

Lotus

@Functions and Macros Guide

Lotus 1-2-3 Release 2.3

Copyright

Under the copyright laws, neither the documentation nor the software may be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine readable form, in whole or part, without the prior written consent of Lotus Development Corporation, except in the manner described in the documentation.

© Copyright 1991 Lotus Development Corporation
55 Cambridge Parkway
Cambridge, MA 02142

All rights reserved. First Edition Printed 1991. Printed in Ireland.

Lotus and 1-2-3 are registered trademarks of Lotus Development Corporation.

Contents

| | |
|---|-----------|
| How to Use This Book | ix |
| Who Should Use This Book | ix |
| Conventions | ix |
| Organization | x |
| | |
| Chapter 1 @Function Basics | 1 |
| What Is an @Function? | 1 |
| @Function Definitions | 1 |
| @Function Format and Rules | 2 |
| @Function Rules | 3 |
| @Function Help | 3 |
| Argument Types | 3 |
| @Function Types | 4 |
| Add-In @Functions | 5 |
| To Use Add-In @Functions | 5 |
| Database Statistical @Functions | 5 |
| Date and Time @Functions | 7 |
| Financial @Functions | 8 |
| Logical @Functions | 9 |
| Mathematical @Functions | 10 |
| Special @Functions | 11 |
| Statistical @Functions | 12 |
| String @Functions | 13 |
| | |
| Chapter 2 @Function Descriptions | 15 |
| @@ | 15 |
| @? | 16 |
| @ABS | 17 |
| @ACOS | 17 |
| @ASIN | 18 |
| @ATAN | 19 |
| @ATAN2 | 19 |
| @AVG | 21 |
| @CELL | 22 |
| @CELLPOINTER | 23 |
| @CHAR | 24 |
| @CHOOSE | 25 |
| @CLEAN | 25 |
| @CODE | 26 |
| @COLS | 27 |
| @COS | 27 |
| @COUNT | 28 |
| @CTERM | 29 |
| @DATE | 30 |

| | |
|------------|----|
| @DATEVALUE | 31 |
| @DAVG | 31 |
| @DAY | 32 |
| @DCOUNT | 33 |
| @DDB | 34 |
| @DMAX | 35 |
| @DMIN | 36 |
| @DSTD | 37 |
| @DSUM | 39 |
| @DVAR | 40 |
| @ERR | 42 |
| @EXACT | 42 |
| @EXP | 43 |
| @FALSE | 44 |
| @FIND | 44 |
| @FV | 45 |
| @HLOOKUP | 46 |
| @HOUR | 47 |
| @IF | 48 |
| @INDEX | 49 |
| @INT | 50 |
| @IRR | 50 |
| @ISAAF | 51 |
| @ISAPP | 52 |
| @ISERR | 53 |
| @ISNA | 53 |
| @ISNUMBER | 54 |
| @ISSTRING | 55 |
| @LEFT | 56 |
| @LENGTH | 56 |
| @LN | 57 |
| @LOG | 57 |
| @LOWER | 58 |
| @MAX | 58 |
| @MID | 59 |
| @MIN | 60 |
| @MINUTE | 60 |
| @MOD | 61 |
| @MONTH | 62 |
| @N | 62 |
| @NA | 63 |
| @NOW | 63 |
| @NPV | 64 |
| @PI | 65 |
| @PMT | 66 |
| @PROPER | 67 |
| @PV | 67 |

| | |
|-------------------------------|-----------|
| @RAND | 69 |
| @RATE | 69 |
| @REPEAT | 70 |
| @REPLACE | 71 |
| @RIGHT | 71 |
| @ROUND | 72 |
| @ROWS | 73 |
| @S | 74 |
| @SECOND | 74 |
| @SIN | 75 |
| @SLN | 75 |
| @SQRT | 76 |
| @STD | 77 |
| @STRING | 78 |
| @SUM | 79 |
| @SYD | 80 |
| @TAN | 81 |
| @TERM | 81 |
| @TIME | 83 |
| @TIMEVALUE | 83 |
| @TRIM | 84 |
| @TRUE | 85 |
| @UPPER | 85 |
| @VALUE | 86 |
| @VAR | 87 |
| @VLOOKUP | 88 |
| @YEAR | 90 |
| Chapter 3 Macro Basics | 91 |
| What Is a Macro? | 91 |
| Macro Definitions | 91 |
| Macro Format and Rules | 92 |
| Keystrokes | 93 |
| Macro Commands | 93 |
| Macro Command Rules | 94 |
| Argument Types | 94 |
| Macro Location | 95 |
| Creating a Macro | 96 |
| To Create a Macro | 96 |
| Tips for Creating a Macro | 97 |
| Naming a Macro | 98 |
| To Name a Macro | 98 |
| Documenting a Macro | 99 |
| Running a Macro | 99 |
| To Run a Backslash Macro | 100 |
| To Run a Range Name Macro | 100 |
| Auto-execute Macros | 100 |

| | |
|--|------------|
| Canceling a Macro | 101 |
| Recalculation During Macros | 101 |
| Dialog Boxes in Macros | 101 |
| Debugging a Macro | 101 |
| Troubleshooting Checklist | 102 |
| Debugging a Macro in STEP Mode | 102 |
| To Use STEP Mode | 103 |
| Creating a Macro with the Learn Feature | 104 |
| To Record Keystrokes | 104 |
| To Edit Keystrokes | 105 |
| To Name and Run a Macro Created with Learn | 105 |
| Macro Command Categories | 106 |
| Data Manipulation | 106 |
| File Manipulation | 107 |
| Flow-Of-Control | 107 |
| Interactive | 108 |
| Key Names | 109 |
| Screen Control | 110 |
| | |
| Chapter 4 Macro Command Descriptions | 111 |
| {?} | 111 |
| {~} | 112 |
| {() and {} | 112 |
| {ABS} | 112 |
| {APP1}, {APP2}, {APP3}, and {APP4} | 113 |
| {APPENDBELOW} and {APPENDRIGHT} | 113 |
| {BACKSPACE} and {BS} | 115 |
| {BEEP} | 115 |
| {BIGLEFT} and {BIGRIGHT} | 116 |
| {BLANK} | 116 |
| {BORDERSOFF} and {BORDERSON} | 117 |
| {BRANCH} | 117 |
| {BREAK} | 118 |
| {BREAKOFF} and {BREAKON} | 118 |
| {CALC} | 119 |
| {CLOSE} | 120 |
| {CONTENTS} | 120 |
| {DEFINE} | 122 |
| {DELETE} and {DEL} | 124 |
| {DISPATCH} | 124 |
| {DOWN} and {D} | 125 |
| {EDIT} | 125 |
| {END} | 125 |
| {ESCAPE} and {ESC} | 125 |
| {FILESIZE} | 125 |
| {FOR} | 126 |
| {FORBREAK} | 127 |

| | |
|------------------------------------|-----|
| {FORM} | 128 |
| Suspending a {FORM} Command | 130 |
| {FORMBREAK} | 132 |
| {FRAMEOFF} and {FRAMEON} | 133 |
| {GET} | 134 |
| {GETLABEL} | 135 |
| {GETNUMBER} | 136 |
| {GETPOS} | 137 |
| {GOTO} | 138 |
| {GRAPH} | 138 |
| {GRAPHOFF} and {GRAPHON} | 138 |
| {HELP} | 139 |
| {HOME} | 139 |
| {IF} | 140 |
| {INDICATE} | 141 |
| {INSERT} and {INS} | 142 |
| {LEFT} and {L} | 142 |
| {LET} | 142 |
| {LOOK} | 143 |
| {MENU} | 144 |
| {MENUBRANCH} and {MENUCALL} | 144 |
| Creating a Macro Menu | 146 |
| {NAME} | 147 |
| {ONERROR} | 147 |
| {OPEN} | 148 |
| {PANELOFF} and {PANELON} | 150 |
| {PGDN} and {PGUP} | 151 |
| {PUT} | 151 |
| {QUERY} | 152 |
| {QUIT} | 152 |
| {READ} | 152 |
| {READLN} | 153 |
| {RECALC} and {RECALCCOL} | 154 |
| {RESTART} | 156 |
| {RETURN} | 157 |
| {RIGHT} and {R} | 157 |
| {SETPOS} | 158 |
| { <i>subroutine</i> } | 158 |
| {SYSTEM} | 160 |
| {TABLE} | 161 |
| {UP} and {U} | 161 |
| {WAIT} | 162 |
| {WINDOW} | 163 |
| {WINDOWSOFF} and {WINDOWSON} | 163 |
| {WRITE} | 164 |
| {WRITELN} | 165 |
| The /X Macro Commands | 166 |

| | |
|---|------------|
| Chapter 5 Sample Macros | 167 |
| Using the Sample Macros | 167 |
| To Use a Sample Macro | 167 |
| Goto Macro (\G) | 168 |
| To Use Macro \G | 168 |
| Row-Shifting Macro (\S) | 168 |
| To Use Macro \S | 169 |
| Date Macro (\D) | 169 |
| To Use Macro \D | 169 |
| Rounding Macro (\R) | 170 |
| To Use Macro \R | 171 |
| Column Macro (\C) | 172 |
| To Use Macro \C | 172 |
| Mailing Labels Macro (\M) | 174 |
| To Use Macro \M | 174 |
| | |
| Chapter 6 Using the Macro Library Manager Add-In | 177 |
| What Is a Macro Library? | 177 |
| Starting Macro Library Manager | 178 |
| To Attach Macro Library Manager | 178 |
| To Invoke Macro Library Manager | 178 |
| Rules for Using a Macro Library | 179 |
| When to Attach and Detach Macro Library Manager | 179 |
| Memory Management | 179 |
| Duplicate Library Names | 179 |
| Ranges with Links to Other Files | 179 |
| Rules for Macro Commands in a Macro Library | 180 |
| Range Names | 180 |
| Executing Subroutines and Menus in a Library | 180 |
| Macro Commands that Reference Data | 181 |
| Macro Commands that Contain Formulas | 181 |
| Recalculation of a Formula Within a Library | 182 |
| Libraries that Contain /File Retrieve Commands | 182 |
| Creating a Macro Library | 182 |
| To Create a Macro Library | 182 |
| Saving Macros in a Library | 183 |
| To Save Macros in a Library | 183 |
| Using a Macro in a Library | 184 |
| To Use a Macro in a Library | 184 |
| Making Changes to Macros in a Library | 185 |
| To Change Macros in a Library | 185 |
| Removing a Macro Library from Memory | 186 |
| To Remove a Macro Library from Memory | 186 |
| Using Macro Library Manager on a Network | 186 |
| Macro Library Command Summary | 187 |
| | |
| Index | 189 |

How to Use This Book

Use the *@Functions and Macros Guide* to learn about Lotus® 1-2-3® @functions and macros. It provides examples of how you can use @functions to perform a variety of calculations and how you can use macros to automate your work.

Who Should Use This Book

The *@Functions and Macros Guide* is designed for readers who have a general knowledge of 1-2-3 procedures and concepts.

Conventions

The following conventions are used throughout the *@Functions and Macros Guide*:

- @Function names and macro keywords are in uppercase letters, but you can use uppercase or lowercase letters.

Example: @SUM(A2..A15)

- Arguments for @functions and macros are in lowercase italics.

Example: @TIME(*hour,minutes,seconds*)

- Optional arguments for @functions and macros are in [] (brackets).

Example: {RECALC *location*,*[condition]*,*[iterations]*}

- Function keys and special keys are in small capitals. Keys are identified by the appropriate key sequence, followed by the 1-2-3 key name.

Example: F1 (HELP)

- Key names separated by a - (hyphen) indicate that you must press and hold down the first key, press the second key, and then release both keys.


Example: CTRL- →

- Key names separated by a space indicate that you must press the first key and release it, and then press the second key and release it.

Example: END HOME

- Information that you type appears in a different typeface.

Example: Expenses

 Indicates information or instructions for network users.

Words that are in bold are defined in text where they appear.

Organization

This book contains six chapters:

- Chapter 1, “@Function Basics,” provides basic rules for entering @functions and describes the different types of @functions in 1-2-3.
- Chapter 2, “@Function Descriptions,” contains descriptions and examples of each @function, arranged alphabetically.
- Chapter 3, “Macro Basics,” introduces macros and explains the basic procedures for creating and running macros.
- Chapter 4, “Macro Command Descriptions,” contains descriptions and examples of each macro command, arranged alphabetically.
- Chapter 5, “Sample Macros,” contains several short, general-purpose macros that illustrate macro concepts and techniques. These macros are in a worksheet file named SAMPMACS.WK1. The Install program transferred this worksheet to your 1-2-3 program directory during installation.
- Chapter 6, “Using the Macro Library Manager Add-In,” explains how to create libraries of macros for use with any worksheet.

Chapter 1

@Function Basics

This chapter provides basic rules for entering @functions and describes the different types of @functions in 1-2-3. Chapter 2, beginning on page 15, describes each @function and its arguments, and provides an example of its use.

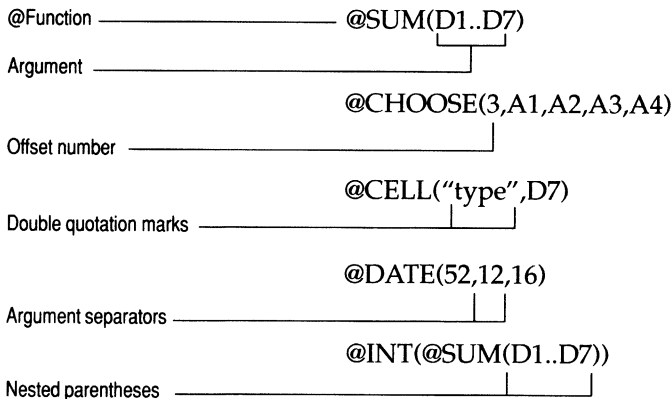
What Is an @Function?

An **@function** is a built-in formula in 1-2-3 that performs a specialized calculation automatically. Some @functions perform simple calculations. For example, @SUM adds the values in a range. @SUM(D1..D7) adds the values in the range D1..D7. Using @SUM is easier than writing out the formula +D1+D2+D3+D4+D5+D6+D7. Other @functions replace complex formulas. For example, @NPV calculates the net present value of a series of future cash-flow values.

You can use an @function by itself as a formula, combine it with other @functions and formulas, or use it in a macro.

@Function Definitions

The following examples show the different elements you use when you enter @functions.



Argument is data you provide for 1-2-3 to use when it calculates the @function. The arguments you provide depend on the @function you use.

Argument separators separate two or more arguments. 1-2-3 allows three argument separators: , (comma), ; (semicolon), and . (period). You can always use a ; (semicolon) to separate arguments and, depending on the setting of /Worksheet Global Default Other International Punctuation, you can also use either . (period) or , (comma).

Double quotation marks (“ ”) enclose text in arguments in string @functions. 1-2-3 assumes that text not in double quotation marks is a range name.

Offset number corresponds to the position of a character in text or a field or row in a database. Offset numbers start at zero. For example, in the label MAHER, the offset number of M is 0, of A is 1, of H is 2, and so on. A field’s offset number corresponds to the position of the column the field occupies in the input range (or database). The first (leftmost) field of the input range (or database) has an offset number of 0, the second field has an offset number of 1, and so on. String and database statistical @functions use offset numbers.

Parentheses enclose arguments. If you use an @function as an argument for another @function, you must nest the parentheses: enclose the @function you are using as an argument within the parentheses of the primary @function. @INT(@SUM(D1..D7)), for example, uses @SUM and its argument as the argument for the primary @function @INT.

@Function Format and Rules

The format of an @function is

@FUNCTION

or

@FUNCTION(*argument1,argument2,...,argumentn*)

@FUNCTION is the name of the @function. The @function name tells 1-2-3 what action to perform. The @ (at sign) identifies the entry in the cell as an @function, rather than a label. You can type @functions in uppercase or lowercase letters, but this book refers to @function names in uppercase letters.

(*argument1,argument2,...,argumentn*) are the arguments for the @function. You can type arguments in uppercase or lowercase letters; this book refers to arguments in lowercase italics. Some @functions have optional arguments (arguments you can omit). This book shows optional arguments in [] (brackets). You enclose arguments in () (parentheses). @Functions that don’t require arguments are not followed by parentheses.

1-2-3 changes the @functions and arguments you enter (except text arguments) to uppercase letters.

@Function Rules

Observe the following rules when you write an @function:

- Enter each @function (@function name and arguments, if any) in a single cell. An @function and its arguments cannot exceed 240 characters. Enclose arguments in () (parentheses).
- Do not type spaces between arguments.
- Use , (comma), ; (semicolon), or . (period) to separate arguments. You can always use a ; (semicolon) to separate arguments and, depending on the setting of /Worksheet Global Default Other International Punctuation, you can also use either . (period) or , (comma).
- Do not use a comma, semicolon, period, or parenthesis as part of an argument, unless you enclose the argument in double quotation marks.
- You do not need to use + (plus) to enter an @function. For example, use @SUM(A1..A5)*2 instead of +@SUM(A1..A5)*2.

@Function Help

You can press F1 (HELP) when you are entering an @function to get information about the @function.

Argument Types

1-2-3 @functions accept four types of arguments.

| Type | Description |
|-----------|---|
| Condition | An expression that uses a relational or logical operator (< > = <> >= <= #NOT# #AND# and #OR#). The @function evaluates the condition argument and proceeds according to whether it is true or false. You can also use a formula or @function, a number, text, or a range name or cell address as a condition argument. |
| Location | The address or name of a cell or range, or a formula or @function that returns a range address or name. A location argument can refer to a single-cell or a multiple-cell range. |
| String | Text (any sequence of letters, numbers, and symbols) enclosed in double quotation marks, the range address or name of a cell that contains a label, or a formula or @function that returns a label. String @functions use text arguments. |
| Value | A number, the address or name of a cell that contains a number, or a formula or @function that returns a number. |

Keep the following information in mind when you use @functions that require arguments:

- Use range names to ensure that location arguments are correct even if you insert or delete rows or columns.
- If the argument you specify for an @function is a single-cell range and the cell is blank, 1-2-3 uses zero as the argument (except for @COUNT, where 1-2-3 uses the value 1 as the argument). Statistical @functions ignore blank cells in multiple-cell range arguments. (A **blank cell** is a cell that does not contain an entry or a label-prefix character.)
- Several @functions require a range for a location argument, but they use only the cell in the upper left corner. Other @functions that require a single cell allow a range only if it is a single-cell range.
- Two @functions (@CELL and @CELLPOINTER) require specific attributes, one of which you must use as the argument. Enclose the attribute in "" (double quotation marks), as you do with all text used as arguments. For more information about attributes, see the table that begins on page22.

@Function Types

1-2-3 has nine types of @functions.

| Type | Description |
|----------------------|---|
| Add-in | Increase the number of @functions you can use to work with data in 1-2-3. Add-in @functions are available from third-party software developers. |
| Database statistical | Perform statistical calculations and queries on a 1-2-3 database. |
| Date and time | Calculate values that represent dates and times. |
| Financial | Calculate values for loans, annuities, cash flow, and depreciation. |
| Logical | Calculate formulas based on conditions that are either true or false. |
| Mathematical | Calculate with numbers. |
| Special | Perform a variety of worksheet tasks, such as looking up a value in a table or providing information about a specific cell. |
| Statistical | Perform statistical analysis on data. |
| String | Evaluate and manipulate labels, text formulas, or text enclosed in double quotation marks. |

Add-In @Functions

Add-in @functions are @functions supplied by add-in programs. Add-in @functions increase the number of @functions you can use to work with data in 1-2-3. Once you use /Add-In Attach (see “Using an Add-In” in Chapter 2 of the *User’s Guide*), to attach an add-in program that has add-in @functions, you can use add-in @functions as you would any built-in 1-2-3 @function. When you use add-in @functions, follow the @function rules on page 3. For information on installing and using add-in @functions, refer to the documentation for your add-in program.

To Use Add-In @Functions

1. Start 1-2-3.
2. Use /Add-In Attach to attach the add-in program that provides the @functions.
3. Use /File Retrieve to retrieve the worksheet that contains the add-in @functions.

You can now use the add-in @functions as you would any built-in 1-2-3 @function. The add-in @functions remain in memory until you end the 1-2-3 session. You cannot detach add-in @functions to free memory during a session.

NOTE If you retrieve a worksheet that contains add-in @functions without first attaching the appropriate add-in, each add-in @function is displayed as @? in the control panel and returns the value NA in the worksheet. To correct this, you must attach the add-in and then retrieve the worksheet again.

Database Statistical @Functions

Database statistical @functions scan a database, select records that meet the criteria in a criteria range, and then perform calculations on the selected records, in the field you specify. You can use database statistical @functions on any range of related data organized in rows and columns, as long as the columns have unique field names and you provide a criteria range. A **field name** is a label in the first row of a field that identifies contents of the field.

Most database statistical @functions have equivalent statistical @functions. Use the database statistical @functions to select and use only those values in a field that meet specific criteria. Use the statistical @functions to calculate all values in a range. For example, use @DAVG to find the average of values that meet criteria you specify, such as the average sales in July by each salesperson. Use @AVG to calculate the average value for a range, such as the average of all sales. Using database statistical @functions is equivalent to using the corresponding statistical @function on the results of a Data Query command.

| | A | B | C | D | E | F | G | |
|----|--|---------------|----|-----------|-----------|-----------|---|---|
| 25 | DATE | ADDRESS | BR | OFFERED | SOLD | BROKER | | ◀ |
| 26 | 03-Apr | 467 Brattle | 4 | \$775,000 | \$720,000 | JCompton | | ▶ |
| 27 | 05-Apr | 183 Hillside | 3 | 325,000 | 318,000 | CGroden | | ▶ |
| 28 | 10-Apr | 64 N. Gate | 2 | 340,000 | 332,000 | AMiller | | ? |
| 29 | 14-Apr | 80 Mt. Auburn | 2 | 280,000 | 278,000 | JCompton | | |
| 30 | 25-Apr | 14 Charles | 4 | 179,000 | 160,000 | AMiller | | |
| 31 | 27-Apr | 1160 Memorial | 1 | 230,000 | 227,000 | AMiller | | |
| 32 | 04-May | 130 Crescent | 3 | 405,000 | 397,000 | CGroden | | |
| 33 | 10-May | 12 Trenton | 2 | 310,000 | 303,000 | AMiller | | |
| 34 | 11-May | 36 Barnes | 4 | 680,000 | 669,000 | JCompton | | |
| 35 | 22-May | 234 Third | 2 | 155,000 | 140,000 | JCompton | | |
| 36 | | | | | | | | |
| 37 | | | BR | | | | | |
| 38 | | | 2 | | | | | |
| 39 | | | | | | | | |
| 40 | Lowest price for a 2-bedroom house, APRIL and MAY: | | | | | \$140,000 | | |
| 41 | | | | | | | | |

Input range SALES (A25..F35)

Field whose offset number is 4

@DMIN(SALES,4,CRIT_RANGE)

Criteria range CRIT_RANGE

All database statistical @functions have the same three arguments: *input*, *field*, and *criteria*.

input is the address or name of a range that contains a database. In the illustration above, SALES is the range name for the *input* range. You do not need to use /Data Query to identify an *input* range before you use it as an argument in a database @function.

field is an offset number (or the address of a cell that contains the offset number) that indicates the field's position in *input*. A **field** is a column of related data in a database, including the field name. Offset numbers begin with 0, so if *field* is a value greater than the number of columns in the database minus 1, the @function returns ERR.

criteria is a range that specifies selection requirements. A criteria range occupies cells in at least two rows. The first row contains some or all the field names in the *input* range. The second row (and any subsequent rows) contains criteria that determine what records 1-2-3 selects. Enter criteria directly below the field name of the field to which they apply. *criteria* can be a range address or the name of a range. In the illustration above, CRIT_RANGE is the range name for the *criteria* range. For more information about criteria, see "Writing Criteria" in Chapter 14 of the *User's Guide*.

In the illustration above, @DMIN(SALES,4,CRIT_RANGE), entered in cell F40, returned the lowest price for which a two-bedroom house was sold.

| @Function | Description |
|------------------|--|
| @DAVG | Averages values in a field. |
| @DCOUNT | Counts the nonblank cells in a field. |
| @DMAX | Finds the greatest value in a field. |
| @DMIN | Finds the least value in a field. |
| @DSTD | Calculates the standard deviation of the values in a field. |
| @DSUM | Sums the values in a field. |
| @DVAR | Calculates the population variance of the values in a field. |

Date and Time @Functions

Date and time @functions calculate with **date numbers**, numbers that correspond to dates from January 1, 1900 (the number 1), through December 31, 2099 (the number 73050), and times from 12:00 midnight (the number 0.000000) through 11:59:59 PM (the number 0.999988). Date and time @functions simplify tasks such as calculating differences between two dates or two times, sorting by dates or times, and comparing a range of dates or times to a particular date or time.

To display a date or time number as the date or time it represents, format the cell that contains the date or time number with /Range Format Date or /Worksheet Global Format Date. For example, @DATE(88,3,7) returns the date number 32209. You can format this number to appear on the screen as 07-Mar-88, 07-Mar, Mar-88, or in an international date format. @TIME(14,30,50) returns the time number 0.604745. You can format this number to appear as 02:30 PM, 02:30:50 PM, or in an international time format.

1-2-3 uses only the integer part of a number in an @function that calculates dates. For example, both @YEAR(31790.45) and @YEAR(31790) return 87. 1-2-3 uses only the decimal portion of a number in an @function that calculates time. For example, both @HOUR(.03125) and @HOUR(31920.03125) return 0 (midnight). 1-2-3 calculates the decimal portion by using the ratio of the number of seconds since midnight to the number of seconds in a day.

Five @functions calculate with dates.

| @Function | Description |
|------------------|---|
| @DATE | Calculates the date number for specified year, month, and day values. |
| @DATEVALUE | Converts text that looks like a date into its equivalent date number. |
| @DAY | Calculates the day of the month from a date number. |
| @MONTH | Calculates the number of the month from a date number. |
| @YEAR | Calculates the number of the year from a date number. |

Five @functions calculate with times.

| @Function | Description |
|------------------|---|
| @HOUR | Calculates the hour from a time number. |
| @MINUTE | Calculates the minutes from a time number. |
| @SECOND | Calculates the seconds from a time number. |
| @TIME | Calculates the time number for specified hour, minutes, and seconds values. |
| @TIMEVALUE | Converts text that looks like a time into its equivalent time number. |

One @function calculates the current date and/or time number.

| @Function | Description |
|------------------|---|
| @NOW | Calculates the date and time number that corresponds to the current date and time set on your computer's clock. |

Financial @Functions

Financial @functions replace complex formulas that analyze investments and annuities, determine depreciation, and calculate cash flow and loan values.

Keep the following rules in mind when you use financial @functions:

- Interest rates must be either percentages or decimal values. For example, you can enter 15.5% either as .155 or as 15.5%, but not as 15.5. When you enter a formula, 1-2-3 converts all values followed by % (percent) to decimal values.
- Interest rates and payment periods must use the same unit of time. For example, if the annual interest rate is 15.5% and the payment period is monthly, divide 15.5% by 12 to find the interest rate per month.
- Financial @functions assume that investments are ordinary annuities. An annuity is an investment in which a series of equal payments are made. An ordinary annuity is an annuity in which a payment is made at the end of each period. An annuity due is an annuity in which a payment is made at the beginning of each period. To convert to an annuity due, multiply the entire @function by $(1 + \textit{interest})$.

Three financial @functions calculate depreciation.

| @Function | Description |
|------------------|---|
| @DDB | Calculates the double-declining balance depreciation allowance of an asset. |
| @SLN | Calculates the straight-line depreciation allowance of an asset. |
| @SYD | Calculates the sum-of-the-years'-digits depreciation allowance of an asset. |

Two financial @functions are used for capital budgeting.

| @Function | Description |
|------------------|--|
| @IRR | Calculates the internal rate of return for a series of cash-flow values. |
| @NPV | Calculates the net present value of a series of future cash-flow values. |

Four financial @functions calculate annuities.

| @Function | Description |
|------------------|---|
| @FV | Calculates the future value of a series of equal payments. |
| @PMT | Calculates the amount of the periodic payment needed to pay off a loan. |
| @PV | Calculates the present value of a series of equal payments. |
| @TERM | Calculates the number of payment periods of an investment. |

Two financial @functions calculate single-sum compounding.

| @Function | Description |
|------------------|---|
| @CTERM | Calculates the number of compounding periods necessary for an investment to grow to a given future value. |
| @RATE | Calculates the periodic interest rate necessary for an investment to grow to a given future value. |

Logical @Functions

Logical @functions evaluate Boolean conditions — conditions that are either true (returning the value 1) or false (returning the value 0). Except for @IF, all the logical @functions return either 1 or 0 — they cannot return any other values, including ERR (error) or NA (not available).

Use logical @functions in the following situations:

- Use @ISERR and @ISNA to test for the values ERR and NA. These values cause a ripple-through effect: when formulas return ERR or NA, other formulas that depend on them also return ERR or NA. For example, if a formula in cell D25 returns ERR, @AVG(D9..D56) and +C25+D25+E25 also return ERR, because both refer to cell D25.
- Use @ISERR and @ISNA with @IF to stop the ripple-through effect. For example, the formula @IF(@ISERR(D25),0,D25/2) returns 0 if cell D25 contains the value ERR, preventing 1-2-3 from evaluating D25/2, which would return ERR.
- Use @ISNUMBER and @ISSTRING to prevent errors that may occur if a cell used in a formula contains the wrong type of data. For example, @IF(@ISNUMBER(D9),@AVG(A9..J9),"Label") returns the result of @AVG(A9..J9) if cell D9 contains a value or is blank. If cell D9 contains a label, it returns the word Label.

- Use @ISAAF and @ISAPP to return information about the status of add-in @functions and add-in programs. For example, @ISAPP("Wysiwyg") returns 1 if the add-in program called Wysiwyg is currently attached.

All logical @functions take a value, a special value (ERR or NA), a cell address, or a single-cell range name as an argument.

| @Function | Description |
|------------------|--|
| @FALSE | Returns the logical value 0 (false). |
| @IF | Returns one value if the condition is true, and another value if the condition is false. |
| @ISAAF | Returns 1 (true) for an attached add-in @function; 0 (false) for any other entry. |
| @ISAPP | Returns 1 (true) for a currently attached add-in; 0 (false) for any other entry. |
| @ISERR | Returns 1 (true) for the value ERR; 0 (false) for any other value. |
| @ISNA | Returns 1 (true) for the value NA; 0 (false) for any other value. |
| @ISNUMBER | Returns 1 (true) for a numeric value, NA, ERR, or a blank cell; 0 (false) for a string. |
| @ISSTRING | Returns 1 (true) for a label or a string; 0 (false) for a numeric value, NA, ERR, or a blank cell. |
| @TRUE | Returns the logical value 1 (true). |

Mathematical @Functions

Mathematical @functions simplify various mathematical operations, such as calculating square roots and complex trigonometric functions. Some mathematical @functions return new values, while others affect values calculated by other formulas or @functions.

When you use mathematical @functions, observe the following rules:

- @SIN, @COS, and @TAN require angles in radians. To convert degrees to radians, multiply the number of degrees by @PI/180.
- @ASIN, @ACOS, @ATAN, and @ATAN2 return angle values in radians. To convert radians to degrees, multiply the number of radians by 180/@PI.

Mathematical @functions use only values (a value, a special value such as ERR or NA, a cell address, or a single-cell range name) as arguments. These @functions do not accept multiple-cell ranges. Except as otherwise noted, any mathematical @function accepts a blank cell or empty string (a string with a length of 0). The result is the same as applying the @function to the value 0.

| @Function | Description |
|------------------|---|
| @ABS | Calculates the absolute (positive) value of a value. |
| @ACOS | Calculates the arc cosine of a value. |
| @ASIN | Calculates the arc sine of a value. |
| @ATAN | Calculates the arc tangent of a value. |
| @ATAN2 | Calculates the arc tangent of the quotient of two values. |
| @COS | Calculates the cosine of an angle. |
| @EXP | Calculates the natural logarithm (base e) raised to a specified power. |
| @INT | Returns the integer portion of a value. |
| @LN | Calculates the natural logarithm (base e) of a value. |
| @LOG | Calculates the common logarithm (base 10) of a value. |
| @MOD | Calculates the remainder (modulus) of two values. |
| @PI | Returns the value π (approximately 3.1415926536). |
| @RAND | Generates a random value between 0 and 1. |
| @ROUND | Rounds a value to a specified number of decimal places. |
| @SIN | Calculates the sine of an angle. |
| @SQRT | Calculates the positive square root of a value. |
| @TAN | Calculates the tangent of an angle. |

Special @Functions

Special @functions return information about cells or ranges, find the contents of a cell, or mark places where information is missing or incorrect. These @functions are particularly useful with macros.

| @Function | Description |
|------------------|---|
| @@ | Returns the contents of a cell referenced through another cell. |
| @? | Indicates an unknown @function from an add-in program. |
| @CELL | Returns information about a cell or its contents or settings. |
| @CELLPOINTER | Returns information about the current cell or its contents or settings. |
| @CHOOSE | Finds a specified value or string in a list of values and/or strings. |
| @COLS | Counts the columns in a range. |
| @ERR | Marks cells that depend on information that is currently incorrect. |
| @HLOOKUP | Finds the contents of a cell in a specified row in a horizontal lookup table. |
| @INDEX | Finds the contents of the cell in a row and column in a range. |
| @NA | Marks cells that depend on information that is not available. |

(continued)

| @Function | Description |
|------------------|--|
| @ROWS | Counts the rows in a range. |
| @VLOOKUP | Finds the contents of a cell in a specified column in a vertical lookup table. |

Statistical @Functions

Statistical @functions perform calculations on a list of values in a range. Each statistical @function has an equivalent database statistical @function. For example, you use @AVG to calculate the average value for a range; you use @DAVG to find the average value of values that meet criteria you specify. Use the database statistical @functions to calculate values in a field that meet specific criteria.

Observe the following guidelines for all statistical @functions:

- The statistical @functions ignore blank cells in a multiple-cell range used as the argument, but they do not ignore references to blank cells listed individually as part of an argument. For example, if you use @AVG to average the values in a range that spans four cells (A1..A4), and cell A2 is a blank cell, 1-2-3 divides the sum by three to find the correct average. If you list those four cells individually, however, (A1,A2,A3,A4), 1-2-3 divides the sum by four.
- All statistical @functions assign the value 0 (not ERR) to all labels in a range or list of values used as the argument, and include the labels in calculations. For example, if you use @AVG to calculate the average value for a range, and the range contains a label, 1-2-3 considers the label to have the value 0 when it calculates the average. Check for labels in the ranges you use in the @function to guard against unexpected results.
- All statistical @functions except @COUNT return ERR or NA if any cell in a range or list used as the argument contains ERR or NA.

All statistical @functions have the argument *list*. There can be one or more entries in *list*. Each entry in *list* can be a value, a cell, a range address, or a range name. For example, the argument *list* in the @function @AVG(12,B3..B8,SALES) contains a value, a multiple-cell range address, and a range name.

| @Function | Description |
|------------------|---|
| @AVG | Averages a list of values. |
| @COUNT | Counts the nonblank cells in a range or list. |
| @MAX | Finds the greatest value in a list of values. |
| @MIN | Finds the least value in a list of values. |
| @STD | Calculates the standard deviation of the values in a list. |
| @SUM | Sums the values in a list. |
| @VAR | Calculates the population variance of the values in a list. |

String @Functions

String @functions provide information about text in cells, and perform other operations on text.

Observe the following rules when you use string @functions:

- Text arguments must be in double quotation marks. For example, enter @LEFT("Monthly Expenses",5) to return the string Month.
- If a cell contains one of the label prefixes " ' ^ or | but contains no text, 1-2-3 treats it as an **empty string**, a string with a length of 0. The cell looks blank, but 1-2-3 will not return the value ERR when you use it as an argument in a string @function.
- Uppercase and lowercase letters have different character codes in the Lotus International Character Set (LICS). For example, @CODE("A") returns the code number 65, but @CODE("a") returns 97 and @CODE("á") returns 225. For information on LICS, see Appendix A of the *User's Guide*.

| @Function | Description |
|-----------|--|
| @CHAR | Returns the character that corresponds to a code number in the Lotus International Character Set (LICS). |
| @CLEAN | Removes control characters from a string. |
| @CODE | Returns the LICS code number that corresponds to the first character in a string. |
| @EXACT | Returns 1 (true) if two strings are the same, and 0 (false) if the strings are different. |
| @FIND | Returns the position of the first character of one string within another string. |
| @LEFT | Returns the specified number of characters from the beginning of a string. |
| @LENGTH | Counts the characters in a string. |
| @LOWER | Converts all the letters in a string to lowercase. |
| @MID | Returns a specified number of characters in a string, starting at a specified character. |
| @N | Returns the value in the first cell in a range or 0 if the cell contains a label. |
| @PROPER | Converts the first letter in each word in a string to uppercase, and the rest to lowercase. |
| @REPEAT | Duplicates a string a specified number of times. |
| @REPLACE | Replaces characters in one string with characters from a different string. |
| @RIGHT | Returns the specified number of characters from the end of a string. |

(continued)

| @Function | Description |
|------------------|--|
| @S | Returns the string value of the first cell in a range. |
| @STRING | Converts a value into a label with a specified number of decimal places. |
| @TRIM | Removes leading, trailing, and consecutive spaces from a string. |
| @UPPER | Converts all the letters in a string to uppercase. |
| @VALUE | Converts a string that looks like a number into a value. |

Chapter 2

@Function Descriptions

This chapter lists the @functions alphabetically. Each @function is described in detail, and includes one or more examples to show the syntax and use of the @function.

The descriptions of the @functions use the following conventions:

- @Function names appear in uppercase. When you enter an @function, you can use either uppercase or lowercase letters.
- Argument names in the @function definitions appear in lowercase italics. You must substitute the type of information required by that argument.
- Arguments used in examples are not italicized.
- Optional arguments are enclosed in [] (brackets). You do not need to specify an optional argument for the @function to work. When you use an optional argument, do not enter the brackets.
- In the examples, ... (ellipses) indicate that the example is an excerpt from a macro or application whose other instructions are not relevant.

@@

@@(*location*) returns the cell or range address produced by *location*.

Argument

location is the address or name of a cell that contains a cell address or name, or a text formula that returns the address or name of a cell. *location* points to another cell whose contents @@ displays in the cell that contains @@. If *location* is not a valid cell address or range name, or is a multiple-cell range, @@ returns ERR.

Uses

@@ is useful as an indirect cell reference. For example, if cell C8 contains the label A10, the formula @@(C8) returns the value in cell A10.

You can also use @@ to return a cell or range address to another @function. For example, if the label A1..A10 is in cell B4, the formula @SUM(@@(B4)) returns the sum of the range A1..A10.

@@ is useful in building conditional formulas because its indirect reference can automatically alter its own value. For example, if A10 contains the formula @IF(DISCOUNT="Y","D8",@IF(DISCOUNT="N","D9",@ERR)), and E2 contains the formula @@(A10), 1-2-3 returns the contents of cell D8 or D9, or ERR, in E2, depending on whether cell DISCOUNT contains Y or N, or something else.

Notes

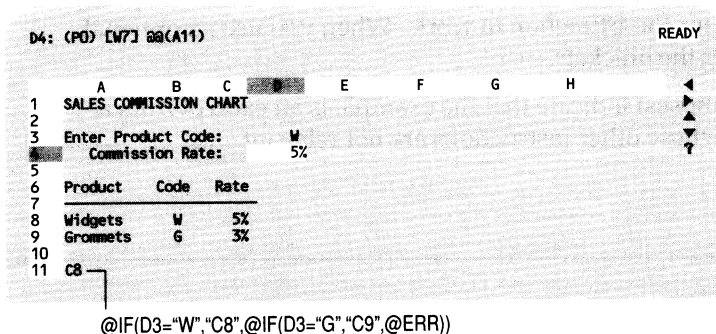
@@ differs from other @functions that manipulate text or return cell addresses. @Functions that manipulate text usually return a label as the result; @@ returns the actual cell address of the label in *location*.

1-2-3 recalculates all @@ functions when the worksheet is recalculated.

Examples

@@(C8) returns the value in cell A10, if cell C8 contains the label A10.

The following illustration shows a sales commission chart. @@(A11) in cell D4 returns the contents of C8, which is the cell specified in cell A11. A11 contains an @IF formula that enters one of two cell addresses, depending on which product code you enter in D3. If you enter anything in D3 other than a valid product code, both the @IF and @@ functions return ERR.



@?

@? is a special @function that 1-2-3 uses to indicate the location of an unknown add-in @function that is referred to by a formula in a worksheet.

Uses

1-2-3 works with add-ins that provide their own @functions. If you retrieve a worksheet that contains add-in @functions without first attaching the appropriate add-in, 1-2-3 translates the @function name to @? and interprets the @function as NA.

Notes

You cannot enter @? directly in a worksheet.

@ABS

@ABS(x) calculates the absolute value of x .

Argument

x is a value, the address or name of a cell that contains a value, or a formula that returns a value.

Uses

Use @ABS when you need numbers to be non-negative, such as percentage differences between actual and budgeted values, or to find the absolute difference between values in a list of positive and negative values.

Use @SQRT with @ABS to prevent ERR when you need to find the square root of a negative number.

Notes

Use -@ABS to force the result of the @function to be negative.

Examples

@ABS(A5) = 25, if cell A5 contains the value 25, -25, or a formula that results in 25 or -25.

-@ABS(A5) = -25, if cell A5 contains the value 25, -25, or a formula that results in 25 or -25.

@ABS(SALES*A9) = 20, if cell SALES contains the value 2 and cell A9 contains the value 10, -10, or a formula that results in 10 or -10.

@ABS(0) = 0.

@ACOS

@ACOS(x) calculates the arc cosine (inverse cosine) using the cosine x of an angle. The result of @ACOS is an angle, in radians, from 0 through π . This represents an angle between 0° and 180° .

Argument

x is the cosine of an angle and can be a value, the address or name of a cell that contains a value, or a formula that returns a value from -1 through 1.

Uses

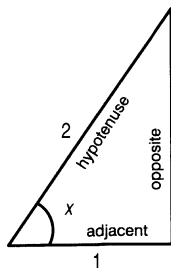
Use @ACOS to find the size of an angle when you know the cosine of the angle. @ACOS calculates the angle between the hypotenuse and the side adjacent to the angle in a right triangle.

Notes

To convert radians to degrees, multiply by $180/@PI$.

Example

In the right triangle below, you calculate the cosine of angle x before you use $@ACOS$ to find the arc cosine of the angle.



The cosine of x (calculated as cosine = adjacent/hypotenuse) is $1/2$, or 0.5 . To determine the arc cosine, use $@ACOS(0.5)$.

The result is 1.04720 , rounded. To convert this to degrees, use $@ACOS(0.5)*180/@PI$.

The result is 60° , the size of angle x .

@ASIN

$@ASIN(x)$ calculates the arc sine (inverse sine) using the sine x of an angle. The result of $@ASIN$ is an angle, in radians, from $-\pi/2$ through $\pi/2$. This represents an angle between -90° and 90° .

Argument

x is the sine of an angle and can be a value, the address or name of a cell that contains a value, or a formula that returns a value from -1 through 1 .

Uses

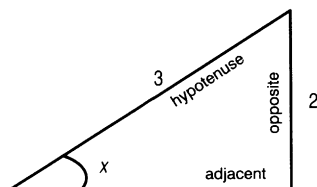
Use $@ASIN$ to find the size of an angle when you know the sine of the angle. $@ASIN$ calculates the angle between the hypotenuse and the side adjacent to the angle in a right triangle.

Notes

To convert radians to degrees, multiply by $180/@PI$.

Example

In the right triangle below, you calculate the sine of angle x before you use $@ASIN$ to find the arc sine of the angle.



The sine of x is $2/3$, or 0.66 (calculated as sine = opposite/hypotenuse). To determine the arc sine, use $@ASIN(.66)$.

The result is $.72082$, rounded. To convert the result to degrees, use $@ASIN(.66)*180/@PI$. The result is 41.3° (rounded), the size of angle x .

@ATAN

@ATAN(x) calculates the arc tangent (inverse tangent) using the tangent x of an angle. The result of @ATAN is an angle, in radians, from $-\pi/2$ through $\pi/2$. This represents an angle between -90° and 90° .

Argument

x is the tangent of an angle and can be any value, the address or name of a cell that contains a value, or a formula that returns a value.

Uses

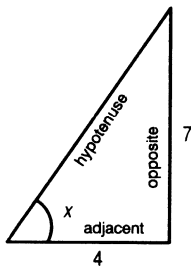
Use @ATAN to find the size of an angle when you know the tangent of the angle. @ATAN calculates the angle between the hypotenuse and the adjacent side.

Notes

To convert radians to degrees, multiply by $180/@PI$.

Example

In the right triangle below, you calculate the tangent of angle x before you use @ATAN to find the arc tangent of the angle.



The tangent of x (calculated as tangent = opposite/adjacent) is $7/4$, or 1.75. To determine the arc tangent, use @ATAN(1.75).

The result is 1.10517, rounded. To convert this to degrees, use @ATAN(1.75)*180/@PI.

The result is 60.3° (rounded), the size of angle x .

@ATAN2

@ATAN2(x,y) calculates the arc tangent using the tangent y/x of an angle. The result of @ATAN2 is an angle, in radians, from $-\pi$ through π . This represents an angle between -180° and 180° , depending on the sign of x and y (see the table on page 20).

Arguments

x and y are values, the addresses or names of cells that contain values, or formulas that return values. If y is 0, @ATAN2 returns 0; if both x and y are 0, @ATAN2 returns ERR.

