PoL Physics of Life TU Dresden

BIOTEC Biotechnology Center TU Dresden

CENTER FOR SYSTEMS BIOLOGY DRESDEN

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

Anna Poetsch, Melissa Sanabria, Allyson Ryan, Robert Haase

@haesleinhuepf

April 2023

The team




Anna Poetsch, Melissa Sanabria, Allyson Ryan, Robert Haase

Credits: Hagen Gebauer

Lecture overview: Programming

- We will focus on Python programming.
- Goal: **Allow you to do things automatically instead of suffering long time when doing it manually.**

Basics

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

```
print(c)
```

```
8
```

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

```
print(h)
```

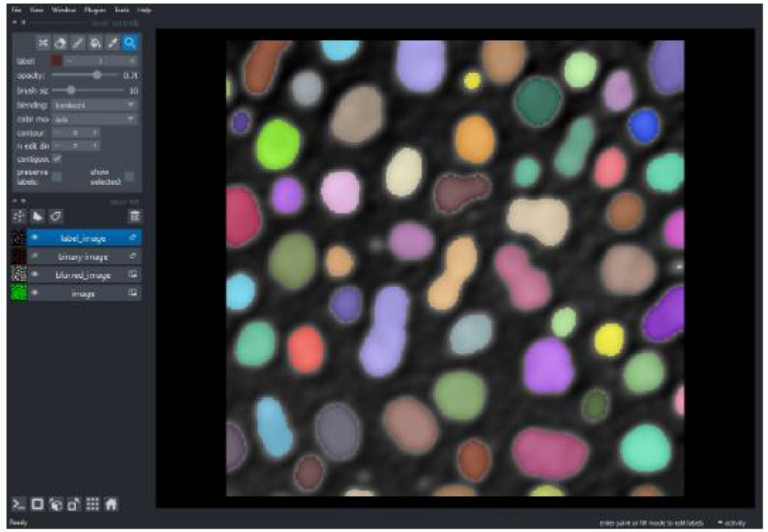
```
5.285714285714286
```

Image Analysis

```
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)

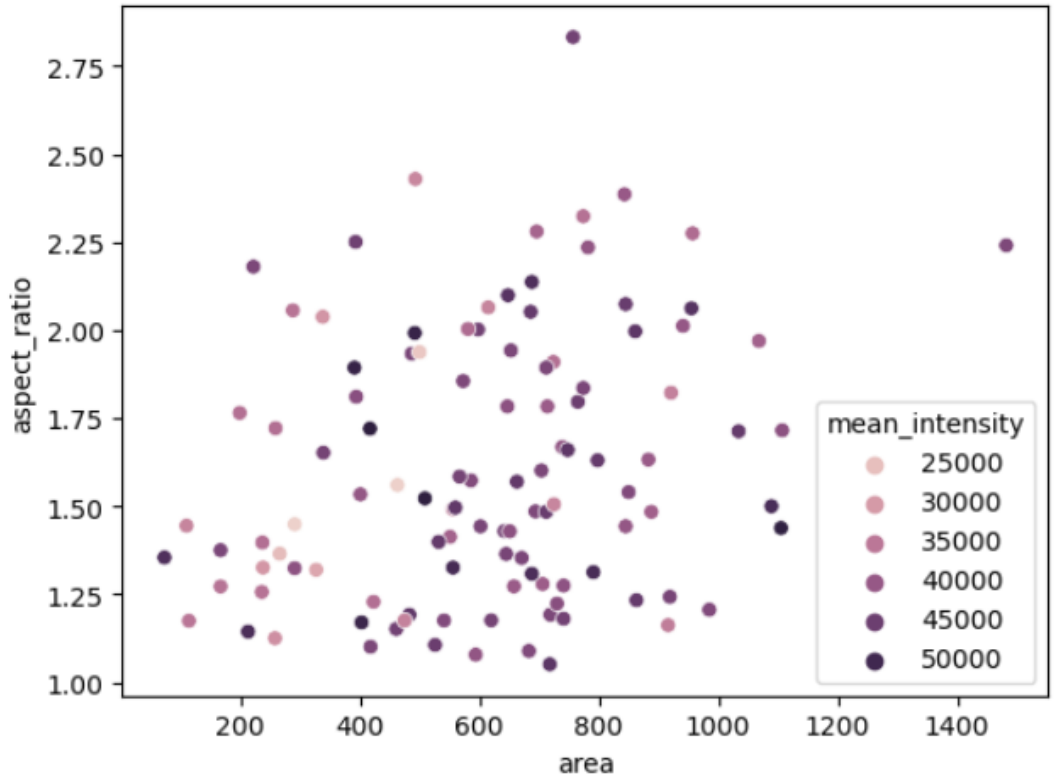
```
napari.utils.nbscreenshot(viewer)
```



Plotting / statistics

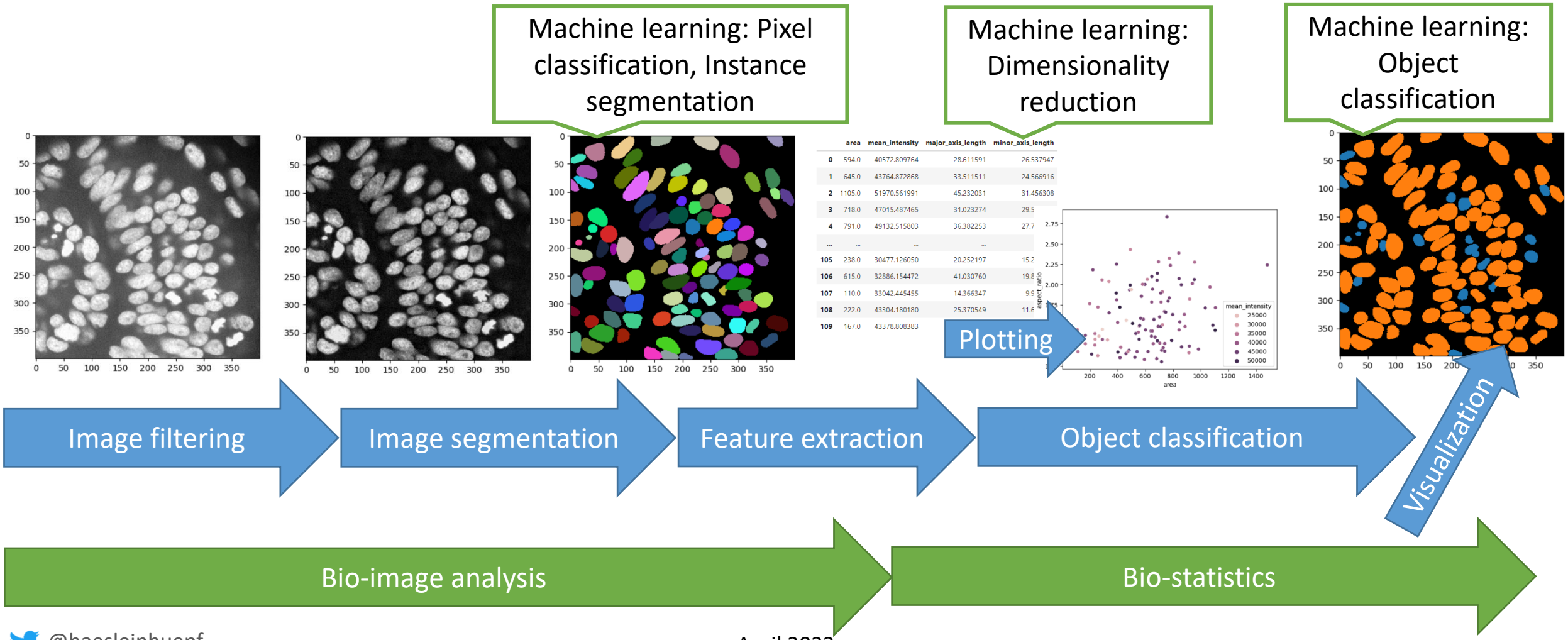
```
seaborn.scatterplot(dataframe, x='area', y='aspect_ratio', hue='mean_intensity')
```

