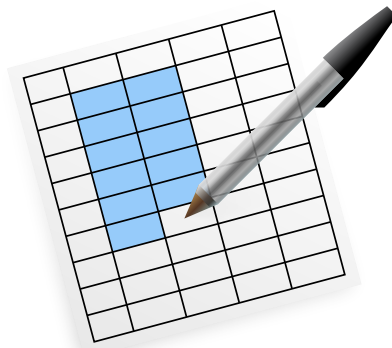


The Calligra Sheets Handbook

Pamela Roberts
Anne-Marie Mahfouf
Gary Cramblitt



The Calligra Sheets Handbook

Contents

1	Introduction	16
2	Calligra Sheets Basics	17
2.1	Spreadsheets for Beginners	17
2.2	Selecting Cells	19
2.3	Entering Data	19
2.3.1	Generic Cell Format	20
2.4	Copy, Cut and Paste	20
2.4.1	Copying and Pasting Cell Areas	21
2.4.2	Other Paste Modes	21
2.5	Insert and Delete	22
2.6	Simple Sums	22
2.6.1	Recalculation	23
2.7	Sorting Data	23
2.8	The Status bar Summary Calculator	24
2.9	Saving your Work	24
2.9.1	Templates	25
2.10	Printing a Spreadsheet	25
3	Spreadsheet Formatting	26
3.1	Cell Format	26
3.1.1	Data Formats and Representation	27
3.1.2	Fonts and Text Settings	29
3.1.3	Text Position and Rotation	30
3.1.4	Cell Border	32
3.1.5	Cell Background	33
3.1.6	Cell Protection	33
3.2	Conditional Cell Attributes	34
3.3	Changing Cell Sizes	34
3.4	Merging Cells	35
3.5	Hiding Rows and Columns	35
3.6	Sheet properties	35

The Calligra Sheets Handbook

4	Advanced Calligra Sheets	38
4.1	Series	38
4.2	Formulae	38
4.2.1	Built in Functions	38
4.2.2	Logical Comparisons	39
4.2.3	Absolute Cell References	40
4.3	Arithmetic using Special Paste	40
4.4	Array Formulas	40
4.5	Goal Seeking	41
4.6	Pivot Tables	41
4.7	Using more than one Worksheet	43
4.7.1	Consolidating Data	43
4.8	Inserting a Chart	43
4.9	Inserting External Data	44
4.10	Link Cells	45
4.11	Validity Checking	45
4.12	Protection	45
4.12.1	Document Protection	45
4.12.2	Sheet protection	47
4.12.3	Cell or selected cells protection	47
4.12.4	Hide cell formula	48
4.12.5	Hide all in the cell	49
4.13	Other Features	49
4.13.1	Named Cells and Areas	49
4.13.2	Cell Comments	50
5	Configuring Calligra Sheets Shortcuts and Toolbars	51
5.1	Shortcuts	51
5.2	Toolbars	52
6	The Calligra Sheets Configuration Dialog Box	53
6.1	Interface	53
6.2	Open/Save	54
6.3	Plugins	55
6.4	Spelling	56
6.5	Author	56

The Calligra Sheets Handbook

7	Command Reference	58
7.1	The File Menu	58
7.2	The Edit Menu	59
7.3	The View Menu	60
7.4	The Go Menu	60
7.5	The Insert Menu	60
7.6	The Format Menu	61
7.7	The Data Menu	62
7.8	The Tools Menu	62
7.9	The Settings Menu	63
7.10	The Help Menu	64
7.11	The Right Mouse Button Menu	64
7.12	Other Shortcuts	66
8	Functions	67
8.1	Supported Functions	67
8.1.1	Bit Operations	67
8.1.1.1	BITAND	67
8.1.1.2	BITLSHIFT	68
8.1.1.3	BITOR	68
8.1.1.4	BITRSHIFT	69
8.1.1.5	BITXOR	69
8.1.2	Conversion	69
8.1.2.1	ARABIC	69
8.1.2.2	ASCIITOCHAR	70
8.1.2.3	BOOL2INT	70
8.1.2.4	BOOL2STRING	71
8.1.2.5	CARX	71
8.1.2.6	CARY	71
8.1.2.7	CHARTOASCII	72
8.1.2.8	DECSEX	72
8.1.2.9	INT2BOOL	73
8.1.2.10	NUM2STRING	73
8.1.2.11	POLA	74
8.1.2.12	POLR	74
8.1.2.13	ROMAN	75
8.1.2.14	SEXDEC	75
8.1.2.15	STRING	75
8.1.3	Database	76
8.1.3.1	DAVERAGE	76
8.1.3.2	DCOUNT	76
8.1.3.3	DCOUNTA	76
8.1.3.4	DGET	77
8.1.3.5	DMAX	77
8.1.3.6	DMIN	77
8.1.3.7	DPRODUCT	78

The Calligra Sheets Handbook

	8.1.3.8 DSTDEV	78
	8.1.3.9 DSTDEVP	79
	8.1.3.10 DSUM	79
	8.1.3.11 DVAR	79
	8.1.3.12 DVARP	80
	8.1.3.13 GETPIVOTDATA	80
8.1.4	Date & Time	80
	8.1.4.1 CURRENTDATE	80
	8.1.4.2 CURRENTDATETIME	81
	8.1.4.3 CURRENTTIME	81
	8.1.4.4 DATE	81
	8.1.4.5 DATE2UNIX	81
	8.1.4.6 DATEDIF	82
	8.1.4.7 DATEVALUE	82
	8.1.4.8 DAY	82
	8.1.4.9 DAYNAME	83
	8.1.4.10 DAYOFYEAR	83
	8.1.4.11 DAYS	84
	8.1.4.12 DAYS360	84
	8.1.4.13 DAYSINMONTH	84
	8.1.4.14 DAYSINYEAR	85
	8.1.4.15 EASTERSONDAY	85
	8.1.4.16 EDATE	85
	8.1.4.17 EOMONTH	86
	8.1.4.18 HOUR	86
	8.1.4.19 HOURS	87
	8.1.4.20 ISLEAPYEAR	87
	8.1.4.21 ISOWEEKNUM	87
	8.1.4.22 MINUTE	88
	8.1.4.23 MINUTES	88
	8.1.4.24 MONTH	88
	8.1.4.25 MONTHNAME	89
	8.1.4.26 MONTHS	89
	8.1.4.27 NETWORKDAY	89
	8.1.4.28 NOW	90
	8.1.4.29 SECOND	90
	8.1.4.30 SECONDS	91
	8.1.4.31 TIME	91
	8.1.4.32 TIMEVALUE	91
	8.1.4.33 TODAY	92
	8.1.4.34 UNIX2DATE	92
	8.1.4.35 WEEKDAY	92
	8.1.4.36 WEEKNUM	93
	8.1.4.37 WEEKS	93
	8.1.4.38 WEEKSINYEAR	94
	8.1.4.39 WORKDAY	94
	8.1.4.40 YEAR	94

The Calligra Sheets Handbook

8.1.4.41	YEARFRAC	95
8.1.4.42	YEARS	95
8.1.5	Engineering	95
8.1.5.1	BASE	95
8.1.5.2	BESSELI	96
8.1.5.3	BESSELJ	96
8.1.5.4	BESSELK	97
8.1.5.5	BESSELY	97
8.1.5.6	BIN2DEC	97
8.1.5.7	BIN2HEX	98
8.1.5.8	BIN2OCT	98
8.1.5.9	COMPLEX	99
8.1.5.10	CONVERT	99
8.1.5.11	DEC2BIN	100
8.1.5.12	DEC2HEX	100
8.1.5.13	DEC2OCT	101
8.1.5.14	DELTA	101
8.1.5.15	ERF	101
8.1.5.16	ERFC	102
8.1.5.17	GESTEP	102
8.1.5.18	HEX2BIN	102
8.1.5.19	HEX2DEC	103
8.1.5.20	HEX2OCT	103
8.1.5.21	IMABS	103
8.1.5.22	IMAGINARY	104
8.1.5.23	IMARGUMENT	104
8.1.5.24	IMCONJUGATE	105
8.1.5.25	IMCOS	105
8.1.5.26	IMCOSH	105
8.1.5.27	IMCOT	106
8.1.5.28	IMCSC	106
8.1.5.29	IMCSCH	106
8.1.5.30	IMDIV	107
8.1.5.31	IMEXP	107
8.1.5.32	IMLN	107
8.1.5.33	IMLOG10	108
8.1.5.34	IMLOG2	108
8.1.5.35	IMPOWER	108
8.1.5.36	IMPRODUCT	109
8.1.5.37	IMREAL	109
8.1.5.38	IMSEC	109
8.1.5.39	IMSECH	110
8.1.5.40	IMSIN	110
8.1.5.41	IMSINH	110
8.1.5.42	IMSQRT	111
8.1.5.43	IMSUB	111

The Calligra Sheets Handbook

	8.1.5.44 IMSUM	111
	8.1.5.45 IMTAN	112
	8.1.5.46 IMTANH	112
	8.1.5.47 OCT2BIN	112
	8.1.5.48 OCT2DEC	113
	8.1.5.49 OCT2HEX	113
8.1.6	Financial	113
	8.1.6.1 ACCRINT	113
	8.1.6.2 ACCRINTM	114
	8.1.6.3 AMORDEGRC	114
	8.1.6.4 AMORLINC	115
	8.1.6.5 COMPOUND	115
	8.1.6.6 CONTINUOUS	116
	8.1.6.7 COUPNUM	116
	8.1.6.8 CUMIPMT	116
	8.1.6.9 CUMPRINC	117
	8.1.6.10 DB	117
	8.1.6.11 DDB	118
	8.1.6.12 DISC	118
	8.1.6.13 DOLLARDE	118
	8.1.6.14 DOLLARFR	119
	8.1.6.15 DURATION	119
	8.1.6.16 DURATION_ADD	120
	8.1.6.17 EFFECT	120
	8.1.6.18 EFFECTIVE	120
	8.1.6.19 EURO	121
	8.1.6.20 EUROCONVERT	121
	8.1.6.21 FV	122
	8.1.6.22 FV_ANNUITY	122
	8.1.6.23 INTRATE	122
	8.1.6.24 IPMT	123
	8.1.6.25 IRR	123
	8.1.6.26 ISPMT	124
	8.1.6.27 LEVEL_COUPON	124
	8.1.6.28 MDURATION	125
	8.1.6.29 MIRR	125
	8.1.6.30 NOMINAL	125
	8.1.6.31 NPER	126
	8.1.6.32 NPV	126
	8.1.6.33 ODDLPRICE	127
	8.1.6.34 ODDLYIELD	127
	8.1.6.35 PMT	128
	8.1.6.36 PPMT	128
	8.1.6.37 PRICEMAT	129
	8.1.6.38 PV	129
	8.1.6.39 PV_ANNUITY	130

The Calligra Sheets Handbook

8.1.6.40	RATE	130
8.1.6.41	RECEIVED	131
8.1.6.42	RRI	131
8.1.6.43	SLN	131
8.1.6.44	SYD	132
8.1.6.45	TBILLEQ	132
8.1.6.46	TBILLPRICE	133
8.1.6.47	TBILLYIELD	133
8.1.6.48	VDB	134
8.1.6.49	XIRR	134
8.1.6.50	XNPV	134
8.1.6.51	YIELDDISC	135
8.1.6.52	YIELDMAT	135
8.1.6.53	ZERO_COUPON	136
8.1.7	Information	136
8.1.7.1	ERRORTYPE	136
8.1.7.2	FILENAME	136
8.1.7.3	FORMULA	137
8.1.7.4	INFO	137
8.1.7.5	ISBLANK	137
8.1.7.6	ISDATE	138
8.1.7.7	ISERR	138
8.1.7.8	ISERROR	138
8.1.7.9	ISEVEN	139
8.1.7.10	ISFORMULA	139
8.1.7.11	ISLOGICAL	139
8.1.7.12	ISNA	140
8.1.7.13	ISNONTEXT	140
8.1.7.14	ISNOTTEXT	140
8.1.7.15	ISNUM	141
8.1.7.16	ISNUMBER	141
8.1.7.17	ISODD	141
8.1.7.18	ISREF	142
8.1.7.19	ISTEXT	142
8.1.7.20	ISTIME	142
8.1.7.21	N	143
8.1.7.22	NA	143
8.1.7.23	TYPE	143
8.1.8	Logical	144
8.1.8.1	AND	144
8.1.8.2	FALSE	144
8.1.8.3	IF	144
8.1.8.4	IFERROR	145
8.1.8.5	IFNA	145
8.1.8.6	NAND	145
8.1.8.7	NOR	146

The Calligra Sheets Handbook

8.1.8.8	NOT	146
8.1.8.9	OR	146
8.1.8.10	TRUE	147
8.1.8.11	XOR	147
8.1.9	Lookup & Reference	148
8.1.9.1	ADDRESS	148
8.1.9.2	AREAS	148
8.1.9.3	CELL	149
8.1.9.4	CHOOSE	149
8.1.9.5	COLUMN	149
8.1.9.6	COLUMNS	150
8.1.9.7	HLOOKUP	150
8.1.9.8	INDEX	151
8.1.9.9	INDIRECT	151
8.1.9.10	LOOKUP	151
8.1.9.11	MATCH	152
8.1.9.12	MULTIPLE.OPERATIONS	152
8.1.9.13	OFFSET	152
8.1.9.14	ROW	153
8.1.9.15	ROWS	153
8.1.9.16	SHEET	154
8.1.9.17	SHEETS	154
8.1.9.18	VLOOKUP	154
8.1.10	Math	155
8.1.10.1	ABS	155
8.1.10.2	CEIL	155
8.1.10.3	CEILING	155
8.1.10.4	COUNT	156
8.1.10.5	COUNTA	157
8.1.10.6	COUNTBLANK	157
8.1.10.7	COUNTIF	157
8.1.10.8	CUR	158
8.1.10.9	DIV	158
8.1.10.10	EPS	159
8.1.10.11	EVEN	159
8.1.10.12	EXP	160
8.1.10.13	FACT	160
8.1.10.14	FACTDOUBLE	160
8.1.10.15	FIB	161
8.1.10.16	FLOOR	161
8.1.10.17	GAMMA	162
8.1.10.18	GCD	162
8.1.10.19	G_PRODUCT	162
8.1.10.20	INT	163
8.1.10.21	INV	163
8.1.10.22	KPRODUCT	163

The Calligra Sheets Handbook

8.1.10.23 LCM	164
8.1.10.24 LN	164
8.1.10.25 LOG	165
8.1.10.26 LOG10	165
8.1.10.27 LOG2	166
8.1.10.28 LOGN	166
8.1.10.29 MAX	167
8.1.10.30 MAXA	167
8.1.10.31 MDETERM	168
8.1.10.32 MIN	168
8.1.10.33 MINA	169
8.1.10.34 MINVERSE	169
8.1.10.35 MMULT	170
8.1.10.36 MOD	170
8.1.10.37 MROUND	170
8.1.10.38 MULTINOMIAL	171
8.1.10.39 MULTIPLY	171
8.1.10.40 MUNIT	172
8.1.10.41 ODD	172
8.1.10.42 POW	172
8.1.10.43 POWER	173
8.1.10.44 PRODUCT	173
8.1.10.45 QUOTIENT	174
8.1.10.46 RAND	174
8.1.10.47 RANDBERNOULLI	174
8.1.10.48 RANDBETWEEN	175
8.1.10.49 RANDBINOM	175
8.1.10.50 RANDEXP	175
8.1.10.51 RANDNEGBINOM	176
8.1.10.52 RANDNORM	176
8.1.10.53 RANDPOISSON	176
8.1.10.54 ROOTN	177
8.1.10.55 ROUND	177
8.1.10.56 ROUNDDOWN	178
8.1.10.57 ROUNDUP	178
8.1.10.58 SERIESSUM	179
8.1.10.59 SIGN	179
8.1.10.60 SQRT	180
8.1.10.61 SQRTPI	180
8.1.10.62 SUBTOTAL	180
8.1.10.63 SUM	181
8.1.10.64 SUMA	182
8.1.10.65 SUMIF	182
8.1.10.66 SUMSQ	183
8.1.10.67 TRANSPOSE	183
8.1.10.68 TRUNC	183

The Calligra Sheets Handbook

8.1.11	Statistical	184
8.1.11.1	AVEDEV	184
8.1.11.2	AVERAGE	184
8.1.11.3	AVERAGEA	185
8.1.11.4	BETADIST	185
8.1.11.5	BETAINV	185
8.1.11.6	BINO	186
8.1.11.7	CHIDIST	186
8.1.11.8	COMBIN	187
8.1.11.9	COMBINA	187
8.1.11.10	CONFIDENCE	187
8.1.11.11	CORREL	188
8.1.11.12	COVAR	188
8.1.11.13	DEVSQ	188
8.1.11.14	EXPONDIST	189
8.1.11.15	FDIST	189
8.1.11.16	FINV	190
8.1.11.17	FISHER	190
8.1.11.18	FISHERINV	190
8.1.11.19	FREQUENCY	191
8.1.11.20	GAMMADIST	191
8.1.11.21	GAMMAINV	191
8.1.11.22	GAMMALN	192
8.1.11.23	GAUSS	192
8.1.11.24	GEOMEAN	192
8.1.11.25	HARMEAN	193
8.1.11.26	HYPGEOMDIST	193
8.1.11.27	INTERCEPT	194
8.1.11.28	INVBINO	194
8.1.11.29	KURT	194
8.1.11.30	KURTP	195
8.1.11.31	LARGE	195
8.1.11.32	LEGACYFDIST	195
8.1.11.33	LOGINV	196
8.1.11.34	LOGNORMDIST	196
8.1.11.35	MEDIAN	196
8.1.11.36	MODE	197
8.1.11.37	NEGBINOMDIST	197
8.1.11.38	NORMDIST	198
8.1.11.39	NORMINV	198
8.1.11.40	NORMSDIST	199
8.1.11.41	NORMSINV	199
8.1.11.42	PEARSON	199
8.1.11.43	PERCENTILE	200
8.1.11.44	PERMUT	200
8.1.11.45	PERMUTATIONA	200

The Calligra Sheets Handbook

8.1.11.46	PHI	201
8.1.11.47	POISSON	201
8.1.11.48	RANK	202
8.1.11.49	RSQ	202
8.1.11.50	SKEW	202
8.1.11.51	SKEWP	203
8.1.11.52	SLOPE	203
8.1.11.53	SMALL	203
8.1.11.54	STANDARDIZE	204
8.1.11.55	STDEV	204
8.1.11.56	STDEVA	204
8.1.11.57	STDEVP	205
8.1.11.58	STDEVPA	205
8.1.11.59	STEYX	206
8.1.11.60	SUM2XMY	206
8.1.11.61	SUMPRODUCT	206
8.1.11.62	SUMX2MY2	207
8.1.11.63	SUMX2PY2	207
8.1.11.64	SUMXMY2	207
8.1.11.65	TDIST	208
8.1.11.66	TREND	208
8.1.11.67	TRIMMEAN	208
8.1.11.68	TTEST	209
8.1.11.69	VAR	209
8.1.11.70	VARA	209
8.1.11.71	VARIANCE	210
8.1.11.72	VARP	211
8.1.11.73	VARPA	211
8.1.11.74	WEIBULL	212
8.1.11.75	ZTEST	212
8.1.12	Text	213
8.1.12.1	ASC	213
8.1.12.2	BAHTTEXT	213
8.1.12.3	CHAR	213
8.1.12.4	CLEAN	214
8.1.12.5	CODE	214
8.1.12.6	COMPARE	214
8.1.12.7	CONCATENATE	215
8.1.12.8	DOLLAR	215
8.1.12.9	EXACT	215
8.1.12.10	FIND	216
8.1.12.11	FINDB	216
8.1.12.12	FIXED	217
8.1.12.13	JIS	217
8.1.12.14	LEFT	218
8.1.12.15	LEFTB	218

The Calligra Sheets Handbook

8.1.12.16	LEN	219
8.1.12.17	LENB	219
8.1.12.18	LOWER	219
8.1.12.19	MID	220
8.1.12.20	MIDB	220
8.1.12.21	PROPER	221
8.1.12.22	REGEXP	221
8.1.12.23	REGEXPRE	221
8.1.12.24	REPLACE	222
8.1.12.25	REPLACEB	222
8.1.12.26	REPT	223
8.1.12.27	RIGHT	223
8.1.12.28	RIGHTB	224
8.1.12.29	ROT13	224
8.1.12.30	SEARCH	224
8.1.12.31	SEARCHB	225
8.1.12.32	SLEEK	225
8.1.12.33	SUBSTITUTE	226
8.1.12.34	T	226
8.1.12.35	TEXT	227
8.1.12.36	TOGGLE	227
8.1.12.37	TRIM	227
8.1.12.38	UNICHAR	228
8.1.12.39	UNICODE	228
8.1.12.40	UPPER	228
8.1.12.41	VALUE	229
8.1.13	Trigonometric	229
8.1.13.1	ACOS	229
8.1.13.2	ACOSH	229
8.1.13.3	ACOT	230
8.1.13.4	ASIN	230
8.1.13.5	ASINH	230
8.1.13.6	ATAN	231
8.1.13.7	ATAN2	231
8.1.13.8	ATANH	232
8.1.13.9	COS	232
8.1.13.10	COSH	232
8.1.13.11	CSC	233
8.1.13.12	CSCH	233
8.1.13.13	DEGREES	233
8.1.13.14	PI	234
8.1.13.15	RADIANS	234
8.1.13.16	SEC	234
8.1.13.17	SECH	235
8.1.13.18	SIN	235
8.1.13.19	SINH	235
8.1.13.20	TAN	236
8.1.13.21	TANH	236

Abstract

Calligra Sheets is a full featured spreadsheet program.

Chapter 1

Introduction

This handbook is dedicated to the memory of [Visicalc](#).

IMPORTANT

Please check <http://docs.kde.org> for updated versions of this document.

Calligra Sheets is a full featured spreadsheet program. It is part of the Calligra productivity suite for KDE.

Other Calligra applications include Calligra Words, (word processing), Calligra Stage (slide presentation creator) among others.

You might care to visit <http://www.kde.org> for more information about KDE in general, or the Calligra web site at <http://www.calligra.org>

Chapter 2

Calligra Sheets Basics

Pamela Robert

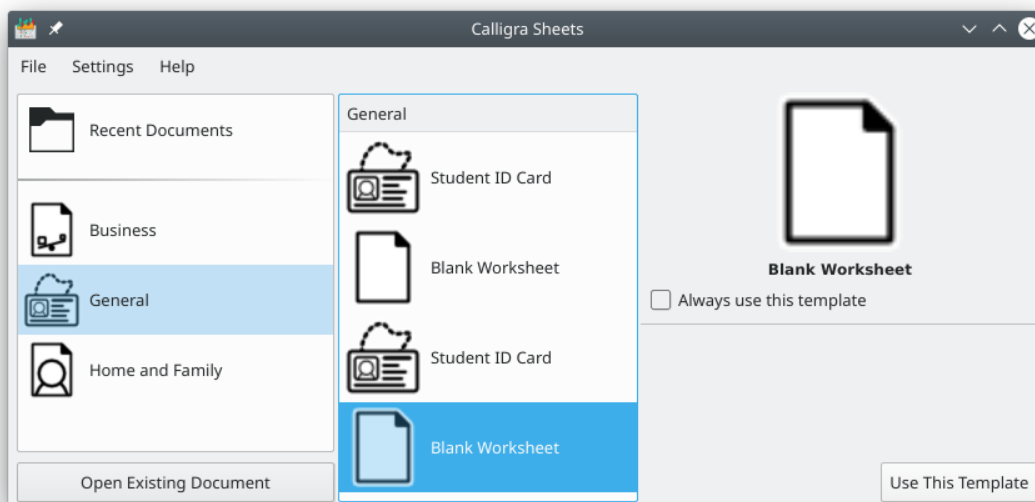
NOTE

Like the rest of KDE, Calligra Sheets is highly configurable, which can cause problems for readers trying to compare the text in a document such as this with what they see on the version of Calligra Sheets running on their desktop. To cut down on some of the possibilities for confusion, it is suggested that when you first start to use Calligra Sheets you set the default options in all pages of the Calligra Sheets configuration dialog (obtained by selecting **Settings** → **Configure Sheets...**).

2.1 Spreadsheets for Beginners

This section attempts to explain by example what a spreadsheet program such as Calligra Sheets actually does, and why it is such a useful tool in any situation where you have to deal with numbers. If you have already used a spreadsheet program you may wish to skip to the next section.

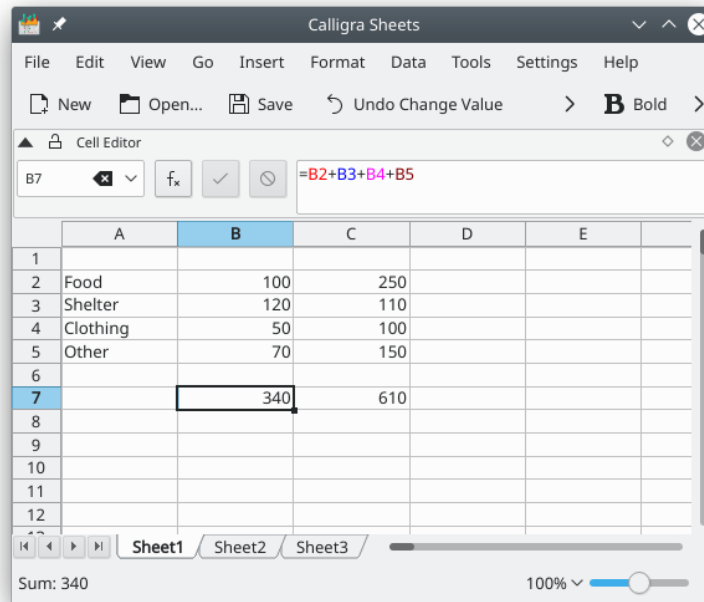
The first thing to do is to start up Calligra Sheets. You can do this by left clicking on a Calligra Sheets icon if there is one on your desktop or panel, or you can select **Office** → **Calligra Sheets** from the application launcher.



The Calligra Sheets Handbook

When it has started you will be given the choice of opening a recent document, creating a new document from a template (with templates categories) or opening an existing document . Select the **General** category on the left and choose the **Blank Worksheet** template. Then click the **Use This Template** button.

Looking at Calligra Sheets once it has started up, you will see a sheet of empty rectangular cells arranged in numbered rows and lettered columns. This is where you enter data or formula, text or charts.



Now, enter the text and values shown in the first 5 rows of the above screenshot into the same cells of your spreadsheet. Ignore what is in row 7 for the moment. To enter anything into a cell first select the cell by left clicking inside it, then type whatever you want, then press **Enter** or use the arrow keys to move the selection point to another cell.

What we have entered so far could be a simple budget for the next two months, listing how much we think we will be spending for Food, Shelter, Clothing and any Other expenditure. Now select cell B7 (column B, row 7), type in `=B2+B3+B4+B5` and press **Enter**. Because it begins with a = symbol Calligra Sheets sees this as a formula, something it has to calculate, in this case by adding together the values in the 4 cells B2 to B5, and what is shown in the cell B7 is the result of that calculation.

You could enter a similar formula into cell C7, except that in this case it would have to be `=C2+C3+C4+C5`, but there is an easier way which is to Copy cell B7 and Paste it into C7. Calligra Sheets will automatically adjust the cell references from B.. to C.. when the Paste is done.

At this point you may think that Calligra Sheets is doing no more than you could manage with pencil, paper and a calculator, and you could be right, but remember that this is a very small example of a spreadsheet, doing simple calculations on only a few numbers. For any reasonably amount of values or data using a spreadsheet to do the calculations is much quicker and more accurate than doing them manually.

Also, a spreadsheet lets you play the 'What if?' game. Because each formula is automatically recalculated whenever any of the values it refers to are changed, you can quickly see what happens if you alter any of them. Using our example you can see the effect of reducing the amount spent on food in December by just entering a new value into cell C2. If you had a spreadsheet that modelled the greenhouse effect accurately you could perhaps see the effect of a 50 percent reduction in the amount of methane released into the atmosphere.

2.2 Selecting Cells

You can select a single cell or a rectangular area of cells in the spreadsheet. The selected cell(s) are displayed with a thick black border.

YOU CAN SELECT A SINGLE CELL IN ONE OF THE FOLLOWING WAYS

- left click on it
- enter the cell reference (for example **B5**) into the cell reference box at the left end of the **Cell Editor** tool options and press **Enter**
- use the **Go** → **Goto Cell...** menu option

You can also steer your way around with the **arrow** keys. Pressing the **Enter** key will move the current selection one position up, down, left or right depending on the setting in the **Interface** page of Calligra Sheets's [configuration dialog box](#).

If you hold the **Shift** key down while using the **arrow** keys the selection will move to the start or end of the block of occupied cells.

To select an area of contiguous cells drag the mouse cursor across the desired area with the left button held down, or enter the references of the top left and bottom right cells separated by a colon into the **Cell Editor** cell reference box (for example **B7:C14**) and press **Enter**, or enter these cell references in a similar format into the dialog box brought up by **Go** → **Goto Cell...**

You can also select an area of cells by selecting the cell in one corner of the wanted area then holding the **Shift** key down while using the left mouse button to select the cell in the opposite corner.

To select non-contiguous cells, click on the first cell you want to select then hold the **Ctrl** key and select the other cells.

To select a complete row or column of cells left click on the row number at the left of the worksheet or on the column letters at the top. To select adjacent rows or columns drag the mouse pointer over the appropriate row numbers or column letters with the left button held down.

To select non-contiguous rows or columns of cells, click on the first row number or column letter then hold the **Ctrl** key and select the other rows or columns of cells.

2.3 Entering Data

Entering data into a cell can be as simple as selecting the cell, typing your data, then pressing **Enter** or moving the selection to another cell with one of the **arrow** keys. Depending on how you enter the data, Calligra Sheets will interpret it as a number, date, time or text:

- Numbers are entered in the obvious way; **123**, **-123**, **456.7** or in scientific notation **-1.2E-5**.
- Dates should be entered in your 'System' format, as defined in the System Settings in **Locale** → **Country/Region & Language** → **Date & Time** tab. If, for example, you are using the DD/MM/YYYY form you should enter **30/03/2012** for 30th March 2012. Leading zeroes can be omitted from the day and month fields and only the last one or two digits of the year need to be entered if the date is in the current century, for example **9/1/9** for 9th January 2009.
- Times should also be entered using the 'System' format. For example if you are using a 12 hour clock then enter times in HH:MIN am | pm or HH:MIN:SS am | pm format such as **9:42 am** or **10:30:52 pm**.
- Calligra Sheets defines any input data as 'text' if it cannot recognize the data as being a number, date or time.

NOTE

By default, Calligra Sheets right justifies numbers, dates and times within a cell and left justifies anything else. This can be a useful guide to whether you have entered a date or time in the correct format. But remember that how items are displayed can be changed by altering the [cell format](#).

The main text entry box in the **Cell Editor** tool options provides an easy way of editing the contents of a selected cell. Press **Enter** or left click on the green tick mark when you are happy with what you have entered, or click on the red cross to cancel your edits.

2.3.1 Generic Cell Format

Calligra Sheets uses the 'Generic' cell format as default. As long as this format is used, Calligra Sheets autodetects the actual data type depending on the current cell data. For example if you enter some text into a cell and later enter a number into the same cell, Calligra Sheets automatically interprets the new data as a number. If you want to define the type of data yourself, you can explicitly set it in the [cell format](#). You can change the format back to 'Generic' at any time.

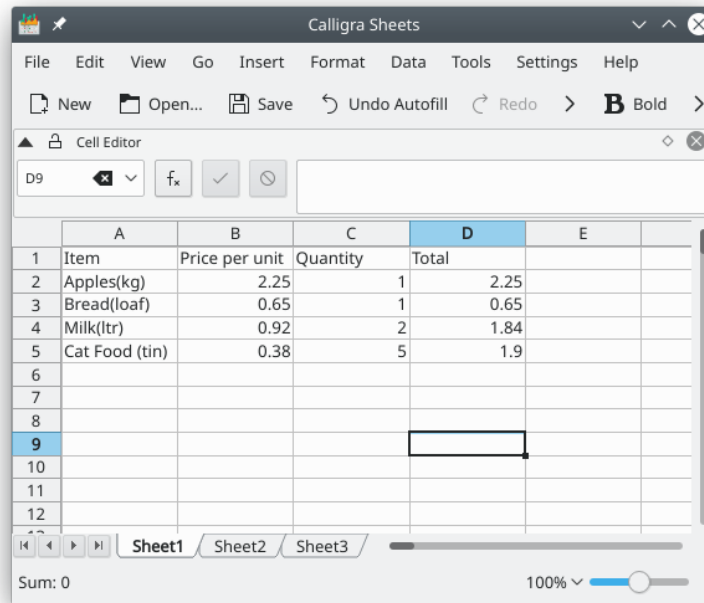
2.4 Copy, Cut and Paste

At first glance, Calligra Sheets's **Cut**, **Copy** and **Paste** appear to be similar to these functions in other KDE applications. Having selected a cell or cells, you can choose **Copy** or **Cut** from the **Edit** menu or from the drop down menu you get by holding the right mouse button down on a selected cell. You can also use the shortcuts **Ctrl+C** or **Ctrl+X**, then move the selection to the target cell and choose **Paste** or press **Ctrl+V**. However there are some subtleties associated with these functions in Calligra Sheets and these are discussed below.

If a cell contains a formula then the formula itself is copied rather than the displayed result, and if the formula contains a reference to another cell, then that reference is changed by the **Cut** or **Copy** and **Paste** operation to point to the cell that is in the same relative position as in the original cell. For example if cell A2 contains the formula **=B3** and is copied to C4, cell C4 will contain **=D5**.

This may seem to be a rather strange way of doing a copy, but 99 percent of the time it is exactly what is wanted (if it is not then see the section about [absolute cell references](#)). For example in the simple shopping list shown below, cell D2 should contain **=B2 * C2**, D3 should be **=B3 * C3**, D4 should be **=B4 * C4** and so on. Instead of having to enter a different formula in each cell, you can just enter the first formula into D2 and then copy it into the cells below, letting Calligra Sheets adjust the cell references to suit.

The Calligra Sheets Handbook



2.4.1 Copying and Pasting Cell Areas

In the above example D2 can be copied into all three cells D3 to D5 at once by just copying D2 then selecting the complete cell area D3:D5 before doing the paste.

A rectangular area of cells can be cut or copied in one operation by selecting the area before doing the cut or copy. Then select the top left corner cell of the area you want to paste into before doing the paste.

If you cut or copy a rectangular area of cells, say B2:C3, and paste it into a larger area such as A10:D13 the original pattern of cells will be repeated to fill the target area.

Calligra Sheets also provides a 'Drag and Copy' method for copying cells down into other cells immediately below or to the right of the original cell(s). To use this method select the cell(s) to be copied then position the mouse pointer over the small black square at the bottom right corner of the selected cell(s) so the cursor changes to a double headed arrow. Then hold the left mouse button down while you drag the selected cell(s) as far as you wish. Note that cell references in formulae are incremented according to the relative position change. Absolute references are not changed.

2.4.2 Other Paste Modes

A cell may contain text, a value, or a formula, and may also contain special font, border or background [formatting information](#). Calligra Sheets has special versions of Paste that let you handle these items in different ways.

Edit → **Special Paste...** brings up the **Special Paste** dialog box. By selecting the appropriate item from the left part of this dialog you can choose to paste **Everything**, just **Text**, the cell **Format**, any **Comment** in the cell(s) or **Everything without border**. The items in the right part of this dialog box allow you to do simple [arithmetic on an area of cells](#).

Paste with Insertion... inserts the copied cell(s) into the sheet by moving the cells that would otherwise be overwritten a suitable number of rows or columns down or to the right. It can also be used to insert complete copied row(s) or column(s) into the worksheet.

2.5 Insert and Delete

Use the **Del** key or **Edit** → **Clear** → **Contents** to remove the text, value or formula from selected cell(s), row(s) or column(s) without affecting anything else.

To delete everything in the selected cell(s), row(s) or column(s), including comments and special formatting, choose the **All** option from the **Edit** → **Clear** menu or from the pop up menu you get when you right click on a selection.

To remove selected row(s) or column(s) completely, use the **Delete Rows** or **Delete Columns** options from the right mouse button pop up menu.

If you select a cell or cells and choose **Remove Cells...** from the right mouse button pop up menu, you can then choose whether other cells in the worksheet will be moved up or to the left to fill in the space left by the cell(s) you have chosen to remove.

If you want to insert new, blank, row(s) or column(s) into the sheet, select row(s) or column(s) where you wish the new row(s) or column(s) to be placed and choose the **Insert Rows**, **Insert Columns** option from the right mouse button pop up menu.

You can insert new cells into the worksheet by selecting the area where you want them to appear then choosing the **Insert Cells...** option from the right mouse button pop up menu. You will then be asked whether the existing cell(s) in the selected area should be moved down or to the right to make room for the new ones.

2.6 Simple Sums

If the first character in a cell is an equals sign (=) Calligra Sheets will take the cell contents to be a formula which is to be calculated. The result of the calculation will be displayed in the cell rather than the formula itself. For example, enter **=2+3** into a cell and it should display 5.

More usefully, a formula can contain references to other cells, so that **=B4+A3** will calculate the sum of the values in cells B4 and A3, and this calculation will be updated whenever cells B4 or A3 are changed.

As well as addition, a formula can make use of the - symbol for subtraction, * for multiplication, and / to perform division. The round bracket symbols (and) can also be used as in normal algebra, so you could enter more complex formulae such as **=(B10 + C3) *5 - F11) / 2**

Cells containing a formula will be marked with a small blue triangle at the bottom left corner if the **Show formula indicator** check box in the **Format** → **Sheet** → **Sheet Properties** dialog is checked.

Calligra Sheets also includes a large number of built-in functions for applications such as statistical, trigonometrical and financial calculations. Their use will be examined in more depth in a [later section](#) of this manual, but if you are interested at this stage choose **Function...** from the **Insert** menu and take a look through the **Function** dialog box that will be displayed.

For the time being, however, the **SUM** function may be of interest as it calculates the sum of all values in a specified area of cells. For example **=SUM(B4 : C10)** calculates the sum of all values in the cell area B4 to C10.

If Calligra Sheets displays **#VALUE!** when you have entered your formula this usually means that it cannot understand what you have entered, but if the row of symbols ends with a small red arrow this just means that the cell is not wide enough to display the complete result, in which case you should either make the cell(s) wider or change their [format](#) so that the result does fit properly.

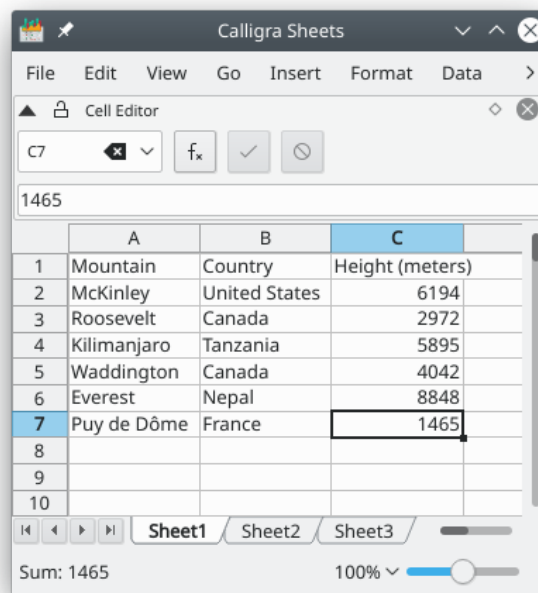
2.6.1 Recalculation

If the **Automatic recalculation** box in the **Format** → **Sheet** → **Sheet Properties** dialog box is checked, Calligra Sheets will recalculate the values of cells whenever anything that affects them is changed in the sheet.

When **Automatic recalculation** is not checked for the current sheet, you can instruct Calligra Sheets to perform a recalculation at any time by using the **Recalculate Sheet** or **Recalculate Document** option in the **Tools** menu or their shortcuts **Shift+F9** or **F9**.

2.7 Sorting Data

In the simple example shown below, the data consist of the names and countries of a number of mountains together with their height above sea level. Calligra Sheets can sort data such as this in different ways.



