Built-in Macro Functions

A

abs(n)
Returns the absolute value of *n*.

acos(n)
Returns the inverse cosine (in radians) of *n*.

Array Functions
These functions operate on arrays. Refer to the ArrayFunctions [https://imagej.nih.gov/ij/macros/examples/ArrayFunctions.txt] macro for examples.

Array.concat(array1,array2) – Returns a new array created by joining two or more arrays or values (examples [https://imagej.nih.gov/ij/macros/examples/ArrayConcatExamples.txt]).
Array.copy(array) – Returns a copy of *array*.
Array.fill(array, value) – Assigns the specified numeric value to each element of *array*.
Array.findMaxima(array, tolerance) – Returns an array holding the peak positions (sorted with descending strength). 'Tolerance' is the minimum amplitude difference needed to separate two peaks. With v1.51n and later, there is an optional 'edgeMode' argument: 0=include edges, 1=exclude edges(default), 2=circular array. Examples [https://imagej.nih.gov/ij/macros/examples/FindMaxima1D.txt] .
Array.findMinima(array, tolerance) – Returns an array holding the minima positions.
Array.fourier(array, windowType) – Calculates and returns the Fourier amplitudes of *array*. *WindowType* can be "none", "Hamming", "Hann", or "flat-top", or may be omitted (meaning "none"). See the TestArrayFourier [https://imagej.nih.gov/ij/macros/examples/TestArrayFourier.txt] macro for an example and more documentation. Requires 1.49i.
Array.getSequence(n) – Returns an array containing the numeric sequence 0,1,2...n-1. Requires 1.49u.
Array.getStatistics(array, min, max, mean, stdDev) – Returns the *min*, *max*, *mean*, and *stdDev* of *array*, which must contain all numbers.
Array.print(array) – Prints the array on a single line.
Array.rankPositions(array) – Returns, as an array, the rank position indexes of *array*, starting with the index of the smallest value (example [https://imagej.nih.gov/ij/macros/examples/ArraySortingDemo.txt]).
Array.resample(array,len) – Returns an array which is linearly resampled to a different length.
Array.reverse(array) – Reverses (inverts) the order of the elements in *array*.
Array.show(array) – Displays the contents of *array* in a window.
Array.show("title", array1, array2, ...) – Displays one or more arrays in a Results window (examples [https://imagej.nih.gov/ij/macros/examples/ShowArrayDemo.txt]). If *title* (optional) is "Results", the window will be the active Results window, otherwise, it will be a dormant Results window (see also **IJ.renameResults**). If *title* ends with " (indexes)", a 0-based Index column is shown. If *title* ends with "(row numbers)", the row number column is shown.
Array.slice(array,start,end) – Extracts a part of an array and returns it. (examples [https://imagej.nih.gov/ij/macros/examples/ArraySliceExamples.txt]).
Array.sort(array) – Sorts *array*, which must contain all numbers or all strings. String sorts are case-insensitive in v1.44i or later.
Array.trim(array, n) – Returns an array that contains the first *n* elements of *array*.
Array.rotate(array, d) – Rotates the array elements by 'd' steps (positive 'd' = rotate right). Requires 1.51n. Examples [https://imagej.nih.gov/ij/macros/examples/RotateArray.txt] .
Array.getVertexAngles(xArr, yArr, arm) – From a closed contour given by 'xArr', 'yArr', an array is returned holding vertex angles in degrees (straight=0, convex = positive if contour is clockwise). Set 'arm'=1 to calculate the angle from the closest vertex points left and right, or use arm>1 for more distant neighbours and

smoother results. Requires 1.51n. Examples [https://imagej.nih.gov/ij/macros/examples/VertexAngles.txt] .

asin(n)
Returns the inverse sine (in radians) of *n*.

atan(n)
Calculates the inverse tangent (arctangent) of *n*. Returns a value in the range -PI/2 through PI/2.

atan2(y, x)
Calculates the inverse tangent of *y/x* and returns an angle in the range -PI to PI, using the signs of the arguments to determine the quadrant. Multiply the result by 180/PI to convert to degrees.

autoUpdate(boolean)
If *boolean* is true, the display is refreshed each time lineTo(), drawLine(), drawString(), etc. are called, otherwise, the display is refreshed only when updateDisplay() is called or when the macro terminates.

B

beep()
Emits an audible beep.

bitDepth()
Returns the bit depth of the active image: 8, 16, 24 (RGB) or 32 (float).

C

calibrate(value)
Uses the calibration function of the active image to convert a raw pixel value to a density calibrated value. The argument must be an integer in the range 0-255 (for 8-bit images) or 0-65535 (for 16-bit images). Returns the same value if the active image does not have a calibration function.

call("class.method", arg1, arg2, ...)
Calls a public static method in a Java class, passing an arbitrary number of string arguments, and returning a string. Refer to the CallJavaDemo [https://imagej.nih.gov/ij/macros/CallJavaDemo.txt] macro and the ImpProps [https://imagej.nih.gov/ij/plugins/imp-props.html] plugin for examples. Note that the call() function does not work when ImageJ is running as an unsigned applet.

changeValues(v1, v2, v3)
Changes pixels in the image or selection that have a value in the range *v1-v2* to *v3*. For example, *changeValues(0,5,5)* changes all pixels less than 5 to 5, and *changeValues(0x0000ff,0x0000ff,0xff0000)* changes all blue pixels in an RGB image to red. In ImageJ 1.52d or later, use changeValues(NaN,NaN,value) to replaces NaN values.

charCodeAt(string, index)
Returns the Unicode value of the character at the specified index in *string*. Index values can range from 0 to lengthOf(*string*). Use the fromCharCode() function to convert one or more Unicode characters to a string.

close()
Closes the active image. This function has the advantage of not closing the "Log" or "Results" window when you meant to close the active image. Use *run("Close")* to close non-image windows.

close(pattern)
Closes windows whose title matches 'pattern', which can contain the wildcard characters '*' (matches any character sequence) and '?' (matches single character). For example, close("Histo*") could be used to dispose all histogram windows. Non-image windows like "Roi Manager" have to be specified without wildcards. For text windows, wildcards are allowed if 'pattern' ends with ".txt", ".ijm", ".js" etc. Use close("*") to close all image windows. Use close(pattern, "keep") to not close text or image windows with changes. If 'pattern' is "\\Others", all images except the front image are closed. The most recent macro window is never closed.

close("*")
Closes all image windows.

close("\\Others")
Closes all images except for the front image.

cos(angle)
Returns the cosine of an angle (in radians).

D

d2s(n, decimalPlaces)
Converts the number *n* into a string using the specified number of decimal places. Uses scientific notation if 'decimalPlaces is negative. Note that d2s stands for "double to string".

Dialog.create("Title")
Creates a dialog box with the specified title. Call *Dialog.addString()*, *Dialog.addNumber(),* etc. to add components to the dialog. Call *Dialog.show()* to display the dialog and *Dialog.getString()*, *Dialog.getNumber()*, etc. to retrieve the values entered by the user. Refer to the DialogDemo [https://imagej.nih.gov/ij/macros/DialogDemo.txt] macro for an example.

Dialog.addMessage(string) - Adds a message to the dialog. The message can be broken into multiple lines by inserting new line characters ("\n") into the string.
Dialog.addString(label, initialText) - Adds a text field to the dialog, using the specified label and initial text.
Dialog.addString(label, initialText, columns) - Adds a text field to the dialog, where *columns* specifies the field width in characters.
Dialog.addNumber(label, default) - Adds a numeric field to the dialog, using the specified label and default value.
Dialog.addNumber(label, default, decimalPlaces, columns, units) - Adds a numeric field, using the specified label and default value. *DecimalPlaces* specifies the number of digits to right of decimal point, *columns* specifies the the field width in characters and *units* is a string that is displayed to the right of the field.
Dialog.addSlider(label, min, max, default) - Adds a slider controlled numeric field to the dialog, using the specified label, and min, max and default values (example [https://imagej.nih.gov/ij/macros/examples/SliderDemo.txt]). Values with decimal points are used when (max-min)<=5 and min, max or default are non-integer.
Dialog.addCheckbox(label, default) - Adds a checkbox to the dialog, using the specified label and default state (true or false).
Dialog.addCheckboxGroup(rows, columns, labels, defaults) - Adds a *rows*x*columns* grid of checkboxes to the dialog, using the specified labels and default states (example [https://imagej.nih.gov/ij/macros/AddCheckboxGroupDemo.txt]).
Dialog.addRadioButtonGroup(label, items, rows, columns, default) - Adds a group of radio buttons to the dialog, where 'label' is the group label, 'items' is an array containing the button labels, 'rows' and 'columns' specify the grid size, and 'default' is the label of the default button. (example [https://imagej.nih.gov/ij/macros/examples/RadioButtonDemo.txt]).
Dialog.addChoice(label, items) - Adds a popup menu to the dialog, where *items* is a string array containing the menu items.
Dialog.addChoice(label, items, default) - Adds a popup menu, where *items* is a string array containing the choices and *default* is the default choice.
Dialog.addHelp(url) - Adds a "Help" button that opens the specified URL in the default browser. This can be used to supply a help page for this dialog or macro. With v1.46b or later, displays an HTML formatted message if 'url' starts with "<html>" (example [https://imagej.nih.gov/ij/macros/examples/DialogWithHelp.txt]).
Dialog.addToSameRow() - Makes the next item added appear on the same row as the previous item. May be used for addNumericField, addSlider, addChoice, addCheckbox, addStringField, addMessage, addPanel, and before the showDialog() method. In the latter case, the buttons appear to the right of the previous item. Note that *addMessage* (and *addString* if a width of more than 8 is specified) use the remaining width, so it must be the last item of a row. Requires 1.51r.
Dialog.setInsets(top, left, bottom) - Overrides the default insets (margins) used for the next component added to the dialog.
Default insets:
addMessage: 0,20,0 (empty string) or 10,20,0
addCheckbox: 15,20,0 (first checkbox) or 0,20,0
addCheckboxGroup: 10,0,0
addNumericField: 5,0,3 (first field) or 0,0,3
addStringField: 5,0,5 (first field) or 0,0,5
addChoice: 5,0,5 (first field) or 0,0,5
Dialog.setLocation(x,y) - Sets the screen location where this dialog will be displayed.

