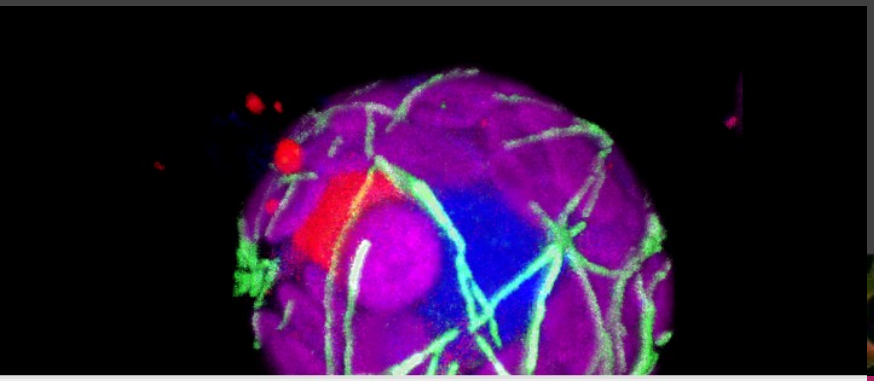
ImageJ Macro Programming for Biological Image Analysis

05/2021



*Macro.ijm

```
setBatchMode(true);
while(next()) {
    run("Analyze Image");
}
setBatchMode("exit and display");
```

File Edit Font

	Skeleton ID	Branch length	V1 x	V1 y	V1 z	V2 x	V2 y	V2 z	Euclidean
136	42.000	0.535	13.401	23.485	6.816	13.265	23.688	7.175	0.434
137	43.000	0.135	8.799	17.123	7.175	8.799	17.259	7.175	0.135
138	44.000	1.101	13.265	8.663	7.175	13.333	8.460	7.534	0.418
139	45.000	1.391	27.952	18.883	7.175	27.546	19.830	7.534	1.092
140	46.000	1.424	8.257	15.161	7.893	7.580	14.957	8.610	1.007
141	47.000	1.081	10.626	21.929	8.252	10.152	21.793	8.969	0.870
142	48.000	0.135	28.697	18.003	8.610	28.832	18.003	8.610	0.135
143	49.000	0.096	28.900	14.551	8.610	28.967	14.484	8.610	0.096
144	50.000	0.096	14.890	25.313	8.969	14.957	25.245	8.969	0.096
145	51.000	0.327	19.018	4.805	8.969	19.289	4.670	8.969	0.303

Total branch length: 342.433

Leslie Bancel-Vallee

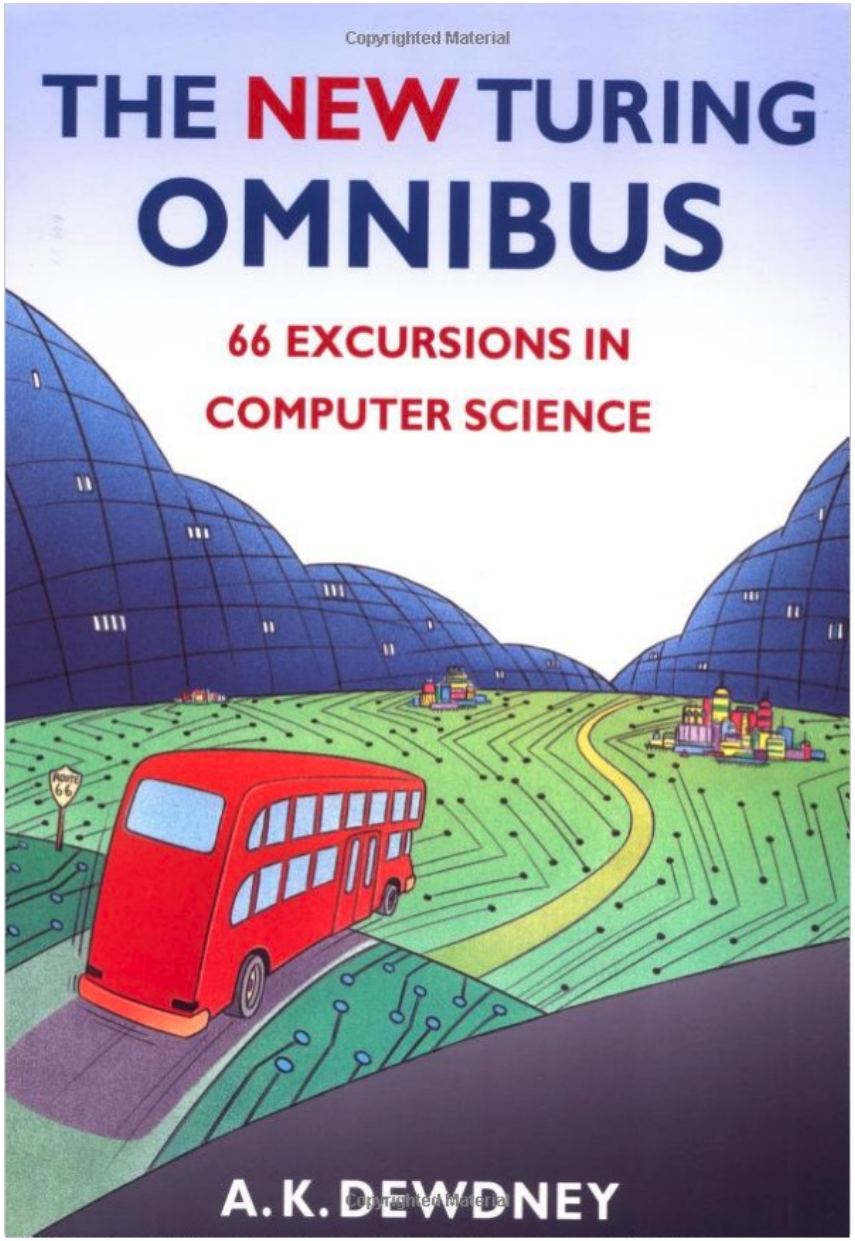
Volker Baecker

Montpellier RIO Imaging

volker.baecker@mri.cnrs.fr

BioCampus Montpellier

Programming and computer science are fun :)