Uses

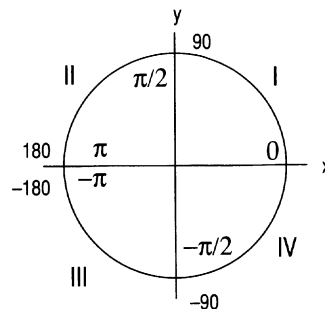
Use @ATAN2 to find the size of an angle in any of the four quadrants when you know the length of the two sides (x and y) that form the right angle in a right triangle. @ATAN2 calculates the angle between the hypotenuse and either of the sides that form the right angle, without first having to calculate the tangent.

Notes

To convert radians to degrees, multiply by $180/@PI$.

@ATAN2 differs from @ATAN in that the result of @ATAN2 is a value from $-\pi$ to π . The table below lists the value ranges for @ATAN2.

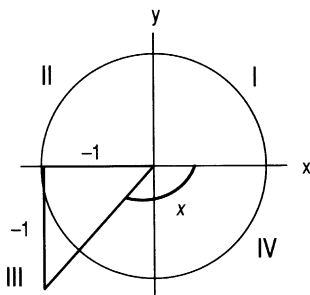
| x | y | @ATAN2(x,y) | Quadrant |
|----------|----------|------------------------------|----------|
| Positive | Positive | From 0 through $\pi/2$ | I |
| Negative | Positive | From $\pi/2$ through π | II |
| Negative | Negative | From $-\pi$ through $-\pi/2$ | III |
| Positive | Negative | From $-\pi/2$ through 0 | IV |



When x and y are both positive (quadrant I), and when x is positive and y is negative (quadrant IV), the results are the same as for @ATAN.

Example

In the right triangle below, use @ATAN2 to find the size of the angle x .



The tangent of x is $-1/-1$, or 1. To determine the arc tangent, use @ATAN2($-1,-1$).

The result is -2.35619 , rounded. To convert this to degrees, use @ATAN2($-1,-1$)* $180/@PI$.

The result is -135° , the size of angle x .

@AVG

@AVG(*list*) calculates the average of a *list* of values.

Argument

list is a series of values separated by argument separators. *list* can contain any combination of values, formulas, and the addresses or names of ranges that contain values.

Uses

Use @AVG to find the average, or mean, of any series of values in a worksheet.

Notes

@AVG ignores blank cells in ranges, but not references to blank cells listed individually. Labels count as zero, as do cells that are apparently empty but contain a label-prefix character or spaces. This means that labels, apparently empty cells, and blank cells listed individually increase the total number of items in *list*; if *list* contains any of these, the result of @AVG may not be what you expect.

Example

In the following illustration, @AVG(A1..A5) entered in G4 returns 43.75 and @AVG(A1,A2,A3,A4,A5) entered in G5 returns 35. This is because the blank cell (A4) is listed individually and causes 1-2-3 to divide the total by five.

| | A | B | C | D | E | F | G | H |
|---|----|---|---|---|---|---|-------------------------------------|---|
| 1 | 66 | | | | | | | |
| 2 | 39 | | | | | | | |
| 3 | 30 | | | | | | | |
| 4 | | | | | | | | |
| 5 | 20 | | | | | | Average value in A1..A5: 43.75 | |
| 6 | | | | | | | Average value in A1,A2,A3,A4,A5: 35 | |

A blank cell

Similar @functions

@DAVG finds the average of values that meet criteria you specify.

@CELL

@CELL(*attribute,range*) returns information about the first cell in *range*. The information depends on the *attribute* you specify.

Arguments

attribute can be any of the 10 items listed in the table below, enclosed in double quotation marks, or the address or name of a cell that contains one of the items.

| Attribute | Result |
|-----------|---|
| address | The absolute cell address (for example, \$A\$1) |
| col | The column letter, as a value from 1 through 256 (1 for column A, 5 for column E, and so on) |
| contents | The contents of the cell |
| filename | The name of the current file including the path |
| format | The cell format: C0 through C15 if Currency, 0 to 15 decimal places F0 through F15 if Fixed, 0 to 15 decimal places G if General P0 through P15 if Percent, 0 to 15 decimal places S0 through S15 if Sci (Scientific), 0 to 15 decimal places ,0 to ,15 if , (Comma), 0 to 15 decimal places + if +/- format D1 if DD-MMM-YY D2 if DD-MMM D3 if MMM-YY D4 if MM/DD/YY, DD/MM/YY, DD.MM.YY, or YY-MM-DD D5 if MM/DD, DD/MM, DD.MM, or MM-DD D6 if HH:MM:SS AM/PM D7 if HH:MM AM/PM D8 if HH:MM:SS (24 hour), HH.MM.SS (24 hour), HH,MM,SS (24 hour), or HHhMMmSSs D9 if HH:MM (24 hour), HH.MM (24 hour), HH,MM, or HHhMMm T if Text format H if Hidden format |
| prefix | The label prefix: ' if the cell contains a left-aligned label " if the cell contains a right-aligned label ^ if the cell contains a centered label \ if the cell contains a repeating label if the cell contains a nonprinting label Blank (no symbol) if the cell is empty or contains a value |

(continued)

| Attribute | Result |
|-----------|--|
| protect | The protection status: 1 if the cell is protected (default) 0 if the cell is unprotected (by /Range Unprotect) |
| row | The row number, from 1 through 8192 |
| type | The type of data in the cell: b if the cell is blank (that is, has no entry) v if the cell contains a numeric value or a formula l if the cell contains a label |
| width | The column width |

range is the address or name of a range, or a formula that returns the address or name of a range.

Uses

@CELL is useful in macros and in combination with @IF. Use @CELL to check input during a macro to guard against certain types of entries, and to run subroutines based on a user's entry. @CELL can also allow an automated application or template to change cell attributes based on a user's entries.

Notes

@CELL returns the named attribute in the upper left corner of *range*. Because the cell is not recalculated by some commands, recalculate with {CALC} before you use @CELL to be sure the results are correct.

Example

The following example uses @CELL with @IF and @ERR to return an error (ERR) if the user does not type a value in the cell named AMOUNT, and to return the contents of AMOUNT (a value) if the user types a value.

```
@IF(@CELL("type",AMOUNT)="v",AMOUNT,@ERR)
```

@CELLPOINTER

@CELLPOINTER(*attribute*) returns information about the current cell. The information depends on the *attribute* you specify.

Argument

attribute can be any of the 10 *attribute* arguments for @CELL, enclosed in double quotation marks, or a cell that contains one of the items. For a list of *attribute* arguments, see the table for @CELL beginning on page 22.

Uses

@CELLPOINTER is useful in macros and in combination with @IF. Use @CELLPOINTER to find the cell pointer's current location or to evaluate a formula based on the contents of the current cell. You can then direct processing depending on the cell's contents or type.

Notes

1-2-3 automatically updates @CELLPOINTER only when you make an entry. To make @CELLPOINTER return information about the current cell, if you have simply moved the cell pointer to it, you must recalculate the worksheet.

Example

The following excerpt from a macro uses @CELLPOINTER to test the current cell in a list of items. If 1-2-3 encounters a blank cell, it beeps and branches to a subroutine.

```
...  
{IF @CELLPOINTER("type")="b"}{BEEP}{BRANCH STEP2}  
...
```

@CHAR

@CHAR(*x*) returns the Lotus International Character Set (LICS) code character that corresponds to the number *x*. For information on LICS, see Appendix A in the *User's Guide*.

Argument

x is an integer, the address or name of a cell that contains an integer, or a formula that returns an integer. Values that do not correspond to character codes return ERR. If *x* is not an integer, @CHAR truncates it to an integer.

Uses

@CHAR is useful for entering foreign language characters and mathematical symbols. Whether a character prints depends on the capabilities of your printer.

Notes

If your monitor cannot display the character that corresponds to *x*, 1-2-3 displays a character that resembles the desired character when possible. If no character approximates the character, 1-2-3 displays nothing. Make sure your printer can print the characters you enter.

Examples

@CHAR(163) = £ (British pound sign).

@CHAR(D9) = A, if cell D9 contains the value 65.

@CHOOSE

@CHOOSE(x ,*list*) returns the x th value or label from *list*.

Arguments

x is a value, the address or name of a cell that contains a value, or a formula that returns a value. x represents an **offset number**. An offset number is any positive number, starting with 0, that corresponds to an item's position in *list*. The first item has the offset number 0, the second item has the offset number 1, and so on.

list is a group of values and labels, or the addresses or names of cells that contain values and labels, separated by argument separators. 1-2-3 numbers each entry in *list*, and then chooses the entry that corresponds to the value of x .

Uses

Use @CHOOSE to enter a list of values to use in formulas without setting up a lookup table.

Notes

x cannot be greater than the number of entries in *list*, minus 1. If x is a blank cell, @CHOOSE treats it as the value 0 and returns the first item in *list*, because 0 is the offset number for the first item.

Example

@CHOOSE(2,"zero","one","two","three") = two.

Similar @functions

@HLOOKUP and @VLOOKUP choose values from lookup tables. @INDEX produces a value from a table using relative locations.

@CLEAN

@CLEAN(*string*) removes the following control characters from *string*:

- Control characters with ASCII codes below 32
- The begin attribute character (LICS code 151), as well as the attribute character itself
- The end attribute character (LICS code 152)
- The merge character (LICS code 155) and the character following

Argument

string is text (enclosed in double quotation marks), the address or name of a cell that contains text, or a formula or @function that results in text.

Uses

Use @CLEAN to remove Wysiwyg control codes from cells before you print in 1-2-3. For example, if a cell uses a Wysiwyg format to show a customer's first and last name in different colors, you can use @CLEAN to remove the control codes so your printed worksheet does not show the control codes and the label in the cell.

Notes

If *string* refers to a blank cell or a value, @CLEAN returns ERR.

Example

You imported data into 1-2-3 from a word processing program. Cell A45 contains the label.

→Second, we must act soon.←

@CLEAN(A45) = Second, we must act soon.

@CODE

@CODE(*string*) returns the Lotus International Character Set (LICS) code that corresponds to the first character in *string*. For information on LICS, see Appendix A in the *User's Guide*.

Argument

string is text (enclosed in double quotation marks), the address or name of a cell that contains text, or a formula or @function that results in text.

Uses

Use @CODE when you need to know the LICS code number for a character.

Notes

If *string* refers to a blank cell or a value, @CODE returns ERR.

Examples

@CODE("A") = 65.

@CODE(C5) = 77, if C5 contains the label Ms. Jones, because 77 is the LICS code for M.

@COLS

@COLS(*range*) counts the number of columns in *range*.

Argument

range is a range address or name.

Uses

Use @COLS to determine the number of columns in a range so that you can perform other tasks based on the size of the range. For example, if you use a macro to print a range, you can determine the size of the range before you print it. In a macro that performs the same task on a series of columns, you can use @COLS to determine when the macro should stop.

Examples

@COLS(D9..J25) = 7, because *range* contains columns D through J (seven columns).

@COLS(SCORES) = 2, if SCORES is the name of the range B3..C45.

@COS

@COS(*z*) calculates the cosine of an angle expressed in radians. The result of @COS is a value from -1 through 1.

Argument

z is any value, in radians, the address or name of a cell that contains a value, or a formula that returns a value from -1.35^{10} through 1.35^{10} .

Uses

Use @COS to find the length of the adjacent side in a right triangle, when you know the length of the hypotenuse and the size of the angle between the hypotenuse and the side adjacent to the angle. You can also use @COS to find the length of the hypotenuse, when you know the length of the side adjacent to the angle.

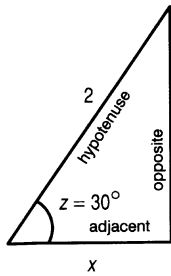
Use @COS to find the secant, or reciprocal of @COS, with the following formula:
 $1/@COS(z)$.

Notes

You must enter the angle *z* in radians. To convert from degrees to radians, multiply degrees by @PI/180.

Example

In the following right triangle, you calculate the cosine of angle *z* (30°) to find the length of side *x*.



You must convert 30° to radians to calculate the cosine:
`@COS(30*@PI/180)`.

The cosine is 0.866. Since cosine = adjacent/hypotenuse, to find the length of side x, use the formula adjacent = cosine * hypotenuse: 0.866*2.

The result is 1.732, which is the length of side x.

@COUNT

`@COUNT(list)` counts the number of cells in a *list* of cells or ranges.

Argument

list is a series of cell addresses or ranges that contain entries, separated by argument separators.

Uses

Use `@COUNT` to count the number of entries in a range. Since `@COUNT` also counts labels (or text), if you want to keep an accurate count of values in a range, make sure the range does not contain any text.

`@COUNT` is also useful to stop (or divert) a macro that performs a task on a series of ranges when the cell pointer reaches a range that has no entries.

Notes

`@COUNT` counts every cell in *list* that contains an entry of any kind, including a label, a label-prefix character, or the values ERR and NA. `@COUNT` does not count blank cells if they occur in a range; however, `@COUNT` counts blank cells if you refer to them by single-cell addresses in *list*.

If *list* includes an undefined range name, `@COUNT` changes the range name in the formula to ERR and counts it as 1.

Examples

`@COUNT(A2..A3,A5) = 1`, if A2..A3 is blank and whether or not A5 is blank, because A5 is a single-cell address.

`{IF @COUNT(SEPTEMBER)=0}{BRANCH YTD}` branches to a macro called YTD (which may calculate year-to-date totals) if the multiple-cell range named SEPTEMBER is blank.

Similar @functions

`@DCOUNT` counts the number of nonblank cells in a range that meet criteria you specify.

@CTERM

@CTERM(*interest*,*future-value*,*present-value*) calculates the number of compounding periods it takes for an investment (*present-value*) to grow to a *future-value*, earning a fixed *interest* rate per compounding period.

Arguments

interest is a value for the periodic interest rate expressed as a decimal or percentage (.1 or 10%), the address or name of a cell that contains a value, or a formula that returns a value.

future-value and *present-value* are values, the addresses or names of cells that contain values, or formulas that return values. Both arguments must be either positive or negative.

Uses

Use @CTERM to determine how long it will take a single investment of a given amount to grow to another given amount at a specified interest rate.

Notes

@CTERM uses the following formula to calculate the compounding period:

$$\frac{\ln(fv/pv)}{\ln(1 + int)}$$

where: *fv* = future value
pv = present value
int = interest rate
ln = natural logarithm

If the annual interest rate is 10% compounded monthly (as in the example below), enter .10/12 (the *interest* divided by the number of compounding periods).

Example

You just deposited \$10,000 in an account that pays an annual interest rate of 10% (.10), compounded monthly. To determine how many years it will take to double your investment, you enter @CTERM(.10/12,20000,10000)/12. 1-2-3 returns 6.960312; it will take about seven years to double the original investment of \$10,000.

Similar @functions

@TERM determines the number of periods it will take to reach a desired future value when equal periodic payments are made at a specified interest rate.

@DATE

@DATE(*year,month,day*) calculates the date number for the specified *year*, *month*, and *day*.

Arguments

year is an integer, the address or name of a cell that contains an integer, or a formula that returns a value from 0 (the year 1900) through 199 (the year 2099).

month is an integer, the address or name of a cell that contains an integer, or a formula that returns a value from 1 through 12.

day is an integer, the address or name of a cell that contains an integer, or a formula that returns a value from 1 through 31. The value you use for *day* must be a valid day for the *month*. For example, you cannot use 31 as the *day* if you use 4 (April) as the *month*.

If *year*, *month*, or *day* is not a value, @DATE returns ERR.

Uses

Use @DATE to create date entries to use in calculations. Dates you enter with @DATE (or with @DATEVALUE) are the only dates you can use in operations that depend on chronological order, such as sorting by date, or using search criteria to search for dates within a range.

Notes

Even though February 29, 1900 did not exist (it was not a leap year), 1-2-3 assigns a date number to this "day." This does not invalidate any of your date calculations, unless you use dates from January 1, 1900 through March 1, 1900. If you are using dates within that period with dates after March 1, 1900, subtract 1 from the result.

@DATE calculates the date number based on the number of days from January 1, 1900, through December 31, 2099. For example, January 1, 1900, corresponds to date number 1; January 2, 1900, corresponds to date number 2, and so on.

If you want the results of an @DATE calculation to appear as an actual date, format the cell that contains the @DATE function using /Range Format Date.

You can specify the format of *year*, *month*, and *day* with /Worksheet Global Default Other International Date.

Examples

@DATE(87,5,23) = 31920, or 23-May-87 in a cell formatted as DD-MMM-YY.

@DATE(199,4,1) = 72776, or 01-Apr-2099 in a cell formatted as DD-MMM-YY.

@DATEVALUE

@DATEVALUE(*string*) calculates the date number for the date specified in *string*.

Argument

string is a label, a text formula, or the address or name of a cell that contains a label or text formula. The label or the result of the text formula must be in one of the five 1-2-3 date formats.

Uses

@DATEVALUE is useful with data imported from another program, such as a word processing program. You can also use @DATEVALUE to convert dates entered as labels to date numbers so that you can use the dates in calculations.

Notes

If you want the results of an @DATEVALUE calculation to appear as an actual date, format the cell that contains the @DATEVALUE function using /Range Format Date.

If you use the Date 3 format, for example @DATEVALUE("Nov-91"), 1-2-3 returns the value for the first day of the month.

You can specify the format to use for *string* with /Worksheet Global Default Other International Date.

Examples

@DATEVALUE(Birthday) = 22890, if the cell named Birthday contains the date string 1-Sep-62.

@DATEVALUE("1-Apr-91") = 33329.

@DATEVALUE("Apr-91") = 33329.

@DAVG

@DAVG(*input,field,criteria*) finds the average value in a *field* of a database (or any range set up like a database) for all values that meet the criteria in the *criteria* range.

Arguments

input is the address or name of a range that contains a database.

field is the field's offset number (a positive number or 0), the address of a cell that contains an offset number, or a formula that returns an offset number. 1-2-3 assigns the first field (column) in *input* the offset number 0, the second field the offset number 1, and so on.

criteria is a range of at least two rows. The top row of the *criteria* range contains exact duplicates of the *input* range's field name(s) for those fields for which you are specifying criteria. The row(s) below the field name(s) contain one or more criteria, or conditions, that each entry in the field must meet to be included by @DAVG. (For more information, see "Writing Criteria" in Chapter 14 of the *User's Guide*.)

Uses

Use @DAVG to find the average value of only those items in a list that meet certain conditions: for example, the average sales for a particular division or the average score for test-takers from a certain city.

Example

In the following illustration, the input range named SALES (A25..F35) lists house sales for April and May. Cell G40 shows the result of using @DAVG to determine the average selling price for a four bedroom house.

G40: (C0) [W9] @DAVG(SALES,4,CRIT_RANGE) READY

| A | B | C | D | E | F |
|--------|---------------|----|-----------|-----------|----------|
| DATE | ADDRESS | BR | OFFERED | SOLD | BROKER |
| 03-Apr | 467 Brattle | 4 | \$775,000 | \$720,000 | JCompton |
| 05-Apr | 183 Hillside | 3 | 325,000 | 318,000 | CGroden |
| 10-Apr | 64 N. Gate | 2 | 340,000 | 332,000 | AMiller |
| 14-Apr | 80 Mt. Auburn | 2 | 280,000 | 278,000 | JCompton |
| 25-Apr | 14 Charles | 4 | 179,000 | 160,000 | AMiller |
| 27-Apr | 1160 Memorial | 1 | 230,000 | 227,000 | AMiller |
| 04-May | 130 Crescent | 3 | 405,000 | 397,000 | CGroden |
| 10-May | 12 Trenton | 2 | 310,000 | 303,000 | AMiller |
| 11-May | 36 Barnes | 4 | 680,000 | 669,000 | JCompton |
| 22-May | 234 Third | 2 | 155,000 | 140,000 | JCompton |

SALES

CRIT_RANGE

40: Average price paid for a 4-bedroom house, APRIL and MAY: \$516,333

Similar @functions

@AVG finds the average value of a range of values.

@DAY

@DAY(*date-number*) calculates the day of the month (1 through 31) using the value of *date-number*.

Argument

date-number is a value, the address or name of a cell that contains a value, or a formula that returns a value from 1 (January 1, 1900) through 73050 (December 31, 2099).

Uses

@DAY is useful when you need to know only the day of the month, and not the entire date. @DAY can also supply the *day* argument for other date or time @functions that build on previously calculated dates (as in the examples below).

Notes

You can use one of the other date or time @functions to supply the value for *date-number*.

Examples

@DAY(@NOW) = the current day of the month.

@DAY(D9) = 25, if cell D9 contains the date number 20723 (the date 25-Sep-56).

@DAY(@DATEVALUE(BIRTHDAY)) = 1, if cell BIRTHDAY contains the date number 33329 (the date 1-Apr-91).

@DCOUNT

@DCOUNT(*input,field,criteria*) counts the nonblank cells in a *field* of a database table that meet the criteria in the *criteria* range.

Arguments

input is the address or name of a range that contains a database.

field is the field's offset number (a positive integer or 0), the address of a cell that contains an offset number, or a formula that returns an offset number. 1-2-3 assigns the first field (column) in *input* the offset number 0, the second field the offset number 1, and so on.

criteria is a range of at least two rows. The top row of the criteria range contains exact duplicates of the *input* range's field name(s) for those fields for which you are specifying criteria. The row(s) below the field name(s) contain one or more criteria, or conditions, that each entry in the field must meet to be included by @DCOUNT. (For more information, see "Writing Criteria" in Chapter 14 of the *User's Guide*.)

Uses

Use @DCOUNT to count the number of records in a database that meet specified criteria: for example, the number of checks written in a certain month.

Example

In the following illustration, the input range named SALES (A25..F35) lists house sales for April and May. Cell F40 uses @DCOUNT to count the number of houses sold by broker J. Compton during April and May.

F40: (G) [W6] @DCOUNT(SALES,5,C37..C38)

READY

| | A | B | C | D | E | F | G |
|----|---|---------------|----------|-----------|-----------|----------|---|
| 25 | DATE | ADDRESS | BR | OFFERED | SOLD | BROKER | |
| 26 | 03-Apr | 467 Brattle | 4 | \$775,000 | \$720,000 | JCompton | |
| 27 | 05-Apr | 183 Hillside | 3 | 325,000 | 318,000 | CGroden | |
| 28 | 10-Apr | 64 N. Gate | 2 | 340,000 | 332,000 | AHiller | |
| 29 | 14-Apr | 80 Mt. Auburn | 2 | 280,000 | 278,000 | JCompton | |
| 30 | 25-Apr | 14 Charles | 4 | 179,000 | 160,000 | AHiller | |
| 31 | 27-Apr | 1160 Memorial | 1 | 230,000 | 227,000 | AHiller | |
| 32 | 04-May | 130 Crescent | 3 | 405,000 | 397,000 | CGroden | |
| 33 | 10-May | 12 Trenton | 2 | 310,000 | 303,000 | AHiller | |
| 34 | 11-May | 36 Barnes | 4 | 680,000 | 669,000 | JCompton | |
| 35 | 22-May | 234 Third | 2 | 155,000 | 140,000 | JCompton | |
| 36 | | | | | | | |
| 37 | | | BROKER | | | | |
| 38 | | | JCompton | | | | |
| 39 | | | | | | | |
| 40 | Number of houses sold by JCompton, APRIL and MAY: | | | | | | 4 |
| 41 | | | | | | | |

Similar @functions

@COUNT counts the nonblank cells in a range.

@DDB

@DDB(*cost,salvage,life,period*) calculates the depreciation allowance of an asset with an initial value of *cost*, an expected useful *life*, and a final *salvage* value for a specified *period* of time, using the double-declining balance method.

Arguments

cost is the amount paid for the asset, and must be a value greater than or equal to *salvage*.

salvage is the estimated value of the asset at the end of its useful life, and can be any value.

life is the number of periods the asset takes to depreciate to its salvage value, and must be a value greater or equal to *period*.

period is the time period for which you want to find the depreciation allowance, and must be a value greater than or equal to 1.

cost, *salvage*, *life*, and *period* are all values, the addresses or names of cells that contain values, or formulas that return values.

Uses

@DDB uses the double-declining balance method of depreciation. The double-declining balance method accelerates the rate of depreciation so that more depreciation expense occurs (and can be written off) in earlier periods than in later ones. Depreciation stops when the book value of the asset — that is, the total cost of the asset minus its total depreciation over all prior periods — reaches the salvage value.

Notes

@DDB uses the following formula to calculate the double-declining balance depreciation for any period:

$$\frac{(bv*2)}{n} \quad \text{where: } bv = \text{book value in that period} \\ n = \text{life of the asset}$$

Example

You purchased an office machine for \$10,000. The useful life of this machine is eight years, and the salvage value after eight years is \$1200. To calculate the depreciation expense for the fifth year using the double-declining balance method, you enter @DDB(10000,1200,8,5). 1-2-3 returns \$791.02, the depreciation expense for the fifth year of the asset's life.

Similar @functions

@SLN calculates depreciation using the straight-line method, and @SYD uses the sum-of-the-years'-digits method.

F40: (C0) [W10] ADMIN(SALES,4,CRIT_RANGE) READY

| | A | B | C | D | E | F | G |
|----|--------|---------------|----|-----------|-----------|----------|-------|
| | DATE | ADDRESS | BR | OFFERED | SOLD | BROKER | |
| 25 | 03-Apr | 467 Brattle | 4 | \$775,000 | \$720,000 | JCompton | |
| 26 | 05-Apr | 183 Hillside | 3 | 325,000 | 318,000 | CGroden | |
| 27 | 10-Apr | 64 N. Gate | 2 | 340,000 | 332,000 | AMiller | |
| 28 | 14-Apr | 80 Mt. Auburn | 2 | 280,000 | 278,000 | JCompton | |
| 29 | 25-Apr | 14 Charles | 4 | 179,000 | 160,000 | AMiller | |
| 30 | 27-Apr | 1160 Memorial | 1 | 230,000 | 227,000 | AMiller | |
| 31 | 04-May | 130 Crescent | 3 | 405,000 | 397,000 | CGroden | |
| 32 | 10-May | 12 Trenton | 2 | 310,000 | 303,000 | AMiller | SALES |
| 33 | 11-May | 36 Barnes | 4 | 680,000 | 669,000 | JCompton | |
| 34 | 22-May | 234 Third | 2 | 155,000 | 140,000 | JCompton | |
| 35 | | | | | | | |
| 36 | | | | | | | |
| 37 | | | | | | | |
| 38 | | | | | | | |
| 39 | | | | | | | |
| 40 | | | | | | | |
| 41 | | | | | | | |

Lowest price for a 2-bedroom house, APRIL and MAY: \$140,000

@DMAX

@DMAX(*input,field,criteria*) finds the greatest value in a *field* of a database that meets the criteria in the *criteria* range.

Arguments

input is the address or name of a range that contains a database.

field is the field's offset number (a positive integer or 0), the address of a cell that contains an offset number, or a formula that returns an offset number. 1-2-3 assigns the first field (column) in *input* the offset number 0, the second field the offset number 1, and so on.

criteria is a range of at least two rows. The top row of the *criteria* range contains exact duplicates of the *input* range's field name(s) for those fields for which you are specifying criteria. The row(s) below the field name(s) contain one or more criteria, or conditions, that each entry in the field must meet to be included by @DMAX. (For more information, see "Writing Criteria" in Chapter 14 of the *User's Guide*.)

Uses

Use @DMAX to find the highest value of only those items that meet certain conditions: the month with the highest sales for a particular product, for example. You can also use @DMAX to find the most recent date or time in a list of dates or times.

@DMAX is also useful when you need to check for unusually large values or to find (and discard) the largest value in some statistical calculations.

Example

In the following illustration, the input range named SALES (A25..G35) lists house sales for April and May. Cell G40 shows the result of using @DMAX to find the highest sale price of a two bedroom house.

G40: (C0) [W10] @DMAX(SALES,4,CRIT_RANGE) READY

| | A | B | C | D | E | F | G |
|----|--|---------------|----|-----------|-----------|----------|------------|
| 25 | DATE | ADDRESS | BR | OFFERED | SOLD | BROKER | COMMISSION |
| 26 | 03-Apr | 467 Brattle | 4 | \$775,000 | \$720,000 | JCompton | \$28,800 |
| 27 | 05-Apr | 183 Hillside | 3 | 325,000 | 318,000 | CGroden | 12,720 |
| 28 | 10-Apr | 64 N. Gate | 2 | 340,000 | 332,000 | AMiller | 13,280 |
| 29 | 14-Apr | 80 Mt. Auburn | 2 | 280,000 | 278,000 | JCompton | 11,120 |
| 30 | 25-Apr | 14 Charles | 4 | 179,000 | 160,000 | AMiller | 6,400 |
| 31 | 27-Apr | 1160 Memorial | 1 | 230,000 | 227,000 | AMiller | 9,080 |
| 32 | 04-May | 130 Crescent | 3 | 405,000 | 397,000 | CGroden | 15,880 |
| 33 | 10-May | 12 Trenton | 2 | 310,000 | 303,000 | AMiller | 12,120 |
| 34 | 11-May | 36 Barnes | 4 | 680,000 | 669,000 | JCompton | 26,760 |
| 35 | 22-May | 234 Third | 2 | 155,000 | 140,000 | JCompton | 5,600 |
| 36 | | | | | | | |
| 37 | | | BR | 2 | | | |
| 38 | | | | | | | |
| 39 | | | | | | | |
| 40 | Highest price paid for a 2-bedroom house, APRIL and MAY: | | | | | | \$332,000 |
| 41 | | | | | | | |

SALES (range A25..G35)

CRIT_RANGE (range B37..C38)

Similar @functions

@MAX finds the greatest value in a range.

@DMIN

@DMIN(*input*,*field*,*criteria*) finds the least value in a field of a database that meets the criteria in the *criteria* range.

Arguments

input is the address or name of a range that contains a database.

field is the field's offset number (a positive integer or 0), the address of a cell that contains an offset number, or a formula that returns an offset number. 1-2-3 assigns the first field (column) in *input* the offset number 0, the second field the offset number 1, and so on.

criteria is a range of at least two rows. The top row of the criteria range contains exact duplicates of the *input* range's field name(s) for those fields for which you are specifying criteria. The row(s) below the field name(s) contain one or more criteria, or conditions, that each entry in the field must meet to be included by @DMIN. (For more information, see "Writing Criteria" in Chapter 14 of the *User's Guide*.)

Uses

Use @DMIN to find the lowest value of only those items that meet certain conditions: the month with the lowest sales for a particular product, for example. You can also use @DMIN to find the earliest date or time in a list of dates or times.

@DMIN is also useful when you need to check for unusually small values, or to find (and discard) the smallest value in some statistical calculations.

Example

In the following illustration, the input range named SALES (A25..F35) lists house sales for April and May. Cell F40 shows the result of using @DMIN to find the lowest sale price of a two bedroom house.

Similar @functions

@MIN finds the least value in a range.

@DSTD

@DSTD(*input,field,criteria*) calculates the standard deviation of the values in a field of a database that meet the criteria in the *criteria* range.

Arguments

input is the address or name of a range that contains a database.

field is the field's offset number (a positive integer or 0), the address of a cell that contains an offset number, or a formula that returns an offset number. 1-2-3 assigns the first field (column) in *input* the offset number 0, the second field the offset number 1, and so on.

criteria is a range of at least two rows. The top row of the criteria range contains exact duplicates of the *input* range's field name(s) for those fields for which you are specifying criteria. The row(s) below the field name(s) contain one or more criteria, or conditions, that each entry in the field must meet to be included by @DSTD. (For more information, see "Writing Criteria" in Chapter 14 of the *User's Guide*.)

Uses

Standard deviation measures the degree to which individual values in a list vary from the mean (average) of all values in the list. The lower the standard deviation, the less individual values vary from the mean, and the more reliable the mean. A standard deviation of 0 indicates that all values in the list are equal.

Use @DSTD to find the standard deviation of values in a field in a database that meet criteria you specify (when those values are the entire population), such as the standard deviation of all sales within a particular sales group, or the standard deviation of all salaries for a certain job level.

@DSTD produces the most accurate results when the population is large.

Notes

@DSTD uses the n, or population, method to calculate the standard deviation of population data. The n method assumes that the selected values are the entire population. If the values are only a sample of the population, the standard deviation is biased because of errors introduced in taking a sample. The n method uses the following formula:

$$\sqrt{\frac{\sum (v_i - avg)^2}{n}}$$

where: n = number of values in *field*
 v_i = the *i*th value in *field*
 avg = average of values in *field*

Standard deviation is the square root of the variance of all individual values from the mean.

Example

In the following illustration, the input range named SALES (A25..G35) lists house sales for April and May. Cell F40 shows the result of using @DSTD to calculate the standard deviation of property taxes for lot sizes greater than .25 acres.

F40: (C2) @DSTD(SALES,5,F37..F38) READY

| | A | B | C | D | E | F | G | H |
|----|--|----|-------|------|-----------|----------|-----|---|
| 25 | ADDRESS | BR | BATHS | LOT | COST | TAX | AGE | |
| 26 | 12 Bartholomew Sq. | 4 | 2 | 0.25 | \$290,000 | \$4,785 | 48 | |
| 27 | 46 Prospect Place | 3 | 1 | 0.40 | 90,000 | 1,485 | 22 | |
| 28 | 103 Cranberry Lane | 2 | 3 | 0.50 | 195,000 | 3,218 | 21 | |
| 29 | 27 Kilburn St. | 2 | 2 | 1.00 | 245,000 | 4,043 | 70 | |
| 30 | 468 Henshaw St. | 4 | 1 | 0.50 | 174,000 | 2,871 | 52 | |
| 31 | 9 Pleasant Street | 1 | 1 | 0.25 | 89,000 | 1,469 | 42 | |
| 32 | 80 E. Beach Street | 3 | 1 | 0.25 | 134,000 | 2,211 | 23 | |
| 33 | 12 Trenton | 2 | 2 | 1.00 | 166,000 | 2,739 | 21 | |
| 34 | 36 Barnes | 4 | 1 | 0.50 | 189,000 | 3,119 | 35 | |
| 35 | 234 Third | 2 | 3 | 0.30 | 140,000 | 2,310 | 60 | |
| 36 | | | | | | | | |
| 37 | | | | | LOT | | | |
| 38 | | | | | >.25 | | | |
| 39 | Standard deviation of property TAX for | | | | | | | |
| 40 | houses with LOT size greater than .25: | | | | | \$736.09 | | |
| 41 | | | | | | | | |

SALES

Similar @functions

@STD calculates the standard deviation of the entire population of values in a range.

@DVAR calculates the population variance of values that meet the criteria you specify.

@DSUM

@DSUM(*input,field,criteria*) calculates the sum of the values in a *field* of a database that meet the criteria in the *criteria* range.

Arguments

input is the address or name of a range that contains a database.

field is the field's offset number (a positive integer or 0), the address of a cell that contains an offset number, or a formula that returns an offset number. 1-2-3 assigns the first field (column) in *input* the offset number 0, the second field the offset number 1, and so on.

criteria is a range of at least two rows. The top row of the criteria range contains exact duplicates of the *input* range's field name(s) for those fields for which you are specifying criteria. The row(s) below the field name(s) contain one or more criteria, or conditions, that each entry in the field must meet to be included by @DSUM. (For more information, see "Writing Criteria" in Chapter 14 of the *User's Guide*.)

Uses

Use @DSUM to find the total of values that meet conditions you specify, such as total sales for a particular month, or total number of goals scored by a soccer team against a particular team over a season.

Example

In the following illustration, the input range named SALES (A25..G35) lists house sales for April and May. Cell F40 shows the result of using @DSUM to calculate the total commission earned by broker A. Miller.

F40: (C0) [W10] @DSUM(SALES,6,CRIT_RANGE) READY

| | A | B | C | D | E | G | |
|----|--|---------------|----|-----------|-----------|----------|------------|
| 25 | DATE | ADDRESS | BR | OFFERED | SOLD | BROKER | COMMISSION |
| 26 | 03-Apr | 467 Brattle | 4 | \$775,000 | \$720,000 | JCompton | \$28,800 |
| 27 | 05-Apr | 183 Hillside | 3 | 325,000 | 318,000 | CGroden | 12,720 |
| 28 | 10-Apr | 64 N. Gate | 2 | 340,000 | 332,000 | AMiller | 13,280 |
| 29 | 14-Apr | 80 Mt. Auburn | 2 | 280,000 | 278,000 | JCompton | 11,120 |
| 30 | 25-Apr | 14 Charles | 4 | 179,000 | 160,000 | AMiller | 6,400 |
| 31 | 27-Apr | 1160 Memorial | 1 | 230,000 | 227,000 | AMiller | 9,080 |
| 32 | 04-May | 130 Crescent | 3 | 405,000 | 397,000 | CGroden | 15,880 |
| 33 | 10-May | 12 Trenton | 2 | 310,000 | 303,000 | AMiller | 12,120 |
| 34 | 11-May | 36 Barnes | 4 | 680,000 | 669,000 | JCompton | 26,760 |
| 35 | 22-May | 234 Third | 2 | 155,000 | 140,000 | JCompton | 5,600 |
| 36 | | | | | | | |
| 37 | | | | | | | |
| 38 | | | | | | | |
| 39 | | | | | | | |
| 40 | Total commission paid to AMiller, APRIL and MAY: | | | | | \$40,880 | |
| 41 | | | | | | | |

BROKER
AMiller

SALES
CRIT_RANGE

Similar @functions

@SUM calculates the sum of the values in a range.

@DVAR

@DVAR(*input,field,criteria*) calculates the population variance of the values in a field of a database that meet the criteria in the *criteria* range.

Arguments

input is the address or name of a range that contains a database.

field is the field's offset number (a positive integer or 0), the address of a cell that contains an offset number, or a formula that returns an offset number. 1-2-3 assigns the first field (column) in *input* the offset number 0, the second field the offset number 1, and so on.

criteria is a range of at least two rows. The top row of the criteria range contains exact duplicates of the *input* range's field name(s) for those fields for which you are specifying criteria. The row(s) below the field name(s) contain one or more criteria, or conditions, that each entry in the field must meet to be included by @DVAR. (For more information, see "Writing Criteria" in Chapter 14 of the *User's Guide*.)

Uses

Variance measures the degree to which individual values in a list vary from the mean (average) of all the values in the list. The lower the variance, the less individual values vary from the mean, and the more reliable the mean. A variance of 0 indicates that all values in the list are equal. Variance is necessary in several ANOVA (analysis of variance) statistical tests.

Use @DVAR to calculate the variance within a group of values in a field in a database that meet criteria you specify (when those values are the entire population), such as the variance of all sales within a particular division, or the variance of salaries for a job level.

@DVAR produces the most accurate results when the population is large.

Notes

@DVAR uses the n, or population, method to calculate variance. The n method assumes that the selected values are the entire population. If the values are only a sample of the population, the variance is biased because of errors introduced in taking a sample. The n method uses the following formula:

$$\frac{\sum (v_i - avg)^2}{n - 1}$$

where: n = number of values in *field*
 v_i = the *i*th value in *field*
 avg = average of values in *field*

Variance is the square of standard deviation.

Example

In the following illustration, the input range named SALES (A25..F35) lists house sales for April and May. Cell E40 shows the result of using @DVAR to calculate the variance of lot size for houses costing more than \$100,000.

E40: (F2) (W9) @DVAR(SALES,3,CRIT_RANGE) EDIT

| | A | B | C | D | E | F | G | H | I |
|----|-----------------------------------|----|-------|------|-----------|-----|---|---|---|
| 25 | ADDRESS | BR | BATHS | LOT | COST | AGE | | | |
| 26 | 12 Bartholomew Sq. | 4 | 2 | 0.25 | \$290,000 | 48 | | | |
| 27 | 46 Prospect Place | 3 | 1 | 0.40 | 90,000 | 22 | | | |
| 28 | 105 Cranberry Lane | 2 | 3 | 0.50 | 195,000 | 21 | | | |
| 29 | 27 Kildurn St. | 2 | 2 | 1.00 | 245,000 | 70 | | | |
| 30 | 468 Hershaw St. | 4 | 1 | 0.50 | 174,000 | 52 | | | |
| 31 | 9 Pleasant Street | 1 | 1 | 0.25 | 89,000 | 42 | | | |
| 32 | 80 E. Beach Street | 3 | 1 | 0.25 | 134,000 | 23 | | | |
| 33 | 12 Trenton | 2 | 2 | 1.00 | 166,000 | 21 | | | |
| 34 | 36 Barnes | 4 | 1 | 0.50 | 189,000 | 35 | | | |
| 35 | 234 Third | 2 | 3 | 0.30 | 140,000 | 60 | | | |
| 36 | | | | | | | | | |
| 37 | | | | | | | | | |
| 38 | | | | | COST | | | | |
| 39 | Variance of LOT size for houses | | | | | | | | |
| 40 | with COST greater than \$100,000: | | | | 0.08 | | | | |
| 41 | | | | | | | | | |

SALES

CRIT_RANGE

Similar @functions

@VAR calculates the population variance of values in a range. @DSTD calculates the population standard deviation of values that meet the criteria you specify.

@ERR

@ERR produces the value ERR.

Uses

@ERR is useful in flagging errors in calculations. It is seldom used by itself. For example, @ERR used as an argument with @IF produces the value ERR when certain conditions exist, such as when a formula results in an unacceptable value (for example, a negative monthly payment).

Notes

ERR is a special value that either 1-2-3 generates to indicate an error in a formula, or you generate with @ERR. ERR will ripple through formulas: a formula that refers to a cell that contains ERR results in ERR, no matter how the value ERR is generated, and any other formula that depends on that formula also results in ERR. When you correct the formula that contains ERR, the results of dependent formulas also become correct.

The label ERR and the value ERR are not equivalent in formulas. For example, the formula $+A2+34 = \text{ERR}$ if cell A2 contains @ERR, but equals 34 if cell A2 contains the label ERR.

Example

$\text{@IF}(B14>3,\text{@ERR},B14) = \text{ERR}$, if the value in cell B14 is greater than 3.

@EXACT

@EXACT(*string1*,*string2*) compares two sets of characters. If the two sets match exactly, @EXACT returns 1 (true); if the two sets are not exactly the same, @EXACT returns 0 (false).

Arguments

string1 and *string2* are text, text formulas, or the addresses or names of cells that contain labels or text formulas.

Uses

Use @EXACT when you need to ensure that a set of characters exactly matches a required entry, such as in macros that compare what a user enters with a required entry before continuing. This allows you to set passwords for macros.

@EXACT is also useful for checking existing entries, as in a database in which you need to ensure that all entries in a field contain the same set of characters.

Notes

@EXACT is more precise than = (the equal operator) in a formula. Unlike =, @EXACT distinguishes between uppercase and lowercase letters and between letters with and without accent marks.

Examples

@EXACT("ATHENS","Athens") = 0 (false).

@EXACT("Overdue",B2) = 1 (true), if cell B2 contains the label Overdue.

@EXACT("400",400) = ERR, because *string2* is not a string.

@IF(@CELL("type",B1)="L"#AND#@CELL("type",C1)="L",@EXACT(B1,C1),0) returns 0 (zero) if B1 or C1 is not a string. You can include the 0 (zero) in another formula or as part of a macro dependency that determines the next commands 1-2-3 performs.

@EXACT("E","e") = 0 (false) because @EXACT is case-sensitive.

@EXP

@EXP(*x*) calculates the value of the constant *e* (approximately 2.718282) raised to the power *x*.

Argument

x is a value, the address or name of a cell that contains a value, or a formula that returns a value less than or equal to 709.

Uses

Use @EXP in scientific calculations that require exponential functions.

Notes

If *x* is greater than 709, the calculation is too large for 1-2-3 to store, and @EXP returns ERR. If *x* is greater than 230, 1-2-3 can calculate and store the value of @EXP, but cannot display it (the cell displays a row of asterisks). 1-2-3 cannot display a value greater than 9.9E99.

Examples

@EXP(-1.25) = 0.286504.

@EXP(B1) = 162754.7, if cell B1 contains the value 12.

@FALSE

@FALSE returns the logical value 0 (false).

Uses

Use @FALSE with @functions such as @IF and @CHOOSE that require a logical value of 0 (false). @FALSE is useful as the *y* argument for @IF, which is the value returned if the condition is not met.

Notes

If a logical statement such as $A1 = B1$ is true, its logical value is 1. If it is false, its logical value is 0.

Example

@IF(A6>500,@TRUE,@FALSE) = 0 if A6 contains a value less than or equal to 500.

@FIND

@FIND(*search-string*,*string*,*start-number*) calculates the position in *string* at which 1-2-3 finds the first occurrence of *search-string*. @FIND begins searching *string* at the position indicated by *start-number*, which represents the offset number of a character in *string*.

Arguments

search-string and *string* are text, text formulas, or the addresses or names of cells that contain text or text formulas.

start-number is a value, the address or name of a cell that contains a value, or a formula that returns a value. *start-number* must be a positive value or 0.

Uses

Use @FIND when you need to determine the position of a particular character or group of characters within a group. @FIND is useful in macros that locate particular sequences of characters for processing.

@FIND is also useful when combined with @MID or @REPLACE to locate and extract or replace text.

Notes

If 1-2-3 does not find *search-string* in *string*, @FIND returns ERR. @FIND also returns ERR if *start-number* is greater than the number of characters in *string*, or if *start-number* is negative.

@FIND is case-sensitive and accent-sensitive; for example, @FIND will not find the search-string "e" in @FIND("e",B1,0) = ERR if B1 contains the string CAMBRIDGE.

Examples

@FIND("P","Accounts Payable",0) = 9 because search-string P is at position 9 in string Accounts Payable.

@FIND("e",@PROPER(B1),0) = 8, if cell B1 contains the string CAMBRIDGE.

@FV

@FV(*payments,interest,term*) calculates the future value of an investment, based on a series of equal *payments*, earning a periodic *interest* rate, over the number of payment periods in *term*.

Arguments

payments is a value, the address or name of a cell that contains a value, or a formula that returns a value.

interest is a value, the address or name of a cell that contains a value, or a formula that returns a value. *interest* must be a decimal or percentage value.

term is a value, the address or name of a cell that contains a value, or a formula that returns a value.