Dialog.show() – Displays the dialog and waits until the user clicks "OK" or "Cancel". The macro terminates if the user clicks "Cancel".
Dialog.getString() – Returns a string containing the contents of the next text field.
Dialog.getNumber() – Returns the contents of the next numeric field.
Dialog.getCheckbox() – Returns the state (true or false) of the next checkbox.
Dialog.getChoice() – Returns the selected item (a string) from the next popup menu.
Dialog.getRadioButton() – Returns the selected item (a string) from the next radio button group.

doCommand("Command")
Runs an ImageJ menu command in a separate thread and returns immediately. As an example, *doCommand("Start Animation")* starts animating the current stack in a separate thread and the macro continues to execute. Use *run("Start Animation")* and the macro hangs until the user stops the animation.

doWand(x, y)
Equivalent to clicking on the current image at (x,y) with the wand tool. Note that some objects, especially one pixel wide lines, may not be reliably traced unless they have been thresholded (highlighted in red) using **setThreshold**.

doWand(x, y, tolerance, mode)
Traces the boundary of the area with pixel values within 'tolerance' of the value of the pixel at (x,y). 'mode' can be "4-connected", "8-connected" or "Legacy". "Legacy" is for compatibility with previous versions of ImageJ; it is ignored if 'tolerance' > 0.

drawLine(x1, y1, x2, y2)
Draws a line between (x1, y1) and (x2, y2). Use setColor() to specify the color of the line and setLineWidth() to vary the line width. See also: **Overlay.drawLine**.

drawOval(x, y, width, height)
Draws the outline of an oval using the current color and line width. See also: **fillOval**, **setColor**, **setLineWidth**, **autoUpdate** and **Overlay.drawEllipse**.

drawRect(x, y, width, height)
Draws the outline of a rectangle using the current color and line width. See also: **fillRect**, **setColor**, **setLineWidth**, **autoUpdate** and **Overlay.drawRect**

drawString("text", x, y)
Draws text at the specified location. Call **setFont()** to specify the font. Call **setJustification()** to have the text centered or right justified. Call **getStringWidth()** to get the width of the text in pixels. Refer to the TextDemo [https://imagej.nih.gov/ij/macros/TextDemo.txt] macro for examples and to DrawTextWithBackground [https://imagej.nih.gov/ij/macros/examples/DrawTextWithBackground.txt] to see how to draw text with a background.

drawString("text", x, y, background)
Draws text at the specified location with a filled background (examples [https://imagej.nih.gov/ij/macros/examples/DrawTextWithBackground.txt]).

dump()
Writes the contents of the symbol table, the tokenized macro code and the variable stack to the "Log" window.

E

endsWith(string, suffix)
Returns *true* (1) if *string* ends with *suffix*. See also: **startsWith**, **indexOf**, **substring**, **matches**.

eval(macro)
Evaluates (runs) one or more lines of macro code. An optional second argument can be used to pass a string to the macro being evaluated. See also: EvalDemo [https://imagej.nih.gov/ij/macros/EvalDemo.txt] macro and **runMacro** function.

eval("script", javascript)
Evaluates the JavaScript [https://imagej.nih.gov/ij/developer/javascript.html] code contained in the string *javascript*, for example *eval("script","IJ.getInstance().setAlwaysOnTop(true);")*. See also: **runMacro(path,arg)**.

eval("bsh", script)
Evaluates the BeanShell [https://imagej.nih.gov/ij/plugins/bsh/index.html] code contained in the string *script*.

eval("python", script)
Evaluates the Python [https://imagej.nih.gov/ij/plugins/jython/index.html] code contained in the string *script*.

exec(string or strings)
Executes a native command and returns the output of that command as a string. Also opens Web pages in the default browser and documents in other applications (e.g., Excel). With commands with multiple arguments, each argument should be passed as a separate string. For example

```
exec("open", "/Users/wayne/test.jpg", "-a", "/Applications/Gimp.app");
```

Refer to the ExecExamples [https://imagej.nih.gov/ij/macros/ExecExamples.txt] macro for examples.

exit() or exit("error message")
Terminates execution of the macro and, optionally, displays an error message.

exp(n)
Returns the exponential number e (i.e., 2.718...) raised to the power of *n*.

Ext (Macro Extension) Functions
These are functions that have been added to the macro language by plugins using the MacroExtension interface. The Image5D_Extensions [https://imagej.nih.gov/ij/plugins/5d-extensions.html] plugin, for example, adds functions that work with Image5D. The Serial Macro Extensions [http://imagejdocu.tudor.lu/doku.php?id=plugin:utilities:serial_macro_extensions:start] plugin adds functions, such as Ext.open("COM8",9600,"") and Ext.write("a"), that talk to serial devices.

F

File Functions
These functions allow you to get information about a file, read or write a text file, create a directory, or to delete, rename or move a file or directory. Note that these functions return a string, with the exception of *File.length*, *File.exists*, *File.isDirectory*, *File.rename* and *File.delete* when used in an assignment statement, for example "length=File.length(path)". The FileDemo [https://imagej.nih.gov/ij/macros/FileDemo.txt] macro demonstrates how to use these functions. See also: **getDirectory** and **getFileList**.

File.append(string, path) – Appends *string* to the end of the specified file.
File.close(f) – Closes the specified file, which must have been opened using File.open().
File.copy(path1, path2) – Copies a file.
File.dateLastModified(path) – Returns the date and time the specified file was last modified.
File.delete(path) – Deletes the specified file or directory. With v1.41e or later, returns "1" (true) if the file or directory was successfully deleted. If the file is a directory, it must be empty. The file must be in the user's home directory, the ImageJ directory or the temp directory.
File.directory – The directory path of the last file opened using a file open dialog, a file save dialog, drag and drop, **open(path)** or **runMacro(path)**.
File.exists(path) – Returns "1" (true) if the specified file exists.
File.getName(path) – Returns the last name in *path*'s name sequence.
File.getParent(path) – Returns the parent of the file specified by *path*.
File.isDirectory(path) – Returns "1" (true) if the specified file is a directory.
File.lastModified(path) – Returns the time the specified file was last modified as the number of milliseconds since January 1, 1970.
File.length(path) – Returns the length in bytes of the specified file as a string, or as a number when used in an assignment statement, for example "length=File.length(path)".
File.makeDirectory(path) – Creates a directory.
File.name – The name of the last file opened using a file open dialog, a file save dialog, drag and drop, or the **open(path)** function.
File.nameWithoutExtension – The name of the last file opened with the extension removed.
File.open(path) – Creates a new text file and returns a file variable that refers to it. To write to the file, pass the file variable to the **print** function. Displays a file save dialog box if *path* is an empty string. The file is closed when the macro exits.

Currently, only one file can be open at a time. For an example, refer to the SaveTextFileDemo [https://imagej.nih.gov/ij/macros/SaveTextFileDemo.txt] macro.
File.openAsString(path) - Opens a text file and returns the contents as a string. Displays a file open dialog box if *path* is an empty string. Use *lines=split(str,"\n")* to convert the string to an array of lines.
File.openAsRawString(path) - Opens a file and returns up to the first 5,000 bytes as a string. Returns all the bytes in the file if the name ends with ".txt". Refer to the First10Bytes [https://imagej.nih.gov/ij/macros/First10Bytes.txt] and ZapGremlins [https://imagej.nih.gov/ij/macros/ZapGremlins.txt] macros for examples.
File.openAsRawString(path, count) - Opens a file and returns up to the first *count* bytes as a string.
File.openUrlAsString(url) - Opens a URL and returns the contents as a string. Returns an emptly string if the host or file cannot be found. With v1.41i and later, returns "<Error: message>" if there any error, including host or file not found.
File.openDialog(title) - Displays a file open dialog and returns the path to the file choosen by the user (example [https://imagej.nih.gov/ij/macros/OpenDialogDemo.txt]). The macro exits if the user cancels the dialog.
File.rename(path1, path2) - Renames, or moves, a file or directory. Returns "1" (true) if successful.
File.saveString(string, path) - Saves *string* as a file.
File.separator - Returns the file name separator character ("/" or "\").

fill()
Fills the image or selection with the current drawing color.

fillOval(x, y, width, height)
Fills an oval bounded by the specified rectangle with the current drawing color. See also: **drawOval**, **setColor**, **autoUpdate**.

fillRect(x, y, width, height)
Fills the specified rectangle with the current drawing color. See also: **drawRect**, **setColor**, **autoUpdate**.

