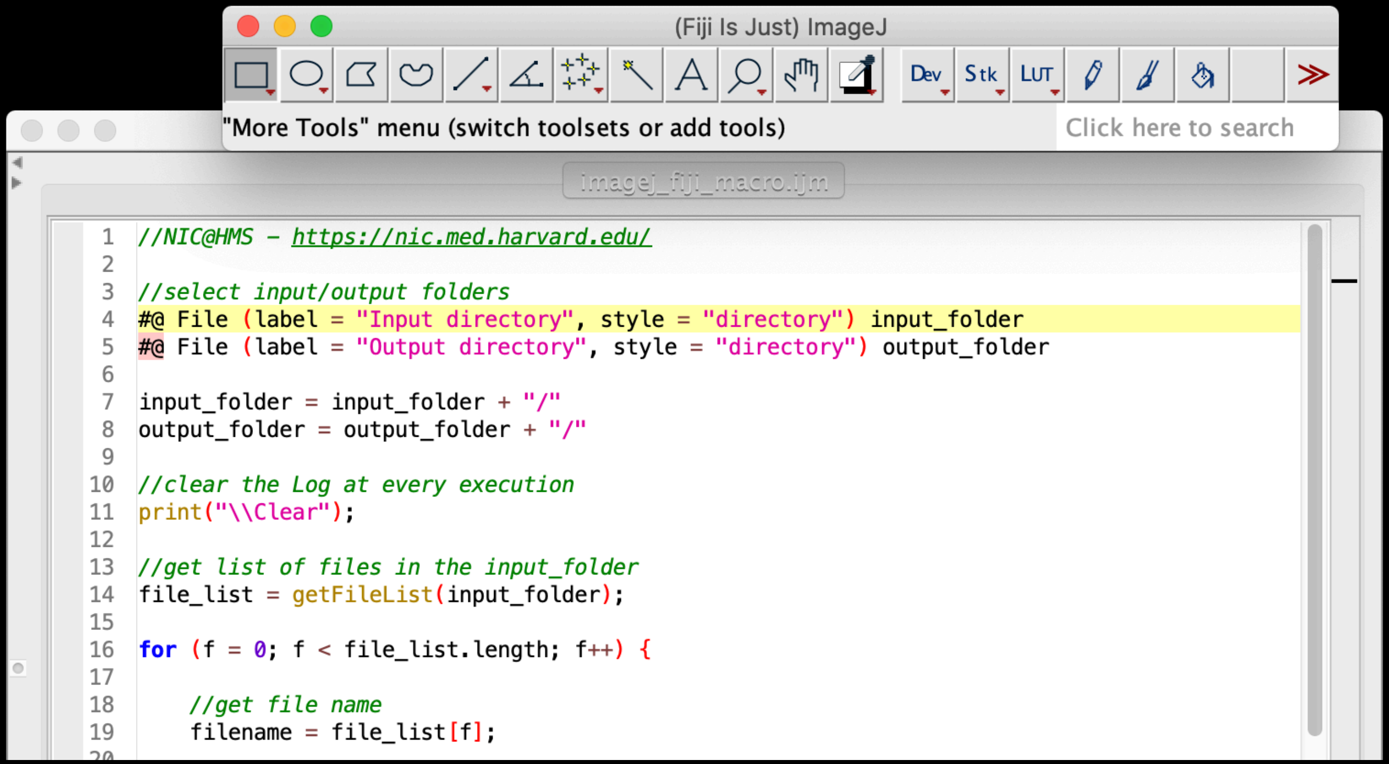
Image Analysis Collaboratory

Automate your work with ImageJ/Fiji Macros



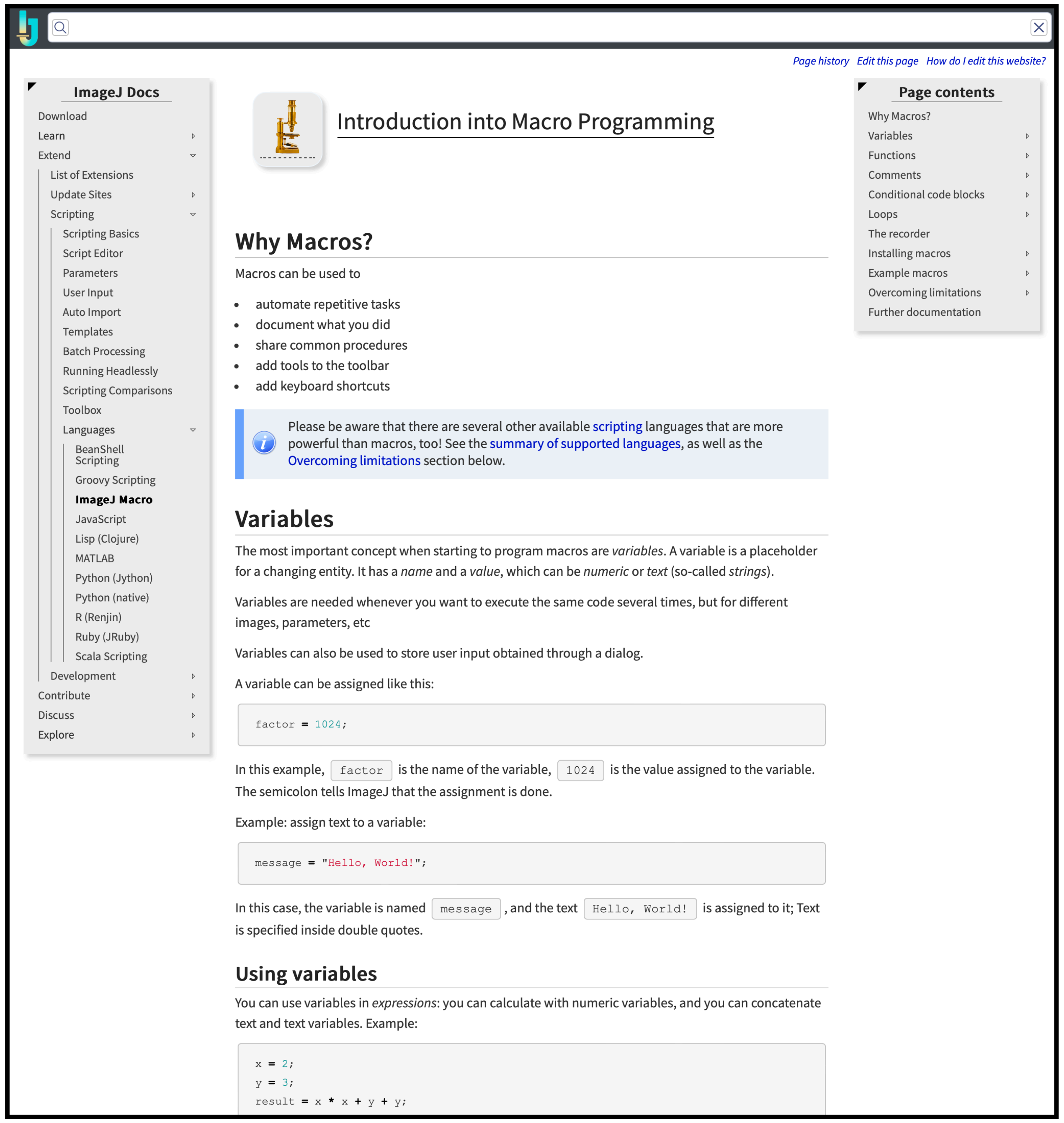
```
//NIC@HMS - https://nic.med.harvard.edu/

//select input/output folders
#@ File (label = "Input directory", style = "directory") input_folder
#@ File (label = "Output directory", style = "directory") output_folder

input_folder = input_folder + "/"
output_folder = output_folder + "/"

//clear the Log at every execution
print("\\Clear");

//get list of files in the input_folder
file_list = getFileList(input_folder);

for (f = 0; f < file_list.length; f++) {

    //get file name
    filename = file_list[f];
```

Federico Gasparoli, MSc, PhD
Research Associate
Image Analysis Collaboratory - Harvard Medical School

Fiji Macro

https://imagej.net/ij/developer/macro/macros.html

https://imagej.net/ij/developer/macro/functions.html



Page history Edit this page How do I edit this website?

ImageJ Docs

- Download
- Learn
- Extend
  - List of Extensions
  - Update Sites
  - Scripting
    - Scripting Basics
    - Script Editor
    - Parameters
    - User Input
    - Auto Import
    - Templates
    - Batch Processing
    - Running Headlessly
    - Scripting Comparisons
    - Toolbox
    - Languages
      - BeanShell Scripting
      - Groovy Scripting
      - **ImageJ Macro**
      - JavaScript
      - Lisp (Clojure)
      - MATLAB
      - Python (Jython)
      - Python (native)
      - R (Renjin)
      - Ruby (JRuby)
      - Scala Scripting
  - Development
- Contribute
- Discuss
- Explore

Introduction into Macro Programming

Page contents

- Why Macros?
- Variables
- Functions
- Comments
- Conditional code blocks
- Loops
- The recorder
- Installing macros
- Example macros
- Overcoming limitations
- Further documentation

Why Macros?

Macros can be used to

- automate repetitive tasks
- document what you did
- share common procedures
- add tools to the toolbar
- add keyboard shortcuts

Please be aware that there are several other available scripting languages that are more powerful than macros, too! See the summary of supported languages, as well as the Overcoming limitations section below.

Variables

The most important concept when starting to program macros are *variables*. A variable is a placeholder for a changing entity. It has a *name* and a *value*, which can be *numeric* or *text* (so-called *strings*).

Variables are needed whenever you want to execute the same code several times, but for different images, parameters, etc

Variables can also be used to store user input obtained through a dialog.

A variable can be assigned like this:

```
factor = 1024;
```

In this example, `factor` is the name of the variable, `1024` is the value assigned to the variable. The semicolon tells ImageJ that the assignment is done.

Example: assign text to a variable:

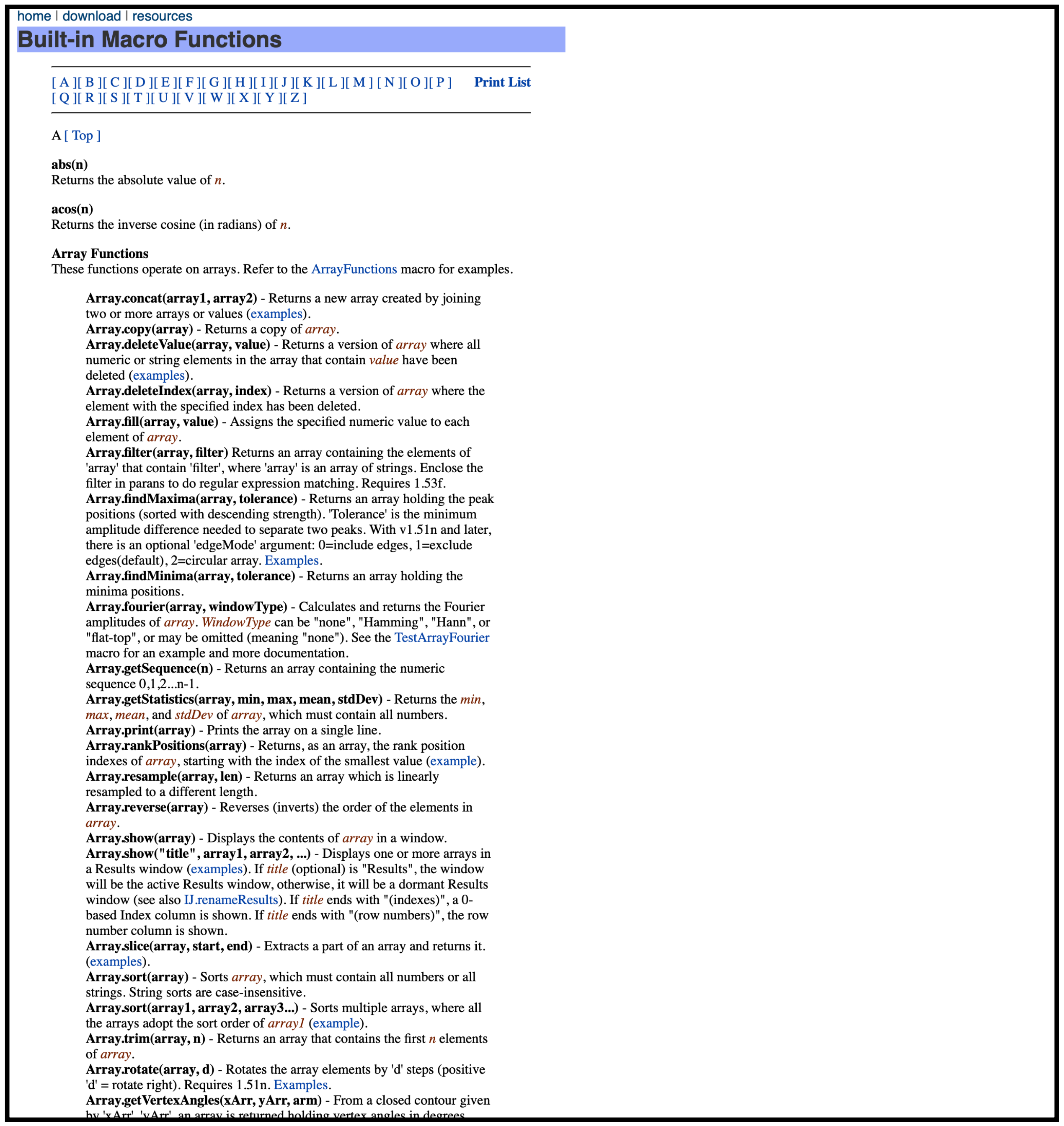
```
message = "Hello, World!";
```

In this case, the variable is named `message`, and the text `Hello, World!` is assigned to it; Text is specified inside double quotes.

Using variables

You can use variables in *expressions*: you can calculate with numeric variables, and you can concatenate text and text variables. Example:

```
x = 2;
y = 3;
result = x * x + y + y;
```



home | download | resources

Built-in Macro Functions

[A][B][C][D][E][F][G][H][I][J][K][L][M] [N][O][P] **Print List**
[Q][R][S][T][U][V][W][X][Y][Z]

A [Top]

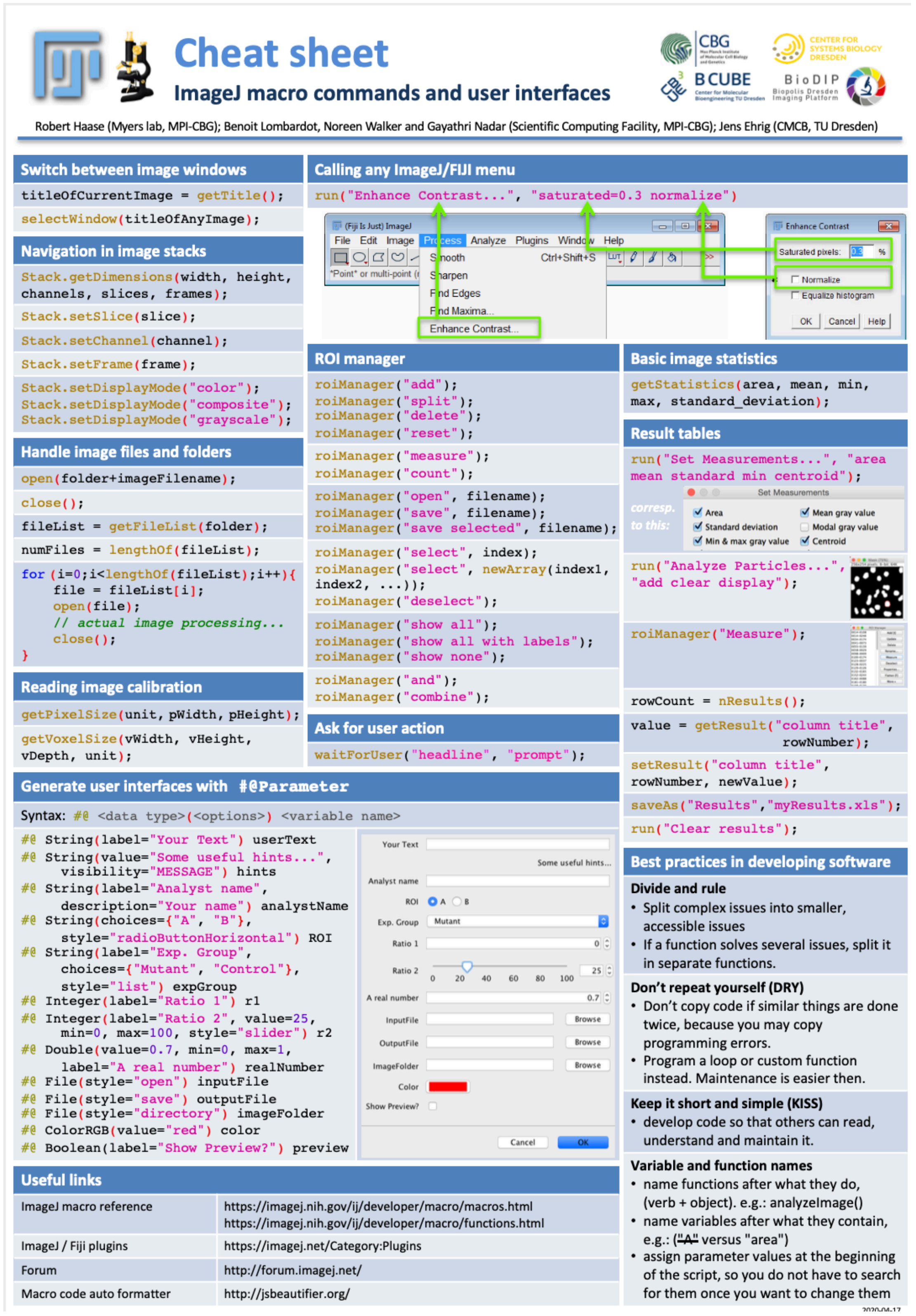
abs(n)
Returns the absolute value of *n*.

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

Array Functions
These functions operate on arrays. Refer to the ArrayFunctions macro for examples.

