

# Bio-image analysis, bio-statistics, programming and machine learning for computational biology

Anna Poetsch, Marcelo Zoccoler, Johannes Müller, Robert Haase

# The team

---



Anna Poetsch, Marcelo Zoccoler, Johannes Müller, Robert Haase

# Lecture overview: Programming

- We will focus on Python programming.
- “Python is the most wanted language for the third year in a row, meaning that developers who do not yet use it say they want to learn it.”
- Goal: **Allow you to do things automatically instead of suffering long time when doing it manually.**

```
b = 3  
c = a + b
```

```
print(c)
```

```
8
```

```
d = 6  
e = 7  
f = a * d  
g = f / e  
h = 1 + g
```

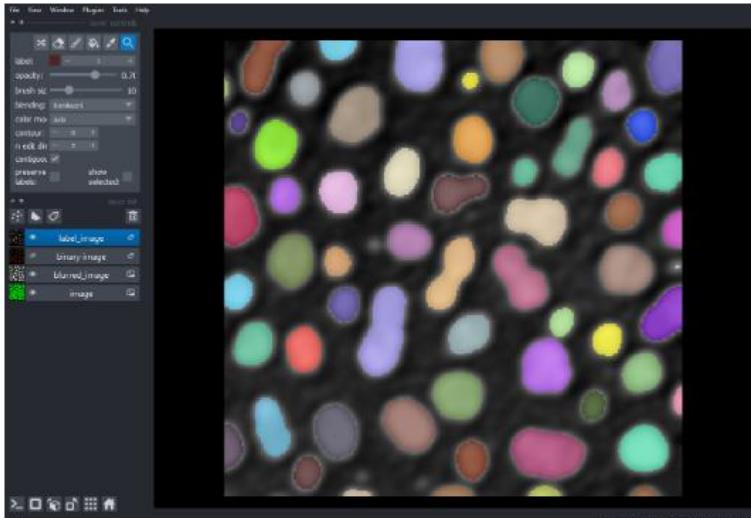
```
print(h)
```

```
5.285714285714286
```

```
from skimage.measure import label  
label_image = label(binary_image)  
  
# add labels to viewer  
label_layer = viewer.add_labels(label_image)
```

You can visualize labelled objects as overlay (per default)

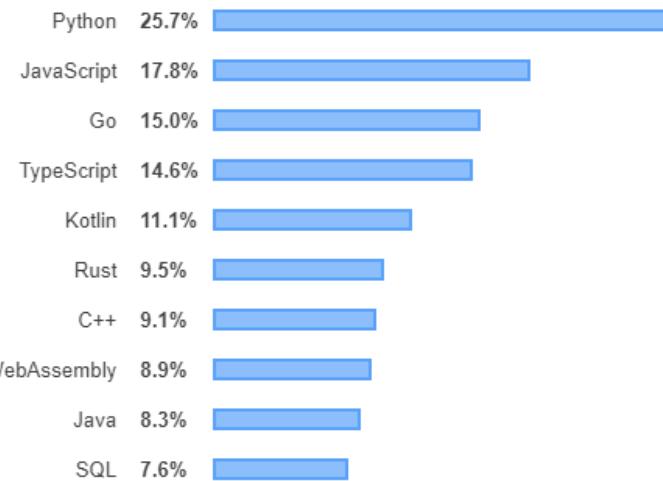
```
napari.utils.nbscreenshot(viewer)
```



## Most Loved, Dreaded, and Wanted

### Most Loved, Dreaded, and Wanted Languages

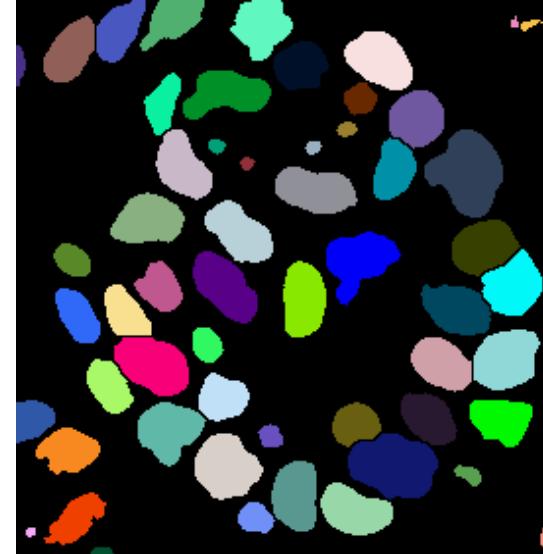
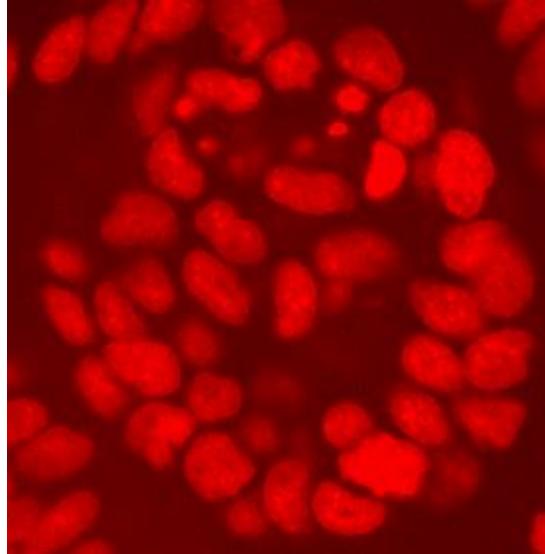
Loved    Dreaded    **Wanted**



Source: <https://insights.stackoverflow.com/survey/2019#technology>

# Lecture overview: Bio-image analysis

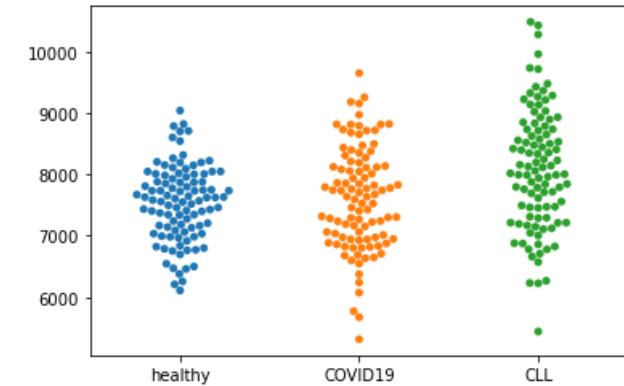
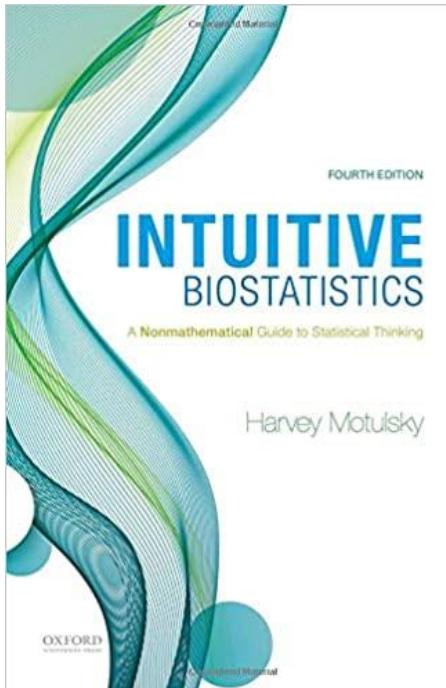
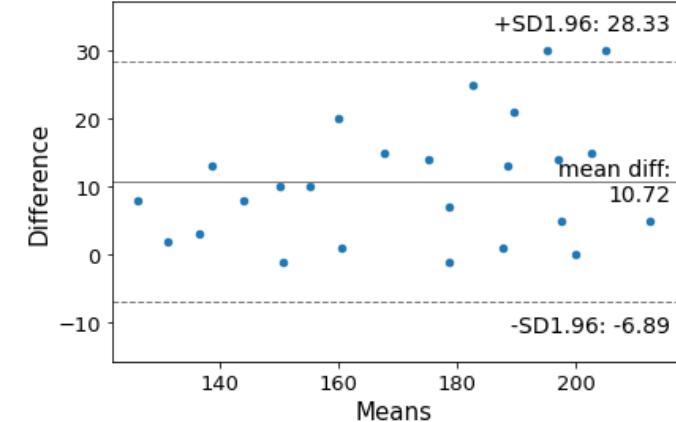
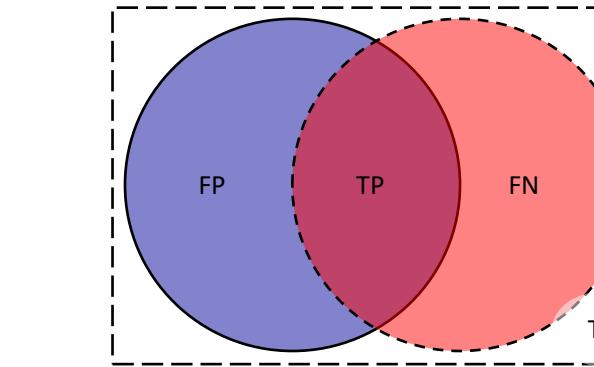
- Quantitative bio-image analysis
  - Transformations, Projections and Filtering
  - Image segmentation
  - Tracking objects
  - Measuring in images
- Goal: Allow you to quantify observations.
- Software
  - Python for automation
  - Napari for interactive image analysis



	label	maximum	mean	median	minimum	sigma
0	1	224.0	137.526132	136.0	112.0	13.360739
1	2	232.0	193.014354	200.0	128.0	28.559077
2	3	224.0	179.846995	184.0	128.0	21.328889
3	4	248.0	207.082171	216.0	120.0	27.772832
4	5	248.0	223.146402	232.0	128.0	30.246515
5	6	248.0	214.906725	224.0	128.0	26.386796
6	7	248.0	211.565891	224.0	136.0	30.197236
7	8	200.0	166.171429	168.0	136.0	16.466894
8	9	224.0	176.932331	176.0	128.0	24.022064
9	10	240.0	191.598174	200.0	128.0	28.239851

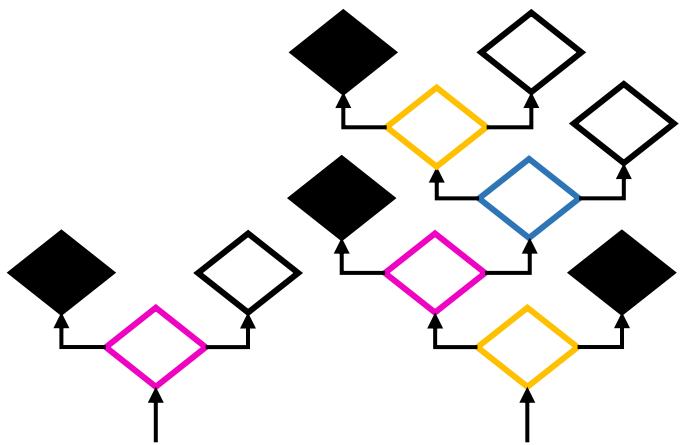
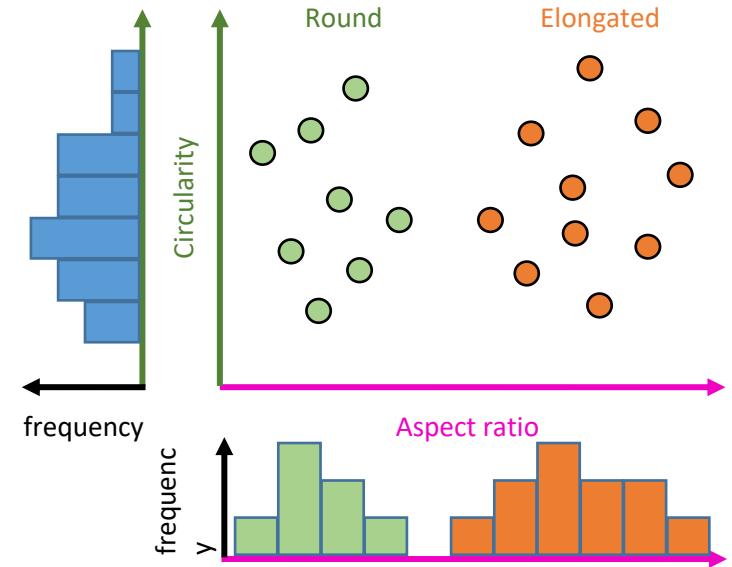
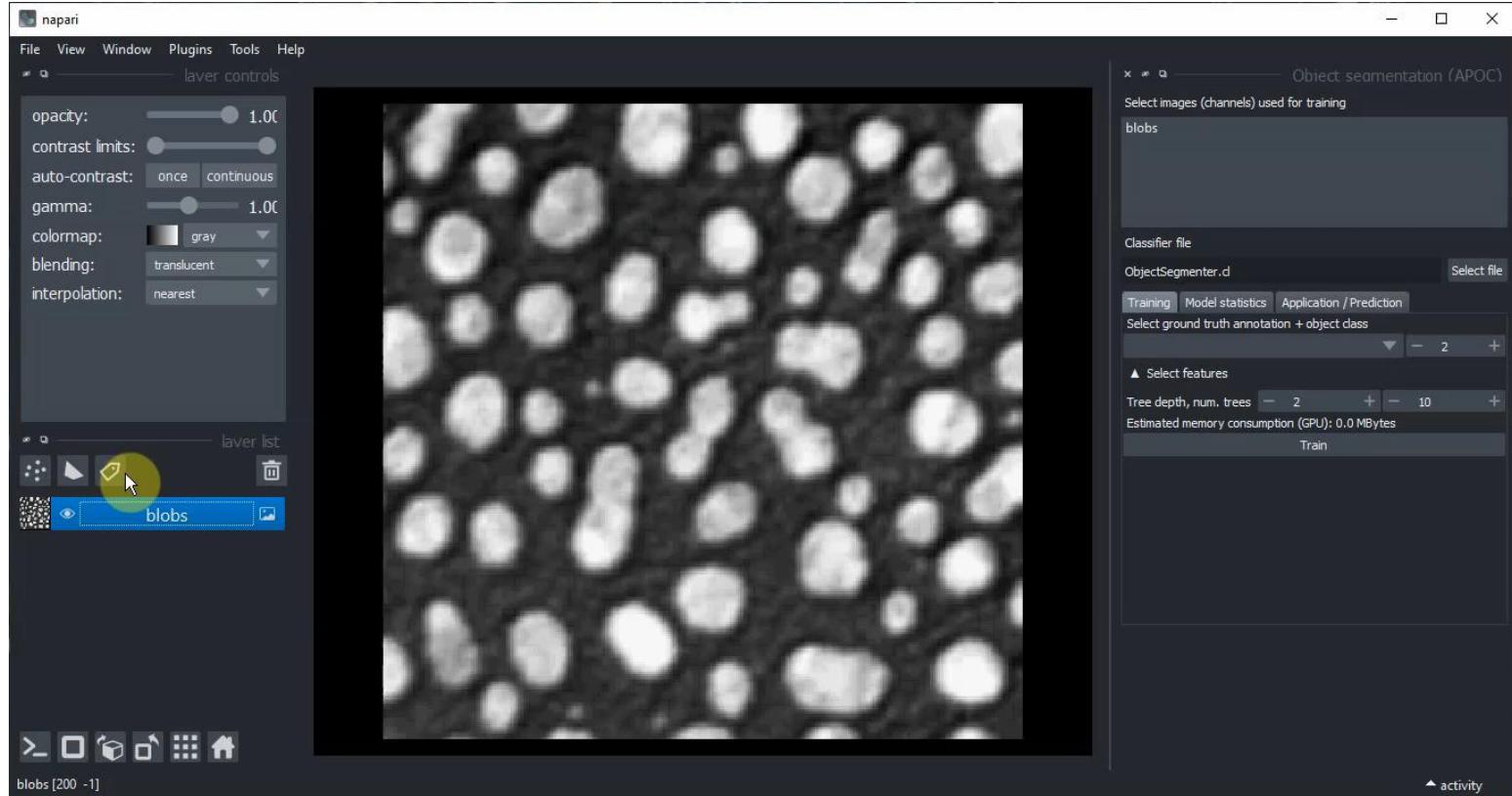
# Lecture overview: Bio-statistics

- Descriptive statistics
    - Distributions
    - Method comparison: Bland-Altman analysis
  - Inferential statistics
    - hypothesis testing
  - Multiple comparisons and correlations
  - Big data, clustering, dimensionality reduction
- 
- **Goal:** Allow you to draw conclusions from quantified observations.