Fit Functions
These functions do curve fitting. The CurveFittingDemo [https://imagej.nih.gov/ij/macros/examples/CurveFittingDemo.txt] macro demonstrates how to use them.

Fit.doFit(equation, xpoints, ypoints) - Fits the specified equation to the points defined by *xpoints*, *ypoints*. *Equation* can be either the equation name or an index. The equation names are shown in the drop down menu in the *Analyze>Tools>Curve Fitting* window. With ImageJ 1.42f or later, *equation* can be a string containing a user-defined equation (example [https://imagej.nih.gov/ij/macros/examples/UserDefinedCurveFits.txt]).
Fit.doFit(equation, xpoints, ypoints, initialGuesses) - Fits the specified equation to the points defined by *xpoints*, *ypoints*, using initial parameter values contained in *initialGuesses*, an array equal in length to the number of parameters in *equation* (example [https://imagej.nih.gov/ij/macros/examples/RodbardSigmoidFit.txt]).
Fit.rSquared - Returns R^2=1-SSE/SSD, where SSE is the sum of the squares of the errors and SSD is the sum of the squares of the deviations about the mean.
Fit.p(index) - Returns the value of the specified parameter.
Fit.nParams - Returns the number of parameters.
Fit.f(x) - Returns the y value at *x* (example [https://imagej.nih.gov/ij/macros/examples/PlotSigmoidDerivatives.txt]).
Fit.nEquations - Returns the number of equations.
Fit.getEquation(index, name, formula) - Gets the name and formula of the specified equation.
Fit.plot - Plots the current curve fit.
Fit.logResults - Causes doFit() to write a description of the curve fitting results to the Log window.
Fit.showDialog - Causes doFit() to display the simplex settings dialog.

floodFill(x, y)
Fills, with the foreground color, pixels that are connected to, and the same color as, the pixel at *(x, y)*. Does 8-connected filling if there is an optional string argument containing "8", for example *floodFill(x, y, "8-connected")*. This function is used to implement the flood fill (paint bucket) [https://imagej.nih.gov/ij/macros/tools/FloodFillTool.txt] macro tool.

floor(n)
Returns the largest value that is not greater than *n* and is equal to an integer. See also: **round**.

fromCharCode(value1,...,valueN)
This function takes one or more Unicode values and returns a string.

G

getArgument()
Returns the string argument passed to macros called by **runMacro(macro, arg)**, **eval(macro)**, *IJ.runMacro(macro, arg)* or *IJ.runMacroFile(path, arg)*.

getBoolean("message")
Displays a dialog box containing the specified message and "Yes", "No" and "Cancel" buttons. Returns *true* (1) if the user clicks "Yes", returns *false* (0) if the user clicks "No" and exits the macro if the user clicks "Cancel".

getBoolean(message, yesLabel, noLabel)
Displays a dialog box containing the specified message and buttons with custom labels.

getBoundingRect(x, y, width, height)
Replace by **getSelectionBounds**.

getCursorLoc(x, y, z, modifiers)
Returns the cursor location in pixels and the mouse event modifier flags. The *z* coordinate is zero for 2D images. For stacks, it is one less than the slice number. Use **toScaled(x,y)** to scale the coordinates. For examples, see the GetCursorLocDemo [https://imagej.nih.gov/ij/macros//GetCursorLocDemo.txt] and the GetCursorLocDemoTool [https://imagej.nih.gov/ij/macros/tools/GetCursorLocDemoTool.txt] macros.

getDateAndTime(year, month, dayOfWeek, dayOfMonth, hour, minute, second, msec)
Returns the current date and time. Note that 'month' and 'dayOfWeek' are zero-based indexes. For an example, refer to the GetDateAndTime [https://imagej.nih.gov/ij/macros/GetDateAndTime.txt] macro. See also: **getTime**.

getDimensions(width, height, channels, slices, frames)
Returns the dimensions of the current image.

getDirectory(string)
Displays a "choose directory" dialog and returns the selected directory, or returns the path to a specified directory, such as "plugins", "home", etc. The returned path ends with a file separator, either "\" (Windows) or "/". Returns an empty string if the specified directory is not found or aborts the macro if the user cancels the "choose directory" dialog box. For examples, see the GetDirectoryDemo [https://imagej.nih.gov/ij/macros/GetDirectoryDemo.txt] and ListFilesRecursively [https://imagej.nih.gov/ij/macros/ListFilesRecursively.txt] macros. See also: **getFileList** and the **File functions**.

getDirectory("Choose a Directory") – Displays a file open dialog, using the argument as a title, and returns the path to the directory selected by the user.
getDirectory("plugins") – Returns the path to the plugins directory.
getDirectory("macros") – Returns the path to the macros directory.
getDirectory("luts") – Returns the path to the luts directory.
getDirectory("image") – Returns the path to the directory that the active image was loaded from.
getDirectory("imagej") – Returns the path to the ImageJ directory.
getDirectory("startup") – Returns the path to the directory that ImageJ was launched from.
getDirectory("home") – Returns the path to users home directory.
getDirectory("temp") – Returns the path to the temporary directory (/tmp on Linux and Mac OS X).

getDisplayedArea(x, y, width, height)
Returns the pixel coordinates of the actual displayed area of the image canvas. For an example, see the Pixel Sampler Tool [https://imagej.nih.gov/ij/macros/tools/Pixel_Sampler_Tool.txt] .

getFileList(directory)
Returns an array containing the names of the files in the specified directory path. The names

of subdirectories have a "/" appended. For an example, see the ListFilesRecursively [https://imagej.nih.gov/ij/macros/ListFilesRecursively.txt] macro.

getFontList()
Returns an array containing the names of available system fonts (example [https://imagej.nih.gov/ij/macros/Fonts.txt]).

getHeight()
Returns the height in pixels of the current image.

getHistogram(values, counts, nBins[, histMin, histMax])
Returns the histogram of the current image or selection. *Values* is an array that will contain the pixel values for each of the histogram counts (or the bin starts for 16 and 32 bit images), or set this argument to 0. *Counts* is an array that will contain the histogram counts. *nBins* is the number of bins that will be used. It must be 256 for 8 bit and RGB image, or an integer greater than zero for 16 and 32 bit images. With 16-bit images, the *Values* argument is ignored if *nBins* is 65536. With 16-bit and 32-bit images, the histogram range can be specified using optional *histMin* and *histMax* arguments. See also: **getStatistics**, HistogramLister [https://imagej.nih.gov/ij/macros/HistogramLister.txt] , HistogramPlotter [https://imagej.nih.gov/ij/macros/HistogramPlotter.txt] , StackHistogramLister [https://imagej.nih.gov/ij/macros/StackHistogramLister.txt] and CustomHistogram [https://imagej.nih.gov/ij/macros/CustomHistogram.txt] .

getImageID()
Returns the unique ID (a negative number) of the active image. Use the *selectImage(id)*, *isOpen(id)* and *isActive(id)* functions to activate an image or to determine if it is open or active.

getImageInfo()
Returns a string containing the text that would be displayed by the *Image>Show Info* command. To retrieve the contents of a text window, use **getInfo("window.contents")**. For an example, see the ListDicomTags [https://imagej.nih.gov/ij/macros/ListDicomTags.txt] macros. See also: **getMetadata**.

getInfo("command.name")
Returns the name of the most recently invoked command. The names of commands invoked using keyboard shortcuts are preceded by "^" (example [https://imagej.nih.gov/ij/macros/examples/GetCommandNameDemo.txt]).

getInfo(DICOM_TAG)
Returns the value of a DICOM tag in the form "xxxx,xxxx", e.g. getInfo("0008,0060"). Returns an empty string if the current image is not a DICOM or if the tag was not found.

getInfo("font.name")
Returns the name of the current font.

getInfo("image.description")
Returns the TIFF image description tag, or an empty string if this is not a TIFF image or the image description is not available.

getInfo("image.directory")
Returns the directory that the current image was loaded from, or an empty string if the directory is not available.

getInfo("image.filename")
Returns the name of the file that the current image was loaded from, or an empty string if the file name is not available.

getInfo("image.subtitle")
Returns the subtitle of the current image. This is the line of information displayed above the image and below the title bar.

