

Copyright

©New Cloud Technologies Ltd., 2016 - 2025

This text (hereinafter, the "Material") is informational documentation on MyOffice software, licensed under the terms of <https://myoffice.ru/en/eula/> or a later version. The rightholder of the Material is New Cloud Technologies Ltd.

You can not use the contents of the file in any way without New Cloud Technologies, Ltd. written permission. To obtain such a permit, you should contact New Cloud Technologies, Ltd. at <http://ncloudtech.com/contact.html>.

Limitation of liability

The information provided here and now by the representatives of New Cloud Technologies Ltd. cannot be interpreted as a commitment or promise to develop, implement or deliver any software product, feature or capability.

A commitment or promise to develop, implement, or deliver any software product, feature, or capability is only valid if it is confirmed by the respective contract.

The upcoming software products' release date, feature set, means of distribution, and the suggested price may differ from what is discussed here and now.

The plans presented are based on what we know today, and this may change due to the expansion of our abilities or any unforeseen changes in the market.

New Cloud Technologies Ltd. reserves the right to change the development plans at any time and without notifying anyone of such changes.

All product names, logos, and trademarks mentioned in this document belong to their respective owners.

The trademark "MyOffice" belongs to New Cloud Technologies Ltd.

All product names, logos, and trademarks referred to in this document shall belong to their owners and have been used here in accordance with the provisions of applicable laws.

Nothing in these terms shall be construed under any circumstances as direct or indirect provision of a license or right to use the trademarks, logos or service marks used herein. Any unauthorized use of these trademarks, logos or service marks without the written permission of their owner is strictly prohibited.

Contents

Copyright	1
Limitation of liability	2
I General functionality	4
1 Understanding formulas	5
1.1 How to build formulas	6
1.1.1 Formulas versus strings	6
1.1.2 Formula elements	6
1.2 How to supply values to formulas	8
1.2.1 Constants	8
1.2.2 References	9
1.3 How to do basic operations	12
1.3.1 Arithmetic operators	12
1.3.2 Comparison operators	13
1.3.3 Text operators	13
1.3.4 Reference operators	14
1.4 Value types	15
1.4.1 String	15
1.4.2 Number	15
1.4.3 Boolean	16
1.4.4 Empty	17
1.4.5 Error	18
1.5 Functions	20

1.5.1	Arguments	20
1.5.2	Supported value types	21
1.5.3	Omitted arguments	22
1.6	Complex numbers	24
1.7	Encoding	26
1.7.1	DBCS functions	26
2	Working with cells and ranges	27
2.1	Introduction	28
2.2	Types of references	28
2.2.1	R1C1 style	30
2.3	Explicit intersection	33
2.4	Implicit intersection	36
2.5	Array formulas	39
2.5.1	Resulting modified arrays	40
2.6	Data tables	41
2.6.1	One-variable data tables	41
2.6.2	Two-variable data tables	42
2.6.3	Data tables in our editor	43
2.7	Database functions	44
3	Cell formatting	48
3.1	Introduction to number formatting	49
3.1.1	Predefined formats	49
3.2	Custom formatting	51
3.2.1	Number tokens	51
3.2.2	Date and time tokens	53
3.2.3	Additional tokens	57
3.2.4	Formats with multiple rules	58
3.3	Autodetect mechanism	60
3.3.1	Detectable formats	61

3.3.2	Autodetect mechanism in formulas	65
3.3.3	Autodetect mechanism in functions	65
4	Localization	67
4.1	Introduction to localization	68
4.2	How the editor chooses locales	70
4.2.1	Replacing the locale based on the system language	70
4.2.2	Replacing the locale based on the system region	72
4.2.3	Partially supported locales	73
4.2.4	Supported locales and replaceable languages and regions	73
5	Pivot tables	76
5.1	Pivot table overview	77
5.1.1	Creating and editing a report	77
5.1.2	Filtration	78
5.1.3	Changing the aggregate function	78
5.1.4	Load only features	79
5.2	Create and edit pivot tables	81
5.2.1	Preparing your data	81
5.2.2	Inserting a pivot table	81
5.2.3	Editing	83
5.2.4	Updating and deleting pivot tables	86
5.3	Pivot grouping	88
5.3.1	Grouped text items	88
5.3.2	Grouped dates	89
5.3.3	Grouped numbers	90
5.4	Filtering data in a pivot table	92
5.4.1	Applying filters to Rows and Columns fields	92
5.4.2	Adding fields as Filters	93
5.5	Aggregate functions	95
5.6	Additional calculations	96

5.7	Custom formulas in pivot tables	97
5.7.1	Calculated fields	97
5.8	Pivot table options	100
5.9	Number formatting in pivot tables	104
5.9.1	Automatic formatting based on your data source	104
5.9.2	Number formatting per field	106

II Description of functions 107

6 Mathematical functions 108

6.1	ABS	109
6.2	ACOS	111
6.3	ACOSH	113
6.4	ACOT	114
6.5	ASIN	116
6.6	ATAN	118
6.7	ATAN2	120
6.8	ATANH	122
6.9	COSH	123
6.10	COTH	124
6.11	CSC	126
6.12	CSCH	128
6.13	DEGREES	130
6.14	EXP	132
6.15	FACTDOUBLE	134
6.16	INT	136
6.17	LN	138
6.18	LOG	140
6.19	LOG10	142
6.20	PI	143

6.21	POWER	144
6.22	PRODUCT	146
6.23	QUOTIENT	149
6.24	RAND	151
6.25	ROUND	153
6.26	ROUNDDOWN	155
6.27	ROUNDUP	157
6.28	SEC	159
6.29	SECH	161
6.30	SINH	163
6.31	SQRT	164
6.32	SQRTPI	166
6.33	SUBTOTAL	167
6.34	SUM	171
6.35	SUMIF	174
6.36	SUMIFS	178
6.37	SUMPRODUCT	181
6.38	SUMSQ	183
6.39	TANH	185
7	Database functions	186
7.1	DSUM	187
8	Logical functions	192
8.1	AND	193
8.2	FALSE	196
8.3	IF	197
8.4	IFERROR	200
8.5	IFNA	203
8.6	OR	206
8.7	SWITCH	209

8.8	TRUE	212
9	Date and time functions	213
9.1	DATE	214
9.2	DATEVALUE	217
9.3	DAY	219
9.4	EDATE	222
9.5	EOMONTH	225
9.6	HOUR	227
9.7	ISOWEEKNUM	229
9.8	MINUTE	232
9.9	MONTH	234
9.10	NOW	236
9.11	TODAY	238
9.12	YEAR	240
10	Statistical functions	242
10.1	AVERAGE	243
10.2	COUNT	245
10.3	COUNTA	247
10.4	COUNTBLANK	249
10.5	COUNTIF	251
10.6	COUNTIFS	255
10.7	F.DIST.RT	259
10.8	F.DIST	261
10.9	FISHER	263
10.10	LARGE	265
10.11	LINEST	267
10.12	MAX	270
10.13	MIN	272
10.14	NORM.DIST	274

10.15	RANK.EQ	276
10.16	SMALL	278
10.17	STDEV.S	280
10.18	VAR	282
10.19	VARA	284
10.20	VARPA	286
11	Financial functions	288
11.1	COUPPCD	289
11.2	DOLLARDE	292
11.3	DOLLARFR	294
11.4	IRR	296
11.5	NPV	298
11.6	PMT	300
11.7	PV	303
12	Text functions	306
12.1	ASC	307
12.2	CLEAN	309
12.3	CODE	311
12.4	CONCATENATE	313
12.5	DBCS (JIS)	316
12.6	EXACT	318
12.7	FIND (FINDB)	321
12.8	LEFT (LEFTB)	324
12.9	LEN (LENB)	326
12.10	LOWER	329
12.11	MID (MIDB)	331
12.12	PROPER	334
12.13	RIGHT(RIGHTB)	336
12.14	SEARCH (SEARCHB)	338

12.15	SUBSTITUTE	341
12.16	TEXT	344
12.17	TRIM	346
12.18	UNICODE	348
12.19	UPPER	350
13	Information functions	352
13.1	CELL	353
13.2	INFO	355
13.3	ISBLANK	356
13.4	ISERR	358
13.5	ISERROR	360
13.6	ISEVEN	361
13.7	ISLOGICAL	363
13.8	ISNA	364
13.9	ISNUMBER	365
13.10	ISODD	367
13.11	ISREF	369
13.12	ISTEXT	370
13.13	NA	372
14	Reference functions	373
14.1	ADDRESS	374
14.2	AREAS	377
14.3	CHOOSE	379
14.4	COLUMN	381
14.5	COLUMNS	383
14.6	HLOOKUP	384
14.7	HYPERLINK	388
14.8	INDEX	390
14.9	INDIRECT	394

14.10	LOOKUP	396
14.11	MATCH	400
14.12	OFFSET	403
14.13	ROW	406
14.14	ROWS	408
14.15	VLOOKUP	409
15	Engineering functions	412
15.1	COMPLEX	413
15.2	IMABS	416
A	Alphabetic index	418

Part I

General functionality

Chapter 1

Understanding formulas

How to build formulas

Related topics: Value types, How to supply values to formulas, How to do basic operations, Functions

Formulas are expressions that perform operations on values. You can use formulas to fill tables with automatically calculated data.

Formulas versus strings

Tip

To indicate that you want to use a formula, ensure that you start your expression with an equal sign (=). Otherwise, your expression is treated as a string.

=1+1

	A	B	C
1	2		
2			

The formula is entered into A1.

1+1

	A	B	C
1	1+1		
2			

The string is entered into A1.

Formula elements

You can build formulas with constants, references, operators, and functions.



- Constants are values that you type directly into a formula. For more information, see [Constants](#).
- References are addresses of cells or ranges with values. For more information, see [References](#).
- Operators are used to perform basic operations, such as subtraction and division. For more information, see [How to do basic operations](#).
- Functions are used to perform complicated data manipulation, such as calculating subtotals and extracting specific substrings from strings. For more information, see [Functions](#).

Note

A valid formula contains at least one value (a constant or a reference) and does not end with an operator other than the percent sign (%), for example, **=17%**.

How to supply values to formulas

Related topics: Value types, How to build formulas, How to do basic operations, Functions

You can supply a value to a formula as a constant or through a reference. Multiple values can be supplied as an array of constants or using a reference to a range of cells.

Constants

Supplying a constant means typing a value directly into a formula.

```
=10-1
```

	A	B	C
1	9		
2			

Tip

When you type text directly into a formula, enclose the text in double quotations marks to indicate that you do not refer to a function or a named range, for example, `=text`.

Multiple constants can form an array.

```
={1, 2; 3, 4}
```

A cell can display only one value, so if you enter the formula above into A1, only the first array element (1) will be displayed.

	A	B	C
1	1		
2			

To display an array correctly, you need an array formula.

```
{={1, 2; 3, 4}}
```


	A	B	C	D
1	1	2		
2	3	4		
3				

Tip

To learn more, see [Array formulas](#).

References

A reference is an address of a cell or a range that can contain constants, references, or formulas. Our editor supports two ways of presenting references: A1 and R1C1. The A1 style is the default one and is presented in this article.

A cell address is specified with a column name and a row number.

=B2+C2

	A	B	C	D
1		April	May	Total
2	Socks	200	100	300
3				

Tip

Use references to make your formulas more flexible and to place all your data on the sheet.

A range address is composed of two cell addresses separated by a colon. The first cell address refers to the top-left corner of a selection, and the second one refers to the bottom-right corner.

=SUM(A1:B2)

	A	B	C	D
1	1	1		
2	1	1		4
3				

The formula in D2 sums the values from the A1: B2 range.

You can also refer to a named range.

```
=SUM(MYRANGE)
```

Note

Currently, you cannot create named ranges, but you can import them from other editors.

By default, all references refer to cells from a current sheet. You can specify another sheet by entering its name with an exclamation mark at the beginning of a reference.

```
=Sheet2!A1
```

The same applies to regular and named ranges.

```
=SUM(Sheet2!A1:B5)
```

```
=SUM(Sheet2!MYRANGE)
```

Tip

A named range can relate to a specific sheet or to a whole document. If a named range belongs to a specific sheet, specify the named range with a sheet name on all the other sheets.

You can specify a range that spans multiple sheets.

```
=SUM(Sheet1:Sheet2!A2:B2)
```

	A	B	C
1	Sheet 1		
2	1	1	
3			

	A	B	C
1	Sheet 2		
2	1	1	4
3			

Also, for the result of the example shown above, you can use an alternative notation.

```
=SUM(Sheet1!A2:Sheet2!B2)
```

Multiple references can form a reference list. To make a reference list, you need to

enter references one by one and separate them with a comma.

```
=SUM(A1, B2:C3, MYRANGE)
```

Tip

If you enter a reference list directly into the formula, enclose the list in parentheses to indicate that it is a single argument.

How to do basic operations

Related topics: Value types, How to build formulas, How to supply values to formulas, Functions

Basic operations are performed with operators subdivided into:

- Arithmetic operators
- Comparative operators
- Text operators
- Reference operators

Arithmetic operators

Arithmetic operators are mostly used with numbers.

Name	Symbol	Example	Result
Unary plus	+	=+1	1
Unary minus	-	=-1	-1
Percent	%	=1%	0.01
Binary plus	+	=1+1	2
Binary minus	-	=1-1	0
Multiplication	*	=2*3	6
Division	/	=10/5	2
Power	^	=2^2^2	16
Priority	()	=(2+3)*6	30

Note

The caret sign (^) is right-associative. For example, `=3^3^3` means 3^{27} , which returns 7.63E+12.

Comparison operators

Comparison operators are used for data comparison. Formulas with them return Boolean (logical) values—either TRUE or FALSE.

Name	Symbol	Example	Result
Equal	=	<code>=1=1</code>	TRUE
Not equal	<>	<code>=1<>1</code>	FALSE
Less than	<	<code>=1<2</code>	TRUE
Less than or equal	<=	<code>=1<=1</code>	TRUE
Greater than	>	<code>=1>2</code>	FALSE
Greater than or equal	>=	<code>=3>=2</code>	TRUE

Text operators

Currently, there is only one text operator—the concatenation operator (&). It connects two strings into a single one.

Name	Symbol	Example	Result
Concatenation operator	&	<code>= "te"&"xt"</code>	text

Reference operators

Reference operations can be performed only on references and return references as a result.

Using the union operator (:)

The union operator (:) combines two references into a single one. The result of a union is a reference to the smallest possible range that covers all cells referred to in the operation.

```
=COUNT(A1:B2:C3)
```

	A	B	C	D
1	1	1	1	
2	1	1	1	
3	1	1	1	
4				
5	9			

The formula in A5 counts nonempty cells in the specified union A1:C3.

Using the intersection operator

The intersection operator is a space character. The result of an intersection is a reference to cells that are present in each of the supplied references.

```
=A1:B2 B2:C3
```

	A	B	C	D
1	1	1	1	
2	1	0	1	
3	1	1	1	
4				
5	0			

The formula in A5 illustrates the so-called explicit intersection.

Tip

To learn more, see [Explicit intersection](#).

Value types

Related topics: [How to build formulas](#), [How to supply values to formulas](#), [How to do basic operations](#), [Functions](#)

The result of a formula depends on the types of values that it receives. There are five types of values.

- String
- Number
- Boolean
- Empty
- Error

String

A string is a sequence of characters used as data in the editor. Strings can include numbers, letters, special symbols, or signs. By default, text strings are left-aligned in a cell as opposed to numbers, which are aligned to the right. Values of this type are typically used with [text operators](#), [text functions](#), and [custom formatting](#).

	A	B	C	D
1	text			

Note

If you work with multiple spreadsheet editors or legacy documents, remember that our editor uses the UTF-8 encoding for all strings. For more information, see [Encoding](#).

Number

Number is the most common value type in the editor. It is used with [arithmetic operators](#) and in most functions.

	A	B	C	D
1	1000000			

Depending on the intended use, you can format numbers in many ways.

	A	B	C	D
1	Format	Number		
2	General	5.1		
3	Currency	\$5.10		
4	Date	01/04/1900		

Tip

For more information about number formatting, see [Cell formatting](#).

Numbers can be automatically detected by the editor. For example, the number below is supplied to the function as a string but is still detected as a number.

```
=ABS("-7")
```

	A	B	C
1	7		
2			

ABS returns the absolute of -7 correctly.

Tip

For more information, see [Autodetect](#).

Boolean

A Boolean, also known as a logical value, is a data type with only two possible expressions, TRUE or FALSE. The Boolean values are mostly used in [logical functions](#). They are also returned as results of [logical comparisons](#).

	A	B	C	D
1	TRUE	FALSE		

Note

In arithmetic operations, the Boolean values TRUE and FALSE are converted to 1 and 0 (zero) correspondingly. For example, **=TRUE+FALSE** returns 1.

Empty

An empty value is a missing value, and it is never returned as the result of a formula. It can only be passed to a formula in the following scenarios.

- **Through a reference to an empty cell.**

=A2

	A	B	C	D
1	Data	Result		
2		0		
3				

The formula returns 0 (zero) because it retrieves the contents of A2, which is an empty value.

- **Through an unspecified array element.**

{={ , 1; 2, 3}}

	A	B	C	D
1	0	1		
2	2	3		
3				

The array formula retrieves an empty value supplied in the first position and replaces the value with 0 (zero).

- **Through an unspecified function argument.**

=SIN()

	A	B	C	D
1	0			
2				
3				

SIN replaces the empty value in the argument with 0 (zero) and returns its sine.

Mostly, formulas convert empty values into zeros or zero-length strings ("").

Tip

Some functions treat empty values in a special way. For example, [PRODUCT](#) can ignore empty values. For more information, see [Empty arguments](#).

Error

Errors are mostly returned as a result of invalid expressions. Some functions, such as [ISERROR](#), accept errors as arguments and can be used to manage errors. Below, you can see the full list of errors.

Error	Common cause
#NULL!	The range specified by the formula is not found.
#DIV/0!	The formula contains division by zero.
#VALUE!	The argument type is invalid, or the number of arguments is incorrect.
#REF!	The formula contains an invalid reference.
#NAME?	A named range or a function does not exist.
#NUM!	The supplied number is invalid.
#N/A	A referenced value is not found.
#GETTING_DATA	An external reference is not valid, or the calculations are not yet processed.

Note

In most cases, formulas return as the result the first error encountered.

=#DIV/0!+1

	A	B	C	D
1	#DIV/0!			
2				
3				

This formula returns #DIV/0!, the division by zero error, even though there is no actual division performed.

Functions

Related topics: How to build formulas, Value types, How to supply values to formulas, How to do basic operations

A function is a predefined expression with a name. The name usually indicates what the function does. For example, the following function returns a square root.

```
=SQRT(9)
```

	A	B	C
1	3		
2			

All functions can be subdivided into several functional groups depending on the tasks that they perform. For example, there are mathematical functions, text functions, and statistical functions among others. Functions of the same group usually have similar values type restrictions and work in a similar way.

Arguments

All functions have one or more arguments.

Tip

Ensure that you separate multiple arguments with a comma.

```
=SUM(1, 2, 3)
```

	A	B	C
1	6		
2			

Arguments can be required or optional. For example, you can enter **SUM** without any arguments at all (actually, with one empty argument), and it will still be a valid function.

```
=SUM()
```

	A	B	C
1	0		
2			

Other functions return errors if you omit their arguments.

=FIND()

	A	B	C
1	#VALUE!		
2			

Here, **FIND** returns an error because its first two arguments are required and cannot, therefore, be omitted.

Supported value types

Functions work with all types of values: numbers, strings, Booleans, empty values, and errors. You can supply values to functions as constants, through references, or using other functions, which is called nesting.

=PROPER(A1)

	A	B	C
1	john	John	
2			

=SQRT(RAND())

	A	B	C
1	0.407		
2			

Depending on the group to which a function belongs, it can have certain restrictions. For example, mathematical functions mostly work with numbers.

Sometimes, restrictions are specific to an individual function or even an argument. For example, **SQRT**, being a mathematical function, neither works with text nor with negative numbers.

=SQRT(-1)

	A	B	C
1	#NUM!		
2			

The value type of a function result is usually predefined for each function. For example, **SQRT** always returns a number (unless an error occurs).

=SQRT(A1)

	A	B	C
1	TRUE	1	
2			

Here, **SQRT** receives the Boolean value TRUE, which is converted to 1. The function then returns a number.

Omitted arguments

When you omit an argument, the function actually takes an empty value, which is then converted to 0 (zero) or a zero-length string for further calculations. However, some functions treat omitted arguments in a special way. For example, **PRODUCT** can ignore them.

=PRODUCT(1, , 2)

	A	B	C
1	2		
2			

Omitted optional arguments are often converted to their own predefined values, which vary depending on the function. For example, **ADDRESS** returns a cell address based on the specified row and column numbers. The third argument of the function is optional and specifies if the resulting address should be absolute or relative.

=ADDRESS(1, 2)

	A	B	C
1	\$B\$1		
2			

Here, the third argument is omitted, but **ADDRESS** does not convert it to 0 (zero) or a zero-length string. Instead, the function uses the default assumption that an absolute address (with dollar symbols) should be returned.

Complex numbers

Related topics: Constants, Values, Engineering functions

A complex number is the sum of a real and an imaginary number.

Tip

In our editor, complex numbers are actually strings. To perform mathematical operations with them, use [engineering functions](#), which can be easily recognized by the prefix **IM-**.

Syntax

a+bi

a+bj

a and **b**

Any real numbers.

i or **j**

The imaginary unit that satisfies the condition $i^2 = -1$.

Tip

Because complex numbers are actually strings, you can receive them with concatenation, for example, `"2" & "+" & "5i"`. Alternatively, use **COMPLEX**, a dedicated function for this purpose.

Examples

Using engineering functions

Each engineering function performs the same operation as its real number counterpart. For example, the same as **SUM**, **IMSUM** sums the supplied arguments.

=IMSUM(A2:A3)

	A	B	C	D
1				
2	5+5i			
3	6-3i		Result	
4			11+2i	
5				

The formula in C4 sums the values and returns 11+2i.

Using complex numbers with different notations

If an engineering function accepts multiple complex numbers, they can be in different notations. In such a case, the resulting value uses the **i** notation.

```
=IMSUM(A2, A3)
```

	A	B	C	D
1				
2	1+1i			
3	-2-2j		Result	
4			-1-1i	
5				

The formula in C4 sums the values in A2 and A3. The number in A2 uses the **i** notation, and the number in A3 uses the **j** notation. The result is in the **i** notation.

Using nonengineering functions

All other functions treat complex numbers as strings.

```
=ABS(A2)
```

	A	B	C	D
1				
2	-14+2i			
3			#VALUE!	
4				

Encoding

To ensure maximum compatibility, all text is stored in the UTF-8 encoding. Implemented in most modern applications and operating systems, it provides support for nearly every language in the world.

DBCS functions

Before the establishment of the Unicode standard, which UTF-8 is part of, languages with multiple unique characters (primarily, Chinese, Japanese, and Korean) used double-byte character sets, or DBCS for short. For backward compatibility, a number of DBSC functions are supported by our editor.

Conversion functions, which are used to convert regular text to and from DBCS, namely **JIS(DBCS)** and **ASC**, return the supplied text without any modification because all text is already stored in the universal encoding, UTF-8. Other functions work exactly as their regular counterparts. For example, **SEARCHB** works in the same way as **SEARCH**, and **FINDB** does not differ from **FIND**.

The supported DBSC functions are:

- **SEARCH (SEARCHB)**
- **FIND (FINDB)**
- **LEFT (LEFTB)**
- **RIGHT(RIGHTB)**
- **MID (MIDB)**
- **LEN (LENB)**
- **DBCS (JIS)**
- **ASC**

Chapter 2

Working with cells and ranges

Introduction

Spreadsheets usually contain large amounts of data, which require special approaches different from the way you work with constants and single cells. In this chapter, you will learn about different types of data sets and how to work with them.

Types of references

Related topics: How to build formulas, Value types

When you move a reference to another cell, it changes according to its new position. This mechanism allows you to apply one formula to multiple cells without editing it. For example, the formula below is copied to other cells in the row and changes accordingly.

	A	B	C	D
1	Data source		Result	Formula
2	10		10\$	=DOLLAR(A2)
3	100			
4	1000			

	A	B	C	D
1	Data source		Result	Formula
2	10		10\$	=DOLLAR(A2)
3	100		100\$	=DOLLAR(A3)
4	1000		1000\$	=DOLLAR(A4)

By default, all references behave in this way because they are relative. A relative reference specifies the location of a cell being referred to in relation to its own position.

	A	B	C	D
1	Data source		Result	Formula
2	10		10\$	=DOLLAR(A2)
3	100		100\$	=DOLLAR(A3)
4	1000		1000\$	=DOLLAR(A4)

Here, **A2** in **=DOLLAR(A2)** refers to the cell in the same row and two columns to the left of its own position.

Note

Relative references change when you copy/paste, cut/paste, or autofill cells.

If you do not want your references to change, you need to make them absolute by using dollar symbols.

	A	B	C	D
1	Data source		Result	Formula
2	10		10\$	=DOLLAR(\$A\$2)
3	100			
4	1000			

	A	B	C	D
1	Data source		Result	Formula
2	10		10\$	=DOLLAR(\$A\$2)
3	100		10\$	=DOLLAR(\$A\$2)
4	1000		10\$	=DOLLAR(\$A\$2)

An absolute reference refers to a fixed point in the table, so it does not change regardless of where you move it.

	A	B	C	D
1	Data source		Result	Formula
2	10		10\$	=DOLLAR(\$A\$2)
3	100		10\$	=DOLLAR(\$A\$2)
4	1000		10\$	=DOLLAR(\$A\$2)

You can also use dollar symbols to lock your references partially. Such references are called mixed. In the example below, the column is locked and does not change.

=DOLLAR(\$A2)

	A	B	C	D
1	Data source		Original reference	
2	10		10\$	
3				Copied reference
4	100			100\$
5	1000			1000\$

R1C1 style

If you mostly use absolute references, consider using the second referencing style in our editor—R1C1. In this style, both rows and columns are labeled by numbers. You can safely switch from one style to the other at any time.

	1	2	3	4
1	R1C1			
2	R2C1			
3	R3C1			
4				

Note

A defined name, which is a name that refers to a single cell, range of cells, constant value, or formula, can coincide with an A1 reference representation if the R1C1 style is chosen, and the other way around. After a style switch, such defined names remain without changes unless edited manually.

By default, references in the R1C1 style are absolute. In the table below, the reference is copied to other cells but continues to refer to the same position in the table.

	1	2	3	4
1	R1C1		R1C1	R1C1
2	R2C1			R1C1
3	R3C1			
4				

A relative R1C1 reference is entered with positive and negative numbers enclosed in

square brackets.

=R[-1]C[-1]

Positive values indicate that the cell referred to is to the bottom (for rows) and to the right (for columns). Negative values indicate that the cell referred to is to the top (for rows) and to the left (for columns).

	1	2	3	4
1	R1C1			
2	R2C1	=R[-1]C[-1]		
3	R3C1			
4				

	1	2	3	4
1	R1C1			
2	R2C1	R1C1		
3	R3C1			
4				

The reference expression **=R[-1]C[-1]** retrieves the value of the cell that is found one row up and one column to the left and returns R1C1.

If a relative R1C1 reference is moved to another cell, the cell reference changes accordingly. In the example below, the reference expression **=R[-1]C[-1]** is copied into the lower cells of the row, and the relative reference adjusts accordingly.

	1	2	3	4
1	R1C1			
2	R2C1	R1C1		
3	R3C1	R2C1		
4		R3C1		

Note

In the R1C1 style, both absolute and relative references always remain unchanged.

To make a mixed R1C1 reference, you need to anchor the row or column part of your reference.

=R1C[-2]

	1	2	3	4
1	R1C1			
2	R2C1		R1C1	
3	R3C1			
4				

The column part is anchored, so the formula refers to the cell in row 1 and two columns to the left. In the current position, the formula returns R1C1.

Explicit intersection

Related topics: Implicit intersection, How to build formulas, Array formulas

Explicit intersection returns shared areas between given references.

Basic usage

=A2:C2 B1:B3

	A	B	C	D
1		Q1	Q2	
2	Lenny	12%	34%	
3	Mick	8%	49%	
4				
5	Lennie's KPI, Q1	=A2:C2 B1:B3		
6				

	A	B	C	D
1		Q1	Q2	
2	Lenny	12%	34%	
3	Mick	8%	49%	
4				
5	Lennie's KPI, Q1	12%		
6				

Warning

If there is no intersection, the #NULL! error is returned. References from different sheets cannot intersect each other.

Syntax

=reference1_reference2_..._referenceN

reference1 ... reference N

Cells, ranges, and reference lists of which you want to find an intersection.

_

The intersection operator.

Examples

1. If the result of your intersection is a range, you cannot display it in a single cell. You need an array formula for such cases:

```
{=A2:C3 B1:B3}
```

	A	B	C	D
1		Q1	Q2	
2	Lenny	12%	34%	
3	Mick	8%	49%	
4				
5	Q1 KPIs	{=A2:C3 B1:B3}		
6				

	A	B	C	D
1		Q1	Q2	
2	Lenny	12%	34%	
3	Mick	8%	49%	
4				
5	Q1 KPIs	12%		
6		8%		

Warning

Currently, you cannot create or edit array formulas. However, you can open documents that already have such formulas inside.

2. Functions such as **SUM** and **MAX** are designed to work with ranges and arrays. You can use these functions to bypass errors when your intersection point is a range:

```
=SUM(A2:D3 D1:D3)
```

	A	B	C	D
1		iPhone	iPad	iMac
2	Lenny	378	492	289
3	Mick	654	549	196
4				
5	Total iMac sales	=SUM(A2:D3 D1:D3)		
6				

	A	B	C	D
1		iPhone	iPad	iMac
2	Lenny	378	492	289
3	Mick	654	549	196
4				
5	Total iMac sales	485		
6				

Implicit intersection

Related topics: Explicit intersection, How to build formulas, Array formulas

A regular formula has only one cell to display the result. So if your formula receives a range and returns multiple results, only one of them can be displayed. Implicit intersection silently selects one value from the given range and gives it to the formula.

Tip

An array formula can have many cells for its results.

Basic Usage

=B1 : B5

This formula is entered into the D2 cell. One cell can not display a range, so this formula is resolved with implicit intersection:

	A	B	C	D
1		Score		
2	John	435		435
3	Paul	973		
4	Ringo	632		
5	George	872		
6				

The cell with the formula implicitly intersects the supplied range in the B2 cell, so the B2 cell is silently selected for calculations instead of the whole range.

Warning

This operation mostly resolves formulas with single-column and single-row ranges. If a formula cannot be resolved, the #VALUE error is returned. A formula cannot be resolved if a shared area is not found or has more than one cell.

Examples

1. Implicit intersection can resolve formulas with single-row ranges:

=A2:E2/\$E\$2*100

This formula is entered into the B5 cell.

	A	B	C	D	E
1		Cats	Dogs	Birds	Total
2	n	10	19	35	64
3					
4					
5	%	15.625	29.6875	54.6875	100
6					

The B5 cell implicitly intersects the A2:E2 range in the B2 cell.

- Implicit intersection resolves formulas with functions:

=SIN(A1:A5)

This formula is entered into the C2 cell.

	A	B	C	D
1	Angle		Sine	
2	0		0.00	
3	30		-0.99	
4	60		-0.30	
5	90		0.89	
6				

The C2 cell implicitly intersects the A1:A5 range in the A2 cell.

Note

Functions such as **SUM** and **MAX** are designed to work with ranges and arrays. They do not need to be resolved with implicit intersection.

- If a formula and a supplied range are on different worksheets, intersection occurs in the three-dimensional space:

=Sheet1!A1:A4

This formula is entered into the B2 cell of Sheet2.

	A	B	C
1	Num		Sheet1
2	0		
3	1		
4	2		

	A	B	C
1			Sheet2
2		0	
3		1	
4		2	

You can even specify multirow or multicolumn ranges:

=Sheet1!A1:A4

This formula is entered into the A2 cell of Sheet2.

	A	B	C
1	Num	Str	Sheet1
2	0	x	
3	1	y	
4	2	z	

	A	B	C
1			Sheet2
2	0	x	
3	1	y	
4	2	z	

Array formulas

Related topics: [How to build formulas](#), [Value types](#), [Implicit intersection](#)

Array formulas work with arrays and ranges and perform the same operation multiple times on each of the specified values. An array formula result can be displayed in multiple cells.

Basic Usage

In most spreadsheets, array formulas are enclosed in curly braces.

```
{=A2:B3+1}
```

	A	B	C	D	E
1	Input			Result	
2	1	2		2	3
3	3	4		4	5
4					

The formula adds 1 to the value of each cell in A2:B3.

You can also supply an array of constants to an array formula.

```
{={1, 2; 3, 4}+1}
```

	A	B	C	D	E
1	Result				
2	2	3			
3	4	5			
4					

Note

Our editor supports only imported array formulas. They cannot be edited or reused in other cells. However, it is possible to edit data in the cells of the supplied ranges and receive valid results. You can also use [Implicit intersection](#) to work with single-row or single-column ranges.

Resulting modified arrays

Some editors require that you specify the output range of an array formula. In tables imported from such editors, an array formula can return an array of a different size than the one produced by its calculation.

Examples

Arrays smaller than those produced by calculations

`{=A2:B3}`

	A	B	C	D	E
1	Input			Result	
2	1	2		1	2
3	3	4			
4					

The result of the formula is a 2x2 array, but the editor truncates the result to fit into the output range (D2:E2).

Arrays larger than those produced by calculations

`{=A2:B2}`

	A	B	C	D	E
1	Input			Result	
2	1	2		1	2
3				1	2
4					

The result of the formula is a 2x1 one-dimensional array, but the editor duplicates the result to fill the 2x2 output range (D2:E3).

Warning

If the resulting range is multidimensional, odd cells are filled with #N/A errors.

Data tables

Related topics: How to build formulas, Functions.

A data table is a tablelike range defined by a single function with one or two variables. Data tables produce multiple resulting values that correspond to multiple input values. Data tables are mainly used in financial forecasting.

Note

Data table cells cannot be edited separately. You can only change the data table function or its arguments to receive new results.

One-variable data tables

One-variable data tables can be column- or row-oriented.

In a column-oriented data table, arguments and results are listed in two columns, with the function found in the cell immediately above the results. For an input variable, the function refers to a cell outside the data table.

