; Documentation for Questor

Functions

#begin_function
alert(message; [buttons]; [icon_type]; [results_range])
Opens an alert panel.
message is the message in the panel.
buttons is the number of buttons (**1** - **OK** only, **2** - **OK** and **Cancel**)
icon_type is not used
result_range is the cell where the result is put
If you click OK, the result will be 1.
If you click Cancel, the result will be 0.
The function also returns the result.
Ex:
alert("You cannot do that!"; ; ;)
returns **1**
alert("Do you want to do this?"; 2; ;A1)
returns **0** (and puts it in cell **A1**)
#end_function

#begin_function
calculate(repeat_count)
Recalculates the worksheet, but does not update it on the screen. Call **redisplay_changed_cells()** to update the screen.
Also see **redisplay_changed_cells()** and **recalculate()**
#end_function

#begin_function
call_macro(name_or_range)
Calls a macro with or without a name. Returns *true* if successful. Otherwise the function returns *false*.
Ex:
call_macro("my_macro")
returns **true**   *; if my_macro is a name of a range with a macro*
call_macro(C7)
returns **true**   *; if cell C7 contains a macro*
#end_function

#begin_function
cell(attribute; range)
Returns information about a cell in the worksheet.
range is the cell you want to examine
attribute can be one of the following:

```
        "alignment"
        "background_gray"
        "bottom_border"
        "column"
        "comma_is_decimal_point"
        "contents"
        "coord"
        "currency"
        "date_format"
        "decimals"
        "filename"
        "filename_only"
        "fontname"
        "fontsize"
        "format_type"
        "has_comment"
        "has_left_border"
        "height"
        "is_alignment_set"
        "is_bezeled"
        "is_bordered"
        "is_circular"
        "is_format_set"
        "is_hidden"
        "is_unlocked"
        "negative_indicator"
        "postfix"
        "prefix"
        "protect"
        "right_border"
        "row"
        "text_gray"
        "thousands"
```

        "top_border"
        "type"
        "width"

Note: Many of the return values are simply integers. The integer value represents the item number in the pop-up list in the cell format inspector.
Also see **cellpointer()**
Ex:
cell("fontname"; A1)
returns something like **Helvetica**
cell("thousands"; A1)
returns **something like 1** *; the item number in the thousands pop-up list in the cell format inspector*
#end_function

#begin_function
cellpointer(attribute)
Returns information about the active cell in the worksheet.
Also see **cell()**
Ex:
cellpointer("fontname")
returns something like **Helvetica**
cellpointer("thousands")
returns something like **1** *; the item number in the "thousands" pop-up list in the cell format inspector*
#end_function

#begin_function
cell_enter(string; [target_location])
Enters **string** in the cell specified by **target_location**. If **target_location** is omitted, the string will be entered in the active cell. You can enter formulas as well.
The function returns *true* if successful.
Note: To make the changes visible in the worksheet, you must use the recalculate() function.
Also see **set_value_at()**
Ex:
cell_enter("hello world")
Returns **true**

cell_enter("=sin(0.5)"; A1)
Returns **true**
#end_function

#begin_function
choose(offset; list)
Selects an item from a **list. offset** determines which item in the list will be chosen. The first item is numbered **1.**
Ex:
choose(2; 10; 20; 30; 40)
returns **20**
choose(4; "Bad"; "OK"; "Good"; "Great")
returns **Great**
#end_function

#begin_function
choose_many(choice_range; result_range; [prompt]; [title])
Opens a panel with up to eight switches, an OK button and a Cancel button.
choice_range is the range that contains the switch descriptions. This range should have three rows and one column for each switch:
- Each cell in the first row contains the text that should appear at the corresponding switch.
- Each cell in the second row contains the initial state for the corresponding switch. 1 means that the switch is on, 0 means that the switch is off.
- Each cell in the third row contains the state for the corresponding switch when the panel closes after the user clicks OK or Cancel. 1 means that the switch is on, 0 means that the switch is off.
result_range is the cell where the function will put **0** if the user clicked Cancel, and **1** if the user clicked OK.
prompt is the prompt in the panel.
title is the tile of the panel.
The function returns *true* if the user clicks OK, and *false* if the user clicks Cancel.
Ex:
Assume the range A1:C3 looks like this:

Snacks	**Popcorn**	**Chips**
0.00	**0.00**	**0.00**

choose_many(A1:C3; D1; "Pick your choices..."; "Choose Panel")
will open a panel with three switches that are not set.
#end_function

#begin_function
choose_one(choice_range; result_range; [prompt]; [title])
Opens a panel with up to eight radio buttons, an OK button and a Cancel button.
choice_range is the range that contains the button descriptions. This range should have two rows and one column for each button:
- Each cell in the first row contains the text that should appear at the corresponding radio button.
- Each cell in the second row contains the initial state for the corresponding radio button. 1 means that the radio button is on, 0 means that the switch is radio button.
result_range is the cell where the function will put **0** if the user clicked Cancel, and **1** if the user clicked OK.
prompt is the prompt in the panel.
title is the tile of the panel.
The function returns the number of the selected radio button if the user clicks OK, and *false* if the user clicks Cancel.
Ex:
Assume the range A1:C2 looks like this:

Yes	**No**	**Maybe**
1.00	**0.00**	**0.00**

choose_one(A1:C2; D1; "Pick your choice..."; "Choose Panel")
will open a panel with three radio buttons.
#end_function

#begin_function
clear_cells_contents([range])
Clears the cells in **range**. The formatting info of the cells is not affected. If **range** is omitted, the active cell will be affected. Returns *true* if successful.
Note: To make the changes visible in the worksheet, you must use the recalculate() function.
Ex:
clear_cells_contents(A1:B6)
returns **true**

clear_cells_contents(coord(1; 2; 1)) *; cell B1*
returns **true**
#end_function

#begin_function
color_clear_background([range])
Sets the background color of the cells in **range** to clear (the default). If **range** is omitted, the active cell will be affected.
The function returns *true* if successful.
Note: To make the changes visible in the worksheet, you must use the recalculate() function.
Ex:
color_clear_background(A1:B6)
returns **true**
color_clear_background(coord(1; 2; 1)) *; cell B1*
returns **true**
#end_function

#begin_function
color_set_background(red; green; blue; [range])
Sets the background color of the cells in **range** to a color specified in RGB. The **red**, **green** and **blue** color components should be between 0 and 1. If **range** is omitted, the active cell will be affected.
The function returns *true* if successful.
Note: To make the changes visible in the worksheet, you must use the recalculate() function.
Also see **color_set_text()**
Ex:
color_set_background(1; 0.4; 0.6; A1:B6)
returns **true**
color_set_background(coord(1; 2; 1)) *; cell B1*
returns **true**
#end_function

#begin_function
color_set_text(red; green; blue; [range])
Sets the text color of the cells in **range** to a color specified in RGB. The **red,**

green and **blue** color components should be between 0 and 1. If **range** is omitted, the active cell will be affected.
The function returns *true* if successful.
Note: To make the changes visible in the worksheet, you must use the recalculate() function.
Also see **color_set_background()**
Ex:
color_set_text(1; 0.4; 0.6; A1:B6)
returns **true**
color_set_text(coord(1; 2; 1)) *; cell B1*
returns **true**
#end_function

#begin_function
cols(range)
Returns the number of columns in **range.**
Also see **rows()**
Ex:
cols(A1:C5)
returns **3**
#end_function

#begin_function
coord(row; column; absolute)
Returns a *cell reference* that can be used by functions like **color_set_text()** and **color_set_background(). absolute** determines if the row or column (or both) should be absolute.
Ex:
coord(1; 2; 1)
returns **r$1c$2**
coord(1; 2; 2)
returns **r$1c[2]**
coord(1; 2; 3)
returns **r[1]c$2**
coord(1; 2; 4)
returns **r[1]c[2]**
@(coord(1; 2; 1))
returns the value in cell **B1** (row 1, column 2)

#end_function

#begin_function
count_empty(args...)
Returns the number of cell without a value in the ranges specified as the argument.
Also see **count_nonempty()**
Ex:
count_empty(A1:E10)
returns the number of cells without a value in the range **A1:E10**
count_empty(A1:E10; A12:D15)
returns the number of cells without a value in the ranges **A1:E10** and **A12:D15**
#end_function

#begin_function
count_nonempty(args...)
Returns the number of cell with a value in the ranges specified as the argument.
Also see **count_empty()**
Ex:
count_nonempty(A1:E10)
returns the number of cells with a value in the range **A1:E10**
count_nonempty(A1:E10; A12:D15)
returns the number of cells with a value in the ranges **A1:E10** and **A12:D15**
#end_function

#begin_function
davg(input_ranges; field; critera_range)
Returns the average of the cells in a column that matches a criteria.
input_ranges is the data that is used.
field is the name of the field that should be calculated
criteria_range is a range that stores the search criteria.
Also see **davg()**, **dcount()**, **dget()**, **dmax()**, **dmin()**, **dstd()**, **dstds()**, **dsum()**, **dvar()** and **dvars()**
Ex:
Assume the range A1:C5 looks like this:

Item	**No**	**Price**
Car	**1092**	**500**

Truck	**1082**	**850**
Bike	**1072**	**200**
MC	**1062**	**350**

Assume the range A7:A8 looks like this:
Price
>400

davg(A1:C5; "Price"; A7:A8)
returns **675**
(The average of the prices that are higher than 400)
#end_function

#begin_function
dcount(input_ranges; field; critera_range)
Returns the number of rows in a column that matches a criteria.
input_ranges is the data that is used.
field is the name of the field that should be calculated
criteria_range is a range that stores the search criteria.
Also see **davg()**, **dcount()**, **dget()**, **dmax()**, **dmin()**, **dstd()**, **dstds()**, **dsum()**, **dvar()** and **dvars()**
Ex:
Assume the range A1:C5 looks like this:

Item	**No**	**Price**
Car	**1092**	**500**
Truck	**1082**	**850**
Bike	**1072**	**200**
MC	**1062**	**350**

Assume the range A7:A8 looks like this:
Price
>400

dcount(A1:C5; "Price"; A7:A8)
returns **2**
(The number of prices that are higher than 400)
#end_function