Uses

Use @FV to determine whether an investment will produce the results you want at the end of *term*.

Notes

@FV assumes the investment you are calculating is an ordinary annuity: an investment in which equal *payments* are made at the end of each period in the *term*.

@FV uses the following formula to calculate future value:

$$pmt * \frac{(1 + int)^n - 1}{int} \quad \text{where: } \begin{array}{l} pmt = \text{periodic payment} \\ int = \text{periodic interest rate} \\ n = \text{number of periods} \end{array}$$

If you make each year's contribution on the first day of the year, you would calculate the amount for an annuity due. To calculate the future value of an annuity due, use the formula @FV(*payments,interest,term*)*(1+*interest*).

For example, @FV(2000,0.075,20)*(1+0.075) = \$93,105, the value of your account in 20 years if you make each deposit on the first day of each year.

Example

You plan to deposit \$2,000 each year for the next 20 years into an account to save for retirement. The account pays 7.5% interest, compounded annually; interest is paid on the last day of each year. You make each year's contribution on the last day of the year. To calculate the value of your account in 20 years, you enter `@FV(2000,0.075,20)`. 1-2-3 returns the value \$86,609.

Similar @functions

`@PV` determines the present value of an investment. `@NPV` computes the net present value of an investment, discounting the future value to present value.

@HLOOKUP

`@HLOOKUP(x,range,row-offset)` returns the contents of a cell in a specified row of a horizontal lookup table.

Arguments

x is a value or text, the address or name of a cell that contains a value or text, or a formula that returns a value or text. If *x* is a value that is less than the first value in *range*, `@HLOOKUP` returns `ERR`. If *x* is greater than the last value in *range*, `@HLOOKUP` stops at the last cell in the row and returns the row number of the greatest value. If *x* is text, it must be an exact match of the text in *range*.

range is the range address or range name of the range that contains the table, including the first row. When 1-2-3 locates a cell in the **index row** (the first row in *range*) that contains the value *x* (or the value closest to, but not greater than, *x*), it moves down that column the number of rows specified by *row-offset* and returns the contents of the cell as the answer.

row-offset is a value, the address or name of a cell that contains a value, or a formula that returns a value from 0 through 8,191. *row-offset* represents an offset number. An offset number corresponds to the position the row occupies in *range*. The top row has an offset number of 0, the second row has an offset number of 1, and so on.

Uses

Use `@HLOOKUP` to choose items from a table or to automate data selection for formulas or in macros.

`@HLOOKUP` is useful for finding entries in tables that contain many different choices that depend on different variables, for example, tax tables or sales commission tables.

Using `@HLOOKUP` to provide data for calculations in macros and other applications helps to ensure that the data is correct and that it is entered correctly, as in a macro that retrieves a product's price from a lookup table based on what a user enters.

Notes

@HLOOKUP is case-sensitive and accent-sensitive; for example, if *x* is the text Paris, @HLOOKUP will not match it with a cell that contains PARIS.

Example

The following illustration shows a horizontal lookup table that lists rates for sending a parcel to several cities. The table range is A2..F7. The simple form in B9..D11 uses @HLOOKUP to retrieve a rate based on the city entered in D9 and the parcel type entered in D10.

D11: (C2) EW11J @HLOOKUP(D9,A2..F7,D10)

READY

| | A | B | C | D | E | F | |
|----|-------------|--------------------|-----------|-----------|-------------------------|-----------|---|
| 1 | | — P A R C E L | | | D E S T I N A T I O N — | | ◀ |
| 2 | Parcel type | London | Paris | Frankfurt | New York | Amsterdam | ▶ |
| 3 | 1 | \$18.36 | \$19.33 | \$20.12 | \$9.25 | \$20.25 | ▲ |
| 4 | 2 | \$20.32 | \$21.66 | \$22.03 | \$11.25 | \$22.25 | ▼ |
| 5 | 3 | \$22.44 | \$23.88 | \$24.00 | \$13.25 | \$24.25 | ? |
| 6 | 4 | \$24.14 | \$25.16 | \$25.75 | \$16.85 | \$26.00 | |
| 7 | 5 | \$28.32 | \$29.00 | \$29.50 | \$19.54 | \$29.90 | |
| 8 | | | | | | | |
| 9 | | Parcel destination | Frankfurt | | | | |
| 10 | | Parcel type | | 3 | | | |
| 11 | | Cost | | \$24.00 | | | |

Similar @functions

@VLOOKUP looks up a value in a vertical lookup table. @INDEX looks up a value when you specify offset numbers for both the row and the column. @CHOOSE looks up values in a single-column lookup table.

@HOUR

@HOUR(*time-number*) returns the hour, a value from 0 (midnight) through 23 (23:00 or 11:00 P.M.), of *time-number*.

Argument

time-number is a value, the address or name of a cell that contains a value, or a formula that returns a value from .000000 (midnight) through .999988 (11:59:59 P.M.). Usually, another time @function supplies *time-number*.

Uses

Use @HOUR to extract the hour portion of time values created with @NOW, @TIME, and @TIMEVALUE. The hour portion is useful in calculations that involve whole hours, such as calculating hourly wages or hours elapsed since you began working on a project, or time-stamping a worksheet.

Examples

@HOUR(.51565) = 12 because .51565 is the time number for 12:22:32 P.M.

@HOUR(@TIME(13,45,18)) = 13 (1:00 P.M.), because 13 is the *hour* argument for @TIME(13,45,18).

@IF

@IF(*condition*,*x*,*y*) evaluates *condition* and returns one of two values, depending on the result of the evaluation. If *condition* is true, @IF returns *x*; if *condition* is false, @IF returns *y*.

Arguments

condition is a logical formula, or the address or name of a cell that contains a logical formula.

x and *y* are values, labels, the addresses or names of cells that contain values or labels, or formulas that return values or labels.

Uses

Use @IF when calculations or processing depend on the result of a test, for example, a test to determine savings-account interest at one rate for balances of \$1,000.00 and over and another rate for balances below that amount.

@IF is useful when combined with @ERR and @NA to document errors or missing data in formulas. It is also useful in preventing ERR, NA, and calculation errors in situations where data may be missing or inaccurate (for example, to prevent division by zero, or, with @ISSTRING, to prevent a label from being included in calculations such as when using @AVG).

Notes

You can nest @IF functions (place one @IF function inside another @IF function) to create a complex condition.

Examples

@IF(BALANCE>=0,BALANCE,"Overdrawn") returns the value in the named range BALANCE when the value in BALANCE is 0 or positive; or returns the label Overdrawn when the value in BALANCE is negative.

@IF(TOT>10000,TOT*.15,@IF(TOT>5000,TOT*.10,TOT*.02)) returns a commission rate based on three levels of sales: total sales greater than \$10,000, total sales greater than \$5,000, and total sales less than or equal to \$5,000.

@INDEX

`@INDEX(range,column-offset,row-offset)` returns the contents of the cell located at the intersection of a specified *column-offset* and *row-offset* of a range.

Arguments

range is a cell address or range name.

column-offset is the offset number of the column that @INDEX uses. 1-2-3 assigns the first column in *range* the offset number 0, the second column the offset number 1, and so on.

row-offset is the offset number of the row that @INDEX uses. 1-2-3 assigns the first row in *range* the offset number 0, the second row the offset number 1, and so on.

column-offset and *row-offset* are values, the addresses or names of cells that contain values, or formulas that return values from 0 through 8,191.

Uses

Use @INDEX when you want to use a lookup table but need to use the relative positions (offset numbers) of the rows or columns, instead of specified values, for both arguments.

Notes

If any argument is larger than the number of columns or rows in *range*, or if any argument is negative, @INDEX returns ERR.

Example

In the following illustration, the table named INCREASE (A3..E8) shows salary increases based on employee performance ratings. @INDEX(INCREASE,2,3) entered in E10 finds the salary increase for an employee who has a rating of 3 and has a salary level of 2. @INDEX(INCREASE,1,2) finds the salary increase for an employee who has a rating of 2 and a salary level of 1.

D10: (P0) @INDEX(INCREASE,2,3) READY

| | A | B | C | E | F | G | H |
|----|---|---|---|---|---|---|---|
| 1 | | | | | | | |
| 2 | | | | | | | |
| 3 | | | | | | | |
| 4 | | | | | | | |
| 5 | | | | | | | |
| 6 | | | | | | | |
| 7 | | | | | | | |
| 8 | | | | | | | |
| 9 | | | | | | | |
| 10 | | | | | | | |
| 11 | | | | | | | |

Rating 1 2 3 4

10% 9% 8% 7%

7% 6% 5% 4%

3% 2% 1% 0%

0% 0% 0% 0%

INCREASE

@INDEX(INCREASE,2,3) ⇒ 5% Salary 2, rating 3

@INDEX(INCREASE,1,2) ⇒ 7% Salary 1, rating 2

@INT

@INT(*x*) returns the integer portion of *x*.

Argument

x is a value, the address or name of a cell that contains a value, or a formula that returns a value.

Uses

Use @INT in calculations that require only the integer portion of values (for example, a calculation of staffing levels).

Notes

@INT truncates a value to its integer portion and eliminates the decimal portion. Use /Range Format Fixed 0 or /Worksheet Global Format Fixed 0 to display values as integers in the worksheet if you want 1-2-3 to calculate values using their full precision. @INT does not round to an integer.

@INT differs from @ROUND in that @ROUND can retain the decimal portion of a value and always rounds the value.

Examples

@INT(35.67) = 35.

@INT(@NOW) = the date number for the current date and time, because the time portion is a decimal value.

F7-@INT(F7) = .8876, if cell F7 contains the value 14.8876.

@IRR

@IRR(*guess,range*) calculates the internal rate of return for a series of cash flow values generated by an investment. The **internal rate of return** is the percentage rate that equates the present value of an expected future series of cash flows to the initial investment.

Arguments

guess is a decimal or percentage value, the address or name of a cell that contains a decimal or percentage value, or a formula that returns a decimal or percentage value. *guess* represents your estimate of the internal rate of return. In most cases, *guess* should be a decimal between 0 (0%) and 1 (100%). With very large cash flows, make *guess* as accurate as possible.

range is the address or name of a range that contains the cash flows. 1-2-3 considers negative numbers as cash outflows and positive numbers as cash inflows. Normally,

the first cash-flow amount in the range is a negative number (a cash outflow) that represents the investment. 1-2-3 ignores empty cells in the range, and treats cells that contain labels as 0.

Uses

Use @IRR to determine the profitability of an investment. Combine @IRR with other financial functions, such as @NPV, to assess an investment.

Notes

1-2-3 assumes the cash flows are received at regular, equal intervals.

@IRR uses a series of approximations, starting with your *guess* value, to calculate the internal rate of return. Start with a *guess* that you feel is reasonable for the internal rate of return. More than one solution may be possible, so try another *guess* if the result doesn't seem correct.

If @IRR cannot approximate the result to within 0.0000001 after 30 calculation iterations, the result is ERR. If your guesses continue to return ERR, use @NPV to determine a better *guess*.

Depending on the *guess* you provide, @IRR can return several different answers. @IRR can also return several different answers if you specify negative cash flows.

Example

Suppose you want to calculate the internal rate of return of a \$10,000 investment (the value -10000 entered in cell B3) that is followed by 12 monthly payments of \$1500 (B4..B15). Using a guess of 12%, your formula would be @IRR(.12,B3..B15) and 1-2-3 would return 10.45%.

Similar @functions

@NPV calculates the net present value of a series of future cash flows. @PV computes the present value of an annuity based on a series of equal payments. @FV calculates the future value of an annuity. @RATE returns the periodic interest rate necessary for an investment to grow to a future value.

@ISAAF

@ISAAF(*name*) tests *name* for an attached add-in @function. If *name* is an attached add-in @function, @ISAAF returns 1 (true); if *name* is not a defined add-in @function, @ISAAF returns 0 (false).

Argument

name is the name of the add-in @function you want to test. *name* is a literal string, a text formula, or a reference to a cell that contains a label. Do not include the initial @ (at sign) in *name*.

Uses

Use @ISAAF in a macro to determine if you need to attach the add-in program required by an add-in @function.

Notes

@ISAAF does not recalculate when you select /Add-In Attach or /Add-In Detach.

If @ISAAF returns 0 (false) because the add-in program required by the @function is not attached, you must attach the add-in program and then retrieve the worksheet again to use the add-in @function.

Example

@ISAAF("dsum") = 0 because @DSUM is a built-in 1-2-3 @function, not an add-in @function.

@ISAPP

@ISAPP(*name*) tests *name* for an attached add-in. If *name* is an attached add-in, @ISAPP returns 1 (true); if *name* is not an attached add-in, @ISAPP returns 0 (false).

Argument

name is the name of the add-in you want to test. *name* is a literal string, a text formula, or a reference to a cell that contains a label. Do not include the .ADN extension in *name*.

Uses

Use @ISAPP in a macro to determine if the add-in program you want to use is attached.

Notes

@ISAPP returns 1 (true) only for add-ins you invoke using /Add-In Invoke. For add-ins that only define add-in @functions, or any add-in installed in your driver set, @ISAPP returns 0 (false). Use @ISAAF to test for add-in @functions.

@ISAPP does not recalculate when you select /Add-In Attach or /Add-In Detach.

Example

@ISAPP("wysiwyg") = 1 if the Wysiwyg add-in is currently attached.

@ISERR

@ISERR(*x*) tests *x* for the value ERR. If *x* is the value ERR, @ISERR returns 1 (true); if *x* is not the value ERR, @ISERR returns 0 (false).

Argument

x can be any value, single-cell location, text, or condition.

Uses

Using @ISERR with @IF in formulas stops the ripple-through effect of the value ERR. For example, @IF(@ISERR(C3),0,C3) returns 0 if cell C3 contains the value ERR. The formula returns the contents of cell C3 if cell C3 contains any other value.

Use @ISERR to block errors that arise from division by zero. For example, the formula @IF(@ISERR(A1/A2),0,A1/A2) tests the result of the division A1/A2. If the result is the value ERR, the formula returns 0. If the result is any other value, the formula returns that result.

Notes

ERR is a value 1-2-3 returns when an error occurs, or you generate with @ERR. The value ERR is not equivalent to the label ERR. @ISERR does not recognize the label ERR.

@ISNA

@ISNA(*x*) tests *x* for the value NA. If *x* is the value NA, @ISNA returns 1 (true); if *x* is not the value NA, @ISNA returns 0 (false).

Argument

x can be any value, single-cell location, text, or condition.

Uses

Using @ISNA in formulas stops the ripple-through effect of the value NA. For example, @IF(@ISNA(C3),0,C3) returns 0 if cell C3 contains the value NA; the formula returns the contents of cell C3 if cell C3 contains any other value.

Notes

The value NA is not equivalent to the label NA. @ISNA does not recognize the label NA. The value NA in a cell generates NA in all formulas that refer to that cell.

Example

The following subroutine QTY_IN tests whether the entry in QTY is a value; if it is, processing transfers to the subroutine PRICE_IN. If QTY does not contain a value, QTY_IN requests a new entry.

PRICE_IN uses @ISNA to determine whether or not a discount applies. If @ISNA is true, the macro calculates the discount. If DISCOUNT does not contain NA and QTY does not contain ERR, the subroutine multiplies the values in the two cells and enters the result in the cell named TOTAL.

```
...
QTY_IN      {GETNUMBER "Enter quantity: ",QTY}~
             {IF @ISERR(QTY)}{BRANCH QTY_IN}
PRICE_IN    {GETNUMBER "Enter price: ",PRICE}~
             {IF @ISERR(PRICE)}{LET PRICE,"No price entered"}~
             {GOTO}TOTAL~
             {IF @ISNUMBER(PRICE)<>1}{BRANCH TOTAL_IN}
             {IF @ISNA(DISCOUNT)}{BRANCH NODISCOUNT}
             +QTY*(PRICE*DISCOUNT)~{QUIT}

NODISCOUNT  +QTY*PRICE~{QUIT}
TOTAL_IN    {LET TOTAL,PRICE}~
...

```

@ISNUMBER

@ISNUMBER(*x*) tests *x* to see if it contains a value. If *x* is a value, NA, ERR, or blank, @ISNUMBER returns 1 (true). If *x* is a string, a range, @ISNUMBER returns 0 (false).

Argument

x can be any value, single-cell location, text, or condition.

Uses

@ISNUMBER is useful in macros to make sure a user enters the correct type of information (values or labels).

Use @ISNUMBER in formulas to avoid the values ERR and NA.

Example

The following subroutine QTY_IN tests whether the entry in QTY is a value; if it is, processing transfers to the subroutine PRICE_IN. If QTY does not contain a value, QTY_IN requests a new entry.

PRICE_IN uses @ISNUMBER to determine whether or not the value entered in PRICE is a number. If PRICE does not contain a number, the macro transfers to the subroutine TOTAL_IN.


```

...
QTY_IN      {GETNUMBER "Enter quantity: ",QTY}~
            {IF @ISERR(QTY)}{BRANCH QTY_IN}
PRICE_IN    {GETNUMBER "Enter price: ",PRICE}~
            {IF @ISERR(PRICE)}{LET PRICE,"No price entered"}~
            {GOTO}TOTAL~
            {IF @ISNUMBER(PRICE)<>1}{BRANCH TOTAL_IN}
            {IF @ISNA(DISCOUNT)}{BRANCH NODISCOUNT}
            +QTY*(PRICE*DISCOUNT)~{QUIT}
NODISCOUNT  +QTY*PRICE~{QUIT}
TOTAL_IN    {LET TOTAL,PRICE}~
...

```

@ISSTRING

@ISSTRING(*x*) tests *x* to see if it is text or a label. If *x* is text or a cell that contains a label, @ISSTRING returns 1 (true); if *x* is a value, NA, ERR, or blank, @ISSTRING returns 0 (false).

Argument

x can be any value, single-cell location, text, or condition.

Uses

@ISSTRING is useful in macros to make sure a user enters the correct type of information (values or labels).

Use @ISSTRING in formulas to avoid the values ERR and NA.

Example

In the following subroutine, CHKSTR checks the contents of the cell named CUSTOMER. If CUSTOMER contains a label (@ISSTRING(CUSTOMER) = 1), the subroutine branches to a macro named FILEORDER. If CUSTOMER does not contain a label, the subroutine requests a new entry.

```

...
CHKSTR      {IF @ISSTRING(CUSTOMER)}{BRANCH FILEORDER}
            {GETLABEL "Enter customer name: ",CUSTOMER}
            {BRANCH CHKSTR}
...

```

@LEFT

@LEFT(*string*,*n*) returns the first *n* characters in *string*.

Arguments

string is text, a text formula, or the address or name of a cell that contains a label or text formula.

n is a positive integer or 0, or the address or name of a cell that contains a positive integer or 0, or a formula that returns a positive integer or 0. If *n* is 0, the result is an empty string. If *n* is greater than the length of *string*, @LEFT returns the entire *string*.

Uses

@LEFT is useful for copying only part of a label into another cell, starting at the beginning of the label (for example, for separating titles such as Dr. and Ms. from names).

In a macro, @LEFT can extract parts of labels the user enters to store them in a database, to use in a subroutine, or to alter the macro itself.

Use @LEFT with @FIND when you do not know the exact value for *n*, or when *n* may vary.

Notes

1-2-3 counts punctuation and spaces as characters in @LEFT.

Example

@LEFT(EUROPE_PHONE,3) = the country code for the telephone number in EUROPE_PHONE.

@LEFT(A1,@FIND("•",A1,0)) = the first name in cell A1 (for example, Cara if cell A1 contains the name Cara Groden). The • (bullet) represents one space.

@LENGTH

@LENGTH(*string*) counts the number of characters in *string*.

Argument

string is text, a text formula, or the address or name of a cell that contains a label or text formula.

Uses

Use @LENGTH with @TRIM to find the length of a string without including leading, trailing, or consecutive spaces.

Use @LENGTH to determine the total length of a line before printing.

@LENGTH is also useful in any application in which labels should be a certain length, such as zip codes and purchase order numbers.

Use @LENGTH to determine spacing for positioning labels in presentations.

Examples

@LENGTH("refrigerator") = 12.

@LENGTH(A5&G12) = the total number of characters in cells A5 and G12, if both A5 and G12 contain labels.

@LN

@LN(x) calculates the natural logarithm (base e) of x .

Argument

x is a value, the address or name of a cell that contains a value, or a formula that returns a value greater than 0.

Uses

Use @LN in scientific calculations that require natural logarithms, such as compound growth or loss.

Notes

A natural logarithm is one that uses the number e (approximately 2.718282) as a base.

Examples

@LN(2) = 0.693147.

@LN(B3) = 1.098612, if cell B3 contains the value 3.

@LOG

@LOG(x) calculates the common logarithm (base 10) of x .

Argument

x is a value, the address or name of a cell that contains a value, or a formula that returns a value greater than 0.

Uses

Use @LOG in any calculation that requires a common logarithm, such as a formula to find a root of a number.

Examples

@LOG(1000) = 3 (because $10^3 = 1000$).

$10^{(@\text{LOG}(8)/3)} = 2$, the cube root of 8.

@LOG(B3) = 0.60206, rounded, if cell B3 contains the value 4.

@LOWER

@LOWER(*string*) converts all uppercase letters in *string* to lowercase.

Argument

string is text, a text formula, or the address or name of a cell that contains a label or text formula.

Uses

@LOWER is useful when you combine labels from several sources, and you want the case of the labels to be consistent throughout the worksheet.

Notes

If you selected the ASCII collating sequence when you installed 1-2-3, capitalization affects the order of labels when you use /Data Sort. Two otherwise identical labels may not appear together if their capitalization is different.

Example

@LOWER("LOTUS Sales Forecast") = lotus sales forecast.

@MAX

@MAX(*list*) finds the greatest value in *list*.

Argument

list is a series of values, or the addresses or names of cells that contain values, separated by argument separators.

Uses

Use @MAX to find the greatest value in a series. @MAX is also useful when you need to check for unusually large values or to find the largest value in some statistical calculations.

Notes

@MAX ignores blank cells in the range, and treats cells that contain labels as 0.

Examples

@MAX(55,39,50,28,67,43) = 67.

@MAX(A1..C10) = the greatest value in A1..C10.

@SUM(DATA_VALUES)-@MAX(DATA_VALUES) = the total of all data values, excluding the largest.

@MID

@MID(*string*,*start-number*,*n*) returns *n* characters from *string*, beginning with the character at *start-number*.

Arguments

string is text, a text formula, or the address or name of a cell that contains a label or text formula.

start-number is any positive value or 0, or the address or name of a cell that contains a positive value or 0, or a formula that returns a positive value or 0. *start-number* is the offset number of a character in *string*. If *start-number* is greater than the length of *string*, the result of @MID is an empty string.

n is a positive integer or 0, the address or name of a cell that contains a positive integer or 0, or a formula that returns a positive value or 0. If *n* is 0, the result of @MID is an empty string. If *n* is greater than the length of *string*, 1-2-3 returns all the characters from *start-number* to the end of *string*.

Uses

Use @MID to extract a part of a label that is not located at the beginning or end of the label. To extract part of a label when you do not know its *start-number*, use @MID with @FIND.

@MID is useful in macros to store parts of labels the user enters, to create subroutine calls, or to alter the macro itself.

Notes

@MID copies punctuation and spaces included in *string*.

Examples

@MID("Daily Account Balance",6,7) = Account.

@MID(A2,@FIND("•",A2,0)+1,50) = the last name in cell A2 (for example, O'Brien if cell A2 contains the name Eoghan O'Brien). The • (bullet) represents one space. Use a large number (such as 50) for *n* if you do not know the length of *string*; 1-2-3 ignores the extra length.

@MIN

@MIN(*list*) finds the least value in *list*.

Argument

list is a series of values, or the addresses or names of cells that contain values, separated by argument separators.

Uses

Use @MIN to find the lowest value in a series, for example, the earliest date or the youngest age. @MIN is also useful when you need to check for unusually small values, or to find the smallest value in some statistical calculations.

Notes

@MIN ignores blank cells in the range, and treats cells that contain labels as 0.

Examples

@MIN(55,39,50,28,67,43) = 28.

@MIN(A1..C10) = the smallest value in A1..C10.

@SUM(DATA_VALUES)-@MIN(DATA_VALUES) returns the total of all data values, excluding the smallest.

@MINUTE

@MINUTE(*time-number*) extracts the minutes, a value from 0 through 59, from *time-number*.

Argument

time-number is a value, the address or name of a cell that contains a value, or a formula that returns a value from .000000 (midnight) through .999988 (11:59:59 P.M.). Usually, another time @function supplies *time-number*.

Uses

Use @MINUTE to extract only the minutes portion of time values created with @TIMEVALUE, @NOW, or @TIME. The minutes portion is useful in calculations that involve only minutes, such as the time that has elapsed since the start of an application.

Notes

@TIME(*hour,minutes,seconds*) returns the time number for the time you specify. For example, @TIME(8,30,0) returns the time number 0.354167.

Examples

@MINUTE(0.333) = 59 because 0.333 is the time number for 7:59:31.

@MINUTE(@TIME(11,15,45)) = 15 because 15 is the *minutes* argument for @TIME(11,15,45).

@MOD

@MOD(*x,y*) calculates the remainder (modulus) of x/y .

Arguments

x and *y* are values, the addresses or names of cells that contain values, or formulas that return values.

Uses

Use the mathematical @function @MOD to determine the remainder in a calculation (for example, the amount of material left over from a production run). You can also use @MOD to determine whether a number is even or odd: with a divisor of 2, an even number has no remainder.

Use @MOD to calculate the day of the week by entering a date number as *x* and 7 (the number of days in a week) as *y*. 1-2-3 assigns a number to each day of the week: 0 for Saturday, 1 for Sunday, and so on to 6 for Friday. If you divide the date number by 7, the remainder is the day of the week. For example, @MOD(@DATE(85,11,18),7) = 2; November 18, 1985 was a Monday.

Notes

The sign (+ or -) of *x* (the dividend) determines the sign of the result. If *x* is 0, @MOD returns 0.

If *y* (the divisor) is 0, @MOD returns ERR.

1-2-3 uses the following formula to calculate the modulus: $x-(y*\text{INT}(x/y))$.

Examples

@MOD(9,4) = 1.

@MOD(-14,3) = -2.

@MONTH

@MONTH(*date-number*) extracts the month (1 to 12) from *date-number*.

Argument

date-number is a value, the address or name of a cell that contains a value, or a formula that returns a value from 1 (January 1, 1900) through 73050 (December 31, 2099).

Uses

Use @MONTH to track months rather than entire dates. Use @MONTH to keep track of events that happen in certain months, such as quarterly sales summaries, or to calculate the months that have elapsed between events.

@MONTH can also supply the *month* argument for other date and time @functions that build on previously calculated dates.

Notes

You can use another date and time @function to supply the argument for @MONTH, as in the following examples.

Examples

@MONTH(@DATE(85,3,27)) = 3.

@MONTH(20181) = 4, because the date number 20181 is the date 02-Apr-55.

@MONTH(@NOW) = 11, if the current month is November.

@N

@N(*range*) returns the entry in the first cell of *range* as a value. If the cell contains a label, @N returns the value 0.

Argument

range is a cell or range address, or a range name.

Uses

@N is useful to check user entries in a macro.

Examples

+100+@N(B5..F5) = 885, if cell B5 contains the value 785.

@N(A5)+@N(B5) = 785, if A5 contains a label and B5 contains the value 785.

@NA

@NA returns the value NA (not available).

Uses

@NA is useful when you are building a worksheet that will contain data that you have not yet determined. Use @NA to flag cells where you will enter the data; formulas that refer to those cells result in the value NA until you supply the correct data.

@NA is also useful to determine which formulas depend on a particular cell. You can also use the Auditor to determine which formulas depend on a particular cell. For more information on the Auditor, see "Exploring Formulas with Auditor," in Chapter 3 of the *User's Guide*.

Notes

NA is a special value that either 1-2-3 or you generate to indicate that a value needed to complete a formula is not available.

NA will ripple through formulas: any formula that refers to a cell that contains NA results in NA (no matter how the value NA is generated), unless the cell contains @ERR. @ERR takes precedence over @NA. This ripple-through effect also means that when you provide the previously unavailable value to a formula that contains NA, the results of dependent formulas also become correct.

The label NA and the value NA are not equivalent in formulas. For example, the formula $+A2+34 = NA$, if cell A2 contains @NA, but equals 34 when cell A2 contains the label NA (because labels have the value zero).

Examples

@IF(@CELL("type",B14)="b",@NA,B14) = the value NA when B14 is blank.

@IF(B14>3,@NA,B14) = NA, if the value in cell B14 is greater than 3.

@NOW

@NOW calculates the number that corresponds to the current date and time. This includes both a date number (integer portion) and a time number (decimal portion).

Uses

@NOW produces a record of the current date and time. This record is useful in any calculation that requires the current date, for example, determining the time a payment has been overdue. Use @NOW with EDIT and CALC to create a fixed record of a date and time for time-stamping worksheets or in calculations of elapsed time.

Notes

Format the value of @NOW in any of the date or time formats. If you format @NOW as a date, 1-2-3 displays only the date (integer) portion of the date and time number. If you format @NOW as time, 1-2-3 displays only the time (decimal) portion of the date and time number. In both cases, 1-2-3 stores the entire date and time number.

1-2-3 recalculates @NOW each time you recalculate your work. If you use /Worksheet Global Recalculation to set recalculation to Automatic, 1-2-3 recalculates @NOW whenever it recalculates another value.

1-2-3 uses the date and time from the current date and time settings on your computer.

Examples

@NOW = 29221.0 at midnight on January 01, 1980.

@NOW = 32688.395 (rounded) at 9:28 A.M., June 29, 1989.

@NPV

@NPV(*interest*,*range*) calculates the net present value of a series of future cash-flow values (*range*), discounted at a fixed periodic *interest* rate.

Arguments

interest is a decimal or percentage value, the address or name of a cell that contains a decimal or percentage value, or a formula that returns a decimal or percentage value.

range is the single-row or single-column range that contains the cash flows.

Uses

Use @NPV to evaluate an investment or to compare one investment with others.

@NPV calculates the initial investment necessary to achieve a certain cash outflow at a certain rate.

Notes

@NPV uses the following formula to calculate the net present value:

$$\sum_{i=1}^n \frac{v_i}{(1 + int)^i}$$

where: $v_1 \dots v_n$ = series of cash flows in range
 int = interest rate
 n = number of cash flows
 i = current iteration (1 through n)

@NPV assumes that the cash outflows occur at equal time intervals, that the first cash outflow occurs at the end of the first period, and that subsequent cash flows occur at the end of subsequent periods.

To determine the net present value of an investment where you make an initial cash outflow immediately, followed by a series of future inflows, factor the initial outflow separately, because it is not affected by the interest. To do this, add the initial cash outflow to the result of the @NPV calculation.

@NPV returns ERR if *range* is not a single row or a single column.

Example

The following illustration shows how you can use @NPV to discount a series of irregular distributions invested at an 11.5% annual rate to today's dollars. The range named DISTRIBUTIONS (column B) contains the list of cash flows. To provide @NPV with the correct number of periods, months in which no distribution is made are included in the range (cells B2, B3 and B12 contain 0). The distributions are made monthly, so @NPV requires the interest (discount rate) to be expressed as a monthly percentage; the cell named DISCOUNT (cell H3, calculated as H2/12) contains the monthly rate.

H6: (C2) [M11] @NPV(DISCOUNT,DISTRIBUTIONS) READY

| | A | B | C | D | E | F | G |
|----|-------|------------|---|---------------------------------|---|-------------|---|
| 1 | Month | Cash Flows | | | | | |
| 2 | 1 | 80.00 | | Annual interest (discount) rate | | 11.50% | |
| 3 | 2 | 80.00 | | Interest (discount) per period | | 0.96% | |
| 4 | 3 | \$2,500.00 | | Total capital distribution | | \$41,000.00 | |
| 5 | 4 | \$2,500.00 | | | | | |
| 6 | 5 | \$3,000.00 | | Net present value | | \$38,084.13 | |
| 7 | 6 | \$5,000.00 | | | | | |
| 8 | 7 | \$6,000.00 | | | | | |
| 9 | 8 | \$9,000.00 | | | | | |
| 10 | 9 | \$3,000.00 | | | | | |
| 11 | 10 | \$2,500.00 | | | | | |
| 12 | 11 | 80.00 | | | | | |
| 13 | 12 | \$7,500.00 | | | | | |
| 14 | | | | | | | |
| 15 | | | | | | | |
| 16 | | | | | | | |
| 17 | | | | | | | |

Similar @functions

@PV determines the present value of an annuity based on a series of equal payments.

@FV calculates the future value of an annuity.

@PI

@PI produces the value π , which 1-2-3 approximates as 3.1415926536. π is the ratio of the circumference of a circle to its diameter.

Uses

Use @PI in calculations that require the value π , particularly in conjunction with trigonometric functions.

Examples

@PI = 3.1415926536.

@PI*4^2 = 50.26548, the area of a circle with a radius of 4.

@SIN(@PI/6) = .5, the sine of an angle that is 30°.

@PMT

@PMT(*principal, interest, term*) calculates the payment on a loan (*principal*) at a given *interest* rate for a specified number of payment periods (*term*).

Arguments

principal is the value of the loan.

interest is the periodic interest rate.

term is the number of payment periods.

principal and *term* are values, the addresses or names of cells that contain values, or formulas that return values. *interest* is a decimal or percentage value, the address or name of a cell that contains a decimal or percentage value, or a formula that returns a decimal or percentage value.

Uses

Use @PMT to calculate the payment necessary to amortize a loan or the periodic payment returned by an annuity.

Notes

The period used to calculate *interest* must be the same period used for *term*; for example, if you are calculating a monthly payment, enter the interest and term in monthly increments. Usually, this means you must divide the *interest* by 12 and multiply the number of years in *term* by 12.

@PMT uses the following formula to calculate periodic payment:

$$prin * \frac{int}{1 - (int + 1)^{-n}}$$

where: *prin* = principal
int = periodic interest rate
n = term

@PMT assumes payments occur at the end of each payment period (an ordinary annuity).

If you make payments at the beginning of each month (an annuity due), you can calculate the amount of the periodic payment on the annuity due with the formula @PMT(*principal, interest, term*)/(1+*interest*).

Example

Suppose an \$8,000 auto loan is available for 3 years at an annual interest rate of 14%, compounded monthly. To determine your monthly payment, you use `@PMT(8000,0.14/12,36)`. 1-2-3 returns \$273.42, the monthly payment.

To calculate the amount of the periodic payment on an annuity due, you use `@PMT(8000,0.14/12,36)/(1 + 0.14/12)`. 1-2-3 returns \$270.27, the monthly payment.

@PROPER

`@PROPER(string)` capitalizes the first letter of each word in *string* and converts the remaining letters to lowercase.

Argument

string is text, a text formula, or the address or name of a cell that contains a label or text formula.

Uses

`@PROPER` is useful when you combine data from several sources and want labels to be consistent throughout the worksheet file. If you selected the ASCII collating sequence when you installed 1-2-3, use `@PROPER` in a database to ensure consistent capitalization of names before sorting the names or before using the names to create mailing labels.

Notes

If you selected the ASCII collating sequence when you installed 1-2-3, capitalization affects the order of labels when you use /Data Sort. Two otherwise identical labels may not appear together if their capitalization is different.

If *string* is a blank cell, `@PROPER` returns ERR.

Example

`@PROPER(A7&" ;•" &G7)` returns Morton Smith; Athens, Georgia if A7 contains the label MORTON SMITH, and G7 contains the label Athens, georgia. The ; (semicolon) is in quotation marks and is therefore treated as a literal string instead of an argument separator. The • (bullet) represents one space.

@PV

`@PV(payments,interest,term)` determines the present value of an investment. `@PV` calculates the present value based on a series of equal *payments*, discounted at a periodic *interest* rate over the number of periods in *term*.

Arguments

payments is the value of the equal investments.

interest is the periodic interest rate.

term is the number of payment periods.

payments and *term* are values, the addresses or names of cells that contain values, or formulas that return values. *interest* is a decimal or percentage value, the address or name of a cell that contains a decimal or percentage value, or a formula that returns a decimal or percentage value.

Uses

Use @PV to evaluate an investment or to compare one investment with others. @PV is useful in comparing different types of investments, for example, comparing a single payment investment from a pension fund with a series of periodic payments. Use @PV with @PMT to create an amortization table.

Notes

@PV complements @PMT: @PV tells you how large a loan you can take out, given the constraint of the size of the monthly payment you can afford. Conversely, @PMT tells you how large your monthly payment will be, given the constraint of the size of the loan you want to take out.

@PV calculates present value with the following formula:

$$pmt * \frac{1 - (1 + int)^{-n}}{int}$$

where: *pmt* = periodic payment
int = periodic interest rate
n = term

@PV assumes each payment is made at the end of the period. To calculate the present value of an annuity due (a payment made at the beginning of each period), use the following formula: @PV(*payments,interest,term*)*(1 + *interest*).

Example

You won \$1,000,000. You can receive either 20 annual payments of \$50,000 at the end of each year or a single payment of \$400,000 instead of the \$1,000,000 annuity. If you were to accept the annual payments of \$50,000, you assume that you would invest the money at a rate of 8%, compounded annually.

To find out which option is worth more in today's dollars, you enter @PV(50000,.08,20). 1-2-3 returns \$490,907, which tells you that the \$1,000,000 paid over 20 years is worth \$490,907 in today's dollars.

Similar @functions

@FV calculates the future value of an investment based on a series of equal payments. @NPV computes the net present value of an investment, discounting future value to present value.

@RAND

@RAND generates a random value between 0 and 1. 1-2-3 calculates @RAND to 17 decimal places. Each time 1-2-3 recalculates the worksheet, @RAND generates a new random value.

Uses

@RAND is useful in situations that require random numbers (for example, generating test data for simulations or choosing returns to audit).

Notes

To convert the value generated by @RAND to a fixed value, use /Range Value or highlight the cell that contains @RAND and then press F2 (EDIT) followed by F9 (CALC) and ENTER.

To generate random values in different numeric intervals, multiply @RAND by the size of the interval. Use @ROUND or @INT with the result to create random whole numbers.

Examples

@RAND = 0.419501 or any value between 0 and 1.

@RAND*10 = 4.19501 or any value between 0 and 10.

@INT(@RAND*50)+1 = 21, or any integer from 0 to 50, inclusive.

@ROUND(@RAND*100,0) = 42, or any integer from 0 to 100, inclusive.

@RATE

@RATE(*future-value*,*present-value*,*term*) returns the periodic interest rate necessary for an investment (*present-value*) to grow to a *future-value* over the number of compounding periods in *term*.

Arguments

future-value, *present-value*, and *term* are values, the addresses or names of cells that contain values, or formulas that return values.

Uses

Use @RATE to evaluate an investment or to compare one investment with others. @RATE is useful when you need to calculate the compound rate of an investment, such as the value of a stock, or in any situation where you know the initial value, the final value, and the time elapsed between the two.

Notes

@RATE uses the following formula to calculate the periodic interest rate:

$$\left(\frac{fv}{pv}\right)^{1/n} - 1$$

where: fv = future value
 pv = present value
 n = term

Example

You invested \$10,000 in a bond that matures in five years and has a maturity value of \$18,000. Interest is compounded monthly. To determine the periodic interest rate for this investment, you enter @RATE(18000,10000,60). 1-2-3 returns .984%, the periodic (monthly) interest rate. To determine the annual interest rate, use the formula $((1+\text{@RATE}(18000,10000,60))^{12})-1$. This yields an annual interest rate of 12.47%.

@REPEAT

@REPEAT(*string*,*n*) duplicates *string* the number of times specified by *n*.

Arguments

string is text, a text formula, the address or name of a cell that contains a label or text formula.

n is a positive integer, the address or name of a cell that contains a positive integer, or a formula that returns a positive integer.

Uses

Use @REPEAT to repeat any printable character, including special mathematical, graphics, or foreign language symbols. @REPEAT is useful in presentations to create lines on either side of labels.

Notes

@REPEAT duplicates the string as many times as you specify; it is not limited by the current column width. This differs from using the repeating label-prefix character \ (backslash), which repeats a label only as many times as will fill the current column.

Examples

@REPEAT("Hello•",3) returns Hello Hello Hello. The • (bullet) represents one space.

@REPEAT("_",@LENGTH(B2)) underlines a label in cell B2 if the current cell is B3.

@REPLACE

@REPLACE(*original-string*,*start-number*,*n*,*new-string*) replaces *n* characters in *original-string* with *new-string*, beginning at *start-number*.

Arguments

original-string and *new-string* are text, text formulas, or the addresses or names of cells that contain labels or text formulas.

start-number is the offset number of a character in *original-string*. It can be any positive value or 0, the address or name of a cell that contains a positive value or 0, or a formula that returns a positive value or 0. If *start-number* is greater than the length of *original-string*, @REPLACE appends *new-string* to *original-string*.

n is a positive integer or 0, the address or name of a cell that contains a positive integer or 0, or a formula that returns a positive integer or 0. If *n* is 0, @REPLACE appends *new-string* to *original-string*.

Uses

Use @FIND with @REPLACE to search for and replace a label or to calculate an unknown *start-number*.

@REPLACE is useful when you need to replace one set of characters with another (for example, to change the area code in a database of telephone numbers).

Notes

@REPLACE counts all the characters in a string, including spaces and punctuation. If you use @REPLACE to append or insert text, remember to include the necessary spaces.

Example

@REPLACE(SOLD,@FIND("-",SOLD,0),1,"/") displays the label in SOLD, 4-24, as 4/24.

@RIGHT

@RIGHT(*string*,*n*) returns the last *n* characters in *string*.

Arguments

string is text, a text formula, or the address or name of a cell that contains a label or text formula.

n is a positive integer or 0, the address or name of a cell that contains a positive integer or 0, or a formula that returns a positive integer or 0. If n is 0, the result is an empty string. If n is greater than the length of *string*, @RIGHT returns the entire string.

Uses

@RIGHT is useful for copying only part of a label into another cell (for example, for extracting last names from labels that include both first and last names).

In a macro, @RIGHT can extract parts of labels the user enters to store them in a database, to use in a subroutine, or to alter the macro itself.

Use @RIGHT with @FIND and @LENGTH when you do not know the exact value for n , or when n may vary.

Notes

@RIGHT counts punctuation and spaces as characters.

Examples

@RIGHT("Average Daily Balance",7) = Balance.

@RIGHT(B3,5) = Sales, if B3 contains the label January Sales.

@RIGHT(A1,@LENGTH(A1)-@FIND("•",A1,0)-1) = the street name in cell A1 (for example, Main Street if cell A1 contains the label 123 Main Street). The • (bullet) represents one space.

@ROUND

@ROUND(x,n) rounds the value x to the nearest multiple of the power of 10 specified by n .

Arguments

x is a value, the address or name of a cell that contains a value, or a formula that returns a value.

n is a value, the address or name of a cell that contains a value, or a formula that returns a value from -15 through 15.

Uses

Use @ROUND in calculations in which you want to use only the rounded portion of a value.

@ROUND is useful in any application that needs a particular number of decimal places (or tens, hundreds, and so on).

Notes

If n is positive, @ROUND affects the decimal portion of the number (moving right from the decimal point). For example, if n is 2, 1-2-3 rounds x to the nearest hundredth.

If n is negative, @ROUND affects the integer portion of the number (moving left from the decimal point) and discards any decimal portion. For example, if n is -2, 1-2-3 rounds x to the nearest hundred.

If n is 0, 1-2-3 rounds x to the nearest integer.

Use /Range Format Fixed or /Worksheet Global Format Fixed to display values with a specified number of decimal places if you want 1-2-3 to calculate the values to their full precision; do not use @ROUND.

@ROUND differs from @INT in that @INT returns the integer portion of a value, without rounding the value.

Examples

@ROUND(134.578,2) = 134.58.

@ROUND(134.578,0) = 135.

@ROUND(134.578,-2) = 100.

@ROWS

@ROWS(*range*) counts the number of rows in *range*.

Argument

range is a cell address or range name.

Uses

@ROWS is useful when you need to determine a value that depends on the number of rows (for example, to find the length of a range you want to print).

Use @ROWS with {FOR} in a macro that repeats the same action on a series of rows in a range when the number of rows is unknown.

Examples

@ROWS(A3..B7) = 5 (rows 3 through 7).

@ROWS(SCORES) = 43, if SCORES is the range B3..B45.

@ROWS(A6..A6) = 1.

@S

`@S(range)` produces the entry in the upper left cell in *range* as a label. If the cell contains a label, `@S` returns that label; if the cell contains a value or is blank, `@S` returns an empty string.

Argument

range is a cell address or range name.

Uses

`@S` is useful with any string `@function` or text formula when a cell may contain a value and the entry must be a label (for example, a cell that contains a ZIP code). Use `@S` to prevent text formulas from resulting in ERR (for example, `+A1&A2` returns ERR if either cell contains a value).

`@S` is also useful in macros to check user entries.

Example

In the following macro instructions, `@S` returns an empty string if B6 contains a value or is a blank cell; 1-2-3 then beeps and changes the mode indicator to ENTRY MUST BE A LABEL.

```
{IF @S(B6)=""}{BEEP}{INDICATE "ENTRY MUST BE A LABEL"}
```

@SECOND

`@SECOND(time-number)` extracts the seconds, an integer from 0 through 59, from *time-number*.

Argument

time-number is a value, the address or name of a cell that contains a value, or a formula that returns a value whose decimal portion is from .000000 (midnight) through .999988 (11:59:59 P.M.). `@SECOND` uses only the decimal portion of *time-number*.

Uses

Use `@SECOND` to use only the seconds portion of time values created with `@NOW`, `@TIME`, or `@TIMEVALUE`.

Examples

`@SECOND(0.333) = 31.`

`@SECOND(@TIME(11,15,45)) = 45`, because 45 is the *seconds* argument for `@TIME(11,15,45)`.