Array.concat(array1, array2) - Returns a new array created by joining two or more arrays or values (examples).
Array.copy(array) - Returns a copy of *array*.
Array.deleteValue(array, value) - Returns a version of *array* where all numeric or string elements in the array that contain *value* have been deleted (examples).
Array.deleteIndex(array, index) - Returns a version of *array* where the element with the specified index has been deleted.
Array.fill(array, value) - Assigns the specified numeric value to each element of *array*.
Array.filter(array, filter) Returns an array containing the elements of 'array' that contain 'filter', where 'array' is an array of strings. Enclose the filter in parans to do regular expression matching. Requires 1.53f.
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.
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 macro for an example and more documentation.
Array.getSequence(n) - Returns an array containing the numeric sequence 0,1,2...n-1.
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).
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). 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).
Array.sort(array) - Sorts *array*, which must contain all numbers or all strings. String sorts are case-insensitive.
Array.sort(array1, array2, array3...) - Sorts multiple arrays, where all the arrays adopt the sort order of *array1* (example).
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.
Array.getVertexAngles(xArr, yArr, arm) - From a closed contour given by 'xArr', 'yArr', an array is returned holding vertex angles in degrees

Macro Cheat Sheet



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/

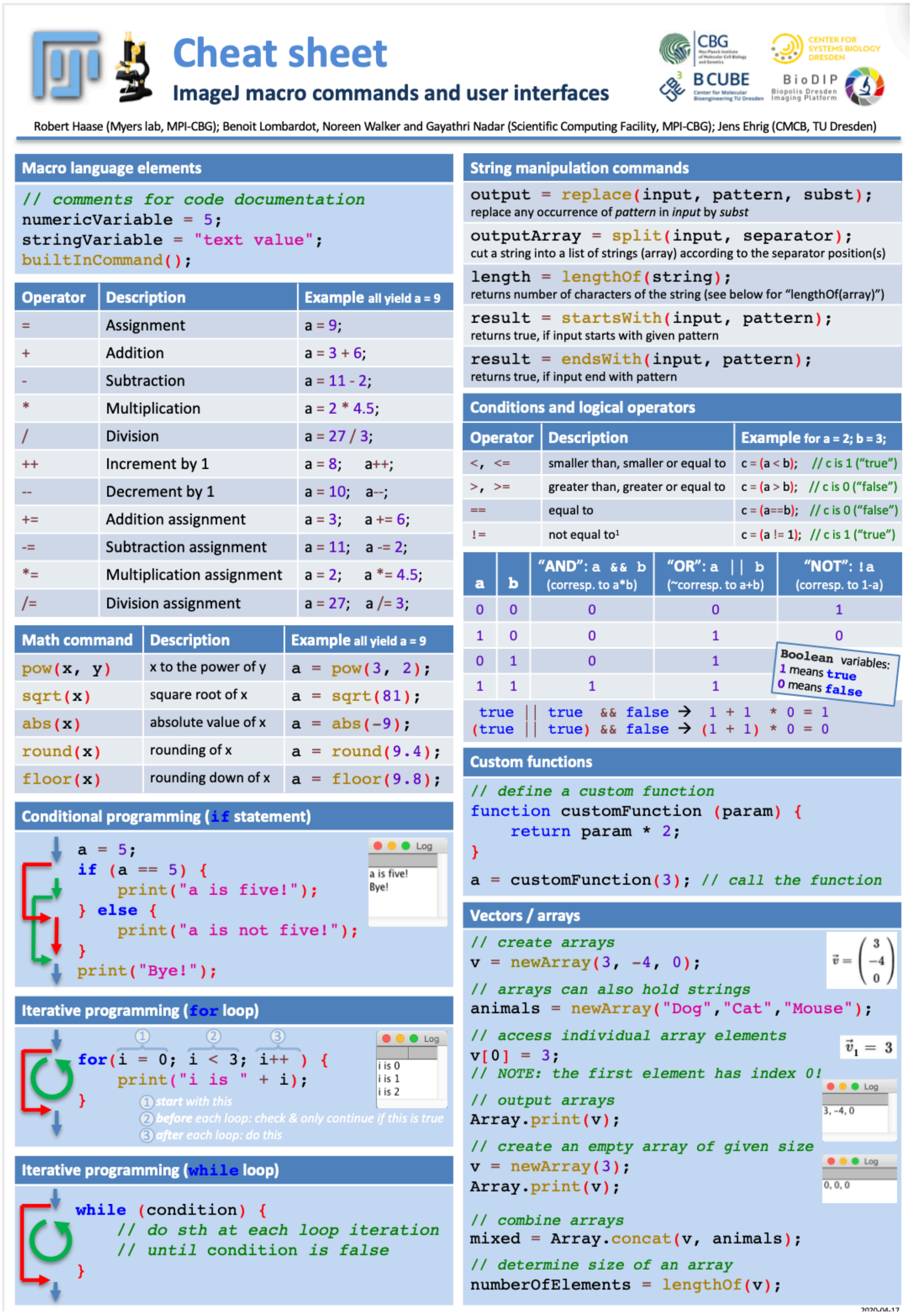
Best practices in developing software
Divide and rule
- Split complex issues into smaller, accessible issues
- If a function solves several issues, split it in separate functions.

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

2020-04-17



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!");
```
Log
```
a is five!
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

Log
```
i is 0
i is 1
i is 2
```

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 *subst*
```
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
```
// create arrays
v = newArray(3, -4, 0);
```
$\vec{v} = \begin{pmatrix} 3 \\ -4 \\ 0 \end{pmatrix}$
```
// arrays can also hold strings
animals = newArray("Dog","Cat","Mouse");

// access individual array elements
v[0] = 3;
// NOTE: the first element has index 0!
```
$\vec{v}_1 = 3$
```
// output arrays
Array.print(v);
```
Log
```
3, -4, 0
```
```
// create an empty array of given size
v = newArray(3);
Array.print(v);
```
Log
```
0, 0, 0
```
```
// combine arrays
mixed = Array.concat(v, animals);

// determine size of an array
numberOfElements = lengthOf(v);
```
2020-04-17

Robert Haase

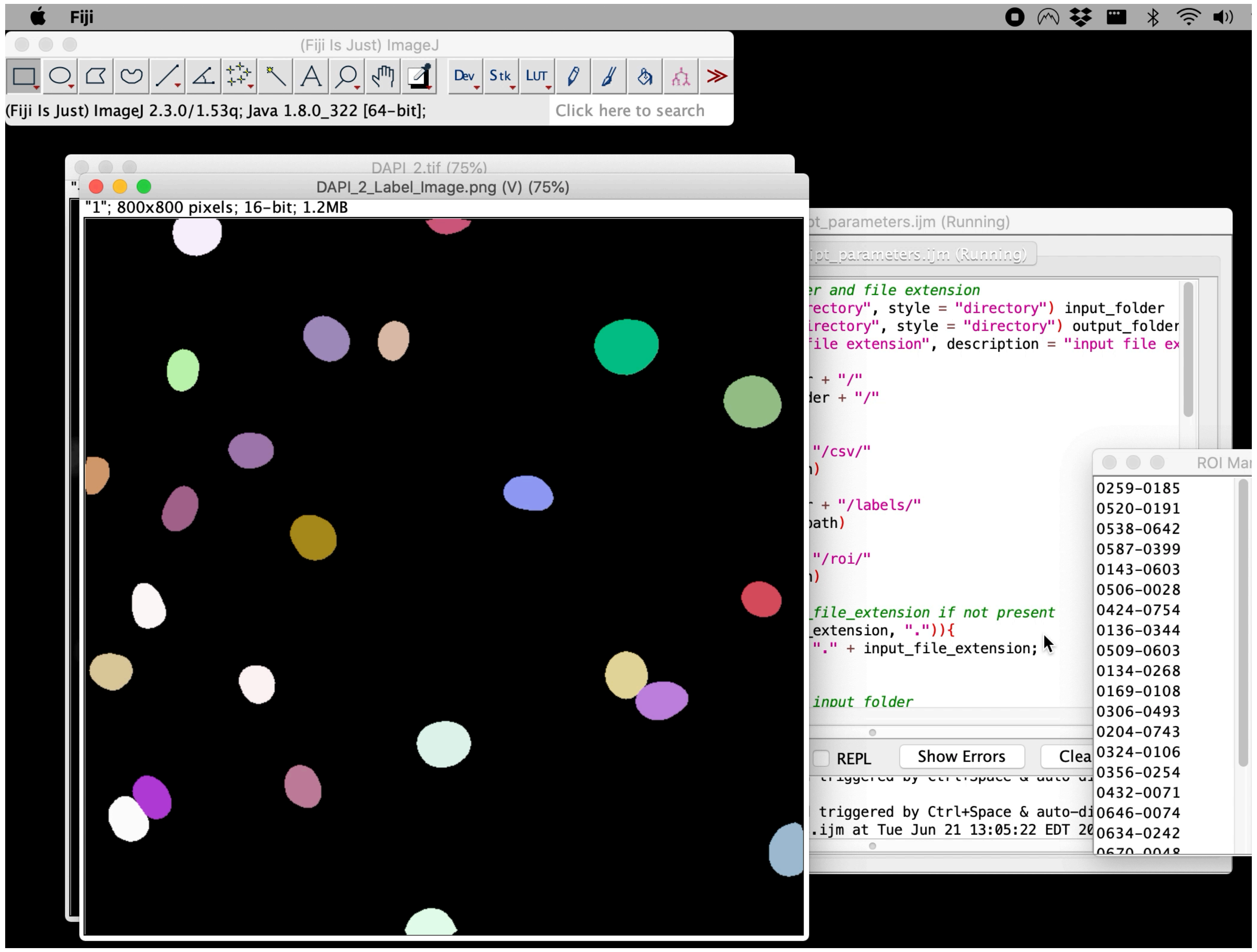
imagej macro

https://imagej.net/scripting/macro

a simple program that automates a series of ImageJ commands (reproducibility)

Example

- *access a folder of nuclei images*
- *for <u>each</u> image:*
    1. *open image*
    2. *segment nuclei*
    3. *measure area and mean*
- *interpret/analyze results*



(Fiji Is Just) ImageJ 2.3.0/1.53q; Java 1.8.0_322 [64-bit];

"1"; 800x800 pixels; 16-bit; 1.2MB

```
er and file extension
rectory", style = "directory") input_folder
irectory", style = "directory") output_folder
file extension", description = "input file ex

 + "/"
der + "/"


 "/csv/"
)

 + "/labels/"
path)

 "/roi/"
)

_file_extension if not present
_extension, ".")){
 "." + input_file_extension;


 input folder
```

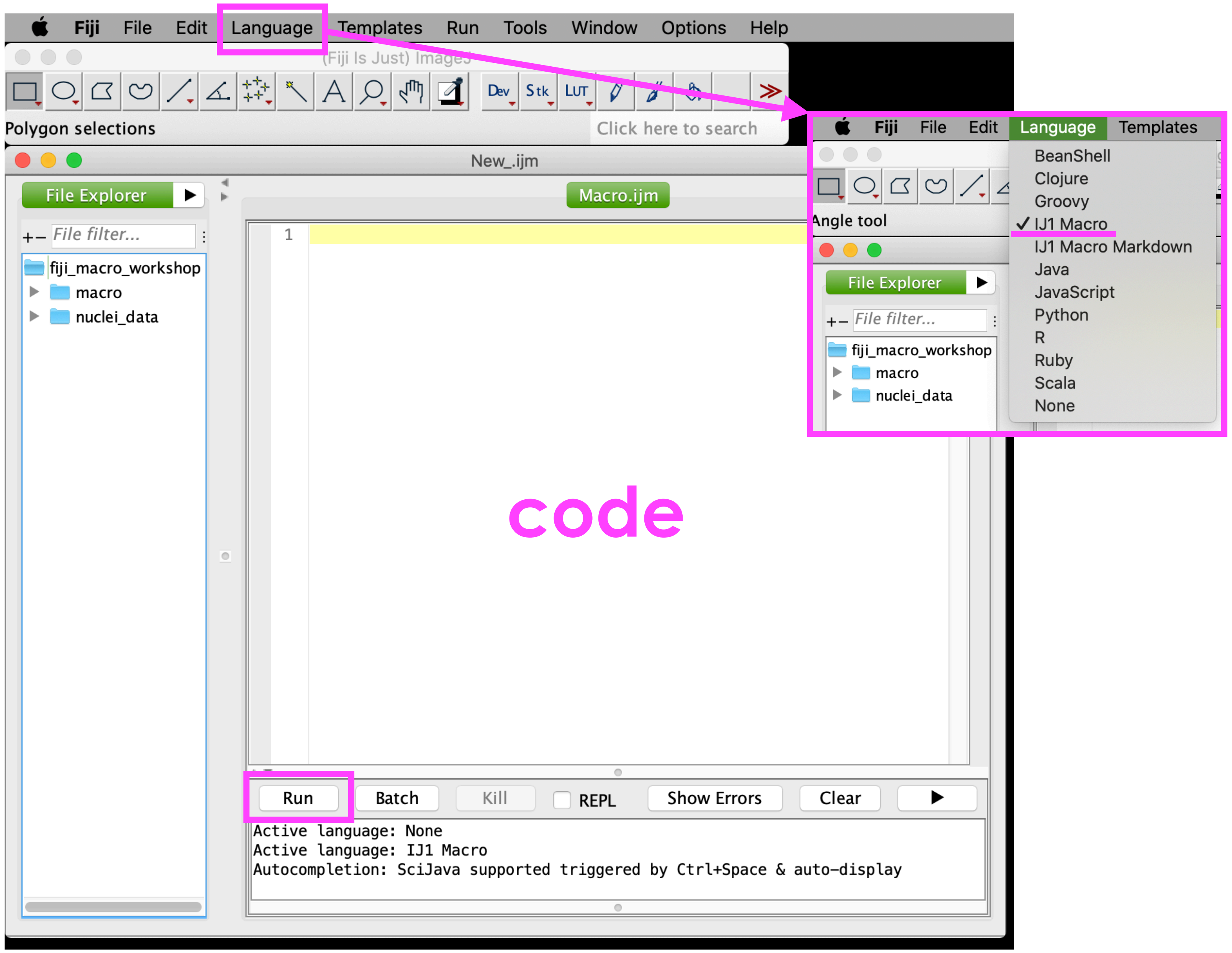
```
 triggered by Ctrl+Space & auto-di
.ijm at Tue Jun 21 13:05:22 EDT 20
```

```
0259-0185
0520-0191
0538-0642
0587-0399
0143-0603
0506-0028
0424-0754
0136-0344
0509-0603
0134-0268
0169-0108
0306-0493
0204-0743
0324-0106
0356-0254
0432-0071
0646-0074
0634-0242
```

script editor

Plugins > New > Macro

File > New > Script…



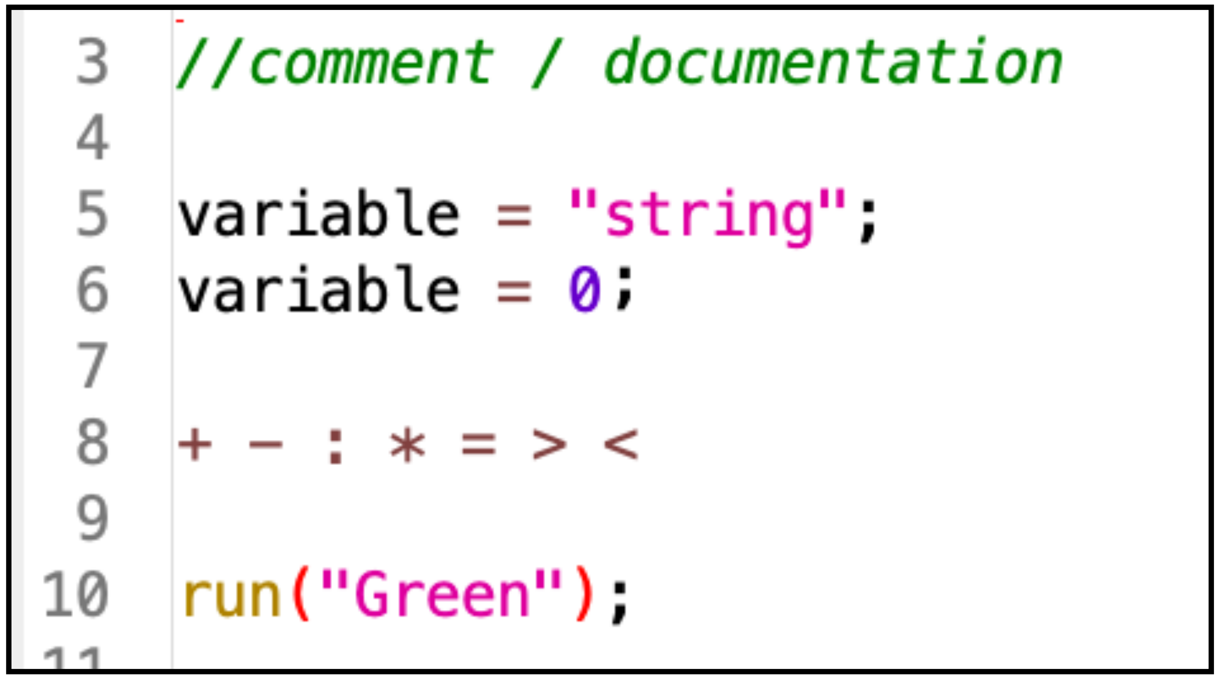
```
Active language: None
Active language: IJ1 Macro
Autocompletion: SciJava supported triggered by Ctrl+Space & auto-display
```

*to open a macro, *drag-and-drop* on the Fiji status bar (or *double-click*)

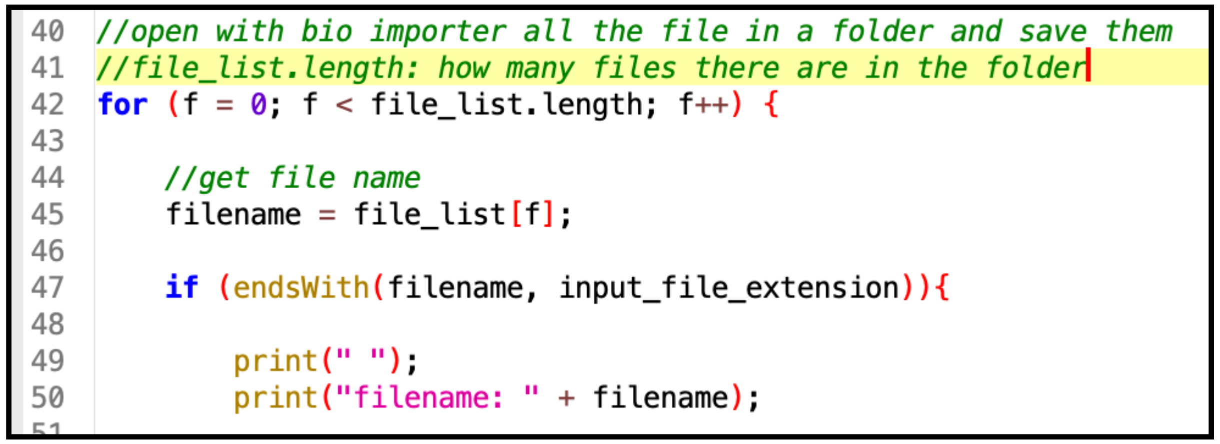
colors depending on type

Colors help you to read and understand the code:

- comments / documentation
- variables
    - strings (text)
    - numbers
    - …
- operators
- commands / action



```
//comment / documentation

variable = "string";
variable = 0;

+ - : * = > <

run("Green");
```

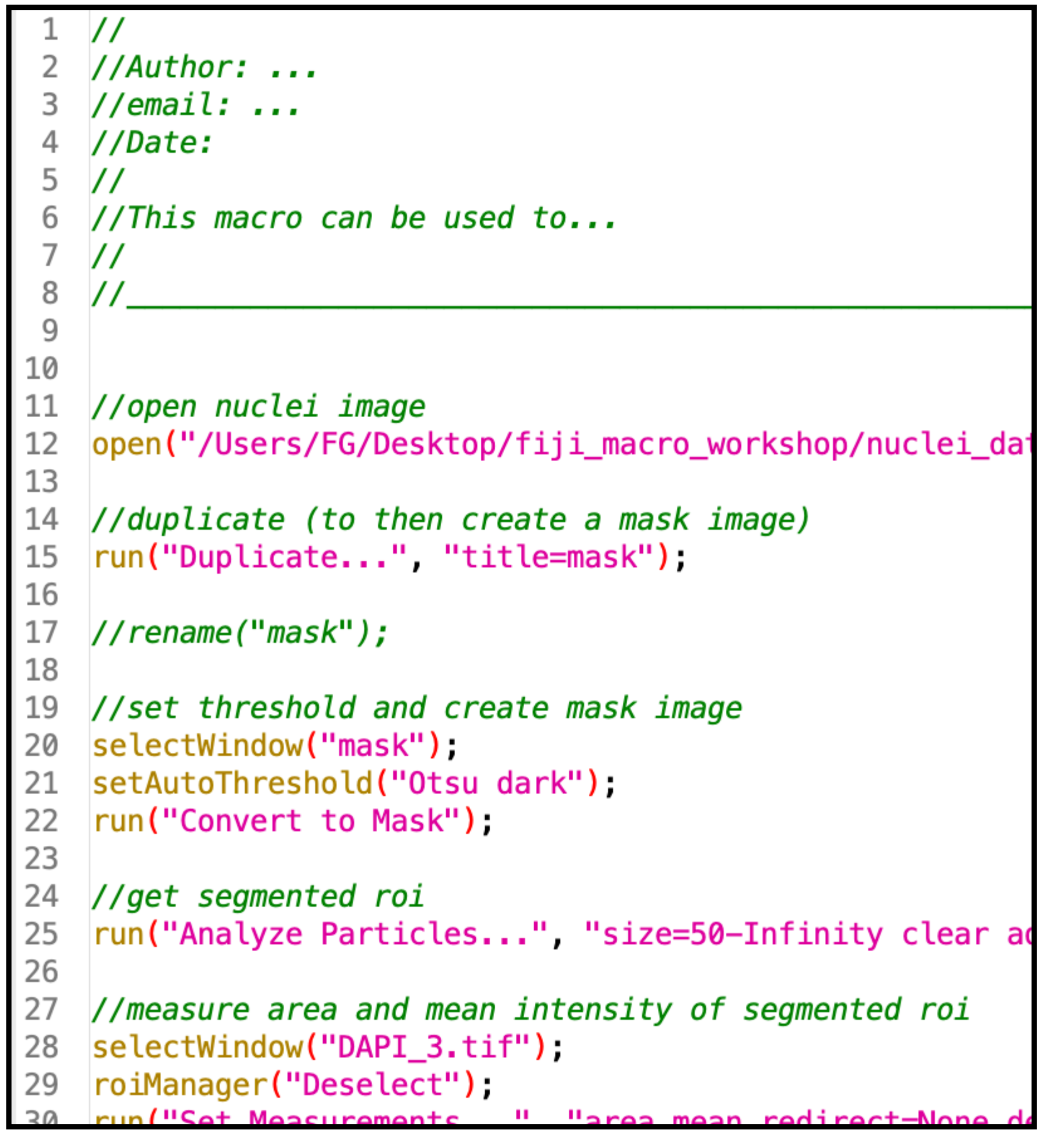


```
//open with bio importer all the file in a folder and save them
//file_list.length: how many files there are in the folder
for (f = 0; f < file_list.length; f++) {

    //get file name
    filename = file_list[f];

    if (endsWith(filename, input_file_extension)){

        print(" ");
        print("filename: " + filename);
```

//comments

Add more **information** to the code.

Every **line** of code that **starts** with // is <u>**not**</u> **executed**.