=A1/10

	A	B	C	D	E
1	10			—arguments	
2				—results	
3		1		—data table	
4	5	0.5			
5	50	5			
6	100	10			
7					

The example shows a column-oriented data table with its function in B3.

In a row-oriented data table, arguments and results are listed in two rows, with the function found immediately left of the results. For an input variable, the function refers to a cell outside the data table.

=A1/10

	A	B	C	D	E
1	10			—arguments	
2				—results	
3			-----	—data table	
4					
5		5	50	100	
6	1	0.5	5	10	
7					

The example shows a row-oriented data table with its function in A6.

Two-variable data tables

In a two-variable data table, the function is located in the top-left corner, with one group of arguments below it and the other to its right. For two input variables, the function refers to two cells outside the data table.

=A1/B1

	A	B	C	D	E
1	10	5		—arguments	
2				—results	
3			-----	—data table	
4					
5					
6	2	10	20	30	40
7	5	2	4	6	8
8	50	0.2	0.4	0.6	0.8
9	100	0.1	0.2	0.3	0.4
10					

The example shows a two-variable data table with its function in A6.

Data tables in our editor

Note

Our editor supports data tables but does not support data table functions. When you open a document with a data table, all its cells are converted to cells with regular formulas.

Example

=A1*B1

	A	B	C	D	E
1	10	5			
2					
3	50	1	2	3	
4	5	5	10	15	
5	10	10	20	30	
6					

The data table is shown with cells that contain values specified and values calculated. The table below demonstrates the converted formulas that underlie the calculated values of particular cells.

	A	B	C	D	E
1	10	5			
2					
3	=A1*B1	1	2	3	
4	5	=B3*A4	=C3*A4	=D3*A4	
5	10	=B3*A5	=C3*A5	=D3*A5	
6					

All the formulas in the table above are independent. You can change the arguments to receive new results, but changing the formula in A3 will not affect the other formulas inside the table.

Database functions

Related topics: Functions.

Database functions allow you to filter your data before calculating it. All of them have the D-prefix in their names and are similar in syntax. To use them, you need to arrange your data as a database, with headings in the top row and records in the lower rows.

Note

All database functions are based on mathematical functions. For example, **DSUM** is based on **SUM**.

Basic Usage

```
=DSUM(A2:C5, 2, A7:C8)
```

	A	B	C	D
1	Database			
2	Model	Price	Condition	Shipping
3	G1	\$100.00	used	\$5.00
4	G2	\$250.00	new	Free
5	G1	\$300.00	new	Free
6	Filtering conditions			
7	Model	Price	Condition	Shipping
8	G1			
9	Result			
10	400			
11				

This formula applies the filter (A7:C8) to the database (A2:C5) and then sums the filtered-out values from the second database column (B2:B5). The result is the total price of all the G1 models.

Note

If cells for filtering conditions are empty or have only headings, no filtering occurs.

Syntax

```
=DFUNCTION(database, field, criteria)
```

database

A range with headings in the top row and records in the lower rows.

field

The column with values that you want to use in calculations.

criteria

A range with **database** headings in the top row and filtering conditions in the lower rows.

Note

Conditions in one row are joined with **AND** logic. All of them must be true for a record to pass the filter.

	A	B	C	D
1	Model	Price (\$)	Discount	
2	A1	>200	>11%	

Conditions in one column are joined with **OR** logic. At least one of them must be true for a record to pass the filter.

	A	B	C	D
1	Model	Price (\$)	Discount	
2	A1	<200	>11%	
3	A2	>100	<20%	

Examples

Using comparison operators

Note

The database functions support all comparison operators available in the editor.

```
=DSUM(A1:D5, C1, A6:D7)
```

	A	B	C	D	E
1	Name	Engines	Flight Range (km)	Price	Result
2	Boeing	2	6,000.00	\$100,000,000.00	6,000.00
3	Boeing	4	13,000.00	\$300,000,000.00	
4	Airbus	2	700.00	\$110,000,000.00	
5	Airbus	4	13,500.00	\$280,000,000.00	
6	Name	Flight Range (km)	Flight Range (km)	Price	
7	Boeing	>5,000	<8,000	>80,000,000	

DSUM returns the total flight range of Boeing planes with individual flight range more than 5,000 and less than 8,000 km and with price under \$80,000,000. Only the first record meets all the specified conditions, so the result is 6,000.

Using wildcards

Note

Database functions support wildcard characters in conditions with text.

```
=DSUM(A1:D6, D1, A7:D9)
```

	A	B	C	D	E
1	Name	Engines	Flight Range (km)	Price	Result
2	Boeing	2	6,000.00	\$100,000,000.00	\$580,000,000.00
3	Boeing	4	13,000.00	\$300,000,000.00	
4	Airbus	2	700.00	\$110,000,000.00	
5	Airbus	4	13,500.00	\$280,000,000.00	
6	IL	4	13,000.00	\$180,000,000.00	
7	Name	Engines	Flight Range (km)		
8	???ing	>2			
9	Air*		>=13,000		

DSUM returns the total price of planes that pass three filters. The function selects planes whose names are either 6 characters long and end in "ing" or start with "Air". The other filters specify that planes are to have more than 2 engines and a flight range of 13,000 km or more.

Tip

In wildcards, a question mark (?) matches any single character, and an asterisk (*) matches any sequence of characters. To use an actual question mark or asterisk, type a tilde (~) before the character.

Chapter 3

Cell formatting

Introduction to number formatting

Related topics: Value types, Localization.

Every number in our editor can be formatted in multiple ways. For example, in the table below, number 10 is formatted into a date.

	A	B	C
1	1/9/1900		
2			

Regardless of how you format a number, it remains a number, though it looks different. For instance, the date in the previous example actually represents the number of days that passed after 12/30/1899 (the editor's base date) and is equal to 10, which you can check with a simple formula.

	A	B	C
1	1/9/1900		
2	=A1-10		

	A	B	C
1	1/9/1900		
2	12/30/1899		

Note

The editor formats values automatically. For example, if you enter **=10%+1**, the result is formatted as a percentage. For more information, see [Autodetect mechanism](#).

Predefined formats

Together with **Date**, there are 11 predefined number formats in our editor. All of them are designed for numbers, so nonnumeric values do not change when you apply these formats to them.

Name	Description	Example
------	-------------	---------

General	Displays a value without changes.	12.1
Number	Displays a number with at least two figures after the decimal separator.	12.10
Currency	Similar to Number but with a currency symbol.	\$12.10
Accounting	Similar to Currency but displays negative numbers in parentheses without a minus sign.	(\$ 12.10)
Date	Displays a number as a date.	1/11/1900
Time	Displays a number as time.	2:24:00 AM
Date and time	Displays a number as a date with time.	1/11/1900 2:24 AM
Percentage	Displays a number as a percentage (1=100%).	1210.00%
Fraction	Displays a number as a fraction.	12 1/10
Scientific	Displays a number in scientific notation.	1.21E+01
Text	Displays a value as a string.	12.1

Note

All predefined formats can be modified into custom ones with the help of special symbols called tokens. Currently, you cannot create custom formats in our editor, but you can import them from other applications. For more information, see [Custom formatting](#).

Custom formatting

Related topics: Introduction to formatting, Value types

Custom formatting lets you modify predefined formats with the help of special symbols called tokens. By combining tokens, you create formatting rules.

Warning!

Currently, you cannot create formatting rules in our editor. You can only import them from other editors. You can also decorate your numbers with **TEXT**, to which certain limitations apply.

For example, the following custom rule displays numbers in red and without the fractional part.

[Red]#

	A	B	C	D
1	Entered	Displayed		
2	12.9999	12		
3	12	12		
4				

Number tokens

Number tokens allow you to present numbers in different notations, for example, as decimal numbers with an adjustable decimal point or as fractions. Formatting rules with these tokens align formatted values to the right.

Token	Formatting	Example
#	Displays a digit or nothing.	$1 \Rightarrow \# \Rightarrow 1$ $1.11 \Rightarrow \#.\# \Rightarrow 1.1$
	You cannot display fewer digits than those in the integer part of your number (which also applies to . , 0 , ?).	$12 \Rightarrow \# \Rightarrow 12$

	If you display fewer decimal places than those in the original number, the number is rounded (which also applies to . , 0 , ?).	$12.15 \Rightarrow \#.\mathbf{0} \Rightarrow 12.2$
.	Separates the integer part and the fractional part. $12.56 \Rightarrow . \Rightarrow 12.$	
0	Displays a digit or 0 (zero).	$1.1 \Rightarrow \mathbf{00}.\mathbf{00} \Rightarrow 01.10$
?	Displays a digit or a single space.	$1.1 \Rightarrow \mathbf{??}.\mathbf{??} \Rightarrow _1.1_$
,	Separates groups of thousands.	$1000 \Rightarrow \#, \# \Rightarrow 1,000$
/	If placed as the last token, displays the number after division by 1000.	$1111 \Rightarrow \#.\#, \Rightarrow 1.1$

Note

., **#**, **0**, **?**

If your rule contains at least one of the tokens, all digits before the decimal separator will be displayed.

/	Separates the numerator and the denominator of a fraction.	$10.5 \Rightarrow \#/\#, \Rightarrow 21/2$
	The denominator can be specified.	$10.5 \Rightarrow \#/\mathbf{10}, \Rightarrow 105/10$
_	Separates the integer and the fractional parts of a fraction.	$10.5 \Rightarrow \mathbf{?} \mathbf{?}/\mathbf{?} \Rightarrow 10 \frac{1}{2}$
%	Multiplies the specified value by 100 and displays it with a percent sign.	$1 \Rightarrow \#\% \Rightarrow 100\%$
E+	Displays the number in scientific notation.	$1999 \Rightarrow \mathbf{?E+?} \Rightarrow 2E+3$
E-	Displays the number in scientific notation without a plus sign.	$1999 \Rightarrow \mathbf{?E-?} \Rightarrow 2E3$

[\$\$]	Displays the second symbol as the currency symbol.	10 ⇒ [\$\$]# ⇒ \$10 10 ⇒ [\$£]# ⇒ £10
	Can be specified with a country code.	10 ⇒ [\$¥-411]# ⇒ ¥10

Note

Exponent and currency tokens can be used only once in a formatting rule.

Date and time tokens

Custom date/time formats are composed of special tokens, which cannot be used in combination with number tokens. The separators that were described above only visually separate date/time tokens and do not perform any additional operations. Rules with these tokens align formatted values to the right.

Token	Formatting	Example
/	Separates the day, month, and year parts of a date. Can be replaced by a dot (.), a hyphen (-), and other separators.	1/1/1900
d	Displays the day part without a leading 0 (zero).	1
dd	Displays the day part with a leading 0 (zero).	01
ddd	Displays the day of the week in short form.	Mon
dddd	Displays the day of the week in full form.	Monday
m	Displays the month part without a leading 0 (zero).	1

mm	Displays the month part with a leading 0 (zero).	01
mmm	Displays the name of the month in short form.	Jan
mmm	Displays the day of the month in full form.	January
mmmm	Displays the first letter of the month name.	J
yy	Displays the year part of the date in 2 digits.	00
yyyy	Displays the year part of the date in 4 digits.	1900
ww	Displays the week of the date.	1
r	Displays the era in short form.	AD
rr	Displays the era in long form.	Anno Domini
q	Displays the quarter of the date in short form.	Q1
qq	Displays the quarter of the date in long form.	1st quarter

Consider some examples in which some of the tokens above are used together.

	A	B	C	D
1	Entered value	Format	Formatted value	
2	1234.567	m/d/yyyy	5/18/1903	
3	1234.567	q yyyy	Q2 1903	
4	1234.567	yyyy r	1903 AD	
5	1234.567	mmm d	May 18	
6				

All dates can be extended with time tokens. Such tokens format decimal parts of numbers into hours, minutes, and seconds.

Token	Formatting	Example
:	Separates hours, minutes, and seconds. Can be replaced with other separators.	12:12
_	Separates date from time. Can be replaced with other separators.	1/1/92 12:12
h	Displays hours without a leading 0 (zero).	1
hh	Displays hours with a leading 0 (zero).	01
m	Displays minutes without a leading 0 (zero) if follows hours or precedes seconds. Otherwise, displays months.	1
mm	Displays minutes with a leading 0 (zero) if follows hours or precedes seconds. Otherwise, displays months.	01
s	Displays seconds without a leading 0 (zero).	1
s.0 s.00 s.000	Displays seconds and milliseconds without a leading 0 (zero).	1.1 1.12 1.123
ss	Displays seconds with a leading 0 (zero).	01
ss.0 ss.00 ss.000	Displays seconds and milliseconds with a leading 0 (zero).	01.1 01.12 01.123
am/pm a/p	Displays time in the 12-hour clock format.	1 AM
[h]	Displays the number of hours that passed after the editor's base date (12/30/1899) without a leading 0 (zero).	1

[hh]	Displays the number of hours that passed after the editor's base date (12/30/1899) with a leading 0 (zero).	01
[m]	Displays the number of minutes that passed after the editor's base date (12/30/1899) without a leading 0 (zero).	1
[mm]	Displays the number of minutes that passed after the editor's base date (12/30/1899) with a leading 0 (zero).	01
[s]	Displays the number of seconds that passed after the editor's base date (12/30/1899) without a leading 0 (zero).	1
[s.0]	Displays the number of seconds and	1.1
[s.00]	milliseconds that passed after the editor's base	1.12
[s.000]	date (12/30/1899) without a leading 0 (zero).	1.123
[ss.0]	Displays the number of seconds and	01.1
[ss.00]	milliseconds that passed after the editor's base	01.12
[ss.000]	date (12/30/1899) with a leading 0 (zero).	01.123

Note

One formatting rule cannot contain more than one **[h]**, **[m]**, or **[s]** token.

Consider some examples in which date and time tokens are used together.

	A	B	C	D
1	Input value	Format	Formatted value	
2	1234.567	m/d/yyyy h:m:s	5/18/1903 13:36:28	
3	1234.567	h:m am/pm	1:36 PM	
4	1234.567	[h]	29629	
5				

Additional tokens

These are tokens that create unconventional formatting styles.

Token	Formatting
*symbol	Aligns the next character to the right and fills the created space with the supplied symbol.
_	Makes the next character invisible.
"any text"	Displays the supplied text at the given position.
@	Displays the value as text. Cannot be used with number tokens and date/time tokens.
general	Displays the value without changes.
[Color 1–56] [Cyan] [Green] [Black] [Blue] [Magenta] [Red] [White] [Yellow]	Displays the value in the specified color. One formatting rule cannot contain more than one color token.

Note

TEXT ignores ***symbol**, **_**, and all color tokens.

Consider some examples.

	A	B	C	D
1	Input value	Format	Formatted value	
2	123	\$*#	\$123	
3	123	#_ \$	123\$	
4	123	"Week" ww "of year" yyyy	Week 18 of year 1900	
5	123	mmmm d"nd", yyyy	May 2nd, 1900	
6	123	[cyan]@@	123123	
7				

Formats with multiple rules

A custom format can consist of several sections, each with its own formatting rule and required condition. Such sections are separated from each other with semicolons (;).

```
##.00; (##.00); ???; @@
```

By default, one of the four sections applies depending on which of the following conditions is met.

- Section 1 applies if the formatted value is greater than 0 (zero).
- Section 2 applies if the formatted value is less than 0 (zero).
- Section 3 applies if the formatted value is equal to 0 (zero).
- Section 4 applies if the formatted value is text.

	A	B	C	D
1	Input value	Format	Formatted value	
2	1	#.0;(#.0);-;@	1.0	
3	-1	#.0;(#.0);0;@	(1.0)	
4	0	#.0;(#.0);"zero";@	zero	
5	text1	#.0;(#.0);0;@@	text1text1	
6				

Custom conditions are entered inside square brackets, at the beginning of each section.

Note

One section cannot contain more than one logical expression.

	A	B	C	D
1	Input value	Format	Formatted value	
2	3	[>2]#.#;(#.#);"zero";@@	3.	
3	1	[>2]#.#;(#.#);"zero";@@	zero	
4	1	[=1]0.0 "kilometer";0.0 "kilometers"	1.0 kilometer	
5	10	[=1]0.0 "kilometer";0.0 "kilometers"	10.0 kilometers	
6				

Autodetect mechanism

Related topics: Introduction to number formatting, Autodetect mechanism in formulas

Everything that you type into a cell is actually a string. The autodetect mechanism analyzes such strings and saves them as values of different types, for example, numbers or errors.

12,000

	A	B	C
1	12,000		
2			

At this input, a number is detected.

Note

Negative numbers in the accounting notation are also recognized. For example, (10) is detected as -10.

If the autodetect mechanism recognizes a number, it also tries to guess how you want to format your number.

may 12

	A	B	C
1	12-May		
2			

A date is detected, which is actually stored as a number in the format **Date**.

Note

The autodetect mechanism ignores letter case. For example, irrespective of whether you type "May" or "mAy", either input is detected as "May".

The autodetect mechanism restarts every time that you enter a new string into a cell.

	A	B	C
1	1 1/2		
2			

A fraction is detected (or, to express it differently, the input is formatted as **Fraction**).

june 12

	A	B	C
1	12-June		
2			

If another string is put into A1 of the example above, the mechanism restarts and the formatting type is changed to **Date**.

Detectable formats

- **General1** means that no other number format is detected.

	A	B	C	D
	Input string	Result		
1	12	12		
2	1.234	1.234		
3	1.234.56	1.234.56		
4				

Note

Numbers in the **General1** format are aligned to the right. Other values in this format are aligned to the left.

- **Number** is applied if you enter a number with at least one thousands separator.

	A	B	C	D
	Input string	Result		
1	1,234	1,234		
2				

Tip

Enter at least one digit before a thousands separator and at least three digits after it. Otherwise, the mechanism detects a string and applies **General1**.

- **Percentage** is applied if you enter decimal numbers with a percent sign (%).

	A	B	C	D
	Input string	Result		
1	1%	1%		
2	%1	1%		
3				

Note

The percent sign can be placed at the start or at the end of your string.

- **Currency** is applied if you enter decimal numbers with a currency sign or abbreviation.

	A	B	C	D
	Entered string	Result		
1	25\$	\$25		
2	USD 23	USD 23		
3	25,100€	€25,100		
4				

Note

The currency sign can be placed at the start or at the end of your string.

- **Fraction** is applied if you enter a fraction with an integer part.

	A	B	C	D
	Input string	Result		
1	1 1/2	1 1/2		
2	1 2/4	1 2/4		
3	1 2/2	2		
4				

Note

All fractions are reduced to lowest terms. If the result is an integer, the autodetect mechanism still applies **Fraction** as the cell format.

- **Scientific** is applied if you enter numbers in scientific E-notation.

	A	B	C	D
	Input string	Result		
1	2.12E+14	2.12E+14		
2	3.23E-28	3.23E-28		
3	2.12E14	2.12E+14		
4				

Tip

Omit the plus sign if you want to enter a positive exponent.

- **Time** is applied if you enter time based on the 24- or 12-hour clock.

	A	B	C	D
	Input string	Result		
1	13:25:00	1:25:00 PM		
2	9:25 am	9:25 AM		
3	13:45:46.123	13:45:46 PM		
4	:45:	12:45 AM		
5				

Notes

- Milliseconds are never displayed after automatic formatting.
- Omitted hours are treated as 12 AM (unless you specify PM).
- Omitted minutes, seconds, and milliseconds are treated as zeros.

- **Date** is applied if you enter dates. Multiple notations are supported.

	A	B	C	D
	Input string	Result		
1	11/23/2009	11/23/2009		
2	11-23-2009	11/23/2009		
3	23 Nov 2009	Nov 23, 2009		
4	November 23	23-Nov		
5	11/2009	Nov-09		
6				

Notes

- If you enter year in short form, the editor adds 2000 years to the specified number. For example, 11/15/19 is detected as 11/15/2019, and 10/30/1 is detected as 10/30/2001.
- If you enter dates with month names or abbreviations, you are allowed to separate the components with any spacing or use no separator at all.
- You can omit the day or the year in a date. If you omit the year, the editor sets the current year. When you omit the day, the editor sets the first day of the specified month.

- **Date and time** is applied if you enter date and time together in either order and separate them from each other with any spacing.

	A	B	C	D
	Input string	Result		
1	7/23/2018 12:45 pm	7/23/2018 12:45 pm		
2	26 April 19:30:30	4/26/2019 7:30 PM		
3	16:45 Aug 1998	8/1/1998 4:45 PM		
4				

Autodetect mechanism in formulas

When you enter a formula, the autodetect mechanism splits the formula into values and analyzes them one by one. The first detected format is then applied to the result.

```
=10% + 1 1/2
```

	A	B	C
1	160%		
2			

Here, the first detected format is **Percentage**.

Sometimes, to avoid ambiguity, you need to enclose your value in double quotation marks. This indicates that the marked-out string is a single value and not an expression.

```
= "12/1/12" + 1
```

	A	B	C
1	12/2/2012		
2			

If a cell address refers to a formatted cell, this formatting is also detected.

```
=A1
```

	A	B	C
1	12%		
2		12%	

Autodetect mechanism in functions

Most functions ignore automatically detected formats and return numbers either in **General** or their predefined formats.

```
=PRODUCT(1 1/2, "1$")
```

	A	B	C
1	1.5		
2			

PRODUCT always returns numbers in **General**.

```
=DATE(2010, 1 1/2, 100%)
```

	A	B	C
1	1/1/2010		
2			

Date always returns numbers in **Date**.

However, there are functions that keep the automatically detected format.

```
=MAX(100%, 2, 3 1/2)
```

	A	B	C
1	350%		
2			

Note

All functions ignore formats that are supplied via cell references.

Chapter 4

Localization

Introduction to localization

Related topics: Functions, Introduction to number formatting, How the editor chooses locales.

Localization is a process of adapting a product to a certain locale.

Note

A locale is a combination of a language and a region. For example, the default locale for our editor is English—United States.

Localization not only includes translation but also addresses many nontextual components of the product. For example, below, you can see how the Currency format is localized for the English—United States locale (left) and for the English—United Kingdom locale (right).

	A	B
1	\$10.00	
2		

	A	B
1	£10.00	
2		

These two locales share the same language but require different localization. Now, consider an opposite example of locales that share the same region: English—Canada (left) and French—Canada (right). Here, the Number format is localized.

	A	B
1	10.00	
2		

	A	B
1	10,00	
2		

As you can see, to perform localization correctly, it is important to consider a locale as a combination of a language and a region.

Note

On some systems, you can directly affect the process of localization. For example, if your system allows you to choose your preferred currency symbol separately, the editor adopts this symbol regardless of your locale.

Documents that are created or changed in our editor are localized for every subsequent use. Also, when you work on a document with other people at the same time, everyone

sees their own localized version.

How the editor chooses locales

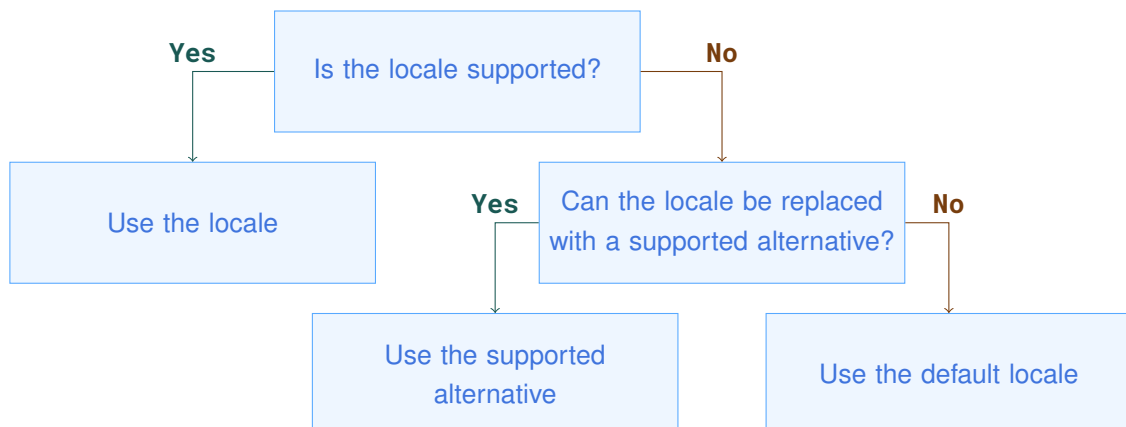
Related topics: [Introduction to localization](#), [Introduction to number formatting](#)

As the first step of localization, the editor checks your system locale. If the locale is supported, it is simply applied to your menus and data.

Tip

If you are unfamiliar with the topic of localization, we strongly advise you to read [Introduction to localization](#) first.

However, it is difficult to support all language and region combinations in the world. To solve the problem, our editor uses a special mechanism, which replaces unsupported locales with their closest supported alternatives.

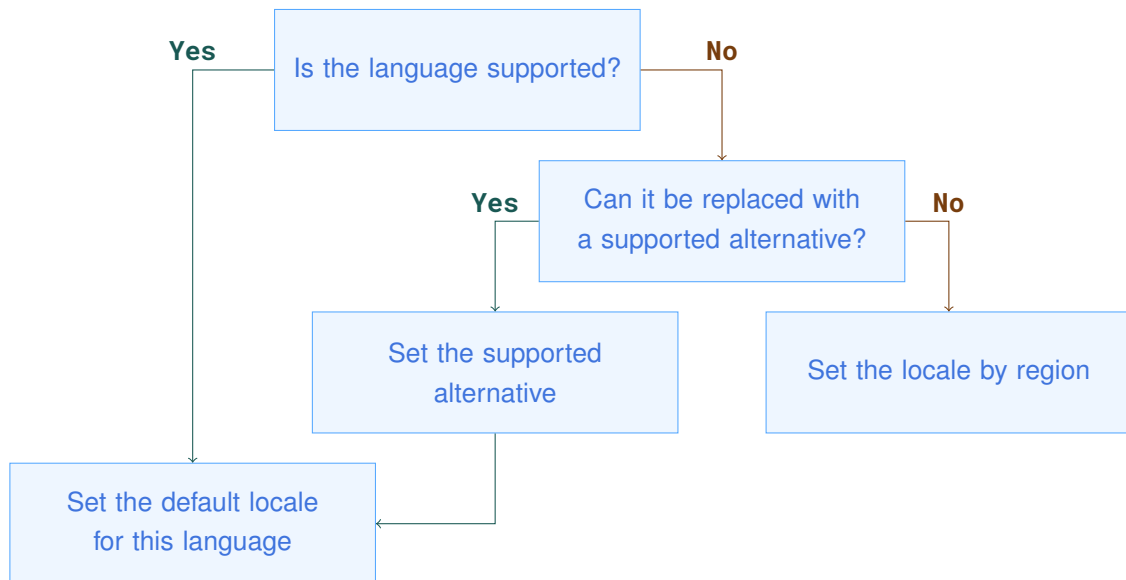


Note

The default locale for the editor is English—United States.

Replacing the locale based on the system language

Language has the biggest impact on localization. Therefore, when the editor receives an unknown locale, it tries to find an alternative based on your system language.



If the language is supported, the editor saves it and combines it with its default region. As a result, unsupported locales that contain English, French, or Russian are converted to English—United States, French—France, and Russian—Russia respectively. If the language can be replaced with Russian, the editor sets Russian—Russia as the locale.

Examples

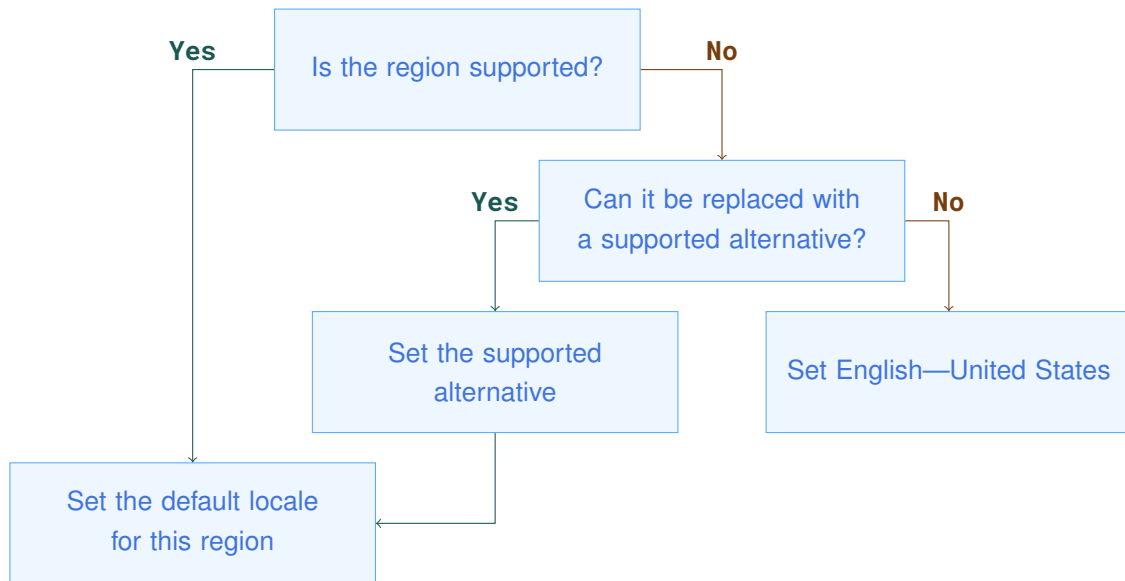
- Your system locale is Russian—Canada, which is not supported. The Russian language, however, is supported, so the editor sets the default locale for this language, which is Russian—Russia.
- If your locale is Bashkir—Canada, Bashkir is replaced with Russian, and the resulting locale is also Russian—Russia.
- The system locale is French—Italy. This combination is not supported because the language is supported, but the region is not. The editor replaces Italy with France and applies French—France, which is the default locale for French.

Note

The chosen locale is not the only factor in the localization result. For example, some operating systems allow you to customize date formatting. Such customizations transfer directly into the editor. For more information, see [Introduction to localization](#).

Replacing the locale based on the system region

If your language is unsupported, the editor tries to set the locale based on your region.



If the region is supported, the editor saves it and combines it with the corresponding supported language. As a result, such regions as Dutch—Australia are converted into English—Australia. If the region can be replaced with Russia, the editor sets Russian—Russia as the locale. If the region is unsupported, the editor sets the default locale, which is English—United States.

Examples

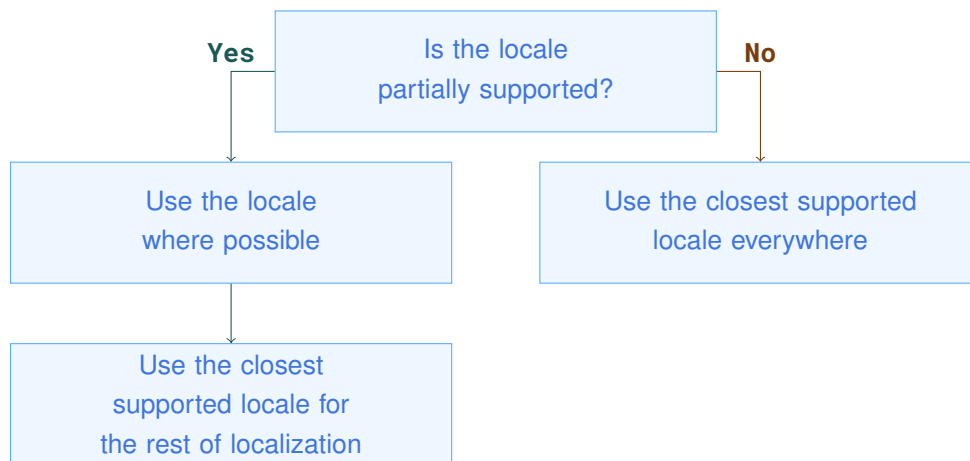
- Your system locale is Chinese—Russia. This combination is not supported. The language is not supported, but the region is. The editor applies Russian—Russia because it is the default locale for Russia. If your locale is Chinese—Kyrgyzstan, Kyrgyzstan is replaced with Russia, and the resulting locale is also Russian—Russia.
- The system locale is Italian—Canada. This combination is not supported. The language is unsupported, but the region is. The editor applies English—Canada because it is the default locale for Canada.
- The system locale is German—Uzbekistan. This combination is not supported. The language is unsupported, but the region can be replaced. The editor applies Russian—Russia because Uzbekistan is replaced with Russia, and Russian is the default language for Russia.

Note

For the full list of the supported and replaceable languages and regions, see [Supported locales and replaceable languages and regions](#).

Partially supported locales

Some locales are partially supported in our editor. During localization, they are applied to some features and only then are replaced with their closest fully supported alternatives.



Example

Bashkir—Russia is a partially supported locale. The editor applies this locale where possible and then uses Russian—Russia as the closest fully supported locale to complete the localization.

Supported locales and replaceable languages and regions

Locale support

- English—United States
- English—Canada
- English—United Kingdom

- English—Australia
- English—New Zealand
- Russian—Russia
- French—France
- French—Burundi
- French—Cameroon
- French—Congo (Republic)
- French—Morocco

Languages that can be replaced with Russian

- Armenian
- Belorussian
- Kazakh
- Kyrgyz
- Romanian
- Tajiki
- Turkmen
- Uzbek
- Ukrainian
- Bashkir
- Tatar

Regions that can be replaced with Russia

- Armenia
- Belarus
- Kazakhstan
- Kyrgyzstan

- Moldova
- Tajikistan
- Turkmenistan
- Uzbekistan
- Ukraine

Chapter 5

Pivot tables

Pivot table overview

A pivot table analyzes your data and creates a report with criteria. When you have large data sets, creating and editing reports with a pivot table is much faster and easier than working with formulas. This article introduces the most commonly used pivot table features.

Creating and editing a report

Here is a fragment of a data table.

	A	B	C	D	E	F	G
1	Date	Country	SalesRep	Product	Prbice	Units	Sales
2	2009-06-07	UK	Ezra	SuperX Core	\$75.00	29	\$2,175.00
3	2009-08-11	Spain	Hatter	SuperX Basic	\$50.00	44	\$2,200.00
4	2009-10-15	Italy	Donovan	SuperX Basic	\$50.00	38	\$1,900.00
5	2009-12-19	Spain	Struksen	SuperX Deluxe	\$100.00	12	\$1,200.00
6	2010-02-22	US	Hatter	SuperX Core	\$75.00	34	\$2,550.00

Below, there is a pivot table report created on the basis of the above data table.