getInfo("log")
Returns the contents of the Log window, or "" if the Log window is not open.

getInfo("macro.filepath")
Returns the filepath of the most recently loaded macro or script.

getInfo("micrometer.abbreviation")
Returns "µm", the abbreviation for micrometer.

getInfo("os.name")
Returns the OS name ("Mac OS X", "Linux" or "Windows").

getInfo("overlay")
Returns information about the current image's overlay.

getInfo("selection.name")
Returns the name of the current selection, or "" if there is no selection or the selection does not have a name. The argument can also be "roi.name".

getInfo("selection.color")
Returns the color of the current selection.

getInfo("slice.label")
Return the label of the current stack slice. This is the string that appears in parentheses in the subtitle, the line of information above the image. Returns an empty string if the current image is not a stack or the current slice does not have a label.

getInfo("threshold.method")
Returns the current thresholding method ("IsoData", "Otsu", etc).

getInfo("threshold.mode")
Returns the current thresholding mode ("Red","B&W" or"Over/Under").

getInfo("window.contents")
If the front window is a text window, returns the contents of that window. If the front window is an image, returns a string containing the text that would be displayed by *Image>Show Info*. Note that **getImageInfo()** is a more reliable way to retrieve information about an image. Use split(getInfo(),'\n') to break the string returned by this function into separate lines. Replaces the getInfo() function.

getInfo("window.title")
Returns the title of the front-most window.

getInfo("window.type")
Returns the type ("Image", "Text", "ResultsTable", "Editor", "Plot", "Histogram", etc.) of the front window.

getInfo(key)
Returns the Java property associated with the specified key (e.g., "java.version", "os.name", "user.home", "user.dir", etc.). Returns an empty string if there is no value associated with the key. See also: **getList("java.properties")**.

getLine(x1, y1, x2, y2, lineWidth)
Returns the starting coordinates, ending coordinates and width of the current straight line selection. The coordinates and line width are in pixels. Sets x1 = -1 if there is no line selection. Refer to the GetLineDemo [https://imagej.nih.gov/ij/macros/GetLineDemo.txt] macro for an example. See also: **makeLine**.

getList("image.titles")
Returns a list (array) of image window titles. For an example, see the DisplayWindowTitles [https://imagej.nih.gov/ij/macros/DisplayWindowTitles.txt] macro.

getList("window.titles")
Returns a list (array) of non-image window titles. For an example, see the DisplayWindowTitles [https://imagej.nih.gov/ij/macros/DisplayWindowTitles.txt] macro.

getList("java.properties")
Returns a list (array) of Java property keys. For an example, see the DisplayJavaProperties [https://imagej.nih.gov/ij/macros/DisplayJavaProperties.txt] macro. See also: **getInfo(key)**.

getList("threshold.methods")
Returns a list of the available automatic thresholding methods (example [https://imagej.nih.gov/ij/macros/examples/AutoThresholdingDemo.txt]).

getList("LUTs")
Returns, as an array, a list of the LUTs in the *Image>Lookup Tables* menu (example [https://imagej.nih.gov/ij/macros/Time-Lapse_Color_Coder.txt]).

getLocationAndSize(x, y, width, height)
Returns the location and size, in screen coordinates, of the active image window. Use **getWidth** and **getHeight** to get the width and height, in image coordinates, of the active image. See also: **setLocation**,

getLut(reds, greens, blues)
Returns three arrays containing the red, green and blue intensity values from the current lookup table. See the LookupTables [https://imagej.nih.gov/ij/macros/LookupTables.txt] macros for examples.

getMetadata("Info")
Returns the metadata (a string) from the "Info" property of the current image. With DICOM images, this is the information (tags) in the DICOM header. See also: **setMetadata**.

getMetadata("Label")
Returns the current slice label. The first line of the this label (up to 60 characters) is display as part of the image subtitle. With DICOM stacks, returns the metadata from the DICOM header. See also: **setMetadata**.

getMinAndMax(min, max)
Returns the minimum and maximum displayed pixel values (display range). See the DisplayRangeMacros [https://imagej.nih.gov/ij/macros/DisplayRangeMacros.txt] for examples.

getNumber("prompt", defaultValue)
Displays a dialog box and returns the number entered by the user. The first argument is the prompting message and the second is the value initially displayed in the dialog. Exits the macro if the user clicks on "Cancel" in the dialog. Returns *defaultValue* if the user enters an invalid number. See also: **Dialog.create**.

getPixel(x, y)
Returns the value of the pixel at *(x,y)*. Note that pixels in RGB images contain red, green and blue components that need to be extracted using shifting and masking. See the Color Picker Tool [https://imagej.nih.gov/ij/macros/tools/ColorPickerTool.txt] macro for an example that shows how to do this.

getPixelSize(unit, pixelWidth, pixelHeight)
Returns the unit of length (as a string) and the pixel dimensions. For an example, see the ShowImageInfo [https://imagej.nih.gov/ij/macros/ShowImageInfo.txt] macro. See also: **getVoxelSize**.

getProfile()
Runs *Analyze>Plot Profile* (without displaying the plot) and returns the intensity values as an array. For an example, see the GetProfileExample [https://imagej.nih.gov/ij/macros/GetProfileExample.txt] macro. See also: **Plot.getValues()**.

getRawStatistics(nPixels, mean, min, max, std, histogram)
This function is similar to **getStatistics** except that the values returned are uncalibrated and the histogram of 16-bit images has a bin width of one and is returned as a *max*+1 element array. For examples, refer to the ShowStatistics [https://imagej.nih.gov/ij/macros/ShowStatistics.txt] macro set. See also: **calibrate** and **List.setMeasurements**

getResult("Column", row)
Returns a measurement from the ImageJ results table or NaN if the specified column is not found. The first argument specifies a column in the table. It must be a "Results" window column label, such as "Area", "Mean" or "Circ.". The second argument specifies the row, where 0<=*row*<nResults. *nResults* is a predefined variable that contains the current measurement count. (Actually, it's a built-in function with the "()" optional.) Omit the second argument and the row defaults to nResults-1 (the last row in the results table). See also: **nResults**, **setResult**, **isNaN**, **getResultLabel**.

getResultString("Column", row)
Returns a string from the ImageJ results table or "null" if the specified column is not found. The first argument specifies a column in the table. The second specifies the row, where 0<=*row*<nResults.

getResultLabel(row)
Returns the label of the specified row in the results table, or an empty string if *Display Label* is not checked in *Analyze>Set Measurements*.

getSelectionBounds(x, y, width, height)
Returns the smallest rectangle that can completely contain the current selection. *x* and *y* are the pixel coordinates of the upper left corner of the rectangle. *width* and *height* are the width and height of the rectangle in pixels. If there is no selection, returns (0, 0, ImageWidth, ImageHeight). See also: **selectionType** and **setSelectionLocation**.

getSelectionCoordinates(xpoints, ypoints)
Returns two arrays containing the X and Y coordinates, in pixels, of the points that define the current selection. See the SelectionCoordinates [https://imagej.nih.gov/ij/macros/SelectionCoordinates.txt] macro for an example. See also: **selectionType**, **getSelectionBounds**.

getSliceNumber()
Returns the number of the currently displayed stack image, an integer between 1 and **nSlices**. Returns 1 if the active image is not a stack. See also: **setSlice**, **Stack.getPosition**.

getStatistics(area, mean, min, max, std, histogram)
Returns the area, average pixel value, minimum pixel value, maximum pixel value, standard deviation of the pixel values and histogram of the active image or selection. The histogram is returned as a 256 element array. For 8-bit and RGB images, the histogram bin width is one. For 16-bit and 32-bit images, the bin width is (*max*-*min*)/256. For examples, refer to the ShowStatistics [https://imagej.nih.gov/ij/macros/ShowStatistics.txt] macro set. Note that trailing arguments can be omitted. For example, you can use *getStatistics(area)*, *getStatistics(area, mean)* or *getStatistics(area, mean, min, max)*. See also: **getRawStatistics** and **List.setMeasurements**

getString("prompt", "default")
Displays a dialog box and returns the string entered by the user. The first argument is the prompting message and the second is the initial string value. Exits the macro if the user clicks on "Cancel" or enters an empty string. See also: **Dialog.create**.

getStringWidth(string)
Returns the width in pixels of the specified string. See also: **setFont**, **drawString**.

getThreshold(lower, upper)
Returns the lower and upper threshold levels. Both variables are set to -1 if the active image is not thresholded. See also: **setThreshold**, **getThreshold**, **resetThreshold**.

getTime()
Returns the current time in milliseconds. The granularity of the time varies considerably from one platform to the next. On Windows NT, 2K, XP it is about 10ms. On other Windows versions it can be as poor as 50ms. On many Unix platforms, including Mac OS X, it actually is 1ms. See also: **getDateAndTime**.

getTitle()
Returns the title of the current image.

getValue("color.foreground")
Returns the current foreground color as a value that can be passed to the **setColor(value)** function. The value returned is the pixel value used by the *Edit>Fill* command and by drawing tools.

getValue("color.background")
Returns the current background color as a value that can be passed to the **setColor(value)** function. The value returned is the pixel value used by the *Edit>Clear* command.

getValue("rgb.foreground")
Returns the current foregound color as an RGB pixel value (example [https://imagej.nih.gov/ij/macros/examples/ForegroundBackgroundColors.txt]).

getValue("rgb.background")
Returns the current backgound color as an RGB pixel value.

getValue("font.size")
Returns the size, in points, of the current font.

getValue("font.height")
Returns the height, in pixels, of the current font.

getValue("selection.width")
Returns the stroke width of the current selection.

getValue("results.count")
Returns the number of lines in the current results table. Unlike **nResults**, works with tables that are not named "Results". Requires 1.49t.

getVoxelSize(width, height, depth, unit)
Returns the voxel size and unit of length ("pixel", "mm", etc.) of the current image or stack. See also: **getPixelSize**, **setVoxelSize**.

getVersion()
Returns the ImageJ version number as a string (e.g., "1.34s"). See also: **IJ.getFullVersion**.

getWidth()
Returns the width in pixels of the current image.

getZoom()
Returns the magnification of the active image, a number in the range 0.03125 to 32.0 (3.1% to 3200%).