```
//
//Author: ...
//email: ...
//Date:
//
//This macro can be used to...
//
//________________________________________________


//open nuclei image
open("/Users/FG/Desktop/fiji_macro_workshop/nuclei_dat

//duplicate (to then create a mask image)
run("Duplicate...", "title=mask");

//rename("mask");

//set threshold and create mask image
selectWindow("mask");
setAutoThreshold("Otsu dark");
run("Convert to Mask");

//get segmented roi
run("Analyze Particles...", "size=50-Infinity clear ac

//measure area and mean intensity of segmented roi
selectWindow("DAPI_3.tif");
roiManager("Deselect");
run("Set Measurements...", "area mean redirect=None de
```

To **add** a **comment**, **type** // and then add the text (“*cmd* + /“ or “*ctrl* + /“).

Comments can be useful for:

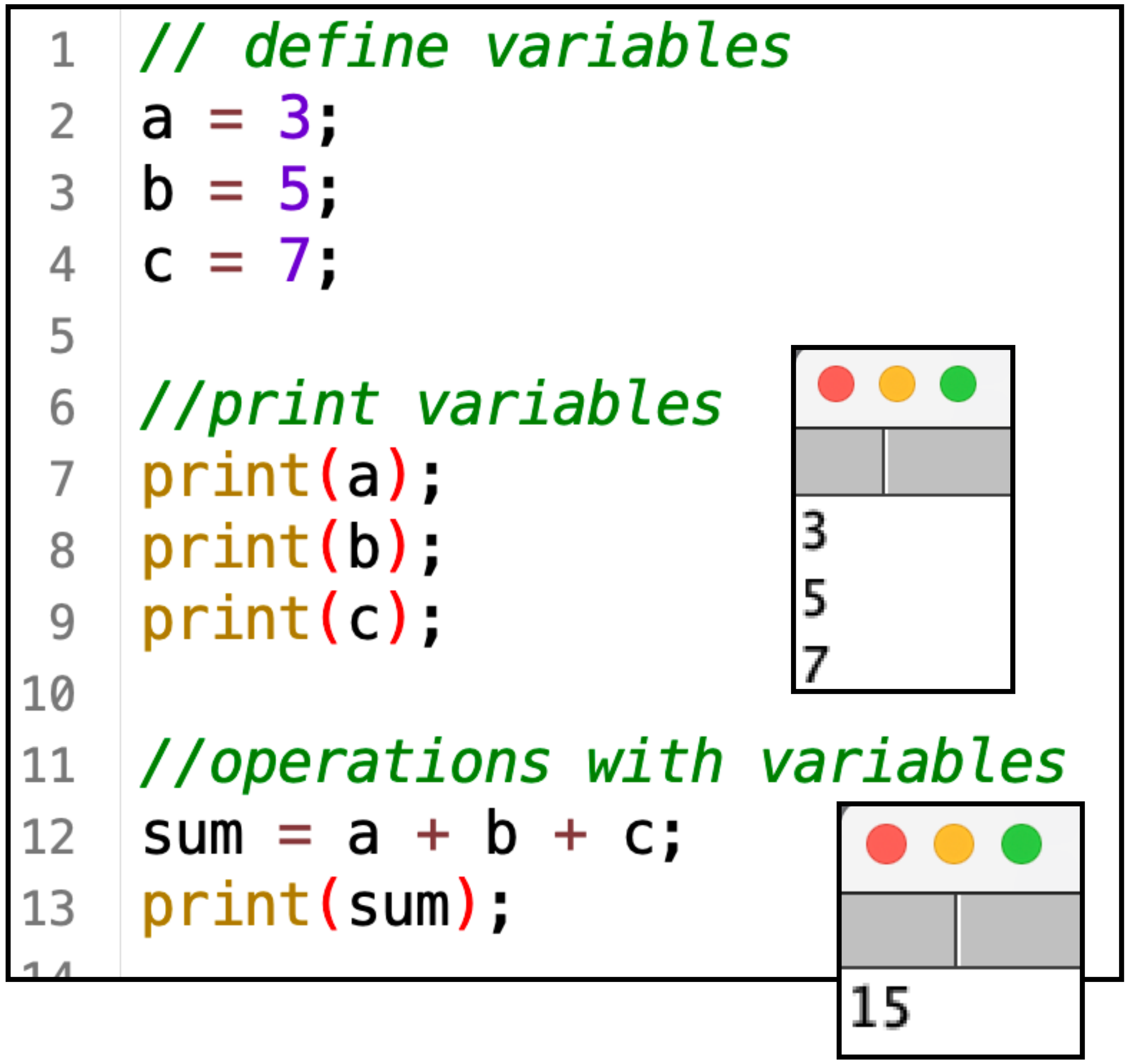
- *add **author info** and **aim** of the **macro**.*
- ***prevent** lines of code to be **executed** (cmd + /).*
- *code **documentation**:*
  explain/describe a specific line/block of code.
- …

Image Analysis Collaboratory

variables

names you give to computer memory locations that you can use to ***store values****.*

numbers

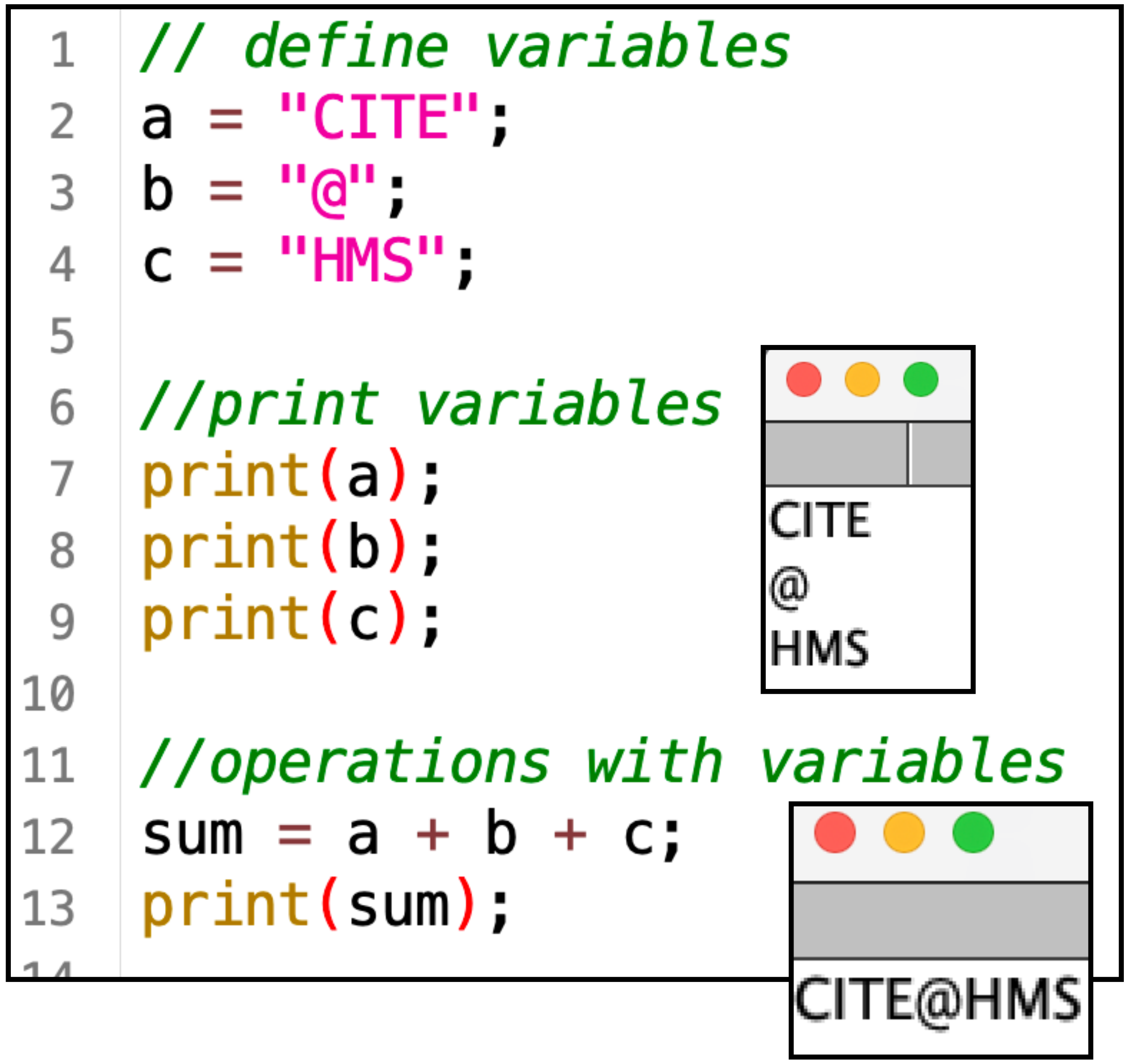


```
// define variables
a = 3;
b = 5;
c = 7;

//print variables
print(a);
print(b);
print(c);

//operations with variables
sum = a + b + c;
print(sum);
```

```
3
5
7
```

```
15
```

strings (text)



```
// define variables
a = "CITE";
b = "@";
c = "HMS";

//print variables
print(a);
print(b);
print(c);

//operations with variables
sum = a + b + c;
print(sum);
```

```
CITE
@
HMS
```

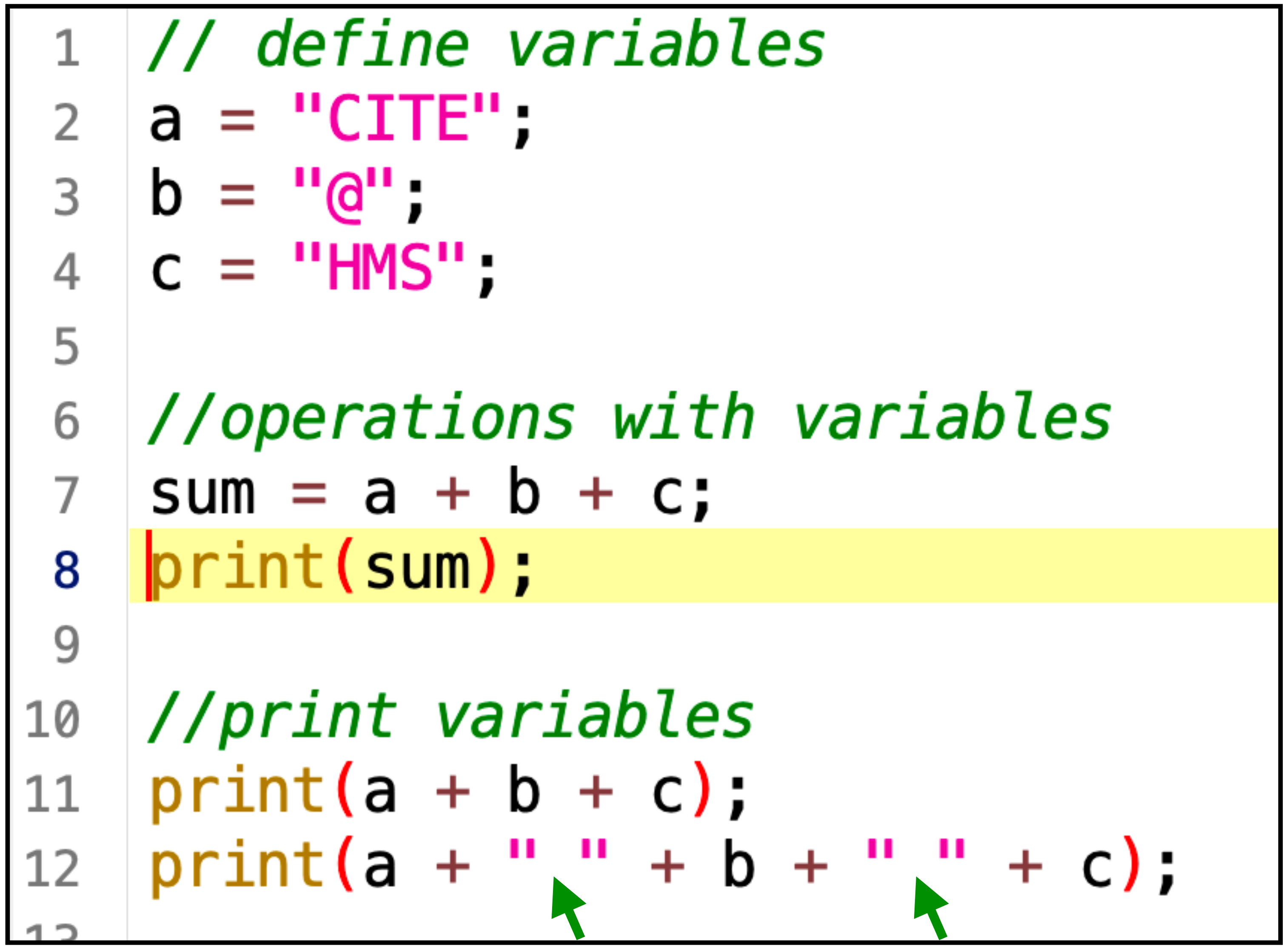
```
CITE@HMS
```

Image Analysis Collaboratory

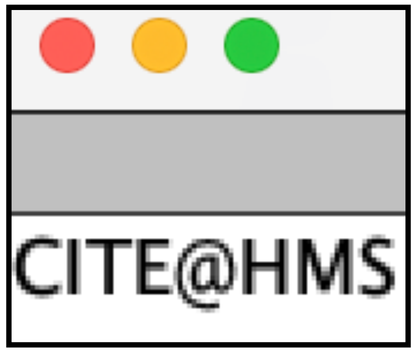
variables

*names you give to computer memory locations that you can use to **store values**.*

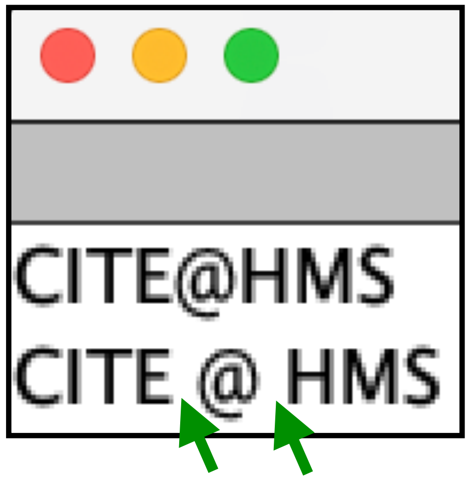
strings (text)



```
// define variables
a = "CITE";
b = "@";
c = "HMS";

//operations with variables
sum = a + b + c;
print(sum);

//print variables
print(a + b + c);
print(a + " " + b + " " + c);
```



```
CITE@HMS
```



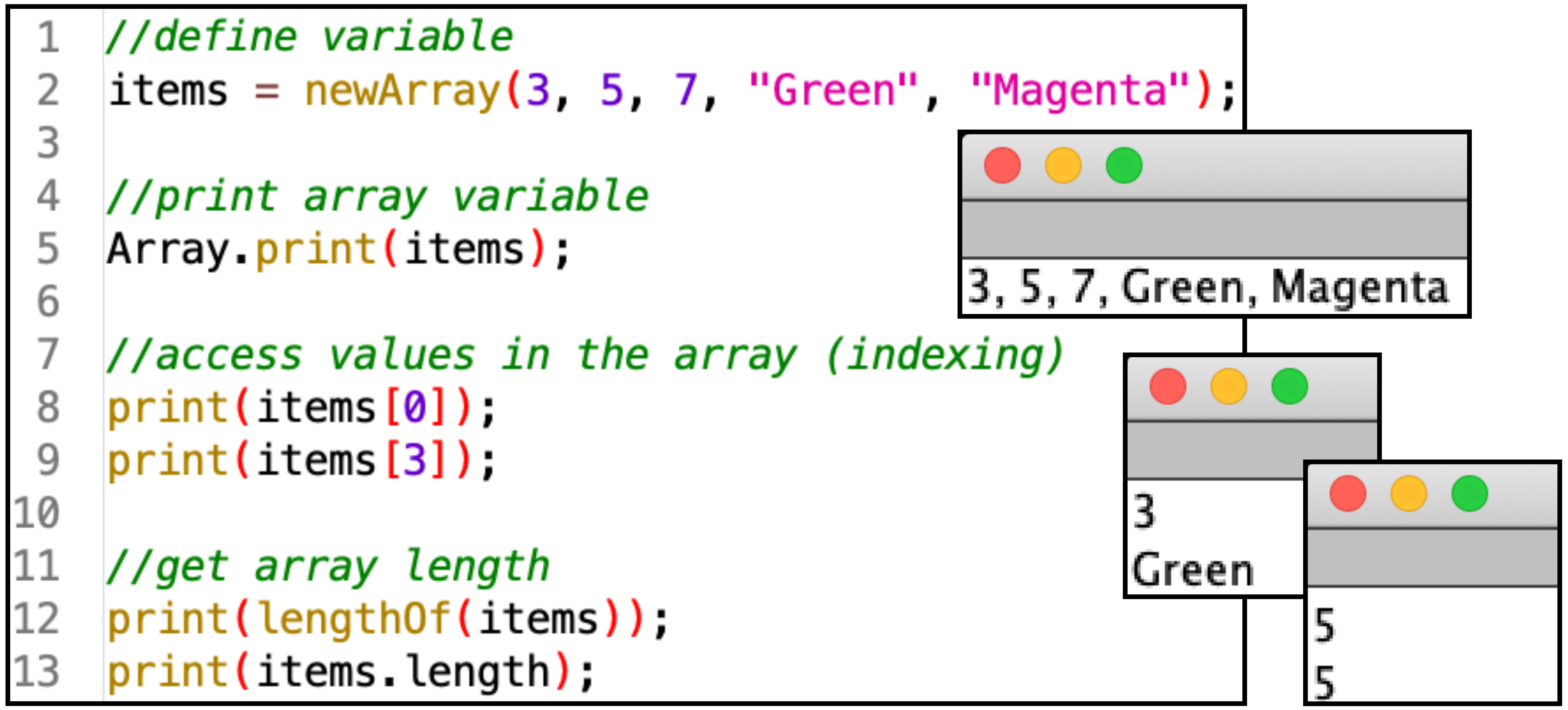
```
CITE@HMS
CITE @ HMS
```

Image Analysis Collaboratory

variables

*names you give to computer memory locations that you can use to **store values**.*

arrays - *variables where you can **store multiple values.***



