td>
<td>25.9</td>
<td>13.6</td>
</tr>
<tr>
<td>1.35</td>
<td>7.7</td>
<td>127.6</td>
<td>27.6</td>
<td>13.9</td>
</tr>
<tr>
<td>1.53</td>
<td>7.3</td>
<td>129.9</td>
<td>29.0</td>
<td>14.0</td>
</tr>
<tr>
<td>1.71</td>
<td>9.9</td>
<td>138.1</td>
<td>33.6</td>
<td>14.6</td>
</tr>
<tr>
<td>1.27</td>
<td>9.3</td>
<td>126.6</td>
<td>27.7</td>
<td>13.9</td>
</tr>
<tr>
<td>1.50</td>
<td>8.1</td>
<td>131.8</td>
<td>30.8</td>
<td>14.5</td>
</tr>
</tbody>
</table>

We want to use this data to extrapolate the maximal oxygen uptake for a boy with the following characteristics:

<table>
<thead>
<tr>
<th>Age (years)</th>
<th>Height (cm)</th>
<th>Weight (kilos)</th>
<th>Chest Depth (cm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>x_1</td>
<td>x_2</td>
<td>x_3</td>
<td>x_4</td>
</tr>
<tr>
<td>10</td>
<td>137</td>
<td>33.5</td>
<td>14.5</td>
</tr>
</tbody>
</table>

The data were entered in a worksheet with the row and column arrangement shown here and a constant value of True. The TREND function returned a value of 1.66295.

See Also

GROWTH, INTERCEPT, LINEST, LOGEST, SLOPE

TRIM

Description

Removes all spaces from text except single spaces between words.

Syntax

TRIM (text)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>Any text string or a reference to a cell that contains a text string.</td>
</tr>
</tbody>
</table>

Remarks

Text that is imported from another environment may require this function.
**Example**  
This function returns Level 3, Gate 45:  
\[
\text{TRIM(" Level 3, Gate 45 ")}
\]

**See Also**  
CLEAN, MID, REPLACE, SUBSTITUTE

---

**TRIMMEAN**

**Description**  
Computes the mean of a list of numbers after reducing a specified percent of the members of the list. The list is sorted in ascending order, then half of \( F \) percent of the list members are eliminated from the top and half from the bottom of the list, and finally the mean of the remaining members is taken.  

This function is useful for eliminating outliers in a sample.

**Syntax**  
\[
\text{TRIMMEAN (} \text{numbers}, \text{ } \text{F} \text{)}
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{numbers} )</td>
<td>A range reference or array constant containing numbers. All numeric values, including 0, are used. Text and logical values in cell references and arrays is ignored. Empty cells are ignored.</td>
</tr>
<tr>
<td>( \text{F} )</td>
<td>The percent of the members of the list of values that you want to eliminate. The list will be sorted in ascending order and half of this percentage of members will be eliminated from the bottom of the list, half from the top. The percentage will be rounded down. For example, if you chose to eliminate 15% of a 100-member list, 7 members will be deleted from the top and 7 from the bottom of the list, totalling only 14% eliminated.</td>
</tr>
</tbody>
</table>

**Example**  
This function returns 5.5:  
\[
\text{TRIMMEAN}((1,2,3,4,5,6,7,8,9,15),0.2)
\]

First, the top 10\% and bottom 10\% of the list members (1 and 15) are eliminated. Then the mean of the remaining numbers is taken.

**See Also**  
AVERAGE

---

**TRUE**

**Description**  
Returns the logical value True. This function always requires the trailing parentheses.

**Syntax**  
\[
\text{TRUE (} \)
\]

**See Also**  
FALSE
TRUNC

Description       Truncates the given number to an integer.
Syntax           TRUNC (number [, precision])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>Any value.</td>
</tr>
<tr>
<td>[precision]</td>
<td>Optional. The number of decimal places allowed in the truncated number. If this argument is omitted, 0 is used.</td>
</tr>
</tbody>
</table>

Remarks         TRUNC removes the fractional part of a number to the specified precision without rounding the number.

Examples         This function returns 123.45:
                  TRUNC(123.456, 2)

                  This function returns 9800:
                  TRUNC(9899.435, -2)

See Also         CEILING, FLOOR, INT, MOD, ROUND

TTEST

Description       Computes the student’s T-distribution from the data in two specified arrays, then computes the probability.
Syntax            TTEST (array1, array2, tails, type)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array1</td>
<td>Two range references or an array constants containing numeric values. Text, logical values, and empty cells are ignored. Array1 and array2 must contain the same potential number of values.</td>
</tr>
<tr>
<td>array2</td>
<td></td>
</tr>
<tr>
<td>tails</td>
<td>Enter 1 for a one-tailed distribution, 2 for two-tailed.</td>
</tr>
<tr>
<td>type</td>
<td>The type of T-test you want to perform on the given arrays. The options are:</td>
</tr>
<tr>
<td></td>
<td>1 Paired values</td>
</tr>
<tr>
<td></td>
<td>2 Two distributions with the same variance</td>
</tr>
<tr>
<td></td>
<td>3 Two distributions with different variances</td>
</tr>
</tbody>
</table>

Remarks          In type option 1, aggregate values for mean, variance, and counts are determined for both arrays. In options 2 and 3, those values are determined for each array independently.
TTEST uses TDIST to compute the T-distribution. TDIST requires that its degrees of freedom argument be an integer. However, in *type* option 3, the computed degrees of freedom will usually not be an integer. Therefore, in this case, the function uses the incomplete beta function instead of TDIST.

**Equation**


**Examples**

This function returns 0.77:

```
TTEST ({2,4,6,8,10,12}, {8,4,6,1,0,18}, 2, 1)
```

**See Also**

CHITEST, FTEST, ZTEST

---

**TYPE**

**Description**

Returns the argument type of the given expression.

**Syntax**

```
TYPE ( expression )
```

**Remarks**

The valid values returned by this argument are:

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Number</td>
</tr>
<tr>
<td>2</td>
<td>Text string</td>
</tr>
<tr>
<td>4</td>
<td>Logical value</td>
</tr>
<tr>
<td>16</td>
<td>Error value</td>
</tr>
</tbody>
</table>

**Examples**

This function returns 1 if cell A1 contains a number:

```
TYPE(A1)
```

This function returns 2:

```
TYPE("Customer")
```

**See Also**

ISBLANK, ISERR, ISERROR, ISLOGICAL, ISNA, ISNONTEXT, ISNUMBER, ISREF, ISTEXT
**UPPER**

**Description**
Changes the characters in the specified string to uppercase characters.

**Syntax**
UPPER (text)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>Any string.</td>
</tr>
</tbody>
</table>

**Remarks**
Numeric characters in the string are not changed.

**Examples**
This function returns 3RD QUARTER:
UPPER("3rd Quarter")

This function returns JOHN DOE:
UPPER("JOHN DOE")

**See Also**
LOWER, PROPER

**USDOLLAR**

**Description**
Returns the specified number as text using the US Dollar format and the supplied precision. Omitting the precision argument assumes two decimal places.