I

IJ Functions
These functions provide access to miscellaneous methods in ImageJ's IJ class.

IJ.deleteRows(index1, index2) - Deletes rows *index1* through *index2* in the results table.
IJ.deleteRows(index1, index2) - Returns the ImageJ version and build number as a string (e.g., "1.52d11").
IJ.getToolName() - Returns the name of the currently selected tool. See also: **setTool**.
IJ.getFullVersion - Returns the ImageJ version and build number as a string (e.g., "1.52d11").
IJ.freeMemory() - Returns the memory status string (e.g., "2971K of 658MB (<1%)") that is displayed when the users clicks in the ImageJ status bar.
IJ.currentMemory() - Returns, as a string, the amount of memory in bytes currently used by ImageJ.
IJ.log(string) - Displays *string* in the Log window.
IJ.maxMemory() - Returns, as a string, the amount of memory in bytes available to ImageJ. This value (the Java heap size) is set in the *Edit>Options>Memory & Threads* dialog box.
IJ.pad(n, length) - Pads 'n' with leading zeros and returns the result (example [https://imagej.nih.gov/ij/macros/examples/StackOverlay.txt]).
IJ.redirectErrorMessages() - Causes next image opening error to be redirected to the Log window and prevents the macro from being aborted (example [https://imagej.nih.gov/ij/macros/examples/BatchMeasureWithRedirectedErrors.txt]).
IJ.renameResults(name) - Changes the title of the "Results" table to the string *name*.
IJ.renameResults(oldName,newName) - Changes the title of a results table from *oldName* to *newName*.

imageCalculator(operator, img1, img2)
Runs the *Process>Image Calculator* tool, where *operator* ("add","subtract","multiply","divide", "and", "or", "xor", "min", "max", "average", "difference" or "copy") specifies the operation, and *img1* and *img2* specify the operands. *img1* and *img2* can be either an image title (a string) or an image ID (an integer). The *operator* string can include up to three modifiers: "create" (e.g., "add create") causes the result to be stored in a new window, "32-bit" causes the result to be 32-bit floating-point and "stack" causes the entire stack to be processed. See the ImageCalculatorDemo [https://imagej.nih.gov/ij/macros/ImageCalculatorDemo.txt] macros for examples.

indexOf(string, substring)
Returns the index within *string* of the first occurrence of *substring*. See also: **lastIndexOf**, **startsWith**, **endsWith**, **substring**, **toLowerCase**, **replace**, **matches**.

indexOf(string, substring, fromIndex)
Returns the index within *string* of the first occurrence of *substring*, with the search starting at *fromIndex*.

is("animated")
Returns *true* if the current image is an animated stack.

is("applet")
Returns *true* if ImageJ is running as an applet.

is("Batch Mode")
Returns *true* if the macro interpreter is running in batch mode.

is("binary")
Returns *true* if the current image is binary (8-bit with only 0 and 255 values).

is("Caps Lock Set")
Returns *true* if the caps lock key is set. Always return *false* on some platforms.

is("changes")
Returns *true* if the current image's 'changes' flag is set.

is("composite")
Returns *true* if the current image is a a multi-channel stack that uses the CompositeImage class.

is("global scale")
Returns *true* if there is global spatial calibration.

is("grayscale")
Returns *true* if the current image is grayscale, or an RGB image with identical R, G and B channels.

is("Inverting LUT")
Returns *true* if the current image is using an inverting (monotonically decreasing) lookup table.

is("InvertY")
Returns *true* if the 'invertY' property of the active image is enabled.

is("locked")
Returns *true* if the current image is locked.

is("Virtual Stack")
Returns *true* if the current image is a virtual stack.

isActive(id)
Returns *true* if the image with the specified ID is active.

isKeyDown(key)
Returns *true* if the specified key is pressed, where *key* must be "shift", "alt" or "space". See also: **setKeyDown**.

isNaN(n)
Returns *true* if the value of the number *n* is NaN (Not-a-Number). A common way to create a NaN is to divide zero by zero. Comparison with a NaN always returns *false* so "if (n!=n)" is equilvalent to (isNaN(n))". Note that the numeric constant NaN is predefined in the macro language. The NaNs [https://imagej.nih.gov/ij/macros/examples/NaNs.txt] macro demonstrates how to remove NaNs from an image.

isOpen(id)
Returns *true* if the image with the specified ID is open.

isOpen("Title")
Returns *true* if the window with the specified title is open.

L

lastIndexOf(string, substring)
Returns the index within *string* of the rightmost occurrence of *substring*. See also: **indexOf**, **startsWith**, **endsWith**, **substring**.

lengthOf(str)
Returns the length of a string or array.

lineTo(x, y)
Draws a line from current location to (*x,y*) . See also: **Overlay.lineTo**.

List Functions
These functions work with a list of key/value pairs. The ListDemo [https://imagej.nih.gov/ij/macros/ListDemo.txt] macro demonstrates how to use them.

List.set(key, value) – Adds a key/value pair to the list.
List.get(key) – Returns the string value associated with *key*, or an empty string if the key is not found.
List.getValue(key) – When used in an assignment statement, returns the value associated with *key* as a number. Aborts the macro if the value is not a number or the key is not found.
List.size – Returns the size of the list.
List.clear() – Resets the list.
List.setList(list) – Loads the key/value pairs in the string *list*.
List.getList – Returns the list as a string.
List.setMeasurements – Measures the current image or selection and loads the resulting keys (Results table column headings) and values into the list. Use *List.setMeasurements("limit")* to measure using the "Limit to threshold" option. All parameters listed in the *Analyze>Set Measurements* dialog box are measured, including those that are unchecked. Use List.getValue() in an assignment statement to retrieve the values. See the DrawEllipse [https://imagej.nih.gov/ij/macros/DrawEllipse.txt] macro for an example.
List.setMeasurements("limit") – This is a version of List.setMeasurements that enables the "Limit to threshold" option.
List.setCommands – Loads the ImageJ menu commands (as keys) and the plugins that implement them (as values).
List.toArrays(keys, values) – Retrieves keys and values as a pair of string arrays, sorted alphabetically for keys.
List.fromArrays(keys, values) – Creates the List from keys and values arrays.
List.indexOf(key) – Returns the alphabetic position of the specified key, or -1 if not found. Note that this function, as well as List.size, returns a string.

log(n)
Returns the natural logarithm (base e) of *n*. Note that log10(n) = log(n)/log(10).

M

makeArrow(x1, y1, x2, y2, style)
Creates an arrow selection, where 'style' is a string containing "filled", "notched", "open", "headless" or "bar", plus the optionial modifiers "outline", "double", "small", "medium" and "large" (example [https://imagej.nih.gov/ij/macros/examples/Arrows.txt]). See also: **Roi.setStrokeWidth** and **Roi.setStrokeColor**. Requires 1.49a.

makeEllipse(x1, y1, x2, y2, aspectRatio)
Creates an elliptical selection, where *x1,y1,x2,y2* specify the major axis of the ellipse and *aspectRatio* (<=1.0) is the ratio of the lengths of minor and major axis.

makeLine(x1, y1, x2, y2)
Creates a new straight line selection. The origin (0,0) is assumed to be the upper left corner of the image. Coordinates are in pixels. You can create segmented line selections by specifying more than two coordinate pairs, for example *makeLine(25,34,44,19,69,30,71,56)*.

makeLine(x1, y1, x2, y2, lineWidth)
Creates a straight line selection with the specified width. See also: **getLine**.

makeOval(x, y, width, height)
Creates an elliptical selection, where (*x,y*) define the upper left corner of the bounding rectangle of the ellipse.

makePoint(x, y)
Creates a point selection at the specified location. Create a multi-point selection by using *makeSelection("point",xpoints,ypoints)*. Use *setKeyDown("shift"); makePoint(x, y);* to add a point to an existing point selection.