```
//define variable
items = newArray(3, 5, 7, "Green", "Magenta");

//print array variable
Array.print(items);

//access values in the array (indexing)
print(items[0]);
print(items[3]);

//get array length
print(lengthOf(items));
print(items.length);
```

```
3, 5, 7, Green, Magenta
```

```
3
Green
```

```
5
5
```

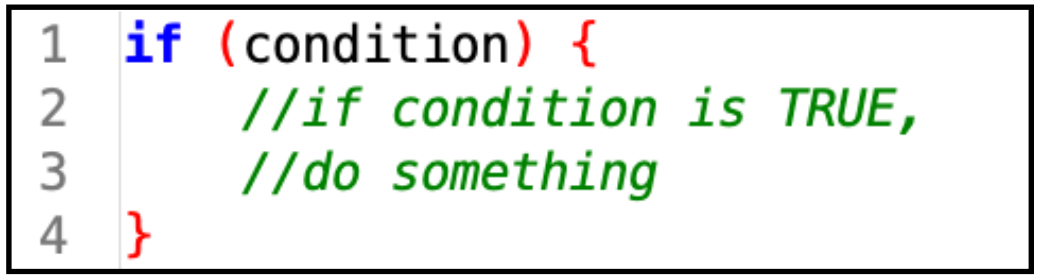
*You can **access** an **array element** by referring to its **index** number.*

items	3	5	7	"Green"	"Magenta
items index	**0**	1	2	**3**	4

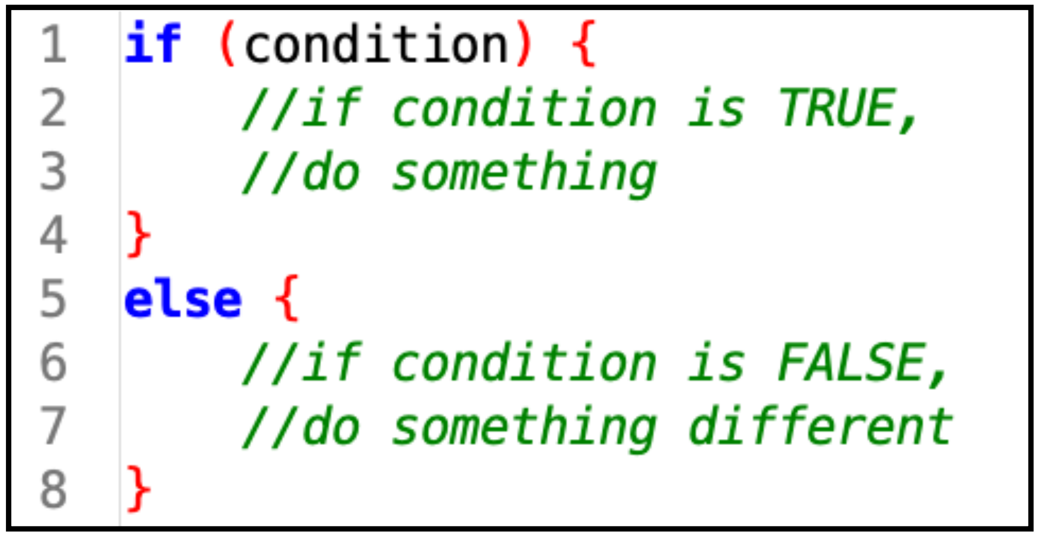
conditions

if…else…

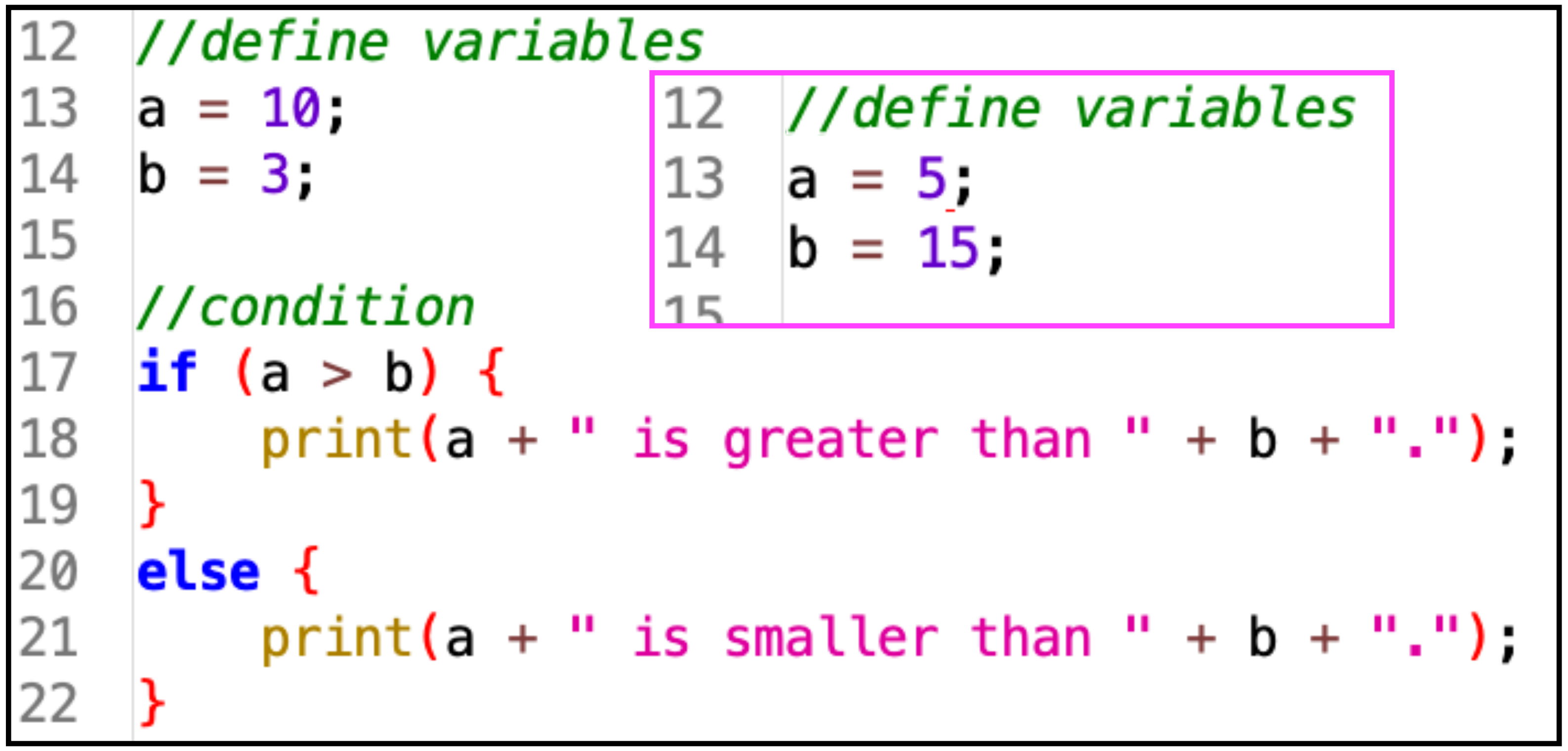
Execute the code only in specific conditions.



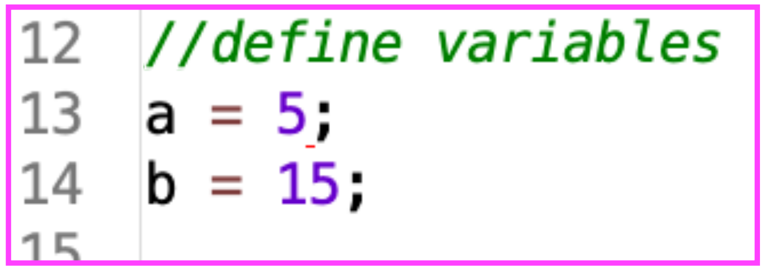
```
if (condition) {
    //if condition is TRUE,
    //do something
}
```



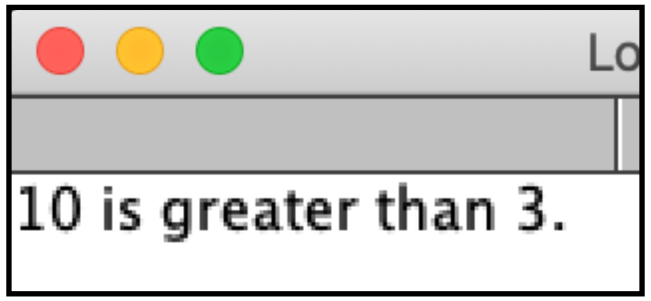
```
if (condition) {
    //if condition is TRUE,
    //do something
}
else {
    //if condition is FALSE,
    //do something different
}
```



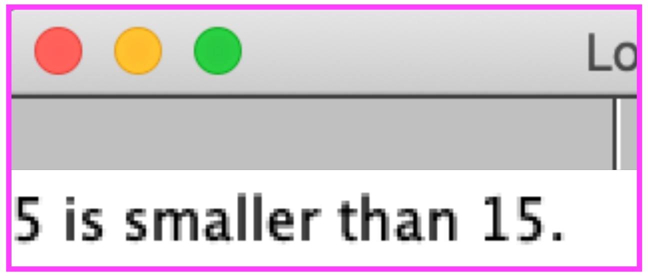
```
//define variables
a = 10;
b = 3;

//condition
if (a > b) {
    print(a + " is greater than " + b + ".");
}
else {
    print(a + " is smaller than " + b + ".");
}
```



```
//define variables
a = 5;
b = 15;
```



```
10 is greater than 3.
```

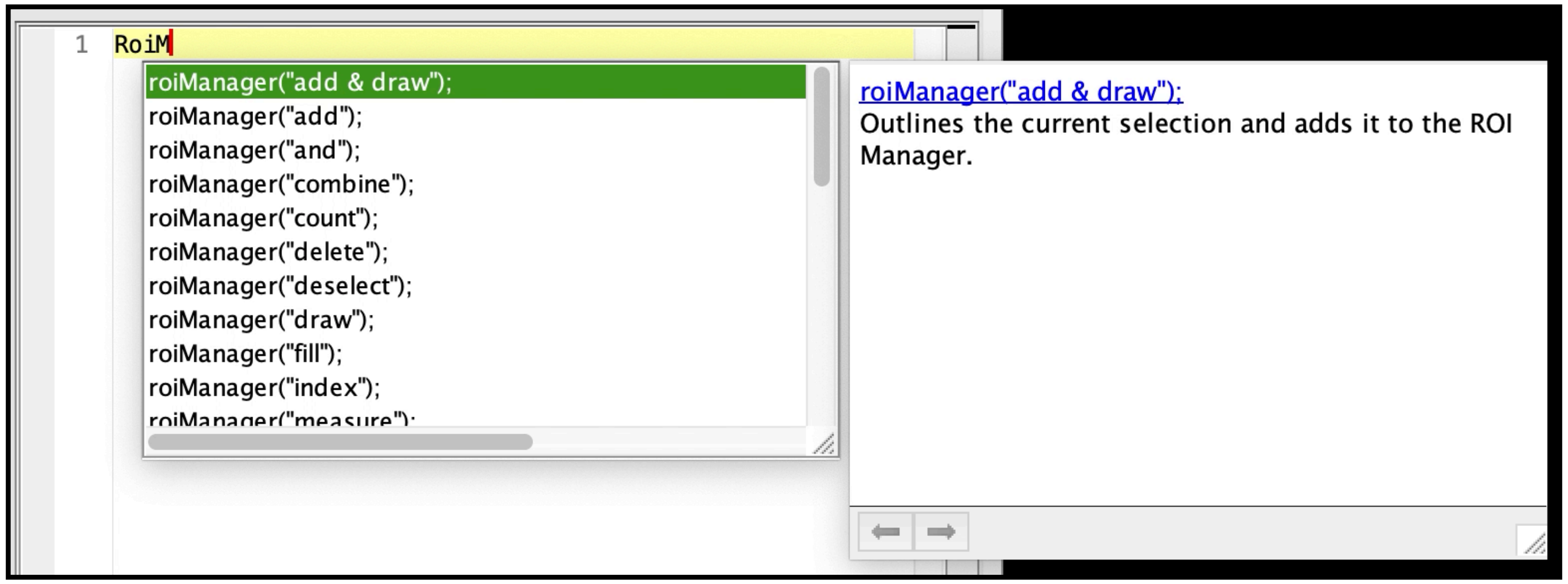


```
5 is smaller than 15.
```

Image Analysis Collaboratory

code auto-completion

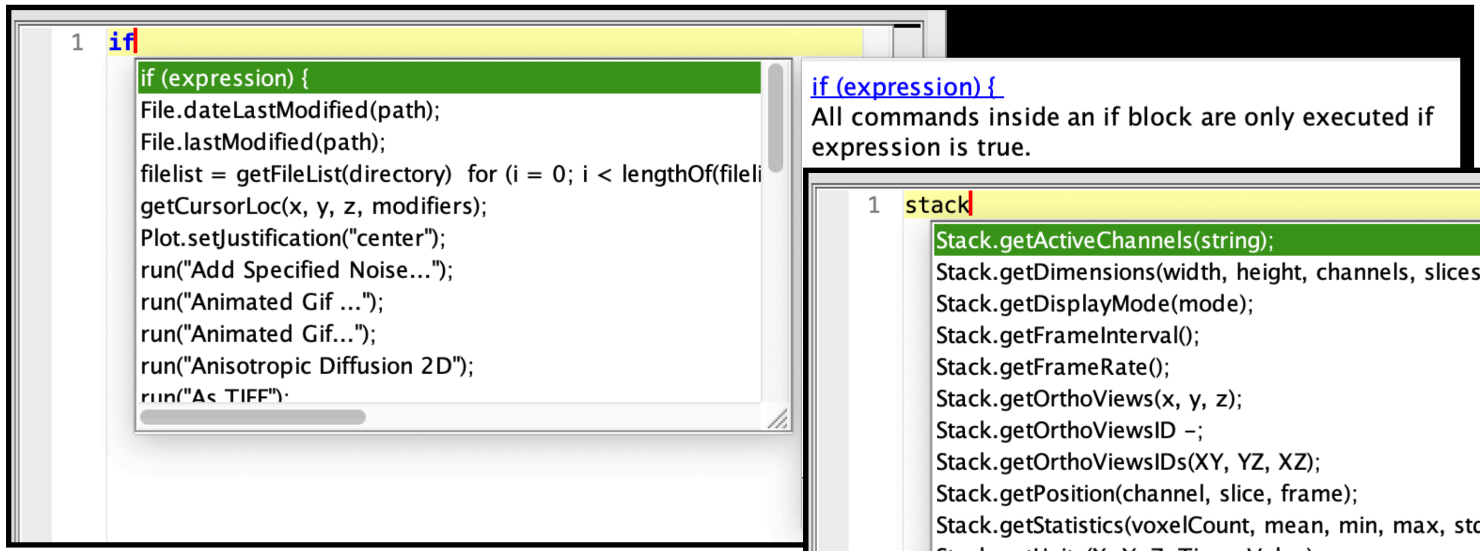
https://imagej.net/ij/developer/macro/functions.html



```
RoiM
```

roiManager("add & draw");
roiManager("add");
roiManager("and");
roiManager("combine");
roiManager("count");
roiManager("delete");
roiManager("deselect");
roiManager("draw");
roiManager("fill");
roiManager("index");
roiManager("measure");

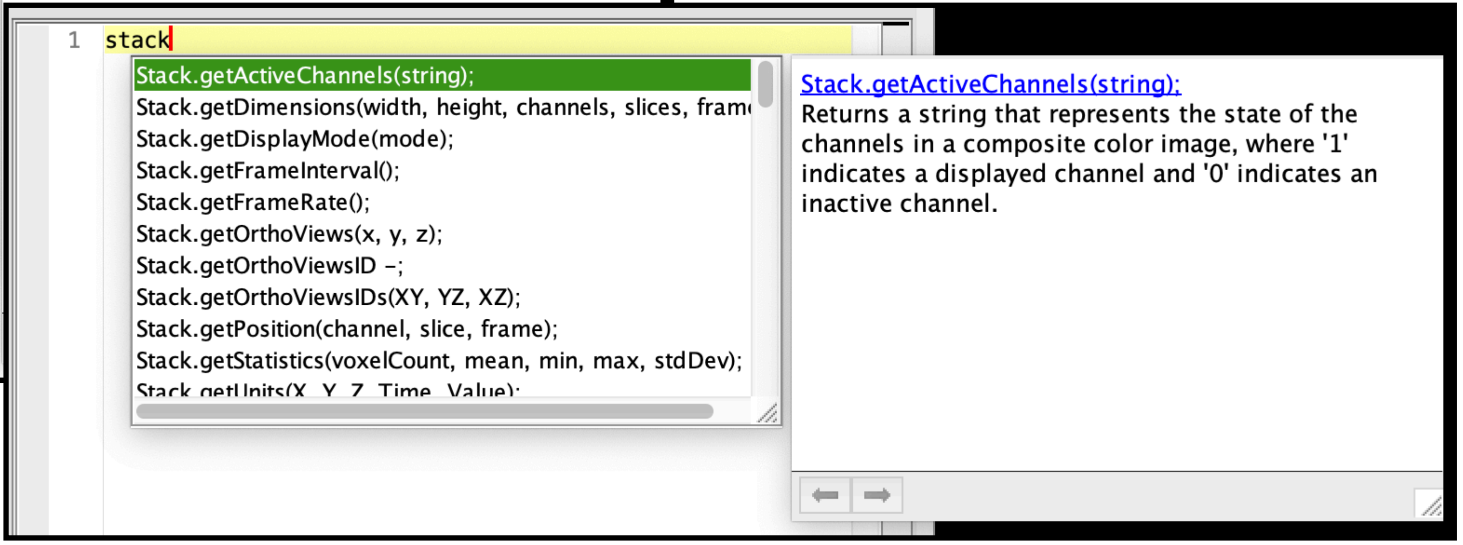
roiManager("add & draw");
Outlines the current selection and adds it to the ROI Manager.



```
if
```

if (expression) {
File.dateLastModified(path);
File.lastModified(path);
filelist = getFileList(directory) for (i = 0; i < lengthOf(fileli
getCursorLoc(x, y, z, modifiers);
Plot.setJustification("center");
run("Add Specified Noise...");
run("Animated Gif ...");
run("Animated Gif...");
run("Anisotropic Diffusion 2D");
run("As TIFF");

if (expression) {
All commands inside an if block are only executed if expression is true.