#begin_function
dget(input_ranges; field; critera_range)
Returns a cell in a row that matches a criteria.
input_ranges is the data that is used.
field is the name of the field that should be calculated
criteria_range is a range that stores the search criteria.
Also see **davg()**, **dcount()**, **dget()**, **dmax()**, **dmin()**, **dstd()**, **dstds()**, **dsum()**, **dvar()** and **dvars()**
Ex:
Assume the range A1:C5 looks like this:

Item	**No**	**Price**
Car	**1092**	**500**
Truck	**1082**	**850**
Bike	**1072**	**200**
MC	**1062**	**350**

Assume the range A7:A8 looks like this:
Item
Truck

dget(A1:C5; "Price"; A7:A8)
returns **850**
(The price for Truck)
#end_function

#begin_function
dmax(input_ranges; field; critera_range)
Returns the maximum of the cells in a column that matches a criteria.
input_ranges is the data that is used.
field is the name of the field that should be calculated
criteria_range is a range that stores the search criteria.
Also see **davg()**, **dcount()**, **dget()**, **dmax()**, **dmin()**, **dstd()**, **dstds()**, **dsum()**, **dvar()** and **dvars()**
Ex:
Assume the range A1:C5 looks like this:

Item	**No**	**Price**
Car	**1092**	**500**
Truck	**1082**	**850**
Bike	**1072**	**200**

MC	**1062**	**350**

Assume the range A7:A8 looks like this:
Price
<400

dmax(A1:C5; "Price"; A7:A8)
returns **350**
(The maximum of the prices that are lower than 400)
#end_function

#begin_function
dmin(input_ranges; field; critera_range)
Returns the minimum of the cells in a column that matches a criteria.
input_ranges is the data that is used.
field is the name of the field that should be calculated
criteria_range is a range that stores the search criteria.
Also see **davg()**, **dcount()**, **dget()**, **dmax()**, **dmin()**, **dstd()**, **dstds()**, **dsum()**, **dvar()** and **dvars()**
Ex:
Assume the range A1:C5 looks like this:

Item	**No**	**Price**
Car	**1092**	**500**
Truck	**1082**	**850**
Bike	**1072**	**200**
MC	**1062**	**350**

Assume the range A7:A8 looks like this:
Price
>400

dmin(A1:C5; "Price"; A7:A8)
returns **500**
(The minimum of the prices that are higher than 400)
#end_function

#begin_function
documents_close_all([discard_changes])

Closes all open documents.
If **discard_changes** is *true*, then the documents will close without a panel asking you to save changed documents.
#end_function

#begin_function
documents_hide_all()
Equivalent to choosing *Hide All* in the menu *Documents*.
#end_function

#begin_function
document_close([discard_changes])
Equivalent to choosing *Close* in the menu *Document*.
If **discard_changes** is *true*, then the document will close without a panel asking you to save a changed document.
#end_function

#begin_function
document_hide()
Equivalent to choosing *Hide* in the menu *Document*.
#end_function

#begin_function
document_inspect()
Equivalent to choosing *Inspect* in the menu *Document*.
#end_function

#begin_function
document_new_report_layout()
Equivalent to choosing *New Report Layout* in the menu *Document*.
#end_function

#begin_function
document_new_window([name])

Equivalent to choosing *New Window* in the menu *Document*.
#end_function

#begin_function
document_new_worksheet()
Equivalent to choosing *New Worksheet* in the menu *Document*.
#end_function

#begin_function
document_open()
Equivalent to choosing *Open* in the menu *Document*.
#end_function

#begin_function
document_recalculate()
Equivalent to choosing *Recalculate* in the menu *Document*.
Same as **recalculate()**
#end_function

#begin_function
document_revert()
Equivalent to choosing *Revert to Saved* in the menu *Document*.
#end_function

#begin_function
document_save()
Equivalent to choosing *Save* in the menu *Document*.
#end_function

#begin_function
document_save_all()
Equivalent to choosing *Save All* in the menu *Document*.
#end_function

#begin_function
document_save_as()
Equivalent to choosing *Save As* in the menu *Document*.
#end_function

#begin_function
document_set_startup()
Sets the current document to the **startup document** for Questor. The startup document can also be specified in the **Launch & Misc Preferences Panel.**
#end_function

#begin_function
dstd(input_ranges; field; critera_range)
Returns the population standard deviation of the cells in a column that matches a criteria.
input_ranges is the data that is used.
field is the name of the field that should be calculated
criteria_range is a range that stores the search criteria.
Also see **davg()**, **dcount()**, **dget()**, **dmax()**, **dmin()**, **dstd()**, **dstds()**, **dsum()**, **dvar()** and **dvars()**
Ex:
Assume the range A1:C5 looks like this:

Item	**No**	**Price**
Car	**1092**	**500**
Truck	**1082**	**850**
Bike	**1072**	**200**
MC	**1062**	**350**

Assume the range A7:A8 looks like this:
Price
<400

dstd(A1:C5; "Price"; A7:A8)
returns **75**
(The population standard deviation of the prices that are lower than 400)
#end_function

#begin_function
dstds(input_ranges; field; critera_range)
Returns the sample standard deviation of the cells in a column that matches a criteria.
input_ranges is the data that is used.
field is the name of the field that should be calculated
criteria_range is a range that stores the search criteria.
Also see **davg(), dcount(), dget(), dmax(), dmin(), dstd(), dstds(), dsum(), dvar()** and **dvars()**
Ex:
Assume the range A1:C5 looks like this:

Item	**No**	**Price**
Car	**1092**	**500**
Truck	**1082**	**850**
Bike	**1072**	**200**
MC	**1062**	**350**

Assume the range A7:A8 looks like this:
Price
<400

dstds(A1:C5; "Price"; A7:A8)
returns **106.06601717798213**
(The sample standard deviation of the prices that are lower than 400)
#end_function

#begin_function
dsum(input_ranges; field; critera_range)
Returns the sum of the cells in a column that matches a criteria.
input_ranges is the data that is used.
field is the name of the field that should be calculated
criteria_range is a range that stores the search criteria.
Also see **davg(), dcount(), dget(), dmax(), dmin(), dstd(), dstds(), dsum(), dvar()** and **dvars()**
Ex:
Assume the range A1:C5 looks like this:

Item	**No**	**Price**
Car	**1092**	**500**

Truck	**1082**	**850**
Bike	**1072**	**200**
MC	**1062**	**350**

Assume the range A7:A8 looks like this:
Price
>400

dsum(A1:C5; "Price"; A7:A8)
returns **1350**
(The sum of the prices that are higher than 400)
#end_function

#begin_function
dvar(input_ranges; field; critera_range)
Returns the population variance of the cells in a column that matches a criteria.
input_ranges is the data that is used.
field is the name of the field that should be calculated
criteria_range is a range that stores the search criteria.
Also see **davg()**, **dcount()**, **dget()**, **dmax()**, **dmin()**, **dstd()**, **dstds()**, **dsum()**, **dvar()** and **dvars()**
Ex:
Assume the range A1:C5 looks like this:

Item	**No**	**Price**
Car	**1092**	**500**
Truck	**1082**	**850**
Bike	**1072**	**200**
MC	**1062**	**350**

Assume the range A7:A8 looks like this:
Price
<400

dvar(A1:C5; "Price"; A7:A8)
returns **5625**
(The population variance of the prices that are lower than 400)
#end_function

#begin_function
dvars(input_ranges; field; critera_range)
Returns the sample variance of the cells in a column that matches a criteria.
input_ranges is the data that is used.
field is the name of the field that should be calculated
criteria_range is a range that stores the search criteria.
Also see **davg()**, **dcount()**, **dget()**, **dmax()**, **dmin()**, **dstd()**, **dstds()**, **dsum()**, **dvar()** and **dvars()**
Ex:
Assume the range A1:C5 looks like this:

Item	**No**	**Price**
Car	**1092**	**500**
Truck	**1082**	**850**
Bike	**1072**	**200**
MC	**1062**	**350**

Assume the range A7:A8 looks like this:
Price
<400

dvars(A1:C5; "Price"; A7:A8)
returns **11250**
(The sample variance of the prices that are lower than 400)
#end_function

#begin_function
edit_clear_cells([range])
Removes the values in the cells in **range**. If **range** is omitted, then the current selection will be cleared.
Equivalent to choosing *Clear* in the menu *Cells*.
Note: To make the changes visible, you should use the function ***recalculate().***
#end_function

#begin_function
edit_copy_cells([range])
Copies the cells in **range** to the pasteboard. If **range** is omitted, then the current selection will be copied.

Equivalent to choosing *Copy* in the menu *Edit*.
#end_function

#begin_function
edit_copy_cells_quick(destination; [origin])
Copies the cells in the range **origin** to the range **destination.** If **origin** is omitted, then the cells in the pasteboard will be used.
*Note: To make the changes visible, you should use the function **recalculate().***
#end_function

#begin_function
edit_cut_cells([range])
Removes the cells in **range** completely. If **range** is omitted, then the current selection will be cut.
Equivalent to choosing *Cut* in the menu *Edit*.
*Note: To make the changes visible, you should use the function **redisplay_windows().***
#end_function

#begin_function
edit_delete_columns([column_range]; [partially])
Deletes the columns that are covered by **column_range.** (Equivalent to choosing *Delete Column* in the menu *Edit).*
If **column_range** is omitted, then the current selection will be used.
If **partially** is *true*, <u>only the cells in **column_range** will be removed</u>. All cells to the right will be shifted to the left. (Equivalent to choosing *Delete Cells* in the menu *Edit)*
*Note: To make the changes visible, you should use the function **redisplay_windows().***
#end_function

#begin_function
edit_delete_rows([row_range]; [partially])
Deletes the rows that are covered by **row_range.** (Equivalent to choosing *Delete Row* in the menu *Edit).*
If **row_range** is omitted, then the current selection will be used.

