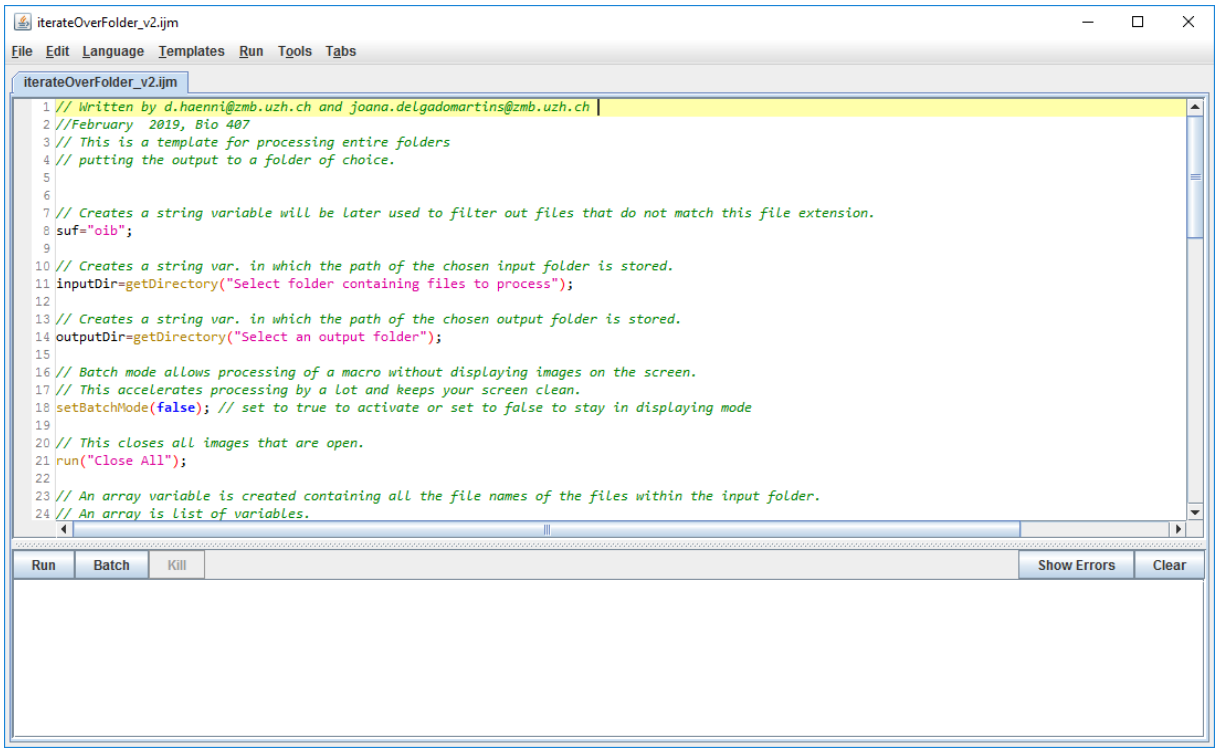
Bio 407
Automated image processing using Fiji and Macros



Center for Microscopy and Image Analysis
University of Zurich

2023

This script was written as a short summary for accompanying the practical course Bio 407 - Image processing - Session 2.

February 2023, Joana Delgado Martins, Center for Microscopy and Image Analysis, University of Zurich.

This script was revised using Fiji, ImageJ 1.53t

Contents

1. Abbreviations and Concepts

ImageJ Macro: a script that automates a series of ImageJ commands and also allows for variables, control flow, etc.

2. Remarks

This script was written for practical training purposes by the Center for Microscopy and Image Analysis, University of Zurich. Theory is kept to a minimum.

The script should be used as a guideline for hands-on training.

Highlighting styles used in this script:

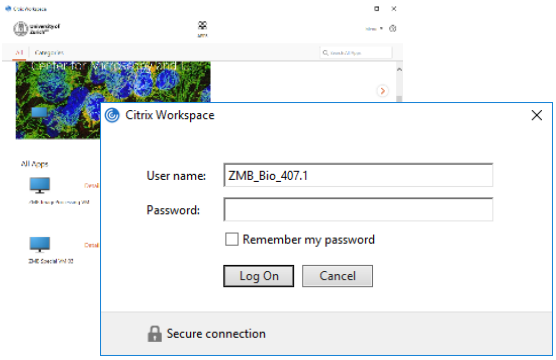
File names are set in green (e.g. Cell_Division_Pinhole_0_5AU_Nyquist_7386.lif)

Step 1.
Practical steps are set in italic. ***Keyboard shortcuts here***

```
Macro code:     run("Enhance Contrast", "saturated=0.35");
```

Or screenshots of the ImageJ macro editor.