makePolygon(x1, y1, x2, y2, x3, y3, ...)
Creates a polygonal selection. At least three coordinate pairs must be specified, but not more

than 200. As an example, *makePolygon(20,48,59,13,101,40,75,77,38,70)* creates a polygon selection with five sides.

makeRectangle(x, y, width, height)
Creates a rectangular selection. The *x* and *y* arguments are the coordinates (in pixels) of the upper left corner of the selection. The origin (0,0) of the coordinate system is the upper left corner of the image.

makeRectangle(x, y, width, height, arcSize)
Creates a rounded rectangular selection using the specified corner arc size.

makeRotatedRectangle(x1, y1, x2, y2, width)
Creates a rotated rectangular selection, which is similar to a wide line where (x1,y1) is the start of the line, (x2,y2) is the end of the line and 'width' is the line width.

makeSelection(type, xcoord, ycoord)
Creates a selection from a list of XY coordinates. The first argument should be "polygon", "freehand", "polyline", "freeline", "angle" or "point", or the numeric value returned by **selectionType**. The *xcoord* and *ycoord* arguments are numeric arrays that contain the X and Y coordinates. See the MakeSelectionDemo [https://imagej.nih.gov/ij/macros/MakeSelectionDemo.txt] macro for examples.

makeText(string, x, y)
Creates a text selection at the specified coordinates. The selection will use the font and size specified by the last call to **setFont()**. The CreateOverlay [https://imagej.nih.gov/ij/macros/examples/CreateOverlay.txt] macro provides an example.

matches(string, regex)
Returns *true* if *string* matches the specified regular expression [http://en.wikipedia.org/wiki/Regular_expression] . See also: **startsWith**, **endsWith**, **indexOf**, **replace**.

maxOf(n1, n2)
Returns the greater of two values.

minOf(n1, n2)
Returns the smaller of two values.

moveTo(x, y)
Sets the current drawing location. The origin is always assumed to be the upper left corner of the image.

N

newArray(size)
Returns a new array containing *size* elements. You can also create arrays by listing the elements, for example newArray(1,4,7) or newArray("a","b","c"). For more examples, see the Arrays [https://imagej.nih.gov/ij/macros/Arrays.txt] macro.

The ImageJ macro language does not directly support 2D arrays. As a work around, either create a blank image and use setPixel() and getPixel(), or create a 1D array using a=newArray(xmax*ymax) and do your own indexing (e.g., value=a[x+y*xmax]).

newImage(title, type, width, height, depth)
Opens a new image or stack using the name *title*. The string *type* should contain "8-bit", "16-bit", "32-bit" or "RGB". In addition, it can contain "white", "black" or "ramp" (the default is "white"). As an example, use "16-bit ramp" to create a 16-bit image containing a grayscale ramp. Precede with *call("ij.gui.ImageWindow.setNextLocation", x, y)* to set the location of the new image. *Width* and *height* specify the width and height of the image in pixels. *Depth* specifies the number of stack slices.

newMenu(macroName, stringArray)
Defines a menu that will be added to the toolbar when the menu tool named *macroName* is installed. Menu tools are macros with names ending in "Menu Tool". *StringArray* is an array containing the menu commands. Returns a copy of *stringArray*. For an example, refer to the Toolbar Menus [https://imagej.nih.gov/ij/macros/toolsets/Toolbar%20Menus.txt] toolset.

nImages
Returns number of open images. The parentheses "()" are optional.

nResults
Returns the current measurement counter value. The parentheses "()" are optional. See also: **getValue("results.count")**.

nSlices
Returns the number of images in the current stack. Returns 1 if the current image is not a stack. The parentheses "()" are optional. See also: **getSliceNumber**, **getDimensions**.

O

open(path)
Opens and displays a tiff, dicom, fits, pgm, jpeg, bmp, gif, lut, roi, or text file. Displays an error message and aborts the macro if the specified file is not in one of the supported formats, or if the file is not found. Displays a file open dialog box if *path* is an empty string or if there is no argument. Use the *File>Open* command with the command recorder running to generate calls to this function. With 1.41k or later, opens images specified by a URL, for example *open("../../images/clown.gif").* With 1.49v or later, opens a folder of images as a stack. Use open("path/to/folder","virtual") to open a folder of images as a virtual stack.

open(path, n)
Opens the *n*th image in the TIFF stack specified by *path*. For example, the first image in the stack is opened if *n*=1 and the tenth is opened if *n*=10.

Overlay Functions
Use these functions to create and manage non-destructive graphic overlays. For an exmple, refer to the OverlayPolygons [https://imagej.nih.gov/ij/macros/examples/OverlayPolygons.txt] macro. See also: **setColor**, **setLineWidth** and **setFont**.

Overlay.moveTo(x, y)
Sets the current drawing location.
Overlay.lineTo(x, y)
Draws a line from the current location to (*x,y*) .
Overlay.drawLine(x1, y1, x2, y2)
Draws a line between (*x1,y1*) and (*x2,y2*)).
Overlay.add
Adds the drawing created by Overlay.lineTo(), Overlay.drawLine(), etc. to the overlay without updating the display.
Overlay.setPosition(n)
Sets the stack position (slice number) of the last item added to the overlay (example [https://imagej.nih.gov/ij/macros/examples/StackOverlay.txt]).
Overlay.setPosition(c, z, t)
Sets the hyperstack position (channel, slice, frame) of the last item added to the overlay.
Overlay.drawRect(x, y, width, height)
Draws a rectangle, where (*x,y*) specifies the upper left corner.
Overlay.drawEllipse(x, y, width, height)
Draws an ellipse, where (*x,y*) specifies the upper left corner of the bounding rectangle.
Overlay.drawString("text", x, y)
Draws text at the specified location and adds it to the overlay. Use **setFont()** to specify the font and **setColor** to set specify the color (example [https://imagej.nih.gov/ij/macros/examples/OverlayDrawStringDemo.txt]).
Overlay.drawString("text", x, y, angle)
Draws text at the specified location and angle and adds it to the overlay (example [https://imagej.nih.gov/ij/macros/examples/RotatedText.txt]).
Overlay.show
Displays the current overlay.
Overlay.hide
Hides the current overlay.
Overlay.hidden
Returns *true* if the active image has an overlay and it is hidden.
Overlay.remove
Removes the current overlay.
Overlay.clear
Resets the overlay without updating the display.
Overlay.size
Returns the size (selection count) of the current overlay. Returns zero if the image does not have an overlay.

Overlay.addSelection
Adds the current selection to the overlay.
Overlay.addSelection(strokeColor)
Sets the stroke color ("red", "green", "ff8800", etc.) of the current selection and adds it to the overlay.
Overlay.addSelection(strokeColor, strokeWidth)
Sets the stroke color ("blue", "yellow", "ffaa77" etc.) and stroke width of the current selection and adds it to the overlay.
Overlay.addSelection("", 0, fillColor)
Sets the fill color ("red", "green", etc.) of the current selection and adds it to the overlay.
Overlay.activateSelection(index)
Activates the specified overlay selection.
Overlay.moveSelection(index, x, y)
Moves the specified selection to the specified location.
Overlay.removeSelection(index)
Removes the specified selection from the overlay.
Overlay.copy
Copies the overlay on the current image to the overlay clipboard.
Overlay.paste
Copies the overlay on the overlay clipboard to the current image.
Overlay.drawLabels(boolean)
Enables/disables overlay labels.
Overlay.selectable(false)
Prevents the selections in this overlay from being activated by clicking on their labels or by long clicking. Requires 1.51r.
Overlay.measure
Measures all the selections in the current overlay.
Overlay.flatten
Creates a new RGB image that has the overlay rendered as pixel data.

P

parseFloat(string)
Converts the string argument to a number and returns it. Returns NaN (Not a Number) if the string cannot be converted into a number. Use the **isNaN()** function to test for NaN. For examples, see ParseFloatIntExamples [https://imagej.nih.gov/ij/macros/ParseFloatIntExamples.txt] .

parseInt(string)
Converts *string* to an integer and returns it. Returns NaN if the string cannot be converted into a integer.

parseInt(string, radix)
Converts *string* to an integer and returns it. The optional second argument (*radix*) specifies the base of the number contained in the string. The radix must be an integer between 2 and 36. For radixes above 10, the letters of the alphabet indicate numerals greater than 9. Set *radix* to 16 to parse hexadecimal numbers. Returns NaN if the string cannot be converted into a integer. For examples, see ParseFloatIntExamples
[https://imagej.nih.gov/ij/macros/ParseFloatIntExamples.txt] .

PI
Returns the constant π (3.14159265), the ratio of the circumference to the diameter of a circle.

Plot Functions
Use these functions to generate and display plots. For examples, check out the Example Plot [https://imagej.nih.gov/ij/macros/examples/Example_Plot.txt] , More Example Plots [https://imagej.nih.gov/ij/macros/ExamplePlots.txt] , AdvancedPlots [https://imagej.nih.gov/ij/macros/examples/AdvancedPlots.txt] , Semi-log Plot [https://imagej.nih.gov/ij/macros/examples/Semi-log_Plot.txt] and Arrow Plot [https://imagej.nih.gov/ij/macros/examples/ArrowPlot.txt] macros.