If **partially** is *true*, only the cells in **row_range** will be removed. All cells below will be shifted upwards.
Note: To make the changes visible, you should use the function ***redisplay_windows().***
#end_function

#begin_function
edit_paste_cells([range])
Pastes the cells in the pasteboard to **range** in the worksheet. If **range** is larger than one cell, then the contents of the pasteboard will be repeated the fill the range. If **range** is omitted, then the current selection will be used.
Equivalent to choosing *Paste Cells* in the menu *Edit*.
Note: To make the changes visible, you should use the function ***recalculate().***
#end_function

#begin_function
firstcell()
Selects the upper left cell of the worksheet (cell **A1**).
Also see **lastcell()**
#end_function

#begin_function
font_bold([range])
Makes the font boldface in all the cells in **range**. If **range** is omitted, then the selected cells will be affected.
Note: To make the changes visible, you should use the function ***recalculate().***
#end_function

#begin_function
font_heavier([range])
Makes the fonts heavier in all the cells in **range**. If **range** is omitted, then the selected cells will be affected.
Note: To make the changes visible, you should use the function ***recalculate().***
#end_function

#begin_function
font_italic([range])
Makes the font italic in all the cells in **range**. If **range** is omitted, then the selected cells will be affected.
*Note: To make the changes visible, you should use the function **recalculate().***
#end_function

#begin_function
font_larger([range])
Makes the fonts larger in all the cells in **range**. If **range** is omitted, then the selected cells will be affected.
*Note: To make the changes visible, you should use the function **recalculate().***
#end_function

#begin_function
font_lighter([range])
Makes the fonts lighter in all the cells in **range**. If **range** is omitted, then the selected cells will be affected.
*Note: To make the changes visible, you should use the function **recalculate().***
#end_function

#begin_function
font_panel()
Opens the font panel. The function returns *true*.
#end_function

#begin_function
font_set([font_name]; [size]; [range])
Sets the fonts in the cells in **range**. The function returns *true* if successful. Otherwise it returns *false*.
*Note: To make the changes visible, you should use the function **recalculate().***
Ex:
font_set("Helvetica-Bold"; 16; A1:F1)
returns **true**
font_set("Courier"; 16; A1:F1)
returns **true**

font_set("Bad-Font"; 16; A1:F1)
returns **false**
#end_function

#begin_function
font_smaller([range])
Makes the fonts smaller in all the cells in **range**. If **range** is omitted, then the selected cells will be affected.
*Note: To make the changes visible, you should use the function **recalculate().***
#end_function

#begin_function
font_unbold([range])
Turns off boldface in the fonts in the cells in **range**. If **range** is omitted, then the selected cells will be affected.
*Note: To make the changes visible, you should use the function **recalculate().***
#end_function

#begin_function
font_unitalic([range])
Turns off italic in the fonts in the cells in **range**. If **range** is omitted, then the selected cells will be affected.
*Note: To make the changes visible, you should use the function **recalculate().***
#end_function

#begin_function
get_next_key([location])
Halts the execution and waits for the next keystroke from the user. Returns the character code for the key. The character code will also be put in the cell **location.**
Also see **look_for_next_key()**
#end_function

#begin_function
grab_shell_output(command_string; target_range)

Evaluates **command_string** in a UNIX shell and returns the output (stdout) in **target_range** in the worksheet.
Ex:
grab_shell_output("date"; A1)
puts **Wed Feb 24 12:21:58 GMT+0100 1993** in cell **A1**
#end_function

#begin_function
hide_columns([range])
Hides the columns that are covered by **range**, i.e the column width is set to 0. If **range** is omitted, then the current selection will be used.
#end_function

#begin_function
hide_questor()
Hides the Questor application in Workspace.
#end_function

#begin_function
hide_rows([range])
Hides the rows that are covered by **range**, i.e the row height is set to 0. If **range** is omitted, then the current selection will be used.
#end_function

#begin_function
hlookup(value; range; offset)
Looks up information in a horizontal table of data on the worksheet. The function looks in the top (index) row of **range** for a value equal to the argument **value**. It then returns the value in the row specified by **offset**.
If there is no match for value in the index row, the function finds the <u>next higher value</u> in the index row and <u>selects the previous value</u>.
Note: The range must be sorted increasingly on the index row.
Also see **vlookup()**
Ex:
Assume the range **A1:C2** looks like this:

```
0       100     200
```

33	**45**	**51**

hlookup(0; A1:C2; 1)

returns **33**

hlookup(50; A1:C2; 1)

returns **33**

hlookup(100; A1:C2; 1)

returns **45**

hlookup(600; A1:C2; 1)

returns **51**

hlookup(-20; A1:C2; 1)

returns **#number_out_of_bounds**

#end_function

#begin_function

index(range; row_offset; col_offset)

Returns the value in the cell in **range** defined by **row_offset** and **col_offset.**

Ex:

Assume the range A1:C5 looks like this:

item	**no**	**price**
a	**111**	**12**
b	**222**	**54**
c	**333**	**23**
d	**444**	**43**

index(A1:C5; 0; 0)

returns **item**

index(A1:C5; 1; 0)

returns **a**

index(A1:C5; 2; 2)

returns **54**

#end_function

#begin_function

info_help()

Equivalent to choosing *Help* in the menu *Info*.

