上課注意事項:

1. 教室內禁止攜帶食物飲料入內, 僅允許“白開水”, 請大家將食物飲料放置於教室外的桌上。
2. 請實體與線上學員掃描以下QR code進行線上簽到。
3. 請線上學員於課程開始前關閉自己的麥克風。
4. 線上學員若有問題, 請先按下“舉手”, 或於聊天室寫下問題, 將於課程結束後在場地時間允許下, 安排QA時間。
5. 現場學員發問時請使用麥克風才可進行收音。

線上簽到

課程材料與資訊連結

課後意見調查

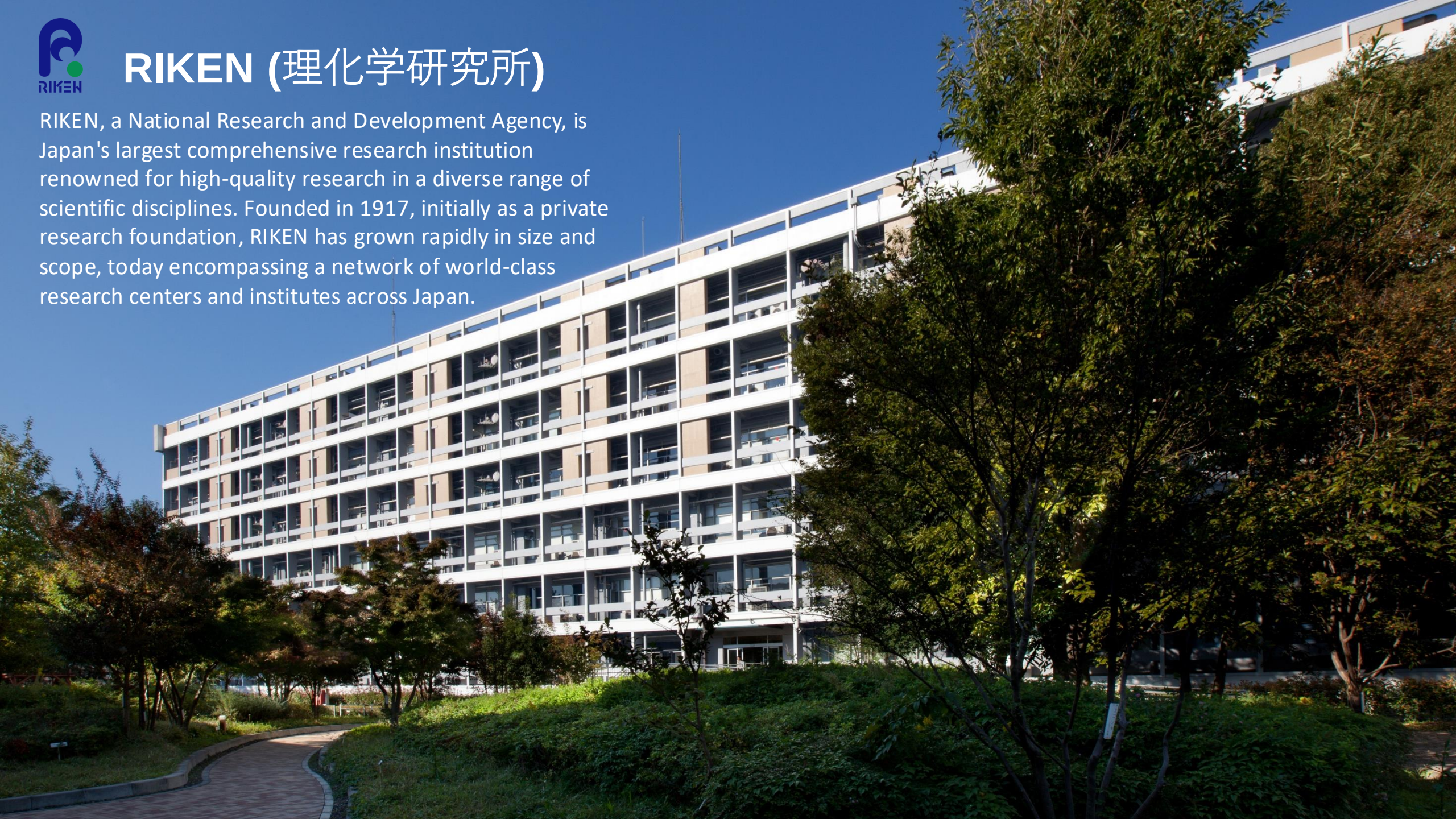
ImageJ 顯微影像分析與程式設計

影像分析自動化 – ImageJ Macro

Jen-Chien Chang (張仁乾)
Research DX Foundation Team, RIKEN
2025.03.17

RIKEN (理化学研究所)

RIKEN, a National Research and Development Agency, is Japan's largest comprehensive research institution renowned for high-quality research in a diverse range of scientific disciplines. Founded in 1917, initially as a private research foundation, RIKEN has grown rapidly in size and scope, today encompassing a network of world-class research centers and institutes across Japan.



Jen-Chien Chang




2008 Taiwan

2014 USA

2023 Riken Yokohama

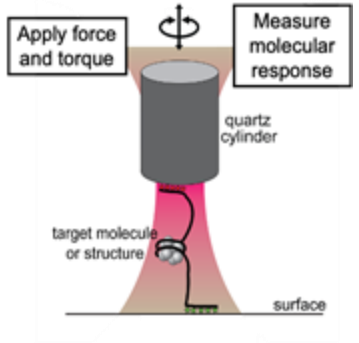
Riken Yokohama

Physics
B92202…

- Quantum optics
- Protein folding simulation

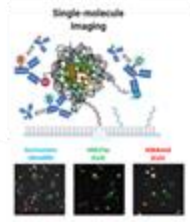
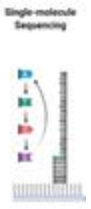
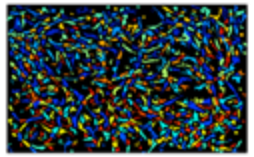
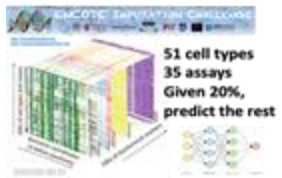
Biophysics

- Single-molecule biophysics
- Optical tweezers



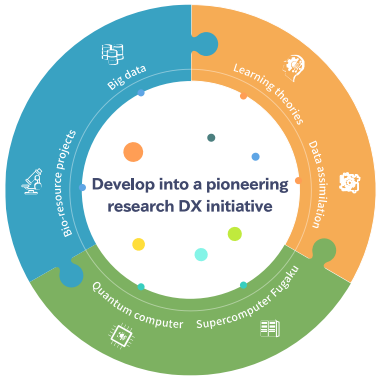
Biology & Medical science

- Epigenomics
- Sequencing technology
- Bioinformatics – genomics
- Bioinformatics – imaging

Research Digital Transformation (DX)

- Technical support for data management & analysis
  - Imaging & Sequencing data
  - Core facilities & individual labs



RIKEN TRIP

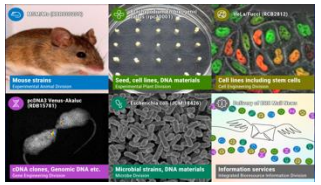
https://researchmap.jp/Jen-Chien_Chang?lang=en

Main RIKEN campus & core facilities

Harima 播磨
(SPring-8 synchrotron radiation facility)
- X-ray CT image

Wako 和光
- Proteomics core lab
- Brain images

Tsukuba 筑波
(BioResource Center)
- Cell bank image
- Mouse CT image



Kobe 神戸
- Proteomics core lab
- Bioimaging facility

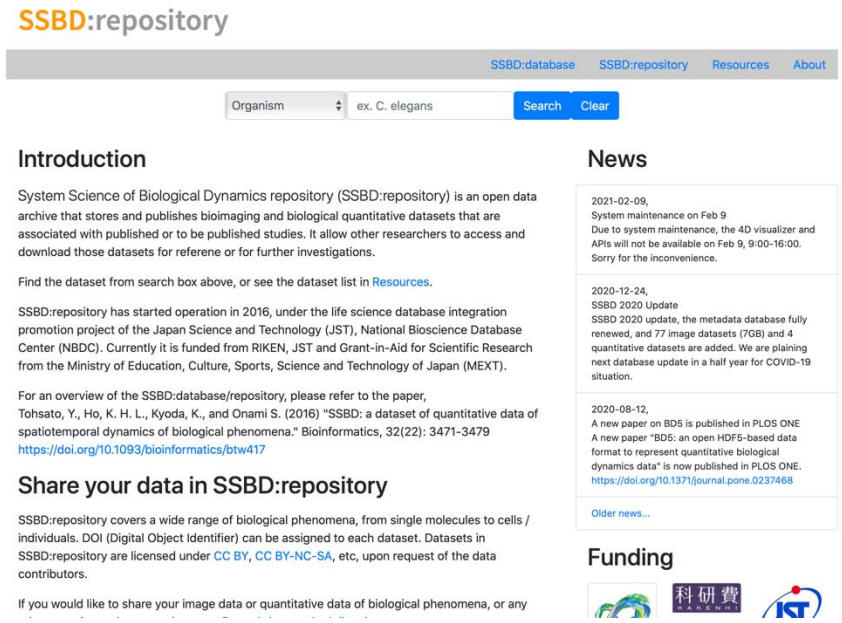
Yokohama 横浜
- NMR facility
- Genome platform