We may want the data sorted so that the names are in alphabetical order. To do this select the area containing the data (A2:C7 in this case) and choose **Sort...** from the **Data** menu. This opens the **Sort** dialog box.

Sorting is done alphanumerically, and the default is case sensitive, numbers coming before uppercase letters which come before lowercase letters, so that cells containing the entries **Cat**, **bar**, **77** and **Bat** would be sorted into the following order: *77 Bat Cat bar*.

In the **Direction** area of this dialog box select to sort in rows or columns. If you check the **First row contains column headers** or **First column contains row headers** box data in the first row or column will not be included in the sort operation.

The rows or columns are sorted in the specified order, which can be changed using the **Move Up** and **Move Down** buttons. Using the example in the above screenshot, choosing column B as the first key and column C as the second would sort the data by country and, for each country, by height.

Uncheck the option **Case Sensitive** to get a sort not depending on capitalization and switch the sort order between **Ascending** and **Descending** by clicking on the cells in the column **Sort Order**.

The **Details** » extension of the dialog allows you to sort using the order of items in a custom list such as January, February... instead of alphanumerically. The cell format is moved with the cell content, if you select **Copy cell formatting (Borders, Colors, Text Style)**.

2.8 The Status bar Summary Calculator

The left hand end of the Status bar by default shows a summary of the values in the selected cell(s). According to the setting of the **Function shown in status bar** combo box in the **Interface** page of Calligra Sheets's configuration dialog the summary can be:

None

No summary calculation is performed.

Average

The value displayed is the average of the values in the selected cells.

Count

The value displayed is the number of cells containing numeric values.

CountA

The value displayed is the number of non empty cells.

Max

The value displayed is the maximum of the values in the selected cells.

Min

The value displayed is the minimum of the values in the selected cells.

Sum

The value displayed is the sum of the values in the selected cells.

The method of calculation can also be changed by right clicking on the summary calculation result area of the Status bar and choosing an item from the pop up menu.

2.9 Saving your Work

Calligra Sheets saves the complete document, which may include more than one worksheet, as a single document file.

If you have created a new document, or want to save an existing one under a different name, use **File** → **Save As...** This will bring up KDE's common **Save Document As** dialog box. Choose the folder where you want to save the document and enter a suitable file name into the **Name** text box. Calligra Sheets documents are normally automatically saved with a **.ods** extension, you do not need to add this to the filename but do make sure that the **Filter** selection is set to **OpenDocument SpreadSheet**.

To save your document without changing its name, just use **File** → **Save**.

You can also save a Calligra Sheets document in a foreign format selecting the format from the **Filter:** combo box.

When you save a modified version of an existing document Calligra Sheets will keep the previous version as a backup file, adding a **~** to the end of the filename.

Calligra Sheets can provide some protection against losing your work because of a computer crash or because you have closed Calligra Sheets without saving the current document. It does

this by automatically saving the latest version of the document you are working on every few minutes using a modified file name. The autosaved version is normally removed when you next save your document, so that it will only exist if it is more up to date than the version that was saved manually. When you open a document Calligra Sheets checks to see if an autosaved version exists, and if it finds one it will offer to open that instead.

Autosaved documents are saved with a file name of the form `.yourfilename.autosave` (note the leading period), so that `spread1.ods` would be autosaved as `.spread1.ods.autosave`. The autosave feature is user configurable in the [settings dialog](#).

2.9.1 Templates

If you are going to be creating a lot of similar documents you can save yourself time and trouble by first creating a template and then using that as the basis for the individual documents.

To do this first create a document containing the common elements, then save it as a template by choosing **File** → **Create Template From Document**. Doing this opens the **Create Template** dialog box. Enter a name for your new template into the **Name:** text box and press **OK**. The next time you start a new document by choosing **File** → **New** or when you next start Calligra Sheets the startup dialog window will give you the option of creating the new document from your template.

The **Create Template** dialog box also lets you choose a different picture to be displayed above the template name in the startup dialog window, and lets you save your templates under different group names, which will appear as different pages in the dialog.

Templates are stored as `.kst` files under `~/.kde/share/apps/tables/templates/`.

2.10 Printing a Spreadsheet

Printing a spreadsheet is basically done by selecting **File** → **Print...** which brings up KDE's common **Print** dialog box where you can choose, among other options, the printer to be used, the number of copies and whether all or only selected pages are to be printed.

By default Calligra Sheets will print all items in the current worksheet, but you can restrict this by first selecting the area that you want to be printed then choosing **Define Print Range** from the **Format** → **Print Range** sub menu.

Calligra Sheets will print as many pages as are necessary to include all items in the current worksheet. You can quickly see how a worksheet will be split into separate pages for printing by checking the **View** → **Page Borders** box. The boundaries of each printed page will then be marked by colored lines in the worksheet.

For a more detailed view of what is to be sent to the printer, including anything you have asked to be included in the page headers and footers (see below), choose **File** → **Print Preview...**

To improve the appearance of the printed output, you can change the fonts, colors, borders and sizes of the cells in the worksheet, see the [Spreadsheet Formatting](#) section for more details about how to do this.

You can also use the **Page Layout** dialog box, invoked by selecting **Format** → **Page Layout...**, to change the orientation of the printed pages, the paper size (this should be suitable for your printer) and the size of the page borders.

The **Sheet** provides more options. The **Print settings** section lets you select whether or not to print the grid, comment indicators and formula indicators, objects and charts. The **Repetitions on each page** section allows you to repeat selected column(s) or row(s) on each printed page. In the section **Scaling** you can set a scalefactor or the page limits for the print.

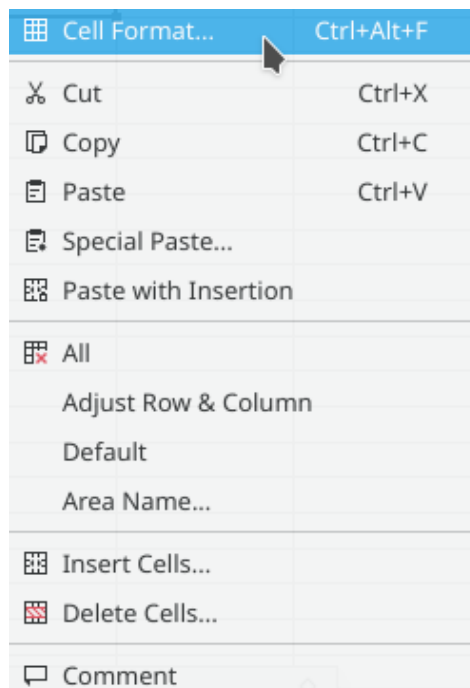
Chapter 3

Spreadsheet Formatting

Pamela Robert
Raphael Langerhorst
Anne-Marie Mahfouf

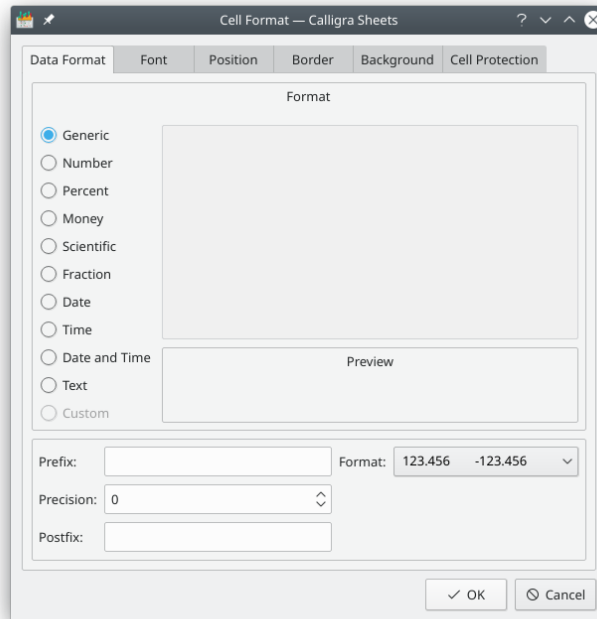
3.1 Cell Format

To change the format and appearance of selected cell(s), row(s) or column(s) use the **Cell Format...** option from the **Format** menu or from the right mouse button popup menu.



This will bring up the **Cell Format** dialog box which has several tabbed pages:


3.1.1 Data Formats and Representation




The **Data Format** page of the **Cell Format** dialog box lets you control how the values of cells are displayed.

The top part of this page lets you select the format to be used when displaying numeric values, dates or times. A **Preview** pane allows you to see the effect of the new format.

You can set the same data format for a row or a column by selecting the row or column and calling the **Cell Format** dialog with the right mouse button.

NOTE
You can increase the precision decimal for any number in **Generic, Number, Percent, Money** or **Scientific** formats using the **Increase precision** icon in the **Format** toolbar: 

You can decrease the precision decimal for any number in **Generic, Number, Percent, Money** or **Scientific** formats using the **Decrease precision** icon in the **Format** toolbar: 

Generic

This is the default format and Calligra Sheets autodetects the actual data type depending on the current cell data. By default, Calligra Sheets right justifies numbers, dates and times within a cell and left justifies anything else.

If the **Generic** format does not suit you, you can change to a specific format among the choices below.

Number

The number notation uses the notation you globally choose in System Settings in **Locale** → **Country/Region & Language** → **Numbers**. Numbers are right justified by default.

Percent

When you have a number in the current cell and you switch the cell format from **Generic** to **Percent**, the current cell number will be multiplied by 100.

For example if you enter 2 and set the cell format to **Percent**, the number will then be 200%. Switching back to **Generic** cell format will bring it back to 2.



You can also use the **Percent** icon in the **Format** Toolbar:

Money

The **Money** format converts your number into money notation using the settings globally fixed in System Settings in **Locale** → **Country/Region & Language** → **Money**. The currency symbol will be displayed and the precision will be the one set in System Settings.

You can also use the **Money Format** icon in the **Format** toolbar to set the cell formatting to

look like your current currency: 

Scientific

The **Scientific** format changes your number using the scientific notation. For example, 0.0012 will be changed to 1.2E-03. Going back using **Generic** cell format will display 0.0012 again. The **Generic** cell data format does not keep scientific notation so if you want this notation, you have to specify it using this menu item.

Fraction

The **Fraction** format changes your number into a fraction. For example, 0.1 can be changed to 1/8, 2/16, 1/10, etc. You define the type of fraction by choosing it in the field on the right. If the exact fraction is not possible in the fraction mode you choose, the nearest closest match is chosen. For example: when we have 1.5 as number, we choose **Fraction** and **Sixteenths 1/16** the text displayed into cell is "1 8/16" which is an exact fraction. If you have 1.4 as number in your cell and you choose **Fraction** and **Sixteenths 1/16** then the cell will display "1 6/16" which is the nearest closest Sixteenth fraction.

Date

To enter a date, you should enter it in one of the formats set in System Settings in **Locale** → **Country/Region & Language** → **Date & Time**. There are two formats set here: the date format and the short date format.

A random natural number NN will be transformed in the date from 30st December 1899 (which is 0) with the number of days NN added. For example if you have a cell with 100 and you choose **Date** format, "1900-04-09" will be displayed in the cell which is 100 days after 30st December 1899. This starting date is two days early as it was a bug in Lotus 123 and then it stayed that way in Excel in order to keep compatibility. Few people will need to calculate from 1st January 1900 anyway and adding 9 days to 1st November 2000 for example will give you 10th November 2000 so all normal calculations on dates are correct.

NOTE

When a cell is in the **Date** format, you can drag this cell down as you do with numbers and the next cells will also get dates, each date being increased by one day.

Time

This formats your cell content as a time. To enter a time, you should enter it in the **Time format** set in System Settings in **Locale** → **Country/Region & Language** → **Date & Time**. In the **Cell Format** dialog box you can set how the time should be displayed by choosing one of the available time format options. The default format is the system format set in System Settings. When the number in the cell does not make sense as a time, Calligra Sheets will display 00:00 in the global format you have in System Settings.

Date and Time

This formats your cell content as date and time. To enter a date and a time, you should enter it in the **Time format** set in System Settings in **Locale** → **Country/Region & Language** → **Date & Time**. In the **Cell Format** dialog box you can set how the date and time should be displayed by choosing one of the available date and time format options. The default format is the system format set in System Settings. When the number in the cell does not make sense as a date and time, Calligra Sheets will display 00:00 in the global format you have in System Settings.

Text

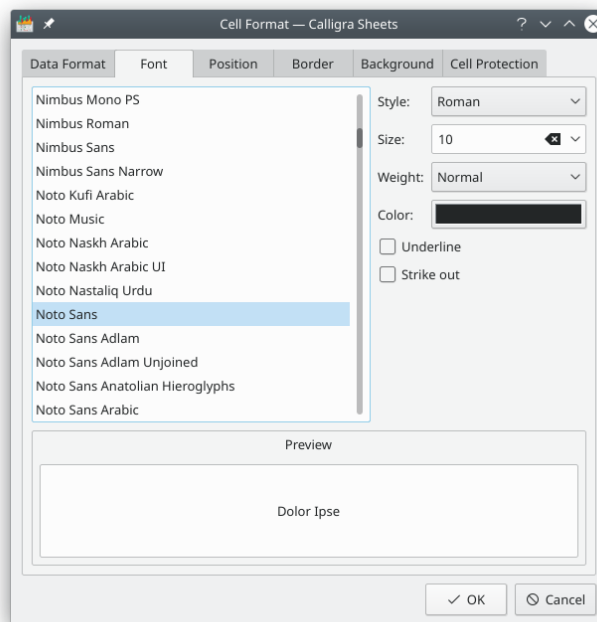
This formats your cell content as text. This can be useful if you want a number treated as text instead as a number, for example for a ZIP code. Setting a number as text format will left justify it. When numbers are formatted as text, they cannot be used in calculations or formulas. It also change the way the cell is justified.

Custom

Does not work yet. To be enabled in the next release.

The lower part of the **Data Format** page lets you add a **Prefix**: such as a \$ symbol at the start of each item or a **Postfix**: such as \$HK to the end. You can also control how many digits are displayed after the decimal point for numeric values, whether positive values are displayed with a leading + sign and whether negative values are shown in red.

3.1.2 Fonts and Text Settings



The **Font** page lets you select the font family, **Style**, **Size**, **Weight** and **Color** for the current cell, including some additional options like underlined or striked out text. The lower part of the page gives a **Preview** of the selected text format.

The default font is set for all cells in the **Format** → **Style Manager** menu with the currently used style.

Style:

Choose the style for your font for the currently selected cells. When you select several cells with different styles, the displayed style is set to **Varying (No Change)** and leaving it that way will keep all your current style settings for each cell. Changing to **Roman** for example will change all the selected cells style text to **Roman**.

Size:

Choose the size for your font for the currently selected cells. When you select several cells with different sizes, the displayed size is set to (no number written) and leaving it that way will keep all your current size settings for each cell. Changing to **14** for example will change all the selected cells font size to **14**.

Weight:

Choose the weight for your font for the currently selected cells. When you select several cells with different font weight, the displayed weight is set to **Varying (No Change)** and leaving it that way will keep all your current weight settings for each cell. Changing to **Bold** for example will change all the selected cells font weight to **Bold**.

Color:

Choose the color for the currently selected cells' text. Clicking on the color bar will bring you the standard KDE **Select Color** dialog where you will be able to choose the new color.

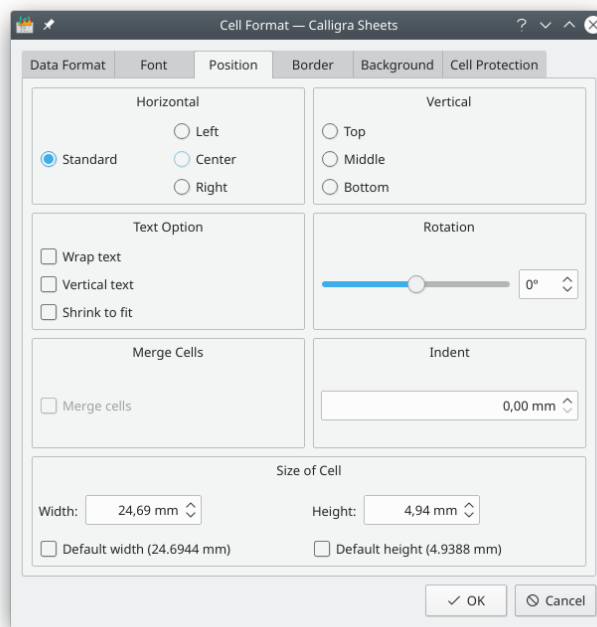
Underline

Underlines the currently selected cells' text if checked. This is not checked per default.

Strike out

This will strike out the currently selected cells' text if this is checked. This is not checked per default.

3.1.3 Text Position and Rotation



From the **Position** page you can control the position of text within a cell by making suitable selections in the **Horizontal** and **Vertical** areas or by setting the **Indent** value. You can also choose to have the text appear vertically rather than horizontally, or even at an angle.

Horizontal

Set the content position horizontally in the cell. **Standard** is default and is set from the data format you choose. **Left** means the content will be displayed on the left of the cell. **Center** means the content will be in the center horizontally in the cell. **Right** means the content of the cell will be displayed on the right of the cell.

Vertical

Set the content position vertically in the cell. **Top** means the content will be displayed on top of the cell. **Middle** means the content will be in the middle vertically in the cell. **Bottom** means the content of the cell will be displayed at the bottom of the cell.

Text Option

This is only available when the rotation is 0. **Wrap text** wraps the text so it fits in the previous cell size. If this is not checked, the text will stay on one line.

Vertical text puts your text vertically.

Rotation

Your text will appear oriented in the angle you set here. Positive values will move it counter-clockwise and negative values will move it clockwise.

Merge Cells

When checked, this has the same effect as **Format** → **Merge Cells**. You need to have at least two consecutive cells selected. Those consecutive cells are then merged into a bigger one.

When a merged cell is selected and when you uncheck this, then all cells come back to their original size as before the merging. It has the same effect as **Format** → **Dissociate Cells**.

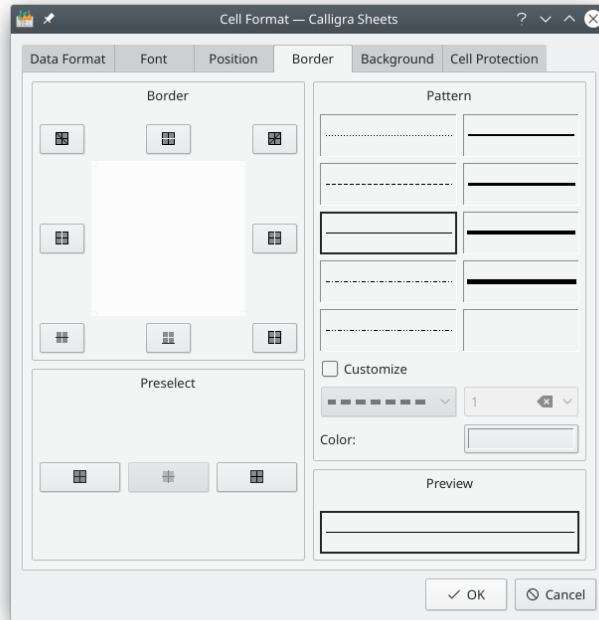
Indent

Set the amount of indent that will be used in the cell when you choose the **Increase Indent/Decrease Indent** actions from the toolbar. These actions are not enabled by default in the toolbar.

Size of Cell

You set here the size of the cell, either a custom width and height or choose the default width and height.

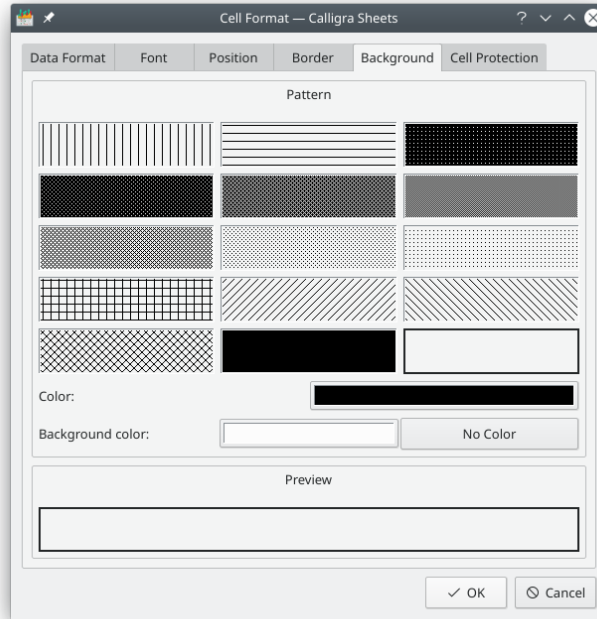
3.1.4 Cell Border



The **Border** page lets you set the appearance of the cell borders. If you have selected more than one cell you can apply different styles to the borders between the cells and that surrounding the selected area.

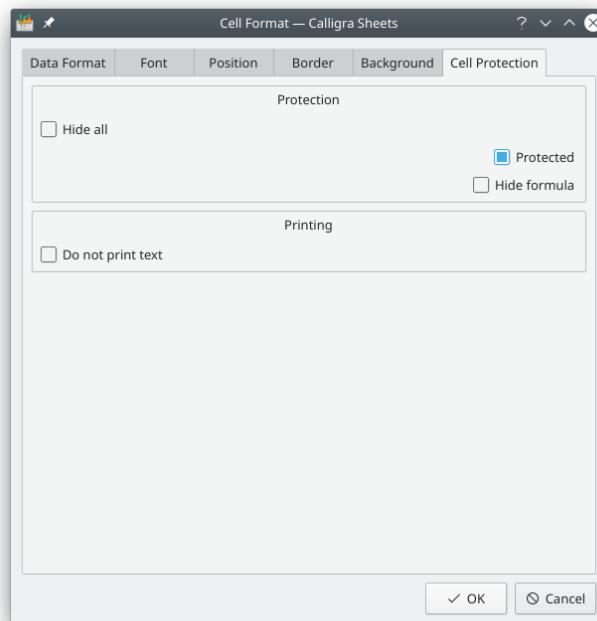
First select the pattern and color from the **Pattern** section of the **Border** page then apply that to different parts of the border by clicking on the appropriate button in the **Border** section, or on one of the **Preselect** buttons. The left hand button in the **Preselect** section will clear any previously applied border(s). Note that you can also add a diagonal strike-through line to the cell(s).

3.1.5 Cell Background



The cell background pattern and color can be selected from the **Background** page. Simply choose a desired **Pattern**, then select the pattern **Color** and the **Background color**. At the bottom of this page you can see a **Preview** of the configured cell background.

3.1.6 Cell Protection



You can change the way the content of a cell is protected in the **Cell Protection** page.

All cells are protected by default (that means cell content cannot be changed) and for the cell protection to be active you also need to protect the sheet using the **Tools** → **Protect Sheet...** menu and to provide a password. You can also hide the cell formula in order to protect the way you calculate the formula. This also needs to enable sheet protection to work. You can hide the cell content with **Hide all** and again this needs sheet protection. You can learn more about all these settings in the [Advanced Calligra Sheets chapter, Protection section](#).

Hide all

This hides the cell content and works only when the sheet is protected which means that changing the **Hide all** attribute of a cell has no effect unless the sheet is protected. Whether the cell itself is protected or not does not matter.

When **Hide all** is selected, **Protected** and **Hide formula** are disabled as when the sheet is protected **Hide all** hides the cell content and the formula and thus masks and protects the cell content.

Protected

If checked, the cell content will be protected. This is the default behaviour. You need to protect the whole sheet using the **Tools** → **Protect Sheet...** menu for this individual cell protection to work. When a cell is protected, its content cannot be changed.

Hide formula

When this is checked, the cell is still visible. However, its contents do not appear in the **Formula** bar. Hiding formula is only working for cells that contain formulae so the user cannot view the formula. And the sheet must be protected for this to work.

Do not print text

If you check **Do not print text** then the text in the cell will not be printed. This is unchecked per default which means that the cell text will always be printed by default.

3.2 Conditional Cell Attributes

You can make the appearance of a cell change according to the value it contains, useful perhaps if you are using Calligra Sheets to keep track of your household expenses and want to highlight any item greater than, say, one thousand dollars.

To do this select the cell(s) then choose **Conditional Styles...** from the **Format** menu. This will bring up the **Conditional Styles** dialog box where you can make the font type and color of a cell change when the value meets one or more conditions. Note that the second and third conditions only apply if the previous condition(s) are not met.

Use **Clear** → **Conditional Styles** from the **Edit** menu to clear any conditional attributes from selected cells.

3.3 Changing Cell Sizes

The **Position** page in the **Cell Format** dialog lets you alter the size of the selected cell(s). Note that changing the height of a single cell will change the height for all cells in that row, similarly changing the width will affect the entire column.

You can also select the row(s) or column(s) to be changed then select **Resize Row...** or **Resize Column...** from the right mouse button pop up menu or from the **Format** → **Row** or **Format** → **Column** menu.

If you move the mouse cursor so that its tip is over the line between two of the row numbers at the left of Calligra Sheets's window the cursor will change to show two parallel lines each with a short arrow headed line coming from it. When the cursor is in this state you can hold the left mouse button down and drag the border between the two rows, changing the height of the upper row. A similar technique can be used to change the width of a column.

To set the row height or column width to the minimum needed to display the contents, select the whole row or column, and click with the right mouse button on the row or column label. In the menu which appears, select **Adjust Row** or **Adjust Column**. The row or column will resize to the minimum necessary. You can also select a single cell or range of cells, and click **Adjust Row & Column** from either the right mouse button popup menu or the **Format** menu.

You can make a number of adjacent rows or columns the same size by selecting them then choosing **Format** → **Row** → **Equalize Row** or **Format** → **Column** → **Equalize Column**.

3.4 Merging Cells

It is often convenient to have one cell that spreads across two or more columns or down more than one row. This can be done by merging two or more cells into one. Select the cells to be merged then choose **Format** → **Merge Cells**.

To reverse this process, select the merged cell then choose **Dissociate Cells** from the **Format** menu.

3.5 Hiding Rows and Columns

A finished spreadsheet can often be made to look more attractive by hiding the cells containing intermediate calculations so that only the important data input and result areas are shown.

In Calligra Sheets you can hide selected rows or columns by using the **Hide Rows** and **Hide Columns** options from the **Format** → **Row**, **Format** → **Column** or right mouse button menus. Hidden rows and columns are not displayed on the screen or included in a print out.

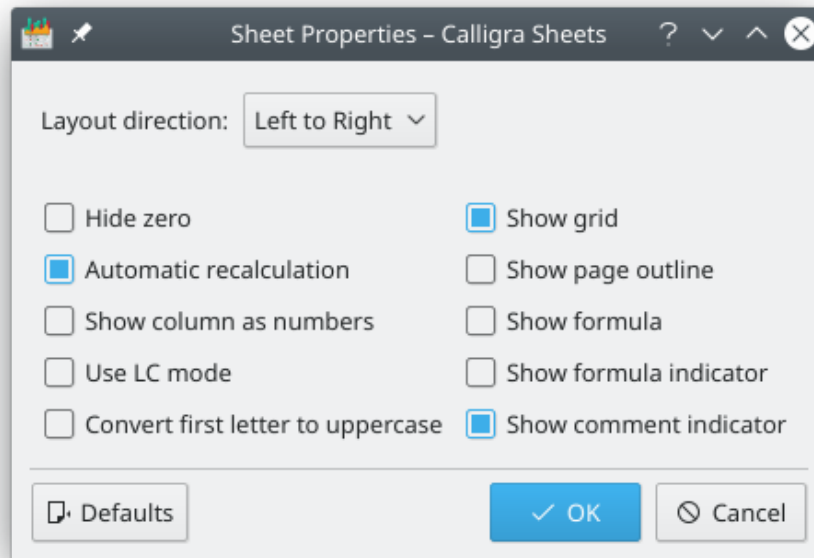
Hiding cells in this way also makes them slightly less prone to accidental change.

To un-hide a row or column select **Row** → **Show Rows...** or **Column** → **Show Columns...** from the **Format** menu. In the dialog box which appears, you can select any number of rows to show.

3.6 Sheet properties

You can access the current sheet properties either by right clicking on the sheet tab and choosing **Sheet Properties** or by using the **Format** → **Sheet** → **Sheet Properties** menu. Please note that you can only access the **Sheet Properties** when the document or the sheet is not protected.

You can set different properties that will be valid in the current sheet. Clicking on **OK** will validate your changes and **Defaults** will bring back the default settings.



Layout direction:

Let you choose the sheet orientation. Default is that the first column of the sheet is on the left. If you choose **Right to Left**, then the first column will be on the right and the others added from right to left.

Hide zero

If this box is checked any cell containing the value zero will appear blank.

Automatic recalculation

This setting controls whether formulae are recalculated automatically when the value of any cell they refer to changes.

Show column as numbers

If this box is checked the column headings will show as numbers rather than as letters. Letters are default.

Use LC mode

If this box is checked the cell reference shown at the left end of the Formula Bar will be displayed in LC mode (i.e. L2C3) rather than in its normal form B3. This does not seem to be of much use at the moment.

Convert first letter to uppercase

Check this box and the first letter of any text you type in will automatically be converted to uppercase.

Show grid

If checked the grid (the cell limits) will be shown. This is default. If you uncheck it, the grid will be hidden.

Show page borders

If you check this option, the page borders will be drawn on your current sheet. Per default the page borders are not displayed. It is useful to see the page borders if you want to print your sheet.

The Calligra Sheets Handbook

Show formula

If this box is checked Calligra Sheets will display the actual formulae in cells rather than the results.

Show formula indicator

If this box is checked Calligra Sheets will display a small blue triangle at the bottom left corner of cells containing formulae. This is useful if you want to protect cells with formulae.

Show comment indicator

If this box is checked cells containing comments will be marked by a small red triangle at the top right corner.

Chapter 4

Advanced Calligra Sheets

Pamela Robert
Anne-Marie Mahfouf

4.1 Series

When constructing a spreadsheet you often need to include a series of values, such as 10, 11, 12..., in a row or column. There are several ways you can do this in Calligra Sheets.

For a simple short series such as 5, 6, 7, 8... the 'Drag and Copy' method is the simplest. Enter the starting value into the starting cell and the next value of the series into an adjacent cell. Then select both cells and move the mouse pointer so that it is over the small square at the bottom right corner; the cursor will change to a diagonal double headed arrow. Then hold the left mouse button down while you drag the cells down or across as needed.

The step size is calculated as the difference between the two starting values that you have entered. For example if you enter **4** into cell A1 and **3.5** into A2 then select both cells and Drag and Copy them down, the step size will be the value in A2 minus the value in A1, -0.5 in this case so you will get the series 4, 3.5, 3, 2.5, 2...

The 'Drag and Copy' method will even cope with series where the step value is not a constant value but is itself a series. So that if you start with 1, 3, 4, 6 Drag and Copy will extend it to 1, 3, 4, 6, 7, 9, 10, 12..., the step value in this example being the series 2, 1, 2, 1...

Calligra Sheets also recognizes some special 'series' such as the days of the week. Try entering **Friday** into a cell (note the capitalization) then Drag and Copy it down. To see what special series are available, and perhaps create your own, select **Tools** → **Custom Lists...**

If you select a cell and choose **Series...** from the **Insert** menu you will see the **Series** dialog box. This is useful for creating series that are too long to be conveniently constructed using the Drag and Copy method, or for creating geometric series such as 1, 1.5, 2.25, 3.375... where the step value, 1.5 in this case, is used as a multiplier.

If the type of series that you want is too complicated for any of the previous methods, consider using a formula and Drag and Copying that. For example to create a series with the values 2, 4, 16, 256... enter **2** into A1, **=A1*A1** into A2, and Drag and Copy cell A2 down.

4.2 Formulae

4.2.1 Built in Functions

Calligra Sheets has a huge range of built in mathematical and other [functions](#) that can be used in a formula cell. They can be seen and accessed by selecting a cell then choosing **Function...** from

the **Insert** menu. This brings up the **Function** dialog box.

Select the function you want to use from the listbox at the left of the dialog box. The **Help** tab page will then display a description, the return type, Syntax, Parameters, and Examples for this function. In addition this page provides often links to Related Functions. Then press the button with the down arrow key symbol on it to paste it into the text edit box at the bottom of the dialog.

The **Parameters** tab page will then be displayed to let you enter the parameter(s) for the function you have just chosen. If you want to enter an actual value for a parameter, just type it into the appropriate text box in the **Parameters** page. To enter a cell reference rather than a value, left click on the appropriate text box in the **Parameters** page; then left click on the target cell in the spreadsheet.

Instead of using the **Parameters** page, cell references such as **B6** can be entered by typing them directly into the edit box at the bottom of the **Function** dialog. If a function has more than one parameter separate them with a semi-colon (;).

Pressing the **OK** button will insert the function into the current cell and close the **Function** dialog.

You can of course do without the **Function** dialog and simply type the complete expression into the main text entry box in the **Cell Editor** tool options. Function names are not case sensitive. Do not forget that all expressions must start with an = symbol.

4.2.2 Logical Comparisons

Logical functions such as IF(), AND(), OR() take parameters which have the logical (boolean) values True or False. This type of value can be produced by other logical functions such as ISEVEN() or by the comparison of values in spreadsheet cells using the comparison expressions given in the following table.

Expression	Description	Example
==	Is equal to	A2==B3 is True if the value in A2 is equal to the value in B3
!=	Is not equal to	A2!=B3 is True if the value in A2 is not equal to the value in B3
<>	Is not equal to	Same as A2!=B3
<	Is less than	A2<B3 is True if the value in A2 is less than the value in B3
<=	Is less than or equal to	A2<=B3 is True if the value in A2 is less than or equal to the value in B3
>	Is greater than	A2>B3 is True if the value in A2 is greater than the value in B3
>=	Is greater than or equal to	A2>=B3 is True if the value A2 is greater than or equal to the value in B3

Thus if you enter **=IF (B3>B1; "BIGGER"; "")** into a cell it will display BIGGER if the value in B3 is greater than that in B1, otherwise the cell will show nothing.

4.2.3 Absolute Cell References

If a formula contains a cell reference that reference will normally be changed when the cell is copied to another part of the worksheet. To prevent this behavior put a \$ symbol before the column letter, row number or both.

- If A1 contains the formula =D5 then on copying the cell to B2 it will become =E6 (the normal behavior).
- If A1 contains the formula =\$D5 then on copying the cell to B2 it will become =D6 (column letter not changed).
- If A1 contains the formula =D\$5 then on copying the cell to B2 it will become =E5 (row number not changed).
- If A1 contains the formula =\$D\$5 then on copying the cell to B2 it will remain as =D5 (neither the column letter nor the row number are changed).

When you are entering or editing a cell reference in a formula the shortcut key **F4** can be used to step through these four possibilities.

[Named cells](#) can be used in a similar way to include a unchanging cell reference in a formula.

4.3 Arithmetic using Special Paste

Sometimes you may want to add a single value to a number of cells, or subtract a value from them, or multiply or divide them all by a single value. The **Special Paste...** option lets you do this quickly and easily.

First, enter the modifier value into any spare cell on your spreadsheet and **Copy** it. Then select the area of cells you want to change, choose **Special Paste...** from the **Edit** or the context menu and select **Addition**, **Subtraction**, **Multiplication** or **Division** from the **Operation** section of the dialog box.

You can also apply different modifier values to different rows or columns of the target area by copying an area containing the wanted modifiers before selecting the target area and doing **Special Paste...** For example, if you enter **5** into cell A1, **10** into B1, select both cells and do a **Copy** then **Special Paste...** **Addition** into cells A10 to D15, 5 will be added to A10:A15 and C10:C15, and 10 to B10:B15 and D10:D15.

Note that a modifier value can be a formula as well as a simple numeric value. If it is a formula then Calligra Sheets will adjust the cell references as for a normal **Paste** operation.

4.4 Array Formulas

Calligra Sheets allows you to use formulas whose result is a matrix or a range of values. Normally, only the first value is displayed in a cell. If you would like to display the entire matrix, simply use **Ctrl-Alt-Enter** when editing a formula, and it will be converted into an array formula, occupying neighboring cells as needed.

Cells that are a part of an array formula are locked for editing.

4.5 Goal Seeking

Calligra Sheets can be used to solve algebraic expressions such as $x + x^2 = 4$ or *For what value of x does $x + x$ squared equal 4 ?*

For this example you could enter `=A2+A2*A2` into A1 then either try different values in A2 until the result in A1 is as close as you wish to 4 or, preferably, use Calligra Sheets's **Goal Seek...** feature which automatically adjusts the value in one cell to try to make the value in another cell as close as possible to a target value.

It is invoked by selecting **Goal Seek...** from the **Data** menu. This brings up a dialog box in which you should enter the reference of the target value cell (**A1** in this case) into the **Set cell:** box, the target value itself (**4**) into the **To value:** box and the reference of the cell that is to be changed (**A2**) into the **By changing cell:** box. Note that you need to have entered some initial value into the cell that is to be changed before starting **Goal Seek**.

Pressing the **OK** button will start the calculation. When it finishes and if it has found a solution press the **OK** button to accept the result or **Cancel** to keep the original value.

4.6 Pivot Tables

Calligra Sheets can be used to construct [pivot tables](#) using the data of the current table.

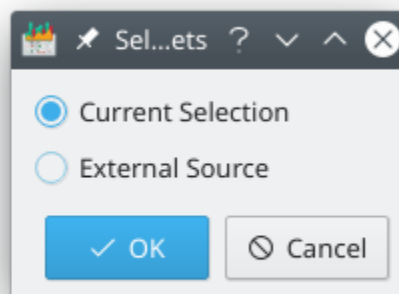
This feature can be invoked by selecting **Pivot...** from the **Data** menu. Below is an example of pivot table generation.

Supposing we have the following data.

	A	B	C
1	Name	Category	Score
2	Jigar	Science	90
3	Smith	Math	80
4	John	Science	95
5	Smith	Science	60
6	Jigar	Math	81
7	John	Math	90

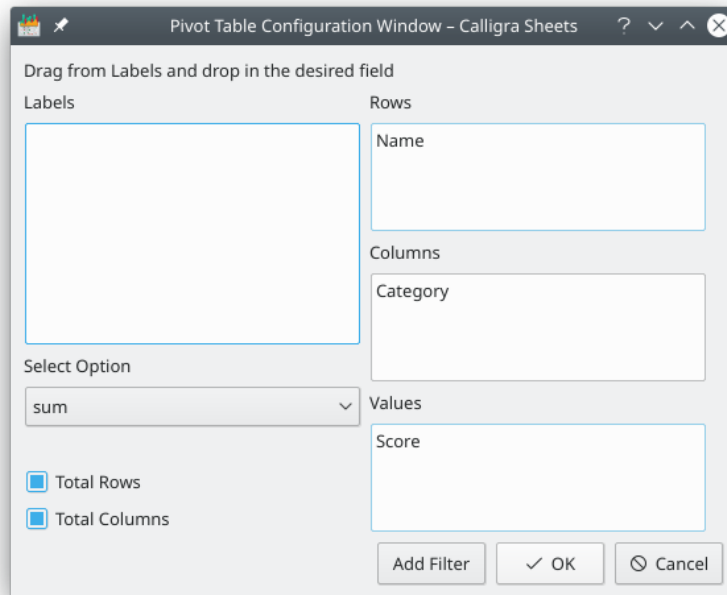
We want to create a pivot table of our choice and requirement. So we choose **Data** → **Pivot...**

The dialog box that will appear allows user to select the source of data. The data can be taken from the current worksheet or from an external source like a database or ODS file.