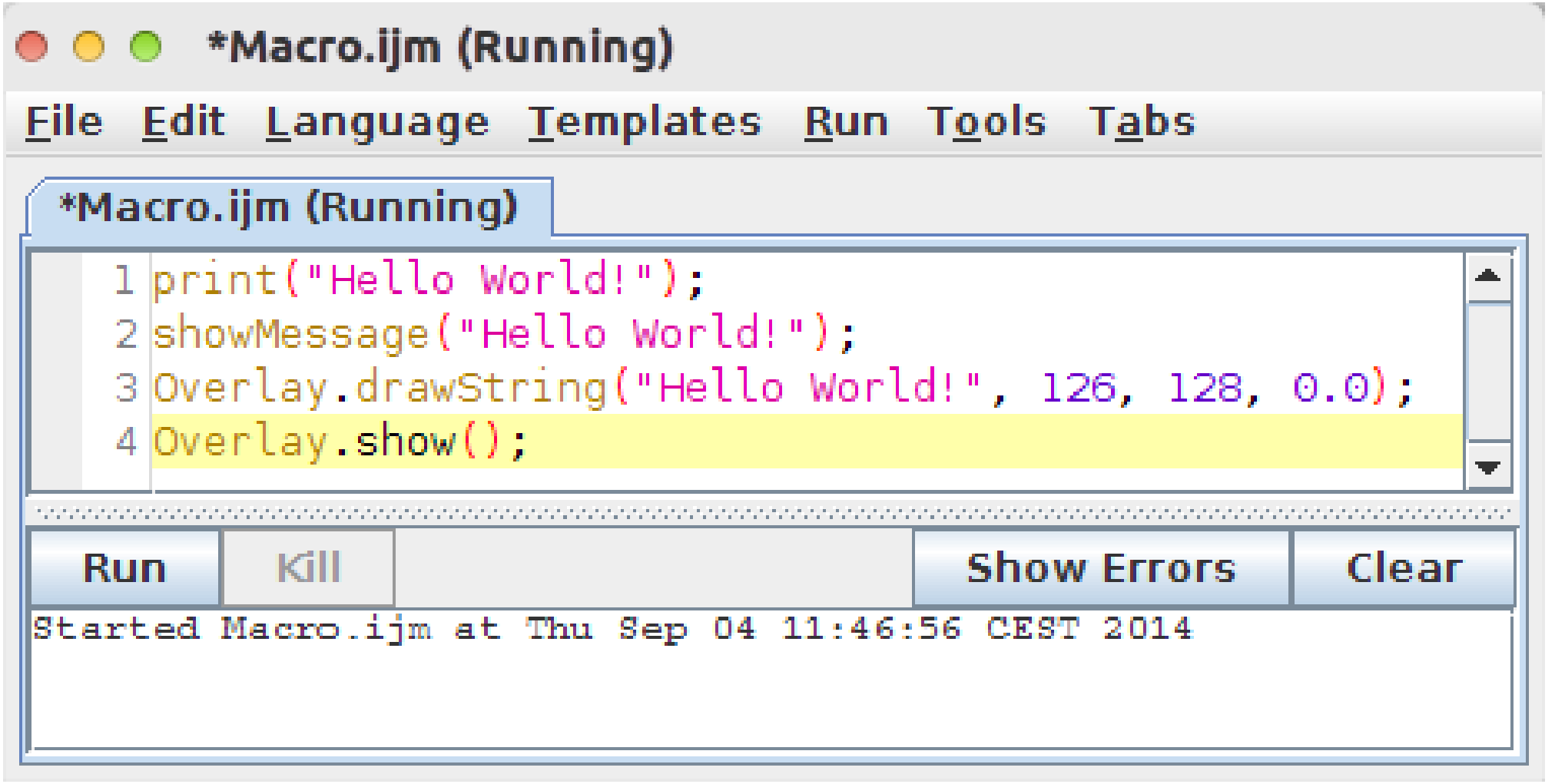


Macros, Scripts and Plugins

- Macro
  - automatize ImageJ
  - interpreted by ImageJ
  - simple
  - missing data structures
  - limited reusability
  - slow
- Script
  - general programming
  - jvm or script interpreter
  - complex
  - datastructures available
  - better reusability
  - slow
- Java Plugin
  - general programming
  - java virtual machine
  - complex
  - datastructures available
  - good reusability
  - fast

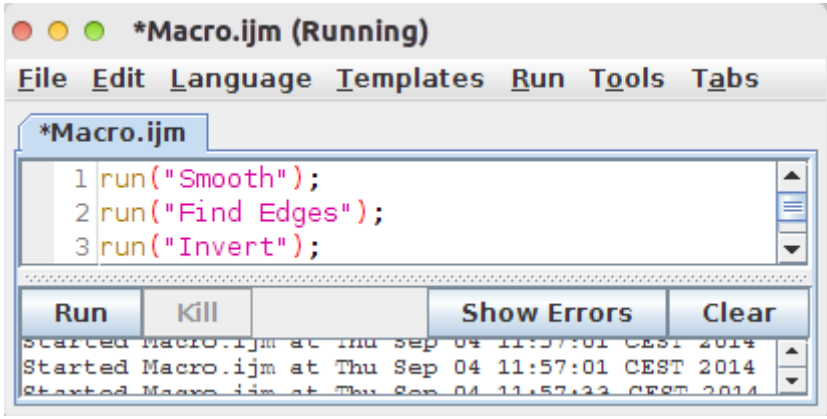
The FIJI Macro Editor

- Plugins>New>Macro



Recording Commands

- You don't know how to use a command in a macro? - Just record it !
- Open sample image File>Open Samples>Clown
- Plugins>Macros>Record…
- Run
  - Process>Smooth
  - Process>Find Edges
  - Edit>Invert
- Create the macro
- Close the image
- Open another sample image
- Run the macro



Exercise 01.01-01.04

Basic Datatypes - Numbers

- floating point
  - special values
    - NaN, Infinity, -Infinity
  - operations
    - - (unary), +, -, *, /, %
  - bitwise operations
    - ~, |, ^, &, <<, >>
  - build in functions
    - parseFloat, sin, cos, sqrt, pow, exp, log, floor, round, ...

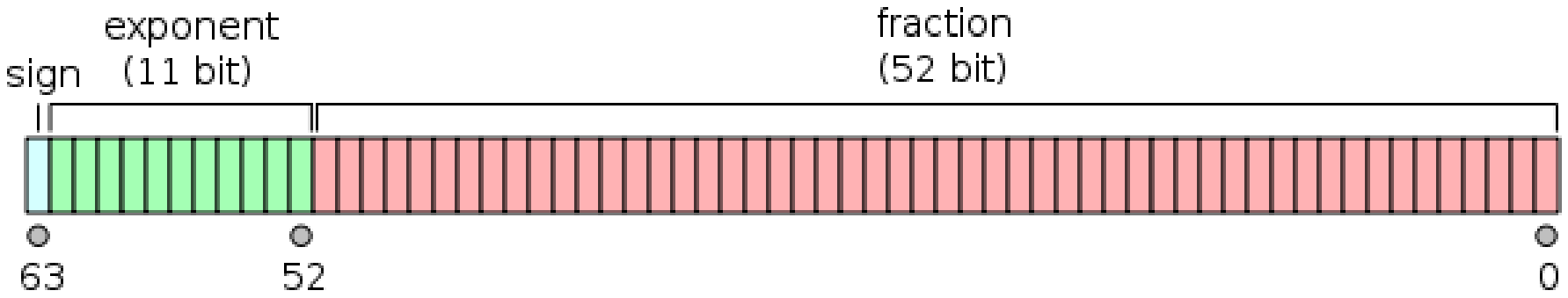
Decimal, Binary, Hexadecimal

$$1234_{10} = 1 \cdot 10^3 + 2 \cdot 10^2 + 3 \cdot 10^1 + 4 \cdot 10^0$$

$$10011010010_2 = 1 \cdot 2^{10} + 1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^4 + 1 \cdot 2^1$$

$$4D2_{16} = 4 \cdot 16^2 + 13 \cdot 16^1 + 2 \cdot 16^0$$

Floating Point



$$(-1)^{sign}(1.b_{51}b_{50}\ldots b_0)_2 \times 2^{e-1023}$$

```
0.3 - 0.2 - 0.1;
(0.3 - 0.2 - 0.1 == 0);
(0.3 - 0.2 - 0.1 < 0.00000000001);
```

Exercise 02.01-02.03

Strings

- literals in quotes or double-quotes
- \ is escape character
- \n newline
- operations
  - concatenation (+)
- build in functions
  - endsWith, indexOf, lastIndexOf, lengthOf, matches, replace, split, substring, toLowerCase