```
stack
```

Stack.getActiveChannels(string);
Stack.getDimensions(width, height, channels, slices, fram
Stack.getDisplayMode(mode);
Stack.getFrameInterval();
Stack.getFrameRate();
Stack.getOrthoViews(x, y, z);
Stack.getOrthoViewsID –;
Stack.getOrthoViewsIDs(XY, YZ, XZ);
Stack.getPosition(channel, slice, frame);
Stack.getStatistics(voxelCount, mean, min, max, stdDev);
Stack.getUnits(X, Y, Z, Time, Value);

Stack.getActiveChannels(string);
Returns a string that represents the state of the channels in a composite color image, where '1' indicates a displayed channel and '0' indicates an inactive channel.

Image Analysis Collaboratory

let’s try!

1. create a **variable** named ***items*** containing **5** random **numbers** (create an array).

2. **print** the **items** variable (note: it is an array)

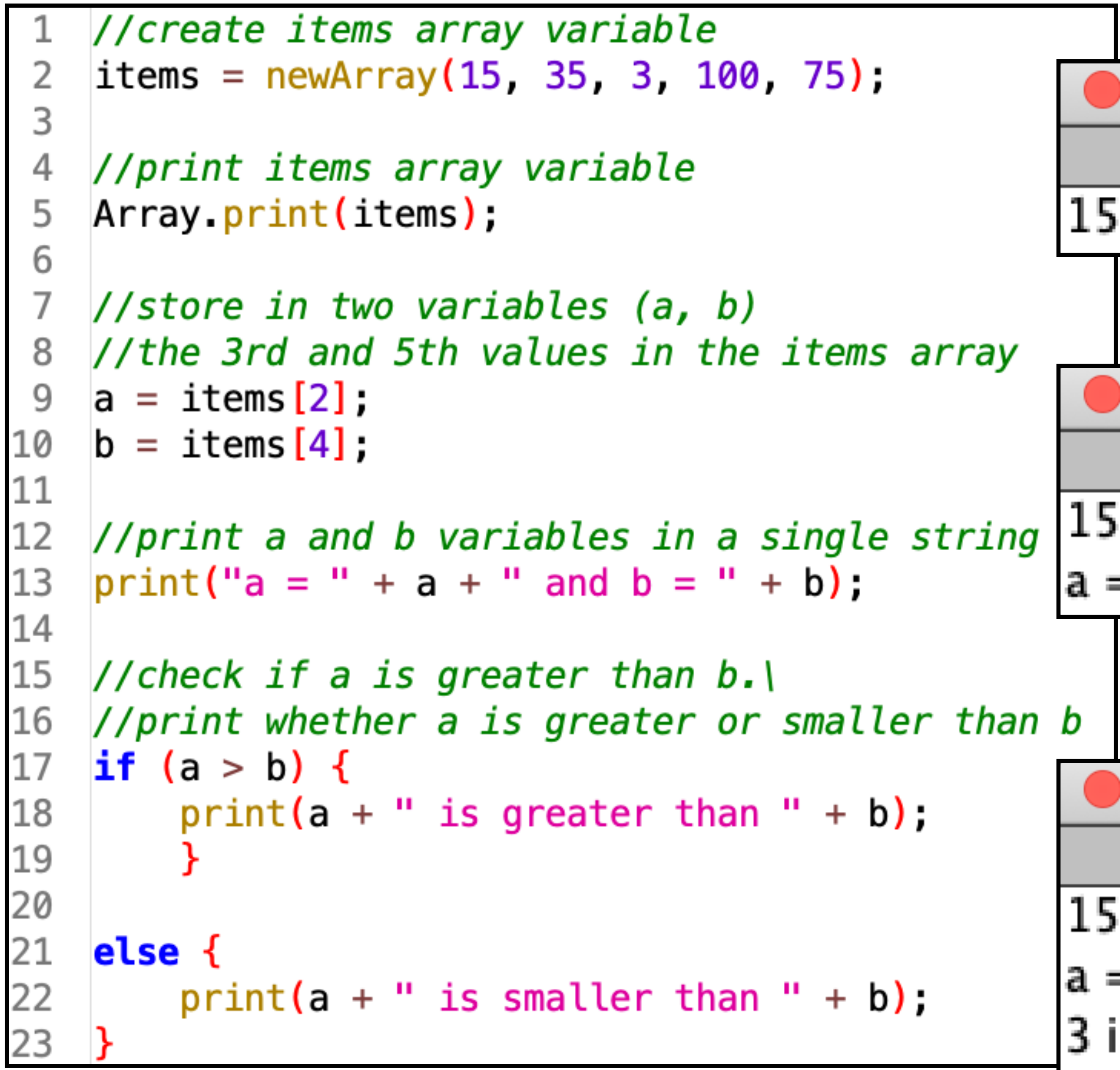
3. create two more variables, named **a** and **b**, and **store** in **a** the ***3rd*** value of the **item array** and in **b** the ***5th*** value of the **item array**.

4. **print a** and **b** variables in a single string (e.g. the output should be something like "*a* = *x and b* = *y*" or "*a* = *x, b* = *y*”).

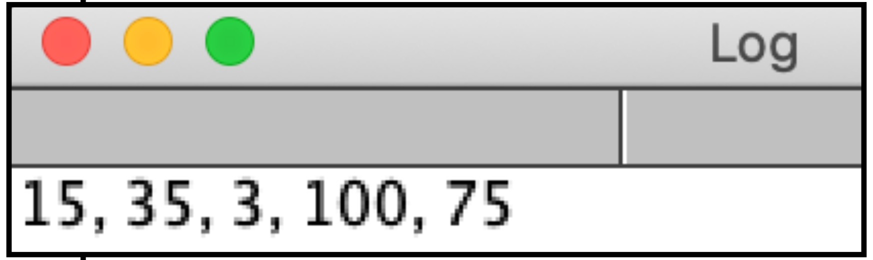
5. **check** and **print** whether **a is greater or smaller than b** (use if… else..)

Image Analysis Collaboratory

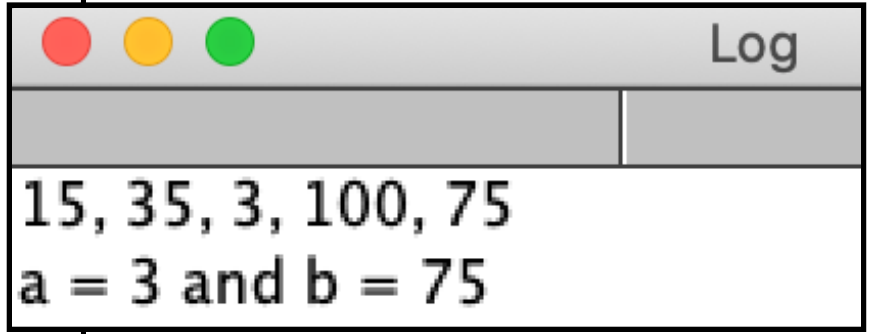
let's try! - solution



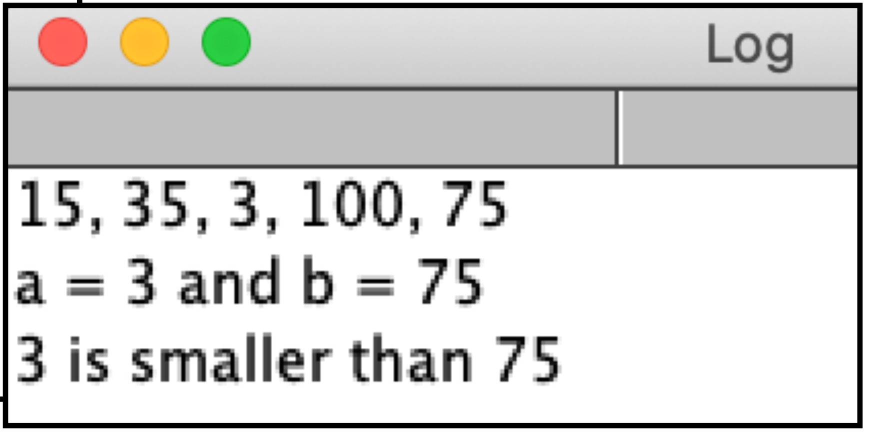
```
//create items array variable
items = newArray(15, 35, 3, 100, 75);

//print items array variable
Array.print(items);

//store in two variables (a, b)
//the 3rd and 5th values in the items array
a = items[2];
b = items[4];

//print a and b variables in a single string
print("a = " + a + " and b = " + b);

//check if a is greater than b.\
//print whether a is greater or smaller than b
if (a > b) {
    print(a + " is greater than " + b);
    }

else {
    print(a + " is smaller than " + b);
}
```



```
15, 35, 3, 100, 75
```



```
15, 35, 3, 100, 75
a = 3 and b = 75
```



```
15, 35, 3, 100, 75
a = 3 and b = 75
3 is smaller than 75
```

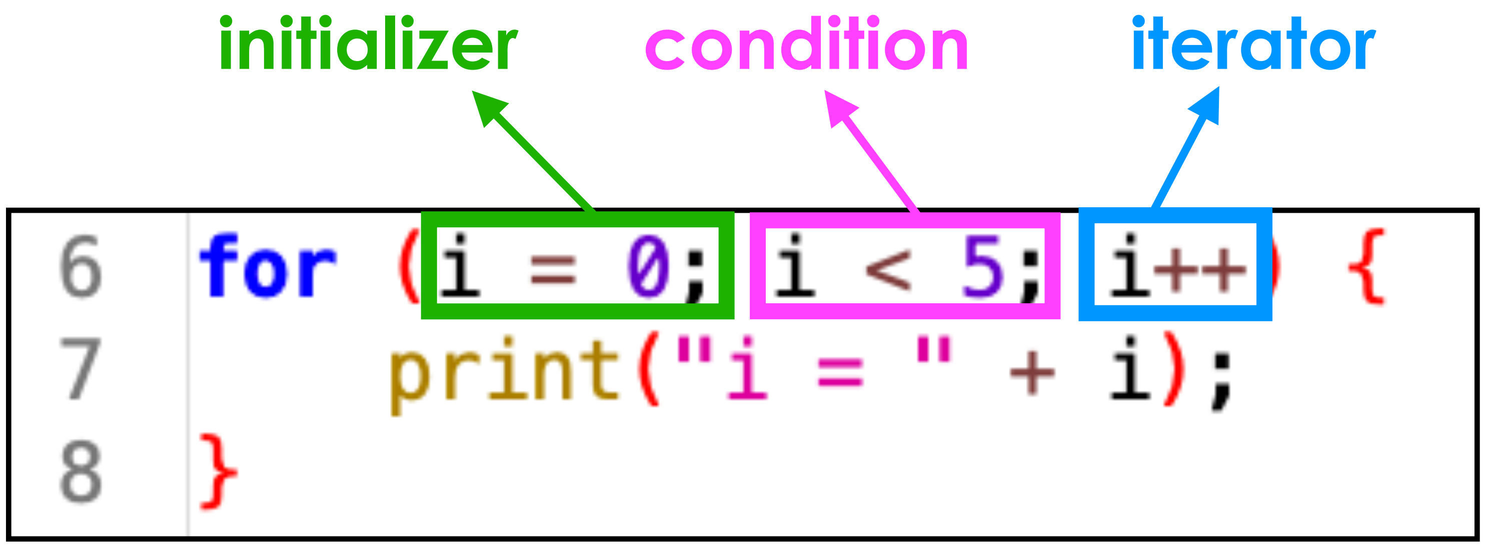
for loops

execute some lines of code ***for* n times**.

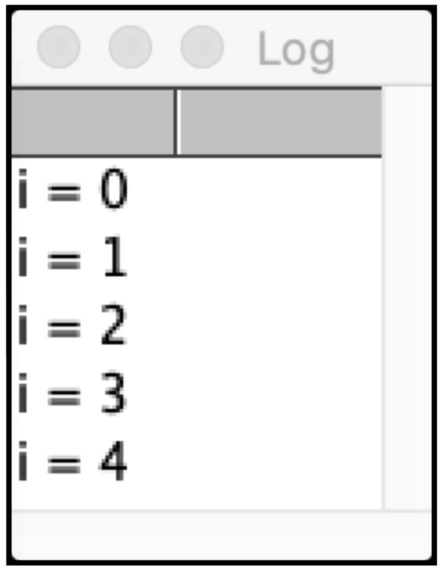
```
for (initializer; condition; iterator) {
    // do something n times
    // until the condition is FALSE
}
```

for loops

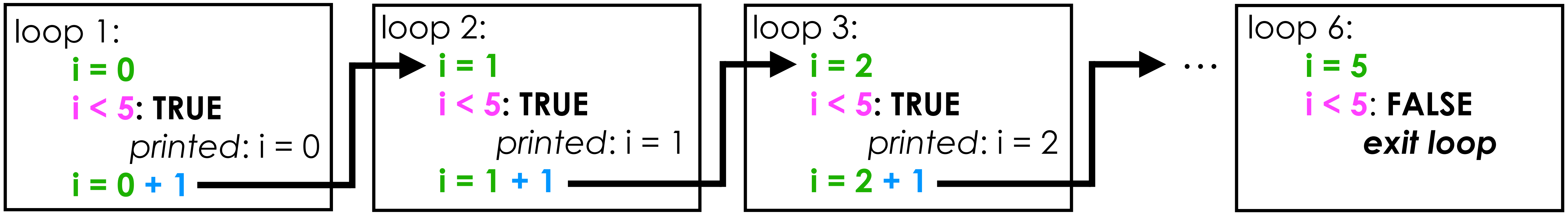
execute some lines of code ***for* n times.**



```
for (i = 0; i < 5; i++) {
    print("i = " + i);
}
```



```
i = 0
i = 1
i = 2
i = 3
i = 4
```



*Within the loop, use **break** to **exit the loop** before the end or **continue** to **skip to the next loop**.*

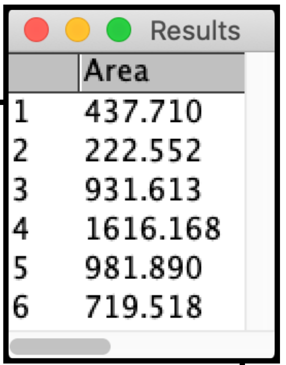
Image Analysis Collaboratory

for loops

execute some lines of code ***for* n times**.

```
//loop through the roi of the ROI Manager
for (i = 0; i < roiManager("count"); i++) {
    roiManager("select", i);
    // do something here;
}
```

```
//loop through a Result table
for (i = 0; i < nResults(); i++) {
    value = getResult("Area", i);
    print(value);
}
```


Results

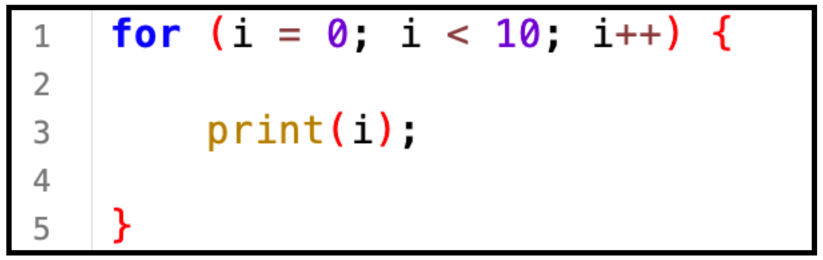
	Area
1	437.710
2	222.552
3	931.613
4	1616.168
5	981.890
6	719.518

```
//loop through the slicies of a stack
for (i = 1; i <= nSlices; i++) {
    setSlice(i);
    // do something here
}
```

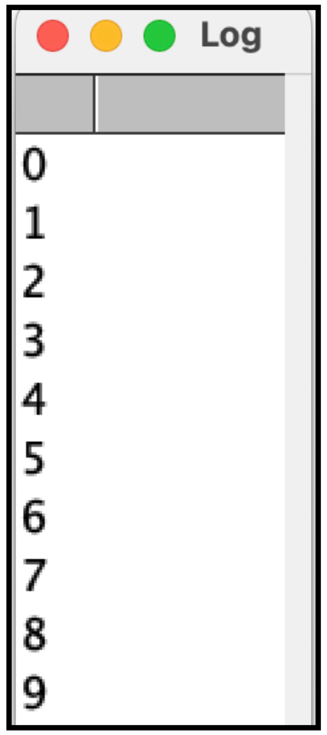
```
//loop through files in a folder
filelist = getFileList(directory)
for (i = 0; i < lengthOf(filelist); i++) {
    print(filelist[i]);
}
```

for loops

for loop that prints the iterator

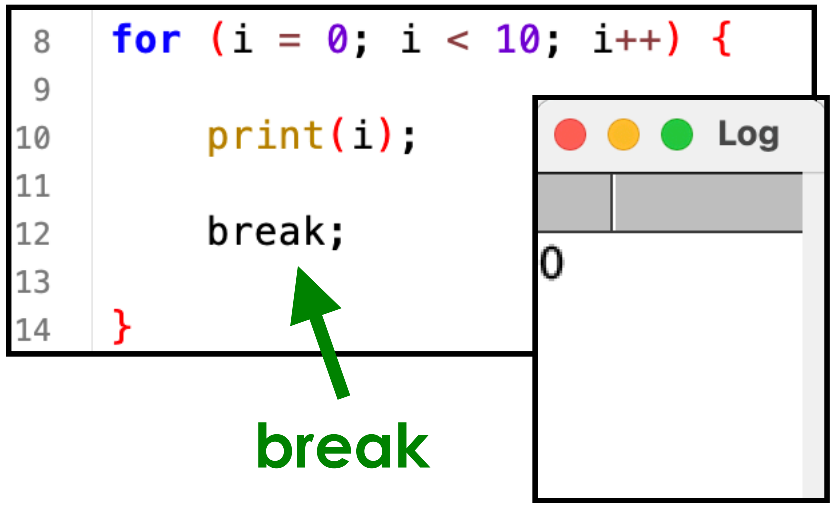


```
for (i = 0; i < 10; i++) {

     print(i);

}
```



```
0
1
2
3
4
5
6
7
8
9
```

exit the loop before the end

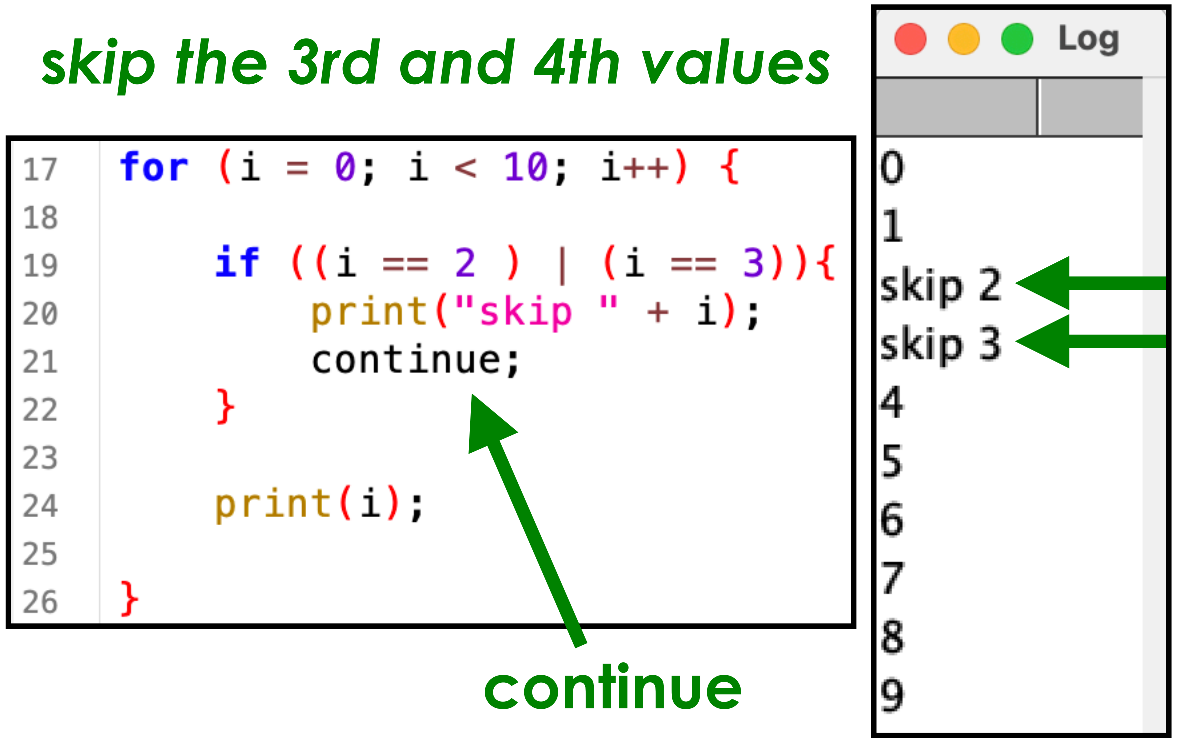


```
for (i = 0; i < 10; i++) {

     print(i);

     break;

}
```

```
0
```

skip to the next iteration

skip the 3rd and 4th values



```
for (i = 0; i < 10; i++) {

     if ((i == 2 ) | (i == 3)){
          print("skip " + i);
          continue;
     }

     print(i);