SSBD

A platform for Systems Science of Biological Dynamics

SSBD:repository

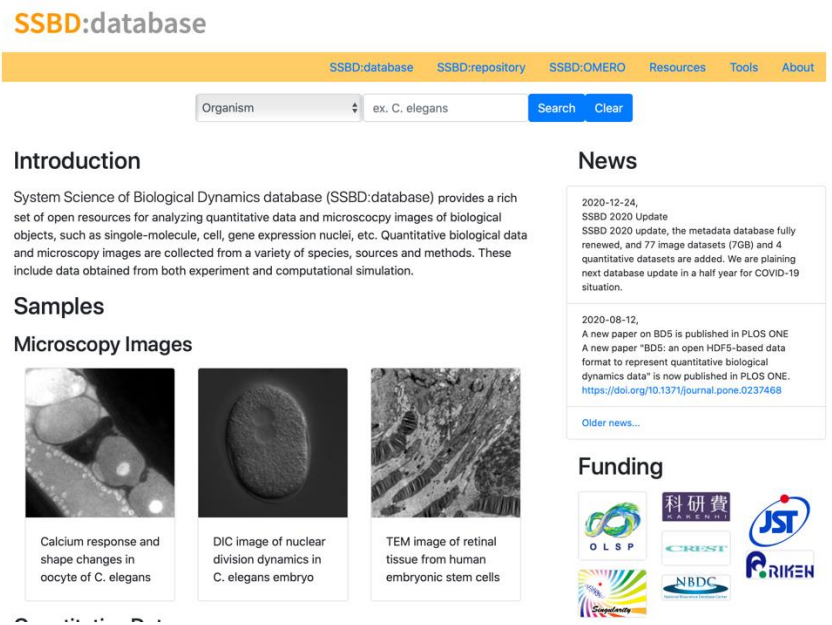
Public repository for sharing bioimaging data and quantitative biosystems dynamic data used in papers



Team Leader:
ONAMI Shuichi
大浪修一

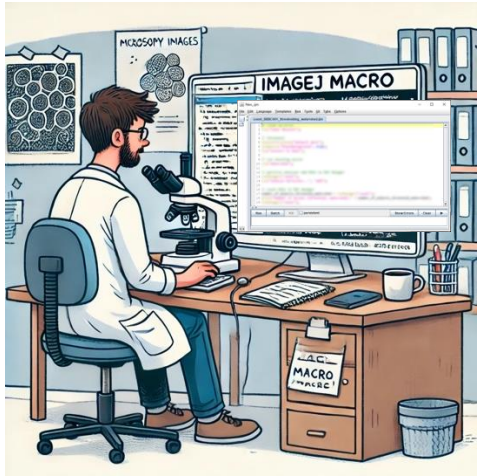
SSBD:database

Added-value database that shares highly reusable bioimaging data and quantitative biosystems dynamics data with rich metadata.



Automate the image analysis workflow with **ImageJ Macro**

- Goal: Automate your analysis workflow using **ImageJ Macro**
  - **For those with programming experience:** Become familiar with ImageJ Macro syntax
  - **If you have no programming experience:** Learn to read and adapt templates for your workflow

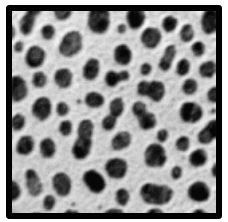


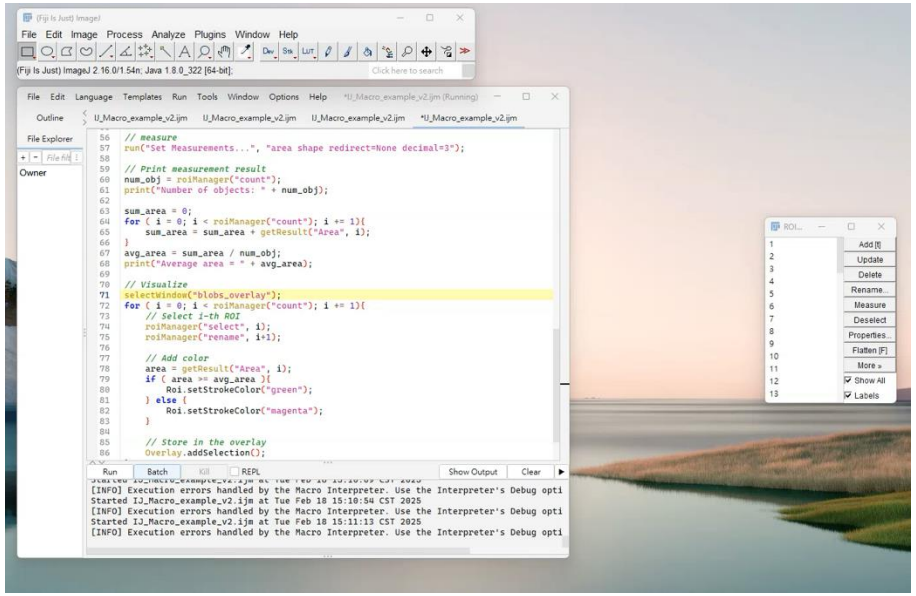
- Outline
  - Part1: Introduction to reproducible analysis with ImageJ Macro

Common concept
- Variables, array, and operations
- Conditions and loops
- Functions
- Writing good code

IJ-specific
- Recording Macros and working with images
- ROIs and Overlays
- Plots and Tables
- Batch processing, File I/O

  - Part2: Hands-on session
    - Ex1: Automate the flow for object counting
    - Ex2: File I/O & batch processing



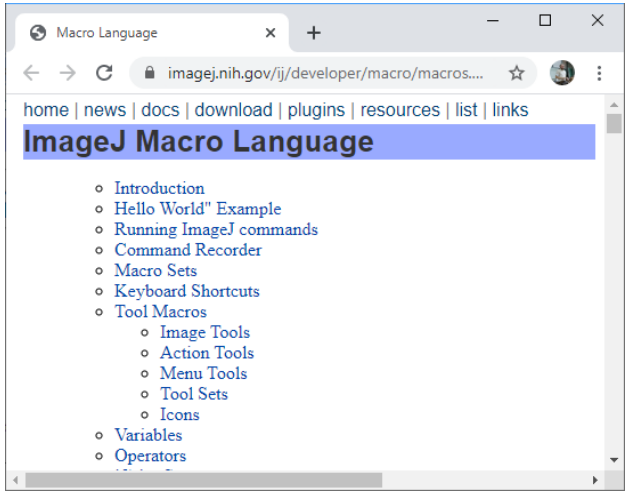


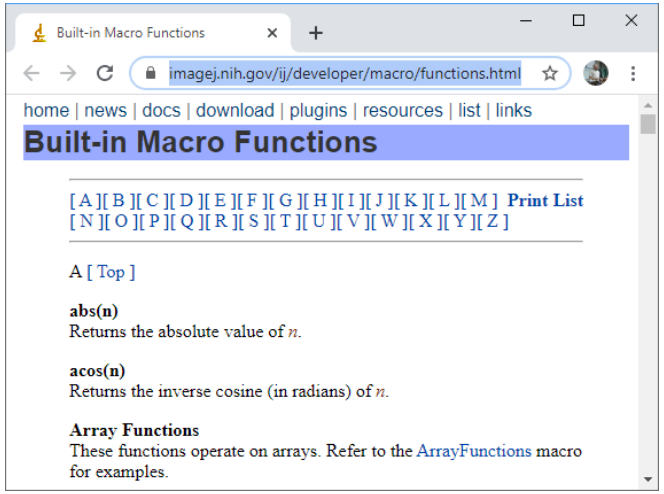
Reference

- This document is adapated mainly from the materials maintained by **Robert Haase** under CC-BY 4.0 license
  - Slide for the virtual course at CCI Gothernburg, October 2021 https://f1000research.com/slides/10-1000
  - Example codes: https://github.com/BiAPoL/CCI_Gothenburg_ImageJ_Macro_Course_2021
  - Youtube: https://www.youtube.com/@haesleinhuepf/videos