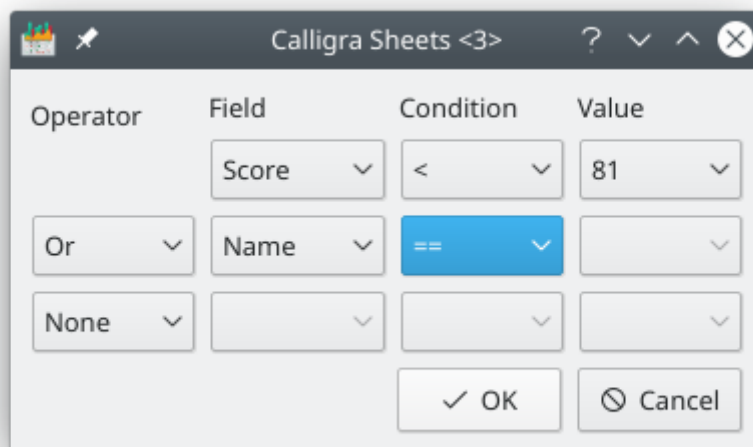
The Calligra Sheets Handbook

Here is the dialog box which allows the user to customize the pivot table. The column labels in the source data are converted to labels which serve as the working fields. The labels can be dragged and dropped into one of three areas (**Rows**, **Columns** or **Values**) to generate the pivot table. You can reset your choices using **Reset DnD** button.



In our example, *Name* is dragged to **Rows**, *Category* to **Columns**, *Score* to **Values**. User defined functions like sum, average, max, min, count, etc. can be selected from the **Select Option** list.

The **Add Filter** button can be used to open filter dialog box to filter the desired data. Using this box you can define multiple filters based on the column label and the relationship between them (**And** or **Or**). This would allow extreme freedom to customize the output.



Total Rows and **Total Columns**: checking these allow automatic totalling of corresponding rows and columns in the pivot table.

4.7 Using more than one Worksheet

When you start a new, empty, document with Calligra Sheets it will create a number of blank worksheets. The number of sheets it creates is determined by the selected template.

Insert → **Sheet** will add another sheet to the document.

You can also switch between worksheets by using the **Ctrl+PgDown** to move to the next sheet, **Ctrl+PgUp** to move to the previous one.

Worksheets are given the default names of *Sheet1*, *Sheet2*... You can give a sheet a different name by right clicking on the tab and selecting **Rename Sheet**...

To remove a sheet from the document use the **Remove Sheet** option in the context menu that pops up when you right click on the tab for the sheet you want to remove.

Other entries in the **Format** → **Sheet** submenu allow you to show or hide a sheet in much the same way as rows and columns can be hidden.

If you want a formula in one sheet to refer to a cell in another sheet, the cell reference must start with the sheet name followed by an exclamation mark (!). For example if you enter **=Sheet2!A2** into a cell in Sheet 1, that cell will take the value from A2 of Sheet2. Note that sheet names are case sensitive.

4.7.1 Consolidating Data

You may have constructed a document containing several worksheets containing similar data but for, say, different months of the year, and wish to have summary sheet containing the consolidated (e.g., sum or average) values of the corresponding data items in the other sheets.

This task can be made slightly easier by using the **Consolidate...** item from the **Data** menu.

Selecting this option brings up the **Consolidate** dialog box.

For each of the source sheets, enter a reference to the desired data area in the **Reference:** box. Press **Add** to transfer it to the **Entered references:** box. The reference should include the name of the sheet containing the source data, such as **January!A1:A10**, and can be entered automatically by selecting the area in the appropriate sheet.

After entering the references for all of the source data sheets select the cell in the target sheet where you want the top left corner of the consolidated results to appear. Then choose the appropriate function from the **Function:** combo box and press the **OK** button.

If you click the **Details** » in the dialog and check the **Copy data** box the values resulting from the consolidation will be placed into the target cells rather than the formulae to calculate them.

4.8 Inserting a Chart

You can insert a chart into a sheet to give a graphical view of your data.

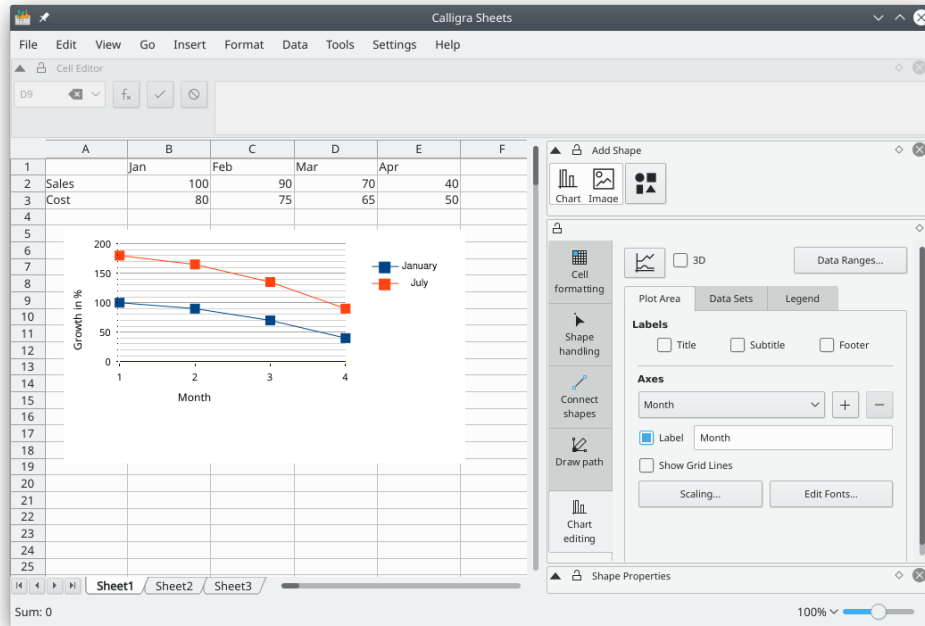
First enable **Add Shape** docker using the **Settings** → **Dockers** menu item.

Then select the area of cells containing the data and choose **Chart** in the **Add Shape**. Drag the cursor across the sheet while holding the left mouse button down to define the area where you want the chart to appear, there is no need to be too accurate at this stage as the chart size can easily be changed at any time. When you release the mouse button a **Chart Options** dialog box will appear.

The data area is already prefilled with the selected cell range. Select the first row and column as labels, check **Data set in rows** and click the **OK** button. The Dialog will vanish and you will see the chart embedded into the worksheet.

The Calligra Sheets Handbook

Now select **Chart Editing Tool** from the **Tools** docker and edit the chart properties like chart type, labels and axis in the **Chart editing**.



To move, resize or even delete the embedded chart switch to the **Basic shape manipulation** tool and click anywhere within the chart area. It should now appear with a green border and with a small yellow square at each corner and in the middle of each edge.

If you move the cursor over any of the squares it should change to a double headed arrow. You can resize the chart by dragging one of these squares with the left mouse button pressed. To delete the chart right click on one of the squares and select **Delete**.

To move the chart move the cursor into the chart. The cursor should then change to a cross, press the left mouse button and you will be able to drag the chart to where you want it to be.

To restore the chart to its normal appearance simply click anywhere outside of the chart area.

To change the format of the chart itself left click twice within the chart area. The chart **Chart editing** should appear in the docker. You can then use these tools to change the chart.

4.9 Inserting External Data

You can insert data from a text file or from the clipboard into a worksheet by first selecting the cell where you want the top left item of the inserted data to appear, then choosing **From Text File...** or **From Clipboard...** from the **Insert** → **External Data** sub menu.

In both cases Calligra Sheets will assume that the data is in CSV form and will open a dialog box allowing you to control how the data is extracted from the file or clipboard and placed into the worksheet cells.

If support for it has been included in your system, Calligra Sheets can also insert data from a SQL database into a worksheet. This is done by using the **Insert** → **External Data** → **From Database...** option.

4.10 Link Cells

A spreadsheet cell can be linked to an action so that left clicking on the cell will, for example, open your browser. To make a cell act in this way select it and choose **Insert** → **Link...** This will bring up the **Insert Link** dialog box, which lets you choose between four types of link:

- An **Internet** link cell will try to open your default browser at the URL entered in the **Internet address:** text box of the **Insert Link** dialog when it is clicked. This could be, for example, **http://www.calligra.org**.
- Clicking on a cell containing a **Mail** link will open your email composer using the address entered in the **Email:** text box as the To: address. For example **anon@example.com**.
- A **File** link cell holds the path to a file or folder, as entered into the **File location:** text box, and will try to open that file or folder with a suitable application when clicked on.
- The **Cell** type of link cell holds a Calligra Sheets cell reference, entered in the **Cell or Named Area** text box. Left clicking on this type of link cell causes Calligra Sheets's focus to move to the target cell.

All four types of link cell need some suitable text to be entered into the **Text to display** field of the **Insert Link** dialog. This is the text that appears in the cell.

4.11 Validity Checking

Calligra Sheets can automatically check the validity of entered data against a number of criteria, and pop up a message box if the data is invalid.

To enable this feature, select the cell(s) to be monitored and choose **Data** → **Validity...** This will bring up Calligra Sheets's **Validity** dialog box which has three tabbed pages.

In the **Criteria** page select what type of data is to be considered valid from the **Allow:** combo box list then define the valid range of values by choosing one of the options in the **Data:** combo box and entering suitable value(s) into one or both of the edit box(es).

When you have done this change to the **Error Alert** tab. Here you can choose the type of message box (**Stop**, **Warning** or **Information**) that will appear when an invalid value is entered, and define the message box title and message text.

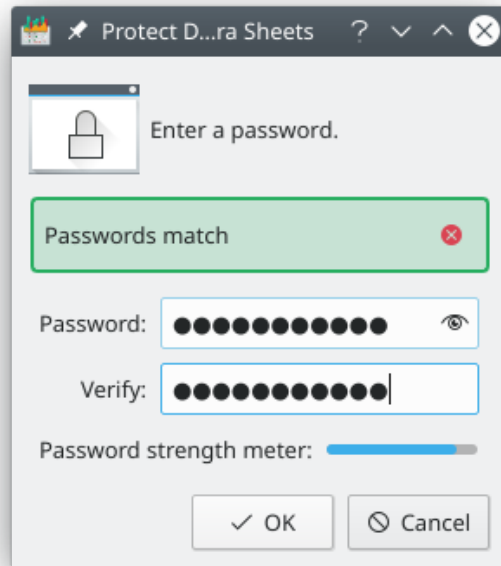
Note that this feature only checks data that you enter into the cell, for a way of checking the results from formulae cells see the [Conditional Cell Attributes](#) section of this Handbook.

4.12 Protection

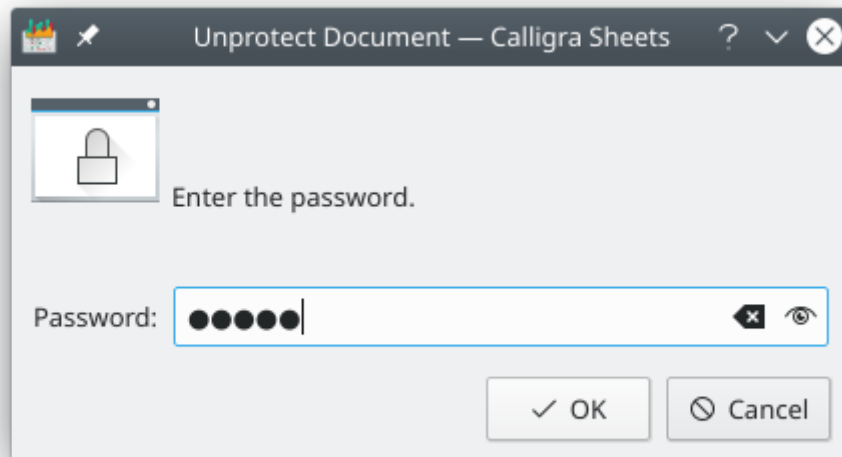
4.12.1 Document Protection

Protecting the document means that without the password a user cannot add or delete sheets. Document protection does not protect cells.

Select **Tools** → **Protect Document...** A dialog appears asking you for a password. The **Password:** strength meter indicates if your password is secure enough. The longer the indicator is, the more secure your password.



That password will then be required to unprotect the document.



When a document is protected, you may not:

- Rename a sheet
- Insert a sheet
- Remove a sheet
- Hide a sheet
- Show a sheet
- See the sheet properties
- Merge or dissociate cells

4.12.2 Sheet protection

Protecting a sheet means protecting the contents of all protected cells and objects on a sheet. Individual cells or a selection of cells can be unprotected within a protected sheet, see [next section](#).

To protect a sheet, select **Tools** → **Protect Sheet...** A dialog appears asking you for a password. The **Password** strength meter indicates if your password is secure enough. The longer the indicator is, the more secure will be your password.

That password will then be required to unprotect the sheet.

When a sheet is protected, you may not:

- Insert any object or chart
- Format any cell
- Insert a row or a column
- Edit and change cell content
- Change any content in the sheet

NOTE

Protecting a sheet is especially useful for preventing accidental erasure of formulae.

4.12.3 Cell or selected cells protection

WARNING

Cell protection is active for all cells by default and is effective when you enable sheet protection. So if you keep the default and if you protect the sheet, all cells will be protected.

If you want only certain cells to be protected, this default protection must be turned off for all other cells. For example you might want most cells to accept user input so you will uncheck **Protected** for those and choose to keep protected cells that should stay unchanged (such as titles). So you need 3 steps in order to protect only some cells: unprotect all the cells, select the cells to protect and protect them and then protect the whole sheet.

To unprotect all the cells:

- Select the entire spreadsheet with the mouse.
- In the menubar, select **Format** → **Cell Format...**
- In the dialog that appears, go to the **Cell Protection** tab.
- Check **Hide all** and uncheck **Protected** to remove the protection on all cells. The cells are now all unprotected.

To protect a range of selected cells or a selection of non-contiguous cells:

- Highlight the range of cells that are to be protected or use the **Ctrl** key to select non-contiguous cells.
- When all of the desired cells are selected, go to the **Format** → **Cell Format...** menu.

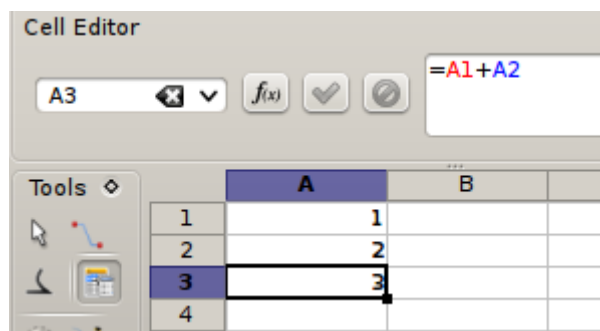
- In the dialog that appears, go to the **Cell Protection** tab.
- Click on the box next to **Protected** then click on **OK**.

Once the cells are marked for protection, the protection option must be enabled at the sheet level, that means you must protect the entire sheet for the cell to be effectively protected:

- Select **Tools** → **Protect Sheet...**
- In the dialog that appears, provide a safe password, then confirm it by typing it again. Click on **OK**.
- Protected cells in a protected sheet cannot be edited without unprotecting the whole sheet, and any sheet changes are disabled. For example, no one can insert rows or columns, change column width, or create embedded charts.

4.12.4 Hide cell formula

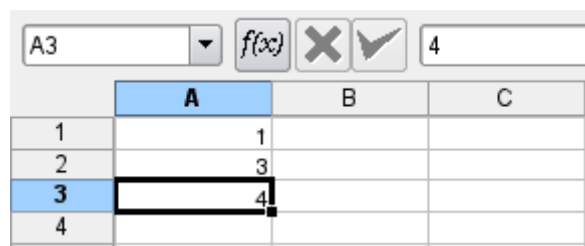
You might want to hide your formulae so other people cannot see them. By default, every cell is protected and not hidden. But it is important to remember that these attributes have no effect unless the sheet itself is protected.



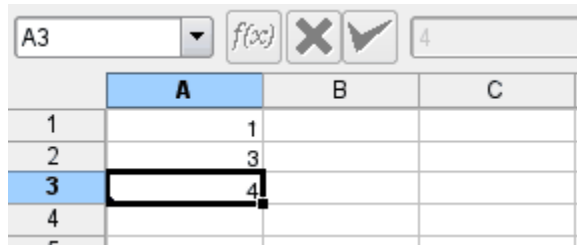
To hide cell formulae, select the appropriate cell or range of cells or non-contiguous cells with **Ctrl** and then choose the **Format** → **Cell Format...** menu. In the Cell format dialog, click the **Cell Protection** tab and select **Hide formula**. After you protect the sheet, the results of the formulae will be visible, but the formulae will not.

You have now to protect the sheet: choose **Tools** → **Protect Sheet...** to display the **Protect Sheet** dialog box. Enter a safe password twice to prevent others from unprotecting the sheet.

When **Hide formula** is enabled and **Protected** is disabled, the formula is hidden after protecting the sheet but the cell content can be changed.



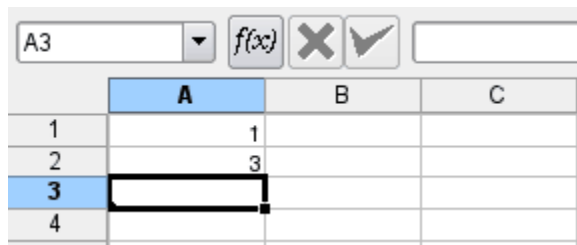
When **Hide formula** and **Protected** are enabled, the formula is hidden after protecting the sheet and the cell content cannot be changed.



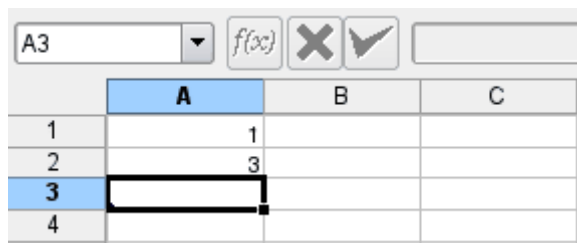
Keep in mind that it is very easy to break the password for a protected sheet so if you are looking for real security, this is not the best solution.

4.12.5 Hide all in the cell

You can hide both the formula and the content of the cell by choosing **Hide all** in the Cell Protection tab in the **Format** → **Cell Format...** menu. In the screenshot below, the cell itself is not protected (**Protected** is unchecked) thus the cell content can be changed.



Here the cell itself is protected so it cannot be overwritten.



4.13 Other Features

4.13.1 Named Cells and Areas

You can give a name such as **foo** to a cell or to any area of a sheet by selecting the cell or area then selecting **Area Name...** from the right mouse button menu. This will bring up the **Area Name** dialog box where you can enter any name you wish.

You can also name a cell or area by selecting it then typing the name into the small text box at the left end of the Formula toolbar, overwriting the cell reference that normally appears here.

If you enter a name that has already been used into this text box Calligra Sheets's selection will change to show the named cell(s).

The **Data** → **Named Areas...** option will give you a list of existing names and let you change Calligra Sheets's focus to any of them or let you remove a name.

Named cells are particularly useful in formulae as an alternative to **absolute cell references** as the names can be used in place of normal cell references and do not change when the cell containing the formula is copied. When a name is used in this way it should be enclosed in single quotation marks.

For example, if cell A1 has been given the name **fred** then you can enter a formula such as **= 'fred' + 2** into another cell which would always give the result of adding 2 to the value in A1 no matter where the formula cell was copied to.

Note that cell and area names are treated as being in lowercase.

4.13.2 Cell Comments

A cell can contain a text comment that can be viewed when working on the spreadsheet but which is not printed and not normally seen.

To add a comment select the cell and choose **Comment...** from the right mouse button menu or from the **Insert** menu and type your comment into the resulting **Cell Comment** dialog box.

To see the comment hover the mouse pointer over the cell. The comment will appear as if it were a Tooltip.

If you check the **Show comment indicator** box of the **Sheet Properties** dialog, those cells containing comments will be highlighted by a small red triangle in the top right corner.

To open this dialog, click with the right mouse button onto the sheet tab at the bottom of the main window and select **Sheet Properties** from the popup menu. Or select it from the **Format** → **Sheet** menu.

To remove a comment from a cell, select **Remove Comment** from the right mouse button menu or choose **Edit** → **Clear** → **Comment**.

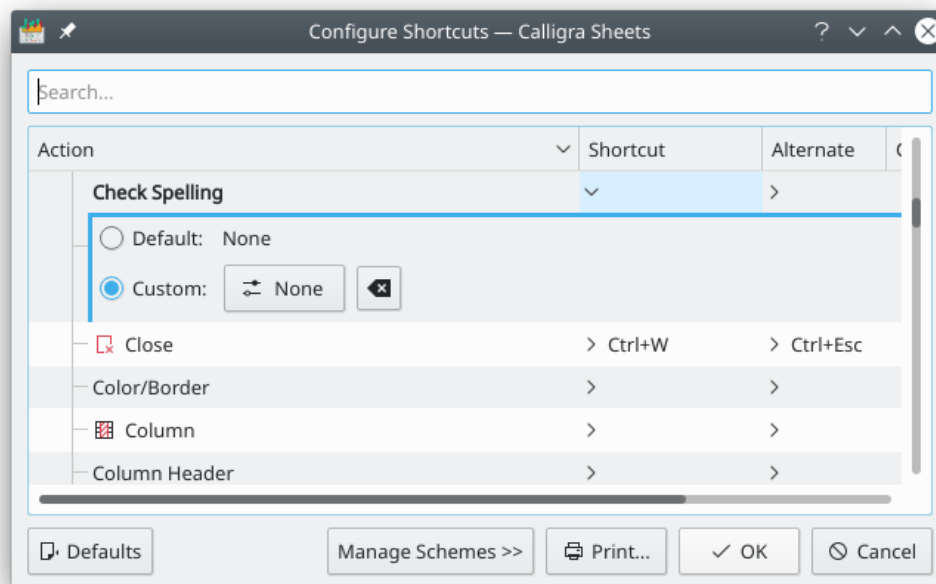
Chapter 5

Configuring Calligra Sheets Shortcuts and Toolbars

Pamela Robert

5.1 Shortcuts

To change the shortcut key arrangements used by Calligra Sheets select **Settings** → **Configure Shortcuts...** . This will launch a dialog box as shown below.



Search through the list box to find the action you want to add or change the shortcut keys for and select it by left clicking on the name. By entering the name of the action in the search bar at the top you can quickly find the desired action. You will then be able to change the shortcut by selecting the **None**, **Default** or **Custom** radio button.

You can now simply press the key combination you want to act as the shortcut, for example **Ctrl+Shift+S**.

5.2 Toolbars

Calligra Sheets has six toolbars: **File**, **Edit**, **Navigation**, **Format**, **Font** and **Color/Border**. each of which may or may not be shown depending on the choices made in the **Settings** menu.

If the toolbars are unlocked, you can choose whether a toolbar appears at the **Top**, **Left**, **Right** or **Bottom** of Calligra Sheets's window by right clicking on the toolbar, which brings up the **Toolbar Menu**, and making a selection from the **Orientation** sub menu. This **Toolbar Menu** also has sub menus for choosing whether the toolbar displays icons, text or both, and the size of the icons.

Another way of moving a toolbar is by positioning the mouse pointer over the two vertical bars at the left end of each toolbar and holding the left mouse button down while you drag the toolbar to the wanted position. When you drag the toolbar in this way you can release the mouse button when it is some distance from any of Calligra Sheets's window sides, and then you will get a floating toolbar, which is not locked to any particular part of Calligra Sheets's window and can in fact be moved outside of the window. To put a floating toolbar back into one of the traditional positions right click on its titlebar to bring up the **Toolbar Menu** then choose one of the options in the **Orientation** sub menu.

Selecting **Configure Toolbars...** from the **Settings** menu will bring up a dialog box which lets you add buttons to or remove them from Calligra Sheets's toolbars.

To use this **Configure Toolbars** dialog box first select a toolbar from the **Toolbar:** combo box. The right hand **Current actions:** window will then show the buttons currently present on the toolbar. You can remove a button by selecting it in this window then pressing the left arrow button, or move it around by pressing the up and down arrow buttons. To add a new button to the toolbar select it in the **Available actions:** list then press the right arrow button.

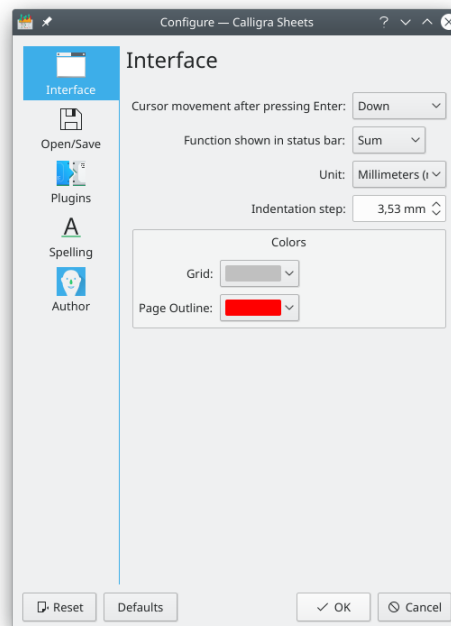
Chapter 6

The Calligra Sheets Configuration Dialog Box

Pamela Robert

Selecting **Settings** → **Configure Calligra Sheets...** opens a dialog box with several pages, selected with the icons at the left of the dialog box, which allow you to change many aspects of Calligra Sheets's operation.

6.1 Interface



Cursor movement after pressing Enter:

Select whether pressing the **Enter** key will move the cursor **Down, Up, Right, Left, Down, First Column** or **None** as determined by the setting in this drop down selection box.

Function shown in the status bar:

This drop down selection box can be used to choose the calculation performed by the [Statusbar Summary](#) function.

Unit:

Choose the default unit that will be used in your sheets.

Indentation step:

Set the amount of indent that will be used in the cell when you choose the **Increase Indent/Decrease Indent** actions from the toolbar. These actions are not enabled by default in the toolbar.

Capture all navigation keys while editing

Captures all navigation keys, i.e. the arrow keys, page up/down, tabulator and backward tabulator key, while editing a cell with the embedded editor. The embedded editor is the one appearing directly in the cell. If captured, these keys are used for navigating in the editor. Otherwise, they are used for cell navigation.

The **Colors** section lets you choose the color of the sheet grid. If you do not want the grid to appear at all, uncheck the **Show grid** box in the **Format** → **Sheet** → **Sheet Properties** dialog.

This section also lets you select the color of the lines used to indicate the printed page borders when the **Page Borders** box in the **View** menu is checked.

Select **Custom** from the current color to display the standard KDE **Select Color** dialog.

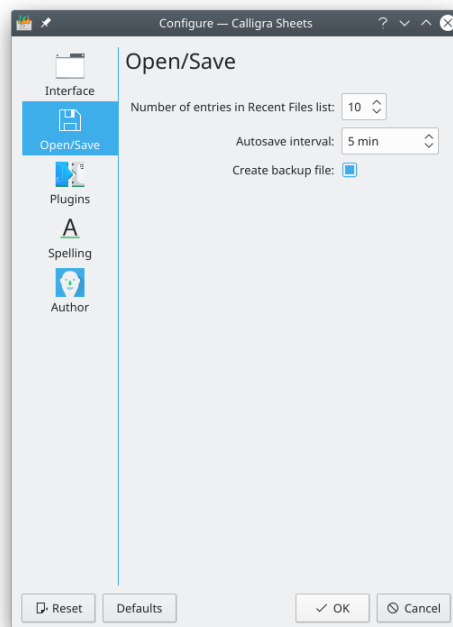
Grid

Click here to change the grid color i.e. the color of the borders of each cell.

Page border

When the **View** → **Show Page Borders** menu item is checked, the page borders are displayed. Click here to choose another color for the borders than the default red.

6.2 Open/Save



Number of entries in Recent Files list:

Controls the maximum number of filenames that are shown when you select **File** → **Open Recent**.

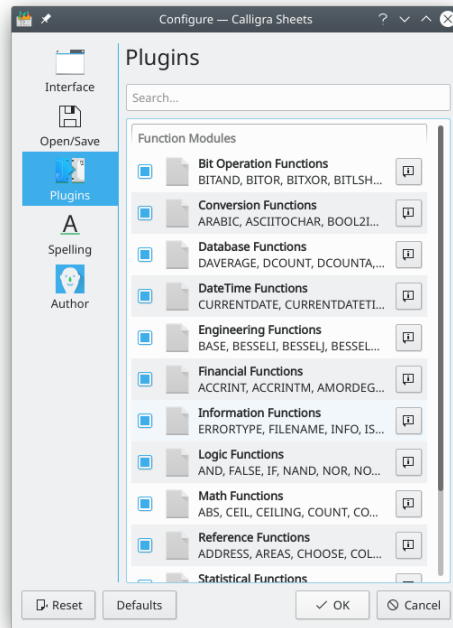
Autosave interval:

Here you can select the time between autosaves, or disable this feature altogether by choosing **No autosave** (by selecting the minimum value).

Create backup file:

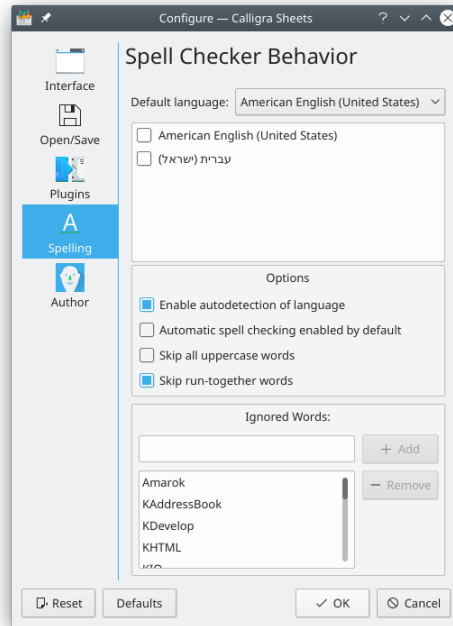
Check this box if you want some backup files created. This is checked per default.

6.3 Plugins



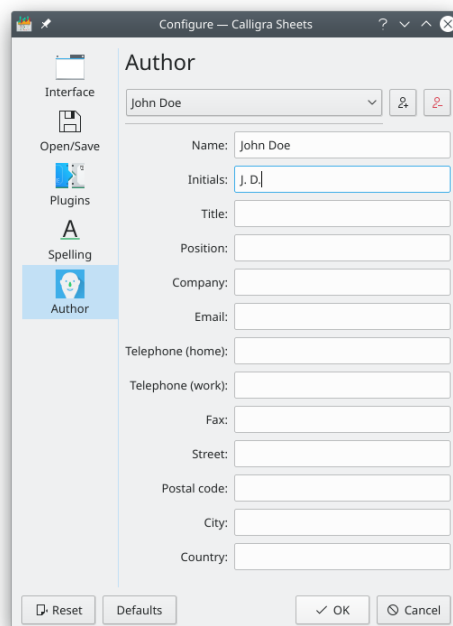
Check a plugin in the list to enable it. Display the **About** dialog by clicking the info button at the right side of the list item.

6.4 Spelling



This page lets you configure the behavior of Calligra Sheets’s spell checker. For more information please refer to the [Spell Checker](#) documentation.

6.5 Author



Choose an author profile for your document.

The Calligra Sheets Handbook

You can add a new profile or delete the current profile using the buttons to the right of the profiles drop-down list.

Chapter 7

Command Reference

Pamela Robert

7.1 The File Menu

File → **New (Ctrl+N)**

Create a new document.

File → **Open... (Ctrl+O)**

Open an existing document.

File → **Open Recent**

Open an existing document by selecting it from a combo box of recently used files.

File → **Save (Ctrl+S)**

Save the document.

File → **Save As...**

Save and change the current document to a new name or format. If you want to keep the name and format of the document use **Export...**

File → **Reload**

Reloads the document.

File → **Import...**

Import other documents.

Tables of data are often held in text files with the values in a line being separated by a comma, space, tab or other character, for example *123, 456, 789, abcd, efgh*. Such files are commonly called 'CSV' (Comma Separated Values) files, even though the separating character may not be a comma.

If you ask Calligra Sheets to open a text file it assumes that the file is in CSV format and launches a dialog box that allows you to specify the delimiter (separating character) used by the file, and shows how the data items will be placed into different spreadsheet cells.

Other options in this dialog box let you define the **Format** of the spreadsheet cells, whether text quote characters should be removed, and whether the first line(s) of the file should be ignored.

File → **Export...**

Save a document to any supported format. The document does not become the exported file.

File → **Mail...**

Send the file as an email attachment.

File → **Create Template From Document...**

Create a Calligra Sheets [template](#) based on this document.

File → **Print... (Ctrl+P)**

Print the document.

File → **Print Preview...**

View the document as it will be printed.

File → **Document Information**

View or enter information about the document and author.

File → **Close (Ctrl+W)**

Close the current document but leave Calligra Sheets running.

File → **Quit (Ctrl+Q)**

Quit Calligra Sheets.

7.2 The Edit Menu

Edit → **Undo (Ctrl+Z)**

Undo the last action.

Edit → **Redo (Ctrl+Shift+Z)**

Redo the last undone action.

Edit → **Cut (Ctrl+X)**

Put selected item(s) into the clipboard and remove them from the original location. If you then do a **Paste** the item(s) will be inserted at the new location.

Edit → **Copy (Ctrl+C)**

Copy selected item(s) to the clipboard.

Edit → **Paste (Ctrl+V)**

Paste item(s) from the clipboard to the selected cell(s).

Edit → **Special Paste...**

Special forms of Paste. See the sections [Other Paste Modes](#) and [Arithmetic using Special Paste](#) for more details.

Edit → **Paste with Insertion**

Move content of the paste area either to the right or down and paste item(s) from the clipboard to the selected cell(s).

Edit → **Fill**

Fills up the selected area with the values from the first item-set. All four directions are supported. Note that the term "item-set" describes the first set of values seen in the fill direction. If the fill direction is Left then the first item-set is the last column of the selection.

Edit → Find... (Ctrl+F)

Find cell containing given text.

Edit → Find Next (F3)

Find the next cell containing given text.

Edit → Find Previous (Shift+F3)

Find the previous cell containing given text.

Edit → Replace... (Ctrl+R)

Find and replace given text in cell(s).

Edit → Clear

Clear **All** or **Contents**, **Comment**, **Conditional Styles**, **Link** or **Validity** from selected cell(s).

Edit → Delete

Delete **Cells**, **Columns**, **Rows** or **Sheet**.

Edit → Modify Cell (F2)

To modify selected cell in-situ.

7.3 The View Menu

View → New View

Open a new instance of Calligra Sheets with the same document.

View → Page Borders

Toggle marking of printed page borders in the sheet with red lines.

View → Zoom

Increase or decrease the magnification used to display the spreadsheet. Range from 33% to 500%.

7.4 The Go Menu

In this menu you find actions to navigate between the sheets of the currently opened spreadsheet and the **Goto Cell** action to jump to a single cell or to select a range of cells.

7.5 The Insert Menu

Insert → Comment

Add or modify a comment.

Insert → Function...

Insert a mathematical function. See the section [Formulae](#) for more details.

Insert → Series...

Insert a series. See the section [Series](#) for more details.

Insert → Link...

Insert a link into the selected cell. See the section [Link Cells](#) for more details.

Insert → Special Character...

Insert a special character into the selected cell.

Insert → External Data

Insert data **From Database...**, **From Text File...** or **From Clipboard...** See the section [Inserting External Data](#) for more details.

7.6 The Format Menu

Format → Cell Format... (Alt+Ctrl+F)

Format selected cell(s). See the [Spreadsheet Formatting](#) section for more details.

Format → Style Manager

Create, Modify or delete cell format styles.

Format → Style

Apply a style to selected cell(s). To manage styles use **Format → Style Manager...**

Format → Create Style From Cell...

Create a new style from the format of the selected cell. To manage styles use **Format → Style Manager...**

Format → AutoFormat...

Autoformat the selected cells: a dialog let you choose between two proposed formats.

Format → Merge Cells

Merge selected cells.

Format → Dissociate Cells

Dissociate (split apart) previously merged cells.

Format → Adjust Row & Column

Set row and column sizes to show selected cell(s) properly.

Format → Row

Resize, equalize, hide or show row(s).

Format → Column

Resize, equalize, hide or show column(s).

Format → Sheet

Hide, show worksheet or configure advanced sheet properties.

Format → Page Layout...

Format printed page layout.

Format → Print Range

Define or reset the print range.

7.7 The Data Menu

Data → Sort...

Sort data in selected cells. See the section [Sorting Data](#) for more details.

Data → Text to Columns...

This option attempts to interpret text in the selected cell(s) as CSV data, placing each item into a different cell in the row.

Data → Named Areas... (Ctrl+Shift+G)

Open the **Named Areas** dialog to select, add, edit and remove named areas. See the section [Named Cells and Areas](#) for further details.

Data → Consolidate...

Consolidate data. See the section [Consolidating Data](#) for more details.

Data → Subtotals...

Create different kinds of subtotals to a database.

Data → Validity...

Set or modify the error checking criteria and error alert message for selected cell(s). See [Validity Checking](#) for more details.

Data → Goal Seek...

Open the Goal Seek dialog box. See [Goal Seeking](#) for details.

Data → Pivot...

Open the Pivot Table configuration dialog. See [Pivot Table](#) for details.

7.8 The Tools Menu

Tools → Spelling...

Check spelling of words in the worksheet.

Tools → Custom Lists...

View or amend the special series of words recognized by Calligra Sheets. These list can be used to insert special [series](#) into the worksheet.

Tools → Protect Sheet...

Protect the sheet with a password. A dialog pops up prompting you for a password. Unchecking this option will prompt you for the password in order to unprotect the sheet. Protecting a sheet means protecting all cells in the sheet. In a protected sheet, the cells cannot be reformatted or overwritten.

Tools → Protect Document...

Protect the whole document with a password. A dialog pops up prompting you for a password. Unchecking this option will prompt you for the password in order to unprotect the document. In a protected document you cannot rename or remove a sheet. Document protection does not mean that each individual sheet is protected.

Tools → Recalculate Sheet (Shift+F9)

Recalculate formulae in the current sheet.

Tools → Recalculate Document (F9)

Recalculate all sheets.

Tools → Execute Script File...

Execute the chosen external script in Calligra Sheets. Calligra Sheets supports scripting in JavaScript, Python and Ruby. Default examples of the scripts can be found in the **Tools** → **Scripts** submenu.

Tools → Scripts

Here you can execute the script to export or import data in various formats, save your Calligra Sheets log into file, use [Orca speech](#) for accessibility, debug Python and Ruby scripts or add functions to display stock or weather condition values. You can even use some [R functions](#) ([RPy module](#) should be installed).

Tools → Script Manager...

Opens the **Scripts Manager** dialog to execute, load, unload, install, uninstall and get more scripts.

Tools → Function Optimizer...

Opens the **Function Optimizer** dialog where you can choose an objective function cell, optimization target (**Maximize**, **Minimize** or enter the **Value**) and a set of decision parameter cells.

7.9 The Settings Menu

Settings → Toolbars Shown

Show or hide the toolbars: **File**, **Edit**, **Navigation**, **Font**, **Format** and **Color/Border**.

Settings → Status Bar

Show or hide the Status Bar. The Status Bar shows additional information for selected items and instant calculations of the selected cells.

Settings → Tab Bar

Show or hide the Tab Bar. All Sheets of the current Document can be accessed through the Tab Bar.

Settings → Configure Shortcuts...

Configure the keyboard shortcuts used by Calligra Sheets. See the section on [configuring shortcuts](#) for more details.

Settings → Configure Toolbars...

Configure the toolbars. The section on [configuring toolbars](#) has more information.

Settings → Themes

Choose the color theme for Calligra Sheets window. You can choose one of the predefined color schemes or select **Configuration...** to open [System Settings color selection module](#).

Settings → Active Author Profile

Configure the author profile for the current document. You can choose one of the profiles defined using [Calligra Sheets configuration window](#), **Default Author Profile** as defined by System Settings data or blank **Anonymous** profile which can ensure your privacy.

Settings → Configure Notifications...

Configure Calligra Sheets notification system. There are no actions that you can be notified in the current version of Calligra Sheets.

Settings → Configure Calligra Sheets...

General Calligra Sheets configuration. See the section on [Calligra Sheets configuration](#) for more details.

7.10 The Help Menu

Help → **Calligra Sheets Handbook (F1)**

Invokes the KDE Help system starting at the Calligra Sheets help pages. (this document).

Help → **What's This? (Shift+F1)**

Changes the mouse cursor to a combination arrow and question mark. Clicking on items within Calligra Sheets will open a help window (if one exists for the particular item) explaining the item's function.

Help → **Report Bug...**

Opens the Bug report dialog where you can report a bug or request a 'wishlist' feature.

Help → **Switch Application Language...**

Opens a dialog where you can edit the **Primary language** and **Fallback language** for this application.

Help → **About Calligra Sheets**

This will display version and author information.