	A	B	C
1	SalesRep ▼	Product ▼	Sum of Sales
2	Dunn		
3		SuperX Basic	3100
4		SuperX Deluxe	1500
5	Dunn Total		4600
6	Ezra		
7		SuperX Core	2175
8		SuperX Deluxe	2700
9	Ezra Total		4875
10	Hatter		
11		SuperX Basic	2200
12		SuperX Core	2550
13	Hatter Total		4750
14	Stoke		
15		SuperX Basic	1650
16		SuperX Deluxe	1200
17	Stoke Total		2850
18	Grand Total		17075

Next, the same report has been edited to use different criteria and data arrangement.



	A	B	C	D	E	F	G
1	Sum of Sales	Country ▼					
2	SalesRep ▼	China	Italy	Spain	US	UK	Grand Total
3	Dunn		1900			2700	4600
4	Ezra			2700	2175		4875
5	Hatter			2200		2550	4750
6	Stoke	1650		1200			2850
7	Grand Total	1650	1900	6100	2175	5250	17075

Tip

See [Create and edit pivot tables](#) to learn how to prepare data for a pivot table and how to edit a pivot table report.

Filtration

To hide unnecessary data in your report, use filters.

	A	B
1	SalesRep	Dunn 
2		
3	Years 	Sum of Sales
4	2009	1900
5	Grand Total	1900

The filters are set to display sales in 2009 and for a sales representative named Dunn.

Tip

See [Filtering data in a pivot table](#) to learn how to use filters in a pivot table.

Changing the aggregate function

By default, a pivot table uses **SUM** as the aggregate function for numeric values (or **COUNT** for text), but you can use other functions as well.

	A	B
1	Country <input type="button" value="v"/>	Average of Sales
2	China	1650
3	Italy	1900
4	Spain	2033
5	UK	2175
6	US	1750
7	Grand Total	1897

In this report, **AVERAGE** is used to summarize the sales.

Tip

See [Aggregate functions](#) for the full list of available functions.

Load only features

The following features cannot be enabled or edited in the editor, but you can load documents that already have them enabled.

Sorting

In the following report, sales representative are sorted from Z to A.

	A	B
1	SalesRep <input type="button" value="v"/>	Sum of Sales
2	Stoke	2850
3	Hatter	4750
4	Ezra	4875
5	Dunn	4600
6	Grand Total	17075

Note

Any changes to a pivot table resets sorting to the ascending order:

- from A to Z for text;
- from Smallest to Largest for numbers;
- from Oldest to Newest for dates.

Grouping

Analyze smaller subsets of your data by loading pivot tables with groups. In the following report, Italy, Spain, and UK are combined into a custom group named Europe.

	A	B
1	Country <input type="button" value="v"/>	Sum of Sales
2	China	
3	China	1650
4	China Total	1650
5	Europe	
6	Italy	1900
7	Spain	6100
8	UK	2175
9	Europe Total	10175
10	US	
11	US	5250
12	US Total	5250
13	Grand Total	17075

Additional calculations

Load pivot tables with additional calculations to enhance data comparison.

	A	B
1	Country <input type="button" value="v"/>	Sum of Sales
2	China	9.66%
3	Italy	11.13%
4	Spain	35.72%
5	UK	12.74%
6	US	30.75%
7	Grand Total	100%

The pivot table performs additional calculations to display all aggregates as percentages of the Grand Total.

Create and edit pivot tables

Related topics: [Pivot table overview](#), [Aggregate functions](#), [Filtering data in a pivot table](#).

A report created with a pivot table differs from a table containing regular ranges with values and formulas. This article describes the main stages of working with such a report.

Preparing your data

A pivot table can only summarize data of specific structure. You need a cell range with columns of values. Each column must start with a heading.

	A	B	C	D
1	Date	Name	Sales	
2	2009-06-07	C. Dickens	\$750.00	
3	2009-08-11	C. Bukowski	\$650.00	

Inserting a pivot table

To start working with a pivot table, insert it into your workbook using the Insert menu or the Insert tab. Both actions open a window, in which you have to specify the location of your data source and your future pivot table, for example.

A1:F10

If you work with multiple worksheets, specify the address with the sheet name.

'Sheet1'!A1:F10

If your range has a name, you can use it instead of the address.

MyDataSource1

Some named ranges work only on the sheets on which they were created. To specify such a range as your data source, add the sheet name.

'Sheet1'!MyNamedRange

Tip

While you are in the Create Pivot Table dialog box, you can click on any cell of your named range to automatically select it as the data source.

If you frequently work with imported documents, you may be familiar with ranges that have banded rows, applied sorting and filtering, and dynamic size.

Note

Microsoft Excel users may know ranges that have banded rows, applied sorting and filtering, and dynamic size as Excel tables.

	A	B	C	D	E
1	Date ▾	Time ▾	Title ▾	Edition ▾	Price/Unit ▾
2	3/1/21	9:12	Women	paperback	\$8.95
3	3/2/21	11.25	Ham on Rye	hadrcover	\$13.95
4	3/2/21	12:40	Ham on Rye	e-book	\$11.99
5	3/2/21	13:45	Post Office	paperback	\$8.95

You can refer to such ranges by their names.

Table1

Also, data in such ranges can be accessed with structured references, which are references composed with column headers and special modifiers instead of letters and numbers. For example, two column names can be used to specify the data source for your pivot table.

Table1[[Date]:[Edition]]

	A	B	C	D	E
1	Date ▾	Time ▾	Title ▾	Edition ▾	Price/Unit ▾
2	3/1/21	9:12	Women	paperback	\$8.95
3	3/2/21	11.25	Ham on Rye	hadrcover	\$13.95
4	3/2/21	12:40	Ham on Rye	e-book	\$11.99
5	3/2/21	13:45	Post Office	paperback	\$8.95

The highlighted range is used as the data source.

To specify the location of your future report, you need to specify a cell address.

'Sheet1'!G7

Alternatively, you can tell the editor to create a new worksheet for your pivot table by selecting the corresponding option in the dialog.

Tip

When you place a pivot table on a worksheet that already has data, the created pivot table can overlap the existing data and result in an error message. To avoid this, place pivot tables in new worksheets.

You can change your data source at any time in the pivot table options.

Editing

For a pivot table, each column in your data source is a field. For example, the following table has seven fields (Date, Time, Title, Edition, Price/Unit, Units, and Total Price).

	A	B	C	D	E	F	G
1	Date	Time	Title	Edition	Price/Unit	Units	Total Price
2	3/1/21	9:12	Women	paperback	\$8.95	8	\$71.60
3	3/2/21	11:25	Ham on Rye	hardcover	\$13.95	3	\$41.85
4	3/2/21	12:40	Ham on Rye	e-book	\$11.99	1	\$11.99
5	3/2/21	13:45	Post Office	paperback	\$8.95	9	\$80.55
6	3/2/21	14:50	Ham on Rye	paperback	\$1.21	6	\$7.26
7	3/2/21	16:40	Women	hardcover	\$14.58	1	\$14.58
8	3/3/21	9:00	Post Office	e-book	\$13.99	1	\$13.99
9	3/3/21	9:06	Ham on Rye	paperback	\$7.15	30	\$214.50

Note

If your data source has multiple columns with identical headings, the pivot table uses numbers to distinguish fields from each other. For example, if you have two Date columns, for a pivot table, they will be Date and Date1.

To fill your report with data, you need to add fields to the report and specify how they should be used by the pivot table. This is done through the pivot table Constructor, where you can add available fields to Rows, Columns, Filters, or Values.

Adding a field to Columns uses the field to create criteria that are placed as column headings.

	A	B	C	D
1				
2	Title ▾			
3	Ham on Rye	Post Office	Women	Grand Total
4				

You can add more fields to create subcriteria.

	A	B	C	D	E
1	Title ▾				
2	Ham on Rye			Ham on Rye Total	Post Office
3	e-book	hardcover	paperback		

Tip

You can also change the hierarchy of your headings.

Adding a field to Rows uses the field to create criteria that are placed as row headings.

	A	B	C	D	E
1		Title ▾			
2	Edition ▾	Ham on Rye	Post Office	Women	Grand Total
3	e-book				
4	hardcover				
5	paperback				
6	Grand Total				

Subcriteria are also possible.

	A
1	Title ▾
2	Ham on Rye
3	e-book
4	hardcover
5	paperback
6	Ham on Rye Total
7	Post Office

Adding a field to Values uses the field as a data set for calculating an aggregate.

	A	B	C	D	E
1	Sum of Units	Title ▼			
2	Edition ▼	Ham on Rye	Post Office	Women	Grand Total
3	e-book	2	3	1	6
4	hardcover	11	2	2	15
5	paperback	44	11	9	66
6	Grand Total	59	16	12	87

You can also add another field to perform calculations on two sets of values.

	A	B	C	D
1	Sum of Units	Title ▼		
2		Ham on Rye		Post Office
3	Edition ▼	Sum of Units	Sum of Total Price	Sum of Units
4	e-book	2	16.99	3
5	hardcover	11	165.69	2
6	paperback	46	293.26	11
7	Grand Total	59	475.94	16

By default, the calculations are performed with **SUM** for numbers and with **COUNT** for other values, but you can use other aggregate functions as well. See [Aggregate functions](#) for the full list.

Tip

You can even add the same field to Values multiple times. This is useful when you want to perform different calculations on one set of values.

Adding a field to Filters uses the field to create criteria that are placed above the report.

	A	B	C
1	Date	3/1/2021 ▼	
2			
3	Sum of Units	Title ▼	
4	Edition ▼	Women	Grand Total
5	paperback	8	8
6	Grand Total	8	8

Note

Filters are also created automatically for each set of headings or values. See [Pivot table filters](#) for more information.

To rearrange data in your report you can

- add new fields,
- remove existing fields,
- respecify fields that have already been added.

Updating and deleting pivot tables

Changes that you make in your data source do not transfer automatically to your pivot table. To get them, you need to update your pivot table using the **Update** option. If you have multiple pivot tables, each of them must be updated separately.

Tip

When you open a document with a pivot table, the pivot table is not connected to its data source. To connect them, use the **Update** option or start editing your pivot table.

A pivot table remembers the location of its data source as a range address in your workbook. If you enter data outside this range (unless its dynamic), your pivot table cannot detect it. To supply your pivot table with new data, change the address of your data source in the pivot table options.

Tip

Another way to supply your pivot table with more data is to insert new rows and columns into your data source. When you add data this way, data source change is not required. To analyze less data, delete unnecessary rows and columns from the range.

To delete a pivot table, you must delete all its cells together because it is a single object.

Pivot grouping

Grouping separates pivot table data into smaller subsets that are easier to analyze.

Note

Currently, you cannot create groups yourself, but you can load documents that already have them. Editing is fully supported.

Grouped text items

This type of grouping usually highlights subtotals for locations, products, or sales representatives.

Here is an example of grouped rows.

	A	B	C	D	E
1	Units Sold	Product <input type="text"/>			
2	City <input type="text"/>	Blanket	Pillow	Socks	Grand Total
3	West				
4	San Diego		368		368
5	San Jose		118		118
6	West Total		486		486
7	East				
8	Boston	2095		421	2516
9	Tampa		741	2074	2815
10	East Total	2095	741	2495	5331
11	Grand Total	2095	1227	2495	5817

Below, you can see an example of grouped columns.

	A	B	C	D	E	F
1	Units Sold	Product <input type="text"/>				
2		Bedroom		Bedroom Total	Socks	Socks Total
3	City <input type="text"/>	Blanket	Pillow		Socks	
4	Boston	2095		2095	421	421
5	San Diego		368	368		
6	San Jose		118	118		
7	Tampa		741	741	2074	2074
8	Grand Total	2095	1227	3322	2495	2495

For a pivot table, each level of grouping is an additional field. You can easily recognize such fields by their names because they are similar to fields with grouped items: for example, **City2** and **Product2**. To remove a grouping level, remove the corresponding field from your report.

	A	B	C	D	E
1	Units Sold	Product ▼			
2	City ▼	Blanket	Pillow	Socks	Grand Total
3	San Diego		368		368
4	San Jose		118		118
5	Boston	2095		421	2516
6	Tampa		741	2074	2815
7	Grand Total	2095	1227	2495	5817

You can also remove the field with grouped items and leave just the groups.

	A	B	C	D	E
1	Units Sold	Product ▼			
2	City ▼	Blanket	Pillow	Socks	Grand Total
3	West		486		486
4	East	2095	741	2495	5331
5	Grand Total	2095	1227	2495	5817

Grouped dates

Dates can be grouped by years, quarters, months, days, hours, minutes, and seconds, with multiple levels of grouping enabled at once.

	A	B
1	Date ▼	Units
2	2010	
3	Q3	4
4	Q4	6
5	2011	
6	Q1	10
7	Q2	7
8	Q3	4
9	Q4	6
10	Grand Total	37

Note

Multiple levels of date grouping come with extra fields, which can be Years, Quarters, Months, Days, Hours, Minutes, and Seconds.

Grouping can cover all dates or a certain period.

	A	B
1	Date <input type="button" value="v"/>	Units
2	<1/1/2011	10
3	Q1	10
4	Q2	7
5	Q3	4
6	Q4	6
7	Grand Total	37

A grouping interval can be an arbitrary number of days.

	A	B
1	Date <input type="button" value="v"/>	Units
2	<10/31/2010	5
3	10/31/2010 - 11/27/2010	2
4	11/28/2010 - 12/25/2010	3
5	>1/1/2011	27
6	Grand Total	37

Grouped numbers

Grouping numbers is a great way to show frequency distribution in your report. Instead of having a very long list of values, you can get a compact summary. In the following table, numbers are grouped by 500.

	A	B	C	D	E	F
1	Orders	Product <input type="button" value="v"/>				
2	Units <input type="button" value="v"/>	San Diego	San Jose	Tampa	Boston	Grand Total
3	100-599	1	5	8	7	21
4	600-1099		5	5	6	16
5	Grand Total	1	10	13	13	37

Grouping can be specified with the starting and ending points.

	A	B
1	Units <input type="button" value="v"/>	Transactions
2	<500	20
3	500-599	1
4	600-700	6
5	700-799	1
6	800-899	5
7	900-999	4
8	Grand Total	37

Once enabled, number grouping can detect changes in your data source and adapt accordingly.

Note

Remember to update your pivot table after any changes in the data source.

	A	B
1	Units <input type="button" value="v"/>	Transactions
2	<500	20
3	500-599	1
4	600-700	6
5	700-799	1
6	800-899	5
7	900-999	4
8	1500-1600	1
9	Grand Total	37

As a result of changes in the data source, an additional group is created (1500-1600).


Filtering data in a pivot table

Related topics: PIVOT TABLE OVERVIEW, CREATE AND EDIT PIVOT TABLES.


Filtration allows you to hide data that does not satisfy specific conditions. This article describes how to enable it.

Applying filters to Rows and Columns fields


When you create a pivot table with at least one field of criteria, you already have a filter. In the following example, the report has a filter by Date.

	A	B
1	Date 	Sales
2	01/01/2020	2248.72
3	01/02/2020	2366.46
4	01/03/2020	2684.92
5	01/04/2020	2402.48
6	01/05/2020	3066.87
7	Grand Total	12769.45

The filter is enabled to show sales for January 1, 2020.

	A	B
1	Date 	Sales
2	01/01/2020	2248.72
3	Grand Total	2248.72



You can set multiple conditions for a filter.

	A	B
1	Date 	Sales
2	01/01/2020	2248.72
2	01/02/2020	2366.46
3	Grand Total	4615.18

Note

Currently, filters can only search for an exact match, in other words, filters work if a value is equal to another value.

To narrow down your results even more, you can enable filters for multiple fields.

	A	B	C
1	Sum of Sales	Product 	
2	Date 	Lion	Grand Total
3	01/01/2020	458.92	458.92
4	01/02/2020	480.2	480.2
5	Grand Total	939.12	939.12




This filter shows only Lion sales for January 1 and January 2.

Note





A pivot table remembers the state of your filter even after you remove the field with the filter. If you want to see all your data again, clear the filter manually.

Adding fields as Filters

To apply additional criteria, you can add fields as filters. For example, the Discount field has been introduced for further filtering.

	A	B	C
1	Discount	10% 	
2			
3	Sum of Sales	Product 	
4	Date 	Snickers	Grand Total
5	01/01/2020	209.52	209.52
6	01/02/2020	725.76	725.76
7	01/03/2020	768.42	768.42
8	01/04/2020	823.5	823.5
9	01/05/2020	1213.38	1213.38
10	Grand Total	2655.43	2578.24

When you add several filter fields, your pivot table shows results that satisfy both conditions at once.

	A	B	C
1	Discount	10% 	
2	City	Swansea 	
3			
4	Sum of Sales	Product 	
5	Date 	Snickers	Grand Total
6	01/03/2020	121.5	121.5
6	01/04/2020	557.28	557.28
7	01/03/2020	858.06	858.06
8	Grand Total	1536.84	1536.84

Note

- Filtration via charts cannot be enabled. However, if you load a pivot table with this type of filtering enabled in other editors, the filtering options are saved.
- Filtration via slicers and timelines is not supported.

Aggregate functions

The full list of functions that can be used to summarize data in a pivot table:

Name	Description
Sum	The sum of all numeric values. The default function for numbers.
Count	The quantity of values. The default function for non-numeric values.
Count Numbers	The quantity of numeric values.
Average	The average of numeric values.
Max	The maximum value.
Min	The minimum value.
Product	The product of numeric values.
StdDev	An estimated standard deviation of a population, where the values are a subset of the entire population.
StdDevP	The standard deviation of a population, where the values are the entire population.
Var	An estimated variance of a population, where the values are a subset of the entire population.
VarP	The variance of a population, where the values are the entire population.

Additional calculations

The full list of additional calculations that can be loaded with a pivot table:

Name	
% of	Shows the value as percentage of an item in a field.
Difference from	Shows the value as difference from an item in a field.
% of difference from	Shows the value as percentage difference from an item in a field.
Running total in	Shows the value as a running total in a field.
% of row total	Shows the value as percentage of row total.
% of column total	Shows the value as percentage of column total.
% of grand total	Shows the value as percentage of grand total.
Index	Calculates the relative weight of the value based on totals: $\frac{(\text{Value}) * (\text{Grand Total Of Grand Totals})}{(\text{Grand Row Total}) * (\text{Grand Column Total})}$

Custom formulas in pivot tables

Pivot tables can use custom formulas to create calculated fields.

Calculated fields

Note

Currently, you cannot create calculated fields in our editor, but you can load pivot tables with calculated fields created in other editors and work with such fields.

Understanding calculated fields

Calculated fields are fields in your pivot tables that perform operations on values in other fields or constants and formulas with constants. As a result, you can use and calculate values that are not in your data sets directly and apply additional logic to work with data in your pivot table. For example, here is a report without calculated fields.

	A	B
1	Product <input type="checkbox"/>	Sales (USD)
2	Bounty	2655.43
3	Lion	2578.24
4	Mars	3795.2
5	Snickers	3740.58
6	Grand Total	12769.45

The report shows sales figures for a list of products along with the grand total. Below, you can see the same report with a calculated field (Sales (EUR)).

	A	B	C
1			
2	Product <input type="checkbox"/>	Sales (USD)	Sales (EUR)
3	Bounty	2655.43	2257.1155
4	Lion	2578.24	2191.504
5	Mars	3795.2	3225.92
6	Snickers	3740.58	3179.493
7	Grand Total	12769.45	10854.0325

The calculated field formula in column C converts USD to EUR.

=Sales*0.85

Using calculated fields

Note

All errors resulting from formulas supplied externally into calculated fields will be displayed in the pivot table.

A pivot table with one or more calculated fields works in a regular way. A calculated field looks like any other field in the table. The original name of the field is not affected by any actions on the pivot table.

	A	B	C	D
1		Title <input type="text"/>		
2		The Shining		
3	Edition <input type="text"/>	Sum of Items	Sum of Price	Sum of Total
4	e-book	2	8	16
5	hardcover	4	24	96
6	paperback	8	12	96
7	Grand Total	14	44	208

The calculated field "Total" (column D) is based on the fields "Items" (column B) and "Price" (column C). "Total" calculates the total for each row and uses the formula below.

=Items*Price

Note

A calculated field can contain items with subitems within.

- The subtotal for an item is the sum of all the subitem values within the item.
- The grand total for a calculated field is the sum of all the subtotal values within the field.

The name of a calculated field appears at the bottom of the Available Fields list sorted in ascending order. You can make a calculated field appear or disappear in your pivot table. If you change data in fields on which a calculated field is based, the changes will apply after refreshing.

Warning

If the formula in a calculated field contains an unsupported function, the #NAME? error will be displayed.

Understanding rules of using calculated fields

Warning

If you change the name of a field on which a calculated field formula is based, the #NAME? error is displayed as a result of calculation.

There are a number of rules in using calculated fields.

- You can add (drag and drop) a calculated field only to the Values area.
- You cannot see the formula supplied into a field or make changes to the formula.
- You cannot change the name of a calculated field.
- Of the Aggregate functions, only **SUM** is permitted for calculated fields.

Note

In other editors, various functions can be used for the subtotal and aggregate of a calculated field, but they have no impact on the calculations of the field itself.

In our editor, if a calculated field was originally set to use a function other than **SUM** for the aggregate, **SUM** is applied automatically after the table is refreshed. The application of **SUM** has no impact on output of the field.

If a calculated field was originally set to use a function other than **SUM** for the subtotal, no change in the function is applied, and the subtotal value for the field remains empty.

Pivot table options

Related topics: Pivot table overview, Create and edit pivot tables.

The appearance of your pivot table is largely affected by its options. Currently, you cannot change them, but pivot tables that were created in other editors can be loaded with modified options.

Data source

This setting allows you to change your data source at any time and does not affect any other settings.



Note

Changing your data source automatically updates your pivot table.



Show totals

This setting enables and disables grand totals for rows and columns:

- **For columns.**

	A	B	C	D
1	Sum of Units	Title 		
2	Edition 	Ham on Rye	Post Office	Women
3	e-book	2	3	1
4	hardcover	11	2	2
5	paperback	44	11	9
6	Grand Total	59	16	12

- **For rows.**

	A	B	C	D	E
1	Sum of Units	Title 			
2	Edition 	Ham on Rye	Post Office	Women	Grand Total
3	e-book	2	3	1	6
4	hardcover	11	2	2	15
5	paperback	44	11	9	66
6					

By default, totals are displayed for both rows and columns:

	A	B	C	D	E
1	Sum of Units	Title ▼			
2	Edition ▼	Ham on Rye	Post Office	Women	Grand Total
3	e-book	2	3	1	6
4	hardcover	11	2	2	15
5	paperback	44	11	9	66
6	Grand Total	59	16	12	87

Table layout

This setting changes the layout form of your pivot table:

- **Compact** displays all row headings in the same column and uses indentation to distinguish levels. This is the default pivot table layout.

	A	B
1	Edition ▼	Sum of Units
2	Ham on Rye	
3	e-book	2
4	hardcover	11
5	paperback	46
6	Ham on Rye Total	59
7	Post Office	16

- **Tabular** displays each level of headings in a separate column.

	A	B	C
1	Title ▼	Edition ▼	Sum of Units
2	Ham on Rye	e-book	2
3		hardcover	11
4		paperback	46
5	Ham on Rye Total		59
6	Post Office	e-book	3

- **Outline** is similar to **Tabular** but skips one row before starting the next level of headings.

	A	B	C
1	Title ▾	Edition ▾	Sum of Units
2	Ham on Rye		
3		e-book	2
4		hardcover	11
5		paperback	46
6	Ham on Rye Total		59
7	Post Office		
8		e-book	3

Captions

This section of options controls the appearance of empty values, errors, and various other captions in your pivot table. For example, below all empty values are replaced with **0**, and captions for rows and columns are changed:

	A	B	C	D	E
1	Sum of Units	Book Title ▾			
2	Transaction Date ▾	Ham on Rye	Post Office	Women	Grand Total
3	03/01/2021	0	0	8	8
4	03/02/2021	10	9	1	20
5	03/03/2021	32	3	0	35

By default, captions are displayed without changes.

Arrange filters

The section controls the arrangement of fields that are added as filters:

- **Down, then to the right** displays filters from top to bottom before starting a new column (works in conjunction with **Filter fields per column**). By default, this option is enabled.
- **Filter fields per column** sets how many filters you want to display before starting a new column when **Down, then to the right** is enabled.

	A	B	C	D	E
1	Date	(All Items) ▾		Title	(All Items) ▾
2	Time	(All Items) ▾			
3	Price/Unit	(All Items) ▾			
4					

- **To the right, then down** displays filters from left to right before starting a new row (works in conjunction with **Filter fields per row**).
- **Filter fields per row** sets how many filters you want to display before starting a new row when **To the right, then down** is enabled.

	A	B	C	D	E
1	Date	(All Items) ▼		Time	(All Items) ▼
2	Price/Unit	(All Items) ▼		Title	(All Items) ▼
3					

Display value fields

When you add more than one field to Values, the constructor creates an additional field with headings to distinguish results. This section controls the arrangement of these headings:

- **As columns** displays them as column headings. By default, this setting is enabled.
- **As rows** displays them as row headings.

Number formatting in pivot tables

Related topics: Pivot table overview, Cell number formatting, Localization.

Number formatting in pivot tables is defined in two ways: automatically based on your data source and manually through field settings.

Automatic formatting based on your data source

When you create a pivot table, it retains formatting from the data source for fields that are added as Rows, Columns, and Filters:

	A	B	C	D	E	F
1	Data Source					
2						
3	Date	Product	Price per unit	Quantity	Discount	Sales
4	01/01/2020	Bounty	\$0.65	374	25.00%	\$182.33
5	01/01/2020	Lion	\$0.40	349	30.00%	\$97.72

	A	B	C	D	E	F
1	Pivot Table					
2						
3	Sum of Sales	Discount ▼				
4	Date ▼	10.00%	20.00%	25.00%	30.00%	Grand Total
5	01/01/2020	209.52	1193.20	387.08	458.92	2248.72

Note

Fields in the **Values** area always use the **General** format no matter how they are formatted in the data source.

If your pivot table does not retain the formatting as you intended, make sure that you have only one format per column in your data source.

Note

Another reason why it is important to keep your data source formatting consistent is that equivalent values in different formats are treated by the editor as equal. For example, if a field has 200% and 2\$, and you add it as a Rows, these two items are detected as one (number 2 in the **General** format).

The following examples demonstrate how a pivot table works with inconsistent formatting.

Multiple formats in a data source column result in the **General** format (except for dates):

	A	B	C	D	E	F
1	Data Source					
2						
3	Date	Product	Price per unit	Quantity	Discount	Sales
4	01/01/2020	Bounty	\$0.65	374	0.25	\$182.33
5	01/01/2020	Lion	\$0.40	349	30.00%	\$97.72

	A	B	C	D	E	F
1	Pivot Table					
2						
3	Sum of Sales	Discount ▼				
4	Date ▼	0.1	0.2	0.25	0.3	Grand Total
5	01/01/2020	209.52	1193.20	387.08	458.92	2248.72

Multiple variations of one format in a data source column result in the first detected format in the column:

	A	B	C	D	E	F
1	Data Source					
2						
3	Date	Product	Price per unit	Quantity	Discount	Sales
4	01/01/2020	Bounty	\$0.65	374	25%	\$182.33
5	01/01/2020	Lion	\$0.40	349	30.00%	\$97.72

	A	B	C	D	E	F
1	Pivot Table					
2						
3	Sum of Sales	Discount ▼				
4	Date ▼	10%	20%	25%	30%	Grand Total
5	01/01/2020	209.52	1193.20	387.08	458.92	2248.72

Note

- **Dates** are always formatted as your system's short date with leading zeros (for example, 01/01/2020).
- **Time** is always formatted according to your system regional preferences.
- Mixing currency signs and currency codes result in the **General** format even if they correspond to one currency.

Number formatting per field

Pivot tables that were created in other editors can have individual number formatting per field:

	A	B
1	Product ▼	Sum of Sales
2	Bounty	\$2,655.43
3	Lion	\$2,578.24
4	Mars	\$3,795.20
5	Snickers	\$3,740.58
6	Grand Total	\$12,769.45

This type of formatting resets to the automatic one after editing or updating a pivot table.

Part II

Description of functions

Chapter 6

Mathematical functions

ABS

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Returns the absolute value of a number (the value without its sign).

Basic Usage

```
=ABS(A2)
```

	A	B	C
1	value	Result	
2	-18	18	

ABS returns 18, which is the absolute value of the number in A2.

Syntax

```
=ABS(value)
```

value

A number for which you need the absolute value.

Note

If **value** is omitted or refers to an empty cell, **ABS** returns 0 (zero).

Example

Using logical values

Note

ABS treats the logical value TRUE as 1 and the logical value FALSE as 0 (zero).

=ABS(A2)

	A	B	C
1	value	Result	
2	TRUE	1	

ABS treats the logical value TRUE as 1 and returns 1.

ACOS

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Calculates the arccosine.

Note

The arccosine is the inverse of the cosine and returns an angle whose cosine is equal to a specified number.

Basic Usage

```
=ACOS(0.2)
```

	A	B	C
1	1.3694384		
2			

ACOS returns 1.3694384, which is the angle whose cosine is 0.2.

Tip

ACOS returns the result in radians. To convert radians to degrees, multiply the result by 180/pi or use [DEGREES](#).

Syntax

```
=ACOS(value)
```

value

A number not less than -1 and not greater than 1 whose arccosine you want to calculate.

Example

Using cell references

```
=ACOS(A2)
```

	A	B	C
1	value	Result	
2	0.5	1.0471975512	

ACOS returns 1.0471975512, which is the arccosine of the number in A2.

Note

If **value** is an empty value, **ACOS** treats it as 0 (zero).

ACOSH

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Calculates the inverse hyperbolic cosine.

Basic Usage

```
=ACOSH(1)
```

	A	B	C
1	0		
2			

Syntax

```
=ACOSH(value)
```

value

A number greater than or equal to 1 whose inverse hyperbolic cosine you want to calculate.

Example

Using cell references

```
=ACOSH(A2)
```

	A	B	C
1	value	Result	
2	2	1.3169578969	

Note

If **value** is an empty value, **ACOSH** treats it as 0 (zero).

ACOT

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Calculates the arccotangent.

Note

The arccotangent is the inverse of the cotangent and returns an angle whose cotangent is equal to a specified number.

Basic Usage

```
=ACOT(2)
```

	A	B	C
1	0.4636476		
2			

Tip

ACOT returns the result in radians. To convert radians to degrees, multiply the result by 180/pi or use [DEGREES](#).

Syntax

```
=ACOT(value)
```

value

A number whose arccotangent you want to calculate.

Example

Using cell references

```
=ACOT(A2)
```

	A	B	C
1	value	Result	
2	-1	2.3561944902	

ASIN

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Calculates the arcsine.

Note

The arcsine is the inverse of the sine and returns an angle whose sine is equal to a specified number.

Basic Usage

```
=ASIN(0.2)
```

	A	B	C
1	0.2013579208		
2			

Tip

ASIN returns the result in radians. To convert radians to degrees, multiply the result by 180/pi or use [DEGREES](#).

Syntax

```
=ASIN(value)
```

value

Any number in the range from -1 to 1.

Note

If **value** is an empty value, **ASIN** treats it as 0 (zero).

Example

Using cell references

```
=ASIN(A2)
```

	A	B	C
1	value	Result	
2	0.7	0.7753974966	

ATAN

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Calculates the arctangent.

Note

The arctangent is the inverse of the tangent and returns an angle whose tangent is equal to a specified number.

Basic Usage

```
=ATAN(0.2)
```

	A	B	C
1	0.1973955598		
2			

Tip

ATAN returns the result in radians. To convert radians to degrees, multiply the result by 180/pi or use [DEGREES](#).

Syntax

```
=ATAN(value)
```

value

A number whose arctangent you want to calculate.

Note

If **value** is an empty value, **ATAN** treats it as 0 (zero).

Example

Using cell references

```
=ATAN(A2)
```

	A	B	C
1	value	Result	
2	0.7	0.6107259644	

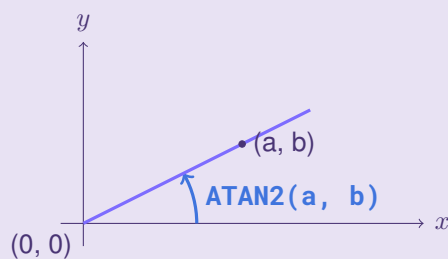
ATAN2

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

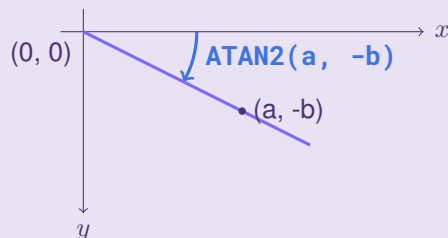
Calculates the arctangent based on x- and y- coordinates.

Note

The arctangent of (a, b) is the angle between the positive x-axis and the ray from the origin (0, 0) to the specified point (a, b):



If the y-coordinate is negative, the angle is measured clockwise (in the opposite direction):



Basic Usage

`=ATAN2(1, 1)`

	A	B	C
1	0.7853982		
2			

Tip

The result is returned in radians. To convert radians to degrees, multiply the result by 180/pi or use [DEGREES](#).

Syntax

=ATAN2(x, y)

x

A number. If **y** is 0 (zero), **x** cannot be 0 (zero).

y

A number. If **x** is 0 (zero), **y** cannot be 0 (zero).

Example

Using cell references

=ATAN2(A2, B2)

	A	B	C	D
1	x	y	Result	
2	-1	1	2.3561945	
3	1	-1	-0.7853982	
4	-1	-1	-2.3561945	
5	1	1	0.7853982	
6	TRUE	FALSE	0	

The formula in C2 returns 2.3561945. Below, you have analogous calculations for various values of **x** and **y**.

Note

ATAN2 treats the logical value TRUE as 1 and the logical value FALSE as 0 (zero).

ATANH

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Calculates the inverse hyperbolic tangent.

Basic Usage

```
=ATANH(0)
```

	A	B	C
1	0		
2			

ATANH returns 0 (zero), which is the inverse hyperbolic tangent of 0 (zero).

Syntax

```
=ATANH(value)
```

value

A number greater than -1 and less than 1 whose inverse hyperbolic tangent you want to calculate.

Note

If **value** is an empty value, **ATANH** treats it as 0 (zero).