@SIN

@SIN(*z*) calculates the sine of an angle. **Sine** is the ratio of the side opposite an acute angle of a right triangle to the hypotenuse: $\sin(\text{angle}) = \text{side opposite the angle} / \text{hypotenuse}$.

Argument

z is a value, the address or name of a cell that contains a value, or a formula that returns a value of an angle measured in radians.

Uses

Use @SIN to calculate the ratio of the side opposite angle *z* to the hypotenuse when you know angle *z*.

Use @SIN to find the cosecant, or reciprocal of @SIN, with the following formula: $1 / \text{@SIN}(z)$.

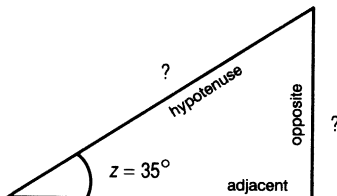
Use @SIN to find either the opposite side or the hypotenuse when only one is known.

Notes

To convert from degrees to radians, multiply degrees by @PI/180.

Examples

In the following right triangle, angle *z* is 35°. Using @SIN(35*@PI/180) is the same as dividing the length of the side opposite angle *z* by the length of the hypotenuse.



@SIN(35*@PI/180) = 0.573576; so the sine of *z* is 0.573576. If the hypotenuse is 10, the side opposite the angle (calculated as @SIN(35*@PI/180)*10) is 5.73576.

@SLN

@SLN(*cost, salvage, life*) calculates the straight-line depreciation allowance of an asset with an initial value of *cost*, an expected useful *life*, and a final value of *salvage*, for one period.

Arguments

cost is the amount paid for the asset.

salvage is the value of the asset at the end of its life.

life is the number of periods the asset takes to depreciate to its salvage value.

cost, *salvage*, and *life* are values, the addresses or names of cells that contain values, or formulas that return values.

Uses

Use @SLN to calculate an asset's depreciation when the situation does not require accelerated depreciation.

Notes

Straight-line depreciation divides the depreciable cost (the actual cost minus the salvage value) equally into each period of the useful life of the asset. The useful life is the number of periods (typically years) over which the asset is depreciated.

@SLN uses the following formula to calculate straight-line depreciation:

$$\frac{(c - s)}{n} \quad \text{where: } \begin{array}{l} c = \text{cost of the asset} \\ s = \text{salvage value of the asset} \\ n = \text{useful life of the asset} \end{array}$$

Example

You have an office machine worth \$10,000. The useful life of this machine is 10 years, and the salvage value in 10 years will be \$1200. To calculate the yearly depreciation expense using the straight-line method, you enter @SLN(10000,1200,10). 1-2-3 returns \$880, the yearly depreciation allowance.

Similar @functions

@DDB calculates depreciation using the double-declining balance method, and @SYD uses the sum-of-the-years'-digits method.

@SQRT

@SQRT(*x*) returns the positive square root of *x*.

Argument

x is a positive value, the address or name of a cell that contains a positive value, or a formula that returns a positive value.

Uses

Use @SQRT to calculate with or find the square root of a number. Use @SQRT with @ABS to force *x* to be positive: @SQRT(@ABS(*x*)).

Examples

@SQRT(B3) = 10, if B3 contains the value 100.

@SQRT(-2) = ERR, because *x* is negative.

@STD

@STD(*list*) calculates the standard deviation in a list of values.

Argument

list is a series of values, or the addresses or names of cells that contain values, separated by argument separators.

Uses

Standard deviation measures the degree to which individual values in a list vary from the mean (average) of all values in the list. The lower the standard deviation, the less individual values vary from the mean, and the more reliable the mean. A standard deviation of 0 indicates that all values in the list are equal.

The standard deviation is the square root of the variance (@VAR).

Use @STD to find the standard deviation within a population of values (for example, the degree to which individual scores or weights vary from the mean). @STD is useful for statistical calculations that involve finite groups, such as test scores, salary distributions, or ages of people in a company.

Notes

@STD uses the n, or population, method to calculate standard deviation of population data. The n method assumes that the values in *list* are the entire population. If *list* is only a sample of the population, the standard deviation is biased because of errors introduced in taking the sample. The n method uses the following formula:

$$\sqrt{\frac{\sum (v_i - avg)^2}{n}}$$

where: n = number of items in *list*
 v_i = the *i*th item in *list*
 avg = average of values in *list*

You use the n-1 method (sample standard deviation) to produce an unbiased standard deviation for a sample. The n-1 method uses the following formula, where *list* is the series of values for which you are calculating standard deviation.

@STD(*list*)*SQRT(@COUNT(*list*)/(@COUNT(*list*)-1))

Example

In the following illustration, cell D37 shows the result of using @STD to calculate the standard deviation of the ages of the houses sold in April and May (the range AGE_LIST in F26..F35).

| A | B | C | D | E | F | G | |
|----|--------|----------------------------|----|-----------|-----------|-----|----------|
| 25 | DATE | ADDRESS | BR | OFFERED | SOLD | AGE | BROKER |
| 26 | 03-Apr | 467 Brattle | 4 | \$775,000 | \$720,000 | 48 | JCompton |
| 27 | 05-Apr | 183 Hillside | 3 | 325,000 | 318,000 | 22 | CGroden |
| 28 | 10-Apr | 64 N. Gate | 2 | 340,000 | 332,000 | 21 | AMiller |
| 29 | 14-Apr | 80 Mt. Auburn | 2 | 280,000 | 278,000 | 70 | JCompton |
| 30 | 25-Apr | 14 Charles | 4 | 179,000 | 160,000 | 52 | AMiller |
| 31 | 27-Apr | 1160 Memorial | 1 | 230,000 | 227,000 | 42 | AMiller |
| 32 | 04-May | 130 Crescent | 3 | 405,000 | 397,000 | 23 | CGroden |
| 33 | 10-May | 12 Trenton | 2 | 310,000 | 303,000 | 21 | AMiller |
| 34 | 11-May | 36 Barnes | 4 | 680,000 | 669,000 | 35 | JCompton |
| 35 | 22-May | 234 Third | 2 | 155,000 | 140,000 | 40 | JCompton |
| 36 | | | | | | | |
| 37 | | Standard deviation of AGE: | | 16.94 | | | |
| 38 | | | | | | | |
| 39 | | | | | | | |
| 40 | | | | | | | |
| 41 | | | | | | | |

AGE_LIST

Similar @functions

@VAR calculates the population variance of *list*. @DSTD calculates the standard deviation of values that meet criteria you specify.

@STRING

@STRING(*x*,*n*) converts the value *x* to a label with *n* decimal places.

Arguments

x is a value, the address or name of a cell that contains a value, or a formula that returns a value.

n is an integer, the address or name of a cell that contains an integer, or a formula that returns an integer from 0 through 15.

Uses

@STRING is useful when you need to use a value as text, such as in a text formula to form a complex label. For example, if cell B3 contains the value 78 and cell B4 contains the label Lincoln Avenue, the formula @STRING(B3,0)&"•"&B4 produces the string 78 Lincoln Avenue. Each • (bullet) represents one space.

Notes

@STRING ignores any formatting characters 1-2-3 uses to display the value *x*. This includes all currency and other numeric formatting symbols, whether you enter them or 1-2-3 creates them after you select a format from /Range Format or /Worksheet Global Format. For example, if cell A7 contains the formatted value \$45.23, @STRING(A7,2) = the label 45.23.

Examples

@STRING(B4,3) = the string 203.000, if cell B4 contains the value 203.

@STRING(1.23587,0) = the string 1.

@STRING(20%,1) = the string .2.

@SUM

@SUM(*list*) adds the values in *list*.

Argument

list is a series of values, or the addresses or names of cells that contain values, separated by argument separators.

Uses

Use @SUM to find the total value of a series of values, such as total sales or gross budget projections. @SUM eases the time-consuming task of adding the values in individual ranges: @SUM(B1..B5) is equivalent to the formula +B1+B2+B3+B4+B5.

You can also use @SUM to add a variety of cells and ranges, for example @SUM(A1..B3,C10,200,RANGE1).

Notes

@SUM ignores blank cells in the range, and treats cells that contain labels as 0.

Example

The following illustration shows house sales for April and May. Cell D37 shows the result of using @SUM to find the total commission paid on sales in April, cell D38 shows the total commission paid on sales in May, and cell D39 shows the total commission paid for both months.

D39: (C0) [C11] @SUM(G25..G35) READY

| | A | B | C | E | F | G | |
|----|--------|----------------------------|----|-----------|-----------|----------|------------|
| 25 | DATE | ADDRESS | BR | OFFERED | SOLD | BROKER | COMMISSION |
| 26 | 03-Apr | 467 Brattle | 4 | \$775,000 | \$720,000 | JCompton | \$28,800 |
| 27 | 05-Apr | 183 Hillside | 3 | 325,000 | 318,000 | CGroden | 12,720 |
| 28 | 10-Apr | 64 N. Gate | 2 | 340,000 | 332,000 | AMiller | 13,280 |
| 29 | 14-Apr | 80 Mt. Auburn | 2 | 280,000 | 278,000 | JCompton | 11,120 |
| 30 | 25-Apr | 14 Charles | 4 | 179,000 | 160,000 | AMiller | 6,400 |
| 31 | 27-Apr | 1160 Memorial | 1 | 230,000 | 227,000 | AMiller | 9,080 |
| 32 | 06-May | 130 Crescent | 3 | 405,000 | 397,000 | CGroden | 15,880 |
| 33 | 10-May | 12 Trenton | 2 | 310,000 | 303,000 | AMiller | 12,120 |
| 34 | 11-May | 36 Barnes | 4 | 680,000 | 669,000 | JCompton | 26,760 |
| 35 | 22-May | 234 Third | 2 | 155,000 | 140,000 | JCompton | 5,600 |
| 36 | | | | | | | |
| 37 | | Total commissions, APRIL: | | \$81,400 | | | |
| 38 | | Total commissions, MAY: | | \$60,360 | | | |
| 39 | | Total comm, APRIL and MAY: | | \$141,760 | | | |
| 40 | | | | | | | |
| 41 | | | | | | | |

Similar @functions

@DSUM calculates the sum of values that meet criteria you specify.

@SYD

@SYD(*cost,salvage,life,period*) calculates the sum-of-the-years'-digits depreciation allowance of an asset with an initial value of *cost*, an expected useful *life*, and a final value of *salvage*, for a specified *period*.

Arguments

cost is the amount paid for the asset.

salvage is the value of the asset at the end of its life.

life is the number of periods (typically, years) the asset takes to depreciate to its salvage value.

period is the time for which you want to find the depreciation allowance.

cost, *salvage*, *life*, and *period* are values, the addresses or names of cells that contain values, or formulas that return values.

Uses

The sum-of-the-years'-digits method accelerates the rate of depreciation so that more depreciation expense occurs in earlier periods than in later ones (although not so much as when you use the double-declining balance method). The depreciable cost is the actual cost minus the salvage value.

Use @SYD when you need a higher depreciation expense early in the life of an asset, such as in preparing tax returns.

Notes

@SYD uses the following formula to calculate depreciation using the sum-of-the-years'-digits method:

$$\frac{(c - s) * (n - p + 1)}{(n * (n + 1) / 2)}$$

where: *c* = cost of the asset
s = salvage value of the asset
p = period for which depreciation is being calculated
n = calculated useful life of the asset

Example

You have an office machine worth \$10,000. The useful life of the machine is 10 years, and the salvage value in 10 years will be \$1200. To calculate the depreciation expense for the fifth year using the sum-of-the-years'-digits method, you enter @SYD(10000,1200,10,5). 1-2-3 returns \$960, the depreciation allowance for the fifth year.

Similar @functions

@DDB calculates depreciation using the double-declining balance method, and @SLN calculates depreciation using the straight-line method.

@TAN

@TAN(*z*) calculates the tangent of angle *z*. The tangent is the ratio of the side opposite an acute angle of a right triangle to the side adjacent to the same acute angle: $\tan(\text{angle}) = \text{side opposite} / \text{side adjacent}$.

Argument

z is a value, the address or name of a cell that contains a value, or a formula that returns a value of an angle measured in radians.

Uses

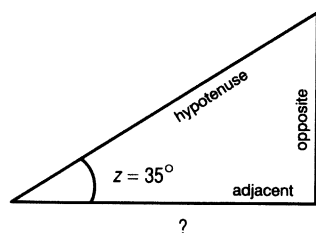
Use @TAN, instead of a tangent table, to calculate the tangent of an angle. Use @TAN to find the cotangent, or reciprocal of @TAN, with the following formula: $1 / @TAN(z)$.

Notes

To convert from degrees to radians, multiply degrees by @PI/180.

Example

In the right triangle below, angle *z* is 35°. Using @TAN(35*@PI/180) is the same as dividing the length of the side opposite angle *z* by the length of the adjacent side.



? $@TAN(35 * @PI / 180) = 0.700208$, rounded; so the tangent of *z* is 0.700208.

@TERM

@TERM(*payments*, *interest*, *future-value*) calculates the number of compounding period (*payments*) required for an investment to accumulate to a *future-value* at a periodic *interest* rate.

Arguments

payments is the value of the equal investments.

interest is the periodic interest rate.

future-value is the amount you want to accumulate.

payments and *future-value* are values, the addresses or names of cells that contain values, or formulas that return values. *interest* is a decimal or percentage value, the address or name of a cell that contains a decimal or percentage value, or a formula that returns a decimal or percentage value.

Uses

Use @TERM when you need to calculate how long it will take for equal periodic deposits, payments, or additions to reach a specific amount at a given interest rate.

Notes

@TERM assumes you are using an ordinary annuity: *payments* are made at the end of each period.

@TERM uses the following formula to calculate the payment term:

$$\frac{\ln(1 + (fv * int/pmt))}{\ln(1 + int)}$$

where: *pmt* = periodic payment
fv = future value
int = periodic interest rate
ln = natural logarithm

If you made payments at the beginning of each year, you would calculate the amount for an annuity due. To calculate the number of payment periods in an annuity due, use the formula @TERM(*payment,interest,future value/(1+interest)*).

For example, @TERM(2000,0.075,100000/(1+0.075)) = 20.8, the number of years it would take to accumulate \$100,000 if you made deposits of \$2,000 at the beginning of each year.

You can calculate the term necessary to pay back a loan by using @TERM with a negative *future value*. For example, you want to know how long it will take to pay back a \$10,000 loan at 10% yearly interest, making payments of \$1,174.60 per year. @ABS(@TERM(1174.6,0.1,-10000)) = 20 years to pay back the loan.

Example

You deposit \$2,000 at the end of each year into a bank account. Your account earns 7.5% a year, compounded annually. To determine how long it will take to accumulate \$100,000, you enter @TERM(2000,.075,100000). 1-2-3 returns 21.5, the number of years it will take to accumulate \$100,000 in your account.

Similar @functions

@CTERM calculates the number of compounding periods for a single-deposit investment.

@TIME

@TIME(*hour,minutes,seconds*) calculates the time number for the specified *hour*, *minutes*, and *seconds*. For an explanation of time numbers, see “Date and Time @Functions” on page 7.

Arguments

hour is a value, the address or name of a cell that contains a value, or a formula that returns a value from 0 (midnight) through 23 (11:00 P.M.).

minutes is a value, the address or name of a cell that contains a value, or a formula that returns a value from 0 through 59.

seconds is a value, the address or name of a cell that contains a value, or a formula that returns a value from 0 through 59.

If *hour*, *minutes*, or *seconds* is not a value, @TIME returns ERR.

Uses

Use @TIME to enter time numbers so that 1-2-3 can perform time arithmetic. Time arithmetic is useful when you need to determine the differences between times (for example, when you need to keep track of elapsed times for billing or in test runs). @TIME is also useful in calendar worksheets.

Notes

/Range Format and /Worksheet Global Format can make the time number appear as the time it represents.

Example

The formula `(@TIME(13,0,0)-@TIME(9,15,0))*95*24` calculates the amount due to a consultant on a given day by subtracting the stop time from the start time and multiplying the result by an hourly rate of \$95.00. The result is \$356.25.

@TIMEVALUE

@TIMEVALUE(*string*) calculates the time number specified in *string*. For an explanation of time numbers, see “Date and Time @Functions” on page 7.

Argument

string is text, a text formula, or the address or name of a cell that contains a label or text formula, in one of the four 1-2-3 time formats: HH:MM:SS AM/PM, HH:MM AM/PM, HH:MM:SS (24 hour), or HH:MM (24 hour).

Uses

The uses of @TIMEVALUE are the same as those for @TIME, but @TIMEVALUE makes entering arguments simpler.

@TIMEVALUE is useful when you need to convert times entered as labels into time numbers for use in calculations. @TIMEVALUE is especially useful with data that has been imported from another program, such as a word processing program.

Notes

/Range Format and /Worksheet Global Format make the time number appear as the time it represents.

Examples

@TIMEVALUE("08:19:27 AM") = 0.34684 or 08:19:27 AM, if the cell is formatted as HH:MM:SS AM/PM.

@TIMEVALUE("15:19:27 AM") = .63851 or 03:19:27 PM, if the cell is formatted as HH:MM:SS AM/PM.

@TIMEVALUE("08:19:27 PM") = .84684.

@TRIM

@TRIM(*string*) removes leading, trailing, and consecutive space characters from *string*.

Argument

string is text, a text formula, or the address or name of a cell that contains a label or text formula.

Uses

Use @TRIM when entries contain unnecessary spaces, such as labels positioned by adding spaces in cells, or labels imported from another program.

@TRIM is also useful to control spaces entered with data or to combine strings that have unknown spacing. @TRIM ensures that database entries do not contain unnecessary spaces that would affect sort order.

Examples

In the following examples, each • (bullet) represents one space.

@TRIM("•45••3/8") = 45 3/8.

@TRIM("•500•••South••St.") = 500 South St.

@TRUE

@TRUE returns the logical value 1.

Uses

Use @TRUE with @functions such as @IF and @CHOOSE to display 1 (true) if a condition is met.

@TRUE is useful as the x argument of @IF, which is the value returned if the condition is met.

Notes

If a logical statement such as $A1 = B1$ is true, its logical value is 1. If it is false, its logical value is 0.

Using @TRUE is the same as using the value 1 in formulas that evaluate logical conditions, but @TRUE makes the formula easier to read.

Use @TRUE with macros or @functions such as @IF and @CHOOSE that require a logical value of 1 (true). You can use either @TRUE or any nonzero value in formulas that evaluate logical conditions, but @TRUE makes the formula easier to read.

Example

@IF(A6>500,@TRUE,@FALSE) = 1, when cell A6 contains a value greater than 500.

@UPPER

@UPPER(*string*) converts all the letters in *string* to uppercase.

Argument

string is text, a text formula, or the address or name of a cell that contains a label or text formula.

Uses

@UPPER is useful when you combine labels from several sources and want capitalization to be consistent throughout the worksheet. Use @UPPER when you want to ensure that labels are consistently uppercase.

Notes

If you selected the ASCII collating sequence when you installed 1-2-3, capitalization affects the order of labels when you use /Data Sort. Two otherwise identical labels may not appear together if their capitalization is different.

Examples

@UPPER("Account Number") = ACCOUNT NUMBER.

@UPPER(B2) = WARNING, if B2 contains the label warning.

@VALUE

@VALUE(*string*) converts a number entered as a *string* to its corresponding value.

Argument

string is text or a label that contains only numbers. *string* can resemble a standard number (456.7), a number in scientific format (4.567E2), a mixed number (45 7/8), or a formatted number (\$32.85).

Uses

Use @VALUE to convert text that consists of numbers into values you can use in mathematical calculations. @VALUE is useful when you need to convert entries retrieved from another source into values.

Notes

@VALUE ignores leading and trailing spaces; however, @VALUE returns ERR when *string* contains spaces that separate symbols from the numbers (such as \$ 32.85 or £ 56.20).

@VALUE results in 0 when *string* is a blank cell or empty string, and returns ERR when *string* contains non-numeric characters.

Use /Range Value to replace @VALUE with its value.

You cannot calculate within a *string* argument in @VALUE, but you can create a formula with several @VALUE functions. For example, @VALUE("22"+"20") = 0, but @VALUE("22")+@VALUE("20") = 42.

The format you use to specify *string* depends on the currency and punctuation settings specified with /Worksheet Global Default Other International.

Examples

@VALUE("543") = the value 543.

@VALUE(B3) = 49.75, if cell B3 contains the label 49 3/4.

@VALUE("85%") = .85.

@VAR

@VAR(*list*) calculates the population variance in a list of values.

Argument

list is a series of values, or the addresses or names of cells that contain values, separated by argument separators.

Uses

Variance measures the degree to which individual values in a list vary from the mean (average) of all the values in the list. The lower the variance, the less individual values vary from the mean, and the more reliable the mean. A variance of 0 indicates that all values in the list are equal.

Use @VAR to calculate the variance from the mean of a list of values when *list* contains all values for the group, as in an employee database or a set of test scores. Variance is necessary in several ANOVA (analysis of variance) statistical tests.

Notes

@VAR uses the *n*, or population, method to calculate variance. The *n* method assumes the selected values are the entire population. If the values are only a sample of the population, the variance is biased because of errors introduced in taking a sample. The *n* method uses the following formula:

$$\frac{\sum (v_i - avg)^2}{n}$$

where: *n* = number of values in *list*
v_i = the *i*th value in *list*
avg = average of values in *list*

You use the *n*-1 method (sample population variance) to produce an unbiased variance for a sample. The *n*-1 method uses the following formula, where *list* is the series of values for which you are calculating the population variance.

@VAR(*list*)*@COUNT(*list*)/(@COUNT(*list*)-1)

Example

The following illustration lists house sales for April and May. Cell D37 shows the result of using @VAR to calculate the variance in the age of the houses in the list.

| | A | B | C | D | E | F | G | |
|----|--------|------------------|----|-----------|-----------|-----|----------|---|
| 25 | DATE | ADDRESS | BR | OFFERED | SOLD | AGE | BROKER | ◀ |
| 26 | 03-Apr | 467 Brattle | 4 | \$775,000 | \$720,000 | 48 | JCompton | ▶ |
| 27 | 05-Apr | 183 Hillside | 3 | 325,000 | 318,000 | 22 | CGroden | ▶ |
| 28 | 10-Apr | 64 N. Gate | 2 | 340,000 | 332,000 | 21 | AMiller | ▶ |
| 29 | 14-Apr | 80 Mt. Auburn | 2 | 280,000 | 278,000 | 70 | JCompton | ? |
| 30 | 25-Apr | 14 Charles | 4 | 179,000 | 160,000 | 52 | AMiller | |
| 31 | 27-Apr | 1160 Memorial | 1 | 230,000 | 227,000 | 42 | AMiller | |
| 32 | 04-May | 130 Crescent | 3 | 405,000 | 397,000 | 23 | CGroden | |
| 33 | 10-May | 12 Trenton | 2 | 310,000 | 303,000 | 21 | AMiller | |
| 34 | 11-May | 36 Barnes | 4 | 680,000 | 669,000 | 35 | JCompton | |
| 35 | 22-May | 234 Third | 2 | 155,000 | 140,000 | 60 | JCompton | |
| 36 | | | | | | | | |
| 37 | | Variance of AGE: | | 286.84 | | | | |
| 38 | | | | | | | | |
| 39 | | | | | | | | |
| 40 | | | | | | | | |
| 41 | | | | | | | | |

Similar @functions

@DVAR calculates the population variance of values that meet criteria you specify.

@VLOOKUP

@VLOOKUP(*x,range,column-offset*) produces the contents of a cell in a specified column of a vertical lookup table.

Arguments

x is a value or text, the address or name of a cell that contains a value or text, or a formula that returns a value or text. If *x* is a value that is less than the first value in *range*, @VLOOKUP returns ERR. If *x* is greater than the last value in *range*, @VLOOKUP stops at the last cell in *column-offset*. If *x* is text, it must be an exact match of the text in *range*.

range is a cell address or range name. The values in the first column of the table (*range*) must be in ascending order. 1-2-3 compares the value *x* to each cell in the first column. When 1-2-3 locates a cell in the first column that contains *x* (or the value closest to, but not greater than, *x*), it moves across that row the number of columns specified by *column-offset* and returns the contents of that cell as the answer.

column-offset is a value, the address or name of a cell that contains a value, or a formula that returns a value from 0 through 255. The *column-offset* number corresponds to the position the column occupies in *range*. The first column has an offset number of 0, the second column has an offset number of 1, and so on. If *column-offset* is positive, 1-2-3 moves across the row the specified number of columns; if *column-offset* is 0, 1-2-3 stays in the first column.

Uses

Use `@VLOOKUP` when you need to choose items from a table, or to automate data selection for formulas or macros, such as in an application that supplies prices from a price list.

`@VLOOKUP` is useful for finding entries in tables, such as tax tables or sales commission tables, which contain many different choices based on different variables.

Notes

The first column of the lookup table must be in ascending order.

`@VLOOKUP` is case-sensitive and accent-sensitive; for example, if *x* is the text Paris, `@VLOOKUP` will not match it with a cell that contains PARIS, and the letter à will not match the letter a.

Example

In the following illustration, a tax table named TAXTABLE (A3..E11) lists tax amounts based on income and filing status. The simple form in G3..H5 uses `@VLOOKUP` to retrieve a tax amount based on the income entered in INCOME (cell H3) and the filing status entered in STATUS (cell H4). When INCOME contains \$35,329 and STATUS contains 1, `@VLOOKUP(INCOME,TAXTABLE,STATUS)` returns \$9351, the tax amount for the income figure that is closest to, but not greater than, the value in INCOME.

H5: (C2) [W11] @VLOOKUP(INCOME,TAXTABLE,STATUS) READY

| | A | B | C | D | E | F | G |
|----|-----------|-------------|---------|-------------|---------|--------|-------------|
| 1 | | F I L I N G | | S T A T U S | | | |
| 2 | Income >= | 1 | 2 | 3 | 4 | | |
| 3 | \$35,000 | \$9,219 | \$7,265 | \$11,315 | \$8,531 | Income | \$35,329.00 |
| 4 | \$35,050 | \$9,241 | \$7,282 | \$11,340 | \$8,552 | Status | 1 |
| 5 | \$35,100 | \$9,263 | \$7,299 | \$11,363 | \$8,573 | Tax | \$9,351.00 |
| 6 | \$35,150 | \$9,285 | \$7,313 | \$11,388 | \$8,594 | | |
| 7 | \$35,200 | \$9,307 | \$7,330 | \$11,411 | \$8,615 | | |
| 8 | \$35,250 | \$9,329 | \$7,347 | \$11,436 | \$8,636 | | |
| 9 | \$35,300 | \$9,351 | \$7,361 | \$11,459 | \$8,657 | | |
| 10 | \$35,350 | \$9,373 | \$7,377 | \$11,483 | \$8,678 | | |
| 11 | \$35,400 | \$9,395 | \$7,393 | \$11,507 | \$8,699 | | |

Similar @functions

`@HLOOKUP` looks up a value in a horizontal lookup table. `@INDEX` finds a value when you specify offset numbers for both the column and the row. `@CHOOSE` replaces a lookup table that requires only one row.

@YEAR

@YEAR(*date-number*) extracts the year, an integer from 0 (1900) through 199 (2099), from a date number.

Argument

date-number is a value, the address or name of a cell that contains a value, or a formula that returns a value from 1 (January 1, 1900) through 73050 (December 31, 2099).

Uses

Use @YEAR when you need only the year, rather than the entire date (for example, to determine whether an item was purchased in or before a particular year for warranty restrictions or bond maturity, or to calculate seniority or length of tenure for benefits).

@YEAR can also supply the *year* argument for other date @functions that build on previously calculated dates.

Notes

Add 1900 to @YEAR to convert it into a 4-digit year. For example, @YEAR(20181)+1900 creates the 4-digit year 1955.

Examples

@YEAR(20181) = 55, because the date number 20181 is the date 02-Apr-55.

@YEAR(@NOW) = the current year.

@YEAR(@DATEVALUE("14-Feb-91")) = 91.

Chapter 3

Macro Basics

This chapter provides basic information on macros: definitions, rules for creating and using them, and descriptions of the types of macros you can create. This chapter also lists the different categories of macro commands, including the macro key equivalents. For a detailed description of each macro command, see Chapter 4 beginning on page 111.

What Is a Macro?

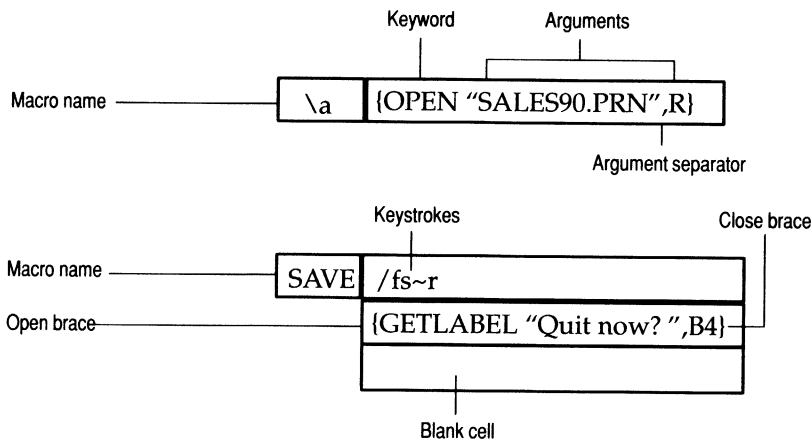
You can use macros to speed up repetitive or complex tasks. A **macro** is a set of instructions that automates tasks in 1-2-3. Macros can include procedures entered from the keyboard and/or macro commands.

- Macros can automate procedures you normally perform from the keyboard, such as using menu commands.
- Macros can perform complex tasks and programming procedures, such as for-loops and if-then-else statements.
- Macros can guide users who are unfamiliar with 1-2-3 through tasks and applications that you create and control.

You can use a macro from the worksheet or a macro library.

Macro Definitions

The following examples show the different elements you use when you enter macros.



Arguments are data you provide for many macro commands. 1-2-3 macro commands accept four types of arguments: number, string, location, and condition. 1-2-3 uses arguments when it runs the macro.

Argument separators separate two or more arguments. 1-2-3 allows three argument separators: , (comma); ; (semicolon), and . (period). You can always use a ; (semicolon) to separate arguments and, depending on the setting of /Worksheet Global Default Other International Punctuation, you can also use either . (period) or , (comma).

Blank cell is a cell that does not contain an entry or a label-prefix character. A blank cell (or a {RETURN} command, a {QUIT} command, or a cell that contains a value) indicates the end of a macro or a subroutine.

Braces enclose all macro commands (including arguments) and key names that replace 1-2-3 keys such as F2 (EDIT) or F9 (CALC).

Keystrokes are ordinary 1-2-3 keystroke or menu commands used as part of a macro. For example /WCS15~ are the keystrokes used to change the current column's width to 15. (The ~ (tilde) is the keystroke equivalent of ENTER.)

Macro keyword is the key name or the name of a 1-2-3 macro command. For example, {DOWN} and {GETLABEL} are macro keywords.

Macro name is the range name you assign to a macro to help you remember what the macro does and to make the macro easier to run. The macro name is the command you use to start the macro.

NOTE Mouse actions are not acceptable as macro instructions. You must use keystrokes to select commands and data in a macro.

Macro Format and Rules

A macro must include instructions for every step of the task or procedure it automates. Before you write a macro, go through the task or procedure manually, noting the commands and keystrokes you use in each step.

If you use keystrokes for every action, 1-2-3 can record the keystrokes if you specify a learn range (/Worksheet Learn) and press ALT-F5 (LEARN). You can view the column that contains the learn range, copy the keystrokes you need, and use them in the worksheet as a macro. (See "Creating a Macro with the Learn Feature" on page 104.) The simplest sort of macro uses only keystrokes — for example, to save the worksheet — and using ALT-F5 (LEARN) is the easiest way to create this sort of macro. You can also type the macro in a single column in the worksheet.

Other macros automate tasks with **macro commands** — commands you enter in the macro that tell 1-2-3 what to do. Macro commands are an easy-to-use programming language within 1-2-3. When you combine macro commands and keystrokes in a macro, you can create applications that simplify complex tasks or automate time-consuming and repetitive chores. For example, use {FORM} to create a data

entry form that prompts for specific information, checks the responses, and enters the data in the worksheet; or use {FOR} to repeat a task a specified number of times.

You can create macros to use with Wysiwyg, Auditor, Viewer, and Macro Library Manager. However, you cannot create macros to use with the mouse, with dialog boxes, or with the Wysiwyg :Graph Edit commands.

NOTE The second confirmation prompt for /Worksheet Erase and /File Retrieve does not appear while a macro is running.

Follow the guidelines below to enter keystrokes and macro commands in a macro.

Keystrokes

Most macros contain keystrokes that automate procedures, such as entering data or choosing menu commands. These procedures consist of keyboard characters (letters, numbers, symbols) and commands. To enter keystrokes in a macro, type (as labels) the keystrokes you want 1-2-3 to perform. Type the keystrokes in a single cell or in adjacent cells in a single column. For example, the following macro inserts a row and types the label Denise's Dairy Parlour in the current cell. (The ~ (tilde) represents ENTER.)

```
'/WIR~
```

```
'Denise's Dairy Parlour~
```

A label-prefix character (' ^ or ") is required if the macro instruction begins with slash (/), backslash (\), a number, or one of the numeric symbols < + <> <= - @ . (# or \$.

Macro Commands

Macros may also contain key names that represent keyboard keys (such as TAB, DOWN, and F5 (GOTO)), and commands similar to those found in programming languages. When you enter these macro commands and key names, use the correct syntax; 1-2-3 cannot perform the macro instructions if the syntax is incorrect.

The format for macro commands and key names is

```
{KEYWORD}
```

or

```
{KEYWORD argument1 argument2,...,argumentn}
```

KEYWORD is the name of a macro command or a key name and is always preceded by { (open brace). Key names and commands that have no arguments must be followed by } (close brace). The keyword tells 1-2-3 what action to perform. You can type keywords in uppercase or lowercase letters, but this book refers to macro keywords in uppercase letters. For a list of macro keywords, see "Macro Command Categories" on page 106.

argument1, argument2,..., argumentn are arguments for the macro command, where *argumentn* is the last of several arguments in a list. Arguments provide information 1-2-3 needs to complete the command and perform its task. You can type arguments

in uppercase or lowercase letters; this book refers to arguments in italics. Some commands have optional arguments (arguments you can omit); this book shows optional arguments in [] (brackets). The last argument must be followed by } (close brace).

Macro Command Rules

To include a macro command in a macro, follow these guidelines:

- Start and end the macro command in the same cell. (The macro itself can span many cells in the same column.)
- Start the command with { (open brace) and end it with } (close brace).
- Type the keyword immediately after the open brace. You can type it in uppercase or lowercase letters.
- Separate the keyword from the first argument (if any) with one space, but do not type spaces between arguments.
- If the command includes two or more arguments, separate the arguments from one another with argument separators. By default, semicolons and commas are valid argument separators for macro commands. The examples in this chapter always use commas. You can, however, use /Worksheet Global Default Other International Punctuation to set a different argument separator.
- Enter any combination of macro commands and keystroke instructions in the same cell, as long as the total number of characters does not exceed 240.
- With the exception of specifying a range address, do not use a comma, semicolon, or period as part of an argument, unless you enclose the argument in double quotation marks. Also, do not use a colon or brace as part of an argument, unless you enclose the argument in double quotation marks.

Argument Types

1-2-3 macro commands accept four types of arguments: condition, location, string, and value.

| Type | Description |
|-----------|--|
| Condition | An expression that evaluates to true or false, or the cell address or range name of a cell that contains such an expression. The macro evaluates the condition argument and proceeds according to whether it is true or false. You can also use a formula or @function, a number, or a range name or cell address as a condition argument. |
| Location | The address or name of a cell or range, or a formula or @function that returns a range address or name. A location argument can refer to a single-cell or multiple-cell range. |

(continued)

| Type | Description |
|--------|---|
| String | Text (any sequence of letters, numbers, and symbols) enclosed in double quotation marks, the range address or name of a cell that contains a label, or a formula or @function that returns a label. |
| Value | A number, the address or name of a cell that contains a number, or a formula or @function that returns a number. |

The following rules apply to argument types:

- Use range names to ensure that location arguments are correct even if you insert or delete rows or columns.
- Location arguments in flow-of-control macro commands (listed on page 107) can be either in the worksheet or in a macro library.
- To make sure that 1-2-3 uses an argument as a string, not a value, add :s or :string to the end of the argument. To make sure that 1-2-3 uses an argument as a value, add :v or :value to the end of the argument.
- Macro commands that require a single cell use the upper left corner cell of a multiple-cell range.

Macro Location

Use the following information to help you select a worksheet location for the macros you create:

- You can save macros in a worksheet with other data or you can save them in a special file that contains only macros (a **macro library**). If you plan to use a macro with only one worksheet, the simplest approach is to enter the macro in that worksheet. If you plan to use the macro with a number of worksheets, enter the macro in a blank worksheet (or any worksheet) and then use Macro Library Manager to save the macro in a macro library. For more information on creating and using macro libraries, see Chapter 6 beginning on page 177.
- If the macro calls subroutines or branches, put the subroutines or branches near the calling macro in the worksheet so that you can see both at once, if possible. If several macros in different worksheet files use the same subroutine, put the subroutine in a macro library file.
- If you enter macros in the same worksheet that contains data, enter the macros below and to the right of the data. This keeps you from writing over data when you enter the macros or damaging the macros when you insert or delete rows and columns in the data area. For example, if the data occupies the range A1..Z240, put the macro below row 240 and to the right of column Z.
- Enter all instructions for a single macro as labels in successive cells in the same column (unless the macro uses branches or subroutines).

Creating a Macro

To create a macro, you must first identify the steps of the 1-2-3 task you are automating.

For example, suppose you want to create a macro that sets the width of a column to 15. To create the macro, you must know that the task involves selecting /Worksheet Column Set-Width, typing 15 as the column width, and pressing ENTER to complete the command. The macro /WCS15~ performs this task.

Identifying the steps means performing the task once, manually, and noting each key that you press. In some cases, mapping out the procedure with a flow chart may help you work out the steps of the task.

To Create a Macro

1. Perform the procedure the macro automates to determine what keystrokes are necessary.

If you perform the procedure, you can use recorded keystrokes from the learn range as macro instructions. See “Creating a Macro with the Learn Feature” beginning on page 104.

2. Move to a worksheet location far away from data (see “Macro Location” on page 95).
3. Enter the macro name in an empty cell.

The macro name provides helpful information about what the macro does. The name is not necessary to run the macro.

4. Enter the first macro instruction as a label in the cell to the right of the name. (If the macro is short, enter the entire macro in this cell.) Use keystrokes, or macro commands, or both.

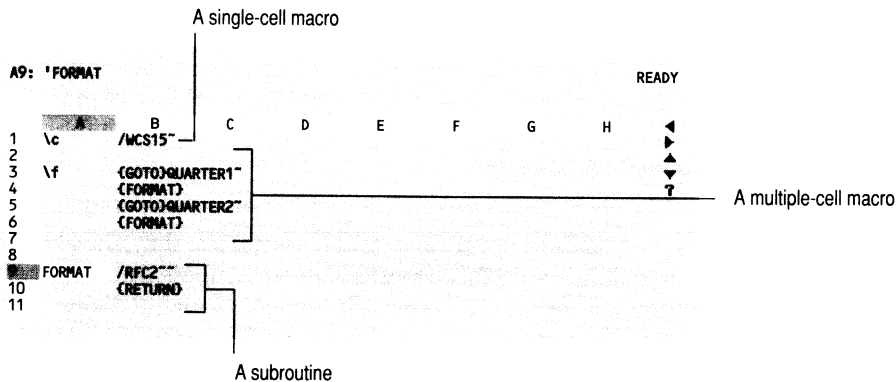
Keystrokes — type a label-prefix character if the first keystroke is a menu command, number, numeric character, \ / < + <> <= \$ # @ - . (\$; then type the keystroke(s).

Macro commands — enter the macro keyword using the syntax described in “Macro Commands,” on page 93. While you are entering a macro, you can also press F1 (HELP) for an explanation of the macro command’s syntax.

5. Enter any subsequent instructions in the cells immediately below the first instruction.
6. Enter subroutines and branch macros as necessary in adjacent columns or below the end of the macro. Provide a blank row between the end of the macro and the first cell of the subroutine. Enter the name of each subroutine or branch macro in the cell to the left of its top cell.

7. Enter the keyword {QUIT} or leave a blank cell after the last line of the macro to end the macro. (If you are creating a macro subroutine, enter the keyword {RETURN} after the last line of the subroutine.)
8. Name the macro and the subroutines and branch macros. (See "Naming a Macro" on page 98.)

The following illustration shows two macros and a subroutine entered in a single column.



Tips for Creating a Macro

- Specify consecutive keystroke instructions by following the key name with a number. For example, {UP 5} moves the cell pointer up five rows. Separate the number from the key name with a space.

Alternatively, you can include a cell reference (address, range name, or a formula that returns a number) after the key name. For example, {DOWN SOME} moves the cell pointer down the number of rows specified by the value in the cell named SOME.

- Use range names when specifying worksheet locations in a macro. If you move a range (for example, if you insert some rows above the range), a macro that refers to the range by name will continue to work correctly, but a macro that refers to the range by address may not (for example, a macro that uses absolute cell addresses in formulas).
- Use { } (open and close braces with nothing inside them) as placeholders in a macro. 1-2-3 ignores { } instructions when running a macro.
- Create several macros and name them all at once. Enter the macros in the same column (with at least one blank cell between them), and enter the name of each macro to the left of the macro's starting cell. Position the cell pointer in the column that contains the names. Then select /Range Name Labels Right and specify this column of names to assign them to all the macros at once.

- Create an auto-execute macro by naming the macro \0. If the Auto-exec Macros on check box is marked in the Worksheet Global Default Settings dialog box, 1-2-3 runs the auto-execute macro when you retrieve the file or load the macro library.
- Create libraries of macros using Macro Library Manager (see Chapter 6 beginning on page 177). Then, when you load a macro library into memory, you can use the macros in the library with any worksheet.

Naming a Macro

After you enter a macro, assign the macro a range name. To name a macro, name the first cell of the macro with /Range Name Create or /Range Name Labels. Name subroutines the same way you name macros. You use the range name to run the macro. (See “Running a Macro” on page 99.)

Macro range names can consist of any combination of up to 15 characters. Like any other range name, however, they should not duplicate cell addresses; they should not include spaces, commas, semicolons, periods, or mathematical symbols; and they should not duplicate @function names, macro command keywords, or 1-2-3 key names. For more information, see “Using Named Ranges” in Chapter 3 of the *User’s Guide*.

You will often want to start macro range names with the same character, such as \ (backslash). Using a naming convention lets you quickly distinguish macro range names from other range names and is helpful in distinguishing macros you plan to store in a macro library. For details about macro libraries, see Chapter 6, beginning on page 177.

To Name a Macro

1. Select /Range Name Create.
2. Specify the macro’s name (up to 15 characters) as the range name. 1-2-3 accepts two kinds of macro names:

Backslash names consist of a backslash followed by a single letter, such as \D. You start this macro by pressing ALT and the letter that follows the backslash. (See “Running a Macro” on page 99.) If you name a macro \0 (zero), you create an auto-execute macro that executes automatically every time you retrieve the worksheet file (for more information, see “Auto-execute Macros” on page 100).

Multiple-character names are ordinary range names. Specify a name that reminds you of what the macro does.

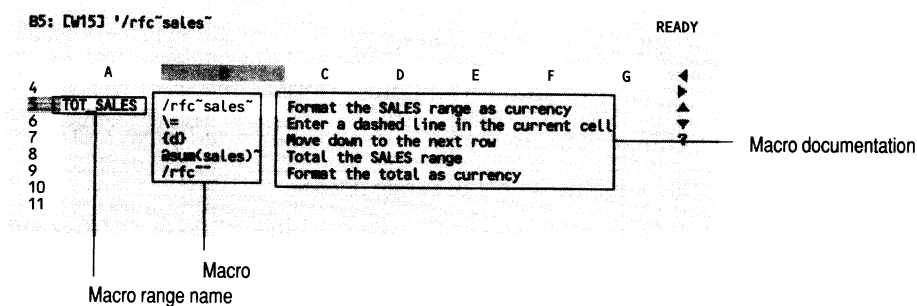
3. Specify the first cell of the macro as the range to name.

If you will later move the macro around the worksheet, specify the entire macro as the range, not just the first cell. You can then use the range name when moving the macro.

Documenting a Macro

After entering and naming a macro, document the macro's range name and the macro instructions. This documentation identifies which range name belongs to the macro (useful in a worksheet that contains many named ranges); clarifies the macro's purpose; and describes the macro's steps.

- Document the macro's range name by entering the name as a label to the left of the first cell of macro instructions. If the name starts with a backslash, such as \N, type a label prefix (" " or ^) before you type the range name.
- Document the macro instructions by entering comments to the right of the cell or cells that contain the macro.



Running a Macro

Running a macro means starting the macro so it carries out the task it automates. If it performs as expected, you can use the macro at any time during the current 1-2-3 work session. To use the macro in future sessions, save the worksheet.

You run a macro in one of two ways: with ALT or with ALT-F3 (RUN). The method you use depends on the macro's name. You can run any backslash macro with ALT when 1-2-3 is in READY mode, EDIT mode, or during a command (MENU or POINT mode). However, ALT-F3 (RUN) only works in READY mode.

1-2-3 runs a macro by performing the commands in the first cell of the macro, then the commands in the next cell in the same column, and so on down the single column. 1-2-3 ends the macro when it reaches a cell that does not contain a label.

CAUTION Use /File Save to save your work before you run the macro. If the macro produces unexpected results, you can then use /File Retrieve to retrieve the original version of the worksheet. If undo is on and no add-in program is attached, you can press ALT-F4 (UNDO) immediately after the macro is finished to restore your original worksheet.

To Run a Backslash Macro

1. Hold down ALT and then press the letter of the macro range name.

For example, to run a macro named \N, press ALT-n. 1-2-3 runs the macro.