Strings and regular expressions

- concatenation
  - "Hello" + " World!" ;
  - "2 + 2 = " + 2+2 ;

- split, matches and replace work with regular expressions

[ ]	set	[aA]	either a or A
-	range	[0-9]	any digit
.	any character	[0-9].	a digit followed by one character
*	zero or more	.*	any string
?	zero or one	[0-9] ?	an optional digit
+	one or more	[0-9]+	one or more digits
^	negation	[^0-9]	any character that is not a digit
&&	and	[0-9&&[^3]]	a digit that is not 3
|	or	[0-9]|[a-zA-Z]	a digit or a lower or upper case letter
()	a group		

```
replace("A01GFP_c001_t001_z001.tif",
        "_t([0-9][0-9][0-9])", "_t0$1");
```

Exercise 03.01-03.04

Booleans

- Comparison operations result in boolean
  - ==, <, >, <=, >=, !=
- Operations on booleans are
  - not (!), and (&&), or(||)
- example
  - (x>0 && x<2048)
- {and, or, not} functionally complete set of boolean operations
  - xor:

    ((!a && b) || (a && !b))

```
"not";
(!false);                1
(!true);

"and";
(false && false);    0
(false && true);     0
(true && false);     0
(true && true);  1

"or"
(false || false);  0
(false || true);       1
(true || false);       1
(true || true);              1
```

Exercise 04.01-04.02

Variables

- Variable
  - has identifier and value
  - must be defined by an assignment
  - copied by value for basic types, by reference for arrays
  - has a scope
  - operations on variables
    - ++, --, +=, -=, *=, /=

```
radius = 11.25;
```

Exercise 05.01-05.03

Array

- indexed variable
  - a[i]
  - can be created literally or with a size
  - elements can have different types
  - elements can only be of basic types
  - only one dimensional arrays
  - a.length gives the number of elements in a

- Array.concat
- Array.copy
- Array.fill
- Array.findMaxima
- Array.findMinima
- Array.getStatistics
- Array.print
- Array.show,
- ...

```
options = newArray(34, true, "Huang");
print("threshold value:", options[0]);
print("dark background:", options[1]);
print("threshold method:", options[2]);
```

Exercise 06.01-06.03

if...then...else

- conditional execution of code
  - depending on input or state

```
if (condition) {
   list of statements 1
} else {
   list of statements 2
}
```

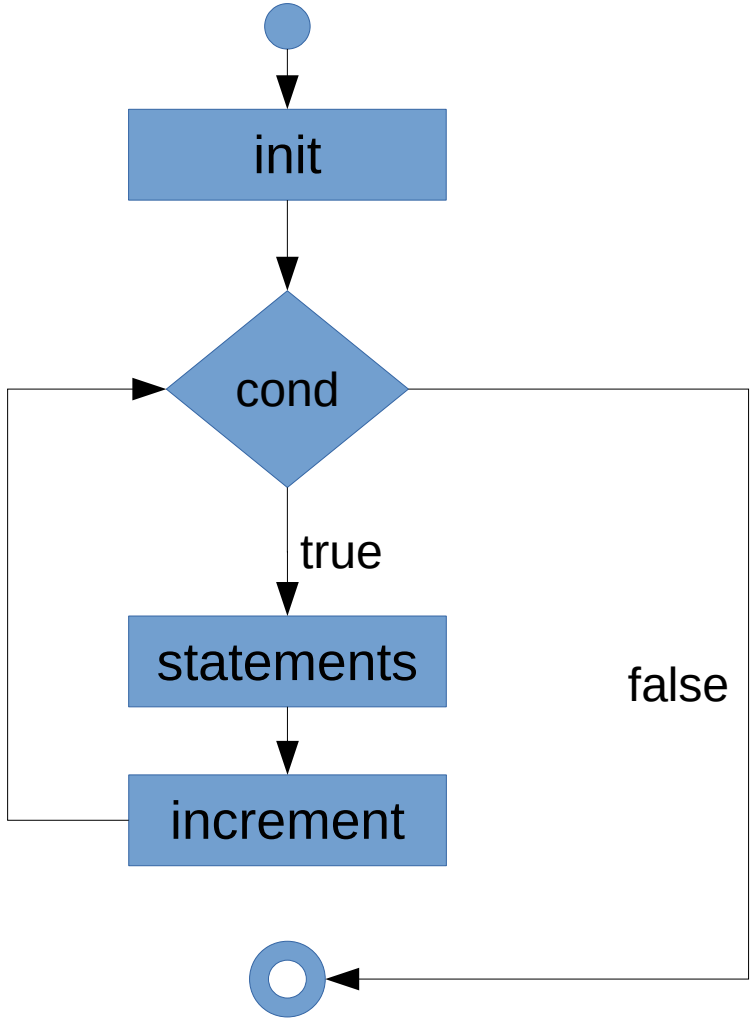
Exercise 07.01-07.02

loops

- repeatedly exexute a code block
  - a condition is evaluated for each iteration and decides when the loop finishes
- three flavours
  - for
    - number of iterations known
  - while
    - condition before each iteration
  - do while
    - condition after each iteration

for

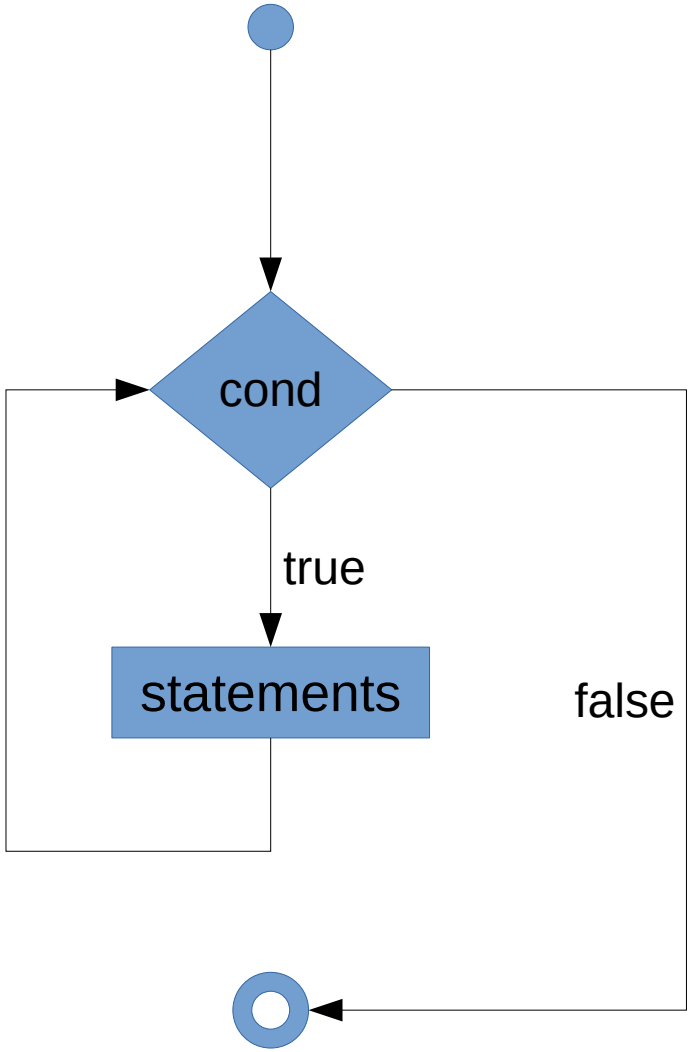
```
for (<initialization>; <condition>; <increment>) {
   <list of statements>
}
```