**Syntax**
USDOLLAR (number [, precision])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number</td>
<td>A number, a formula that evaluates to a number, or a reference to a cell that contains a number.</td>
</tr>
<tr>
<td>[precision]</td>
<td>Optional. A value representing the number of decimal places to the right of the decimal point. If this argument is omitted, 2 is used.</td>
</tr>
</tbody>
</table>

**Examples**
This function returns $1023.79:
USDOLLAR(1023.789)

This function returns $500:
USDOLLAR(495.301, -2)

**See Also**
DOLLAR, FIXED, TEXT, VALUE
### VALUE

**Description**
Returns the specified text as a number.

**Syntax**
```
VALUE ( text )
```

**Examples**
- This function returns 9800:
  
  ```
  VALUE(9800)
  ```
- This function returns 123:
  
  ```
  VALUE("123")
  ```

**See Also**
DOLLAR, FIXED, TEXT

### VAR

**Description**
Returns the variance of a population based on a sample of values.

**Syntax**
```
VAR ( number_list )
```

**Example**
- This function returns .31:
  
  ```
  VAR(4.0, 3.0, 3.0, 3.5, 2.5, 4.0, 3.5)
  ```

**See Also**
STDEV, STDEVP, VARP
VARA

Description
Returns the variance of a population based on a sample of values.
This function is equivalent to the VAR function, but its implementation treats text and logical values in cell and range references differently.

Syntax
VARA (number_list)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number_list</td>
<td>A list of up to 30 arguments separated by commas. The list may contain numeric values, cell references, range references, or array constants. Text in cells referenced by this function is treated as the number 0 (this includes zero-length text). Text entered in the argument list will be evaluated as a number, if possible; otherwise it will cause a #VALUE! error. Text and logical values in arrays are ignored. Logical values referenced in cells or entered into the argument list are evaluated as numeric 1 if TRUE and numeric 0 if FALSE.</td>
</tr>
</tbody>
</table>

Example
This function returns .30952381:
VARA(4.0, 3.0, 3.0, 3.5, 2.5, 4.0, 3.5)

See Also
STDEV, STDEVA, STDEVP, STDEVP A, VAR, VARP, VARPA

VARP

Description
Returns the variance of a population based on an entire population of values.

Syntax
VARP (number_list)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number_list</td>
<td>A list of 1 to 30 numbers separated by commas. The list may contain numeric values, cell references, range references, or array constants. All numeric values, including 0, are used. Text and logical values in cells and arrays are ignored. Text entered in the argument list will be evaluated as a number, if possible; otherwise it will cause a #VALUE! error. Logical values entered into the argument list are evaluated as numeric 1 if TRUE and numeric 0 if FALSE.</td>
</tr>
</tbody>
</table>

Example
This function returns .27:
VARP(4.0, 3.0, 3.0, 3.5, 2.5, 4.0, 3.5)

See Also
STDEV, STDEVP, VAR
**VARPA**

**Description**
Returns the variance of a population based on the entire population of values.

This function is equivalent to the VARP function, but its implementation treats text and logical values in cell and range references differently.

**Syntax**
VARPA( number_list)

**Example**
This function returns .26530612:

VARPA(4.0, 3.0, 3.0, 3.5, 2.5, 4.0, 3.5)

**See Also**
STDEV, STDEVA, STDEVP, STDEVPA, VAR, VARA, VARP

---

**VDB**

**Description**
Returns the depreciation of an asset for a specified period using a variable method of depreciation.

**Syntax**
VDB( cost, salvage, life, start_period, end_period [, factor] [, method])

**Arguments**
- **cost**: The initial cost of the asset.
- **salvage**: The salvage value of the asset.
- **life**: The number of periods in the useful life of the asset.
- **start_period**: The beginning period for which to calculate the depreciation. The time units used to determine start_period and life must match.
- **end_period**: The ending period for which to calculate the depreciation. The time units used to determine end_period and life must match.
- **factor** (optional): The rate at which the balance declines. If this argument is omitted, 2 (the double-declining balance factor) is used.
- **method** (optional): The method used to calculate the depreciation.
VLOOKUP

Description
Searches the first column of a table for a value and returns the contents of a cell in that table that corresponds to the location of the search value.

Syntax
VLOOKUP (search_item, search_range, column_index)

Remarks
VLOOKUP compares the information in the first column of search_range to the supplied search_item. When a match is found, information located in the same row and supplied column (column_index) is returned.

If search_item cannot be found in the first column of search_range, the largest value that is less than search_item is used. When search_item is less than the smallest value in the first column of the search_range, #REF! is returned.

Examples
The following examples use this worksheet.
This function returns $28,700:

VLOOKUP("Clark", A2:E9, 4)

This function returns 3961:

VLOOKUP("Lee", A2:E9, 3)

See Also
HLOOKUP, INDEX (non-array type), LOOKUP, MATCH

### WEEKDAY

**Description**
Returns the day of the week that corresponds to the supplied date.

**Syntax**

WEEKDAY (serial_number [order])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serial_number</td>
<td>The date as a serial number or as text (for example, 06-21-94 or 21-Jun-94).</td>
</tr>
<tr>
<td>[order]</td>
<td>Optional. This argument determines how the days of the week are numbered for the purposes of this function. If this argument is omitted, 1 is used. The options are:</td>
</tr>
<tr>
<td></td>
<td>1 = Sunday, 2 = Monday, 3 = Tuesday ... 7 = Saturday</td>
</tr>
<tr>
<td></td>
<td>2 = Monday, 2 = Tuesday, 3 = Wednesday ... 7 = Sunday</td>
</tr>
<tr>
<td></td>
<td>0 = Monday, 1 = Tuesday, 2 = Wednesday ... 6 = Sunday</td>
</tr>
</tbody>
</table>

**Examples**

This function returns 1, indicating Sunday:

WEEKDAY(34399.92)
This function returns 3, indicating Tuesday:

\[ \text{WEEKDAY("06/21/94")} \]

**See Also**

DAY, NOW, TEXT, TODAY

---

### WEEKNUM

**Description**

Returns a number indicating which week of the year the specified date falls in.

**Syntax**

\[ \text{WEEKNUM (date [, week\_begins])} \]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>Any date. Dates in the argument list must be entered as text, with quotation marks.</td>
</tr>
<tr>
<td>[week_begins]</td>
<td>Optional. Indicates whether Sunday or Monday is used as the starting point for a week. If this argument is omitted, Sunday is used. Decimal values will be truncated to integers. The options are: 1 Sunday starts the week. 2 Monday starts the week.</td>
</tr>
</tbody>
</table>

**Remarks**

Dates in the first week of January will return 1. The last few dates in a year will return 53.

**Examples**

This function returns 20:

\[ \text{WEEKNUM("5/9/99",1)} \]

This function returns 19:

\[ \text{WEEKNUM("5/9/99",2)} \]

**See Also**

WEEKDAY

---

### WEIBULL

**Description**

Computes the Weibull distribution.