To Run a Range Name Macro

1. Make sure that 1-2-3 is in READY mode.
2. Press ALT-F3 (RUN).

1-2-3 displays a menu of all range names in the worksheet (including macro range names and backslash macros) and in any macro libraries in memory. If you have many range names, press F3 (NAME) to see a full-screen menu. Highlight a macro range name in the full-screen menu to see the range address or the macro library name (if the range name is in a macro library in memory).

3. To specify the macro to run, do one of the following:
 - Type the macro range name or address and press ENTER.
 - Highlight the macro range name in the list of range names and press ENTER.
 - Click the macro range name in the list of range names.
 - Press ESC to switch 1-2-3 to POINT mode, move the cell pointer to the first cell of the macro, and press ENTER.

Auto-execute Macros

An **auto-execute macro** is a macro that 1-2-3 runs automatically when you retrieve the worksheet file that contains it. An auto-execute macro is useful in custom applications and for worksheet files that you use often.

An auto-execute macro is like any other macro, except that you give it a special name: \0 (zero). 1-2-3 runs the auto-execute macro whenever you retrieve the file that contains it, if the current session has the Auto-execute Macros on check box marked in the Worksheet Global Default Settings dialog box.

You can retrieve files while running a macro from a macro library. If you mark the Auto-execute Macros on check box in the Worksheet Global Default Settings dialog box and the file contains an auto-execute macro, 1-2-3 will run the auto-execute macro. Otherwise, 1-2-3 continues to execute the original macro in the macro library. You can also load a macro library automatically.

If you run a macro that uses /File Retrieve in the worksheet where the macro is currently running, the macro ends as soon as the new file is retrieved as long as the new file does not contain an auto-execute macro with the Auto-execute Macros on check box marked.

Canceling a Macro

Press **CTRL-BREAK** to cancel a macro while it is running. Unless the macro contains a {BREAKOFF} or an {ONERROR} command, 1-2-3 stops the macro after it completes the current macro instruction. After interrupting a macro, press **ESC** or **ENTER** to clear the error message and return 1-2-3 to **READY** mode. You can then resume working with 1-2-3.

For more information, see {BREAKOFF} and {BREAKON} on page 118.

Recalculation During Macros

When you run a macro with the worksheet recalculation method set to **Automatic**, 1-2-3 does not recalculate all data continuously. **Automatic** recalculation occurs if you enter data in the worksheet in response to a {?} command, if the macro enters data, or if you followed a command such as {LET} or {GET} with a ~ (tilde) to represent **ENTER** (which 1-2-3 interprets as user input).

For example, suppose your macro has several {LET} commands, but no user data entry in response to a {?} command. If other macro commands use a formula cell that references the cell that contains the {LET} command, you need to recalculate the worksheet. You can recalculate the worksheet by following the last {LET} command with a ~ (tilde) or by including a {CALC} command.

Dialog Boxes in Macros

When you run a macro, 1-2-3 does not display dialog boxes. Dialog boxes appear only if you use {WINDOW} in your macro. 1-2-3 suspends the macro when a dialog box appears, if you use {EDIT} to switch to **SETTINGS** mode. When you select **OK** or press **ENTER**, 1-2-3 uses the values you specified in the dialog box and continues to run the macro.

Debugging a Macro

If, when you run a macro, it does not perform as you expected it to, or if 1-2-3 does not finish running it because of an error, you need to **debug** the macro — find out which macro instructions are causing the problem and edit them.

Sometimes you can identify the faulty instructions by looking at the results of the macro. For example, if a macro makes a typographical error when entering a label in a cell, look for the same typographical error in the macro instructions.

Often, however, it is more difficult to resolve the problem. In these instances, use the following troubleshooting checklist to identify possible solutions. If you still cannot figure out the problem, go through the macro in **STEP** mode, as explained in “Debugging a Macro in STEP Mode” on page 102.

Troubleshooting Checklist

If 1-2-3 displays an error message when you run a macro, press **F1 (HELP)** while the error message is on the screen to get an explanation of the message. When you are done using the Help system, press **ESC** or **ENTER** to clear the error message. Then move to the macro and look for the problem. Here are some common mistakes made when entering macro instructions:

- Typing or spelling errors, including incorrect spelling of keywords and range names
- Spaces where they shouldn't be, especially between arguments, or missing spaces between keywords and arguments
- Incorrect or incomplete menu command sequences, for example a missing tilde in /rfp0~ instead of /rfp0~~
- Missing braces, or brackets or parentheses instead of braces, for example, [up] or (up) instead of {up}
- A blank cell or a cell that contains a number before the end of the macro
- Missing or incorrect arguments or argument separators
- Incorrect cell or range references, such as undefined or unacceptable range names
- Range names that duplicate keyword names or function-key names

Macro error messages include the location of the instruction 1-2-3 was performing at the time it encountered the error. Check the cell cited in the error message for typographical errors, missing braces or tildes, or anything else listed in the troubleshooting checklist.

If you find no problems in the referenced cell, check to see if a macro instruction in a cell above it could have caused the problem. For example, if 1-2-3 reports an error in a cell that apparently contains none, but the cell does contain a range name assigned at an earlier point in the macro, check the cell that contains the range name assignment instructions to see whether you spelled the range name the same way in both places.

When you find the error, move the cell pointer to the appropriate cell, press **F2 (EDIT)**, correct the error, and press **ENTER**.

Debugging a Macro in STEP Mode

The instructions that cause an error in a lengthy or complicated macro may not be easy to find. To help you diagnose problems in a macro, 1-2-3 has a feature called **STEP mode**. **STEP mode** lets you run a macro one instruction at a time, until you locate the error.

To Use STEP Mode

1. When 1-2-3 is waiting for you to press a key or when 1-2-3 is in READY mode, press ALT-F2 (STEP) to turn on STEP mode.

The STEP indicator appears in the status line at the bottom of the screen.

2. Start the macro using one of the methods described in “Running a Macro” on page 99.

1-2-3 displays the cell address of the current macro instruction and the contents of the cell in the status line.

3. Press a key (the space bar is recommended, but you can press any key) to run the first macro instruction.
4. Repeat step 3 as many times as necessary to find the part of the macro that contains the error.

Each time you press a key, 1-2-3 performs the next instruction in the macro and replaces the status line with the cell address of the macro instruction and the cell's contents. 1-2-3 also highlights the current macro instruction (the one that will run).

If the instruction is not enclosed in { } (braces), such as an @function or label, each time you press a key, 1-2-3 steps through one character of the @function or label. If the macro pauses for user input, a blinking SST indicator also appears to remind you that you are in STEP mode.

If you're using STEP mode with a macro in a library (see Chapter 6, beginning on page 177), you see the macro library name instead of a range address. If the library is protected with a password, however, you see only the library name, not the contents of the macro.

5. Once you find the error, end the macro by pressing CTRL-BREAK.

When you end the macro to edit it, the STEP indicator reappears to remind you that STEP mode is still on. You do not need to turn off STEP mode to edit the macro.

6. Edit the macro to correct the problem.
7. Run the macro in STEP mode again if you need to locate the source of other problems.
8. Press ALT-F2 (STEP) to turn off STEP mode.
9. Press any key to continue the macro.

NOTE You can turn STEP mode on or off during a macro. To do so, press ALT-F2 (STEP) when 1-2-3 is waiting for input during an interactive command.

Creating a Macro with the Learn Feature

In addition to typing macros directly into the worksheet, you can use the 1-2-3 learn feature to create a macro. When you use this feature, 1-2-3 records your keystrokes in a **learn range**, a single column range that you define.

Use the learn feature to record a macro and test it at the same time. Because you are performing the procedure that the macro will automate, you can see on the screen exactly what will happen when the macro runs. The learn feature also minimizes syntax errors because it records keystrokes in the correct format. You simply press keys and respond to the menus and prompts as they appear.

1-2-3 records all your keystrokes in the learn range; you do not enter anything into the learn range directly. If you type more characters than 1-2-3 can enter in the learn range you specified, 1-2-3 turns off the learn feature and tells you the learn range is full. If the task you were recording used only a few keystrokes, you can make the learn range larger, erase its contents with `/Worksheet Learn Erase`, and start again. If the task was long, just make the learn range larger and start again where you left off. 1-2-3 also turns off the learn feature and cancels the learn range if you delete the column that contains the learn range.

1-2-3 records keystrokes in macro instruction format. For example, when you press `F5 (GOTO)`, type `a5`, and press `ENTER`, 1-2-3 records `{GOTO}a5~`. 1-2-3 abbreviates keystrokes (for example, `{D}` instead of `{DOWN}`), and uses a number for duplicate keystrokes (for example `{D 2}`, instead of `{D}{D}`).

1-2-3 does not record 1-2-3 mouse actions, actions in a dialog box, or the following keys in the learn range: `ALT-F1 (COMPOSE)`, `ALT-F2 (STEP)`, `ALT-F3 (RUN)`, `ALT-F4 (UNDO)`, `ALT-F5 (LEARN)`, `CTRL-F1 (BOOKMARK)`, `CTRL-BREAK`, `SHIFT`, `NUM LOCK`, `CAPS LOCK`, `PRINT SCREEN`, or `SCROLL LOCK`. Learn records `HELP (F1)`, but it will not record any keystrokes you enter while using Help.

The learn range does not use any memory until you start recording keystrokes.

To Record Keystrokes

1. Select `/Worksheet Learn Range`.
2. Specify a single column range in an empty part of the worksheet where the macro cannot interfere with data.

Specify a column with a large number of blank cells for the learn range so 1-2-3 doesn't run out of space when recording your keystrokes.

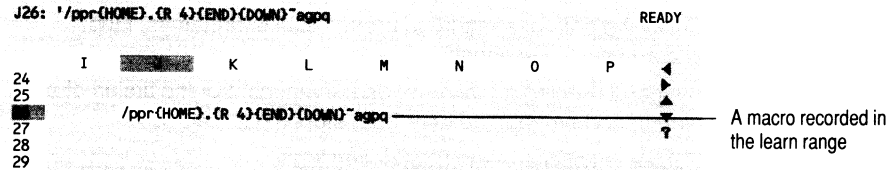
3. Move the cell pointer to the cell where you want to start the task.
4. Press `ALT-F5 (LEARN)`.

The LEARN indicator appears at the bottom of the screen. As long as the LEARN indicator remains on the screen, 1-2-3 records each of your keystrokes in the learn range.

5. Perform the task or series of tasks you want to record.

If you want to include another macro in the one you're recording, enter its range name in { } (braces), for example, {TOTAL}. When you run the macro, 1-2-3 runs the macro you included and then returns control to the original macro. For more information, refer to the description of {*subroutine*} on page 158.

6. When you finish the task, press ALT-F5 (LEARN) again to stop recording keystrokes.



To Edit Keystrokes

When you finish recording keystrokes, move the cell pointer to the learn range and examine the recorded keystrokes. If you made mistakes while performing the task, 1-2-3 recorded them, so edit them before going any further. As you edit, be sure not to leave any empty cells or values in the middle of the macro or 1-2-3 will interpret them as the end of the macro.

If you made many mistakes and want to start over, erase the learn range with /Worksheet Learn Erase and record the keystrokes again.

To Name and Run a Macro Created with Learn

1. Name the macro by assigning it a range name, as described in "Naming a Macro" on page 98.
2. Depending on how you named the macro, run it by using either ALT or ALT-F3 (RUN). See "Running a Macro" on page 99.
3. If the macro isn't working as you expected it to, debug and edit it as explained in "Debugging a Macro" on page 101.
4. Save the worksheet to save the macro.

Keep the following information in mind when you use the learn feature:

- You can save a macro created with the learn feature in a macro library. For information about macro libraries, see "Saving Macros in a Library" on page 183.
- Avoid using /Range Justify to make a macro created in a learn range more compact. When you use /Range Justify, 1-2-3 inserts a space in front of each keystroke that it moves up from the row below. These spaces will alter the way the macro works.

Macro Command Categories

1-2-3 macro commands fall into six categories.

| Category | Description |
|-------------------|--|
| Data manipulation | Enter data, edit existing entries, erase entries, and recalculate data. |
| File manipulation | Work with text files. |
| Flow-of-control | Direct the path of a macro to branch, call subroutines, loop, and process conditionally. |
| Interactive | Suspend running a macro for user input, control the timing of a macro, and prevent the user from stopping a macro. |
| Macro key names | Replicate actions of the nonprinting keys. |
| Screen control | Control the screen display, control the mode indicator, sound various tones, and clear control panel prompts. |

The following sections list macro keywords by category. Chapter 4, beginning on page 111, lists each macro command in alphabetical order.

Data Manipulation

These commands enter, copy, edit, erase, and recalculate data.

| Macro instruction | Description |
|--------------------------|--|
| {APPENDBELOW} | Copies a range of data to the bottom of a second range. For a named range, extends the second range to include the data. |
| {APPENDRIGHT} | Copies a range of data to the right of a second range. For a named range, extends the second range to include the data. |
| {BLANK} | Erases a cell or range. |
| {CONTENTS} | Copies the contents of a cell to another cell as a label. |
| {LET} | Enters a label or number in a cell. |
| {PUT} | Enters a label or number in a range. |
| {RECALC} | Recalculates formulas row by row. |
| {RECALCCOL} | Recalculates formulas column by column. |

File Manipulation

These commands work with text files (also called print files or ASCII files).

| Macro instruction | Description |
|-------------------|---|
| {CLOSE} | Closes the open text file. |
| {FILESIZE} | Determines the number of bytes in the open text file and records the number in a cell. |
| {GETPOS} | Determines the file pointer's position in the open text file and records that position in a cell. |
| {OPEN} | Opens a new or existing text file. |
| {READ} | Copies data from the open text file to a cell. |
| {READLN} | Copies a line from the open text file to a cell. |
| {SETPOS} | Repositions the file pointer in the open text file. |
| {WRITE} | Writes a string to the open text file. |
| {WRITELN} | Writes a string to the open text file and adds an end-of-line sequence. |

Flow-Of-Control

These commands direct the path of a macro using subroutines, branches, calls, for-loops, and conditional processing.

| Macro instruction | Description |
|-----------------------|---|
| { <i>subroutine</i> } | Calls a subroutine. The <i>subroutine</i> is a range name that you assigned. Runs the subroutine before running the rest of the original macro. When the subroutine finishes, returns control to the command after { <i>subroutine</i> }. |
| {BRANCH} | Transfers macro control from the current column of macro instructions to another location. |
| {DEFINE} | Evaluates and stores information that you pass to a subroutine in a { <i>subroutine</i> } command. |
| {DISPATCH} | Branches by transferring macro control to the branch location specified in a cell. |
| {FOR} | Creates a for-loop: repeats a subroutine or a series of instructions a specified number of times. |
| {FORBREAK} | Cancels a for-loop. |
| {IF} | Evaluates a condition and continues depending on the results: if the condition is true, continues with the macro instructions that follow {IF} in the same cell; if it is false, goes directly to the instructions in the next cell. |
| {ONERROR} | Branches if an error occurs while a macro is running, so the macro continues instead of terminating at the error. |
| {QUIT} | Ends a macro, returning keyboard control to the user. |

(continued)

| Macro instruction | Description |
|-------------------|---|
| {RESTART} | Keeps 1-2-3 from returning to the location from which the subroutine call was issued. When {RETURN} or a blank cell is encountered, the macro ends. Used in subroutines. |
| {RETURN} | Ends a subroutine and returns control to the instruction following the command that called it. In a for-loop, ends the current repetition immediately and starts the next repetition. |
| {SYSTEM} | Temporarily suspends 1-2-3 and passes a command to the operating system. When the command is completed, resumes the 1-2-3 session and continues the macro |

Interactive

These commands suspend the running of a macro for user input, control macro interruption and the timing of the macro, and create custom prompts.

| Macro instruction | Description |
|-------------------|--|
| {?} | Suspends the running of a macro to let you move the cell pointer, enter data, or select commands. Returns control to the macro when you press ENTER. |
| {BREAK} | Issues a CTRL-BREAK (equivalent to pressing ESC one or more times in a menu), so you can return 1-2-3 to READY mode and continue the macro. |
| {BREAKOFF} | Disables CTRL-BREAK while a macro is running, so the macro cannot be interrupted. |
| {BREAKON} | Restores CTRL-BREAK, undoing a {BREAKOFF} command. |
| {FORM} | Suspends the running of a macro so you can enter and edit data in a specified range. Similar to Range Input, but provides more control over allowable keystrokes. |
| {FORMBREAK} | Ends a {FORM} command. |
| {GET} | Suspends the running of a macro until you press a key and then records that keystroke in a cell. |
| {GETLABEL} | Displays a prompt in the control panel, waits for a response to the prompt, and enters the response as a label in a cell. |
| {GETNUMBER} | Displays a prompt in the control panel, waits for a response to the prompt, and enters the response as a number in a cell. |
| {LOOK} | Checks the computer's typeahead buffer (the buffer in which 1-2-3 stores keystrokes during noninteractive parts of a macro) and records the first keystroke (if any) the buffer contains in a cell. |
| {MENUBRANCH} | Displays a customized menu in the control panel, waits for you to select a menu item, then branches to the macro instructions associated with that menu item. |

(continued)

| Macro instruction | Description |
|--------------------------|--|
| {MENUCALL} | Displays a customized menu in the control panel, waits for you to select a menu item, and then runs the macro instructions associated with that menu item as a subroutine. |
| {WAIT} | Suspends running a macro and displays the WAIT indicator until the time you specify. |

Key Names

The table below lists the macro keywords that correspond to the standard keyboard keys and the 1-2-3 function keys.

| Key | Macro key name |
|---|-----------------------|
| ↓ | {DOWN} or {D} |
| ↑ | {UP} or {U} |
| ← | {LEFT} or {L} |
| → | {RIGHT} or {R} |
| } (close brace) | { } |
| { (open brace) | { { } |
| / (slash) or < (less-than symbol) | /, <, or {MENU} |
| ~ (tilde) | {~} |
| ALT-F7 (APP1) | {APP1} |
| ALT-F8 (APP2) | {APP2} |
| ALT-F9 (APP3) | {APP3} |
| ALT-F10 (APP4) | {APP4} |
| BACKSPACE | {BACKSPACE} or {BS} |
| CTRL-← (BIG LEFT) or SHIFT-TAB (BACKTAB) | {BIGLEFT} |
| CTRL-→ (BIG RIGHT) or TAB | {BIGRIGHT} |
| DEL | {DELETE} or {DEL} |
| END | {END} |
| ENTER | ~ |
| ESC | {ESCAPE} or {ESC} |
| F1 (HELP) | {HELP} |
| F2 (EDIT) | {EDIT} |
| F3 (NAME) | {NAME} |
| F4 (ABS) | {ABS} |
| F5 (GOTO) | {GOTO} |
| F6 (WINDOW) | {WINDOW} |

(continued)

| Key | Macro key name |
|-------------|-----------------------|
| F7 (QUERY) | {QUERY} |
| F8 (TABLE) | {TABLE} |
| F9 (CALC) | {CALC} |
| F10 (GRAPH) | {GRAPH} |
| HOME | {HOME} |
| INS | {INSERT} or {INS} |
| PG UP | {PGUP} |
| PG DN | {PGDN} |

Screen Control

These commands control different parts of the screen display, including the mode indicator, and sound the computer's bell.

| Macro instruction | Description |
|--------------------------|--|
| {BEEP} | Sounds a tone of specified frequency and duration. |
| {BORDERSOFF} | Turns off display of the worksheet column letters and row numbers. (Works the same as {FRAMEOFF}.) |
| {BORDERSON} | Restores display of the worksheet column letters and row numbers, undoing {BORDERSOFF}. (Works the same as {FRAMEON}.) |
| {FRAMEOFF} | Turns off display of the worksheet column letters and row numbers. (Works the same as {BORDERSOFF}.) |
| {FRAMEON} | Restores display of the worksheet column letters and row numbers, undoing {FRAMEOFF}. (Works the same as {BORDERSON}.) |
| {GRAPHOFF} | Removes a graph displayed by {GRAPHON} and redisplay the worksheet. |
| {GRAPHON} | Without suspending the macro, creates a full-screen view of the current graph or makes a named graph the current graph (with or without displaying the graph). |
| {INDICATE} | Changes the mode indicator to the string you specify or restores the standard mode indicator. |
| {PANELOFF} | Freezes the control panel either in its current state or after clearing it. |
| {PANELON} | Unfreezes and displays the control panel, undoing {PANELOFF}. |
| {WINDOW} | Turns the display of dialog boxes on or off. |
| {WINDOWSOFF} | Freezes the worksheet area of the screen. |
| {WINDOWSON} | Unfreezes the worksheet area of the screen, undoing {WINDOWSOFF}. |

Chapter 4

Macro Command Descriptions

This chapter lists the macro commands alphabetically by keyword. Each macro command is described in detail and includes one or more examples to show its syntax and use. Descriptions of the macro commands use the following conventions:

- Macro commands appear in uppercase letters except for the *{subroutine}* command (*subroutine* is not the keyword; you must replace *subroutine* with the name of the subroutine you want to run). You can use either uppercase or lowercase letters when you enter a macro command.
- Argument names for which you must supply information appear in lowercase italics. Substitute the type of information the argument type requires.
- Actual arguments used in examples are not italicized.
- In the examples, ... (ellipses) indicate that the macro example is an excerpt from a longer macro.
- Range names appear in uppercase.
- An argument that requires a value can be either a number or a formula that results in a number. A string argument can be either text, or a formula that results in text. A location argument can be a range name or address, or a formula that results in a range name or address.

{?}

{?} (pause) suspends further execution of the macro until you press ENTER, letting you type any number of keystrokes.

Uses

Use {?} to stop the macro temporarily. You can then move the cell or menu pointer, complete part of a command, or enter data for the macro to process. The macro continues when you press ENTER.

Notes

When you press ENTER, 1-2-3 ends the {?} (pause) command and continues the macro. 1-2-3 does not enter data or complete a menu command, unless the next character in the macro is a ~ (tilde) or, in READY mode, a pointer-movement key name (for example {DOWN}).

Examples

The following macro selects /Range Format Currency with two decimal places and pauses so you can specify the range to format. After you specify the range, 1-2-3 formats the specified range as Currency, with two decimal places.

```
'/rfc2~{?}~
```

The following macro moves the cell pointer to the cell named ERR_MSG, which contains an error message, and pauses to let you read the message. When you press ENTER, the macro continues.

```
{GOTO}ERR_MSG~  
{?}  
{HOME}
```

{~}

{~} (tilde) lets you enter a ~ (tilde) that 1-2-3 does not interpret as ENTER.

{{}} and {} }

{{} and {} } let you enter { (open brace) and } (close brace) without 1-2-3 interpreting them as a macro command.

{ABS}

{ABS [*number*]} is equivalent to pressing F4 (ABS). For more information about 1-2-3 function keys, see the *Quick Reference*.

Argument

number is an optional argument that tells 1-2-3 how many times to press F4 (ABS). *number* is a value, the address or name of a cell that contains a value, or a formula that returns a value. {ABS} without an argument is equivalent to {ABS 1}.

Notes

Use {ABS} to cycle through absolute references and to specify ranges prior to selecting commands.

{APP1}, {APP2}, {APP3}, and {APP4}

{APP1}, {APP2}, {APP3}, and {APP4} invoke add-in programs.

Notes

{APP1}, {APP2}, {APP3}, and {APP4} are equivalent to the following keys.

| Macro name | Description |
|-------------------|---|
| {APP1} | Invokes the add-in program assigned to ALT-F7. |
| {APP2} | Invokes the add-in program assigned to ALT-F8. |
| {APP3} | Invokes the add-in program assigned to ALT-F9. |
| {APP4} | Invokes the add-in program assigned to ALT-F10. If no add-in is assigned to ALT-F10, {APP4} displays the Add-In menu. |

For information about invoking add-in programs, see “Using an Add-In” in Chapter 2 of the *User’s Guide*. For information about invoking the Macro Library Manager add-in, see “To Invoke Macro Library Manager” on page 178.

Example

The following macro invokes the Macro Library Manager add-in, if 9 (ALT-F9) was the key to which you assigned Macro Library Manager.

```
{APP3}
```

{APPENDBELOW} and {APPENDRIGHT}

{APPENDBELOW *target-location,source-location*} copies the contents of *source-location* to the rows immediately below the bottom row of *target-location*.

{APPENDRIGHT *target-location,source-location*} copies the contents of *source-location* to the columns immediately to the right of *target-location*.

Arguments

source-location and *target-location* are named ranges or range addresses. If you specify a named range, 1-2-3 expands the *target-location* after the appended data to include the rows or columns that contain the added data.

Uses

Use {APPENDBELOW} with /Range Input or {FORM} to transfer records from an entry form to a database.

Use {APPENDRIGHT} to add a new field to a database or to add a column of data to a spreadsheet application.

Notes

In the following situations, {APPENDBELOW} or {APPENDRIGHT} fails and the macro stops. 1-2-3 then displays an error message explaining why the macro stopped.

- When you append more rows or columns than can fit between *target-location* and the worksheet boundaries (column IV or row 8192)
- When appending *source-location* to *target-location* would write over data
- When cells below or columns to the right of *target-location* are protected

When *source-location* contains formulas, {APPENDBELOW} or {APPENDRIGHT} copies the current values of the formulas to *target-location*, not the formulas themselves.

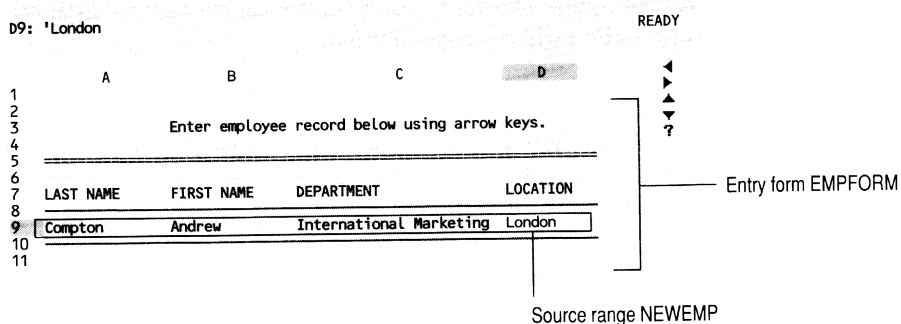
In {APPENDBELOW}, if any range's last row is also the last row of *target-location*, that range name will be expanded to accommodate the appended data. In {APPENDRIGHT}, if any range's right column is also the right column of *target-location*, that range will be expanded to accommodate the appended data.

To use {APPENDBELOW} or {APPENDRIGHT} from a macro library, *source-location* and *target-location* must be worksheet ranges.

Example

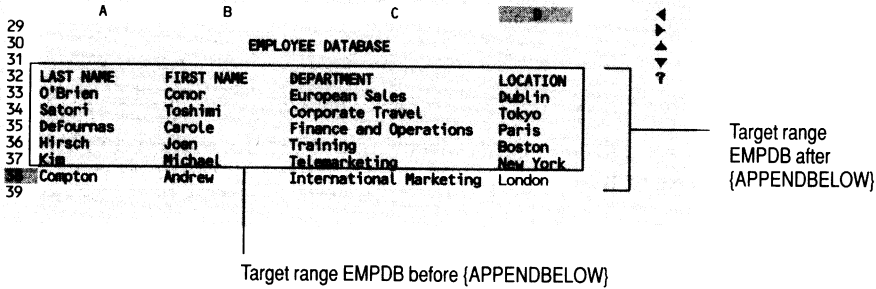
The following macro lets you enter new employee information in an unprotected range named NEWEMP. NEWEMP is in an entry form named EMPFORM. The macro appends the information in NEWEMP to the employee database named EMPDB and expands EMPDB to include the new record.

```
{BLANK NEWEMP}~  
{FORM EMPFORM}  
{APPENDBELOW EMPDB,NEWEMP}
```



D38: 'London

READY



{BACKSPACE} and {BS}

{BACKSPACE [number]} and {BS [number]} are equivalent to pressing BACKSPACE.

Argument

number is an optional argument that tells 1-2-3 how many times to press BACKSPACE. *number* is a value, the address or name of a cell that contains a value, or a formula that returns a value. {BACKSPACE} without an argument is equivalent to {BACKSPACE 1}.

{BEEP}

{BEEP [tone-number]} sounds one of four tones.

Argument

tone-number is an optional argument that tells 1-2-3 which of four tones to sound. *tone-number* must be a value, the address or name of a cell that contains a value, or a formula that returns a value. If you use a number other than 1, 2, 3, or 4 for *tone-number*, 1-2-3 divides the number by four and uses the remainder as *tone-number* (if the remainder is 0, 1-2-3 sounds tone 4). For example, {BEEP 7} is equivalent to {BEEP 3}. {BEEP} without an argument is equivalent to {BEEP 1}.

Uses

Use {BEEP} to get the user's attention: to signal the end of a macro or the end of a waiting period (see {WAIT} on page 162), to alert a user to an on-screen message, or to signal the beginning of an interactive command.

Notes

{BEEP} does not produce a tone when the computer sounds are turned off with /Worksheet Global Default Other Beep No.

Examples

The following excerpt from a macro sounds two tones to draw attention to the subsequent interactive command.

```
{BEEP}{BEEP 4}
{?}
```

The following macro sounds a tone to draw attention to the information displayed in cell `ERR_MSG`. The tone of the bell depends on the cell named `ERR_TONE`.

```
{GOTO}ERR_MSG~
{BEEP ERR_TONE}
```

The following macro ends with three `{BEEP}` commands that sound the same tone three times to signal the macro's end.

```
...
{BEEP 3}{BEEP 3}{BEEP 3}{QUIT}
```

{BIGLEFT} and {BIGRIGHT}

`{BIGLEFT [number]}` is equivalent to pressing `CTRL←` or `SHIFT-TAB`. `{BIGRIGHT [number]}` is equivalent to pressing `TAB` or `CTRL→`.

Argument

number is an optional argument that tells 1-2-3 how many times to press `CTRL←` or `TAB`. *number* is a value, the address or name of a cell that contains a value, or a formula that returns a value. `{BIGLEFT}` or `{BIGRIGHT}` without an argument is equivalent to `{BIGLEFT 1}` or `{BIGRIGHT 1}`.

{BLANK}

`{BLANK location}` erases the contents of *location*. `{BLANK}` does not change the format of the cells in *location*.

Argument

location is the address or name of a cell or range.

Notes

`{BLANK}` is similar to `/Range Erase`, but it is faster and does not force recalculation.

To see the result of a `{BLANK}` command, follow `{BLANK}` with a command that redraws the screen (for example, `{D}` or `~`), or with a `{CALC}` command.

Example

The following macro erases the contents of the range named DATARANGE.

```
{BLANK DATARANGE}
```

{BORDERSOFF} and {BORDERSON}

{BORDERSOFF} and {BORDERSON} are identical to {FRAMEOFF} and {FRAMEON}. Refer to the descriptions of {FRAMEOFF} and {FRAMEON} on page 133.

{BRANCH}

{BRANCH *location*} transfers macro control from the current macro instruction to *location* and does not return to the calling macro.

Argument

Location is the address or name of another macro or subroutine. If you specify a range, 1-2-3 branches to the first cell in the range.

Uses

Use {BRANCH} with {IF} to implement if-then-else processing or to transfer control to another macro.

Use {BRANCH} to create a loop by branching to a cell above the {BRANCH} command in the same macro. This structure is useful for repetitive data entry tasks.

Notes

{BRANCH} is not the same as {GOTO}. {GOTO} moves the cell pointer to another cell. {BRANCH} transfers macro execution to the commands that begin in *location*.

{BRANCH} is one-way; you can return from a {BRANCH} to the calling macro only by branching back to it with a {BRANCH} command or a {DISPATCH} command in the instructions at *location*. {DISPATCH} branches indirectly out of the current macro by branching to the address or range name specified in *location*. To branch out of the current macro and then return to it, use *{subroutine}*. *{subroutine}* automatically returns control to the calling macro once the subroutine is completed.

Examples

The following macro transfers control to either the macro named BIG or the macro named SMALL, depending on the value in the cell named SIZE.

```
{IF SIZE>100}{BRANCH BIG}  
{BRANCH SMALL}
```

The following macro transfers control to the current cell.

```
{BRANCH @CELLPOINTER("address")}
```

{BREAK}

{BREAK} clears the control panel and returns 1-2-3 to READY mode.

Uses

Use {BREAK} at the beginning of a macro to ensure that the macro runs, even if a user runs it while entering data or selecting a 1-2-3 command.

Example

{BREAK} has the same effect as pressing ESC several times to leave a menu. {BREAK} does not stop a macro. To stop a macro, press CTRL-BREAK.

Example

The following macro leaves the current menu and displays a range named MYSCREEN.

```
{BREAK}{GOTO}MYSCREEN~
```

{BREAKOFF} and {BREAKON}

{BREAKOFF} prevents you from canceling a macro with CTRL-BREAK.

{BREAKON} restores the operation of CTRL-BREAK, undoing a {BREAKOFF} command.

Uses

Normally, you can stop a macro while it is running by pressing CTRL-BREAK. While {BREAKOFF} is in effect, however, CTRL-BREAK does not stop the macro. Use {BREAKOFF} to keep users from stopping a macro to alter data or look at restricted data in a protected application.

Notes

{BREAKOFF} stays in effect until 1-2-3 performs a {BREAKON} command or until the macro ends.

If {BREAKOFF} is in effect and the macro goes into an infinite loop, you cannot return to 1-2-3. To stop the macro, you must restart the computer. In either case, all data entered or changed since the last time the worksheet file was saved is lost. For this reason, do not use {BREAKOFF} while designing and testing a macro. To prevent data loss, precede {BREAKOFF} with /File Save in the macro.

{BREAKOFF} and {BREAKON} have no effect on {BREAK}.

Example

The following excerpt from a macro disables CTRL-BREAK before starting the PAYROLL subroutine, preventing you from gaining access to proprietary information by stopping the macro while the PAYROLL subroutine is running. When the PAYROLL subroutine ends, {BREAKON} restores CTRL-BREAK for the rest of the macro.

```
...  
{BREAKOFF}  
{PAYROLL}  
{BREAKON}  
...
```

{CALC}

{CALC *number*} is equivalent to pressing F9 (CALC). For more information about 1-2-3 function keys, see the *Quick Reference*.

Argument

number is an optional argument that tells 1-2-3 how many times to press F9 (CALC). *number* is a value, the address or name of a cell that contains a value, or a formula that returns a value greater than 1. *number* specifies the number of times you want 1-2-3 to perform the calculations. {CALC} without an argument is equivalent to {CALC 1}.

Uses

Use {CALC} if you use Manual recalculation when running a macro and you need to control when 1-2-3 recalculates formulas.

Include {CALC} at the end of a macro that uses {RECALC} or {RECALCCOL} to ensure that 1-2-3 updates all formulas in the worksheet.

If worksheet recalculation is set to manual, use {CALC} before you select /Range Transpose or /Range Value to make sure the values in the cells you specify are up to date.

Use {CALC} to see the results of a {BLANK}, {LET}, {APPENDRIGHT}, or {APPENDBELOW} command.

Notes

{CALC} does not update linked formulas. Use /File Admin Link-Refresh to update linked formulas.

{CLOSE}

{CLOSE} closes a text file that you opened with {OPEN} (if one is open) and saves any changes made to the file.

Uses

Use {CLOSE} to close text files when you finish using them. If you don't include a {CLOSE} command in a macro that contains an {OPEN} command, the text file remains open until you end your 1-2-3 work session.

Notes

Only one text file can be open at one time, so using an {OPEN} command when a text file is already open closes the first text file before it opens the second text file. You do not need a {CLOSE} command between {OPEN} commands.

Following the {CLOSE}, 1-2-3 continues with the command in the next cell.

Example

The following macro opens a text file named STOCKS with append access (see {OPEN} on page 148), adds a line to the file to report the day's volume for a stock, and closes the file before ending the macro. (Without the {CLOSE} command, STOCKS would remain open at the end of the macro, and you could continue processing STOCKS in a subsequent macro without using an {OPEN} command.)

```
{OPEN "STOCKS.TXT","a"}  
{WRITELN "Today's volume: "&@STRING(VOLUME,0)}  
{CLOSE}  
{QUIT}
```

{CONTENTS}

{CONTENTS *target-location,source-location,[width],[cell-format]*} copies a value from *source-location* to *target-location* as a label.

Arguments

source-location and *target-location* are the names or addresses of cells or ranges. If you specify ranges, 1-2-3 uses the first cells of the ranges.

width is an optional argument that specifies the width of the label that 1-2-3 creates. *width* is a value, the address or name of a cell that contains a value, or a formula that returns a value from 0 through 240.

If you copy a value without specifying the optional *width* and *cell-format* arguments, the label 1-2-3 creates in *target-location* has the same width and format as *source-location*.

cell-format is an optional argument that specifies the format of the label 1-2-3 creates. *cell-format* must be a value corresponding to a code number, or the address or name of a cell that contains such a value.

| Code number | Format |
|--------------------|--|
| 0 through 15 | Fixed, 0 through 15 decimal places |
| 16 through 31 | Scientific, 0 through 15 decimal places |
| 32 through 47 | Currency, 0 through 15 decimal places |
| 48 through 63 | Percent, 0 through 15 decimal places |
| 64 through 79 | Comma, 0 through 15 decimal places |
| 112 | +/- |
| 113 | General |
| 114 | D1 (DD- <i>MMM</i> - <i>YY</i>) |
| 115 | D2 (DD- <i>MMM</i>) |
| 116 | D3 (<i>MMM</i> - <i>YY</i>) |
| 117 | Text |
| 118 | Hidden |
| 119 | D6 (HH:MM:SS AM/PM) |
| 120 | D7 (HH:MM AM/PM) |
| 121 | D4 (Long International date format) |
| 122 | D5 (Short International date format) |
| 123 | D8 (Long International date format) |
| 124 | D9 (Short International date format) |
| 127 | Global cell format (specified with /Worksheet Global Format) |

Uses

Use {CONTENTS} to store a value as a label so you can use it in a text formula. Use {CONTENTS} with the Text format number 117 to obtain a formula as text rather than its result.

Notes

If you include a *cell-format* argument, you must include a *width* argument.

To copy a label from one cell to another, use {LET} instead of {CONTENTS}.

Examples

In the examples that follow, the cell named INCOME contains the formula +GROSS-EXP, which results in \$167.24. INCOME is formatted as Currency, 2

decimal places, and its column width is 9. In the explanations of the examples, each bullet represents one space.

```
{CONTENTS REPORT,INCOME}  
+“Today we earned”&REPORT{CALC}~
```

The macro enters the 9-character label •\$167.24• in cell REPORT, then creates the sentence “Today we earned \$167.24 ” and enters it in the current cell.

```
{CONTENTS REPORT,INCOME,11,117}  
+“The formula we use to calculate earnings is: ”&REPORT{CALC}~
```

The macro enters the label +GROSS-EXP• in REPORT (Code 117 is Text format), then creates the sentence “The formula we use to calculate earnings is:•+GROSS-EXP•” and enters it in the current cell.

```
{CONTENTS REPORT,INCOME,3}
```

The macro places the 3-character label *** in REPORT, because the specified width is not wide enough to display \$167.24.

{DEFINE}

{DEFINE *location1,location2,...locationn*} specifies where to store arguments for a {*subroutine*} command where *locationn* is the last of several arguments in a list.

Arguments

location is the address or name of a cell or range that is unprotected. If *location* is a range, 1-2-3 uses the first cell of the range as the storage location.

Specify a *location* argument for each argument in the {*subroutine*} command. If you do not, 1-2-3 ends the macro when it reaches the {DEFINE} command and displays an error message.

Uses

Use {DEFINE} at the beginning of any subroutine to which you pass arguments.

Notes

{DEFINE} must be the first macro command in the subroutine.

You can specify the type of data (string or value) that 1-2-3 is to store in *location*. 1-2-3 stores the subroutine arguments as labels unless you add the suffix :value (or :v) to the *location* arguments.

| Suffix | What it does |
|---------------|--|
| :s or :string | Stores the argument as a label, even if the argument looks like a number, formula, or cell or range address. |
| :v or :value | Stores the argument as a value. |

You can define up to 31 arguments. However, when you use suffixes with {DEFINE}, 1-2-3 treats the suffix as an additional argument. This means that you can have up to 15 arguments passed by value to a subroutine.

Examples

The following examples illustrate how to use the {DEFINE} command. The {*subroutine*} command in macro \A passes three arguments to SUBR1. The {DEFINE} command at the beginning of SUBR1 stores the label @NOW in cell ONE, the label "Closing Price:" in cell TWO, and the label FINAL in cell THREE. The {LET} commands then enter the labels in three consecutive cells in a row.

```
\A      {SUBR1 @NOW,"Closing Price:",FINAL}
...
SUBR1   {DEFINE ONE,TWO,THREE}
        {LET @CELLPOINTER("address"),ONE}{RIGHT}
        {LET @CELLPOINTER("address"),TWO}{RIGHT}
        {LET @CELLPOINTER("address"),THREE}~
```

The result is a row that contains the date number for today's date followed by the labels:

Closing Price: FINAL

The {*subroutine*} command in macro \B passes three arguments to SUBR2. The {DEFINE} command at the beginning of SUBR2 evaluates the arguments before storing them. Thus, it stores the value of the first argument, today's date, as a number in cell ONE; the second argument, the string Closing Price:, as a label in cell TWO; and the value of the third argument, the contents of the cell named FINAL, as a number in cell THREE.

SUBR2 then formats the current cell as DD-MMM and enters the number stored in cell ONE, moves right one cell and enters the label stored in cell TWO, moves right one cell again, formats the cell as Currency with two decimal places, and enters the number stored in cell THREE.

```
\B      {SUBR2 @NOW,"Closing Price:",FINAL}
...
SUBR2   {DEFINE ONE:V,TWO,THREE:V}
        /rfd2~~{LET @CELLPOINTER("address"),ONE}{RIGHT}
        {LET @CELLPOINTER("address"),TWO}{RIGHT}
        /rfc2~~{LET @CELLPOINTER("address"),THREE}
```

The result is a row that reads (depending on the date and closing price)

01-Apr Closing Price: \$9.32

{DELETE} and {DEL}

{DELETE [*number*]} and {DEL [*number*]} are equivalent to pressing DEL.

Argument

number is an optional argument that tells 1-2-3 how many times to press DEL. *number* is a value, the address or name of a cell that contains a value, or a formula that returns a value. {DELETE} without an argument is equivalent to {DELETE 1}.

{DISPATCH}

{DISPATCH *location*} transfers macro control to the cell whose address or name you specify in the *location* cell.

Argument

location is a single cell that contains the address or name of the cell to which macro control is transferred.

Uses

Use {DISPATCH} to have 1-2-3 branch to one of several possible macros, depending on the contents of *location*. {DISPATCH} is particularly useful in macros that change depending on conditions in the worksheet.

Notes

{DISPATCH} does not return control to the calling macro. If you want to return control to the calling macro, use {*subroutine*}. {*subroutine*} automatically returns control to the calling macro once the subroutine is completed.

Example

The following excerpt from a macro sets the label in the cell named SWITCH, and then transfers macro control to the macro whose name is in SWITCH.

```
...
{IF COST<0}{LET SWITCH,"negative"}~
{IF COST=0}{LET SWITCH,"zero"}~
{IF COST>0}{LET SWITCH,"positive"}~
{DISPATCH SWITCH}
```

{DOWN} and {D}

{DOWN [*number*]} and {D [*number*]} are equivalent to pressing ↓.

Argument

number is an optional argument that tells 1-2-3 how many times to press ↓. *number* is a value, the address or name of a cell that contains a value, or a formula that returns a value. {DOWN} without an argument is equivalent to {DOWN 1}.

{EDIT}

{EDIT} is equivalent to pressing F2 (EDIT). For more information about 1-2-3 function keys, see the *Quick Reference*.

Uses

Use {EDIT} to alter the contents of a cell or to enter SETTINGS mode.

{END}

{END} is equivalent to pressing END.

{ESCAPE} and {ESC}

{ESCAPE [*number*]} and {ESC [*number*]} are equivalent to pressing ESC.

Argument

number is an optional argument that tells 1-2-3 how many times to press ESC. *number* is a value, the address or name of a cell that contains a value, or a formula that returns a value. {ESCAPE} without an argument is equivalent to {ESCAPE 1}.

{FILESIZE}

{FILESIZE *location*} counts the number of bytes in the open text file and stores that number in *location*.

Argument

location is the address or name of a cell or range, or a formula that returns the address or name of a cell or range. If you specify a range, 1-2-3 enters the number in the first cell of the range.

Uses

Use {FILESIZE} after every {OPEN} command to determine the size of the file before the macro reads or writes data.

Notes

Before you use {FILESIZE}, use {OPEN} to retrieve the text file that you want to use.

Example

The following macro enters in cell MAXBYTES the number of bytes in the open text file. The {READ} command then copies 72 bytes from the text file into the current cell and moves down one cell to the next row. The macro then uses {GETPOS} and {IF} to determine if the byte pointer has reached the end of the file. If the byte pointer has not reached the end of the file, the macro branches to R_LOOP to read an additional 72 characters of text. When all text has been read, the macro closes the file.

```
\R          {FILESIZE MAXBYTES}
R_LOOP     {READ 72,@CELLPOINTER("address")}
           {D}{GETPOS CURRBYTE}
           {IF CURRBYTE=MAXBYTES}{BRANCH FILE_CLOSE}
           {BRANCH R_LOOP}
```

{FOR}

{FOR *counter,start,stop,step,subroutine*} creates a for-next loop — it repeatedly performs a call to *subroutine*. The *start*, *stop*, and *step* numbers determine the total number of repetitions, and *counter* keeps a running count of the repetitions.