# Lecture overview: Machine learning

- Machine learning
  - in the context of image analysis
- Computers can *learn* where, what objects in images are...
- Goal: **Give you an insight into state-of-the-art methods**



Random forest classifiers

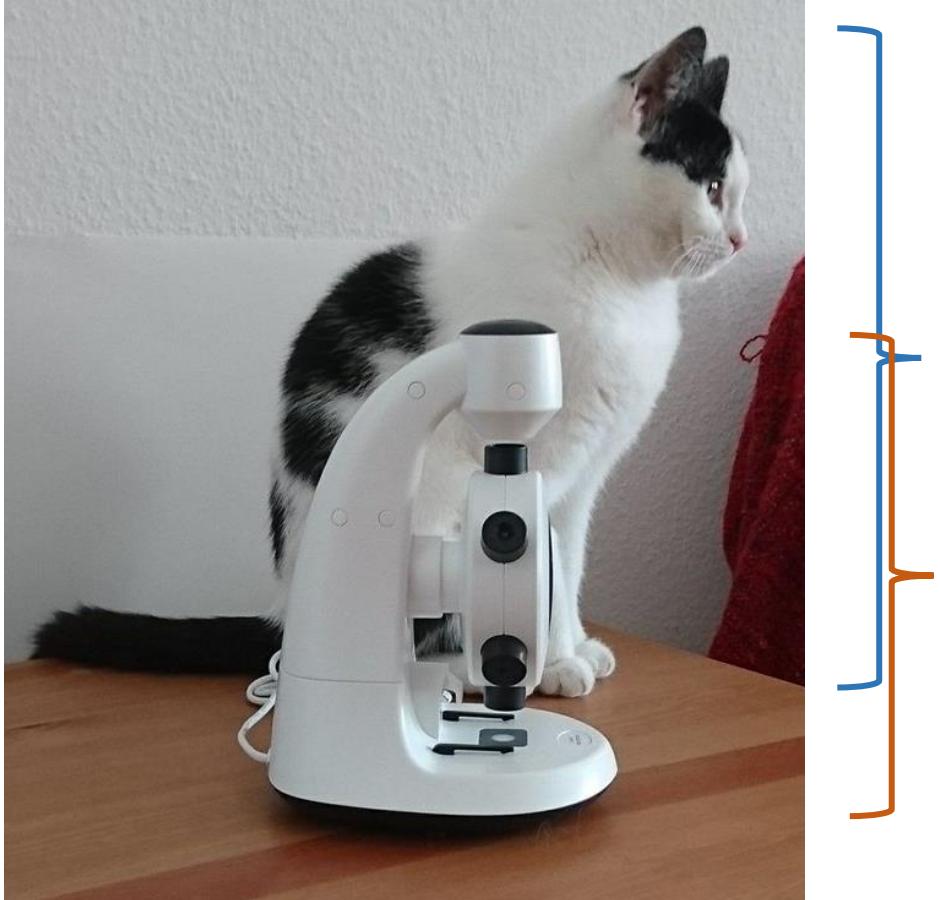
- 5.4.2022 - General introduction, setting up Conda and Python
- 12.4.2022 - Introduction to python programming, data structures and image data
- 19.4.2022 - Introduction to python algorithms: loops, conditions, functions, image processing
- 26.4.2022 - Image filtering + segmentation
- 3.5.2022 - Quantitative image analysis / feature extraction
- 10.5.2022 - Machine learning for bio-image analysis
- 17.5.2022 - Introduction to Biostatistics
- 24.5.2022 - Descriptive statistics
- 31.5.2022 - Method Comparison - Bland-Altman analysis
- 14.6.2022 - Hypothesis testing
- 21.6.2022 - Multiple comparisons and correlations
- 28.6.2022 - Big data, clustering, dimensionality reduction
- 5.7.2022 - Deep Learning
- 12.7.2022 – Summary, exam preparation

- Every week will follow the same rough scheme
  - 13:00 - 13:30: Discussion about exercise last week
  - 13:30 : 1 h lecture
  - < 1h demonstration
  - < 1h homework
- Questions can be asked in the zoom meetings any time by speaking up. Or you post them anonymously: (see link to etherpad)
- Exam will cover the semester content accordingly
  - Theory of
    - image analysis,
    - statistics and
    - machine learning
  - Basics of programming
    - write simple < 10 line programs and
    - read code and describe what it does

# Introduction to Bio-Image Analysis

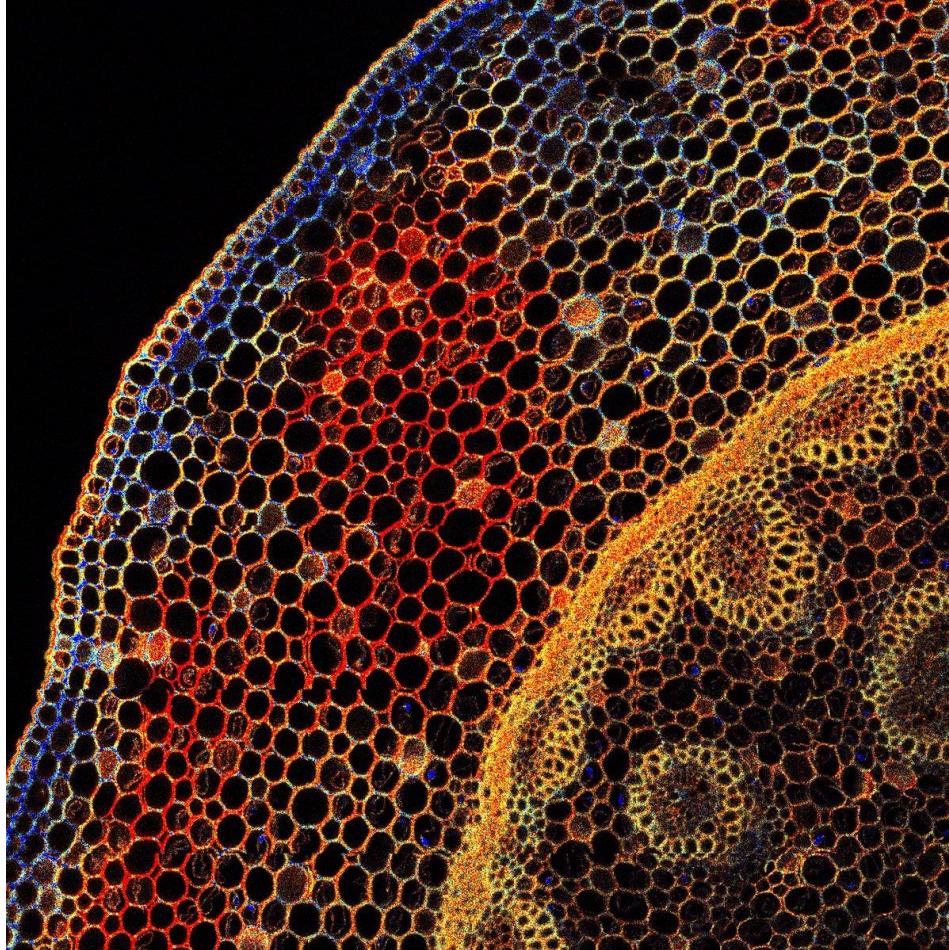
Robert Haase

- Deriving quantitative information from images of biological samples taken with microscopes



cat height = 1.5 x microscope height

- Deriving quantitative information from images of biological samples taken with microscopes



How many cells are there in the outer ring?

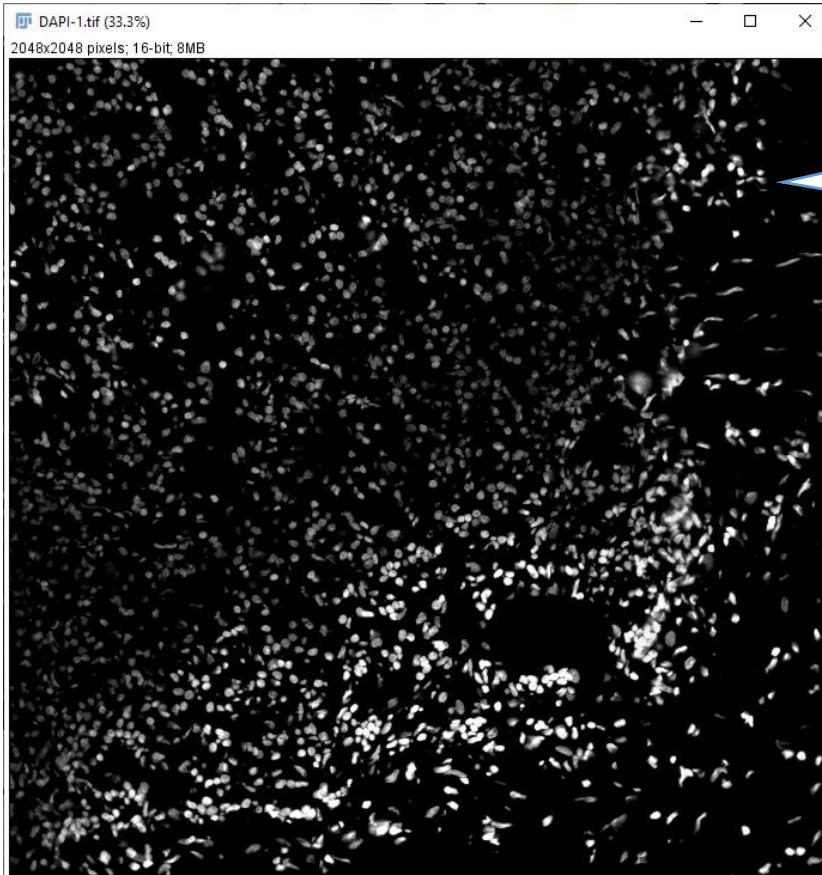
How high is the intensity in the inner ring?

How is the signal in the blue channel distributed?

Image data source: Lorenzo Scipioni, U. California / Irvine

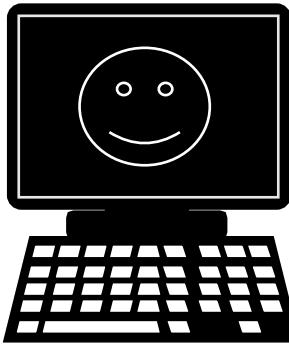
# Objective bio-image analysis

- Measurements should be objective, not influenced by human interpretation



Nuclei in this image are ...  
... more dense than in this image.

Use automation for less subjective analysis.



April 2022

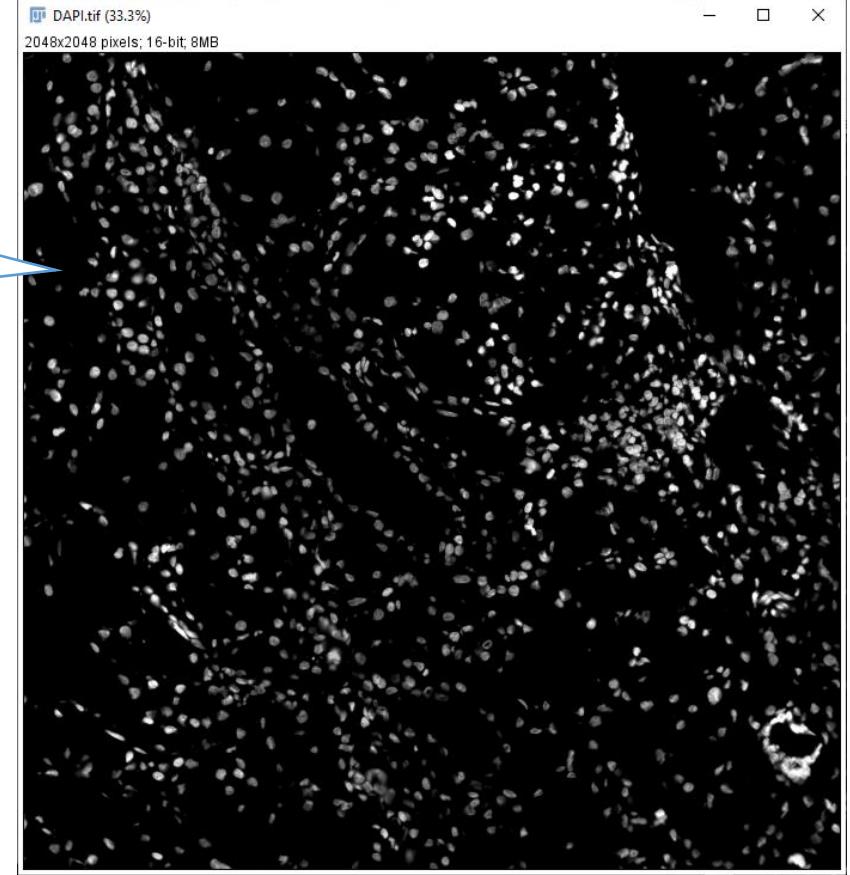
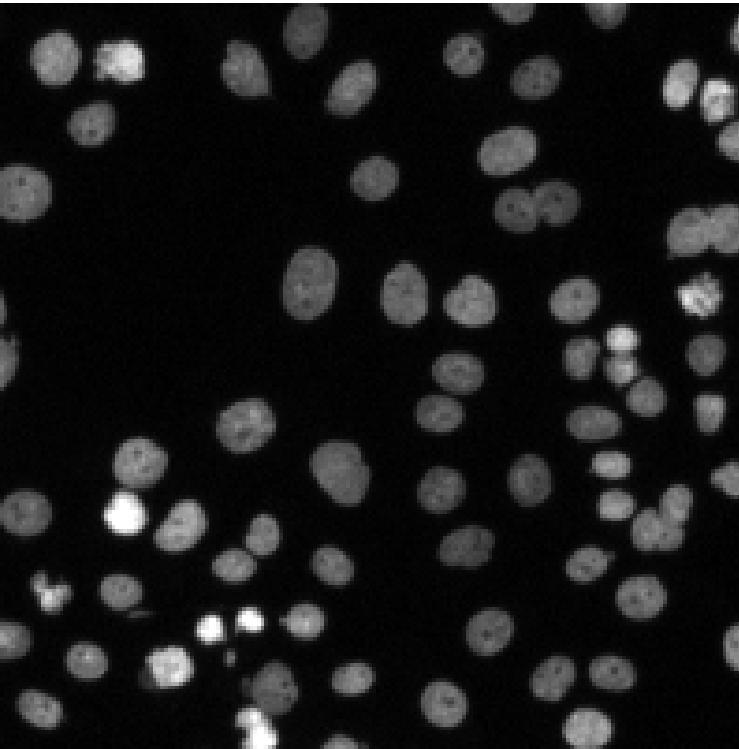


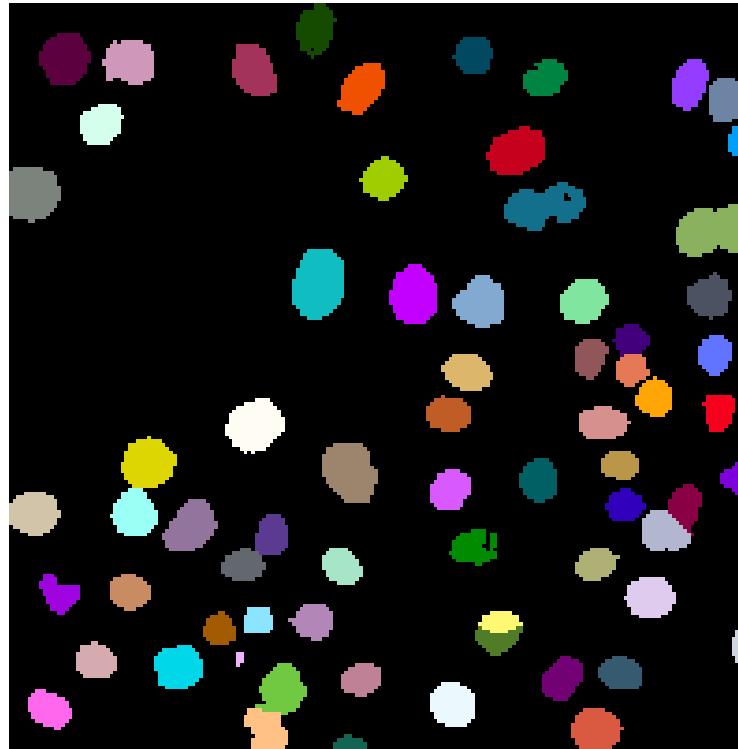
Image data source: Pascual-Reguant, Anna. (2021). Immunofluorescence staining of a human kidney (#2, peri-tumor area) obtained by MELC [Data set]. Zenodo. <http://doi.org/10.5281/zenodo.4434462> licensed CC-BY 4.0

- Algorithms must be reliable (trustworthy). Visualization helps gaining trust in automated methods.

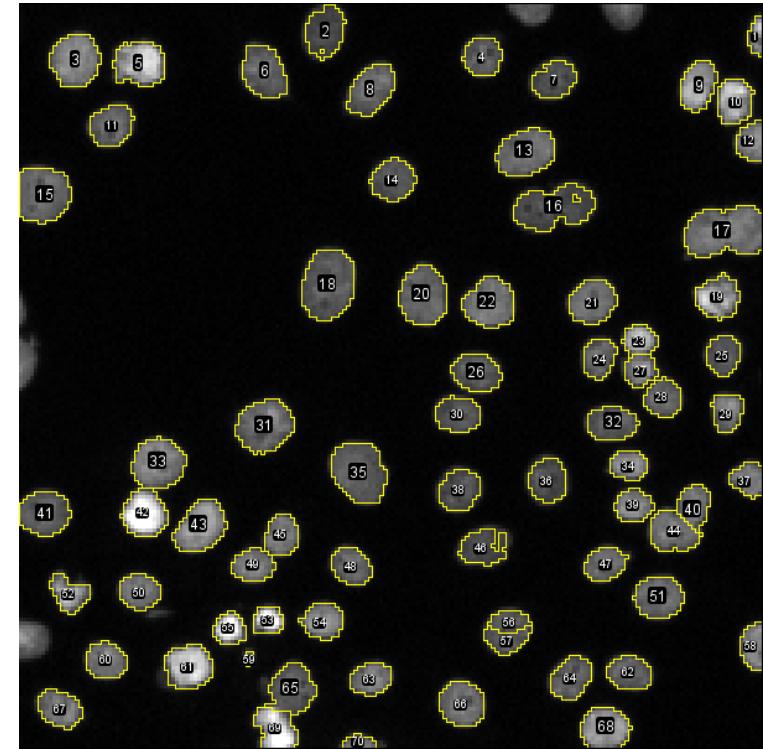
Original image



Label image



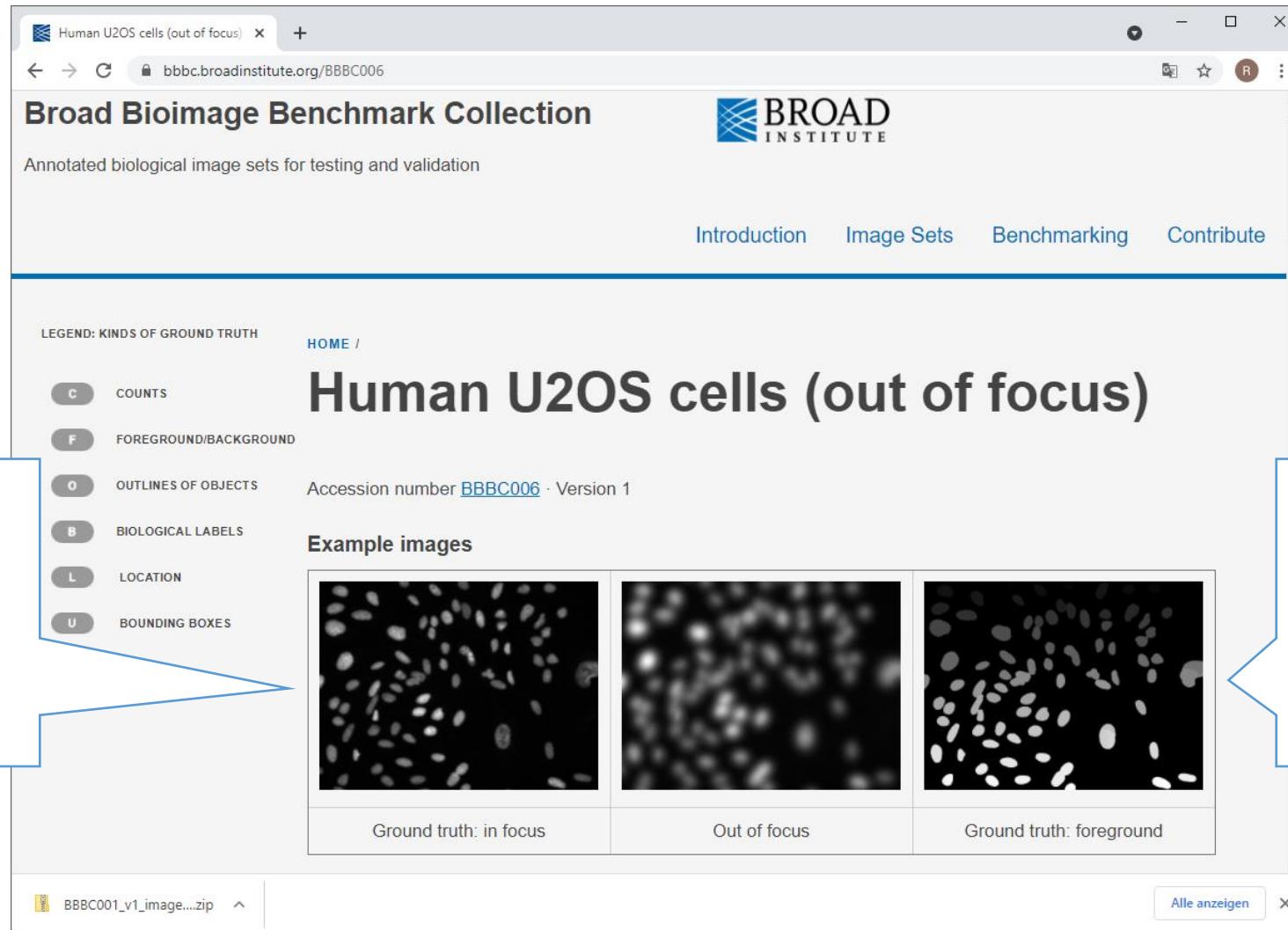
Overlay



There are 70 nuclei  
in this image.