3. Image Processing: Virtual Machines (VMs)



Usernames: [redacted] to [redacted]
Password: [redacted]
Please use the account number which corresponds to the number in the participants list (See slide "work teams")

Shortcut to common Bio 407 course share "**ZMB_Bio_407**" in "**files.core.uzh.ch (Z:)**" on microscopes and virtual machines

➔ Make a team folder where you can store and work on your data

You can find your groups credentials as well as the paths to the course network drives in the slides of the kick-off session.

4. Software: Fiji

This script gives a basic introduction to scientific image processing. All exercises can be done using ImageJ or **Fiji** which is just a version of ImageJ supplemented with many useful plugins. It can be downloaded for free from

https://fiji.sc/

Downloadable distributions are available for Windows, Mac OS X and Linux. It can read many image and microscope formats including TIFF, GIF, JPEG, BMP, DICOM, FITS, IMS, LIF, CZI, 'raw'.

If you are using the virtual machines available to the users of the Center of Microscopy and Image Analysis use the available version on the desktop for this session. Otherwise save it locally on your computer.

5. Macros

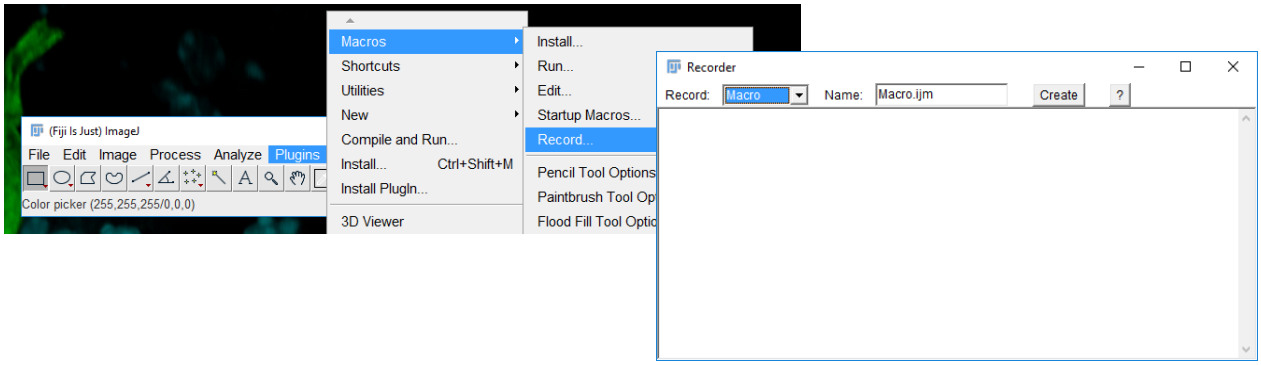
Fiji macros are very useful tools not only to **automatize repetitive tasks**, but also to **standardize** and **document** your image processing efforts.

You can easily start recording the commands used in your image processing through the macro recorder.

Step 2.
Open an image from the provided folder for this training session.

Image1_Widefield_Actin.tif.

Step 3.
Plugins>Macro>Record…



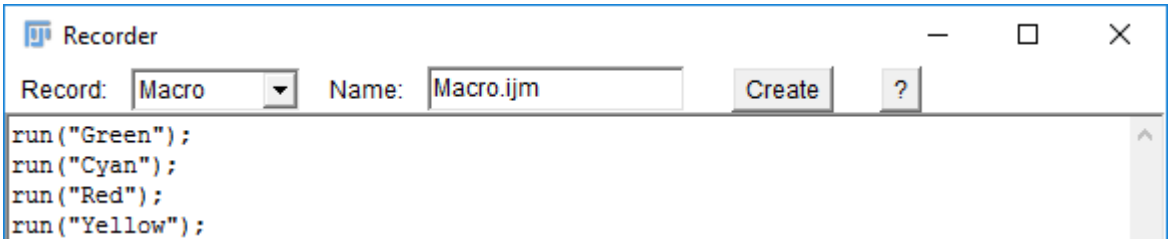
Create a copy and rename your image through:

Step 4.
Image>Duplicate ***Ctrl+Shift+ D***

Choose a LUT of your choice for the selected image through:

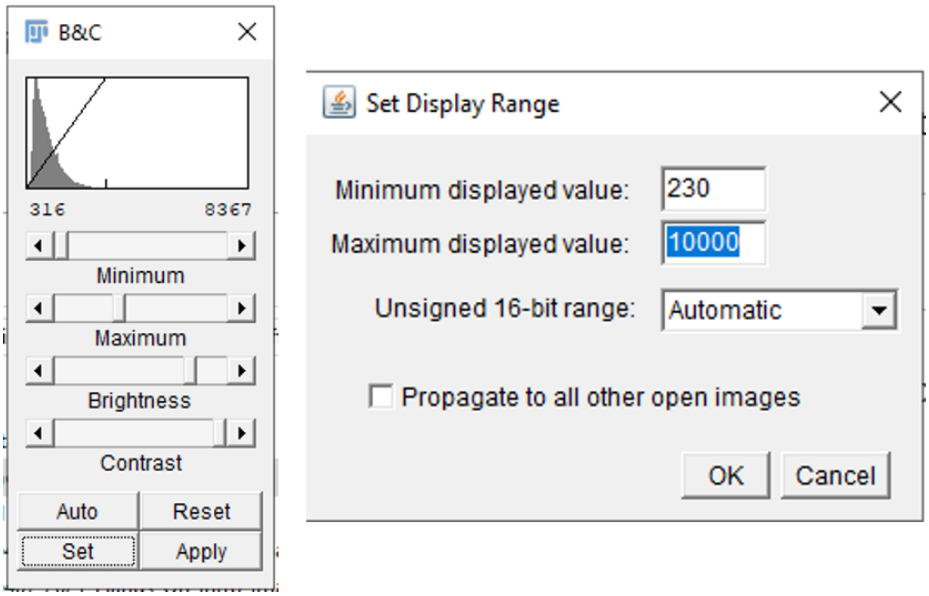
Step 5.
Image>Lookup tables

You will notice that each operation is being recorded in the macro recorder.



Step 6.

Adjust the brightness and contrast of you channels accordingly by using the "Set" button
Select Image>Adjust>Brightness/Contrast ***Ctrl+ Shift+C***




```
//run("Brightness/Contrast...");

setMinAndMax(230, 10000);
```

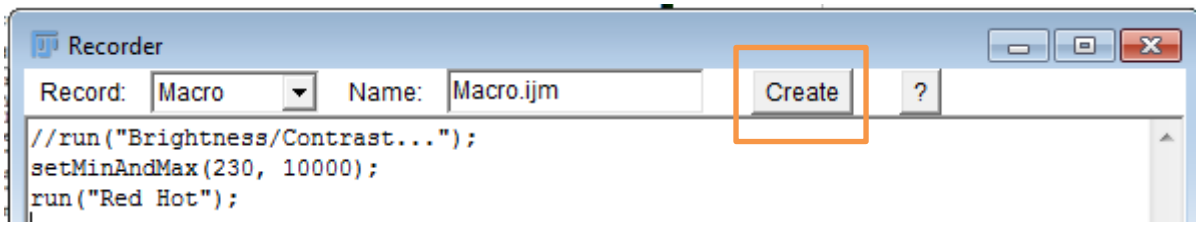
Step 7.

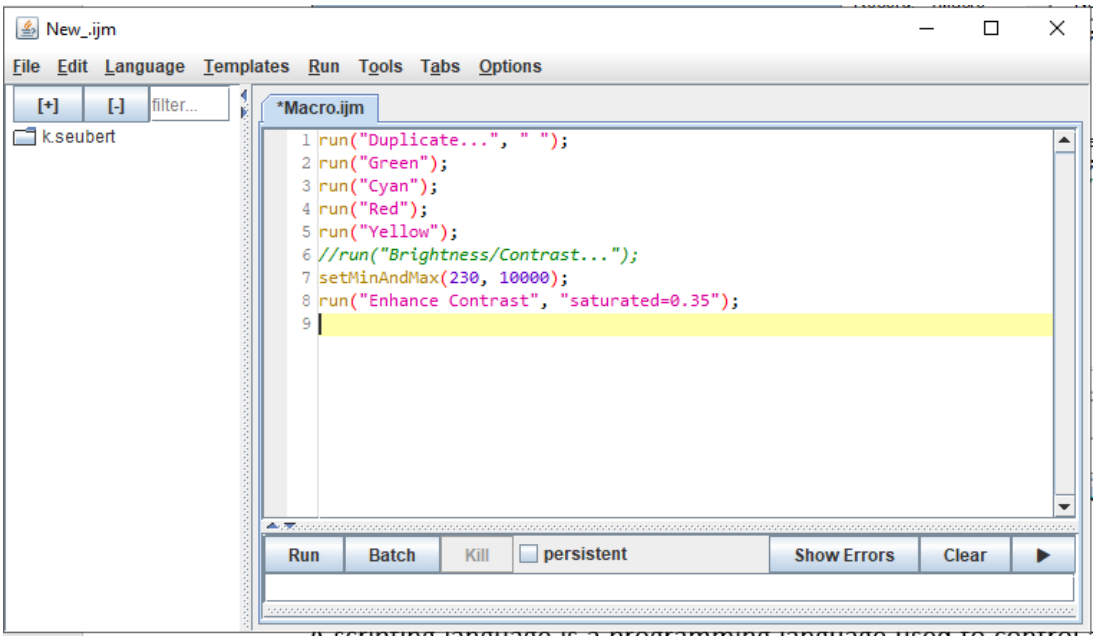
You can also automatically set these values by choosing "Auto". In this case the following command will appear

```
run("Enhance Contrast", "saturated=0.35");
```

Step 8.

You can now create your own macro by pressing "Create" on the recorder window.