Example

Using cell references

```
=ATANH(A2)
```

	A	B	C
1	value	Result	
2	0.5	0.5493061443	

COSH

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Calculates the hyperbolic cosine.

Basic Usage

```
=COSH(0)
```

	A	B	C
1	1		

Syntax

```
=COSH(angle)
```

angle

An angle in radians whose hyperbolic cosine you want to calculate.

Note

If **angle** is an empty value, **COSH** treats it as 0 (zero).

Example

Using cell references

```
=COSH(A2)
```

	A	B	C
1	angle	Result	
2	-3	10.067661996	

COTH

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Calculates the hyperbolic cotangent.

Basic Usage

```
=COTH(4)
```

	A	B	C
1	1.0006711504		

Syntax

```
=COTH(angle)
```

angle

An angle in radians different from 0 (zero) whose hyperbolic cotangent you want to calculate.

Note

The hyperbolic cotangent is the ratio of the hyperbolic sine and cosine. **=SINH(0)** is equal to 0 (zero), so **=COTH(0)** results in the #DIV/0! error. If **angle** is an empty value, **COTH** treats it as 0 (zero).

Example

Using cell references

```
=COTH(A2)
```

	A	B	C
1	angle	Result	
2	-4	-1.0006711504	

CSC

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Calculates the cosecant.

Basic Usage

```
=CSC(1.5)
```

	A	B	C
1	1.0025113		

CSC returns 1.0025113, which is the cosecant of 1.5.

Syntax

```
=CSC(angle)
```

angle

An angle in radians different from 0 (zero) whose cosecant you want to calculate.

Note

The cosecant is the reciprocal of the sine. It means that the cosecant of x is 1 divided by the sine of x.

Example

Using cell references

```
=CSC(A2)
```

	A	B	C
1	angle	Result	
2	10	-1.838164	

CSC returns -1.838164, which is the cosecant for the value in A2.

Note

CSC treats the logical value TRUE as 1 and the logical value FALSE as 0 (zero).

CSCH

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Calculates the hyperbolic cosecant.

Basic Usage

```
=CSCH(3)
```

	A	B	C
1	0.0998215697		

Syntax

```
=CSCH(angle)
```

angle

An angle in radians different from 0 (zero) whose hyperbolic cosecant you want to calculate.

Note

The hyperbolic cosecant is the reciprocal of the hyperbolic sine. **=SINH(0)** is equal to 0 (zero), so **=CSCH(0)** results in the #DIV/0! error.

Example

Using cell references

```
=CSCH(A2)
```

	A	B	C
1	angle	Result	
2	-3	-0.0998215697	

Note

CSCH treats the logical value TRUE as 1 and the logical value FALSE as 0 (zero).

DEGREES

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Converts an angle in radians to an angle in degrees.

Basic Usage

```
=DEGREES(3.1415927)
```

	A	B	C
1	180		

Syntax

```
=DEGREES(angle)
```

angle

An angle in radians that you want to express in degrees.

Examples

Using cell references

```
=DEGREES(A2)
```

	A	B	C
1	angle	Result	
2	6.2831853	360	

Using nested formulas

```
=DEGREES(COS(0))
```

	A	B	C
1	57.295779513		
2			

The formula converts the cosine of 0 (zero) expressed in radians to the value in degrees, which equals 57.295779513.

EXP

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Returns the number e (~2.718) raised to a specified power.

Tip

For the inverse function, the natural logarithm of a number, see [LN](#).

Basic Usage

```
=EXP(1)
```

	A	B	C
1	2.7182818		
2			

EXP raises the number e to the power of 1.

Syntax

```
=EXP(exponent)
```

exponent

A power to which you want to raise the number e.

Examples

Using cell references

```
=EXP(A2)
```

	A	B	C	D
1	exponent	Result		
2	2	7.3890560989		

Using negative **exponent**

=EXP(-2)

	A	B	C
1	0.1353352832		

The formula in A1 is equivalent to **=1/EXP(2)**.

FACTDOUBLE

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Calculates the double factorial of a number.

Note

A double factorial is calculated differently for even and odd numbers.

- Even

$$n!! = n(n-2)(n-4)\dots(4)(2)$$

- Odd

$$n!! = n(n-2)(n-4)\dots(3)(1)$$

Basic Usage

=FACTDOUBLE(7)

	A	B	C
1	105		
2			

FACTDOUBLE returns 105, which is the double factorial of 7.

Syntax

=FACTDOUBLE(value)

value

A number greater than or equal to 0 (zero) whose double factorial you want to calculate.

Note

If **value** is not an integer, its fractional part is truncated.

Examples

Using cell references

```
=FACTDOUBLE(A2)
```

	A	B	C
1	value	Result	
2	5	15	

Note

FACTDOUBLE treats the logical value TRUE as 1 and the logical value FALSE as 0 (zero).

Using nonintegers

```
=FACTDOUBLE(A2)
```

	A	B	C
1	value	Result	
2	5.7	15	

FACTDOUBLE truncates the noninteger in A2 and returns 15, which is the double factorial of 5.

INT

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Rounds a specified number down to the nearest integer.

Basic Usage

```
=INT(9.7)
```

	A	B	C
1	9		
2			

Syntax

```
=INT(value)
```

value

A number that you want to round down to the nearest integer.

Examples

Using cell references

```
=INT(A2)
```

	A	B	C
1	value	Result	
2	5.7	5	
3			

Working with negative numbers

```
=INT(A2)
```

	A	B	C
1	value	Result	
2	-5.7	-6	
3			

LN

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Calculates the natural logarithm of a number.

Note

A natural logarithm is a logarithm to the base of Euler's number (~2.718).

Basic Usage

```
=LN(2.718)
```

	A	B	C
1	0.9998963157		
2			

Syntax

```
=LN(value)
```

value

A number greater than 0 (zero) whose natural logarithm you want to calculate.

Note

LN treats the logical value TRUE as 1 and the logical value FALSE as 0 (zero).

Example

Working with cell references

```
=LN(A2)
```

	A	B	C
1	value	Result	
2	6	1.79175595	

LOG

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Calculates the logarithm of a number to a specified base.

Tip

To calculate a common logarithm (base 10) or a natural logarithm (base e), use [LOG10](#) and [LN](#) respectively.

Basic Usage

```
=LOG(9, 3)
```

	A	B	C
1	2		
2			

LOG uses the formula $\log_3 9$ and returns 2.

Syntax

```
=LOG(value, [base])
```

value

A number greater than 0 (zero) for which you want to calculate the logarithm.

[base] (optional)

A number greater than 0 (zero) that is the logarithm base. If **[base]** is omitted, 10 is used by default.

Example

Working with cell references

```
=LOG(A2, B2)
```

	A	B	C	D
1	value	[base]	Result	
2	81	3	4	
3				

LOG10

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Calculates the base-10 logarithm of a number.

Tip

If you want to specify the base of a logarithm, use [LOG](#).

Basic Usage

```
=LOG10(1000)
```

	A	B	C
1	3		
2			

LOG10 uses the formula $\log_{10}1000$ and returns 3.

Syntax

```
=LOG10(value)
```

value

A number greater than 0 (zero) for which you want to calculate the base-10 logarithm.

Example

Using cell references

```
=LOG10(A2)
```

	A	B	C
1	value	Result	
2	15	1.1760913	

PI

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Returns the number pi.

Basic Usage

```
=PI()
```

	A	B	C
1	3.1415926536		
2			

Syntax

```
=PI()
```

Note

PI does not have arguments.

Example

Using nested formulas

```
=LN(PI())
```

	A	B	C
1	1.1447298858		
2			

The formula returns the natural logarithm of pi.

POWER

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Calculates the result of a number raised to a power.

Note

POWER is equivalent to using the caret sign (^), for example, **=2^3**.

Basic Usage

```
=POWER(2, 3)
```

	A	B	C
1	8		
2			

POWER uses the formula 2^3 and returns 8.

Syntax

```
=POWER(base, exponent)
```

base

A number that you want to raise to a certain power.

exponent

A number for the power.

Examples

Using cell references

```
=POWER(A2, B2)
```

	A	B	C	D
1	base	exponent	Result	
2	4	0.5	2	

Using a negative exponent

```
=POWER(A2, B2)
```

	A	B	C	D
1	base	exponent	Result	
2	15	-5	0.0000013	

PRODUCT

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS, CONSTANTS, ARRAYS

Returns the product of specified arguments.

Basic Usage

```
=PRODUCT(15, 2, 6)
```

	A	B	C
1	180		

PRODUCT calculates the product of 15, 2, and 6 and returns 180.

Syntax

```
=PRODUCT(factor1, [factor2, ...])
```

factor1

The first number or set of numbers that you want to multiply.

[factor2, ...] (optional)

Additional numbers or sets of numbers that you want to multiply.

Note

If only one argument is supplied, **PRODUCT** returns the **factor1** value.

Examples

Using cell references

```
=PRODUCT(A1, A2)
```

	A	B	C
1	13	Result	
2	34	442	

PRODUCT calculates the product of the values in A1 and A2 and returns 442.

Using cell ranges

Note

You can present a set of numbers as a cell range or an array.

=PRODUCT(A1:A3)

	A	B	C
1	23		
2	2	Result	
3	13	598	

PRODUCT calculates the product of the values in A1:A3 and returns 598.

Supplying various types of arguments

=PRODUCT(A2:A4, C5, 2)

	A	B	C	D	E
1				Result	
2	2			240	
3	4				
4	5				
5			3		
6					

PRODUCT calculates the product of the values in A2:A4 and C5, then multiplies that by 2, and returns 240.

Note

PRODUCT ignores empty, logical, and text values within an array or a cell range. If all arguments are ignored, the function returns 0 (zero).

Using arrays

```
=PRODUCT({1, 2; 3, 4}, {1, 2; 3, 4}, A2)
```

	A	B	C
1		Result	
2	price	576	

PRODUCT multiplies the numbers from the arrays, ignores the text in A2, and returns 576.

QUOTIENT

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Returns the result of division without a remainder.

Tip

For regular division operations, use the slash sign (/), for example, **=10/5**.

Basic Usage

```
=QUOTIENT(7, 2)
```

	A	B	C	D
1	Result			
2	3			
3				

QUOTIENT divides 7 by 2 and returns 3 truncating the remainder (the fractional part).

Syntax

```
=QUOTIENT(dividend, divisor)
```

dividend

A number that you want to divide.

divisor

A number other than 0 (zero) by which you want to divide.

Note

QUOTIENT treats the logical value TRUE as 1 and the logical value FALSE as 0 (zero).

Example

Using cell references

```
=QUOTIENT(A2, B2)
```

	A	B	C	D
1	dividend	divisor	Result	
2	-25	3	-8	

RAND

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Returns a random number greater than 0 (zero) and less than 1.

Basic Usage

```
=RAND( )
```

	A	B	C
1	0.182175		

Syntax

```
=RAND( )
```

Warning

RAND has no arguments. If you supply any value to it, **RAND** returns the #VALUE error.

Examples

Extending the range of positive numbers

```
=RAND( ) * 22
```

	A	B	C
1	4.3561202		

The formula returns a random number greater than 0 (zero) and less than 22.

Extending the range to negative numbers

```
=RAND( ) * -90
```

	A	B	C
1	-84.7327835		

The formula returns a random number greater than -90 and less than 0 (zero).

Note

The result changes every time that you introduce a change to the document.

ROUND

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS, CONSTANTS

Rounds a number to a specified number of decimal places using the standard rounding rules.

Basic Usage

```
=ROUND(20.3, 0)
```

	A	B	C
1	20		

ROUND rounds 20.3 to the nearest integer and returns 20.

Syntax

```
=ROUND(value, places)
```

value

A number that you want to round.

places

The number of places to which you want to round.

Tip

To round **value** to the nearest integer, use 0 (zero) for **places**. To round **value** to the left of the decimal point, use negative **places**.

Examples

Using cell references

```
=ROUND(A2, B2)
```


	A	B	C	D
1	value	places	Result	
2	36.909	2	36.91	
3				
4				

ROUND rounds 36.909 to two decimal places and returns 36.91.

Rounding to the left of the decimal point

=ROUND(A2, B2)

	A	B	C	D
1	value	places	Result	
2	46.345	-1	50	
3				
4				

ROUND rounds 46.345 to the nearest multiple of 10 and returns 50.

Note

ROUND rounds numbers according to the standard rules, which means that the next most significant digit (a digit to the right) is considered. If a digit is greater than or equal to 5, the number is rounded up. Otherwise, it is rounded down. If you want to ignore the standard rules, use **ROUNDUP** or **ROUNDDOWN**.

ROUNDDOWN

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Rounds a value down to a specified number of decimal places.

Basic Usage

```
=ROUNDDOWN(12.7, 0)
```

	A	B	C
1	12		

ROUNDDOWN rounds 12.7 to zero decimal places (to the nearest integer) and returns 12.

Syntax

```
=ROUNDDOWN(value, places)
```

value

A number that you want to round down.

places

A number of decimal places to which you want to round down.

Tip

To round **value** to the nearest integer, use 0 (zero) for **places**. To round **value** to the left of the decimal point, use negative **places**.

Examples

Using cell references

```
=ROUNDDOWN(A2, B2)
```

	A	B	C	D
1	value	places	Result	
2	123.456	1	123.4	

ROUNDOWN rounds 123.456 down to one decimal place and returns 123.4.

Rounding down to the left of the decimal point

```
=ROUNDOWN(A2, B2)
```

	A	B	C	D
1	value	places	Result	
2	123.456	-1	120	

ROUNDOWN rounds 123.456 down to the nearest multiple of 10 and returns 120.

ROUNDUP

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Rounds a value up to a specified number of decimal places.

Basic Usage

```
=ROUNDUP(12.7, 0)
```

	A	B	C
1	13		

ROUNDUP rounds 12.7 up to zero decimal places (to the nearest integer) and returns 13.

Syntax

```
=ROUNDUP(value, places)
```

value

A number that you want to round up.

places

A number of decimal places to which you want to round up **value**.

Tip

To round **value** to the nearest integer, use 0 (zero) for **places**. To round **value** to the left of the decimal point, use negative **places**.

Examples

Using cell references

```
=ROUNDUP(A2, B2)
```

	A	B	C	D
1	value	places	Result	
2	123.456	1	123.5	

ROUNDUP rounds 123.456 up to one decimal place and returns 123.5.

Rounding up to the left of the decimal point

```
=ROUNDUP(A2, B2)
```

	A	B	C	D
1	value	places	Result	
2	123.456	-1	130	

ROUNDUP rounds 123.456 up to the nearest multiple of 10 and returns 130.

SEC

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Calculates the secant.

Note

The secant is the reciprocal of the cosine. It means that the secant of x is 1 divided by the cosine of x .

Basic Usage

```
=SEC(1.5)
```

	A	B	C
1	1.0025113		

SEC returns 1.0025113, which is the secant for the angle of 1.5.

Syntax

```
=SEC(angle)
```

angle

An angle in radians for which you want to calculate the secant.

Note

If **angle** is an empty value, **SEC** treats it as 0 (zero).

Example

Using cell references

```
=SEC(A2)
```

	A	B	C
1	angle	Result	
2	-10	-1.1917935	

SECH

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Calculates the hyperbolic secant.

Note

The hyperbolic secant is the ratio of 1 and the hyperbolic cosine.

Basic Usage

```
=SECH(3)
```

	A	B	C
1	0.0993279274		

Syntax

```
=SECH(angle)
```

angle

An angle in radians whose hyperbolic secant you want to calculate.

Note

If **angle** is an empty value, **SECH** treats it as 0 (zero) and returns 1.

Example

Using cell references

```
=SECH(A2)
```


	A	B	C
1	angle	Result	
2	-3	0.0993279274	

SINH

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Calculates the hyperbolic sine.

Basic Usage

```
=SINH(3)
```

	A	B	C
1	10.017874927		

Syntax

```
=SINH(angle)
```

angle

An angle in radians whose hyperbolic sine you want to calculate.

Example

Using cell references

```
=SINH(A2)
```

	A	B	C
1	angle	Result	
2	-2	-3.6268604078	

SQRT

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Calculates the positive square root of a number.

Basic Usage

```
=SQRT(16)
```

	A	B	C
1	4		
2			

SQRT returns the positive square root of 16.

Syntax

```
=SQRT(value)
```

value

A number greater than or equal to 0 (zero) whose positive square root you want to calculate.

Examples

Using cell references

```
=SQRT(A2)
```

	A	B	C
1	value	Result	
2	625	25	

Using negative values

Warning

If you supply a negative number to [SQRT](#), the function returns the #NUM! error. To avoid the error, use [ABS](#).

```
=SQRT(ABS(A2))
```

	A	B	C
1	value	Result	
2	-625	25	

SQRTPI

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Multiplies a number by pi (~3.1415927) and returns the square root of the resulting product.

Basic Usage

```
=SQRTPI(16)
```

	A	B	C
1	7.0898154		
2			

SQRTPI returns the square root of 16*pi.

Syntax

```
=SQRTPI(value)
```

value

A number greater than or equal to 0 (zero) that you want to multiply by pi.

Example

Using cell references

```
=SQRTPI(A2)
```

	A	B	C
1	value	Result	
2	625	44.3113463	

SUBTOTAL

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Uses another function to calculate a subtotal for a set of numbers.

Basic Usage

```
=SUBTOTAL(2, A2:A5)
```

	A	B	C
1	Data	Result	
2	200	4	
3	300		
4	400		
5	500		
6			

SUBTOTAL uses **COUNT** to count the values in A2:A5. Code 2 corresponds to **COUNT**.

Syntax

```
=SUBTOTAL(function_code, range1, [range2, ...])
```

function_code

A numeric code that indicates the function that you want to use for the subtotal. Use codes 1–11 to include hidden cells or 101–111 to ignore them.

range1

The first range or array for which you want to calculate the subtotal.

[range2, ...] (optional)

Additional ranges or arrays that you want to include in the subtotal.

function_code (with hidden cells)	function_code (without hidden cells)	Function
1	101	AVERAGE
2	102	COUNT

3	103	COUNTA
4	104	MAX
5	105	MIN
6	106	PRODUCT
7	107	STDEV.S
8	108	STDEV.P
9	109	SUM
10	110	VAR
11	111	VAR.P

Note

If you use a filter to hide a cell, **SUBTOTAL** excludes the cell from the subtotal regardless of **function_code**. If you filter a range, the function treats all cells hidden manually as hidden by the filter.

Examples

Working with other subtotals

=SUBTOTAL(9, B2:C10)

	A	B	C	D
1		Store 1	Store 2	
2	Socks	287	230	
3	Scarves	346	481	
4	Hats	498	405	
5	Subtotal 1	1131	1116	
6				
7	Bombers	128	61	
8	Parkas	220	392	
9	Trenches	474	476	
10	Subtotal 2	822	929	
11				
12	Result	3998		

SUBTOTAL sums all the numbers in the range except for the subtotals and returns 3998.

Note

SUBTOTAL ignores the results of **AGGREGATE**.

Including hidden cells

=SUBTOTAL(2, A2:A7)

		A	B	C	D
Hidden		1	Data	Result	
3	180	2	120	6	
		4	240		
		5	320		
		6	560		
		7	880		

SUBTOTAL counts the numbers including the number in the hidden cell and returns 6.

Excluding hidden cells

=SUBTOTAL(102, A2:A7)

		A	B	C	D
Hidden		1	Data	Result	
3	180	2	120		
		4	240	5	
		5	320		
		6	560		
		7	880		


SUBTOTAL counts the numbers excluding the number in the hidden cell and returns 5.

Warning

If all data is excluded from the subtotal, **SUBTOTAL** returns the #VALUE! error.

Working with filtration

=SUBTOTAL(2, A2:A7)

		A	B	C	D
	1	Data		Result	
Hidden manually	2	120	4		
3	4	240			
	5	320			
Hidden by filter	6	560			
7	8				

Because of the applied filtration, **SUBTOTAL** excludes the cells hidden both manually and by the filter from the subtotal and returns 4.

Tip

To include a hidden cell in the subtotal, hide the cell after applying the filter.

SUM

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Returns the sum of specified numbers.

Note

SUM is equivalent to the use of the plus sign (+). For example, **=2+3** is equivalent to **=SUM(2, 3)**.

Basic Usage

```
=SUM(1, 2, 3)
```

	A	B	C
1	6		

Syntax

```
=SUM(value1, [value2, ...])
```

value1

The first number that you want to add.

value2 (optional)

Additional numbers that you want to add.

Examples

Using cell ranges

```
=SUM(B2:B5)
```

	A	B	C
1		May, 2020	
2	Tim	34	
3	Tom	23	
4	Tiffany	68	
5	Sam	VAC	
6			
7	Result	125	

SUM ignores the text in B5 and returns 125.

Note

SUM ignores text and logical values in ranges and arrays and converts TRUE and FALSE supplied as single constants to 1 and 0 (zero) respectively.

Using arrays

```
=SUM({TRUE, , "VAC", 12})
```

	A	B	C
1	12		

SUM ignores the Boolean value and the text in the array and returns 12.

Warning

If you enter text directly into the formula, for example, **=SUM("ten")**, **SUM** returns the #VALUE error.

Using various kinds of input

```
=SUM(A2:A3, A4, 1)
```

	A	B	C
1	value	Result	
2		3	20
3		7	
4		9	

SUMIF

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Sums values in an array or a range that meet a single condition.

Tip

To test values against multiple conditions, use [SUMIFS](#).

Basic Usage

```
=SUMIF({1, 22, 22, 3}, 22)
```

	A	B	C
1	44		
2			

SUMIF sums the array elements that meet the condition (22) and returns 44.

Tip

If the condition supplied directly to the formula contains text, mathematical symbols, or logical values, enclose the condition in double quotation marks, for example, "<0".

Syntax

```
=SUMIF(range, criterion, [sum_range])
```

range

A range or an array that you want to test against **criterion**.

criterion

A condition that you want to apply to **range**.

[sum_range] (optional)

A range of values that you want to sum if the corresponding values in **range** meet the specified condition.

Tip

To test and sum values from the same range, omit **[sum_range]**.

Examples

Using cell references

```
=SUMIF(A2:B4, C2)
```

	A	B	C	D	E
1	range		criterion	Result	
2	-15	15	>0	50	
3	-10	10			
4	10	15			
5					

SUMIF returns 50. It is the sum of the values from A2:B4 that meet the specified **criterion** (>0).

Using **[sum_range]**

Warning

Do not use spaces in a series of mathematical symbols supplied as **criterion**, for example, "<=40".

```
=SUMIF(A2:A4, B2, C2:C4)
```

	A	B	C	D	E
1	range	criterion	[sum_range]	Result	
2	-10	>=10	1	5	
3	10		2		
4	15		3		
5					

SUMIF returns 5. It is the sum of the values from C2:C4 that correspond to the values in A2:A4 meeting the specified **criterion** (>=10).

Tip

When you specify **[sum_range]**, ensure that it is of the same size as **range**. Otherwise, **SUMIF** crops or expands **[sum_range]** to make the two ranges equal in size. Example:

Tested range	Specified range to sum	Actually summed range
A1:A2	B1:C2	B1:B2 (cropped)
A1:B2	C1:C2	C1:D2 (expanded)

Using text

```
=SUMIF(A2:A4, B2, C2:C4)
```

	A	B	C	D	E
1	range	criterion	[sum_range]	Result	
2	John	john	145\$	393	
3	Lenny		178\$		
4	john		248\$		
5					

Note

Uppercase and lowercase text values entered as **criterion** are considered equivalent.

SUMIF returns 393. It is the sum of the values from C2:C4 that correspond to the values in A2:A4 meeting the specified **criterion** ("john").

Using wildcards

Note

By default, the question mark (?) and the asterisk (*) are wildcard characters. The question mark represents any single character, and the asterisk represents any sequence of characters. If you want a wildcard character to be treated as a regular character, put the tilde (~) before it (for example, ~?).

=SUMIF(A2:B4, B2, C2:C4)

	A	B	C	D	E
1	range	criterion	[sum_range]	Result	
2	US East	US*	100	300	
3	US West		200		
4	EU East		500		
5					

SUMIF returns 300. It is the sum of the values from C2:C4 that correspond to the values in A2:A4 meeting the specified **criterion** ("US*").

SUMIFS

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Returns the sum of a range of values that meet multiple conditions.

Basic Usage

```
=SUMIFS(A2:A5, B2:B5, 9348, C2:C5, 20513)
```

	A	B	C	D
1	Sold	Shop code	Product code	
2	40	9348	20513	
3	90	9535	25366	
4	100	9949	20513	
5	60	9348	20513	
6				
7	Result	Shop code	Product code	
8	100	9348	20513	
9				

SUMIFS searches for the specified Shop and Product codes (9348 and 20513) in B2:B5 and C2:C5 respectively, sums the corresponding values from the Sold column (40 and 60), and returns 100.

Tip

If the condition supplied directly to the formula contains text or comparison operators, enclose the condition in double quotation marks, for example, "John" or ">10".

Syntax

```
=SUMIFS(sum_range, criteria_range1, criterion1,  
[criteria_range2, criterion2, ...])
```

sum_range

A range or an array with values that you want to sum.

criteria_range1

A range or an array that you want to test. The argument is the same size as **sum_range**.

criterion1

A condition that you want to apply.

[criteria_range2, criterion2, ...] (optional)

Additional ranges or arrays and conditions that you want to test and apply.

Warning

Do not use spaces in a series of mathematical symbols supplied as **criterion**, for example, "<=40".

Examples

Using wildcards

Note

By default, the question mark (?) and the asterisk (*) are wildcard characters. The question mark represents any single character, and the asterisk represents any sequence of characters. If you want a wildcard character to be treated as a regular character, put the tilde (~) before it (for example, ~?).

```
=SUMIFS(A2:A5, B2:B5, B8, C2:C5, C8)
```

	A	B	C	D
1	Sold	Product	Representative	
2	153	Jeans	Marciano	
3	470	Shorts	Clayton	
4	370	Jeans	Clay	
5	488	Jeans	Marciano	
6				
7	Result	Product	Representative	
8	470	Shorts	Clay*	
9				

SUMIFS searches for the specified Product and Representative values ("Shorts" and "Clay*") in B2:B5 and C2:C5 respectively, finds the corresponding value in the Sold column (470), and returns 470.

Using comparison operators

```
=SUMIFS(A2:A5, B2:B5, B8, C2:C5, C8)
```

	A	B	C	D
1	Sales	Quantity	Representative	
2	\$1,530.00	153	Mancini	
3	\$2,350.00	470	Rossi	
4	\$1,850.00	370	Mancini	
5	\$2,440.00	488	Rossi	
6				
7	Result	Quantity	Representative	
8	1850	>200	Mancini	
9				

SUMIFS searches for the specified Quantity and Representative values (>200 and "Mancini") in B2:B5 and C2:C5 respectively, finds the corresponding value in the Sales column (\$1,850.00), and returns 1850.

SUMPRODUCT

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Sums the products of corresponding elements in specified sets of numbers.

Basic Usage

```
=SUMPRODUCT({1, 2}, {10, 20})
```

	A	B	C
1	50		
2			

SUMPRODUCT applies the formula $=1*10+2*20$ and returns 50.

Syntax

```
=SUMPRODUCT(array1, [array2, ...])
```

array1

An array or a range whose values you want to multiply by values from other arrays or ranges.

[array2, ...] (optional)

Other arrays or ranges of the same size as **array1**.

Warning

If you supply arrays or ranges of unequal sizes to **SUMPRODUCT**, the function returns the #VALUE! error.

Examples

Using cell ranges

```
=SUMPRODUCT(A2:A5, B2:B5)
```

	A	B	C	D
1	array1	array2	Result	
2	1	5	50	
3	2	5		
4	3	5		
5	4	5		
6				

SUMPRODUCT applies the formula $=1*5+2*5+3*5+4*5$ and returns 50.

Using text, Boolean, and empty values

Note

If the range or array contains text, Boolean, or empty values, **SUMPRODUCT** treats them as 0 (zero).

=SUMPRODUCT(A2:A5, B2:B5)

	A	B	C	D
1	array1	array2	Result	
2	One	5	20	
3	TRUE	5		
4		5		
5	4	5		
6				

SUMPRODUCT casts the text, Boolean, and empty values in A2, A3, and A4 respectively to 0 (zero), applies the formula $=4*5$, and returns 20.

Note

If a single array or range is supplied, **SUMPRODUCT** returns the sum of numbers in the array or range.

SUMSQ

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Sums the squares of specified numbers.

Basic Usage

```
=SUMSQ(2, 3)
```

	A	B	C
1	13		
2			

SUMSQ sums the squares of 2 and 3 and returns 13.

Syntax

```
=SUMSQ(value1, [value2, ...])
```

value1

A number or a set of numbers that you want you square and add up.

value2 (optional)

Additional numbers that you want you square and add up.

Note

SUMSQ ignores text, Boolean, and empty values inside ranges and arrays.

Examples

Using cell references

```
=SUMSQ(A2, B2)
```

	A	B	C	D
1	value1	value2	Result	
2	1	2	5	

Using cell ranges

```
=SUMSQ(A2:A3, B2:B3)
```

	A	B	C	D
1	value1	value2	SUMSQ	
2	2		8	
3	2			

SUMSQ sums the squares of the values in A2:A3 and ignores the empty values in B2:B3.

Note

If you enter Boolean values directly into the formula, **SUMSQ** converts TRUE and FALSE to 1 and 0 (zero) respectively.

TANH

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Returns the hyperbolic tangent of an angle.

Basic Usage

```
=TANH(3)
```

	A	B	C
1	0.9950547537		

Syntax

```
=TANH(angle)
```

angle

An angle in radians whose hyperbolic tangent you want to calculate.

Example

Using cell references

```
=TANH(A2)
```

	A	B	C
1	angle	Result	
2	-2	-0.9640275801	

Chapter 7

Database functions

DSUM

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, REFERENCE TYPES, COMPARATIVE OPERATORS, COMMON PROBLEMS

Calculates the sum of database values that match specified conditions.

Basic Usage

```
=DSUM(A1:C6, 3, A8:A9)
```

	A	B	C	D	E
1	Product	Supplier	Revenue	Result	
2	Puzzles	Tom	240	776	
3	Checkers	Sam	435		
4	Darts	Peter	360		
5	Checkers	Tom	341		
6	Puzzles	Peter	595		
7					
8	Product				
9	checkers				
10					

DSUM sums the values of the cells from column 3 (C) whose corresponding cells in the Product column contain "Checkers" and returns 776.

Note

DSUM ignores letter case.

Syntax

```
=DSUM(database, field, criteria)
```

database

A cell range with column headers in the uppermost row and a set of related data in the lower rows.

field

A **database** column with values that you want to sum. You can specify **field** as a column number or a column header in double quotation marks. Number 1 matches the first column of **database**.

criteria

A cell range with **database** column headers in the uppermost row and conditions for filtering in each of the lower rows.

Examples

Using column headers as **field**

```
=DSUM(B1:C6, "Revenue", B8:B9)
```

	A	B	C	D	E
1	Product	Supplier	Revenue	Result	
2	Puzzles	Tom	240	1016	
3	Checkers	Sam	435		
4	Darts	Peter	360		
5	Checkers	Tom	341		
6	Puzzles	Peter	595		
7					
8		Supplier			
9		??m			
10					

DSUM sums the values from the cells in the Revenue column whose corresponding cells in the Supplier column contain three-character strings ending in "m" and returns 1016. The result is the total revenue from Tom and Sam.

Note

If **criteria** is an empty cell range or has an empty second row, **DSUM** calculates the sum of all values in **field**.

Applying numbers as **criteria**

```
=DSUM(B1:C6, "Revenue", C8:C9)
```

	A	B	C	D	E
1	Product	Supplier	Revenue	Result	
2	Puzzles	Tom	240	1030	
3	Checkers	Sam	435		
4	Darts	Peter	360		
5	Checkers	Tom	341		
6	Puzzles	Peter	595		
7					
8			Revenue		
9			>=400		
10					

Note

DSUM supports predicates (>, <, >=, <=, =, <>).

DSUM sums the values from the Revenue column that are greater than or equal to 400 and returns 1030.

Using wildcards

=DSUM(A1:C6, "Revenue", A8:A10)

	A	B	C	D	E
1	Product	Supplier	Revenue	Result	
2	Puzzles	Tom	240	1611	
3	Checkers	Sam	435		
4	Darts	Peter	360		
5	Checkers	Tom	341		
6	Puzzles	Peter	595		
7					
8	Product				
9	checkers				
10	p*				

DSUM sums the values in the cells of the Revenue column whose corresponding cells in the Product column contain "Checkers" or text that starts with "p". **DSUM** returns 1611, which is the total revenue from checkers and puzzles.

Note

DSUM supports wildcard characters in **criteria** that contain text. By default, the function treats a question mark (?) as any single character and an asterisk (*) as any sequence of characters. To use an actual question mark or asterisk in **criteria**, type a tilde (~) before the character.

Referring to cells with text

```
=DSUM(A1:B6, 2, A8:A9)
```

	A	B	C	D	E
1	Product	Supplier	Revenue	Result	
2	Puzzles	Tom	240	0	
3	Checkers	Sam	435		
4	Darts	Peter	360		
5	Checkers	Tom	341		
6	Puzzles	Peter	595		
7					
8	Product				
9	checkers				
10					

Note

DSUM ignores text in nonheader rows inside the **field** column.

DSUM returns 0 (zero) because the nonheader rows of column 2 (B) contain only text.

Setting multiple criteria

```
=DSUM(A1:C6, 3, A8:C10)
```

	A	B	C	D	E
1	Product	Supplier	Revenue	Result	
2	Puzzles	Tom	240	675	
3	Checkers	Sam	435		
4	Darts	Peter	360		
5	Checkers	Tom	341		
6	Puzzles	Peter	595		
7					
8	Product	Supplier	Revenue		
9	checkers	??m	>=400		
10	p*		<500		

DSUM calculates the sum of the values from the cells in column 3 (C) whose corresponding cells match conditions specified either in the second or in the third row of **criteria** and returns 675. The result is the combined revenue from puzzles sold by Tom and checkers sold by Sam.

Chapter 8

Logical functions

AND

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, BOOLEAN, OR, COMMON PROBLEMS

Tests multiple logical conditions together.

Note

If all conditions are true, **AND** returns the logical value TRUE. If at least one condition is false, **AND** returns the logical value FALSE.

Basic Usage

Tip

If you enter text into the formula, enclose the text in double quotation marks, for example, "text".

```
=AND(A2>9, B2="socks")
```

	A	B	C	D	E
1				Result	
2	10	socks		TRUE	

AND returns TRUE because both conditions for A2 and B2 are met.