Exercise 08.01-08.04

while

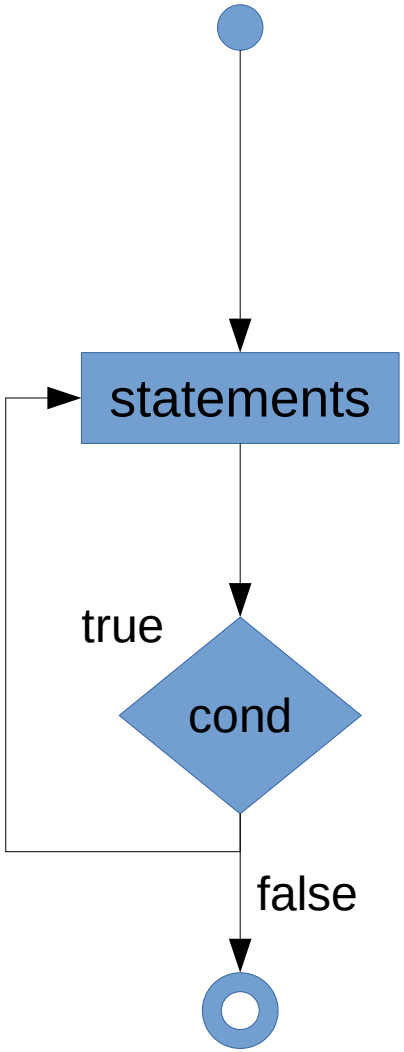
```
while (<condition>) {
   <list of statements>
}
```



Exercise 09.01-09.02

do-while

```
do {
   <list of statements>
} while (<condition>);
```



Exercise 10.01

user defined functions

```
function <name_of_the_function>(<list of args>) {
   <list of statements>
}

function max(x,y) {
   if (x>y) return x;
   else return y;
}

r = max(19, 42);
print(r);
```

Variable scope and global variables

- functions
  - have their own variable scope
  - can communicate via global variables

```
a = "outer";
show();
print(a);
function show() {
   a = "inner";
   print(a);
}

inner
outer
```

```
var a = "outer";
show();
print(a);
function show() {
   a = "inner";
   print(a);
}

inner
inner
```

recursion

- functions
  - can call themselves
  - can call each other mutually
  - termination condition needed
- useful for
  - compact programs
  - traversing recursive structures (for example trees)

```
f(0) = 1
f(n) = n * f(n-1)

f(5)
5 * f(4)
5 * 4 * f(3)
5 * 4 * 3 * f(2)
5 * 4 * 3 * 2 * f(1)
5 * 4 * 3 * 2 * 1 * f(0)
5 * 4 * 3 * 2 * 1 * 1
5 * 4 * 3 * 2 * 1
5 * 4 * 3 * 2
5 * 4 * 6
5 * 24
120
```

Exercise 11.01-11.04

Macros as plugins

- save macro with underscore in the name in plugins folder
- run it from the menu or
- create shortcut via
  - Plugins>Shortcuts>Create Shortcut...

Defining macros

- macro sets
  - multiple macros in one file
  - assign a shortcut for each macro

```
macro "<name> [<short-cut>]" {
   <list of commands>
}
```

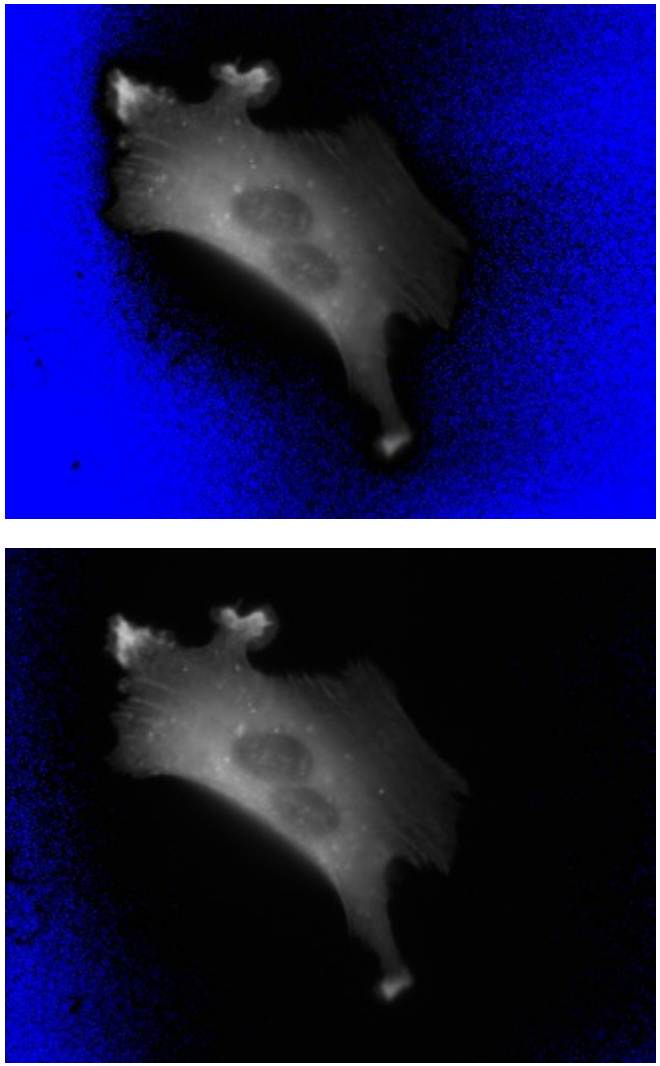
Exercise 12.01-12.02

Part II

- Background correction on a stack
- Batch Merge channels
- Batch Measure cells
- Separate touching objects
- Sort rois and measurements
- Show rois of selected measurements (Link the results table to the image)