**Syntax**

\[ \text{WEIBULL (x, alpha, beta, cumulative)} \]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>The value at which the function will be evaluated. It must be larger than or equal to 0.</td>
</tr>
<tr>
<td>alpha</td>
<td>The alpha parameter to the distribution. It must be larger than 0.</td>
</tr>
</tbody>
</table>
Equations

For the cumulative distribution: \( F(x; \alpha, \beta) = 1 - e^{-\left(\frac{x}{\beta}\right)^\alpha} \)

For the probability density function: \( f(x; \alpha, \beta) = \frac{\alpha}{\beta^{\alpha}} x^{\alpha-1} e^{-\left(\frac{x}{\beta}\right)^\alpha} \)

If \( \alpha = 1 \), \( \text{WEIBULL} \) returns the exponential distribution with \( \lambda = \frac{1}{\beta} \).

Examples

This function returns 0.439:

\( \text{WEIBULL}(100, 3, 120, \text{TRUE}) \)

This function returns 0.010:

\( \text{WEIBULL}(100, 3, 120, \text{FALSE}) \)

See Also

BINOMDIST, BETADIST, CHIDIST, COMBIN, CRITBINOM, EXPONDIST, FDIST, GAMMADIST, NORMDIST, POISSON

WORKDAY

Description

Returns the date that is a specified number of days before or after a specified date, not counting weekends and specified holidays.

Syntax

\( \text{WORKDAY} \ (\text{start\_date}, \text{days} \ [\text{holidays}]) \)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>start_date</td>
<td>The date from which calculations will start. Dates in the argument list must be entered as text, with quotation marks. Decimal values are truncated to integers.</td>
</tr>
<tr>
<td>days</td>
<td>The number of days before or after start_date. Enter a negative number to find days before, a positive number to find days after. Decimal values are truncated to integers.</td>
</tr>
<tr>
<td>[holidays]</td>
<td>Optional. A range reference or array constant containing a list of dates that represent holidays. The dates specified in holidays will not be counted as workdays in the function’s computation. Dates in array references must be entered as text, with quotation marks. Decimal values are truncated to integers.</td>
</tr>
</tbody>
</table>

Examples

This function returns 1/14/93:

\( \text{WORKDAY} \("12/15/92", 20, \{"12/24/92","12/25/92","1/1/93"\}) \)
This function returns 6/29/99:

\[
\text{WORKDAY("7/14/99",-10,"7/5/99")}
\]

See Also

NETWORKDAYS, WEEKDAY

XIRR

Description

Computes the internal rate of return for an investment with flexible periods.

Syntax

\[
\text{XIRR (values, dates [, guess])}
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>values</td>
<td>A range reference or array constant containing a series of payments to the investment. The first value in the range or array constant is the cost at the beginning of the investment; it is usually negative or 0. Positive values represent payments. Empty cells are evaluated as 0. The function requires at least one negative and at least one positive value in values. You may use a series of positive and negative values to show payments and costs.</td>
</tr>
<tr>
<td>dates</td>
<td>A range reference or array constant containing the dates each of the payments in values was made. The first date in the range or array constant is the initial investment date. All the other dates must be later than the initial date, but need not be in chronological order. All dates are truncated to integers. There must be the same number of potential values in dates as in values.</td>
</tr>
<tr>
<td>[guess]</td>
<td>Optional. An estimate of the rate of return. If you omit this argument, 0.1 or 10% will be used. Use this argument when the function returns the #NUM! error (see below).</td>
</tr>
</tbody>
</table>

Remarks

XIRR calculates rate of return similarly to IRR, only XIRR allows you to enter the exact dates of the payments. This may be useful in cases where, for example, the institution offering the investment is writing a contract that sets down the exact payment schedule.

All computations are based on a 365-day year.

Unlike other investment functions, XIRR may have a negative rate result.

This is an iterative function, which means that Formula One for Java will attempt to calculate the result over and over to a finer and finer estimate until it reaches a stable value. If after 100 tries the function still has not found a stable value, it returns the #NUM! error. In this case, you can re-enter the function with an estimate in the guess argument that may help the iteration converge. The guess argument is especially helpful if the data in values alternates between positive and negative values.
Equation

\[ 0 = \sum_{i=0}^{N} \frac{P_i}{(1 + rate) \frac{d_i - d_0}{365}} \]

...where \( d_i \) is the \( i \)th, or last, payment date, \( d_0 \) is the 0th payment date, and \( P_i \) is the \( i \)th, or last, payment. The software iterates to a solution to this equation.

Example

This function returns 0.024851, or just above 2%:

\[
XIRR([-1000,600,600],"1/1/85","1/1/90","1/1/95")
\]

See Also

PV, NPV, IRR, XNPV

---

XNPV

Description

Computes the net present value of an investment with flexible periods.

Syntax

\[ XNPV(rate, values, dates) \]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rate</td>
<td>The annual interest rate.</td>
</tr>
<tr>
<td>values</td>
<td>A range reference or array constant containing a series of payments to the investment. The first value in the range or array constant is the cost at the beginning of the investment; it is usually negative or 0. Positive values represent payments to the investment. Empty cells are evaluated as 0.</td>
</tr>
</tbody>
</table>
| dates    | A range reference or array constant containing the dates each of the payments in \( values \) was made. The first date in the range or array constant is the initial investment date. All the other dates must be later than the initial date, but need not be in chronological order. All dates are truncated to integers. 

There must be the same number of potential values in \( dates \) as in \( values \). |

Remarks

XNPV calculates net present value similarly to NPV, only XNPV allows you to enter the exact dates of the payments. This may be useful in cases where, for example, the institution offering the investment is writing a contract that sets down the exact payment schedule.

All computations are based on a 365-day year.

Equation

\[ \sum_{i=0}^{N} \frac{P_i}{(1 + rate) \frac{d_i - d_0}{365}} \]

...where \( d_i \) is the \( i \)th, or last, payment date, \( d_0 \) is the 0th payment date, and \( P_i \) is the \( i \)th, or last, payment.
Examples

This function returns -937.41:

\[ XNPV(0.6,\{-1000,600,600\},\{"1/1/85","1/1/90","1/1/95"\}) \]

See Also

PV, NPV, IRR, XIRR

---

YEAR

Description

Returns the year that corresponds to the supplied date.

Syntax

\[ \text{YEAR} \left( \text{serial\_number} \right) \]

Arguments and Description

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serial_number</td>
<td>The date as a serial number or as text (for example, 06-21-94 or 21-Jun-94).</td>
</tr>
</tbody>
</table>

Examples

This function returns 1993:

\[ \text{YEAR}(34328) \]

This function returns 1994:

\[ \text{YEAR}("06/21/94") \]

See Also

DAY, NOW, HOUR, MINUTE, MONTH, SECOND, TODAY, WEEKDAY

---

YEARFRAC

Description

Computes the fraction of a year between two specified dates.