Syntax

```
=AND(logical_expression1, [logical_expression2, ...])
```

logical_expression1

A logical condition that you want to test.

[logical_expression2, ...] (optional)

Additional conditions that you want to test.

Note

Numbers, logical values, or formulas that result in TRUE or FALSE can be used as conditions. For example, **=AND(1, 2, -3)** returns TRUE because positive and negative numbers are treated as TRUE. 0 (zero) is treated as FALSE, so **=AND(0, TRUE)** returns FALSE. Numbers enclosed in double quotation marks, for example, "12", are treated as regular numbers. Text and empty values have no expression in Boolean values and are ignored.

Examples

Using references to Boolean values or formulas with Boolean results

=AND(B1:B5)

	A	B	C	D	E
1	Formula	Cell			
2	=1+1>=2	TRUE			
3	= "q" = "a"	FALSE			
4	= "q" <> "a"	TRUE			
5			Result		
6	=AND(B2:B4)		FALSE		

AND returns FALSE because a cell in B1:B5 contains FALSE. **AND** ignores the text in B1 and the empty value in B5.

Warning

If an argument in your formula has only text in double quotation marks, for example, **=AND("text", TRUE)**, **AND** returns the #VALUE! error.

Combining **AND** and **IF**

Tip

Use **AND** in a nested formula with **IF** to replace TRUE or FALSE with numbers or text.

```
=IF(AND(B2>B3, C2>C3), "Glory!!!", "Shame")
```

	A	B	C	D	E
1		Infantry	Cavalry	Result	
2		500	150	Glory!!!	
3		300	50		

AND returns "Glory!!!" because both specified conditions are met. If at least one condition were not satisfied, the formula would return "Shame".

Combining **AND** and **OR**

```
=AND(D2>55, OR(B2>60, C2>60))
```

	A	B	C	D	E	F
1		Writing	Speaking	Average	Result	
2		20	90	55	FALSE	
3						

AND returns FALSE because condition one is not met, although condition two, specified via **OR**, is fulfilled.

FALSE

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, BOOLEAN, TRUE, COMMON PROBLEMS

Returns the Boolean value FALSE.

Tip

Use **FALSE** for compatibility with other spreadsheets. It is equivalent to the logical value FALSE.

Basic Usage

```
=FALSE()
```

	A	B	C
1	FALSE		
2			

FALSE returns FALSE.

Syntax

```
=FALSE()
```

Example

Using nested functions

```
=IF(A2<10, FALSE())
```

	A	B	C
1	Data	Result	
2	5	FALSE	

The formula returns FALSE because the condition for the value in A2 is met.

IF

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, BOOLEAN, COMMON PROBLEMS

Tests comparisons.

Basic Usage

```
=IF(A2>10, "Pass", "Fail")
```

	A	B	C	D	E
1				Result	
2	12			Pass	

IF returns "Pass" because the condition for the value in A2 is met.

Tip

If you enter text into the formula, enclose it in double quotation marks, for example, "text".

Syntax

```
=IF(condition, if_true, [if_false])
```

condition

A comparison that you want to test.

if_true

A value that you want **IF** to return if your comparison is true.

[if_false] (optional)

A value that you want **IF** to return if your comparison is false. If you omit the argument, it is set to FALSE.

Note

IF ignores letter case when comparing text values.

Examples

Using cell references

```
=IF(C2/B2>0.8, "In stock")
```

	A	B	C	D	E
1	Name	In stock	Sold	Result	
2	Socks	600	300	FALSE	

IF returns FALSE because the condition is not met.

Using references to cells with Boolean values

```
=IF(C2, 10, 30)
```

	A	B	C	D	E
1	Name	In stock	Popular	Result	
2	Socks	600	FALSE	30	

IF returns 30 because the value in C2 is FALSE. If it were TRUE, **IF** would return 10.

Note

IF treats an empty cell as a cell with an empty value that is not equal to 0 (zero).

Using formulas in `if_true` and `[if_false]`

```
=IF(B2>300, C2*0.75, C2*1.1)
```

	A	B	C	D	E
1	Name	In stock	Price	Result	
2	Socks	400	24	18	

IF returns 18, which results from 24×0.75 applied because the condition for B2 is met.

Nesting IF inside `if_true` and `[if_false]`

```
=IF(B2="standard", IF(C2>25, 30, 50), 0)
```

	A	B	C	D	E
1	Album	Edition	Price	Result	
2	Amnesiac	standard	24.99	50	

IF returns 50 applying the following algorithm:

- If the edition is standard and the price is greater than 25, return 30.
- If the edition is standard, but the price is less than 25, return 50.
- If the edition is not standard, return 0 (zero).

Note

When you omit both `if_true` and `[if_false]`, for example, `=IF(A1>B2, ,)`, IF considers them equal to 0 (zero).

For more information, see [Empty value](#).

IFERROR

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, BOOLEAN, COMMON PROBLEMS

Detects and manages broken formulas.

Basic Usage

```
=IFERROR(0/0, "wrong")
```

	A	B	C
1	wrong		
2			

IFERROR returns "wrong" because the specified formula results in an error (#DIV/0!).

Tip

If you enter text into the formula, enclose the text in double quotation marks, for example, "text".

Syntax

```
=IFERROR(value, value_if_error)
```

value

A value to test. If the value is not an error, **IFERROR** returns the value as its result.

value_if_error

A value that you want **IFERROR** to return if an error is detected.

Tip

To make **IFERROR** return an empty value, put a blank space in double quotation marks after the first argument, for example, **=IFERROR(0/0, " ")**.

Examples

Using nested formulas

Tip

Use **IFERROR** to make your worksheet cleaner and easier to understand.

=AVERAGE (B2 : C2)

	A	B	C	D	E
1		Math	English	Average	
2	Stranger	100	100	100	
3	Grizzly	67	56	61.5	
4	Plotter			#DIV/0!	
5					

The formula in D2 is applied to the other cells in the column with autofill.

=IFERROR(AVERAGE(B2:C2), "no marks")

	A	B	C	D	E
1		Math	English	Average	
2	Stranger	100	100	100	
3	Grizzly	67	56	61.5	
4	Plotter			no marks	
5					

The nested formula with **IFERROR** returns "no marks" in D4 instead of the error message in the previous formula.

Tip

Be careful when you check whole ranges and arrays for errors. For single-column ranges, **IFERROR** applies Implicit intersection. To check other ranges properly, use Array formulas.

Simplifying nested formulas in older spreadsheets

Note

IFERROR can help you avoid nested **IF** statements, which you can often find in older spreadsheets.

```
=IF(NOT(ISERROR(AVERAGE(B2:C2))), AVERAGE(B2:C2), "no marks")
```

	A	B	C	D	E
1		Math	English	Average	
2	Stranger	100	100	100	
3	Grizzly	67	56	61.5	
4	Plotter			no marks	
5					

The **IF** formula is equivalent to the **IFERROR** formula of the previous example. **IFERROR** is a shorter substitute for many complicated nested formulas.

Tip

IFERROR supports all types of errors, but to manage the #N/A error specifically, use **IFNA**.

IFNA

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, BOOLEAN, COMMON PROBLEMS

Detects and manages #N/A errors.

Basic Usage

```
=IFNA(B2, "Invalid ID")
```

	A	B	C	D	E
1		Student		Result	
2		#N/A		Invalid ID	

IFNA returns "Invalid ID" because B2 contains the #N/A error.

Tip

If you enter text into the formula, enclose the text in double quotation marks, for example, "text".

Syntax

```
=IFNA(value, value_if_na)
```

value

A value that you want to test. If the value is not a #N/A error, **IFNA** returns this value.

value_if_na

A value that you want **IFNA** to return if a #N/A error is detected.

Tip

IFNA returns an empty value if you put a blank space in double quotation marks after **value**, for example, `=IFNA(A1, " ")`.

Examples

Making your worksheet easier to read by ignoring errors

For example, you have the following formula:

```
=VLOOKUP(D3, A3:B5, 2, FALSE())
```

	A	B	C	D	E	F
1	MENU			ORDER		
2	Pizza	Price		Pizza	Price	Total
3	Chicken	5.99		chicken	5.99	#N/A
4	4 Seasons	8.99		veggie	#N/A	
5	Pepperoni	6.99				

Because of the error in E4, you cannot calculate the total order price. You can solve the problem with **IFNA**.

```
=IFNA(VLOOKUP(D3, A3:B5, 2, FALSE()), "sold out")
```

	A	B	C	D	E	F
1	MENU			ORDER		
2	Pizza	Price		Pizza	Price	Total
3	Chicken	5.99		chicken	5.99	5.99
4	4 Seasons	8.99		veggie	sold out	
5	Pepperoni	6.99				

IFNA returns "sold out", so you can calculate the total order price.

```
=SUM(E:E)
```

	A	B	C	D	E	F
1	MENU			ORDER		
2	Pizza	Price		Pizza	Price	Total
3	Chicken	5.99		chicken	5.99	5.99
4	4 Seasons	8.99		veggie	sold out	
5	Pepperoni	6.99				

Tip

Be careful when you check whole ranges and arrays for errors. If you have a single-column range, your formula is resolved with Implicit intersection. To check other ranges properly, use Array formulas.

Avoiding nested IF formulas found in older spreadsheets

```
=IFNA((B2), "Invalid ID")
```

	A	B	C	D	E
1	ID	Student	Result		
2	#743450	#N/A	Invalid ID		
3	#743412	D. Smith	D. Smith		
4					
5					

IFNA returns "Invalid ID" for the value in B2.

The following formula found in older spreadsheets returns the same result.

```
=IF(NOT(ISNA(B2))), B2, "Invalid ID")
```

You can replace the formula with the shorter **IFNA** equivalent above.

Tip

If you want to manage all types of errors, use IFERROR.

OR

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, BOOLEAN, AND, COMMON PROBLEMS

Tests multiple logical conditions together.

Note

If at least one condition is true, **OR** returns the logical value TRUE. If all conditions are false, **OR** returns the logical value FALSE.

Basic Usage

```
=OR(A2>9, B2="animal")
```

	A	B	C	D
1			Result	
2	10	plant	TRUE	

OR returns TRUE because one of the specified conditions (for A2) is fulfilled.

Tip

If you enter text into the formula, enclose the text in double quotation marks, for example, "text".

Syntax

```
=OR(logical_expression1, [logical_expression2, ...])
```

logical_expression1

A logical condition that you want to test.

[logical_expression2, ...] (optional)

Additional conditions that you want to test.

Examples

Specifying conditions via formulas

Tip

To make your formula shorter, specify conditions through cell references. Any logical value or expression that results in a logical value can be a condition.

=OR(B1:B5)

	A	B	C	D
1	In the formula bar	In a cell		
2	=1+1>=2	TRUE		
3	= "q" = "a"	FALSE		
4	= "q" <> "a"	TRUE		
5				
6		Result		
7	=AND(B1:B5)	TRUE		

OR returns TRUE because some of the specified conditions for B1:B5 are fulfilled. Note that **OR** ignores the text in B1 and the empty value in B5.

Note

In addition to predicates (comparison operators), you can set conditions via logical functions or formulas that return a number. Positive and negative numbers are treated as TRUE, and 0 (zero) is treated as FALSE.

Using nested formulas to return numbers or text

=IF(OR(B2>B3, C2>C3), "Glory!!!", "Shame")

	A	B	C	D
1		Infantry	Cavalry	Result
2	Brown	500	150	Glory!!!
3	Hockney	300	200	

OR returns "Glory!!!" because one of the two specified conditions (B2>B3) is met.

Combining OR with AND

```
=OR(B2>95, C2>95, AND(B2>60, C2>60))
```

	A	B	C	D	E	F
1		Writing	Speaking	Average	Pass	
2	Curtis A.	74	86	80	TRUE	

OR returns TRUE because one of the three specified conditions is met (the values in both B2 and C2 are greater than 60).

Checking conditions inside arrays

```
=OR({TRUE; FALSE})
```

OR returns TRUE.

```
=OR({0; 0})
```

OR returns FALSE.

Note

OR treats numbers in double quotation marks, for example, "12", as valid **logical_expression** values. However, the function ignores text enclosed in double quotation marks, for example, "text".

SWITCH

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, BOOLEAN, COMMON PROBLEMS

Compares a specified value with a list of values and returns a specified result.

Basic Usage

```
=SWITCH(1, 1, "1=1", 2, "1=2")
```

	A	B	C
1	1=1		
2			

SWITCH uses the formula: if 1 equals 1, return 1=1, and if 1 equals 2, return 1=2. **SWITCH** returns 1=1.

Tip

If you enter text into the formula, enclose the text in double quotation marks, for example, "text".

Syntax

```
=SWITCH(expression, value1, result1, [value2, result2, ...],  
[default])
```

expression

A value that you want to compare with a list of values.

valueN

A value that you want to compare with **expression**.

resultN

A value that is returned if a corresponding **valueN** is equal to **expression**.

[default] (optional)

A value that is returned if no matches are found. If omitted, the argument is set to #N/A.

Note

The maximum number of arguments for **SWITCH** is limited to 254.

Examples

Using cell references

```
=SWITCH(A2, "D. J.", "Dan James", "D. I.", "Danny Ings", "H. K.", "Harry Kane", "Unknown")
```

	A	B	C
1	Short	Result	
2	D. J.	Dan James	

SWITCH returns "Dan James", which corresponds to the specified value match in A2.

Tip

Specify **expression** as a reference to make your **SWITCH** formula easier to reuse.

Reusing formulas with autofill

Now, you can change **expression** without editing your formula above. Enter values into the cells under A2, and autofill the cells under your formula from B2.

```
=SWITCH(A2, "D. J.", "Dan James", "D. I.", "Danny Ings", "H. K.", "Harry Kane", "Unknown")
```

	A	B	C
1	Short	Result	
2	D. J.	Dan James	
3	D. I.	Danny Ings	
4	H. K.	Harry Kane	
5	K. P.	Unknown	

In B2:B5, **SWITCH** returns values specified for matches in A2:A5.

Simplifying nested formulas

Note

SWITCH can help you avoid nested **IF** formulas.

```
=IF(A2=5, "cake", IF(A2=4, "candy", IF(A2=3, "apple", "nothing")))
```

```
=SWITCH(A2, 5, "cake", 4, "candy", 3, "apple", "nothing")
```

Both of the formulas return "cake", "candy", or "apple" if A2 equals 5, 4, or 3 respectively. Otherwise, they return "nothing".

	A	B	C
1	Points	Result	
2	2	nothing	
3	3	apple	
4	4	candy	
5	5	cake	

In B2:B5, **SWITCH** returns values specified for matches in A2:A5.

Managing errors

```
=SWITCH(A2, #DIV/0!, "division by zero", #N/A, "not found", "Unknown error")
```

	A	B	C
1		Result	
2	#N/A	not found	

SWITCH returns "not found" for the value match in A2.

TRUE

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, BOOLEAN, FALSE, COMMON PROBLEMS

Returns the Boolean value TRUE.

Tip

Use **TRUE** for compatibility with other spreadsheets. It is equivalent to the logical value TRUE.

Basic Usage

```
=TRUE()
```

	A	B	C
1	TRUE		
2			

TRUE returns TRUE.

Syntax

```
=TRUE()
```

Using nested functions

```
=IF(A2<10, TRUE())
```

	A	B	C
1	Data	Result	
2	5	TRUE	

The formula returns TRUE because the condition for the value in A2 is met.

Chapter 9

Date and time functions

DATE

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, DATE AND TIME, COMMON PROBLEMS

Composes a date based on the specified values of day, month, and year.

Basic Usage

```
=DATE(2021, 5, 27)
```

	A	B	C
1	5/27/2021		
2			

Tip

Use **DATE** to ensure that date values are formatted correctly.

Syntax

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

year

A number that specifies the year component in a date.

month

A number that specifies the month component in a date.

day

A number that specifies the day component in a date.

Warning

The date and time functions support dates from 1/1/0001 to 12/31/9999. If the specified date falls outside the supported range, **DATE** returns the #VALUE! error.

Examples

Using nonstandard **day** values

Note

If **day** is less than 1 or greater than the number of days in the specified **month**, **DATE** recalculates the supplied values and returns the result in the correct date format.

```
=DATE(1882, 6, 34)
```

	A	B	C
1	7/4/1882		
2			

The specified **day** is greater than the number of days in June. As a result, **DATE** adds 34 days to the specified **month**, which is June, and returns 7/4/1882.

Using nonstandard **month** values

Note

If **month** is less than 1 or greater than 12, **DATE** recalculates the specified values and returns the result in the correct date format.

```
=DATE(2001, 14, 22)
```

	A	B	C
1	2/22/2002		
2			

The specified **month** is greater than 12. As a result, **DATE** adds 14 months to the specified **year**, which is 2001, and returns 2/22/2002.

Using nonintegers

Note

If the input is not an integer, **DATE** truncates its fractional part.

```
=DATE(2008, 5.3, 9.6)
```

	A	B	C
1	5/9/2008		
2			

DATE discards the fractional part of the specified **day** and **month** and converts the received numbers into a date, which is 5/9/2008.

Using negative numbers

```
=DATE(2016, 9, -1)
```

	A	B	C
1	8/30/2016		
2			

DATE subtracts 1 day from the supplied month, September, and returns 8/30/2016.

```
=DATE(1997, -3, 2)
```

	A	B	C
1	9/2/1996		
2			

DATE subtracts 3 months from the supplied year, 1997, and returns 9/2/1996.

DATEVALUE

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, DATE AND TIME, COMMON PROBLEMS

Returns the numeric value of a date specified in a common format.

Basic Usage

```
=DATEVALUE("05-29-2011")
```

	A	B	C
1	40692		
2			

Tip

If you enter a date directly into the formula, use double quotation marks, for example, "8 March 2020".

Syntax

```
=DATEVALUE(date_string)
```

date_string

A date in a common format.

Warning

The date and time functions support dates from 1/1/0001 to 12/31/9999. If **date_string** falls outside the supported date range, **DATEVALUE** returns the #VALUE! error.

Examples

Using nested functions

Tip

To ensure that **date_string** is supplied correctly, use [DATE](#).

```
=DATEVALUE(DATE (2009, 1, 3))
```

	A	B	C
1	39816		
2			

DATEVALUE returns 39816, which is the numeric value of the date supplied via [DATE](#).

Using cell references

```
=DATEVALUE(A2)
```

	A	B	C
1	Date	Result	
2	3/22/2011	40624	
3			

DATEVALUE returns 40624, which is the numeric value of the date in A2.

DAY

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, DATE AND TIME, COMMON PROBLEMS

Returns the day component of a specified date.

Basic Usage

```
=DAY("08 June 2021")
```

	A	B	C
1	8		
2			

Tip

If you enter a date directly into the formula, use double quotation marks, for example, "1/10/2017".

Syntax

```
=DAY(date)
```

date

A date from which you want to retrieve the day component.

Warning

The date and time functions support dates from 1/1/0001 to 12/31/9999. If **date** falls outside the supported date range, **DAY** returns the #VALUE! error.

Examples

Using nested functions

Tip

To ensure that **date** is supplied correctly, use [DATE](#).

```
=DAY(DATE(2003, 8, 29))
```

	A	B	C
1	8		
2			

DAY returns 8, which is the day of the date supplied via **DATE**.

```
=DAY(TODAY())
```

	A	B	C
1	25		
2			

DAY returns 25, which is the day of the current date. The result varies depending on the current date.

Using cell references

```
=DAY(A2)
```

	A	B	C
1	date	Result	
2	2/10/1990	10	
3			

DAY returns 10, which is the day of the date in A2.

Using numeric date values

```
=DAY(32898)
```

	A	B	C
1	25		
2			

DAY returns 25. It is the day of the specified numeric date value, which corresponds to January 25, 1990.

EDATE

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, DATE AND TIME, COMMON PROBLEMS

Adds or subtracts a specified number of months to or from a date and returns its numeric value.

Note

A numeric date value represents a date as the number of days that passed after December 30, 1899, which is the editor's base date.

Basic Usage

```
=EDATE("10-Jan-1991", 4)
```

	A	B	C
1	42821		
2			

Tip

If you enter a date directly into the formula, use double quotation marks, for example, "5/10/1991".

Syntax

```
=EDATE(start_date, months)
```

start_date

The initial date from which the calculation starts.

months

The number of months that you want to add to or subtract from **start_date**. The negative **months** argument subtracts months. The positive **months** argument adds months.

Warning

The date and time functions support dates from 1/1/0001 to 12/31/9999. If **start_date** falls outside the supported date range, **EDATE** returns the #VALUE! error.

Examples

Using nested functions

Tip

To ensure that **start_date** is supplied correctly, use **DATE**.

```
=EDATE(DATE(1877, 4, 21), -2)
```

	A	B	C
1	-8347		
2			

EDATE subtracts two months from the date supplied via **DATE** and returns -8347. The result is the numeric value of 2/21/1877.

```
=EDATE(NOW(), -3)
```

	A	B	C
1	42821		
2			

EDATE subtracts 3 months from the current date, 6/27/2017, and returns the numeric value of 3/27/2017. The result varies depending on the current date.

Using cell references

```
=EDATE(A2, 2)
```

	A	B	C
1	Data	Result	
2	4/18/2002	37425	
3			

EDATE adds two months to the date in A2 and returns 37425. The result is the numeric value of 6/18/2002.

EOMONTH

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, DATE AND TIME, COMMON PROBLEMS

Adds or subtracts a specified number of months to or from a given date. Returns the numeric date value that corresponds to the last calendar day of the resulting month.

Note

EOMONTH considers only integers and truncates the fractional parts of supplied numbers.

Basic Usage

```
=EOMONTH("11 Jun 2021", +1)
```

	A	B	C
1	44408		
2			

Tip

If you enter a date directly into the formula, use double quotation marks, for example, "5/10/1991".

Syntax

```
=EOMONTH(start_date, months)
```

start_date

The initial date from which the calculation starts.

months

The number of months that you want to add to or subtract from **start_date**. The negative **months** argument subtracts months. The positive **months** argument adds months.

Warning

The date and time functions support dates from 1/1/0001 to 12/31/9999. If **start_date** falls outside the supported date range, **EOMONTH** returns the #VALUE! error.

Examples

Using nested functions

Tip

To ensure that **start_date** is supplied correctly, use **DATE**.

```
=EOMONTH(DATE(2011, 2, 12), -3)
```

	A	B	C
1	40512		
2			

EOMONTH subtracts three months from the date supplied via **DATE** and returns 40512. The result is the numeric value of the last calendar day in November.

Using cell references

```
=EOMONTH(A3, 2)
```

	A	B	C
1	Data	Result	
2	11/01/99	36556	
3			

EOMONTH adds two months to the date in A2 and returns 36556. The result is the numeric value of the last calendar day in January.

HOUR

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, DATE AND TIME, COMMON PROBLEMS

Returns the hour component of a specified time value. The result is an integer in the range from 0 (zero) to 23.

Basic Usage

```
=HOUR("17:21")
```

	A	B	C
1	17		
2			

Tip

If you enter a time value directly into the formula, use double quotation marks, for example, "10:37:00 AM" or "17:21".

Syntax

```
=HOUR(time)
```

time

A time value from which you want to retrieve the hour component.

Examples

Using the 24:00 time value

```
=HOUR("24:00")
```

	A	B	C
1	0		
2			

HOUR returns 0 (zero) because the specified time is equivalent to 00:00:00.

Using nested functions

```
=HOUR(NOW())
```

	A	B	C
1	19		
2			

The formula returns 19, which is the hour component of the current time (19:08:07). The result varies based on the current time.

```
=HOUR(TIME(12, 34, 50))
```

	A	B	C
1	12		
2			

The formula returns 12 because **TIME** assembles 12:34:50 PM.

Using cell references

```
=HOUR(A2)
```

	A	B	C
1	time	Result	
2	8:37:00 AM	8	
3			

HOUR returns 8, which is the hour component of the time in A2.

ISOWEEKNUM

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, DATE AND TIME, COMMON PROBLEMS

Returns the ISO week number for a specified date. The result ranges from 1 to 54.

Note

ISOWEEKNUM returns a week number based on the ISO-8601 standard. According to this standard, the first day of the week is Monday, and the first week of the year is the first week that contains a Thursday.

Basic Usage

```
=ISOWEEKNUM("Jan-31-2011")
```

	A	B	C
1	5		
2			

Tip

If you enter a date directly into the formula, use double quotation marks, for example, "1/10/2017".

Syntax

```
=ISOWEEKNUM(date)
```

date

A date for which you want to retrieve the ISO week number.

Warning

The date and time functions support dates from 1/1/0001 to 12/31/9999. If **date** falls outside the supported date range, **ISOWEEKNUM** returns the #VALUE! error.

Examples

Using nested functions

Tip

To ensure that **date** is supplied correctly, use **DATE**.

```
=ISOWEEKNUM(DATE(2010, 4, 25))
```

	A	B	C
1	16		
2			

ISOWEEKNUM returns 16, which is the ISO week number for the date supplied via **DATE**.

```
=ISOWEEKNUM(TODAY())
```

	A	B	C
1	2		
2			

The formula returns 2, which is the ISO week number for the current date, 01/13/2022. The result varies depending on the current date.

Using cell references

```
=ISOWEEKNUM(A2)
```

	A	B	C
1	date	Result	
2	4/23/1997	17	
3			

ISOWEEKNUM returns 17, which is the ISO week number for the date in A2.

MINUTE

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, DATE AND TIME, COMMON PROBLEMS

Returns the minute component of a specified time value. The result is an integer in the range from 0 (zero) to 59.

Basic Usage

```
=MINUTE("12:45")
```

	A	B	C
1	45		
2			

Tip

If you enter a time value directly into the formula, use double quotation marks, for example, "10:37:00 AM" or "17:21".

Syntax

```
=MINUTE(time)
```

time

A time value from which you want to retrieve the minute component.

Examples

Using nested functions

```
=MINUTE(NOW())
```

	A	B	C
1	18		
2			

MINUTE extracts the minute component from the current time, which is 5:18:07 PM, and returns 18. The result varies based on the current time.

Using cell references

```
=MINUTE(A2)
```

	A	B	C
1	time	Result	
2	9:25:00 AM	25	
3			

MINUTE extracts the minute component from the time in A2 and returns 25.

MONTH

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, DATE AND TIME, COMMON PROBLEMS

Returns the month component of a specified date. The result is an integer in the range from 1 to 12.

Basic Usage

```
=MONTH("08 Jul 2016")
```

	A	B	C
1	7		
2			

Tip

If you enter a date directly into the formula, use double quotation marks, for example, "5/10/1991".

Syntax

```
=MONTH(date)
```

date

A date from which you want to retrieve the month component.

Warning

The date and time functions support dates from 1/1/0001 to 12/31/9999. If **date** falls outside the supported date range, **MONTH** returns the #VALUE! error.

Examples

Using nested functions

Tip

To ensure that **date** is supplied correctly, use [DATE](#).

```
=MONTH(DATE(1994, 3, 15))
```

	A	B	C
1	3		
2			

MONTH returns 3, which is the month of the date supplied via **DATE**.

Using cell references

```
=MONTH(A2)
```

	A	B	C
1	date	Result	
2	5/12/2000	5	
3			

MONTH returns 5, which is the month of the date in A2.

Using numeric date values

```
=MONTH(32898)
```

	A	B	C
1	1		
2			

MONTH returns 1. It is the month of the specified numeric date value, which corresponds to January 25, 1990.

NOW

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, DATE AND TIME, COMMON PROBLEMS

Returns the current date and time.

Basic Usage

```
=NOW()
```

	A	B	C
1	1/8/2017 1:42:10 PM		
2			

Note

In **NOW**, the result varies based on the current date and time and is updated every time that you open or recalculate a worksheet.

Syntax

```
=NOW()
```

Note

NOW allows you to calculate the date and time value that falls a specified number of days or hours before or after the current date and time. In the formula, negative numbers subtract from and positive numbers add to the current date and time.

Examples

Adding days

```
=NOW()+1
```

	A	B	C
1	1/9/2017 1:42:10 PM		
2			

The formula calculates the date and time value that falls one day (24 hours) after the current date and time and returns 1/9/2017 1:42:10 PM. The current date and time value is 1/8/2017 1:42:10 PM.

Subtracting days and hours

Note

In nonintegers, digits to the left of the decimal point represent the number of days, whereas digits to the right specify the value for hours (0.5 stands for 12 hours).

=NOW() -1.5

	A	B	C
1	1/7/2017 1:42:10 AM		
2			

The formula subtracts 1 day and 12 hours from the current date and time (1/8/2017 1:42:10 PM) and returns 1/7/2017 1:42:10 AM.

TODAY

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, DATE AND TIME, COMMON PROBLEMS

Returns the current date.

Basic Usage

```
=TODAY()
```

	A	B	C
1	2/23/2021		
2			

TODAY returns the current date.

Syntax

```
=TODAY()
```

Note

In **TODAY**, the result varies based on the current date.

Examples

Calculating a date that comes before or after the current date

```
=TODAY() -1
```

	A	B	C
1	2/22/2021		
2			

The formula returns yesterday's date.

=TODAY()+1

	A	B	C
1	01/14/2022		
2			

The formula returns tomorrow's date.

Using nested functions

=YEAR(TODAY())

	A	B	C
1	2021		
2			

YEAR returns the year component of the current date.

YEAR

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, DATE AND TIME, COMMON PROBLEMS

Returns the year component of a specified date.

Basic Usage

```
=YEAR("5/31/2022")
```

	A	B	C
1	2022		
2			

Tip

If you enter a date directly into the formula, use double quotation marks, for example, "5/10/1991".

Syntax

```
=YEAR(date)
```

date

A date from which you want to retrieve the year component.

Warning

The date and time functions support dates from 1/1/0001 to 12/31/9999. If **date** falls outside the supported date range, **YEAR** returns the #VALUE! error.

Examples

Using nested functions

Tip

To ensure that **date** is supplied correctly, use [DATE](#).

```
=YEAR(DATE(2011, 10, 10))
```

	A	B	C
1	2011		
2			

YEAR returns 2011, which is the year of the date supplied via **DATE**.

Using cell references

```
=YEAR(A2)
```

	A	B	C
1	date	Result	
2	8/12/1999	1999	
3			

YEAR returns 1999, which is the year of the date in A2.

Using numeric date values

```
=YEAR(32898)
```

	A	B	C
1	1990		
2			

YEAR returns 1990. It is the year of the specified numeric date value, which corresponds to January 25, 1990.

Chapter 10

Statistical functions

AVERAGE

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Returns the average (arithmetic mean) of numbers in a data set.

Basic Usage

```
=AVERAGE(A2:A5)
```

	A	B	C
1	Data	Result	
2	48	43	
3	six		
4	66		
5	15		
6			

Syntax

```
=AVERAGE(value1, [value2, ...])
```

value1

The first number, array, cell reference, or range for which you want to calculate the average.

[value2, ...] (optional)

Additional numbers, arrays, cell references, or ranges for which you want to calculate the average.

Warning

AVERAGE ignores logical, text, and empty values supplied to an array or a cell and returns the #DIV/0! error if all **value** arguments are not numbers.

Examples

Working with arrays

```
=AVERAGE({3, 5, 6}, {-7, "5", 6})
```

	A	B	C
1	3		
2			

AVERAGE returns 3, which is the average of the numbers in the arrays.

Working with cell references

```
=AVERAGE(A2, A3)
```

	A	B	C
1	Data	Result	
2	66	40.5	
3	15		
4			

AVERAGE returns 40.5, which is the average of the numbers in A2 and A3.

COUNT

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Returns the count of numbers in a data set.

Basic Usage

```
=COUNT(A2:A5)
```

	A	B	C
1	Data	Result	
2	7	4	
3	2		
4	16		
5	5		
6			

Syntax

```
=COUNT(value1, [value2, ...])
```

value1

The first value, array, cell reference, or range which or in which you want to count.

[value2, ...] (optional)

Additional values, arrays, cell references, or ranges which or in which you want to count.

Note

COUNT considers all formatted numeric values such as date, time, percentage and currency values. The function ignores text and references to empty cells. Logical values are counted if they are supplied directly to the formula.

Example

Working with cell references and arrays

```
=COUNT(A2, A3, A5, A6 {"4", 6})
```

	A	B	C
1	Data	Result	
2		3	
3	value		
4			
5	TRUE		
6	01/01/1990		

COUNT returns 3. A2, which is an empty cell, the text in A3, and the logical value in A5 are ignored.

COUNTA

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Returns the number of values in a data set.

Note

COUNTA considers any type of values but ignores empty cells.

Basic Usage

```
=COUNTA(A2:A6)
```

	A	B	C
1	Data	Result	
2	7	4	
3	7/29/2020		
4	amount		
5			
6	5.5		
7			

Syntax

```
=COUNTA(value1, [value2, ...])
```

value1

The first value, array, cell reference, or range which or in which you want to count.

[value2, ...] (optional)

Additional values, arrays, cell references, or ranges which or in which you want to count.

Examples

Using cell references

```
=COUNTA(A2, A3)
```

	A	B	C
1	Data	Result	
2	25	2	
3	-		
4			

COUNTA counts the values in A2 and A3 and returns 2.

Counting empty strings

```
=COUNTA("price", A2:A4, "")
```

	A	B	C
1	Data	Result	
2		4	
3	\$10		
4	TRUE		
5			

COUNTA counts the values in the specified data set and returns 4. The empty string ("") supplied directly to the formula is counted, but A2, which is an empty cell, is ignored.

COUNTBLANK

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Returns the number of empty values in an array or a range.

Basic Usage

```
=COUNTBLANK(A2:A4)
```

	A	B	C
1	Data	Result	
2		2	
3	14		
4			

COUNTBLANK counts empty values in the specified range and returns 2.

Note

COUNTBLANK also counts cells that contain formulas returning an empty string ("").

Syntax

```
=COUNTBLANK(range)
```

range

An array or a range in which you want to count empty values.

Example

Working with arrays

```
=COUNTBLANK({"", , "", 5})
```


	A	B	C
1	3		
2			

COUNTBLANK counts empty values and empty strings in the array and returns 3.

COUNTIF

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Calculates the number of cells with values that meet a specified condition.

Basic Usage

```
=COUNTIF(A2:A7, "Draughts")
```

	A	B	C
1	range	Result	
2	Cards	2	
3	Draughts		
4	Darts		
5	draughts		
6	Puzzles		
7	Billiards		
8			

In A2:A7, **COUNTIF** counts cells with "Draughts" and returns 2.