```
<AxesSubplot: xlabel='area', ylabel='aspect_ratio'>
```



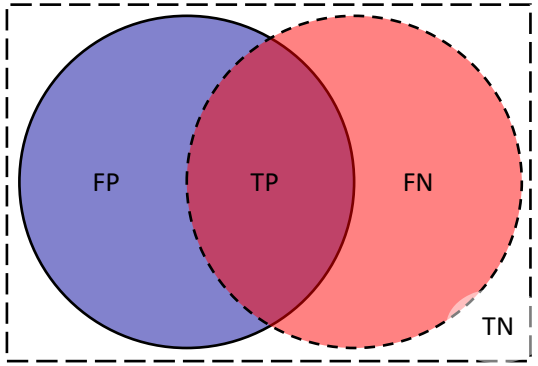
Lecture overview: Bio-image Analysis

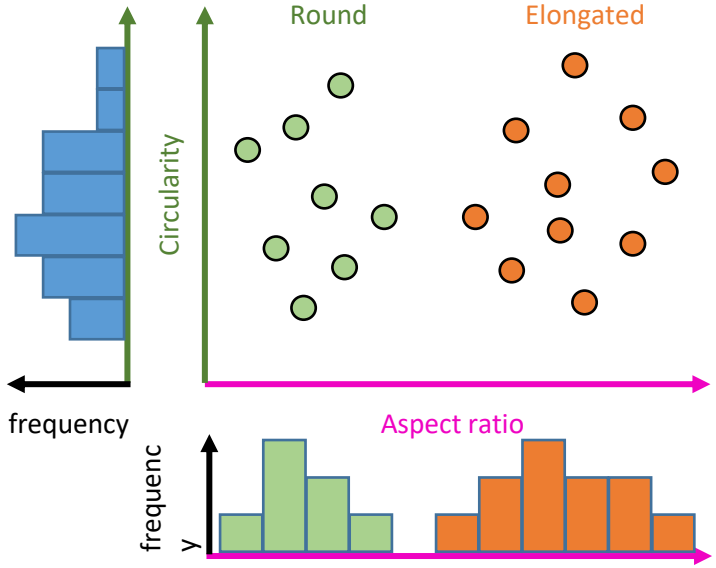
- Image Data Analysis workflows
- Goal: **Quantify observations, substantiate conclusions with numbers**

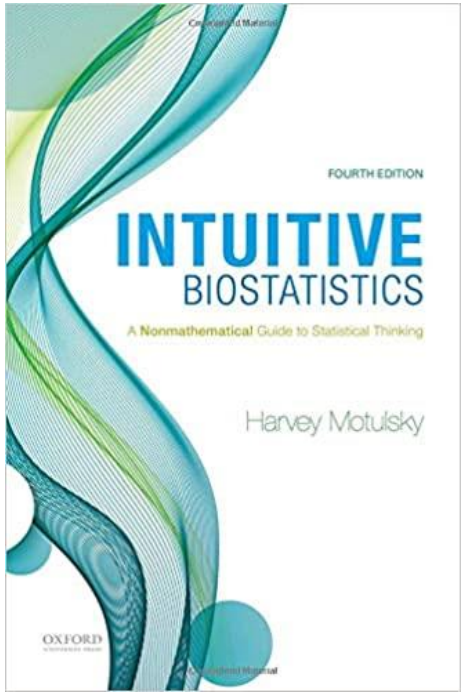


Lecture overview: Bio-statistics

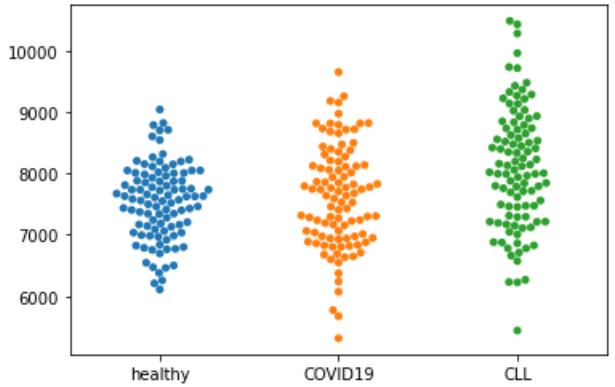
- Descriptive statistics
  - Distributions
- Inferential statistics
  - Hypothesis testing
- Multiple comparisons and correlations
- Clustering, dimensionality reduction





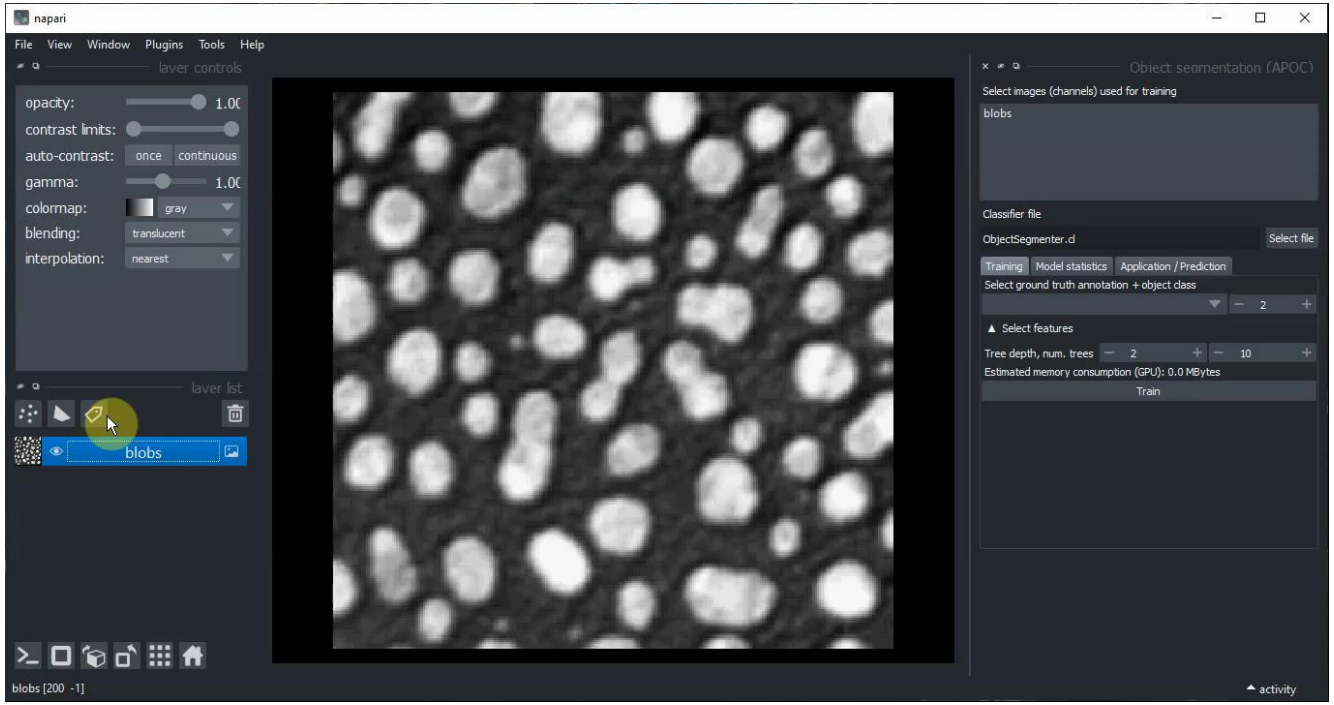


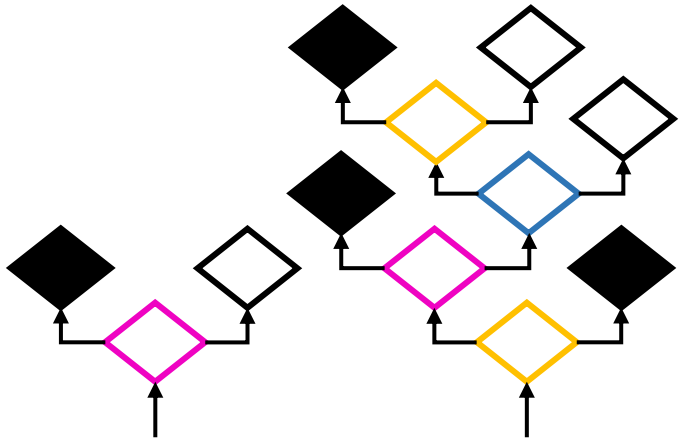
- Goal: **Allow you to draw conclusions from quantified observations.**



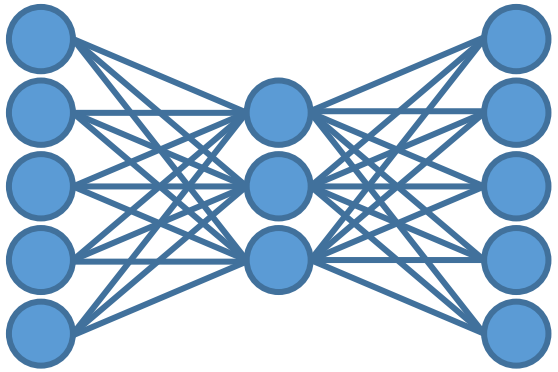
Lecture overview: Machine learning

- Machine learning
  - in the context of image analysis and genetics
- Computers can *learn* from data, potentially revealing relationships that are not obvious to a human
- Goal: **Give you an insight into state-of-the-art methods**





Random forest classifiers



Neural networks

Lecture overview

- 4.4.2022 - General introduction, introduction to Python Programming I
- 11.4.2022 - Introduction to Python programming II
- 18.4.2022 - Image Processing
- 25.4.2022 - Image Segmentation + Quality Assurance
- 2.5.2022 - Feature extraction
- 9.5.2022 - Introduction to Biostatistics
- 16.5.2022 - Descriptive Statistics
- 23.5.2022 – Hypothesis Testing
- 6.6.2022 – Introduction to Machine Learning + Random Forest Classifiers
- 13.6.2022 – Unsupervised Machine Learning
- 20.6.2022 – Supervised Machine Learning / Deep Learning
- 27.6.2022 – Introduction to Genomic Data
- 4.7.2022 – Multimodal Machine Learning
- 11.7.2022 – Summary, exam preparation

Lecture overview

- Every week will follow the same rough scheme
  - 13:00 : 90 min lecture
  - 14:20 : 90 min exercises
  - when you're done, enjoy the sun!
- 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
  - "closed book exam"

Exercise: Survey

In which category do you see yourself?

Molecular Bioengineering Bachelor / Master student	Other Bachelor / Master student	PhD student	Other

PoL Physics of Life TU Dresden

CENTER FOR SYSTEMS BIOLOGY DRESDEN

Introduction to Bio-Image Analysis

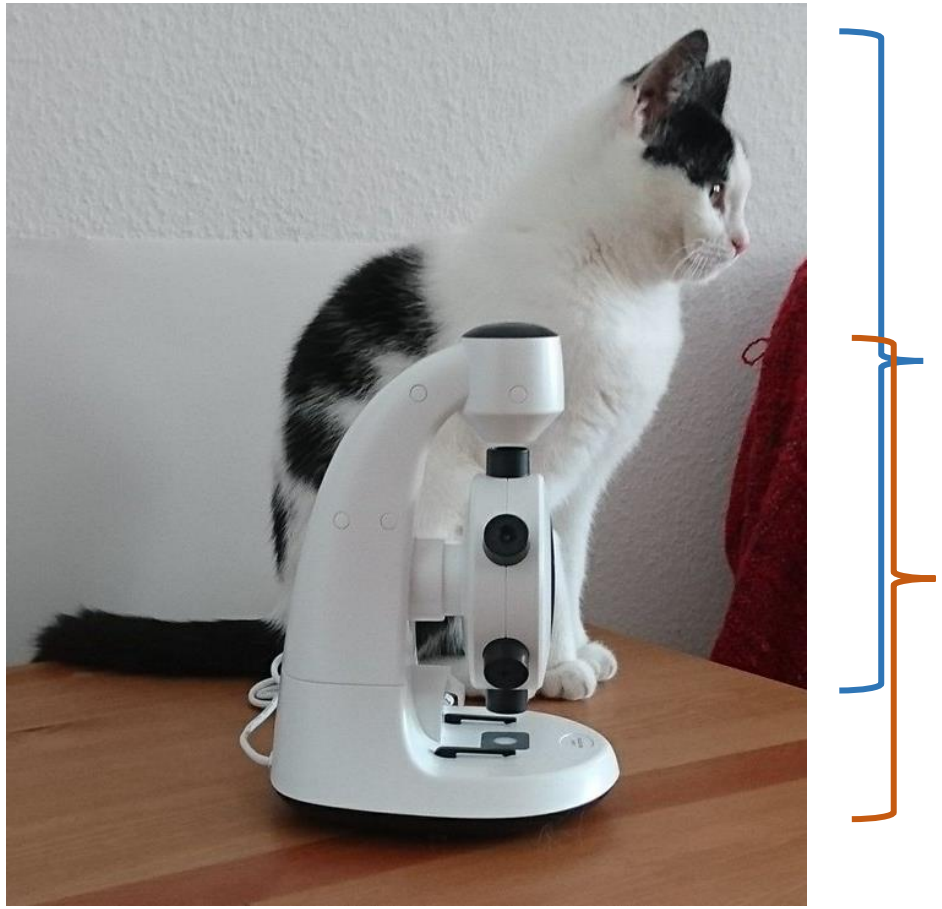
Robert Haase

@haesleinhuepf

April 2023

Quantitative bio-image analysis

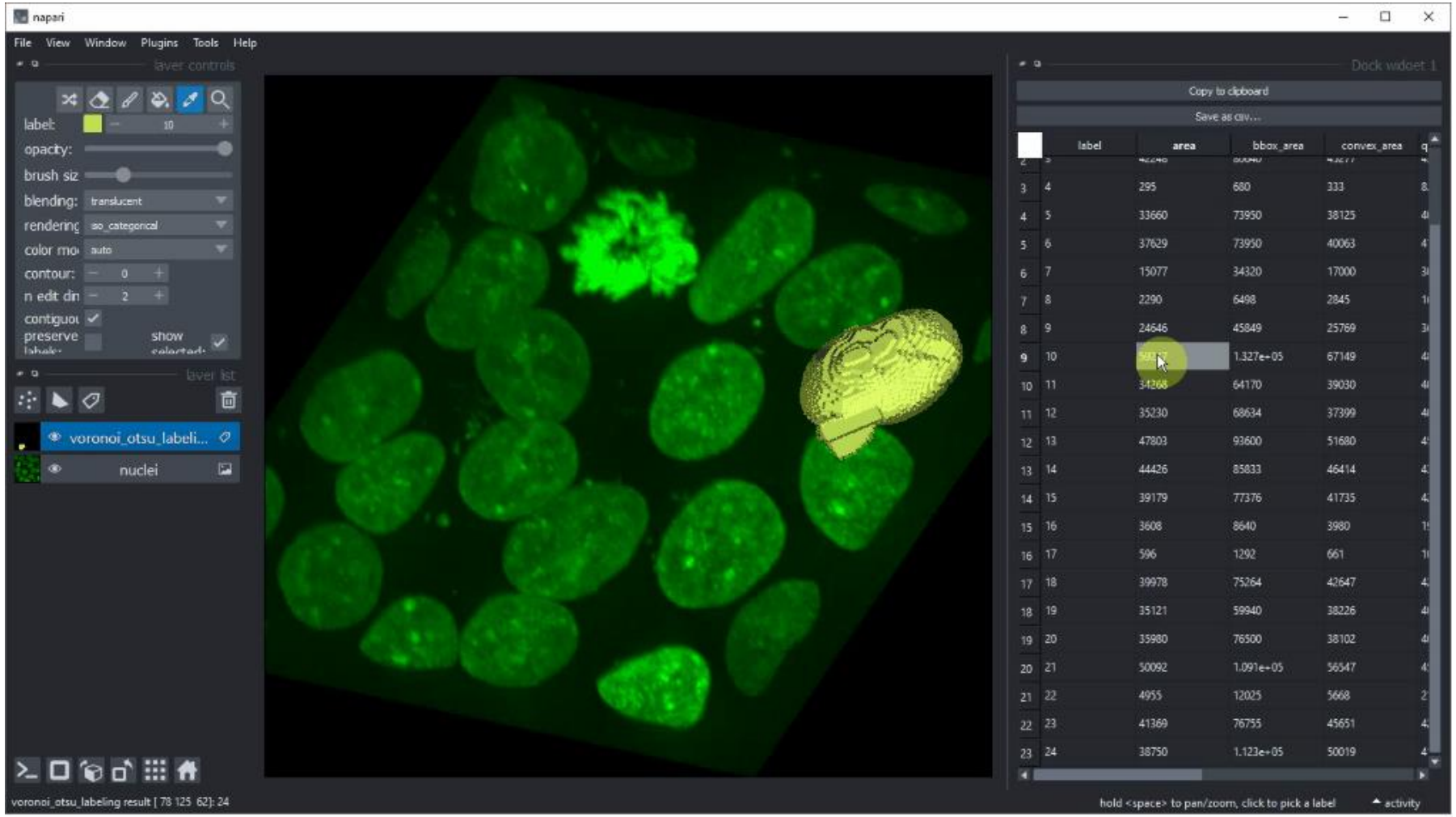
- Deriving quantitative information from images of biological samples taken with microscopes



cat height = 1.5 x microscope height

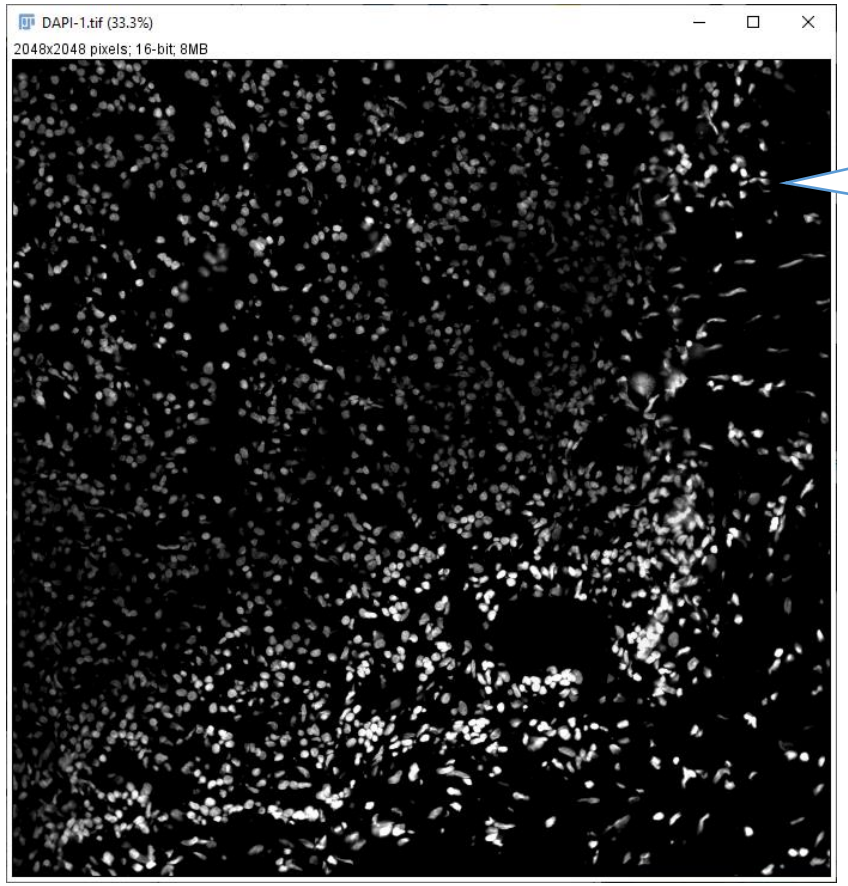
Quantitative bio-image analysis

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



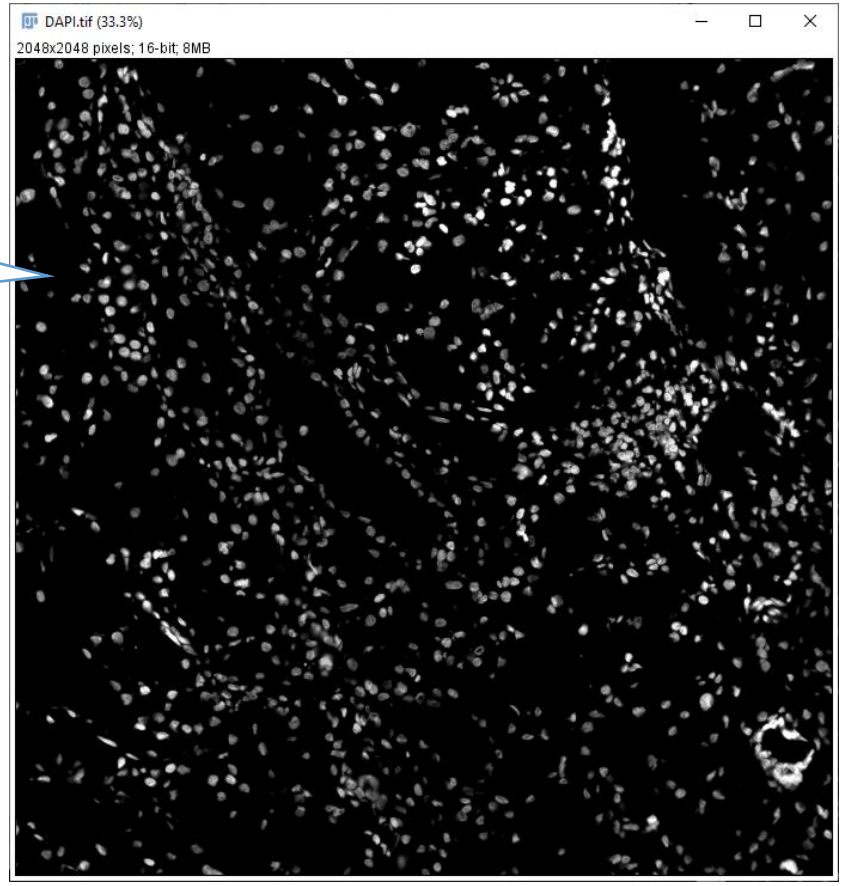
Objective bio-image analysis

- Measurements should be objective, not influenced by human interpretation



Nuclei in this image are …

… more dense than in this image.



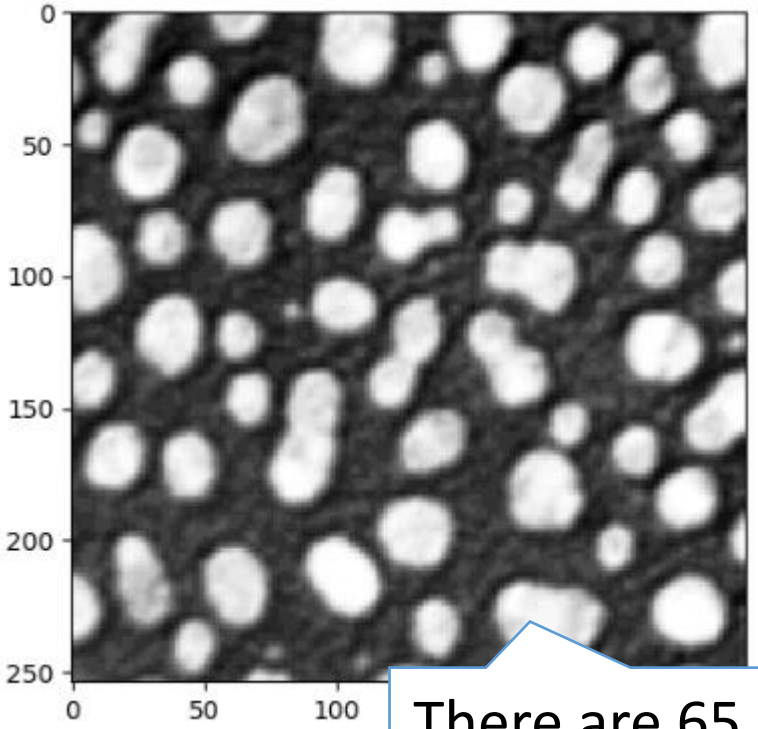
Use automation for less subjective analysis.

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

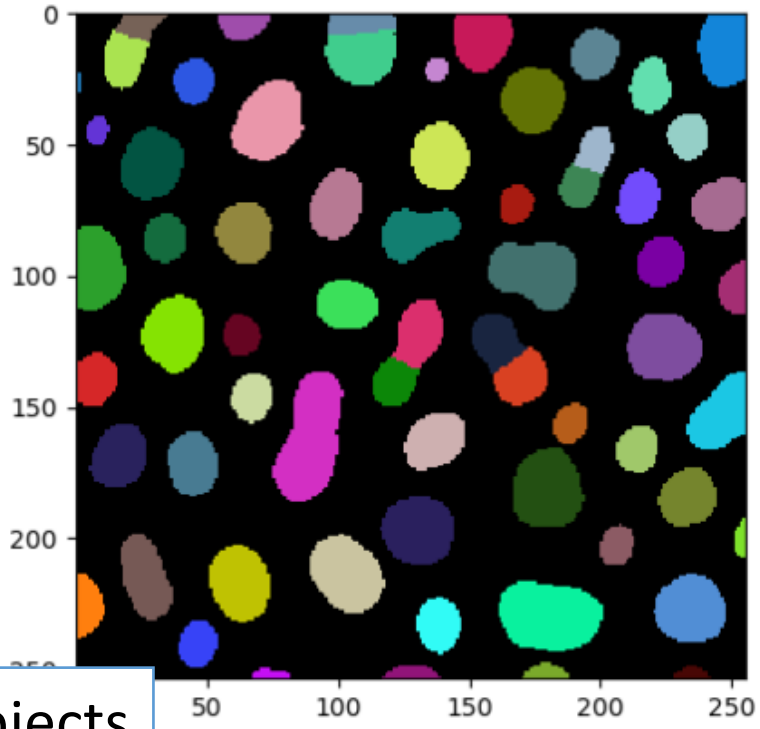
Reliable bio-image analysis

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

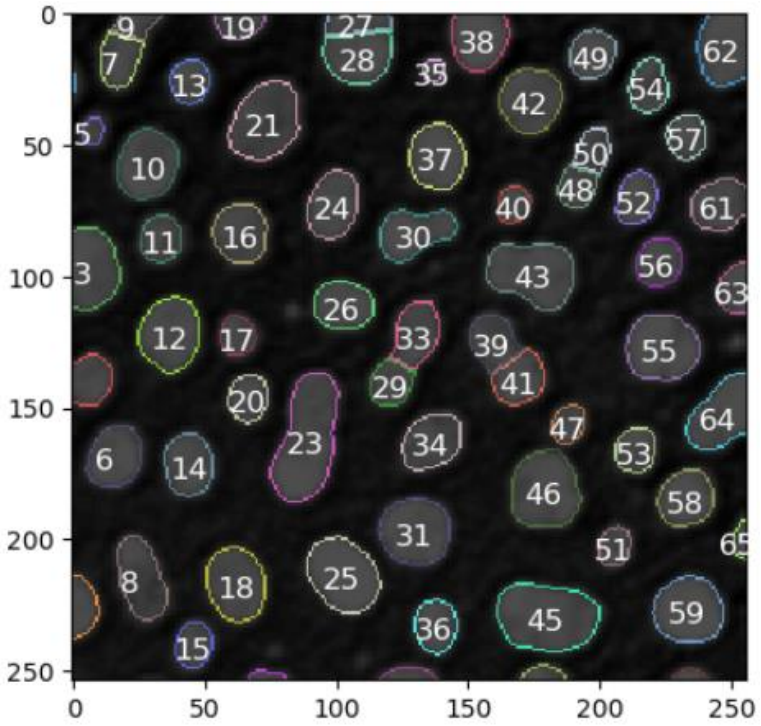
Original image



Label image



Overlay



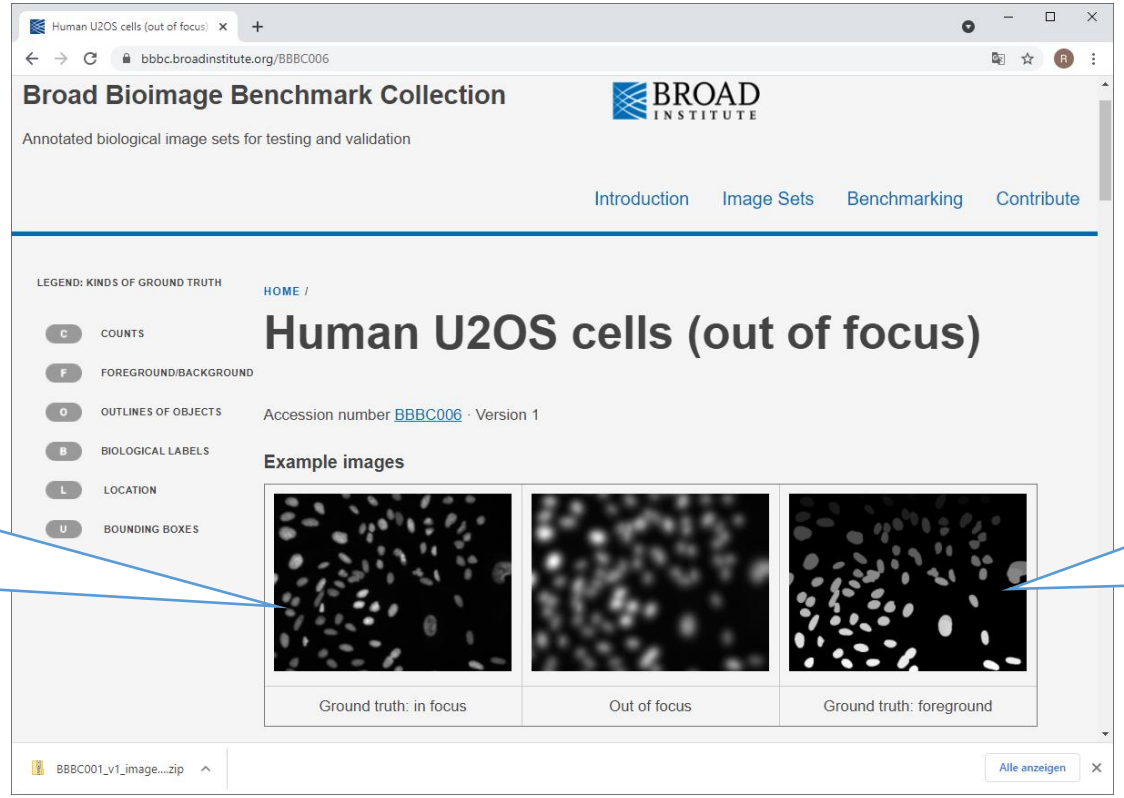
There are 65 objects in this image.

Source: M. Zoccoler & R. Haase licensed CC-BY https://haesleinhuepf.github.io/BioImageAnalysisNotebooks/60_data_visualization/overlay_text_on_image.html

Reliable bio-image analysis

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

Original image data



"Ground truth" label images

Reproducible data analysis

- Allowing others to do your experiment again.