6. Scripting and Supported Languages

A scripting language is a programming language used to control another software such as a runtime environment or in this case ImageJ. We will be using **ImageJ Macro language** (IJM) which is designed to be easy to read, learn and use. Programs written in the IJM, or macros, can be used to perform a sequence of actions in ImageJ.

ImageJ also supports several other scripting languages. To find more you can check here https://imagej.net/Scripting.

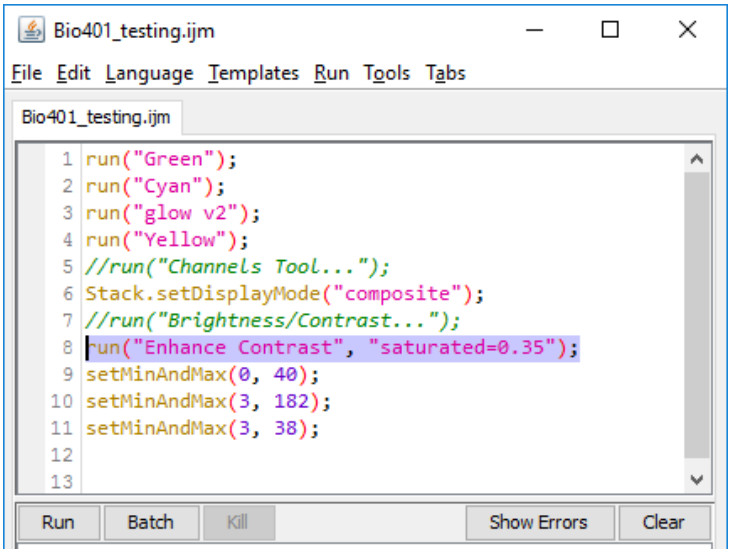
Like other programming languages, the IJM has basic structures that can be used for expressing algorithms. Those include variables, control structures (e.g. looping statements) and user-defined functions. In addition, the IJM provides access to the ImageJ functions available from the graphical user interface menu commands and to many built-in functions aimed at working with the different data structures used in ImageJ (images and image windows, regions of interest, the results table, plots, image overlays, etc.).

Remark: Not all but most ImageJ plugins are scriptable. Only if the developer of a specific plugin included the scripting functionality properly it can be seen in the macro editor and used in your code.

In your macro editor, save your macro

Step 9.
File>Save As

You will realize that your text will appear formatted in a specific way (Syntax highlighting):



Commands will automatically appear in brown,

"text in magenta between inverted commas",

values in blue and

comments in green.

All commands end with **;**

You can check other available scripting languages under the menu Language. Select IJ1 Macro Language in case your macro is shown in plain text.

At this point you probably also recorded unnecessary operations and mistakes.

Step 10.
You should now remove unnecessary operations from your code.

Step 11.
Now try to open the image
Image2_Widefield_Actin.tif
and apply the same settings using your recorded macro by pressing "Run".

7. Commenting

It is essential not only for you but also for your successors that you correctly document your macro. Also include a header where you write

- who programmed the macro (also include a contact email)
- when
- where
- for what purpose
- assumptions, known limitations, data requirements, data preprocessing, needed plugins
- other useful information for running the macro

This will be helpful not only when you need to share the code with other colleagues, but also for your own recollection.

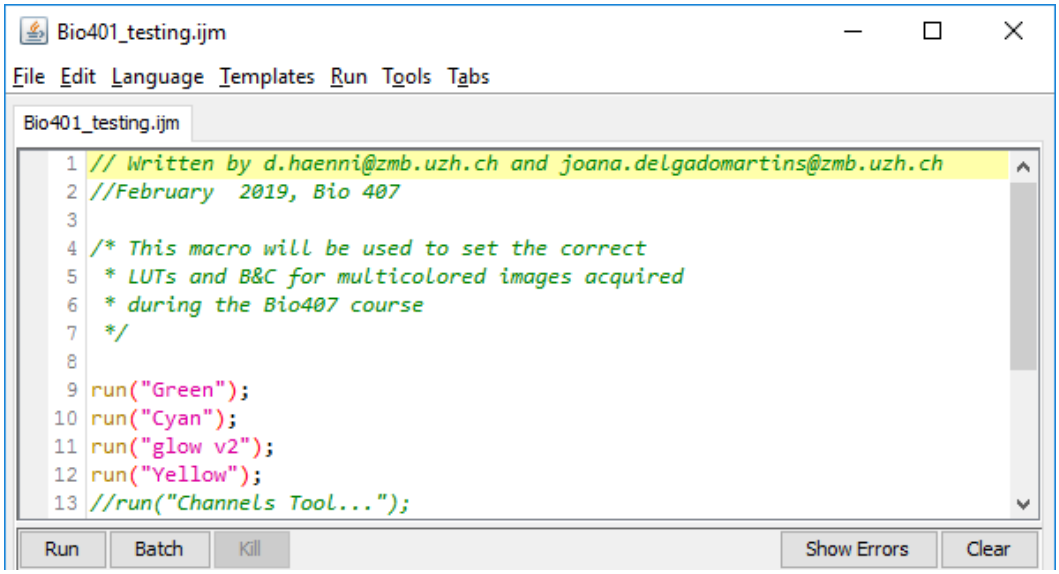
To add text that is only intended to be used for commenting your macro you can either include