}
```

```
0
1
skip 2
skip 3
4
5
6
7
8
9
```

let’s try

1. create a **variable** named ***items*** containing **6** random **numbers**.

2. **print** the **items** variable.

3. create another variable named **a** and **store** the **1st** value of the **item** variable.

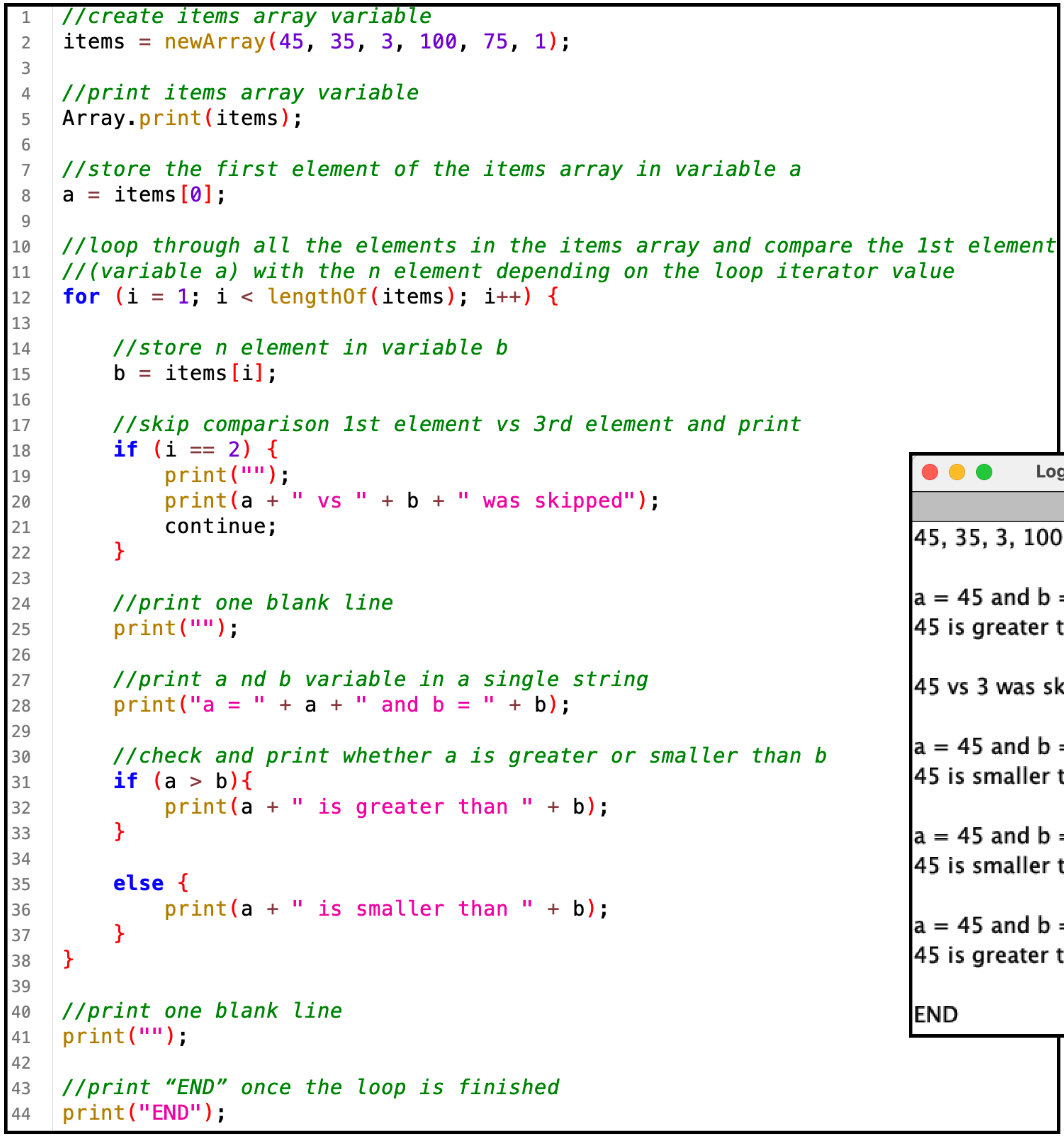
4. **loop through** all the elements in the **items** array: the goal is to **compare** the **1st element** (variable **a**) **with** the **n element depending on the loop *iterator value***. Within the loop you should:
    a. create a variable named **b and** store the **n element**.
    b. **skip** the **comparison:**
        - **1st element vs 1st element** (use for loop *initializer* value)
        - **1st element vs 3rd element**; in this case print first a blank line and then that you skipped this comparison (e.g. *“45 vs 3 was skipped”)*
    c. **print** one **blank line**.
    d. **print a** and **b** variables in a single string (e.g. "*a* = x *and b* = *y*”).
    e. **check** and **print** whether **a is greater or smaller than b**.

5. **print** one **blank line**.

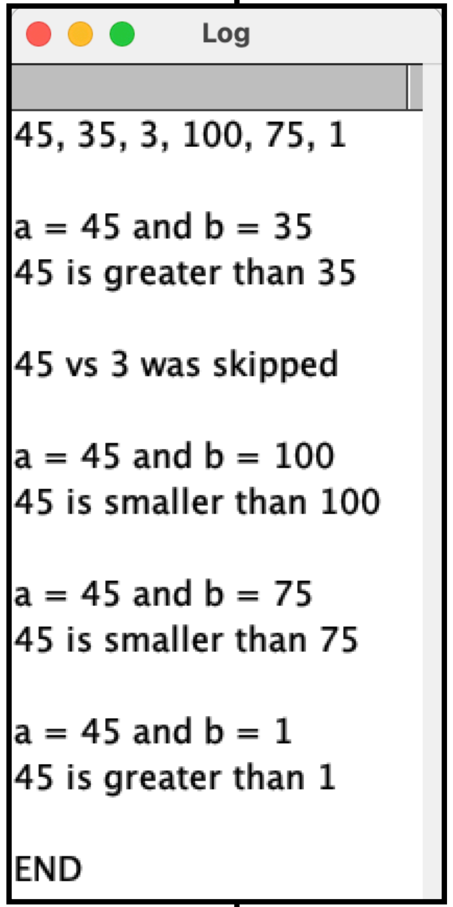
6. **print “END”** once the loop is finished.

Image Analysis Collaboratory

let’s try - solution



```
//create items array variable
items = newArray(45, 35, 3, 100, 75, 1);

//print items array variable
Array.print(items);

//store the first element of the items array in variable a
a = items[0];

//loop through all the elements in the items array and compare the 1st element
//(variable a) with the n element depending on the loop iterator value
for (i = 1; i < lengthOf(items); i++) {

    //store n element in variable b
    b = items[i];

    //skip comparison 1st element vs 3rd element and print
    if (i == 2) {
        print("");
        print(a + " vs " + b + " was skipped");
        continue;
    }

    //print one blank line
    print("");

    //print a nd b variable in a single string
    print("a = " + a + " and b = " + b);

    //check and print whether a is greater or smaller than b
    if (a > b){
        print(a + " is greater than " + b);
    }

    else {
        print(a + " is smaller than " + b);
    }
}

//print one blank line
print("");

//print “END” once the loop is finished
print("END");
```

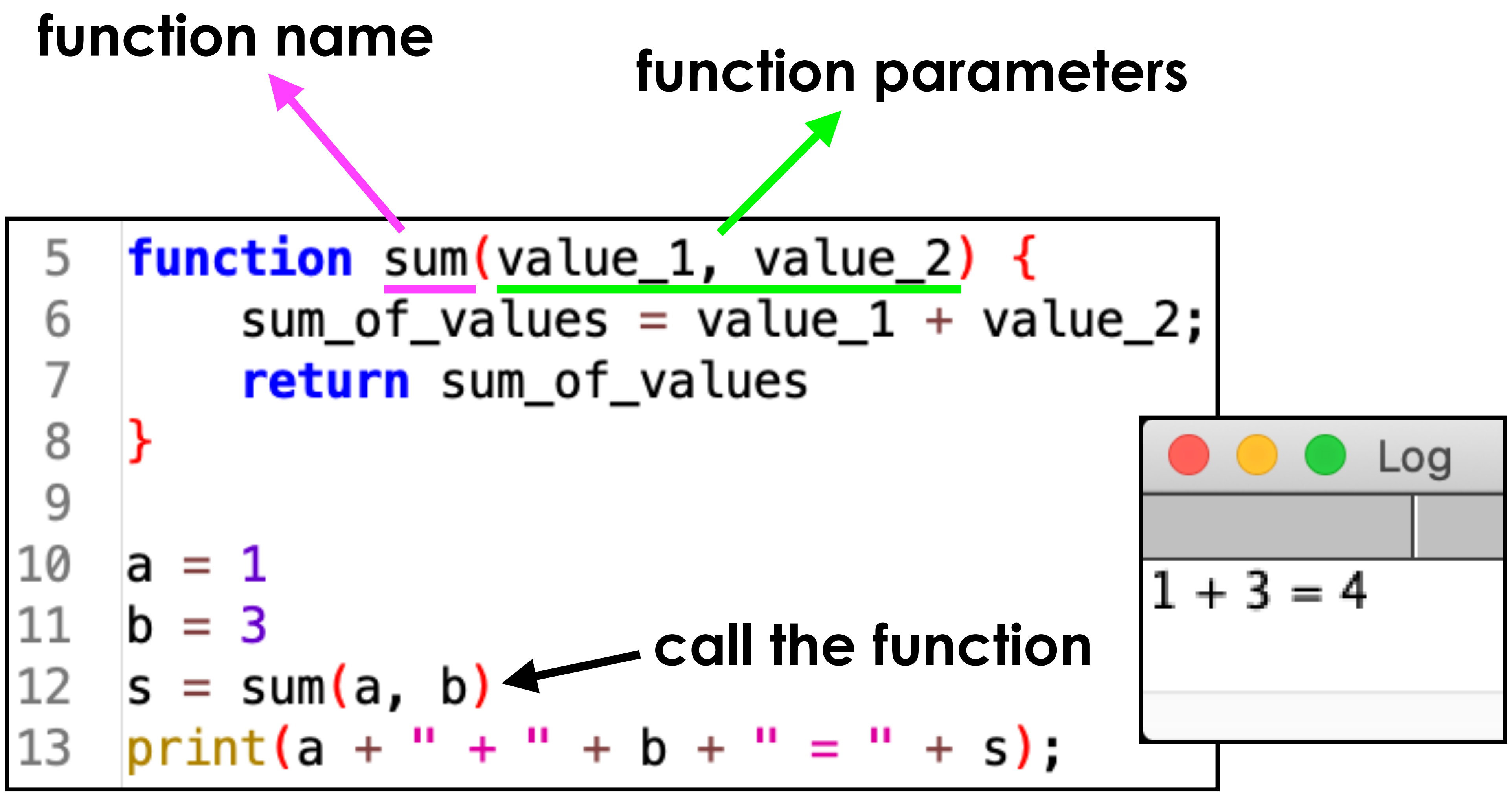


Log

```
45, 35, 3, 100, 75, 1

a = 45 and b = 35
45 is greater than 35

45 vs 3 was skipped

a = 45 and b = 100
45 is smaller than 100

a = 45 and b = 75
45 is smaller than 75

a = 45 and b = 1
45 is greater than 1

END
```

functions

If there are **lines of code** that are **repetitive** you can **replace** the code **with** a **function**.



```
function sum(value_1, value_2) {
    sum_of_values = value_1 + value_2;
    return sum_of_values
}

a = 1
b = 3
s = sum(a, b)
print(a + " + " + b + " = " + s);
```

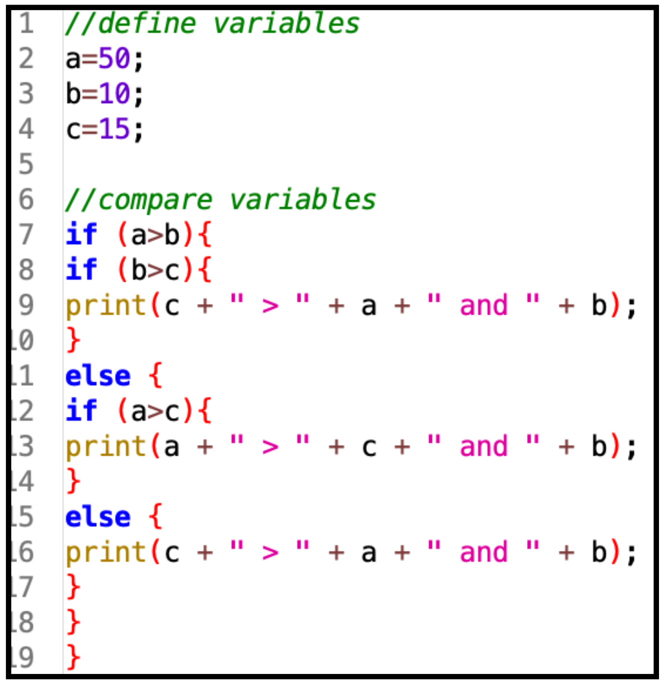
```
1 + 3 = 4
```

Image Analysis Collaboratory

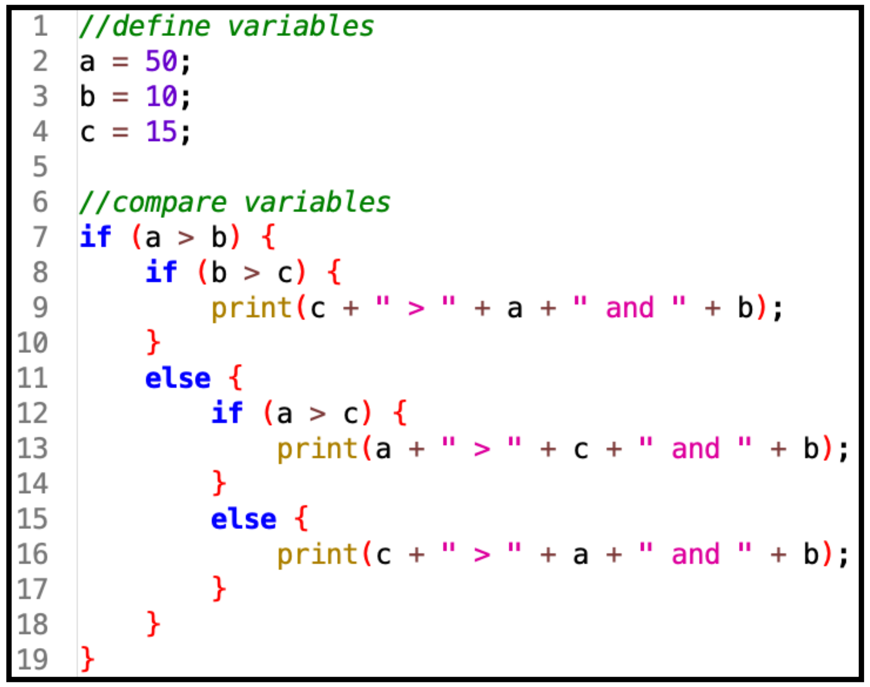
how to write the code

some useful tips...

- use **comments** to describe code lines/ blocks.
- **empty lines** to separate code lines/blocks.
- **one command per line**.
- give **variables** meaningful **names** (c vs channel) and place them **at** the **beginning** of the **code** for easy access.
- **space between operators** (a=1 vs a = 1).
- **indentation**.



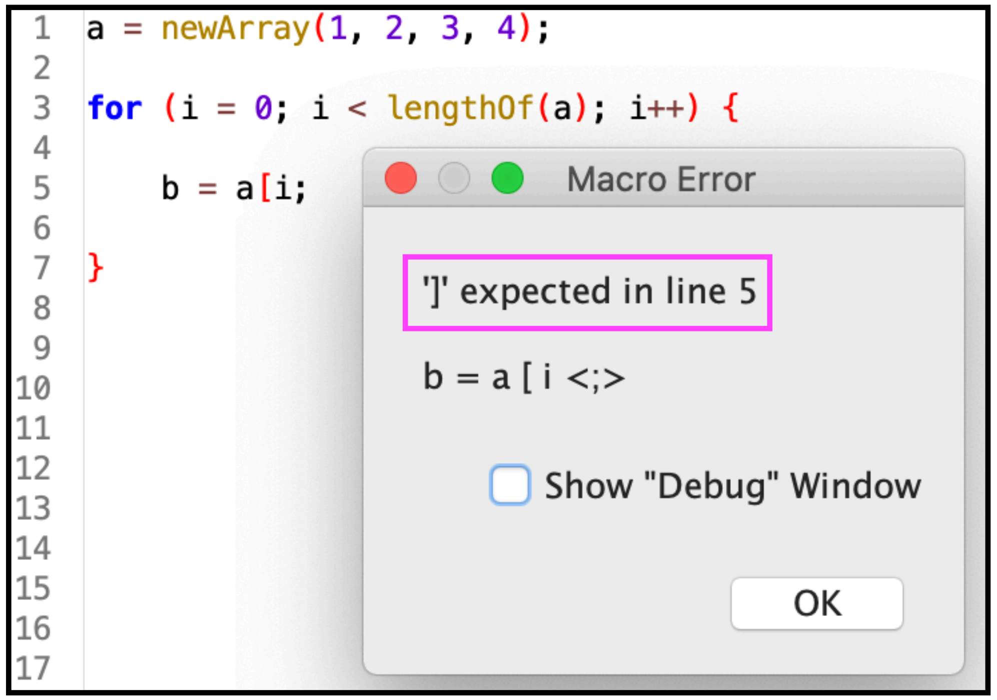
```
//define variables
a=50;
b=10;
c=15;

//compare variables
if (a>b){
if (b>c){
print(c + " > " + a + " and " + b);
}
else {
if (a>c){
print(a + " > " + c + " and " + b);
}
else {
print(c + " > " + a + " and " + b);
}
}
}
```



```
//define variables
a = 50;
b = 10;
c = 15;

//compare variables
if (a > b) {
    if (b > c) {
        print(c + " > " + a + " and " + b);
    }
    else {
        if (a > c) {
            print(a + " > " + c + " and " + b);
        }
        else {
            print(c + " > " + a + " and " + b);
        }
    }
}
```

how to troubleshoot your code

error message: try to understand **where** the error happened and **what** appears to be **wrong**.

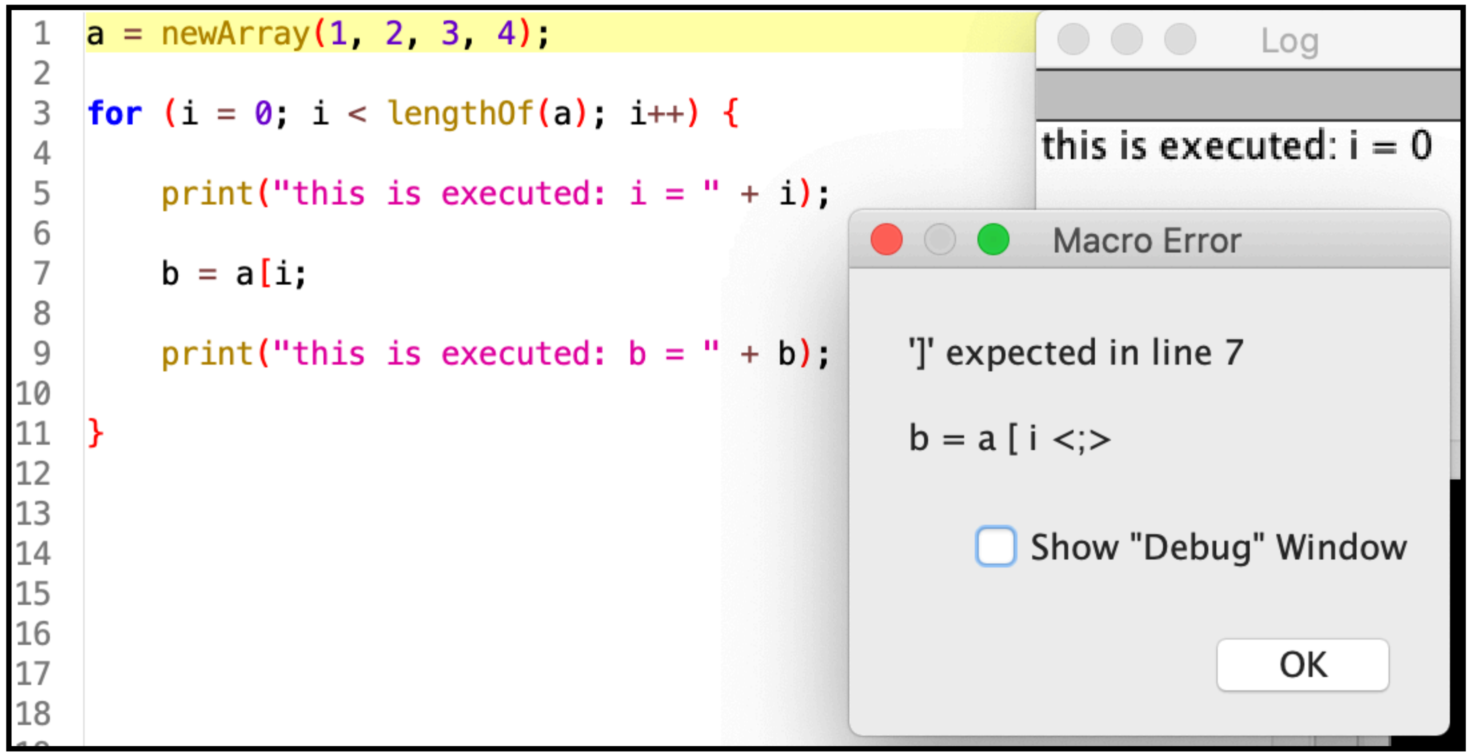