Arguments

counter is the address or name of a blank cell where 1-2-3 keeps track of the number of times the *subroutine* will run during the for-next loop. *counter* should be a blank cell, since anything in *counter* is replaced.

start is the initial value for *counter*.

stop is the value that tells 1-2-3 when to terminate the for-next loop.

step is the value added to counter each time 1-2-3 runs the subroutine.

start, *stop*, and *step* are values, the addresses or names of cells that contain values, or formulas that return values.

subroutine is the range address or name of the subroutine that 1-2-3 runs in the for-next loop.

Uses

Use {FOR} to control the execution of a for-next loop.

Notes

1-2-3 evaluates the *start*, *stop*, and *step* values before it runs a subroutine in a for-next loop. Before each pass through the subroutine, 1-2-3 does the following:

1. Compares the number in *counter* with the *stop* number. If the number in *counter* has not passed *stop*, 1-2-3 runs *subroutine*.
2. Adds *step* to the number in *counter* at the end of each pass through the subroutine. (Initially, the value in *counter* is set to *start*).
3. Repeats steps 1 and 2 until *counter* passes *stop*. 1-2-3 then ends the for-next loop and completes the instructions following the {FOR} command.

If *step* is 0, the number in counter can never exceed the *stop* number, and the for-next loop becomes an infinite loop. Press CTRL-BREAK to stop an infinite for-next loop.

You can modify a for-next loop by using range names for *start*, *stop*, or *step*.

Examples

The following examples show different combinations of using a for-next loop to call a subroutine called FORMAT. The *counter* is the cell named NUM.

Repeat FORMAT 10 times:

```
{FOR NUM,1,10,1,FORMAT}
```

Repeat FORMAT 4 times:

```
{FOR NUM,1,10,2.5,FORMAT}
```

Repeat FORMAT 5 times:

```
{FOR NUM,10,1,-2,FORMAT}
```

Repeat FORMAT indefinitely (*step* is 0 so *counter* never reaches *stop*):

```
{FOR NUM,1,10,0,FORMAT}
```

FORMAT is never called because *start* (10) is greater than *stop* (1), and *step* (2) is positive:

```
{FOR NUM,10,1,2,FORMAT}
```

{FORBREAK}

{FORBREAK} cancels a for-next loop created by a {FOR} command.

Uses

Use {FORBREAK} to return to the calling macro and perform the instruction following the {FOR} command.

Notes

Use {FORBREAK} only within a for-next loop. Using {FORBREAK} anywhere else ends the macro and causes 1-2-3 to display an error message.

Examples

In macro \A below, 1-2-3 repeats subroutine ENTRY up to ten times, to let you enter names in a roster. If you press ENTER at the {GETLABEL} command instead of typing a name, the {FORBREAK} command terminates the for-next loop and 1-2-3 continues immediately with the instructions following the {FOR} command.

```
\A      {BLANK ROSTER}
        {GOTO}ROSTER~
        {FOR COUNTER,1,10,1,ENTRY}
        ...
ENTRY   {GETLABEL "Enter name: ",@CELLPOINTER("address")}
        {IF @CELLPOINTER("contents")=""}{FORBREAK}
        {DOWN}
```

The following excerpt from a macro uses {FORBREAK} to end a for-next loop if the cell named TOTAL contains ERR. If TOTAL contains a value greater than 500, 1-2-3 performs the {RETURN} command and begins the next repetition of the loop (see the {RETURN} command later in this chapter). If TOTAL contains anything else, 1-2-3 continues running the subroutine.

```
...
        {IF @ISERR(TOTAL)}{FORBREAK}
        {IF TOTAL>500}{RETURN}
...
```

{FORM}

{FORM *input-location*,[*call-table*],[*include-list*],[*exclude-list*]} suspends a macro temporarily so you can enter and edit data in the unprotected cells in *input-location*.

Arguments

input-location is a range of any size that contains at least one unprotected cell. This is the range where you enter data. *input-location* cannot include hidden columns.

call-table is an optional two-column range. Each cell in the first column contains one or more macro names of keyboard keys (see page 109). Each adjacent cell in the second column contains a set of macro instructions that 1-2-3 performs when you press the key(s) listed in the first column.

include-list is an optional range that lists allowable keystrokes. You can specify any character key and any key name. Each cell in the range can contain one or more of the character keys and key names listed on page 109.

exclude-list is an optional range that lists unacceptable keystrokes. Each cell in the range can contain one or more keystrokes. 1-2-3 beeps when you press an excluded key. If you specify an *include-list*, do not specify an *exclude-list*, and vice-versa. 1-2-3 uses only one list argument. If you specify both an *include-list* and an *exclude-list*, 1-2-3 uses the *include-list*.

You can use any of the optional arguments without using the ones that precede it by inserting an extra argument separator as a placeholder. For example, to use a *call-table* and an *exclude-list* without an *include-list*, use the syntax {FORM *input-location,call-table,,exclude-list*} (two argument separators between *call-table* and *exclude-list*).

There should be no punctuation between the names of the keyboard keys in *call-table*, *include-list*, and *exclude-list*. Also, the table and lists are case-sensitive: for example, if the *include-list* contains an uppercase B, but not a lowercase b, 1-2-3 allows you to enter only uppercase B's during the {FORM} command and ignores lowercase b's.

To use {FORM} from a macro library, *input-location* must be a worksheet range. You can specify a *call-table*, *include-list*, and *exclude-list* from a macro library or the worksheet.

Uses

Use {FORM} to fill data entry forms.

Notes

{FORM} is similar to /Range Input, but the three optional arguments (*call-table*, *include-list*, and *exclude-list*) give you more control over user entries than /Range Input allows.

When 1-2-3 performs a {FORM} command, it moves the cell pointer to the first unprotected cell in *input-location*, suspends the macro, and waits for you to press a key. What happens next depends on whether {FORM} uses optional arguments.

| If {FORM} uses | What 1-2-3 does |
|-----------------------|--|
| No optional arguments | 1-2-3 processes keystrokes exactly as it does during /Range Input. To cancel the {FORM} command in READY mode, you press ENTER or ESC. 1-2-3 continues the macro at the instruction following the {FORM} command, and leaves the cell pointer where it was when you pressed ENTER or ESC. If you use an <i>include-list</i> , you must specify ~ (tilde) and {ESC} in the list or you cannot use ENTER or ESC to end the {FORM} command. |

(continued)

| If {FORM} uses | What 1-2-3 does |
|--------------------|--|
| Optional arguments | <p>1-2-3 proceeds as follows:</p> <p>If you press a key and the {FORM} command includes a <i>call-table</i> argument, 1-2-3 checks the first column of the <i>call-table</i>. If the keystroke is listed, 1-2-3 performs the instructions in the second column as a subroutine, and then returns to the {FORM} command and waits for you to press another key. If you specified {ESC} or ~ (tilde) in a <i>call-table</i> subroutine, 1-2-3 suspends the {FORM} command and allows you to select 1-2-3 keys and menus (see “Suspending a {FORM} Command” below).</p> <p>If a keystroke is not listed in <i>call-table</i>, and the {FORM} command includes an <i>include-list</i>, 1-2-3 checks the <i>include-list</i>. If the keystroke is in the <i>include-list</i>, 1-2-3 performs the keystroke; otherwise, 1-2-3 ignores the keystroke and beeps.</p> <p>If a keystroke is not listed in <i>call-table</i>, and the {FORM} command includes an <i>exclude-list</i>, 1-2-3 checks the <i>exclude-list</i>. If the keystroke is in the <i>exclude-list</i>, 1-2-3 ignores the keystroke and beeps; otherwise, 1-2-3 performs the keystroke.</p> |

You can **nest forms** (place one form within another form) by making a call to {FORM}. 1-2-3 Release 2.3 lets you nest up to eight forms.

Suspending a {FORM} Command

Including {ESC} or ~ (tilde) in a *call-table* subroutine lets you move the cell pointer out of *input-range*'s unprotected area and use all 1-2-3 keys and menus for the rest of the *call-table* subroutine. When the *call-table* subroutine ends, 1-2-3 moves the cell pointer to wherever it was when the *call-table* subroutine started (unless the cell pointer is within *input range*'s unprotected area when the subroutine ends, in which case 1-2-3 leaves the cell pointer where it is) and reinstates use of 1-2-3 keys as defined by the {FORM} command.

To end a macro from within a *call-table* subroutine, use {RESTART} or {QUIT} in the subroutine. (See the example below.) To end a {FORM} command from within a *call-table* subroutine and continue the macro, use {FORMBREAK} to leave the {FORM} command and continue the macro at the instruction immediately following the {FORM} command. See {FORMBREAK} on page 132.

Examples

The {FORM} command in the following example uses a *call-table* and an *exclude-list*. This {FORM} command processes inventory orders in a database using the entry form shown in the following illustration.

```

E10: U 0.19
                                                READY
      A      B      C      D      E      F      G
1     INVENTORY ORDER FORM
2     _____
3     _____
4     Enter order information below.
5     Press [INS] to transfer order to database.
6     Press [END] to stop entering orders.
7
8     DATE      ITEM      ITEM #    QTY      UNIT PRICE
9     01-Apr-91  Pen      122      2000     0.19
10
11

```

Entry form (range ENTRYFORM)

Input area (unprotected range INPUT_AREA)

The {FORM} command, its *call-table*, and its *exclude-list* are shown in the following illustration. The extra argument separator in the {FORM} command (between SIGKEYS and BADKEYS) indicates the absence of an *include-list*.

```

I21: 'VA
                                                READY
      J      K      L      M      N      O      P
21  VA      {FORM ENTRYFORM,,SIGKEYS,,BADKEYS}
22
23  BADKEYS {WINDOW}{QUERY}
24
25  SIGKEYS {INS}  {APPENDBELOW ORDER_DB,,INPUT_AREA}{BLANK INPUT_AREA}
26  {END}   {BRANCH CONFIRM}
27
28  CONFIRM {GETLABEL "Stop entering orders? (y/n)",CHOICE}
29  {IF BUPPER(CHOICE)="Y"}{FORMBREAK}
30
31
32  CHOICE  y
33
34
35
36
37

```

{FORM} command macro VA

Exclude-list (range BADKEYS)

Call-table (range SIGKEYS)

{FORM ENTRYFORM,SIGKEYS,,BADKEYS} stops the macro temporarily so you can enter an order in the entry form. The *call-table*, SIGKEYS (J25..K26), includes two key names: {INS} and {END}.

- If you press INS during the {FORM} command, 1-2-3 appends the data in INPUT_AREA to the order database (ORDER_DB), erases INPUT_AREA, and returns to the {FORM} command. See {APPENDBELOW} on page 113.
- If you press END during the {FORM} command, 1-2-3 branches to subroutine CONFIRM, which uses a {GETLABEL} command to confirm that you want to stop entering orders. If you enter y at the {GETLABEL} prompt, the {FORMBREAK} ends the {FORM} command. If you type any other letter, 1-2-3 returns to the {FORM} command.

The *exclude-list*, BADKEYS (J23), contains two key names: {WINDOW} and {QUERY}. If you press either of these keys during the {FORM} command, 1-2-3 ignores the keystroke.

{FORMBREAK}

{FORMBREAK} ends a {FORM} command canceling the current form.

Uses

Use {FORMBREAK} to leave the current form. You can also use {FORMBREAK} to end a nested {FORM} command and have 1-2-3 return you to the previous form. 1-2-3 continues at the command following the {FORM} command.

Use {FORMBREAK} only within a {FORM} command.

Notes

Use {FORMBREAK} only within a *call-table* subroutine or a subroutine to which you transfer control from a *call-table* subroutine with {BRANCH} or {DISPATCH}. If you use {FORMBREAK} without first using a {FORM} command, 1-2-3 ends the macro and displays an error.

Example

The following macro lets you enter an invoice number, the date, and a company name in an INVOICE form. Pressing + (plus) moves you to the ITEM form where you can enter one or more items for the current invoice.

INVOICE form

```
10 .....
11 |VA      {FORM INVOICE, INV_CALL}
12 INV_CALL +   {FORM ITEM, IT_CALL}
13           {INS}   {APPENDBELOW ORDERS, INVOICE} {BLANK INVOICE}
14
15 IT_CALL  {INS}   {BRANCH ADD_ITEM}
16           {DEL}   {FORMBREAK}
17
18 ADD_ITEM {APPENDBELOW ORDERED, ITEM}
19           {BLANK ITEM}
20           {FORMBREAK}
```

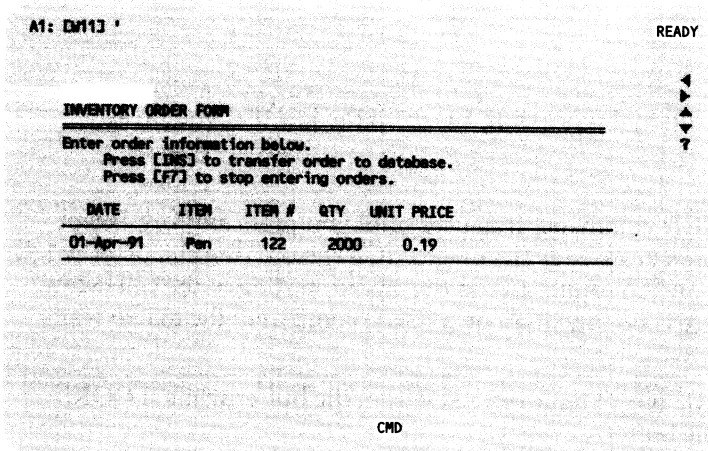
{FORM INVOICE, INV_CALL} stops the macro temporarily so you can enter data in the INVOICE entry form. Pressing + (plus) runs the nested ITEM form.

ITEM has a call table named IT_CALL (A15..C16), which includes the keys named DEL and INS. If you press DEL while entering items for the current invoice, a {FORMBREAK} returns you to the INVOICE form. If you press INS, the macro adds the items to the database and uses {FORMBREAK} to return you to the INVOICE form. You can then press INS to insert the invoice and display a blank form or you can press + (plus) to add another item to the form.

{FRAMEOFF} and {FRAMEON}

{FRAMEOFF} suppresses display of the worksheet frame (column letters and row numbers). The frame is suppressed until 1-2-3 performs a {FRAMEON} command or the macro ends. {FRAMEON} redisplay the worksheet frame hidden by a {FRAMEOFF} command. {FRAMEOFF} and {FRAMEON} are identical to {BORDERSOFF} and {BORDERSON}.

The following illustration shows a screen without the worksheet frame.



Uses

Use {FRAMEOFF} and {FRAMEON} when column letters and row numbers might distract users.

Notes

If a {WINDOWSOFF} command was performed earlier in the macro, precede {FRAMEOFF} or {FRAMEON} with a {WINDOWSON} command. The effects of {FRAMEOFF} or {FRAMEON} will not be visible until you issue a command that redraws the screen (for example, {D} or ~).

Examples

The following excerpt from a macro turns off the worksheet frame during a {FORM} command and then redisplay the frame.

```
{FRAMEOFF}  
{FORM ORDERFORM}  
{FRAMEON}  
...
```

{GET}

{GET *location*} suspends a macro until you press a key, and then records the keystroke in *location*.

Argument

location is the address or name of a cell or range, or a formula that returns the address of a cell or range. If you specify a range, 1-2-3 records the keystroke in the first cell in the range.

Uses

Use {GET} to perform conditional processing based on the key that you press.

Notes

When 1-2-3 performs a {GET} command, it pauses until you press any key except CTRL-BREAK. Pressing CTRL-BREAK ends the macro. {GET} stores the character or key name as a label in *location* and continues to run the macro starting in the cell below the {GET} command. There is no time limit on a {GET} command; the macro waits indefinitely for a keystroke.

{GET} differs from {LOOK}. {GET} removes a key from the buffer while {LOOK} copies a keystroke from the buffer.

When you press ESC, {GET} records {ESC} and when you press DEL, {GET} records {DEL}. For all other keys, {GET} records the full key name (for example, {BACKSPACE} or {RIGHT} instead of {BS} or {R}).

Example

Macro \A prompts you to choose Daily or Monthly (by typing D or M). It then stores the keystroke in the cell named CHOICE. If the keystroke in CHOICE is D, 1-2-3 branches to DAY; if it is M, 1-2-3 branches to MONTH. If the keystroke is anything else, 1-2-3 beeps and starts the macro again.


```

\A      {GOTO}EXPENSES~
        Choose D(aily) or M(onthly)
        {GET CHOICE}
        {ESC}
        {IF @UPPER(CHOICE)="D"}{BRANCH DAY}
        {IF @UPPER(CHOICE)="M"}{BRANCH MONTH}
        {BEEP}{BRANCH \A}

```

{GETLABEL}

{GETLABEL *prompt,location*} prompts you to type information, suspends the macro while you type a response, and stores the response as a label in *location*.

Arguments

prompt is text (up to 72 characters can be displayed), the address or name of a cell that contains the prompt, or a text formula that results in the prompt.

location is the address or name of a cell or a range, or a formula that returns the address or name of a cell or a range. If you specify a range, 1-2-3 stores the user's response in the first cell of the range.

Notes

When 1-2-3 performs a {GETLABEL} command, it displays *prompt* in the control panel and pauses for you to enter a response (up to 80 characters). When you press ENTER, 1-2-3 stores the entry as a label in *location* and continues the macro. If you press ENTER without typing anything, 1-2-3 enters a left-aligned label prefix in *location*, creating a label without any text.

If *prompt* is a cell address or name, {GETLABEL} displays the cell contents as the prompt if the cell contains a label. If the cell contains a text formula, {GETLABEL} displays the result of the formula as the prompt. If the cell contains a value or is blank, {GETLABEL} returns an error.

The {GETLABEL} prompt and your response appear in the control panel even after a {PANELOFF} command.

Examples

The following {GETLABEL} command prompts you to type a product name, and then stores the name in a cell named PRODUCT:

```
{GETLABEL "Type a product name and press Enter: ",PRODUCT}
```

In the following {GETLABEL} command, the macro gets the prompt from CITY_PROMPT, and then enters your response as a label in cell CITY:

```
{GETLABEL CITY_PROMPT,CITY}
```

In the following {GETLABEL} command, the macro gets the prompt from PHONE_PROMPT, and then enters your response in the current cell:

```
{GETLABEL PHONE_PROMPT,@CELLPOINTER("address")}
```

{GETNUMBER}

{GETNUMBER *prompt,location*} prompts you to type a number, suspends the macro while you respond, and stores the response in *location*.

Arguments

prompt is any text (up to 72 characters can be displayed), or the address or name of a cell that contains the prompt, or a text formula that results in the prompt.

location can be the address or name of a cell or a range, or a formula that returns the address or name of a cell or a range. If you specify a range, 1-2-3 stores the user's response in the first cell of the range.

Notes

When 1-2-3 performs a {GETNUMBER} command, it displays *prompt* in the control line and pauses for you to enter a value (up to 80 characters for a formula). When you press ENTER, 1-2-3 stores the entry as a value in *location* and continues the macro.

If you enter a label, text formula, or the address or name of a cell that contains a label or text formula, 1-2-3 enters ERR in *location*. 1-2-3 also enters ERR if you press ENTER without typing anything.

Examples

The following {GETNUMBER} command prompts for your age, and then enters the response in the current cell.

```
{GETNUMBER "Enter your age: ",@CELLPOINTER("address")}
```

The following macro stores your response to the prompt in cell NEWSALES, and then checks the contents of NEWSALES. If NEWSALES contains ERR (you entered a non-numeric response), the macro branches to RETRY, which contains instructions to beep and start the macro again. If NEWSALES contains a number, the macro adds the number to the current value in YTD_SALES and stores the result in YTD_SALES.

```
\S      {GETNUMBER "Enter month's sales: ",NEWSALES}  
        {IF @ISERR(NEWSALES)}{BRANCH RETRY}  
        {LET YTD_SALES,YTD_SALES+NEWSALES}  
  
RETRY  {BEEP}{BRANCH \S}
```

{GETPOS}

{GETPOS *location*} retrieves the character offset position of the **byte pointer** (the pointer that moves character by character through an open text file) and enters the number of characters between the beginning of the file and the byte pointer in *location*.

Arguments

location is the address or name of a cell or a range, or a formula that returns the address or name of a cell or a range. If you specify a range, 1-2-3 enters the number in the first cell in the range.

Uses

Use {GETPOS} to track how much data has been read from a text file. For example, if you want to read the first 100 bytes from a text file when you are not sure how many bytes at a time will be read, use {GETPOS} in the loop that reads and processes the bytes. When *location* contains a number greater than 100, branch out of the loop.

Notes

You must use {OPEN} before you can use {GETPOS}.

After 1-2-3 stores the character offset position in *location*, the macro continues in the cell below the {GETPOS} command. If no text file is open, 1-2-3 ignores a {GETPOS} command and continues to the next macro instruction in the same cell.

The byte pointer indicates the next byte in the text file to read or write. The first character position in a text file is 0, not 1, so if the byte pointer is on the first byte in the file, {GETPOS} enters 0 in *location*. If the byte pointer is on the tenth byte, {GETPOS} enters 9; and so on.

Example

The following line from a macro records the current position of the byte pointer in cell POINTER. If the {GETPOS} command succeeds, the macro continues in the next cell. If a text file is not open, the macro branches to FAIL, which contains further instructions.

```
...
{GETPOS POINTER}{BRANCH FAIL}
{IF POINTER<MAXBYTES}{BRANCH READLOOP}
{QUIT}
```

{GOTO}

{GOTO} is equivalent to pressing F5 (GOTO). For more information about 1-2-3 function keys, see the *Quick Reference*.

Notes

Use ~ (tilde) after a {GOTO} command (for example, {GOTO}C1~).

{GRAPH}

{GRAPH} is equivalent to pressing F10 (GRAPH). For more information about 1-2-3 function keys, see the *Quick Reference*.

Notes

If no current graph is defined when you use {GRAPH}, 1-2-3 beeps and displays a blank screen. To return control to the macro, press any key.

{GRAPHOFF} and {GRAPHON}

{GRAPHON [*named-graph*],[*nodisplay*]} displays a graph using the current settings, or makes *named-graph* the current graph and optionally displays it. {GRAPHON} displays the graph while the macro continues to run. This differs from {GRAPH}, which displays the graph while the macro pauses.

{GRAPHOFF} removes a graph displayed by a {GRAPHON} command and redisplay the worksheet.

Arguments

named-graph is an optional label that matches an available named graph. If *named-graph* is not a graph name, the macro terminates with an error.

nodisplay is the optional label you specify if you want to use the *named-graph* settings, but you do not want to display the graph.

Uses

Use {GRAPHON} with no arguments to display a full-screen view of the current graph while the macro continues to run. The graph remains displayed until 1-2-3 encounters a {GRAPHOFF} command, another {GRAPHON} command, an {INDICATE} or {?} command, a command that displays a prompt or menu in the control panel ({GETLABEL}, {GETNUMBER}, {MENUCALL}, {MENUBRANCH}), /XL, /XM, or /XN), or the end of the macro.

Use {GRAPHON *named-graph*} to make the *named-graph* settings the current graph settings and display a full-screen view of *named-graph* while the macro continues to run. When 1-2-3 reaches a {GRAPHOFF} command, another {GRAPHON} command, an {INDICATE} or {?} command, a command that displays a prompt or menu in the control panel, or the end of the macro, it removes *named-graph* from the screen.

Use {GRAPHON *named-graph*,nodisplay} to make the *named-graph* settings the current graph settings without displaying the graph.

Notes

named-graph refers to a name assigned to graph settings with /Graph Name Create, not the name of the .PIC file that contains the graph.

Current settings are the settings 1-2-3 uses the next time it draws a graph from this worksheet. If you use the *nodisplay* argument, *named-graph* becomes the current group of graph settings, but 1-2-3 does not display the graph.

If no current graph is defined, {GRAPHON} beeps and then returns control to the macro; you do not need to press a key.

Example

This macro displays three named graphs (LINE, BAR, and PIE) at two-second intervals.

```
{GRAPHON LINE}
{WAIT @NOW+@TIME(0,0,5)}
{GRAPHON BAR}
{WAIT @NOW+@TIME(0,0,5)}
{GRAPHON PIE}
{WAIT @NOW+@TIME(0,0,5)}
{GRAPHOFF}
```

{HELP}

{HELP} is equivalent to pressing F1 (HELP). For more information about 1-2-3 function keys, see the *Quick Reference*.

Notes

The macro pauses while you work in Help. When you leave Help, 1-2-3 performs the next command in the macro.

{HOME}

{HOME} is equivalent to pressing HOME.

{IF}

{IF *condition*} evaluates *condition* to determine if it is true or false. If *condition* is true, the macro continues with the next instruction in the same cell as the {IF} command. If *condition* is false, the macro continues with the first instruction in the cell below the {IF} command.

Arguments

condition is a logical expression or the address or name of a cell that contains a logical expression. (A **logical expression** uses one of the logical operators = <> < > <= >= #AND# #NOT# or #OR#.) You can use any formula, number, text, cell address, or cell name in the logical expression as *condition*.

Uses

The {IF} command can implement if-then-else processes like those in programming languages. The instructions that follow the {IF} command in the same cell are the then clause. The instructions in the cell below the {IF} command are the else clause. Include a {BRANCH} or {RETURN} command in the then clause to prevent 1-2-3 from executing the else clause after the then clause.

Notes

When 1-2-3 performs an {IF} command, it evaluates *condition* first. If *condition* results in any value except 0 (zero), 1-2-3 evaluates it as true, and the macro continues in the same cell, with the instruction immediately following the {IF} command.

If *condition* results in 0 (zero), 1-2-3 evaluates it as false, and the macro continues in the cell below the {IF} command. The values that 1-2-3 interprets as zero are 0, a false logical expression, a blank cell, text, ERR, and NA.

If *condition* is true, 1-2-3 performs both the instruction in the same cell as the {IF} command and the instruction in the cell below the {IF} command. If you do not want 1-2-3 to perform both instructions, use {QUIT}, {BRANCH}, or another {IF} following the {IF} command.

Examples

In the following macro, if the entry in the cell named DATE is a value from 21002 through 31959 (the date numbers for 1 July 1957 through 1 July 1987), the macro copies the contents of DATE to the current cell and returns to the calling macro. Otherwise, the macro continues to the {BRANCH} command in the cell below.

```
{IF DATE>21002}{IF DATE<31959}/cDATE~~{QUIT}  
{BRANCH INVALID_DATE}
```

In the following macro, if the entry in the cell named SALARY is greater than \$70,000, the macro performs subroutine HIBRACKET. If SALARY is less than or equal to \$70,000, the macro performs subroutine LOWBRACKET.

```
{IF SALARY>70000}{BRANCH HIBRACKET}
{BRANCH LOWBRACKET}
```

In the following macro, if the cell named TESTVAL contains a true logical expression or any other entry that does not result in zero, the macro completes subroutine {RTN1}, then branches to NEXTRTN. Otherwise, the macro performs subroutine {RTN2}.

```
{IF TESTVAL=1}{RTN1}{BRANCH NEXTRTN}
{RTN2}
```

{INDICATE}

{INDICATE *string*} replaces READY (or another mode indicator) with *string* as the mode indicator. The mode indicator continues to display *string* until 1-2-3 reaches another {INDICATE} command or until you retrieve another file, select /Worksheet Erase Yes, or leave 1-2-3. {INDICATE} with no argument restores the mode indicator that reflects the current mode (READY or WAIT, for example).

Arguments

string is any text that fits in the first line of the control panel, the address or name of a cell that contains the text, or a text formula. Using an empty string as *string* ({INDICATE ""}) removes the mode indicator from the control panel.

Notes

Enclose *string* in quotation marks so it is not confused with a range name.

In FILES and NAMES modes, *string* may hide the mouse and drive icons.

Examples

The following command displays Database Maintenance Macro in the indicator in the control panel. The indicator expands to fit the text.

```
{INDICATE "Database Maintenance Macro"}
```

The following command displays 1 in the mode indicator in the control panel. The mode indicator shrinks to fit the text.

```
{INDICATE "1"}
```

The following command displays the contents of the cell named MSG in the mode indicator in the control panel.

```
{INDICATE MSG}
```

The following command restores the current 1-2-3 mode to the mode indicator in the control panel.

```
{INDICATE}
```

The following command lets you move to a cell and then displays the address of that cell. {GET} stores your next keystroke, and the last {INDICATE} command restores the current mode to the mode indicator in the control panel.

```
{INDICATE "Move the cell pointer to a cell and press ENTER"}  
{?}  
{INDICATE "You moved to cell "&@CELLPOINTER("address")}  
{GET KEY}  
{INDICATE}
```

{INSERT} and {INS}

{INSERT} and {INS} are equivalent to pressing INS.

{LEFT} and {L}

{LEFT [*number*]} and {L [*number*]} are equivalent to pressing ←.

Argument

number is an optional argument that tells 1-2-3 how many times to press ←. *number* is a value, the address or name of a cell that contains a value, or a formula that returns a value. {LEFT} without an argument is equivalent to {LEFT 1}.

{LET}

{LET *location,entry*} enters a number or label in *location*.

Arguments

location is the address or name of a cell or a range, or a formula that returns the address or name of a cell or range. If you specify a range, 1-2-3 enters *entry* in the first cell of the range.

entry can be a number, text, a formula, or the address or name of a cell that contains a number, a label, or a formula.

Uses

Use {LET} with {IF} to vary cell contents depending on a condition or to change entries in a database when you know the actual cell address of the entries you want to change.

Notes

If you use a formula for *entry*, 1-2-3 evaluates the formula and enters the result in *location*. {LET} does not enter formulas.

{LET} does not cause 1-2-3 to automatically recalculate the worksheet. Use a ~ (tilde) or {CALC} after {LET} to recalculate.

```
{LET QTR_2,1.5*QTR_1}~
```

Examples

In the following macro, the {LET} command enters the result of 1.5 times the value in QTR_1 in the cell named QTR_2 if QTR_1 is a defined range name. If QTR_1 is not a range name, the {LET} command enters 1.5*QTR_1 as a label in cell QTR_2.

```
{LET QTR_2,1.5*QTR_1}~
```

The following macro enters 1.5*QTR_1 as a label in the cell named QTR_2.

```
{LET QTR_2,"1.5*QTR_1"}~
```

The following macro enters the result of the text formula +“Ms. ”&FULLNAME in the cell named CUSTOMER.

```
{LET CUSTOMER,+“Ms. ”&FULLNAME}~
```

{LOOK}

{LOOK *location*} checks the typeahead buffer for keystrokes and then records the first keystroke it contains (if any) as a label in *location*.

Argument

location is the address or name of a cell or range, or a formula that returns the address or name of a cell or range. If you specify a range, 1-2-3 records the keystroke in the first cell in the range.

Uses

Use {LOOK} to stop a long macro, break out of an infinite loop, or tell a macro to branch elsewhere. The macro keeps running unless the {LOOK} command records a character that tells the macro to do something else.

Notes

The **typeahead buffer** is a region in memory where 1-2-3 stores keystrokes you type during noninteractive parts of a macro. It contains all the keystrokes typed since the last interactive command or since the macro began. If the buffer is empty when 1-2-3 performs a {LOOK} command, it enters a left-aligned label-prefix character in *location*, creating an empty string.

{LOOK} differs from {GET}. {LOOK} copies a keystroke from the buffer while {GET} removes a keystroke from the buffer. If your macro is waiting for a specific character, use {BLANK} before each {LOOK} command to erase the previous keystroke.

When you press ESC, {LOOK} records {ESC} and when you press DEL, {LOOK} records {DEL}. For all other keys, {LOOK} records the full key name (for example, {BACKSPACE} or {RIGHT} instead of {BS} or {R}).

Example

The following excerpt from a macro requires you to perform specific tasks. The macro begins by erasing the contents of the cell named KEYCELL. Later, the macro sounds two tones and checks whether you typed a character. If the typeahead buffer is empty, the macro loops back to the beginning of TASK. If the typeahead buffer contains a character, the macro branches to the subroutine named NEWTASK.

```
TASK      {BLANK KEYCELL}
          ...
          {BEEP 4}{BEEP 2}
          {LOOK KEYCELL}
          {IF KEYCELL=""}{BRANCH TASK}
          {BRANCH NEWTASK}
```

{MENU}

{MENU} is equivalent to pressing / (slash) or < (less-than symbol), or moving the mouse pointer to the control panel.

Notes

Use : (colon) instead of {MENU} to activate the Wysiwyg menu.

{MENUBRANCH} and {MENUCALL}

{MENUBRANCH *location*} displays in the control panel the macro menu that starts in the first cell of *location*. 1-2-3 waits for you to select an item from the menu and then branches to the macro instructions associated with that item. {MENUBRANCH} does not return to the calling macro when 1-2-3 completes the menu's macro instructions. (See {BRANCH} on page 117.)

{MENUCALL *location*} displays in the control panel the macro menu in *location*. 1-2-3 waits until you select an item from the menu and then calls the subroutine associated with that menu item. (See *subroutine* on page 158.) {MENUCALL} returns to the calling macro when 1-2-3 completes the macro menu instructions.

The following illustrations show a macro menu in a range and the activated macro menu.

```

A13: CW123 'WM
READY
12
13 \M {GOTO}MSG CELL"Select a file from the menu..." Menu macro (M)
14 {MENUBRANCH FILE_CHOICE}
15
16 FILE_CHOICE PERS.WK1 CASH.WK1 INVEN.WK1 RECVS.WK1 PAYS.WK1
17 Personnel Cashflow Inventory Accts_ReceivableAccts_Payables Macro menu
18 /frPERS" /frCASH" /frINVEN" /frRECVS" /frPAYS"
19
20
21
22

```

```

A1: CW123 'Select a file from the menu...
PERS.WK1 CASH.WK1 INVEN.WK1 RECVS.WK1 PAYS.WK1 MENU Menu items
Personnel
2 Select a file from the menu...
3
4
5
6
Menu item description

```

Arguments

location is the address or name of a cell, or a formula that returns the address or name of a cell. *location* must be the first cell of a row that contains the macro menu items (branch or subroutine names).

Notes

Macro menus you create with {MENUBRANCH} and {MENUCALL} work the same way as 1-2-3 menus.

A macro menu appears in the control panel even after a {PANELOFF} command.

Pressing ESC when a macro menu appears in the control panel cancels the {MENUBRANCH} or {MENUCALL} command and returns control to the location from which the {MENUBRANCH} or {MENUCALL} command was issued. The macro continues at the next instruction following the {MENUBRANCH} or {MENUCALL} command.

For more information about creating a macro menu, you can use the \C macro in the sample macro worksheet, SAMPMACS.WK1 (see page 172).

Examples

The following macro uses {MENUBRANCH} to display the macro menu that starts in the cell named REPORTMENU. When you select one of the items in the report menu, 1-2-3 branches to the macro instructions associated with that item. 1-2-3 performs the {BEEP} command only if you press ESC instead of selecting a menu item.

```
{MENUBRANCH REPORTMENU}
{BEEP}
```

The following macro uses {MENUCALL} to display the macro menu that starts in the cell named REPORTMENU. When you select one of the items from the report menu, 1-2-3 calls the set of macro instructions associated with that item as a subroutine. When it completes those instructions, 1-2-3 executes the macro command immediately after the {MENUCALL} (saves the revised file), and then the macro ends. If you press ESC instead of selecting a menu item, 1-2-3 saves the file and the macro ends.

```
{MENUCALL REPORTMENU}
/fs~r
{QUIT}
```

Creating a Macro Menu

1. Decide on a worksheet location for the macro menu.
2. Enter up to eight menu items in consecutive cells in the same row, beginning at *location*. Leave the cell to the right of the final item blank.

Follow these guidelines when entering menu items:

- Menu items can be labels or text formulas. If you enter a formula, 1-2-3 displays the result as the menu item.
 - Each menu item should start with a different character so you can select an item by typing the first character. If two or more menu items have the same first character, 1-2-3 selects the first item (reading from left to right) when you press that character. However, you can select a menu item that has a non-unique first letter by highlighting it and pressing ENTER.
 - Try to make each menu item a single word. If you use multiple-word items, connect the words with a - (hyphen), for example, First-Quarter. Otherwise, a user might think the words are separate menu items.
 - The combined menu items and delimiting spaces are restricted to the screen width. If they exceed the screen width, 1-2-3 displays an error.
 - The column to the right of the last menu item must be blank. A blank cell tells 1-2-3 there are no more items.
3. Enter the command description for each menu item in the cell directly below the menu item. Command descriptions can be labels or text formulas.
 4. Enter the macro instructions for 1-2-3 to branch to ({MENUBRANCH}) or call as subroutines ({MENUCALL}) immediately below the command descriptions (that is, the second row below *location*).

5. Use /Range Name Create to assign a range name to the first menu item in your macro menu. For example, if cell B5 is your first menu item, you might name it File_choice. Using /Range Name Create allows your macro to run in the event that you add or delete rows or columns.

{NAME}

{NAME [*number*]} is equivalent to pressing F3 (NAME). For more information about 1-2-3 function keys, see the *Quick Reference*.

Argument

number is an optional argument that tells 1-2-3 how many times to press F3 (NAME). *number* is a value, the address or name of a cell that contains a value, or a formula that returns a value. {NAME} without an argument is equivalent to {NAME 1}.

{ONERROR}

{ONERROR *branch-location*,*message-location*}} transfers a macro to *branch-location* if certain 1-2-3 errors occur while a macro is running. If you use the optional argument, {ONERROR} records the error message in *message-location*.

Arguments

branch-location is the address or name of a cell or range, or a formula that returns the address or name of a cell or range. *branch-location* contains the macro instructions to which 1-2-3 branches after an error occurs. If you specify a range, 1-2-3 branches to the first cell in the range.

message-location is the address or name of a cell or range, or a formula that returns the address or name of a cell or range where 1-2-3 is to store the error message. If you specify a range, 1-2-3 uses the first cell in the range.

Uses

Use {ONERROR} before any point at which there is a possibility of a fatal error. To continue finding errors, include another {ONERROR} command in the instructions at *branch-location*.

Notes

{ONERROR} traps only **fatal errors** (errors that return 1-2-3 to READY mode), such as the error "Disk drive not ready" during a /File Save). {ONERROR} does not trap macro **syntax errors** (typographical errors in macro instructions that prevent 1-2-3 from interpreting the instructions). When 1-2-3 encounters a syntax error, it displays a macro error message and ends the macro.

An {ONERROR} command remains in effect until an error occurs (an {ONERROR} command can handle only one error), until 1-2-3 performs another {ONERROR} command, or until the macro ends. You can use looping to reset {ONERROR}.

If an error occurs while a macro is running, 1-2-3 normally displays an error message and ends the macro. If an {ONERROR} command is in effect when the error occurs, however, 1-2-3 branches to *branch-location* for further macro instructions instead of ending the macro. If you include the optional *message-location* argument, 1-2-3 records the error message in *message-location*.

{ONERROR} clears the subroutine stack: if the error occurs in a subroutine, 1-2-3 does not return to the macro that called the subroutine after completing the instructions at *branch-location* unless the instructions at *branch-location* specifically branch back to the calling macro.

Pressing CTRL-BREAK causes a 1-2-3 error. {ONERROR} will trap CTRL-BREAK unless a {BREAKOFF} command earlier in the macro disabled CTRL-BREAK.

If an error occurs, {ONERROR} does not prevent 1-2-3 from automatically issuing a {BORDERSOFF}, {FRAMEOFF}, {PANELOFF}, or {WINDOWSOFF} command.

Example

The following excerpt from a macro branches to a subroutine named CHANGEDIR if a file retrieve operation fails, and stores the error message in a cell named FILE_ERR.

```
...  
{ONERROR CHANGEDIR,FILE_ERR}  
...
```

{OPEN}

{OPEN *file-name,access-type*} makes a text file available for reading, writing, or both, depending on *access-type*. An open text file does not appear on the screen. It is open only in the sense that 1-2-3 can use it.

Arguments

file-name is the full name of a text file, including the extension (for example, JOBMEMO.PRN), or the address or name of a cell that contains a text file name. If the text file is not in the working directory, specify the path as part of *file-name* and enclose the argument in double quotation marks (for example, "C:\PERSONAL\JOBMEMO.PRN").

access-type is one of the four characters r, w, m, or a, or the address or name of a cell that contains one of those characters. If you are specifying a character for *access-type*, enclose the character in quotation marks so it is not confused with a range name. The character specifies the type of access you have to the file once it is open.

| Character | Access type | Task |
|-----------|-------------|--|
| r | read | Opens an existing file with the byte pointer at the beginning. Access to the file is read-only: you can use {READ} and {READLN}, but not {WRITE} or {WRITELN}. |
| w | write | Opens a new file with read and write access: you can use {READ}, {READLN}, {WRITE}, and {WRITELN}. CAUTION Opening an existing file with write access erases the file's contents. Use modify or append to write to an existing file. |
| m | modify | Opens an existing file with the byte pointer at the beginning. You can use {READ}, {READLN}, {WRITE}, and {WRITELN}. |
| a | append | Opens an existing file with the byte pointer at the end. You can use {READ}, {READLN}, {WRITE}, and {WRITELN}. |

Uses

Use {OPEN} before you use any of the other file manipulation macro commands: {CLOSE}, {FILESIZE}, {GETPOS}, {READ}, {READLN}, {SETPOS}, {WRITE}, and {WRITELN}.

Use {OPEN} with write access to create a new text file.

Notes

A **text file** (sometimes called a print file or an ASCII file) is a file stored on disk.

Only one text file can be open at one time, so using an {OPEN} command when a text file is already open closes the open text file before it opens the specified text file. You do not need a {CLOSE} command between {OPEN} commands.

If {OPEN} succeeds, the macro continues in the cell below the {OPEN} command. If {OPEN} fails, the macro continues to the next macro instruction in the same cell as the {OPEN} command.

Follow {OPEN} with an error handling routine to test that the file was opened successfully.

Examples

The following macro opens a new text file named PASTDUE.PRN on drive C, enters the contents of the cell named OVERDUE as the first line of the file, and closes the file; then the macro ends. If the macro is unable to find PASTDUE.PRN on drive C, the macro branches to CONTINUE for further instructions.

```
{OPEN "C:\PASTDUE.PRN","w"}{BRANCH CONTINUE}
{WRITELN OVERDUE}
{CLOSE}
```

In the following macro, if the working directory contains a file named ACCOUNTS.PRN, 1-2-3 opens the file with read access, enters the first line of the file in the cell named NEW_BALANCE, closes the file, and ends the macro. If the working directory does not contain a file named ACCOUNTS.PRN, the macro branches to CONTINUE.

```
{OPEN ACCOUNTS.PRN,"r"}{BRANCH CONTINUE}
{READLN NEW_BALANCE}
{CLOSE}
```

{PANELOFF} and {PANELON}

{PANELOFF [clear]} freezes the status line and control panel until 1-2-3 encounters a {PANELON} command or the macro ends.

{PANELON} unfreezes and displays the status line and control panel.

Argument

clear clears the control panel and status line before freezing them.

Notes

{PANELOFF} suppresses control-panel activity that results only from keystroke instructions. Macro commands that cause changes in the control panel – {MENUBRANCH}, {MENUCALL}, {GETLABEL}, {GETNUMBER}, {WAIT}, and {INDICATE} – override a {PANELOFF} condition. (/XL, /XM, and /XN also override a {PANELOFF}.)

You can use {INDICATE} to display a mode indicator after a {PANELOFF} command.

Uses

Use {PANELOFF} in interactive macros to freeze the status line and control panel when activity in that area would confuse users.

You can also use {PANELOFF} to speed up a macro.

Example

The following macro freezes the status line and control panel so that you do not see the series of prompts and menus that normally appear while 1-2-3 is performing GOTO (F5) and /Range Erase. The macro then pauses for five seconds before unfreezing the status line and control panel.

```
{PANELOFF}
{GOTO}DATA_RANGE~
/reDATA_RANGE~
{WAIT @NOW+@TIME(0,0,5)}
{PANELON}
```


{PGDN} and {PGUP}

{PGDN [*number*]} and {PGUP [*number*]} are equivalent to pressing PG DN and PG UP.

Argument

number is an optional argument that tells 1-2-3 how many times to press PG DN or PG UP. *number* is a value, the address or name of a cell that contains a value, or a formula that returns a value. {PGUP} and {PGDN} without arguments are equivalent to {PGUP 1} and {PGDN 1}.

{PUT}

{PUT *location,column-offset,row-offset,entry*} enters a number or a label in a cell within *location*.

Arguments

location is the address or name of a range of any size, or a formula that returns the address or name of a cell or range that contains a cell where you want to enter data.

column-offset and *row-offset* are numbers that identify the column and row position of a cell within *location*. The offset number for the first column and row is 0, for the second column and row is 1, for the third column and row is 2, and so on. (For example, 2,4 specifies the third column and fifth row of *location*.)

entry is a number, text, a formula, or the address or name of a cell that contains a number, label, or formula. If *entry* is a text formula that begins with double quotation marks, precede it with a + (plus).

Uses

Use {PUT} to change entries in a database when you know the relative position in the database, but not the specific cell address.

Notes

Using an offset number that specifies a column or row outside the specified range (for example, using 4 for *column-offset* when *location* includes only three columns) ends the macro with an error message.

Examples

The following examples refer to a range named COSTS (A1..D5) in the worksheet.

The command {PUT COSTS,3,2,45} places the number 45 in cell D3.

The command {PUT COSTS,2,0,MONTH} copies the contents of the cell named MONTH to cell C1. If MONTH contains a formula, it copies the current value of the formula to cell C1.

The command {PUT COSTS,0,8,500} results in an error. Range COSTS has only five rows, so a *row-offset* of 8 is invalid.

{QUERY}

{QUERY} is equivalent to pressing F7 (QUERY). For more information about 1-2-3 function keys, see the *Quick Reference*.

{QUIT}

{QUIT} ends a macro immediately, returning control to the user. 1-2-3 never performs instructions that follow a {QUIT} command.

Notes

When 1-2-3 performs a {QUIT} command, it stops executing the macro and returns control of the 1-2-3 work session to the user. If {QUIT} is included in a subroutine, the command ends the entire macro, not just the subroutine.