Note

COUNTIF regards uppercase and lowercase text values as equivalent.

Syntax

```
=COUNTIF(range, criterion)
```

range

A range in which you want to count.

criterion

A value that determines which cells you want to count.

Tip

Enclose text, mathematical symbols, and wildcard characters supplied as **criterion** in double quotation marks, for example, "Cards", "<=2", and "????s".

Examples

Using wildcard characters as **criterion**

```
=COUNTIF(A2:A7, "*s")
```

	A	B	C
1	range	Result	
2	Cards	5	
3	Draughts		
4	Darts		
5	Dice		
6	Puzzles		
7	Billiards		
8			

In A2:A7, **COUNTIF** counts cells with values that end in "s" and returns 5.

Tip

Apply wildcard characters as **criterion** or part of **criterion** to filter data based on a partial match. For example, the asterisk (*) denotes zero or more characters anywhere in a string, and the question mark (?) matches a single character in a specific position.

Using question mark wildcards

```
=COUNTIF(A2:A7, "????s")
```

	A	B	C
1	range	Result	
2	Cards	2	
3	Draughts		
4	Darts		
5	Dice		
6	Puzzles		
7	Billiards		
8			

In A2:A7, **COUNTIF** counts five-character values that end in "s" and returns 2.

Using mathematical symbols as **criterion**

```
=COUNTIF(A2:A7, ">=5")
```

	A	B	C
1	range	Result	
2	4	3	
3	5		
4	8		
5	2		
6	10		
7	1		
8			

In A2:A7, **COUNTIF** counts cells with values greater than or equal to 5 and returns 3.

Warning

Do not use spaces in a series of mathematical symbols supplied as **criterion**, for example, "<=40".

Using arrays as **criterion**

Tip

COUNTIF allows you to apply more than one condition to the same range. To do this, enter the conditions as an array.

```
=COUNTIF(A2:A7, {4, 2, 5})
```

	A	B	C
1	range	Result	
2	4	3	
3	5		
4	8		
5	2		
6	10		
7	1		
8			

In A2:A7, **COUNTIF** counts cells with values that match the conditions specified in the array and returns 3.

Using nested functions as **criterion**

```
=COUNTIF(A2:A7, DATE(2008,1,1))
```

	A	B	C
1	range	Result	
2	01/01/2008	2	
3	01/04/2009		
4	01/04/2009		
5	06/01/2008		
6	01/01/2008		
7	05/12/2008		
8			

In A2:A7, **COUNTIF** counts cells with the date supplied via **DATE** and returns 2.

COUNTIFS

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Returns the number of times that multiple conditions are met across ranges or arrays.

Basic Usage

```
=COUNTIFS(A2:A6, "*s", B2:B6, "<01/04/2009")
```

	A	B	C	D
1	Data	Sales dates	Result	
2	Oranges	01/01/2008	2	
3	Bananas	01/04/2009		
4	Mangoes	01/04/2009		
5	Grapefruit	01/01/2008		
6	Apples	01/01/2009		
7				

COUNTIFS returns 2. The result is the number of rows that contain both values ending in "s" in A2:A6 and sales dates that precede 01/04/2009 in B2:B6.

Warning

Ensure that ranges and arrays involved in the calculation are equal in size. Otherwise, **COUNTIFS** returns the #VALUE! error.

Syntax

```
=COUNTIFS(criteria_range1, criterion1, [criteria_range2, criterion2, ...])
```

criteria_range1

A range in which you want to count.

criterion1

A value that determines which cells you want to count.

[criteria_range2, criterion2, ...] (optional)

Additional ranges and values. The maximum number of arguments is limited to 254.

Tip

Enclose text, mathematical symbols, and wildcard characters supplied as **criterion** in double quotation marks, for example, "Cards", "<=2", and "????s".

Examples

Using wildcard characters as **criterion**

```
=COUNTIFS(A2:A6, "*s", B2:B6, "<70")
```

	A	B	C	D
1	Data	Sales	Result	
2	Oranges	71	3	
3	Bananas	32		
4	Mangoes	69		
5	Grapefruit	65		
6	Apples	45		
7				

COUNTIFS returns 3. The result is the number of rows that contain both values ending in "s" in A2:A6 and sales figures less than 70 in B2:B6.

Tip

Apply wildcard characters as **criterion** or part of **criterion** to filter data based on a partial match. For example, the asterisk (*) denotes zero or more characters, and the question mark (?) matches a single character in a specific position.

Using mathematical symbols as **criterion**

```
=COUNTIFS(A2:A6, "??????s", B2:B6, ">=40")
```

	A	B	C	D
1	Data	Sales	Result	
2	Oranges	71	2	
3	Bananas	32		
4	Mangoes	69		
5	Grapefruit	65		
6	Apples	45		
7				

COUNTIFS returns 2. The result is the number of rows that contain both seven-character values ending in "s" in A2:A6 and sales figures greater than or equal to 40 in B2:B6.

Warning

Do not use spaces in a series of mathematical symbols supplied as **criterion**, for example, "<=40".

Using arrays as **criterion**

Note

COUNTIFS allows you to apply more than one condition to the same range. To do this, enter the conditions as an array.

COUNTIFS(A2:A6, {"Oranges", "Mangoes"}, B2:B6, {71, 61})

	A	B	C	D
1	Data	Sales	Result	
2	Oranges	71	1	
3	Bananas	32		
4	Mangoes	69		
5	Grapefruit	65		
6	Apples	45		
7				

COUNTIFS returns 1. The result is the number of rows that contain both "Oranges" or "Mangoes" in A2:A6 and sales figures equal to 71 or 61 in B2:B6.

Using nested functions as **criterion**

```
=COUNTIFS(A2:A6,"mangoes", B2:B6, DATE(2009, 1, 4))
```

	A	B	C	D
1	Data	Sales Dates	Result	
2	Oranges	01/01/2008	1	
3	Bananas	01/04/2009		
4	Mangoes	01/04/2009		
5	Grapefruit	01/01/2008		
6	Mangoes	01/07/2009		
7				

COUNTIFS returns 1. The result is the number of rows that contain both "mangoes" in A2:A6 and the date supplied via **DATE** in B2:B6.

Note

COUNTIFS regards uppercase and lowercase text values as equivalent.

F.DIST.RT

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Calculates the right-tailed F probability distribution (degree of diversity) for two sets of data.

Note

FDIST and **F.DIST.RT** are identical. If you enter **FDIST**, it is automatically replaced with **F.DIST.RT**.

Basic Usage

```
=F.DIST.RT(2/4, 9, 9)
```

	A	B	C
1	0.841761297		
2			

Syntax

```
=F.DIST.RT(x, freedom_degrees1, freedom_degrees2)
```

x

A positive value at which you want to calculate the distribution.

freedom_degrees1

A value equal to or greater than 1 that represents the numerator of the number of degrees of freedom.

freedom_degrees2

A value equal to or greater than 1 that represents the denominator of the number of degrees of freedom.

Warning

The largest allowed value for the **freedom_degrees** arguments is 10000000000 (10^{10}). For larger values, **F.DIST.RT** returns the #NUM! error.

Example

Using cell references

```
=F.DIST.RT(A2, A3, A4)
```

	A	B	C
1	Data	Result	
2	10.97	0.000451429	
3	2		
4	23		
5			

F.DIST.RT calculates the right-tailed F probability distribution based on the data in A2, A3, and A4 and returns 0.000451429.

Note

If **freedom_degrees1** or **freedom_degrees2** is not an integer, its fractional part is truncated.

F.DIST

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Calculates the left-tailed F probability distribution (degree of diversity) for two sets of data.

Basic Usage

```
=F.DIST(5, 3, 2, FALSE)
```

	A	B	C
1	0.029252692		
2			

F.DIST calculates the probability density function for the specified data and returns 0.029252692.

Syntax

```
=F.DIST(x, freedom_degrees1, freedom_degrees2, cumulative)
```

x

A positive value at which you want to calculate the distribution.

freedom_degrees1

A value equal to or greater than 1 that represents the numerator of the number of degrees of freedom.

freedom_degrees2

A value equal to or greater than 1 that represents the denominator of the number of degrees of freedom.

cumulative

A logical value that indicates whether you want to calculate the cumulative distribution function (TRUE) or probability density function (FALSE).

Example

Using cell references

```
=F.DIST(A2, A3, A4, TRUE)
```

	A	B	C
1	Data	Result	
2	20.76	0.984979066	
3	8		
4	3		
5			

F.DIST calculates the left-tailed F cumulative distribution function for the data in A2, A3, and A4 and returns 0.984979066.

Note

If **freedom_degrees1** or **freedom_degrees2** is a noninteger, its fractional part is truncated.

FISHER

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Calculates the Fisher transformation for a specified value.

Tip

Use **FISHER** to test hypotheses on the correlation coefficient.

Basic Usage

```
=FISHER(0.1)
```

	A	B	C
1	0.100335348		
2			

Syntax

```
=FISHER(value)
```

value

A number greater than -1 and less than 1 for which you want to calculate the Fisher transformation.

Warning

If **value** is a logical value, text, or a reference to an empty cell or falls outside the preset limits, **FISHER** returns the #VALUE! error or the #NUM! error respectively.

Examples

Using cell references

```
=FISHER(A2)
```

	A	B	C
1	value	Result	
2	-0.99	-2.646652412	
3			

FISHER returns -2.646652412, which is the Fisher transformation for the value in A2.

Referring to empty values

Note

If **value** is a reference to a cell with an empty value, the empty value is cast to 0 (zero).

```
=FISHER(A2)
```

	A	B	C
1	value	Result	
2		0	
3			

The empty value in A2 is cast to 0 (zero), for which **FISHER** returns 0 (zero).

LARGE

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, SMALL, COMMON PROBLEMS

Returns the **n**-th largest value in a data set.

Basic Usage

```
=LARGE(A2:A6, 3)
```

	A	B	C
1	data	Result	
2	25	55	
3	43		
4	89		
5	87		
6	55		
7			

LARGE arranges the numbers in descending order and returns 55, which is the third largest number in A2:A6.

Note

LARGE ignores Boolean values, text, and empty cells and covers both sorted and unsorted arrays and ranges.

Syntax

```
=LARGE(data, n)
```

data

An array or a range that contains numbers for which you want to find the **n**-th largest value.

n

A number greater than or equal to 1 that represents the position (from the largest to the smallest value) in the array or range.

Warning

If **n** is less than 1 or equal to 0 (zero), **LARGE** returns the #NUM! error.
If **n** is less than 0 (zero), **LARGE** returns the #VALUE! error.

Example

Working with arrays

```
=LARGE({4, 1, 7, "5", 9}, 2)
```

	A	B	C
1	7		
2			

LARGE arranges the numbers in descending order and returns 7, which is the second largest number in the unsorted array.

LINEST

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Calculates parameters of a linear trend based on the method of least squares.

Basic Usage

```
=LINEST(A2:A5, B2:B5, TRUE)
```

	A	B	C	D
1	Data set 1	Data set 2	Result	
2	4	5	0.213740458	
3	5	13		
4	3	9		
5	8	10		
6				

Based on the two equal-sized data sets in A2:A5 and B2:B5, **LINEST** calculates the b variable in the $y = m \cdot x + b$ linear equation and returns 0.213740458.

Note

LINEST returns an array of values that describe a line. However, only the first value is displayed as a result.

Syntax

```
=LINEST(known_data_y, [known_data_x], [calculate_b], [stats])
```

known_data_y

An array or a range of cells with dependent (y) values that you already know.

[known_data_x] (optional)

An array or a range of cells with values of independent (x) variables corresponding to **known_data_y**. If omitted, **[known_data_x]** is assumed to be an array or a range of the same size as **known_data_y**.

[calculate_b] (optional)

A logical value (TRUE or FALSE) that determines whether to calculate the b or m variable in the $y = m \cdot x + b$ linear equation respectively. If **[calculate_b]** is omitted, FALSE is applied by default.

[stats] (optional)

A logical value (TRUE or FALSE) that determines whether to return the additional regression statistics for variables or to calculate only the b and/or m variables in the $y = m \cdot x + b$ linear equation respectively. If **[stats]** is omitted, FALSE is set by default.

Warning

Ensure that **known_data_y** and **[known_data_x]** data sets are equal in size. Otherwise, **LINEST** returns the #REF! error.

Examples

Calculating the m variable in the $y = m \cdot x + b$ linear equation

```
=LINEST(A2:A5, B2:B5, FALSE)
```

	A	B	C	D
1	Data set 1	Data set 2	Result	
2	4	5	0.512	
3	5	13		
4	3	9		
5	8	10		
6				

Based on the two equal-sized data sets in A2:A5 and B2:B5, **LINEST** calculates the m variable and returns 0.512. The $y = m \cdot x + b$ equation, where b equals 0 (zero), transforms into the $y = m \cdot x$ equation.

Calculating the additional regression statistics for the m variable in the $y = m \cdot x + b$ linear equation

```
=LINEST(A2:A4, B2:B4, FALSE, TRUE)
```

	A	B	C	D
1	Data set 1	Data set 2	Result	
2	4	5	0.407272727	
3	5	13		
4	3	9		
5	8	10		
6				

Based on the two equal-sized data sets in A2:A5 and B2:B5, **LINEST** calculates the additional regression statistics for the m variable and returns 0.407272727. The $y = m \cdot x + b$ equation, where b equals 0 (zero), transforms into the $y = m \cdot x$ equation.

Calculating the additional regression statistics for the b variable in the $y = m \cdot x + b$ linear equation

=LINEST(A2:A4, B2:B4, TRUE, TRUE)

	A	B	C	D
1	Data set 1	Data set 2	Result	
2	4	5	0.125	
3	5	13		
4	3	9		
5	8	10		
6				

Based on the two equal-sized data sets in A2:A5 and B2:B5, **LINEST** calculates the additional regression statistics for the b variable and returns 0.125.

MAX

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, MIN, COMMON PROBLEMS

Returns the largest number in a data set.

Basic Usage

```
=MAX(A2:A4)
```

	A	B	C
1	Data	Result	
2	25	80	
3	80		
4	63		
5			

Syntax

```
=MAX(value1, [value2, ...])
```

value1

The first number, array, cell reference, or range for which you want to find the largest number.

[value2, ...] (optional)

Additional numbers, arrays, cell references, or ranges for which you want to find the largest number.

Warning

If you enter text directly into the formula, **MAX** returns the #VALUE! error.

Examples

Using arrays

Note

MAX ignores text supplied to an array or a cell.

```
=MAX({"text", -11, 4, 2, -5})
```

	A	B	C
1	4		
2			

MAX ignores the text in the array and returns 4, which is the largest number in the specified set of values.

Using cell references

```
=MAX(A2, A3, A5, "38")
```

	A	B	C
1	Data	Result	
2	25	43	
3	43		
4			
5	n/a		
6			

MAX ignores the text in A5 and returns 43, which is the largest number in the specified set of values.

MIN

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, MAX, COMMON PROBLEMS

Returns the smallest number in a data set.

Basic Usage

```
=MIN(A2:A5)
```

	A	B	C
1	Data	Result	
2	57	21	
3	39		
4	21		
5	97		
6			

Syntax

```
=MIN(value1, [value2, ...])
```

value1

The first number, array, cell reference, or range for which you want to find the smallest number.

[value2, ...] (optional)

Additional numbers, arrays, cell references, or ranges for which you want to find the smallest number.

Warning

If you enter text directly into the formula, **MIN** returns the #VALUE! error.

Examples

Using arrays

Note

MAX ignores text supplied to an array or a cell.

```
=MIN({"value", 28, 10, 18})
```

	A	B	C
1	10		
2			

MIN ignores the text in the array and returns 10, which is the smallest number in the specified set of values.

Using cell references

```
=MIN(A2, A3, A5, "14", 5)
```

	A	B	C
1	Data	Result	
2	21	5	
3	n/a		
4			
5	9		
6			

MIN ignores the text in A3 and returns 5, which is the smallest number in the specified set of values.

NORM.DIST

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Returns the normal distribution for a specified value, the arithmetic mean of the distribution, and the standard deviation of the distribution.

Basic Usage

```
=NORM.DIST(12, 33, 30, TRUE)
```

	A	B	C
1	0.2419637		
2			

NORM.DIST returns 0.2419637. The result is the cumulative normal distribution for 12 with 33 as the arithmetic mean of the distribution and 30 as the standard deviation of the distribution.

Syntax

```
=NORM.DIST(x, mean, standard_deviation, cumulative)
```

x

A number for which you want to calculate the normal distribution.

mean

A number that represents the arithmetic mean of the distribution.

standard_deviation

A positive number that represents the standard deviation of the normal distribution.

cumulative

A logical value that specifies whether you want to calculate the cumulative normal distribution (TRUE or any negative or positive number) or probability mass (FALSE or 0 (zero)) function.

Warning

If **standard_deviation** is less than or equal to 0 (zero), **NORM.DIST** returns the #NUM! error.

Example

Using cell references

```
=NORM.DIST(A2, A3, A4, FALSE)
```

	A	B	C
1	Data	Result	
2	35	0.006683519	
3	44		
4	59		
5			

NORM.DIST returns 0.006683519, which is the value of the probability mass function for the data in A2, A3, and A4.

RANK.EQ

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Arranges numbers in ascending or descending order and returns the rank of a specified number in a data set.

Basic Usage

```
=RANK.EQ(23, A2:A5, TRUE)
```

	A	B	C
1	data	Result	
2	17	2	
3	23		
4	23		
5	38		
6			

RANK.EQ arranges the numbers in A2:A5 in ascending order and returns 2, which is the rank of 23.

Note

For duplicate values, **RANK.EQ** returns the rank of the first value.

Syntax

```
=RANK.EQ(value, data, [is_ascending])
```

value

A number whose rank you want to determine.

data

An array or a range with data that you want to analyze.

[is_ascending] (optional)

A logical value (TRUE or FALSE) that indicates whether you want to analyze values in

ascending or descending order respectively. If **[is_ascending]** is omitted, FALSE is used by default.

Warning

If **value** is a number that is not available in **data**, **RANK.EQ** returns the #N/A error.

Example

Using arrays

```
=RANK.EQ(8, {8, 6, 5, 10})
```

	A	B	C
1	2		
2			

RANK.EQ arranges the numbers in the array in descending order and returns 2, which is the rank of 8.

Note

RANK.EQ ignores text, logical, and empty values supplied to an array or a range.

SMALL

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, LARGE, COMMON PROBLEMS

Arranges numbers in ascending order with the smallest value in the first position and returns the **n**-th smallest value in the data set.

Basic Usage

```
=SMALL({1, 4, "3", 8, 9}, 2)
```

	A	B	C
1	3		
2			

SMALL arranges the numbers of the array in ascending order and returns 3, which is the second smallest value in the data set.

Syntax

```
=SMALL(data, n)
```

data

An array or a range that contains numbers for which you want to find the **n**-th smallest value.

n

A number greater than or equal to 1 that represents the position (from the smallest to the largest value) in the array or range.

Warning

If **n** is less than 1 or equal to 0 (zero), **SMALL** returns the #NUM! error.
If **n** is less than 0 (zero), **SMALL** returns the #VALUE! error.

Example

Using ranges

```
=SMALL(A2:A6, 3)
```

	A	B	C
1	data	Result	
2	23	45	
3	45		
4	64		
5	90		
6	31		
7			

SMALL arranges the numbers in ascending order and returns 45, which is the third smallest number in the unsorted range A2:A6.

Note

SMALL ignores text, logical, and empty values supplied to an array or a range and covers both sorted and unsorted arrays and ranges.

STDEV.S

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Calculates the standard deviation for a sample of the population.

Note

The standard deviation is a measure of dispersion that shows how far values are from their arithmetic mean.

Basic Usage

```
=STDEV.S(A2:A5)
```

	A	B	C
1	Data	Result	
2	88	9.899494937	
3	TRUE		
4			
5	74		
6			

STDEV.S calculates the standard deviation for the data in A2:A5 and returns 9.899494937. The logical value in A3 and the empty value in A4 are ignored.

Note

STDEV.S considers logical values that you enter directly into the formula. TRUE and FALSE are cast to 1 and 0 (zero) respectively. The function ignores references to cells with text, logical, and empty values.

Syntax

```
=STDEV.S(value1, [value2, ...])
```

value1

The first value, array, cell reference, or range that represents a sample of the population.

[value2, ...] (optional)

Additional values, arrays, cell references, or ranges that represent a sample of the population.

Warning

If the total number of supplied values is fewer than two, **STDEV.S** returns the #DIV/0! error.

Examples

Using arrays

```
STDEV.S({23, 15}, {"31", 17})
```

	A	B	C
1	7.187952884		
2			

STDEV.S calculates the standard deviation for the data in the arrays and returns 7.187952884.

Using cell references

```
=STDEV.S(A2, A4, 10)
```

	A	B	C
1	Data	Result	
2	88	55.154328933	
3			
4	n/a		
5			

STDEV.S calculates the standard deviation for the specified data and returns 55.154328933. The text in A4 is ignored.

VAR

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Calculates variance based on a sample.

Basic Usage

```
=VAR(A2:A6)
```

	A	B	C
1	Data	Result	
2	10	49	
3	17		
4	24		
5			
6	value		
7			

VAR calculates the variance for the values in A2:A6 and returns 49. A5, which is an empty cell, and the text in A6 are ignored.

Note

VAR ignores empty cells and text but considers logical values entered directly into the formula. TRUE and FALSE are cast to 1 and 0 (zero) respectively.

Syntax

```
=VAR(value1, [value2, ...])
```

value1

The first value, array, cell reference, or range that represents a sample.

[value2, ...] (optional)

Additional values, arrays, cell references, or ranges that represent a sample.

Warning

If the total number of supplied values is fewer than two, **VAR** returns the #DIV/0! error.

Examples

Using arrays

```
=VAR({10, "12"}, {16, 14})
```

	A	B	C
1	6.6666667		
2			

VAR calculates the variance for the arrays of values and returns 6.6666667.

Using cell references

```
=VAR(A2, A4)
```

	A	B	C
1	Data	Result	
2	14	8	
3			
4	18		
5			

VAR calculates the variance for the values in A2 and A4 and returns 8.

VARA

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Calculates variance based on a sample.

Note

In contrast to [VAR](#), **VARA** treats text as 0 (zero).

Basic Usage

```
=VARA(A2:A6)
```

	A	B	C
1	Data	Result	
2	9	48.25	
3			
4	0		
5	undefined		
6	14		
7			

VARA calculates the variance for the values in A2:A6 and returns 48.25. The text in A5 is treated as 0 (zero), and A3, which is an empty cell, is ignored.

Note

VARA ignores empty cells but considers logical values. TRUE and FALSE are cast to 1 and 0 (zero) respectively.

Syntax

```
=VARA(value1, [value2, ...])
```

value1

The first value, array, cell reference, or range that represents a sample.

[value2, ...] (optional)

Additional values, arrays, cell references, or ranges that represent a sample.

Warning

If the total number of supplied values is fewer than two, **VARA** returns the #DIV/0! error.

Examples

Using arrays

```
=VARA({14, 20}, {"5", 15, "price"})
```

	A	B	C
1	65.7		
2			

VARA calculates the variance for the arrays of values and returns 65.7. The text is treated as 0 (zero).

Using cell references

```
=VARA(A2, A3)
```

	A	B	C
1	Data	Result	
2	FALSE	0.5	
3	1		
4			

VARA calculates the variance for the values in A2 and A3 and returns 0.5. The logical value in A2 is cast to 0 (zero).

VARPA

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Calculates variance based on an entire population.

Basic Usage

```
=VARPA(A2:A4)
```

	A	B	C
1	Data	Result	
2	text	3.5555556	
3	FALSE		
4	4		
5			

VARPA calculates the variance of the entire population for the values in A2:A4 and returns 3.5555556. The text in A2 and the logical value in A3 are cast to 0 (zero).

Note

VARPA ignores empty cells and treats text as 0 (zero). Logical values (TRUE and FALSE) are cast to 1 and 0 (zero) respectively.

Syntax

```
=VARPA(value1, [value2, ...])
```

value1

The first value, array, cell reference, or range that represents an entire population.

[value2, ...] (optional)

Additional values, arrays, cell references, or ranges that represent an entire population.

Example

Using arrays

```
=VARPA({10, 11, "day"}, 9, "12")
```

	A	B	C
1	18.64		
2			

VARPA calculates the variance of the entire population for the specified values and returns 18.64. The text in the array is cast to 0 (zero).

Chapter 11

Financial functions

COUPPCD

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Returns a numeric date value that corresponds to the previous coupon date prior to the settlement date of a security.

Note

A numeric date value represents a date as the number of days that passed after December 30, 1899, which is the editor's base date.

Basic Usage

```
=COUPPCD(DATE(2002, 3, 20), DATE(2003, 12, 25), 4, 4)
```

	A	B	C
1	37250		
2			

The formula receives the following input: the settlement date of the security—03/20/2002; the maturity date of the security—12/25/2003; the number of coupon payments a year—4; the day count method—European 30/360. **COUPPCD** returns 37250, which is the numeric date value that corresponds to December 25, 2001.

Syntax

```
=COUPPCD(settlement, maturity, frequency, [day_count_convention])
```

settlement

The settlement date of a security (the date following the issue date when the security is transferred to the buyer).

maturity

The expiration date of the security.

frequency

The number of coupon payments a year: 1, 2, or 4 (annual, semiannual, or quarterly payments respectively).

[day_count_convention] (optional)

A number that indicates which day count method you want to use:

0 (zero) or omitted—US (NASD) 30/360 assumes a 30-day month and a 360-day year.

1—Actual/actual represents the actual length of months and years (including a leap year, which consists of 366 days).

2—Actual/360 calculates the actual number of days between the specified dates and assumes a 360-day year.

3—Actual/365 calculates the actual number of days between the specified dates and assumes a 365-day year.

4—European 30/360 assumes a 30-day month and a 360-day year and adjusts month-end dates in compliance with European financial conventions.

Tip

To enter **settlement** and **maturity** correctly, use **DATE** or a function that returns a date, or make a reference to a cell that contains a date. If you supply a date as **settlement** or **maturity** directly to the formula, use double quotation marks, for example, "1/10/2017".

Example

Using cell references

```
=COUPPCD(A2, B2, 1)
```

	A	B	C	D
1	settlement	maturity	Result	
2	09/21/2001	10/13/2007	36812	
3				

Based on the data in A2 and B2, **COUPPCD** returns 36812, which corresponds to the previous coupon date prior to the settlement date (October 13, 2000). The US (NASD) 30/360 day count method is applied by default because **[day_count_convention]** is omitted.

Warning

If **maturity** precedes **settlement**, **COUPPCD** returns the #NUM! error.

DOLLARDE

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Converts a currency price in fractional notation into a currency price in decimal notation.

Note

DOLLARDE is primarily intended for calculations that involve security prices.

Basic Usage

```
=DOLLARDE(5.1, 2)
```

	A	B	C
1	5.5		
2			

DOLLARDE converts the specified fractional price ($5\frac{1}{2}$) into its decimal representation and returns 5.5.

Syntax

```
=DOLLARDE(fractional_price, unit)
```

fractional_price

A currency price specified as a number that contains an integer part and a fractional part, preceded by a decimal separator. The fractional part is the nominator of the fraction.

unit

A positive integer that acts as the denominator of the fraction.

Warning

If **unit** is 0 (zero) or a negative number, **DOLLARDE** returns the #DIV/0! error and the #NUM! error respectively.

Example

Using cell references

```
=DOLLARDE(A2, 4)
```

	A	B	C
1	Data	Result	
2	4.004	4.01	

DOLLARDE converts the fractional price in A2 ($4\frac{0.04}{4}$) into its decimal representation and returns 4.01.

Note

If **unit** is a noninteger, **DOLLARDE** truncates its fractional part.

DOLLARFR

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Converts a currency price in decimal notation into a currency price in fractional notation. **DOLLARFR** is primarily intended for calculations that involve security prices.

Basic Usage

```
=DOLLARFR(0.75, 4)
```

	A	B	C
1	0.3		
2			

DOLLARFR converts the specified price in decimal notation into its fractional representation ($\frac{3}{4}$) and returns 0.3.

Syntax

```
=DOLLARFR(decimal_price, unit)
```

decimal_price

A currency price specified as a decimal number.

unit

A positive integer that acts as the denominator of the fraction.

Warning

If **unit** is 0 (zero) or a negative number, **DOLLARFR** returns the #DIV/0! error and the #NUM! error respectively.

Example

Working with cell references

```
=DOLLARFR(A2, "8.2")
```

	A	B	C
1	Data	Result	
2	1.625	1.5	

DOLLARFR converts the price in decimal notation supplied in A2 into its fractional representation ($1\frac{5}{8}$) and returns 1.5.

Note

If **unit** is a noninteger, **DOLLARFR** truncates its fractional part.

IRR

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Calculates the internal rate of return on an investment based on regular cash flows that include both payments and income.

Basic Usage

```
=IRR(A2:A5, 0.3)
```

	A	B	C
1	Cash flows	Results	
2	-\$5,500.00	-9.88%	
3	\$2,000.00		
4	\$1,000.00		
5	\$1,500.00		
6			

IRR calculates the internal rate of return based on the cash flows in A2:A5 and the estimate for the internal rate of return at 0.3 and returns -9.88%.

Note

IRR formats numbers as a percentage and adds the percent sign (%) to the calculation result.

Syntax

```
=IRR(cashflow_amounts, [rate_guess])
```

cashflow_amounts

An array or a range of cells with numbers that indicate payments and income for which you want to calculate the internal rate of return.

[rate_guess] (optional)

A number that represents your estimate for the internal rate of return. If **[rate_guess]** is omitted, 0.1 (10 percent) is applied by default.

Tip

Enter both payments and income into **cashflow_amounts**. Ensure that you first place negative values to specify payments and then positive values to specify income.

Example

Using arrays

```
=IRR({-3000, 2000, 1500, 5000})
```

	A	B	C
1	61.51%		
2			

IRR calculates the internal rate of return based on the cash flows supplied to the array and returns 61.51%. The estimate of 10% for the internal rate of return is applied by default.

Warning

If you enter **cashflow_amounts** directly into the formula, do not use a currency symbol, for example, \$. Otherwise, **IRR** returns the #VALUE! error.

NPV

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Returns the net present value of an investment based on a discount rate and a series of future cash flows. Cash flows include both payments and income and occur at regular intervals.

Basic Usage

```
=NPV(7%, -5000, -500, 2000, 3000, 8000)
```

	A	B	C
1	\$4,516		
2			

Based on the supplied cash flows and a 7% discount rate, **NPV** calculates the net present value of the investment at \$4,516.

Note

NPV formats numbers as currency and adds a locale-dependent currency symbol to the calculation result.

Syntax

```
=NPV(discount, cashflow1, [cashflow2, ...])
```

discount

A rate of discount for one period.

cashflow1

The first cash flow in a series of future regular cash flows.

[cashflow2, ...] (optional)

One or more additional future regular cash flows. The maximum number of **cashflow** arguments is limited to 254.

Warning

If you enter **cashflow** arguments directly into the formula, do not use a currency symbol, for example, \$. Otherwise, **NPV** returns the #VALUE! error.

Example

Using cell references and ranges

```
=NPV(A2, B2:B4)
```

	A	B	C	D
1	discount	Cash flows	Result	
2	6%	-\$9,000	-\$6,786	
3		\$500		
4		\$1,500		
5				

NPV calculates the net present value of the investment based on the discount rate in A2 and the cash flows in B2:B4 and returns -\$6,786.

Tip

Enter both payments and income as **cashflow** arguments. Ensure that you first place negative values to present payments and then positive values to indicate income.

PMT

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Calculates the payment amount for an investment or a loan based on fixed periodic payments and a fixed interest rate.

Basic Usage

```
=PMT(5%/12, 24, 10000)
```

	A	B	C
1	-\$439		
2			

PMT calculates the payment for the loan of \$10,000 and returns -\$439. The payments will be made for the length of two years at the annual interest rate of 5% and are due at the end of each month.

Note

PMT formats numbers as currency and adds a locale-dependent currency symbol to the calculation result.

Syntax

```
=PMT(rate, number_of_payments, present_value,  
[future_value], [end_or_beginning])
```

rate

An annual interest rate for an investment or a loan.

number_of_payments

The total number of payments for the investment or loan.

present_value

The current value of future payments.

[future_value] (optional)

A future amount that remains after the final payment has been made. If omitted, **[future_value]** is assumed to be 0 (zero) by default.

[end_or_beginning] (optional)

A number that indicates whether payments are due at the end (0) or beginning (1) of each period. If omitted, **[end_or_beginning]** is assumed to be 0 (zero) by default.

Tip

Ensure that you set **rate** and **number_of_payments** correctly depending on whether the payments are due monthly or quarterly. For example, if a two-year loan is paid out monthly, **rate** is divided by 12, and **number_of_payments** is set to 24. If a loan of the same length is paid out quarterly, **rate** is divided by 4, and **number_of_payments** is set to 8.

Examples

Using cell references

```
=PMT(A2/12, B2, C2, 0, 1)
```

	A	B	C	D	E
1	rate	number_of_payments	present_value	Result	
2	7%	12	\$5,000.00	-\$430	
3					

PMT calculates the payment for the loan of \$5,000 and returns -\$430. The payments will be made for the length of one year at the interest rate of 7% and are due at the beginning of each month.

Calculating a regular deposit to savings

Note

PMT allows you to calculate the amount of savings that needs to be deposited monthly or quarterly to obtain the desired amount at the end of the payment period. To do this, supply 0 (zero) to **present_value**, and indicate the amount that you want to obtain in **[future_value]**.

```
=PMT(A2/12, B2, C2, 35000, 1)
```

	A	B	C	D	E
1	rate	number_of_payments	present_value	Result	
2	12%	36	0	-\$804	
3					

PMT calculates the deposit for the three-year savings plan with monthly payments at the annual interest rate of 12% and returns -\$804. The amount that you want to save is \$35,000. The payments are due at the beginning of each month.

Warning

If you enter **present_value** and **[future_value]** directly into the formula, do not use a currency symbol, for example, \$. Otherwise, **PMT** returns the #VALUE! error.

PV

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Calculates the present value of an investment or a loan based on fixed periodic payments and a fixed interest rate.