- “The image data was analyzed with Python.”

Can you reproduce what they did?

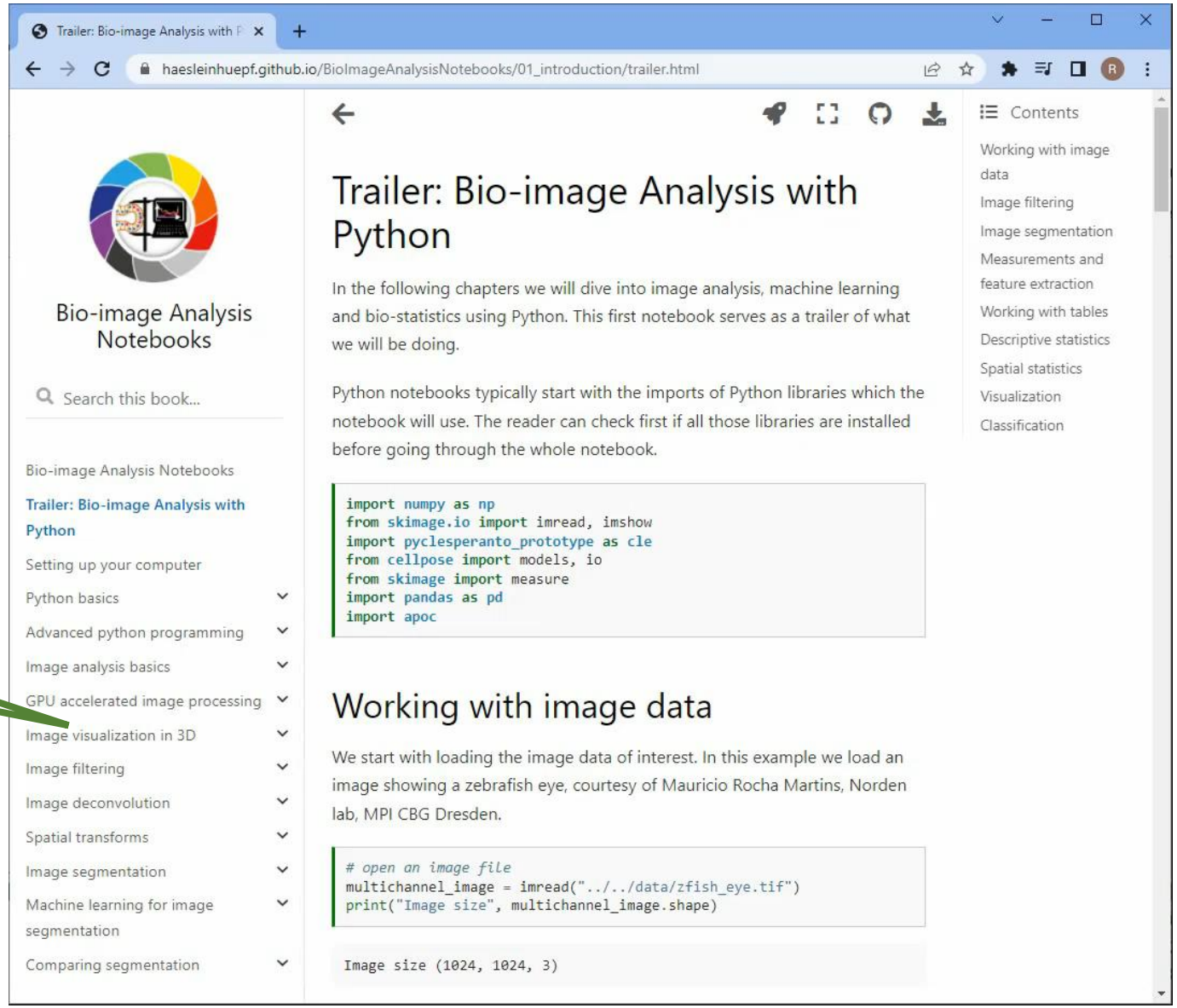
Reproducible bio-image analysis

- Allowing others to do your experiment again.

- "The image data was analyzed with Python."

Can you reproduce what they did?

Can you reproduce what they did?



Replicable bio-image analysis

- Others run the same analysis on their data and have consistent results / <u>same conclusions</u>.
- Can only be achieved if data analysis protocol was documented <u>reproducibly</u>.

- See also: *Replication crisis*
  - In Psychology (surveys)
  - In Medicine (clinical trials)
  - In Computer Science (executable code)
  - …

Open access, freely available online

Essay

Why Most Published Research Findings Are False

John P. A. Ioannidis

Summary

There is increasing concern that most current published research findings are false. The probability that a research claim is true may depend on study power and bias, the number of other studies on the same question, and, importantly, the ratio of true to no relationships among the relationships probed in each scientific field. In this framework, a research finding is less likely to be true when the studies conducted in a field are smaller; when effect sizes are smaller; when there is a greater number and lesser preselection of tested relationships; where there is greater flexibility in designs, definitions, outcomes, and analytical modes; when there is greater financial and other interest and prejudice; and when more teams are involved in a scientific field in chase of statistical significance. Simulations show that for most study designs and settings, it is more likely for

factors that influence this problem and some corollaries thereof.

Modeling the Framework for False Positive Findings

Several methodologists have pointed out [9–11] that the high rate of nonreplication (lack of confirmation) of research discoveries is a consequence of the convenient, yet ill-founded strategy of claiming conclusive research findings solely on the basis of a single study assessed by formal statistical significance, typically for a *p*-value less than 0.05. Research is not most appropriately represented and summarized by *p*-values, but, unfortunately, there is a widespread notion that medical research articles

It can be proven that most claimed research findings are false.

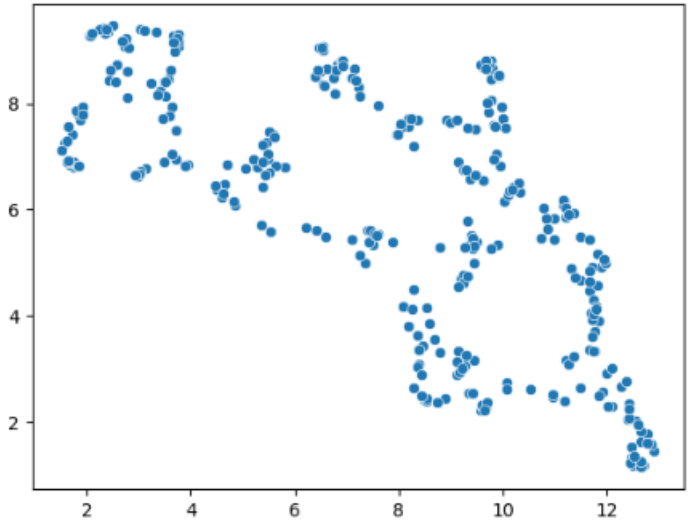
is characteristic of the field and can vary a lot depending on whether the field targets highly likely relationships or searches for only one or a few true relationships among thousands and millions of hypotheses that may be postulated. Let us also consider, for computational simplicity, circumscribed fields where either there is only one true relationship (among many that can be hypothesized) or the power is similar to find any of the several existing true relationships. The pre-study probability of a relationship being true is $R/(R+1)$. The probability of a study finding a true relationship reflects the power $1-\beta$ (one minus the Type II error rate). The probability of claiming a relationship when none truly exists reflects the Type I error rate, α. Assuming that c relationships are being probed in the field, the expected values of the 2×2 table are given in Table 1. After a research finding has been claimed based on

https://en.wikipedia.org/wiki/Replication_crisis

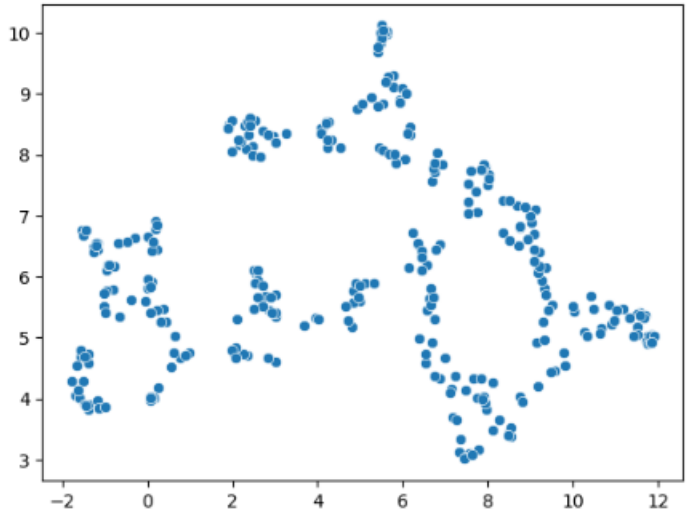
Repeatable data analysis

- In wet-lab experiments, samples may get destroyed while executing the experiment.
- Repeatability is a property of the experiment / algorithm. You cannot improve repeatability by better documentation.