```
a = newArray(1, 2, 3, 4);

for (i = 0; i < lengthOf(a); i++) {

    b = a[i;

}
```

Macro Error

']' expected in line 5

b = a [i <;>



```
a = newArray(1, 2, 3, 4);

for (i = 0; i < lengthOf(a); i++) {

    print("this is executed: i = " + i);

    b = a[i;

    print("this is executed: b = " + b);

}
```

Log

```
this is executed: i = 0
```

Macro Error

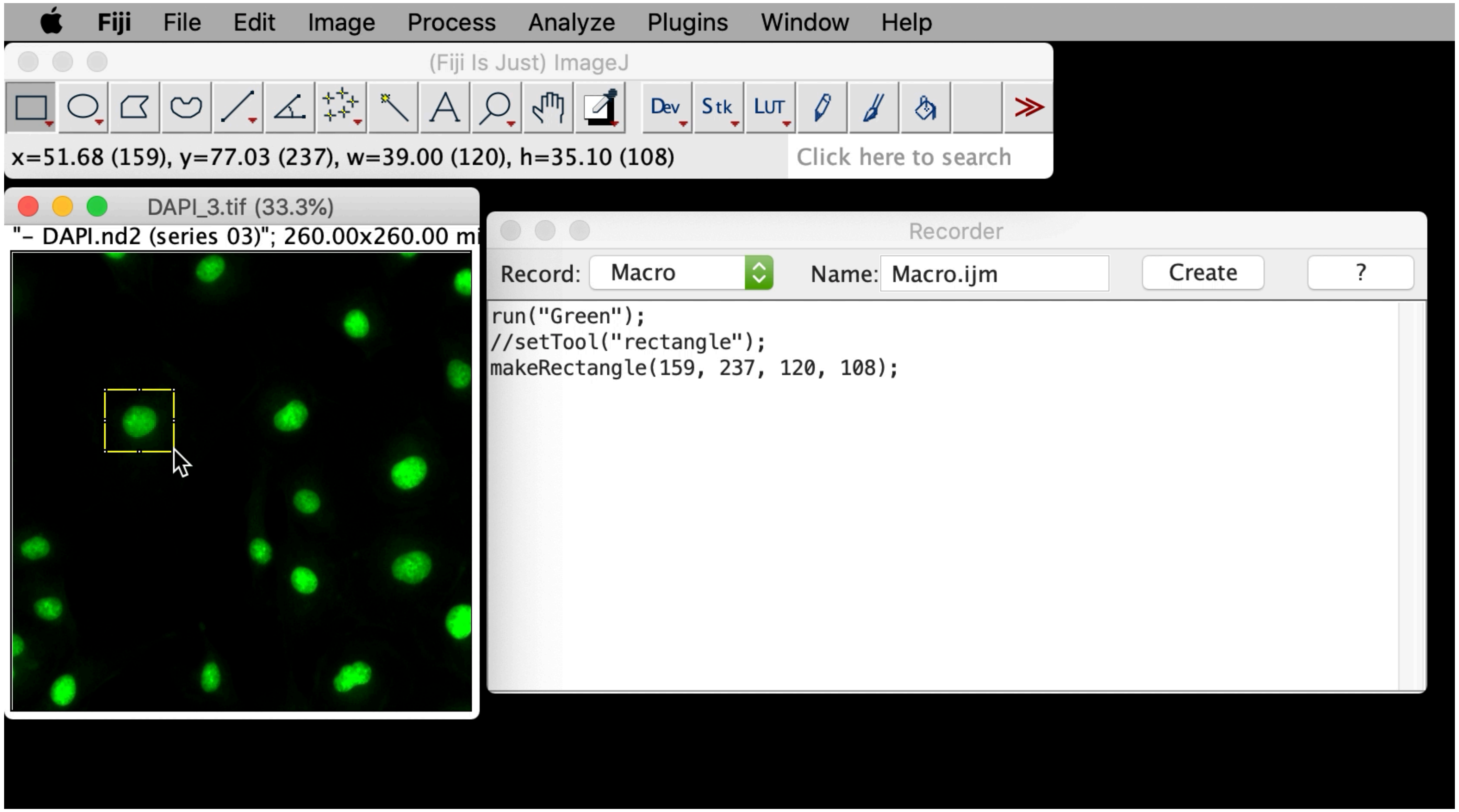
']' expected in line 7

b = a [i <;>

it can be **useful** to follow the progress of your code using the **print** function (**tracing**).

macro recorder

Plugins > Macros > Record…


Fiji File Edit Image Process Analyze Plugins Window Help
(Fiji Is Just) ImageJ
Dev Stk LUT
x=51.68 (159), y=77.03 (237), w=39.00 (120), h=35.10 (108)
Click here to search
DAPI_3.tif (33.3%)
"– DAPI.nd2 (series 03)"; 260.00x260.00 mi
Recorder
Record: Macro
Name: Macro.ijm
Create
?

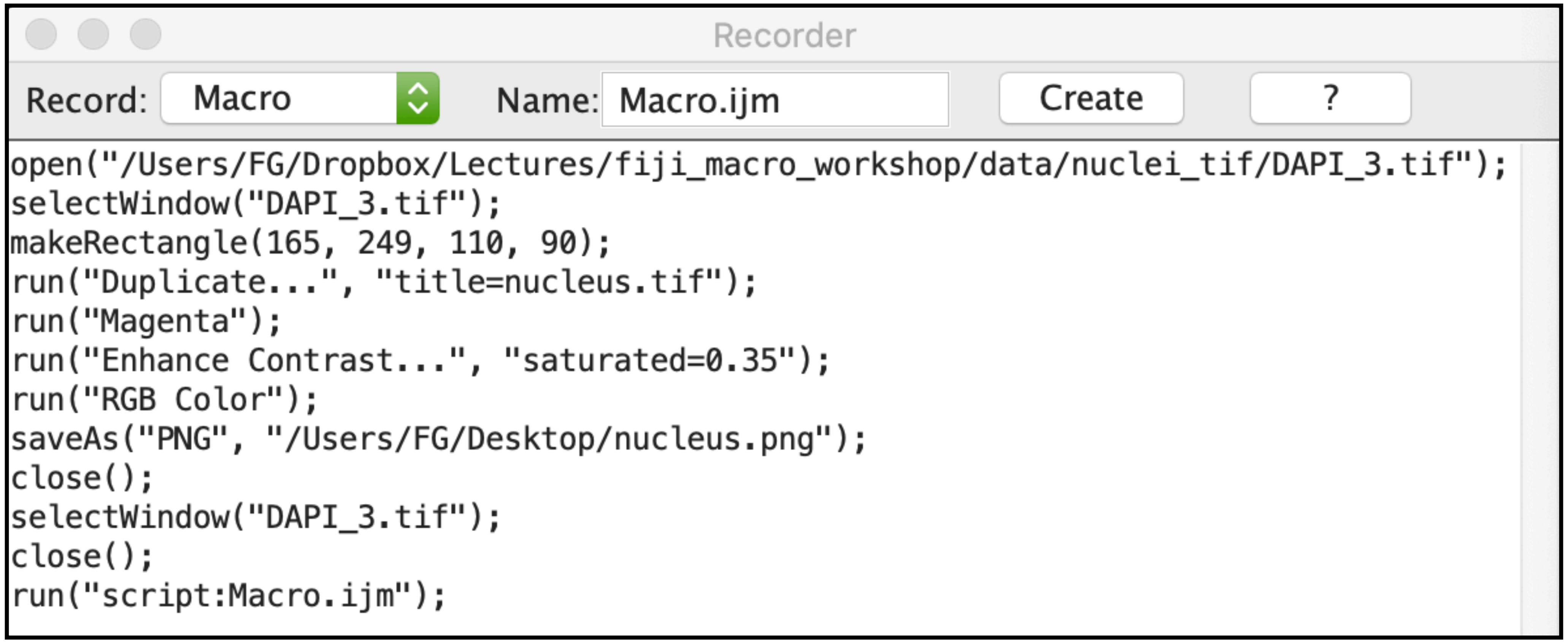
```
run("Green");
//setTool("rectangle");
makeRectangle(159, 237, 120, 108);
```

Image Analysis Collaboratory

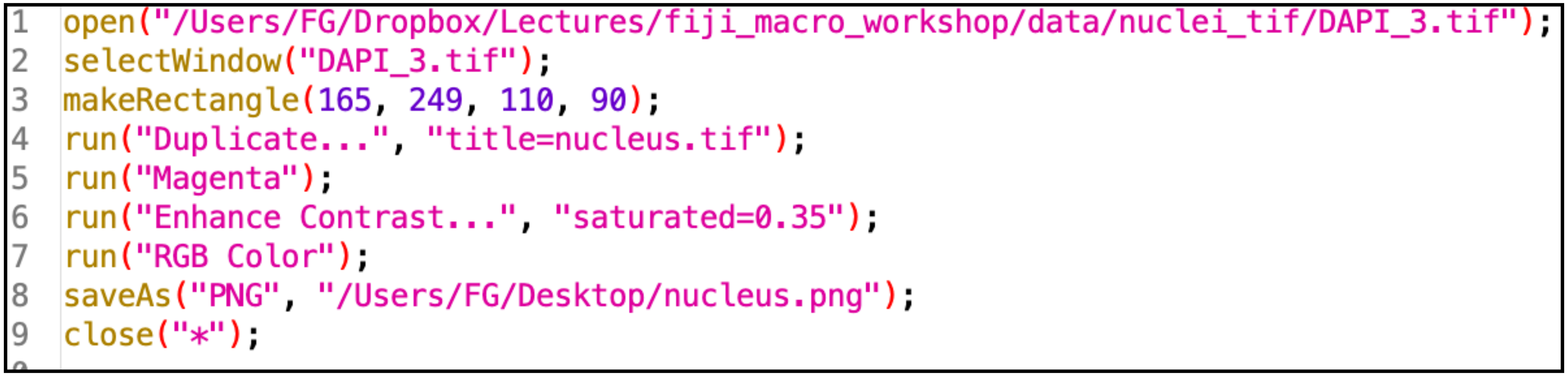
let’s try

1. **open** the **macro Recorder** (Plugins > Macros > Record…)
2. **open** an image from the *nuclei_tif* folder .
3. **draw** a **ROI** around one of the nuclei.
4. **duplicate** and **rename** the image as “nucleus.tif”.
5. **apply** a **LUT** to “nucleus.tif” (e.g. Green, Magenta…).
6. **enhance** the image **contrast** of “nucleus.tif” (Process > Enhance Contrast…).
7. **convert** “nucleus.tif” to **RGB Color**.
8. **save** “nucleus.tif” as **PNG** on the desktop.
9. **close** all the images.

let's try



```
open("/Users/FG/Dropbox/Lectures/fiji_macro_workshop/data/nuclei_tif/DAPI_3.tif");
selectWindow("DAPI_3.tif");
makeRectangle(165, 249, 110, 90);
run("Duplicate...", "title=nucleus.tif");
run("Magenta");
run("Enhance Contrast...", "saturated=0.35");
run("RGB Color");
saveAs("PNG", "/Users/FG/Desktop/nucleus.png");
close();
selectWindow("DAPI_3.tif");
close();
run("script:Macro.ijm");
```

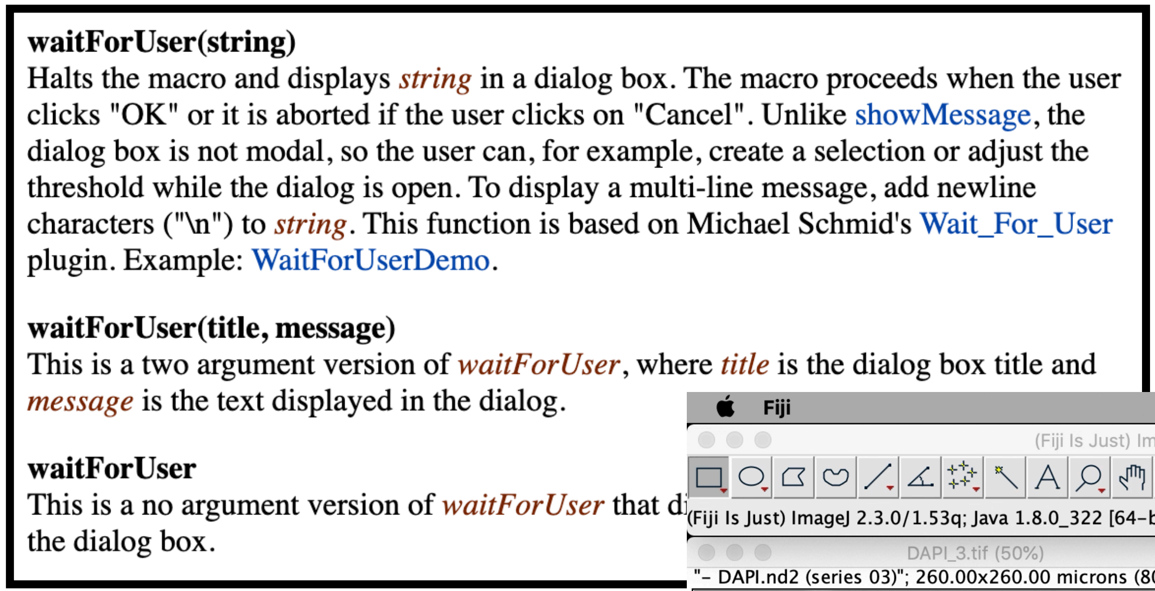


```
open("/Users/FG/Dropbox/Lectures/fiji_macro_workshop/data/nuclei_tif/DAPI_3.tif");
selectWindow("DAPI_3.tif");
makeRectangle(165, 249, 110, 90);
run("Duplicate...", "title=nucleus.tif");
run("Magenta");
run("Enhance Contrast...", "saturated=0.35");
run("RGB Color");
saveAs("PNG", "/Users/FG/Desktop/nucleus.png");
close("*");
```

macro “GUI” for user interaction

wait for user

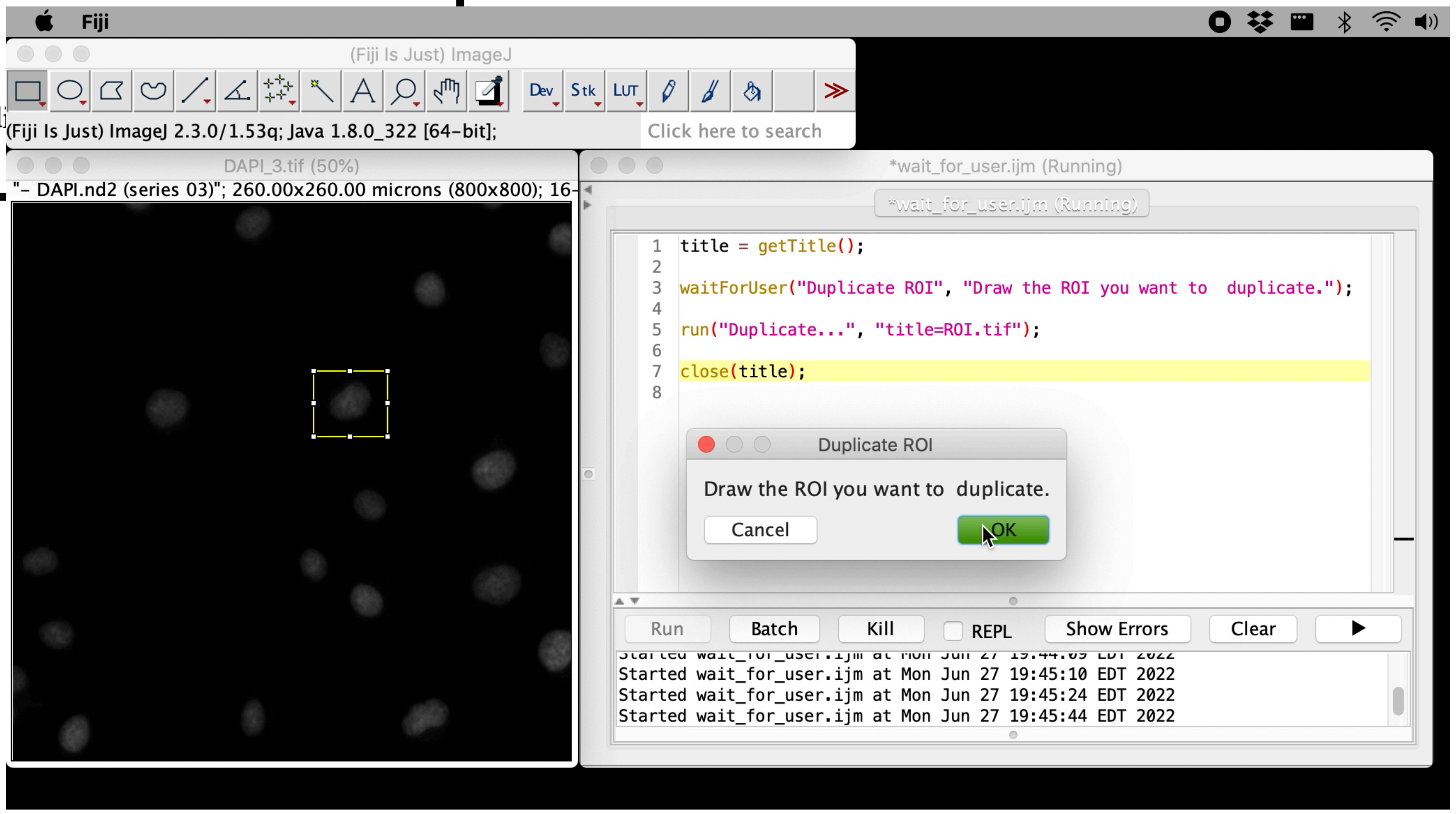
https://imagej.net/ij/developer/macro/functions.html



waitForUser(string)
Halts the macro and displays *string* in a dialog box. The macro proceeds when the user clicks "OK" or it is aborted if the user clicks on "Cancel". Unlike showMessage, the dialog box is not modal, so the user can, for example, create a selection or adjust the threshold while the dialog is open. To display a multi-line message, add newline characters ("\n") to *string*. This function is based on Michael Schmid's Wait_For_User plugin. Example: WaitForUserDemo.

waitForUser(title, message)
This is a two argument version of *waitForUser*, where *title* is the dialog box title and *message* is the text displayed in the dialog.

waitForUser
This is a no argument version of *waitForUser* that di
the dialog box.


Fiji
(Fiji Is Just) ImageJ
Dev
Stk
LUT
(Fiji Is Just) ImageJ 2.3.0/1.53q; Java 1.8.0_322 [64-bit];
Click here to search
DAPI_3.tif (50%)
"– DAPI.nd2 (series 03)"; 260.00x260.00 microns (800x800); 16-
*wait_for_user.ijm (Running)
Duplicate ROI
Draw the ROI you want to duplicate.
Cancel
OK
Run
Batch
Kill
REPL
Show Errors
Clear