// At the beginning of the line

Or

```
/*
 *
 */
```

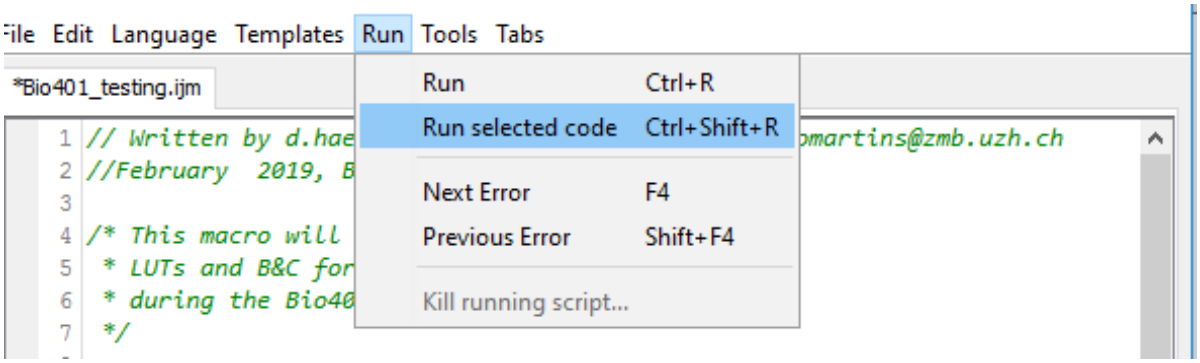
For multiple lines (once you've written /* and press Enter in your macro editor it will automatically fill the following 2 lines. You only need to write in front of the * and press Enter for a new line.



8. Run Selected Code

When debugging/optimizing your macro you might find it useful to run only selected parts of your code rather than the whole macro. You can do this in your macro editor through

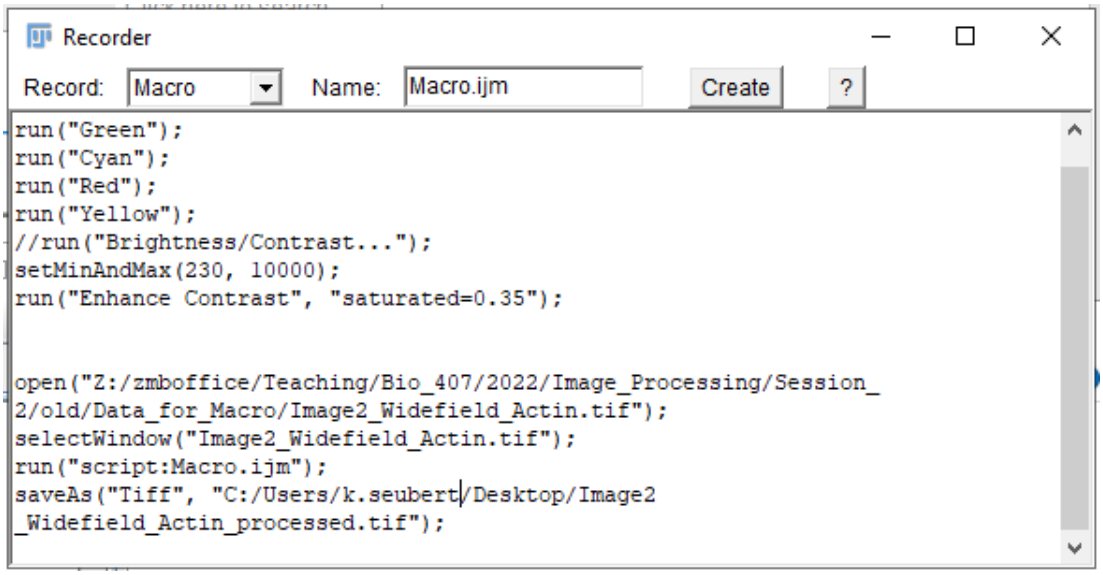
Step 12.
Run>Run selected code ***Ctrl+Shift+R***



9. Saving your Image

Step 13.
Save your image as a tif file.
File>SaveAs>Tiff

Make sure you indicate in the name that this image has been processed (not only add "-1").



You will realize that all the steps of the new file are also recorded and for saving, the **exact path** will be given.

10. Adding a Scale Bar and Saving a Second Image

Record the commands for the following steps and add them to your macro:

Step 14.
Convert into RGB
Image>Type>RGB Color
Step 15.
Add a scale bar through
Analyze>Tools>Scale Bar...
Step 16.
Save the second image with a different name.

11. Variables, adding Flexibility to your Macro

You would now love to apply this macro to a whole folder and save the corresponding images in an output folder. But how?