Syntax

\[ \text{YEARFRAC} \left( \text{start\_date}, \text{end\_date}[, \text{calendar\_type}] \right) \]

Arguments and Description

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>start_date</td>
<td>The starting date. Dates in the argument list must be in the form of a serial number or text.</td>
</tr>
<tr>
<td>end_date</td>
<td>The ending date. Dates in the argument list must be in the form of a serial number or text.</td>
</tr>
<tr>
<td>[calendar_type]</td>
<td>Optional. One of five methods of counting days for computing interest. See “The calendar_type Argument” on page 19 for more information.</td>
</tr>
</tbody>
</table>

Remarks

This function truncates all arguments to integers.
Examples

This function returns 0.344:

YEARFRAC("1/11/97","5/15/97")

This function returns 1.344:

YEARFRAC("1/11/97","5/15/98")

See Also

DAYS360

YIELD

Description

Computes the annual yield of a security that pays periodic interest. Rate, redemption value, and price must be specified.

Syntax

YIELD (settlement, maturity, rate, price, redemption, frequency [, calendar_type])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>settlement</td>
<td>The date when the security is traded to the buyer. Decimal values will be truncated to integers.</td>
</tr>
<tr>
<td>maturity</td>
<td>The date the security expires and the remaining amount is paid to the investor. It must be later than settlement. Decimal values will be truncated to integers.</td>
</tr>
<tr>
<td>rate</td>
<td>The security’s annual coupon rate. The coupons pay at this rate divided by frequency.</td>
</tr>
<tr>
<td>price</td>
<td>The amount the buyer pays, per $100 face value.</td>
</tr>
<tr>
<td>redemption</td>
<td>The security’s redemption value per $100 face value. This is the amount paid at the settlement date that is not part of any final coupon payment.</td>
</tr>
<tr>
<td>frequency</td>
<td>The number of interest payments per year. See “The frequency Argument” on page 18 for more information.</td>
</tr>
<tr>
<td>[calendar_type]</td>
<td>Optional. One of five methods of counting days for computing interest. See “The calendar_type Argument” on page 19 for more information.</td>
</tr>
</tbody>
</table>

Examples

This function returns 0.0921:

YIELD("5/6/97","12/31/99",0.06,0.9253,100,4)

See Also

PRICE, PRICEDISC, PRICEMAT, YIELDDISC, YIELDMAT
## YIELDDISC

**Description**
Computes the annual yield of a discounted security, given a specified price and redemption value.

**Syntax**
YIELDDISC (settlement, maturity, price, redemption [calendar_type])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>settlement</td>
<td>The date when the security is traded to the buyer. Decimal values will be</td>
</tr>
<tr>
<td></td>
<td>truncated to integers.</td>
</tr>
<tr>
<td>maturity</td>
<td>The date the security expires and the remaining amount is paid to the</td>
</tr>
<tr>
<td></td>
<td>investor. It must be later than settlement. Decimal values will be</td>
</tr>
<tr>
<td></td>
<td>truncated to integers.</td>
</tr>
<tr>
<td>price</td>
<td>The amount the buyer pays, per $100 face value.</td>
</tr>
<tr>
<td>redemption</td>
<td>The security's redemption value per $100 face value. This is the amount</td>
</tr>
<tr>
<td></td>
<td>paid at maturity.</td>
</tr>
<tr>
<td>[calendar_type]</td>
<td>Optional. One of five methods of counting days for computing interest.</td>
</tr>
<tr>
<td></td>
<td>See “The calendar_type Argument” on page 19 for more information.</td>
</tr>
</tbody>
</table>

**Equation**

\[
\text{redemption} - \text{price} \over \text{price} \times \text{YEARFRAC (settlement, maturity, calendar_type)}
\]

**Examples**
This function returns 0.0244:

YIELDDISC("10/23/94","7/7/95",98.31,100)

**See Also**
DISC, PRICE, PRICEDISC, PRICEMAT, YIELD, YIELDMAT

## YIELDMAT

**Description**
Computes the annual yield on a security that pays interest only at maturity, given specified dates, interest rate, and price.

**Syntax**
YIELDMAT (settlement, maturity, issue, rate, price [calendar_type])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>settlement</td>
<td>The date when the security is traded to the buyer. Decimal values will be</td>
</tr>
<tr>
<td></td>
<td>truncated to integers.</td>
</tr>
<tr>
<td>maturity</td>
<td>The date the security expires and the remaining amount is paid to the</td>
</tr>
<tr>
<td></td>
<td>investor. It must be later than settlement. Decimal values will be</td>
</tr>
<tr>
<td></td>
<td>truncated to integers.</td>
</tr>
<tr>
<td>issue</td>
<td>The date the security was originally issued and began earning interest.</td>
</tr>
<tr>
<td></td>
<td>Decimal values will be truncated to integers.</td>
</tr>
</tbody>
</table>
Equation

\[
\frac{1}{\text{DSM}} \times \frac{100 + \text{DIM} \times \text{rate} \times 100}{(\text{price} + \text{DIS} \times \text{rate} \times 100) - 1}
\]

where the three-letter codes correspond to values you can compute using the YEARFRAC function, as shown in the following table.

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIM</td>
<td>Days from issue to maturity.</td>
<td>YEARFRAC(issue, maturity, calendar_type)</td>
</tr>
<tr>
<td>DIS</td>
<td>Days from issue to settlement.</td>
<td>YEARFRAC(issue, settlement, calendar_type)</td>
</tr>
<tr>
<td>DSM</td>
<td>Days from settlement to maturity.</td>
<td>YEARFRAC(settlement, maturity, calendar_type)</td>
</tr>
</tbody>
</table>

Example

This function returns 0.046:

\[ \text{YIELDMAT(“9/19/86”,”2/28/94”,”2/28/84”,0.0525,100.0154)} \]

See Also

PRICE, PRICEDISC, PRICEMAT, YIELD, YIELDDISC

ZTEST

Description

Computes the two-tailed probability of a z-test, which is a test of a specified value against a specified set of numbers. ZTEST returns the number of standard deviations that separate the specified value from the mean of the set of numbers.

Syntax

\[ \text{ZTEST (array, x [, \sigma ])} \]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>A list of numbers separated by commas. The list can be in the form of a range reference or array constant. Text and logical values are ignored.</td>
</tr>
<tr>
<td>x</td>
<td>The value you want to test. A blank cell will be evaluated as 0. Text will be evaluated as a number, if possible; otherwise it will cause a #VALUE! error. Logical values are evaluated as numeric 1 if TRUE and numeric 0 if FALSE.</td>
</tr>
<tr>
<td>[\sigma ]</td>
<td>Optional. The population standard deviation. If this argument is omitted, ZTEST calculates a sample standard deviation based on the values in array.</td>
</tr>
</tbody>
</table>
Equation

\[ 1 - \text{NORMSDIST}\left( \frac{\bar{x} - \mu}{\sigma/\sqrt{n}} \right) \]

Example

This function returns 0.99973395:

\[ \text{ZTEST}\left(\{1,2,3\}, 4\right) \]

See Also

CHITEST, FTEST, TTEST
Index

Symbols

! 7
- 11
% 11
& 4, 11, 67
* 11
+ 11
, 12
/ 11
<= 12
<> 12
= 12
> 12
>= 12
^ 11
[ ] 9, 14
' 7