```
[11]: reducer = umap.UMAP()
      embedding2 = reducer.fit_transform(scaled_statistics)

      seaborn.scatterplot(x=embedding2[:, 0],
                          y=embedding2[:, 1])

[11]: <AxesSubplot: >
```




```
[12]: reducer = umap.UMAP()
      embedding2 = reducer.fit_transform(scaled_statistics)

      seaborn.scatterplot(x=embedding2[:, 0],
                          y=embedding2[:, 1])

[12]: <AxesSubplot: >
```



https://haesleinhuepf.github.io/BioImageAnalysisNotebooks/47_clustering/umap.html?highlight=umap#a-note-on-repeatability

Bio-image Analysis: good scientific practice

PoL Physics of Life TU Dresden

Preface — Introduction to Bioima

bioimagebook.github.io/chapters/0-preamble/preface/preface.html

Introduction to Bioimage Analysis

Front matter

Acknowledgements

License

Disclaimer

When I'm confronted by an image analysis problem, my goal is never really to find the *right* way to do the analysis. That generally doesn't exist.

Instead, my goal is to find the *least wrong* way to do the analysis – and to be able to understand and explain whatever lingering limitations and biases can't be entirely overcome. It can be frustrating, I still don't feel terribly good at it, but it is – in its own strange way – kind of *enjoyable*. There's always something new to learn, and some new angle from which to look at the problem. And each new angle can help us wring more drops of knowledge out of our data.

My hope is that this book will help introduce others find the weird, frustrating pleasure of thinking more deeply about scientific images. Through this, I hope it might make a small contribution towards helping us do image analysis a bit better.

Pete Bankhead

April 2022

Previous
Changelog

Next
How to read this book

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**
    - Enabling others to re-do the experiment. For this, documentation is crucial!
  - **Replicability**
    - Others *do* execute the same analysis, potentially on other data, and see consistent results.
  - **Repeatable**
    - We can do the same experiment twice under the *same conditions* and get the same measurements.

Quiz

INSTITUTE OF PHYSICS PUBLISHING EUROPEAN JOURNAL OF PHYSICS
Eur. J. Phys. 23 (2002) 21–26 PII: S0143-0807(02)26048-1

Demonstration of the exponential decay law using beer froth

A Leike
Ludwig–Maximil
E-mail: leike@th

Received 22 J
Published 17
Online at stac

Abstract
The volume o
to demonstrat
depends on th
beers. The an
commonly us
data, paramet

In the following, the demonstration is described in detail. In our experiment, a cylindrical beer mug with a diameter of 7.2 cm was filled with beer immediately after opening the bottle. The temperature of the beer was 19 °C.

The froth appears while filling the mug with the beer. The froth reaches its maximum height within a few seconds. This indicates that the typical time scale of the expansion of the froth is a few seconds. On the other hand, the froth lasts for a few minutes (see table 1). Therefore, the time scale for the decay is a few minutes. The two time scales are very different. We therefore assume that a few seconds after the time where the froth reaches its maximum height only the decay plays a significant role.

We began with the measurement at the time where the froth dropped to a certain initial height $h^{\exp}(0) = h(0)$. The error $\Delta h^{\exp}(0)$ in the measurement of $h(0)$ was estimated to be 2 mm.

We investigated three different beers. With every beer, the experiment was repeated several times. We performed seven experiments with Erdinger Weissbier (the author's favourite!), four experiments with Augustinerbräu München and four experiments with Budweiser Budvar. Our data are shown in table 1. The entries for $h^{\exp}(t_i)$ are obtained by averaging over all individual measurements at time t_i. To obtain the errors $\Delta h^{\exp}(t_i)$ of the measurements $h^{\exp}(t_i)$, we first

Is this experiment reproducible?

Yes

No

Quiz

INSTITUTE OF PHYSICS PUBLISHING EUROPEAN JOURNAL OF PHYSICS
Eur. J. Phys. 23 (2002) 21–26 PII: S0143-0807(02)26048-1

Demonstration of the exponential decay law using beer froth

A Leike
Ludwig–Maximil
E-mail: leike@th

Received 22 J
Published 17
Online at stac

Abstract
The volume o
to demonstrat
depends on th
beers. The an
commonly us
data, paramet

In the following, the demonstration is described in detail. In our experiment, a cylindrical beer mug with a diameter of 7.2 cm was filled with beer immediately after opening the bottle. The temperature of the beer was 19 °C.

The froth appears while filling the mug with the beer. The froth reaches its maximum height within a few seconds. This indicates that the typical time scale of the expansion of the froth is a few seconds. On the other hand, the froth lasts for a few minutes (see table 1). Therefore, the time scale for the decay is a few minutes. The two time scales are very different. We therefore assume that a few seconds after the time where the froth reaches its maximum height only the decay plays a significant role.

We began with the measurement at the time where the froth dropped to a certain initial height $h^{\exp}(0) = h(0)$. The error $\Delta h^{\exp}(0)$ in the measurement of $h(0)$ was estimated to be 2 mm.

We investigated three different beers. With every beer, the experiment was repeated several times. We performed seven experiments with Erdinger Weissbier (the author's favourite!), four experiments with Augustinerbräu München and four experiments with Budweiser Budvar. Our data are shown in table 1. The entries for $h^{\exp}(t_i)$ are obtained by averaging over all individual measurements at time t_i. To obtain the errors $\Delta h^{\exp}(t_i)$ of the measurements $h^{\exp}(t_i)$, we first

Is this experiment repeatable?

Yes

No

Quiz

INSTITUTE OF PHYSICS PUBLISHING EUROPEAN JOURNAL OF PHYSICS
Eur. J. Phys. 23 (2002) 21–26 PII: S0143-0807(02)26048-1

Demonstration of the exponential decay law using beer froth

A Leike
Ludwig–Maximil
E-mail: leike@th

Received 22 J
Published 17
Online at stac

Abstract
The volume o
to demonstrat
depends on th
beers. The an
commonly us
data, paramet

In the following, the demonstration is described in detail. In our experiment, a cylindrical beer mug with a diameter of 7.2 cm was filled with beer immediately after opening the bottle. The temperature of the beer was 19 °C.

The froth appears while filling the mug with the beer. The froth reaches its maximum height within a few seconds. This indicates that the typical time scale of the expansion of the froth is a few seconds. On the other hand, the froth lasts for a few minutes (see table 1). Therefore, the time scale for the decay is a few minutes. The two time scales are very different. We therefore assume that a few seconds after the time where the froth reaches its maximum height only the decay plays a significant role.

We began with the measurement at the time where the froth dropped to a certain initial height $h^{\exp}(0) = h(0)$. The error $\Delta h^{\exp}(0)$ in the measurement of $h(0)$ was estimated to be 2 mm.

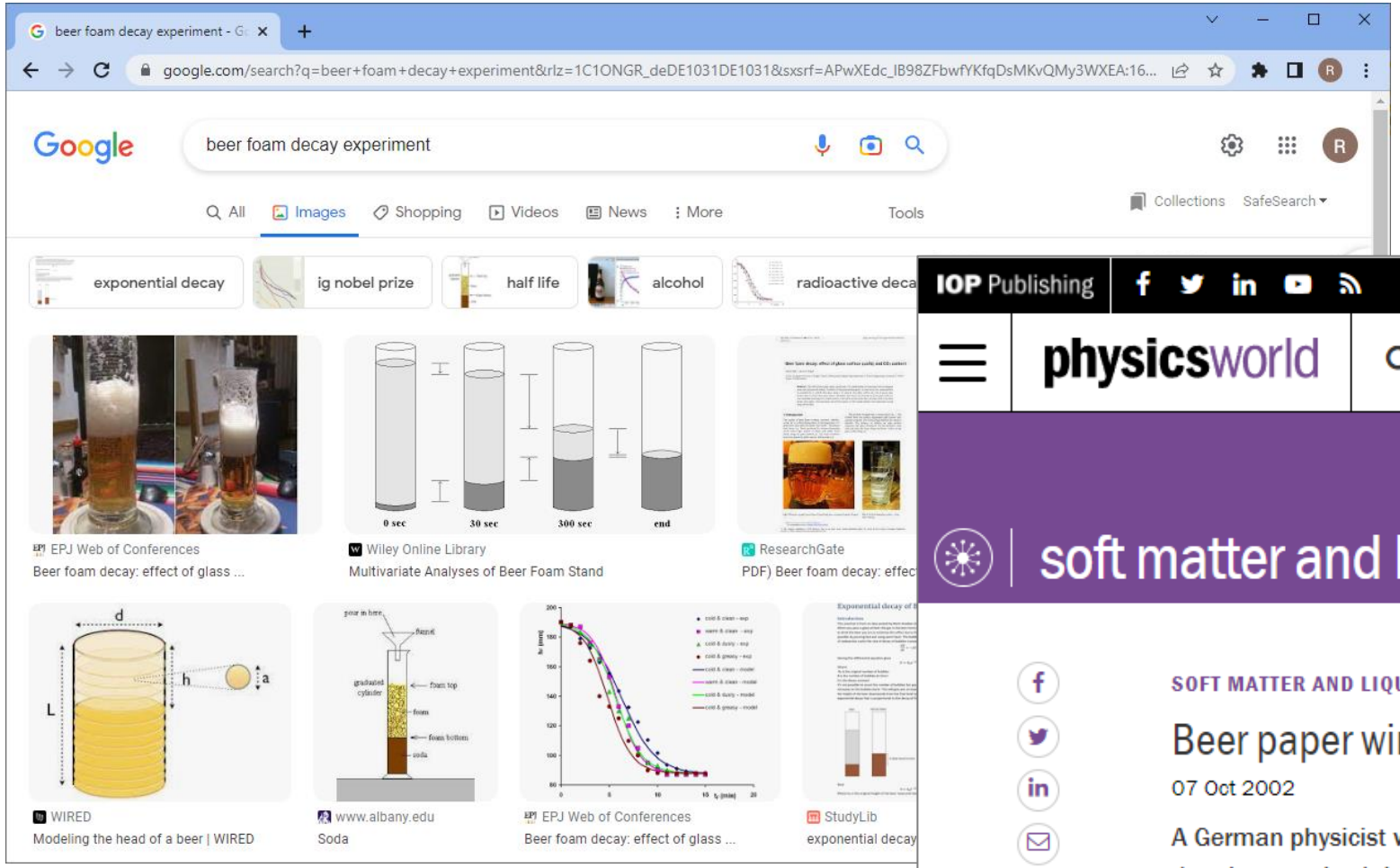
We investigated three different beers. With every beer, the experiment was repeated several times. We performed seven experiments with Erdinger Weissbier (the author's favourite!), four experiments with Augustinerbräu München and four experiments with Budweiser Budvar. Our data are shown in table 1. The entries for $h^{\exp}(t_i)$ are obtained by averaging over all individual measurements at time t_i. To obtain the errors $\Delta h^{\exp}(t_i)$ of the measurements $h^{\exp}(t_i)$, we first

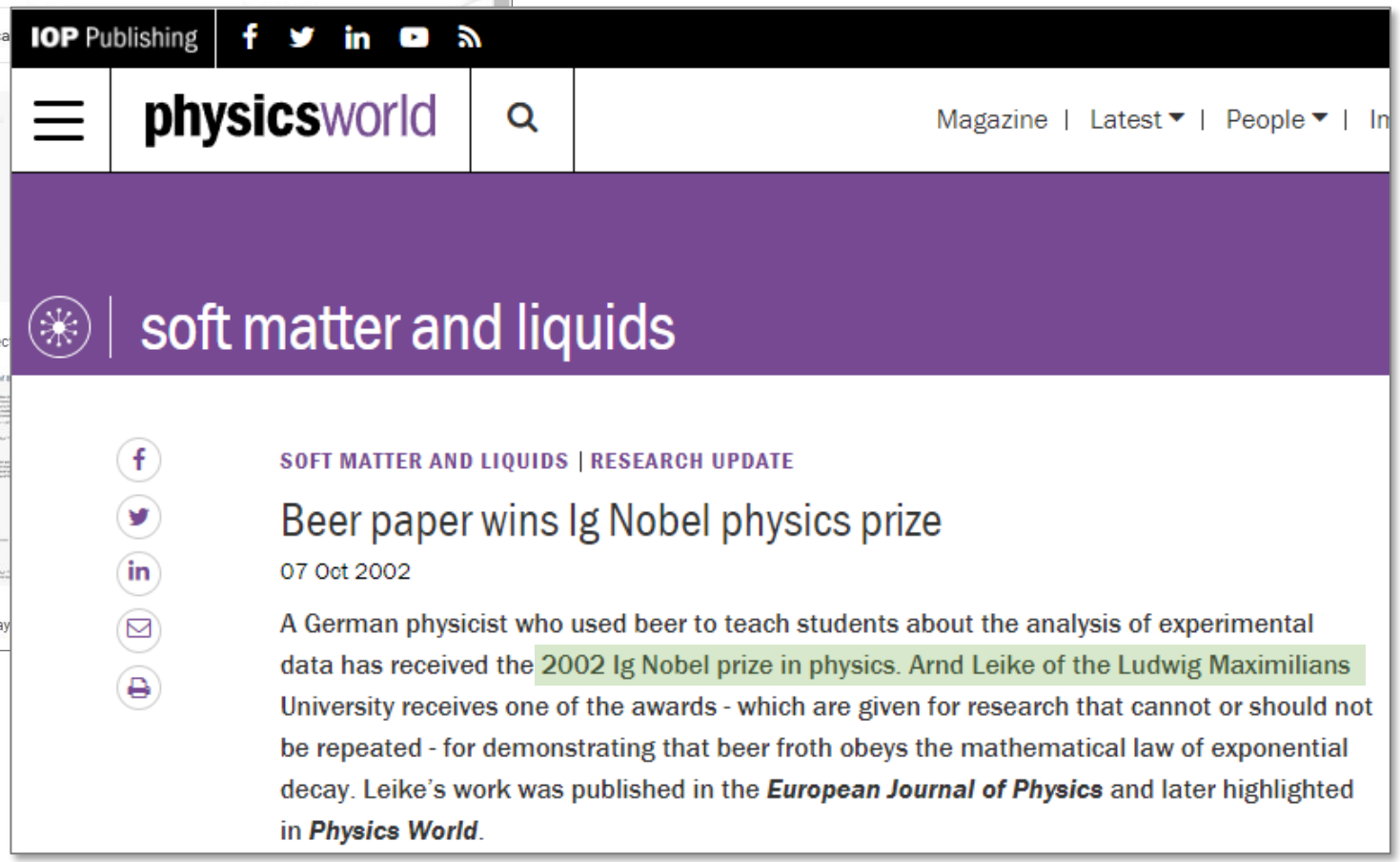
Do you think this experiment was replicated?

Yes

No

Quiz

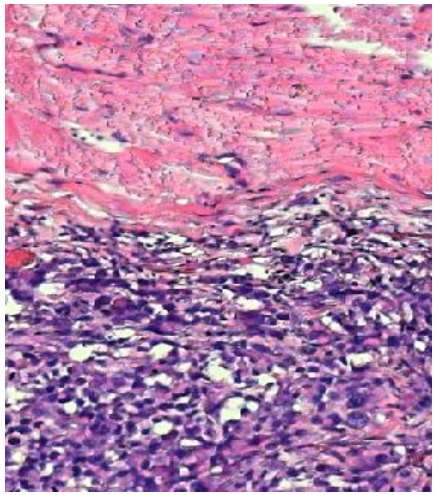




https://physicsworld.com/a/beer-paper-wins-ig-nobel-physics-prize/

Common topics

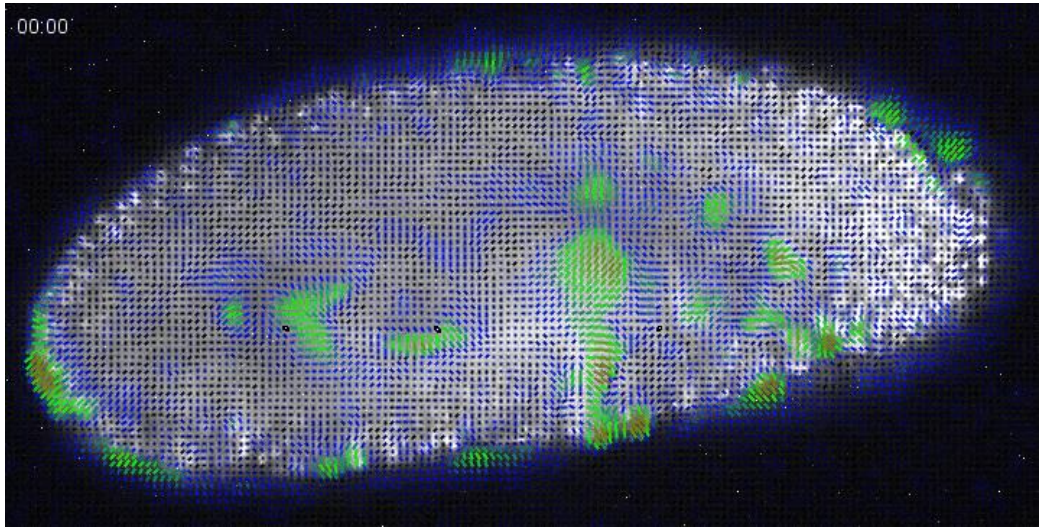
- Typical questions bio-image analysts deal with
  - Is signal intensity different under varying conditions?
  - How many cells are in my image?
  - How high is cell density?
    - ➢ Bio-statistics / medicine / disease staging
  - How are different tissues characterized?
    - ➢ Machine learning



muscle, normal tissue

squamous-cell carcinoma

- 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?



Hypothesis-driven quantitative biology

PoL Physics of Life TU Dresden

- Hypothesis: Cell shape can be influenced by modifying X.
- Null-Hypothesis: Circularity of modified cells is similar to cells in the control group.

- Sample preparation
- Imaging

Shall we use a different microscope?

Should we use a different segmentation algorithm?

- Cell segmentation
- Circularity measurement

Is circularity the right parameter to measure?

- Statistics

Hypothesis generating quantitative biology

PoL Physics of Life TU Dresden

- ~~Hypothesis: Cell shape can be influenced by modifying X.~~
- Question: Which image-derived parameter is influenced when modifying X?
  - Sample preparation
  - Imaging
  - Cell segmentation algorithm A, algorithm B, algorithm C
  - Measurement of circularity, solidity, elongation, extend, texture, intensity, topology …
  - Statistics

Which segmentation algorithms allow measurements that show a relationship with X?

Why?

Which parameter shows any relationship with X?

PoL Physics of Life TU Dresden

CENTER FOR SYSTEMS BIOLOGY DRESDEN

Python Programming

Robert Haase

April 2023

@haesleinhuepf

Data science with python

- Why Python?

Because copy&paste works so great.

stardist/stardist: StardDist - Object

github.com

README.md

pypi package 0.6.2

forum 86 topics

StarDist - C

Star-conve

This repository contain

object detection for 2

- Uwe Schmidt, Ma
  Broaddus, and Ge
  Cell Detection wit
  International Con
  Image Computing
  Intervention (MIC
  September 2018.

stardist/stardist: StardDist - Object

github.com/stardist/stardist

README.md

You can access these pretrained models from

`stardist.models.StarDist2D`

```
from
# pri
StarD
# cre
model
```

Annot

To train
annotat
correspo
labeled
with 0).
among
options:
having a

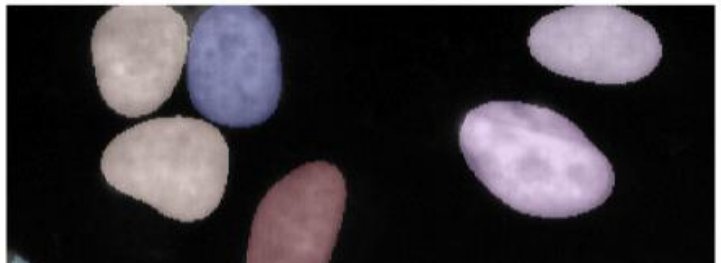
stardist/3_prediction.ipynb at ma

github.com/stardist/stardist/blob/master/examples/2D/3_ diction.ipynb

remaining polygons

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[...,0], clim=(0,1), cmap=
plt.imshow(labels, cmap=lbl_cmap, alpha=0.5)
plt.axis('off');
```

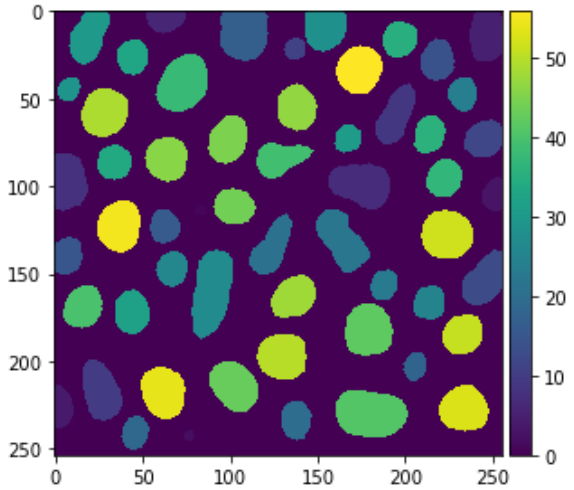


In [3]:
```
# normalize image
from csbdeep.utils import normalize
normalized_image = normalize(image, 1,99.8, axis=(0,1))

# 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
```

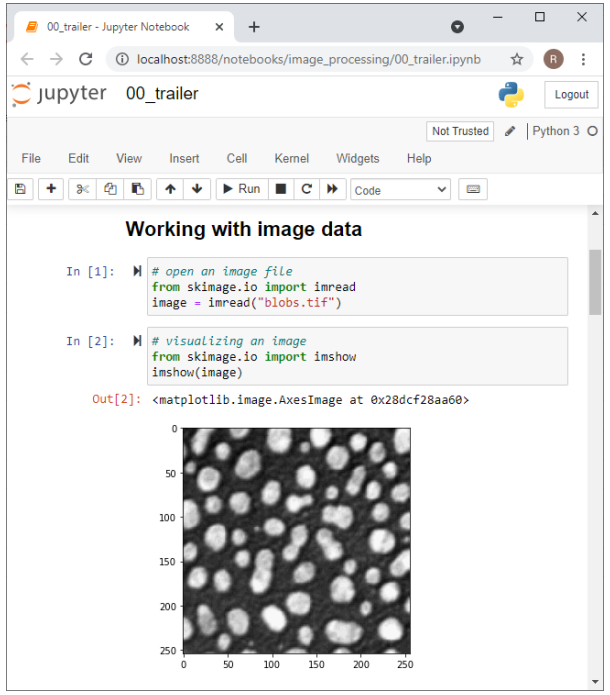
]: `<matplotlib.image.AxesImage at 0x28dd9f991c0>`