The command recorder writes down exactly what you do, including the names of the images you use as well as the absolute paths. To use a more general macro on more than one image, specific names must be replaced with either generic names or **variables**. You can store your filenames as variables that can be later called in the macro and used for other operations.

The ImageJ macro language is typeless. Variables do not need to be declared and do not have explicit data types. They are automatically initialized when used in an assignment statement. A variable can contain a number, a boolean, a string or an array. Numbers are stored in 64-bit double-precision floating point format. Booleans are represented by the numbers 0 and 1. Strings are sequences of characters and to be defined by "..". The most typical example of a string is a filename. The same variable can be any of these things at different times.

Strings are important variables to work with in the ImageJ macro language. They are used to handle file names and file paths, window titles and user interaction messages, but also to feed arguments to built-in commands.

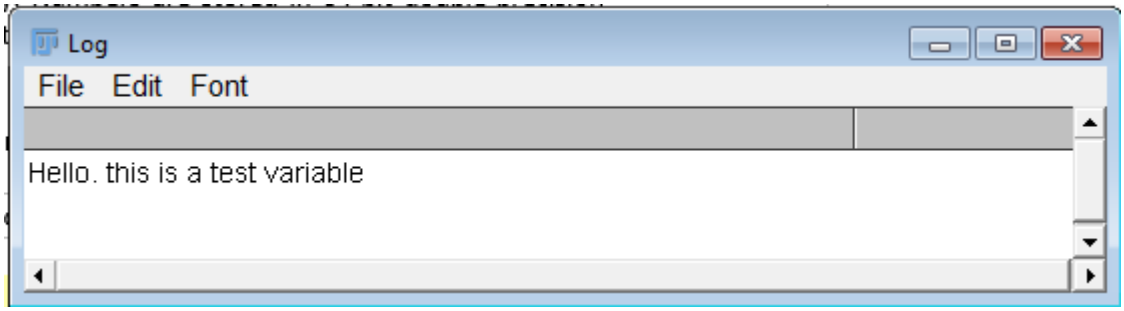
Variable names are case-sensitive. "Name" and "name" are different variables.

Step 17.
Close the macro recorder.
Step 18.
Open the macro editor via Plugins>New>Macro.
Step 19.
*Store and print a variable **testvariable** with your text of choice (a string).*

```
//Variables

testvariable="Hello. this is a test variable";
print(testvariable);
```

This will be shown in the Log window, where you can find useful information about ongoing operations (crucial for debugging).



Step 20.
Store a variable "testvariable2" with a different text. Print it.

Be careful when assigning variables as you might overwrite previously defined ones. Choose informative variable names and a coherent naming scheme that you can easily identify or debug later on.

Try marking several lines of your code, "run selected" code and see how the results change.

12. Built in Functions

Besides the commands recorded in the macro recorder ImageJ also has many useful built in functions such as

```
getTitle();
```

This function allows you to access the name of your image.

Step 21.
*Store a variable **image** which contains your image title by opening a TIFF and adding*

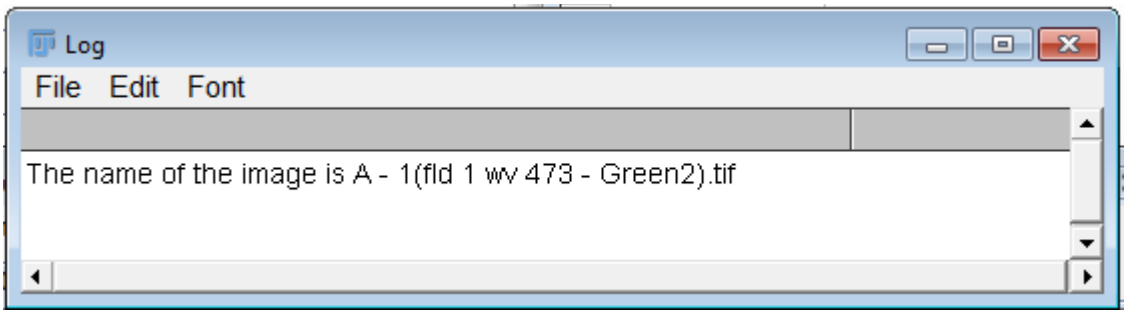
```
image=getTitle();
```

Step 22.
As the output of such instructions is not always visible you can use the command print so you

can easily keep track. This will be shown in the Log window.

```
print("The name of the image is "+image);
selectWindow(image);
```

You can also print variables and additional text by using the + sign as you can see in the example. Remember your text must be between " ". You will easily recognize this due to its different syntax highlight (magenta).



You will also find many useful built in functions in

https://imagej.nih.gov/ij/developer/macro/functions.html

13. String Manipulation and Filenames

Additionally, you can change your string such as "image" by using different built-in functions:

`indexOf(string,substring)`
Returns the index within string of the first occurrence of substring.