# Reliable bio-image analysis

- Algorithms must be reliable (validated methods). Publicly available benchmark data sets allow to compare algorithms on common data.



# Reproducible bio-image analysis

- “The image data was analyzed with ImageJ.”

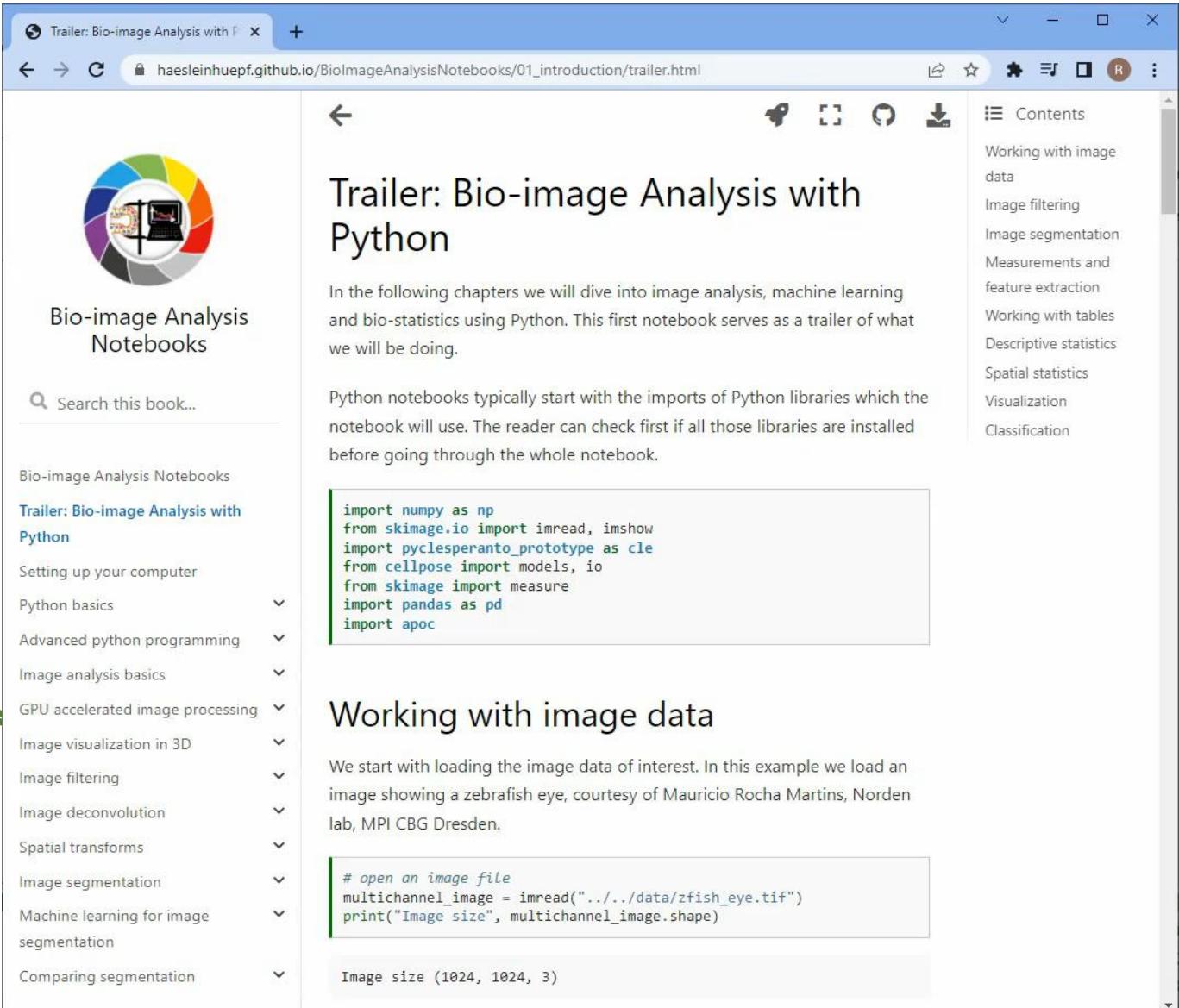
Can you reproduce  
what they did?

# Reproducible bio-image analysis

- “The image data was analyzed with ImageJ.”

Can you reproduce what they did?

Can you reproduce what they did?



The screenshot shows a web browser window displaying a trailer notebook for "Bio-image Analysis with Python". The URL in the address bar is [haesleinhuepf.github.io/BioImageAnalysisNotebooks/01\\_introduction/trailer.html](https://haesleinhuepf.github.io/BioImageAnalysisNotebooks/01_introduction/trailer.html). The page title is "Trailer: Bio-image Analysis with Python". On the left, there is a sidebar with a circular logo and a search bar. Below the search bar is a list of topics: Bio-image Analysis Notebooks, Trailer: Bio-image Analysis with Python, Setting up your computer, Python basics, Advanced python programming, Image analysis basics, GPU accelerated image processing, Image visualization in 3D, Image filtering, Image deconvolution, Spatial transforms, Image segmentation, Machine learning for image segmentation, and Comparing segmentation. The main content area starts with a brief introduction: "In the following chapters we will dive into image analysis, machine learning and bio-statistics using Python. This first notebook serves as a trailer of what we will be doing." It then provides instructions for setting up the environment and lists the required Python libraries: numpy, skimage.io, pyclesperanto\_prototype, cellpose, skimage.measure, pandas, and apoc. A code block shows the import statements:

```
import numpy as np
from skimage.io import imread, imshow
import pyclesperanto_prototype as cle
from cellpose import models, io
from skimage import measure
import pandas as pd
import apoc
```

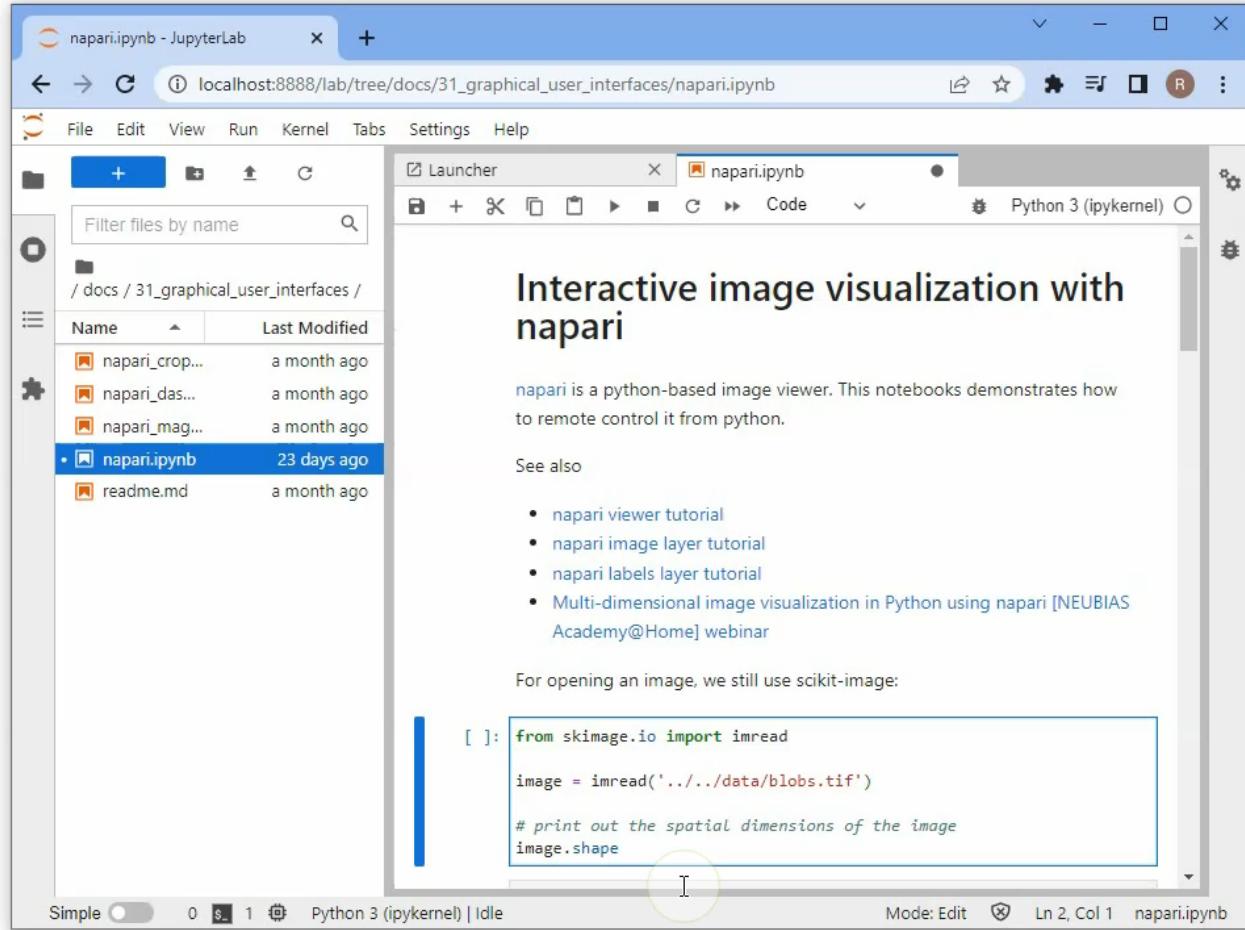
Below this, a section titled "Working with image data" begins with the text: "We start with loading the image data of interest. In this example we load an image showing a zebrafish eye, courtesy of Mauricio Rocha Martins, Norden lab, MPI CBG Dresden." A code block shows the command to load the image:

```
# open an image file
multichannel_image = imread("../data/zfish_eye.tif")
print("Image size", multichannel_image.shape)
```

The output of the print statement is "Image size (1024, 1024, 3)".

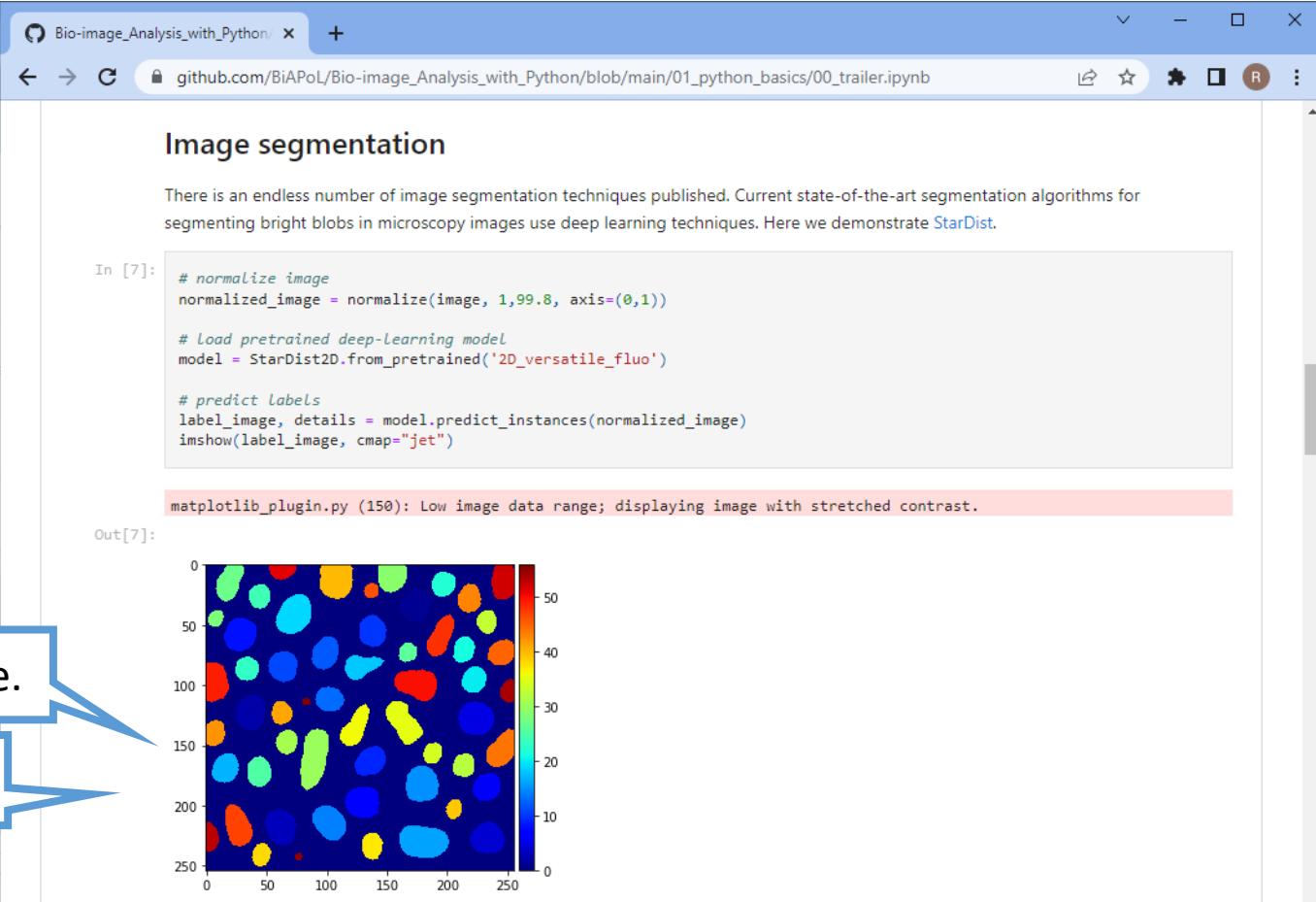
# Interactivity versus reproducibility

- Remote controlling graphical user interfaces



- Compared to wet-lab experiments, image analysis (in-silico) experiments are typically repeatable.
- In wet-lab experiments, samples may get destroyed while executing the experiment.
- Repeatability is a property of the experiment. You cannot improve repeatability by better documentation.

- However, you need to pay some extra attention to have repeatable image processing
  - Scripts need to be written in a way that their execution is repeatable.
  - Test it!



The screenshot shows a Jupyter Notebook cell titled "Image segmentation". The cell contains Python code for image segmentation using the StarDist2D model. The code includes normalization of the image, loading a pretrained model, and predicting labels. A warning message from matplotlib is displayed: "matplotlib\_plugin.py (150): Low image data range; displaying image with stretched contrast." Below the code, a heatmap visualizes the segmented blobs in various colors (red, orange, yellow, green, blue) against a dark background. A color bar on the right indicates intensity levels from 0 to 50. Three callout boxes with arrows point from the text below to the code and the resulting image:

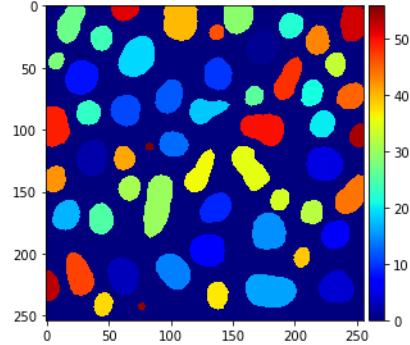
- "Run it once." points to the "# predict labels" line in the code.
- "Run it twice." points to the "imshow(label\_image, cmap="jet")" line in the code.
- "Are results identical?" points to the heatmap.

```
# normalize image
normalized_image = normalize(image, 1, 99.8, axis=(0,1))

# load pretrained deep-learning model
model = StarDist2D.from_pretrained('2D_versatile_fluo')

# predict labels
label_image, details = model.predict_instances(normalized_image)
imshow(label_image, cmap="jet")

matplotlib_plugin.py (150): Low image data range; displaying image with stretched contrast.
```



Out[7]:

# Introduction to bio-image analysis

- Bio-image analysis is supposed to be
  - **Quantitative**
    - We derive numbers from images which describe physical properties of the observed sample.
  - **Objective**
    - The derived measurement does not depend on who did the measurement. The measurement is free of interpretation.
  - **Reliable (trustworthy / validated)**
    - We are confident that the measurement is describing what it is supposed to describe.
  - **Reproducible**
    - Somebody else can do the experiment under *different conditions* and gets similar measurements. For this, documentation is decisive!
  - **Repeatable**
    - We can do the same experiment twice under the *same conditions* and get similar measurements.