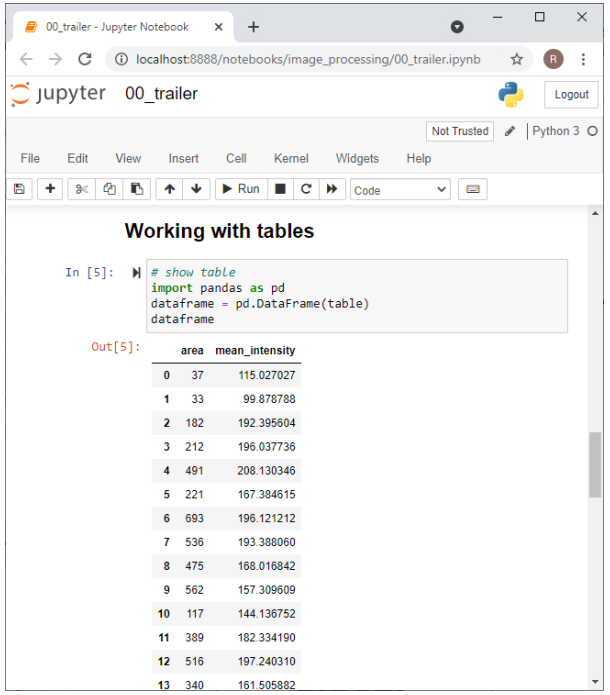


https://github.com/stardist/stardist

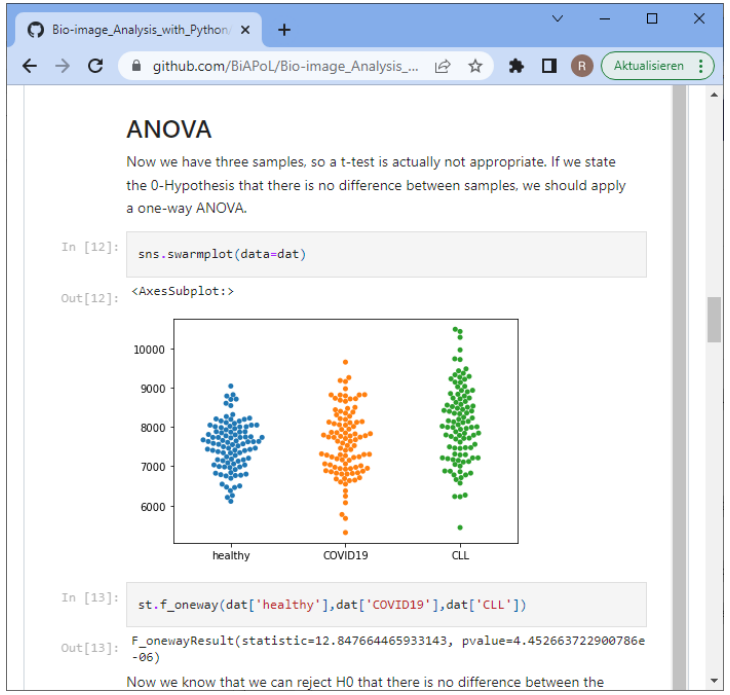
Python

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





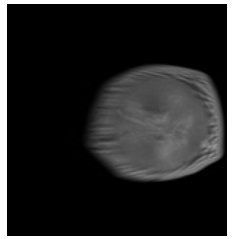
	area	mean_intensity
0	37	115.027027
1	33	99.878788
2	182	192.395604
3	212	196.037736
4	491	208.130346
5	221	167.384615
6	693	196.121212
7	536	193.388060
8	475	168.016842
9	562	157.309609
10	117	144.136752
11	389	182.334190
12	516	197.240310
13	340	161.505882



How to eat a banana the computational way using Python

PoL Physics of Life TU Dresden

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

```python
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))
```


Image data source: Nasreddin Abolmaali, TU Dresden

PoL Physics of Life TU Dresden

CENTER FOR SYSTEMS BIOLOGY DRESDEN

Python
Data structures

Robert Haase

April 2023

@haesleinhuepf

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)
```

```
8
```

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
```

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

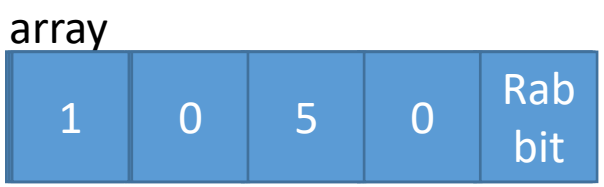
Arrays

- Arrays are variables, where you can store multiple values

Give me a “0”, five times!

```
array = [0] * 5
```

Computer memory



Subsets

```python
# Arrays
numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
print(numbers)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

- Creating subsets of arrays

Start

End

```python
subset = numbers[2:4]
print(subset)
```

```
[2, 3]
```

Step

```python
subset_with_gaps = arr[1:8:2]
print(subset_with_gaps)
```

```
[1, 3, 5, 7]
```

data[start:stop:step]

Indexing, cropping, subsets

- “Indexing” is addressing certain elements in arrays. The first element is “0” away from the start.

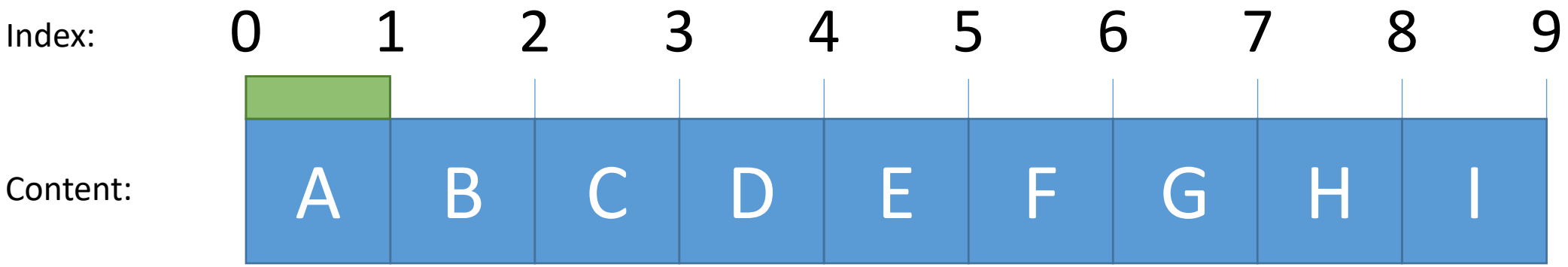
```
data = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I']
```

Index:	0	1	2	3	4	5	6	7	8	9
Content:	A	B	C	D	E	F	G	H	I	

Indexing, cropping, subsets

- “Indexing” is addressing certain elements in arrays. The first element is “0” away from the start.

```
data = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I']
```



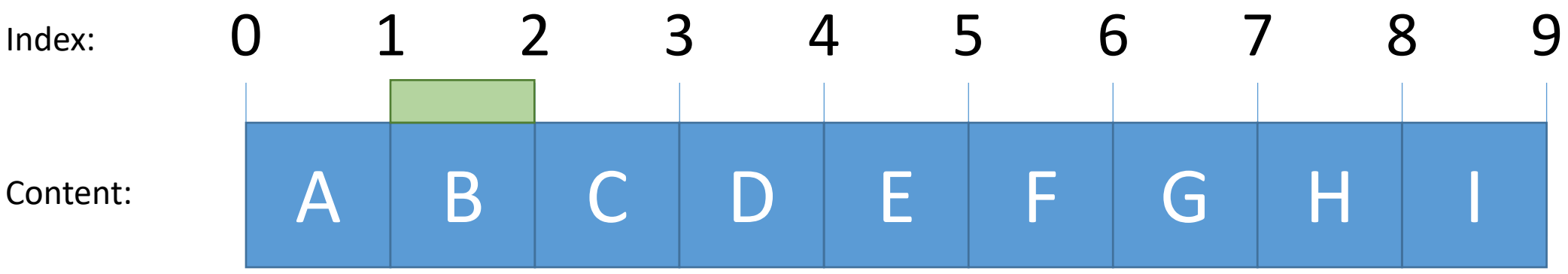

```
data[0]
```

```
'A'
```

Indexing, cropping, subsets

- “Indexing” is addressing certain elements in arrays. The first element is “0” away from the start.

```
data = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I']
```




```
data[0]
```

```
'A'
```

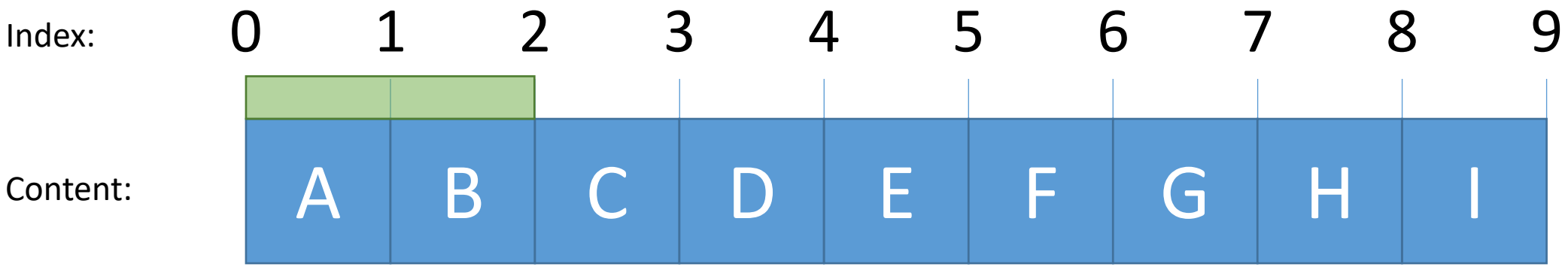
```
data[1]
```

```
'B'
```

Indexing, cropping, subsets

- “Indexing” is addressing certain elements in arrays. The first element is “0” away from the start.

```
data = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I']
```




```
data[0]
```

```
'A'
```

```
data[1]
```

```
'B'
```

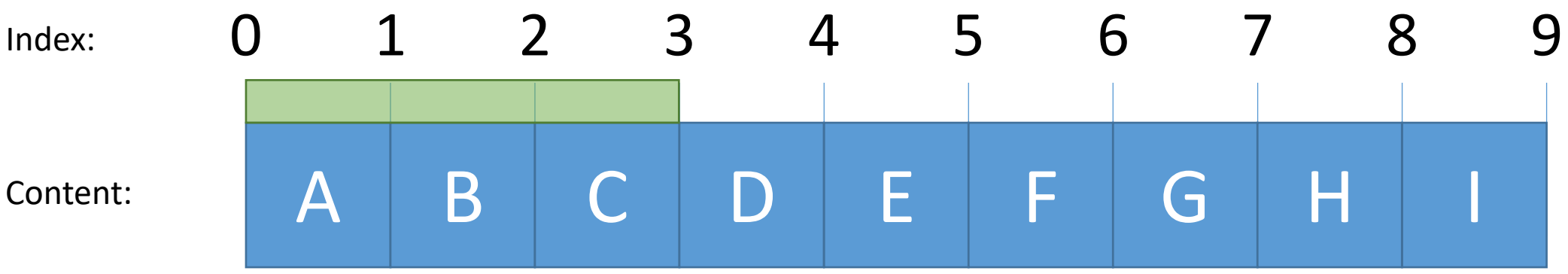
```
data[0:2]
```

```
['A', 'B']
```

Indexing, cropping, subsets

- “Indexing” is addressing certain elements in arrays. The first element is “0” away from the start.

```
data = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I']
```




```
data[0]
```

```
'A'
```

```
data[1]
```

```
'B'
```

```
data[0:2]
```

```
['A', 'B']
```

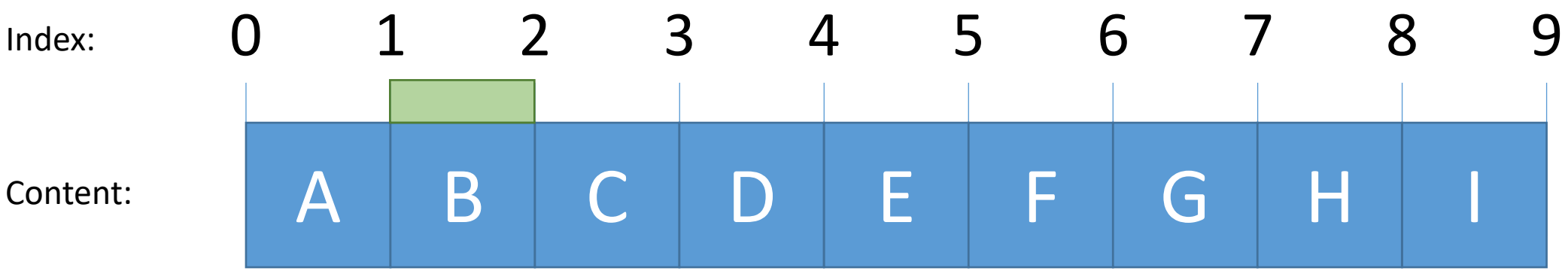
```
data[0:3]
```

```
['A', 'B', 'C']
```

Indexing, cropping, subsets

- “Indexing” is addressing certain elements in arrays. The first element is “0” away from the start.

```
data = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I']
```




```
data[0]
```

```
'A'
```

```
data[1]
```

```
'B'
```

```
data[0:2]
```

```
['A', 'B']
```

```
data[0:3]
```

```
['A', 'B', 'C']
```

```
data[1:2]
```

```
['B']
```

Indexing, cropping, subsets

- “Indexing” is addressing certain elements in arrays. The first element is “0” away from the start.

```
data = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I']
```

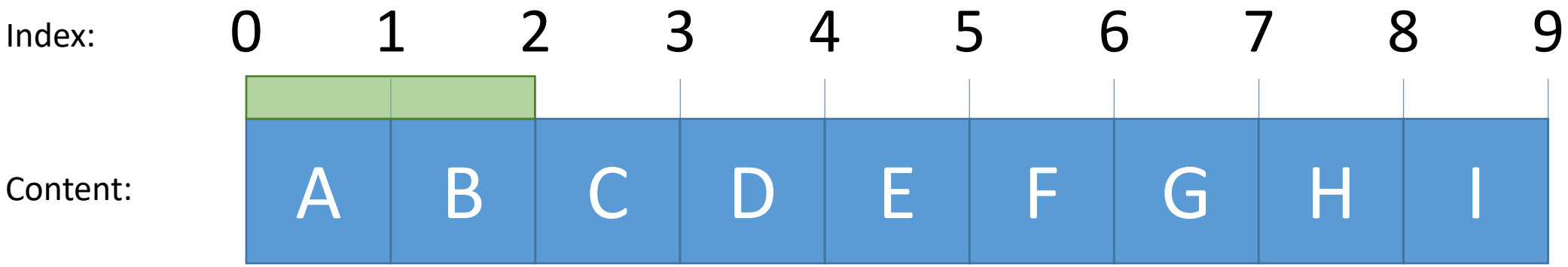
Index:	0	1	2	3	4	5	6	7	8	9
Content:	A	B	C	D	E	F	G	H	I	

`data[0]`	`data[1]`	`data[0:2]`	`data[0:3]`	`data[1:2]`	`len(data)`
`'A'`	`'B'`	`['A', 'B']`	`['A', 'B', 'C']`	`['B']`	`9`

Indexing, cropping, subsets

- You can leave start and end out when specifying index ranges

```
data = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I']
```



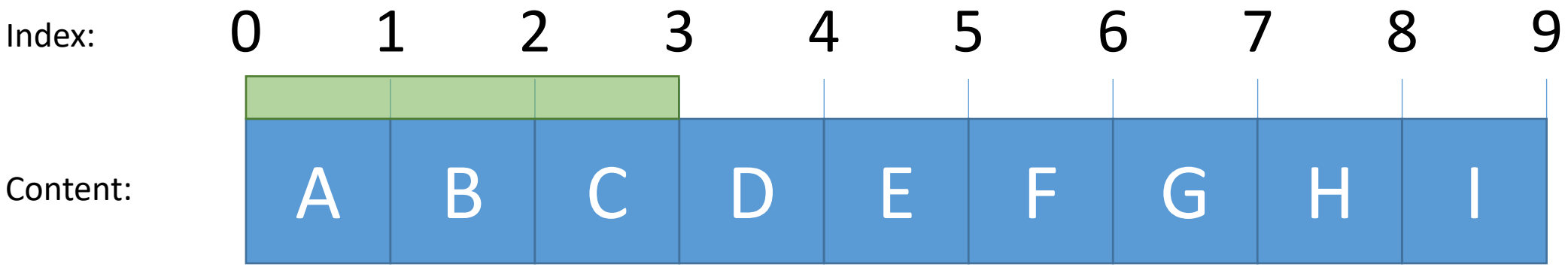

```
data[:2]
```

```
['A', 'B']
```

Indexing, cropping, subsets

- You can leave start and end out when specifying index ranges

```
data = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I']
```




```
data[:2]
```

```
['A', 'B']
```

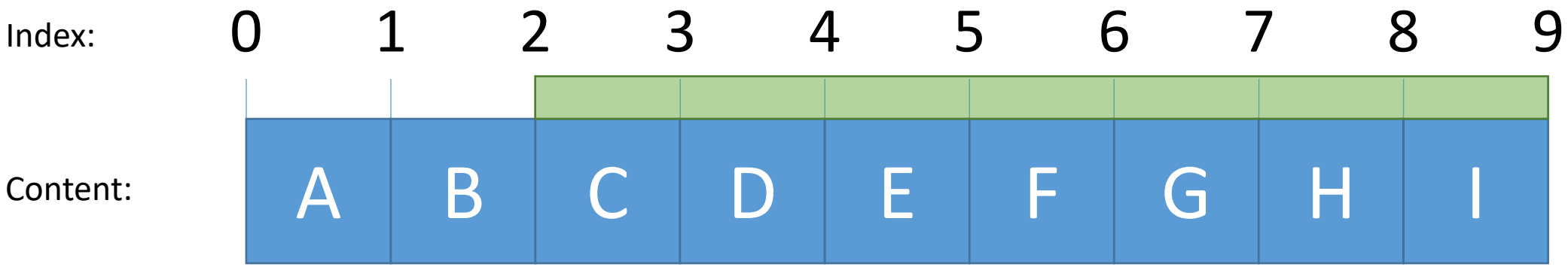
```
data[:3]
```

```
['A', 'B', 'C']
```

Indexing, cropping, subsets

- You can leave start and end out when specifying index ranges