`substring(string,index1,index2)`
Returns a new string that is a substring of string. The substring begins at index1 and extends to the character at index2 - 1.

`replace(string,old,new)`
Returns the new string that results from replacing all occurrences of old in string with new.

`toString(number)`
Returns a decimal string representation of number.

Step 23.

*Use **replace(string, old, new)** to change string **`image`** containing your file name.*

```
newname=replace(image,"Actin","Actin2");
print("new image name is " + newname);
```

Step 24.

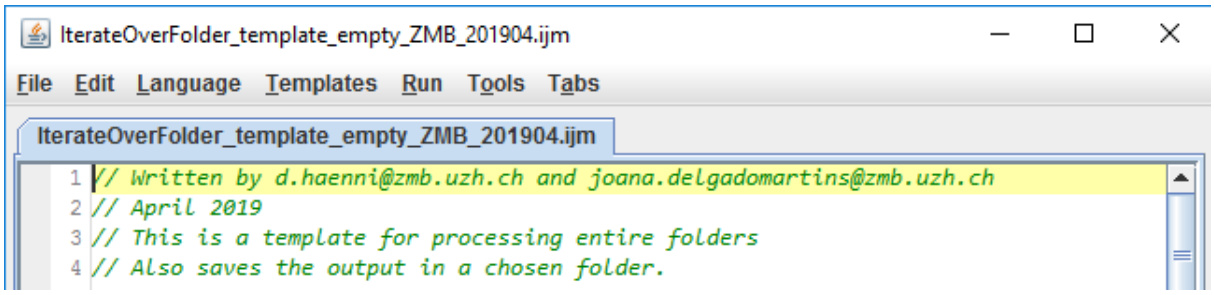
As you may have noticed, you can just add multiple strings using string + string:

```
text1="It is just easy ";
text2="to add two strings";
fulltext=text1+text2;
print(fulltext);
```

14. Iterate over a Folder

Drag & Drop the file **6_iterateOverFolder_Bio407_template_with comments.ijm**

Step 25.
Take a look at the comments and structure of the template.



Using additional built-in functions will help you get the path for your input and output folders.

```
// Creates a string var. in which the path of the chosen input folder is stored.
inputDir=getDirectory("Select folder containing files to process");

// Creates a string var. in which the path of the chosen output folder is stored.
outputDir=getDirectory("Select an output folder");
```

The variable "list" will be used to store all the file names inside the input directory:

```
list = getFileList(inputDir);
```

For loops

In order to go through all the files contained in your folder we will need to create an **iteration loop.** Looping statements are used to repeatedly execute a block of code.

Remark: Be careful that you only have the images you want to process in your folder. Otherwise you need to include additional code which instructs ImageJ regarding the types of files that will be used.

The ImageJ macro language has several loop statements for controlling the flow of the processing. In this case we will set up a **for** loop.

The *for* statement has the form:

```
for (initialization; condition; increment) {
   statement(s)
}
```

You will find this structure in the template:

```
list = getFileList(inputDir);

for (i=0; i<list.length; i++) {

        run("Bio-Formats Importer", "open=[" + inputDir + list[i] + "] color_mode=De
        title=getTitle();

        // Enter your code here ///////////

        // END of inserted code   ///////////

        saveTitle= outputDir+File.separator+"processed_"+title;

        saveAs("Tiff", saveTitle);

}
```

By adding **.length** to your "list"-variable the **for** loop knows how many times to iterate (the length of your file list). Alternatively you could manually select how many files you wish to process by placing a number instead of `list.length`.

15. Batch Mode

If you do not need to visualize each image as it is being processed you can

Step 26.
*Set Batch mode to **true**.*

```
setBatchMode(true);
```

The batch mode allows processing of a macro without displaying the images on the screen. This improves the performance of your macro dramatically and keeps your screen clean.

If you are running your macro in batch mode, print commands to the console will be particularly helpful to follow what is going on and to debug your code.

As good practice always set Batch Mode to false again

```
setBatchMode(false);
```

at the end of your macro.

16. Running the Template Macro

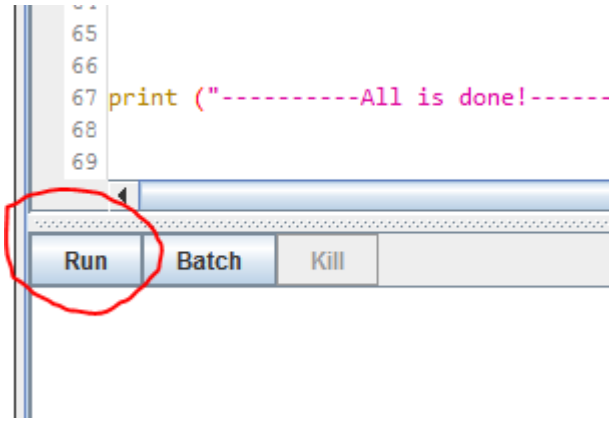
You can now try to apply your instructions (do not include the save options) to a test input folder.