A

ABS 29, 37
Absolute cell references 5
Absolute row and column reference 41
Absolute value 37
Absolute value of complex numbers 133
Accelerated rate in computing depreciation 90
ACCRINT vii, 25, 37
ACCRINTM vii, 25, 39
Accrued interest 37
ACOS 29, 39
ACOSH 29, 40
Add-in functions error 19
Adding complex numbers 143
Adding functions 237–242
Addition
operator 11
symbol 12
ADDRESS 28, 40
Addressing cells 4
AMORDEGRC vii, 25, 41
AMORLINC vii, 25, 43
AND 28, 44
Angles
arctangents of 46
computing arc cosine 39
tangents of 46

Annuities
computing interest per period 217
computing present value 213
computing principle paid on 207
computing the periodic payment of 205
Arc cosine 39
Arcsine, calculating 45
Arctangent 46
AREAS vii, 28, 44
Areas, formula to count, in a range 44
Argument ranges in array functions 13
Arguments in functions 3
Arithmetic mean 48
Array constants in function arguments 9
Array functions 13–14, 179, 249, 254
entering 14
ASCII text
and the CHAR function 61
and the CODE function 65
ASIN 29, 45
ASINH 29, 45
Assets
depreciating 259
depreciation using sum of years 242
ATAN 29, 46
ATAN2 29, 46
ATANH 29, 47
AVEDEV vii, 31, 47
AVERAGE 31, 48
AVERAGEA vii, 31, 48

B

Base-10, logarithm, computing 164
Bell curve, computing the 189
Bessel functions 49–51
BESSELI vii, 23, 49
BESSELJ vii, 23, 49
BESSELK vii, 23, 50
BESSELY vii, 23, 51
Beta distribution, computing cumulative 51
BETADIST vii, 31, 51
BETAINV vii, 31, 52
BIN2DEC vii, 23, 53
BIN2HEX vii, 23, 54
BIN2OCT vii, 23, 54
BIN2OCT, function 54
Binary numbers
  converting from  53–55
  converting to  90, 128, 192
BINOMDIST vii, 31, 55
Binomial distribution, computing  55
Bond-equivalent yield, computing for a T-bill  244
Borders, colored, around cells  10

C
  calendar_type argument  42, 43
CALL vii, 57
Case, changing  171, 212
CEILING 29, 60
CELL vii, 27, 57
Cell references
  database search criteria guidelines  17
  in function arguments  4
Cells
  address, creating as text  40
  detecting contents using the CELL function  58
  formatting  5
  detecting with the CELL function  57
  references in criteria ranges  17
  references to  4–8
CHAR 34, 61
Characters
  changing to uppercase  256
  CLEANing nonprintable  64
  maximum in a function  2
CHIDIST vii, 31, 61
CHIINV vii, 31, 62
Chi-square distribution  61, 62
CHOOSE 28, 63
CLEAN 34, 64
CODE 34, 64
Coefficient
  and the PEARSON function  201
  of depreciation  42
  of imaginary number portion of complex number  133
  of the real portion of a complex number  140
COLUMN 28, 65
COLUMNS 28, 65
COMBIN vii, 66
Combinations, computing the number of possible  203
Commas, using in arguments  4
Common logarithm, computing 138
Comparison, symbols used in  12
Complementary
  error function, computing  106
  F-distribution, computing  110
  student’s T-distribution, computing  246
COMPLEX vii, 23, 67
Complex
  common logarithm, computing  138
  conjugate, computing  135
  cosine, computing  135
  exponential, computing  136
  logarithm base-2, computing  138
  natural logarithm, computing  137
  numbers 67, 133–143
    dividing  136
  numbers, computing real power of  139
Compounding, computing effects of  104
CONCATENATE 34, 67
Concatenation
  operator  11
  symbol for text  12
  using text  4
CONFIDENCE vii, 31, 68
Confidence interval, computing  68
Conjugate, computing complex  135
Constants  9
CONVERT vii, 23, 68
Converting
  among number systems  53
  arabic numerals to roman 218
  arabic numerals to roman numerals 220
  CALL functions to Java add-in functions  57
  columns to rows  249
  date to serial number  249
  date to year  266
  decimal dollar to fractional  98
  fractional dollars to decimals  98
  numbers to text  247
  numbers to text in dollars  256
  period between dates to fraction of year  266
  rows to columns  249
  text to numbers  257
  time to a serial number  247
Coordinates  46
CORREL vii, 31, 72
Correlation coefficient
  and the PEARSON function  201
  computing  72
  computing the square of the  224
COS 29, 73
COSH 29, 74
Cosine
  arc  39
  computing complex  135
  hyperbolic  74
  inverse hyperbolic  40
COUNT 31, 74
COUNTA 31, 75
COUNTBLANK vii, 27, 75
COUNTIF 29, 76
Counting formulas 74–76
COUPDAYBS vii, 25, 76
COUPDAYS vii, 25, 77
COUPDAYSNC vii, 25, 78
COUPNCD vii, 25, 78
COUPNUM vii, 25, 79
Coupon dates 37
  computing 76–80
  period length and security price 194, 197
  period length and security yield 196, 198
COUPPCD vii, 25, 80
COVAR vii, 31, 80
Covariance, computing 80
CRITBINOM vii, 31, 81
Criteria argument in database functions 16–18
CUMIPMT vii, 25, 82
CUMPRINC vii, 25, 83
Cumulative beta distribution, computing 51
  interest, computing 82
  lognormal distribution for x, computing 169
  principal paid, computing 83
Curly brackets, in array functions 14
Currency, converting to local 97

D
Data entry
  formulas and 19
Database
  argument in database functions 15
  example 14
  list of 22
DATE 22, 83
Date
  and database searches 17
  and time functions
    list of 22
  and time functions, list of 22
  and time, entering in arguments 3
  associated with securities, computing 105
  computing 263
  computing coupons 76–80
  computing dates associated with securities 104
  computing day of the week 261
  computing fraction of year 266
  computing today’s 249
  computing week number of the year 262
Date (continued)
  computing year of 266
  functions 83–87
  returning 190
DATEDIF 22, 84
DATEVALUE 22, 85
DAVERAGE vii, 14, 15, 22, 85
DAY 22, 86
Day of the week, computing 261
Days, computing working 186
DAY360 23, 86
  used in AMORDEGRC 42
  used in AMORLINC 43
DB 25, 87
DCOUNT vii, 14, 22, 88
DCOUNTA vii, 14, 22, 89
DDB 25, 89
DEC2BIN vii, 23, 90
DEC2BIN, function 90
DEC2HEX vii, 24, 91
DEC2HEX, function 91
DEC2OCT vii, 24, 91
DEC2OCT, function 91
Decimal numbers
  converting from 90–92
  converting to 53, 129, 193
  symbols in function arguments 12
DEGREES vii, 29, 92
Degrees
  converted from radians 92
  converting from 215
DELT A vii, 24, 93, 123
Depreciation 89
  coefficient 42
  computing 228
  computing using French tax rules 41, 43
  of an asset 259
  of an asset using sum-of-years method 242
Determinant, computing 174
Deviation, calculating average 47
DEVSQ vii, 31, 93
DGET vii, 14, 22, 94
DISC vii, 25, 95
Discounted rate, computing 95
Displaying formulas in cells 10
Distance abbreviations and conversion units 69
Distribution, normal, computing 189
#DIV/0! error message 19
Division
  by zero error 19
  operator 11
  symbol 12
DMAX vii, 14, 22, 96
DMIN vii, 14, 22, 96
Documentation v
DOLLAR 34, 97
Dollar
  converting fractional to decimal 98
  sign 6
  using U.S. 97
DOLLARDE vii, 25, 98
DOLLARFR vii, 25, 98
Double-declining balance method 89
Double-factorial 110
DPRODUCT vii, 14, 22, 99
DSTDEV vii, 14, 22, 99
DSTDEVP vii, 14, 22, 100
DSUM vii, 14, 22, 101
DURATION vii, 25, 101
dVAR vii, 14, 22, 103
dVARP vii, 14, 22, 103