Background correction on a stack

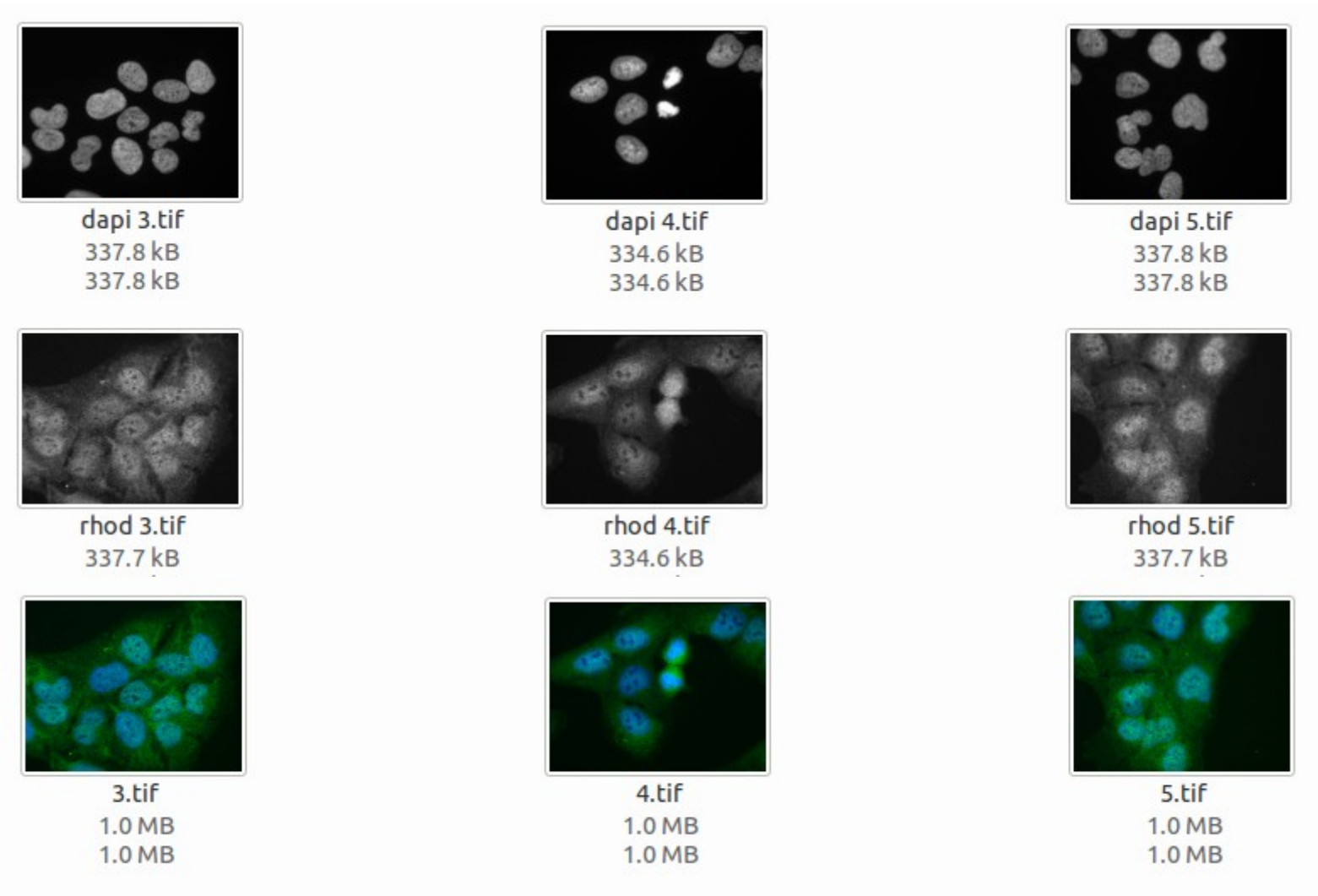
- the user selects a background region
- the macro
  - iterates over the series
    - measures the mean intensity
    - removes the region
    - subtracts the value from the image
    - restores the region
  - useful commands
    - nSlices - the number of slilces in the stack
    - setSlice(i) - set the current slice
    - getStatistics(area, mean);
    - run("Select None");
    - run("Restore Selection");
    - run("Subtract...", "value=5 slice");



actine-stack.tif

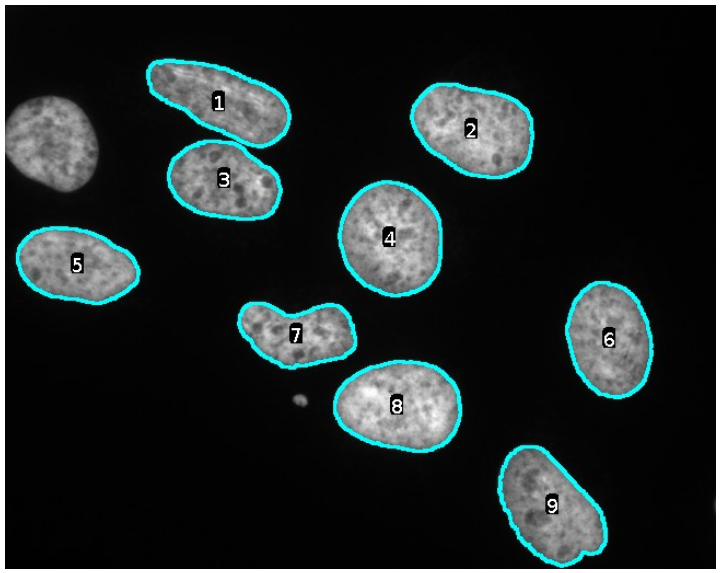
Exercise 13

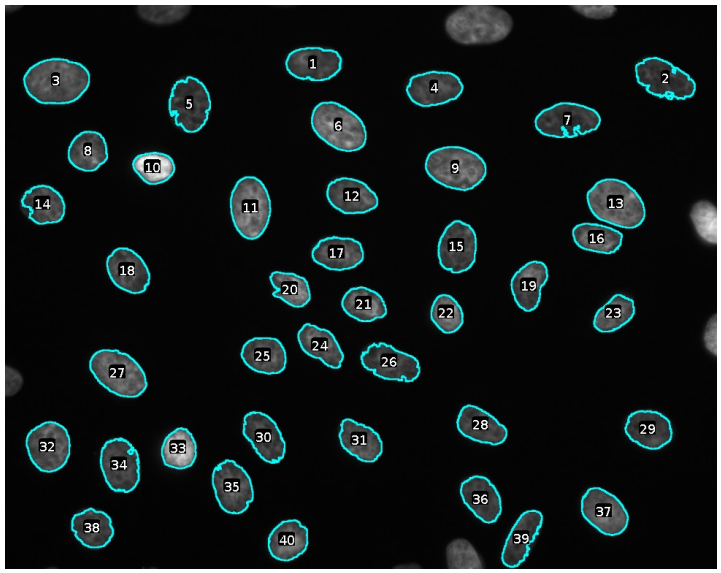
Merge channels of all Images in a folder