Help → **About KDE**

This displays the KDE version and other basic information.

7.11 The Right Mouse Button Menu

This section describes the items in the pop up menu obtained by right clicking on a selected cell or cells, row(s) or column(s).

Cell Format... (Ctrl+Alt+F)

Format selected cell(s). See the [Spreadsheet Formatting](#) section for more details.

Cut (Ctrl+X)

Put selected item(s) into the clipboard. If you then do a **Paste** the item(s) will be moved from the original location to the new one.

Copy (Ctrl+C)

Copy selected item(s) into the clipboard.

Paste (Ctrl+V)

Paste item(s) from the clipboard to the selected cells.

Special Paste...

Special forms of Paste. See the sections [Other Paste Modes](#) and [Arithmetic using Special Paste](#) for more details.

Paste with Insertion

Paste from the clipboard to the selected cell(s), moving the previous cell(s) to make room.

All

Delete contents of selected cell(s).

Adjust Row & Column

Change size of row and column to display selected cell(s) completely.

The Calligra Sheets Handbook

Default

Set default formats for selected cell(s).

Area Name...

Name selected area. See the section [Named Areas](#) for more details.

Resize Row...

Change height of selected row.

Adjust Row

Change height of selected row to display cell(s) completely.

Resize Column...

Change width of selected column.

Adjust Column

Change width of selected column to display cell(s) completely.

Insert Cells...

Insert new cell(s) at selected location, moving existing cell(s) to make room.

Delete Cells...

Remove selected cell(s), moving other cell(s) to occupy the space left by the removed cell(s).

Insert Rows

Insert new row(s) above selected row(s).

Delete Rows

Delete selected row(s).

Hide Rows

Hides selected row(s).

Show Rows

Shows selected row(s). In order to show hidden rows you need to select a range of rows that includes the hidden rows.

Insert Columns

Insert new column(s) at left of selected column(s).

Delete Columns

Delete selected column(s).

Hide Columns

Hides selected column(s).

Show Columns

Shows selected column(s). In order to show hidden columns you need to select a range of columns that includes the hidden columns.

Comment...

Add or modify a comment to the selected cell.

Selection List...

Lets you select and paste text from any cell of the current selection of cells into the selected cell.

7.12 Other Shortcuts

This section describes those Calligra Sheets shortcut keys used for operations that do not appear in any of the menus.

Ctrl+H

Toggle the display of the dockers.

Ctrl+Arrow keys

If the selected cell is occupied then move the cell cursor to the start or end of the occupied block in the current row or column. If the selected cell is not occupied then move the cell cursor to the start or end of the block of unoccupied cells in the current row or column.

Ctrl+Shift+Arrow keys

If the selected cell is occupied then select all occupied cells to the start or end of that block of occupied cells in the current row or column. If the selected cell is not occupied then select all unoccupied cells to the start or end of that block of unoccupied cells in the current row or column.

Page Down

Move the cell cursor 10 cells down.

Page Up

Move the cell cursor 10 cells up.

Ctrl+Page Down

Move to the next sheet.

Ctrl+Page Up

Move to the previous sheet.

F4

Change cell reference between normal and [absolute reference](#) types.

Chapter 8

Functions

Calligra Sheets has a huge range of built in mathematical and other functions that can be used in a formula cell.

8.1 Supported Functions

This chapter holds a brief overview of all supported functions in the following groups:

- Bit Operations
- Conversion
- Database
- Date & Time
- Engineering
- Financial
- Information
- Logical
- Lookup & Reference
- Math
- Statistical
- Text
- Trigonometric

8.1.1 Bit Operations

8.1.1.1 BITAND

The BITAND() function performs a bit-wise AND operation for the two integer parameters.

Return type: Whole number (like 1, 132, 2344)

Syntax

BITAND(value; value)

Parameters

Comment: First number, *Type:* Whole number (like 1, 132, 2344)

Comment: Second number, *Type:* Whole number (like 1, 132, 2344)

Examples

BITAND(12;10) returns 8 (because decimal 12 is binary 1100, and decimal 10 is binary 1010; and 1100 "anded" with 1010 is 1000, which is integer 8).

Related Functions

[BITOR](#)
[BITXOR](#)

8.1.1.2 BITLSHIFT

The BITLSHIFT() function performs a bit-wise left shift operation of the first parameter. The number of bits to shift by is specified by the second parameter. Note that a negative number of bits to left shift by becomes a right shift.

Return type: Whole number (like 1, 132, 2344)

Syntax

BITLSHIFT(value; shift size)

Parameters

Comment: First number, *Type:* Whole number (like 1, 132, 2344)

Comment: Amount to left shift by, *Type:* Whole number (like 1, 132, 2344)

Related Functions

[BITLSHIFT](#)

8.1.1.3 BITOR

The BITOR() function performs a bit-wise OR operation for the two integer parameters.

Return type: Whole number (like 1, 132, 2344)

Syntax

BITOR(value; value)

Parameters

Comment: First number, *Type:* Whole number (like 1, 132, 2344)

Comment: Second number, *Type:* Whole number (like 1, 132, 2344)

Examples

BITOR(12;10) returns 14 (because decimal 12 is binary 1100, and decimal 10 is binary 1010; and 1100 "ored" with 1010 is 1110, which is integer 14).

Related Functions

[BITAND](#)
[BITXOR](#)

8.1.1.4 BITRSHIFT

The BITRSHIFT() function performs a bit-wise right shift operation of the first parameter. The number of bits to shift by is specified by the second parameter. Note that a negative number of bits to right shift by becomes a left shift.

Return type: Whole number (like 1, 132, 2344)

Syntax

BITRSHIFT(value; shift size)

Parameters

Comment: First number, *Type:* Whole number (like 1, 132, 2344)

Comment: Amount to right shift by, *Type:* Whole number (like 1, 132, 2344)

Related Functions

[BITLSHIFT](#)

8.1.1.5 BITXOR

The BITXOR() function performs a bit-wise exclusive-OR operation for the two integer parameters.

Return type: Whole number (like 1, 132, 2344)

Syntax

BITXOR(value; value)

Parameters

Comment: First number, *Type:* Whole number (like 1, 132, 2344)

Comment: Second number, *Type:* Whole number (like 1, 132, 2344)

Examples

BITXOR(12;10) returns 6 (because decimal 12 is binary 1100, and decimal 10 is binary 1010; and 1100 "xored" with 1010 is 0110, which is integer 6).

Related Functions

[BITAND](#)

[BITOR](#)

8.1.2 Conversion

8.1.2.1 ARABIC

The ARABIC() function converts a roman numeral into a number.

Return type: Whole number (like 1, 132, 2344)

Syntax

ARABIC(Numeral)

Parameters

Comment: Numeral, *Type:* Text

Examples

ARABIC("IV") returns 4

Examples

ARABIC("XCIX") returns 99

Related Functions

[ROMAN](#)

8.1.2.2 ASCIITOCHAR

The ASCIITOCHAR() function returns the character for each given ASCII code

Return type: Text

Syntax

ASCIITOCHAR(value)

Parameters

Comment: The ASCII values to convert, *Type:* Whole number (like 1, 132, 2344)

Examples

ASCIITOCHAR(118) returns "v"

Examples

ASCIITOCHAR(75; 68; 69) returns "KDE"

8.1.2.3 BOOL2INT

The BOOL2INT() function returns an integer value for a given boolean value. This method is intended for using a boolean value in methods which require an integer.

Return type: Whole number (like 1, 132, 2344)

Syntax

BOOL2INT(value)

Parameters

Comment: Bool value to convert, *Type:* A truth value (TRUE or FALSE)

Examples

BOOL2INT(True) returns 1

Examples

BOOL2INT(False) returns 0

Related Functions

[INT2BOOL](#)

8.1.2.4 BOOL2STRING

The BOOL2STRING() function returns a string value for a given boolean value. This method is intended for using a boolean in methods which require a string

Return type: Text

Syntax

BOOL2STRING(value)

Parameters

Comment: Bool value to convert, *Type:* A truth value (TRUE or FALSE)

Examples

BOOL2STRING(true) returns "True"

Examples

BOOL2STRING(false) returns "False"

Examples

upper(BOOL2STRING(find("nan";"banana"))) returns TRUE

8.1.2.5 CARX

The CARX() function returns the X position corresponding to the position of a point in a polar landmark.

Return type: Double

Syntax

CARX(Radius;Angle)

Parameters

Comment: Radius, *Type:* Double

Comment: Angle (radians), *Type:* Double

Examples

CARX(12;1.5707) returns 0.00115592

Examples

CARX(12;0) returns 12

Related Functions

[CARY](#)

[POLA](#)

[POLR](#)

8.1.2.6 CARY

The CARY() function returns the Y position corresponding to the position of a point in a polar landmark.

Return type: Double

Syntax

CARY(Radius;Angle)

Parameters

Comment: Radius, *Type:* Double

Comment: Angle (radians), *Type:* Double

Examples

CARY(12;1.5707) returns 12

Examples

CARY(12;0) returns 0

Related Functions

[CARX](#)

[POLA](#)

[POLR](#)

8.1.2.7 CHARTOASCII

The CHARTOASCII() function returns the ASCII code for the given character.

Return type: Whole number (like 1, 132, 2344)

Syntax

CHARTOASCII(value)

Parameters

Comment: A one character string to convert, *Type:* Text

Examples

CHARTOASCII("v") returns 118

Examples

CHARTOASCII(r) is an error. The character must be in quotes.

8.1.2.8 DECSEX

The DECSEX() function converts a double value to a time value.

Return type: Double

Syntax

DECSEX(double)

Parameters

Comment: Value, *Type:* Double

Examples

DECSEX(1.6668) returns 1:40

Examples

DECSEX(7.8) returns 7:47

8.1.2.9 INT2BOOL

The INT2BOOL() function returns a boolean value for a given integer number. This method is intended for using an integer in methods which require a boolean. It only accepts 0 or 1. If any other value is given, false is returned.

Return type: A truth value (TRUE or FALSE)

Syntax

INT2BOOL(value)

Parameters

Comment: Integer value to convert, *Type:* Whole number (like 1, 132, 2344)

Examples

INT2BOOL(1) returns true

Examples

INT2BOOL(0) returns false

Examples

OR(INT2BOOL(1); false) returns true

Related Functions

[BOOL2INT](#)

8.1.2.10 NUM2STRING

The NUM2STRING() function returns a string value for a given number. Note that Calligra Sheets can auto-convert numbers to strings if needed, so this function should rarely be needed.

Return type: Text

Syntax

NUM2STRING(value)

Parameters

Comment: Number to convert into string, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

NUM2STRING(10) returns "10"

Examples

NUM2STRING(2.05) returns "2.05"

Examples

=find("101";NUM2STRING(A1)) (A1 = 2.010102) returns True

Related Functions

[STRING](#)

8.1.2.11 POLA

The POLA() function returns the angle (in radians) corresponding to the position of a point in a cartesian landmark.

Return type: Double

Syntax

POLA(X;Y)

Parameters

Comment: Value in X, *Type:* Double

Comment: Value in Y, *Type:* Double

Examples

POLA(12;12) returns 0.78539816

Examples

POLA(12;0) returns 0

Examples

POLA(0;12) returns 1.5707

Related Functions

[POLR](#)

[CARX](#)

[CARY](#)

8.1.2.12 POLR

The POLR() function returns the radius corresponding to the position of a point in a cartesian landmark.

Return type: Double

Syntax

POLR(X;Y)

Parameters

Comment: Value in X, *Type:* Double

Comment: Value in Y, *Type:* Double

Examples

POLR(12;12) returns 16.9705

Examples

POLR(12;0) returns 12

Related Functions

[POLA](#)

[CARX](#)

[CARY](#)

8.1.2.13 ROMAN

The ROMAN() function returns the number in Roman format. Only positive whole numbers can be converted. The optional Format argument specifies the level of conciseness, and defaults to 0.

Return type: Text

Syntax

ROMAN(Number)

Parameters

Comment: Number, *Type:* Whole number (like 1, 132, 2344)

Comment: Format, *Type:* Whole number (like 1, 132, 2344)

Examples

ROMAN(99) returns "XCIX"

Examples

ROMAN(-55) returns "Err"

Related Functions

[ARABIC](#)

8.1.2.14 SEXDEC

The SEXDEC() function returns a decimal value. You can also supply a time value.

Return type: Double

Syntax

SEXDEC(time value) or SEXDEC(hours;minutes;seconds)

Parameters

Comment: Hours, *Type:* Whole number (like 1, 132, 2344)

Comment: Minutes, *Type:* Whole number (like 1, 132, 2344)

Comment: Seconds, *Type:* Whole number (like 1, 132, 2344)

Examples

SEXDEC(1;5;7) returns 1.0852778

Examples

DECSEX("8:05") returns 8.08333333

8.1.2.15 STRING

The STRING() function returns a string value for a given number. It is the same as the NUM2STRING function.

Return type: Text

Syntax

Parameters

Comment: Number to convert into string, *Type:* A floating point value (like 1.3, 0.343, 253)

Related Functions

[NUM2STRING](#)

8.1.3 Database

8.1.3.1 DAVERAGE

Calculates the average in a column of a database specified by a set of conditions for values that are numbers

Return type: FLOAT

Syntax

DAVERAGE(Database; "Header"; Conditions)

Parameters

Comment: Range marking the database, *Type:* A range of strings

Comment: String marking the column in the database, *Type:* Text

Comment: Range marking the conditions, *Type:* A range of strings

Examples

DAVERAGE(A1:C5; "Salary"; A9:A11)

8.1.3.2 DCOUNT

Counts the cells containing numeric values in a column of a database specified by a set of conditions.

Return type: FLOAT

Syntax

DCOUNT(Database; "Header"; Conditions)

Parameters

Comment: Range marking the database, *Type:* A range of strings

Comment: String marking the column in the database, *Type:* Text

Comment: Range marking the conditions, *Type:* A range of strings

Examples

DCOUNT(A1:C5; "Salary"; A9:A11)

Related Functions

[DCOUNTA](#)

8.1.3.3 DCOUNTA

Counts the cells containing numeric or alphanumeric values in a column of a database specified by a set of conditions.

Return type: FLOAT

Syntax

DCOUNTA(Database; "Header"; Conditions)

Parameters

Comment: Range marking the database, *Type:* A range of strings

Comment: String marking the column in the database, *Type:* Text

Comment: Range marking the conditions, *Type:* A range of strings

Examples

DCOUNTA(A1:C5; "Salary"; A9:A11)

Related Functions

[DCOUNT](#)

8.1.3.4 DGET

Returns a single value from a column of a database specified by a set of conditions. This function returns an error if no value or more than one value exist.

Return type: FLOAT

Syntax

DGET(Database; "Header"; Conditions)

Parameters

Comment: Range marking the database, *Type:* A range of strings

Comment: String marking the column in the database, *Type:* Text

Comment: Range marking the conditions, *Type:* A range of strings

Examples

DGET(A1:C5; "Salary"; A9:A11)

8.1.3.5 DMAX

Returns the largest value in a column of a database specified by a set of conditions.

Return type: FLOAT

Syntax

DMAX(Database; "Header"; Conditions)

Parameters

Comment: Range marking the database, *Type:* A range of strings

Comment: String marking the column in the database, *Type:* Text

Comment: Range marking the conditions, *Type:* A range of strings

Examples

DMAX(A1:C5; "Salary"; A9:A11)

Related Functions

[DMIN](#)

8.1.3.6 DMIN

Returns the smallest values in a column of a database specified by a set of conditions.

Return type: FLOAT

Syntax

DMIN(Database; "Header"; Conditions)

Parameters

Comment: Range marking the database, *Type:* A range of strings
Comment: String marking the column in the database, *Type:* Text
Comment: Range marking the conditions, *Type:* A range of strings

Examples

DMIN(A1:C5; "Salary"; A9:A11)

Related Functions

[DMAX](#)

8.1.3.7 DPRODUCT

Returns the product of all numeric values in a column of a database specified by a set of conditions.

Return type: FLOAT

Syntax

DPRODUCT(Database; "Header"; Conditions)

Parameters

Comment: Range marking the database, *Type:* A range of strings
Comment: String marking the column in the database, *Type:* Text
Comment: Range marking the conditions, *Type:* A range of strings

Examples

DPRODUCT(A1:C5; "Salary"; A9:A11)

8.1.3.8 DSTDEV

Returns the estimate of the standard deviation of a population based on a sample using all numeric values in a column of a database specified by a set of conditions.

Return type: FLOAT

Syntax

DSTDEV(Database; "Header"; Conditions)

Parameters

Comment: Range marking the database, *Type:* A range of strings
Comment: String marking the column in the database, *Type:* Text
Comment: Range marking the conditions, *Type:* A range of strings

Examples

DSTDEV(A1:C5; "Salary"; A9:A11)

Related Functions

[DSTDEVP](#)

8.1.3.9 DSTDEVP

Returns the standard deviation of a population based on the entire population using all numeric values in a column of a database specified by a set of conditions.

Return type: FLOAT

Syntax

DSTDEVP(Database; "Header"; Conditions)

Parameters

Comment: Range marking the database, *Type:* A range of strings

Comment: String marking the column in the database, *Type:* Text

Comment: Range marking the conditions, *Type:* A range of strings

Examples

DSTDEVP(A1:C5; "Salary"; A9:A11)

Related Functions

[DSTDEV](#)

8.1.3.10 DSUM

Sums up the numbers in a column of a database specified by a set of conditions.

Return type: FLOAT

Syntax

DSUM(Database; "Header"; Conditions)

Parameters

Comment: Range marking the database, *Type:* A range of strings

Comment: String marking the column in the database, *Type:* Text

Comment: Range marking the conditions, *Type:* A range of strings

Examples

DSUM(A1:C5; "Salary"; A9:A11)

8.1.3.11 DVAR

Returns the estimate of the variance of a population based on a sample using all numeric values in a column of a database specified by a set of conditions.

Return type: FLOAT

Syntax

DVAR(Database; "Header"; Conditions)

Parameters

Comment: Range marking the database, *Type:* A range of strings

Comment: String marking the column in the database, *Type:* Text

Comment: Range marking the conditions, *Type:* A range of strings

Examples

DVAR(A1:C5; "Salary"; A9:A11)

Related Functions

[DVARP](#)

8.1.3.12 DVARP

Returns the variance of a population based on the entire population using all numeric values in a column of a database specified by a set of conditions.

Return type: FLOAT

Syntax

DVARP(Database; "Header"; Conditions)

Parameters

Comment: Range marking the database, *Type:* A range of strings

Comment: String marking the column in the database, *Type:* Text

Comment: Range marking the conditions, *Type:* A range of strings

Examples

DVARP(A1:C5; "Salary"; A9:A11)

Related Functions

[DVAR](#)

8.1.3.13 GETPIVOTDATA

Fetches summary data from a pivot table.

Return type: FLOAT

Syntax

GETPIVOTDATA(Database; "Sales")

Parameters

Comment: Range containing the pivot table, *Type:* A range of strings

Comment: Name of the field of which you want the summary data, *Type:* Text

8.1.4 Date & Time

8.1.4.1 CURRENTDATE

The CURRENTDATE() function returns the current date. It is equivalent to the TODAY function.

Return type: Date

Syntax

CURRENTDATE()

Parameters

Examples

CURRENTDATE() returns "Saturday 13 April 2002"

Related Functions

[CURRENTTIME](#)

[TODAY](#)

8.1.4.2 CURRENTDATETIME

The CURRENTDATETIME() function returns the current date and time.

Return type: Date

Syntax

CURRENTDATETIME()

Parameters

Examples

CURRENTDATETIME() returns "Saturday 13 April 2002 19:12:01"

8.1.4.3 CURRENTTIME

The CURRENTTIME() function returns the current time formatted with local parameters.

Return type: Date

Syntax

CURRENTTIME()

Parameters

Examples

CURRENTTIME() returns "19:12:01"

8.1.4.4 DATE

The DATE() function returns the date formatted with local parameters.

Return type: Text

Syntax

DATE(year;month;date)

Parameters

Comment: Year, *Type:* Whole number (like 1, 132, 2344)

Comment: Month, *Type:* Whole number (like 1, 132, 2344)

Comment: Day, *Type:* Whole number (like 1, 132, 2344)

Examples

DATE(2000;5;5) returns Friday 05 May 2000

8.1.4.5 DATE2UNIX

DATE2UNIX() function converts a date and time value to unix time.

A unix time is the number of seconds after midnight January 1st, 1970.

Return type: Whole number (like 1, 132, 2344)

Syntax

DATE2UNIX(date)

Parameters

Comment: Date, *Type:* Text

Examples

DATE2UNIX("01/01/2000") returns 946,684,800

8.1.4.6 DATEDIF

The DATEDIF() function returns the difference between two dates.

Interval must be one of the following: "m": month; "d": days; "y": complete years; "ym": month excluding years; "yd": days excluding years; "md": days excluding months and years

Return type: Whole number (like 1, 132, 2344)

Syntax

DATEDIF(first date; second date; interval)

Parameters

Comment: First date, *Type:* Text

Comment: Second date, *Type:* Text

Comment: interval, *Type:* Text

Examples

DATEDIF(A1;A2;"d") A1 is "1st of January 1995" and A2 is "15th of June 1999" returns number of days 1626

Examples

DATEDIF(A1;A2;"m") A1 is "1st of January 1995" and A2 is "15th of June 1999" returns number of months 53

8.1.4.7 DATEVALUE

The DATEVALUE function returns a number representing the day, i.e the number of days elapsed since December 31, 1899.

Return type: Whole number (like 1, 132, 2344)

Syntax

DATEVALUE(date)

Parameters

Comment: Date, *Type:* Text

Examples

DATEVALUE("2/22/2002") returns 37309

Related Functions

[TIMEVALUE](#)

8.1.4.8 DAY

The DAY functions returns the day of a date. If no parameter is specified the current day gets returned.

Return type: Whole number (like 1, 132, 2344)

Syntax

DAY(date)

Parameters

Comment: Date, *Type:* Text

Examples

DAY("2/22/2002") returns 22

Examples

DAY(2323.1285) returns 11

Related Functions

[MONTH](#)
[YEAR](#)

8.1.4.9 DAYNAME

The DAYNAME() function returns the name of the day of the week (1..7). In some countries the first day of the week is Monday, while in others the first day of the week is Sunday.

Return type: Text

Syntax

DAYNAME(weekday)

Parameters

Comment: Number of day in week (1..7), *Type:* Whole number (like 1, 132, 2344)

Examples

DAYNAME(1) returns Monday (if the week starts on Monday)

Related Functions

[WEEKDAY](#)

8.1.4.10 DAYOFYEAR

The DAYOFYEAR() function returns the number of the day in the year (1...365).

Return type: Whole number (like 1, 132, 2344)

Syntax

DAYOFYEAR(year;month;date)

Parameters

Comment: Year, *Type:* Whole number (like 1, 132, 2344)

Comment: Month, *Type:* Whole number (like 1, 132, 2344)

Comment: Day, *Type:* Whole number (like 1, 132, 2344)

Examples

DAYOFYEAR(2000;12;1) returns 336

Examples

DAYOFYEAR(2000;2;29) returns 60

8.1.4.11 DAYS

The DAYS() function returns the difference between two dates in days.

Return type: Whole number (like 1, 132, 2344)

Syntax

DAYS(date2; date1)

Parameters

Comment: First (earlier) date value, *Type:* Text

Comment: Second date value, *Type:* Text

Examples

DAYS("2002-02-22"; "2002-02-26") returns 4

8.1.4.12 DAYS360

The DAYS360() function returns the number of days from date1 to date2 using a 360-day calendar in which all months are assumed to have 30 days. If method is false (default) the US method will be used, the European otherwise.

Return type: Whole number (like 1, 132, 2344)

Syntax

DAYS360(date1; date2; method)

Parameters

Comment: Date1, *Type:* Text

Comment: Date2, *Type:* Text

Comment: Method, *Type:* A truth value (TRUE or FALSE)

Examples

DAYS360("2/22/2002"; "4/21/2002"; FALSE) returns 59

Related Functions

[DAYS](#)

[MONTHS](#)

[WEEKS](#)

[YEARS](#)

8.1.4.13 DAYSINMONTH

The function DAYSINMONTH() returns the number of days in the given year and month.

Return type: Whole number (like 1, 132, 2344)

Syntax

DAYSINMONTH(year;month)

Parameters

Comment: Year, *Type:* Whole number (like 1, 132, 2344)

Comment: Month, *Type:* Whole number (like 1, 132, 2344)

Examples

DAYSINMONTH(2000;2) returns 29

8.1.4.14 DAYSINYEAR

The function DAYSINYEAR() returns the number of days in the given year.

Return type: Whole number (like 1, 132, 2344)

Syntax

DAYSINYEAR(year)

Parameters

Comment: Year, *Type:* Whole number (like 1, 132, 2344)

Examples

DAYSINYEAR(2000) returns 366

8.1.4.15 EASTERSUNDAY

The EASTERSUNDAY() function returns the date which corresponds to Easter Sunday in the year given as the parameter.

Return type: Date

Syntax

EASTERSUNDAY(year)

Parameters

Comment: Year, *Type:* Whole number (like 1, 132, 2344)

Examples

EASTERSUNDAY(2003) returns "20th April 2003"

8.1.4.16 EDATE

The EDATE functions returns the date that is specified by a given date and a number of months before or after that date.

Return type: Date

Syntax

EDATE(date; months)

Parameters

Comment: Date, *Type:* Text

Comment: Months, *Type:* Whole number (like 1, 132, 2344)

Examples

EDATE("2/22/2002"; 3) returns "5/22/2002"

Examples

EDATE("3/31/2002"; -1) returns "2/28/2002"

Related Functions

[DATE](#)

[EOMONTH](#)

8.1.4.17 EOMONTH

The EOMONTH functions returns the last day in the month specified by a date and the number of months from that date.

Return type: Date

Syntax

EOMONTH(date; months)

Parameters

Comment: Date, *Type:* Text

Comment: Months, *Type:* Whole number (like 1, 132, 2344)

Examples

EOMONTH("2/22/2002"; 3) returns "5/31/2002"

Examples

EOMONTH("3/12/2002"; -1) returns "2/28/2002"

Examples

EOMONTH("3/12/2002"; 0) returns "3/31/2002"

Related Functions

[EDATE](#)
[MONTH](#)

8.1.4.18 HOUR

The HOUR functions returns the hour of a time. If no parameter is specified the current hour gets returned.

Return type: Whole number (like 1, 132, 2344)

Syntax

HOUR(time)

Parameters

Comment: Time, *Type:* Text

Examples

HOUR("22:10:12") returns 22

Examples

HOUR(0.1285) returns 3

Related Functions

[MINUTE](#)
[SECOND](#)

8.1.4.19 HOURS

The HOURS() function returns the value of the hours in a time expression.

Return type: Whole number (like 1, 132, 2344)

Syntax

HOURS(time)

Parameters

Comment: Time, *Type:* Text

Examples

HOURS("10:5:2") returns 10

8.1.4.20 ISLEAPYEAR

The function ISLEAPYEAR() returns True if the given year is leap.

Return type: A truth value (TRUE or FALSE)

Syntax

ISLEAPYEAR(year)

Parameters

Comment: Year, *Type:* Whole number (like 1, 132, 2344)

Examples

ISLEAPYEAR(2000) returns True

8.1.4.21 ISOWEEKNUM

The ISOWEEKNUM() function returns number of the week which the date falls into. Note that this function is compliant with the ISO8601 standard: a week always begins on a Monday, and ends on a Sunday. The first week of a year is that week which contains the first Thursday of the year.

Return type: Whole number (like 1, 132, 2344)

Syntax

ISOWEEKNUM(date)

Parameters

Comment: Date, *Type:* Text

Examples

ISOWEEKNUM(A1) returns 51 when A1 is "21st of Dec".

Related Functions

[WEEKNUM](#)

8.1.4.22 MINUTE

The MINUTE functions returns the minutes of a time. If no parameter is specified the current minute is returned.

Return type: Whole number (like 1, 132, 2344)

Syntax

MINUTE(time)

Parameters

Comment: Time, *Type:* Text

Examples

MINUTE("22:10:12") returns 10

Examples

MINUTE(0.1234) returns 57

Related Functions

[HOUR](#)
[SECOND](#)

8.1.4.23 MINUTES

The MINUTES() function returns the value of the minutes in a time expression.

Return type: Whole number (like 1, 132, 2344)

Syntax

MINUTES(time)

Parameters

Comment: Time, *Type:* Text

Examples

MINUTES("10:5:2") returns 5

8.1.4.24 MONTH

The MONTH functions returns the month of a date. If no parameter is specified the current month gets returned.

Return type: Whole number (like 1, 132, 2344)

Syntax

MONTH(date)

Parameters

Comment: Date, *Type:* Text

Examples

MONTH("2/22/2002") returns 2

Examples

MONTH(2323.1285) returns 5

Related Functions

[DAY](#)
[YEAR](#)

8.1.4.25 MONTHNAME

The MONTHNAME() function returns the name of the month (1...12).

Return type: Text

Syntax

MONTHNAME(number)

Parameters

Comment: Number of month (1..12), *Type:* Whole number (like 1, 132, 2344)

Examples

MONTHNAME(5) returns May

8.1.4.26 MONTHS

The MONTHS() function returns the difference between two dates in months. The third parameter indicates the calculation mode: if the mode is 0, MONTHS() returns the maximal possible number of months between those days. If the mode is 1, it only returns the number of complete months in between.

Return type: Whole number (like 1, 132, 2344)

Syntax

MONTHS(date2; date1; mode)

Parameters

Comment: First (earlier) date value, *Type:* Text

Comment: Second date value, *Type:* Text

Comment: Calculation mode, *Type:* Whole number (like 1, 132, 2344)

Examples

MONTHS("2002-01-18"; "2002-02-26"; 0) returns 1, because there is 1 month and 8 days in between

Examples

MONTHS("2002-01-19"; "2002-02-26"; 1) returns 0, because there is not a whole month in between, starting at the first day of the month

8.1.4.27 NETWORKDAY

The NETWORKDAY() function returns the number of working days between startdate and end-date.

Holidays must be one of the following: number = days to add, a single date or an array of dates.

Return type: Whole number (like 1, 132, 2344)

Syntax

NETWORKDAY(start date; end date; holidays)

Parameters

Comment: Start date, *Type:* Text

Comment: End date, *Type:* Text

Comment: Holidays, *Type:* Text

Examples

NETWORKDAY("01/01/2001";"01/08/2001") returns 5 workdays

Examples

NETWORKDAY("01/01/2001";"01/08/2001";2) returns 3 workdays

8.1.4.28 NOW

The NOW() function returns the current date and time. It is identical with CURRENTDATETIME and provided for compatibility with other applications.

Return type: Date

Syntax

NOW()

Parameters

Examples

NOW() returns "Saturday 13 April 2002 19:12:01"

Related Functions

[CURRENTTIME](#)
[TODAY](#)

8.1.4.29 SECOND

The SECOND functions returns the seconds of a time. If no parameter is specified the current second is returned.

Return type: Whole number (like 1, 132, 2344)

Syntax

SECOND(time)

Parameters

Comment: Time, *Type:* Text

Examples

SECOND("22:10:12") returns 12

Examples

SECOND(0.1234) returns 42

Related Functions

[HOUR](#)
[MINUTE](#)

8.1.4.30 SECONDS

The SECONDS() function returns the value of the seconds in a time expression.

Return type: Whole number (like 1, 132, 2344)

Syntax

SECONDS(time)

Parameters

Comment: Time, *Type:* Text

Examples

SECONDS("10:5:2") returns 2

8.1.4.31 TIME

The TIME() function returns the time formatted with local parameters.

Return type: Text

Syntax

TIME(hours;minutes;seconds)

Parameters

Comment: Hours, *Type:* Whole number (like 1, 132, 2344)

Comment: Minutes, *Type:* Whole number (like 1, 132, 2344)

Comment: Seconds, *Type:* Whole number (like 1, 132, 2344)

Examples

TIME(10;2;2) returns 10:02:02

Examples

TIME(10;70;0) returns 11:10:0

Examples

TIME(10;-40;0) returns 9:20:0

8.1.4.32 TIMEVALUE

The TIMEVALUE() function returns a number (between 0 and 1) representing the time of day.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

TIMEVALUE(time)

Parameters

Comment: Time, *Type:* Text

Examples

TIMEVALUE("10:05:02") returns 0.42

Related Functions

[DATEVALUE](#)

8.1.4.33 TODAY

The TODAY() function returns the current date.

Return type: Date

Syntax

TODAY()

Parameters

Examples

TODAY() returns "Saturday 13 April 2002"

Related Functions

[CURRENTTIME](#)

[NOW](#)

8.1.4.34 UNIX2DATE

UNIX2DATE() function converts unix time to a date and time value.

A unix time is the number of seconds after midnight January 1st, 1970.

Return type: Date

Syntax

UNIX2DATE(unixtime)

Parameters

Comment: Unixtime, *Type:* Whole number (like 1, 132, 2344)

Examples

UNIX2DATE(0) returns 1970-01-01

8.1.4.35 WEEKDAY

The WEEKDAY() function returns the weekday of given date. If the method is 1 (default) WEEKDAY() returns 1 for sunday, 2 for monday,.. If the method is 2, monday is 1, tuesday 2, ... and if the method is 3 WEEKDAY() returns 0 for monday, 1 for tuesday,...

Return type: Whole number (like 1, 132, 2344)

Syntax

WEEKDAY(date; method)

Parameters

Comment: Date, *Type:* Text

Comment: Method (optional), *Type:* Whole number (like 1, 132, 2344)

Examples

WEEKDAY("2002-02-22"; 2) returns 5

Related Functions

[DAYNAME](#)

8.1.4.36 WEEKNUM

The WEEKNUM() function returns the non-ISO week number in which the date falls into.

Return type: Whole number (like 1, 132, 2344)

Syntax

WEEKNUM(date; method)

Parameters

Comment: Date, *Type:* Text

Comment: Method (optional), *Type:* Whole number (like 1, 132, 2344)

Examples

WEEKNUM(A1; 1) returns 11 when A1 is "9th of March 2008". Number of the week in the year, with a week beginning on Sunday (1, this is the default if Method is omitted.)

Examples

WEEKNUM(A1; 2) returns 10 when A1 is "9th of March 2008". Number of the week in the year, with a week beginning on Monday (2)

Related Functions

[ISOWEEKNUM](#)

8.1.4.37 WEEKS

The WEEKS() function returns the difference between two dates in weeks. The third parameter indicates the calculation mode: if the mode is 0, WEEKS() returns the maximal possible number of weeks between those days. If the mode is 1, it only returns the number of whole weeks in between.

Return type: Whole number (like 1, 132, 2344)

Syntax

WEEKS(date2; date1; mode)

Parameters

Comment: First (earlier) date value, *Type:* Text

Comment: Second date value, *Type:* Text

Comment: Calculation mode, *Type:* Whole number (like 1, 132, 2344)

Examples

WEEKS("2002-02-18"; "2002-02-26"; 0) returns 1, because there is one week and 1 day in between

Examples

WEEKS("2002-19-02"; "2002-19-02"; 1) returns 0, because there is not a whole week in between, starting at the first day of the week (monday or sunday, depending on your local settings)

8.1.4.38 WEEKSINYEAR

The function WEEKSINYEAR() returns the number of weeks in the given year.

Return type: Whole number (like 1, 132, 2344)

Syntax

WEEKSINYEAR(year)

Parameters

Comment: Year, *Type:* Whole number (like 1, 132, 2344)

Examples

WEEKSINYEAR(2000) returns 52

8.1.4.39 WORKDAY

The WORKDAY() function returns the date which is working days from the start date.

Holidays must be one of the following: number = days to add, a single date or an array of dates.

Return type: Date

Syntax

WORKDAY(start date; days; holidays)

Parameters

Comment: Start date, *Type:* Text

Comment: Working days, *Type:* Whole number (like 1, 132, 2344)

Comment: Holidays, *Type:* Text

Examples

if B9 is "01/01/2001", D3 is "01/03/2001", D4 is "01/04/2001" then WORKDAY(B9;2;D3:D4) returns "Fri Jan 5 2001"

8.1.4.40 YEAR

The YEAR functions returns the year of a date. If no parameter is specified the current year gets returned.

Return type: Whole number (like 1, 132, 2344)

Syntax

YEAR(date)

Parameters

Comment: Date, *Type:* Text

Examples

YEAR("2/22/2002") returns 2002

Examples

YEAR(2323.1285) returns 1906

Related Functions

[DAY](#)

[MONTH](#)

8.1.4.41 YEARFRAC

The YEARFRAC() function returns the number of full days between start date and end date according to the basis.

Basis must be one of the following: 0 = 30/360 US, 1 = Actual/actual, 2 = Actual/360, 3 = Actual/365, 4 = European 30/360

Return type: Whole number (like 1, 132, 2344)

Syntax

YEARFRAC(start date; end date; basis)

Parameters

Comment: First date, *Type:* Text

Comment: Second date, *Type:* Text

Comment: interval, *Type:* Text

8.1.4.42 YEARS

The YEARS() function returns the difference between two dates in years. The third parameter indicates the calculation mode: if the mode is 0, YEARS() returns the maximal possible number of years between those days. If the mode is 1, it only returns whole years, starting at the 1st Jan and ending on the 31st Dec.

Return type: Whole number (like 1, 132, 2344)

Syntax

YEARS(date2; date1; mode)

Parameters

Comment: First (earlier) date value, *Type:* Text

Comment: Second date value, *Type:* Text

Comment: Calculation mode, *Type:* Whole number (like 1, 132, 2344)

Examples

YEARS("2001-02-19"; "2002-02-26"; 0) returns 1, because there is one year and 7 days in between

Examples

YEARS("2002-02-19"; "2002-02-26"; 1) returns 0, because there is not a whole year in between, starting at the first day of the year

8.1.5 Engineering

8.1.5.1 BASE

The BASE() function converts a number from base-10 to a string value in a target base from 2 to 36.

Return type: Text

Syntax

BASE(number;base;prec)

Parameters

Comment: Number, *Type:* A floating point value (like 1.3, 0.343, 253)
Comment: Base, *Type:* Whole number (like 1, 132, 2344)
Comment: MinLength, *Type:* Whole number (like 1, 132, 2344)

Examples

BASE(128;8) returns "200"

8.1.5.2 BESSELI

The BESSELI() function returns the modified Bessel function In(x).

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

BESSELI(X;N)

Parameters

Comment: Where the function is evaluated, *Type:* A floating point value (like 1.3, 0.343, 253)
Comment: Order of the function, *Type:* Whole number (like 1, 132, 2344)

Examples

BESSELI(0.7;3) returns 0.007367374

Related Functions

[BESSELJ](#)
[BESSELK](#)
[BESSELY](#)

8.1.5.3 BESSELJ

The BESSELJ() function returns the Bessel function.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

BESSELJ(X;N)

Parameters

Comment: Where the function is evaluated, *Type:* A floating point value (like 1.3, 0.343, 253)
Comment: Order of the function, *Type:* Whole number (like 1, 132, 2344)

Examples

BESSELJ(0.89;3) returns 0.013974004

Related Functions

[BESSELI](#)
[BESSELK](#)
[BESSELY](#)

8.1.5.4 BESSELK

The BESSELK() function returns the modified Bessel function, which is equivalent to the Bessel function evaluated for purely imaginary arguments.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

BESSELK(X;N)

Parameters

Comment: Where the function is evaluated, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Order of the function, *Type:* Whole number (like 1, 132, 2344)

Examples

BESSELK(3;9) returns 397.95880

Related Functions

[BESSELI](#)

[BESSELJ](#)

[BESSELY](#)

8.1.5.5 BESSELY

The BESSELY() function returns the Bessel function, which is also called the Weber function or the Neumann function.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

BESSELY(X;N)

Parameters

Comment: Where the function is evaluated, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Order of the function, *Type:* Whole number (like 1, 132, 2344)

Examples

BESSELY(4;2) equals 0.215903595

Related Functions

[BESSELI](#)

[BESSELJ](#)

[BESSELK](#)

8.1.5.6 BIN2DEC

The BIN2DEC() function returns the value formatted as a decimal number.