```
data = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I']
```




```
data[:2]
```

```
['A', 'B']
```

```
data[:3]
```

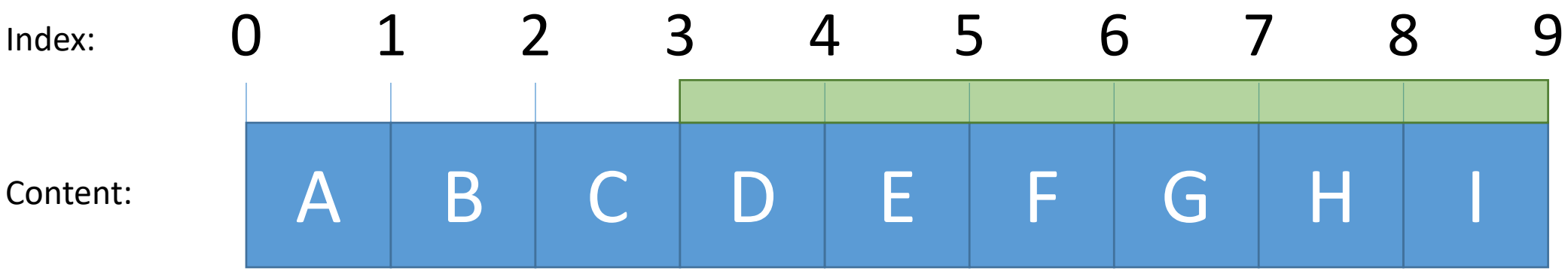
```
['A', 'B', 'C']
```

```
data[2:]
```

```
['C', 'D', 'E', 'F', 'G', 'H', 'I']
```

Indexing, cropping, subsets

```
data = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I']
```



What's the output of

```
data[3:]
```

?

ABC

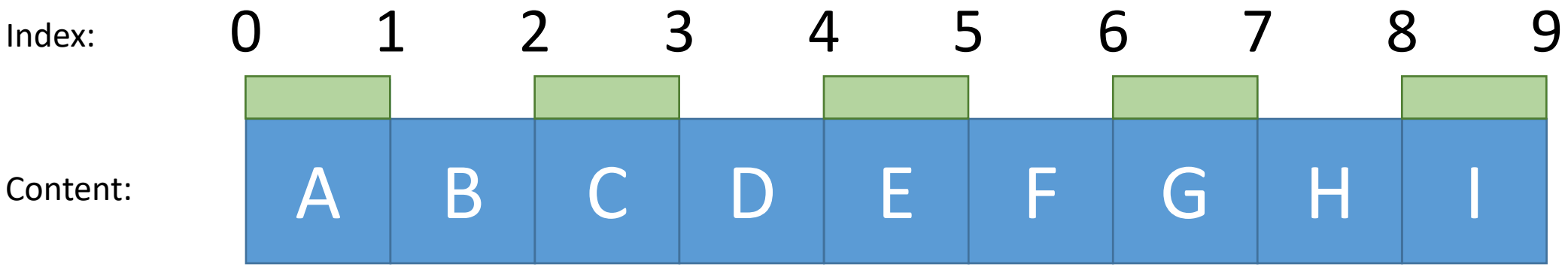
CDEFGHI

DEFGHI

Indexing, cropping, subsets

- The step-size allows skipping elements

```
data = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I']
```



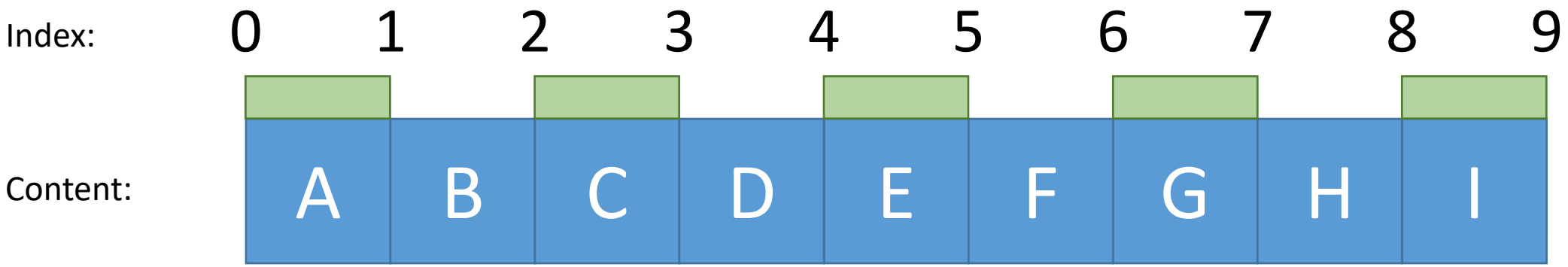

```
data[0:10:2]
```

```
['A', 'C', 'E', 'G', 'I']
```

Indexing, cropping, subsets

- The step-size allows skipping elements

```
data = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I']
```




```
data[0:10:2]
```

```
['A', 'C', 'E', 'G', 'I']
```

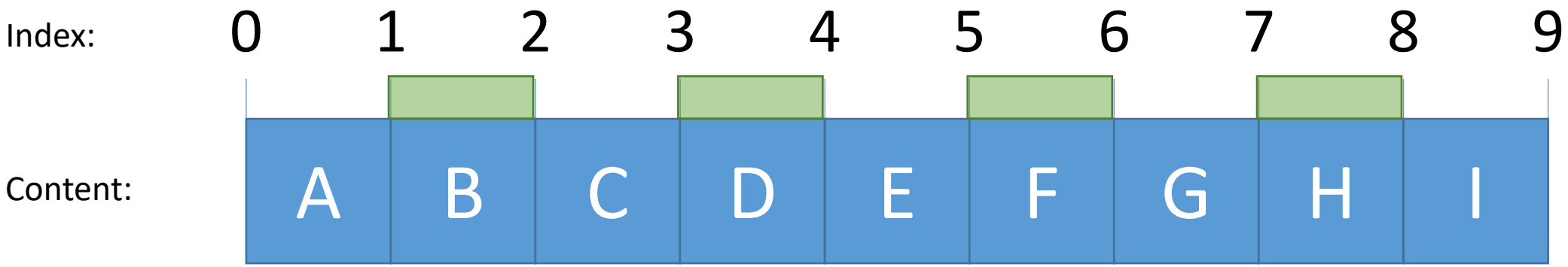
```
data[::2]
```

```
['A', 'C', 'E', 'G', 'I']
```

Indexing, cropping, subsets

- The step-size allows skipping elements

```
data = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I']
```




```
data[0:10:2]
```

```
['A', 'C', 'E', 'G', 'I']
```

```
data[::2]
```

```
['A', 'C', 'E', 'G', 'I']
```

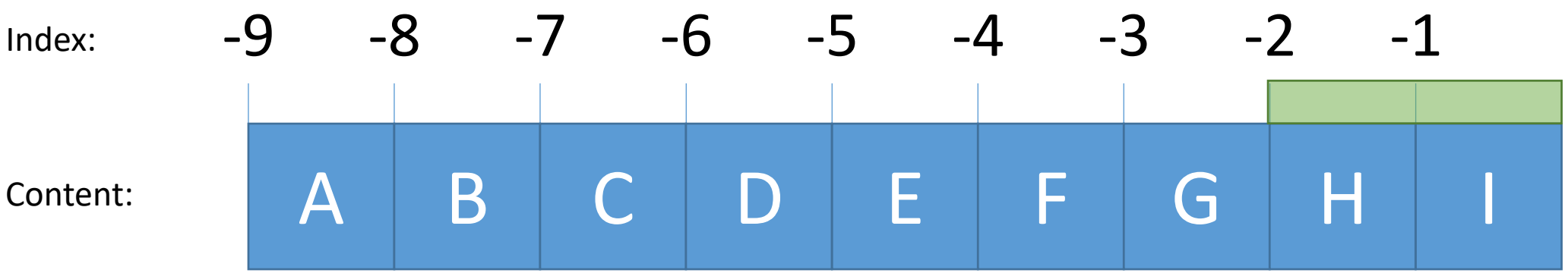
```
data[1::2]
```

```
['B', 'D', 'F', 'H']
```

Indexing, cropping, subsets

- Indexing also works with negative indices

```
data = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I']
```



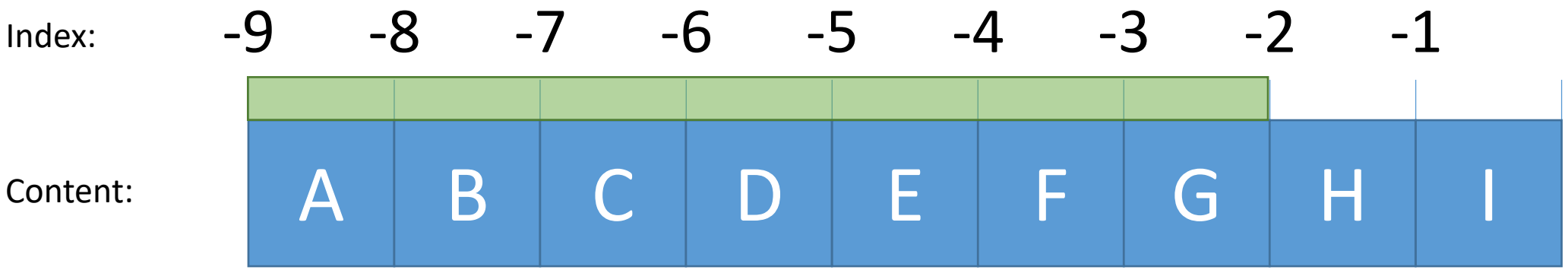

```
data[-2:]
```

```
['H', 'I']
```

Indexing, cropping, subsets

- Indexing also works with negative indices

```
data = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I']
```




```
data[-2:]
```

```
['H', 'I']
```

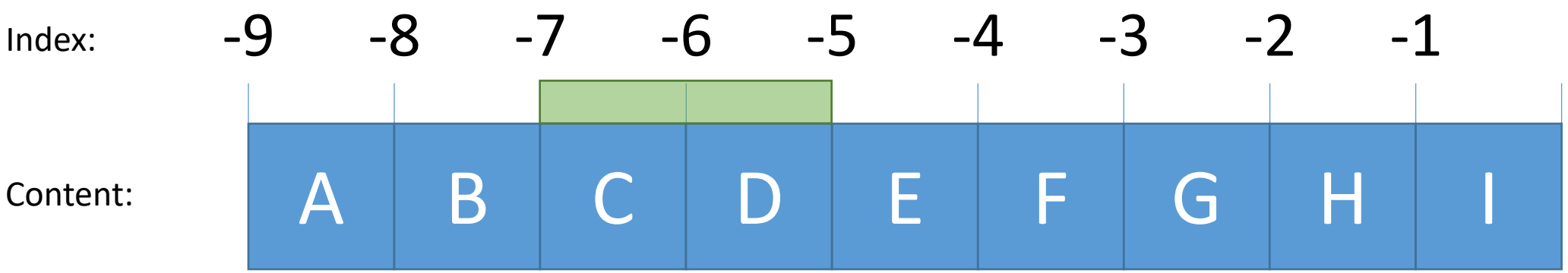
```
data[:-2]
```

```
['A', 'B', 'C', 'D', 'E', 'F', 'G']
```

Indexing, cropping, subsets

- Indexing also works with negative indices

```
data = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I']
```




```
data[-2:]
```

```
['H', 'I']
```

```
data[:-2]
```

```
['A', 'B', 'C', 'D', 'E', 'F', 'G']
```

```
data[-7:-5]
```

```
['C', 'D']
```

Indexing, cropping, subsets

- Indexing also works with negative indices

```
data = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I']
```

Index:	-9	-8	-7	-6	-5	-4	-3	-2	-1
Content:	A	B	C	D	E	F	G	H	I

```
data[-2:]
```

```
['H', 'I']
```

```
data[:-2]
```

```
['A', 'B', 'C', 'D', 'E', 'F', 'G']
```

```
data[-7:-5]
```

```
['C', 'D']
```

```
data[-5:-7]
```

```
[]
```

Indexing, cropping, subsets

- Negative stepping also works

```
data = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I']
```

Index:	0	1	2	3	4	5	6	7	8	9
Content:	A	B	C	D	E	F	G	H	I	

```
data[::-1]
```

```
['I', 'H', 'G', 'F', 'E', 'D', 'C', 'B', 'A']
```

Quiz

```
data = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I']
```

Index:	0	1	2	3	4	5	6	7	8	9
Content:	A	B	C	D	E	F	G	H	I	

What's the output of

```
data[::-2]
```

?

IGECA

HGFEDCBA

GFEDCBA

Arrays in Python

- Modifying array elements

```
numbers = [0, 1, 2, 3, 4]

# write in one array element
numbers[1] = 5

print(numbers)
```

```
[0, 5, 2, 3, 4]
```

Note: The first element has index 0!

- Creating arrays of defined size

What?

How many?

```
zeros = [0] * 10
print(zeros)
```

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

- Concatenating arrays

```
ones = [1, 1, 1]
twos = [2, 2, 2, 2]

# concatenate arrays
numbers = ones + twos

print(numbers)
```

```
[1, 1, 1, 2, 2, 2, 2]
```

+ means appending

Arrays: Lists versus Tuples

- Lists can be modified

```
measurements = [5.5, 6.3, 7.2, 8.0, 8.8]
```

```
measurements[1] = 25
```

```
measurements.append(10.2)
```

```
measurements
```

```
]: [5.5, 25, 7.2, 8.0, 8.8, 10.2]
```

- Tuples not

Note: round brackets

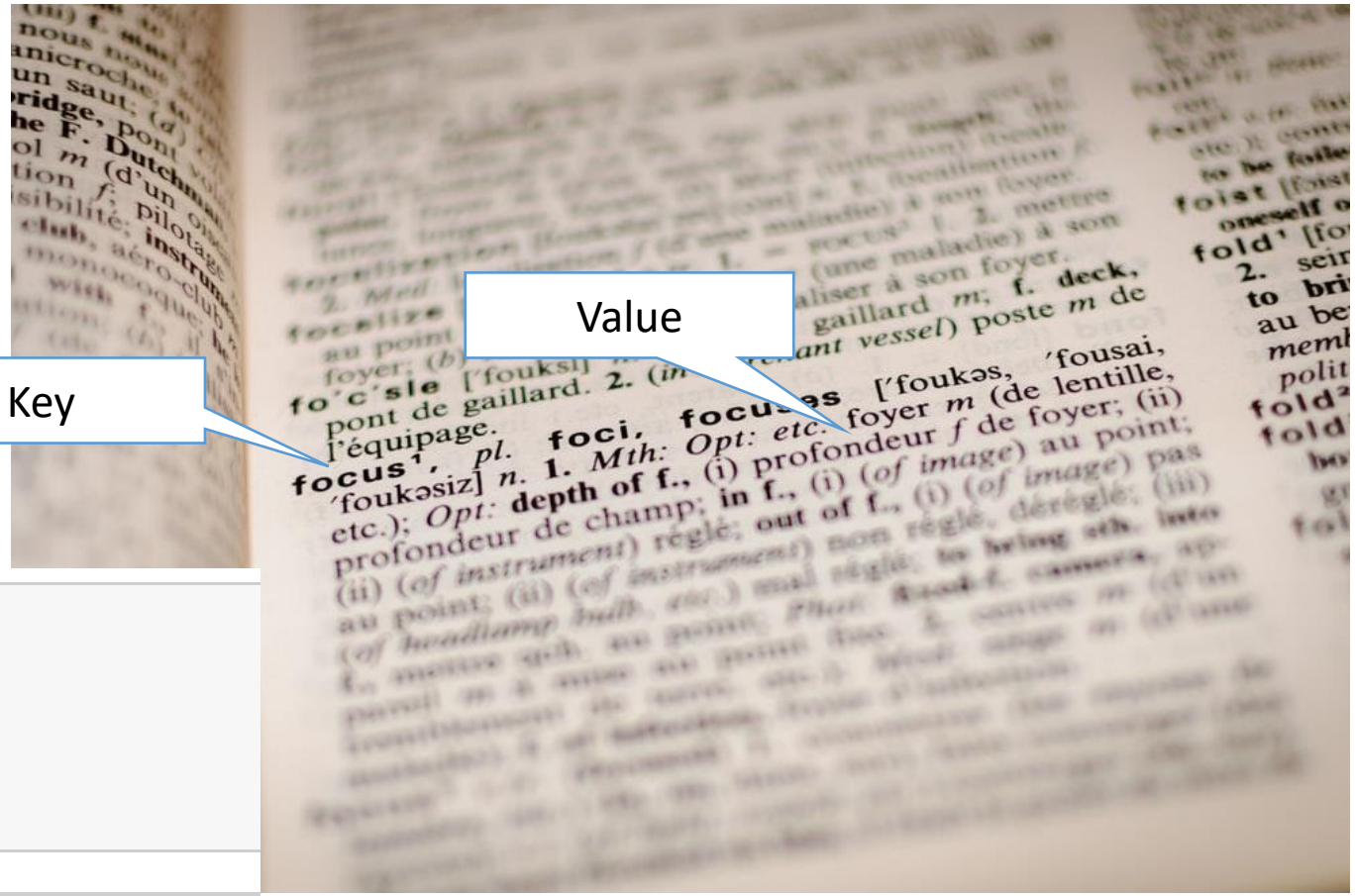
```
immutable = (4, 3, 7.8)
```

```
immutable[1] = 5
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-49-a01b13633c23> in <module>
----> 1 immutable[1] = 5

TypeError: 'tuple' object does not support item assignment
```

Dictionaries

- Dictionary: a list of key-value pairs



Key Value

```
german_english_dictionary = {
    'Vorlesung':'Lecture',
    'Gleichung':'Equation'
}
```

```
german_english_dictionary
```

```
]: {'Vorlesung': 'Lecture', 'Gleichung': 'Equation'}
```


Image source: Romain Vignes

Dictionaries

- Dictionary: a list of key-value pairs

```
german_english_dictionary = {
    'Vorlesung':'Lecture',
    'Gleichung':'Equation'
}
```

- Look up something in the dictionary: it's an array with named entries!