```
title = getTitle();

waitForUser("Duplicate ROI", "Draw the ROI you want to  duplicate.");

run("Duplicate...", "title=ROI.tif");

close(title);

```

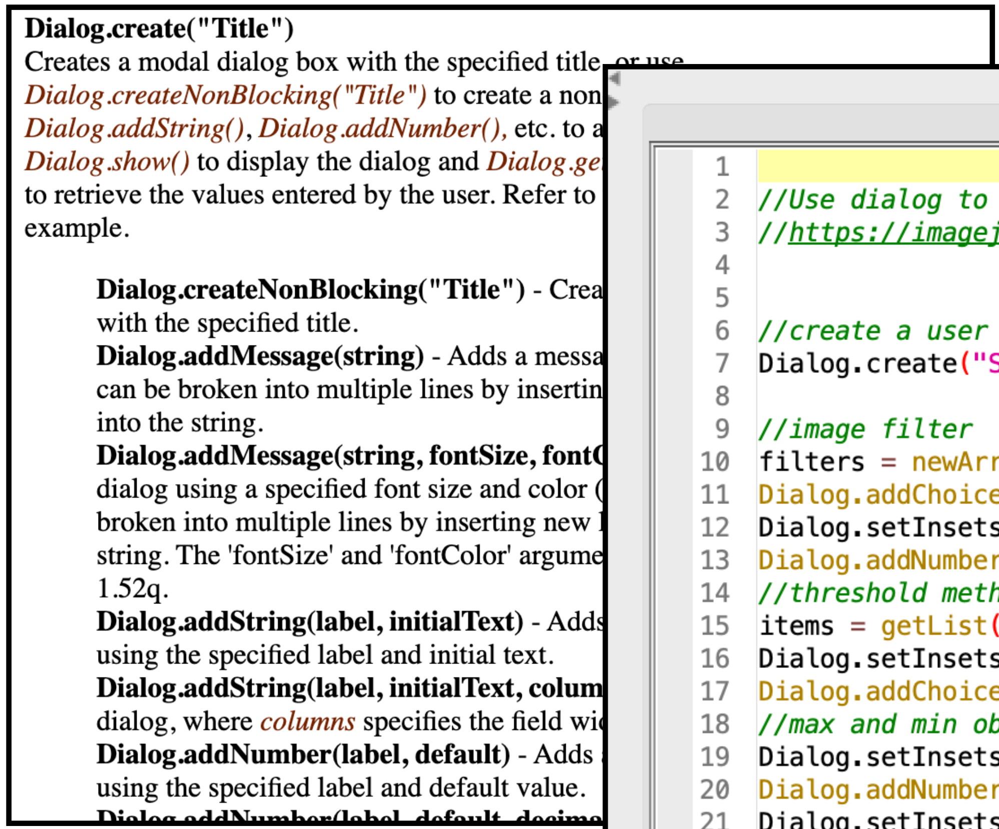
```
Started wait_for_user.ijm at Mon Jun 27 19:45:10 EDT 2022
Started wait_for_user.ijm at Mon Jun 27 19:45:24 EDT 2022
Started wait_for_user.ijm at Mon Jun 27 19:45:44 EDT 2022
```

Image Analysis Collaboratory

macro “GUI” for user interaction

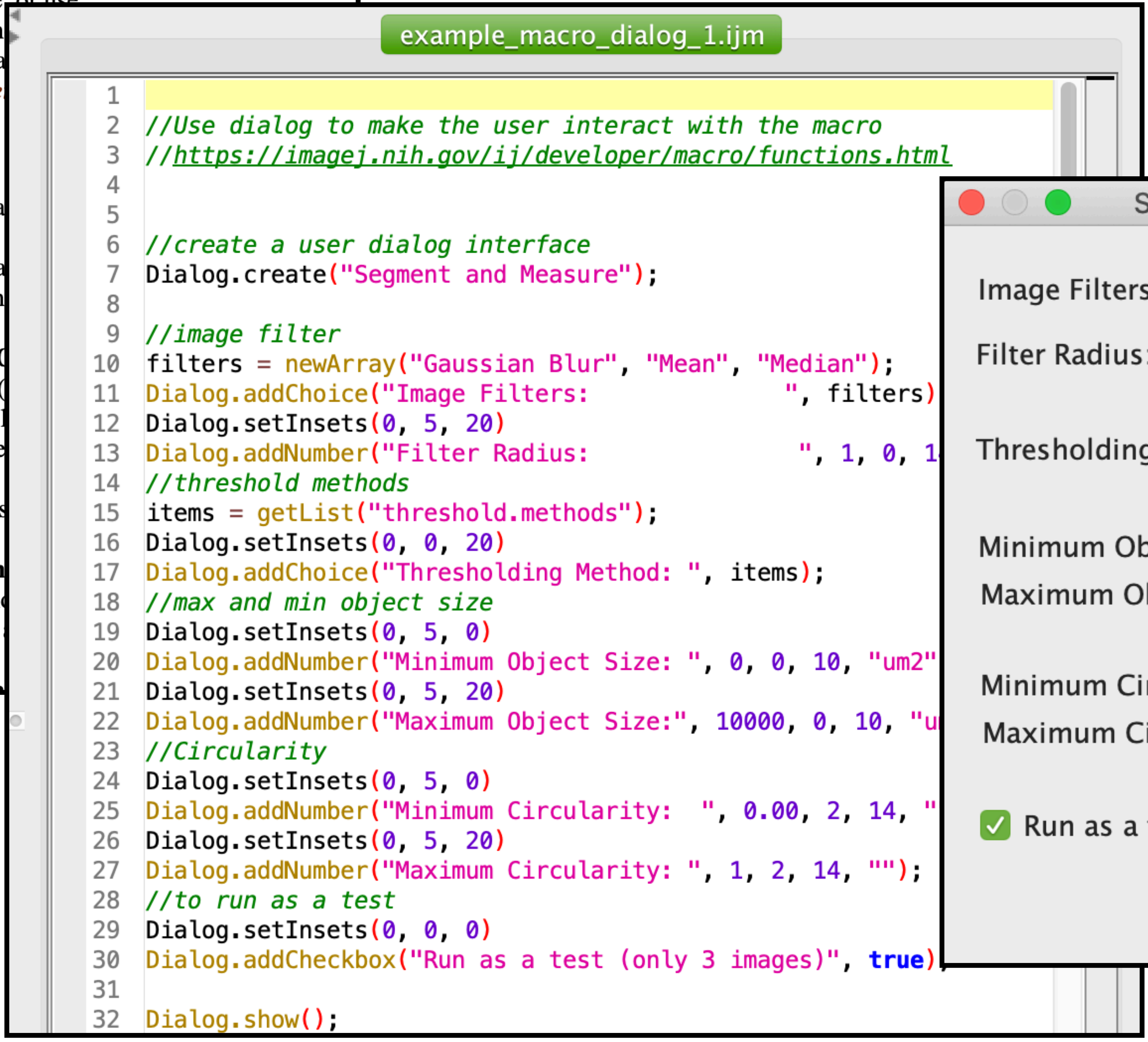
dialogs

https://imagej.net/ij/developer/macro/functions.html



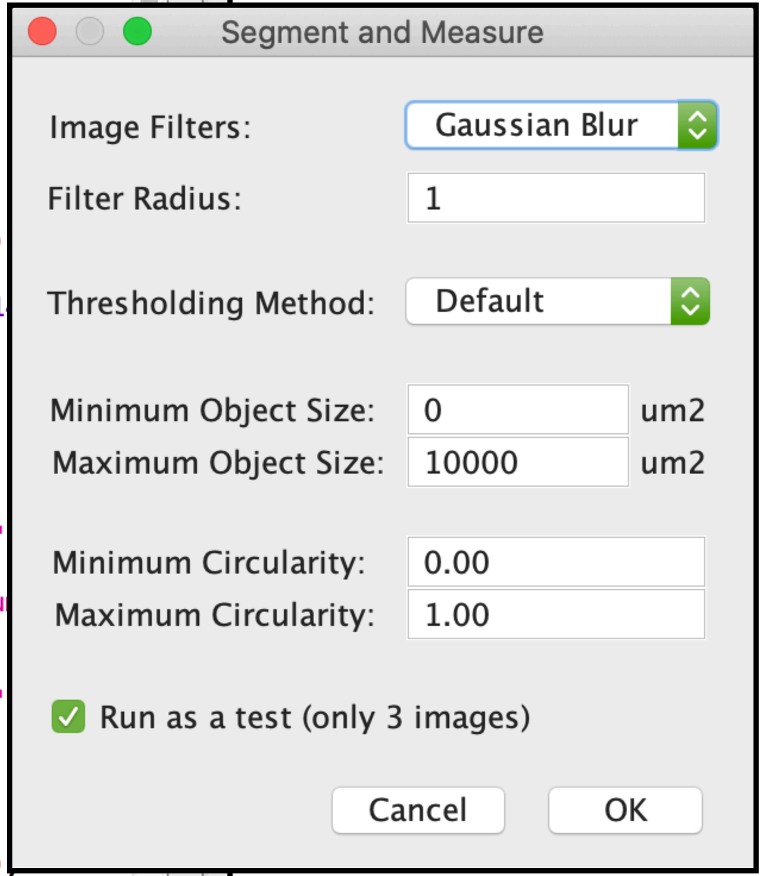
Dialog.create("Title")
Creates a modal dialog box with the specified title, or use
Dialog.createNonBlocking("Title") to create a non
Dialog.addString(), *Dialog.addNumber()*, etc. to a
Dialog.show() to display the dialog and *Dialog.ge*
to retrieve the values entered by the user. Refer to
example.

Dialog.createNonBlocking("Title") - Crea
with the specified title.
Dialog.addMessage(string) - Adds a messa
can be broken into multiple lines by insertin
into the string.
Dialog.addMessage(string, fontSize, font
dialog using a specified font size and color (
broken into multiple lines by inserting new l
string. The 'fontSize' and 'fontColor' argume
1.52q.
Dialog.addString(label, initialText) - Adds
using the specified label and initial text.
Dialog.addString(label, initialText, colum
dialog, where *columns* specifies the field wi
Dialog.addNumber(label, default) - Adds
using the specified label and default value.



example_macro_dialog_1.ijm

```

//Use dialog to make the user interact with the macro
//https://imagej.nih.gov/ij/developer/macro/functions.html


//create a user dialog interface
Dialog.create("Segment and Measure");

//image filter
filters = newArray("Gaussian Blur", "Mean", "Median");
Dialog.addChoice("Image Filters:                ", filters)
Dialog.setInsets(0, 5, 20)
Dialog.addNumber("Filter Radius:                ", 1, 0, 1
//threshold methods
items = getList("threshold.methods");
Dialog.setInsets(0, 0, 20)
Dialog.addChoice("Thresholding Method: ", items);
//max and min object size
Dialog.setInsets(0, 5, 0)
Dialog.addNumber("Minimum Object Size: ", 0, 0, 10, "um2"
Dialog.setInsets(0, 5, 20)
Dialog.addNumber("Maximum Object Size:", 10000, 0, 10, "u
//Circularity
Dialog.setInsets(0, 5, 0)
Dialog.addNumber("Minimum Circularity:  ", 0.00, 2, 14, "
Dialog.setInsets(0, 5, 20)
Dialog.addNumber("Maximum Circularity: ", 1, 2, 14, "");
//to run as a test
Dialog.setInsets(0, 0, 0)
Dialog.addCheckbox("Run as a test (only 3 images)", true)

Dialog.show();
```



Segment and Measure

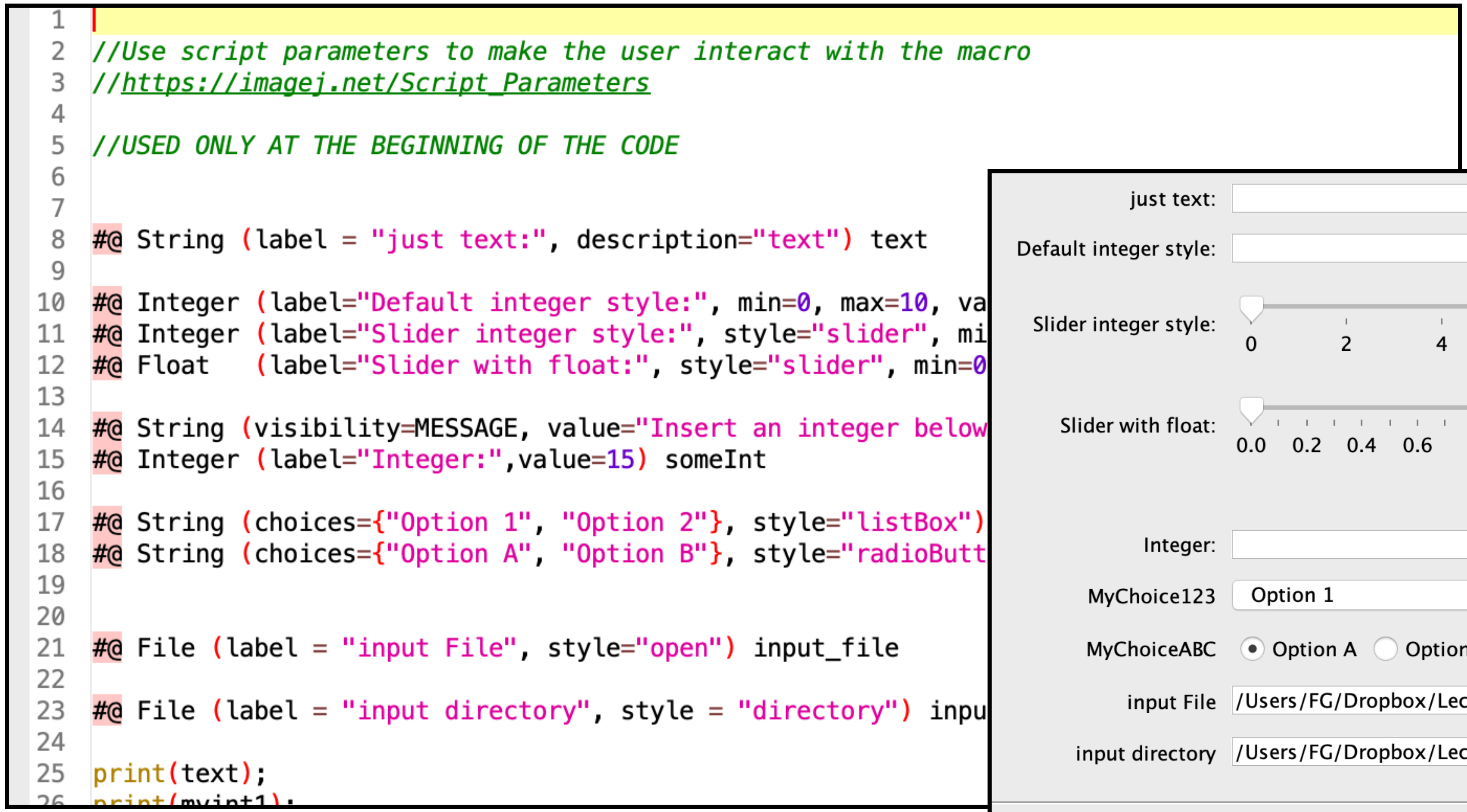
Image Filters: Gaussian Blur
Filter Radius: 1
Thresholding Method: Default
Minimum Object Size: 0 um2
Maximum Object Size: 10000 um2
Minimum Circularity: 0.00
Maximum Circularity: 1.00
✓ Run as a test (only 3 images)

***fiji_macro_workshop/macro/examples_script_parameters_and_dialog**

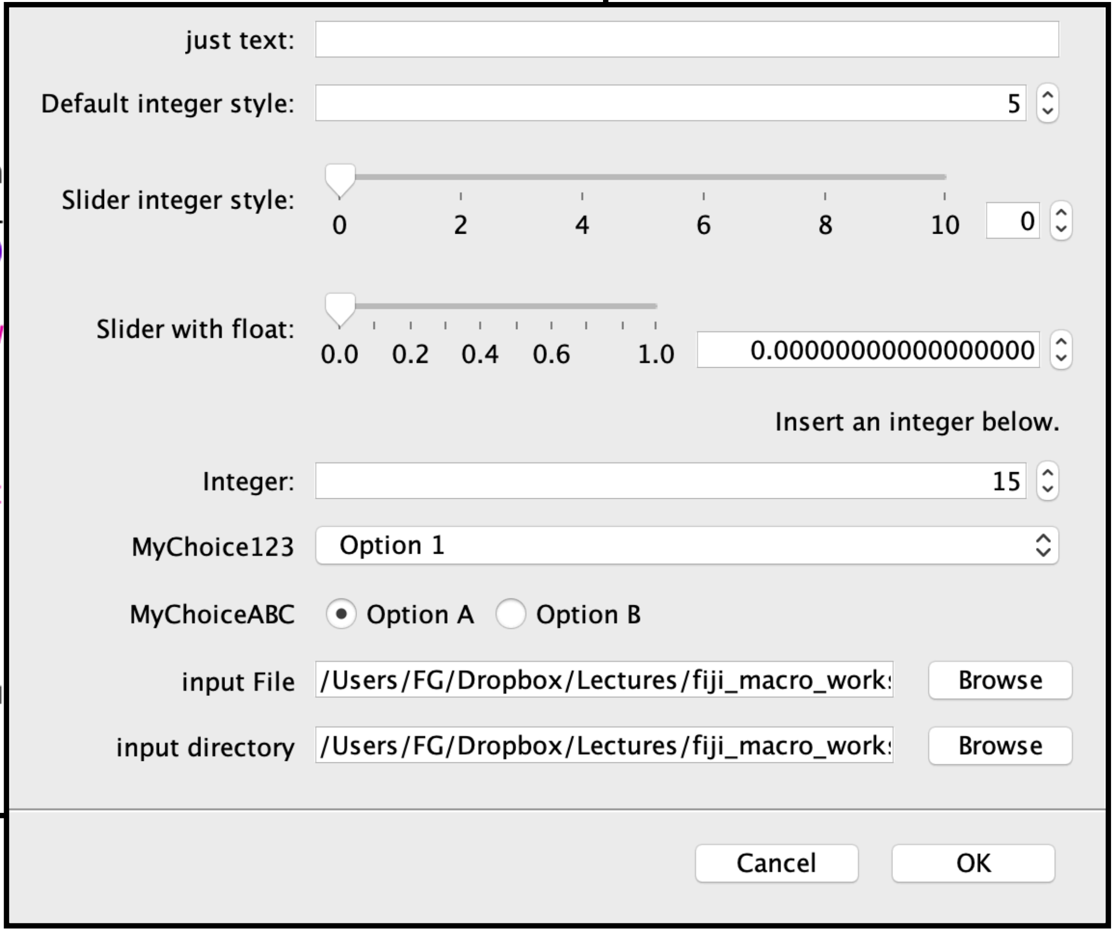
macro “GUI” for user interaction

script parameters

https://imagej.net/scripting/parameters



```
//Use script parameters to make the user interact with the macro
//https://imagej.net/Script_Parameters

//USED ONLY AT THE BEGINNING OF THE CODE


#@ String (label = "just text:", description="text") text

#@ Integer (label="Default integer style:", min=0, max=10, va
#@ Integer (label="Slider integer style:", style="slider", mi
#@ Float   (label="Slider with float:", style="slider", min=0

#@ String (visibility=MESSAGE, value="Insert an integer below
#@ Integer (label="Integer:",value=15) someInt

#@ String (choices={"Option 1", "Option 2"}, style="listBox")
#@ String (choices={"Option A", "Option B"}, style="radioButt


#@ File (label = "input File", style="open") input_file

#@ File (label = "input directory", style = "directory") inpu

print(text);
print(myint1);
```



just text:	
Default integer style:	5
Slider integer style:	0
Slider with float:	0.0000000000000000
	Insert an integer below.
Integer:	15
MyChoice123	Option 1
MyChoiceABC	[✓ Option A] Option B
input File	/Users/FG/Dropbox/Lectures/fiji_macro_work:
input directory	/Users/FG/Dropbox/Lectures/fiji_macro_work:

*fiji_macro_workshop/macro/examples_script_parameters_and_dialog