Plot.create("Title", "X-axis Label", "Y-axis Label", xValues, yValues)
Generates a plot using the specified title, axis labels and X and Y coordinate arrays. If only one array is specified it is assumed to contain the Y values and a 0..n-1 sequence is used as the X values. It is also permissible to specify no arrays and use *Plot.setLimits()* and *Plot.add()* to generate the plot. Use *Plot.show()* to display the plot in a window, or it will be displayed automatically when the macro exits.

Plot.create("Title", "{cat1,cat2,cat3}", "Y-axis Label")
Draws category labels instead of x-axis numbers 0, 1, 2. Requires 1.49u.
Plot.add(type, xValues, yValues)
Adds a curve, set of points or error bars to a plot created using *Plot.create()*. If only one array is specified it is assumed to contain the Y values and a 0..n-1 sequence is used as the X values. The first argument (*type*) can be "line", filled", "bars", "circles", "boxes", "triangles", "crosses", "diamonds", "dots", "x", "connected", "error bars" (in y direction) or "xerror bars". In 1.49t or later, error bars apply to the last dataset provided by *Plot.create* or *Plot.add*.
Plot.drawVectors(xStarts, yStarts, xEnds, yEnds)
Draws arrows from the starting to ending coordinates contained in the arrays.
Plot.drawLine(x1, y1, x2, y2)
Draws a line between *x1,y1* and *x2,y2*, using the coordinate system defined by *Plot.setLimits()*.
Plot.drawNormalizedLine(x1, y1, x2, y2)
Draws a line using a normalized 0-1, 0-1 coordinate system, with (0,0) at the top left and (1,1) at the lower right corner.
Plot.addText("A line of text", x, y)
Adds text to the plot at the specified location, where (0,0) is the upper left corner of the the plot frame and (1,1) is the lower right corner. Call *Plot.setJustification()* to have the text centered or right justified. **Plot.setLimits(xMin, xMax, yMin, yMax)**
Sets the range of the x-axis and y-axis of plots. With version 1.50g and later, when 'NaN' is used as a limit, the range is calculated automatically from the plots that have been added so far.
Plot.getLimits(xMin, xMax, yMin, yMax)
Returns the current axis limits. Note that min>max if the axis is reversed. Requires 1.49t.
Plot.setLimitsToFit()
Sets the range of the x and y axes to fit all data. Requires 1.49t.
Plot.setColor(color)
Specifies the color used in subsequent calls to *Plot.add()* or *Plot.addText()*. The argument can be "black", "blue", "cyan", "darkGray", "gray", "green", "lightGray", "magenta", "orange", "pink", "red", "white", "yellow", or a hex value like "#ff00ff". Note that the curve specified in *Plot.create()* is drawn last.
Plot.setColor(color1, color2)
This is a two argument version of Plot.setColor, where the second argument is used for filling symbols or as the line color for connected points. Requires 1.49t.
Plot.setBackgroundColor(color)
Sets the background color of the plot frame (example [https://imagej.nih.gov/ij/macros/examples/PlotBackgroundColorDemo.txt]). Requires 1.49h.
Plot.setLineWidth(width)
Specifies the width of the line used to draw a curve. Points (circle, box, etc.) are also drawn larger if a line width greater than one is specified. Note that the curve specified in *Plot.create()* is the last one drawn before the plot is dispayed or updated.
Plot.setJustification("center")
Specifies the justification used by *Plot.addText()*. The argument can be "left", "right" or "center". The default is "left".
Plot.setLegend("label1\tlabel2...", "options")
Creates a legend for each of the data sets added with *Plot.create* and *Plot.add*. In the first argument, the labels for the data sets should be separated with tab or newline characters. The optional second argument can contain the legend position ("top-left", "top-right", "bottom-left", "bottom-right"; default is automatic positioning), "bottom-to-top" for reversed sequence of the labels, and "transparent" to make the legend background transparent. Requires 1.49t.
Plot.setFrameSize(width, height)
Sets the plot frame size in pixels, overriding the default size defined in the *Edit>Options>Profile Plot Options* dialog box.
Plot.getFrameBounds(x, y, width, height)
Returns the plot frame bounds.
Plot.setLogScaleX(boolean)
Sets the x axis scale to Logarithmic, or back to linear if the optional boolean argument is false. In versions up to 1.49s, it must be called immediately after *Plot.create* and before *Plot.setLimits*. See the LogLogPlot [https://imagej.nih.gov/ij/macros/examples/LogLogPlot.txt] macro for an example.
Plot.setLogScaleY(boolean)
Sets the y axis scale to Logarithmic, or back to linear if the optional boolean argument is false.

Plot.setXYLabels("x Label", "y Label")
Sets the axis labels. Requires 1.49t.
Plot.setFontSize(size, "options")
Sets the default font size in the plot (otherwise specified in *Profile Plot Options*), used e.g. for axes labels. Can be also called prior to *Plot.addText*. The optional second argument can include "bold" and/or "italic". Requires 1.49t.
Plot.setAxisLabelSize(size, "options")
Sets the fort size of the axis labels. The optional second argument can include "bold" and/or "italic". Requires 1.49t. **Plot.setFormatFlags("11001100001111")**
Controls whether to draw axes labels, grid lines and ticks (major/minor/ticks for log axes). Use the macro recorder and *More>>Axis Options* in any plot window to determine the binary code. Requires 1.49t.
Plot.useTemplate("plot name" or id)
Transfers the formatting of an open plot window to the current plot. Requires 1.49t.
Plot.makeHighResolution(factor)
Creates a high-resolution image of the plot enlarged by *factor*. Add the second argument "disable" to avoid antialiased text. Requires 1.49t. **Plot.show()**
Displays the plot generated by *Plot.create()*, *Plot.add()*, etc. in a window. This function is automatically called when a macro exits.
Plot.update()
Draws the plot generated by *Plot.create()*, *Plot.add()*, etc. in an existing plot window. Equivalent to *Plot.show* if no plot window is open.
Plot.getValues(xpoints, ypoints)
Returns the values displayed by clicking on "List" in a plot or histogram window (example [https://imagej.nih.gov/ij/macros/examples/PlotGetValuesDemo.txt]).
Plot.showValues()
Displays the plot values in the Results window. Must be preceded by *Plot.show*.
Plot.drawGrid()
Redraws the grid above previous plots. Requires 1.49u.
Plot.drawShapes("rectangles", lefts, tops, rights, bottoms)
Draws one or more rectangles. The four arguments (values or arrays) hold rectangle coordinates. Requires 1.49u.
Plot.drawBoxes("boxes width=30", x, y1, y2, y3, y4, y5)
Draws a boxplot, where 'width' is in pixels, array 'x' holds x-positions and arrays 'y1'..'y5' hold the quartile borders in ascending order. Secondary color will fill the box. For horizontal boxes, use "boxesx width=30" instead. Requires 1.49u.

pow(base, exponent)
Returns the value of *base* raised to the power of *exponent*.

print(string)
Outputs a string to the "Log" window. Numeric arguments are automatically converted to strings. The print() function accepts multiple arguments. For example, you can use *print(x,y,width, height)* instead of *print(x+" "+y+" "+width+" "+height)*. If the first argument is a file handle returned by **File.open(path)**, then the second is saved in the refered file (see SaveTextFileDemo [https://imagej.nih.gov/ij/macros/SaveTextFileDemo.txt]).

Numeric expressions are automatically converted to strings using four decimal places, or use the **d2s** function to specify the decimal places. For example, *print(2/3)* outputs "0.6667" but *print(d2s(2/3,1))* outputs "0.7".

The print() function accepts commands such as *"\|\|Clear"*, *"\|\|Update:<text>"* and *"\|\|Update<n>:<text>"* (for n<26) that clear the "Log" window and update its contents. For example, *print("\|\|Clear")* erases the Log window, *print("\|\|Update:new text")* replaces the last line with "new text" and *print("\|\|Update8:new 8th line")* replaces the 8th line with "new 8th line". Refer to the LogWindowTricks [https://imagej.nih.gov/ij/macros/LogWindowTricks.txt] macro for an example.

The second argument to print(arg1, arg2) is appended to a text window or table if the first argument is a window title in brackets, for example *print("[My Window]", "Hello, world")*. With text windows, newline characters ("\n") are not automatically appended and text that starts with *"\|\|Update:"* replaces the entire contents of the window. Refer to the PrintToTextWindow [https://imagej.nih.gov/ij/macros/PrintToTextWindow.txt] , Clock [https://imagej.nih.gov/ij/macros/Clock.txt] and ProgressBar [https://imagej.nih.gov/ij/macros/ProgressBar.txt] macros for examples.