Return type: Whole number (like 1, 132, 2344)

Syntax

BIN2DEC(value)

Parameters

Comment: The value to convert, *Type:* Whole number (like 1, 132, 2344)

Examples

BIN2DEC("1010") returns 10

Examples

BIN2DEC("11111") returns 31

8.1.5.7 BIN2HEX

The BIN2HEX() function returns the value formatted as a hexadecimal number.

Return type: Text

Syntax

BIN2HEX(value)

Parameters

Comment: The value to convert, *Type:* Text

Comment: The minimum length of the output, *Type:* Whole number (like 1, 132, 2344)

Examples

BIN2HEX("1010") returns "a"

Examples

BIN2HEX("11111") returns "1f"

8.1.5.8 BIN2OCT

The BIN2OCT() function returns the value formatted as an octal number.

Return type: Text

Syntax

BIN2OCT(value)

Parameters

Comment: The value to convert, *Type:* Text

Comment: The minimum length of the output, *Type:* Whole number (like 1, 132, 2344)

Examples

BIN2OCT("1010") returns "12"

Examples

BIN2OCT("11111") returns "37"

8.1.5.9 COMPLEX

The COMPLEX(real;imag) returns a complex number of form $x+yi$.

Return type: Text

Syntax

COMPLEX(real;imag)

Parameters

Comment: Real coefficient, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Imaginary coefficient, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

COMPLEX(1.2;3.4) returns "1.2+3.4i"

Examples

COMPLEX(0;-1) returns "-i"

8.1.5.10 CONVERT

The CONVERT() function returns a conversion from one measurement system to another.

Supported mass units: g (gram), sg (pieces), lbm (pound), u (atomic mass), ozm (ounce), stone, ton, grain, pweight (pennyweight), hweight (hundredweight).

Supported distance units: m (meter), in (inch), ft (feet), mi (mile), Nmi (nautical mile), ang (Angstrom), parsec, lightyear.

Supported pressure units: Pa (Pascal), atm (atmosphere), mmHg (mm of Mercury), psi, Torr.

Supported force units: N (Newton), dyn, pound.

Supported energy units: J (Joule), e (erg), c (Thermodynamic calorie), cal (IT calorie), eV (electronvolt), HPh (Horsepower-hour), Wh (Watt-hour), flb (foot-pound), BTU.

Supported power units: W (Watt), HP (horsepower), PS (Pferdestaerke).

Supported magnetism units: T (Tesla), ga (Gauss).

Supported temperature units: C (Celsius), F (Fahrenheit), K (Kelvin).

Supported volume units: l (liter), tsp (teaspoon), tbs (tablespoon), oz (ounce liquid), cup, pt (pint), qt (quart), gal (gallon), barrel, m3 (cubic meter), mi3 (cubic mile), Nmi3 (cubic Nautical mile), in3 (cubic inch), ft3 (cubic foot), yd3 (cubic yard), GRT or regton (gross register ton).

Supported area units: m2 (square meter), mi2 (square mile), Nmi2 (square Nautical mile), in2 (square inch), ft2 (square foot), yd2 (square yard), acre, ha (hectare).

Supported speed units: m/s (meters per second), m/h (meters per hour), mph (miles per hour), kn (knot).

For metric units any of the following prefixes can be used: E (exa, 1E+18), P (peta, 1E+15), T (tera, 1E+12), G (giga, 1E+09), M (mega, 1E+06), k (kilo, 1E+03), h (hecto, 1E+02), e (deka, 1E+01), d (deci, 1E-01), c (centi, 1E-02), m (milli, 1E-03), u (micro, 1E-06), n (nano, 1E-09), p (pico, 1E-12), f (femto, 1E-15), a (atto, 1E-18).

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

CONVERT(Number; From Unit; To Unit)

The Calligra Sheets Handbook

Parameters

Comment: Number, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: From unit, *Type:* Text

Comment: To unit, *Type:* Text

Examples

CONVERT(32;"C";"F") equals 89.6

Examples

CONVERT(3;"lbm";"kg") equals 1.3608

Examples

CONVERT(7.9;"cal";"J") equals 33.0757

8.1.5.11 DEC2BIN

The DEC2BIN() function returns the value formatted as a binary number.

Return type: Text

Syntax

DEC2BIN(value)

Parameters

Comment: The value to convert, *Type:* Whole number (like 1, 132, 2344)

Comment: The minimum length of the output, *Type:* Whole number (like 1, 132, 2344)

Examples

DEC2BIN(12) returns "1100"

Examples

DEC2BIN(55) returns "110111"

8.1.5.12 DEC2HEX

The DEC2HEX() function returns the value formatted as a hexadecimal number.

Return type: Text

Syntax

DEC2HEX(value)

Parameters

Comment: The value to convert, *Type:* Whole number (like 1, 132, 2344)

Comment: The minimum length of the output, *Type:* Whole number (like 1, 132, 2344)

Examples

DEC2HEX(12) returns "c"

Examples

DEC2HEX(55) returns "37"

8.1.5.13 DEC2OCT

The DEC2OCT() function returns the value formatted as an octal number.

Return type: Text

Syntax

DEC2OCT(value)

Parameters

Comment: The value to convert, *Type:* Whole number (like 1, 132, 2344)

Comment: The minimum length of the output, *Type:* Whole number (like 1, 132, 2344)

Examples

DEC2OCT(12) returns "14"

Examples

DEC2OCT(55) returns "67"

8.1.5.14 DELTA

The DELTA() function returns 1 if x equals y, otherwise returns 0. y defaults to 0.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

DELTA(x; y)

Parameters

Comment: Floating point value, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Floating point value, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

DELTA(1.2; 3.4) returns 0

Examples

DELTA(3; 3) returns 1

Examples

DELTA(1; TRUE) returns 1

8.1.5.15 ERF

The ERF() function returns the error function. With a single argument, ERF() returns the error function between 0 and that argument.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

ERF(Lower limit; Upper limit)

Parameters

Comment: Lower limit, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Upper limit, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

ERF(0.4) equals 0.42839236

Related Functions

[ERFC](#)

8.1.5.16 ERFC

The ERFC() function returns the complementary error function.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

ERFC(Lower limit; Upper limit)

Parameters

Comment: Lower limit, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Upper limit, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

ERFC(0.4) equals 0.57160764

Related Functions

[ERF](#)

8.1.5.17 GESTEP

The GESTEP() function returns 1 if x greater or equals y, otherwise returns 0. y defaults to 0.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

GESTEP(x; y)

Parameters

Comment: Floating point value, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Floating point value, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

GESTEP(1.2; 3.4) returns 0

Examples

GESTEP(3; 3) returns 1

Examples

GESTEP(0.4; TRUE) returns 0

Examples

GESTEP(4; 3) returns 1

8.1.5.18 HEX2BIN

The HEX2BIN() function returns the value formatted as a binary number.

Return type: Text

Syntax

HEX2BIN(value)

Parameters

Comment: The value to convert, *Type:* Text

Examples

HEX2BIN("a") returns "1010"

Examples

HEX2BIN("37") returns "110111"

8.1.5.19 HEX2DEC

The HEX2DEC() function returns the value formatted as a decimal number.

Return type: Whole number (like 1, 132, 2344)

Syntax

HEX2DEC(value)

Parameters

Comment: The value to convert, *Type:* Text

Examples

HEX2DEC("a") returns 10

Examples

HEX2DEC("37") returns 55

8.1.5.20 HEX2OCT

The HEX2OCT() function returns the value formatted as an octal number.

Return type: Text

Syntax

HEX2OCT(value)

Parameters

Comment: The value to convert, *Type:* Text

Examples

HEX2OCT("a") returns "12"

Examples

HEX2OCT("37") returns "67"

8.1.5.21 IMABS

The IMABS(complex number) returns the norm of a complex number of form $x+yi$.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

IMABS(complex number)

Parameters

Comment: Complex number, *Type:* Text

Examples

IMABS("1.2+5i") returns 5.1419

Examples

IMABS("-i") returns 1

Examples

IMABS("12") returns 12

8.1.5.22 IMAGINARY

The IMAGINARY(string) returns the imaginary coefficient of a complex.

Return type: Double

Syntax

IMAGINARY(string)

Parameters

Comment: Complex number, *Type:* Text

Examples

IMAGINARY("1.2+3.4i") returns 3.4

Examples

IMAGINARY("1.2") returns 0

8.1.5.23 IMARGUMENT

The IMARGUMENT(complex number) returns the argument of a complex number of form $x+yi$.

Return type: Text

Syntax

IMARGUMENT(complex number)

Parameters

Comment: Complex number, *Type:* Text

Examples

IMARGUMENT("1.2+5i") returns 0.6072

Examples

IMARGUMENT("-i") returns -1.57079633

Examples

IMARGUMENT("12") returns "#Div/0"

8.1.5.24 IMCONJUGATE

The IMCONJUGATE(complex number) returns the conjugate of a complex number of form $x+yi$.

Return type: Text

Syntax

IMCONJUGATE(complex number)

Parameters

Comment: Complex number, *Type:* Text

Examples

IMCONJUGATE("1.2+5i") returns "1.2-5i"

Examples

IMCONJUGATE("-i") returns "i"

Examples

IMCONJUGATE("12") returns "12"

8.1.5.25 IMCOS

The IMCOS(string) returns the cosine of a complex number.

Return type: Text

Syntax

IMCOS(string)

Parameters

Comment: Complex number, *Type:* Text

Examples

IMCOS("1+i") returns "0.83373-0.988898i"

Examples

IMCOS("12i") returns 81 377.4

8.1.5.26 IMCOSH

The IMCOSH(string) returns the hyperbolic cosine of a complex number.

Return type: Text

Syntax

IMCOSH(string)

Parameters

Comment: Complex number, *Type:* Text

Examples

IMCOSH("1+i") returns "0.83373+0.988898i"

Examples

IMCOSH("12i") returns 0.84358

8.1.5.27 IMCOT

The IMCOT(string) returns the cotangent of a complex number.

Return type: Text

Syntax

IMCOT(string)

Parameters

Comment: Complex number, *Type:* Text

Examples

IMCOT("1+i") returns "0.21762-0.86801i"

8.1.5.28 IMCSC

The IMCSC(string) returns the cosecant of a complex number.

Return type: Text

Syntax

IMCSC(string)

Parameters

Comment: Complex number, *Type:* Text

Examples

IMCSC("1+i") returns "0.62151-0.30393i"

8.1.5.29 IMCSCH

The IMCSCH(string) returns the hyperbolic cosecant of a complex number.

Return type: Text

Syntax

IMCSCH(string)

Parameters

Comment: Complex number, *Type:* Text

Examples

IMCSCH("1+i") returns "0.30393-i0.62151"

8.1.5.30 IMDIV

The IMDIV() returns the division of several complex numbers of form $x+yi$.

Return type: Text

Syntax

IMDIV(value;value;...)

Parameters

Comment: Complex number, *Type:* A range of strings

Comment: Complex number, *Type:* A range of strings

Comment: Complex number, *Type:* A range of strings

Comment: Complex number, *Type:* A range of strings

Comment: Complex number, *Type:* A range of strings

Examples

IMDIV(1.2;"3.4+5i") returns "0.111597-0.164114i"

Examples

IMDIV("12+i";"12-i") returns "0.986207+0.16551i"

8.1.5.31 IMEXP

The IMEXP(string) returns the exponential of a complex number.

Return type: Text

Syntax

IMEXP(string)

Parameters

Comment: Complex number, *Type:* Text

Examples

IMEXP("2-i") returns "3.99232-6.21768i"

Examples

IMEXP("12i") returns "0.843854-0.536573i"

8.1.5.32 IMLN

The IMLN(string) returns the natural logarithm of a complex number.

Return type: Text

Syntax

IMLN(string)

Parameters

Comment: Complex number, *Type:* Text

Examples

IMLN("3-i") returns "1.15129-0.321751i"

Examples

IMLN("12") returns 2.48491

8.1.5.33 IMLOG10

The IMLOG10(string) returns the base-10 logarithm of a complex number.

Return type: Text

Syntax

IMLOG10(string)

Parameters

Comment: Complex number, *Type:* Text

Examples

IMLOG10("3+4i") returns "0.69897+0.402719i"

8.1.5.34 IMLOG2

The IMLOG2(string) returns the base-2 logarithm of a complex number.

Return type: Text

Syntax

IMLOG2(string)

Parameters

Comment: Complex number, *Type:* Text

Examples

IMLOG2("3+4i") returns "2.321928+1.337804i"

8.1.5.35 IMPOWER

The IMPOWER(string) returns a complex number raised to a power.

Return type: Text

Syntax

IMPOWER(string)

Parameters

Comment: Complex number, *Type:* Text

Comment: Power, *Type:* Whole number (like 1, 132, 2344)

Examples

IMPOWER("4-i";2) returns "15-8i"

Examples

IMPOWER("1.2";2) returns 1.44

8.1.5.36 IMPRODUCT

The IMPRODUCT() returns the product of several complex numbers of form $x+yi$.

Return type: Text

Syntax

IMPRODUCT(value;value;...)

Parameters

Comment: Complex number, *Type:* A range of strings

Comment: Complex number, *Type:* A range of strings

Comment: Complex number, *Type:* A range of strings

Comment: Complex number, *Type:* A range of strings

Comment: Complex number, *Type:* A range of strings

Examples

IMPRODUCT(1.2;"3.4+5i") returns "4.08+6i"

Examples

IMPRODUCT(1.2;"1i") returns "+1.2i"

8.1.5.37 IMREAL

The IMREAL(string) returns the real coefficient of a complex.

Return type: Double

Syntax

IMREAL(string)

Parameters

Comment: Complex number, *Type:* Text

Examples

IMREAL("1.2+3.4i") returns 1.2

Examples

IMREAL("1.2i") returns 0

8.1.5.38 IMSEC

The IMSEC(string) returns the secant of a complex number.

Return type: Text

Syntax

IMSEC(string)

Parameters

Comment: Complex number, *Type:* Text

Examples

IMSEC("1+i") returns "0.49833+i0.59108"

8.1.5.39 IMSECH

The IMSECH(string) returns the hyperbolic secant of a complex number.

Return type: Text

Syntax

IMSECH(string)

Parameters

Comment: Complex number, *Type:* Text

Examples

IMSECH("1+i") returns "0.49833-i0.59108"

8.1.5.40 IMSIN

The IMSIN(string) function returns the sine of a complex number.

Return type: Text

Syntax

IMSIN(string)

Parameters

Comment: Complex number, *Type:* Text

Examples

IMSIN("1+i") returns "1.29846+0.634964i"

Examples

IMSIN("1.2") returns -0.536573

8.1.5.41 IMSINH

The IMSINH(string) function returns the hyperbolic sine of a complex number.

Return type: Text

Syntax

IMSINH(string)

Parameters

Comment: Complex number, *Type:* Text

Examples

IMSINH("1+i") returns "0.63496+1.29846i"

Examples

IMSINH("1.2") returns 1.50946

8.1.5.42 IMSQRT

The IMSQRT(string) returns the square root of a complex number.

Return type: Text

Syntax

IMSQRT(string)

Parameters

Comment: Complex number, *Type:* Text

Examples

IMSQRT("1+i") returns "1.09868+0.45509i"

Examples

IMSQRT("1.2i") returns "0.774597+0.774597i"

8.1.5.43 IMSUB

The IMSUB() returns the difference of several complex numbers of form x+yi.

Return type: Text

Syntax

IMSUB(value;value;...)

Parameters

Comment: Complex number, *Type:* A range of strings

Comment: Complex number, *Type:* A range of strings

Comment: Complex number, *Type:* A range of strings

Comment: Complex number, *Type:* A range of strings

Comment: Complex number, *Type:* A range of strings

Examples

IMSUB(1.2;"3.4+5i") returns "-2.2-5i"

Examples

IMSUB(1.2;"1i") returns "1.2-i"

8.1.5.44 IMSUM

The IMSUM() returns the sum of several complex numbers of form x+yi.

Return type: Text

Syntax

IMSUM(value;value;...)

Parameters

Comment: Complex number, *Type:* A range of strings

Comment: Complex number, *Type:* A range of strings

Comment: Complex number, *Type:* A range of strings

Comment: Complex number, *Type:* A range of strings

Comment: Complex number, *Type:* A range of strings

Examples

IMSUM(1.2;"3.4+5i") returns "4.6+5i"

Examples

IMSUM(1.2;"1i") returns "1.2+i"

8.1.5.45 IMTAN

The IMTAN(string) function returns the tangent of a complex number.

Return type: Text

Syntax

IMTAN(string)

Parameters

Comment: Complex number, *Type:* Text

Examples

IMTAN("1+i") returns "0.27175+1.08392i"

Examples

IMTAN("1.2") returns 2.57215

8.1.5.46 IMTANH

The IMTANH(string) function returns the hyperbolic tangent of a complex number.

Return type: Text

Syntax

IMTANH(string)

Parameters

Comment: Complex number, *Type:* Text

Examples

IMTANH("1+i") returns "1.08392+0.27175i"

Examples

IMTANH("1.2") returns 0.83365

8.1.5.47 OCT2BIN

The OCT2BIN() function returns the value formatted as a binary number.

Return type: Text

Syntax

OCT2BIN(value)

Parameters

Comment: The value to convert, *Type:* Text

Comment: The minimum length of the output, *Type:* Whole number (like 1, 132, 2344)

Examples

OCT2BIN("12") returns "1010"

Examples

OCT2BIN("55") returns "101101"

8.1.5.48 OCT2DEC

The OCT2DEC() function returns the value formatted as a decimal number.

Return type: Whole number (like 1, 132, 2344)

Syntax

OCT2DEC(value)

Parameters

Comment: The value to convert, *Type:* Text

Examples

OCT2DEC("12") returns 10

Examples

OCT2DEC("55") returns 45

8.1.5.49 OCT2HEX

The OCT2HEX() function returns the value formatted as a hexadecimal number.

Return type: Text

Syntax

OCT2HEX(value)

Parameters

Comment: The value to convert, *Type:* Text

Comment: The minimum length of the output, *Type:* Whole number (like 1, 132, 2344)

Examples

OCT2HEX("12") returns "A"

Examples

OCT2HEX("55") returns "2D"

8.1.6 Financial

8.1.6.1 ACCRINT

The ACCRINT function returns accrued interest for a security which pays periodic interest. Allowed frequencies are 1 - annual, 2 - semi-annual or 4 - quarterly. Basis is the type of day counting you want to use: 0: US 30/360 (default), 1: real days, 2: real days/360, 3: real days/365 or 4: European 30/365.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

ACCRINT(issue; first interest; settlement; rate; par; frequency; basis)

Parameters

Comment: Issue date, *Type:* Date

Comment: First interest, *Type:* Date

Comment: Settlement, *Type:* Date

Comment: Annual rate of security, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Par value, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Number of payments per year, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Day counting basis, *Type:* Whole number (like 1, 132, 2344)

Examples

ACCRINT("2/28/2001"; "8/31/2001"; "5/1/2001"; 0.1; 1000; 2; 0) returns 16,944

Related Functions

[ACCRINTM](#)

8.1.6.2 ACCRINTM

The ACCRINTM function returns accrued interest for a security which pays interests at maturity date. Basis is the type of day counting you want to use: 0: US 30/360 (default), 1: real days, 2: real days/360, 3: real days/365 or 4: European 30/365.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

ACCRINTM(issue; settlement; rate; par; basis)

Parameters

Comment: Issue date, *Type:* Date

Comment: Settlement, *Type:* Date

Comment: Annual rate of security, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Par value, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Day counting basis, *Type:* Whole number (like 1, 132, 2344)

Examples

ACCRINTM("2/28/2001"; "8/31/2001"; 0.1; 100) returns 5.0278

Related Functions

[ACCRINT](#)

8.1.6.3 AMORDEGRC

The AMORDEGRC function calculates the amortization value for the French accounting system using degressive depreciation.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

AMORDEGRC(Cost; purchaseDate; firstPeriodEndDate; salvage; period; rate; basis)

Parameters

Comment: Cost, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Pv, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Fv, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

AMORDEGRC(1000; "2006-02-01"; "2006-12-31"; 10; 0; 0.1; 1) returns 228

Related Functions

[AMORLINC](#)
[DB](#)
[DDB](#)
[YEARFRAC](#)

8.1.6.4 AMORLINC

The AMORLINC function calculates the amortization value for the French accounting system using linear depreciation.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

AMORLINC(Cost; purchaseDate; firstPeriodEndDate; salvage; period; rate; basis)

Parameters

Comment: P, *Type:* Whole number (like 1, 132, 2344)

Comment: Pv, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Fv, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

AMORLINC(1000; "2004-02-01"; "2004-12-31"; 10; 0; 0.1; 1) returns 91.256831

Related Functions

[AMORDEGRC](#)
[DB](#)
[DDB](#)
[YEARFRAC](#)

8.1.6.5 COMPOUND

The COMPOUND() function returns the value of an investment, given the principal, nominal interest rate, compounding frequency and time. For example: \$5000 at 12% interest compounded quarterly for 5 years will become COMPOUND(5000;0.12;4;5) or \$9030.56.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

COMPOUND(initial;interest;periods;periods_per_year)

Parameters

Comment: Principal, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Interest rate, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Periods per year, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Years, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

COMPOUND(5000;0.12;4;5) equals 9030.56

8.1.6.6 CONTINUOUS

The CONTINUOUS() function calculates the return on continuously compounded interest, given the principal, nominal rate and time in years. For example: \$1000 earning 10% for 1 year becomes CONTINUOUS(1000;1;1) or \$1105.17.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

CONTINUOUS(principal;interest;years)

Parameters

Comment: Principal, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Interest rate, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Years, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

CONTINUOUS(1000;0.1;1) equals 1105.17

8.1.6.7 COUPNUM

The COUPNUM function returns the number of coupons to be paid between the settlement and the maturity. Basis is the type of day counting you want to use: 0: US 30/360 (default), 1: real days, 2: real days/360, 3: real days/365 or 4: European 30/365.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

COUPNUM(settlement; maturity; frequency; basis)

Parameters

Comment: Settlement, *Type:* Date

Comment: Maturity, *Type:* Date

Comment: Frequency, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Day counting basis, *Type:* Whole number (like 1, 132, 2344)

Examples

COUPNUM("2/28/2001"; "8/31/2001"; 2; 0) returns 1

8.1.6.8 CUMIPMT

Calculates the cumulative interest payment.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

CUMIPMT(rate, periods, value, start, end, type)

Parameters

Comment: rate, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: periods, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: value, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: start, *Type:* Whole number (like 1, 132, 2344)

Comment: end, *Type:* Whole number (like 1, 132, 2344)

Comment: type, *Type:* Whole number (like 1, 132, 2344)

Examples

CUMIPMT(0.06/12; 5*12; 100000; 5; 12; 0) equals -3562,187023

Related Functions

[IPMT](#)
[CUMPRINC](#)

8.1.6.9 CUMPRINC

Calculates the cumulative principal payment.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

CUMPRINC(rate, periods, value, start, end, type)

Parameters

Comment: rate, *Type:* A floating point value (like 1.3, 0.343, 253)
Comment: periods, *Type:* A floating point value (like 1.3, 0.343, 253)
Comment: value, *Type:* A floating point value (like 1.3, 0.343, 253)
Comment: start, *Type:* Whole number (like 1, 132, 2344)
Comment: end, *Type:* Whole number (like 1, 132, 2344)
Comment: type, *Type:* Whole number (like 1, 132, 2344)

Examples

CUMPRINC(0.06/12; 5*12; 100000; 5; 12; 0) equals -11904.054201

Related Functions

[PPMT](#)
[CUMIPMT](#)

8.1.6.10 DB

The DB() function will calculate the depreciation of an asset for a given period using the fixed-declining balance method. Month is optional, if omitted it is assumed to be 12.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

DB(cost; salvage value; life; period [;month])

Parameters

Comment: Cost, *Type:* A floating point value (like 1.3, 0.343, 253)
Comment: Salvage, *Type:* A floating point value (like 1.3, 0.343, 253)
Comment: Life, *Type:* A floating point value (like 1.3, 0.343, 253)
Comment: Period, *Type:* A floating point value (like 1.3, 0.343, 253)
Comment: Month, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

DB(8000;400;6;3) equals 1158.40

Examples

DB(8000;400;6;3;2) equals 1783.41

Related Functions

[DDB](#)
[SLN](#)

8.1.6.11 DDB

The DDB() function calculates the depreciation of an asset for a given period using the arithmetic-declining method. The factor is optional, if omitted it is assumed to be 2. All the parameter must be greater than zero.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

DDB(cost; salvage value; life; period [;factor])

Parameters

Comment: Cost, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Salvage, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Life, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Period, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Factor, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

DDB(75000;1;60;12;2) returns 1721.81

Related Functions

[SLN](#)

8.1.6.12 DISC

The DISC function returns the discount rate for a security. Basis is the type of day counting you want to use: 0: US 30/360 (default), 1: real days, 2: real days/360, 3: real days/365 or 4: European 30/365.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

DISC(settlement; maturity; par; redemption [; basis])

Parameters

Comment: Settlement, *Type:* Date

Comment: Maturity, *Type:* Date

Comment: Price per \$100 face value, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Redemption, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Day counting basis, *Type:* Whole number (like 1, 132, 2344)

Examples

DISC("2/28/2001"; "8/31/2001"; 12; 14) returns 0.2841

Related Functions

[YEARFRAC](#)

8.1.6.13 DOLLARDE

The DOLLARDE() function returns a dollar price expressed as a decimal number. The fractional dollar is the number to be converted and the fraction is the denominator of the fraction

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

DOLLARDE(fractional dollar; fraction)

Parameters

Comment: Fractional Dollar, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Fraction, *Type:* Whole number (like 1, 132, 2344)

Examples

DOLLARDE(1.02; 16) - stands for 1 and 2/16 - returns 1.125

Related Functions

[DOLLARFR](#)
[TRUNC](#)

8.1.6.14 DOLLARFR

The DOLLARFR() function returns a dollar price expressed as a fraction. The decimal dollar is the number to be converted and the fraction is the denominator of the fraction

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

DOLLARFR(fractional dollar; fraction)

Parameters

Comment: Decimal Dollar, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Fraction, *Type:* Whole number (like 1, 132, 2344)

Examples

DOLLARFR(1.125; 16) returns 1.02. (1 + 2/16)

Related Functions

[DOLLARDE](#)
[TRUNC](#)

8.1.6.15 DURATION

Returns the number of periods needed for an investment to retain a desired value.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

DURATION(rate; pv; fv)

Parameters

Comment: Rate, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Present value (PV), *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Future value (FV), *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

DURATION(0.1; 1000; 2000) returns 7.27

Related Functions

[FV](#)
[PV](#)

8.1.6.16 DURATION_ADD

Returns the Macauley duration of a fixed interest security in years.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

DURATION_ADD(Settlement; Maturity; Coupon; Yield; Frequency; Basis)

Parameters

Comment: Settlement, *Type:* Date

Comment: Maturity, *Type:* Date

Comment: Coupon, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Yield, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Frequency, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Basis, *Type:* Whole number (like 1, 132, 2344)

Examples

DURATION_ADD("1998-01-01"; "2006-01-01"; 0.08; 0.09; 2; 1) returns 5.9937749555

Related Functions

[MDURATION](#)

8.1.6.17 EFFECT

The EFFECT() function calculates the effective yield for a nominal interest rate (annual rate or APR). For example: 8% interest compounded monthly provides an effective yield of EFFECT(.08;12) or 8.3%.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

EFFECT(nominal;periods)

Parameters

Comment: Nominal interest rate, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Periods, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

EFFECT(0.08;12) equals 0.083

Related Functions

[EFFECTIVE](#)

[NOMINAL](#)

8.1.6.18 EFFECTIVE

The EFFECTIVE() function calculates the effective yield for a nominal interest rate (annual rate or APR). It is the same as the EFFECT function.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

EFFECTIVE(nominal;periods)

Parameters

Comment: Nominal interest rate, *Type:* A floating point value (like 1.3, 0.343, 253)
Comment: Periods, *Type:* A floating point value (like 1.3, 0.343, 253)

Related Functions

[EFFECT](#)

8.1.6.19 EURO

The EURO() function converts one Euro to a given national currency in the European monetary union. Currency is one of the following: ATS (Austria), BEF (Belgium), DEM (Germany), ESP (Spain), EUR (Euro), FIM (Finland), FRF (France), GRD (Greece), IEP (Ireland), ITL (Italy), LUF (Luxembourg), NLG (Netherlands), or PTE (Portugal).

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

EURO(currency)

Parameters

Comment: Currency, *Type:* Text

Examples

EURO("DEM") equals 1.95583

Related Functions

[EUROCONVERT](#)

8.1.6.20 EUROCONVERT

The EUROCONVERT() function converts a number from one national currency to another currency in the European monetary union by using EURO an intermediary. Currency is one of the following: ATS (Austria), BEF (Belgium), DEM (Germany), ESP (Spain), EUR (Euro), FIM (Finland), FRF (France), GRD (Greece), IEP (Ireland), ITL (Italy), LUF (Luxembourg), NLG (Netherlands), or PTE (Portugal).

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

EUROCONVERT(number; source currency; target currency)

Parameters

Comment: Number, *Type:* A floating point value (like 1.3, 0.343, 253)
Comment: Source currency, *Type:* Text
Comment: Target currency, *Type:* Text

Examples

EUROCONVERT(1; "EUR"; "DEM") equals 1.95583

Related Functions

[EURO](#)

8.1.6.21 FV

The FV() function returns the future value of an investment, given the yield and the time elapsed. If you have \$1000 in a bank account earning 8% interest, after two years you will have FV(1000;0.08;2) or \$1166.40.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

FV(present value;yield;periods)

Parameters

Comment: Present value, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Rate, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Periods, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

FV(1000;0.08;2) equals 1166.40

Related Functions

[PV](#)
[NPER](#)
[PMT](#)
[RATE](#)

8.1.6.22 FV_ANNUIITY

The FV_ANNUIITY() function returns the future value of a stream of payments given the amount of the payment, the interest rate and the number of periods. For example: If you receive \$500 per year for 20 years, and invest it at 8%, the total after 20 years will be FV_annuity(500;0.08;20) or \$22,880.98. This function assumes that payments are made at the end of each period.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

FV_ANNUIITY(amount;interest;periods)

Parameters

Comment: Payment per period, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Interest rate, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Periods, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

FV_ANNUIITY(1000;0.05;5) equals 5525.63

8.1.6.23 INTRATE

The INTRATE function returns the interest rate for a fully invested security. Basis is the type of day counting you want to use: 0: US 30/360 (default), 1: real days, 2: real days/360, 3: real days/365 or 4: European 30/365.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

INTRATE(settlement; maturity; investment; redemption; basis)

Parameters

Comment: Settlement, *Type:* Date

Comment: Maturity, *Type:* Date

Comment: Investment, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Redemption, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Day counting basis, *Type:* Whole number (like 1, 132, 2344)

Examples

INTRATE("2/28/2001"; "8/31/2001"; 1000000; 2000000; 1) returns 1.98

8.1.6.24 IPMT

IPMT calculates the amount of a payment of an annuity going towards interest.

Rate is the periodic interest rate.

Period is the amortization period. 1 for the first and NPER for the last period.

NPER is the total number of periods during which annuity is paid.

PV is the present value in the sequence of payments.

FV (optional) is the desired (future) value. default: 0.

Type (optional) defines the due date. 1 for payment at the beginning of a period and 0 (default) for payment at the end of a period.

The example shows the interest to pay in the last year of a three year loan. The interest rate is 10 percent.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

IPMT(Rate; Period; NPer; PV; FV; Type)

Parameters

Comment: Rate, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Period, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Number of periods, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Present values, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Future value (optional), *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Type (optional), *Type:* Whole number (like 1, 132, 2344)

Examples

IPMT(0.1;3;3;8000) equals -292.45

Related Functions

[PPMT](#)

[PV](#)

[PMT](#)

8.1.6.25 IRR

The IRR function calculates the internal rate of return for a series of cash flows.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

IRR(Values[; Guess = 0.1])

Parameters

Comment: Values, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Guess, *Type:* A floating point value (like 1.3, 0.343, 253)

Related Functions

XIRR

8.1.6.26 ISPMT

Calculates the interest paid on a given period of an investment.

Rate is the periodic interest rate.

Period is the amortization period. 1 for the first and NPer for the last period.

NPer is the total number of periods during which annuity is paid.

PV is the present value in the sequence of payments.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

ISPMT(Rate; Period; NPer; PV)

Parameters

Comment: Rate, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Period, *Type:* Whole number (like 1, 132, 2344)

Comment: Number of periods, *Type:* Whole number (like 1, 132, 2344)

Comment: Present values (PV), *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

ISPMT(0.1; 1; 3; 8000000) equals -533333

Related Functions

PV

FV

NPER

PMT

RATE

8.1.6.27 LEVEL_COUPON

The LEVEL_COUPON() function calculates the value of a level-coupon bond. For example: if the interest rate is 10%, a \$1000 bond with semi-annual coupons at a rate of 13% that matures in 4 years is worth LEVEL_COUPON(1000;.13;2;4;.1) or \$1096.95.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

LEVEL_COUPON(face value;coupon rate;coupons per year;years;market rate)

Parameters

Comment: Face value, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Coupon rate, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Coupons per year, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Years, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Market interest rate, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

LEVEL_COUPON(1000;.13;2;4;.1) equals 1096.95

8.1.6.28 MDURATION

The MDURATION() function will calculate the modified Macauley duration of a fixed interest security in years.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

MDURATION(Settlement; Maturity; Coupon; Yield; Frequency; [Basis=0])

Parameters

Comment: Settlement, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Maturity, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Coupon, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Yield, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Frequency, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Basis, *Type:* Whole number (like 1, 132, 2344)

Examples

MDURATION("2004-02-01"; "2004-05-31"; 0.08; 0.09; 2; 0) returns 0.316321106

Related Functions

[DURATION](#)

8.1.6.29 MIRR

The MIRR() function will calculate the modified internal rate of return (IRR) of a series of periodic investments.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

MIRR(values; investment; reinvestment)

Parameters

Comment: Values, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Investment, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Reinvestment, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

MIRR({100;200;-50;300;-200}, 5%, 6%) equals 34.2823387842%

Related Functions

[IRR](#)

8.1.6.30 NOMINAL

The NOMINAL() function calculates the nominal (stated) interest rate for an effective (annualized) interest rate compounded at given intervals. For example: to earn 8% on an account compounded monthly, you need a return of NOMINAL(.08;12) or 7.72%.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

NOMINAL(effective;periods)

The Calligra Sheets Handbook

Parameters

Comment: Effective interest rate, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Periods, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

NOMINAL(0.08;12) equals 0.0772

Related Functions

[EFFECT](#)

8.1.6.31 NPER

Returns the number of periods of an investment.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

NPER(rate;payment;pv;fv;type)

Parameters

Comment: Rate, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Payment, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Present value (PV), *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Future value (FV - optional), *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Type (optional), *Type:* Whole number (like 1, 132, 2344)

Examples

NPER(0.1; -100; 1000) equals 11

Examples

NPER(0.06; 0; -10000; 20000 ;0) returns 11.906

Related Functions

[FV](#)

[RATE](#)

[PMT](#)

[PV](#)

8.1.6.32 NPV

The net present value (NPV) for a series of periodic cash flows.

Computes the net present value for a series of periodic cash flows with the discount rate Rate. Values should be positive if they are received as income, and negative if the amounts are expenditure.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

NPV(Rate; Values)

Parameters

Comment: Rate, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Values (array), *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

NPV(100%;4;5;7) = 4.125

Related Functions

FV
IRR
NPER
PMT
PV

8.1.6.33 ODDLPRICE

The ODDLPRICE function calculates the value of the security per 100 currency units of face value. The security has an irregular last interest date.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

ODDLPRICE(Settlement; Maturity; Last; Rate; AnnualYield; Redemption; Frequency [; Basis = 0])

Parameters

Comment: Settlement, *Type:* Date

Comment: Maturity, *Type:* Date

Comment: Last, *Type:* Date

Comment: Rate, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: AnnualYield, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Redemption, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Frequency, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Basis, *Type:* Whole number (like 1, 132, 2344)

Examples

ODDLPRICE(DATE(1990;6;1);DATE(1995;12;31);DATE(1990;1;1);3%;5%;100;2) returns
90.991042345

8.1.6.34 ODDLYIELD

The ODDLYIELD function calculates the yield of the security which has an irregular last interest date.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

ODDLYIELD(Settlement; Maturity; Last; Rate; Price; Redemption; Frequency [; Basis = 0])

Parameters

Comment: Settlement, *Type:* Date

Comment: Maturity, *Type:* Date

Comment: Last, *Type:* Date

Comment: Rate, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Price, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Redemption, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Frequency, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Basis, *Type:* Whole number (like 1, 132, 2344)

Examples

ODDLYIELD(1990;6;1;1995;12;31;1990;1;1;3%;91;100;2) returns
4.997775351

Related Functions

[ODDLPRICE](#)

8.1.6.35 PMT

PMT returns the amount of payment for a loan based on a constant interest rate and constant payments (each payment is equal amount).

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

PMT(rate; nper ; pv [; fv = 0 [; type = 0]])

Parameters

Comment: Rate, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Number of periods (NPer), *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Present value (PV), *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Future value (FV - optional), *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Type (optional), *Type:* Whole number (like 1, 132, 2344)

Examples

PMT(0.1; 4; 10000) equals -3154.71

Related Functions

[NPER](#)

[IPMT](#)

[PPMT](#)

[PV](#)

8.1.6.36 PPMT

PPMT calculates the amount of a payment of an annuity going towards principal.

Rate is the periodic interest rate.

Period is the amortization period. 1 for the first and NPER for the last period.

NPER is the total number of periods during which annuity is paid.

PV is the present value in the sequence of payments.

FV (optional) is the desired (future) value. default: 0.

Type (optional) defines the due date. 1 for payment at the beginning of a period and 0 (default) for payment at the end of a period.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

PPMT(Rate; Period; NPer; PV [; FV = 0 [; Type = 0]])

Parameters

Comment: Rate, *Type:* A floating point value (like 1.3, 0.343, 253)
Comment: Period, *Type:* A floating point value (like 1.3, 0.343, 253)
Comment: Number of periods, *Type:* A floating point value (like 1.3, 0.343, 253)
Comment: Present value, *Type:* A floating point value (like 1.3, 0.343, 253)
Comment: Future value (optional), *Type:* A floating point value (like 1.3, 0.343, 253)
Comment: Type (optional), *Type:* Whole number (like 1, 132, 2344)

Examples

PPMT(0.0875;1;36;5000;8000;1) equals -18.48

Related Functions

IPMT
PMT
PV

8.1.6.37 PRICEMAT

PRICEMAT Calculate the price per 100 currency units of face value of the security that pays interest on the maturity date.