Examples

In the following macro, if the cell named YEAR contains the value 1990, the macro ends; otherwise, 1-2-3 branches to NEW.

```
{IF YEAR=1990}{QUIT}
{BRANCH NEW}
```

In the following macro, if the value in the cell named YEAR is less than 1990, 1-2-3 branches to OLD; otherwise, the macro ends.

```
{IF YEAR<1990}{BRANCH OLD}
{QUIT}
```

{READ}

{READ *byte-count,location*} copies the number of characters specified in *byte-count* from an open text file to *location*.

Arguments

byte-count is a value, the address or name of a cell that contains a value, or a formula that returns a value from 0 through 240.

location is the address or name of a cell or range, or a formula that returns the address or name of a cell or range. If you specify a range, 1-2-3 enters the data in the first cell of the range.

Notes

Before you use {READ}, use {OPEN} to specify the text file that you want to use. If no text file is open, 1-2-3 ignores {READ} and the macro continues to the next instruction in the same cell. If {READ} succeeds, the macro copies the data to *location* and continues in the cell below the {READ} command.

After each {READ} command, the byte pointer advances by the number specified in *byte-count*, so that a subsequent {READ} command begins reading at the next character in the file.

{READ} copies the carriage-return and line-feed characters at the end of text lines. If you don't want to copy the carriage-return and line-feed characters, use {READLN}.

To see the results of a {READ} command immediately, use a ~ (tilde) or a command that redraws the screen (for example, {DOWN}).

Example

The byte pointer is at the first character (offset 0) of the text:

Total Sales for the Year Ending 1988

The following {READ} command copies the word Total and the space that follows it to the cell named CHARS. The byte pointer moves forward six characters, to the beginning of the word Sales.

```
...  
{READ 6,CHARS}  
...
```

{READLN}

{READLN *location*} copies a line from an open text file and stores the characters in *location*.

Argument

location is a single cell or a range. If you specify a range, 1-2-3 enters the data in the first cell of the range.

Uses

Use {READLN} to copy a line of text that has an undetermined length.

Notes

Before you use {READLN}, use {OPEN} to specify the text file that you want to use.

{READLN} copies a line of characters starting at the position of the byte pointer and ending before a carriage return. 1-2-3 moves the byte pointer to the character after

the carriage return in the text file, so that a subsequent {READLN} command begins there. 1-2-3 does not copy the carriage return (and line-feed characters) with the line of text.

If no text file is open, 1-2-3 ignores {READLN} and the macro continues with the next instruction in the same cell as the {READLN} command. If {READLN} succeeds, the macro copies the data to *location* and continues with the cell below the {READLN} command.

Follow {READLN} with an error handling routine to test that the line was read successfully.

Example

The byte pointer is at the beginning of the line that contains the word January. Each line ends with a carriage-return.

```
January
February
...
```

The first {READLN} command copies the word January to cell MONTH1. The next {READLN} command copies the word February to cell MONTH2. Any subsequent {READLN} commands store subsequent lines in the specified locations.

```
...
{READLN MONTH1}
{READLN MONTH2}
...
```

{RECALC} and {RECALCCOL}

{RECALC *location*,[*condition*],[*iterations*]} recalculates the values in *location*, proceeding row-by-row.

{RECALCCOL *location*,[*condition*],[*iterations*]} recalculates the values in *location*, proceeding column-by-column.

Arguments

location is the address or name of the cell or range to recalculate, or a formula that returns the address or name of the cell or range to recalculate.

condition is an optional argument that tells 1-2-3 to recalculate once and then repeat the recalculation until *condition* is true. *condition* is usually a logical expression or the address or name of a cell that contains a logical expression, but it can be any formula, number, or address or name of a cell. 1-2-3 evaluates any *condition* that does not equal 0 (zero) as true and any *condition* that does equal 0 (zero) as false.

If *condition* refers to a cell that contains a formula and the formula needs to be recalculated for the {RECALC} or {RECALCCOL} command to work correctly, be sure the cell is in *location*.

iterations is an optional argument that tells 1-2-3 to perform the specified number of recalculation passes. *iterations* can be a value, or the address or name of a cell that contains a value. If *iterations* is 0 (zero), 1-2-3 performs the recalculation once. If you specify the *iterations* argument, you must specify the *condition* argument.

Uses

Use {RECALC} to recalculate formulas located below the cells on which they depend. You can also use {RECALC} to recalculate formulas located to the right and in the same row of cells on which they depend.

Use {RECALCCOL} to recalculate formulas located to the right of the cells on which they depend. You can also use {RECALCCOL} to recalculate formulas located in the same column and below the cells on which they depend.

Use {RECALC} and {RECALCCOL} to recalculate values entered by macro commands without recalculating the entire worksheet.

Notes

When you include both optional arguments, 1-2-3 repeats the recalculation until *condition* is true or until it has performed the specified number of recalculation passes, whichever happens first.

When 1-2-3 uses {RECALC} or {RECALCCOL} to recalculate a range, it updates formulas only in the range. To ensure that all formulas are up to date at the end of a macro that uses {RECALC} or {RECALCCOL}, include a {CALC} command in the macro or press F9 (CALC) when the macro ends.

Examples

The first set of instructions in the following excerpt from a macro sets recalculation to Manual. Other macro instructions change the value in cell D4 that the formula in cell A9 uses. The {RECALC} command at the end tells 1-2-3 to recalculate the range named NEWPRICES, which includes cells D4 and A9. Recalculation proceeds row-by-row, so 1-2-3 recalculates D4 before A9, and the result is accurate.

```
/WGRM  
...  
{RECALC NEWPRICES}
```

The following example of {RECALCCOL} continuously recalculates the range named PAYMENT, column-by-column, until the value in the cell named VAL falls below 100 or the number of recalculations equals 50.

```
{RECALCCOL PAYMENT,VAL<100,50}
```

{RESTART}

{RESTART} cancels the return sequence of nested subroutines, ending the macro when the current subroutine ends.

Uses

Use {RESTART} to control whether a macro continues or ends after certain subroutine tasks are performed. For more information on subroutines, see the description of {*subroutine*} on page 158.

Use {RESTART} to cancel a {FORM} command from within a *call-table*.

Notes

1-2-3 tracks the order in which nested subroutines are called up to 32 subroutines. After a {RESTART}, 1-2-3 performs the remaining instructions in the current subroutine, but the macro ends when the subroutine ends instead of returning control to the calling macro. If the instructions that follow {RESTART} in the subroutine transfer macro control elsewhere, the macro continues.

Examples

The following excerpt from a subroutine combines {RESTART} with {IF} to cancel the return sequence and branch to NEXTPLAN if the cell named STATUS contains the label Not OK or a text formula that results in Not OK. If STATUS contains anything else or is blank, macro control returns to the calling macro after 1-2-3 completes the remainder of the subroutine.

```
...  
{IF STATUS="Not OK"}{RESTART}{BRANCH NEXTPLAN}  
...
```

In the following example, the {FOR} command calls the subroutine BALANCE. The subroutine stores the number you enter in cell PURCHASE and enters the new balance in cell BAL. If the new balance is 0 or less, {RESTART} cancels the return sequence and branches to BROKE; otherwise, 1-2-3 repeats the subroutine BALANCE ten times, as specified in the {FOR} command. When the for-next loop is complete, 1-2-3 returns to the instruction that follows the {FOR} command in the calling macro.

```
...  
{FOR Count,1,10,1,BALANCE}  
...  
BALANCE {GETNUMBER "Cost of purchase? ",PURCHASE}  
{LET BAL,BAL-PURCHASE}  
{IF BAL<=0}{RESTART}{BRANCH BROKE}
```

{RETURN}

{RETURN} returns macro control from a subroutine to the calling macro.

Uses

Use {RETURN} in a subroutine that was called by a *{subroutine}* or {MENUCALL} command.

Use {RETURN} as the then clause of an {IF} statement in a subroutine to return directly to the calling macro without performing the else clause.

Notes

When it encounters {RETURN}, 1-2-3 returns control to the macro or subroutine that called the subroutine and performs the instructions that follow the command that called the subroutine. If {RETURN} is in the original calling macro, it performs as {QUIT}.

In a subroutine called by a {FOR} command, {RETURN} ends the current iteration of the subroutine and immediately starts the next iteration.

If the subroutine ends with a blank cell, {RETURN} is unnecessary. Macro control automatically returns to the calling macro.

Example

In the SAVE subroutine that follows, {GETLABEL} prompts you to type a response. If you type N or n, the {RETURN} command forces the subroutine to return to the calling macro before the subroutine ends. If you type Y or y, 1-2-3 saves the current version of the file and then returns to the calling macro. If you type any other character, 1-2-3 repeats the SAVE subroutine from the beginning.

```
SAVE      {GETLABEL "Save file? (Y/N)",INPUT}~
          {IF @UPPER(INPUT)="N"}{RETURN}
          {IF @UPPER(INPUT)="Y"}/fs~r{RETURN}
          {BRANCH SAVE}
```

{RIGHT} and {R}

{RIGHT [*number*]} and {R [*number*]} are equivalent to pressing →.

Argument

number is an optional argument that tells 1-2-3 how many times to press →. *number* is a value, the address or name of a cell that contains a value, or a formula that returns a value. {RIGHT} without an argument is equivalent to {RIGHT 1}.

{SETPOS}

{SETPOS *offset-number*} moves the byte pointer to the *offset-number* position in an open text file.

Argument

offset-number is a value, the address or name of a cell that contains a value, or a formula that returns a value. It specifies the character position in the file to which you want to move the byte pointer, relative to the first byte in the file. The first character in the file is at position 0, the second at position 1, and so on.

Notes

A text file (sometimes called a print file or an ASCII file) is a file stored on disk.

If no text file is open, 1-2-3 ignores {SETPOS} and the macro continues with the next instruction in the same cell as the {SETPOS} command. If a file is open, 1-2-3 moves the byte pointer to the *offset-number* position in the text file, and the macro continues with the instruction in the cell below the {SETPOS} command.

1-2-3 does not prevent you from placing the byte pointer past the end of the file. Before using {SETPOS}, use {FILESIZE} to determine the last character position in the file.

Examples

Suppose that the byte pointer is at the beginning of a text file that consists of 250 bytes and begins with the following text:

This report contains information based on last year's performance

The command {SETPOS 10} moves the byte pointer to the letter t at the end of the word report.

The following command, acting on the same text file, enters the value 250 in the cell named BYTES and then moves the byte pointer to the position after the last character in the file. If no text file is open, the macro branches to location NEXT.

```
{FILESIZE BYTES}  
{SETPOS BYTES}{BRANCH NEXT}
```

{*subroutine*}

{*subroutine* [*arg1*],[*arg2*],...[*argn*]} calls a subroutine.

Arguments

subroutine is the range name (or address) of the subroutine that you want the macro to call. The range name (or address) can refer to the first cell of the subroutine or to the entire subroutine. It's safest to use a range name rather than a cell address for the

subroutine in case you insert or delete rows in the worksheet or move the subroutine. If you specify a range for *subroutine*, 1-2-3 begins the subroutine in the upper left corner of the range.

arg1 to *argn* are optional arguments. You can include up to 31 optional arguments. Arguments can be values or text, including formulas and the names or addresses of cells. *{subroutine}* passes the arguments to the subroutine, which must begin with a {DEFINE} command if arguments are specified. {DEFINE} evaluates and stores the optional arguments in worksheet cells. For more information, see {DEFINE} on page 122.

Uses

Use subroutines to divide long macros into smaller, more specific tasks. If these tasks are shared by several macros, using a subroutine means you only have to write the shared task once. You can call it from all macros that use it. 1-2-3 can transfer control from one subroutine to another to perform many different tasks during a macro.

Notes

To call a subroutine, use a *{subroutine}* command in the calling macro where you want 1-2-3 to begin performing the subroutine. Specify the address or name of the subroutine as *{subroutine}*. 1-2-3 temporarily passes control from the calling macro to the subroutine when it encounters a *{subroutine}* command. After it performs the subroutine, it returns to the calling macro and performs the instructions that follow the *{subroutine}* command. A subroutine ends when 1-2-3 performs a {RETURN} command or encounters a blank cell. A {QUIT} command in the subroutine stops the macro.

One subroutine can call another subroutine. This is known as **nesting**. Using nested subroutines lets you create large macro applications that are clearly structured, accessible, and easy to revise. Nesting is limited to 32 subroutines.

When 1-2-3 encounters a *{subroutine}* command in a subroutine, it immediately starts performing the new subroutine. When the second subroutine ends, 1-2-3 returns to the subroutine that called it and finishes performing that subroutine, and then returns control to the original calling macro. If there is more than one nested subroutine, 1-2-3 returns to each previous subroutine until it finally returns to the original calling macro.

If you do not want 1-2-3 to return from a particular subroutine, use {RESTART} in that subroutine. {RESTART} cancels the return sequence that 1-2-3 tracks as it completes nested subroutines. When 1-2-3 completes the subroutine that contains {RESTART}, the macro ends. For more information, see {RESTART} on page 156.

Examples

The following two macros perform the same task, but the second one uses arguments to pass range names to a subroutine. The macros format four ranges as Currency with two decimal places. The {RETURN} command at the end of the CURRSUB subroutine is optional as long as the subroutine ends with a blank cell.

The DATAPREP macro calls the subroutine FMT four times. Each time, it formats the current cell. After each subroutine call, 1-2-3 returns to the next {GOTO} command in DATAPREP.

```
DATAPREP {GOTO}QUAR1~
          {FMT}
          {GOTO}QUAR2~
          {FMT}
          {GOTO}QUAR3~
          {FMT}
          {GOTO}QUAR4~
          {FMT}

FMT      /RFC2~~
          {RETURN}
```

In the following macro, DATAPREP specifies the same four ranges as arguments passed to FMT. FMT stores the argument in range ONE and uses a nested subroutine call to format each range.

```
DATAPREP {FMT "QUAR1"}
          {FMT "QUAR2"}
          {FMT "QUAR3"}
          {FMT "QUAR4"}

FMT      {DEFINE ONE}
          {GOTO}{ONE}~
          /RFC2~~
          {RETURN}

ONE
```

{SYSTEM}

{SYSTEM *command*} temporarily suspends the 1-2-3 session and performs the specified DOS command.

Argument

command is any DOS command, including batch commands, up to 128 characters.

Notes

After DOS performs the specified command, the 1-2-3 session automatically resumes and the macro continues.

{SYSTEM *command*} is similar to the /System command, except that control returns from DOS automatically after *command* is completed. If you want to temporarily suspend the 1-2-3 session without specifying an operating system command, use the System command (/S) in the macro. Type exit to return to 1-2-3 and continue the macro.

If you use 1-2-3 with DOS 3.0 or higher, you can precede *command* with an explicit path, such as {SYSTEM "c:\bats\reports"}. If you use an earlier version of DOS, you can include the directory that contains the program(s) you want to run in the PATH statement in your AUTOEXEC.BAT file.

Do not use the {SYSTEM} command to load memory-resident programs such as terminate-and-stay-resident programs (for example, the DOS PRINT program). If you do so, you may not be able to resume 1-2-3.

Examples

The following command suspends the 1-2-3 session, runs a batch file called COPYFILE, and returns to 1-2-3. The macro continues after the batch file is finished.

```
...  
{SYSTEM "COPYFILE"}  
...
```

The following command suspends the 1-2-3 session and performs the operating system command entered in cell SYS_CMD. For example, if SYS_CMD contains the entry C:\DOS\CHKDSK, {SYSTEM} performs a CHKDSK command (assuming the CHKDSK command file is in the DOS directory on drive C) and then returns to 1-2-3 and continues the macro.

```
...  
{SYSTEM SYS_CMD}  
...
```

{TABLE}

{TABLE} is equivalent to pressing F8 (TABLE). For more information about 1-2-3 function keys, see the *Quick Reference*.

{UP} and {U}

{UP [number]} and {U [number]} are equivalent to pressing ↑.

Argument

number is an optional argument that tells 1-2-3 how many times to press ↑. *number* is a value, the address or name of a cell that contains a value, or a formula that returns a value. {UP} without an argument is equivalent to {UP 1}.

{WAIT}

{WAIT *time-number*} suspends a macro and displays the WAIT mode indicator until the time specified by *time-number*. When the specified time arrives, 1-2-3 removes the WAIT indicator and continues the macro.

Argument

time-number is a value, the address or name of a cell that contains a value, or a formula that returns a value. The value must represent a future time as a 1-2-3 date and time number. If the value represents a nonexistent time or a time that has already passed, 1-2-3 ignores the {WAIT} command and continues to the next macro instruction in the same cell. You can use the @functions @NOW, @TIME, and @TIMEVALUE to specify *time-number*.

Uses

Use {WAIT} to let you read what is on the screen before moving to a new location.

Notes

1-2-3 uses the date and time settings on your PC to keep track of time. Be sure these settings are correct before you use {WAIT}.

To interrupt a {WAIT} command, press **CTRL-BREAK** unless a {BREAKOFF} command is in effect. 1-2-3 ignores all other keystrokes while a {WAIT} command is in effect.

Examples

The READCOL macro that follows displays the message “Press CTRL-BREAK to stop” in the Status indicator area, moves the cell pointer down one row, and pauses 5 seconds. It repeats this process until you press **CTRL-BREAK**. This macro is useful to examine a long column of entries, or to scroll through a long document while you are reading it.

```
READCOL  {INDICATE "Press CTRL-BREAK to stop"}
PAUSE    {DOWN}
          {WAIT @NOW+@TIME(0,0,5)}
          {BRANCH PAUSE}
```

The following macro suspends the running of a macro for the amount of time specified by @TIMEVALUE(0.012), about 17.5 minutes.

```
{WAIT @NOW+@TIMEVALUE(0.012)}
```

{WINDOW}

{WINDOW} is equivalent to pressing F6 (WINDOW). For more information about 1-2-3 function keys, see the *Quick Reference*.

Uses

Use {WINDOW} to display dialog boxes during a macro. 1-2-3 suspends the macro when a dialog box appears, if you use {EDIT} to switch to SETTINGS mode. When you select OK or press ENTER, 1-2-3 uses the values you specified in the dialog box and continues to run the macro.

{WINDOWSOFF} and {WINDOWSON}

{WINDOWSOFF} stops screen updates while a macro is running.

{WINDOWSON} cancels {WINDOWSOFF} and resumes normal worksheet display.

Uses

Use {WINDOWSOFF} when you do not want macro activity (changes flashing on the screen), to appear. {WINDOWSOFF} also speeds up the macro because 1-2-3 does not update the screen display.

Notes

{WINDOWSOFF} remains in effect until the macro ends or until 1-2-3 performs a {WINDOWSON} command.

Example

The following macro uses {WINDOWSOFF} before formatting several ranges in the worksheet, then uses {WINDOWSON} when formatting is complete. The macro continues, resuming screen updating.

```
{WINDOWSOFF}
{CURRENCY FIRST_RANGE}
{PERCENT SECOND_RANGE}
{DATE THIRD_RANGE}
{WINDOWSON}
```

...

{WRITE}

{WRITE *string*} copies *string* to the current byte pointer position in the open text file.

Argument

string is text, a text formula, or the address or name of a cell that contains a label or a text formula.

Notes

A text file (sometimes called a print file or an ASCII file) is a file stored on disk.

When 1-2-3 performs a {WRITE} command, it evaluates *string* and copies the resulting text to the open text file, starting at the current byte pointer position. The byte pointer advances to the position after the last character written. If the byte pointer is at the end of the file, 1-2-3 expands the length of the file to accommodate the incoming string. A subsequent {WRITE} or {WRITELN} command begins writing at the new byte pointer position unless you reset the pointer with {SETPOS}. If the byte pointer is in the middle of the file, the incoming string replaces existing data.

{WRITE} works only if the text file was opened with write, append, or modify access (see {OPEN} on page 148). If no text file is open, or if the file was opened with read access, 1-2-3 ignores {WRITE} and the macro continues with the next instruction in the same cell as the {WRITE} command. If {WRITE} succeeds, the macro copies the string to the text file and continues with the instruction in the cell below the {WRITE} command.

Examples

The following macro writes this character string to the open text file.

Ben's Dairy

If no text file is open, or if the file was opened with read-only access, the macro branches to FAIL. Otherwise, the macro branches to REPORT.

```
...
{WRITE "Ben's Dairy"}{BRANCH FAIL}
{BRANCH REPORT}
...
```

The following {WRITE} command writes the label from the cell named FULLNAME to the open text file (assuming the file was opened with write, append, or modify access). If FULLNAME contains a text formula such as +*"FIRST"&"•"&"LAST"*, 1-2-3 evaluates the formula and writes the resulting text in the file. (The bullet represents one space.) If FULLNAME contains a value, 1-2-3 ends the macro with an error message. If the range name FULLNAME does not exist, 1-2-3 writes FULLNAME in the text file.

```
{WRITE FULLNAME}
```

{WRITELN}

{WRITELN *string*} writes *string* at the byte pointer position in the open text file, adding a carriage return and line feed.

Argument

string is text, a text formula, or the address or name of a cell that contains a label or a text formula. If you use an empty string ("") as *string*, 1-2-3 writes a carriage return and line feed.

Notes

A text file (sometimes called a print file or an ASCII file) is a file stored on disk.

When 1-2-3 performs a {WRITELN} command, it evaluates *string* and copies the resulting text to the current byte pointer position in the open text file, followed by a carriage return and a line feed. The byte pointer advances to the position after the last character written. If the byte pointer is at the end of the file, 1-2-3 extends the file to accommodate the incoming string. A subsequent {WRITE} or {WRITELN} command begins writing at the new byte pointer position unless you reset the pointer with {SETPOS}. If the byte pointer is not at the end of the file, the incoming string replaces existing data.

{WRITELN} works only if the text file was opened with write, append, or modify access (see {OPEN} on page 148). If no text file is open, or if the open file was opened with read-only access, 1-2-3 ignores {WRITELN} and the macro continues with the next instruction in the same cell as the {WRITELN} command. If {WRITELN} succeeds, the macro copies the string to the text file and continues with the instruction in the cell below the {WRITELN} command.

Example

The following macro writes a line to the open text file, adds a carriage return and line feed to start a new line, and then writes four more lines that end with a carriage return and line feed. If no text file is open, or if the text file was opened for read-only access, the macro branches to FAIL.

```
{WRITE "Musical Instruments in My Band"}{BRANCH FAIL}
{WRITELN ""}
{WRITELN "Keyboard"}
{WRITELN "Saxophone"}
{WRITELN "Drums"}
{WRITELN "Guitars"}
```

The /X Macro Commands

1-2-3 Release 2.3 keeps the /X commands (originally used in 1-2-3 Release 1A) for compatibility. Each /X command has a corresponding macro command.

| /X command | Function | Macro command |
|--|--|-----------------------|
| <i>/XC</i> location~ | Calls the subroutine at <i>location</i> . | { <i>subroutine</i> } |
| <i>/XG</i> location~ | Branches to <i>location</i> . | {BRANCH} |
| <i>/Xl</i> condition~... | If <i>condition</i> is true, performs the next instruction in the same cell. Otherwise, skips to the next cell for further instructions. | {IF} |
| <i>/XL</i> prompt~[<i>location</i>]~ | Displays <i>prompt</i> in the control panel. Enters your response as a label in <i>location</i> . | {GETLABEL} |
| <i>/XM</i> location~ | Activates the macro menu at <i>location</i> . | {MENUBRANCH} |
| <i>/XN</i> prompt~[<i>location</i>]~ | Displays <i>prompt</i> in the control panel. Enters your response as a number in <i>location</i> . | {GETNUMBER} |
| <i>/XQ</i> | Ends the macro. | {QUIT} |
| <i>/XR</i> | Returns control from the current subroutine to the main macro, or ends the current loop through the subroutine and starts the next loop. | {RETURN} |

Arguments

condition is a logical expression or the address or name of a cell that contains a logical expression. (A logical expression uses one of the logical operators = <> < > <= >= #AND# #NOT# or #OR#.) You can use any formula, number, text, cell address, or cell name in the logical expression as *condition*.

location is a range name or address, or a formula that results in a range name or address.

prompt is any text that fits in the first line of the control panel, the address or name of a cell that contains the text, or a text formula.

Notes

Preface /X commands with {BREAK} to ensure that the macro starts in READY mode. Do not start a /X command between steps of another 1-2-3 command.

Chapter 5

Sample Macros

This chapter presents several macros. The macros are short and simple, and they provide concrete examples of how you can use macros in your everyday work. They also illustrate programming techniques that you can apply to any macro you create.

| Macro name | Purpose |
|------------|--|
| \G | Moves the cell pointer to a different range. |
| \S | Shifts the current row to the top of the screen. |
| \D | Enters today's date in the current cell. |
| \R | Rounds a column of values to the number of decimal places you specify. |
| \C | Creates a macro menu of column-related commands. |
| \M | Creates and prints mailing labels using records in a database. |

Using the Sample Macros

The sample macros are in the file called SAMPMACS.WK1. The Install program transferred the SAMPMACS.WK1 file to your 1-2-3 program directory.

To Use a Sample Macro

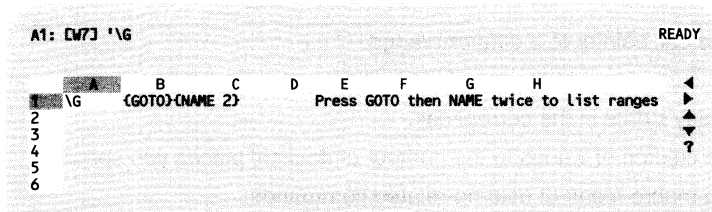
1. Start 1-2-3.
2. If necessary, use /File Directory to make the 1-2-3 program directory the current directory.
3. Use /File Retrieve to retrieve the SAMPMACS.WK1 worksheet.
4. Move the cell pointer to the macro you want to run.
5. Start the macro. To do this, hold down ALT while pressing the letter of the macro's name. For example, to start macro \G, press ALT-G.

You can also use ALT-F3 (RUN) to list the names of all macros in the worksheet. To start a macro this way, select a macro name from the menu and press ENTER.

6. (Optional) Copy a macro to your worksheet using /File Combine Copy.
Make sure you reassign range names to the macro (and its subroutines) that you copy; 1-2-3 does not preserve the range names in the copy.
7. (Optional) Save the macros in a macro library to use with any worksheet (see "Saving Macros in a Library" on page 183).

Goto Macro (\G)

In a large worksheet that has different data areas (such as an income statement with a balance sheet and various cost analyses), using pointer-movement keys to move from one area to another can be time-consuming. If you assign range names to the different areas, you can use macro \G to move the cell pointer more quickly from area to area.



To Use Macro \G

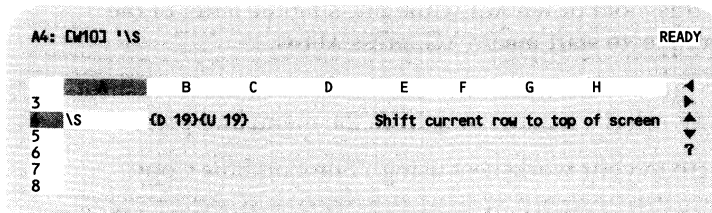
1. Start macro \G.
2. Highlight the name of the range you want to move to and press ENTER.

Explanation of Macro \G

{GOTO}{NAME 2} presses F5 (GOTO) and then F3 (NAME) twice to display a full-screen list of range names in the worksheet.

Row-Shifting Macro (\S)

Macro \S moves the row that contains the cell pointer to the top of the screen.



To Use Macro \S

1. Move the cell pointer to the row you want to move to the top of the screen.
2. Start macro \S.
3. (Optional) Modify the \S macro to use with your worksheet.

If your screen displays more than 20 rows, specify the number of rows your screen displays minus 1. For example, if your screen displays 38 rows, the macro should read {D 37}{U 37}.

Explanation of Macro \S

{D 19}{U 19} moves the cell pointer down 19 rows and then back up 19 rows, shifting to the top of the screen the row the cell pointer was in when you started the macro. (The macro instructions {D} and {U} are equivalent to {DOWN} and {UP}.)

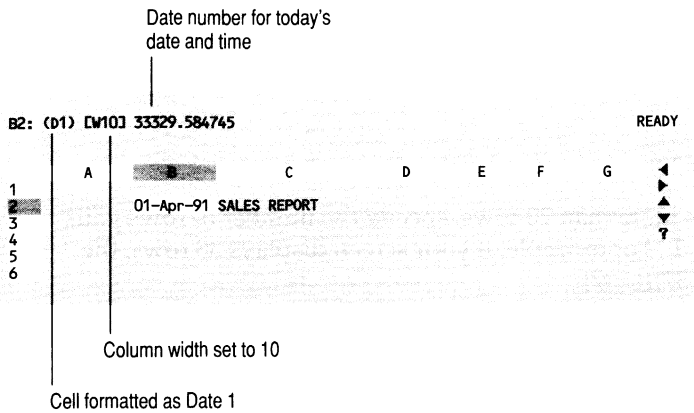
Date Macro (\D)

Macro \D enters today's date in the current cell by converting the result of @NOW to a value. Use the macro to document the data in a worksheet, insert the date in a memo, or create an entry for a print header or footer.

```
A7: [M1] \D                                READY
6      * * * * * B C D E F G H
7      \D      (PANELOFF)(WINDOWSOFF)  Freeze screen
8      /rfd1   Format current cell as Date 1
9      @NOW(CALC)~ Type @NOW, convert formula to value
10     (WINDOWSOND)(PANELOND)  Unfreeze screen
11
```

To Use Macro \D

1. Move to a blank cell.
2. Start macro \D.
3. If 1-2-3 displays asterisks, use /Worksheet Global Column-Width, /Worksheet Column Set-Width, or the column macro described on page 172 to widen the column to 10. The asterisks will disappear and 1-2-3 will display the current date. (You can also modify this macro to check the current column width with @CELLPOINTER and, if necessary, widen the column. For example, use {IF @CELLPOINTER("width")<10}/wcs10~.)

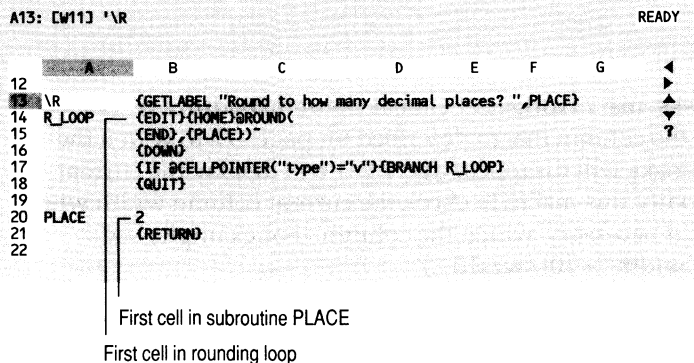


Explanation of Macro \D

- {PANELOFF}{WINDOWSOFF} freezes the control panel and worksheet area, preventing flashing on the screen during the macro and allowing the macro to work at maximum speed.
- /rfd1~ formats the current cell as Date 1 format.
- @NOW{CALC}~ types @NOW, converts @NOW to its current value, and enters the value in the current cell.
- {WINDOWSON}{PANELON} unfreezes the control panel and worksheet area prior to ending the macro.

Rounding Macro (\R)

Macro \R rounds a column of values by converting the values to @ROUND formulas, using the number of decimal places you specify.



To Use Macro \R

1. Move the cell pointer to the first cell in the column of values you are rounding.
2. Start macro \R.

B1: (G) [M10] 24.759

| | A | C | D |
|---|----------|---|---|
| 1 | | | |
| 2 | 24.759 | | |
| 3 | 21.888 | | |
| 4 | 35.23 | | |
| 5 | 890.111 | | |
| 6 | 1.7 | | |
| | 11456.45 | | |

Unrounded values

B6: [M10] @ROUND(11456.45,1)

| | A | C | D |
|---|---------|---|---|
| 1 | | | |
| 2 | 24.8 | | |
| 3 | 21.9 | | |
| 4 | 35.2 | | |
| 5 | 890.1 | | |
| 6 | 1.7 | | |
| | 11456.5 | | |

Values rounded to one decimal place

Explanation of Macro \R

{GETLABEL "Round to how many decimal places? ",PLACE} prompts you for the number of decimal places to round the values, waits for you to type a number and press ENTER, and then enters that number as a label in cell PLACE for use later in the macro. You must use {GETLABEL} because macro instructions are labels so anything you want the macro to use must also be a label.

The rounding loop R_LOOP converts the value in the current cell to an @ROUND formula. For example, it converts 24.759 to @ROUND(24.759,1), as follows:

- {EDIT}{HOME}@ROUND(presses F2 (EDIT) to begin editing the current cell's contents, moves the cursor to the beginning of the entry, and inserts @ROUND(in front of the entry.
- {END}, moves the cursor to the end of the entry and types a comma.
- {PLACE} calls subroutine PLACE (located in cells B20..B21). PLACE uses the number you specified in the {GETLABEL} command for the *decimal-place* argument in the @ROUND formula. {RETURN} ends the subroutine, so macro control shifts back to the main macro where)~ completes and enters the @ROUND formula.
- {DOWN} moves the cell pointer down one cell.
- {IF @CELLPOINTER("type")="v"} tests the new cell's contents. If the cell contains a value, the macro continues to the next instruction, {BRANCH R_LOOP}, which branches macro control to cell R_LOOP (the second cell in the macro). The macro then repeats the value-to-@ROUND-formula conversion. If the cell is blank or contains a label, the macro skips to the next line, where {QUIT} ends the macro.

Column Macro (\C)

1-2-3 has several commands for adjusting columns; remembering which command to select for which type of column change can be confusing. You can use macro \C to change the width of individual columns, change the width of columns globally, reset columns to the default column width, hide and redisplay columns, and/or insert and delete columns, all from one macro menu.

```

Main menu
A24: [W9] '\C
READY

  A   B   C   D   E   F   G   H   I
24  \C  {MNUCALL COL_MENU} Display column-macro menu
25      {BRANCH \C} Restart column macro
26
27  COL_MENU Global Set Reset Hide Unhide AddCol DelCol Quit
28      Set gloSet Reset a Hide a RedisplaInsert adelete aEnd macro
29      {GLOBAL{SET{RESET} {HIDE} {UNHIDE}{ADDCOL}{DELCOL}{QUIT}}
30
31  GLOBAL /wgc{?}" Set global column width to user-specified width
32
33  SET /wccs{?}"{?}Set column width for user-specified range
34
35  RESET /wccr{?}" Reset column width for user-specified range
36
37  HIDE /wch{?}" Hide user-specified range of columns
38
39  UNHIDE /wcd{?}" Redisplay user-specified range of columns
40
41  ADDCOL /wic{?}" Insert user-specified range of columns
42
43  DELCOL /wdc{?}" Delete user-specified range of columns
  
```

Macro menu

Subroutines for menu items

To Use Macro \C

1. Start macro \C.

1-2-3 displays a macro menu.

```

A1: [W9] '\G
MENU
Global Set Reset Hide Unhide AddCol DelCol Quit
Set global column width for worksheet
  
```

2. Select an item from the menu.

After 1-2-3 completes the selected subroutine, it returns to the instruction following the {MNUCALL} command; in this case, {BRANCH \C}, which redisplay the column macro menu.

3. To end macro \C, select Quit from the macro menu.

Explanation of Macro \C

{MENUCALL COL_MENU} displays the macro menu stored in range COL_MENU. The range has three rows:

- The first row contains the items that will appear in the second line of the control panel: Global, Set, Reset, Hide, Unhide, AddCol, DelCol, and Quit. Each item is in its own column.
- The second row contains the descriptions that 1-2-3 displays in the third line of the control panel as you highlight each menu item. For example, "Set global column width for current worksheet" is the description for the Global menu item.
- The third row contains the macro instructions 1-2-3 performs after you select a menu item. Depending on the menu item you select, the third row either calls a subroutine (GLOBAL, SET, RESET, HIDE, UNHIDE, ADDCOL, or DELCOL) or ends the macro (QUIT).

| | |
|--------|---|
| GLOBAL | /wgc selects /Worksheet Global Column-Width. {?} suspends the macro to let you specify a column width, and ~ completes the command. |
| SET | /wccs selects /Worksheet Column Column-Range Set-Width. {?}~{?}~ suspends the macro twice to let you specify a range of columns and a column width (the tildes enter your specifications and complete the command). |
| RESET | /wccr selects /Worksheet Column Column-Range Reset-Width. {?} suspends the macro to let you specify the range of columns whose width you want to reset, and ~ completes the command. |
| HIDE | /wch selects /Worksheet Column Hide. {?} suspends the macro to let you specify the range of columns to hide, and ~ completes the command. |
| UNHIDE | /wcd selects /Worksheet Column Display. {?} suspends the macro to let you specify the range of columns to redisplay, and ~ completes the command. |
| ADDCOL | /wic selects /Worksheet Insert Column. {?} suspends the macro to let you specify the range of columns to insert, and ~ completes the command. |
| DELCOL | /wdc selects /Worksheet Delete Column. {?} suspends the macro to let you specify the range of columns to delete, and ~ completes the command. |

NOTE If you run the ADDCOL or DELCOL subroutine in the SAMPMACS.WK1 worksheet, specify columns other than A through I.

Mailing Labels Macro (\M)

Macro \M creates and prints mailing labels.

```

A47: EW7J \M                                READY
      A      B      C      D      E      F      G      H      I
47 \M /pprLABEL"ouuqaq                      Set LABEL as print range
48 P_LOOP (IF @CELLPOINTER("type")="b")<QUIT> Check current cell
49 /c(RIGHT 5)"LAST"                          Copy current record
50 (RECALC LABEL)                             Recalculate mailing label str
51 /ppgq(DOWN)                                Print mailing label
52 (BRANCH P_LOOP)                            Restart print loop
  
```

To Use Macro \M

1. Enter the names and addresses in a database.

```

A56: 'Compton                                READY
      A      B      C      D      E      F
55 LAST FIRST STREET CITY STATE ZIP
56 Compton Julia 90 Stratford Drive Englishtown NJ 07726
57 O'Brien Conor 1055 W. 7th Street Los Angeles CA 90017
58 Miller Alana 1 Camelback Road Phoenix AZ 85012
59 Groden Cara 938 Lafayette Street New Orleans LA 70113
60 Yarrow Maurice 1990 Post Oak Blvd. Houston TX 77056
  
```

A sample database

2. Set up the cells to which the text formulas refer. To do this, copy the field names from the name-and-address database to another area of the worksheet. For example, in the following illustration, the field names in A55..F55 have been copied to A63..F63.

```

A55: 'LAST                                    READY
      A      B      C      D      E      F
55 LAST FIRST STREET CITY STATE ZIP
56 Compton Julia 90 Stratford Drive Englishtown NJ 07726
57 O'Brien Conor 1055 W. 7th Street Los Angeles CA 90017
58 Miller Alana 1 Camelback Road Phoenix AZ 85012
59 Groden Cara 938 Lafayette Street New Orleans LA 70113
60 Yarrow Maurice 1990 Post Oak Blvd. Houston TX 77056
61
62
63 LAST FIRST STREET CITY STATE ZIP
64
65
  
```

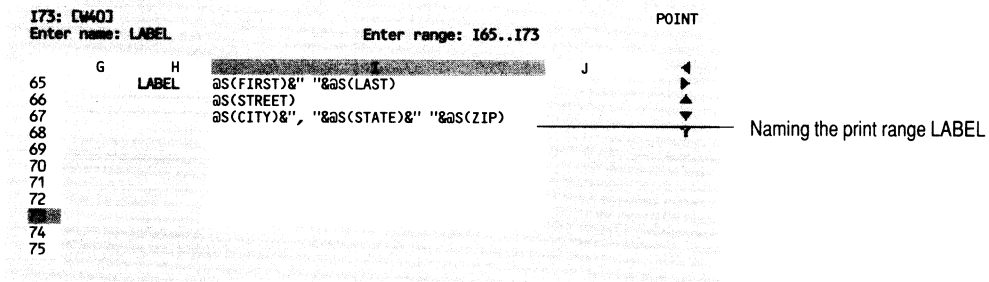
Field names copied from database to another area

/RNLD used to assign names to cells below

3. Use /Range Name Labels Down to assign each field name as the range name for the cell below it (in the illustration above, to A64..F64).

4. Choose a column for the print range.
5. Use /Worksheet Column Set-Width to set the column width of the print range column to 40 (or any setting that will be at least one character wider than the longest mailing label). In this example, column I is the print range column.
6. Create your mailing label in the worksheet. To do this, enter the following formulas in three consecutive cells in the print range column. (Each • (bullet) represents a space. The @S function prevents blank cells in a database record from causing formulas to display and print ERR.)


```
@S(FIRST)&"•"•&@S(LAST)
@S(STREET)
@S(CITY)&"•"•&@S(STATE)&"•"•&@S(ZIP)
```
7. Select /Range Name Create.
8. Enter LABEL as the name for the range; and as the range to name, specify a one-column range that includes the three formulas and as many additional rows as are needed for the printer to skip from one mailing label to the next. In this example, range LABEL, I65..I73, includes six extra rows.



NOTE Cells that contain the formulas are formatted as Text so you can see them; leave the formulas as they are in your worksheet.

9. Move the cell pointer to the leftmost cell in the first record (for example, cell A56).
10. Start macro \M.

Explanation of Macro \M

- /pprLABEL~ selects /Print Printer Range and enters LABEL as the print range.
- oouqaq selects Options Other Unformatted for no top-and-bottom margins or page breaks, leaves the /Print Printer Options menu, selects Align to align the paper in the printer, and leaves the /Print menu.
- {IF @CELLPOINTER("type")="b"} checks to see whether the current cell is blank (a blank cell signals the end of the database records).

- If the cell is blank, the macro ends with {QUIT}. Otherwise the macro continues to /c{RIGHT 5}~LAST~, which copies the current record to cells LAST, FIRST, STREET, CITY, STATE, and ZIP, the cells that the mailing-label text formulas use.
- {RECALC LABEL} recalculates the mailing-label text formulas using the new information in cells LAST, FIRST, STREET, CITY, STATE, and ZIP.
- /ppgq{DOWN}{BRANCH P_LOOP} selects /Print Printer Go Quit to print the current mailing label and then leave the /Print menu, moves the cell pointer down one row, and branches back to cell P_LOOP to create the next mailing label.

Chapter 6

Using the Macro Library Manager Add-In

This chapter describes the Macro Library Manager add-in. It provides steps for using Macro Library Manager, and for creating, updating, and working with macro libraries.

Macro Library Manager lets you create macro libraries and work with them. As a 1-2-3 **add-in** program, Macro Library Manager runs in 1-2-3, thus increasing the power and versatility of 1-2-3. Because Macro Library Manager is an add-in, it does not have to be stored in your computer's memory all the time as does 1-2-3. Using the Add-In commands, you can attach Macro Library Manager when you need to use a macro library and detach it when you need more memory for completing other tasks.

Macro Library Manager has its own commands that you can use to create and edit macro libraries. Other commands let you load libraries into memory, remove them from memory, or list the range names in a library.

What Is a Macro Library?

A **macro library** makes a set of macros available for use in any worksheet. To store macros in a library, you specify the range that contains the macros from a 1-2-3 worksheet. Macro Library Manager stores the range in memory (in an area that is separate from the worksheet) and saves it in a file on disk with a .MLB extension (called a **library file**).

A macro library can make your work easier and more efficient in the following ways:

- Macros are independent of worksheets that store applications, so you can use library macros and have your worksheets free for other data.
- Macros are stored in one location so you can update them centrally, instead of updating individual worksheets.
- Tasks you perform routinely with more than one worksheet are independent of all worksheets. For example, a macro that performs the same financial analysis on data in several worksheets or a graph macro that graphs sales data in several worksheets.
- Macros you create are available for others to use.

Starting Macro Library Manager

To start Macro Library Manager, you **attach** the add-in into your computer's memory. You then **invoke** it to use Macro Library Manager commands.

To Attach Macro Library Manager

1. Select /Add-In Attach.

1-2-3 displays a menu of add-ins (*.ADN) in the current directory. Macro Library Manager is in a file called MACROMGR.ADN.

2. Specify MACROMGR.ADN as the add-in to attach.

If MACROMGR.ADN is not in the menu, specify the drive and/or directory that contains this file. Press **ESC** to clear the currently displayed file names, edit the drive and/or directory name, and then press **ENTER**.

3. Select the key you want to use to invoke Macro Library Manager:

| | |
|--------|--|
| No-Key | Does not assign Macro Library Manager to any key. |
| 7 | Assigns Macro Library Manager to ALT-F7 (APP1) . |
| 8 | Assigns Macro Library Manager to ALT-F8 (APP2) . |
| 9 | Assigns Macro Library Manager to ALT-F9 (APP3) . |
| 10 | Assigns Macro Library Manager to ALT-F10 (APP4) . |

4. Select Quit to return 1-2-3 to READY mode.

To Invoke Macro Library Manager

How you invoke Macro Library Manager depends on whether you selected No-Key or a key when you attached Macro Library Manager.

| If you selected | Do this |
|-----------------|---|
| No-Key | Select /Add-In Invoke, and then select MACROMGR.ADN from the menu of attached add-ins that appears. |
| A key | Press the key you specified. For example, if you selected 7, press ALT-F7 (APP1) . |

The Macro Library Manager menu appears.

NOTE You can attach Macro Library Manager automatically whenever you start 1-2-3. To do this, select /Worksheet Global Default Other Add-In Set, specify an add-in setting (1, 2, 3, 4, 5, 6, 7, or 8), and then specify MACROMGR.ADN as the add-in to attach automatically. You will also need to use /Worksheet Global Default Update to update the 1-2-3 configuration file.

Rules for Using a Macro Library

The rules in this section apply to anyone who wants to work with macro libraries.

When to Attach and Detach Macro Library Manager

You must attach Macro Library Manager before you can save data in a library, load a library file into memory, or run a macro in a library.

If you detach Macro Library Manager during a work session, the macro libraries you have saved or loaded disappear from memory. The library files already on disk are not affected.