#end_function

#begin_function
info_license()
Equivalent to choosing *License* in the menu *Info*.
#end_function

#begin_function
info_panel()
Equivalent to choosing *Info Panel* in the menu *Info*.
#end_function

#begin_function
info_preferences()
Equivalent to choosing *Preferences* in the menu *Info*.
#end_function

#begin_function
info_release_notes()
Equivalent to choosing *Release Notes* in the menu *Info*.
#end_function

#begin_function
isna(value)
Returns *true* if **value** is *nil*. (Same as **NA** in 1-2-3). Otherwise the function returns *false*.
Ex:
isna(12)
returns **false**
isna(nil)
returns **true**
#end_function

#begin_function
isrange(value)
Same as **is_range()**

Used for 1-2-3 function compatibility only.
#end_function

#begin_function
is_name(value)
Returns *true* if value is a *name* in the worksheet. Otherwise the function returns *false*.
Names are defined in the **names aspect** of the **document inspector**.
#end_function

#begin_function
is_range(value)
Returns *true* if **value** is ***a range***. Otherwise the function returns *false*.
Ex:
isrange(12)
returns **false**
isrange(A1:B5)
returns **true**
#end_function

#begin_function
lastcell()
Selects the lower right cell of the <u>used part</u> of the worksheet.
Also see **firstcell()**
#end_function

#begin_function
look_for_next_key([location])
Checks the type-ahead buffer to see if it contains any characters and places the first found (if any) in **location**.
Also see **get_next_key()**
#end_function

#begin_function
n(range)

Returns the number value of the top left cell in **range**. If the cell contains a string value, the function returns 0.
#end_function

#begin_function
na()
Always returns *nil*. *nil* is the same as **NA** in 1-2-3.
#end_function

#begin_function
print_page_layout()
Equivalent to choosing *Page Layout* in the menu *Print*.
#end_function

#begin_function
print_print()
Equivalent to choosing *Print* in the menu *Print*.
#end_function

#begin_function
print_report_layout()
Equivalent to choosing *Report Layout* in the menu *Print*.
#end_function

#begin_function
print_status(string)
Displays **string** in the **information field** in the lower left corner of the worksheet window.
Note: Questor will continue to display information messages in the information field, so the string will eventually be overwritten.
#end_function

#begin_function
queries_fetch_data()

Executes all queries in the document.
Also see **query_fetch_data()**
#end_function

#begin_function
query_fetch_data(query)
Executes a specific query in the document.
query is the name or number of the query.
Also see **queries_fetch_data()**
#end_function

#begin_function
query_last_row(query)
Returns the last row that the query wrote data into.
#end_function

#begin_function
query_set_qualifier(query; qualifier_string)
Sets the qualifier for a query.
query is the name or number of the query.
qualifier_string is the qualifier.
Returns *true* if successful.
*Note: This can also be specified in the **Qualifier aspect** of the **query inspector**.*
Ex:
query_set_qualifier(1; "name = \"John\"")
query_set_qualifier("my_query"; "name = \"John\" and age > 25")

Note the backslash (\) that is used to protect the quotation marks in the qualifier strings.
#end_function

#begin_function
query_set_target(query; target_range)
Sets the target range for a query.
query is the name or number of the query.
target_range is the range where the result from the query should appear in the

worksheet.
Returns *true* if successful.
Note: This can also be specified in the ***Output Range aspect*** *of the* ***query inspector****.*
Ex:
query_set_target(1; B2)
query_set_target("my_query"; A1:D23)
#end_function

#begin_function
quit_questor([discard_changes])
Quits the Questor application. If **discard_changes** is *true*, then Questor will close without a panel asking you to save changed documents.
#end_function

#begin_function
recalculate()
Recalculates and updates the worksheet on the screen. It is the same as a **calculate()** followed by a **redisplay_changed_cells()**.
Note: This function should always be called after changing cell values or cell formatting.
Also see **calculate()** and **redisplay_changed_cells()**
#end_function

#begin_function
redisplay_changed_cells()
Redisplays the changed cells only in the worksheet. This function should be called after a **calculate()**
Also see **calculate()** and **recalculate()**
#end_function

#begin_function
redisplay_windows()
Redisplays all worksheet windows completely. Usually it is much faster to use **redisplay_changed_cells()**.
Also see **redisplay_changed_cells()**

#end_function

#begin_function
rows(range)
Returns the number of rows in **range**.
Also see **cols()**
Ex:
rows(A1:C5)
returns **5**
#end_function

#begin_function
s(range)
Returns the string value of the top left cell in **range**. If the cell contains a number value, the function returns the *empty string* "".
#end_function

#begin_function
scroll_columns([amount])
Scrolls the current worksheet window **amount** number of columns. If **amount** is omitted, the worksheet window will be scrolled 1 column. The function returns *true* if successful.
#end_function

#begin_function
scroll_rows([amount])
Scrolls the current worksheet window **amount** number of rows. If **amount** is omitted, the worksheet window will be scrolled 1 row. The function returns *true* if successful.
#end_function

#begin_function
scroll_to_cell(position)
Scrolls the current worksheet window so that the cell **position** scrolls to the upper left corner of the window. The function returns *true* if successful.