# Image analysis is part of the experiment



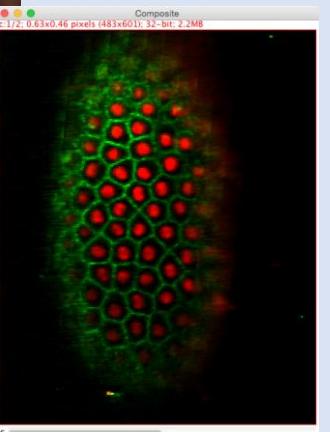
**PoL**  
Physics of Life  
TU Dresden



Observation

$$p_{ij}(t) = \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{j=1}^n [\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}$$

Modeling



Imaging

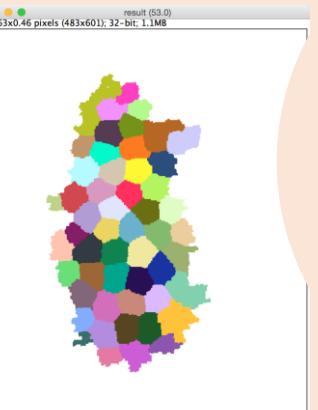


Image processing

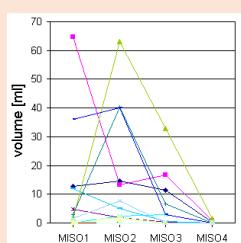
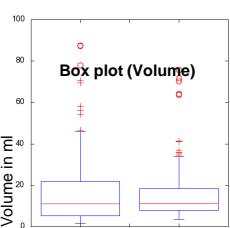


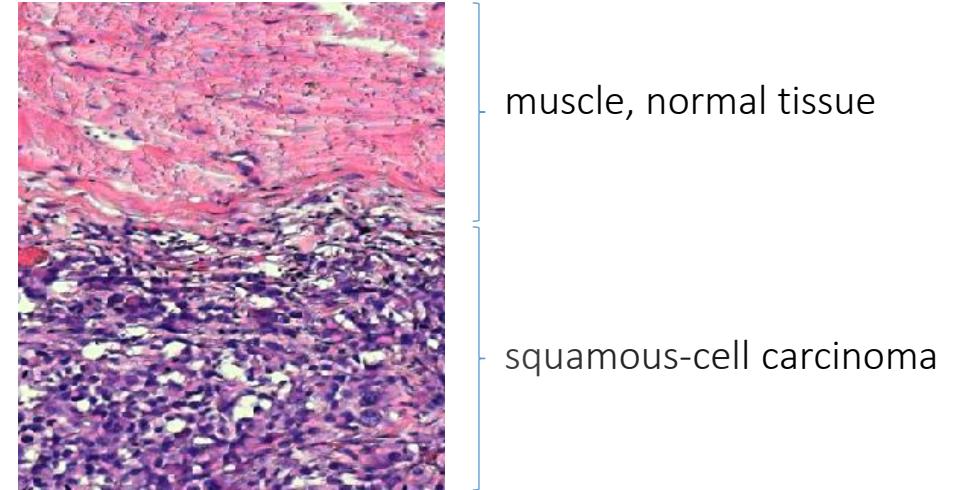
Image analysis  
Bio-statistics



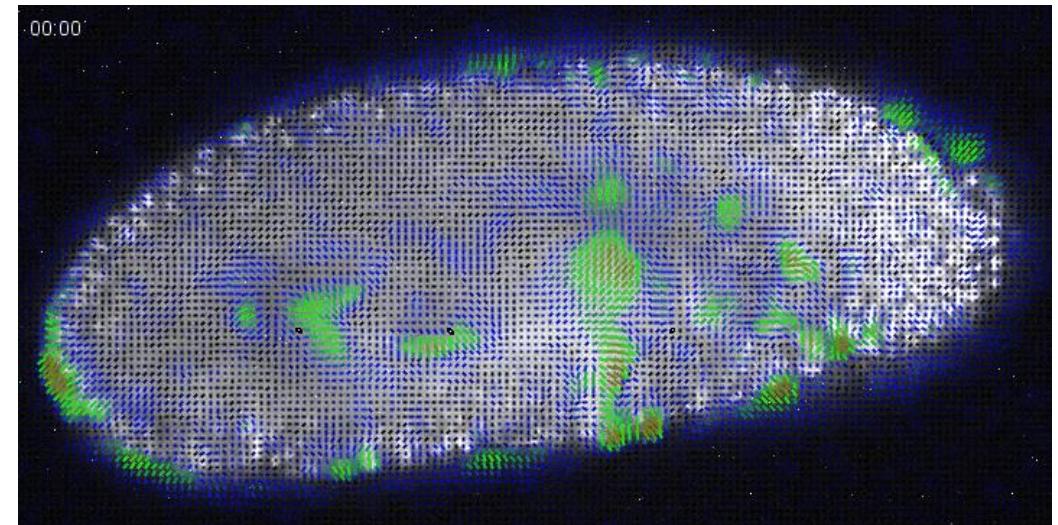
April 2022

# Common questions

- Typical questions bio-image analysts deal with
  - Is signal intensity different under varying conditions?
  - How many cells are in my image?
  - How high is cell density?
    - Bio-statistics / medicine
  - How are different tissues characterized?
    - Machine learning



- Typical questions bio-image analysts struggle with
  - What force drives the observed processes?
  - What is the lineage tree of one particular cell?
  - Are observation A and observation B related?
  - Are structures observed in different color channels colocalized?



# Image analysis is part of the experiment

- Think about how to analyze your images before starting the experiment.
  - Consider adapting your experiment so that quantitative image analysis can be performed easily.
- Think about controls, counter-proves, an easy to falsify null-hypothesis.
  - Be a lazy scientist. Do simple experiments.
- How can you exclude yourself from the experiment?
  - Think of blinding yourself or fully automate analysis.
- One experiment usually answers just one or less questions.

# Image analysis is part of the experiment

Talk to image-analysis / data-science experts early during your project.

Not one week before the  
submission deadline.

# The command line

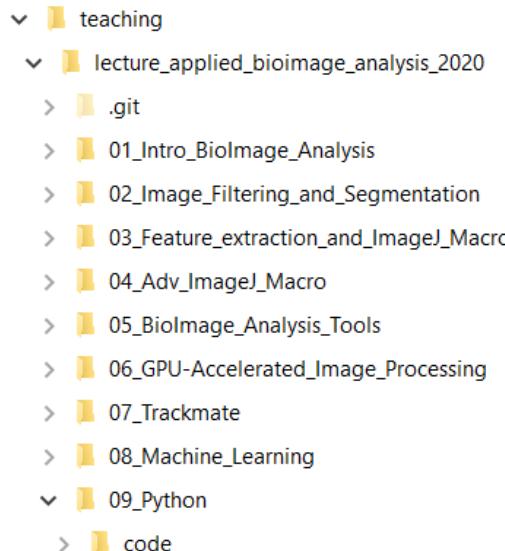
Robert Haase

April 2022

# The command line

- A.k.a. the Terminal or Eingabeaufforderung: Welcome to the 20<sup>th</sup> century!

- The `dir` command tells you what's in the current directory
- On Mac and Linux the command is called `ls -l`



```
C:\structure\teaching\lecture_applied_bioimage_analysis_2020>dir
```

```
C:\structure\teaching\lecture_applied_bioimage_analysis_2020>dir
Volume in drive C has no label.
Volume Serial Number is CE24-80E9

Directory of C:\structure\teaching\lecture_applied_bioimage_analysis_2020

05/25/2020  10:01 AM    <DIR>        .
05/25/2020  10:01 AM    <DIR>        ..
05/14/2020  08:38 AM    <DIR>        01_Intro_BioImage_Analysis
05/14/2020  08:38 AM    <DIR>        02_Image_Filtering_and_Segmentation
05/14/2020  08:38 AM    <DIR>        03_Feature_extraction_and_ImageJ_Macro
05/14/2020  08:38 AM    <DIR>        04_Adv_ImageJ_Macro
05/14/2020  08:38 AM    <DIR>        05_BioImage_Analysis_Tools
05/14/2020  08:38 AM    <DIR>        06_GPU-Accelerated_Image_Processing
05/28/2020  09:24 AM    <DIR>        07_Trackmate
05/25/2020  10:04 AM    <DIR>        08_Machine_Learning
05/28/2020  09:24 AM    <DIR>        09_Python
                  0 File(s)   811,171,487,744 bytes free
                  11 Dir(s)
```

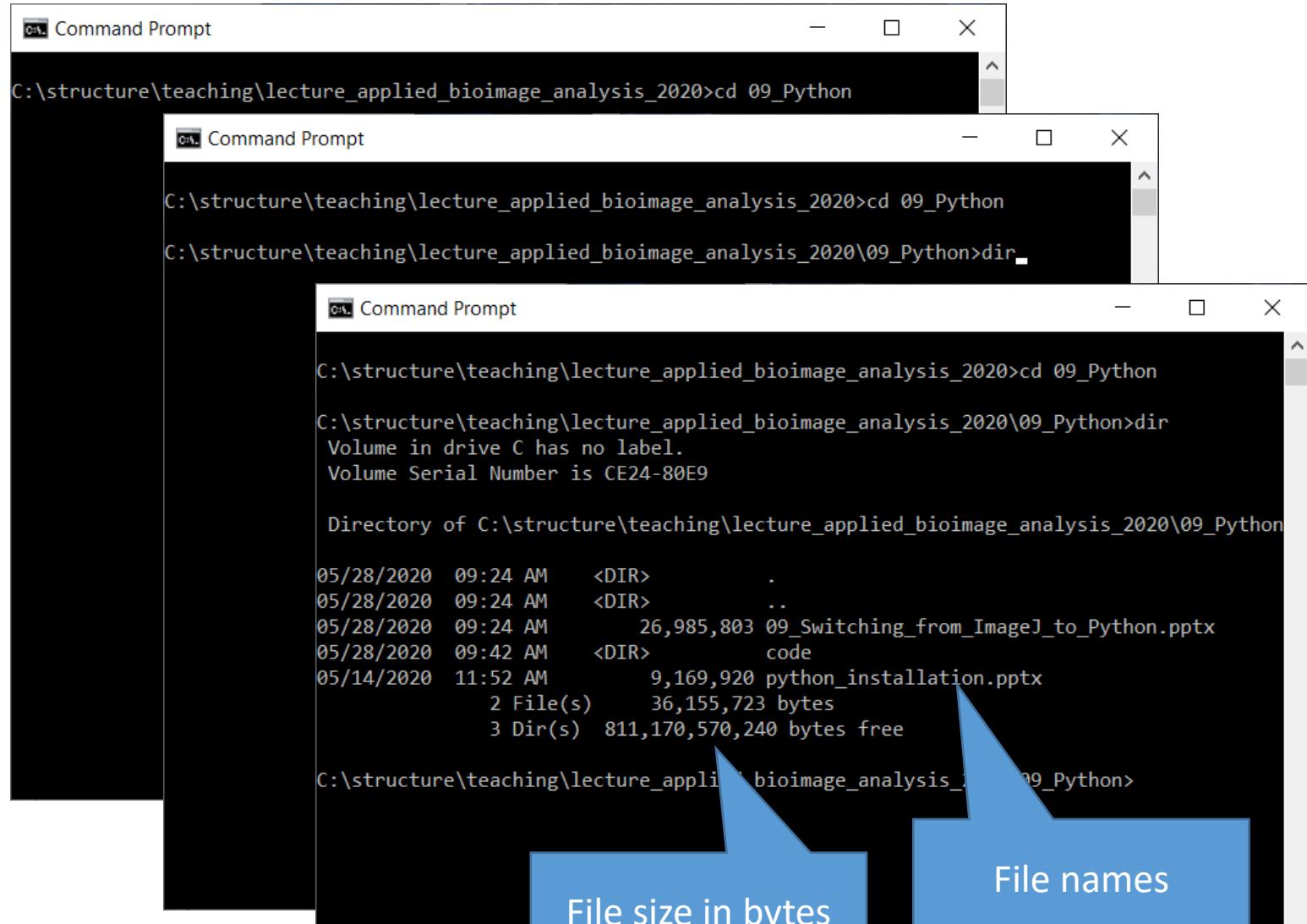
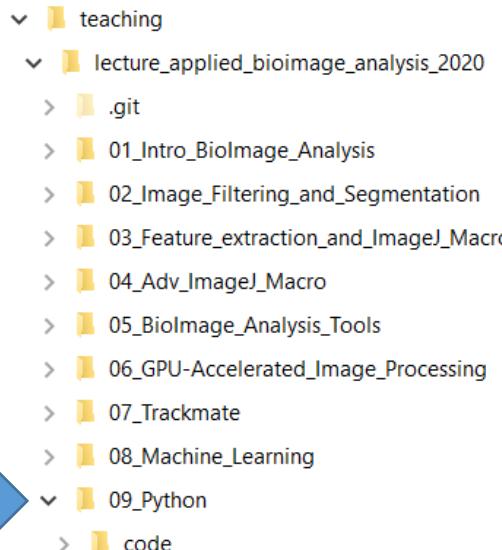
**Types (directories)**

**Names of files and subdirectories**

# The command line

- A.k.a. the Terminal or Eingabeaufforderung: Welcome to the 20<sup>th</sup> century!

- The `cd` command let's you move between different directories.
- With `cd <pathname>` you go into a sub-directory



```
C:\structure\teaching\lecture_applied_bioimage_analysis_2020>cd 09_Python
C:\structure\teaching\lecture_applied_bioimage_analysis_2020>cd 09_Python
C:\structure\teaching\lecture_applied_bioimage_analysis_2020\09_Python>dir
C:\structure\teaching\lecture_applied_bioimage_analysis_2020>cd 09_Python
C:\structure\teaching\lecture_applied_bioimage_analysis_2020\09_Python>dir
Volume in drive C has no label.
Volume Serial Number is CE24-80E9

Directory of C:\structure\teaching\lecture_applied_bioimage_analysis_2020\09_Python

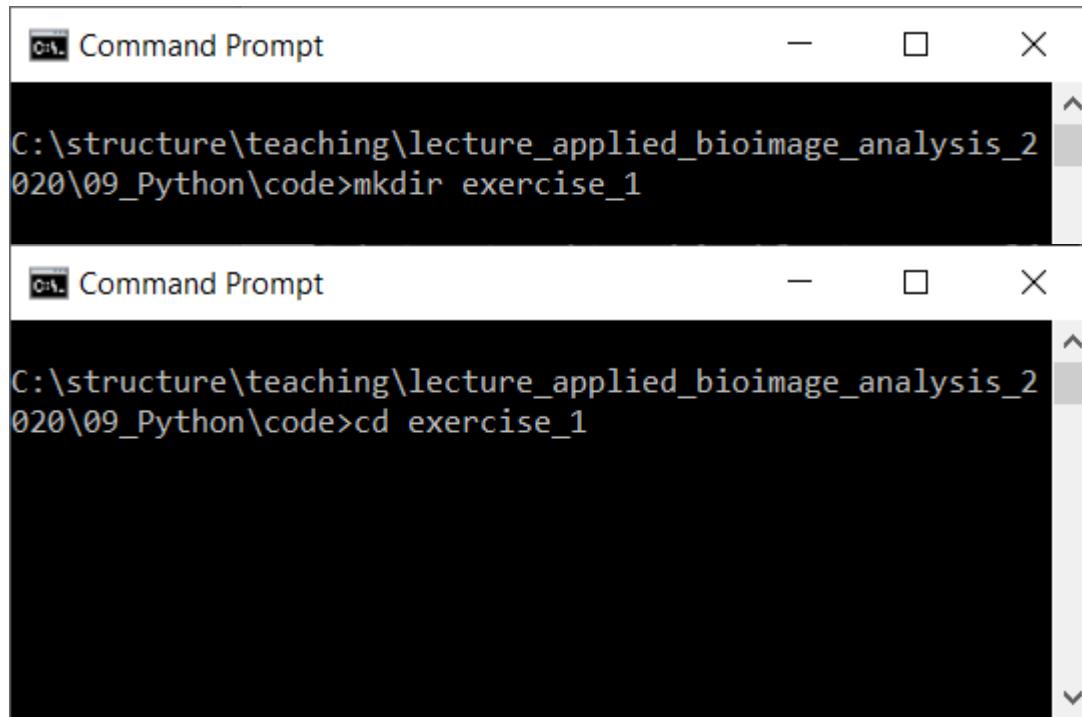
05/28/2020  09:24 AM    <DIR>          .
05/28/2020  09:24 AM    <DIR>          ..
05/28/2020  09:24 AM        26,985,803 09_Switching_from_ImageJ_to_Python.pptx
05/28/2020  09:42 AM    <DIR>          code
05/14/2020  11:52 AM        9,169,920 python_installation.pptx
                           2 File(s)     36,155,723 bytes
                           3 Dir(s)   811,170,570,240 bytes free
```

File size in bytes

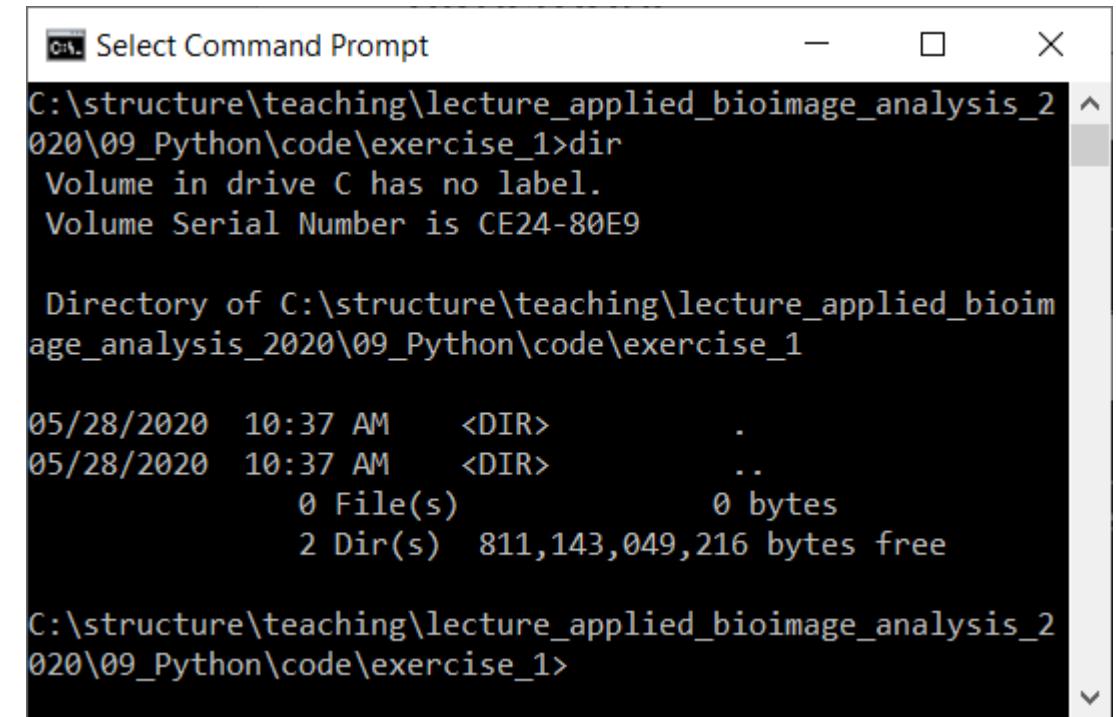
File names

# The command line

- A.k.a. the Terminal or Eingabeaufforderung: Welcome to the 20<sup>th</sup> century!
- The `mkdir` command creates new directories.



```
C:\structure\teaching\lecture_applied_bioimage_analysis_2  
020\09_Python\code>mkdir exercise_1  
  
C:\structure\teaching\lecture_applied_bioimage_analysis_2  
020\09_Python\code>cd exercise_1
```