Basis Calculation method

0 US method, 12 months, each month with 30 days

1 Actual number of days in year, actual number of days in months

2 360 days in a year, actual number of days in months

4 365 days in a year, actual number of days in months

5 European method, 12 months, each month has 30 days

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

PRICEMAT(settlement; maturity; issue; rate; yield [; basis = 0])

Parameters

Comment: Settlement, *Type:* Date
Comment: Maturity, *Type:* Date
Comment: Issue, *Type:* Date
Comment: Discount rate, *Type:* A floating point value (like 1.3, 0.343, 253)
Comment: Yield, *Type:* A floating point value (like 1.3, 0.343, 253)
Comment: Basis, *Type:* Whole number (like 1, 132, 2344)

Examples

PRICEMAT(DATE(1990;6;1); DATE(1995;12;31); DATE(1990;1;1); 6%; 5%) returns
103.819218241

8.1.6.38 PV

The PV() function returns the present value of an investment -- the value today of a sum of money in the future, given the rate of interest or inflation. For example if you need \$1166.40 for your new computer and you want to buy it in two years while earning 8% interest, you need to start with PV(1166.4;0.08;2) or \$1000.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

PV(future value;rate;periods)

Parameters

Comment: Future value, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Interest rate, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Periods, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

PV(1166.4;0.08;2) equals 1000

8.1.6.39 PV_ANNUITY

The PV_ANNUITY() function returns the present value of an annuity or stream of payments. For example: a "million dollar" lottery ticket that pays \$50,000 a year for 20 years, with an interest rate of 5%, is actually worth PV_ANNUITY(50000;0.05;20) or \$623,111. This function assumes that payments are made at the end of each period.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

PV_ANNUITY(amount;interest;periods)

Parameters

Comment: Payment per period, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Interest rate, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Periods, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

PV_ANNUITY(1000;0.05;5) equals 4329.48

8.1.6.40 RATE

The RATE() function computes the constant interest rate per period of an investment.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

RATE(nper;pmt;pv;fv;type;guess)

Parameters

Comment: Payment period, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Regular payments, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Present value, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Future value, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Type, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Guess, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

RATE(4*12;-200;8000) equals 0.007701472

8.1.6.41 RECEIVED

The RECEIVED function returns the amount received at the maturity date for a invested security. Basis is the type of day counting you want to use: 0: US 30/360 (default), 1: real days, 2: real days/360, 3: real days/365 or 4: European 30/365. The settlement date must be before maturity date.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

RECEIVED(settlement; maturity; investment; discount; basis)

Parameters

Comment: Settlement, *Type:* Date

Comment: Maturity, *Type:* Date

Comment: Investment, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Discount rate, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Basis, *Type:* Whole number (like 1, 132, 2344)

Examples

RECEIVED("2/28/2001"; "8/31/2001"; 1000; 0.05; 0) returns 1,025.787

8.1.6.42 RRI

The RRI function calculates the interest rate resulting from the profit (return) of an investment.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

RRI(P; Pv; Fv)

Parameters

Comment: P, *Type:* Whole number (like 1, 132, 2344)

Comment: Pv, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Fv, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

RRI(1;100;200) returns 1

Related Functions

[FV](#)
[NPER](#)
[PMT](#)
[PV](#)
[RATE](#)

8.1.6.43 SLN

The SLN() function will determine the straight line depreciation of an asset for a single period. Cost is the amount you paid for the asset. Salvage is the value of the asset at the end of the period. Life is the number of periods over which the asset is depreciated. SLN divides the cost evenly over the life of an asset.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

SLN(cost; salvage value; life)

Parameters

Comment: Cost, Type: A floating point value (like 1.3, 0.343, 253)

Comment: Salvage, Type: A floating point value (like 1.3, 0.343, 253)

Comment: Life, Type: A floating point value (like 1.3, 0.343, 253)

Examples

SLN(10000;700;10) equals 930

Related Functions

[SYD](#)

[DDB](#)

8.1.6.44 SYD

The SYD() function will calculate the sum-of-years digits depreciation for an asset based on its cost, salvage value, anticipated life, and a particular period. This method accelerates the rate of the depreciation, so that more depreciation expense occurs in earlier periods than in later ones. The depreciable cost is the actual cost minus the salvage value. The useful life is the number of periods (typically years) over which the asset is depreciated.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

SYD(cost; salvage value; life; period)

Parameters

Comment: Cost, Type: A floating point value (like 1.3, 0.343, 253)

Comment: Salvage, Type: A floating point value (like 1.3, 0.343, 253)

Comment: Life, Type: A floating point value (like 1.3, 0.343, 253)

Comment: Period, Type: A floating point value (like 1.3, 0.343, 253)

Examples

SYD(5000; 200; 5; 2) equals 1280

Related Functions

[SLN](#)

[DDB](#)

8.1.6.45 TBILLEQ

The TBILLEQ functions returns the bond equivalent for a treasury bill. The maturity date must be after the settlement date but within 365 days.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

TBILLEQ(settlement; maturity; discount)

Parameters

Comment: Settlement, Type: Date

Comment: Maturity, Type: Date

Comment: Discount rate, Type: A floating point value (like 1.3, 0.343, 253)

Examples

TBILLEQ("2/28/2001"; "8/31/2001"; 0.1) returns 0.1068

Related Functions

TBILLPRICE
TBILLYIELD

8.1.6.46 TBILLPRICE

The TBILLPRICE functions returns the price per \$100 value for a treasury bill. The maturity date must be after the settlement date but within 365 days. The discount rate must be positive.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

TBILLPRICE(settlement; maturity; discount)

Parameters

Comment: Settlement, *Type:* Date

Comment: Maturity, *Type:* Date

Comment: Discount rate, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

TBILLPRICE("2/28/2001"; "8/31/2001"; 0.05) returns 97.4444

Related Functions

TBILLEQ
TBILLYIELD

8.1.6.47 TBILLYIELD

The TBILLYIELD functions returns the yield for a treasury bill. The maturity date must be after the settlement date but within 365 days. The price must be positive.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

TBILLYIELD(settlement; maturity; price)

Parameters

Comment: Settlement, *Type:* Date

Comment: Maturity, *Type:* Date

Comment: Price per \$100 face value, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

TBILLYIELD("2/28/2001"; "8/31/2001"; 600) returns -1.63

Related Functions

TBILLEQ
TBILLPRICE

8.1.6.48 VDB

VDB calculates the depreciation allowance of an asset with an initial value, an expected useful life, and a final value of salvage for a period specified, using the variable-rate declining balance method.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

VDB(cost; salvage; life; start-period; end-period; [; depreciation-factor = 2 [; switch = false]])

Parameters

Comment: Settlement, *Type:* Date

Comment: Maturity, *Type:* Date

Comment: Price, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Redemption, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Basis, *Type:* Whole number (like 1, 132, 2344)

Examples

VDB(10000;600;10;0;0.875;1.5) returns 1312.5

8.1.6.49 XIRR

The XIRR function calculates the internal rate of return for a non-periodic series of cash flows.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

XIRR(Values; Dates[; Guess = 0.1])

Parameters

Comment: Values, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Dates, *Type:* Date

Comment: Guess, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

XIRR(B1:B4;C1:C4) Suppose B1:B4 contains -20000, 4000, 12000, 8000 while C1:C4 contains "=DATE(2000;1;1)", "=DATE(2000;6;1)", "=DATE(2000;12;30)", "=DATE(2001;3;1)" returns 0.2115964

Related Functions

[IRR](#)

8.1.6.50 XNPV

The XNPV function calculates the net present value of a series of cash flows.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

XNPV(Rate; Values; Dates)

Parameters

Comment: Rate, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Values, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Dates, *Type:* Date

Examples

XNPV(5%;B1:B4;C1:C4) suppose B1:B4 contains -20000, 4000, 12000, 8000 while C1:C4 contains "=DATE(2000;1;1)", "=DATE(2000;6;1)", "=DATE(2000;12;30)", "=DATE(2001;3;1)" returns 2907.83187

Related Functions

[NPV](#)

8.1.6.51 YIELDDISC

YIELDDISC calculates the yield of a discounted security per 100 currency units of face value.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

YIELDDISC(settlement; maturity; price, redemp, basis)

Parameters

Comment: Settlement, *Type:* Date

Comment: Maturity, *Type:* Date

Comment: Price, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Redemption, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Basis, *Type:* Whole number (like 1, 132, 2344)

Examples

YIELDDISC(DATE(1990;6;1);DATE(1990;12;31);941.66667;1000) returns 0.106194684

8.1.6.52 YIELDMAT

The YIELDMAT function calculates the yield of the security that pays interest on the maturity date.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

YIELDMAT(Settlement; Maturity; Issue; Rate; Price; Basis)

Parameters

Comment: Settlement, *Type:* Date

Comment: Maturity, *Type:* Date

Comment: Issue, *Type:* Date

Comment: Discount rate, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Price, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Basis, *Type:* Whole number (like 1, 132, 2344)

Examples

YIELDMAT(DATE(1990;6;1);DATE(1995;12;31);DATE(1990; 1; 1); 6%;103.819218241) returns 0.050000000

Related Functions

[YIELDDISC](#)

8.1.6.53 ZERO_COUPON

The ZERO_COUPON() function calculates the value of a zero-coupon (pure discount) bond. For example: if the interest rate is 10%, a \$1000 bond that matures in 20 years is worth ZERO_COUPON(1000;.1;20) or \$148.64.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

ZERO_COUPON(face value;rate;years)

Parameters

Comment: Face value, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Interest rate, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Years, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

ZERO_COUPON(1000;.1;20) equals 148.64

8.1.7 Information

8.1.7.1 ERRORTYPE

The ERRORTYPE() function converts an error to a number. If the value is not an error, an error is returned. Otherwise, a numerical code is returned. Error codes are modelled on Excel.

Return type: Whole number (like 1, 132, 2344)

Syntax

ERRORTYPE(value)

Parameters

Comment: Error, *Type:* Any kind of value

Examples

ERRORTYPE(NA()) returns 7

Examples

ERRORTYPE(0) returns an error

8.1.7.2 FILENAME

Returns the current filename. If the current document is not saved, an empty string is returned.

Return type: Text

Syntax

FILENAME()

Parameters

8.1.7.3 FORMULA

The FORMULA() function returns the formula of a cell as string.

Return type: Text

Syntax

FORMULA(x)

Parameters

Comment: Reference, *Type:* Reference

Examples

FORMULA(A1) returns "=SUM(1+2)" if the cell A1 contains such a formula.

8.1.7.4 INFO

The INFO() function returns information about the current operating environment. Parameter type specifies what type of information you want to return. It is one of the following: "directory" returns the path of the current directory, "numfile" returns the number of active documents, "release" returns the version of Calligra Sheets as text, "recalc" returns the current recalculation mode: "Automatic" or "Manual", "system" returns the name of the operating environment, "osversion" returns the current operating system.

Return type: Text

Syntax

INFO(type)

Parameters

Comment: Type of information, *Type:* Text

8.1.7.5 ISBLANK

The ISBLANK() function returns True if the parameter is empty. Otherwise it returns False.

Return type: A truth value (TRUE or FALSE)

Syntax

ISBLANK(x)

Parameters

Comment: Any value, *Type:* Any kind of value

Examples

ISBLANK(A1) returns True if A1 is empty

Examples

ISBLANK(A1) returns False if A1 holds a value

8.1.7.6 ISDATE

The ISDATE() function returns True if the parameter is a date value. Otherwise it returns False

Return type: A truth value (TRUE or FALSE)

Syntax

ISDATE(x)

Parameters

Comment: Any value, *Type:* Any kind of value

Examples

ISDATE("2000-2-2") returns True

Examples

ISDATE("hello") returns False

8.1.7.7 ISERR

The ISERR() function returns True if its parameter is an error other than N/A. Otherwise, it returns False. Use ISERROR() if you want to include the N/A error as well.

Return type: A truth value (TRUE or FALSE)

Syntax

ISERR(x)

Parameters

Comment: Any value, *Type:* Any kind of value

Related Functions

[ISERROR](#)
[ISNA](#)

8.1.7.8 ISERROR

The ISERROR() function returns True if its parameter is an error of any type. Otherwise, it returns False.

Return type: A truth value (TRUE or FALSE)

Syntax

ISERROR(x)

Parameters

Comment: Any value, *Type:* Any kind of value

Related Functions

[ISERR](#)
[ISNA](#)

8.1.7.9 ISEVEN

The ISEVEN() function returns True if the number is even. Otherwise returns False.

Return type: A truth value (TRUE or FALSE)

Syntax

ISEVEN(x)

Parameters

Comment: Any value, *Type:* Any kind of value

Examples

ISEVEN(12) returns True

Examples

ISEVEN(-7) returns False

8.1.7.10 ISFORMULA

The ISFORMULA() function returns True if the referenced cell contains a formula. Otherwise it returns False

Return type: A truth value (TRUE or FALSE)

Syntax

ISFORMULA(x)

Parameters

Comment: Reference, *Type:* Reference

8.1.7.11 ISLOGICAL

The ISLOGICAL() function returns True if the parameter is a boolean value. Otherwise it returns False.

Return type: A truth value (TRUE or FALSE)

Syntax

ISLOGICAL(x)

Parameters

Comment: Any value, *Type:* Any kind of value

Examples

ISLOGICAL(A1>A2) returns True

Examples

ISLOGICAL(12) returns False

8.1.7.12 ISNA

The ISNA() function returns True if its parameter is a N/A error. In all other cases, it returns False.

Return type: A truth value (TRUE or FALSE)

Syntax

ISNA(x)

Parameters

Comment: Any value, *Type:* Any kind of value

Related Functions

[ISERR](#)

[ISERROR](#)

8.1.7.13 ISNONTEXT

The ISNONTEXT() function returns True if the parameter is not a string. Otherwise it returns False. It's the same as ISNOTTEXT.

Return type: A truth value (TRUE or FALSE)

Syntax

ISNONTEXT(x)

Parameters

Comment: Any value, *Type:* Any kind of value

Examples

ISNONTEXT(12) returns True

Examples

ISNONTEXT("hello") returns False

Related Functions

[ISNOTTEXT](#)

8.1.7.14 ISNOTTEXT

The ISNOTTEXT() function returns True if the parameter is not a string. Otherwise it returns False. It's the same as ISNONTEXT.

Return type: A truth value (TRUE or FALSE)

Syntax

ISNOTTEXT(x)

Parameters

Comment: Any value, *Type:* Any kind of value

Examples

ISNOTTEXT(12) returns True

Examples

ISNOTTEXT("hello") returns False

Related Functions

[ISNONTEXT](#)

8.1.7.15 ISNUM

The ISNUM() function returns True if the parameter is a numerical value. Otherwise it returns False. It's the same as ISNUMBER.

Return type: A truth value (TRUE or FALSE)

Syntax

ISNUM(x)

Parameters

Comment: Any value, *Type:* Any kind of value

Examples

ISNUM(12) returns True

Examples

ISNUM(hello) returns False

Related Functions

[ISNUMBER](#)

8.1.7.16 ISNUMBER

The ISNUMBER() function returns True if the parameter is a numerical value. Otherwise it returns False. It's the same as ISNUM.

Return type: A truth value (TRUE or FALSE)

Syntax

ISNUMBER(x)

Parameters

Comment: Any value, *Type:* Any kind of value

Examples

ISNUMBER(12) returns True

Examples

ISNUMBER(hello) returns False

Related Functions

[ISNUM](#)

8.1.7.17 ISODD

The ISODD() function returns True if the number is odd. Otherwise returns False.

Return type: A truth value (TRUE or FALSE)

Syntax

ISODD(x)

Parameters

Comment: Any value, *Type:* Any kind of value

Examples

ISODD(12) returns False

Examples

ISODD(-7) returns True

8.1.7.18 ISREF

The ISREF() function returns True if the parameter refers to a reference. Otherwise it returns False

Return type: A truth value (TRUE or FALSE)

Syntax

ISREF(x)

Parameters

Comment: Any value, *Type:* Any kind of value

Examples

ISREF(A12) returns true

Examples

ISREF("hello") returns false

8.1.7.19 ISTEEXT

The ISTEEXT() function returns True if the parameter is a string. Otherwise it returns False

Return type: A truth value (TRUE or FALSE)

Syntax

ISTEEXT(x)

Parameters

Comment: Any value, *Type:* Any kind of value

Examples

ISTEEXT(12) returns False

Examples

ISTEEXT("hello") returns True

8.1.7.20 ISTIME

The ISTIME() function returns True if the parameter is a time value. Otherwise it returns False.

Return type: A truth value (TRUE or FALSE)

Syntax

ISTIME(x)

Parameters

Comment: Any value, *Type:* Any kind of value

Examples

ISTIME("12:05") returns True

Examples

ISTIME("hello") returns False

8.1.7.21 N

The N() function converts a value to a number. If value is or refers to a number, this function returns the number. If value is True, this function returns 1. If a value is a date, this function returns the serial number of that date. Anything else will cause the function to return 0.

Return type: Whole number (like 1, 132, 2344)

Syntax

N(value)

Parameters

Comment: Value, *Type:* Any kind of value

Examples

N(3.14) returns 3.14

Examples

N("7") returns 0 (because "7" is text)

8.1.7.22 NA

The NA() function returns the constant error value, N/A.

Return type: Error

Syntax

NA()

Parameters

Related Functions

[ISNA](#)

[ISERR](#)

[ISERROR](#)

8.1.7.23 TYPE

The TYPE() function returns 1 if the value is a number, 2 if it is text, 4 if the value is a logical value, 16 if it is an error value or 64 if the value is an array. If the cell the value represents contains a formula you get its return type.

Return type: Whole number (like 1, 132, 2344)

Syntax

TYPE(x)

Parameters

Comment: Any value, *Type:* Any kind of value

Examples

TYPE(A1) returns 2, if A1 contains "Text"

Examples

TYPE(-7) returns 1

Examples

TYPE(A2) returns 1, if A2 contains "=CURRENTDATE()"

8.1.8 Logical

8.1.8.1 AND

The AND() function returns True if all the values are true. Otherwise it returns False (unless any of the values in an error - then it returns an error).

Return type: A truth value (TRUE or FALSE)

Syntax

AND(value;value;...)

Parameters

Comment: Boolean values, *Type:* A range of truth values (TRUE or FALSE)

Comment: Boolean values, *Type:* A range of truth values (TRUE or FALSE)

Comment: Boolean values, *Type:* A range of truth values (TRUE or FALSE)

Comment: Boolean values, *Type:* A range of truth values (TRUE or FALSE)

Comment: Boolean values, *Type:* A range of truth values (TRUE or FALSE)

Examples

AND(true;true;true) returns True

Examples

AND(true;false) returns False

8.1.8.2 FALSE

The FALSE() function returns the boolean value FALSE.

Return type: A truth value (TRUE or FALSE)

Syntax

FALSE()

Parameters

Examples

FALSE() returns FALSE

8.1.8.3 IF

The IF() function is a conditional function. This function returns the second parameter if the condition is True. Otherwise it returns the third parameter (which defaults to being false).

Return type: Any kind of value

Syntax

IF(condition;if_true;if_false)

Parameters

Comment: Condition, *Type:* A truth value (TRUE or FALSE)

Comment: If true, *Type:* Any kind of value

Comment: If false, *Type:* Any kind of value

Examples

A1=4;A2=6;IF(A1>A2;5;3) returns 3

8.1.8.4 IFERROR

Return X unless it is an Error, in which case return an alternative value.

Return type: Any kind of value

Syntax

IFERROR(AnyX;AnyAlternative)

Parameters

Comment: Any X, *Type:* Any kind of value

Comment: Any Alternative, *Type:* Any kind of value

Examples

IFERROR(A1;A2) returns the content of A1 if that content is not an error-value else the content of A2 is returned.

8.1.8.5 IFNA

Return X unless it is an NA, in which case return an alternative value.

Return type: Any kind of value

Syntax

IFNA(AnyX;AnyAlternative)

Parameters

Comment: Any X, *Type:* Any kind of value

Comment: Any Alternative, *Type:* Any kind of value

Examples

IFNA(A1;A2) returns the content of A1 if that content is not an #N/A error-value else the content of A2 is returned.

8.1.8.6 NAND

The NAND() function returns True if at least one value is not true. Otherwise it returns False.

Return type: A truth value (TRUE or FALSE)

Syntax

NAND(value;value;...)

Parameters

Comment: Boolean values, *Type:* A range of truth values (TRUE or FALSE)

Comment: Boolean values, *Type:* A range of truth values (TRUE or FALSE)

Comment: Boolean values, *Type:* A range of truth values (TRUE or FALSE)

Comment: Boolean values, *Type:* A range of truth values (TRUE or FALSE)

Comment: Boolean values, *Type:* A range of truth values (TRUE or FALSE)

Examples

NAND(true;false;false) returns True

Examples

NAND(true>true) returns False

8.1.8.7 NOR

The NOR() function returns True if all the values given as parameters are of boolean type and have the value false. Otherwise it returns False.

Return type: A truth value (TRUE or FALSE)

Syntax

NOR(value;value;...)

Parameters

Comment: Boolean values, *Type:* A range of truth values (TRUE or FALSE)

Comment: Boolean values, *Type:* A range of truth values (TRUE or FALSE)

Comment: Boolean values, *Type:* A range of truth values (TRUE or FALSE)

Comment: Boolean values, *Type:* A range of truth values (TRUE or FALSE)

Comment: Boolean values, *Type:* A range of truth values (TRUE or FALSE)

Examples

NOR(true;false;false) returns False

Examples

NOR(false;false) returns True

8.1.8.8 NOT

The NOT() function returns True if the value is False and returns False if the value is True. It returns an error if the input is an error.

Return type: A truth value (TRUE or FALSE)

Syntax

NOT(bool)

Parameters

Comment: Boolean value, *Type:* A truth value (TRUE or FALSE)

Examples

NOT(false) returns True

Examples

NOT(true) returns False

8.1.8.9 OR

The OR() function returns True if at least one of the values is true. Otherwise it returns False (unless any of the values is an error, then it returns an error).

Return type: A truth value (TRUE or FALSE)

Syntax

OR(value;value;...)

Parameters

Comment: Boolean values, *Type:* A range of truth values (TRUE or FALSE)
Comment: Boolean values, *Type:* A range of truth values (TRUE or FALSE)
Comment: Boolean values, *Type:* A range of truth values (TRUE or FALSE)
Comment: Boolean values, *Type:* A range of truth values (TRUE or FALSE)
Comment: Boolean values, *Type:* A range of truth values (TRUE or FALSE)

Examples

OR(false;false;false) returns False

Examples

OR(true;false) returns True

8.1.8.10 TRUE

The TRUE() function returns the boolean value TRUE.

Return type: A truth value (TRUE or FALSE)

Syntax

TRUE()

Parameters

Examples

TRUE() returns TRUE

8.1.8.11 XOR

The XOR() function returns False if the number of True values is even. Otherwise it returns True. It returns an error if any argument is an error.

Return type: A truth value (TRUE or FALSE)

Syntax

XOR(value;value;...)

Parameters

Comment: Boolean values, *Type:* A range of truth values (TRUE or FALSE)
Comment: Boolean values, *Type:* A range of truth values (TRUE or FALSE)
Comment: Boolean values, *Type:* A range of truth values (TRUE or FALSE)
Comment: Boolean values, *Type:* A range of truth values (TRUE or FALSE)
Comment: Boolean values, *Type:* A range of truth values (TRUE or FALSE)

Examples

XOR(false;false;false) returns True

Examples

XOR(true;false) returns True

8.1.9 Lookup & Reference

8.1.9.1 ADDRESS

The ADDRESS creates a cell address. Parameter Row is the row number and Column is the column number.

Absolute number specifies the type of reference: 1 or omitted = Absolute, 2 = Absolute row, relative column, 3 = Relative row; absolute column and 4 = Relative.

A1 Style specifies the style of the address to return. If A1 is set to TRUE (default) the address is returned in A1 style if it is set to FALSE in R1C1 style.

Sheet name is the text specifying the name of the sheet.

Return type: Text

Syntax

ADDRESS(row; col; absolute; style; sheet name)

Parameters

Comment: Row number, *Type:* Whole number (like 1, 132, 2344)

Comment: Column number, *Type:* Whole number (like 1, 132, 2344)

Comment: Absolute number (optional), *Type:* Whole number (like 1, 132, 2344)

Comment: A1 style (optional), *Type:* A truth value (TRUE or FALSE)

Comment: Sheet name, *Type:* Text

Examples

ADDRESS(6; 4) returns \$D\$6

Examples

ADDRESS(6; 4; 2) returns D\$6

Examples

ADDRESS(6; 4; 2; FALSE; "Sheet1") returns Sheet1!R6C[4]

Examples

ADDRESS(6; 4; 1; FALSE; "Sheet1") returns Sheet1!R6C4

Examples

ADDRESS(6; 4; 4; TRUE; "Sheet1") returns Sheet1!D6

8.1.9.2 AREAS

Returns the number of areas in the reference string. An area can be a single cell or a set of cells.

Return type: Whole number (like 1, 132, 2344)

Syntax

AREAS(reference)

Parameters

Comment: Reference, *Type:* A range of strings

Examples

AREAS(A1) returns 1

Examples

AREAS((A1; A2:A4)) returns 2

8.1.9.3 CELL

Returns information about position, formatting or contents in a reference.

Return type: Any kind of value

Syntax

CELL(type; reference)

Parameters

Comment: Type, *Type:* Text

Comment: Reference, *Type:* Reference

Examples

CELL("COL", C7) returns 3

Examples

CELL("ROW", C7) returns 7

Examples

CELL("ADDRESS", C7) returns \$C\$7

8.1.9.4 CHOOSE

Returns the parameter specified by the index.

Return type: Any kind of value

Syntax

CHOOSE(index; parameter1; parameter2;...)

Parameters

Comment: Index, *Type:* Whole number (like 1, 132, 2344)

Comment: Arguments, *Type:*

Examples

CHOOSE(1; "1st"; "2nd") returns "1st"

Examples

CHOOSE(2; 3; 2; 4) returns 2

8.1.9.5 COLUMN

The COLUMN function returns the column of given cell reference. If no parameter is specified the column of the current cell gets returned.

Return type: Whole number (like 1, 132, 2344)

Syntax

COLUMN(reference)

Parameters

Comment: Reference, *Type:* Text

Examples

COLUMN(A1) returns 1

Examples

COLUMN(D2) returns 4

Related Functions

[COLUMNS](#)
[ROW](#)

8.1.9.6 COLUMNS

The COLUMNS function returns the number of columns in a reference.

Return type: Whole number (like 1, 132, 2344)

Syntax

COLUMNS(reference)

Parameters

Comment: Reference, *Type:* Text

Examples

COLUMNS(A1:C3) returns 3

Examples

COLUMNS(D2) returns 1

Related Functions

[COLUMN](#)
[ROWS](#)

8.1.9.7 HLOOKUP

Look for a matching value in the first row of the given table, and return the value of the indicated row.

Looks up the 'lookup value' in the first row of the 'data source'. If a value matches, the value in the 'row' and the column, the value was found in, is returned. If 'sorted' is true (default), the first row is assumed to be sorted. The search will end, if the 'lookup value' is lower than the value, currently compared to.

Return type: String/Numeric

Syntax

HLOOKUP(Lookup value; data source; Row; Sorted)

Parameters

Comment: Lookup value, *Type:* String/Numeric

Comment: Data source, *Type:* Array

Comment: Row, *Type:* Whole number (like 1, 132, 2344)

Comment: Sorted (optional), *Type:* A truth value (TRUE or FALSE)

8.1.9.8 INDEX

If a range is given, returns value stored in a given row/column. If one cell is given, which contains an array, then one element of the array is returned.

Return type: Whole number (like 1, 132, 2344)

Syntax

INDEX(cell, row, column)

Parameters

Comment: Reference, *Type:* Text

Comment: Row, *Type:* Whole number (like 1, 132, 2344)

Comment: Column, *Type:* Whole number (like 1, 132, 2344)

Examples

INDEX(A1:C3;2;2), returns contents of B2

Examples

INDEX(A1;2;2), if A1 is a result of array calculation, returns its (2,2) element.

8.1.9.9 INDIRECT

Returns the content of the cell specified by the reference text. The second parameter is optional.

Return type: Whole number (like 1, 132, 2344)

Syntax

INDIRECT(referenceText, a1 style)

Parameters

Comment: Reference, *Type:* Text

Comment: A1 style (optional), *Type:* A truth value (TRUE or FALSE)

Examples

INDIRECT(A1), A1 contains "B1", and B1 1 => returns 1

Examples

INDIRECT("A1"), returns content of A1

8.1.9.10 LOOKUP

The LOOKUP function looks up the first parameter in the lookup vector. It returns a value in the result Vector with the same index as the matching value in the lookup vector. If value is not in the lookup vector it takes the next lower one. If no value in the lookup vector matches an error is returned. The lookup vector must be in ascending order and lookup and result vector must have the same size. Numeric values, string and boolean values are recognized. Comparison between strings is case-insensitive.

Return type: Whole number (like 1, 132, 2344)

Syntax

LOOKUP(value; lookup vector; result vector)

Parameters

Comment: Lookup value, *Type:* String/Numeric

Comment: Lookup vector, *Type:* String/Numeric

Comment: Result vector, *Type:* String/Numeric

Examples

LOOKUP(1.232; A1:A6; B1:B6) for A1 = 1, A2 = 2 returns the value of B1.

8.1.9.11 MATCH

Finds a search value in a search region, and returns its position (starting from 1). Match type can be either -1, 0 or 1 and determines how is searched for the value. If match type is 0, the index of the first value that equals search value is returned. If match type is 1 (or omitted), the index of the first value that is less than or equal to the search value is returned and the values in the search region must be sorted in ascending order. If match type is -1, the smallest value that is greater than or equal to the search value is found, and the search region needs to be sorted in descending order.

Return type: Whole number (like 1, 132, 2344)

Syntax

MATCH(Search value; Search region; Match type)

Parameters

Comment: Search value, *Type:* String/Numeric

Comment: Search region, *Type:* Reference/Array

Comment: Match type (optional), *Type:* Whole number (like 1, 132, 2344)

8.1.9.12 MULTIPLE.OPERATIONS

MULTIPLE.OPERATIONS executes the formula expression pointed to by FormulaCell and all formula expressions it depends on while replacing all references to RowCell with references to RowReplacement respectively all references to ColumnCell with references to ColumnReplacement. The function may be used to easily create tables of expressions that depend on two input parameters.

Return type: String/Numeric

Syntax

MULTIPLE.OPERATIONS(Formula cell; Row cell; Row replacement; Column cell; Column replacement)

Parameters

Comment: Formula cell, *Type:* Reference

Comment: Row cell, *Type:* Reference

Comment: Row replacement, *Type:* Reference

Comment: Column cell (optional), *Type:* Reference

Comment: Column replacement (optional), *Type:* Reference

8.1.9.13 OFFSET

Modifies a reference's position and dimension.

Return type: Reference

Syntax

OFFSET(Reference reference; Integer rowOffset; Integer columnOffset; Integer newHeight; Integer newWidth)

Parameters

Comment: Reference or range, *Type:* Reference

Comment: Number of rows to offset, *Type:* Whole number (like 1, 132, 2344)

Comment: Number of columns to offset, *Type:* Whole number (like 1, 132, 2344)

Comment: Height of the offset range (optional), *Type:* Whole number (like 1, 132, 2344)

Comment: Width of the offset range (optional), *Type:* Whole number (like 1, 132, 2344)

8.1.9.14 ROW

The ROW function returns the row of given cell reference. If no parameter is specified the row of the current cell gets returned.

Return type: Whole number (like 1, 132, 2344)

Syntax

ROW(reference)

Parameters

Comment: Reference, *Type:* Text

Examples

ROW(A1) returns 1

Examples

ROW(D2) returns 2

Related Functions

[ROWS](#)
[COLUMN](#)

8.1.9.15 ROWS

The ROWS function returns the number of rows in a reference.

Return type: Whole number (like 1, 132, 2344)

Syntax

ROWS(reference)

Parameters

Comment: Reference, *Type:* Text

Examples

ROWS(A1:C3) returns 3

Examples

ROWS(D2) returns 1

Related Functions

[ROW](#)
[COLUMNS](#)

8.1.9.16 SHEET

Returns the sheet number of the reference or the string representing a sheet name.

Return type: Whole number (like 1, 132, 2344)

Syntax

SHEET(reference)

Parameters

Comment: Reference, *Type:* Reference

Examples

SHEET(Sheet1!C7) returns 1

Examples

SHEET(Sheet2!C7) returns 2

8.1.9.17 SHEETS

Returns the number of sheets in a reference or current document.

Return type: Whole number (like 1, 132, 2344)

Syntax

SHEETS(reference)

Parameters

Comment: Reference, *Type:* Reference

8.1.9.18 VLOOKUP

Look for a matching value in the first column of the given table, and return the value of the indicated column.

Looks up the 'lookup value' in the first column of the 'data source'. If a value matches, the value in the 'column' and the row, the value was found in, is returned. If 'sorted' is true (default), the first column is assumed to be sorted. The search will end, if the 'lookup value' is lower than the value, currently compared to.

Return type: String/Numeric

Syntax

VLOOKUP(Lookup value; data source; Column; Sorted)

Parameters

Comment: Lookup value, *Type:* String/Numeric

Comment: Data source, *Type:* Array

Comment: Column, *Type:* Whole number (like 1, 132, 2344)

Comment: Sorted (optional), *Type:* A truth value (TRUE or FALSE)

8.1.10 Math

8.1.10.1 ABS

The ABS() function returns the absolute value of the floating-point number x.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

ABS(x)

Parameters

Comment: A floating point value, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

ABS(12.5) equals 12.5

Examples

ABS(-12.5) equals 12.5

8.1.10.2 CEIL

The CEIL() function rounds x up to the nearest integer which is greater than the input, returning that value as a double.

Return type: An integer (like 0, -5, 14)

Syntax

CEIL(x)

Parameters

Comment: A floating point value, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

CEIL(12.5) equals 13

Examples

CEIL(-12.5) equals -12

Related Functions

[CEILING](#)
[FLOOR](#)
[ROUND](#)
[ROUNDUP](#)

8.1.10.3 CEILING

The CEILING() function rounds x up (away from zero) to the nearest multiple of Significance which is greater than the input. The default value for Significance is 1 (or -1 if the value is negative), which means rounding up to the nearest integer. If the Mode parameter is non-zero, the function rounds away from zero, instead of up towards the positive infinity.

Return type: An integer (like 0, -5, 14)

Syntax

CEILING(x)

Parameters

Comment: A floating point value, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Significance (optional), *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Mode (optional), *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

CEILING(12.5) equals 13

Examples

CEILING(6.43; 4) equals 8

Examples

CEILING(-6.43; -4; 1) equals -8

Examples

CEILING(-6.43; -4; 0) equals -4

Related Functions

[CEIL](#)
[FLOOR](#)
[ROUND](#)
[ROUNDUP](#)

8.1.10.4 COUNT

This function returns the count of integer or floating arguments passed. You can count using a range: COUNT(A1:B5) or using a list of values like COUNT(12;5;12.5).

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

COUNT(value;value;value...)

Parameters

Comment: Values, *Type:* FLOAT

Examples

COUNT(-5;"KSpread";2) returns 2

Examples

COUNT(5) returns 1

Related Functions

[COUNTA](#)
[COUNTIF](#)
[SUM](#)

8.1.10.5 COUNTA

This function returns the count of all non empty arguments passed. You can count using a range: COUNTA(A1:B5) or using a list of values like COUNTA(12;5;12.5).

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

COUNTA(value;value;value...)

Parameters

Comment: Values, *Type:* FLOAT

Examples

COUNTA(-5;"KSpread";2) returns 3

Examples

COUNTA(5) returns 1

Related Functions

[COUNT](#)
[COUNTIF](#)

8.1.10.6 COUNTBLANK

This function returns the count of all empty cells within the range.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

COUNTBLANK(range)

Parameters

Comment: Cell range, *Type:* Range

Examples

COUNTBLANK(A1:B5)

Related Functions

[COUNT](#)
[COUNTA](#)
[COUNTIF](#)

8.1.10.7 COUNTIF

The COUNTIF() function returns the number of cells in the given range that meet the given criteria.

Return type: Whole number (like 1, 132, 2344)

Syntax

COUNTIF(range;criteria)

Parameters

Comment: Range, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Comment: Criteria, *Type:* Text

Examples

COUNTIF(A2:A3;"14") returns 1 if A2 is -4 and A3 is 14

Related Functions

COUNT
SUMIF

8.1.10.8 CUR

The CUR() function returns the non-negative cube root of x.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

CUR(x)

Parameters

Comment: A floating point value, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

CUR(27) equals 3

Related Functions

SQRT

8.1.10.9 DIV

The DIV() function divides the first value by the other values in turn.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

DIV(value;value;...)

Parameters

Comment: Values, *Type:* FLOAT

Examples

DIV(20;2;2) returns 5

Examples

DIV(25;2.5) returns 10

Related Functions

MULTIPLY
MOD

8.1.10.10 EPS

EPS() returns the machine epsilon; this is the difference between 1 and the next largest floating-point number. Because computers use a finite number of digits, roundoff error is inherent (but usually insignificant) in all calculations.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

EPS()

Parameters

Examples

On most systems, this returns $2^{-52}=2.2204460492503131e-16$

Examples

$0.5*EPS()$ returns the "unit round"; this value is interesting because it is the largest number x where $(1+x)-1=0$ (due to roundoff errors).

Examples

EPS() is so small that Calligra Sheets displays $1+eps()$ as 1

Examples

Pick a number x between 0 and EPS(). Observe that $1+x$ rounds x to either 0 or EPS() by using the equation $(1+x)-1$

8.1.10.11 EVEN

The EVEN() function returns the number rounded up to the nearest even integer.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

EVEN(value)

Parameters

Comment: Floating point value, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

EVEN(1.2) returns 2

Examples

EVEN(2) returns 2

Related Functions

[ODD](#)

8.1.10.12 EXP

The EXP() function returns the value of e (the base of natural logarithms) raised to the power of x.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

EXP(x)

Parameters

Comment: A floating point value, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

EXP(9) equals 8 103.08392758

Examples

EXP(-9) equals 0.00012341

Related Functions

[LN](#)

8.1.10.13 FACT

The FACT() function calculates the factorial of the parameter. The mathematical expression is (value)!.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

FACT(number)

Parameters

Comment: A floating point value, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

FACT(10) returns 3628800

Examples

FACT(0) returns 1

8.1.10.14 FACTDOUBLE

The FACTDOUBLE() function calculates the double factorial of a number, i.e. x!!.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

FACTDOUBLE(number)

Parameters

Comment: A floating point value, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

FACTDOUBLE(6) returns 48

Examples

FACTDOUBLE(7) returns 105

8.1.10.15 FIB

Function FIB calculates the Nth term of a Fibonacci sequence (1, 1, 2, 3, 5, 8, 13, 21...), in which each number, after the first two, is the sum of the two numbers immediately preceding it. FIB(0) is defined to be 0.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

FIB(n)

Parameters

Comment: Nth term, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

FIB(9) returns 34

Examples

FIB(26) returns 121393

8.1.10.16 FLOOR

Round a number x down to the nearest multiple of the second parameter, Significance.

The FLOOR() function rounds x down (towards zero) to the nearest multiple of Significance which is smaller than the input. The default value for Significance is 1, if x is positive. It is -1, if the value is negative, which means rounding up to the nearest integer. If mode is given and not equal to zero, the amount of x is rounded toward zero to a multiple of significance and then the sign applied. Otherwise, it rounds toward negative infinity. If any of the two parameters x or Significance is zero, the result is zero.

Return type: An integer (like 0, -5, 14)

Syntax

FLOOR(x)

Parameters

Comment: A floating point value, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Significance (optional), *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Mode (optional), *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

FLOOR(12.5) equals 12

Examples

FLOOR(-12.5) equals -13

Examples

FLOOR(5; 2) equals 4

Examples

FLOOR(5; 2.2) equals 4.4

Related Functions

CEIL
CEILING
ROUND
ROUNDDOWN

8.1.10.17 GAMMA

The GAMMA() function returns the gamma function value.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

GAMMA(value)

Parameters

Comment: Floating point value, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

GAMMA(1) returns 1

Related Functions

[FACT](#)

8.1.10.18 GCD

The GCD() function returns the greatest common denominator for two or more integer values.

Return type: Whole number (like 1, 132, 2344)

Syntax

GCD(value; value)

Parameters

Comment: First number, *Type:* A range of whole numbers (like 1, 132, 2344)

Comment: Second number, *Type:* A range of whole numbers (like 1, 132, 2344)

Comment: Third number, *Type:* A range of whole numbers (like 1, 132, 2344)

Examples

GCD(6;4) returns 2

Examples

GCD(10;20) returns 10

Examples

GCD(20;15;10) returns 5

Related Functions

[LCM](#)

8.1.10.19 G_PRODUCT

The G_PRODUCT() function is the same as KPRODUCT. It is provided for Gnumeric compatibility.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

G_PRODUCT(value;value;...)

Parameters

Comment: Values, *Type:* FLOAT

Related Functions

[KPRODUCT](#)

8.1.10.20 INT

The INT() function returns the integer part of the value.

Return type: Whole number (like 1, 132, 2344)

Syntax

INT(x)

Parameters

Comment: A floating point value, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

INT(12.55) equals 12

Examples

INT(15) equals 15

Related Functions

[FLOOR](#)
[QUOTIENT](#)

8.1.10.21 INV

This function multiplies each value by -1.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

INV(value)

Parameters

Comment: Floating point value, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

INV(-5) equals 5

Examples

INV(5) equals -5

Examples

INV(0) equals 0

8.1.10.22 KPRODUCT

The KPRODUCT() function calculates the product of all the values given as parameters. You can calculate the product of a range: KPRODUCT(A1:B5) or a list of values like KPRODUCT(12;5;12.5). If no numeric values are found 1 is returned.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

KPRODUCT(value;value;...)