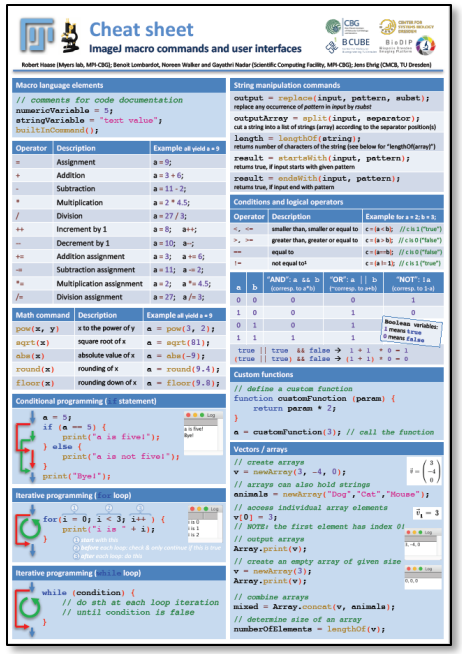
Resources

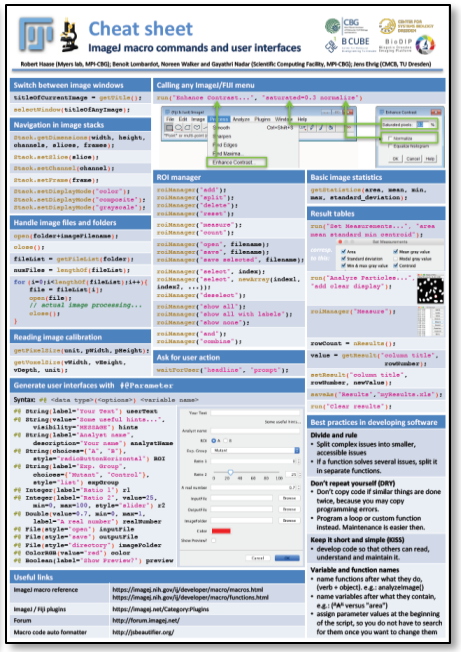
- **ImageJ Macro official websites**
  - https://imagej.net/ij/developer/macro/macros.html
  - https://imagej.net/ij/developer/macro/functions.html



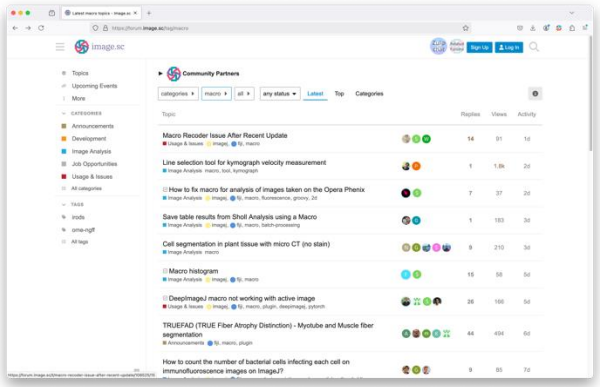


- **Cheat sheet**
  - https://github.com/BiAPoL/imagej-macro-cheat-sheet/blob/master/ImageJ_macro_cheatsheet.pdf





- **image.sc**
  - https://forum.image.sc/



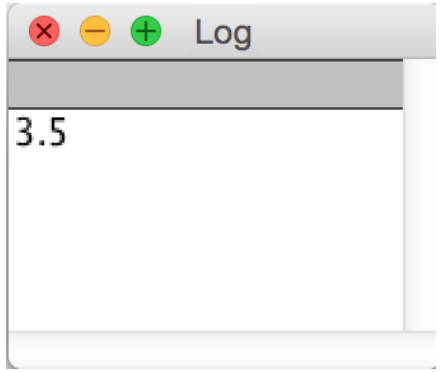
Part1: Introduction to reproducible analysis with **ImageJ Macro**

Variables

- Variables hold **values** or **text** and can be connected using **operators**

```
// initialise program
a = 1;
b = 2.5;

// run complicated algorithm
final_result = a + b;

// print the result
print( final_result );
```

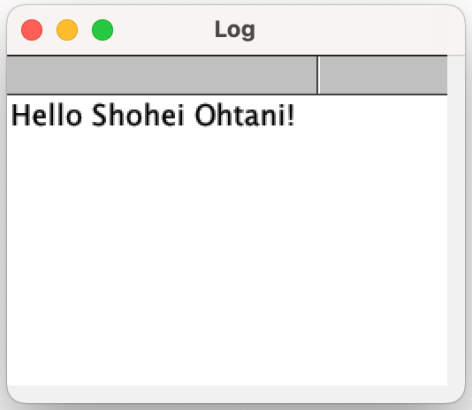
variables.ijm



- Texts are called “strings” in the world of programming
  - They represent a chain of characters
  - Strings can be concatenated with “+”

```
// initialise program
firstname = ”Shohei";
lastname = ”Ohtani";

// run complicated algorithm
name = firstname + " " + lastname;

print("Hello " + name + "!");
```

string_variables.ijm

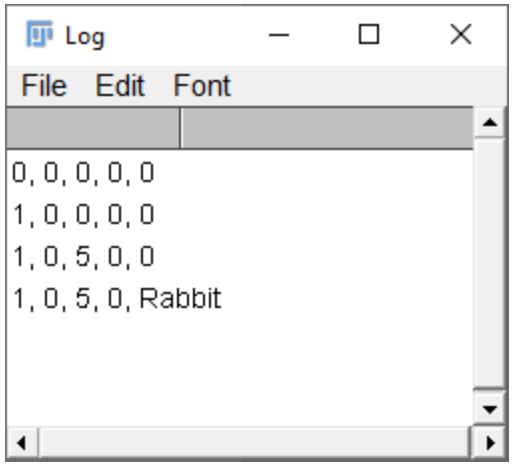


Array

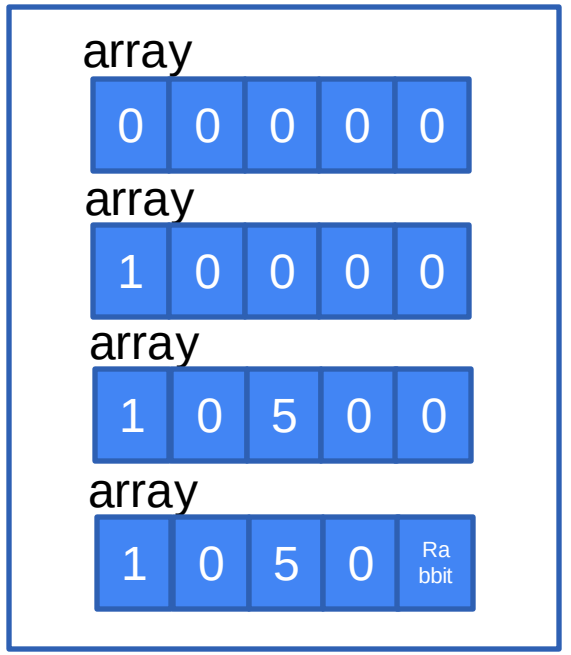
- Arrays are variables, where you can store multiple stuff

```
1
2  array = newArray(5);
3  Array.print(array);
4
5  array[0] = 1;
6  Array.print(array);
7
8  array[2] = 5;
9  Array.print(array);
10
11 array[4] = "Rabbit";
12 Array.print(array);
13
```

arrays_access.ijm



Computer memory



- Arrays can be initialized with values (default: 0)

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

animals = newArray("dog", "cat", "mouse");
```

- Arrays can be concatenated:

```
mixed = Array.concat(v, animals);
```

// Comments

- Comments should contain additional information such as
  - User documentation
    - What does the program do?
    - How can this program be used?
  - Your name / institute in case a reader has a question
  - Comment why things are done.
  - Do not comment what is written in the code already!

