Cheat sheet

ImageJ macro commands and user interfaces

Robert Haase (Myers lab, MPI-CBG); Benoit Lombardot, Noreen Walker and Gayathri Nadar (Scientific Computing Facility, MPI-CBG); Jens Ehrig (CMCB, TU Dresden)

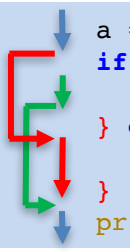
Macro language elements

```
// comments for code documentation
numericVariable = 5;
stringVariable = "text value";
builtInCommand();
```

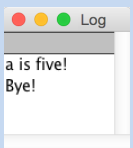
Operator	Description	Example all yield a = 9
=	Assignment	a = 9;
+	Addition	a = 3 + 6;
-	Subtraction	a = 11 - 2;
*	Multiplication	a = 2 * 4.5;
/	Division	a = 27 / 3;
++	Increment by 1	a = 8; a++;
--	Decrement by 1	a = 10; a--;
+=	Addition assignment	a = 3; a += 6;
-=	Subtraction assignment	a = 11; a -= 2;
*=	Multiplication assignment	a = 2; a *= 4.5;
/=	Division assignment	a = 27; a /= 3;

Math command	Description	Example all yield a = 9
pow(x, y)	x to the power of y	a = pow(3, 2);
sqrt(x)	square root of x	a = sqrt(81);
abs(x)	absolute value of x	a = abs(-9);
round(x)	rounding of x	a = round(9.4);
floor(x)	rounding down of x	a = floor(9.8);

Conditional programming (if statement)



```
a = 5;
if (a == 5) {
    print("a is five!");
} else {
    print("a is not five!");
}
print("Bye!");
```



Iterative programming (for loop)

```
for(i = 0; i < 3; i++) {
    print("i is " + i);
}
```

① *start* with this
② *before* each loop: check & only continue if this is true
③ *after* each loop: do this

Iterative programming (while loop)

```
while (condition) {
    // do sth at each loop iteration
    // until condition is false
}
```

String manipulation commands

```
output = replace(input, pattern, subst);
```
replace any occurrence of *pattern* in *input* by *rsubst*

```
outputArray = split(input, separator);
```
cut a string into a list of strings (array) according to the separator position(s)

```
length = lengthOf(string);
```
returns number of characters of the string (see below for "lengthOf(array)")

```
result = startsWith(input, pattern);
```
returns true, if input starts with given pattern

```
result = endsWith(input, pattern);
```
returns true, if input end with pattern

Conditions and logical operators

Operator	Description	Example for a = 2; b = 3;
<, <=	smaller than, smaller or equal to	c = (a < b); // c is 1 ("true")
>, >=	greater than, greater or equal to	c = (a > b); // c is 0 ("false")
==	equal to	c = (a==b); // c is 0 ("false")
!=	not equal to[1]	c = (a != 1); // c is 1 ("true")

a	b	"AND": a && b (corresp. to a*b)	"OR": a || b (~corresp. to a+b)	"NOT": !a (corresp. to 1-a)
0	0	0	0	1
1	0	0	1	0
0	1	0	1	
1	1	1	1	

Boolean variables:
1 means true
0 means false

```
true || true  && false →  1 + 1  * 0 = 1
(true || true) && false → (1 + 1) * 0 = 0
```

Custom functions

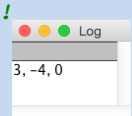
```
// define a custom function
function customFunction (param) {
    return param * 2;
}

a = customFunction(3); // call the function
```

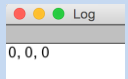
Vectors / arrays

$$\vec{v} = \begin{pmatrix} 3 \\ -4 \\ 0 \end{pmatrix}$$

```
// create arrays
v = newArray(3, -4, 0);

// arrays can also hold strings
animals = newArray("Dog","Cat","Mouse");

// access individual array elements
v[0] = 3;
// NOTE: the first element has index 0!

// output arrays
Array.print(v);

// create an empty array of given size
v = newArray(3);
Array.print(v);

// combine arrays
mixed = Array.concat(v, animals);

// determine size of an array
numberOfElements = lengthOf(v);
```

$\vec{v}_1 = 3$





Cheat sheet

ImageJ macro commands and user interfaces

Robert Haase (Myers lab, MPI-CBG); Benoit Lombardot, Noreen Walker and Gayathri Nadar (Scientific Computing Facility, MPI-CBG); Jens Ehrig (CMCB, TU Dresden)

Switch between image windows

```
titleOfCurrentImage = getTitle();
selectWindow(titleOfAnyImage);
```

Navigation in image stacks

```
Stack.getDimensions(width, height,
channels, slices, frames);
Stack.setSlice(slice);
Stack.setChannel(channel);
Stack.setFrame(frame);
Stack.setDisplayMode("color");
Stack.setDisplayMode("composite");
Stack.setDisplayMode("grayscale");
```