The second argument to print(arg1, arg2) is appended to a table (e.g., ResultsTable) if the first argument is the title of the table in brackets. Use the *Plugins>New* command to open a blank table. Any command that can be sent to the "Log" window (*"\|\|Clear"*, *"\|\|Update:<text>"*

, etc.) can also be sent to a table. Refer to the SineCosineTable2 [https://imagej.nih.gov/ij/macros/SineCosineTable2.txt] and TableTricks [https://imagej.nih.gov/ij/macros/TableTricks.txt] macros for examples.

R

random
Returns a random number between 0 and 1.

random("seed", seed)
Sets the seed (a whole number) used by the *random()* function.

rename(name)
Changes the title of the active image to the string *name*.

replace(string, old, new)
Returns the new string that results from replacing all occurrences of *old* in *string* with *new*, where *old* and *new* are single character strings. If *old* or *new* are longer than one character, each substring of *string* that matches the regular expression [http://en.wikipedia.org/wiki/Regular_expression] *old* is replaced with *new*. When doing a simple string replacement, and *old* contains regular expression metacharacters ('.', '[', ']', '^', '$', etc.), you must escape them with a "\\". For example, to replace "[xx]" with "yy", use *string=replace(string,"\\[xx\\]","yy")*. See also: **matches**.

requires("1.29p")
Display a message and aborts the macro if the ImageJ version is less than the one specified. See also: **getVersion**.

reset
Restores the backup image created by the **snapshot** function. Note that reset() and run("Undo") are basically the same, so only one run() call can be reset.

resetMinAndMax
With 16-bit and 32-bit images, resets the minimum and maximum displayed pixel values (display range) to be the same as the current image's minimum and maximum pixel values. With 8-bit images, sets the display range to 0-255. With RGB images, does nothing. See the DisplayRangeMacros [https://imagej.nih.gov/ij/macros/DisplayRangeMacros.txt] for examples.

resetThreshold
Disables thresholding. See also: **setThreshold**, **setAutoThreshold**, **getThreshold**.

restoreSettings
Restores *Edit>Options* submenu settings saved by the **saveSettings** function.

ROI Functions
Use these functions to get information about the current selection or to get or set selection properties. Refer to the RoiFunctionsDemo [https://imagej.nih.gov/ij/macros/examples/RoiFunctionsDemo.txt] macro for examples. These functions require ImageJ 1.48h or later.

Roi.contains(x, y)
Returns "1" if the point *x,y* is inside the current selection or "0" if it is not. Aborts the macro if there is no selection. See also: **Roi.getContainedPoints** and **selectionContains**.
Roi.getBounds(x, y, width, height)
Returns the location and size of the selection's bounding rectangle.
Roi.getCoordinates(xpoints, ypoints)
Returns, as two arrays, the x and y coordinates that define this selection.
Roi.getContainedPoints(xpoints, ypoints)
Returns, as two arrays, the x and y coordinates of the pixels inside the current selection. Aborts the macro if there is no selection.
Roi.getDefaultColor
Returns the current default selection color.
Roi.getStrokeColor
Returns the selection stroke color.
Roi.getFillColor
Returns the selection fill color.
Roi.getName
Returns the selection name or an empty string if the selection does not have a name.

Roi.getProperty(key)
Returns the value (a string) associated with the specified key or an empty string if the key is not found.
Roi.setProperty(key, value)
Adds the specified key and value pair to the selection properties. Assumes a value of "1" (true) if there is only one argument.
Roi.getProperties
Returns all selection properties or an empty string if the selection does not have properties.
Roi.getSplineAnchors(x, y)
Returns the x and y coordinates of the anchor points of a spline fitted selection (example [https://imagej.nih.gov/ij/macros/examples/GetSetAnchors.txt]).
Roi.setPolygonSplineAnchors(x, y)
Creates or updates a spline fitted polygon selection (example [https://imagej.nih.gov/ij/macros/examples/GetSetAnchors.txt]).
Roi.setPolylineSplineAnchors(x, y)
Creates or updates a spline fitted polyline selection (example [https://imagej.nih.gov/ij/macros/examples/GetSetAnchors.txt]).
Roi.getType
Returns, as a string, the type of the current selection.
Roi.move(x, y)
Moves the selection to the specified location.
Roi.setStrokeColor(color)
Sets the selection stroke color ("red", "5500ff00". etc.).
Roi.setStrokeColor(red, green, blue)
Sets the selection stroke color, where 'red', 'green' and 'blue' are integers (0-255).
Roi.setStrokeColor(rgb)
Sets the selection stroke color, where 'rgb' is an integer.
Roi.setFillColor(color)
Sets the selection fill color ("red", "5500ff00". etc.).
Roi.setFillColor(red, green, blue)
Sets the selection fill color, where 'red', 'green' and 'blue' are integers (0-255).
Roi.setFillColor(rgb)
Sets the selection fill color, where 'rgb' is an integer.
Roi.setName(name)
Sets the selection name.
Roi.setStrokeWidth(width)
Sets the selection stroke width.
Roi.remove
Deletes the selection, if any, from the active image.

ROI Manager Functions
These function run ROI Manager commands. The ROI Manager is opened if it is not already open. Use *roiManager("reset")* to delete all ROIs on the list. Use **setOption("Show All", boolean)** to enable/disable "Show All" mode. For examples, refer to the RoiManagerMacros [https://imagej.nih.gov/ij/macros/RoiManagerMacros.txt] , ROI Manager Stack Demo [https://imagej.nih.gov/ij/macros/ROI_Manager_Stack_Demo.txt] and RoiManagerSpeedTest [https://imagej.nih.gov/ij/macros/RoiManagerSpeedTest.txt] macros.

roiManager("and")
Uses the conjunction operator on the selected ROIs, or all ROIs if none are selected, to create a composite selection.

roiManager("add")
Adds the current selection to the ROI Manager.

roiManager("add & draw")
Outlines the current selection and adds it to the ROI Manager.

roiManager("combine")
Uses the union operator on the selected ROIs to create a composite selection. Combines all ROIs if none are selected.

roiManager("count")
Returns the number of ROIs in the ROI Manager list.

roiManager("delete")
Deletes the selected ROIs from the list, or deletes all ROIs if none are selected.

roiManager("deselect")
Deselects all ROIs in the list. When ROIs are deselected, subsequent ROI Manager commands are applied to all ROIs.

roiManager("draw")
Draws the selected ROIs, or all ROIs if none are selected, using the equivalent of the *Edit>Draw* command.

roiManager("fill")
Fills the selected ROIs, or all ROIs if none are selected, using the equivalent of the *Edit>Fill* command.

roiManager("index")
Returns the index of the currently selected ROI on the list, or -1 if the list is empty or no ROIs are selected. Returns the index of the first selected ROI if more than one is selected

roiManager("measure")
Measures the selected ROIs, or if none is selected, all ROIs on the list.

roiManager("multi measure")
Measures all the ROIs on all slices in the stack, creating a Results Table with one row per slice.

roiManager("multi-measure append")
Measures all the ROIs on all slices in the stack, appending the measurements to the Results Table, with one row per slice.

roiManager("multi-measure one")
Measures all the ROIs on all slices in the stack, creating a Results Table with one row per measurement.

roiManager("multi plot")
Plots the selected ROIs, or all ROIs if none are selected, on a single graph.

roiManager("open", file-path)
Opens a .roi file and adds it to the list or opens a ZIP archive (.zip file) and adds all the selections contained in it to the list.

roiManager("remove slice info")
Removes the information in the ROI names that associates them with particular stack slices.

roiManager("rename", name)
Renames the selected ROI. You can get the name of an ROI on the list using *call("ij.plugin.frame.RoiManager.getName", index)*.

roiManager("reset")
Deletes all ROIs on the list.

roiManager("save", file-path)
Saves all the ROIs on the list in a ZIP archive.

roiManager("save selected", file-path)
Saves the selected ROI as a .roi file.

roiManager("select", index)
Selects an item in the ROI Manager list, where *index* must be greater than or equal zero and less than the value returned by *roiManager("count")*. Note that macros that use this function sometimes run orders of magnitude faster in batch mode. Use *roiManager("deselect")* to deselect all items on the list. For an example, refer to the ROI Manager Stack Demo [https://imagej.nih.gov/ij/macros/ROI_Manager_Stack_Demo.txt] macro.

roiManager("select", indexes)
Selects multiple items in the ROI Manager list, where *indexes* is an array of integers, each of which must be greater than or equal to 0 and less than the value returned by *roiManager("count")*. The selected ROIs are not highlighted in the ROI Manager list and are no longer selected after the next ROI Manager command is executed.

roiManager("show all")
Displays all the ROIs as an overlay.

roiManager("show all with labels")
Displays all the ROIs as an overlay, with labels.

roiManager("show all without labels")
Displays all the ROIs as an overlay, without labels.

roiManager("show none")
Removes the ROI Manager overlay.

roiManager("sort")
Sorts the ROI list in alphanumeric order.

roiManager("split")
Splits the current selection (it must be a composite selection) into its com