Excel (continued)
  and the COVAR function 81
  and the GETPIVOTDATA function 123
  and the HYPRELINK function 131
  and the PEARSON function 201
  and the SQLQUERY function 230
  and units of length 69
  evaluation rules 13
Exclamation point 8
EXP 29, 108
EXPONDIST vii, 31, 109
Exponential
  distribution, computing 109
  regression, computing multiple 165
Exponentials
  computing complex 136
Exponentiation
  operator 11
  symbol 12
Exponents, entering in arguments 3
Expressions 9
External references 8

FACT 29, 109
FACTDOUBLE vii, 29, 110
Factorial 109
FALSE 28, 110
FDIST vii, 31, 110
F-distribution, computing 110
Field argument in database function 15
Financial functions 18
  list of 25
FIND 34, 111
FIXED vii, 29, 113
FIXED v 34, 113
FLOOR 29, 114
Force abbreviations and conversion units 70
FORECAST vii, 31, 114
Formula evaluation rules 13
Formula One for Java, technical support for vi
Formulas
  cell references in 4–8
  complex 11
  creating with functions 11
  displaying in cells 10
  evaluation rules 13
Fractions, entering in arguments 3
French accounting system, computing depreciation for 43
FREQUENCY vii, 13, 31, 115
FTEST vii, 31, 117
F-test probability, computing 117
Function Reference, about v
Functions
and arguments 3
arguments
array constants in 9
cell references in 4
error values in 19
logical values in 4
numbers in 3
range references in 4
array 13
A-Z reference 37–270
creating formulas with 11
database 14
defined 2
displaying formulas in cells 11
elements 2
entering 2
equal signs in 2
financial 18
keywords in 2
listed by category 21–35
maximum number of characters in 2
multiple nested 17
nesting 11
reference v
Future value, computing 118–119
FV 25, 118
FVSCHEDULE vii, 25, 119

G
Gamma distribution, computing 120
GAMMADIST vii, 31, 120
GAMMAINV vii, 31, 120
GAMMALN vii, 31, 121
GCD vii, 29, 121
GEOMEAN vii, 31, 122
Geometric mean, computing 122
GESTEP vii, 24, 123
GETPIVOTDATA vii, 123
Greater than symbol 12
Greatest common divisor, computing 121
GROWTH vii, 13, 32, 124
Guides
Function Reference v
Technical v
User’s v

H
HARMEAN vii, 32, 127
Harmonic mean, computing 127
HEX2BIN vii, 24, 128
HEX2BIN, function 128
HEX2DEC vii, 24, 129
HEX2OCT vii, 24, 130
HEX2OCT, function 130
Hexadecimal numbers
converting from 128–130
converting to 54, 193
HLOOKUP 28, 127
HOUR 23, 130
Hyperbolic
sine, computing 227
tangent, computing 244
Hypergeometric distribution, computing 131
HYPERLINK vii, 28, 131
HYPERGEOMDIST vii, 32, 131

I
IF 28, 132
IMABS vii, 24, 133
IMAGINARY vii, 24, 133
Imaginary numbers 133–143
IMARGUMENT vii, 24, 134
IMCONJUGATE vii, 24, 135
IMCOS vii, 24, 135
IMDIV vii, 24, 136
IMEXP vii, 24, 136
IMLN vii, 24, 137
IMLOG10 vii, 24, 138
IMLOG2 vii, 24, 138
IMPOWER vii, 24, 139
IMPRODUCT vii, 24, 139
IMREAL vii, 24, 140
IMSIN vii, 24, 141
IMSQRT vii, 24, 141
IMSUB vii, 24, 142
IMSUM vii, 24, 143
INDEX (array type) 13, 28, 144
INDEX (non-array type) 28, 143
INDIRECT 28, 145
INFO vii, 27, 146
Information functions, list of 27
INT 29, 147
INTERCEPT vii, 32, 148
Interest 38, 39
cumulative 82
effective annual rate 104
in a security 267–269
Interest (continued)
nominal annual 187
payment 149
payment to 82
rate 148
rate of an annuity per period 217
Internal rate of return, computing 264
International system of measurement 69
INTRATE vii, 25, 148
Inverse
hyperbolic
cosine 40
sine 45
tangent 47
of bell curve, computing the 189
of complementary F-distribution 112
of cumulative gamma distribution 120
of cumulative lognormal distribution, computing 169
of cumulative normal distribution 188
of Fisher’s z-transformation 113
of normal cumulative distribution, computing 189
of square matrix, computing 179
Investments
net present value 191
number of periods of 190
IPMT 25, 149
IRR 25, 150
ISBLANK 27, 151
ISERR 27, 151
ERROR vii, 27, 152
ISEVEN vii, 27, 153
ISLOGICAL vii, 27, 153
ISNA 27, 153
ISNONTEXT 27, 153
ISNUMBER 27, 154
ISODD vii, 27, 154
ISREF 27, 155
ISTEXT 27, 155