```
german_english_dictionary['Vorlesung']
```

```
]: 'Lecture'
```

Tables

- Tables can be dictionaries with arrays as values

```
measurements_week = {
    'Monday':[2.3, 3.1, 5.6],
    'Tuesday':[1.8, 7.0, 4.3],
    'Wednesday':[4.5, 1.5, 3.2],
    'Thursday':[1.9, 2.0, 6.4],
    'Friday':[4.4, 2.3, 5.4]
}
```

```
measurements_week
```

```
]: {'Monday': [2.3, 3.1, 5.6],
    'Tuesday': [1.8, 7.0, 4.3],
    'Wednesday': [4.5, 1.5, 3.2],
    'Thursday': [1.9, 2.0, 6.4],
    'Friday': [4.4, 2.3, 5.4]}
```

- Retrieve a column

```
measurements_week['monday']
```

```
]: [2.3, 3.1, 5.6]
```

Troubleshooting

If your program throws error messages:

- Don't panic.
- *"There are two ways to write error-free programs; only the third one works."*

  Alan J. Perlis, Yale University
- Read where the error happened.
  - You may see your fault immediately, when looking at the right point.
- Read what appears to be wrong.
  - If you know, what's missing, you may see it, even if it's missing in a slightly different place.
  - Sometimes, something related is missing

```
print(round(4.5)

  File "<ipython-input-15-09a9be4a90c5>", line 1
    print(round(4.5)
                   ^
SyntaxError: unexpected EOF while parsing
```

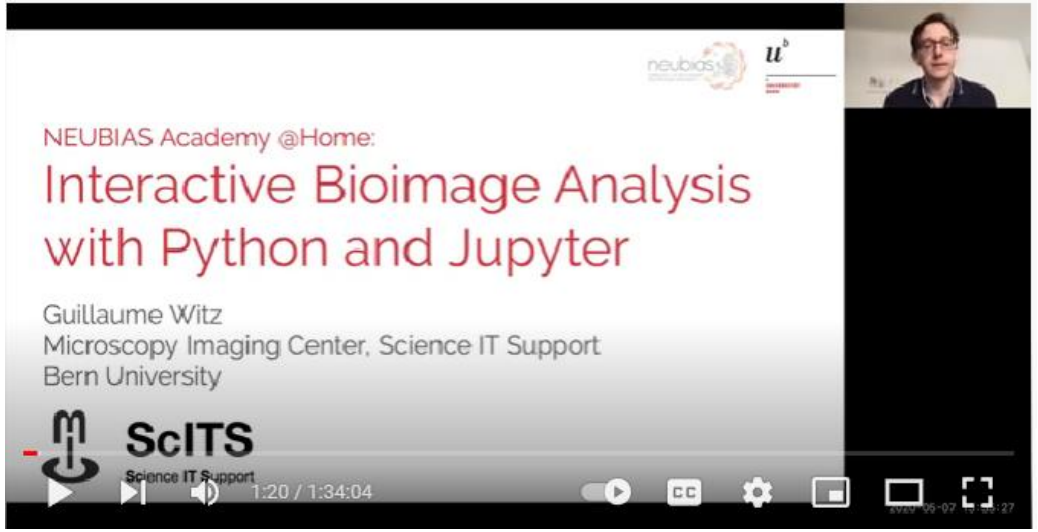
Comments

Comments should contain additional information such as

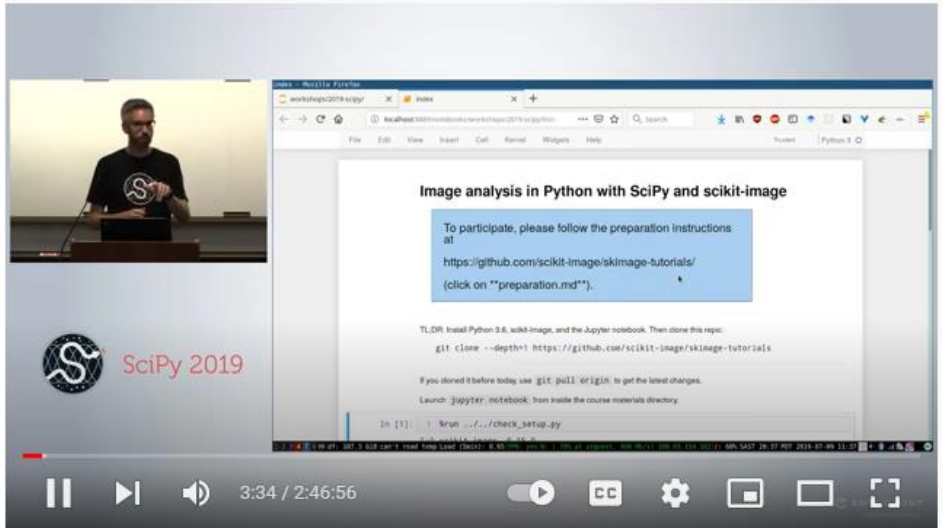
- User documentation
  - What does the program do?
  - How can this program be used?
- Your name / institute in case a reader has a question
- Comment why things are done.
- Do not comment what is written in the code already!

```
#
# This program sums up two numbers.
#
# Usage:
# * Run it in 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 )
```

More resources



Guillaume Witz, NEUBIAS Academy 2020



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

Watch more:

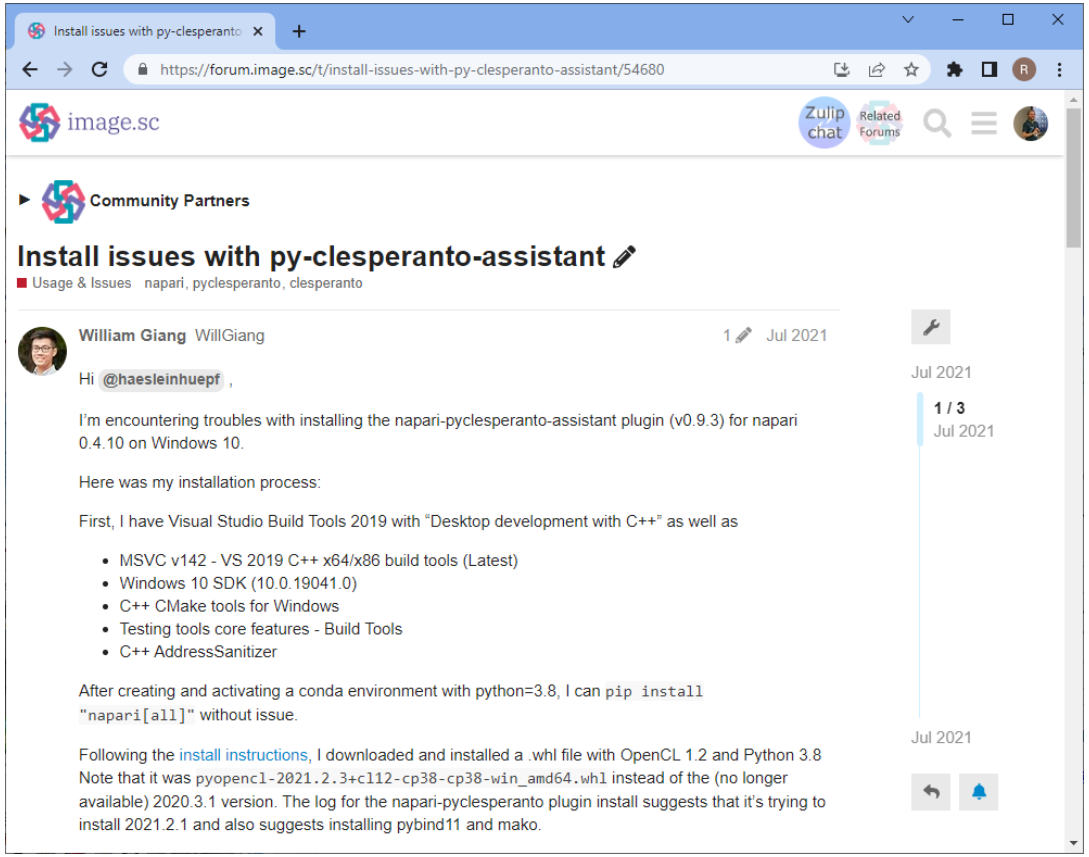
- https://www.youtube.com/watch?v=2KF8vBrp3Zw
- https://www.youtube.com/watch?v=d1CIV9irQAY
- https://www.youtube.com/watch?v=X_pCiVQ4c4E

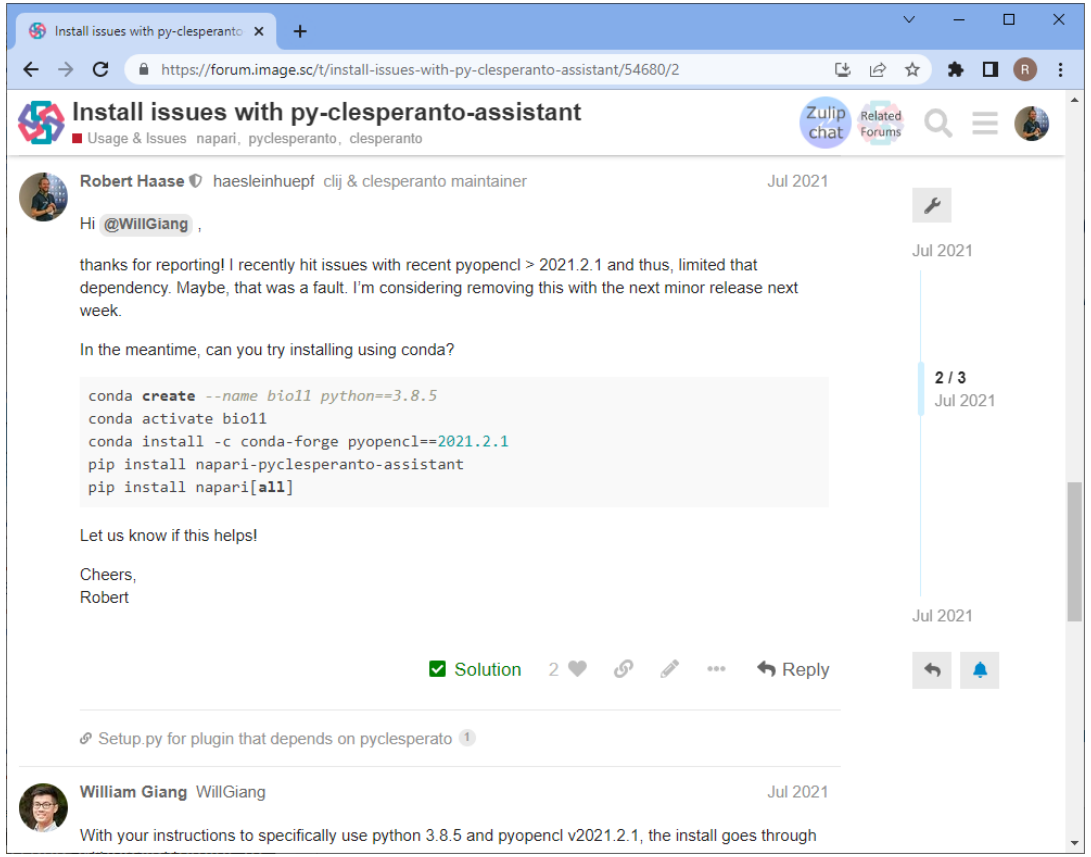
Sreenivas Bhattiprolu, Python for Microscopists @Youtube 2019-…

The Image Science Community

PoL Physics of Life TU Dresden

- Ask your question online and an expert will likely reply the same day ☺





Summary

Today, you learned

- *Bio-image analysis*
  - Quantitative
  - Objective
  - Reproducible
  - Repeatable
  - Reliable
- The command line
- Jupyter Lab
- Python programming
  - Variables
  - Arrays (lists, tuples)
  - Dictionaries

`data[start:stop:step]`

Coming up next

- Loops
- Conditions
- Functions
- Libraries