```
//
// This program sums up two numbers.
//
// Usage:
// * Run it in FIJI (www.fiji.sc)
//
// Author: Robert Haase, MPI CBG,
//         rhaase@mpi-cbg.de
// July 2016

// initialise program
a = 1;
b = 2.5;

// run complicated algorithm
final_result = a + b;

// print the final result
print( final_result );
```

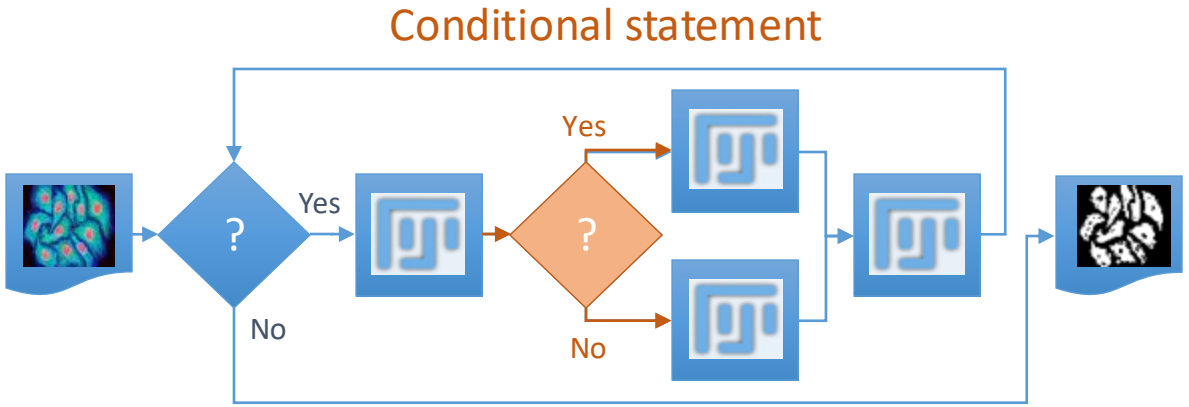
good_comments.ijm

Mathematical operations

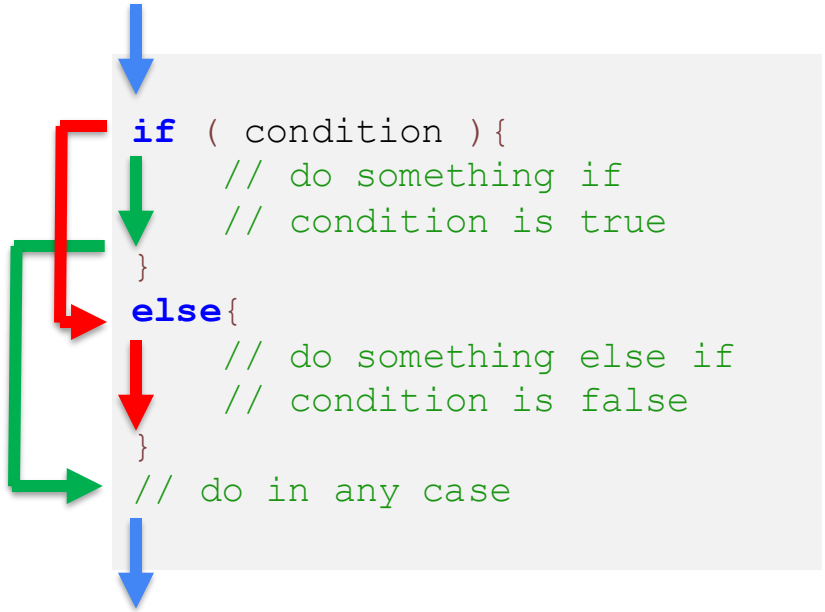
- Math commands supplement operators to be able to implement any form of calculations

Command	Description	Example
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);
sin(x)	sinus of x given in radians	b = sin(PI);
cos(x)	cosinus of x given in radians	b = cos(PI);
tan(x)	tangens of x given in radians	b = tan(PI);

Conditions (if / else)



- The if / else statement controls analysis workflow, allowing to program alternatives.
- Depending on a condition, either the one or the other block is executed.
- Curly brackets {} are used to mark where a block starts and ends.
- Indentation helps reading blocks.



```
if ( condition ){
    // do something if
    // condition is true
}
else{
    // do something else if
    // condition is false
}
// do in any case
```

```
// initialise program
quality = 99.5;

// evaluate result
if (quality > 99.9) {
    print(”Everything is fine.");
} else {
    print(”We need to improve!");
}
```

Comparison operators & logic operators

- Comparison operators always have *true* or *false* as results

Operator	Description	Example
<, <=	smaller than, smaller or equal to[1]	a < b
>, >=	greater than, greater or equal to[1]	a > b
==	equal to[1]	a == b
!=	not equal to[1]	a != 1

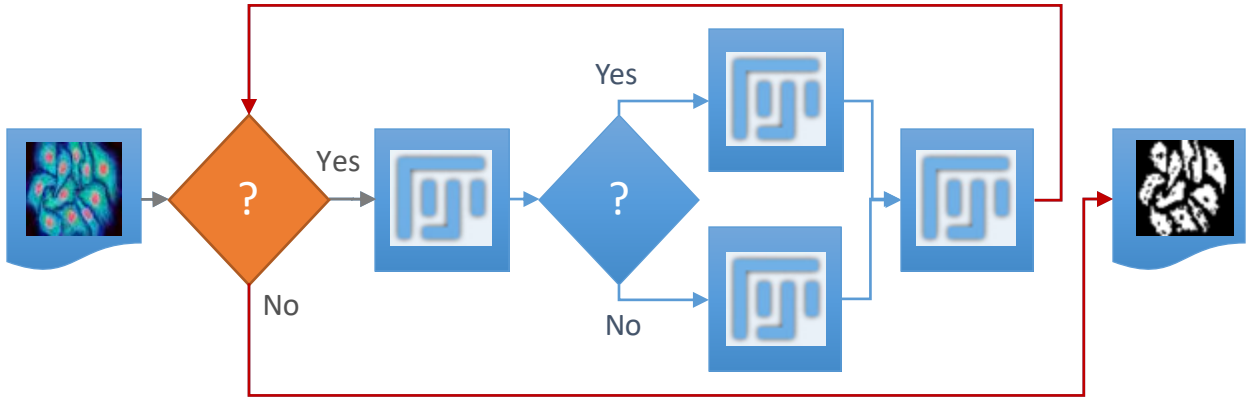
[1] these operators work also with string values

- Logic operators always take conditions as operands and result in a condition

```
// initialise program
quality = 99.9;
age = 3;

if ((quality >= 99.9) && !(age > 5)) {
    print("The item is ok.");
}
```

Operator	Description	Example
&&	and	a < b && b < c
||	or	a < b || b > c
!	negation (“not”)	!(a == b)

Loops (for)



Loop statement

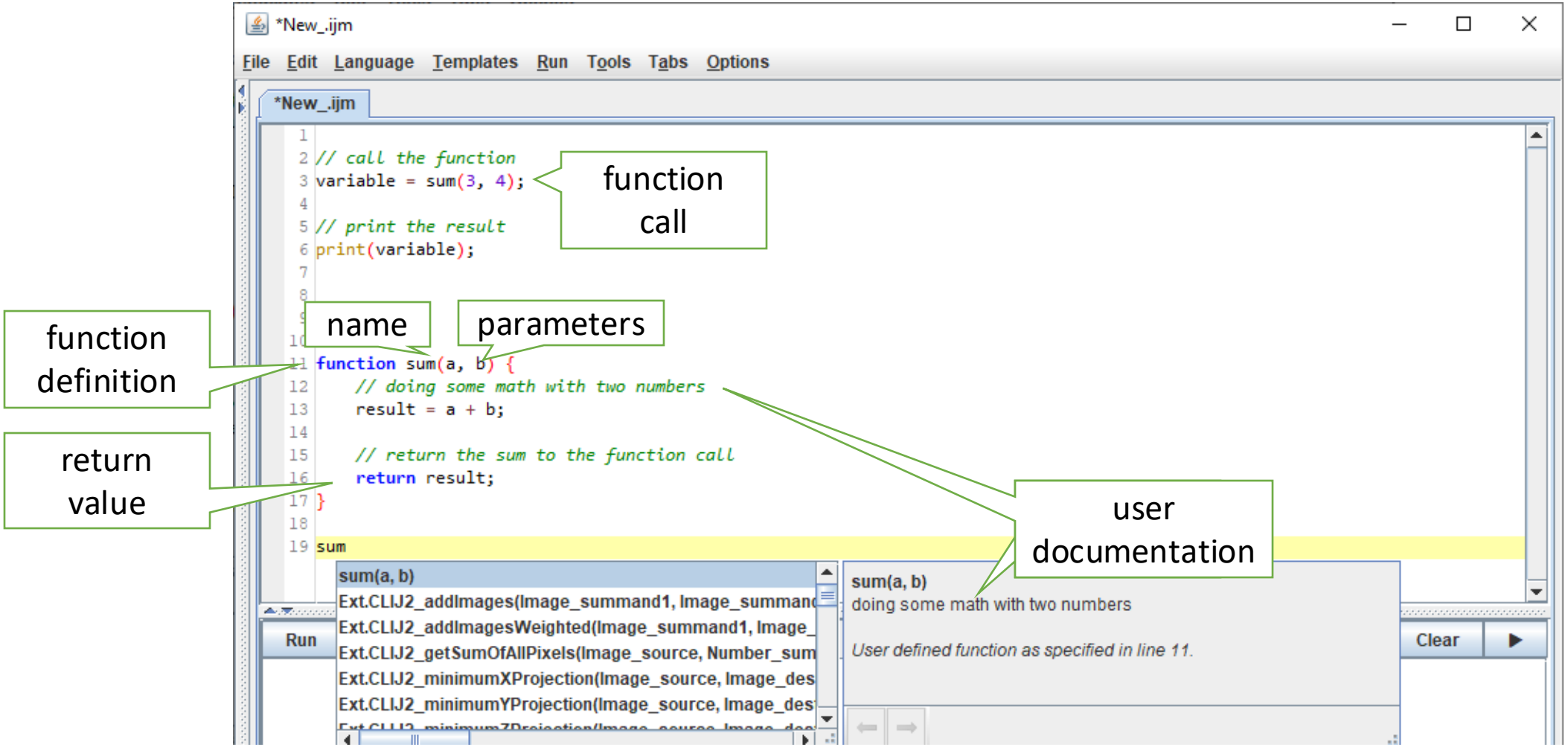
- The `for` statement allows us to execute some lines of code *for* several times
  - The initializer is only executed once; at the beginning.
  - The condition is checked every time
  - After block execution, the counter is increased.