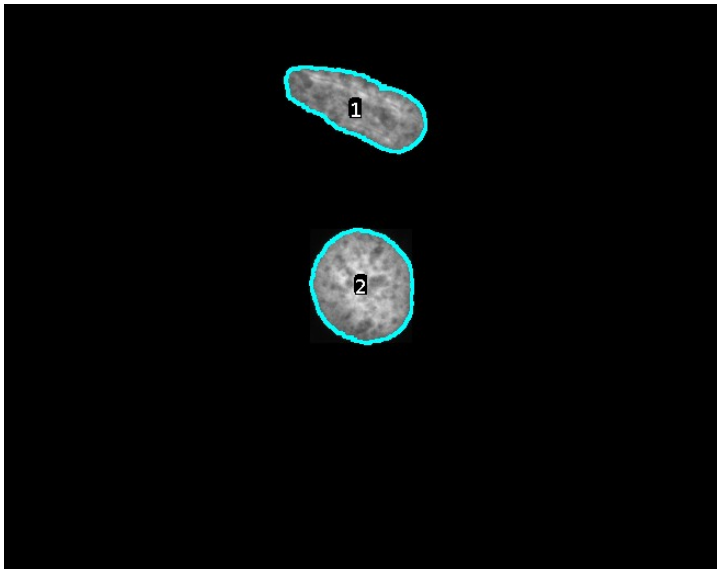


Exercise 14.01-14.05

Batch - measure cells



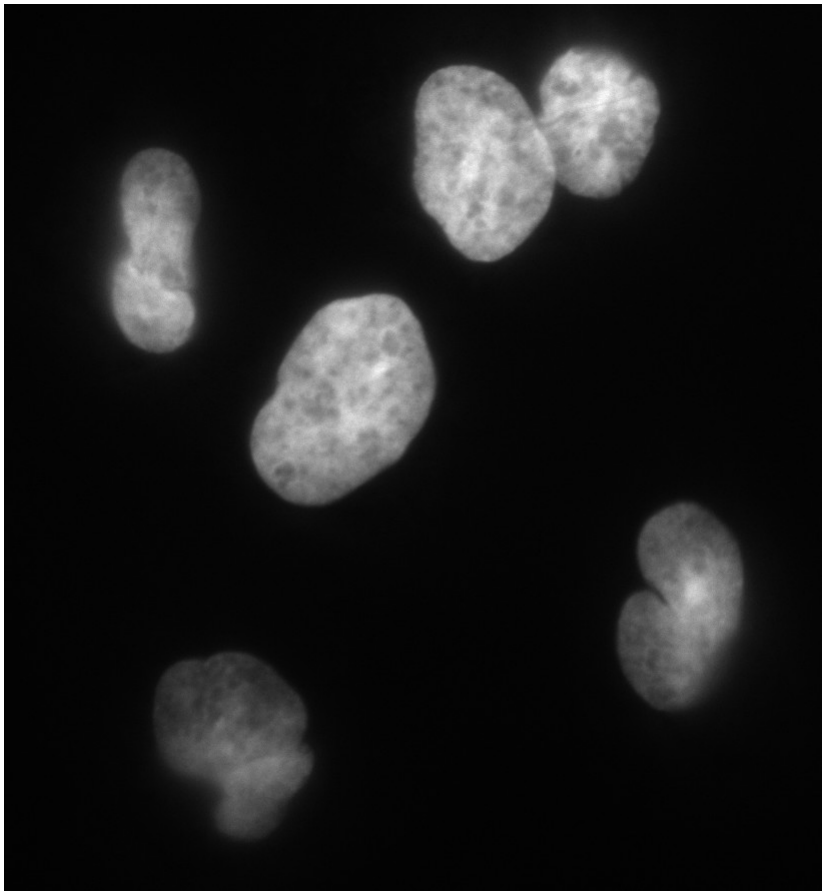


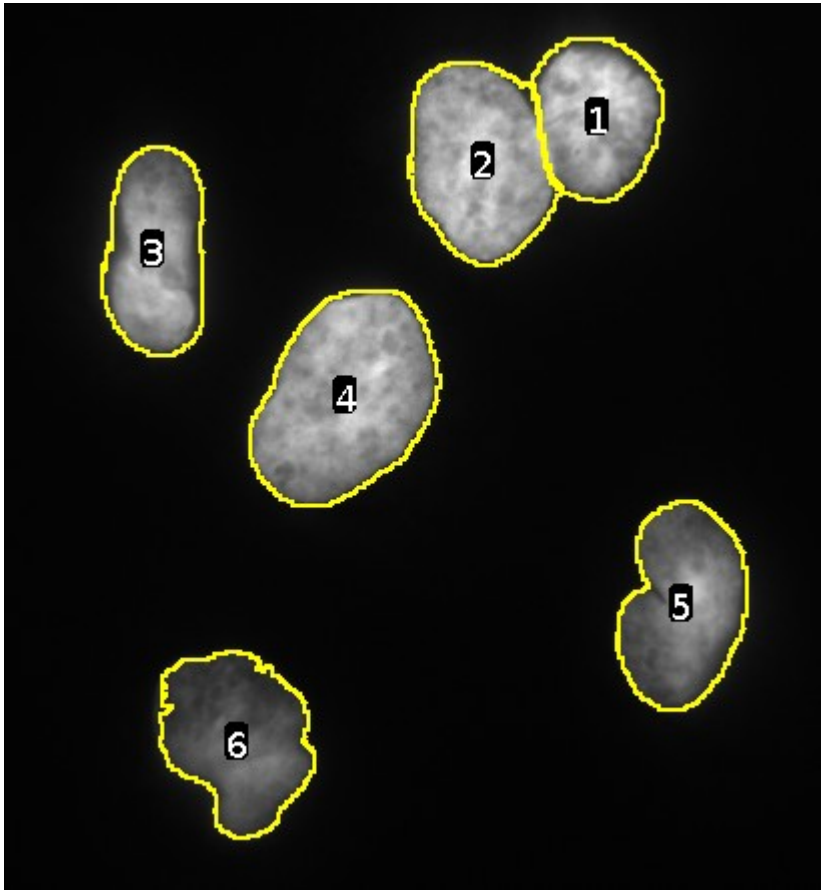


	Label	Area	Mean	StdDev	Mode	Min	Max
1	A4 dapi 1.tif	6101	126.281	25.558	116	72	222
2	A4 dapi 1.tif	7047	149.474	31.401	157	72	239
3	A4 dapi 1.tif	5455	126.024	26.887	135	72	235
4	A4 dapi 1.tif	7524	145.870	32.919	150	72	246
5	A4 dapi 1.tif	5653	135.360	23.931	145	72	198
6	A4 dapi 1.tif	6178	132.127	25.064	132	72	211
7	A4 dapi 1.tif	4583	137.211	31.462	130	72	224
8	A4 dapi 1.tif	7312	167.040	36.387	167	72	255
9	A4 dapi 1.tif	6820	123.350	25.957	126	72	210
10	nuclei.tif	4656	60.545	12.363	54	41	108

Exercise 15

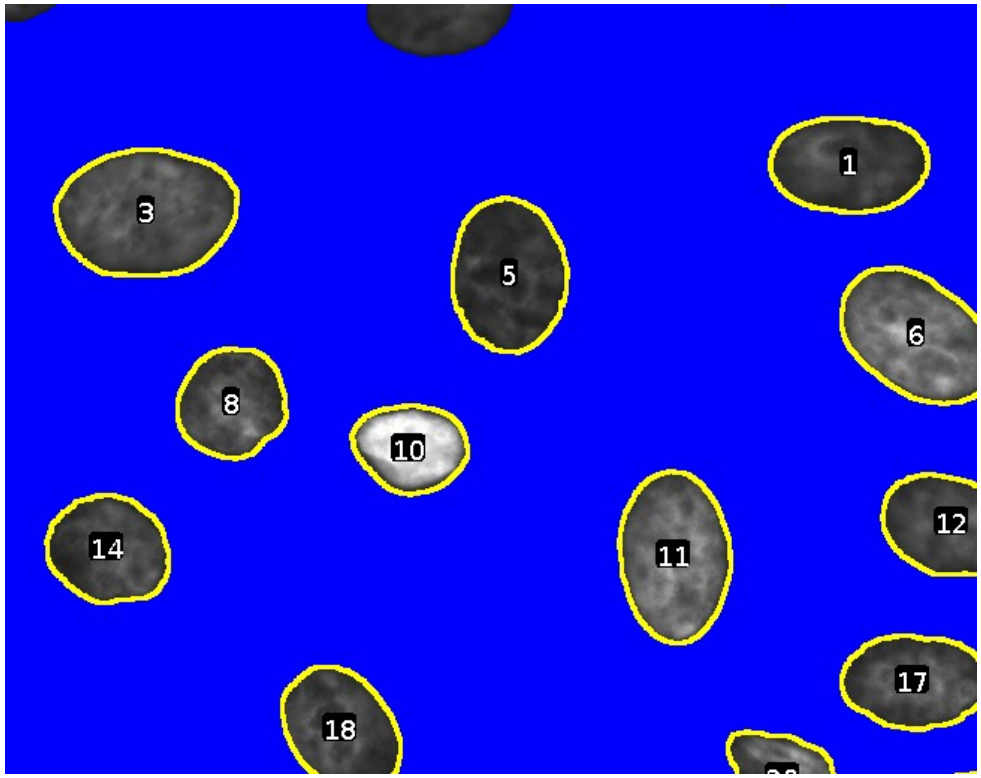
Separate touching objects using a watershed