Parameters

Comment: Values, *Type:* FLOAT

Examples

KPRODUCT(3;5;7) equals 105

Examples

KPRODUCT(12.5;2) equals 25

Related Functions

[G_PRODUCT](#)
[MULTIPLY](#)
[PRODUCT](#)

8.1.10.23 LCM

The LCM() function returns the least common multiple for two or more float values

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

LCM(value; value)

Parameters

Comment: First number, *Type:* FLOAT

Comment: Second number, *Type:* FLOAT

Examples

LCM(6;4) returns 12

Examples

LCM(1.5;2.25) returns 4.5

Examples

LCM(2;3;4) returns 12

Related Functions

[GCD](#)

8.1.10.24 LN

The LN() function returns the natural logarithm of x.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

LN(x)

Parameters

Comment: A floating point value, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

LN(0.8) equals -0.22314355

Examples

LN(0) equals -inf

Related Functions

LOG
LOG10
LOG2

8.1.10.25 LOG

The LOG() function returns the base-10 logarithm of x.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

LOG(x)

Parameters

Comment: A floating point value, greater than zero, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

LOG(0.8) equals -0.09691001

Examples

LOG(0) is an error.

Related Functions

LN
LOGN
LOG10
LOG2

8.1.10.26 LOG10

The LOG10() function returns the base-10 logarithm of the argument.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

LOG10(x)

Parameters

Comment: A positive floating point value, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

LOG10(10) equals 1.

Examples

LOG10(0) is an error.

Related Functions

LN
LOGN
LOG
LOG2

8.1.10.27 LOG2

The LOG2() function returns the base-2 logarithm of x.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

LOG2(x)

Parameters

Comment: A floating point value, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

LOG2(0.8) equals -0.32192809

Examples

LOG2(0) equals -inf.

Related Functions

[LN](#)
[LOGN](#)
[LOG](#)
[LOG10](#)

8.1.10.28 LOGN

The LOGn() function returns the base n logarithm of x.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

LOGn(value;base)

Parameters

Comment: A floating point value, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Base, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

LOGn(12;10) equals 1.07918125

Examples

LOGn(12;2) equals 3.5849625

Related Functions

[LOG](#)
[LN](#)
[LOG10](#)
[LOG2](#)

8.1.10.29 MAX

The MAX() function returns the largest value given in the parameters. String and logical values are ignored.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

MAX(value;value;...)

Parameters

Comment: Values, *Type:* FLOAT

Examples

MAX(12;5; 7) returns 12

Examples

MAX(12.5; 2) returns 12.5

Examples

MAX(0.5; 0.4; TRUE; 0.2) returns 0.5

Related Functions

COUNT
COUNTA
MAXA
MIN
MINA

8.1.10.30 MAXA

The MAXA() function returns the largest value given in the parameters. TRUE evaluates to 1, FALSE evaluates to 0. String values are ignored.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

MAXA(value;value;...)

Parameters

Comment: Values, *Type:* FLOAT

Examples

MAXA(12;5; 7) returns 12

Examples

MAXA(12.5; 2) returns 12.5

Examples

MAXA(0.5; 0.4; TRUE; 0.2) returns 1

Related Functions

COUNT
COUNTA
MAX
MIN
MINA

8.1.10.31 MDETERM

Function MDETERM returns the determinant of a given matrix. The matrix must be of type $n \times n$.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

MDETERM(matrix)

Parameters

Comment: Range, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Examples

MDETERM(A1:C3)

Related Functions

[MMULT](#)

8.1.10.32 MIN

The MIN() function returns the smallest value given in the parameters. String and logical values are ignored.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

MIN(value;value;...)

Parameters

Comment: Values, *Type:* FLOAT

Examples

MIN(12;5; 7) returns 5

Examples

MIN(12.5; 2) returns 2

Examples

MIN(0.4; 2; FALSE; 0.7) returns 0.4

Related Functions

[COUNT](#)
[COUNTA](#)
[MAX](#)
[MAXA](#)
[MINA](#)

8.1.10.33 MINA

The MINA() function returns the smallest value given in the parameters. TRUE evaluates to 1, FALSE to 0. String values are ignored.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

MINA(value;value;...)

Parameters

Comment: Values, *Type:* FLOAT

Examples

MINA(12;5; 7) returns 5

Examples

MINA(12.5; 2) returns 2

Examples

MINA(0.4; 2; FALSE; 0.7) returns 0.

Related Functions

COUNT
COUNTA
MAX
MAXA
MIN

8.1.10.34 MINVERSE

Calculates the inverse of the matrix.

The matrix multiplied with its inverse results in the unity matrix of the same dimension.

Invertible matrices have a non-zero determinant.

Return type: A range of floating point values (like 1.3, 0.343, 253)

Syntax

MINVERSE(matrix)

Parameters

Comment: Matrix, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Examples

MINVERSE(A1:C3)

Related Functions

MDETERM
MUNIT

8.1.10.35 MMULT

Function MMULT multiplies two matrices. Number of columns of the first matrix must be the same as row count of the second one. The result is a matrix.

Return type: A range of floating point values (like 1.3, 0.343, 253)

Syntax

MMULT(matrix1;matrix2)

Parameters

Comment: First matrix, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Comment: Second matrix, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Examples

MMULT(A1:C3)

Related Functions

[MDETERM](#)

8.1.10.36 MOD

The MOD() function returns the remainder after division. If the second parameter is null the function returns #DIV/0.

Return type: Whole number (like 1, 132, 2344)

Syntax

MOD(value;value)

Parameters

Comment: Floating point value, *Type:* Whole number (like 1, 132, 2344)

Comment: Floating point value, *Type:* Whole number (like 1, 132, 2344)

Examples

MOD(12;5) returns 2

Examples

MOD(5;5) returns 0

Related Functions

[DIV](#)

8.1.10.37 MROUND

The MROUND() function returns the value rounded to the specified multiple. The value and the multiple must have the same sign

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

MROUND(value; multiple)

Parameters

Comment: Floating point value, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Multiple, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

MROUND(1.252; 0.5) equals 1.5

Examples

MROUND(-1.252; -0.5) equals -1.5

Related Functions

[ROUND](#)

8.1.10.38 MULTINOMIAL

The MULTINOMIAL() function returns the multinomial of each number in the parameters. It uses this formula for MULTINOMIAL(a,b,c):

$(a+b+c)! / a!b!c!$

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

MULTINOMIAL(value;value;...)

Parameters

Comment: Values, *Type:* FLOAT

Examples

MULTINOMIAL(3;4;5) equals 27720

8.1.10.39 MULTIPLY

The MULTIPLY() function multiplies all the values given in the parameters. You can multiply values given by a range MULTIPLY(A1:B5) or a list of values like MULTIPLY(12;5;12.5). It's equivalent to PRODUCT.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

MULTIPLY(value;value;...)

Parameters

Comment: Values, *Type:* FLOAT

Examples

MULTIPLY(12;5;7) equals 420

Examples

MULTIPLY(12.5;2) equals 25

Related Functions

[DIV](#)

[PRODUCT](#)

[KPRODUCT](#)

8.1.10.40 MUNIT

Creates the unity matrix of the given dimension.

Return type: A range of floating point values (like 1.3, 0.343, 253)

Syntax

MUNIT(dimension)

Parameters

Comment: Dimension, *Type:* Whole number (like 1, 132, 2344)

Examples

MUNIT(3) creates a 3x3 unity matrix

Related Functions

[MINVERSE](#)

8.1.10.41 ODD

The ODD() function returns the number rounded up (or down, for negative values) to the nearest odd integer. By definition, ODD(0) is 1.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

ODD(value)

Parameters

Comment: Floating point value, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

ODD(1.2) returns 3

Examples

ODD(2) returns 3

Examples

ODD(-2) returns -3

Related Functions

[EVEN](#)

8.1.10.42 POW

The POW(x;y) function returns the value of x raised to the power of y. It's the same as POWER.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

POW(value;value)

Parameters

Comment: Floating point value, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Floating point value, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

POW(1.2;3.4) equals 1.8572

Examples

POW(2;3) equals 8

Related Functions

[POWER](#)

8.1.10.43 POWER

The POWER(x;y) function returns the value of x raised to the power of y.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

POWER(value;value)

Parameters

Comment: Floating point value, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Floating point value, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

POWER(1.2;3.4) equals 1.8572

Examples

POWER(2;3) equals 8

Related Functions

[POW](#)

8.1.10.44 PRODUCT

The PRODUCT() function calculates the product of all the values given as parameters. You can calculate the product of a range: PRODUCT(A1:B5) or a list of values like product(12;5;12.5). If no numeric values are found 0 is returned.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

PRODUCT(value;value;...)

Parameters

Comment: Values, *Type:* FLOAT

Examples

PRODUCT(3;5;7) equals 105

Examples

PRODUCT(12.5;2) equals 25

Related Functions

[MULTIPLY](#)
[KPRODUCT](#)

8.1.10.45 QUOTIENT

Function QUOTIENT returns the integer portion of numerator/denominator.

Return type: Whole number (like 1, 132, 2344)

Syntax

QUOTIENT(numerator;denominator)

Parameters

Comment: Numerator, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Denominator, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

QUOTIENT(21;4) returns 5

Related Functions

[INT](#)

8.1.10.46 RAND

The RAND() function returns a pseudo-random number between 0 and 1.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

RAND()

Parameters

Examples

RAND() equals for example 0.78309922...

Related Functions

[RANDBETWEEN](#)

[RANDEXP](#)

8.1.10.47 RANDBERNOULLI

The RANDBERNOULLI() function returns a Bernoulli-distributed pseudo-random number.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

RANDBERNOULLI(x)

Parameters

Comment: A floating point value (between 0 and 1), *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: A floating point value, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

RANDBERNOULLI(0.45)

Related Functions

[RAND](#)

8.1.10.48 RANDBETWEEN

The RANDBETWEEN() function returns a pseudo-random number between bottom and top value. If bottom > top this function returns Err.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

RANDBETWEEN(bottom;top)

Parameters

Comment: Bottom value, *Type:* Whole number (like 1, 132, 2344)

Comment: Top value, *Type:* Whole number (like 1, 132, 2344)

Examples

RANDBETWEEN(12;78) equals for example 61.0811...

Related Functions

[RAND](#)

8.1.10.49 RANDBINOM

The RANDBINOM() function returns a binomially-distributed pseudo-random number.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

RANDBINOM(x)

Parameters

Comment: A floating point value (between 0 and 1), *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Trials (greater 0), *Type:* Whole number (like 1, 132, 2344)

Examples

RANDBINOM(4)

Related Functions

[RAND](#)

[RANDNEGBINOM](#)

8.1.10.50 RANDEXP

The RANDEXP() function returns an exponentially-distributed pseudo-random number.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

RANDEXP(x)

Parameters

Comment: A floating point value (greater 0), *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

RANDEXP(0.88)

Related Functions

[RAND](#)

8.1.10.51 RANDNEGBINOM

The RANDNEGBINOM() function returns a negative binomially-distributed pseudo-random number.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

RANDNEGBINOM(x)

Parameters

Comment: A floating point value (between 0 and 1), *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Failures (greater 0), *Type:* Whole number (like 1, 132, 2344)

Examples

RANDNEGBINOM(4)

Related Functions

[RAND](#)

[RANDBINOM](#)

8.1.10.52 RANDNORM

The RANDNORM() function returns a Normal(Gaussian)-distributed pseudo-random number.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

RANDNORM(mu; sigma)

Parameters

Comment: Mean value of the normal distribution, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Dispersion of the normal distribution, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

RANDNORM(0; 1)

Related Functions

[RAND](#)

8.1.10.53 RANDPOISSON

The RANDPOISSON() function returns a poisson-distributed pseudo-random number.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

RANDPOISSON(x)

Parameters

Comment: A floating point value (greater 0), *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: A floating point value, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

RANDPOISSON(4)

Related Functions

[RAND](#)

8.1.10.54 ROOTN

The ROOTN() function returns the non-negative nth root of x.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

ROOTN(x;n)

Parameters

Comment: A floating point value, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Value, *Type:* Whole number (like 1, 132, 2344)

Examples

ROOTN(9;2) equals 3

Related Functions

[SQRT](#)

8.1.10.55 ROUND

The ROUND(value;[digits]) function returns value rounded. Digits is the number of digits to which you want to round that number. If digits is zero or omitted, value is rounded up to the nearest integer. If digits is smaller than zero, the corresponding integer part of the number is rounded.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

ROUND(value;[digits])

Parameters

Comment: Floating point value, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Digits, *Type:* Whole number (like 1, 132, 2344)

Examples

ROUND(1.252;2) equals 1.25

Examples

ROUND(-1.252;2) equals -1.25

Examples

ROUND(1.258;2) equals 1.26

Examples

ROUND(-12.25;-1) equals -10

Examples

ROUND(-1.252;0) equals -1

Related Functions

[MROUND](#)

[ROUNDDOWN](#)

[ROUNDUP](#)

8.1.10.56 ROUNDDOWN

The ROUNDDOWN(value;[digits]) function returns value rounded so that its absolute value is lesser. Digits is the number of digits to which you want to round that number. If digits is zero or omitted, value is rounded down to the nearest integer.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

ROUNDDOWN(value;[digits])

Parameters

Comment: Floating point value, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Digits, *Type:* Whole number (like 1, 132, 2344)

Examples

ROUNDDOWN(1.252) equals 1

Examples

ROUNDDOWN(1.252;2) equals 1.25

Examples

ROUNDDOWN(-1.252;2) equals -1.25

Examples

ROUNDDOWN(-1.252) equals -1

Related Functions

[ROUND](#)
[ROUNDUP](#)

8.1.10.57 ROUNDUP

The ROUNDUP(value;[digits]) function returns value rounded so that its absolute value is greater. Digits is the number of digits to which you want to round that number. If digits is zero or omitted, value is rounded up to the nearest integer.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

ROUNDUP(value;[digits])

Parameters

Comment: Floating point value, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Digits, *Type:* Whole number (like 1, 132, 2344)

Examples

ROUNDUP(1.252) equals 2

Examples

ROUNDUP(1.252;2) equals 1.26

Examples

ROUNDUP(-1.252;2) equals -1.26

Examples

ROUNDUP(-1.252) equals -2

Related Functions

ROUND
ROUNDDOWN

8.1.10.58 SERIESSUM

The SERIESSUM() function returns the sum of a power series.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

SERIESSUM(X; N; M; Coefficients)

Parameters

Comment: X the independent variable of the power series, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: N the initial power to which X is to be raised, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: M the increment by which to increase N for each term in the series, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Coefficients a set of coefficients by which each successive power of the variable X is multiplied, *Type:* FLOAT

Examples

SERIESSUM(2;0;2;{1;2}) return 9

8.1.10.59 SIGN

This function returns -1 if the number is negative, 0 if the number is null and 1 if the number is positive.

Return type: Whole number (like 1, 132, 2344)

Syntax

SIGN(value)

Parameters

Comment: Floating point value, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

SIGN(5) equals 1

Examples

SIGN(0) equals 0

Examples

SIGN(-5) equals -1

8.1.10.60 SQRT

The SQRT() function returns the non-negative square root of the argument. It is an error if the argument is negative.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

SQRT(x)

Parameters

Comment: A floating point value, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

SQRT(9) equals 3

Examples

SQRT(-9) is an error

Related Functions

[IMSQRT](#)

8.1.10.61 SQRTPI

The SQRTPI() function returns the non-negative square root of $x * \text{PI}$. It is an error if the argument is negative.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

SQRTPI(x)

Parameters

Comment: A floating point value, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

SQRTPI(2) equals 2.506628

8.1.10.62 SUBTOTAL

The SUBTOTAL() function returns a subtotal of a given list of arguments ignoring other subtotal results in there. Function can be one of the following numbers: 1 - Average, 2 - Count, 3 - CountA, 4 - Max, 5 - Min, 6 - Product, 7 - StDev, 8 - StDevP, 9 - Sum, 10 - Var, 11 - VarP.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

SUBTOTAL(function; value)

Parameters

Comment: Function, *Type:* Whole number (like 1, 132, 2344)

Comment: Values, *Type:* FLOAT

Examples

If A1:A5 contains 7, 24, 23, 56 and 9:

Examples

SUBTOTAL(1; A1:A5) returns 23.8

Examples

SUBTOTAL(4; A1:A5) returns 56

Examples

SUBTOTAL(9; A1:A5) returns 119

Examples

SUBTOTAL(11; A1:A5) returns 307.76

Related Functions

AVERAGE
COUNT
COUNTA
MAX
MIN
PRODUCT
STDEV
STDEVP
SUM
VAR
VARP

8.1.10.63 SUM

The SUM() function calculates the sum of all the values given as parameters. You can calculate the sum of a range SUM(A1:B5) or a list of values like SUM(12;5;12.5).

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

SUM(value;value;...)

Parameters

Comment: Values, *Type:* FLOAT

Examples

SUM(12;5;7) equals 24

Examples

SUM(12.5;2) equals 14.5

Related Functions

SUMA
SUMSQ
SUMIF

8.1.10.64 SUMA

The SUMA() function calculates the sum of all the values given as parameters. You can calculate the sum of a range SUMA(A1:B5) or a list of values like SUMA(12;5;12.5). If a parameter contains text or the boolean value FALSE it is counted as 0, if a parameter evaluates to TRUE it is counted as 1.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

SUM(value;value;...)

Parameters

Comment: Values, *Type:* FLOAT

Examples

SUMA(12;5; 7) equals 24

Examples

SUMA(12.5; 2; TRUE) equals 15.5

Related Functions

[SUM](#)
[SUMSQ](#)

8.1.10.65 SUMIF

The SUMIF() function calculates the sum of all values given as parameters which match the criteria. The sum range is optional. If not supplied, the values in the check range are summed. The length of the check range should be equal or less than the length of the sum range.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

SUMIF(checkrange;criteria;sumrange)

Parameters

Comment: Check range, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Comment: Criteria, *Type:* Text

Comment: Sum range, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Examples

SUMIF(A1:A4;">1") sums all values in range A1:A4 which match >1

Examples

SUMIF(A1:A4;"=0";B1:B4) sums all values in range B1:B4 if the corresponding value in A1:A4 matches =0

Related Functions

[SUM](#)
[COUNTIF](#)

8.1.10.66 SUMSQ

The SUMSQ() function calculates the sum of all the squares of values given as parameters. You can calculate the sum of a range SUMSQ(A1:B5) or a list of values like SUMSQ(12;5;12.5).

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

SUMSQ(value;value;...)

Parameters

Comment: Values, *Type:* FLOAT

Examples

SUMSQ(12;5;7) equals 218

Examples

SUMSQ(12.5;2) equals 173

Related Functions

[SUM](#)

8.1.10.67 TRANSPOSE

Returns the transpose of a matrix, i.e. rows and columns of the matrix are exchanged.

Return type: A range of floating point values (like 1.3, 0.343, 253)

Syntax

TRANSPOSE(matrix)

Parameters

Comment: Matrix, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Examples

TRANSPOSE(A1:C3)

8.1.10.68 TRUNC

The TRUNC() function truncates a numeric value to a certain precision. If the precision is omitted 0 is assumed.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

TRUNC(value; precision)

Parameters

Comment: Floating point value, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Precision, *Type:* Whole number (like 1, 132, 2344)

Examples

TRUNC(1.2) returns 1

Examples

TRUNC(213.232; 2) returns 213.23

Related Functions

ROUND
ROUNDDOWN
ROUNDUP

8.1.11 Statistical

8.1.11.1 AVEDEV

The AVEDEV() function calculates the average of the absolute deviations of a data set from their mean.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

AVEDEV(value;value;...)

Parameters

Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Examples

AVEDEV(11.4;17.3;21.3;25.9;40.1) returns 7.84

Examples

AVEDEV(A1:A5) ...

8.1.11.2 AVERAGE

The AVERAGE() function calculates the average of all the values given as parameters. You can calculate the average of a range AVERAGE(A1:B5) or a list of values like AVERAGE(12;5;12.5).

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

AVERAGE(value;value;...)

Parameters

Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Examples

AVERAGE(12;5;7) equals 8

Examples

AVERAGE(12.5;2) equals 7.25

8.1.11.3 AVERAGEA

The AVERAGEA() calculates the average of the given arguments. Numbers, text and logical values are included in the calculation too. If the cell contains text or the argument evaluates to FALSE, it is counted as value zero (0). If the argument evaluates to TRUE, it is counted as one (1). Note that empty cells are not counted.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

AVERAGEA(value;value;...)

Parameters

Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Comment: String values, *Type:* Text

Examples

AVERAGEA(11.4;17.3;"sometext";25.9;40.1) equals 18.94

8.1.11.4 BETADIST

The BETADIST() function returns the cumulative beta probability density function.

The third and fourth parameters are optional. They set the lower and upper bounds, otherwise defaulting to 0.0 and 1.0 respectively.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

BETADIST(number;alpha;beta;start;end;[cumulative=TRUE])

Parameters

Comment: Number, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Alpha parameter, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Beta parameter, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Start, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: End, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Cumulative, *Type:* A truth value (TRUE or FALSE)

Examples

BETADIST(0.2859;0.2606;0.8105) equals 0.675444

Examples

BETADIST(0.2859;0.2606;0.8105;0.2;0.9) equals 0.537856

8.1.11.5 BETAINV

The BETAINV() function returns the inverse of BETADIST(x;alpha;beta;a;b;TRUE()).

The start and end parameters are optional. They set the lower and upper bounds, otherwise defaulting to 0.0 and 1.0 respectively.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

BETAINV(number;alpha;beta [; start=0 [; end=1]])

Parameters

Comment: Number, *Type:* A floating point value (like 1.3, 0.343, 253)
Comment: Alpha parameter, *Type:* A floating point value (like 1.3, 0.343, 253)
Comment: Beta parameter, *Type:* A floating point value (like 1.3, 0.343, 253)
Comment: Start, *Type:* A floating point value (like 1.3, 0.343, 253)
Comment: End, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

BETADIST(BETAINV(0.1;3;4);3;4) equals 0.1

Examples

BETADIST(BETAINV(0.3;3;4);3;4) equals 0.3

8.1.11.6 BINO

The BINO() function returns the binomial distribution.

The first parameter is the number of trials, the second parameter is the number of successes, and the third is the probability of success. The number of trials should be greater than the number of successes and the probability should be smaller or equal to 1.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

BINO(trials;success;prob_of_success)

Parameters

Comment: Number of trials, *Type:* Whole number (like 1, 132, 2344)
Comment: Number of successful trials, *Type:* Whole number (like 1, 132, 2344)
Comment: Probability of success, *Type:* Double

Examples

BINO(12;9;0.8) returns 0.236223201

8.1.11.7 CHIDIST

The CHIDIST() function returns the probability value from the indicated Chi square that a hypothesis is confirmed.

CHIDIST compares the Chi square value to be given for a random sample that is calculated from the sum of (observed value-expected value)²/expected value for all values with the theoretical Chi square distribution and determines from this the probability of error for the hypothesis to be tested.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

CHIDIST(number;degrees_freedom)

Parameters

Comment: Number, *Type:* A floating point value (like 1.3, 0.343, 253)
Comment: Degrees of freedom, *Type:* Whole number (like 1, 132, 2344)

Examples

CHIDIST(13.27;5) returns 0.021

8.1.11.8 COMBIN

The COMBIN() function calculates the count of possible combinations. The first parameter is the total count of elements. The second parameter is the count of elements to choose. Both parameters should be positive and the first parameter should not be less than the second. Otherwise the function returns an error.

Return type: Whole number (like 1, 132, 2344)

Syntax

COMBIN(total;chosen)

Parameters

Comment: Total number of elements, *Type:* Whole number (like 1, 132, 2344)

Comment: Number of elements to choose, *Type:* Whole number (like 1, 132, 2344)

Examples

COMBIN(12;5) returns 792

Examples

COMBIN(5;5) returns 1

8.1.11.9 COMBINA

The COMBINA() function calculates the count of possible combinations. The first parameter is the total count of elements. The second parameter is the count of elements to choose. Both parameters should be positive and the first parameter should not be less than the second. Otherwise the function returns an error.

Return type: Whole number (like 1, 132, 2344)

Syntax

COMBIN(total;chosen)

Parameters

Comment: Total number of elements, *Type:* Whole number (like 1, 132, 2344)

Comment: Number of elements to choose, *Type:* Whole number (like 1, 132, 2344)

Examples

COMBIN(12;5) returns 792

Examples

COMBIN(5;5) returns 1

8.1.11.10 CONFIDENCE

The CONFIDENCE() function returns the confidence interval for a population mean.

The alpha parameter must be between 0 and 1 (non-inclusive), stddev must be positive and size must be greater or equal to 1.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

CONFIDENCE(alpha;stddev;size)

Parameters

Comment: Level of the confidence interval, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Standard deviation for the total population, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Size of the total population, *Type:* Whole number (like 1, 132, 2344)

Examples

CONFIDENCE(0.05;1.5;100) equals 0.294059

8.1.11.11 CORREL

The CORREL() function calculates the correlation coefficient of two cell ranges.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

CORREL(range1; range2)

Parameters

Comment: Cell range of values, *Type:* Double

Comment: Second cell range of values, *Type:* Double

Examples

CORREL(A1:A3; B1:B3)

Related Functions

[PEARSON](#)

8.1.11.12 COVAR

The COVAR() function calculates the covariance of two cell ranges.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

COVAR(range1; range2)

Parameters

Comment: Cell range of values, *Type:* Double

Comment: Second cell range of values, *Type:* Double

Examples

COVAR(A1:A3; B1:B3)

8.1.11.13 DEVSQ

The DEVSQ() function calculates the sum of squares of deviations.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

DEVSQ(value; value;...)

Parameters

Comment: Floating point values, *Type:* Double
Comment: Floating point values, *Type:* Double
Comment: Floating point values, *Type:* Double
Comment: Floating point values, *Type:* Double
Comment: Floating point values, *Type:* Double

Examples

DEVSQ(A1:A5)

Examples

DEVSQ(21; 33; 54; 23) returns 684.75

8.1.11.14 EXPONDIST

The EXPONDIST() function returns the exponential distribution.

The lambda parameter must be positive.

Cumulative = 0 calculates the density function; cumulative = 1 calculates the distribution.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

EXPONDIST(number;lambda;cumulative)

Parameters

Comment: Number, *Type:* A floating point value (like 1.3, 0.343, 253)
Comment: Lambda parameter, *Type:* A floating point value (like 1.3, 0.343, 253)
Comment: 0 = density, 1 = distribution, *Type:* Whole number (like 1, 132, 2344)

Examples

EXPONDIST(3;0.5;0) equals 0.111565

Examples

EXPONDIST(3;0.5;1) equals 0.776870

8.1.11.15 FDIST

The FDIST() function returns the f-distribution.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

FDIST(number;degrees_freedom_1;degrees_freedom_2)

Parameters

Comment: Number, *Type:* A floating point value (like 1.3, 0.343, 253)
Comment: Degrees of freedom 1, *Type:* Whole number (like 1, 132, 2344)
Comment: Degrees of freedom 2, *Type:* Whole number (like 1, 132, 2344)

Examples

FDIST(0.8;8;12) yields 0.61

8.1.11.16 FINV

The FINV() function returns the unique non-negative number x such that $FDIST(x;r1;r2) = p$.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

FINV(number; r1; r2)

Parameters

Comment: Number, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Number r1, *Type:* Whole number (like 1, 132, 2344)

Comment: Number r2, *Type:* Whole number (like 1, 132, 2344)

Examples

FDIST(FINV(0.1;3;4);3;4) equals 0.1

8.1.11.17 FISHER

The FISHER() function returns the Fisher transformation for x and creates a function close to a normal distribution.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

FISHER(number)

Parameters

Comment: Number, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

FISHER(0.2859) equals 0.294096

Examples

FISHER(0.8105) equals 1.128485

8.1.11.18 FISHERINV

The FISHERINV() function returns the inverse of the Fisher transformation for x and creates a function close to a normal distribution.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

FISHERINV(number)

Parameters

Comment: Number, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

FISHERINV(0.2859) equals 0.278357

Examples

FISHERINV(0.8105) equals 0.669866

8.1.11.19 FREQUENCY

Counts the number of values for each interval given by the border values in the second parameter. The values in the second parameter determine the upper boundaries of the intervals. The intervals include the upper boundaries. The returned array is a column vector and has one more element than the second parameter; the last element represents the number of all elements greater than the last value in second parameter. If the second parameter is empty, all values in the first parameter are counted.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

FREQUENCY(Range data; Range bins)

Parameters

Comment: Floating point values, that should be counted., *Type:* A range of floating point values (like 1.3, 0.343, 253)

Comment: Floating point values, representing the upper boundaries of the intervals., *Type:* A range of floating point values (like 1.3, 0.343, 253)

8.1.11.20 GAMMADIST

The GAMMADIST() function returns the gamma distribution.

If the last parameter (cumulated) is 0, it calculates the density function; if it's 1, the distribution is returned.

The first three parameters must be positive.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

GAMMADIST(number;alpha;beta;cumulated)

Parameters

Comment: Number, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Alpha parameter, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Beta parameter, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Cumulated flag, *Type:* Whole number (like 1, 132, 2344)

Examples

GAMMADIST(0.758;0.1;0.35;1) equals 0.995450

Examples

GAMMADIST(0.758;0.1;0.35;0) equals 0.017179

8.1.11.21 GAMMAINV

The GAMMAINV() function returns the unique number $x \geq 0$ such that $GAMMAINV(x;alpha;beta;TRUE()) = p$.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

GAMMAINV(number;alpha;beta)

Parameters

Comment: Number, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Alpha parameter, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Beta parameter, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

GAMMADIST(GAMMAINV(0.1;3;4);3;4) equals 0.1

Examples

GAMMADIST(GAMMAINV(0.3;3;4);3;4) equals 0.3

8.1.11.22 GAMMALN

The GAMMALN() function returns the natural logarithm of the gamma function: G(x). The number parameter must be positive.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

GAMMALN(Number)

Parameters

Comment: Number, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

GAMMALN(2) returns 0

8.1.11.23 GAUSS

The GAUSS() function returns the integral values for the standard normal cumulative distribution.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

GAUSS(value)

Parameters

Comment: The number for which the integral value of standard normal distribution is to be calculated, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

GAUSS(0.25) equals 0.098706

8.1.11.24 GEOMEAN

The GEOMEAN() function returns the geometric mean of the given arguments. This is equal to the Nth root of the product of the terms.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

GEOMEAN(value; value;...)

Parameters

Comment: Floating point values, *Type:* Double
Comment: Floating point values, *Type:* Double
Comment: Floating point values, *Type:* Double
Comment: Floating point values, *Type:* Double
Comment: Floating point values, *Type:* Double

Examples

GEOMEAN(A1:A5)

Examples

GEOMEAN(21; 33; 54; 23) returns 30.45886

Related Functions

[HARMEAN](#)

8.1.11.25 HARMEAN

The HARMEAN() function returns the harmonic mean of the N data points (N divided by the sum of the inverses of the data points).

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

HARMEAN(value; value;...)

Parameters

Comment: Floating point values, *Type:* Double
Comment: Floating point values, *Type:* Double
Comment: Floating point values, *Type:* Double
Comment: Floating point values, *Type:* Double
Comment: Floating point values, *Type:* Double

Examples

HARMEAN(A1:A5)

Examples

HARMEAN(21; 33; 54; 23) returns 28.588

Related Functions

[GEOMEAN](#)

8.1.11.26 HYPGEOMDIST

The HYPGEOMDIST() function returns the hypergeometric distribution.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

HYPGEOMDIST(x; n; M; N)

Parameters

Comment: Number of success in the sample, *Type:* Whole number (like 1, 132, 2344)
Comment: Number of trials, *Type:* Whole number (like 1, 132, 2344)
Comment: Number of success overall, *Type:* Whole number (like 1, 132, 2344)
Comment: Population size, *Type:* Whole number (like 1, 132, 2344)

Examples

HYPGEOMDIST(2; 5; 6; 20) returns 0.3522

8.1.11.27 INTERCEPT

The INTERCEPT() function calculates the interception of the linear regression line with the y axis.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

INTERCEPT(y;x)

Parameters

Comment: y values (array), *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: x values (array), *Type:* A floating point value (like 1.3, 0.343, 253)

8.1.11.28 INVBINO

The INVBINO() function returns the negative binomial distribution. The first parameter is the number of trials, the second parameter is the number of failures, and the third is the probability of failure. The number of trials should be larger than the number of failures and the probability should be smaller or equal to 1.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

INVBINO(trials;failure;prob_of_failure)

Parameters

Comment: Number of trials, *Type:* Whole number (like 1, 132, 2344)

Comment: Number of failures, *Type:* Whole number (like 1, 132, 2344)

Comment: Probability of failure, *Type:* Double

Examples

INVBINO(12;3;0.2) returns 0.236223201

8.1.11.29 KURT

The KURT() function calculates an unbiased estimate of the kurtosis of a data set. You have to provide at least 4 values, otherwise an error is returned.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

KURT(value; value;...)

Parameters

Comment: Floating point values, *Type:* Double

Comment: Floating point values, *Type:* Double

Comment: Floating point values, *Type:* Double

Comment: Floating point values, *Type:* Double

Comment: Floating point values, *Type:* Double

Examples

KURT(A1:A5)

Examples

KURT(21; 33; 54; 23) returns 1.344239

Related Functions

[KURTP](#)

8.1.11.30 KURTP

The KURTP() function calculates an population kurtosis of a data set. You have to provide at least 4 values, otherwise an error is returned.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

KURTP(value; value;...)

Parameters

Comment: Floating point values, *Type:* Double

Comment: Floating point values, *Type:* Double

Comment: Floating point values, *Type:* Double

Comment: Floating point values, *Type:* Double

Comment: Floating point values, *Type:* Double

Examples

KURTP(A1:A5)

Examples

KURTP(21; 33; 54; 23) returns -1.021

Related Functions

[KURT](#)

8.1.11.31 LARGE

The LARGE() function returns the k-th largest value from the data set.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

LARGE(range; k)

Parameters

Comment: Cell range of values, *Type:* Double

Comment: Position (from the largest), *Type:* Whole number (like 1, 132, 2344)

Examples

A1: 3, A2: 1, A3: 5 => LARGE(A1:A3; 2) returns 3

8.1.11.32 LEGACYFDIST

The LEGACYFDIST() function returns the f-distribution.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

LEGACYFDIST(number;degrees_freedom_1;degrees_freedom_2)

Parameters

Comment: Number, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Degrees of freedom 1, *Type:* Whole number (like 1, 132, 2344)

Comment: Degrees of freedom 2, *Type:* Whole number (like 1, 132, 2344)

Examples

LEGACYFDIST(0.8;8;12) yields 0.61

8.1.11.33 LOGINV

The LOGINV() function returns the inverse of the lognormal cumulative distribution.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

LOGINV(p; mean; stdev)

Parameters

Comment: Probability, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Mean value of the standard logarithmic distribution, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Standard deviation of the standard logarithmic distribution, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

LOGINV(0.1;0;1) equals 0.2776

8.1.11.34 LOGNORMDIST

The LOGNORMDIST() function returns the cumulative lognormal distribution.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

LOGNORMDIST(Number;MV;STD)

Parameters

Comment: Probability value for which the standard logarithmic distribution is to be calculated, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Mean value of the standard logarithmic distribution, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Standard deviation of the standard logarithmic distribution, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

LOGNORMDIST(0.1;0;1) equals 0.01

8.1.11.35 MEDIAN

The MEDIAN() function calculates the median of all the values given as parameters. You can calculate the median of a range like MEDIAN(A1:B5) or a list of values like MEDIAN(12; 5; 12.5). Blank cells will be considered as a zero, and cells with text will be ignored.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

MEDIAN(value;value;...)

Parameters

Comment: Floating point value or range of values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Comment: Floating point values or range of values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Comment: Floating point values or range of values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Comment: Floating point values or range of values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Comment: Floating point values or range of values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Examples

MEDIAN(12; 5; 5.5) equals 5.5

Examples

MEDIAN(12; 7; 8;2) equals 7.5

8.1.11.36 MODE

The MODE() function returns the most frequently occurring value in the data set.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

MODE(number; number2; ...)

Parameters

Comment: Float, *Type:* Double

Comment: Float, *Type:* Double

Comment: Float, *Type:* Double

Comment: Float, *Type:* Double

Examples

MODE(12; 14; 12; 15) returns 12

8.1.11.37 NEGBINOMDIST

The NEGBINOMDIST() function returns the negative binomial distribution.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

NEGBINOMDIST(failures; success; prob_of_success)

Parameters

Comment: Number of failures, *Type:* Whole number (like 1, 132, 2344)

Comment: Number of successful trials, *Type:* Whole number (like 1, 132, 2344)

Comment: Probability of success, *Type:* Double

Examples

NEGBINOMDIST(2;5;0.55) returns 0.152872629

8.1.11.38 NORMDIST

The NORMDIST() function returns the normal cumulative distribution.

Number is the value of the distribution based on which the normal distribution is to be calculated.

MV is the linear middle of the distribution.

STD is the standard deviation of the distribution.

K = 0 calculates the density function; K = 1 calculates the distribution.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

NORMDIST(Number;MV;STD;K)

Parameters

Comment: Number, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Linear middle of the distribution, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Standard deviation of the distribution, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: 0 = density, 1 = distribution, *Type:* Whole number (like 1, 132, 2344)

Examples

NORMDIST(0.859;0.6;0.258;0) equals 0.934236

Examples

NORMDIST(0.859;0.6;0.258;1) equals 0.842281

8.1.11.39 NORMINV

The NORMINV() function returns the inverse of the normal cumulative distribution. The number must be between 0 and 1 (non-inclusive) and STD must be positive.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

NORMINV(number;MV;STD)

Parameters

Comment: Probability value for which the standard logarithmic distribution is to be calculated, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Middle value in the normal distribution, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Standard deviation of the normal distribution, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

NORMINV(0.9;63;5) equals 69.41

8.1.11.40 NORMSDIST

The NORMSDIST() function returns the standard normal distribution.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

NORMSDIST(Number)

Parameters

Comment: Value to which the standard normal distribution is calculated, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

NORMSDIST(1) equals 0.84

8.1.11.41 NORMSINV

The NORMSINV() function returns the inverse of the standard normal cumulative distribution. The number must be between 0 and 1 (non-inclusive).

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

NORMSINV(Number)

Parameters

Comment: Number, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

NORMSINV(0.908789) returns 1.3333

8.1.11.42 PEARSON

The PEARSON() function calculates the correlation coefficient of two cell ranges. It is the same as the CORREL function.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

PEARSON(range1; range2)

Parameters

Comment: Cell range of values, *Type:* Double

Comment: Second cell range of values, *Type:* Double

Examples

PEARSON(A1:A3; B1:B3)

Related Functions

[CORREL](#)

8.1.11.43 PERCENTILE

The PERCENTILE() function returns the x-th sample percentile of data values in Data. A percentile returns the scale value for a data series which goes from the smallest (alpha=0) to the largest value (alpha=1) of a data series. For alpha = 25%, the percentile means the first quartile; alpha = 50% is the MEDIAN. Blank cells will be considered as a zero, and cells with text will be ignored.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

PERCENTILE(data;alpha)

Parameters

Comment: Range of values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Comment: The percentile value between 0 and 1, inclusive., *Type:* A floating point value (like 1.3, 0.343, 253)

Related Functions

[MEDIAN](#)

8.1.11.44 PERMUT

The PERMUT() function returns the number of permutations. The first parameter is the number of elements, and the second parameter is the number of elements used in the permutation.

Return type: Whole number (like 1, 132, 2344)

Syntax

PERMUT(total;permutated)

Parameters

Comment: Total number of elements, *Type:* Whole number (like 1, 132, 2344)

Comment: Number of elements to permutate, *Type:* Whole number (like 1, 132, 2344)

Examples

PERMUT(8;5) equals 6720

Examples

PERMUT(1;1) equals 1

8.1.11.45 PERMUTATIONA

The PERMUTATIONA() function returns the number of ordered permutations when allowing repetition. The first parameter is the number of elements, and the second parameter is the number of elements to choose. Both parameters must be positive.