```
// do something once

for(<initializer>; <condition>; <iterator>) ){
    // do something at each
    // iteration of the loop
}

// do another thing once
```

```
for( i = 0; i < 10; i += 1 ){
    print(i);
}
```

Functions

- For reusing code and for organizing code: Use functions!



Writing good (readable) codes

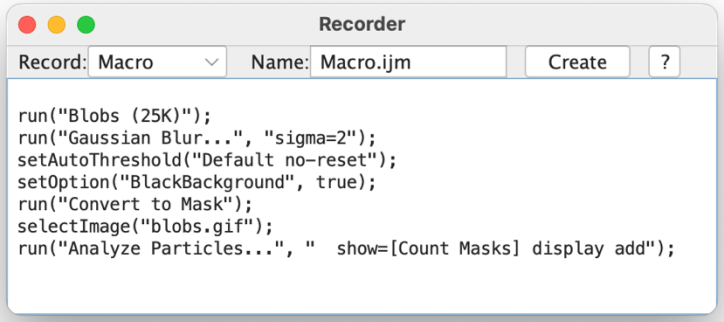
- Every command belongs on its own line
- Insert empty lines to separate important processing steps
- Put spaces between operators and operands, because:

This is easier to read thanthat,orisnt'it?

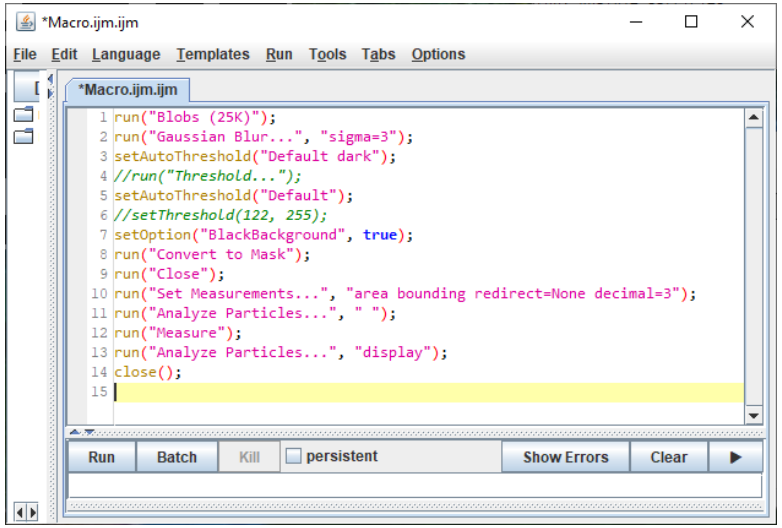
- Indent every conditional block (if/else) using the TAB key
- Hint: put the "{" behind the if; it makes your program shorter.
- Make use of tools: http://jsbeautifier.org

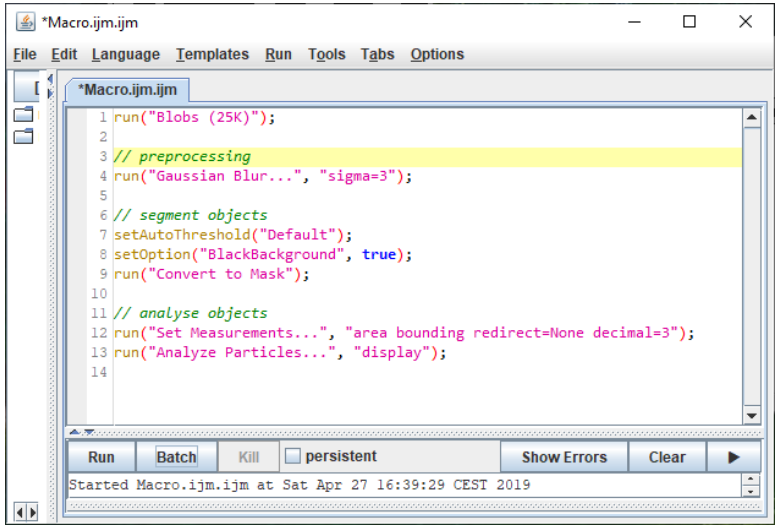
```
// initialise program
a = 5;
b = 3;
c = 8;

// execute algorithm
d = (a + b) / c;