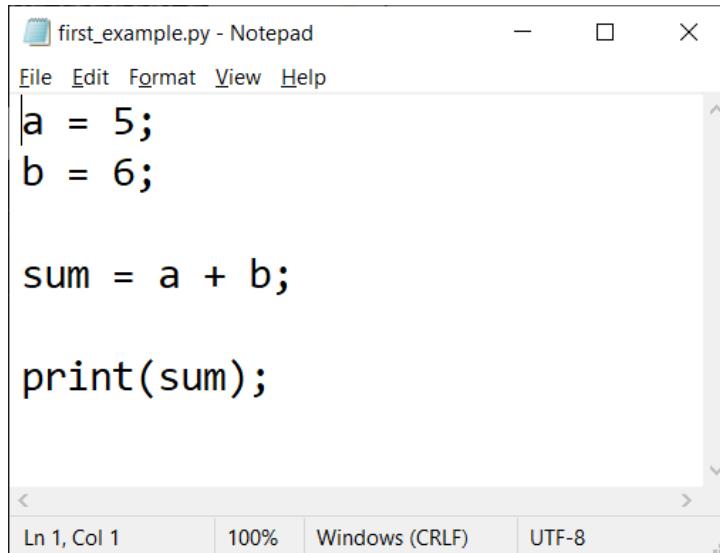
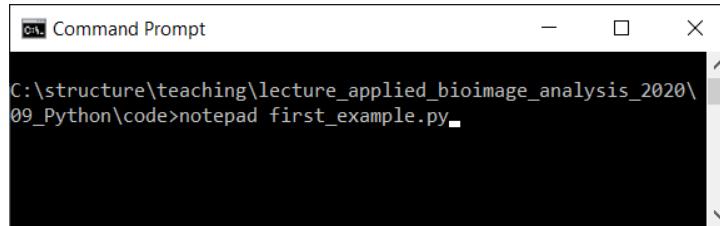
```
C:\structure\teaching\lecture_applied_bioimage_analysis_2  
020\09_Python\code\exercise_1>dir  
Volume in drive C has no label.  
Volume Serial Number is CE24-80E9  
  
Directory of C:\structure\teaching\lecture_applied_bioim  
age_analysis_2020\09_Python\code\exercise_1  
  
05/28/2020 10:37 AM <DIR> .  
05/28/2020 10:37 AM <DIR> ..  
 0 File(s) 0 bytes  
 2 Dir(s) 811,143,049,216 bytes free  
  
C:\structure\teaching\lecture_applied_bioimage_analysis_2  
020\09_Python\code\exercise_1>
```

# The command line

- Windows specific

- Notepad text editor

```
notepad <filename>
```

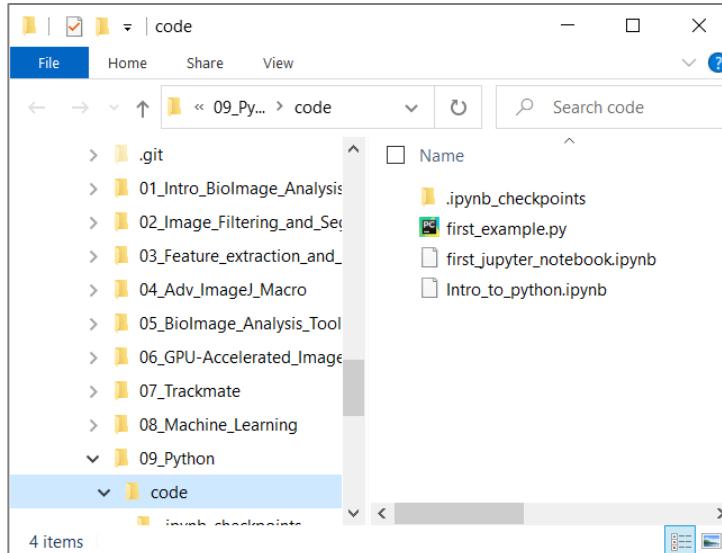
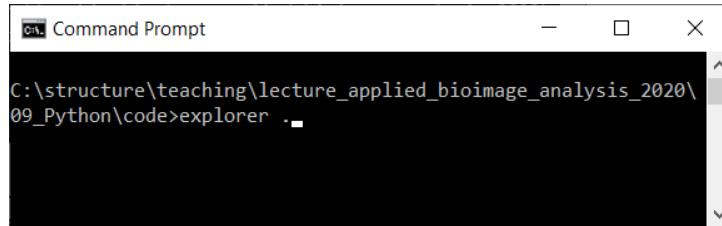


A screenshot of a Notepad application window. The title bar says "first\_example.py - Notepad". The menu bar includes File, Edit, Format, View, Help. The code in the text area is:

```
a = 5;  
b = 6;  
  
sum = a + b;  
  
print(sum);
```

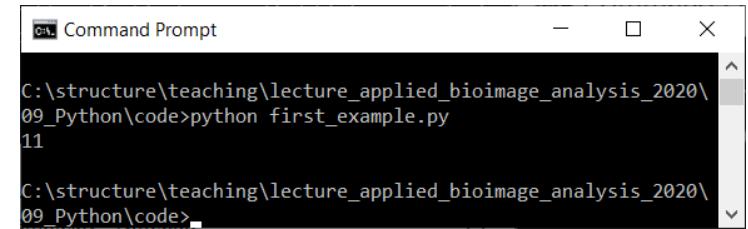
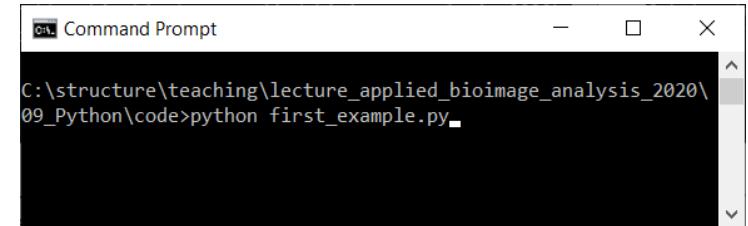
At the bottom, it shows "Ln 1, Col 1" and "100% Windows (CRLF) UTF-8".

- Windows Explorer



- Execute Python script

```
python <filename>
```



# The command line

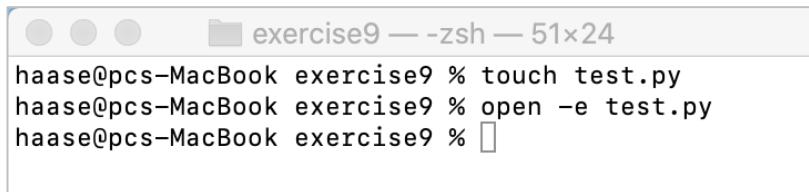
- Mac OS specific

- Text editor

```
touch <filename>
```

```
open -e <filename>
```

Create a new file



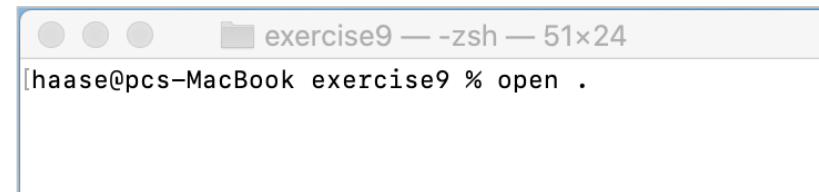
```
exercise9 -- zsh -- 51x24
haase@pcs-MacBook exercise9 % touch test.py
haase@pcs-MacBook exercise9 % open -e test.py
haase@pcs-MacBook exercise9 %
```

- Finder

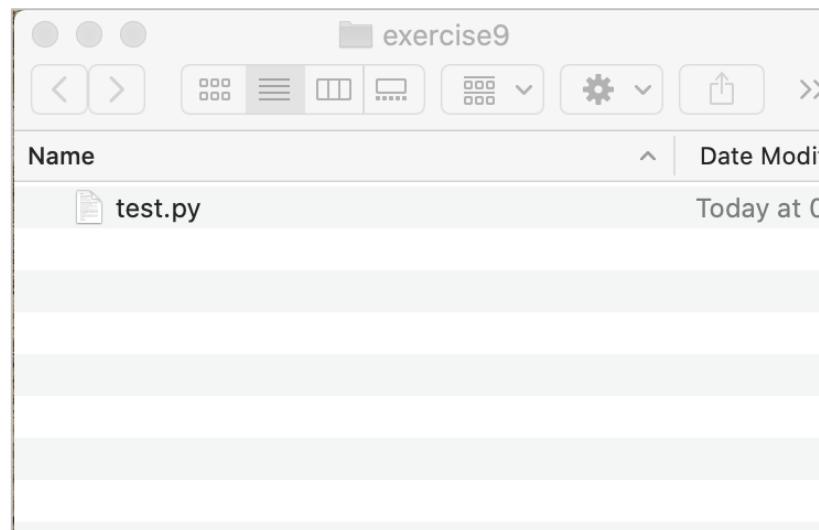
```
open .
```

- Execute Python script

```
python <filename>
```



```
exercise9 -- zsh -- 51x24
haase@pcs-MacBook exercise9 % open .
```

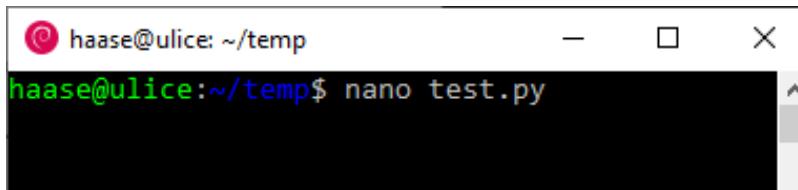


# The command line

- Linux specific

- Nano text editor

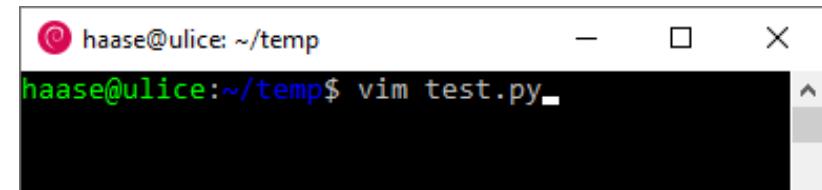
```
nano <filename>
```



```
haase@ulice:~/temp$ nano test.py
```

- vim

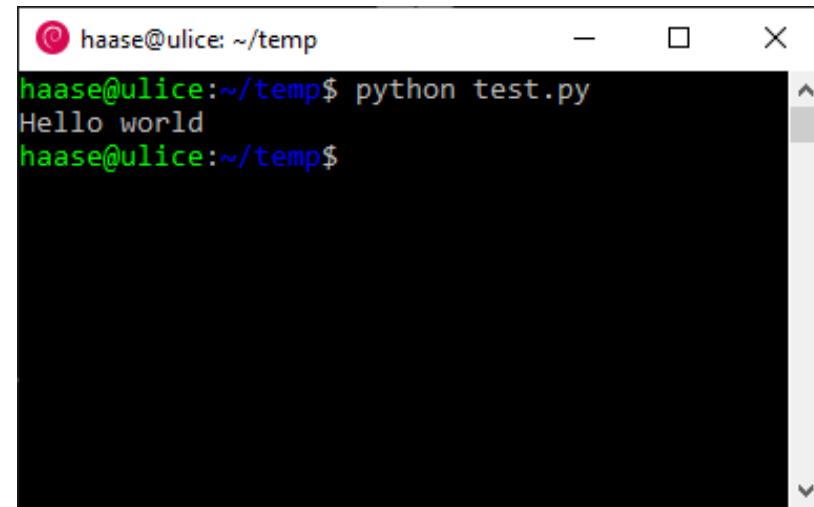
```
vim <filename>
```



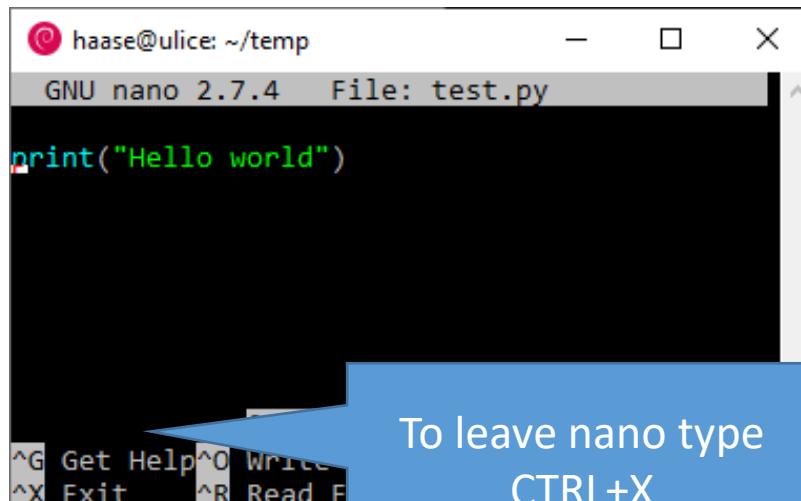
```
haase@ulice:~/temp$ vim test.py
```

- Execute Python script

```
python <filename>
```



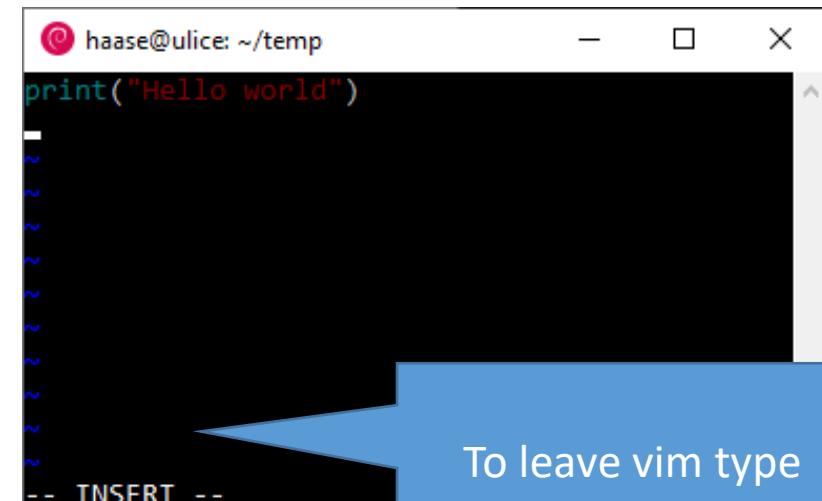
```
haase@ulice:~/temp$ python test.py
Hello world
haase@ulice:~/temp$
```



```
GNU nano 2.7.4  File: test.py
print("Hello world")
```

^G Get Help ^O Write ^R Read F  
^X Exit

To leave nano type  
CTRL+X  
N



```
-- INSERT --
```

```
print("Hello world")
```

To leave vim type  
:q!