Memory Management

- A macro library can contain up to 16,376 cells.
- You can have up to 10 macro libraries in memory simultaneously.
- When you specify the range you want to save in a library, Macro Library Manager allocates a cell in conventional memory for each cell in the range, even if it is empty. To save memory, make your macros as compact as possible and specify ranges with as few empty cells as possible.
- Load only the libraries you need into memory and remove them when you are finished.

Duplicate Library Names

You can have several macro libraries in memory at the same time, but the macro libraries must have different names. If you save a library using the name of an existing library in memory or on disk, a prompt appears asking whether you want to write over the existing library.

Ranges with Links to Other Files

A macro library cannot include ranges that reference, or **link**, to data in another file.

Rules for Macro Commands in a Macro Library

The rules in this section apply to those who want to create macros to save in a macro library.

Range Names

- Refer to a named range in a macro library to run a macro or to perform a macro command. For example, the macro command {LET QTR2,1.5*QTR1} can be in the worksheet or in any library. The ranges QTR1 and QTR2 can be in a library.
- You cannot use library range names to specify a location with 1-2-3 commands, for example, /Move and /Copy.
- Data in a library that you want a macro to use must be in a named range.
- When you specify a range either to start a macro or as an argument in a macro command, 1-2-3 searches the worksheet first, then the first macro library that is in memory. If the range name is not in that library, 1-2-3 looks for it in the next library in memory, and so on, until it finds the named range. If 1-2-3 cannot find the range, an error results.
- Avoid using the same range name in more than one library. If you have two ranges with the same name, 1-2-3 uses the first one it finds, so you may not get the macro you wanted.

Executing Subroutines and Menus in a Library

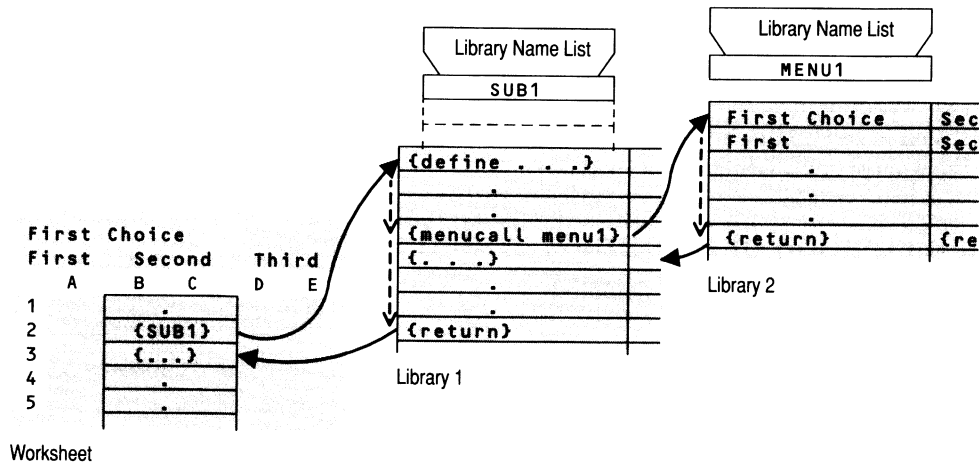
The following macro commands can use subroutines, menus, and other named ranges in all libraries. You can put these macro commands in a worksheet or in any library, regardless of where the range that they reference is stored.

{BRANCH} {MENUBRANCH}

{DISPATCH} {MENUCALL}

{FOR} {ONERROR}

The following illustration shows macro commands that reference two libraries. Part of a macro in a worksheet makes a subroutine call to a routine in Library 1 called SUB1. SUB1, in turn, makes a call to a menu in Library 2. The menu choices appear in the worksheet in the control panel. The user chooses a command from the menu. The subroutine performs the command and returns control to SUB1 in Library 1. SUB1 continues processing and returns to the macro in the worksheet.



Macro Commands that Reference Data

You can reference data in libraries, either from the worksheet or from another library, using macro commands that accept range arguments. These commands can use range references in three ways:

- Macro commands can move data between libraries, or between the worksheet and a library. In the statement {GET CHOICE}, for example, CHOICE can be in a library or in the worksheet. The following {LET} command moves data selected from the range named SALES in a library to cell A1 in the worksheet.

```
{LET A1,@INDEX(SALES,1,2)}
```

- A macro can use cell values in a library or in the worksheet. For example, in the following statements, cell TEST_VAL can be in the worksheet or in a library.

```
{IF TEST_VAL}{BRANCH RTN_1}
{BRANCH RTN_2}
```

- Macro commands may act on the contents of the ranges specified as arguments. For example, {BLANK CAPTURE_TOT} erases the contents of a range named CAPTURE_TOT. CAPTURE_TOT can be in a library or in the worksheet.

Macro Commands that Contain Formulas

Formulas in a macro library can refer only to cells or ranges in the same library; the cells cannot be in a different library or in the worksheet. For example, if a formula references a range called SALES, SALES must be in the same library as the formula. Similarly, formulas in the worksheet cannot refer to cells or ranges in a library.

1-2-3 adjusts cell or range references in a library formula to maintain their relative addresses. For example, when a formula is in the worksheet it may reference a cell whose relative address is two columns to the left and one row up from the formula.

After you move the formula to a library, it will continue to search two columns to the left and one row up in the library for the cell that it references.

NOTE When you save a range in a library, any formulas in the worksheet that were not saved in the library and that reference cells in the range will continue to refer to the worksheet cells, which are now empty.

Recalculation of a Formula Within a Library

A formula is automatically recalculated

- When you save it in a library
- When a macro instruction changes a cell in a library
- When you use {RECALC} or {RECALCCOL} to calculate a library range

1-2-3 calculates formulas in libraries in Natural order, except when you use {RECALC} and {RECALCCOL}. You cannot turn off recalculation of formulas in a library by choosing manual recalculation in the worksheet.

Libraries that Contain /File Retrieve Commands

You can create a library that contains a macro that issues /File Retrieve and provides additional instructions after it retrieves the file. If, however, the file that the macro retrieves contains an auto-execute macro, the auto-execute macro will run instead of the library macro.

For example, a library may contain a macro that retrieves the file SALES and sets the column width to 10. If SALES already contains an auto-execute macro that sets the column width to 15, the auto-execute macro will take over and the worksheet's columns will be set to 15. You can use /Worksheet Global Default Autoexec No to prevent an auto-execute macro from running.

A macro library may contain an auto-execute macro. 1-2-3 runs the auto-execute macro when you retrieve any worksheet.

Creating a Macro Library

To create a macro library, you enter macros and data in a worksheet. You then use Macro Library Manager to name the library and specify the range you want it to contain. Macro Library Manager moves the range to the library, removing it from the worksheet.

To Create a Macro Library

1. Attach Macro Library Manager. (See "Starting Macro Library Manager" on page 178.)
2. Select /Worksheet Erase to start with a blank worksheet.

3. Enter the macros and data you want to store in the library.
4. Use /Range Name Create to assign unique names to the macros. For more information about macro names, see “Naming a Macro” on page 98.
5. Test your macros in the worksheet so you know they work correctly.

Saving Macros in a Library

After you create the macros, use Macro Library Manager to move the macros to a library in memory as well as to a library (.MLB) file on disk.

Macro Library Manager removes the range that contains your macros from the worksheet. Range names associated with the cells no longer refer to the worksheet; they now refer to library locations.

To Save Macros in a Library

1. Invoke Macro Library Manager.
2. Select Save.

Macro Library Manager displays a menu of the macro libraries (files with the .MLB extension) in the current directory.

3. Specify the name of the library in which you want to save the macros or data ranges.

You can specify a path to store the library in another directory. You do not need to specify a file extension because Macro Library Manager automatically adds .MLB. If the library already exists, Macro Library Manager asks whether you want to replace it.

4. Specify the range that contains the macros and data you want to save in the macro library.

The range can contain a single macro, a number of macros, a combination of macros and data (the data can include formulas), or just data. Make sure you include any ranges and cells that the library formulas will reference. A library formula can refer only to cells or ranges in the same library.

5. (Optional) Specify a password (up to 80 characters).

You must remember the exact combination of uppercase or lowercase letters you type, or you will not be able to edit the library in the future. Anyone will, however, be able to use the library without specifying the password.

The macros are now saved in the library.

Using a Macro in a Library

You start a macro stored in a macro library just as you start a macro in any worksheet.

1-2-3 runs the macro from the macro library as if the macro were stored in the worksheet. In addition, 1-2-3 refers to the macro library for any {subroutine}, {BRANCH}, {DISPATCH}, {FOR}, {ONERROR}, {MENUBRANCH}, and {MENUCALL} macro commands, unless the worksheet contains routines with the same name(s).

When you start Macro Library Manager, it can load a macro library into memory automatically. To do this, specify the name AUTOLOAD.MLB when you save the macro library. No macros in the AUTOLOAD.MLB library will be executed automatically (not even \0 macros), but they will be ready for you to use.

To Use a Macro in a Library

1. Retrieve the worksheet in which you want to use the macros.
2. Start the Macro Library Manager. (See “Starting Macro Library Manager” on page 178.)
3. Select Load.
Macro Library Manager displays a menu of files with a .MLB extension in the current directory.
4. If you want to display libraries in another drive and/or directory, press ESC to clear the file names, edit the drive and/or directory name, and then press ENTER.
5. Specify the library you want to load.
If a library with the name you specify is already loaded, you can choose to write over the library already in memory.
6. If necessary, move the cell pointer to the cell where you want the macro to start.
7. Start the macro:

Backslash macros — Hold down ALT and press the single letter of the macro range name.

Range name macros — Press ALT-F3 (RUN) and then select the macro name from the menu that appears. Optionally, you can then press F3 (NAME) to display a full-screen menu of range names. 1-2-3 lists the worksheet’s range names followed by the range names in each macro library that is currently loaded. When you highlight a worksheet range name, 1-2-3 displays its range coordinates. When you highlight a library range name, 1-2-3 displays the name of its library.

Making Changes to Macros in a Library

Macro Library Manager lets you copy a macro library from memory into the worksheet so you can make changes to the library or use its contents in the worksheet. Macro Library Manager leaves a copy of the library in memory and in the library (.MLB) file on disk.

To Change Macros in a Library

1. Start Macro Library Manager. (See “Starting Macro Library Manager” on page 178.)

2. Select Edit.

Macro Library Manager displays a menu of the libraries in memory. The libraries appear in the order in which you saved them or loaded them from disk during the current work session.

3. Specify the library you want to edit.

4. If the macro library is password-protected, type the password and press **ENTER**.

You must type the same combination of uppercase and lowercase letters that you used when you created the password.

5. A prompt appears asking you to specify the action you want 1-2-3 to take if any range names in the library match range names in the worksheet. Select one of the following options:

| | |
|-----------|--|
| Ignore | Uses range names in the worksheet instead of those in the library. |
| Overwrite | Uses range names in the library instead of those in the worksheet. |

6. Specify an unprotected range in the worksheet for the library. You need to specify only the cell for the upper left corner of the range.

CAUTION Make sure that the range you specify in the worksheet is blank or contains unimportant data because Macro Library Manager writes over existing data when it copies the library into the worksheet. If you accidentally write over existing data and the undo feature is on, press **ALT-F4 (UNDO)** immediately to restore the worksheet to its original state.

7. Edit the contents of the library in the same way as you edit worksheet data: by changing it, erasing it, or adding to it.
8. Save the edited library if you want to use it in the future. (For more information, see “Saving Macros in a Library” on page 183.)

Keep the following information in mind when you change macros in a library:

- You can use the same information in several worksheets. For example, if you have a template for your monthly budget that you want to use in individual files, you could retrieve each file and use Edit to copy the template into each worksheet.
- When you save a library you have edited, Macro Library Manager treats it as if it were loaded into memory for the first time. A newly edited library appears last when Macro Library Manager lists libraries that are in memory. When you run a macro, Macro Library Manager will search this library for range names after it searches the worksheet and all other libraries.

Removing a Macro Library from Memory

When you remove a macro library from memory, Macro Library Manager leaves a copy of the library on disk.

To Remove a Macro Library from Memory

1. Select Remove.


Macro Library Manager displays a menu of the libraries in memory. The libraries appear in the order in which you saved them or loaded them from disk during the current work session.

2. Specify the library you want to remove.

Macro Library Manager removes the library from memory.

TIP To delete a library file from disk, select /File Erase Other, press ESC, type *.mlb, press ENTER, and select the library you want to delete.

Using Macro Library Manager on a Network

 Although Macro Library Manager does not support network use, you can keep libraries on a network file server for shared use. If, however, you want to edit the contents of the library, coordinate your efforts with others on your network.

CAUTION If both you and another person are editing the same library, one set of changes will be lost. If you save your changes before the other person, the other person's changes will eventually replace yours, and vice versa.

Macro Library Command Summary

The Macro Library Manager commands let you create and manage the macro libraries that you use with 1-2-3. The Macro Library Manager menu appears when you invoke the add-in.

| Command | Task |
|----------------|---|
| Edit | Copies the contents of a macro library in memory to a range in the worksheet so you can make changes to the library. |
| Load | Copies the contents of library (.MLB) file on disk into memory so you can use the library. |
| Name-List | Enters, in the worksheet, a list of the range names contained in a macro library. |
| Quit | Leaves the Macro Library Manager menu and returns you to 1-2-3. |
| Remove | Erases a macro library from memory. |
| Save | Moves the contents of a range and its range names into a macro library in memory as well as to a library (.MLB) file on disk. |

Index

Symbols

. (period), argument separator,
2, 3, 92, 94
{?}, 111 to 112
, (comma), argument separator,
2, 3, 92, 94
; (semicolon), argument
separator, 2, 3, 92, 94
"" (empty string), 10, 13, 56, 59,
72, 74, 86, 128, 141, 143,
144, 165
{ (open brace), 92, 93
{ } and { }, 112
} (close brace), 92, 93
@?, 16
@@, 15 to 16
/X commands, 166
\ (backslash), macro names, 98
{~}, 112
~ (tilde), 93, 112

A

{ABS}, 112
@ABS, 17
Absolute value, 17
Access type (files), 149
@ACOS, 17 to 18
Add-in @function, testing for,
52
Add-in @functions, 5
Adding, 79 to 80
field values, 39 to 40
ALT-F2 (STEP), 103 to 104
ALT-F3 (RUN), 100, 104, 105
ALT-F4 (UNDO), 99, 104
ALT-F5 (LEARN), 92, 104
Amortized payments, 66 to 67
Angle, arc tangent, 19 to 20
Annual interest rate, 29, 67, 70
Annuity
calculating, 9
due, 8, 45
future value, 45
ordinary, 8, 45
payments from, 66
{APP1}, 113
{APP2}, 113
{APP3}, 113
{APP4}, 113
{APPENDBELOW}, 113 to 115
{APPENDRIGHT}, 113 to 115
Arc cosine, 17 to 18
Arc sine, 18
Arc tangent, 19 to 20
Arguments, 1, 92
and blank cells, 4
for database statistical
@functions, 6
for financial @functions, 8
for logical @functions, 10
for mathematical @functions,
10
for statistical @functions, 12
for string @functions, 13
@functions, 2 to 4
macro commands, 93 to 94
nested, 2
optional, 2
passed to a subroutine,
122 to 123
string, 2, 3, 95, 111
value, 95, 111
Argument separators, 3
changing, 2, 92
defined, 2, 92
macro, 94
@ASIN, 18
Asset, depreciation allowance,
34 to 35, 75 to 76, 80 to 81
@ATAN, 19
@ATAN2, 19 to 20
Attribute
argument for @CELL,
22 to 23
argument for @functions, 4
assigned to a cell, 22 to 23
removing, 25 to 26
Auto-execute macro, 100
in macro library, 182
Worksheet Global Default,
100
AUTOLIB.MLB, 184
Average, 21, 31 to 32
variance from, 40 to 41,
87 to 88
@AVG, 21

B

Backslash names, 98
{BACKSPACE}, 115
Base 10 numbers, 57 to 58
{BEEP}, 110, 115 to 116
in example, 24
{BIGLEFT}, 116
{BIGRIGHT}, 116
{BLANK}, 116 to 117
Blank cell, 28
and @AVG, 21
and @MAX, 58
and @MIN, 60
and @SUM, 79
as location argument, 4
for a learn range, 104
to end a macro, 92, 97
Boolean conditions, evaluating,
9
{BORDERSOFF}, 117
{BORDERSON}, 117
Braces, 92
{BRANCH}, 117 to 118, 171
compared with {GOTO}, 117
in example, 24, 28
with {IF}, 117
Branching, 117 to 118
in response to an error,
147 to 148
{BREAK}, 118
{BREAKOFF}, 101, 118 to 119
{BREAKON}, 118 to 119
Buffer, typeahead, 143
Byte pointer
moving, 158
position of, 137

C

{CALC}, 119
Calculation
decimal portion of date
number, 7
flagging errors, 42
in @ISNUMBER, 54
preventing errors, 53
remainder, 61
text only, 74
unavailable data, 53 to 54, 63

- Calling a subroutine, 159
- Calling macro, returning to, 157
- Canceling a macro, 101
- Capital budgeting, 9
- Capitalization
 - @LOWER, 58
 - @PROPER, 67
 - @UPPER, 85 to 86
- Cash flows, 64 to 65
- @CELL, 22 to 23
- Cell
 - addresses, 3
 - attributes, 22 to 23
 - changing settings, 23
 - counting, 28, 33 to 34
 - location of current, 24
 - obtaining information about, 22 to 23
- Cell contents
 - erasing, 116 to 117
 - label, 55, 74
 - numeric, 54 to 55, 62
 - obtaining, 15 to 16, 25, 46 to 47, 49, 88 to 89
 - testing for, 54 to 55
 - text, 55, 74
- @CELLPOINTER, 23 to 24, 171, 175
- @CHAR, 24
- Characters
 - capitalization, 58, 67, 85 to 86
 - code for, 24, 26
 - comparing, 42 to 43
 - converting to number, 31, 83 to 84
 - counting number of, 56 to 57
 - extracting, 44
 - finding, 44 to 45, 56, 59, 71 to 72
 - manipulating with string
 - @functions, 13 to 14
 - repeating, 70
 - replacing, 44, 71
 - retrieving, 56, 59, 71 to 72
- Checking entries, 23, 42
- @CHOOSE, 25
- @CLEAN, 25 to 26
- {CLOSE}, 120
- Closing a text file, 120
- @CODE, 26
- Code
 - character, 24, 26
 - number, 26
- @COLS, 27
- Column, number in a range, 27
- Column letters, suppressing
 - display of, 133 to 134

- Column width, set with
 - macros, 172 to 173
- Columnwise recalculation, 154 to 155
- Comma, as argument separator, 2, 92
- Commands
 - example in macros, 93
 - Range Name Create, 98
 - Range Name Labels, 98
- Comments, 99
- Common logarithm, 57 to 58
- Comparing two sets of
 - characters, 42 to 43
- Compound growth or loss,
 - @LN to calculate, 57
- Compound term of an
 - investment, 29
- Condition argument, 3, 94
- Conditional formulas
 - @FALSE, 44
 - @TRUE, 85
- Conditional processing, 140 to 141
- {CONTENTS}, 120 to 122
- Control line, freezing, 150
- Control panel, freezing, 170
- Control panel area
 - disabling, 150
 - enabling, 150
- Copying
 - data as a label, 120 to 122
 - from a text file, 152 to 154
 - to a text file, 164 to 165
- Copying data
 - with {APPENDBELOW}, 113 to 115
 - with {APPENDRIGHT}, 113 to 115
- @COS, 27 to 28
- Cosecant, 75
- Cosine, 27 to 28
 - inverse, 17 to 18
- Cotangent, 81
- @COUNT, 28
- Counting entries, 28, 33 to 34
- Criteria argument, 6, 32
- @CTERM, 29
- CTRL-BREAK**, 101, 104
 - disabling, 118 to 119
 - in a for-next loop, 127
- CTRL-F1 (BOOKMARK)**, 104
- Current cell, obtaining
 - information about, 23 to 24

D

- Data
 - checking, 23
 - erasing, 116 to 117
- Data manipulation macro
 - commands, 106
 - {APPENDBELOW}, 113 to 115
 - {APPENDRIGHT}, 113 to 115
 - {BLANK}, 116 to 117
 - {CONTENTS}, 120 to 122
 - {LET}, 142 to 143
 - {PUT}, 151 to 152
 - {RECALC}, 154 to 155
 - {RECALCCOL}, 154 to 155
- Database, 5
 - average value in field, 31 to 32
 - criteria for, 6
 - field in, 6
 - input range, 6
 - offset number, 2
- Database statistical @functions, 5 to 7
 - arguments for, 6
 - compared with statistical
 - @functions, 5
 - @DAVG, 31 to 32
 - @DCOUNT, 33 to 34
 - @DMAX, 35 to 36
 - @DMIN, 36 to 37
 - @DSTD, 37 to 39
 - @DSUM, 39 to 40
 - @DVAR, 40 to 41
- @DATE, 30
- Dates
 - current, 8
 - entering, 30
 - formats, 7
 - @functions, 7
- Date and Time @functions, 7 to 8
 - current date and time, 63 to 64
 - @DATE, 30
 - @DATEVALUE, 31
 - @DAY, 32 to 33
 - day argument, 33
 - @HOUR, 47 to 48
 - @MINUTE, 60 to 61
 - @MONTH, 62
 - month argument, 62
 - @NOW, 63 to 64

- @SECOND, 74
- @TIME, 83
- @TIMEVALUE, 83 to 84
- @YEAR, 90
 - year argument, 90
- Date numbers, 7, 30 to 31
- @DATEVALUE, 31
- @DAVG, 31 to 32
- @DAY, 32 to 33
- Day of month, determining,
 - 32 to 33
- @DCOUNT, 33 to 34
- @DDB, 34 to 35
- Debugging, macros, 101 to 103
- {DEFINE}, 122 to 123
- Degrees
 - converting radians to, 18
 - converting to radians, 27
- {DELETE}, 124
- Depreciation, 8
 - double-declining balance
 - method, 34 to 35
 - straight-line, 75 to 76
 - sum-of-the-years'-digits,
 - 80 to 81
- Dialog box, 93, 101, 163
- Discounting to present value,
 - 64, 67 to 68
- {DISPATCH}, 124
- @DMAX, 35 to 36
- @DMIN, 36 to 37
- Documenting, macro, 99
- DOS, command in macro, 160
- {DOWN}, 125
- @DSTD, 37 to 39
- @DSUM, 39 to 40
- @DVAR, 40 to 41

E

- {EDIT}, 125
- EDIT mode, running a macro
 - in, 99
- Empty cells
 - and @AVG, 21
 - and @MAX, 58
 - and @MIN, 60
 - and @SUM, 79
 - in macro libraries, 179
- Empty string, as argument, 13
- {END}, 125
- Ending a macro, 97, 152
- ENTER, 112
 - representing with ~ (tilde),
 - 93
- Entering
 - dates, 31

- @function, help for, 3
- macros, 96 to 97
- Entry
 - checking, 23, 42
 - copying as a label, 120 to 122
 - counting, 28, 33 to 34
 - date, 30
 - entering with {PUT},
 - 151 to 152
 - in macro, 135 to 136
 - number or label, 62, 74
 - removing spaces in, 84
 - time, 83
- Entry form
 - creating with {FORM},
 - 128 to 131
 - leaving with {FORMBREAK},
 - 132 to 133
- Equal operator (=), compared
 - with @EXACT, 43
- Erasing a range, 116 to 117
- @ERR, 42
 - in example, 15
- ERR, 9, 28
 - in example, 23
 - label versus value, 42
 - preventing, 53, 74
 - produced by @ERR, 42
 - testing for, 53

Error

- branching as a result of,
 - 147 to 148
- during macro, 101 to 103
- fatal, 147
- flagging with @ERR, 42
- preventing, 48
- syntax, 147
- testing for, 9, 53

- Error message, recording,
 - 147 to 148

- {ESCAPE}, 125
- @EXACT, 42 to 43
- @EXP, 43

- Exponential functions, 43,
 - 57 to 58

- Extracting text, 44

F

- F3 (NAME), 100
- @FALSE, 44
- FALSE, as returned value, 9
- Field
 - average value in, 31 to 32
 - counting nonblank cells, 33
 - defined, 6
 - greatest value in, 35 to 36

- least value in, 36 to 37
- name in argument, 31

- Field argument, 6
- Field name, 5

- File manipulation macro
 - commands, 107

- {CLOSE}, 120
- {FILESIZE}, 125 to 126
- {GETPOS}, 137
- {OPEN}, 148 to 150
- {READ}, 152 to 153
- {READLN}, 153 to 154
- {SETPOS}, 158
- {WRITE}, 164
- {WRITELN}, 165
- {FILESIZE}, 125 to 126

- Financial @functions, 8 to 9

- @CTERM, 29
- @DDB, 34 to 36
- @FV, 45 to 46
- @IRR, 50 to 51
- @NPV, 64 to 65
- @PMT, 66 to 67
- @PV, 67 to 68
- @RATE, 69 to 70
- @SLN, 75 to 76
- @SYD, 80 to 81
- @TERM, 81 to 82

- @FIND, 44 to 45

Finding

- list entries, 25
- table entries, 49

- Flow-of-control macro
 - commands, 107 to 108

- {BRANCH}, 117 to 118
- {DEFINE}, 122 to 123
- {DISPATCH}, 124
- {FOR}, 126 to 127
- {FORBREAK}, 127 to 128
- {IF}, 140 to 141
- {ONERROR}, 147 to 148
- {QUIT}, 152
- {RESTART}, 156
- {RETURN}, 157
- {*subroutine*}, 158 to 160

- {FOR}, 126 to 127

- ending current iteration, 157
- repeating action on
 - subsequent rows, 73

- For-next loop, 126 to 127

- ending current iteration, 157
- stopping, 127
- stopping with {FORBREAK},
 - 127 to 128

- {FORBREAK}, 127 to 128

- {FORM}, 128 to 131

canceling with {RESTART},
156

Format
date and time, 7
@functions, 2
macro commands, 93
truncating values, 50
{FORMBREAK}, 132 to 133

Formulas
determining dependent
values, 42, 63
ERR or NA in, 9
{FRAMEOFF}, 133 to 134
{FRAMEON}, 133 to 134

@Functions
add-in, 5
arguments, 3 to 4
database statistical, 5 to 7
date and time, 7 to 8
defined, 1 to 2
financial, 8 to 9
format, 2
logical, 9 to 10
mathematical, 10 to 11
rules, 3
special, 11 to 12
statistical, 12
string, 13 to 14

@Function types, 4 to 14

@Functions, unknown, 16

Future value, 9, 45 to 46
compounding term required,
29
rate needed to achieve,
69 to 70

@FV, 45 to 46

G

{GET}, 134 to 135
{GETLABEL}, 135 to 136, 171
{GETNUMBER}, 136
{GETPOS}, 137
{GOTO}, 138
{GRAPH}, 138
{GRAPHOFF}, 138 to 139
{GRAPHON}, 138 to 139

H

{HELP}, 139
Help, 3
@HLOOKUP, 46 to 47
{HOME}, 139
Horizontal lookup, 46 to 47

@HOUR, 47 to 48
Hour, finding the, 47 to 48

I

IBM Multilingual Character Set,
24, 26

{IF}, 140 to 141
in example, 24, 28
with {BRANCH}, 117
with {LET}, 142

@IF, 48
nesting, 48
with @@, 15
with @ABS, 17
with @CELL, 23
with @ERR, 42, 48
with @ISERR, 53
with @ISNA, 53
with @ISSTRING, 48
with @NA, 48

If-then-else, 140

@INDEX, 49

Index row, 46

{INDICATE}, 141 to 142
reversing {PANELOFF} with,
150

Indicator, 110

Indirect cell reference, 15 to 16

Input argument, 6

{INSERT}, 142

@INT, 50

Integer
calculating value, 50
random number, 69

Interactive macro commands,
108 to 109
{?}, 111 to 112
{BREAKOFF}, 118 to 119
{BREAKON}, 118 to 119
{FORM}, 128 to 131
{FORMBREAK}, 132 to 133
{GET}, 134 to 135
{GETLABEL}, 135 to 136
{GETNUMBER}, 136
{LOOK}, 143 to 144
{MENUBRANCH},
144 to 147
{MENUCALL}, 144 to 147
{SYSTEM}, 160 to 161
{WAIT}, 162

Interest rate
argument for financial
@functions, 8
calculating, 8

to achieve future value,
69 to 70

Internal rate of return, 50
International characters, codes,
24

Inverse

cosine, 17 to 18
sine, 18
tangent, 19 to 20

Investment

calculating periodic
payments, 66
comparing, 64, 68
compound term, 29
future value, 45
internal rate of return, 50
net present value, 64
present value, 67 to 68
rate to achieve future value,
69 to 70
term, 81 to 82

@IRR, 50 to 51
@ISAAF, 51 to 52
@ISAPP, 52
@ISERR, 53
@ISNA, 53 to 54
@ISNUMBER, 54 to 55
@ISSTRING, 55

K

Key names

{ } and {}, 112
{~}, 112
{ABS}, 112
{APP1}, 113
{APP2}, 113
{APP3}, 113
{APP4}, 113
{BACKSPACE}, 115
{BIGLEFT}, 116
{BIGRIGHT}, 116
{CALC}, 119
{DELETE}, 124
{DOWN}, 125
{EDIT}, 125
{END}, 125
{ESCAPE}, 125
{GOTO}, 138
{GRAPH}, 138
{HELP}, 139
{HOME}, 139
{INSERT}, 142
{LEFT}, 142
{MENU}, 144

{NAME}, 147
 {PGDN}, 151
 {PGUP}, 151
 {QUERY}, 152
 {RIGHT}, 157
 {TABLE}, 161
 {UP}, 161
 {WINDOW}, 163

Keys
See also Key names
ALT-F2 (STEP), 103 to 104
ALT-F3 (RUN), 100
ALT-F4 (UNDO), 104
ALT-F5 (LEARN), 104
CTRL-BREAK, 101
CTRL-F1 (BOOKMARK), 104
F3 (NAME), 100
 for add-ins, 109
 macro key names, 109 to 110

Keystrokes
 excluding with {FORM}, 129
 in a macro, 92, 93
 recording during macro, 134 to 135, 143 to 144

Keyword, 106

L

Label
 and @AVG, 21
 and @MAX, 58
 and @MIN, 60
 and @SUM, 79
 checking, 74
 comparing, 42 to 43
 converting to number, 86
 converting to time, 83 to 84
 determining length, 56 to 57
 entering in a macro, 135 to 136
 entering with {PUT}, 151 to 152
 from value, 78 to 79
 macro instructions, 93
 removing spaces in, 84
 returning number of characters in, 56 to 57
 testing for with @ISSTRING, 55

Learn feature, 104 to 105
 LEARN indicator, 104
 {LEFT}, 142
 @LEFT, 56
 @LENGTH, 56 to 57
 {LET}, 142 to 143
 @LN, 57

Loan, determining payment, 66 to 67

Location argument, 3, 94, 166
 @LOG, 57 to 58

Logarithm
 common (base 10), 57 to 58
 inverse natural logarithm, 43
 natural (base *e*), 57

Logical expression, 140, 166
 false, 44
 true, 85

Logical @functions, 9 to 10
 arguments for, 10
 @FALSE, 44
 @IF, 48
 @ISAAF, 51 to 52
 @ISAPP, 52
 @ISERR, 53
 @ISNA, 53 to 54
 @ISNUMBER, 54 to 55
 @ISSTRING, 55
 @NA, 63
 @TRUE, 85

Logical operators, 3
 {LOOK}, 143 to 144

Lookup, in a list, 25

Lookup table, 46 to 47, 49, 88 to 89

Loop
 creating a, 117 to 118
 rounding, 170 to 171

Lotus International Character Set (LICS), 24, 26
 @LOWER, 58

Lowercase, converting to, 58

M

Macro
 / (slash) in, 93
 argument separators, 94
 branching, 117 to 118
 canceling, 101
 categories, 106
 common errors, 102
 controlling screen, 110
 creating, 96
 defined, 91
 DOS commands in, 160
 ending, 97, 152, 156, 157
 entering in a worksheet, 95, 96
 entry during, 128 to 131
 erasing data, 116 to 117
 errors in, 101 to 103
 excluding user keystrokes, 129

freezing screen display during, 163
 interactive, 108 to 109
 keystrokes, 92 to 93
 library, 177 to 187
 looping, 171
 mouse actions in, 92
 naming, 98
 numeric characters in, 93
 pausing, 111 to 112, 162
 placement, 95
 preventing users from stopping, 118 to 119
 recalculation during, 101
 recording user keystrokes, 134 to 135, 143 to 144
 resuming, 132 to 133
 returning from subroutine, 157
 running, 99
 running automatically, 100
 running from library, 184
 sample, 167 to 176
 spaces in, 94
 steps to create, 96 to 98
 stopping, 118 to 119
 suspending, 101, 111 to 112, 128 to 131, 162, 163
 symbols, 93
 tips for creating, 97

Macro commands
See also Flow-of-control macro commands;
 Interactive macro commands;
 Screen control macro commands

argument types, 94 to 95
 arguments, 93
 data manipulation, 106
 file manipulation, 107
 flow-of-control, 107 to 108
 interactive, 108 to 109
 keyword, 93
 screen control, 110
 syntax, 93

Macro key names, 109 to 110
 Macro keyword, 92
 Macro library, 177 to 187
 and memory management, 179
 creating, 182 to 183
 erasing from disk, 186
 loading automatically, 184
 moving data between, 181
 multiple, 179
 recalculation of formulas in, 182

- referencing data in, 181
- removing from memory, 186
- saving formulas in, 181
- size of, 179
- updating, 185 to 186
- using macros in, 184
- Macro Library Manager
 - assigning to a key, 178
 - attaching automatically, 178
 - commands, 187
 - invoking, 178
 - starting, 178
 - using, 177 to 187
 - using on a network, 186
- Macro names, 92, 98
 - \ (backslash), 98
 - range names, 98
- MACROMGR.ADN, 178
- Mailing labels, creating, 174
- Mathematical @functions,
 - 10 to 11
 - @ABS, 17
 - @ACOS, 17 to 18
 - arguments for, 10
 - @ASIN, 18
 - @ATAN, 19
 - @ATAN2, 19 to 20
 - @COS, 27 to 28
 - @EXP, 43
 - @INT, 50
 - @LN, 57
 - @LOG, 57 to 58
 - @MOD, 61
 - @PI, 65 to 66
 - @RAND, 69
 - @ROUND, 72 to 73
 - @SIN, 75
 - @SQRT, 76
 - @TAN, 81
- Mathematical symbols, codes,
 - 24
 - @MAX, 58 to 59
- Maximum value, 35 to 36,
 - 58 to 59
- Mean, 21, 31 to 32
 - standard deviation from,
 - 37 to 39, 77 to 78
 - variance from, 40 to 41,
 - 87 to 88
- Memory, for macro libraries,
 - 179
- {MENU}, 144
- Menu
 - creating, 144 to 147
 - in macro libraries, 180
 - macro, 173
- Menu bar, freezing, 150

- MENU mode, running a macro
 - in, 99
- {MENUBRANCH}, 144 to 147
- {MENCALL}, 144 to 147, 173
- @MID, 59
- @MIN, 60
- Minimum value, finding,
 - 36 to 37, 60
- @MINUTE, 60 to 61
- @MOD, 61
 - to calculate day of the week,
 - 61
- Mode indicator, 110, 141 to 142
- Modulus, 61
- @MONTH, 62
- Month, determining, 62
- Mouse, and learn feature, 104
- N**
 - @N, 62
 - @NA, 63
 - NA, 9, 28
 - label versus value, 53, 63
 - preventing, 48, 53
 - produced by @NA, 63
 - testing for, 53 to 54
 - {NAME}, 147
 - Name, macro, 98
 - Natural logarithm, 57
 - inverse, 43
 - Negative values, and @ABS, 17
 - Nested subroutines, 159
 - canceling return sequence,
 - 156
 - Nesting, @IF, 48
 - Net present value, 9, 64 to 65
 - Network, Macro Library
 - Manager on, 186
 - @NOW, 63 to 64, 169
 - extracting hour, 47
 - extracting minutes, 60
 - extracting seconds, 74
 - @NPV, 64 to 65
 - Number
 - calculating root, 57
 - converting to label, 78 to 79
 - even, 61
 - from label, 86
 - in a macro, 93
 - integer value, 50
 - odd, 61
 - random, 69
 - square root, 76
 - testing for, 54 to 55

O

- Offset number, 2, 6, 25
 - field, 31
- {ONERROR}, 147 to 148
- {OPEN}, 148 to 150

P

- {PANELOFF}, 150, 170
- {PANELON}, 150, 170
- Parentheses, and arguments, 2,
 - 3
- Pause, in a macro, 111 to 112
- Payments, 9, 66 to 67
 - investment, 81 to 82
- Period, compounding, 29
- {PGDN}, 151
- {PGUP}, 151
- @PI, 65 to 66
 - in example, 18, 28
 - to convert
 - degrees to radians, 27
 - to convert
 - radians to degrees, 18
- Pi, 65 to 66
- @PMT, 66 to 67
- Pointer-movement keys, macro
 - for, 168
- Positive values
 - forcing with @ABS, 17
 - testing for, 17
- Present value, 9, 67 to 68
 - of an investment, 50
- Printing
 - foreign-language characters,
 - 24
 - line length, 57
 - size of range, 27, 73
 - symbols, 24
- Prompt, creating for user entry,
 - 135 to 136
- @PROPER, 67
- {PUT}, 151 to 152
- @PV, 67 to 68

Q

- {QUERY}, 152
- {QUIT}, 152
- Quotation marks and string
 - arguments, 2

R

Radians, converting to degrees, 18
@RAND, 69
Random number, 69
Range
 counting nonblank cells, 28
 determining size, 27, 73
 greatest value, 58 to 59
 learn, 104
 least value, 60
Range name, 95, 98
 and subroutine, 158 to 159
 in macro libraries, 180
 location argument, 4
 macro name, 98
Range references, in macros, 97
@RATE, 69 to 70
Rate of return, 9
{READ}, 152 to 153
Reading a text file, 153 to 154
{READLN}, 153 to 154
READY mode, running a macro
 in, 99
{RECALC}, 154 to 155
{RECALCCOL}, 154 to 155
Recalculation
 during macros, 101
 in a macro, 154 to 155
Reciprocal
 of sine, 75
 of tangent, 81
Relational operators, 3
Remainder, 61
Removing
 See also @TRIM
 control characters from a
 string, 25 to 26
@REPEAT, 70
Repeating characters, 70
@REPLACE, 71
Replacing characters, 44, 71
{RESTART}, 156
Retrieving, a file that contains
 auto-execute macro, 100
{RETURN}, 157
 and {FOR}, 157
{RIGHT}, 157
@RIGHT, 71 to 72
Right triangle, 18, 27, 75, 81
Ripple-through effect, avoiding,
 42, 53, 63
Root, square, 76
@ROUND, 72 to 73, 170
Rounding, 72 to 73
 versus truncating, 50

Row, shifting to top of screen,
 168
Rows, number in a range, 73
Row numbers, suppressing
 display of, 133 to 134
@ROWS, 73
Running a macro, 99 to 101

S

@S, 74
Sample macros, 167 to 176
SAMPMACS.WK1, 167
Screen control macro
 commands, 110
 {BEEP}, 115 to 116
 {BORDERSOFF}, 117
 {BORDERSON}, 117
 {FRAMEOFF}, 133 to 134
 {FRAMEON}, 133 to 134
 {GRAPHOFF}, 138 to 139
 {GRAPHON}, 138 to 139
 {INDICATE}, 141 to 142
 {PANELOFF}, 150
 {PANELON}, 150
 {WINDOWSOFF}, 163
 {WINDOWSON}, 163
Searching, 44 to 45
 by dates, 30
@SECOND, 74
Second, determining, 74
{SETPOS}, 158
Settings
 assigned to a cell, 22 to 23
 changing, 23
@SIN, 75
Sine, 75
 inverse, 18
Single cell
 entire macro in a, 95
 macro comand, 95
Size, of a range, 27, 73
@SLN, 75 to 76
Sorting, chronological order, 30
Spaces, removing, 84
Spacing, in a macro, 94
Special @functions, 11 to 12
 @?, 16
 @@, 15 to 16
 @CELL, 22 to 23
 @CELLPOINTER, 23 to 24
 @CHOOSE, 25
 @COLS, 27
 @ERR, 42
 @HLOOKUP, 46 to 47
 @INDEX, 49
 @ROWS, 73
 @VLOOKUP, 88 to 89
@SQRT, 76
 and @ABS, 17
Square root, 57, 76
 and @ABS, 17
Standard deviation, 37 to 39,
 77 to 78
 relation to variance, 38
Statistical @functions, 12
 arguments for, 12
 @AVG, 21
 compared with database
 statistical @functions, 12
 @COUNT, 28
 @MAX, 58 to 59
 @MIN, 60
 @STD, 77 to 78
 @SUM, 79 to 80
 @VAR, 87 to 88
@STD, 77 to 78
STEP mode, 102
Straight-line depreciation,
 75 to 76
@STRING, 78 to 79
String argument, 3, 95, 111
 See also Text
 empty string, 13
String @functions, 13 to 14
 arguments for, 2, 3, 13
 @CHAR, 24
 @CLEAN, 25 to 26
 @CODE, 26
 @EXACT, 42 to 43
 @FIND, 44 to 45
 @LEFT, 56
 @LENGTH, 56 to 57
 @LOWER, 58
 @MID, 59
 @N, 62
 @PROPER, 67
 @REPEAT, 70
 @REPLACE, 71
 @RIGHT, 71 to 72
 @S, 74
 @STRING, 78 to 79
 @TRIM, 84
 @UPPER, 85 to 86
 @VALUE, 86
Subroutine, 158 to 160
 arguments for, 122 to 123
 calling, 144 to 147, 159
 canceling return sequence,
 156, 159
 ending, 92, 157
 in macro libraries, 180
 name, 158 to 159
 nesting, 159

- (QUIT) in, 152
- range name, 158 to 159
- returning from, 157, 159
- Subroutine call, 173
- Suffix, in a macro, 123
- @SUM, 79 to 80
- Suspending macro
 - for specified time, 162
 - for user entry, 128 to 131
 - temporarily, 111 to 112
- @SYD, 80 to 81
- Symbols, 93
- Syntax, macro commands, 93
- {SYSTEM}, 160 to 161

T

- {TABLE}, 161
- Table entries
 - finding with @HLOOKUP, 46 to 47
 - finding with @VLOOKUP, 88 to 89
- Table, lookup, 46 to 47, 88 to 89
- @TAN, 81
- Tangent, 81
 - inverse, 19 to 20
- @TERM, 81 to 82
- Term
 - argument for financial
 - @functions, 8
 - calculating, 9
 - compound, 9, 29
 - of an investment, 81 to 82
- Text
 - capitalization, 58, 67, 85 to 86
 - comparing, 42 to 43
 - converting to number, 86
 - determining length, 56 to 57
 - ensuring in text formulas, 74
 - removing spaces in, 84
 - repeating, 70
 - replacing, 71
 - string @functions, 13 to 14
 - testing for, 55
 - user entry in macro, 135 to 136
- Text file, 107
 - access, 148 to 150
 - byte pointer position, 137
 - closing, 120
 - copying from, 152 to 154
 - copying to, 164 to 165
 - moving the byte pointer, 158
 - opening, 148 to 150

- reading, 152
- size, 125 to 126
- Text formula, 4
- Tilde, 92
- @TIME, 83
 - extracting hour, 47
 - extracting minutes, 60
 - extracting seconds, 74
- Time
 - converting labels, 83 to 84
 - current, 8, 63 to 64
 - entering, 83 to 84
 - formats, 7
 - @functions, 7 to 8
- Time number, 7, 83 to 84
- @TIMEVALUE, 83 to 84
 - extracting hour, 47
 - extracting minutes, 60
 - extracting seconds, 74
- Title bar, freezing, 150
- Tone
 - macro for sounding a, 110
 - sounding computer, 115
- Total, 39 to 40, 79 to 80
- Trigonometric @functions
 - @ACOS, 17 to 18
 - arc cosine, 17 to 18
 - arc sine, 18
 - arc tangent, 19
 - @ASIN, 18
 - @ATAN, 19
 - @ATAN2, 19 to 20
 - @COS, 27 to 28
 - cosine, 27 to 28
 - @PI used with, 65
 - @SIN, 75
 - sine, 75
 - @TAN, 81
 - tangent, 81
- @TRIM, 84
- Troubleshooting macros, 101 to 103
- @TRUE, 85
- TRUE, as returned value, 9
- Typeahead buffer, 143

U

- {UP}, 161
- @UPPER, 85 to 86
- Uppercase, changing to, 85 to 86
- User entry, in a macro, 108 to 109

V

- @VALUE, 86
- Value, 86
 - adding, 39 to 40, 79 to 80
 - checking for, 62
 - converting to a label, 78 to 79
 - e* (2.718282), 43
 - even, 61
 - greatest in a field, 35 to 36
 - greatest in a range, 58 to 59
 - least in a field, 36 to 37
 - least in a range, 60
 - odd, 61
 - pi, 65
 - random number, 69
 - square root, 76
 - testing for, 54 to 55
- Value argument, 3, 95, 111
- @VAR, 87 to 88
- Variance, 40 to 41, 87 to 88
 - relationship to standard deviation, 41
- Vertical lookup, 88 to 89
- @VLOOKUP, 88 to 89

W

- {WAIT}, 162
- {WINDOW}, 163
- Window
 - display in macro, 163
 - freezing during macro, 163
- {WINDOWSOFF}, 163, 170
- {WINDOWSON}, 163, 170
- Worksheet frame, suppressing display of, 133 to 134
- Worksheet, recalculating, 154
- {WRITE}, 164
- {WRITELN}, 165

Y

- @YEAR, 90
- Year, determining, 90