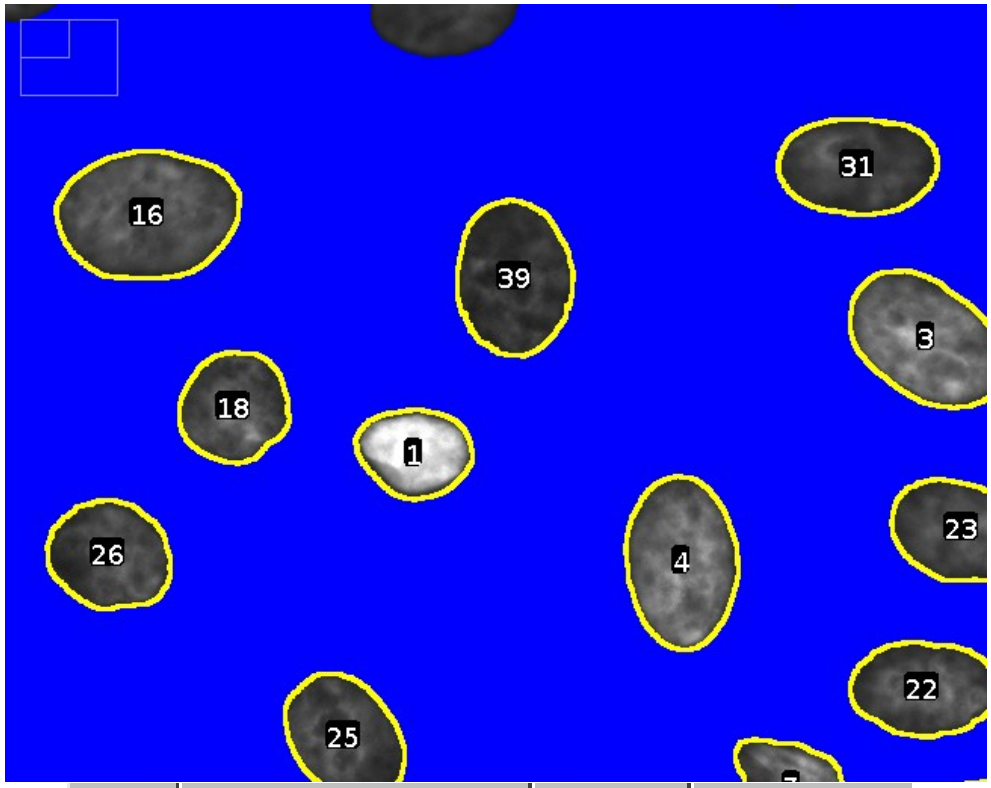


Exercise 16

Sort rois and measurements by a column



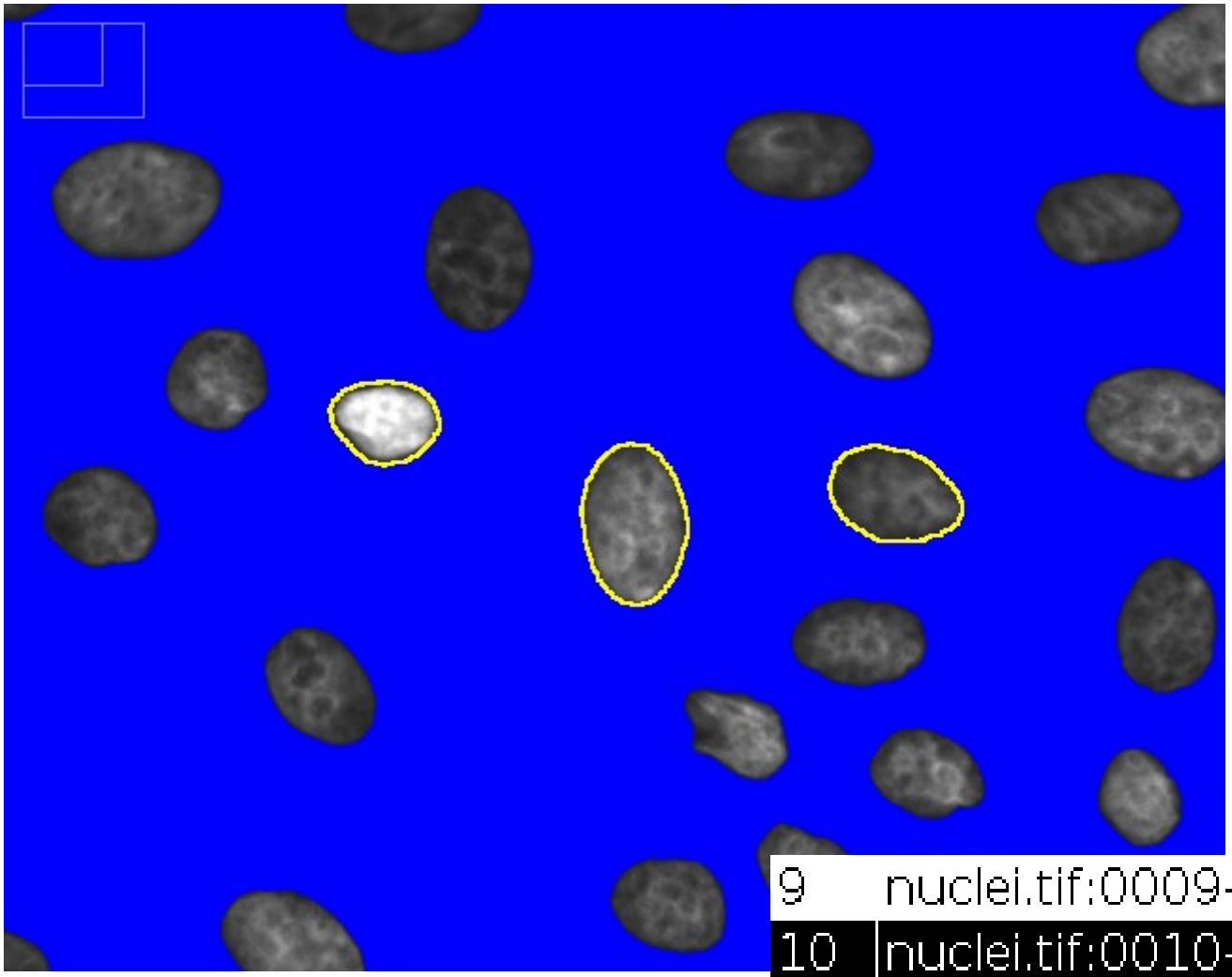
	Label	Area	Mean
1	nuclei.tif:0001-0107	5342	57.268
2	nuclei.tif:0002-0134	6119	47.881
3	nuclei.tif:0003-0139	8048	72.366
4	nuclei.tif:0004-0153	5304	58.734
5	nuclei.tif:0005-0181	6140	48.238
6	nuclei.tif:0006-0221	6556	97.595
7	nuclei.tif:0007-0210	5937	50.747
8	nuclei.tif:0008-0267	4110	70.521
9	nuclei.tif:0009-0298	6745	84.885
10	nuclei.tif:0010-0298	3462	167.340