# jupyter lab

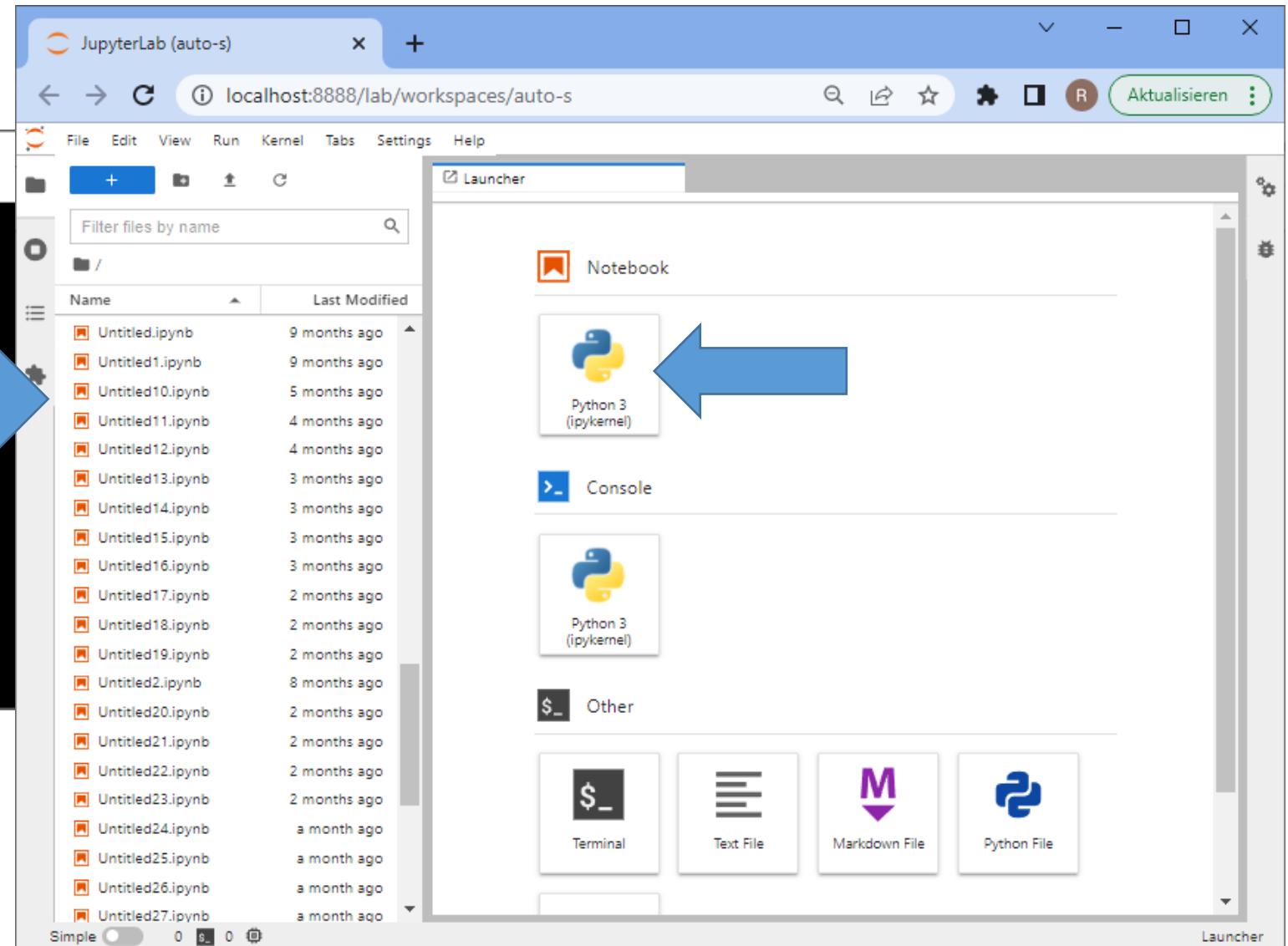
Robert Haase

April 2022

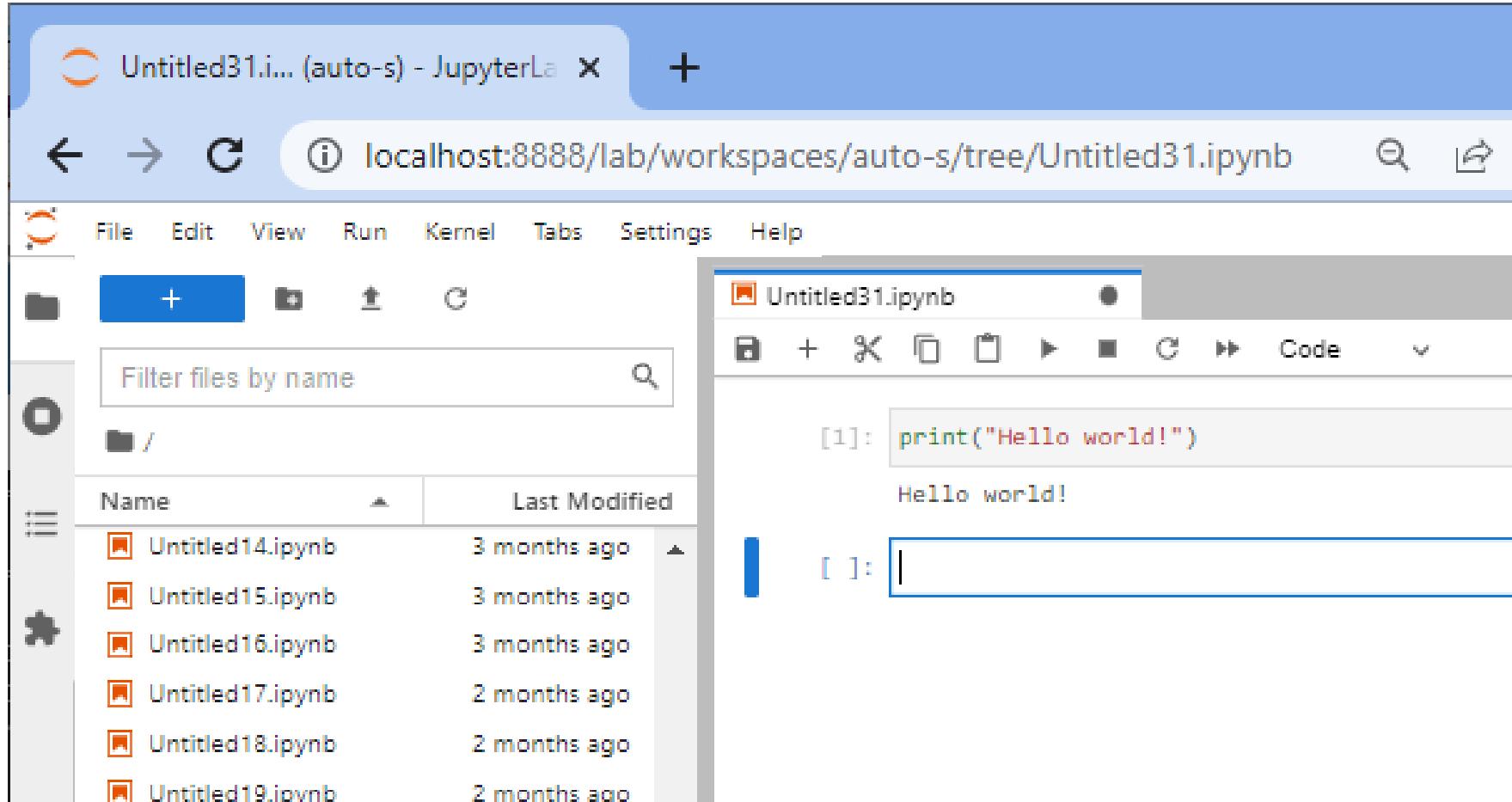
# Jupyter lab

- Our programming environment for this semester

```
c:\ Command Prompt - conda deactivate - cond...
c:\Users\rober>conda activate bio_39
(bio_39) c:\Users\rober>jupyter lab
```

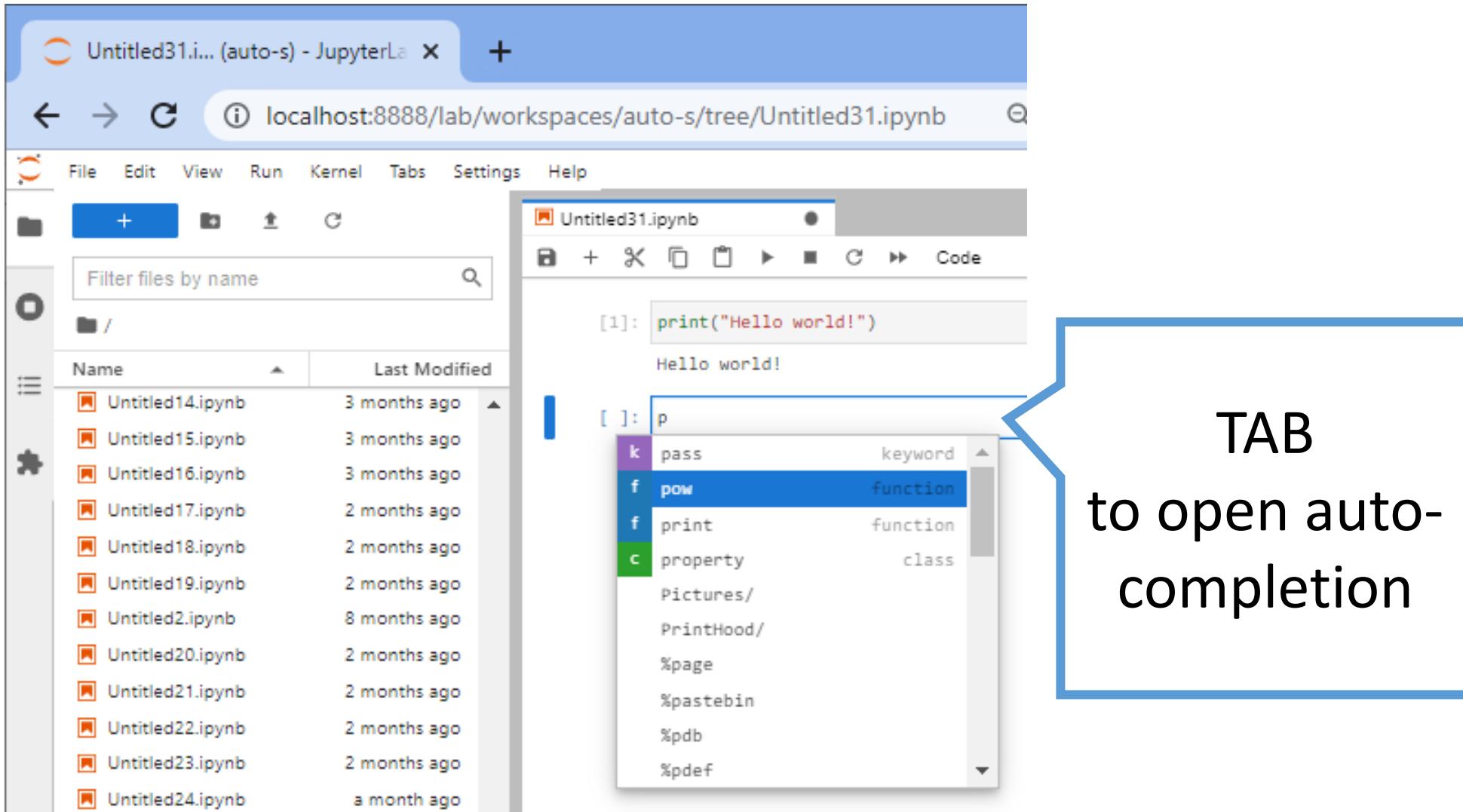


- Execute code cell-by-cell and see results instantaneously

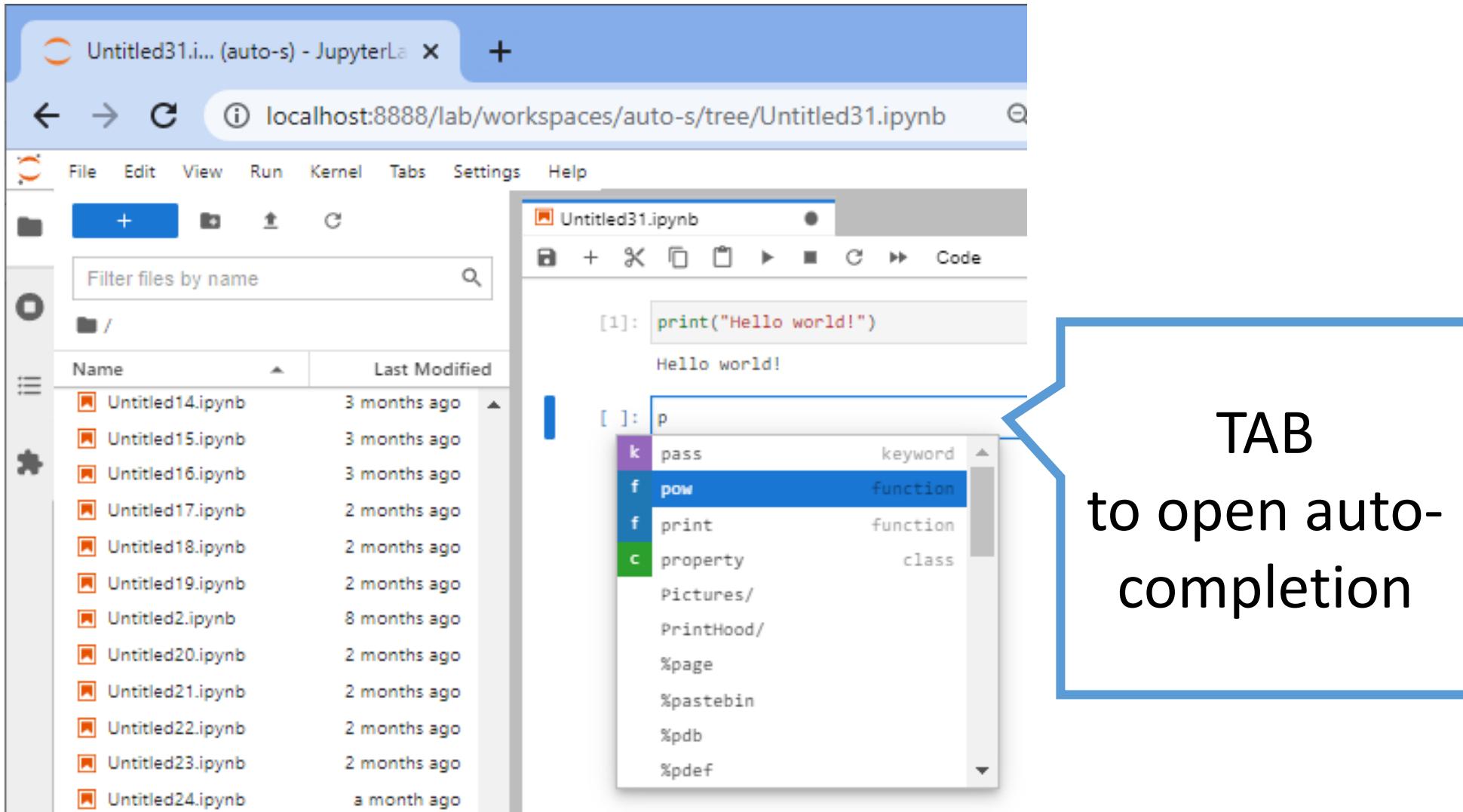


SHIFT + ENTER  
to execute a  
code cell

- Context-specific help, auto-completion

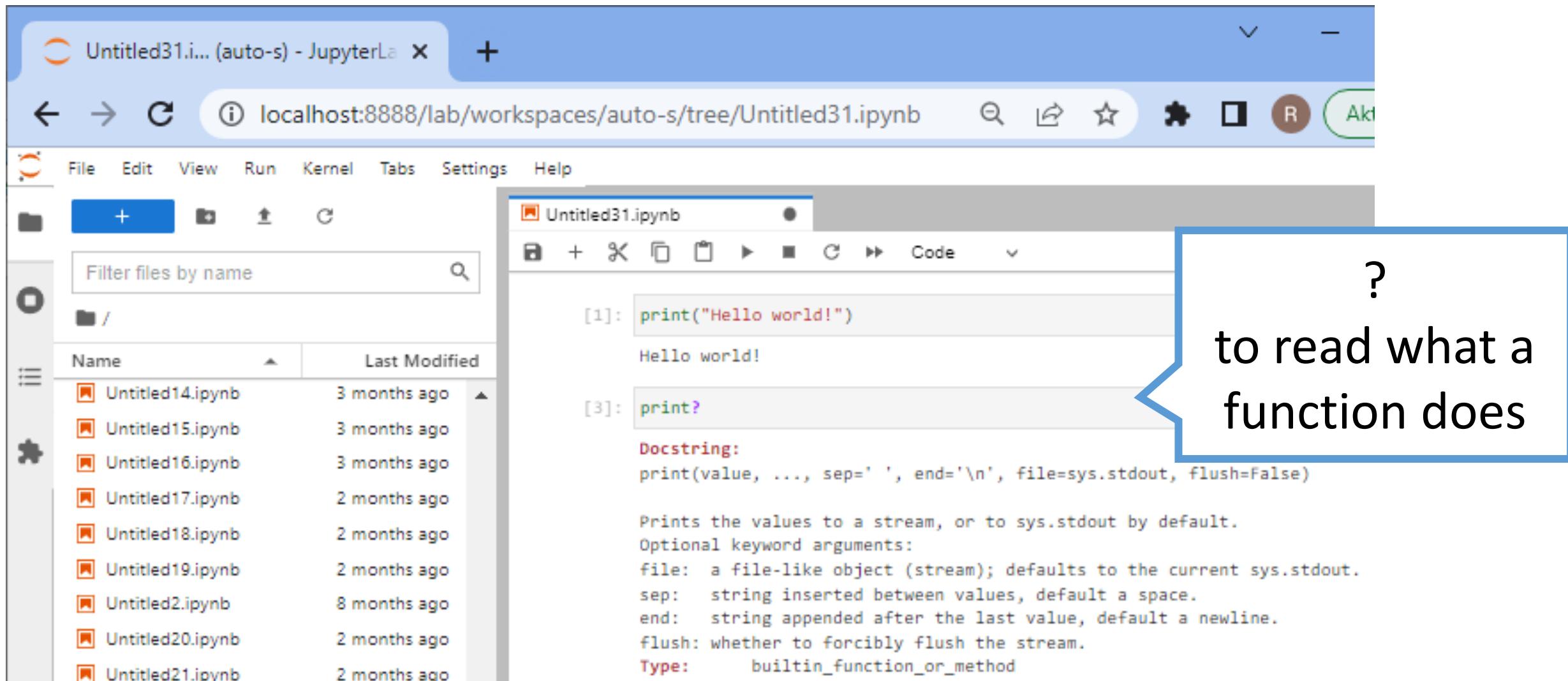


- Context-specific help, auto-completion



TAB  
to open auto-  
completion

- Help / “docstrings”



The screenshot shows the Jupyter Lab interface. On the left is a file browser with a sidebar containing icons for file operations like creating (+), deleting (trash), and moving (up/down). A search bar says "Filter files by name". Below it is a list of notebooks:

Name	Last Modified
Untitled14.ipynb	3 months ago
Untitled15.ipynb	3 months ago
Untitled16.ipynb	3 months ago
Untitled17.ipynb	2 months ago
Untitled18.ipynb	2 months ago
Untitled19.ipynb	2 months ago
Untitled2.ipynb	8 months ago
Untitled20.ipynb	2 months ago
Untitled21.ipynb	2 months ago

The main area is a code editor with tabs for "Code" and "Text". A code cell at the top contains:

```
[1]: print("Hello world!")
```

The output below it is:

```
Hello world!
```

Another code cell below it contains:

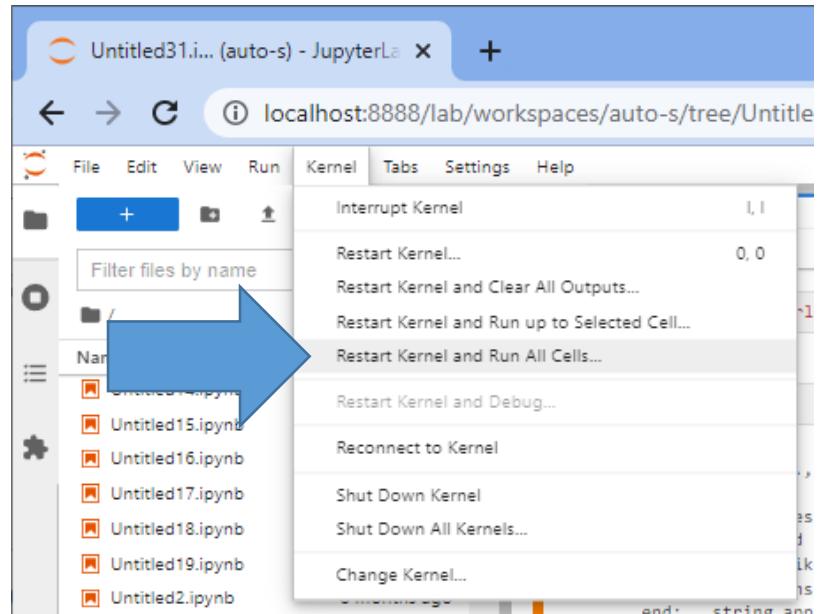
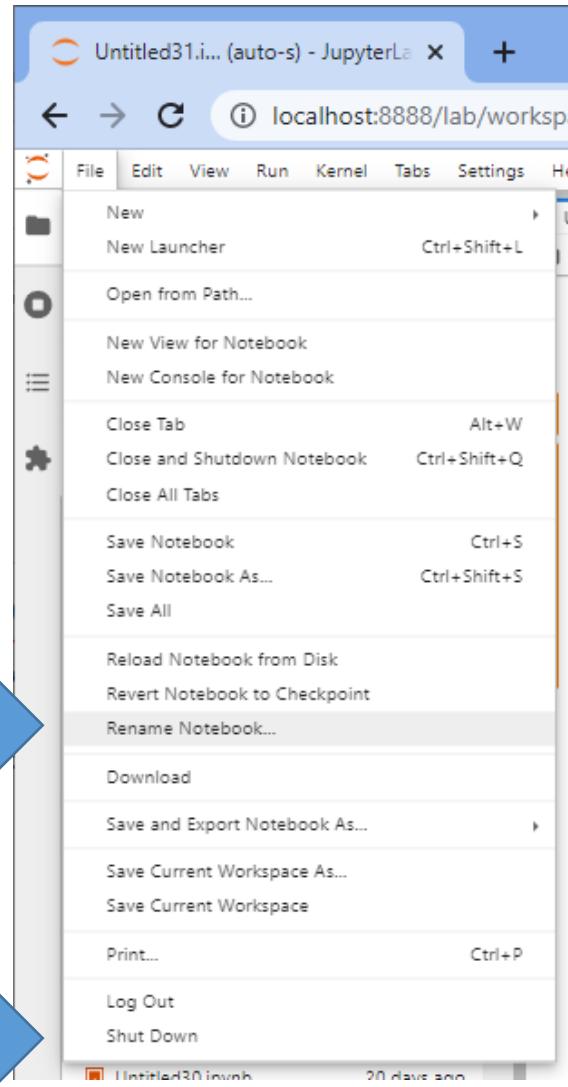
```
[3]: print?
```

The output is the docstring for the `print` function:

```
Docstring:  
print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)  
  
Prints the values to a stream, or to sys.stdout by default.  
Optional keyword arguments:  
file: a file-like object (stream); defaults to the current sys.stdout.  
sep: string inserted between values, default a space.  
end: string appended after the last value, default a newline.  
flush: whether to forcibly flush the stream.  
Type: builtin_function_or_method
```

A blue callout points from the question mark in the docstring to the text "to read what a function does".