J
JDBC support, as replacement for SQLREQUEST 230

K
Keywords in functions 2
KURT vii, 32, 156
Kurtosis, computing 156

L
LARGE vii, 32, 156
LCM vii, 29, 157
Least common multiple, computing 157
Least squares linear regression line, computing slope 229
Least squares regression line, constructing 114
LEFT 34, 158
LEN 34, 158
Less than or equal to symbol 12
Less than symbol 12
Linear regression, multiple, computing 159
LINEST vii, 13, 32, 159
Liquid measure abbreviations and conversion units 71
List
of database functions 22
of date and time functions 22
of engineering functions 23
of financial functions 25
of information functions 27
of logical functions 28
of lookup and reference functions 28
of math and trigonometry functions 29
of operator types 11
of statistical functions 31
of table of operator symbols 12
of text functions 34
LN 29, 163
Loans, computing cumulative interest on 82
LOG 29, 164
LOG10 29, 164
Logarithm 164
base-10 164
base-2 complex 138
common complex 138
natural 163
natural complex 137
of the gamma function 121
LOGEST vii, 13, 32, 165
Logical functions, list of 28
Logical values in function arguments 4
LOGINV vii, 32, 169
Lognormal distribution of x, computing cumulative 169
LOGNORMDIST vii, 32, 169
Long text, detecting with the CELL function 57
LOOKUP 28, 170
Lookup and reference functions, list of 28
Lotus 1-2-3
and AVERAGEA 48
and the CELL function 60
evaluation rules, using 13
LOWER 34, 171
<table>
<thead>
<tr>
<th><strong>M</strong></th>
<th></th>
<th><strong>N</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Macaulay duration, computing 101</td>
<td></td>
<td>N 27, 185</td>
<td></td>
</tr>
<tr>
<td>Magnetism abbreviations and conversion units 70</td>
<td>MATCH 28, 172</td>
<td>#N/A error message 19</td>
<td></td>
</tr>
<tr>
<td>MATCH 28, 172</td>
<td>Math and trigonometry functions, list of 29</td>
<td>NA 27, 185</td>
<td></td>
</tr>
<tr>
<td>Matrix multiplication 181</td>
<td>MAX 32, 173</td>
<td>Name command (Insert menu) 10</td>
<td></td>
</tr>
<tr>
<td>MAXA vii, 32, 173</td>
<td>MAXTERM 174</td>
<td>#NAME? error message 19</td>
<td></td>
</tr>
<tr>
<td>MDETERM vii, 32, 174</td>
<td>MDURATION vii, 26, 175</td>
<td>Names, defining 9</td>
<td></td>
</tr>
<tr>
<td>MDETERM 174</td>
<td>MEDIAN vii, 32, 176</td>
<td>Natural logarithm, computing 137, 163</td>
<td></td>
</tr>
<tr>
<td>MDURATION vii, 26, 175</td>
<td>Method, setGroup 8</td>
<td>Negation symbol 12</td>
<td></td>
</tr>
<tr>
<td>MIN 32, 177</td>
<td>MINVERSE vii, 32, 178</td>
<td>Negative binomial distribution, computing 185</td>
<td></td>
</tr>
<tr>
<td>MINA vii, 32, 178</td>
<td>MINUTE 23, 178</td>
<td>Negative numbers, entering in arguments 3</td>
<td></td>
</tr>
<tr>
<td>MIR 180</td>
<td>MIXED vii, 26, 29, 179</td>
<td>NEGBINOMDIST vii, 32, 185</td>
<td></td>
</tr>
<tr>
<td>Mixed references 8</td>
<td>MMULT vii, 13, 29, 181</td>
<td>Nesting functions 11</td>
<td></td>
</tr>
<tr>
<td>MOD 29, 182</td>
<td>MODE vii, 32, 182</td>
<td>Net present value, computing 191, 265</td>
<td>NETWORKDAYS vii, 23, 186</td>
</tr>
<tr>
<td>MODE: vii, 32, 182</td>
<td>Modified internal rate of return 180</td>
<td></td>
<td>NIST Guide for the Use of the International System of Units (SI) 69</td>
</tr>
<tr>
<td>MONTH 23, 183</td>
<td>Most frequent value, computing 182</td>
<td>OCT2BIN vii, 24, 192</td>
<td>NOMINAL vii, 26, 187</td>
</tr>
<tr>
<td>Mouse, entering cell references with 5</td>
<td>MULTINOMIAL vii, 29, 184</td>
<td>OCT2DEC vii, 24, 193</td>
<td>Nominal annual interest, computing 187</td>
</tr>
<tr>
<td>MROUND vii, 29, 183</td>
<td>Multiple criteria in a database search 18</td>
<td>OCT2HEX vii, 24, 193</td>
<td>Nominal interest rate 104</td>
</tr>
<tr>
<td>MULTINOMIAL vii, 29, 184</td>
<td>Multiple exponential regression, computing 165</td>
<td>OCT2HEX vii, 24, 193</td>
<td>Nonprintable characters, CLEANing 64</td>
</tr>
<tr>
<td>Multiple linear regression, computing 159</td>
<td>Multiple nested functions 17</td>
<td>OCT2HEX vii, 24, 193</td>
<td>Normal cumulative distribution, computing 189</td>
</tr>
<tr>
<td>Multiple linear regression, computing 159</td>
<td>Multiplication operator 11</td>
<td>ODD 29, 194</td>
<td>Normal distribution, computing 187</td>
</tr>
<tr>
<td>Multiple nested functions 17</td>
<td>Multiplication symbol 12</td>
<td>ODDFPRICE vii, 26, 194</td>
<td>NORMDIST vii, 32, 187</td>
</tr>
<tr>
<td>Multiplication symbol 12</td>
<td>Multiplying 212</td>
<td>ODDFYIELD vii, 26, 196</td>
<td>NORMINV vii, 32, 188</td>
</tr>
<tr>
<td>Multiplying 212</td>
<td></td>
<td>ODDLYIELD vii, 26, 197</td>
<td>NORMSINV vii, 32, 189</td>
</tr>
<tr>
<td>Operators, table 12</td>
<td></td>
<td>OFFSET 28, 199</td>
<td>NOT 28, 190</td>
</tr>
<tr>
<td>Order of evaluation 12</td>
<td></td>
<td>Operator symbols, table 12</td>
<td>Not equal to symbol 12</td>
</tr>
<tr>
<td>Order of precedence of operators in a formula 12</td>
<td>Paths in External References 9</td>
<td>OR 28, 200</td>
<td>Null hypothesis, testing 112</td>
</tr>
<tr>
<td>Outliers, eliminating using the TRIMMEAN function 253</td>
<td>Parentheses 12</td>
<td>ORDER 28, 200</td>
<td>Null intersection 19</td>
</tr>
<tr>
<td>P</td>
<td></td>
<td>ORDER 28, 200</td>
<td>#NUM! error message 19</td>
</tr>
<tr>
<td>Parentheses 12</td>
<td></td>
<td>OFFSET 28, 199</td>
<td></td>
</tr>
<tr>
<td>Paths in External References 9</td>
<td></td>
<td>Order of evaluation 12</td>
<td></td>
</tr>
<tr>
<td>Payment for principal, computing 83</td>
<td></td>
<td>Order of precedence of operators in a formula 12</td>
<td></td>
</tr>
</tbody>
</table>
Payment to interest, computing 82
PEARSON viii, 32, 201
Percent signs 3
Percentage
operator 11
symbol 12
Percentages, entering in arguments 3
PERCENTILE viii, 32, 202
PERCENTRANK viii, 32
PERMUT viii, 32, 204
Permutations, computing the number of possible 203
PI 29, 204
Pivot table data 123
PMT 26, 205
POISSON viii, 33, 206
Population
standard deviation, computing 100
variance, computing 103
Possibility, calculating 56
POWER viii, 29, 206
Power abbreviations and conversion units 70
PPMT 26, 207
Prefix character abbreviations table 71
Present value of an annuity, computing 213
Pressure abbreviations and conversion units 70
PRICE viii, 26, 208
Price
of a discounted security, computing 209
of a security at maturity, computing 210
of a security, computing 208
PRICEDISC viii, 26, 209
PRICEMAT viii, 26, 210
Principle paid on an annuity, computing 207
PROB viii, 33, 211
Probabilities, computing 56
Probability
and degrees of freedom 248
and negative binomial distribution 185
and the BINOMDIST function 55
and the CHIDINV function 62
and the CHITEST function 62
and the HYPGEOMDIST function 131
and the NEGBINOMDIST function 186
and the POISSON function 205
computing 211
of a z-test, computing 269
of T-distribution 254
PRODUCT 29, 212
Product of complex numbers, computing 139
PROPER 34, 212
Pseudo-periods 38
PV 26, 213