Basic Usage

```
=PV(5%/12, 24, -500)
```

	A	B	C
1	\$11,396.95		
2			

PV calculates the present value of the loan taken out for the length of two years at the annual interest rate of 5% and returns \$11,396.95. The amount of \$500 is paid out at the end of every month.

Note

PV formats numbers as currency and adds a locale-dependent currency symbol to the calculation result.

Syntax

```
=PV(rate,      number_of_periods,      payment_per_period,  
[future_value], [end_or_beginning])
```

rate

An interest rate per period.

number_of_periods

The number of payment periods in an investment or a loan.

payment_per_period

An amount paid per period. Enter a negative number if you make a payment and a positive number if you receive an amount of money.

[future_value] (optional)

A future amount that remains after the final payment. If omitted, **[future_value]** is assumed to be 0 (zero) by default.

[end_or_beginning] (optional)

A number that indicates whether payments are due at the end (0) or beginning (1) of each period. If omitted, **[end_or_beginning]** is assumed to be 0 (zero) by default.

Tip

Ensure that you set **rate**, **number_of_periods**, and **payment_per_period** correctly depending on whether the payments are due monthly or quarterly. For example, if a two-year loan is paid out monthly, **rate** is divided by 12, and **number_of_periods** is set to 24. If a loan of the same length is paid out quarterly, **rate** is divided by 4, and **number_of_periods** is set to 8.

Examples

Using cell references

```
=PV(A2/12, B2*12, C2, 0, 1)
```

	A	B	C	D	E
1	rate	Years	payment_per_period	Result	
2	8%	5	-\$300.00	\$14,894.17	
3					

PV returns \$14,894.17. It calculates the present value of the loan taken out for the length of five years at the interest rate of 8%. The amount of \$300 is paid out at the beginning of every month.

Calculating the present value of a deposit

Note

PV allows you to calculate the present value of a deposit based on monthly or quarterly payments. To do this, supply 0 (zero) to **payment_per_period**, and indicate the amount that you want to obtain in **[future_value]**.

```
=PV(A2/12, B2, C2, 27000, 1)
```

	A	B	C	D	E
1	rate	number_of_periods	payment_per_period	Result	
2	12%	36	0	- \$18,870.97	
3					

PV calculates the present value of the three-year deposit plan with monthly payments at the annual interest rate of 12% and returns -\$18,870.97. The sum that you want to obtain is \$27,000. The payments are due at the beginning of each month.

Warning

If you enter **payment_per_period** or **[future_value]** directly into the formula, do not use a currency symbol, for example, \$. Otherwise, **PV** returns the #VALUE! error.

Chapter 12

Text functions

ASC

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, ENCODING, COMMON PROBLEMS

Converts a string encoded in a double-byte character set (DBCS) to a string encoded in a single-byte character set (SBCS).

Note

ASC is designed for backward compatibility with older spreadsheet applications. In our editor, the function has no effect on a specified string.

Basic Usage

```
=ASC("value")
```

	A	B	C
1	value		
2			

Our editor uses neither DBCS nor SBCS to encode data, so the specified string remains unchanged.

Tip

If you enter text directly into the formula, enclose the text in double quotation marks.

Syntax

```
=ASC(string)
```

string

A string or a reference to a cell containing a string that you want to convert.

Example

Using cell references

```
=ASC(A2)
```

	A	B	C
1	string	Result	
2	567	567	
3			

CLEAN

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, ENCODING, COMMON PROBLEMS

Removes nonprintable characters from a specified string.

Tip

Use **CLEAN** to remove trailing or leading spaces and nonprintable characters from imported strings.

Basic Usage

=CLEAN(A2)

	A	B	C
1	text	Result	
2	q	q	
3			

Copied from another application, the string in A2 starts with a nonprintable character (a horizontal tab). **CLEAN** returns the strings without the nonprintable character.

Tip

CLEAN removes only the first 32 ASCII characters (from 0 (zero) to 31). To remove unnecessary space characters (character 32), use **TRIM**. You cannot remove some nonprintable characters, such as nonbreaking space characters, with **CLEAN** or **TRIM**, but you can replace them using **SUBSTITUTE**.

Syntax

=CLEAN(text)

text

A string from which you want to remove nonprintable characters.

Examples

Removing unwanted line breaks

```
=CLEAN(A2)
```

	A	B	C	D
1	text	Result		
2	many line breaks	many line breaks		
3				
4				

The string in A2 has a line break after every word. **CLEAN** returns the string without any line breaks.

Removing all nonprintable ASCII characters

Tip

Use **CLEAN** with **TRIM** to remove all nonprintable ASCII characters, including space characters.

```
=TRIM(CLEAN(A2))
```

	A	B	C	D
1	text	Result		
2	line breaks and spaces	line breaks and spaces		
3				
4				

The string in A2 has unnecessary line breaks and spaces. The formula returns the string without them.

CODE

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, ENCODING, COMMON PROBLEMS

Returns the Unicode numeric code for the first character in a specified string.

Note

CODE is designed for backward compatibility with older spreadsheets and covers only the first 255 Unicode characters. To access all the codes, use [UNICODE](#).

Basic Usage

=CODE(A2)

	A	B	C	D
1	string	Result		
2	¢	162		
3	£25	163		
4	α	164		

The formula is entered into B2 and then copied to the cells below it with autofill. The codes in the Result column correspond to the characters in the **string** column.

Syntax

=CODE(string)

string

A string in which you want to find the Unicode numeric code for the first character.

Note

CODE is the inverse of **CHAR**, which returns a character by its code.

Examples

Using **string** whose first character has a code above 255

Warning

Characters with codes above 255 are treated as question marks. Using **CODE** with such characters makes the function return 63 instead of a proper code.

=CODE(A2)

	A	B	C	D
1	string	Result	Proper code	
2	Đ	63	272	
3	đ	63	273	
4	?	63	63	

The formula is entered into B2 and then copied to the cells below it with autofill. The codes in the Result column and the proper codes in the Proper code column correspond to the characters in the **string** column.

Removing unwanted leading characters

Tip

Imported strings often contain leading characters that you cannot remove with **CLEAN** and **TRIM**. To remove an unwanted leading character, replace the character with an empty string using **SUBSTITUTE**.

=SUBSTITUTE(A2, CHAR(CODE(A2)), "")

	A	B	C
1	string	Result	
2	□45	45	
3			

The formula returns 45. **SUBSTITUTE** nests **CHAR** (which, in turn, nests **CODE**) and replaces the unwanted first character of the string in A2 with an empty string.

CONCATENATE

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, ENCODING, COMMON PROBLEMS

Joins multiple strings into a single string.

Basic Usage

```
=CONCATENATE("zoo", ".com")
```

	A	B	C
1	zoo.com		
2			
3			

Tip

Enclose text in double quotation marks if you enter it into the formula.

Syntax

```
=CONCATENATE(string1, [string2, ...])
```

string1

The first string.

[string2, ...] (optional)

Additional strings that you want to join with **string1**.

Warning

Supply strings to **CONCATENATE** one by one. If you supply a range to the function, it is resolved with Implicit intersection. If the function cannot be resolved, it returns the #VALUE! error.

Examples

Combining numbers into a string

Note

Numbers do not need to be enclosed in double quotation marks.

```
=CONCATENATE(1, 2, 3)
```

	A	B	C
1	123		
2			
3			

Preventing strings from running together by adding a space character at the end of a string

Tip

To prevent strings from running into each other, separate them with spaces. For example, add a space at the end of a string.

```
=CONCATENATE("LG ", A1, B1)
```

	A	B	C	
1	Line-up	Model	Product name	
2	GL	850	LG GL850	
3				

Preventing strings from running together by adding a space character as an argument

Tip

To prevent strings from running into each other, alternatively, add a space character as one of the arguments.

=CONCATENATE(A1, " ", B1)

	A	B	C	
1	Line-up	Model	Product name	
2	Pro	16	Pro 16	
3				

Joining strings with the ampersand operator (&)

Tip

You can also apply the ampersand operator (&) to join strings without using a function.

=A2&A3&A4

	A	B	C
1	Data	Result	
2	a	abc	
3	b		
4	c		
5			

DBCS (JIS)

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, ENCODING, COMMON PROBLEMS

Converts a string encoded in a single-byte character set (SBCS) to a string in a double-byte character set (DBCS).

Note

Designed for backward compatibility with older spreadsheet applications, **DBCS** and **JIS** are identical in their syntax and usage. In our editor, the functions have no effect on a specified string.

Basic Usage

```
=DBCS("value")
```

	A	B	C
1	value		
2			

Our editor uses neither DBCS nor SBCS to encode data, so the specified string remains unchanged.

Tip

If you enter text directly into the formula, enclose the text in double quotation marks.

Syntax

```
=DBCS(string)
```

string

A string or a reference to a cell containing a string that you want to convert.

Example

Using cell references

```
=DBCS(A2)
```

	A	B	C
1	string	Result	
2	567	567	
3			

EXACT

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, ENCODING, COMMON PROBLEMS

Tests whether two strings are identical.

Basic Usage

```
=EXACT("R2D2", "R2D2")
```

	A	B	C
1	TRUE		
2			

EXACT returns TRUE because the tested strings are identical.

Note

EXACT is case-sensitive. Uppercase and lowercase values are not equivalent.

Syntax

```
=EXACT(string1, string2)
```

string1

The first string that you want to test.

string2

The second string that you want to test.

Examples

Testing nonidentical strings

```
=EXACT("R2D2", "r2d2")
```

	A	B	C
1	FALSE		
2			

EXACT returns FALSE because the tested strings are not identical.

Using cell references

```
=EXACT(A2, B2)
```

	A	B	C	
1	string1	string2	Result	
2	GX750	GX750	TRUE	
3				

EXACT returns TRUE because the tested strings are identical.

Testing hidden characters

Note

EXACT detects nonprintable characters, such as spaces and tabulations.

```
=EXACT(CHAR(32), CHAR(160))
```

	A	B	C
1	FALSE		
2			

EXACT nests **CHAR** twice as its arguments and compares a regular space (**CHAR(32)**) and a nonbreaking space (**CHAR(160)**). The characters are different, so the result is FALSE.

Case-insensitive string testing

Tip

If you want to apply case-insensitive comparison, use a formula with the equal sign (=) instead of **EXACT**.

= "case" = "CaSe"

	A	B	C
1	TRUE		
2			

The formula uses the equal sign (=) instead of **EXACT**, applies case-insensitive comparison, and returns TRUE.

FIND (FINDB)

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, ENCODING, COMMON PROBLEMS

Returns the position of a string within another string.

Tip

FIND and **FINDB** are identical. Apply **FINDB** only if you need backward compatibility with older applications that use DBCS encodings.

Basic Usage

```
=FIND("p", "Pop")
```

	A	B	C
1	3		
2			

FIND performs a case-sensitive search and returns 3, which is the position of "p" in "Pop" starting from the first character.

Tip

If you enter a string into the formula, enclose the string in double quotation marks.

Syntax

```
=FIND(search_for, text_to_search, [starting_at])
```

search_for

A string for which you want to search within **text_to_search**.

text_to_search

A string within which you want to search for the first occurrence of **search_for**.

[starting_at] (optional)

A number greater than 0 (zero) for a character within **text_to_search** at which you want to start searching. If **[starting_at]** is omitted, 1 is used by default.

Tip

Use **FIND** (**FINDB**) to perform a case-sensitive search without wildcard characters. To ignore letter case and enable wildcards, use **SEARCH**.

Examples

Using cell references

```
=FIND(B1, B2)
```

	A	B	C	D
1	search_for	3		
2	text_to_search	123		
3	Result	3		
4				

Note

FIND automatically recognizes numbers as strings, so enclosing numbers in double quotation marks is not required. For example, both **=FIND(3, 123)** and **=FIND("3", "123")** are valid.

Using [starting_at]

```
=FIND(B1, B2, 2)
```

Tip

If **text_to_search** contains several identical **search_for** matches some of which you want to skip, specify **[starting_at]** to make **FIND** search from the specified position.

	A	B	C	D
1	search_for	chop		
2	text_to_search	chop chop!		
3	Result	6		
4				

FIND searches for "chop" starting from the second character of "chop chop!", so the first "chop" is skipped. The function returns 6, which is the starting position of the second "chop".

Note

FIND (**FINDB**) treats white space and punctuation marks as individual characters.

LEFT (LEFTB)

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, ENCODING, RIGHT, COMMON PROBLEMS

Returns a substring from the left side of a string.

Tip

LEFT and **LEFTB** are identical. Apply **LEFTB** only if you need backward compatibility with older applications that use DBCS encodings.

Basic Usage

```
=LEFT("text", 2)
```

	A	B	C
1	te		
2			

LEFT returns a two-character substring.

Tip

If you enter a string into the formula, enclose the string in double quotation marks.

Syntax

```
=LEFT(string, [number_of_characters])
```

string

A string from which you want to extract a substring.

[number_of_characters] (optional)

A number greater than or equal to 0 (zero) for how many characters you want to extract. If the argument is omitted, 1 is used by default. If **[number_of_characters]** is greater than the number of characters in **string**, **LEFT** returns the whole string.

Warning

If you supply a negative number to `[number_of_characters]`, `LEFT` returns the #VALUE! error.

Examples

Using cell references

Tip

Make the formula more flexible and less prone to errors by using cell references rather than strings in the formula.

`=LEFT(B1, B2)`

	A	B	C	D
1	string	HD 660		
2	[number_of_characters]	2		
3	Result	HD		
4				

Retrieving the first character of string

Tip

Omit `[number_of_characters]` to extract only the first character of string.

`=LEFT(B1)`

	A	B	C
1	string	9888	
2	Result	9	
3			

LEN (LENB)

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, ENCODING, FORMATTING, FORMULA STRUCTURE, COMMON PROBLEMS

Returns the number of characters in a specified string.

Tip

LEN and **LENB** are identical. Apply **LENB** only if you need backward compatibility with older applications that use DBCS encodings.

Basic Usage

```
=LEN("string")
```

	A	B	C
1	6		
2			

LEN returns 6, which is the number of characters in the supplied string.

Tip

If you enter a string into the formula, enclose the string in double quotation marks.

Syntax

```
=LEN(text)
```

text

A string in which you want to count characters.

Note

LEN counts all characters, including spaces and nonprintable characters.

Examples

Using cell references

Tip

Make your formula more flexible and less prone to errors by using cell references rather than strings in the formula.

```
=LEN(A2)
```

	A	B	C
1	text	Result	
2	Count me	8	
3	and me!	7	
4	875.6	5	

The formula is entered into B2 and then copied to B3 and B4 with autofill.

Counting characters in numbers

Note

You can use **LEN** to count characters in a number, but remember that formatting a number does not change its length.

```
=LEN(A2)
```

	A	B	C	D
1	Number 10 in three formats	Result		
2	10	2		
3	\$10.00	2		
4	1900/01/09	2		

The formula is entered into B2 and then copied to B3 and B4 with autofill. **LEN** returns 2 in all the three cases because formatting a number does not affect its length.

Tip

To make better use of the text functions with formatted numbers, use double quotation marks, for example, "1900/01/09", or [TEXT](#).

LOWER

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, UPPER, PROPER, COMMON PROBLEMS

Returns text in lowercase.

Basic Usage

```
=LOWER("New YORK")
```

	A	B	C
1	new york		
2			

Tip

If you enter a string into the formula, enclose the string in double quotation marks.

Syntax

```
=LOWER(text)
```

text

A string that you want to appear in lowercase.

Examples

Using cell references

Tip

Make your formula more flexible and less prone to errors by using cell references rather than strings in the formula.

=LOWER(A1)

	A	B	C
1	text	Result	
2	Rose	rose	
3	Fern	fern	
4	Hope	hope	

The formula is entered into B2 and then copied to B3 and B4 with autofill.

Using nonletter characters

Note

LOWER does not change nonletter characters.

=LOWER(A1)

	A	B	C
1	text	Result	
2	J&R	j&r	
3	R2D2	r2d2	
4	\$\$\$	\$\$\$	

The formula is entered into B2 and then copied to B3 and B4 with autofill.

MID (MIDB)

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, ENCODING, FORMATTING, FORMULA STRUCTURE, COMMON PROBLEMS

Extracts a substring from a string.

Tip

MID and **MIDB** are identical. Use **MIDB** only if you need backward compatibility with older applications that use DBCS encodings.

Basic Usage

Tip

If you enter a string into the formula, enclose the string in double quotation marks.

```
=MID("text", 1, 2)
```

	A	B	C
1	te		
2			

MID returns "te", which starts with the first character of the string "text" and is two characters in length.

Tip

Use **MID** to extract a substring from a specified position in a string. To extract from the beginning or the end, use **LEFT** or **RIGHT** respectively.

Syntax

```
=MID(string, starting_at, extract_length)
```

string

A string from which you want to extract a substring.

starting_at

A number greater than 0 (zero) for a character at which you want to start extracting. If **starting_at** is greater than the actual length of **string**, **MID** returns an empty value.

extract_length

A number for the length of the substring in characters. If **extract_length** is greater than the actual length of the initial string, **MID** returns the whole string without any trailing spaces.

Warning

If **starting_at** is a negative number, **MID** returns the #VALUE! error.

Examples

Using cell references

Tip

Make your formula more flexible and less prone to errors by using cell references rather than strings in the formula.

=MID(C1, C2, C3)

	A	B	C	D	E
1	string		i5 10600k		
2	starting_at		4		
3	extract_length		5		
4					
5	Result				
6	10600				

MID returns 10600, which begins with the fourth character of the string in C1 and is five characters long.

Extracting subsets from numbers

=MID(A2, B2, C2)

	A	B	C	D	E
1				Result	
2	1234	2	3	234	
3	\$10.00	1	4	10	
4	05/18/1903	2	3	234	
5	"05/18/1903"	2	3	05/	

The formula is entered into D2 and then copied to the cells below it with autofill. The numbers in A3 and A4 are formatted. Formatting a number does not change its length, which affects the output of **MID**.

Tip

To make better use of the text functions with formatted numbers, use double quotation marks, for example, "1900/01/09", or **TEXT**.

PROPER

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, UPPER, LOWER, COMMON PROBLEMS

Capitalizes every word in a specified string.

Tip

Use **PROPER** for proper nouns, for example, city names.

Basic Usage

```
=PROPER("new y0RK")
```

	A	B	C
1	New York		
2			

Tip

If you enter a string into the formula, enclose the string in double quotation marks.

Syntax

```
=PROPER(text_to_capitalize)
```

text_to_capitalize

A string in which you want to capitalize all the words.

Examples

Using cell references

Tip

Make your formula more flexible and less prone to errors by using cell references rather than strings in the formula.

=PROPER(A1)

	A	B	C	D
1	Input		Result	
2	mr. robot		Mr. Robot	
3	mr. bLACk		Mr. Black	
4	mr. aliEN		Mr. Alien	

The formula is entered into C2 and then copied into C3 and C4 with autofill.

Using strings with nonletter characters

Note

PROPER does not change nonletter characters and treats them as word dividers.

=PROPER(A1)

	A	B	C	D
1	Input		Result	
2	d&g		D&G	
3	r1c1		R1C1	
4	up-to-date		Up-To-Date	

The formula is entered into C2 and then copied into C3 and C4 with autofill.

RIGHT(RIGHTB)

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, ENCODING, FORMATTING, FORMULA STRUCTURE, LEFT, COMMON PROBLEMS

Returns a substring from the right side of a string.

Tip

RIGHT and **RIGHTB** are identical. Use **RIGHTB** only if you need backward compatibility with older applications that use DBCS encodings.

Basic Usage

```
=RIGHT("text", 2)
```

	A	B	C
1	xt		
2			

RIGHT returns a substring of two rightmost characters.

Tip

If you enter a string into the formula, enclose the string in double quotation marks.

Syntax

```
=RIGHT(string, [number_of_characters])
```

string

A string from which you want to extract a substring.

[number_of_characters] (optional)

A number greater than or equal to 0 (zero) for how many characters you want to extract. If the argument is omitted, 1 is used by default. If **[number_of_characters]** is greater than the number of characters in **string**, **RIGHT** returns the whole string.

Warning

If you supply a negative number to `[number_of_characters]`, **RIGHT** returns the #VALUE! error.

Examples

Using cell references

Tip

Make your formula more flexible and less prone to errors by using cell references rather than strings in the formula.

```
=RIGHT(A2, A3)
```

	A	B	C	D	E
1	Input		Result		
2	HD 660		660		
3	3				

RIGHT returns 660, which is the three rightmost characters of the string in A2.

Extracting the last character of a string

Tip

Omit `[number_of_characters]` to extract only the last character of **string**.

```
=RIGHT(A2)
```

	A	B	C
1	string	Result	
2	9998	8	

SEARCH (SEARCHB)

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, ENCODING, FORMATTING, FORMULA STRUCTURE, COMMON PROBLEMS

Returns the starting position of a specified substring within a string.

Tip

SEARCH and **SEARCHB** are identical. Use **SEARCHB** only if you need backward compatibility with older applications that use DBCS encodings.

Basic Usage

Tip

If you enter a string into the formula, enclose the string in double quotation marks.

```
=SEARCH("p", "Pot")
```

	A	B	C
1	1		
2			

SEARCH returns 1, which is the position of "p" in "Pot".

Tip

Use **SEARCH** to perform a case-insensitive search with wildcard characters. For a case-sensitive search without wildcard support, use **FIND**.

Syntax

```
=SEARCH(search_for, text_to_search, [starting_at])
```

search_for

A string that you want to find.

text_to_search

A string within which you want to search.

[starting_at] (optional)

An integer greater than 0 (zero) for the position of a character within **text_to_search** at which you want to start searching. If the argument is omitted, 1 is used by default.

Warning

If no match is found for **search_for** or if **[starting_at]** is less than 1 or greater than the length of **text_to_search**, **SEARCH** returns the #VALUE! error.

Examples

Using cell references

Tip

Make your formula more flexible and less prone to errors by using cell references rather than strings in the formula.

```
=SEARCH(A2, A3)
```

	A	B	C	D	E
1	Input		Result		
2	3		2		
3	538				

SEARCH returns 2, which is the position of the string from A2 in the string from A3.

Tip

SEARCH automatically recognizes numbers as strings, so you do not need to use double quotation marks. For example, both **=SEARCH(3, 123)** and **=SEARCH("3", "123")** are valid.

Initiating a search from a specified position

Tip

If your string contains several search keys and you want to skip some of them, specify **[starting_at]** to initiate a search from a specified position.

=SEARCH(A2, A3, 3)

	A	B	C	D	E
1	Input		Result		
2	clap		6		
3	CLAP CLAP!				

SEARCH starts searching from the third character of the string in A3 and returns 6, which is the starting position of the identified match.

Using wildcards

Note

White space and punctuation marks (except for those in wildcards) are treated as individual characters.

=SEARCH(A2, A3)

	A	B	C	D	E
1	Input		Result		
2	?tx		1		
3	GTX 780				

Tip

In wildcards, a question mark (?) represents any single character. An asterisk (*) represents any sequence of characters. To find an actual question mark or asterisk, type a tilde (~) before the character.

SUBSTITUTE

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, ENCODING, FORMATTING, FORMULA STRUCTURE, COMMON PROBLEMS

Replaces specified content in a string in one or all occurrences.

Tip

Use **SUBSTITUTE** to replace string segments if you know their content but not their position in the string. To replace a string based on its location, use **REPLACE**.

Basic Usage

Note

SUBSTITUTE is case-sensitive and does not support wildcard characters.

```
=SUBSTITUTE("Roar", "R", "S")
```

	A	B	C
1	Soar		
2			

SUBSTITUTE replaces every "R" with an "S" in "Roar" and returns "Soar".

Tip

If you enter a string into the formula, enclose the string in double quotation marks.

Syntax

```
=SUBSTITUTE(text_to_search, search_for, replace_with,  
[occurrence_number])
```

text_to_search

A string in which you want to replace a segment.

search_for

An old string that you want to replace.

replace_with

A new string with which you want to replace the old string.

[occurrence_number] (optional)

A number that indicates the occurrence of the string that you want to replace. If the argument is omitted, all occurrences are replaced.

Examples

Using cell references

Tip

Make your formula more flexible and less prone to errors by using cell references rather than strings in the formula.

```
=SUBSTITUTE(C1, C2, C3)
```

	A	B	C	D	E	F
1	text_to_search		Blob			
2	search_for		b			
3	replace_with		ck			
4						
5	Result				Block	

SUBSTITUTE affects only the lowercase character specified in C2.

Specifying a particular occurrence for substitution

Tip

By default, **SUBSTITUTE** replaces all occurrences of specified content. If you want to replace only one, specify **[occurrence_number]**.

=SUBSTITUTE(C1, C2, C3, C4)

	A	B	C	D	E
1	text_to_search		1995		
2	search_for		9		
3	replace_with		8		
4	[occurrence_number]		2		
5	Result			1985	

SUBSTITUTE affects only one occurrence of the specified string and returns 1985.

Tip

SUBSTITUTE automatically recognizes numbers as strings, so you do not need to use double quotation marks. For example, both **=SUBSTITUTE(1995, 9, 8, 2)** and **=SUBSTITUTE("1995", "9", "8", 2)** are valid.

Troubleshooting

Problem	Possible cause
The string is returned without changes.	No specified search_for was found. Example =SUBSTITUTE("text", "r", "s") The specified search_for was found fewer times than the specified [occurrence_number] . Example =SUBSTITUTE("text", "x", "s", 2)
A #VALUE! error is returned	Strings for arguments were entered without double quotation marks. Example =SUBSTITUTE(text, r, s)

TEXT

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, ENCODING, FORMATTING, FORMULA STRUCTURE, COMMON PROBLEMS

Returns a formatted number as a string.

Basic Usage

Tip

Use **TEXT** to receive a formatted number to be treated as a string.

```
=TEXT(1, "m/d/y")
```

	A	B	C
1	12/31/99		
2			

TEXT formats the number as a date and returns 12/31/99.

Syntax

```
=TEXT(value, format)
```

value

A number, a date, or a time value that you want to format.

format

A formatting pattern you want to apply to **value**. Formatting patterns are strings. If you enter them directly into the formula, enclose them in double quotation marks.

Tip

For more information on formatting patterns, see [Custom formatting](#).

Example

Using cell references

Tip

Make your formula more flexible and less prone to errors by using cell references rather than strings in the formula.

```
=TEXT(A2, B2)
```

	A	B	C	D
1	value	format	Result	
2	35	"\$00"	\$35	
3	45.678	000	046	
4	24849	#.45	24849.45	
5	123.223	h:m am/pm	5:21 AM	
6	45	"m"	2	

The formula is entered into C2 and then copied to the other cells in the column. **TEXT** formats the number in A2 as currency and rounds the number in A3, adding a leading zero to the result in C3. The function adds a fractional part to the number in A4, displays the number in A5 as a 12-hour time stamp, and returns 2 for the number in A6, which corresponds to the second month of the year.

Note

Formatting styles vary depending on the locale that you specify in the system settings of your computer. Formatting initiated under the settings of one locale may not return proper results under the settings of another locale. For more information, see [Introduction to localization](#).

TRIM

Related topics: [FUNCTIONS](#), [VALUE TYPES AND PROPERTIES](#), [ENCODING](#), [FORMATTING](#), [FORMULA STRUCTURE](#), [COMMON PROBLEMS](#)

Removes unnecessary space characters from strings.

Basic Usage

Tip

Imported strings often contain trailing or leading spaces and extra spaces between words. Use **TRIM** to rid your strings of them.

```
=TRIM(A2)
```

	A	B	C
1	text	Result	
2	12 boys	12 boys	

TRIM returns the supplied string without unnecessary space characters.

Tip

TRIM removes only 7-bit ASCII space characters (character 32 in the Unicode character set). Any other unwanted white space can be removed with [CLEAN](#) or [SUBSTITUTE](#) (see examples below).

Syntax

```
=TRIM(text)
```

text

A string or a reference to a cell with a string from which you want to remove unnecessary spaces.

Examples

Removing all nonprintable ASCII characters

Tip

To remove all nonprintable ASCII characters, including the space character, use [CLEAN](#) with [TRIM](#).

```
=TRIM(CLEAN(A2))
```

	A	B	C	D
1	text	Result		
2	line breaks and spaces	line breaks and spaces		
3				

The nested formula returns the supplied string without line breaks and unnecessary spaces.

Removing nonbreaking spaces

Tip

You cannot remove a nonbreaking space (character 160 in the Unicode character set) with [TRIM](#). However, you can replace it with a normal space character and then remove it.

```
=TRIM(SUBSTITUTE(A2, UNICHAR(160), UNICHAR(32)))
```

	A	B	C	D
1	text	Result		
2	non-breaking spaces	non-breaking spaces		
3				

The string in A2 has extra nonbreaking spaces at the beginning and between words. [SUBSTITUTE](#) replaces all nonbreaking spaces ([UNICHAR\(160\)](#)) with regular spaces ([UNICHAR\(32\)](#)), then [TRIM](#) removes the unnecessary nonbreaking spaces.

UNICODE

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, ENCODING, FORMATTING, FORMULA STRUCTURE, COMMON PROBLEMS

Returns the numeric code for the first character in a specified string.

Basic Usage

=UNICODE(A2)

	A	B	C	D
1	string	Result		
2	Ð	272		
3	đ	273		
4	Æ	198		
4	range	114		

The formula is entered into B2 and then copied to the cells below it with autofill. **UNICODE** returns the codes that correspond to the lone characters or the first character of the string in column A.

Syntax

=UNICODE(string)

string

A string starting with the character for which you want to get the Unicode value.

Note

UNICODE is the inverse of **UNICHAR**, which returns a character by its code.

Example

Using nested functions

Tip

Imported strings often contain leading characters that cannot be removed with [CLEAN](#) and [TRIM](#). Use nested functions with [UNICHAR](#) to replace such characters with an empty string.

```
=SUBSTITUTE(A2, UNICHAR(UNICODE(A2)), "")
```

	A	B	C	D
1	string	Result		
2	□45	45		
3				

UPPER

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, LOWER, PROPER, COMMON PROBLEMS

Returns text in uppercase.

Basic Usage

```
=UPPER("Sold out")
```

	A	B	C
1	SOLD OUT		
2			

Tip

If you enter a string into the formula, enclose the string in double quotation marks.

Syntax

```
=UPPER(text)
```

text

A string that you want to appear in uppercase.

Examples

Using cell references

Tip

Make your formula more flexible and less prone to errors by using cell references rather than strings in the formula.

=UPPER(A2)

	A	B	C	D
1	text	Result		
2	sos	SOS		
3	kdt	KDT		
4	mia	MIA		

The formula is entered into B2 and then copied to B3 and B4 with autofill.

Using nonletter characters

Note

UPPER does not change nonletter characters.

=UPPER(A2)

	A	B	C	D
1	text	Result		
2	m&m	M&M		
3	r9 3900x	R9 3900X		
4	£££	£££		

The formula is entered into B2 and then copied to B3 and B4 with autofill.

Chapter 13

Information functions

CELL

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Returns information on a specified cell.

Basic Usage

```
=CELL("col", A2)
```

	A	B	C
1	1		
2			

The formula requests the column number of the specified cell (A2) and returns 1.

Syntax

```
=CELL(info_type, [reference])
```

info_type

A text string that specifies what information you want to request:

- **"address"** is for the absolute address of the cell.
- **"col"** is for the column number, starting from column A.
- **"contents"** is for the value in the cell.
- **"prefix"** is for the text alignment in the cell, returning the following:
 - ' for right-aligned text or nontext values.
 - " for left-aligned text.
 - \ for fill-aligned text.
 - ^ for centered text.
- **"row"** is for the row number.
- **"type"** is for the type of data in the cell, returning the following:

- "e" for a blank cell.
- "1" for text.
- "v" for everything else.

[reference] (optional)

A cell on which you want to request information. Omit **[reference]** to choose the cell with the formula.

Note

If **[reference]** is a range, **CELL** returns information about the top-left cell.

Examples

Requesting text alignment

=CELL(B2, A2)

	A	B	C	D
1		info_type	Result	
2	text	prefix	'	

The single quotation mark indicates that the text in A2 is left-aligned.

Warning

If you specify **info_type** through a cell reference, do not use double quotation marks.

Requesting data type

=CELL(B2, A2)

	A	B	C	D
1		info_type	Result	
2		type	e	

The letter "e" indicates that A2 is blank.

INFO

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Returns information of a specified type on the current working environment.

Basic Usage

```
=INFO("numfile")
```

	A	B	C
1	1		
2			

The formula returns 1 as the number of sheets in the workbook currently open.

Syntax

```
=INFO(info_type)
```

info_type

A text string that specifies what type of information you want to return:

- **"numfile"** is the number of sheets in the current workbook.
- **"osversion"** is the current version of the operating system.
- **"recalc"** is the recalculation mode set in the current workbook (automatic or manual).
- **"release"** is the current version of the editor.
- **"system"** is the operating system name.

ISBLANK

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Checks whether a cell is blank and returns TRUE or FALSE.

Basic Usage

```
=ISBLANK(A2)
```

	A	B	C
1	value	Result	
2		TRUE	
3			

Syntax

```
=ISBLANK(value)
```

value

A cell that you want to check.

Example

Checking cells that are not blank

```
=ISBLANK(A2)
```

	A	B	C
1	value	Result	
2	1	FALSE	
3			

Tip

If a cell looks blank, but the function returns `FALSE`, check whether the cell contains any white space.

ISERR

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Checks whether a cell contains an error (other than the #N/A error) and returns TRUE or FALSE.

Basic Usage

```
=ISERR(A2)
```

	A	B	C
1	value	Result	
2	#DIV/0!	TRUE	
3			

Syntax

```
=ISERR(value)
```

value

A cell that you want to check.

Examples

Error is the #N/A error

```
=ISERR(A2)
```

	A	B	C
1	value	Result	
2	#N/A	FALSE	
3			

No error

=ISERR(A2)

	A	B	C
1	value	Result	
2	12	FALSE	
3			

ISERROR

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Checks whether a cell contains an error and returns TRUE or FALSE.

Basic Usage

```
=ISERROR(#REF!)
```

	A	B	C
1	TRUE		
2			

Syntax

```
=ISERROR(value)
```

value

A value or reference to a cell that you want to check.

Example

Using cell references

```
=ISERROR(A2)
```

	A	B	C	D
1	value	Result		
2	#VALUE!	TRUE		
3				

ISEVEN

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Checks whether a number is even and returns TRUE or FALSE.

Basic Usage

```
=ISEVEN(12)
```

	A	B	C
1	TRUE		
2			

Syntax

```
=ISEVEN(value)
```

value

A number that you want to check.