Handle image files and folders

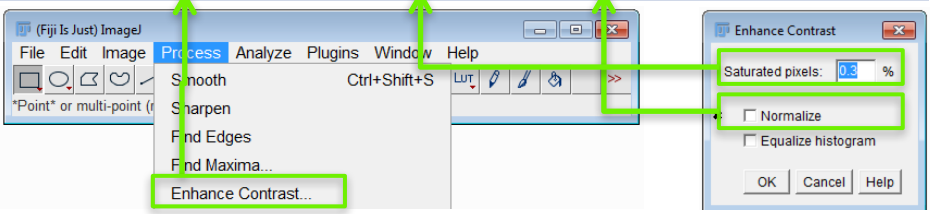
```
open(folder+imageFilename);
close();
fileList = getFileList(folder);
numFiles = lengthOf(fileList);
for (i=0;i<lengthOf(fileList);i++){
    file = fileList[i];
    open(file);
    // actual image processing...
    close();
}
```

Reading image calibration

```
getPixelSize(unit, pWidth, pHeight);
getVoxelSize(vWidth, vHeight,
vDepth, unit);
```

Calling any ImageJ/FIJI menu

```
run("Enhance Contrast...", "saturated=0.3 normalize")
```



ROI manager

```
roiManager("add");
roiManager("split");
roiManager("delete");
roiManager("reset");
roiManager("measure");
roiManager("count");
roiManager("open", filename);
roiManager("save", filename);
roiManager("save selected", filename);
roiManager("select", index);
roiManager("select", newArray(index1,
index2, ...));
roiManager("deselect");
roiManager("show all");
roiManager("show all with labels");
roiManager("show none");
roiManager("and");
roiManager("combine");
```

Ask for user action

```
waitForUser("headline", "prompt");
```

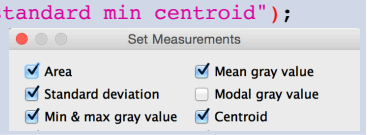
Basic image statistics

```
getStatistics(area, mean, min,
max, standard_deviation);
```

Result tables

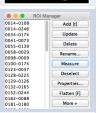
```
run("Set Measurements...", "area
mean standard min centroid");
```

corresp. to this:




```
run("Analyze Particles...",
"add clear display");
```

```
roiManager("Measure");
```

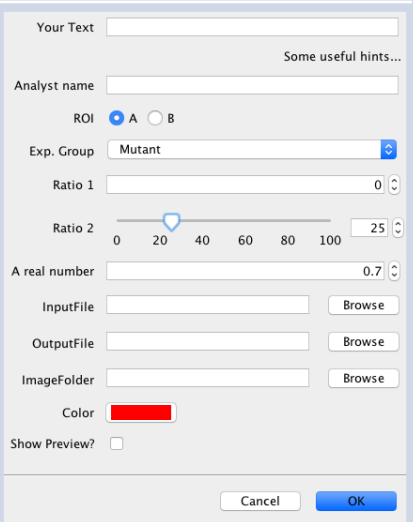


```
rowCount = nResults();
value = getResult("column title",
                  rowNumber);
setResult("column title",
rowNumber, newValue);
saveAs("Results","myResults.xls");
run("Clear Results");
```

Generate user interfaces with `#@Parameter`

Syntax: `#@ <data type>(<options>) <variable name>`

```
#@ String(label="Your Text") userText
#@ String(value="Some useful hints...",
     visibility="MESSAGE") hints
#@ String(label="Analyst name",
     description="Your name") analystName
#@ String(choices={"A", "B"},
     style="radioButtonHorizontal") ROI
#@ String(label="Exp. Group",
     choices={"Mutant", "Control"},
     style="list") expGroup
#@ Integer(label="Ratio 1") r1
#@ Integer(label="Ratio 2", value=25,
     min=0, max=100, style="slider") r2
#@ Double(value=0.7, min=0, max=1,
     label="A real number") realNumber
#@ File(style="open") inputFile
#@ File(style="save") outputFile
#@ File(style="directory") imageFolder
#@ ColorRGB(value="red") color
#@ Boolean(label="Show Preview?") preview
```



Useful links

ImageJ macro reference	https://imagej.nih.gov/ij/developer/macro/macros.html https://imagej.nih.gov/ij/developer/macro/functions.html
ImageJ / Fiji plugins	https://imagej.net/Category:Plugins
Forum	http://forum.imagej.net/
Macro code auto formatter	http://jsbeautifier.org/

Best practices in developing software

Divide and rule

- Split complex issues into smaller, accessible issues
- If a function solves several issues, split it in separate functions.

Don't repeat yourself (DRY)

- Don't copy code if similar things are done twice, because you may copy programming errors.
- Program a loop or custom function instead. Maintenance is easier then.

Keep it short and simple (KISS)

- develop code so that others can read, understand and maintain it.

Variable and function names

- name functions after what they do, (verb + object). e.g.: analyzeImage()
- name variables after what they contain, e.g.: (~~"A"~~ versus "area")
- assign parameter values at the beginning of the script, so you do not have to search for them once you want to change them