- Saving / renaming / closing



Enforcing a “clean” execution state is important for ensuring reproducibility and repeatability

# Python Programming



Robert Haase

April 2022

# Data science with python

- Why Python?

Because copy&paste  
works so great.

The screenshot shows a Jupyter Notebook interface with four tabs:

- stardist/stardist: StarDist - Object (README.md)
- stardist/stardist: StarDist - Object (README.md)
- stardist/stardist: StarDist - Object (3\_prediction.ipynb)
- stardist/3\_prediction.ipynb at master · stardist/stardist · GitHub

The bottom tab displays a prediction result for a microscopy image of cells. The image is labeled with various colors (purple, green, yellow) representing detected objects. A color bar on the right indicates intensity values from 0 to 50. The code in the notebook is as follows:

```
# normalize image
from csbdeep.utils import normalize
normalized_image = normalize(image, 1, 99.8, axis=(0,1))

# Load pretrained deep-Learning model
from stardist.models import StarDist2D
model = StarDist2D.from_pretrained('2D_versatile_fluo')

# predict labels
label_image, details = model.predict_instances(normalized_image)
imshow(label_image)
```

Found model '2D\_versatile\_fluo' for 'StarDist2D'.  
Loading network weights from 'weights\_best.h5'.  
Loading thresholds from 'thresholds.json'.  
Using default values: prob\_thresh=0.479071, nms\_thresh=0.3.

```
matplotlib_plugin.py (150): Low image data range; displaying image
```

```
In [5]: img = normalize(X[16], 1, 99.8, axis=axis_norm)
labels, details = model.predict_instances(img)

In [6]: plt.figure(figsize=(8,8))
plt.imshow(img if img.ndim==2 else img[...,:], clim=(0,1), cmap=lbl_cmap)
plt.imshow(labels, cmap=lbl_cmap, alpha=0.5)
plt.axis('off');
```

In [3]:

```
# normalize image
from csbdeep.utils import normalize
normalized_image = normalize(image, 1, 99.8, axis=(0,1))

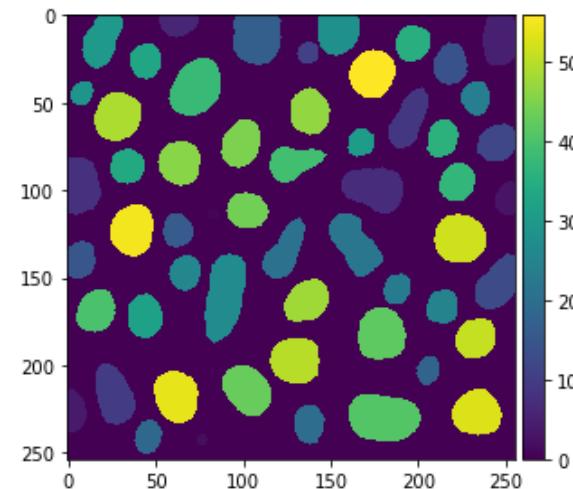
# Load pretrained deep-Learning model
from stardist.models import StarDist2D
model = StarDist2D.from_pretrained('2D_versatile_fluo')

# predict labels
label_image, details = model.predict_instances(normalized_image)
imshow(label_image)
```

Found model '2D\_versatile\_fluo' for 'StarDist2D'.  
Loading network weights from 'weights\_best.h5'.  
Loading thresholds from 'thresholds.json'.  
Using default values: prob\_thresh=0.479071, nms\_thresh=0.3.

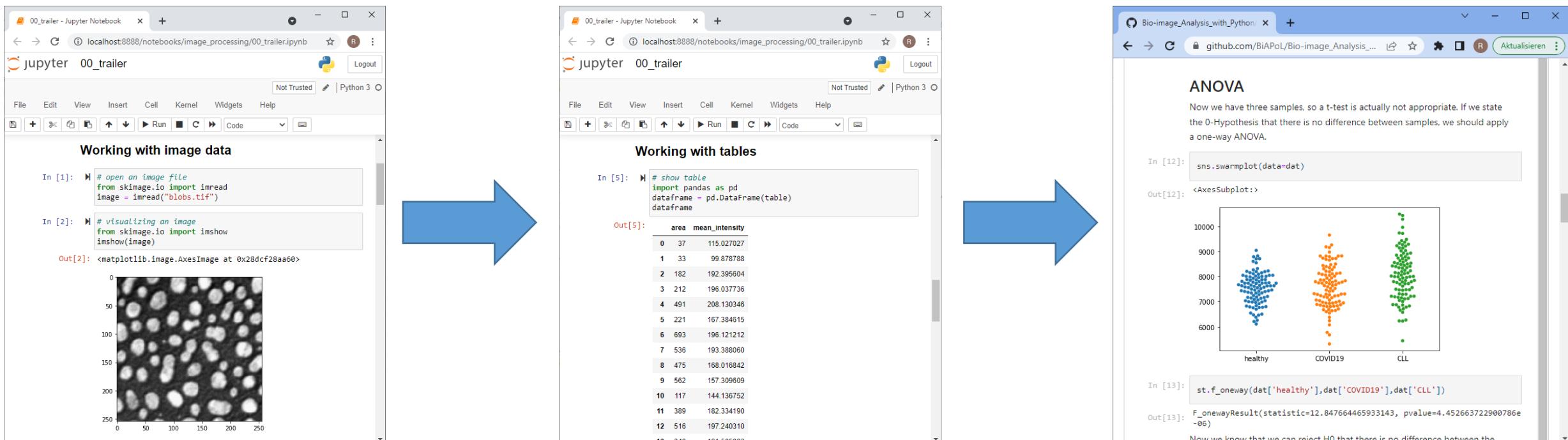
```
matplotlib_plugin.py (150): Low image data range; displaying image
```

```
In [3]: <matplotlib.image.AxesImage at 0x28dd9ff991c0>
```

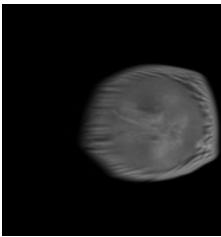


<https://github.com/stardist/stardist>

- Major goals of image analysis via scripting:
  - reproducible workflows for processing images (raw data) into quantitative information and biological insights.
  - automation
  - Sharing code, knowledge
  - Prevent reinventing the wheel



banana0008.tif  
banana0009.tif  
banana0010.tif  
banana0011.tif  
banana0012.tif



- Remove shell
- Repeat until nothing left:
  - Take a bite
  - Chew
  - Swallow
  - Digest

- Access folder
- Repeat for all images:
  - Open an image file
  - Segment the banana slice
  - Analyse it
  - Save measurements

```

slice_areas = []
for root, dirs, files in os.walk(data_folder):
    for file in files:
        if file.endswith('tif'):

            # Load data
            from skimage.io import imread
            image = imread(root + file)

            # segment it
            from skimage.filters import threshold_otsu
            binary_image = image > threshold_otsu(image)

            from skimage.measure import label
            labels = label(binary_image)

            # measure radius
            from skimage.measure import regionprops
            statistics = regionprops(labels)
            areas = [s.area for s in statistics]

            # store result in array
            import numpy as np
            slice_areas.append(np.max(areas))

```

# Python programming basics

Robert Haase

April 2022

# Working with variables

- Variables can hold numeric values and you can do math with them

```
▶ # initialize program
a = 5
b = 3

# run algorithm on given parameters
sum = a + b

# print out result
print (sum)
```

# Mathematical operations

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

- Power

```
▶ pow(3, 2)
```

```
]: 9
```

- Absolute

```
▶ abs(-8)
```

```
]: 8
```

- Rounding

```
▶ round(4.6)
```

```
]: 5
```

Be careful with  
some of them!

```
▶ round(4.5)
```

```
]: 4
```

[https://en.wikipedia.org/wiki/Rounding#Round\\_half\\_to\\_even](https://en.wikipedia.org/wiki/Rounding#Round_half_to_even)

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 Python 3.8  
#  
# Author: Robert Haase, PoL TUD  
#         Robert.haase@tu-dresden.de  
# April 2021  
  
# initialise program  
a = 1  
b = 2.5  
  
# run complicated algorithm  
final_result = a + b  
  
# print the final result  
print( final_result )
```

# Working with variables and string values

- Also strings as values for variables are supported

Single and double quotes  
allowed

```
▶firstname = "Robert"  
lastname = 'Haase'  
  
print("Hello " + firstname + " " + lastname)
```

Hello Robert Haase

# Working with variables and string values

- Also strings as values for variables are supported
- When combining strings and numbers, you need to explicitly define what you want to do.

```
# mixing types

a = 5
b = "2"

print (a + b)
```

```
-----  
TypeError                                     Traceback (most recent call last)
<ipython-input-4-51629e6a285f> in <module>
      4 b = "2"
      5
----> 6 print (a + b)

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

```
# mixing types to make numbers

a = 5
b = "2"

print (a + int(b))
```

7

```
# mixing types

a = "5"
b = 2

print (a + b)
```

```
-----  
TypeError                                     Traceback (most recent call last)
<ipython-input-5-85ae49867097> in <module>
      4 b = 2
      5
----> 6 print (a + b)

TypeError: can only concatenate str (not "int") to str
```

```
# mixing types to make strings

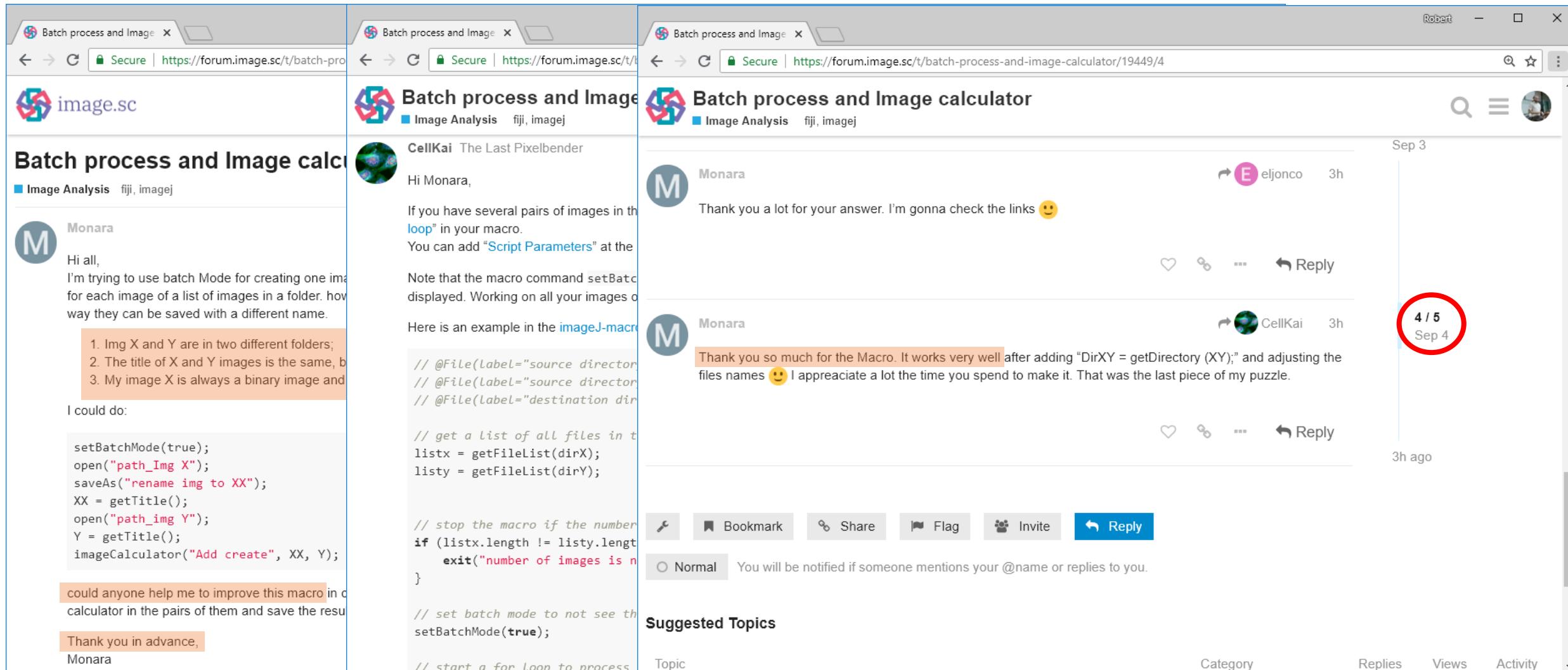
a = "5"
b = 2

print (a + str(b))
```

52

- Conversion to a floating point number: float()

- Visit <http://image.sc>, an online Q&A platform for the image sciences hosted by University Wisconsin and the Broad institute of Harvard and MIT.



The screenshot shows three browser tabs from the image.sc forum:

- Batch process and Image calculator**: Monara asks how to use batch mode to save images with different names. CellKai responds with a macro example.
- Batch process and Image calculator**: Monara thanks CellKai for the macro.
- Batch process and Image calculator**: CellKai replies, noting the macro works well after some adjustments.

A red circle highlights the "4 / 5" rating and the date "Sep 4" next to the last post.

**Monara's message:**

Hi all,  
I'm trying to use batch Mode for creating one image for each image of a list of images in a folder. how way they can be saved with a different name.

1. Img X and Y are in two different folders;  
2. The title of X and Y images is the same, b  
3. My image X is always a binary image and

I could do:

```
setBatchMode(true);
open("path_Img X");
saveAs("rename img to XX");
XX = getTitle();
open("path_img Y");
Y = getTitle();
imageCalculator("Add create", XX, Y);
```

could anyone help me to improve this macro in calculator in the pairs of them and save the resu

Thank you in advance,  
Monara

**CellKai's response:**

CellKai The Last Pixelbender

Hi Monara,

If you have several pairs of images in the "loop" in your macro.  
You can add "Script Parameters" at the top of the macro.

Note that the macro command `setBatchMode(true)` is displayed. Working on all your images one by one.

Here is an example in the [imageJ-macros](#) repository:

```
// @File(Label="source directory")
// @File(Label="source directory")
// @File(Label="destination directory")

// get a List of all files in the source directory
listx = getFileList(dirX);
listy = getFileList(dirY);

// stop the macro if the number of files is not equal
if (listx.length != listy.length)
    exit("number of images is not equal");

// set batch mode to not see the progress bar
setBatchMode(true);

// start a for loop to process
```

**Monara's reply:**

Monara

Thank you a lot for your answer. I'm gonna check the links 😊

**CellKai's final message:**

CellKai

Thank you so much for the Macro. It works very well after adding "DirXY = getDirectory (XY);" and adjusting the file names 😊 I appreciate a lot the time you spend to make it. That was the last piece of my puzzle.

# Demo

Robert Haase

April 2022

# Homework

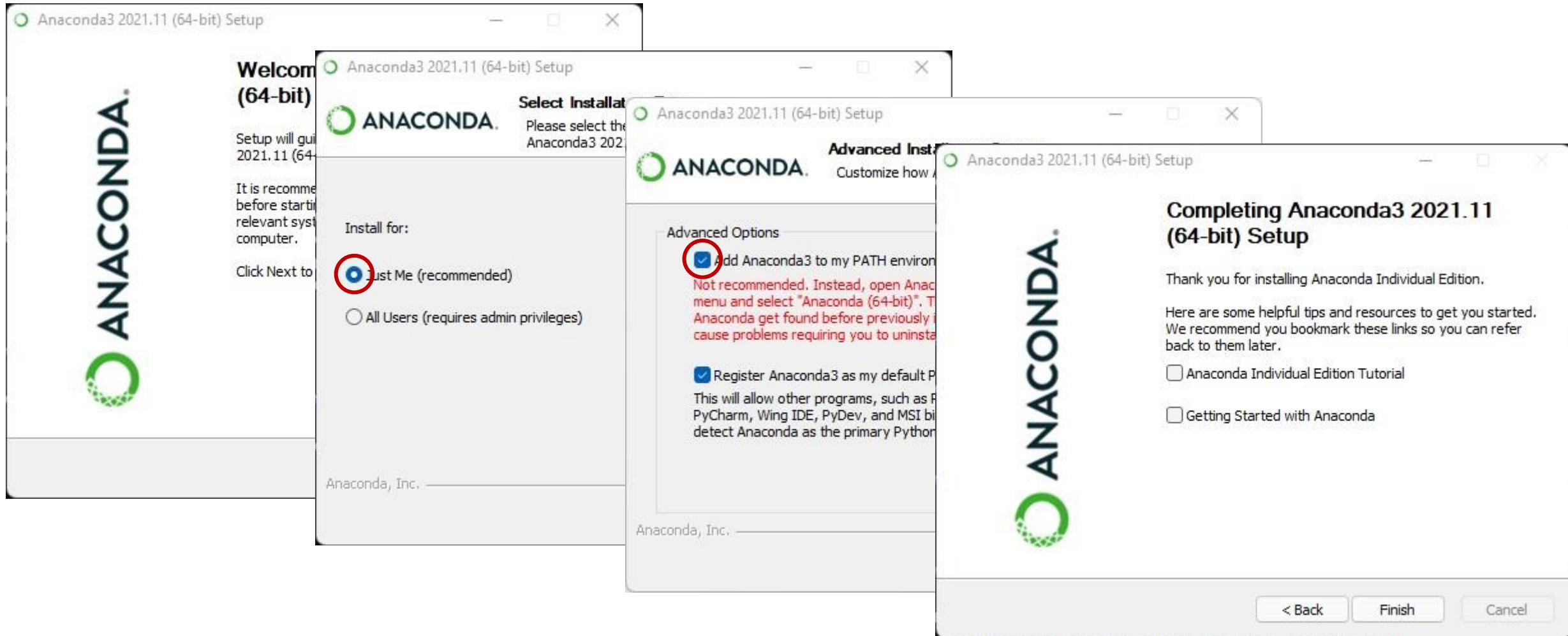
Robert Haase

April 2022

# Homework: Install anaconda and test python

Detailed instructions:

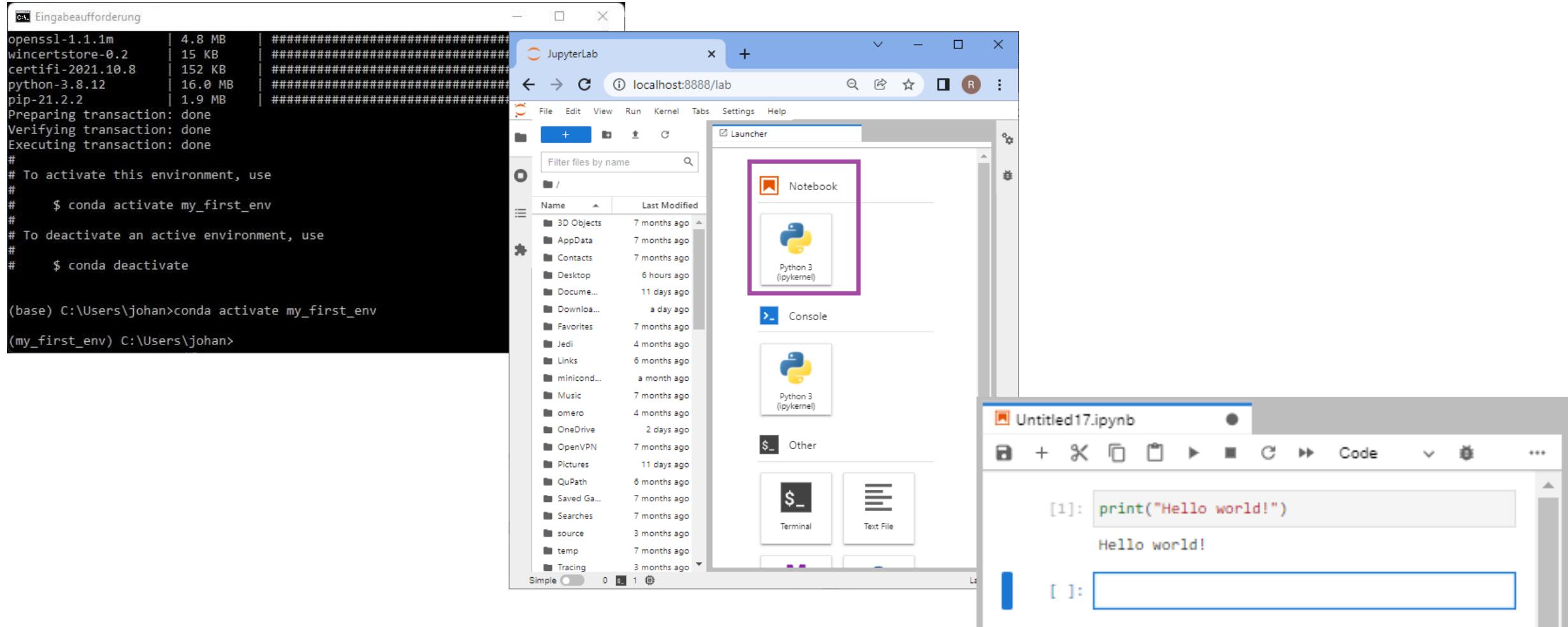
- [https://biapol.github.io/blog/johannes\\_mueller/anaconda\\_getting\\_started/](https://biapol.github.io/blog/johannes_mueller/anaconda_getting_started/)



# Homework: Install anaconda and test python

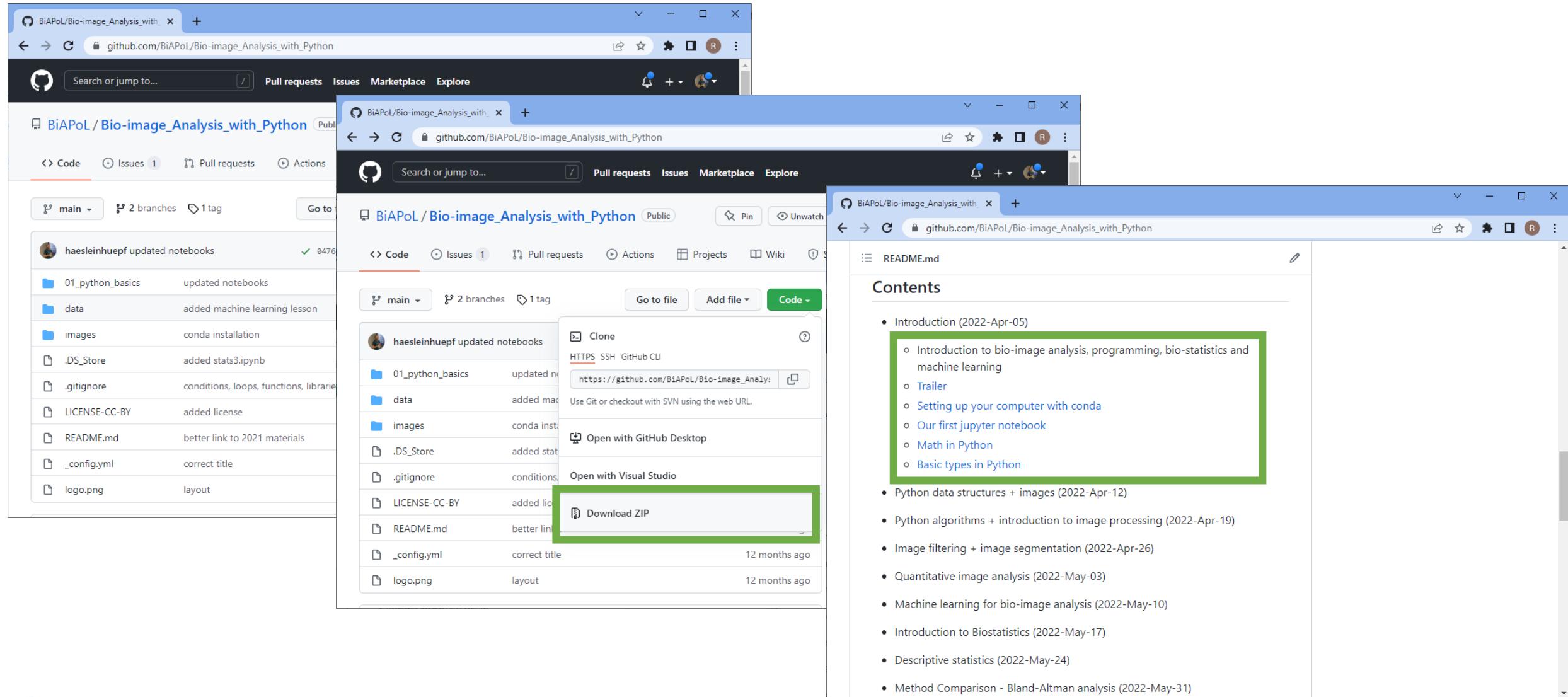
Detailed instructions:

- [https://biapol.github.io/blog/johannes\\_mueller/anaconda getting started/](https://biapol.github.io/blog/johannes_mueller/anaconda_getting_started/)



# Homework: Explore our github repository

- [https://github.com/BiAPoL/Bio-image Analysis with Python](https://github.com/BiAPoL/Bio-image_Analysis_with_Python)

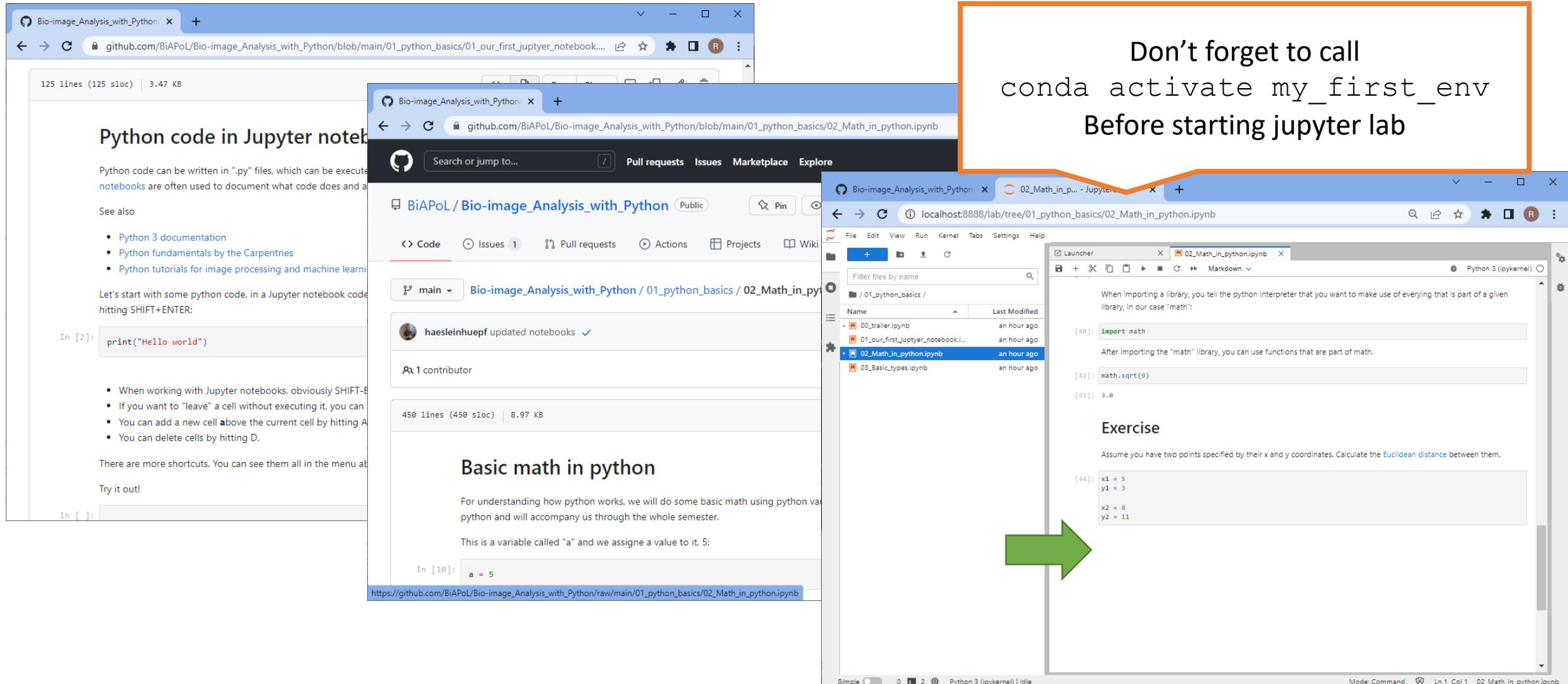


The image displays three side-by-side screenshots of a GitHub repository page for "BiAPoL/Bio-image\_Analysis\_with\_Python".

- Screenshot 1 (Left):** Shows the repository's main page with a list of files and commits. A commit by "haesleinhuepf" is highlighted, showing updates to notebooks, data, images, and other files.
- Screenshot 2 (Middle):** Shows the same repository page with a focus on the "Code" tab. It displays a "Clone" section with links for HTTPS, SSH, and GitHub CLI, and options to "Open with GitHub Desktop" or "Open with Visual Studio". A "Download ZIP" button is highlighted with a green box.
- Screenshot 3 (Right):** Shows the contents of the "README.md" file. The file lists a series of topics and their corresponding dates:
  - Introduction (2022-Apr-05)
    - Introduction to bio-image analysis, programming, bio-statistics and machine learning
    - Trailer
    - Setting up your computer with conda
    - Our first jupyter notebook
    - Math in Python
    - Basic types in Python
  - Python data structures + images (2022-Apr-12)
  - Python algorithms + introduction to image processing (2022-Apr-19)
  - Image filtering + image segmentation (2022-Apr-26)
  - Quantitative image analysis (2022-May-03)
  - Machine learning for bio-image analysis (2022-May-10)
  - Introduction to Biostatistics (2022-May-17)
  - Descriptive statistics (2022-May-24)
  - Method Comparison - Bland-Altman analysis (2022-May-31)

# Homework: Basic python

- [https://github.com/BiAPoL/Bio-image Analysis with Python](https://github.com/BiAPoL/Bio-image_Analysis_with_Python)



Don't forget to call  
conda activate my\_first\_env  
Before starting jupyter lab

Python code in Jupyter notebook

Python code can be written in ".py" files, which can be executed in a Jupyter notebook. Notebooks are often used to document what code does and a lot more.

See also

- Python 3 documentation
- Python fundamentals by the Carpentries
- Python tutorials for image processing and machine learning

Let's start with some python code, in a Jupyter notebook code cell. You can execute it by hitting SHIFT+ENTER:

```
In [2]: print("Hello world")
```

When working with Jupyter notebooks, obviously SHIFT+ENTER is the key combination to execute a cell. If you want to "leave" a cell without executing it, you can hit ESC. You can add a new cell above the current cell by hitting A. You can delete cells by hitting D.

There are more shortcuts. You can see them all in the menu above the code cell.

Try it out!

```
In [ ]:
```

Basic math in python

For understanding how python works, we will do some basic math using python variables. We will use the variable "a" and assign it a value of 5. This is a variable called "a" and we assign a value to it: 5.

```
In [10]: a = 5
```

https://github.com/BiAPoL/Bio-image\_Analysis\_with\_Python/raw/main/01\_python\_basics/02\_Math\_in\_python.ipynb

File Edit View Run Kernel Tabs Settings Help

00\_trailer.ipynb 01\_our\_first\_jupyter\_notebook.ipynb 02\_Math\_in\_python.ipynb 03\_Basic\_types.ipynb

[40]: import math

After importing the "math" library, you can use functions that are part of math.

[41]: math.sqrt(9)

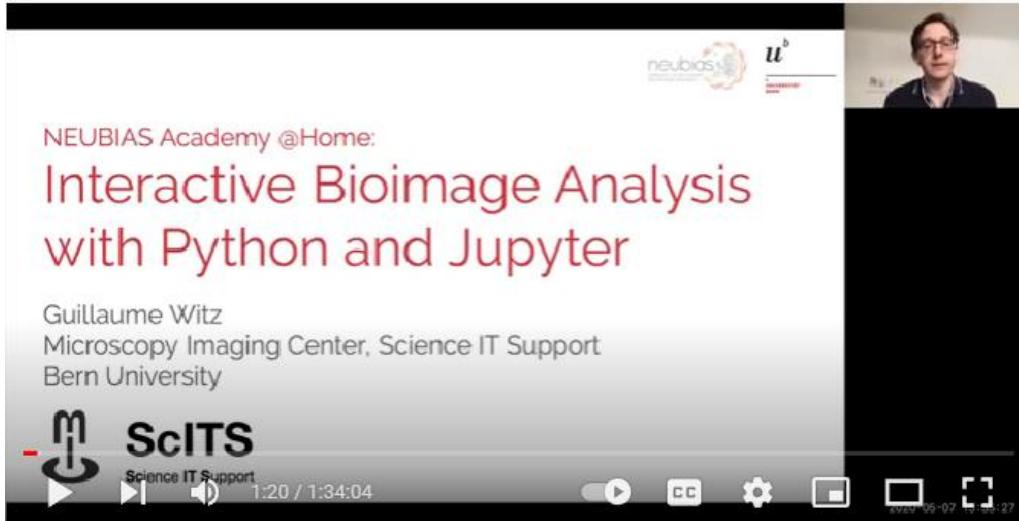
[41]: 3.0

Exercise

Assume you have two points specified by their x and y coordinates. Calculate the Euclidean distance between them.

```
[44]: x1 = 5
y1 = 3
x2 = 8
y2 = 11
```

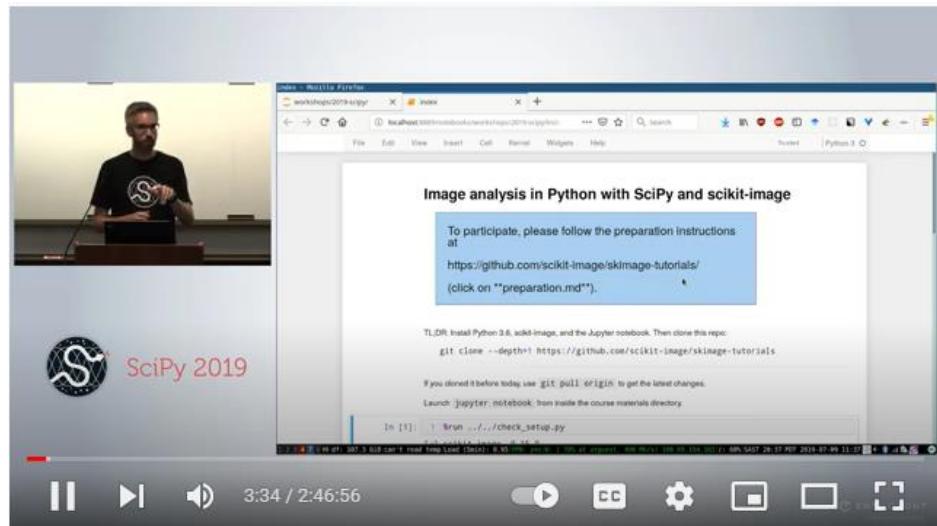
# More resources



Guillaume Witz, NEUBIAS Academy 2020

Watch more:

- <https://www.youtube.com/watch?v=2KF8vBrp3Zw>
- <https://www.youtube.com/watch?v=d1CIV9irQAY>
- [https://www.youtube.com/watch?v=X\\_pCiVQ4c4E](https://www.youtube.com/watch?v=X_pCiVQ4c4E)



Stéfan van der Walt, Juan Nunez-Iglesias, SciPy 2019



Sreenivas Bhattiprolu, Python for Microscopists @Youtube 2019-...  
April 2022

Today, you learned

- *Bio-image analysis*
  - Quantitative
  - Objective
  - Reproducible
  - Repeatable
  - Reliable
- The command line
- Python basics
- Reminder:
  - Use the etherpad to ask questions anonymously!
  - Alternatively: Ask on <https://image.sc> in public.

Coming up next

- Python programming
  - Lists, tuples and dictionaries
  - Tables
  - Image data

```
# going through arrays pair-wise
measurement_1 = [1, 9, 7, 1, 2, 8, 9, 2, 1, 7, 8]
measurement_2 = [4, 5, 5, 7, 4, 5, 4, 6, 6, 5, 4]

for m_1, m_2 in zip(measurement_1, measurement_2):
    print("Paired measurements: " + str(m_1) + " and " + str(m_2))
```

```
Paired measurements: 1 and 4
Paired measurements: 9 and 5
Paired measurements: 7 and 5
Paired measurements: 1 and 7
Paired measurements: 2 and 4
Paired measurements: 8 and 5
Paired measurements: 9 and 4
Paired measurements: 2 and 6
Paired measurements: 1 and 6
Paired measurements: 7 and 5
Paired measurements: 8 and 4
```