Note

If **value** is a reference to an empty cell, **ISEVEN** treats it as 0 (zero).

Examples

Using cell references

```
=ISEVEN(A2)
```

	A	B	C	D
1	value	Result		
2	9	FALSE		

Using nonintegers

Note

If **value** is not an integer, the function truncates its fractional part.

=ISEVEN(A2)

	A	B	C	D
1	value	Result		
2	10.9	TRUE		

Note

ISEVEN treats the logical value TRUE as 1 and the logical value FALSE as 0 (zero).

ISLOGICAL

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Checks whether a value is a Boolean value and returns TRUE or FALSE.

Basic Usage

```
=ISLOGICAL(FALSE)
```

	A	B	C
1	TRUE		
2			

Syntax

```
=ISLOGICAL(value)
```

value

A value that you want to check.

Example

Using cell references

```
=ISLOGICAL(A2)
```

	A	B	C	D
1	value	Result		
2	FALSE	TRUE		

ISNA

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Checks whether a value is the #N/A error and returns TRUE or FALSE.

Basic Usage

```
=ISNA(#N/A)
```

	A	B	C
1	TRUE		
2			

Syntax

```
=ISNA(value)
```

value

A value that you want to check.

Example

Using cell references

```
=ISNA(A2)
```

	A	B	C	D
1	value	Result		
2	#N/A	TRUE		

ISNUMBER

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Checks whether a value is a number and returns TRUE or FALSE.

Basic Usage

```
=ISNUMBER(22)
```

	A	B	C
1	TRUE		
2			

Syntax

```
=ISNUMBER(value)
```

value

A value that you want to check.

Examples

Using cell references

```
=ISNUMBER(A2)
```

	A	B	C	D
1	value	Result		
2		FALSE		

Using numbers in double quotation marks

```
=ISNUMBER("10")
```

	A	B	C
1	TRUE		
2			

ISODD

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Checks whether a number is odd and returns TRUE or FALSE.

Basic Usage

```
=ISODD(11)
```

	A	B	C
1	TRUE		
2			

Syntax

```
=ISODD(value)
```

value

A number that you want to check.

Note

If **value** is a reference to an empty cell, **ISODD** treats it as 0 (zero).

Examples

Using cell references

```
=ISODD(A2)
```

	A	B	C	D
1	value	Result		
2	9	TRUE		

Working with nonintegers

Note

If a number supplied to **ISODD** is a noninteger, its fractional part is truncated.

=ISODD(A2)

	A	B	C	D
1	value	Result		
2	10.9	FALSE		

Note

ISODD treats the logical value TRUE as 1 and the logical value FALSE as 0 (zero).

ISREF

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Checks whether a value is a reference and returns TRUE or FALSE.

Basic Usage

```
=ISREF(A1)
```

	A	B	C
1	TRUE		
2			

Syntax

```
=ISREF(value)
```

value

A value that you want to check.

ISTEXT

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Checks whether a value is text and returns TRUE or FALSE.

Basic Usage

```
=ISTEXT("abc")
```

	A	B	C
1	TRUE		
2			

Syntax

```
=ISTEXT(value)
```

value

A value that you want to check.

Examples

Using numbers in double quotation marks

Note

ISTEXT detects numbers in double quotation marks as text but treats logical values as nontext.

```
=ISTEXT("10")
```

	A	B	C
1	TRUE		
2			

Using cell references

```
=ISTEXT(A2)
```

	A	B	C	D
1	value	Result		
2	Pot	TRUE		

Tip

If your cell looks empty, but **ISTEXT** detects text, check your cell for white space.

NA

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMMON PROBLEMS

Returns the #N/A error, which means "value is not available".

Basic Usage

```
=NA()
```

	A	B	C
1	#N/A		
2			

NA returns #N/A, indicating that A1 is empty.

Tip

Use **NA** to mark empty cells. By entering #N/A into cells, you can avoid the problem of unintentionally including empty cells in your calculations.

Syntax

```
=NA()
```

Chapter 14

Reference functions

ADDRESS

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, REFERENCE TYPES, COMPARATIVE OPERATORS, COMMON PROBLEMS

Returns a cell location on a worksheet based on specified row and column numbers.

Basic Usage

```
=ADDRESS(1, 1)
```

	A	B	C
1	\$A\$1		
2			

ADDRESS returns the address of the cell based on the specified row and column numbers.

Syntax

```
=ADDRESS(row, column, [absolute_relative_mode],  
[use_a1_notation], [sheet])
```

row

A row number.

column

A column number.

[absolute_relative_mode] (optional)

A number corresponding to the reference type that you want to use. Omit the argument to retrieve a reference with an absolute row and an absolute column. For different reference types, use:

- **1** for an absolute row and an absolute column (for example, \$A\$1).
- **2** for an absolute row and a relative column (for example, A\$1).
- **3** for a relative row and an absolute column (for example, \$A1).

- **4** for a relative row and a relative column (for example, A1).

[use_a1_notation] (optional)

A Boolean value that specifies which reference notation you want to use. Omit the argument or use TRUE for A1. Use FALSE for R1C1.

[sheet] (optional)

A sheet name on which a cell is located. Omit the argument to use the current sheet.

Tip

If you omit an argument between other arguments, remember to place a comma after the omitted argument.

Examples

Using A1: relative rows and columns

```
=ADDRESS(1, 1, 4, TRUE)
```

	A	B	C
1	A1		
2			

Using R1C1: absolute rows and columns

```
=ADDRESS(1, 1, 1, FALSE)
```

	A	B	C
1	R1C1		
2			

Using R1C1: absolute rows and relative columns

```
=ADDRESS(1, 1, 2, FALSE)
```

	A	B	C
1	R1C[1]		
2			

AREAS

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, REFERENCE TYPES, COMPARATIVE OPERATORS, COMMON PROBLEMS

Counts references in a reference or a list of references.

Basic Usage

```
=AREAS(A1:A2)
```

	A	B	C
1		1	
2			

Syntax

```
=AREAS(reference)
```

reference

A reference to a cell or a range or a list of references.

Examples

Counting in a list of references

```
=AREAS((B1, C1:C2))
```

	A	B	C
1	2		
2			

Tip

Always enclose a list of references in parentheses.

Nesting other functions

```
=AREAS(INDIRECT("B1"))
```

	A	B	C
1	1		
2			

Note

You can nest another function inside **AREAS** only if this function returns a reference.

CHOOSE

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, REFERENCE TYPES, COMPARATIVE OPERATORS, COMMON PROBLEMS

Returns a value from a data set based on the position number of the value.

Basic Usage

```
=CHOOSE(1, 10, 20, 30)
```

	A	B	C
1	10		
2			

The formula chooses the first value from the list.

Syntax

```
=CHOOSE(index, choice1, [choice2, ...])
```

index

The position of a value that you want to choose, starting from 1 and not greater than the total number of values on a list.

choice1

The first value on the list.

[choice2, ...] (optional)

Additional values.

Note

If **index** is a noninteger, **CHOOSE** truncates its fractional part.

Examples

Using cell references

```
=CHOOSE(A2, B2, B3, B4)
```

	A	B	C	D
1	index	choice	Result	
2	1	John	John	
3		Jane		
4		Jim		

Using **index** in A2, **CHOOSE** identifies the first item on the list and returns "John".

Using **CHOOSE** as a nested function

```
=SUM(CHOOSE(A2, B3, B4, B5, B6), B8)
```

	A	B	C	D
1	index	choice	Result	
2	2	9	10	
3		8		
4		7		
5		6		
6		5		
7		4		
8		3		

Using **index** in A2, **CHOOSE** identifies the second item on the list and returns 7, which is then added to 3 in B8, resulting in 10.

COLUMN

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, REFERENCE TYPES, COMPARATIVE OPERATORS, COMMON PROBLEMS

Returns the number of the column where a specified cell is located.

Basic Usage

```
=COLUMN(A2)
```

	A	B	C
1	1		
2			

The formula returns 1, which is the number of the column where A2 is located.

Syntax

```
=COLUMN([cell_reference])
```

[cell_reference]

A cell whose column number you want to determine. Omit **[cell_reference]** to find the column number of the cell with the formula.

Note

COLUMN works with single-row ranges. The result is determined by Implicit intersection.

Examples

Locating columns with formulas

```
=COLUMN()
```

	A	B	C
1	1		
2			

Working with single-row ranges as arguments

```
=COLUMN(B1:C1)
```

	A	B	C
1	Range		
2	Result	2	
3			

When you supply a single-row range to **COLUMN**, the function is resolved with implicit intersection. The cell with the formula (B2) implicitly intersects B1:C1 in B1, so the function returns 2, which is the number of the column where B1 is located.

Working with single-column ranges as arguments

```
=COLUMN(A2:A3)
```

	A	B	C
1	Range	Result	
2		1	
3			

When you supply a single-column range to **COLUMN**, the function is resolved with implicit intersection. The cell with the formula (B2) implicitly intersects A2:A3 in A2, so the function returns 1, which is the number of the column where A2 is located.

COLUMNS

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, REFERENCE TYPES, COMPARATIVE OPERATORS, COMMON PROBLEMS

Counts columns in a range or an array.

Basic Usage

```
=COLUMNS(B1:C1)
```

	A	B	C
1	2		
2			

COLUMNS counts the number of columns in B1:C1 and returns 2.

Syntax

```
=COLUMNS(range)
```

range

A range or an array in which you want to count columns.

Example

Counting in arrays

```
=COLUMNS({1, 2; 4, 6, 5})
```

	A	B	C
1	3		
2			

COLUMNS counts the number of columns in the array and returns 3.

HLOOKUP

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, REFERENCE TYPES, COMPARATIVE OPERATORS, COMMON PROBLEMS

Searches for a value in the top row of a range and returns another value from a specified row of the same column.

Basic Usage

```
=HLOOKUP("Clyde", A1:D3, 3, FALSE)
```

	A	B	C	D	E
1	Date	Bond	Clyde	Donovan	
2	2009	\$10,620	\$9,558	\$12,744	
3	2011	\$30,798	\$28,674	\$31,860	
4		Result			
5		28674			

HLOOKUP searches for "Clyde" in the top row of A1:D3 and returns the value from row 3 of the same column. FALSE indicates that the function searches for an exact match.

Syntax

```
=HLOOKUP(search_key, range, index, [match_type])
```

search_key

A case-insensitive value for which you want to search.

range

A range that you want to search.

index

The number of a row from which you want to retrieve a value, starting from 1 and not greater than the total number of rows.

[match_type] (optional)

A Boolean value that specifies a type of search. To search for an approximate match, omit **[match_type]** or use TRUE. Use FALSE to search only for exact matches.

Warning

If **index** is greater than the number of rows in the searched table, **HLOOKUP** returns the #REF! error.

Examples

Using approximate matching

```
=HLOOKUP(A5, A1:D2, 2, TRUE)
```

	A	B	C	D	E
1	Rating	90	80	70	
2	Class	S	A	B	
3					
4	Key	Result			
5	87	A			

Note

When approximate matching is enabled and no exact match is present, **HLOOKUP** searches for the next smaller value.

HLOOKUP searches for the position of the key (87) in the top row of A1:D2. There is no exact match, so the function finds the next smaller value (80) and returns A, which is the value in the corresponding position in row 2 of the same column.

Using approximate matching with wildcards

```
=HLOOKUP(A5, A1:D2, 2, TRUE)
```

	A	B	C	D	E
1	Food	Chicken	Ham	Lamb	
2	Protein (g)	28	14	23	
3					
4	Key	Result			
5	Lam?	14			

Approximate matching sometimes returns undesirable results. Although "Lam?" looks similar to "Lamb", "Ham" is actually the next smaller value, so **HLOOKUP** returns 14. As an alternative, you can search for an exact match with wildcards.

Using exact matching with wildcards

```
=HLOOKUP(A5, A1:D2, 2, FALSE)
```

	A	B	C	D	E
1	Food	Chicken	Ham	Lamb	
2	Protein (g)	28	14	23	
3					
4	Key	Result			
5	Lam?	23			

HLOOKUP searches for the position of the key ("Lam?") in the top row of A1:D2, finds the position of the matching value ("Lamb"), and returns 23, which is the corresponding value in row 2 of the same column.

Note

By default, the question mark (?) and the asterisk (*) are wildcard characters. The question mark represents any single character, and the asterisk represents any sequence of characters. If you want a wildcard character to be treated as a regular character, put the tilde (~) before it (for example, ~?). To ensure a more reliable result when working with wildcards, enable exact matching by supplying FALSE as **[match_type]**.

Ensuring that the search range contains the key

```
=HLOOKUP(A5, A1:D2, 2, FALSE)
```

	A	B	C	D	E
1	Food	Chicken	Cheese	Lamb	
2	Protein (g)	28	14	23	
3					
4	Key	Result			
5	Ham	#N/A			

HLOOKUP fails to find a match for the key ("Ham") in the top row of A1:D2 and returns the #N/A error.

HYPERLINK

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, REFERENCE TYPES, COMPARATIVE OPERATORS, COMMON PROBLEMS

Creates a hyperlink in a cell of a worksheet.

Basic Usage

```
=HYPERLINK("JohnBalance.com")
```

	A	B	C
1	JohnBalance.com		
2			

Syntax

```
=HYPERLINK(url, [link_label])
```

url

A web address, a link to a file, or a reference to a cell that contains such data.

[link_label] (optional)

A label that you want to show instead of the web address or link to the file or a reference to a cell that contains a label.

Tip

Ensure that you enclose web addresses and links to files in double quotation marks if you enter them directly into the formula.

Examples

Creating hyperlinks to files on your computer

```
=HYPERLINK("Desktop/JUNE.pdf")
```

	A	B	C
1	Desktop/JUNE.pdf		
2			

Using labels instead of web addresses

```
=HYPERLINK("JohnBalance.com","Official website")
```

	A	B	C
1	Official website		
2			

Using cell references as arguments

```
=HYPERLINK(A2, B2)
```

	A	B	C	D
1	url	[link_label]	Result	
2	JohnBalance.com	John Balance	John Balance	

INDEX

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, REFERENCE TYPES, COMPARATIVE OPERATORS, COMMON PROBLEMS

Returns an element of an array or a range based on row and column positions.

Basic Usage

```
=INDEX({1, 2; 3, 4}, 2, 2)
```

	A	B	C
1	4		
2			

INDEX returns the array element from the second row of the second column.

Syntax

```
=INDEX(area, row_num, [column_num], [range_num])
```

area

An array, a range, or a list of ranges from which you want to retrieve an element.

row_num

The row number of an element in the specified area; not greater than the total number of rows.

[column_num] (optional)

The column number of an element in the specified area; not greater than the total number of columns.

[range_num] (optional)

The range from which you want to retrieve an element if **area** consists of multiple ranges. Omit the argument to retrieve an element from the first range.

Note

If the range or array has only one column ({1; 2; 3}), you can use **row_num** to specify the position of the element that you want to retrieve. For example, **=INDEX({1; 2; 3}, 3)** returns 3.

Examples

Retrieving values from ranges

=INDEX(A1:C3, A6, B6)

	A	B	C	D
1	Date	Product	Quantity	
2	03/27/2015	Checkers	43	
3	06/18/2015	Chess	89	
4				
5	Row	Column	Result	
6	2	3	43	

Retrieving rows

Tip

To retrieve a whole row, omit **[column_num]** or specify it as 0 (zero).

=SUM(INDEX(B2:D3, A6, B6))

	A	B	C	D	E
1		Bond	Clyde	Donovan	
2	2009	\$10,620	\$9,558	\$12,744	
3	2011	\$30,798	\$28,674	\$31,860	
4					
5	Row	Column	Result		
6	1	0	32922		

INDEX returns the first row of the specified range. Then the formula becomes equivalent to **=SUM(B2:D2)**.

Retrieving columns

Tip

To retrieve a whole column, supply an empty value (a space) into **row_num** or specify it as 0 (zero).

=SUM(INDEX(B2:D3, A6, B6))

	A	B	C	D	E
1		Bond	Clyde	Donovan	
2	2009	\$10,620	\$9,558	\$12,744	
3	2011	\$30,798	\$28,674	\$31,860	
4					
5	Row	Column	Result		
6	0	1	41418		

The formula is equivalent to **=SUM(B2:B3)**.

Retrieving ranges

Tip

To return a whole range, supply an empty value (a space) into **row_num** and omit **[column_num]**. Alternatively, specify them both as 0 (zero).

=SUM(INDEX((A1:B3, C1:C3), A6, B6, 2))

	A	B	C	D
1	Date	Product	Quantity	
2	03/27/2015	Checkers	43	
3	06/18/2015	Chess	89	
4				
5	Row	Column	Result	
6	0	0	132	

The formula is equivalent to **=SUM(C1:C3)**.

Tip

Always enclose a list of ranges in parentheses, and separate the ranges with a comma.

Retrieving from unknown positions

=INDEX(A1:A6, MATCH("Hayashi", B1:B3))

	A	B	C
1	Date	Name	
2	2009-06-07	Ezra	
3	2009-08-11	Hatter	
4	2009-10-15	Donovan	
5	2009-12-19	Tiersen	
6	2010-02-22	Hayashi	
7			
8	Result		
9	2010-02-22		

If you do not know the position of a value, you can use **MATCH** to find it. After **MATCH** finds the specified text, the formula becomes equivalent to **=INDEX(A1:A6, 6)**.

Tip

If you want to supply an empty value into an argument between other arguments, put a comma after the space for the empty value to ensure proper input: **=FUNCTION(arg1, , arg3)**.

INDIRECT

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, REFERENCE TYPES, COMPARATIVE OPERATORS, COMMON PROBLEMS

Converts a string into a cell reference.

Basic Usage

```
=INDIRECT("A1")
```

	A	B	C
1	4	4	
2			

INDIRECT converts a string into a cell address (A1) and displays the content of the cell.

Syntax

```
=INDIRECT(cell_reference_as_string, [is_A1_notation])
```

cell_reference_as_string

A string or reference to a string.

[is_A1_notation] (optional)

A Boolean value that specifies a reference style. For the A1 reference style, omit the argument or use TRUE. Use FALSE for the R1C1 reference style.

Examples

Using R1C1 style

```
=INDIRECT("R2C1", FALSE)
```

	A	B	C
1	Reference	INDIRECT	
2	75	75	

INDIRECT uses the reference to A2 and returns its value (75).

Supplying strings through cell references

=INDIRECT(B2)

	A	B	C	D
1	Data	String	INDIRECT	
2	25	A2	25	

INDIRECT uses the reference to B2, interprets its content as a reference to A2, and returns the value in A2 (25).

Assembling cell references

=INDIRECT(B2&C2)

	A	B	C	D	E
1	Data	String 1	String 2	INDIRECT	
2	60	A	2	60	

The concatenation operator (&) connects two strings into a single one. **INDIRECT** uses the newly created reference (A2) and returns the value in A2 (60).

LOOKUP

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, REFERENCE TYPES, COMPARATIVE OPERATORS, COMMON PROBLEMS

Searches for the position of a key in a row or column of data and returns the value in the corresponding position from another row or column.

Note

LOOKUP does not support wildcards and ignores letter case. To search for partial matches, use **HLOOKUP** and **VLOOKUP**.

Basic Usage

```
=LOOKUP(1, {1, 2}, {10, 20})
```

	A	B	C
1	10		
2			

The function finds the key (1) in the first array and returns 10, which is the corresponding value in the second array.

Syntax

```
=LOOKUP(search_key, search_range, [result_range])
```

search_key

A value for which you want to search.

search_range

An array or a range that you want to search.

[result_range] (optional)

An array or a range of the same size as **search_range** from which you want to retrieve a value. Omit **[result_range]** to search for and retrieve values from the same range.

Note

If the range that you want to search is two-dimensional (at least two rows and two columns), **LOOKUP** activates the array mode, which works differently depending on the shape of your range. For a detailed explanation, see [Using array mode with wide ranges](#).

Examples

Working with single-row ranges

```
=LOOKUP(A5, A1:D1, A2:D2)
```

	A	B	C	D	E
1	Food	Chicken	Ham	Lamb	
2	Protein (g)	28	14	23	
3					
4	Key	Result			
5	Ham	14			

LOOKUP searches for the position of the key in A5 ("Ham") within A1:D1 and returns 14, which is the corresponding value in A2:D2.

Using array mode with wide ranges

Note

If the width of your range is greater than its height, **LOOKUP** searches for the position of the key in the top row and returns a corresponding value from the bottom row.

```
=LOOKUP(B6, A1:D3)
```

	A	B	C	D	E
1	Points	90	80	70	
2	Class	First	Business	Economy	
3	Stars	5	4	3	
4					
5	Hotel	Key	Result		
6	Stilton	95	5		

Note

If the exact match cannot be found, **LOOKUP** searches for the next smaller value.

LOOKUP searches for the position of the key from B6 (95) in the top row of A1:D3. There is no exact match, so the function finds the next smaller value (90) and returns 5, which is the value in the corresponding position in the bottom row.

Using array mode with tall ranges

Note

If the height of your range is greater than its width, **LOOKUP** searches for the key in the leftmost column and returns a corresponding value from the rightmost row.

=LOOKUP(C2, A1:B6)

	A	B	C	D	E
1	Planet	°C	Key	Result	
2	Mercury	430	Earth	16	
3	Venus	471			
4	Earth	16			
5	Mars	-28			
6	Jupiter	-108			

LOOKUP searches for the position of the key in C2 ("Earth") in the leftmost column of A1:B6 and returns 16, which is the value in the corresponding position in the rightmost column.

Ensuring that the search range contains the key

Warning

If the search range contains no key, **LOOKUP** returns the #N/A error.

=LOOKUP(A5, A1:D1, A2:D2)

	A	B	C	D	E
1	Food	Chicken	Cheese	Lamb	
2	Protein (g)	28	14	23	
3					
4	Key	Result			
5	Ham	#N/A			

LOOKUP fails to find the key ("Ham") in A1:D1 and returns the #N/A error.

MATCH

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, REFERENCE TYPES, COMPARATIVE OPERATORS, COMMON PROBLEMS

Returns the position of a value in a row or column of data.

Basic Usage

```
=MATCH(10, {10, 20, 30, 40})
```

	A	B	C
1	1		
2			

MATCH returns 1, which is the position of the specified number (10) in the array.

Syntax

```
=MATCH(search_key, range, [search_type])
```

search_key

A value of any kind, including errors and empty values, that you want to find.

range

A range or an array of one row or one column in size that you want to search.

[search_type] (optional)

A type of search that you want to use, which is one of the following:

- any positive number or omitted—search first for the position of the last exact match and then for the position of the last value smaller than **search_key**.
- 0 (zero)—search for the position of the first exact match.
- any negative number—search first for the position of the last exact match and then for the position of the last value greater than **search_key**.

Note

MATCH searches from left to right or from top to bottom and ignores letter case.

Examples

Searching for equal or smaller values

```
=MATCH(A8, B2:B5, 1)
```

	A	B	C	D
1	Index	Item ID	Item	
2	1	1035	apple	
3	2	1032	grapefruit	
4	3	1033	mango	
5	4	1036	peach	
6				
7	search_key	Result		
8	1034	3		

MATCH finds no exact match for the key in A8 (1034) and returns 3, which is the position of the last smaller value in B2:B5 (1033).

Searching only for equal values

```
=MATCH(A8, B2:B5, 0)
```

	A	B	C	D
1	Index	Item	Price	
2	1	apple	23	
3	2	grapefruit	40	
4	3	mango	20	
5	4	peach	17	
6				
7	search_key	Result		
8	p*	4		

MATCH finds an exact match for the key in A8 ("p*") and returns 4, which is the position of the match in B2:B5 ("peach").

Note

By default, the question mark (?) and the asterisk (*) are wildcard characters. The question mark represents any single character, and the asterisk represents any sequence of characters. If you want a wildcard character to be treated as a regular character, put the tilde (~) before it (for example, ~?).

Searching for equal or greater values

=MATCH(A8, B2:B5, -1)

	A	B	C	D
1	Index	Price		
2	1	17		
3	2	18		
4	3	23		
5	4	40		
6				
7	search_key	Result		
8	20	4		

MATCH finds no exact match for the key in A8 (20) and returns 4, which is the position of the last greater value in B2:B5 (40).

OFFSET

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, REFERENCE TYPES, COMPARATIVE OPERATORS, COMMON PROBLEMS

Returns the value(s) of a cell or range that is located a specified number of rows and columns away from a specified point.

Basic Usage

```
=OFFSET(B2, 1, 1)
```

	A	B	C	D	E
1		June	July	August	
2	2009	\$10,620	\$9,558	\$12,744	
3	2011	\$30,798	\$28,674	\$31,860	
4					
5	Result	28674			

OFFSET returns 28674, which is the value in a single-cell range that is one row below and one column to the right of B2.

Syntax

```
=OFFSET(cell_reference, offset_rows, offset_columns, [height], [width])
```

cell_reference

A cell or range that you want to use as the starting point.

offset_rows

A number of rows that you want to shift from the starting point. Use positive values to shift right and negative values to shift left.

offset_columns

A number of columns that you want to shift from the starting point. Use positive values to shift down and negative values to shift up.

[height] (optional)

A number of rows in the output range. Omit **[height]** to use the size of **cell_reference**.

[width] (optional)

A number of columns in the output range. Omit **[width]** to use the size of **cell_reference**.

Note

By default, when **OFFSET** finishes shifting, it assumes that the found position is the top-left corner of the range to return. You can change this with negative **[height]** and **[width]**:

- Negative **[height]** implies that the range to return is above the found position.
- Negative **[width]** implies that the range to return is to the left of the found position.

Examples

Using positive values in nested functions

```
=SUM(OFFSET(A1, 1, 1, 4, 1))
```

	A	B	C	D	E
1		John	Jane		
2	Apples	1	5		
3	Bananas	3	15		
4	Oranges	7	20		
5	Mangoes	13	25		
6					
7		John			
8	Total	24			

OFFSET shifts one row down and one column to the right of A1 and returns a range with four rows and one column (B2:B5), whose values are then added up to return 24.

Using negative values in nested functions

```
=SUM(OFFSET(D6, -1, -1, 1, -2))
```

	A	B	C	D	E
1		John	Jane		
2	Apples	1	5		
3	Bananas	3	15		
4	Oranges	7	20		
5	Mangoes	13	25		
6					
7		Total			
8	Mangoes	38			

OFFSET shifts one row up and one column to the left of D6 and returns a range with two columns and one row (B5:C5), whose values are then added up to return 38.

ROW

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, REFERENCE TYPES, COMPARATIVE OPERATORS, COMMON PROBLEMS

Returns the number of the row where a specified cell is located.

Basic Usage

```
=ROW(A2)
```

	A	B	C
1	Cell	Result	
2		2	

Syntax

```
=ROW([cell_reference])
```

[cell_reference]

A cell whose row number you want to retrieve. Omit the argument to return the row number of the cell with the formula.

Note

ROW works with single-column ranges. The result is determined by Implicit intersection.

Examples

Locating the row with the formula

```
=ROW()
```

	A	B	C
1	1		
2			

Working with single-column ranges

=ROW(A1:A3)

	A	B	C
1	Range	Result	
2	A2	2	
3	A3		

The cell with the formula (B2) implicitly intersects the range in A2, so **ROW** returns 2, which is the row number of A2.

Working with single-row ranges

=ROW(B1:C1)

	A	B	C	D
1	Range	B1	C1	
2	Result	1		
3				

The cell with the formula (B2) implicitly intersects the range in B1, so **ROW** returns 1, which is the row number of B1.

ROWS

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, REFERENCE TYPES, COMPARATIVE OPERATORS, COMMON PROBLEMS

Counts rows in a range or an array.

Basic Usage

```
=ROWS(A1:B1)
```

	A	B	C
1	1		
2			

Syntax

```
=ROWS( range )
```

range

A range or an array in which you want to count rows.

Example

Counting in arrays

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

	A	B	C
1	2		
2			

VLOOKUP

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, REFERENCE TYPES, COMPARATIVE OPERATORS, COMMON PROBLEMS

Searches for a value in the first column of a range and returns the corresponding value from another column.

Note

VLOOKUP searches from top to bottom and ignores letter case.

Basic Usage

```
=VLOOKUP("Earth", A1:B6, 2)
```

	A	B	C	D	E
1	Planet	°C		Result	
2	Mercury	430		16	
3	Venus	471			
4	Earth	16			
5	Mars	-28			
6	Jupiter	-108			

VLOOKUP searches for "Earth" in the first column of A1:B6 and returns 16, which is the corresponding value from the second column.

Syntax

```
=VLOOKUP(search_key, range, index, [match_type])
```

search_key

A value for which you want to search.

range

A range that you want to search.

index

The number of a column (starting from 1 and not greater than the total number of columns in **range**) from which you want to retrieve a value.

[match_type] (optional)

A Boolean value that specifies a search type. Omit the argument or use TRUE for approximate matching. Use FALSE for exact matching.

Examples

Approximate matching

Note

When approximate matching is enabled and the exact match is not present, **VLOOKUP** searches for the next smaller value.

```
=VLOOKUP(A8, A2:B5, 2, TRUE)
```

	A	B	C
1	Price	Discount	
2	10	0.01	
3	50	0.1	
4	100	0.25	
5	500	0.5	
6			
7	Price	Result	
8	60	0.1	

The next smaller value is 50, so **VLOOKUP** returns 0.1.

Partial matching with wildcards

```
=VLOOKUP(A10, A2:C7, 3, FALSE)
```

	A	B	C	D
1	Product	Price	Supplier	
2	Apples	68	Tom	
3	Mangoes	56	Peter	
4	Grapefruits	87	Garry	
5	Bananas	59	Michael	
6	Peaches	67	Mark	
7	Apricots	82	Jerry	
8				
9	Product	Result		
10	m*	Peter		

VLOOKUP searches for the product that starts with "m" and returns "Peter", which is the name of the corresponding supplier.

Chapter 15

Engineering functions

COMPLEX

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMPLEX NUMBERS, COMMON PROBLEMS

Creates a complex number based on specified real and imaginary components.

Basic Usage

```
=COMPLEX(-5, -8)
```

	A	B	C
1	-5-8i		
2			

Based on the specified real and imaginary components, **COMPLEX** returns the complex number -5-8i.

Syntax

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

real_number

A number that represents the real component of a complex number.

imaginary_number

A number that represents the imaginary component of a complex number.

[suffix] (optional)

A case-sensitive character indicating which suffix you want to use: "i" or "j". If the argument is omitted, "i" is used by default.

Warning

If you enter the capital "I" or "J" into **[suffix]**, **COMPLEX** results in the #VALUE! error.

Examples

Using suffix "j"

```
=COMPLEX(34, 91, "j")
```

	A	B	C
1	34+91j		
2			

COMPLEX returns the complex number 34+91j based on the supplied real and imaginary components and the suffix "j".

Using cell references

```
=COMPLEX(A2, A3, "i")
```

	A	B	C
1	Data	Result	
2	9	9-24i	
3	-24		

COMPLEX returns the complex number 9-24i based on the real and imaginary components in A2 and A3 and the suffix "i".

Using logical values

Note

COMPLEX treats the logical value TRUE as 1 and the logical value FALSE as 0 (zero).

```
=COMPLEX(A2, -8)
```

	A	B	C
1	Data	Result	
2	FALSE	-8i	
3			

COMPLEX returns the complex number $-8i$. The real component is 0 (zero) because FALSE is cast to 0, and the imaginary component is -8.

IMABS

Related topics: FUNCTIONS, VALUE TYPES AND PROPERTIES, FORMULA STRUCTURE, WORKING WITH CELLS AND RANGES, COMPLEX NUMBERS, COMMON PROBLEMS

Returns the absolute value for a specified complex number.

Basic Usage

```
=IMABS("-5-8j")
```

	A	B	C
1	9.4339811321		
2			

IMABS returns 9.4339811321, which is the absolute value of the complex number -5-8j.

Syntax

```
=IMABS(complex_number)
```

complex_number

A complex number for which you want the absolute value.

Examples

Using cell references

```
=IMABS(A2)
```

	A	B	C
1	Data	Result	
2	7+8i	10.630145813	

IMABS returns 10.630145813, which is the absolute value of the complex number in A2.

Using nested functions

```
=IMABS(COMPLEX(4, -2))
```

	A	B	C
1	4.472135955		
2			

IMABS returns 4.472135955, which is the absolute value of the complex number 4-2i created by **COMPLEX**.

Appendix A

Alphabetic index

ABS
ACOS
ACOSH
ACOT
ADDRESS
AND
AREAS
ASC
ASIN
ATAN
ATAN2
ATANH
AVERAGE
CELL
CHOOSE
CLEAN
CODE
COLUMN
COLUMNS
COMPLEX
CONCATENATE
DBCS (JIS)
COSH
COTH
COUNT
COUNTA
COUNTBLANK

COUNTIF
COUNTIFS
COUPPCD
CSC
CSCH
DATE
DATEVALUE
DAY
DEGREES
DOLLARDE
DOLLARFR
DSUM
EDATE
EOMONTH
EXACT
EXP
F.DIST.RT
F.DIST
FACTDOUBLE
FALSE
FIND (FINDB)
FISHER
HLOOKUP
HOUR
HYPERLINK
IF
IFERROR
IFNA
IMABS
INDEX
INDIRECT
INFO
INT
IRR
ISBLANK
ISERR
ISERROR
ISEVEN
ISLOGICAL
ISNA

ISNUMBER
ISODD
ISOWEEKNUM
ISREF
ISTEXT
LARGE
LEFT (LEFTB)
LEN (LENB)
LINEST
LN
LOG
LOG10
LOOKUP
LOWER
MATCH
MAX
MID (MIDB)
MIN
MINUTE
MONTH
NA
NORM.DIST
NOW
NPV
OFFSET
OR
PI
PMT
POWER
PRODUCT
PROPER
PV
QUOTIENT
RAND
RANK.EQ
RIGHT(RIGHTB)
ROUND
ROUNDDOWN
ROUNDUP
ROW

ROWS
SEARCH (SEARCHB)
SEC
SECH
SINH
SMALL
SQRT
SQRTPI
STDEV.S
SUBTOTAL
SUBSTITUTE
SUM
SUMIF
SUMIFS
SUMPRODUCT
SUMSQ
SWITCH
TANH
TEXT
TODAY
TRIM
TRUE
UNICODE
UPPER
VAR
VARA
VARPA
VLOOKUP
YEAR