Ex:
scroll_to_cell(B2)
#end_function

#begin_function
scroll_to_column(number)
Scrolls the current worksheet window so that the column **number** scrolls to the left side of the window. You can also specify the column as a cell position. The function returns *true* if successful.
Ex:
scroll_to_column(3)
scroll_to_column(B2)
#end_function

#begin_function
scroll_to_row(number)
Scrolls the current worksheet window so that the row **number** scrolls to the top of the window. You can also specify the row as a cell position. The function returns *true* if successful.
Ex:
scroll_to_row(3)
scroll_to_row(B2)
#end_function

#begin_function
selection_down([steps])
Moves the current selection the specified number of **steps** rows down. If **steps** is omitted, then the selection will be moved one row.
#end_function

#begin_function
selection_home()
Selects the upper left cell of the worksheet (cell **A1**).
#end_function

#begin_function
selection_left([steps])
Moves the current selection the specified number of **steps** columns to the left. If **steps** is omitted, then the selection will be moved one column.
#end_function

#begin_function
selection_right([steps])
Moves the current selection the specified number of **steps** columns to the right. If **steps** is omitted, then the selection will be moved one column.
#end_function

#begin_function
selection_up([steps])
Moves the current selection the specified number of **steps** rows up. If **steps** is omitted, then the selection will be moved one row.
#end_function

#begin_function
select_active_cell(location; [index])
Positions the active cell within the current selection. If the current selection consists of more than one range, and they overlap, then **index** specifies which of the ranges that should be used.
The function returns *true* if successful
Ex:
select_range(A1:B3)
select_range_append(B3:C6)
select_active_cell(B3; 1)
selects cell **B3** in range **A1:B3**
select_active_cell(B3; 2)
selects cell **B3** in range **B3:C6**
#end_function

#begin_function
select_range(range)
Makes **range** the current selection.

Ex:
select_range(A2:B6)
#end_function

#begin_function
select_range_append(range)
Adds **range** to the current selection.
Ex:
select_range(A2:B6)
select_range_append(C2:D6)
#end_function

#begin_function
select_range_relative([col_offset]; [row_offset]; [sheet_offset])
Selects a range whose corners are the active cell and a cell specified by offsets from the active cell.
sheet_offset is not used in version 1.0
The function returns *true* if successful.
Ex:
Assume the active cell is **B2**
select_range_relative(2; 2)
will select the range **B2:D4**
#end_function

#begin_function
select_range_remove([index])
Removes a specified range from the current selection.
#end_function

#begin_function
select_range_reshape(location; [index])
Moves a specified range in the current selection to **location.**
#end_function

#begin_function

send_range(host; application; range)
Sends a **range** of data to another **application.** The application must have a Listener object that understands the Objective-C message

```
- (int)questorData: (char *)buf
        len:(int)len
        fromRow: (int)fromRow
        fromCol: (int)fromCol
        toRow: (int)toRow
        toCol: (int)toCol
        sheet: (char *)aPath
        ok: (int *)ok;
```

host is the machine that runs the application. host should be *nil* if the application runs on the same machine.
application is the name of the application.
range is the cell range that contains the data.
The function returns **0** if successful. Otherwise the function returns **-1.**
Please refer to the printed documentation about the Questor API for details.
Ex:
send_range(nil; "My_App", A1:B4)
send_range("next2"; "My_App", A1:B4)
#end_function

#begin_function
set_trace_granularity(granularity)
Specifies the granularity that should be used by the **Macro Tracer.**
granularity can be:
0 - trace only subroutine calls
1 - trace each subroutine row
2 - trace each element on each row
#end_function

#begin_function
set_trace_mode(mode)
Specifies the mode of the **Macro Tracer.**
mode can be:
0 - disabled
1 - trace enabled

2 - step enabled
#end_function

#begin_function
set_value_at(row; col; value)
Puts **value** in the cell specified by **row** and **col.** To make the new value visible and to recalculate the worksheet, you should use the function **recalculate().** For best performance, only use the function **recalculate()** once after several calls to **set_value_at().**
Returns **value** if successful. Otherwise the function returns *nil.*
Note: If you want to enter a formula in a cell, you should use the function ***cell_enter().***
Also see **cell_enter()** and **recalculate()**
Ex:
set_value_at(1; 1; 123)
puts **123** in cell **A1**
set_value_at(2; 2; "a string")
puts **a string** in cell **B2**
#end_function

#begin_function
sheet_fill([output_range]; [start]; [step]; [stop]; [units]; [by_columns])
Fills **output_range** with values.
To fill a range, you use three values: **start**, **step** and **stop**.
You can specify **now()** or **today()** as the start value to start filling from the current time or date.
- If **step** is not zero, the **stop** value is ignored. The range will be filled with values starting with the **start** value and using the **step** value to generate the next value.
- If **step** is zero, the range will be filled with values starting with the **start** value, stopping with the **stop** value, and automatically generating a step value.
units specifies the type of fill:
- **"linear"** will fill the range with values:
        start, start + 1 * step, start + 2 * step etc.
- **"geometric"** will fill the range with values:
        start, start * step, start * step ^ 2, start * step ^ 3 etc.
- **"seconds"** should be used if you enter a time as a start value. The step value will then represent seconds. The stop value is ignored.

- **"minutes"** should be used if you enter a time as a start value. The step value will then represent minutes. The stop value is ignored.
- **"hours"** should be used if you enter a time as a start value. The step value will then represent hours. The stop value is ignored.
- **"days"** should be used if you enter a date as a start value. The step value will then represent days. The stop value is ignored.
- **"weeks"** should be used if you enter a date as a start value. The step value will then represent weeks. The stop value is ignored.
- **"months"** should be used if you enter a date as a start value. The step value will then represent months. The stop value is ignored.
- **"years"** should be used if you enter a date as a start value. The step value will then represent years. The stop value is ignored.
- **"random"** will fill the selected range with random values between the start and the stop value. The step value is ignored.

by_columns is used to define how the data should be filled: if it is true, the fill will be done by columns.