// evaluate result


if (a == 5) {
    print("Yin");
} else {
    print("Yang");
}
```

Macro recording



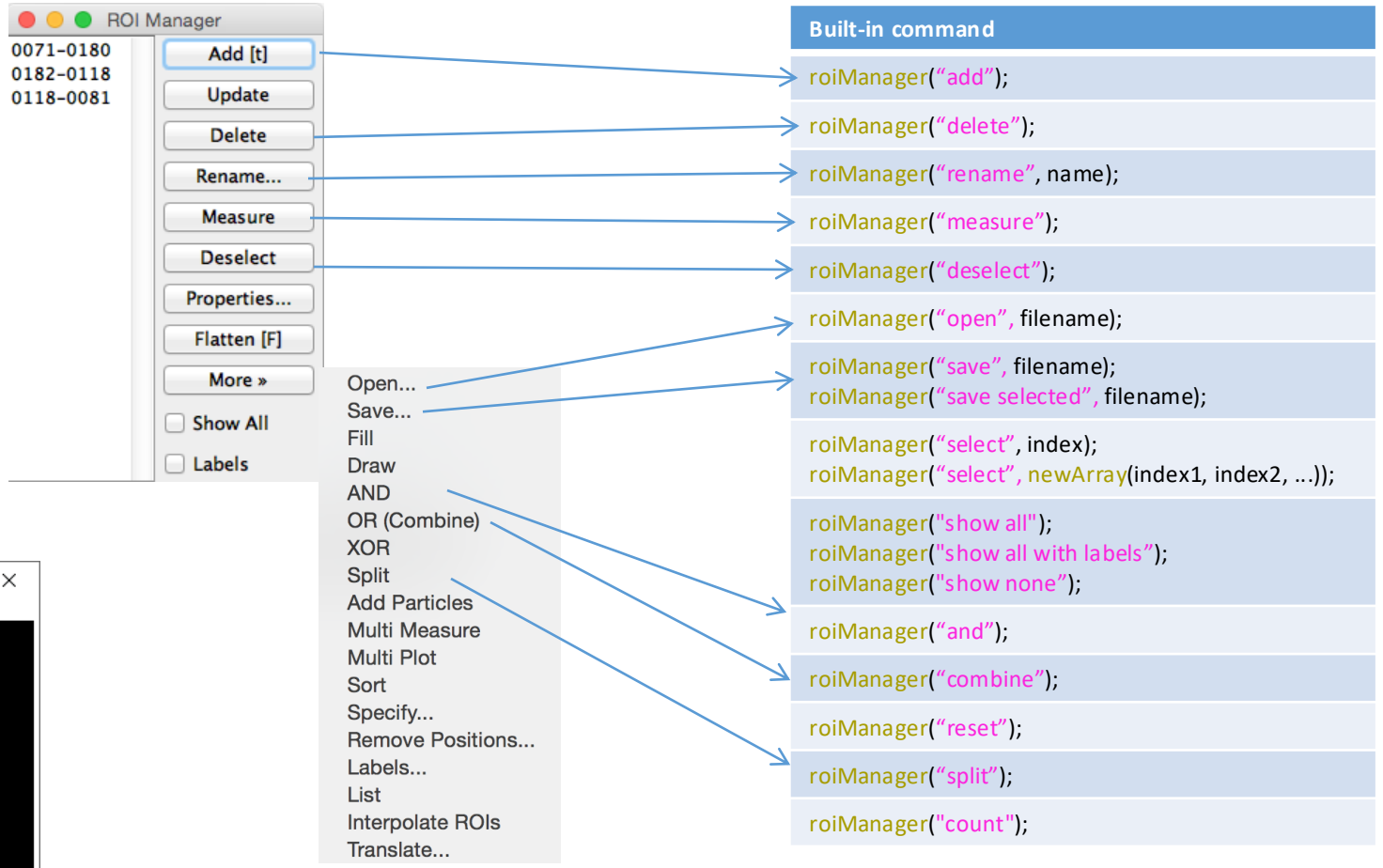
- Almost every action you perform in ImageJ by clicking is recorded!
- The script in recorder can be edited and used for creating .ijm file
- Editing recorded macros needs to be trained. It's 80% reading and 20% writing
- Hints:
  - Put comments first. Try to understand what was recorded and why.
  - Do it in tiny steps. As soon as you have a working workflow consisting of 4-5 steps, create a macro.
  - Collect macros. When you do something new, do cherry picking from the old macros.





ROI Manager

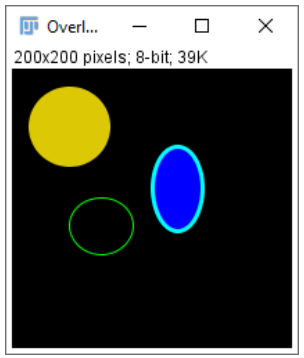
- ROI (Regions of interest)
  - ROIs can be used for visualization of (intermediate) results
  - Every button in the ROI Manager can be called from a macro.



- Overlay: collection of ROIs

```
3  // Draw filled elipses
4  newImage("Overlays", "8-bit black", 20
5  makeOval(11, 12, 59, 58);
6  Roi.setFillColor(220, 200, 4);
7  Overlay.addSelection();
8
9  // Draw outline elipses
10 makeOval(40, 92, 46, 41);
11 Roi.setStrokeColor("Green");
12 Overlay.addSelection();
13
```

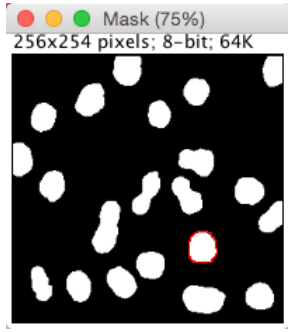
overlays.ijm



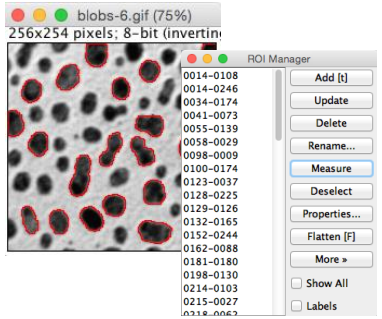
Results tables

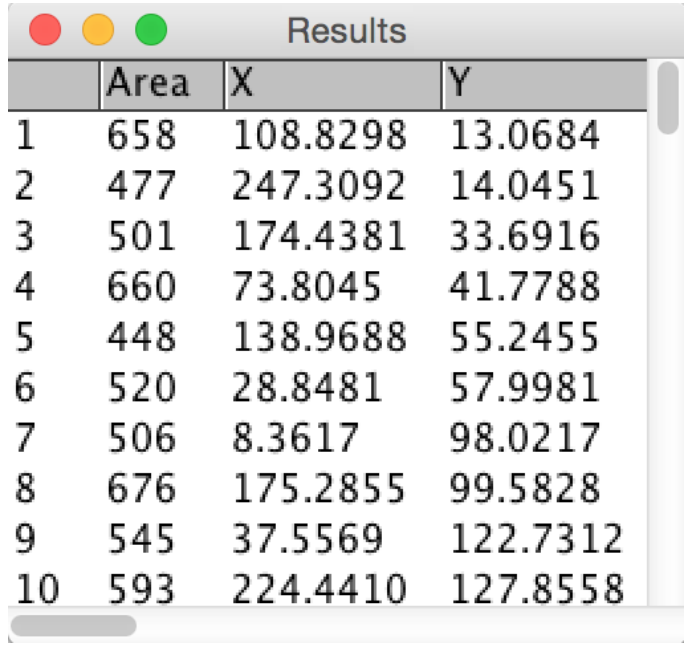
- ImageJ/FIJIs tool for collecting data/measurements
- Two ways for creating result tables

```
run("Analyse Particles...", "display");
```



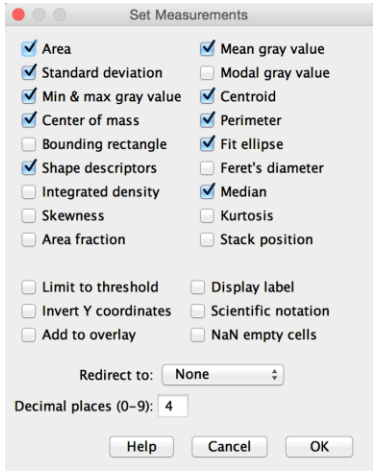

```
roiManager("Measure");
```





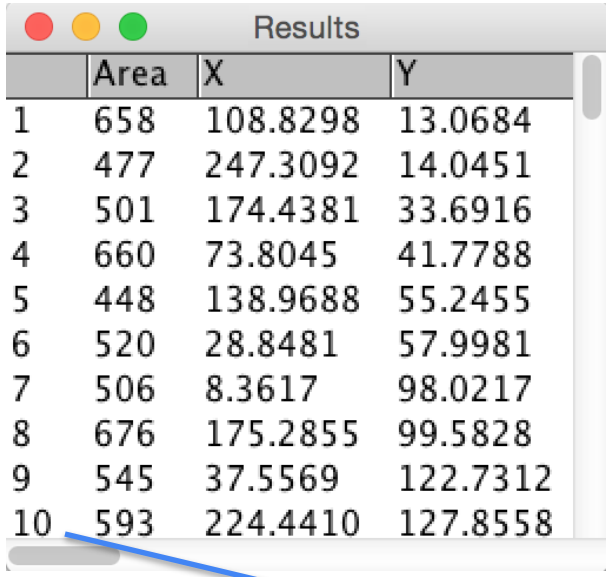
	Area	X	Y
1	658	108.8298	13.0684
2	477	247.3092	14.0451
3	501	174.4381	33.6916
4	660	73.8045	41.7788
5	448	138.9688	55.2455
6	520	28.8481	57.9981
7	506	8.3617	98.0217
8	676	175.2855	99.5828
9	545	37.5569	122.7312
10	593	224.4410	127.8558

- Set Measurements

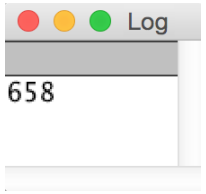


Results tables

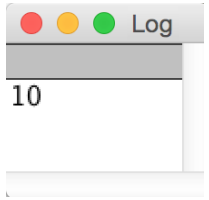
- Reading elements in tables