Return type: Whole number (like 1, 132, 2344)

Syntax

PERMUTATIONA(total;chosen)

Parameters

Comment: Total number of elements, *Type:* Whole number (like 1, 132, 2344)

Comment: Number of elements to choose, *Type:* Whole number (like 1, 132, 2344)

Examples

PERMUTATIONA(2,3) returns 8

Examples

PERMUTATIONA(0,0) returns 1

8.1.11.46 PHI

The PHI() function returns value of the distribution function for a standard normal distribution.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

PHI(value)

Parameters

Comment: The number for which the standard normal distribution is to be calculated, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

PHI(0.25) equals 0.386668

8.1.11.47 POISSON

The POISSON() function returns the Poisson distribution.

The lambda and number parameters must be positive.

Cumulative = 0 calculates the density function; cumulative = 1 calculates the distribution.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

POISSON(number;lambda;cumulative)

Parameters

Comment: Number, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Lambda parameter (the middle value), *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: 0 = density, 1 = distribution, *Type:* Whole number (like 1, 132, 2344)

Examples

POISSON(60;50;0) equals 0.020105

Examples

POISSON(60;50;1) equals 0.927840

8.1.11.48 RANK

The RANK() function returns the rank of a number in a list of numbers.

Order specifies how to rank the numbers:

If 0 or omitted, Data is ranked in descending order.

If not 0, Data is ranked in ascending order.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

RSQ(Value; Data; Order)

Parameters

Comment: Value, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Data (array), *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Order, *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

RANK (2;{1;2;3}) equals 2

8.1.11.49 RSQ

The RSQ() function returns the square of the Pearson product moment correlation coefficient through data points in known_y's and known_x's.

If "arrayY" and "arrayX" are empty or have a different number of data points, then #N/A is returned.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

RSQ(known Y; known X)

Parameters

Comment: known Y (array), *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: known X (array), *Type:* A floating point value (like 1.3, 0.343, 253)

8.1.11.50 SKEW

The SKEW() function returns an estimate for skewness of a distribution

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

SKEW(number; number2; ...)

Parameters

Comment: Float, *Type:* Double

Comment: Float, *Type:* Double

Comment: Float, *Type:* Double

Comment: Float, *Type:* Double

Examples

SKEW(11.4; 17.3; 21.3; 25.9; 40.1) returns 0.9768

Related Functions

[SKEWP](#)

8.1.11.51 SKEWP

The SKEWP() function returns the population skewness of a distribution

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

SKEWP(number; number2; ...)

Parameters

Comment: Float, *Type:* Double

Comment: Float, *Type:* Double

Comment: Float, *Type:* Double

Comment: Float, *Type:* Double

Examples

SKEWP(11.4; 17.3; 21.3; 25.9; 40.1) returns 0.6552

Related Functions

[SKEW](#)

8.1.11.52 SLOPE

The SLOPE() function calculates the slope of the linear regression line.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

SLOPE(y;x)

Parameters

Comment: y values (array), *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: x values (array), *Type:* A floating point value (like 1.3, 0.343, 253)

8.1.11.53 SMALL

The SMALL() function returns the k-th smallest value from the data set.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

SMALL(range; k)

Parameters

Comment: Cell range of values, *Type:* Double

Comment: Position (from the smallest), *Type:* Whole number (like 1, 132, 2344)

Examples

A1: 3, A2: 1, A3: 5 => SMALL(A1:A3; 1) returns 1

8.1.11.54 STANDARDIZE

The STANDARDIZE() function calculates a normalized value.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

STANDARDIZE(x; mean, stdev)

Parameters

Comment: Number to be normalized, *Type:* Double

Comment: Mean of the distribution, *Type:* Double

Comment: Standard deviation, *Type:* Double

Examples

STANDARDIZE(4; 3; 7) returns 0.1429

8.1.11.55 STDEV

The STDEV() function returns the estimate standard deviation based on a sample. The standard deviation is a measure of how widely values are dispersed from the average value.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

STDEV(value;value;...)

Parameters

Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Examples

STDEV(6;7;8) equals 1

Related Functions

[STDEVP](#)

8.1.11.56 STDEVA

The STDEVA() function returns the estimate standard deviation based on a sample. The standard deviation is a measure of how widely values are dispersed from the average value. If a referenced cell contains text or contains the boolean value FALSE, it is counted as 0. If the boolean value is TRUE it is counted as 1.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

STDEVA(value;value;...)

Parameters

Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)
Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)
Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)
Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)
Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Examples

STDEVA(6; 7; A1; 8) equals 1, if A1 is empty

Examples

STDEVA(6; 7; A1; 8) equals 3.109, if A1 is TRUE

Related Functions

[STDEV](#)
[STDEVP](#)

8.1.11.57 STDEVP

The STDEVP() function returns the standard deviation based on an entire population

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

STDEVP(value;value;...)

Parameters

Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)
Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)
Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)
Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)
Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Examples

STDEVP(6;7;8) equals 0.816497...

Related Functions

[STDEV](#)

8.1.11.58 STDEVPA

The STDEVPA() function returns standard deviation based on an entire population. If a referenced cell contains text or contains the boolean value FALSE, it is counted as 0. If the boolean value is TRUE it is counted as 1.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

STDEVPA(value;value;...)

Parameters

Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)
Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)
Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)
Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)
Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Examples

STDEVPA(6; 7; A1; 8) equals 0.816497..., if A1 is empty

Examples

STDEVPA(6; 7; A1; 8) equals 2.69..., if A1 is TRUE

Examples

STDEVPA(6; 7; A1; 8) equals 3.11..., if A1 is FALSE

Related Functions

[STDEV](#)
[STDEVP](#)

8.1.11.59 STEYX

The STEYX() function calculates the standard error of the predicted y value for each x in the regression.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

SLOPE(y;x)

Parameters

Comment: y values (array), *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: x values (array), *Type:* A floating point value (like 1.3, 0.343, 253)

8.1.11.60 SUM2XMY

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

SLOPE(y;x)

Parameters

8.1.11.61 SUMPRODUCT

The SUMPRODUCT() function (SUM(X*Y)) returns the sum of the product of these values. The number of values in the two arrays should be equal. Otherwise this function returns Err.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

SUMPRODUCT(array1;array2)

Parameters

Comment: Value (array), *Type:* Double

Comment: Value (array), *Type:* Double

Examples

SUMPRODUCT(A1:A2;B1:B2) with A1=2, A2=5, B1=3 and B2=5, returns 31

8.1.11.62 SUMX2MY2

The SUMX2MY2() function ($SUM(X^2-Y^2)$) returns the difference of the squares of these values. The number of values in the two arrays should be equal. Otherwise this function returns Err.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

SUMX2MY2(array1;array2)

Parameters

Comment: Value (array), *Type:* Double

Comment: Value (array), *Type:* Double

Examples

SUMX2MY2(A1:A2;B1:B2) with A1=2, A2=5, B1=3 and B2=5, returns -5

8.1.11.63 SUMX2PY2

The SUMX2PY2() function ($SUM(X^2+Y^2)$) returns the sum of the squares of these values. The number of values in the two arrays should be equal. Otherwise this function returns Err.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

SUMX2PY2(array1;array2)

Parameters

Comment: Value (array), *Type:* Double

Comment: Value (array), *Type:* Double

Examples

SUMX2PY2(A1:A2;B1:B2) with A1=2, A2=5, B1=3 and B2=5, returns 63

8.1.11.64 SUMXMY2

The SUMXMY2() function ($SUM((X-Y)^2)$) returns the square of the differences of these values. The number of values in the two arrays should be equal. Otherwise this function returns Err.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

SUMXMY2(array1;array2)

Parameters

Comment: Value (array), *Type:* Double

Comment: Value (array), *Type:* Double

Examples

SUMXMY2(A1:A2;B1:B2) with A1=2, A2=5, B1=3 and B2=5, returns 1

8.1.11.65 TDIST

The TDIST() function returns the t-distribution.

Mode = 1 returns the one-tailed test, Mode = 2 returns the two-tailed test.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

TDIST(number;degrees_freedom;mode)

Parameters

Comment: Number, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Degrees of freedom for the t-distribution, *Type:* Whole number (like 1, 132, 2344)

Comment: Mode (1 or 2), *Type:* Whole number (like 1, 132, 2344)

Examples

TDIST(12;5;1) returns 0.000035

8.1.11.66 TREND

The TREND() function calculates a sequence of values based on a linear regression of known value pairs.

Constraints: COUNT(knownY) = COUNT(knownX).

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

TREND(knownY[;knownX[;newX[;allowOffset = TRUE]])

Parameters

Comment: KnownY, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: KnownX, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: NumberSequence newX, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: allowOffset, *Type:* A truth value (TRUE or FALSE)

8.1.11.67 TRIMMEAN

The TRIMMEAN() function calculates the mean of a data set's fraction.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

TRIMMEAN(dataSet; cutOffFraction)

Parameters

Comment: dataSet, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: cutOffFraction, *Type:* A floating point value (like 1.3, 0.343, 253)

8.1.11.68 TTEST

The TTEST() function calculates the probability of a t-test.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

TTEST(x; y; type; mode)

Parameters

Comment: x (array), *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: y (array), *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: type, *Type:* Whole number (like 1, 132, 2344)

Comment: mode, *Type:* Whole number (like 1, 132, 2344)

8.1.11.69 VAR

The VAR() function calculates the estimates variance based on a sample.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

VAR(value;value;...)

Parameters

Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Examples

VAR(12;5;7) equals 13

Examples

VAR(15;80;3) equals 1716.333...

Examples

VAR(6;7;8) equals 1

Related Functions

[VARIANCE](#)

[VARA](#)

[VARP](#)

[VARPA](#)

8.1.11.70 VARA

The VARA() function calculates the variance based on a sample.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

VARA(value;value;...)

Parameters

Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)
Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)
Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)
Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)
Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Examples

VARA(12;5;7) equals 13

Examples

VARA(15;80;3) equals 1716.333...

Examples

VARA(6;7;8) equals 1

Related Functions

[VAR](#)
[VARP](#)
[VARPA](#)

8.1.11.71 VARIANCE

The VARIANCE() function calculates the estimates variance based on a sample. It's the same as the VAR function.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

VARIANCE(value;value;...)

Parameters

Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)
Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)
Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)
Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)
Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Examples

VARIANCE(12;5;7) equals 13

Examples

VARIANCE(15;80;3) equals 1716.333...

Examples

VARIANCE(6;7;8) equals 1

Related Functions

[VAR](#)
[VARA](#)
[VARP](#)
[VARPA](#)

8.1.11.72 VARP

The VARP() function calculates the variance based on an entire population.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

VARP(value;value;...)

Parameters

Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Examples

VARP(12;5;7) equals 8.666...

Examples

VARP(15;80;3) equals 1144.22...

Examples

VARP(6;7;8) equals 0.6666667...

Related Functions

[VAR](#)

[VARA](#)

[VARPA](#)

8.1.11.73 VARPA

The VARPA() function calculates the variance based on an entire population. Text and boolean values that evaluate to FALSE are counted as 0, boolean value that evaluate to TRUE are counted as 1.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

VARPA(value;value;...)

Parameters

Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Comment: Floating point values, *Type:* A range of floating point values (like 1.3, 0.343, 253)

Examples

VARPA(12;5;7) equals 8.666...

Examples

VARPA(15;80;3) equals 1144.22...

Examples

VARPA(6;7;8) equals 0.6666667...

Related Functions

VAR
VARA
VARP

8.1.11.74 WEIBULL

The WEIBULL() function returns the Weibull distribution.

The alpha and beta parameters must be positive, the number (first parameter) must be non-negative.

Cumulative = 0 calculates the density function; cumulative = 1 calculates the distribution.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

WEIBULL(number;alpha;beta;cumulative)

Parameters

Comment: Number, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Alpha parameter, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Beta parameter, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: 0 = density, 1 = distribution, *Type:* Whole number (like 1, 132, 2344)

Examples

WEIBULL(2;1;1;0) equals 0.135335

Examples

WEIBULL(2;1;1;1) equals 0.864665

8.1.11.75 ZTEST

The ZTEST() function calculates the two tailed probability of a z-test with normal distribution.

Performs a test of the null hypothesis, that sample is a sample of a normal distributed random variable with mean mean and standard deviation sigma. A return value of 1 indicates, that the null hypothesis is rejected, i.e. the sample is not a random sample of the normal distribution. If sigma is omitted, it is estimated from sample, using STDEV.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

ZTEST(x; mean; standardDeviation)

Parameters

Comment: x (array), *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: mean, *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: standardDeviation, *Type:* A floating point value (like 1.3, 0.343, 253)

8.1.12 Text

8.1.12.1 ASC

The ASC() function returns the half-width characters corresponding to the full-width argument.

Return type: Text

Syntax

ASC(text)

Parameters

Comment: Full width characters, *Type:* Text

Related Functions

[JIS](#)

8.1.12.2 BAHTTEXT

The BAHTTEXT() function converts a number to a text in Thai characters (baht).

Return type: Text

Syntax

BAHTTEXT(number)

Parameters

Comment: Number, *Type:* Whole number (like 1, 132, 2344)

Examples

BAHTTEXT(23) returns "๒๓"

8.1.12.3 CHAR

The CHAR() function returns the character specified by a number.

Return type: Text

Syntax

CHAR(code)

Parameters

Comment: Character code, *Type:* Whole number (like 1, 132, 2344)

Examples

CHAR(65) returns "A"

Related Functions

[CODE](#)

8.1.12.4 CLEAN

The CLEAN() function removes every non-printable character from the string

Return type: Text

Syntax

CLEAN(text)

Parameters

Comment: Source string, *Type:* Text

Examples

CLEAN(AsciiToChar(7) + "HELLO") returns "HELLO"

8.1.12.5 CODE

The CODE() function returns a numeric code for the first character in a text string.

Return type: Whole number (like 1, 132, 2344)

Syntax

CODE(text)

Parameters

Comment: Text, *Type:* Text

Examples

CODE("KDE") returns 75

Related Functions

[CHAR](#)

8.1.12.6 COMPARE

The COMPARE() function returns 0 if the two strings are equal; -1 if the first one is lower in value than the second one; otherwise it returns 1.

Return type: Whole number (like 1, 132, 2344)

Syntax

COMPARE(string1; string2; true | false)

Parameters

Comment: First string, *Type:* Text

Comment: String to compare with, *Type:* Text

Comment: Compare case-sensitive (true/false), *Type:* A truth value (TRUE or FALSE)

Examples

COMPARE("Calligra"; "Calligra"; true) returns 0

Examples

COMPARE("calligra"; "Calligra"; true) returns 1

Examples

COMPARE("kspread"; "Calligra"; false) returns 1

Related Functions

[EXACT](#)

8.1.12.7 CONCATENATE

The CONCATENATE() function returns a string which is the concatenation of the strings passed as parameters.

Return type: Text

Syntax

CONCATENATE(value;value;...)

Parameters

Comment: String values, *Type:* A range of strings

Comment: String values, *Type:* A range of strings

Comment: String values, *Type:* A range of strings

Comment: String values, *Type:* A range of strings

Comment: String values, *Type:* A range of strings

Examples

CONCATENATE("Sheets";"Calligra";"KDE") returns "SheetsCalligraKDE"

8.1.12.8 DOLLAR

The DOLLAR() function converts a number to text using currency format, with the decimals rounded to the specified place. Although the name is DOLLAR, this function will do the conversion according to the current locale.

Return type: Text

Syntax

DOLLAR(number;decimals)

Parameters

Comment: Number, *Type:* Double

Comment: Decimals, *Type:* Whole number (like 1, 132, 2344)

Examples

DOLLAR(1403.77) returns "\$ 1,403.77"

Examples

DOLLAR(-0.123;4) returns "\$-0.1230"

8.1.12.9 EXACT

The EXACT() function returns True if these two strings are equal. Otherwise, it returns False.

Return type: A truth value (TRUE or FALSE)

Syntax

EXACT(string1;string2)

Parameters

Comment: String, *Type:* Text

Comment: String, *Type:* Text

Examples

EXACT("Calligra";"Calligra") returns True

Examples

EXACT("KSpread";"Calligra") returns False

Related Functions

[COMPARE](#)

8.1.12.10 FIND

The FIND() function finds one text string (find_text) within another text string (within_text) and returns the number of the starting point of find_text, from the leftmost character of within_text.

Parameter start_num specifies the character at which to start the search. The first character is character number 1. If start_num is omitted, it is assumed to be 1.

You can also use function SEARCH, but unlike SEARCH, FIND is case-sensitive and does not allow wildcard characters.

Return type: Whole number (like 1, 132, 2344)

Syntax

FIND(find_text;within_text;start_num)

Parameters

Comment: The text you want to find, *Type:* Text

Comment: The text which may contain find_text, *Type:* Text

Comment: Specifies index to start the search, *Type:* Whole number (like 1, 132, 2344)

Examples

FIND("Cal";"Calligra") returns 1

Examples

FIND("i";"Calligra") returns 5

Examples

FIND("a";"Sheets in Calligra";4) returns 12

Related Functions

[FINDB](#)

[SEARCH](#)

[REPLACE](#)

[SEARCHB](#)

[REPLACEB](#)

8.1.12.11 FINDB

The FINDB() function finds one text string (find_text) within another text string (within_text) and returns the number of the starting point of find_text, from the leftmost character of within_text using byte positions.

Parameter BytePosition specifies the character at which to start the search. The first character is character number 2. If start_num is omitted, it is assumed to be 2.

Return type: Whole number (like 1, 132, 2344)

Syntax

FINDB(find_text;within_text;BytePosition Start)

Parameters

Comment: The text you want to find, *Type:* Text

Comment: The text which may contain find_text, *Type:* Text

Comment: Specifies byte position to start the search, *Type:* Whole number (like 1, 132, 2344)

Related Functions

[FIND](#)
[SEARCH](#)
[REPLACE](#)
[SEARCHB](#)
[REPLACEB](#)

8.1.12.12 FIXED

The FIXED() function rounds a number to the specified number of decimals, formats the number in decimal format string, and returns the result as text. If decimals is negative, number is rounded to the left of the decimal point. If you omit decimals, it is assumed to be 2. If optional parameter no_commas is True, thousand separators will not show up.

Return type: Text

Syntax

FIXED(number;decimals;no_commas)

Parameters

Comment: Number, *Type:* Double

Comment: Decimals, *Type:* Whole number (like 1, 132, 2344)

Comment: No_commas, *Type:* A truth value (TRUE or FALSE)

Examples

FIXED(1234.567;1) returns "1,234.6"

Examples

FIXED(1234.567;1;FALSE) returns "1234.6"

Examples

FIXED(44.332) returns "44.33"

8.1.12.13 JIS

The JIS() function returns the full-width characters corresponding to the half-width argument.

Return type: Text

Syntax

JIS(text)

Parameters

Comment: Half-width characters, *Type:* Text

Related Functions

[ASC](#)

8.1.12.14 LEFT

The LEFT() function returns a substring that contains the 'length' leftmost characters of the string. The whole string is returned if 'length' exceeds the length of the string. It is an error for the number of characters to be less than 0.

Return type: Text

Syntax

LEFT(text;length)

Parameters

Comment: Source string, *Type:* Text

Comment: Number of characters, *Type:* Whole number (like 1, 132, 2344)

Examples

LEFT("hello";2) returns "he"

Examples

LEFT("KSpread";10) returns "KSpread"

Examples

LEFT("KSpread") returns "K"

Related Functions

[RIGHT](#)

[MID](#)

[RIGHTB](#)

[MIDB](#)

8.1.12.15 LEFTB

The LEFTB() function returns a substring that contains the 'length' leftmost characters of the string using byte positions. The whole string is returned if 'length' exceeds the length of the string. It is an error for the number of characters to be less than 0.

Return type: Text

Syntax

LEFTB(text;ByteLength)

Parameters

Comment: Source string, *Type:* Text

Comment: Byte Length, *Type:* Whole number (like 1, 132, 2344)

Related Functions

[RIGHT](#)

[MID](#)

[RIGHTB](#)

[MIDB](#)

8.1.12.16 LEN

The LEN() function returns the length of the string.

Return type: Whole number (like 1, 132, 2344)

Syntax

LEN(text)

Parameters

Comment: String, *Type:* Text

Examples

LEN("hello") returns 5

Examples

LEN("KSpread") returns 7

Related Functions

[LENB](#)

8.1.12.17 LENB

The LENB() function returns the length of the string using byte positions.

Return type: Whole number (like 1, 132, 2344)

Syntax

LENB(text)

Parameters

Comment: String, *Type:* Text

8.1.12.18 LOWER

The LOWER() function converts a string to lower case.

Return type: Text

Syntax

LOWER(text)

Parameters

Comment: Source string, *Type:* Text

Examples

LOWER("hello") returns "hello"

Examples

LOWER("HELLO") returns "hello"

Related Functions

[UPPER](#)
[TOGGLE](#)

8.1.12.19 MID

The MID() function returns a substring that contains 'length' characters of the string, starting at 'position' index.

Return type: Text

Syntax

MID(text;position;length)

Parameters

Comment: Source string, *Type:* Text

Comment: Position, *Type:* Whole number (like 1, 132, 2344)

Comment: Length, *Type:* Whole number (like 1, 132, 2344)

Examples

MID("Calligra";2;3) returns "all"

Examples

MID("Calligra";2) returns "alligra"

Related Functions

[LEFT](#)

[RIGHT](#)

[LEFTB](#)

[RIGHTB](#)

[MIDB](#)

8.1.12.20 MIDB

The MIDB() function returns a substring that contains 'length' characters of the string, starting at 'position' index using byte positions.

Return type: Text

Syntax

MIDB(text;BytePosition Start;ByteLength)

Parameters

Comment: Source string, *Type:* Text

Comment: Byte Position, *Type:* Whole number (like 1, 132, 2344)

Comment: Byte Length, *Type:* Whole number (like 1, 132, 2344)

Related Functions

[LEFT](#)

[RIGHT](#)

[LEFTB](#)

[RIGHTB](#)

[MID](#)

8.1.12.21 PROPER

The PROPER() function converts the first letter of each word to uppercase and the rest of the letters to lowercase.

Return type: Text

Syntax

PROPER(string)

Parameters

Comment: String, *Type:* Text

Examples

PROPER("this is a title") returns "This Is A Title"

8.1.12.22 REGEXP

Returns a part of the string that matches a regular expression. If the string does not match the given regular expression, value specified as default is returned.

If a back-reference is provided, then the value of that back-reference is returned.

If no default value is given, an empty string is assumed. If no back-reference is given, 0 is assumed (so that entire matching part is returned).

Return type: Text

Syntax

REGEXP(text; regexp; default; backref)

Parameters

Comment: Searched text, *Type:* Text

Comment: Regular expression, *Type:* Text

Comment: Default value (optional), *Type:* Text

Comment: Back-reference (optional), *Type:* Number

Examples

REGEXP("Number is 15. "; "[0-9]+") = "15"

Examples

REGEXP("15, 20, 26, 41"; "([0-9]+), *[0-9]+\$"; ""; 1) = "26"

8.1.12.23 REGEXPRE

Replaces all matches of a regular expression with the replacement text

Return type: Text

Syntax

REGEXPRE(text; regexp; replacement)

Parameters

Comment: Searched text, *Type:* Text

Comment: Regular expression, *Type:* Text

Comment: Replacement, *Type:* Text

Examples

REGEXPRE("14 and 15 and 16"; "[0-9]+"; "num") returns "num and num and num"

8.1.12.24 REPLACE

The REPLACE() function replaces part of a text string with a different text string.

Return type: Text

Syntax

REPLACE(text;position;length;new_text)

Parameters

Comment: Text which you want to replace some characters, *Type:* Text

Comment: Position of the characters to replace, *Type:* Whole number (like 1, 132, 2344)

Comment: Number of characters to replace, *Type:* Whole number (like 1, 132, 2344)

Comment: The text that will replace characters in old text, *Type:* Text

Examples

REPLACE("abcdefghijk";6;5;"-") returns "abcde-k"

Examples

REPLACE("2002";3;2;"03") returns "2003"

Related Functions

[FIND](#)

[MID](#)

[FINDB](#)

[MIDB](#)

8.1.12.25 REPLACEB

The REPLACEB() function replaces part of a text string with a different text string using byte positions.

Return type: Text

Syntax

REPLACEB(text;BytePosition;ByteLength Len;new_text)

Parameters

Comment: Text which you want to replace some characters using byte position, *Type:* Text

Comment: Byte position of the characters to replace, *Type:* Whole number (like 1, 132, 2344)

Comment: The byte length of characters to replace, *Type:* Whole number (like 1, 132, 2344)

Comment: The text that will replace characters in old text, *Type:* Text

Related Functions

[FINDB](#)

[MIDB](#)

[FIND](#)

[MID](#)

8.1.12.26 REPT

The REPT() function repeats the first parameter as many times as by the second parameter. The second parameter must not be negative, and this function will return an empty string if the second parameter is zero (or rounds down to zero).

Return type: Text

Syntax

REPT(text;count)

Parameters

Comment: Source string, *Type:* Text

Comment: Count of repetitions, *Type:* Whole number (like 1, 132, 2344)

Examples

REPT("KSpread";3) returns "KSpreadKSpreadKSpread"

Examples

REPT("KSpread";0) returns ""

8.1.12.27 RIGHT

The RIGHT() function returns a substring that contains the 'length' rightmost characters of the string. The whole string is returned if 'length' exceeds the length of the string.

Return type: Text

Syntax

RIGHT(text;length)

Parameters

Comment: Source string, *Type:* Text

Comment: Number of characters, *Type:* Whole number (like 1, 132, 2344)

Examples

RIGHT("hello";2) returns "lo"

Examples

RIGHT("KSpread";10) returns "KSpread"

Examples

RIGHT("KSpread") returns "d"

Related Functions

LEFT
MID
LEFTB
MIDB

8.1.12.28 RIGHTB

The RIGHTB() function returns a substring that contains the 'length' rightmost characters of the string using byte positions. The whole string is returned if 'length' exceeds the length of the string.

Return type: Text

Syntax

RIGHTB(text;ByteLength)

Parameters

Comment: Source string, *Type:* Text

Comment: Byte Length, *Type:* Whole number (like 1, 132, 2344)

Related Functions

[LEFT](#)
[MID](#)
[LEFTB](#)
[MIDB](#)

8.1.12.29 ROT13

The ROT13() function encrypts text by replacing each letter with the one 13 places along in the alphabet. If the 13th position is beyond the letter Z, it begins again at A (rotation).

By applying the encryption function again to the resulting text, you can decrypt the text.

Return type: Text

Syntax

ROT13(Text)

Parameters

Comment: Text, *Type:* Text

Examples

ROT13("KSpread") returns "XFcernq"

Examples

ROT13("XFcernq") returns "KSpread"

8.1.12.30 SEARCH

The SEARCH() function finds one text string (find_text) within another text string (within_text) and returns the number of the starting point of find_text, from the leftmost character of within_text.

You can use wildcard characters, question mark (?) and asterisk (*). A question mark matches any single character, an asterisk matches any sequences of characters.

Parameter start_num specifies the character at which to start the search. The first character is character number 1. If start_num is omitted, it is assumed to be 1. SEARCH does not distinguish between uppercase and lowercase letters.

Return type: Whole number (like 1, 132, 2344)

Syntax

SEARCH(find_text;within_text;start_num)

Parameters

Comment: The text you want to find, *Type:* Text

Comment: The text which may contain find_text, *Type:* Text

Comment: Specified index to start the search, *Type:* Whole number (like 1, 132, 2344)

Examples

SEARCH("e";"Statements";6) returns 7

Examples

SEARCH("margin";"Profit Margin") returns 8

Related Functions

[FIND](#)
[FINDB](#)
[SEARCHB](#)

8.1.12.31 SEARCHB

The SEARCHB() function finds one text string (find_text) within another text string (within_text) and returns the number of the starting point of find_text, from the leftmost character of within_text using byte positions.

You can use wildcard characters, question mark (?) and asterisk (*). A question mark matches any single character, an asterisk matches any sequences of characters.

Parameter BytePosition specifies the character at which to start the search. The first character is character number 2. If BytePosition is omitted, it is assumed to be 2. SEARCHB does not distinguish between uppercase and lowercase letters.

Return type: Whole number (like 1, 132, 2344)

Syntax

SEARCHB(find_text;within_text;BytePosition Start)

Parameters

Comment: The text you want to find, *Type:* Text

Comment: The text which may contain find_text, *Type:* Text

Comment: Specified byte position to start the search, *Type:* Whole number (like 1, 132, 2344)

Related Functions

[FINDB](#)
[FIND](#)
[SEARCH](#)

8.1.12.32 SLEEK

The SLEEK() function removes all spaces from the string.

Return type: Text

Syntax

SLEEK(text)

Parameters

Comment: Source string, *Type:* Text

Examples

SLEEK("This is some text ") returns "Thisissometext"

Related Functions

[TRIM](#)

8.1.12.33 SUBSTITUTE

The SUBSTITUTE() substitutes new_text for old_text in a text string. If instance_num is specified, only that instance of old_text is replaced. Otherwise, every occurrence of old_text is changed to new_text. Use SUBSTITUTE when you want to replace specific text, use REPLACE when you want to replace any text that occurs in a specific location.

Return type: Text

Syntax

SUBSTITUTE(text; old_text; new_text; instance_num)

Parameters

Comment: Text for which you want to substitute, *Type:* Text

Comment: Part of text you want to replace, *Type:* Text

Comment: New text which will be replacement, *Type:* Text

Comment: Which occurrence to replace, *Type:* Whole number (like 1, 132, 2344)

Examples

SUBSTITUTE("Cost Data";"Cost";"Sales") returns "Sales Data"

Examples

SUBSTITUTE("Qtr 1, 2001";"1";"3";1) returns "Qtr 3, 2001"

Examples

SUBSTITUTE("Qtr 1, 2001";"1";"3";4) returns "Qtr 3, 2003"

Related Functions

[REPLACE](#)

[REPLACEB](#)

[FIND](#)

[FINDB](#)

8.1.12.34 T

The T() function returns the text referred to by value. If value is, or refers to, text then T returns value. If value does not refer to text then T returns empty text.

Return type: Text

Syntax

T(value)

Parameters

Comment: Value, *Type:* Any kind of value

Examples

T("Calligra") returns "Calligra"

Examples

T(1.2) returns "" (empty text)

8.1.12.35 TEXT

The TEXT() function converts a value to text.

Return type: Text

Syntax

TEXT(value)

Parameters

Comment: Value, *Type:* Any kind of value

Examples

TEXT(1234.56) returns "1234.56"

Examples

TEXT("KSpread") returns "KSpread"

8.1.12.36 TOGGLE

The TOGGLE() function changes lowercase characters to uppercase and uppercase characters to lowercase.

Return type: Text

Syntax

TOGGLE(text)

Parameters

Comment: Source string, *Type:* Text

Examples

TOGGLE("hello") returns "HELLO"

Examples

TOGGLE("HELLO") returns "hello"

Examples

TOGGLE("HeLlO") returns "hElLo"

Related Functions

[UPPER](#)

[LOWER](#)

8.1.12.37 TRIM

The TRIM() function returns text with only single spaces between words.

Return type: Text

Syntax

TRIM(text)

Parameters

Comment: String, *Type:* Text

Examples

TRIM(" hello KSpread ") returns "hello KSpread"

8.1.12.38 UNICHAR

The UNICHAR() function returns the character specified by a unicode code point.

Return type: Text

Syntax

UNICHAR(code)

Parameters

Comment: Character code, *Type:* Whole number (like 1, 132, 2344)

Examples

UNICHAR(65) returns "A"

Related Functions

[UNICODE](#)
[CHAR](#)

8.1.12.39 UNICODE

The UNICODE() function returns a unicode code point for the first character in a text string.

Return type: Whole number (like 1, 132, 2344)

Syntax

UNICODE(text)

Parameters

Comment: Text, *Type:* Text

Examples

UNICODE("KDE") returns 75

Related Functions

[UNICHAR](#)
[CODE](#)

8.1.12.40 UPPER

The UPPER() function converts a string to upper case.

Return type: Text

Syntax

UPPER(text)

Parameters

Comment: Source string, *Type:* Text

Examples

UPPER("hello") returns "HELLO"

Examples

UPPER("HELLO") returns "HELLO"

Related Functions

[LOWER](#)
[TOGGLE](#)

8.1.12.41 VALUE

Converts text string that represents a value to the real value.

Return type: Double

Syntax

VALUE(text)

Parameters

Comment: Text, *Type:* Text

Examples

VALUE("14.03") returns 14.03

8.1.13 Trigonometric

8.1.13.1 ACOS

The ACOS() function returns the arc cosine in radians and the value is mathematically defined to be 0 to PI (inclusive).

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

ACOS(Float)

Parameters

Comment: Angle (radians), *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

ACOS(0.8) equals 0.6435011

Examples

ACOS(0) equals 1.57079633

Related Functions

[COS](#)

8.1.13.2 ACOSH

The ACOSH() function calculates the inverse hyperbolic cosine of x. That is the value whose hyperbolic cosine is x. If x is less than 1.0, acosh() returns not-a-number (NaN) and errno is set.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

ACOSH(Float)

Parameters

Comment: Angle (radians), *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

ACOSH(5) equals 2.29243167

Examples

ACOSH(0) equals NaN

Related Functions

[COSH](#)

8.1.13.3 ACOT

The ACOT() function returns the inverse cotangent of a number.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

ACOT(Float)

Parameters

Comment: Angle (radians), *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

ACOT(0) equals 1.57079633

8.1.13.4 ASIN

The ASIN() function returns the arc sine in radians and the value is mathematically defined to be $-\pi/2$ to $\pi/2$ (inclusive).

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

ASIN(Float)

Parameters

Comment: Angle (radians), *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

ASIN(0.8) equals 0.92729522

Examples

ASIN(0) equals 0

Related Functions

[SIN](#)

8.1.13.5 ASINH

The ASINH() function calculates the inverse hyperbolic sine of x; that is the value whose hyperbolic sine is x.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

ASINH(Float)

Parameters

Comment: Angle (radians), *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

ASINH(0.8) equals 0.73266826

Examples

ASINH(0) equals 0

Related Functions

[SINH](#)

8.1.13.6 ATAN

The ATAN() function returns the arc tangent in radians and the value is mathematically defined to be $-\pi/2$ to $\pi/2$ (inclusive).

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

ATAN(Float)

Parameters

Comment: Angle (radians), *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

ATAN(0.8) equals 0.67474094

Examples

ATAN(0) equals 0

Related Functions

[TAN](#)
[ATAN2](#)

8.1.13.7 ATAN2

This function calculates the arc tangent of the two variables x and y. It is similar to calculating the arc tangent of y/x , except that the signs of both arguments are used to determine the quadrant of the result.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

ATAN2(value;value)

Parameters

Comment: Angle (radians), *Type:* A floating point value (like 1.3, 0.343, 253)

Comment: Angle (radians), *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

ATAN2(0.5;1.0) equals 1.107149

Examples

ATAN2(-0.5;2.0) equals 1.815775

Related Functions

[ATAN](#)

8.1.13.8 ATANH

The ATANH() function calculates the inverse hyperbolic tangent of x ; that is the value whose hyperbolic tangent is x . If the absolute value of x is greater than 1.0, ATANH() returns not-a-number (NaN).

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

ATANH(Float)

Parameters

Comment: Angle (radians), *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

ATANH(0.8) equals 1.09861229

Examples

ATANH(0) equals 0

Related Functions

[TANH](#)

8.1.13.9 COS

The COS() function returns the cosine of x , where x is given in radians.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

COS(Float)

Parameters

Comment: Angle (radians), *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

COS(0) equals 1.0

Examples

COS(PI()/2) equals 0

Related Functions

[SIN](#)
[ACOS](#)

8.1.13.10 COSH

The COSH() function returns the hyperbolic cosine of x , which is defined mathematically as $(\exp(x) + \exp(-x)) / 2$.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

COSH(Float)

Parameters

Comment: Angle (radians), *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

COSH(0.8) equals 1.33743495

Examples

COSH(0) equals 1

Related Functions

[ACOSH](#)

8.1.13.11 CSC

The CSC() function returns the cosecant of x, where x is given in radians.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

CSC(Float)

Parameters

Comment: Angle (radians), *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

CSC(PI()/2) equals 1

8.1.13.12 CSCH

The CSCH() function returns the hyperbolic cosecant of x, where x is given in radians.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

CSCH(Float)

Parameters

Comment: Angle (radians), *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

CSCH(PI()/2) equals 0.434537208...

8.1.13.13 DEGREES

This function transforms a radian angle to a degree angle.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

DEGREES(Float)

Parameters

Comment: Angle (radians), *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

DEGREES(0.78) equals 44.69

Examples

DEGREES(1) equals 57.29

Related Functions

[RADIANS](#)

8.1.13.14 PI

The PI() function returns the value of PI.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

PI()

Parameters

Examples

PI() equals 3.141592654...

8.1.13.15 RADIANS

This function transforms a degree angle to a radian angle.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

RADIANS(Float)

Parameters

Comment: Angle (degrees), *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

RADIANS(75) equals 1.308

Examples

RADIANS(90) equals 1.5707

Related Functions

[DEGREES](#)

8.1.13.16 SEC

The SEC() function returns the secant of x, where x is given in radians.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

SEC(Float)

Parameters

Comment: Angle (radians), *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

SEC(0) equals 1

8.1.13.17 SECH

The SECH() function returns the hyperbolic secant of x , where x is given in radians.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

SECH(Float)

Parameters

Comment: Angle (radians), *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

SECH(0) equals 1

8.1.13.18 SIN

The SIN() function returns the sine of x , where x is given in radians.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

SIN(Float)

Parameters

Comment: Angle (radians), *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

SIN(0) equals 0

Examples

SIN(PI()/2) equals 1

Related Functions

[COS](#)

[ASIN](#)

8.1.13.19 SINH

The SINH() function returns the hyperbolic sine of x , which is defined mathematically as $(\exp(x) - \exp(-x)) / 2$.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

SINH(Float)

Parameters

Comment: Angle (radians), *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

SINH(0.8) equals 0.88810598

Examples

SINH(0) equals 0

Related Functions

[ASINH](#)

8.1.13.20 TAN

The TAN() function returns the tangent of x , where x is given in radians.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

TAN(Float)

Parameters

Comment: Angle (radians), *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

TAN(0.7) equals 0.84228838

Examples

TAN(0) equals 0

Related Functions

[ATAN](#)

8.1.13.21 TANH

The TANH() function returns the hyperbolic tangent of x , which is defined mathematically as $\sinh(x)/\cosh(x)$.

Return type: A floating point value (like 1.3, 0.343, 253)

Syntax

TANH(Float)

Parameters

Comment: Angle (radians), *Type:* A floating point value (like 1.3, 0.343, 253)

Examples

TANH(0.8) equals 0.66403677

Examples

TANH(0) equals 0

Related Functions

[ATANH](#)

Chapter 9

Credits and License

Calligra Sheets

Program copyright 1998-2019 The Calligra Sheets Team:

- Torben Weis weis@kde.org
- Laurent Montel lmontel@mandrakesoft.com
- David Faure faure@kde.org
- John Dailey dailey@vt.edu
- Philipp Müller philipp.mueller@gmx.de
- Ariya Hidayat ariya@kde.org
- Norbert Andres nandres@web.de
- Shaheed Haque srhaque@iee.org
- Werner Trobin trobin@kde.org
- Nikolas Zimmermann wildfox@kde.org
- Helge Deller deller@kde.org
- Percy Leonhart percy@eris23.org
- Eva Brucherseifer eva@kde.org
- Phillip Ezolt phillipezolt@hotmail.com
- Enno Bartels ebartels@nwn.de
- Graham Short grahshrt@netscape.net

Documentation copyright 2002 Pamela Roberts pamroberts@blueyonder.co.uk

Minor updates to documentation for KOffice 1.3 by Philip Rodrigues phil@kde.org.

Screenshot updates for Calligra 3.1 by Carl Schwan carl@carlschwan.eu

This documentation is licensed under the terms of the [GNU Free Documentation License](#).

This program is licensed under the terms of the [GNU General Public License](#).