Q
QUARTILE viii, 33, 213
QUOTIENT viii, 30, 214

R
RADIANS viii, 30, 215
Radians
converting to 215
converting to degrees 40, 45, 46, 92
result of arc cosine calculation 40
result of arcsine calculation 45
result of arctangent calculation 46
RAND 30, 215
RANDBETWEEN viii, 30, 216
Random numbers, computing 215–216
Ranges
argument in array functions 13
formula to count areas in 44
in arguments 4
references to 12
results in array functions 13
symbols 12
RANK viii, 33, 203, 216
RATE 26, 217
Rate of return, computing internal 150, 264
Rate of return, modified internal 180
RECEIVED viii, 26, 218
Record
defined 15
#REF! error message 19
References
absolute 5
error 19
external 8
mixed 8
paths in external 9
relative 5
to other workbooks 8
to other worksheets 7
REGISTER.ID viii, 218
Relative cell references 5
Relative row and column reference 41
REPLACE 34, 219
Replacing text using SUBSTITUTE function 235
REPT 34, 219
Results ranges 13
RIGHT 34, 220
ROMAN viii, 30, 220
Roman numerals, converting to 220
ROUND 30, 221
ROUNDDOWN 30, 222
Rounding
  down 222
  to a specified decimal place 221
  to a supplied precision 113
  to an even integer multiple of a factor 183
  to nearest integer 147
  to nearest multiple 114
  to nearest odd integer 194
  up 107, 222
ROUNDUP 30, 222
ROW 28, 223
ROWS 28, 223
RSQ viii, 33, 224

S
Scientific notation, entering in arguments 3
SEARCH 34, 224
Search for text strings 111
SECOND 23, 225
Securities
  computing accrued interest for 37
  computing amount received 218
  computing dates associated with 105
  computing discount rate 95
  computing discounted price 209
  computing Macaulay duration 101
  computing modified duration 175
  computing price 208
  computing price at maturity 210
  computing price based on period length 194, 197
  computing yield 267–269
  computing yield based on period length 196, 198
SERIESSUM viii, 30, 226
setGroup method 8
Settlement date 37
SIGN 30, 226
Simultaneous equations, solving with MINVERSE and MMULT 179
SIN 30, 227
Sine
  calculating the inverse hyperbolic 45
  computing 227
  computing complex of complex number 141
  computing hyperbolic 227
  used in arcsine calculation 45
SINH 30, 227
SKEW viii, 33, 228
SLN 26, 228
SLOPE viii, 33, 229
Slope, computing in least squares linear regression line 229
SMALL viii, 33, 230
Spaces, removing using the TRIM function 252

Spreadsheets v
SQLREQUEST viii, 230
SQRT 30, 231
SQRTPI viii, 24, 30, 231
Square of the correlation coefficient, computing 224
Square root
  computing 231
  computing complex number 141
Standard deviation, computing 99, 232–234
Standard error, computing 234
STANDARDIZE viii, 33, 232
Statistical functions, list of 31
STDEV 33, 232
STDEVA viii, 33, 233
STDEVPA viii, 33, 234
STEYX viii, 33, 234
SUBSTITUTE 34, 235
SUBTOTAL viii, 30, 236
Subtracting complex numbers 142
Subtraction
  operator 11
  symbol 12
SUM 30, 237
SUM functions 237–242
SUMIF 30, 238
Sum-of-years method 242
SUMPRODUCT viii, 30, 238
SUMSQ 30, 239
SUMX2MY2 viii, 30, 240, 241
SUMX2PY2 viii, 30
SUMXMY2 viii, 30
Survey system of measurement 69
SYD 26, 242
Syntax error, in entering functions 3
Syntax errors
  list of error messages 19

T
T 34, 243
TAN 30, 243
Tangent 46, 243
  hyperbolic 244
  inverse hyperbolic 47
TANH 30, 244
T-bill functions 244–246
TBILLEQ viii, 26, 244
TBILLEQ function 244
TBILLPRIICE viii, 26, 245
TBILLPRIICE function 245
TBILLYIELD viii, 26, 245
TBILLYIELD function 245
TDIST viii, 33, 246
T-distribution, computing 246, 254
Technical
documents vi
support vi
Temperature abbreviations and conversion units 71
TEXT 34, 247
Text
changing character case 171
changing to uppercase 256
formatting 212
functions, list of 34
joining using the CONCATENATE function 67
removing all spaces 252
Theta, computing as an angle in radians 134
TIME 23, 247
Time
abbreviations and conversion units 70
computing as serial number 247
computing day of the week 261
computing fraction of year 266
computing week number of the year 262
computing year of date 266
returning 130, 190
seconds computed corresponding to date 225
TIMEVALUE 23, 248
TINV viii, 33, 248
TODAY 23, 249
TRANSPOSE viii, 13, 28, 249
Treasury bill, computing bond-equivalent yield 244
TREND viii, 13, 33, 250
TRIM 34, 252
TRIMMEAN viii, 33, 253
TRUE 28, 253
TRUNC 30, 254
TTEST viii, 33, 254
Two-tailed probability of a z-test, computing 269
Two-tailed test 248
TYPE 27, 255
Type markers 10

U
Unicode values
and the CODE function 64
in CHAR function 61
Union symbol 12
UPPER 35, 256
USDOLLAR 35, 256

V
VALUE 35, 257
#VALUE! error message 19
VAR 33, 257
VARA viii, 33, 258
Variance, computing 102, 257–259
VARP 33, 258
VARPA viii, 34, 259
VDB 27, 259
View options 10
VLOOKUP 28, 260

W
Week number, computing 262
WEEKNUM viii, 262
WEIBULL viii, 34, 262
Weibull distribution, computing 262
Weight and mass abbreviations and conversion units 69
WORKDAY viii, 23, 263
Worksheet functions
A-Z reference 37–270
Worksheets 19
defining names in 9

X
XIRR viii, 27, 264
XNPV viii, 27, 265

Y
YEAR 23, 266
YEARFRAC viii, 23, 266
used in AMORDEGRC 42
used in AMORLINC 43
YIELD viii, 27, 267
YIELDDISC viii, 27, 268
YIELDMAT viii, 27, 268
Y-intercept, computing 148

Z
ZTEST viii, 34, 269
Z-transformation, Fisher’s, computing 112