```
area = getResult("Area", 0);
print(area);
```



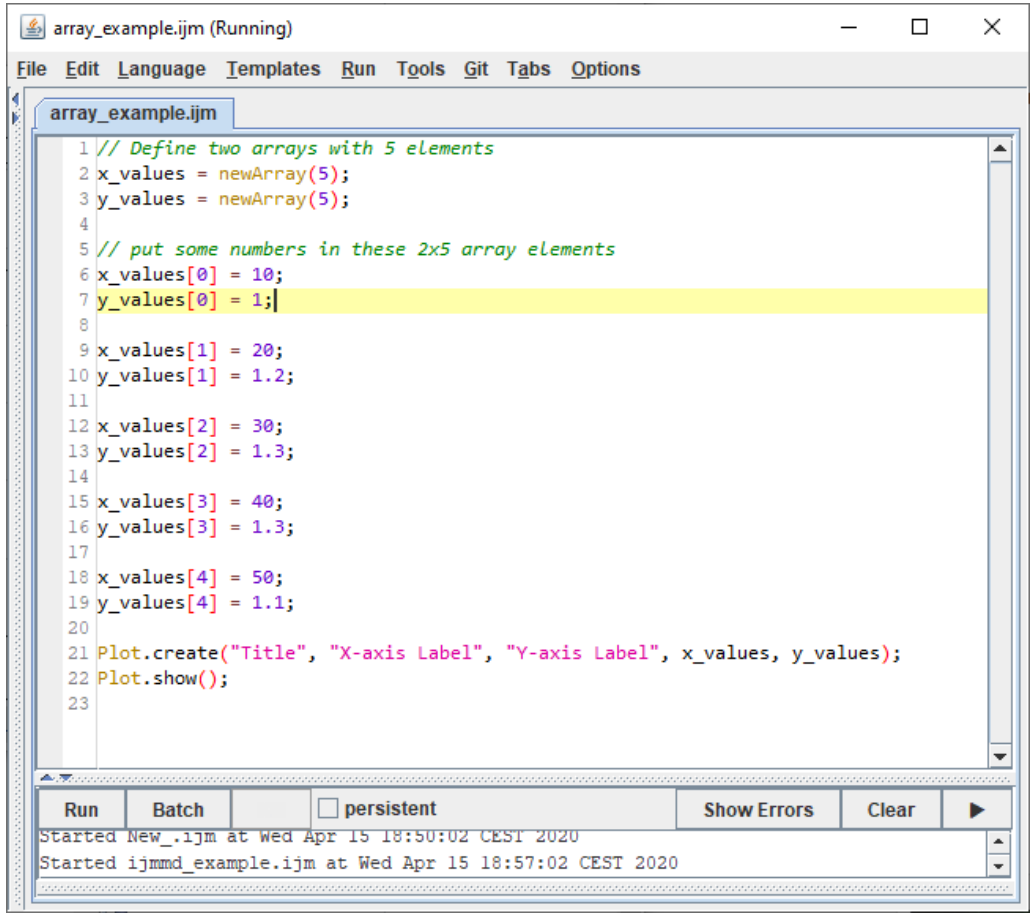

```
r = nResults();
print(r); // number of rows
```

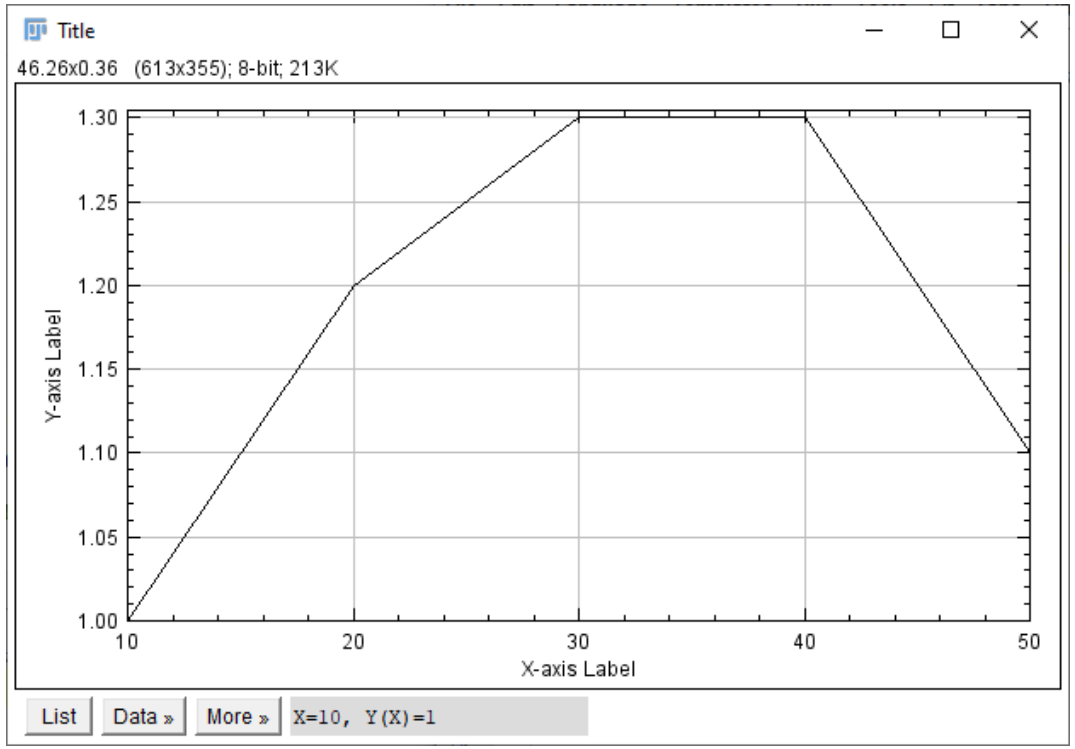


Built-in command
value = getResult (“column title”, rowNumber);
rowCount = nResults();

Plots in ImageJ

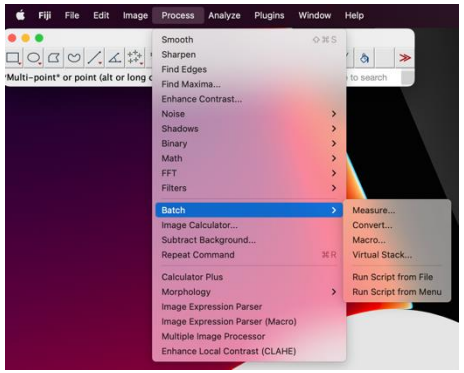
- Simple plots can be drawn using arrays

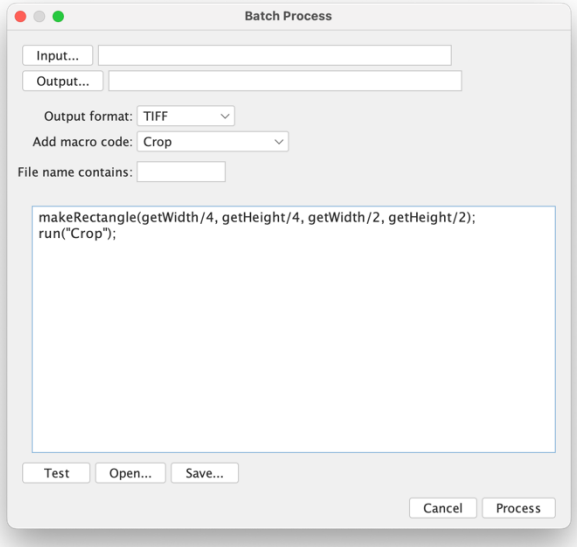




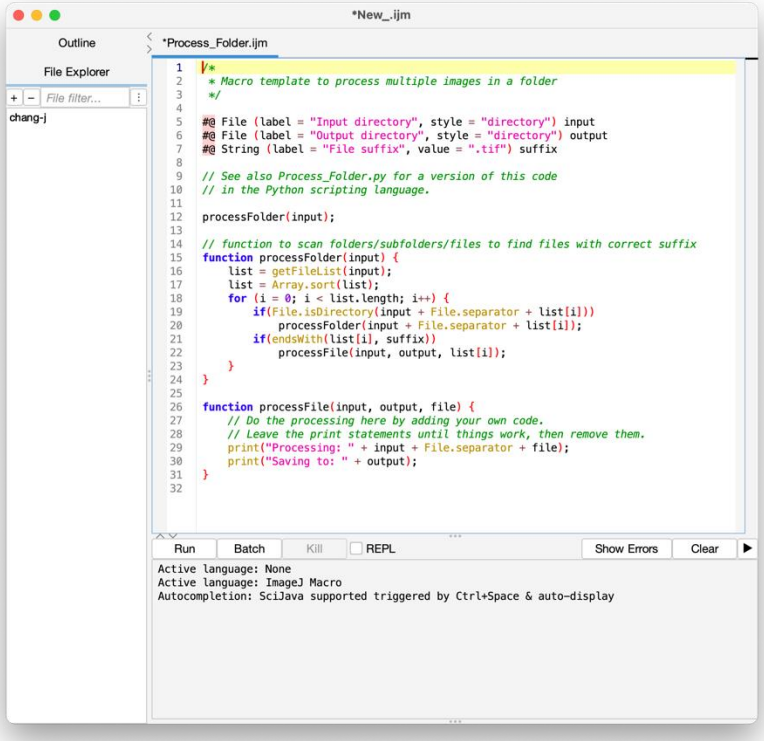
Batch processing

- Use ImageJ submenu
  - Measure
  - Convert
  - Macro
  - Virtual Stack
  - Note: silently override existing files with the same name





- Create your own
  - Start with a template in script editor
    - *Template › ImageJ 1.x › Batch › Process Folder (ImageJ Macro)*



Working with image files

- Open and close commands allow handling image files.

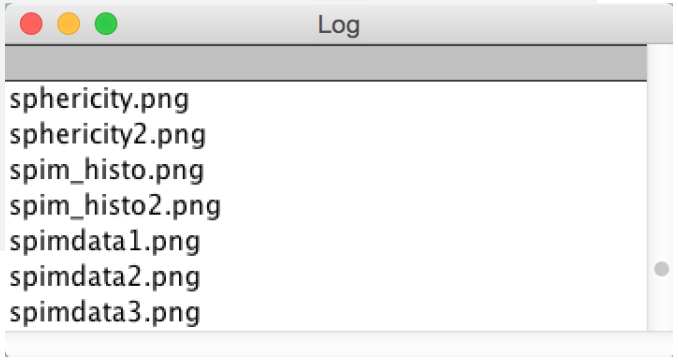
```
// initialise program
imageFilename = "/Users/rhaase/images/blobs.gif";