Step 27.
Try applying the same B&C settings to all the images inside the folder by copying the code from your recorded macro into the section

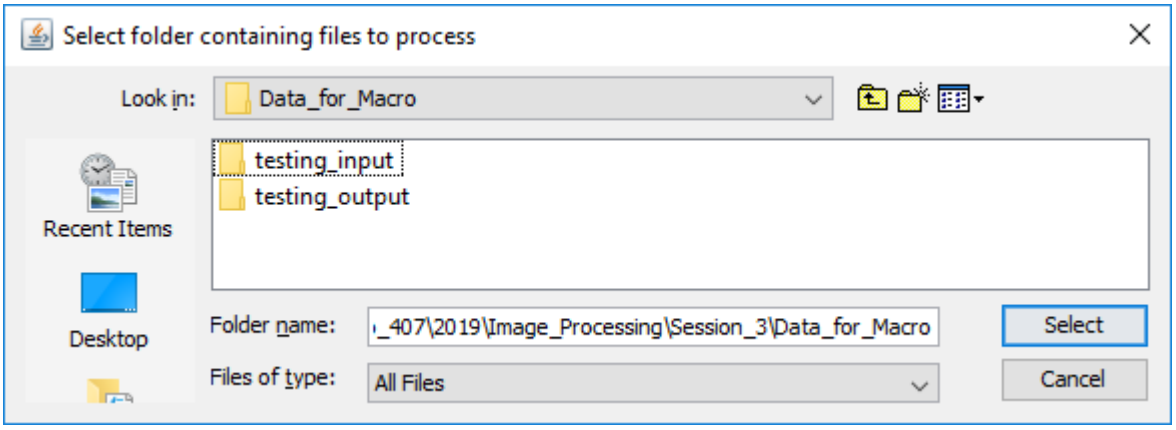
```
// Enter your code here ///////////
```

Step 28.

Run the macro.

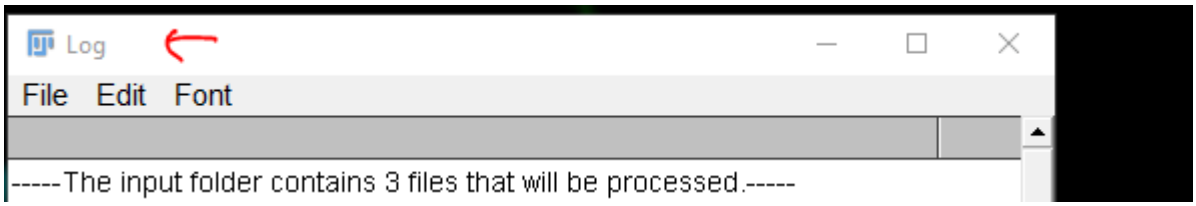


You will be asked to choose and input folder and an output folder.



After you choose your origin and destination folders the macro will store this paths and try to open the files inside.

In the Log window you will be able to see how many files your input folder contains.



Step 29.

How would you proceed if you would like to save two images?
One with and one without the scale bar?

If you need some help, look at the hints below. If this is not enough, you may also ask for the respective template.

In this example, after the B&C settings are adjusted the image is duplicated.

The name of the second image is stored under “title2”.

To save each image the command `selectWindow()` allows selection of the appropriate window.

```
run("Duplicate...", " ");
title2=getTitle();

//Here you add the scale bar
run("Scale Bar...", "width=50 height=16 font=56 color=White background=Non

        // END of inserted code   //////////


        //Saving first image

        selectWindow(title);
        saveTitle= outputDir+File.separator+"noscalebar_"+title;
        print("new file path is "+saveTitle);

        saveAs("Tiff", saveTitle);

        close();

        //Saving second image

        selectWindow(title2);
        saveTitle= outputDir+File.separator+"withscalebar_"+title;
        print("new file path is "+saveTitle);

        saveAs("Tiff", saveTitle);

        close();

}


setBatchMode(false);
print ("----------All is done!----------------");
```

17. Literature and Further Information

- Image J Macro language
    - https://imagej.nih.gov/ij/developer/macro/macros.html
- Introduction to Macro Programming
    - https://imagej.net/Introduction_into_Macro_Programming
- The ImageJ Built-in Macro Functions
    - https://imagej.nih.gov/ij/developer/macro/functions.html
- You will also find various forums and channels where you can place your questions and find solutions.
    - https://imagej.net/Help

Ways to get help

There are several popular communication channels:

Forum	The recommended way to get help. Very active. Rich modern interface. Public, archived discussion.
Mailing Lists	An established and trusted resource for help. Thousands of subscribers. Public, archived discussion.
Chat	ImageJ developers and experts frequent real-time chat services including Gitter and IRC.
Stack Overflow	Some people like to post about ImageJ on Stack Overflow, a programming Q&A site.
Reddit	Some people like to post about ImageJ on Reddit, a community bulletin board.