	Label	Area	Mean
1	nuclei.tif:0000	3462	167.340
2	nuclei.tif:0001	3850	133.805
3	nuclei.tif:0002	6556	97.595
4	nuclei.tif:0003	6552	94.887
5	nuclei.tif:0004	3120	91.873
6	nuclei.tif:0005	6935	88.955
7	nuclei.tif:0006	3450	87.855
8	nuclei.tif:0007	6420	85.566
9	nuclei.tif:0008	6745	84.885
10	nuclei.tif:0009	4273	79.643

Exercise 17

Show rois of selected measurements

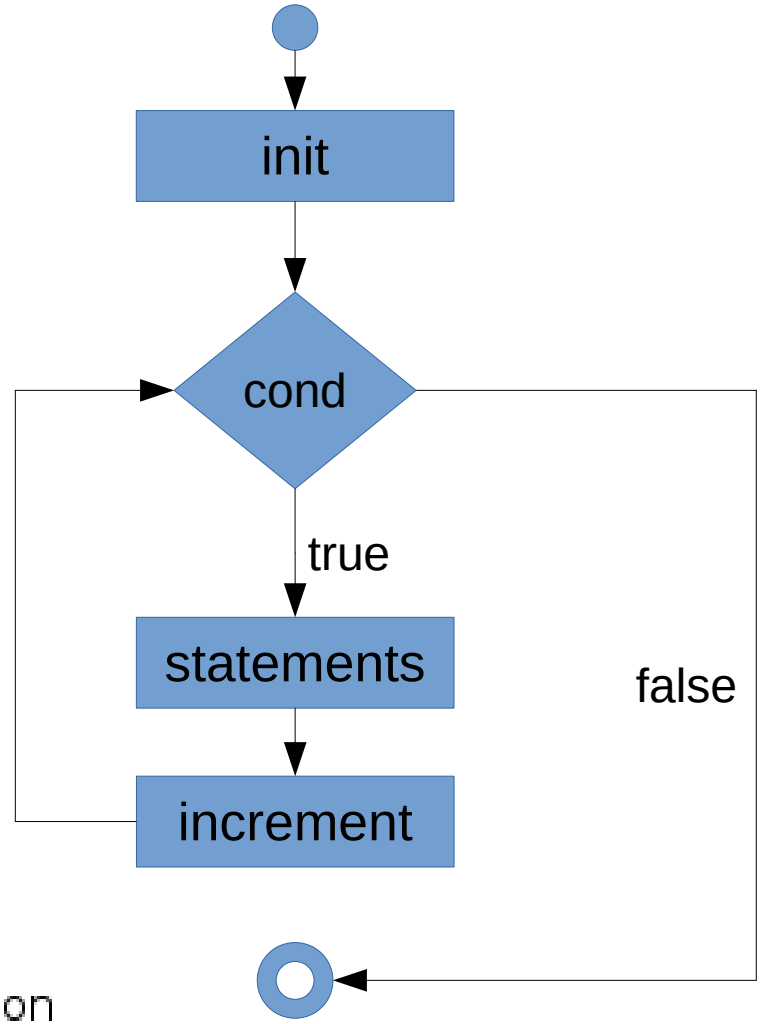


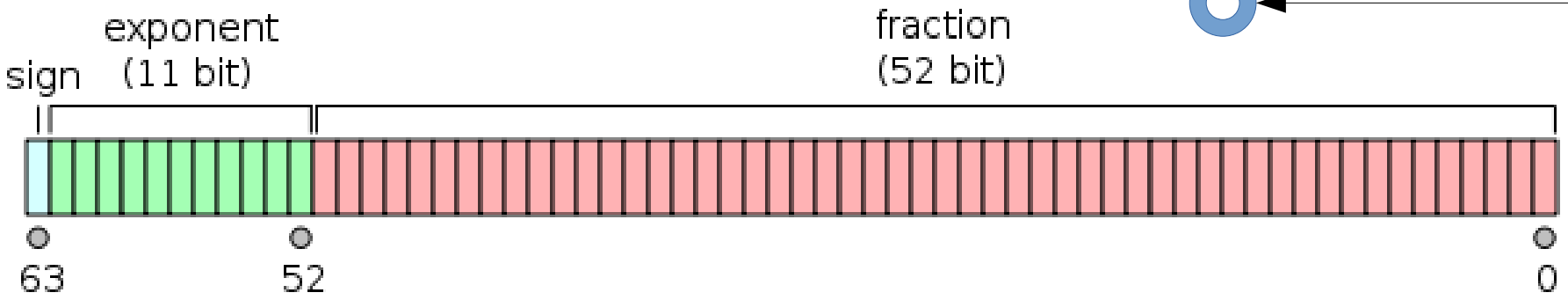
9	nuclei.tif:0009-0298	6745	84.885
10	nuclei.tif:0010-0298	3462	167.340
11	nuclei.tif:0011-0369	6552	94.887
12	nuclei.tif:0012-0347	4777	66.389
13	nuclei.tif:0013-0362	6935	88.955

Exercise 18

The End

```
while(questions) {
  ask();
}
print("Thank you!");
```





Bibliography

[1] Wikimedia. File:IEEE 754 Double Floating Point Format.svg.

[2] ASA standard x3.4-1963. Technical report, June 1963.

[3] IEEE 754-2008. Technical report, August 2008.

[4] Michael R. Berthold, Nicolas Cebron, Fabian Dill, Thomas R. Gabriel, Tobias Ktter, Thorsten Meinl, Peter Ohl, Kilian Thiel, and Bernd Wiswedel. KNIME - the konstanz information miner: version 2.0 and beyond. ACM SIGKDD Explorations Newsletter, 11(1):26, November 2009.

[5] Michael R. Lamprecht, David M. Sabatini, and Anne E. Carpenter. CellProfiler: free, versatile software for automated biological image analysis. BioTechniques, 42(1):71–75, January 2007.

[6] Jérôme Mutterer. Programming with the ImageJ macro language. In ImageJ User and Developer Conference 2010, Luxembourg, 2010. Centre de Recherche Public Henri Tudor.

Bibliography

[7] Seymour Papert. Mindstorms: children, computers, and powerful ideas. Basic Books, New York, 2nd ed edition, 1993.

[8] Johannes Schindelin, Ignacio Arganda-Carreras, Erwin Frise, Verena Kaynig, Mark Longair, Tobias Pietzsch, Stephan Preibisch, Curtis Rueden, Stephan Saalfeld, Benjamin Schmid, Jean-Yves Tinevez, Daniel James White, Volker Hartenstein, Kevin Eliceiri, Pavel Tomancak, and Albert Cardona. Fiji: an open-source platform for biological-image analysis. Nature Methods, 9(7):676–682, June 2012.

[9] Caroline A Schneider, Wayne S Rasband, and Kevin W Eliceiri. NIH image
to ImageJ: 25 years of image analysis. Nature methods, 9(7):671–675, July 2012. PMID: 22930834.

[10] Kenneth Slonneger. Formal syntax and semantics of programming languages: a laboratory based approach. Addison-Wesley Pub. Co, Reading, Mass, 1995.

[11] Wikipedia. Double-precision floating-point format — wikipedia, the free encyclopedia, 2014. [Online; accessed 28-July-2014].

[12] Wikipedia. Ieee 754-1985 — wikipedia, the free encyclopedia, 2014. [Online; accessed 28-July-2014].