open( imageFilename );

// process the image
// ...

close();
```

Built-in command	Description	Parameters
open(filename);	open an image	filename of the image
close();	close current image	

Working with image files in a folder

- The files in a folder are represented as an `array[]`

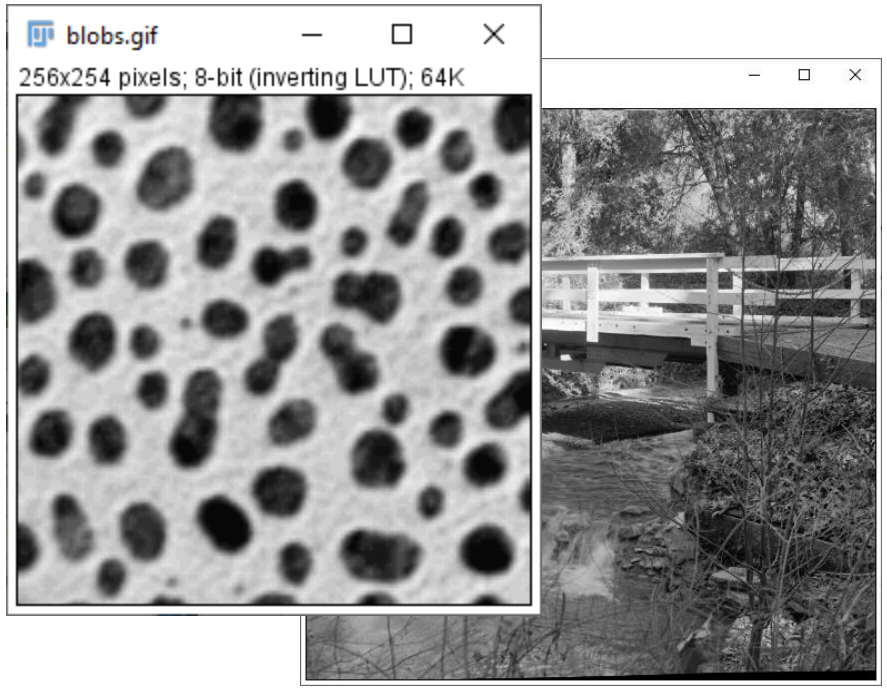
```
// initialise program
foldername = "/Users/rhaase/temp/";

// get all files in the folder as array
list = getFileList( foldername );

// print out the array; item by item
for (i = 0; i < lengthOf(list); i += 1 ) {
    filename = list[i];

    print( filename );
}
```

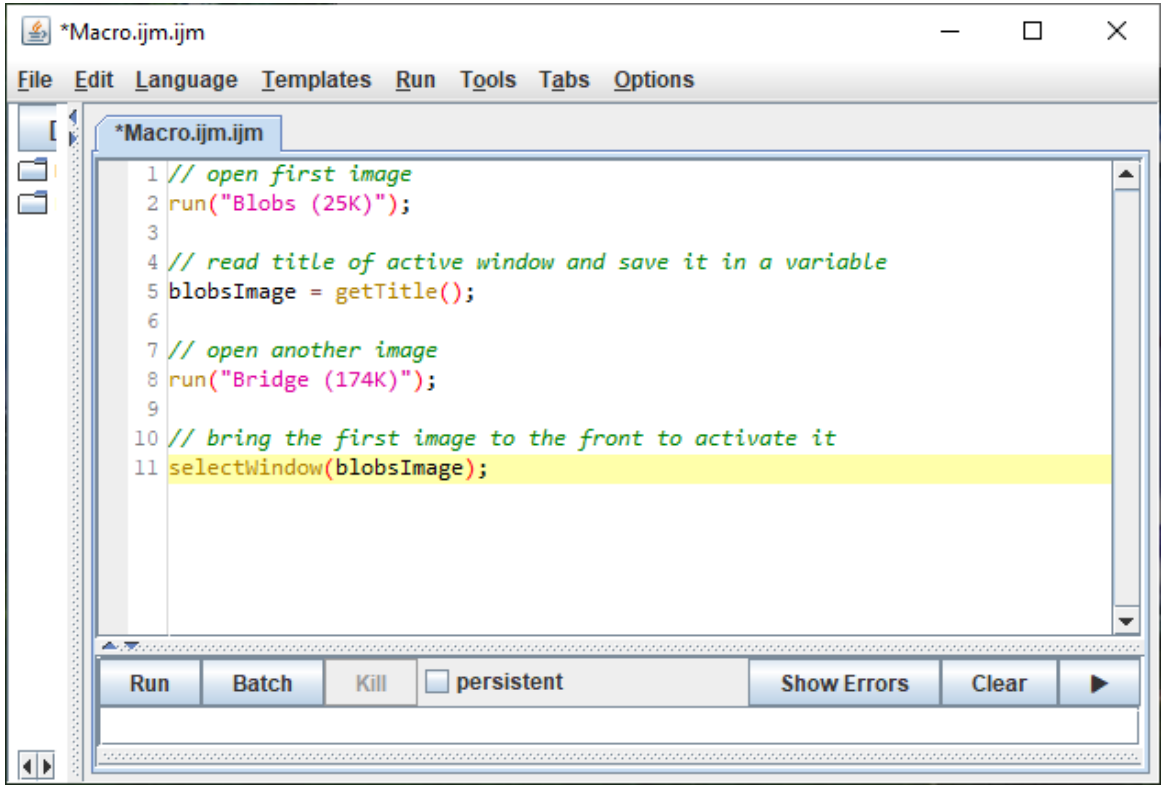


Built-in command	Description	Parameters
filenameList = getFileList(foldername)	return a list of file names	location name of the folder

Hint: switching between windows

- If you work with several images, you may want to switch between them. Use
  - ***getTitle()*** to get the headline of the current window
  - ***selectWindow(title)*** to select a window to bring it to the front
  - ***rename(new_title)*** to change the name of a window





Hint: logging

- There is an alternative to write simple tables and text files to disc

```
path = "C:/structure/teaching/lecture_applied_bioimage_analysis/06_example_code/test.csv";

headline = "Number, number squared";

File.append(headline, path);

for (i = 0; i < 10; i++) {
    contentline = "" + i + ", " + pow(i, 2);
    File.append(contentline, path);
}
```

save_csv_file.ijm

test.csv

```
Number, number squared
0, 0
1, 1
2, 4
3, 9
4, 16
5, 25
6, 36
7, 49
8, 64
9, 81
```

Part2: Hands-on session

Summary & Acknowledgement

- ImageJ Macro intro
  - Common programming concept
  - ImageJ Macro syntax and functions
- Exercise
  - Ex1: automate a basic workflow
  - Ex2: batch processing
- Next week: Interactive analysis with CLIJ

課後意見調查

國立臺灣大學 National Taiwan University
台大醫學院 National Taiwan University College of Medicine
NSTC 國家科學及技術委員會 National Science and Technology Council
iBC