Note: To make the changes visible in the worksheet, you must use the recalculate() function.

Ex:

sheet_fill(A1:A4; 93-01-01; 3; 0; "months"; true)
will put the following in the range **A1:A4**
01-Jan-93
01-Apr-93
01-Jul-93
01-Oct-93

sheet_fill(A1:A4; 0; 0; 1200; "linear"; true)
will put the following in the range **A1:A4**
0.00
400.00
800.00
1 200.00
#end_function

#begin_function
sheet_height()
Returns the number of rows of the worksheet.
#end_function

#begin_function
sheet_width()
Returns the number of columns of the worksheet.
#end_function

#begin_function
sumproduct(args...)
Multiplies the values the values in a number of ranges and sums all the products. All the ranges must be the same size.
Ex:
Assume the range A1:B2 looks like this:

```
0.00    1.00
2.00    3.00
```

Assume the range C1:C2 looks like this:

```
0.00    1.00
2.00    3.00
```

Assume the range E1:E2 looks like this:

```
0.00    1.00
2.00    3.00
```

sumproduct(A1:B2; C1:D2; E1:F2)
returns **36**

*This is the same as 0*0*0 + 1*1*1 + 2*2*2 + 3*3*3*
#end_function

#begin_function
text_align(style; [data_range])
Sets the text alignment of the cells in data_range to style, that can be one of the following:
"left" - left aligned
"centered" - centered aligned
"right" - right aligned
"smart" - smart aligned (numbers right, strings left and dates centered)
If **data_range** is omitted, then the current selection will be used.

The function always returns *true*.
Note: To make the changes visible in the worksheet, you must use the recalculate() function.
Ex:
text_align("centered"; A1:B3)
#end_function

#begin_function
this_cell()
Returns the address of the cell that contains the function call.
#end_function

#begin_function
this_column()
Returns the column number of the cell that contains the function call.
#end_function

#begin_function
this_row()
Returns the row number of the cell that contains the function call.
#end_function

#begin_function
tools_colors()
Equivalent to choosing *Colors* in the menu *Tools*.
#end_function

#begin_function
tools_console()
Equivalent to choosing *Console* in the menu *Tools*.
#end_function

#begin_function
tools_databases()

Equivalent to choosing *Databases* in the menu *Tools*.
#end_function

#begin_function
tools_input()
Equivalent to choosing *Input* in the menu *Tools*.
#end_function

#begin_function
tools_inspector()
Equivalent to choosing *Inspector* in the menu *Tools*.
#end_function

#begin_function
tools_macro_tracer()
Equivalent to choosing *Macro Tracer* in the menu *Tools*.
#end_function

#begin_function
tools_toolbox()
Equivalent to choosing *ToolBox* in the menu *Tools*.
#end_function

#begin_function
value_at(row; col)
Returns the value in the cell specified by **row** and **column**. If the cell does not exist, then it returns *nil*.
#end_function

#begin_function
vlookup(value; range; offset)
Looks up information in a vertical table of data on the worksheet. The function looks in the left (index) column of **range** for a value equal to the argument **value**. It then returns the value in the column specified by **offset**.

If there is no match for value in the index column, the function finds the next higher value in the index column and selects the previous value.
Note: The range must be sorted increasingly on the index column.
Also see **hlookup()**
Ex:
Assume the range **A1:B5** looks like this:

0	**25**
100	**33**
200	**45**
300	**51**
400	**57**

vlookup(0; A1:B5; 1)
returns **25**
vlookup(50; A1:B5; 1)
returns **25**
vlookup(100; A1:B5; 1)
returns **33**
vlookup(600; A1:B5; 1)
returns **57**
vlookup(-20; A1:B5; 1)
returns **#number_out_of_bounds**
#end_function

#begin_function
windows_arrange()
Equivalent to choosing *Arrange in Front* in the menu *Windows*.
#end_function

#begin_function
windows_close([name])
Closes the current worksheet window. Equivalent to choosing *Close Window* in the menu *Windows*.
The **name** argument is ignored in version 1.0
#end_function

#begin_function

windows_miniaturize([name])
Miniaturizes the current worksheet window. Equivalent to choosing *Miniaturize Window* in the menu *Windows*.
The **name** argument is ignored in version 1.0
#end_function

#begin_function
windows_open([name])
Makes the current worksheet window the key (topmost) window.
The **name** argument is ignored in version 1.0
#end_function

#begin_function
windows_split()
Splits the key worksheet window into two windows. Equivalent to choosing *Split Window* in the menu *Windows*.
#end_function

#begin_function
windows_tile()
Tiles all the worksheet windows to fill the screen. Equivalent to choosing *Tile* in the menu *Windows*.
#end_function

#begin_function
@(location)
Returns the value of the cell that is referred to by the contents of the cell **location.**
Also see **coord()**
Ex:
Assume cell **A1** contains the value **123:**

B1
returns **a1**
@(B1)
returns **123**

```
@(r1c2)
returns 123
#end_function
```