```python
# 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
```

PoL Physics of Life TU Dresden

CENTER FOR SYSTEMS BIOLOGY DRESDEN

Exercises

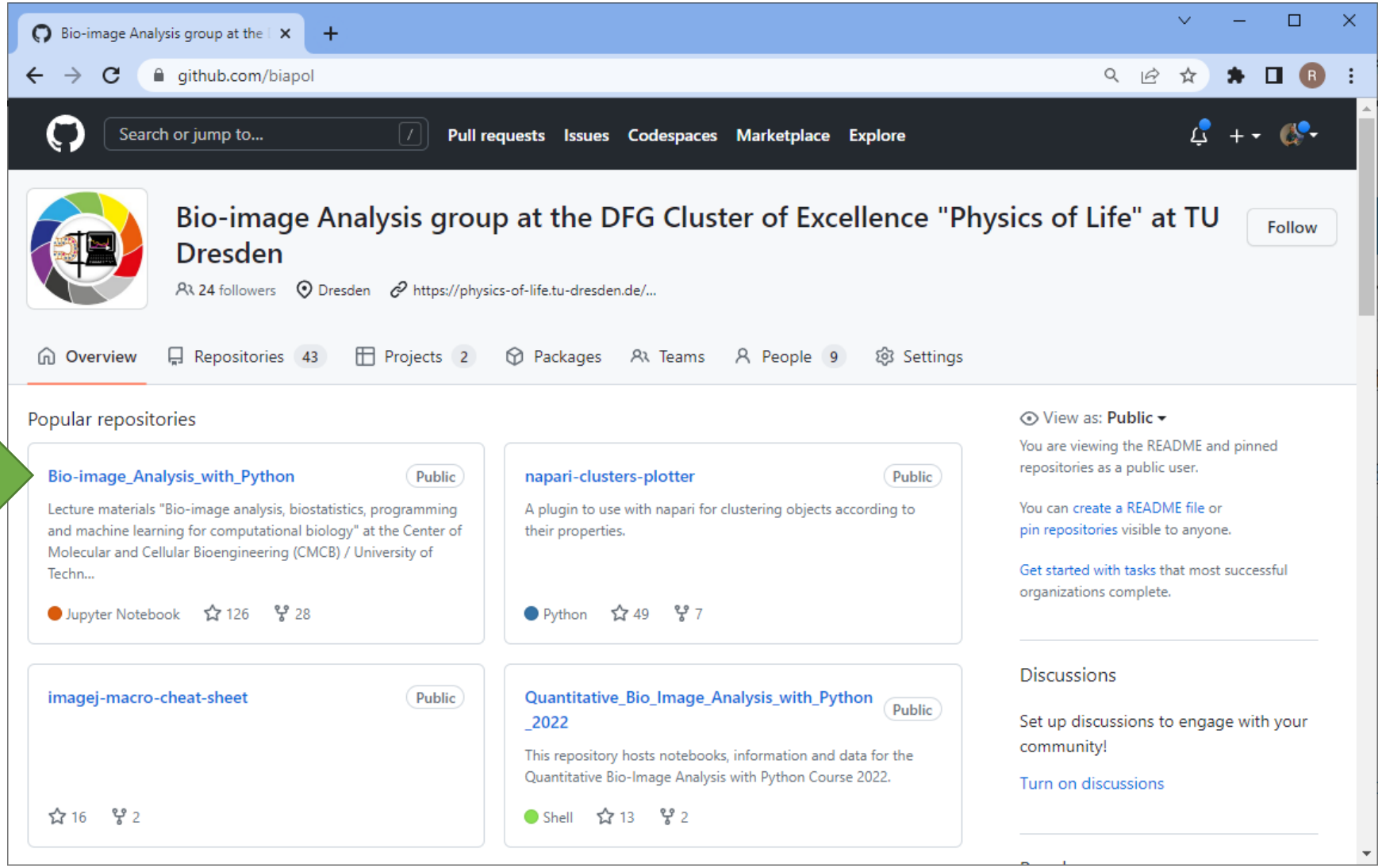
Robert Haase

April 2023

@haesleinhuepf

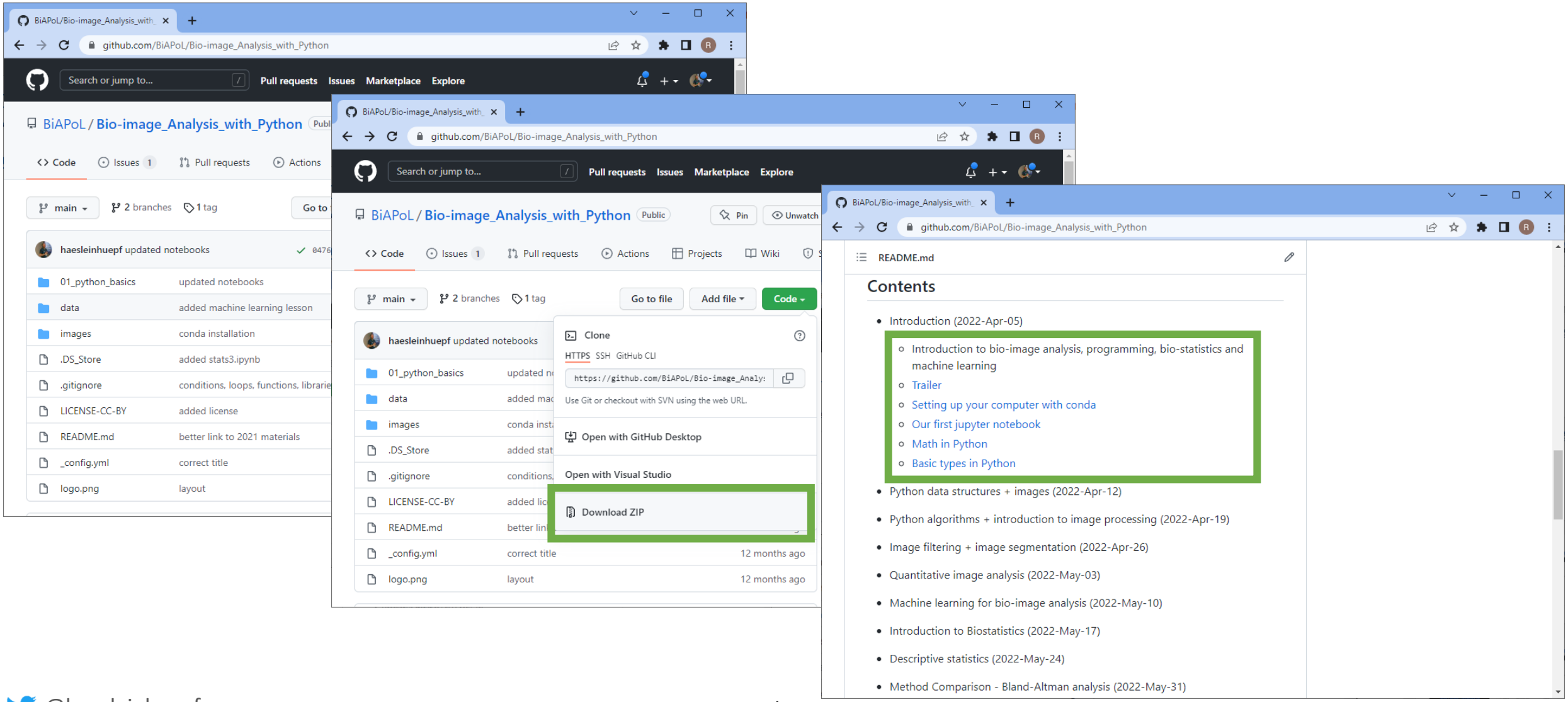
Exercise: Explore our GitHub repository

- https://github.com/biapol



Exercise: Explore our GitHub repository

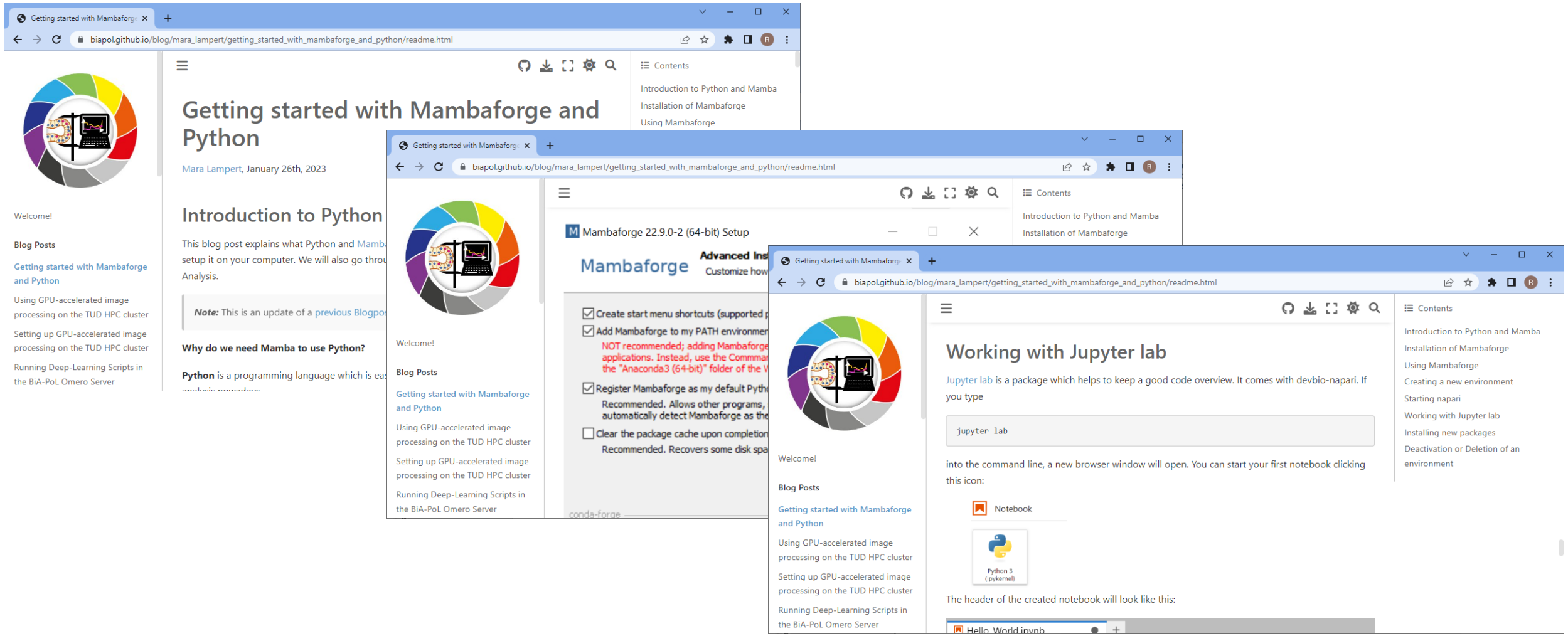
- https://github.com/BiAPoL/Bio-image_Analysis_with_Python



Exercise: Install mambaforge and test Python

Detailed instructions:

- https://biapol.github.io/blog/mara_lampert/getting_started_with_mambaforge_and_python/readme.html

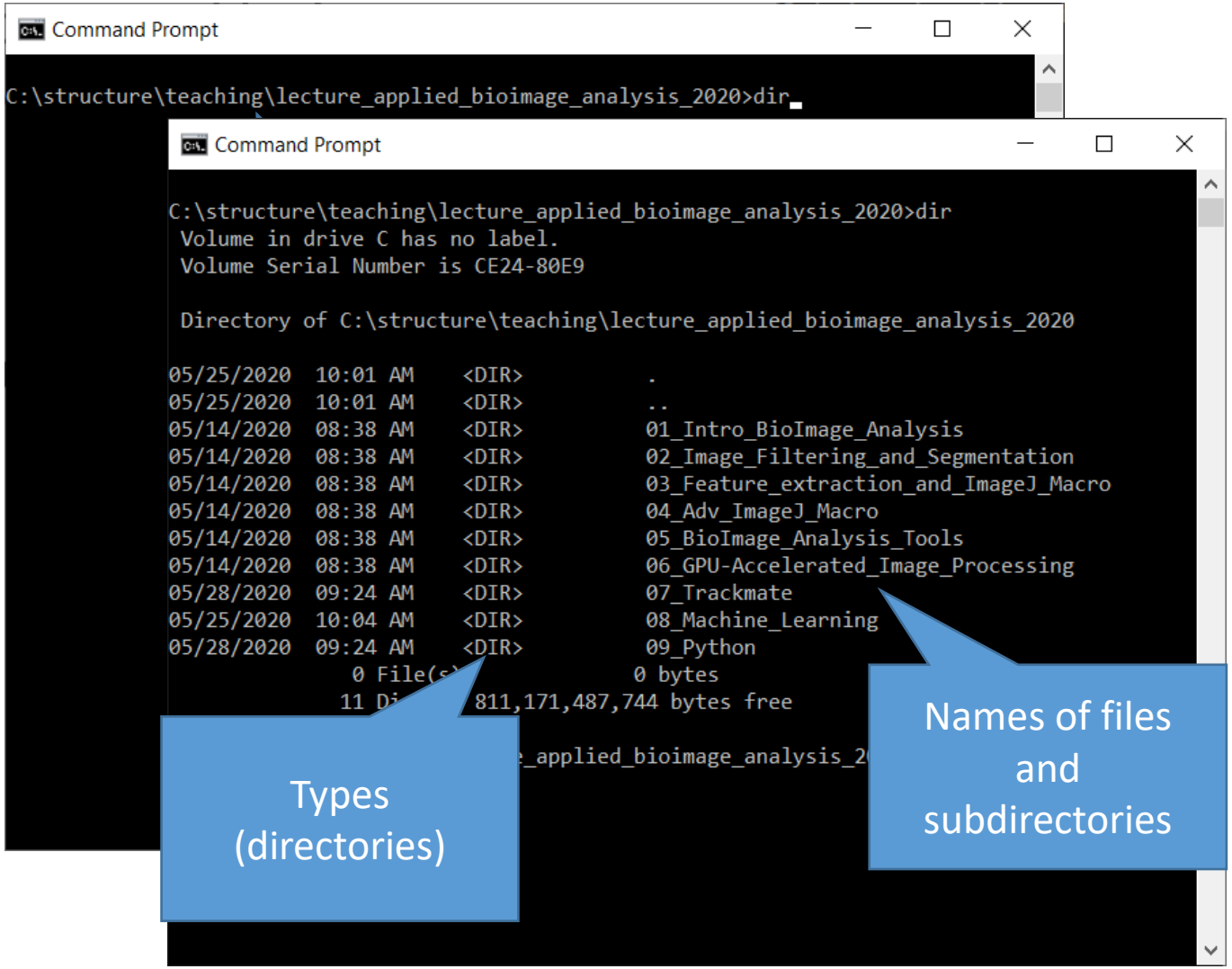


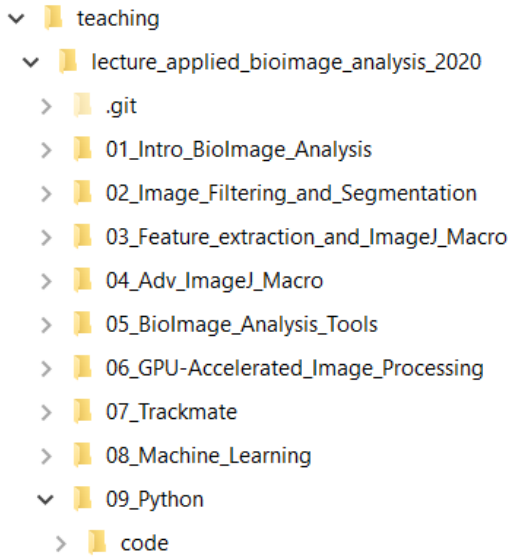
The command line

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

- The `dir` command tells you what's in the current directory

- On Mac and Linux the command is called `ls -l`



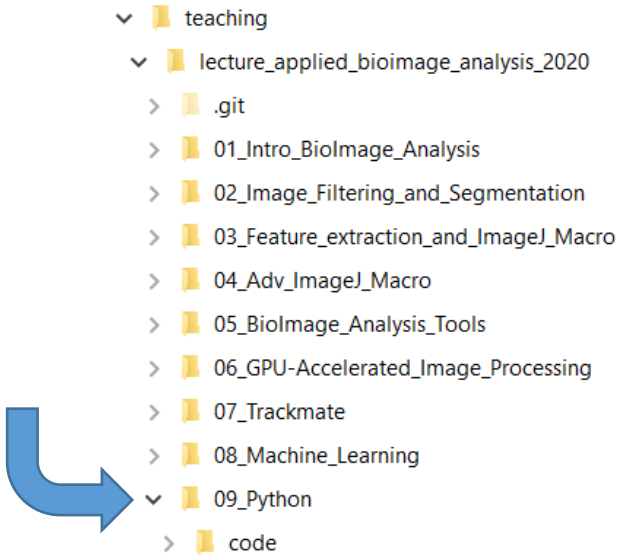


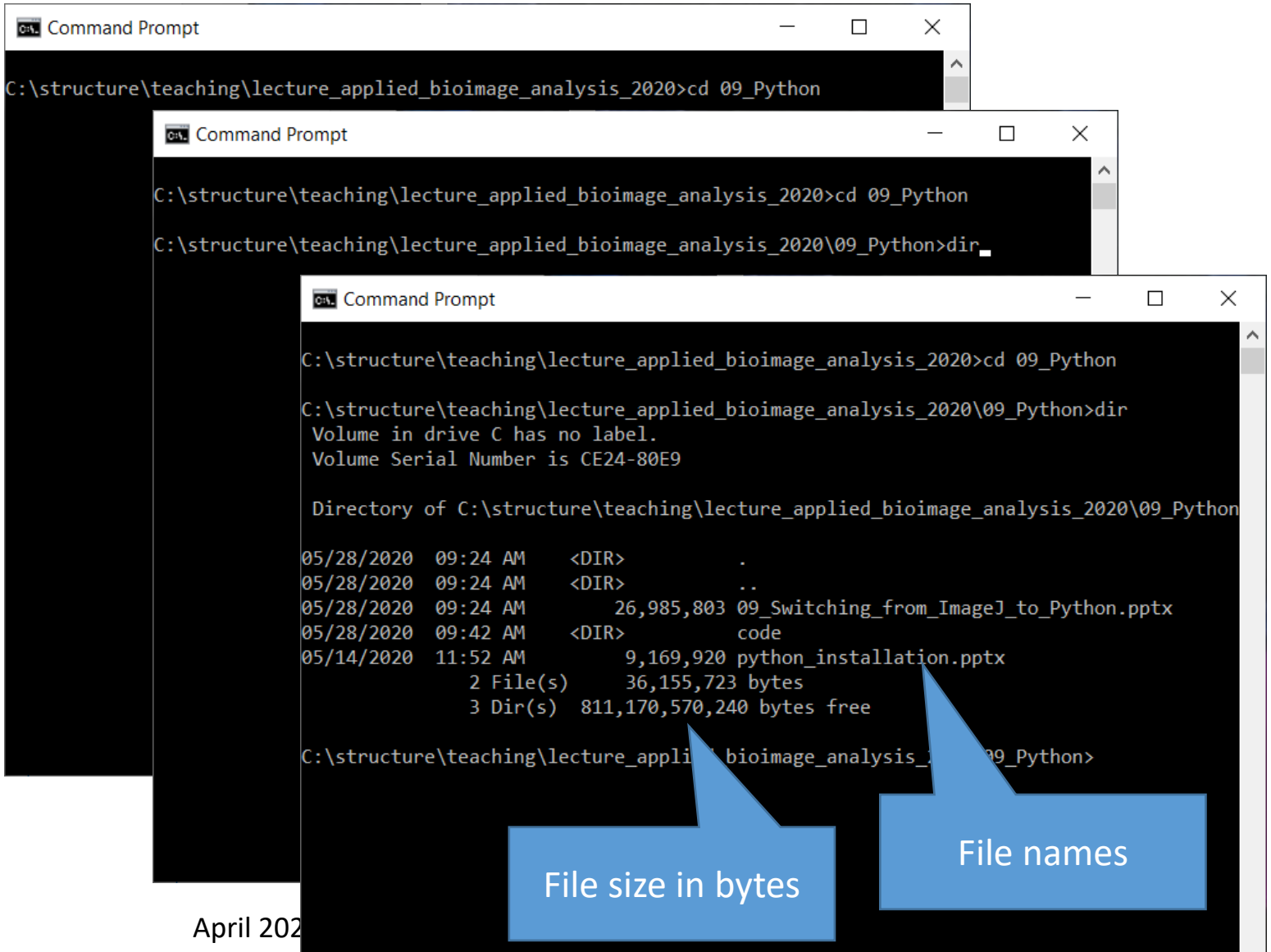
The command line

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

- The `cd` command let's you move between different directories.

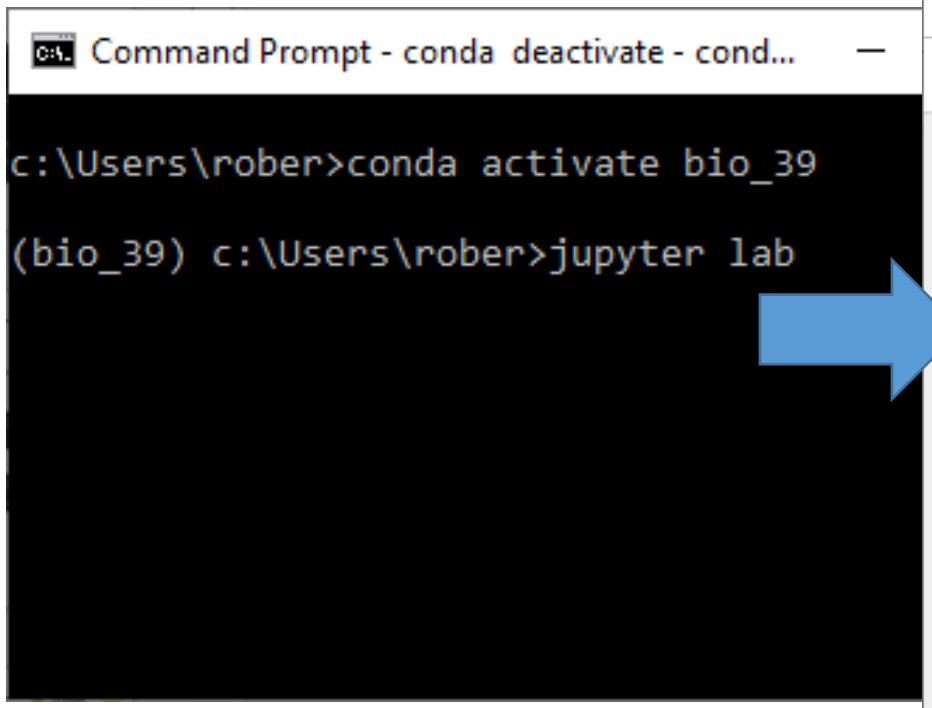
- With `cd <pathname>` you go into a sub-directory

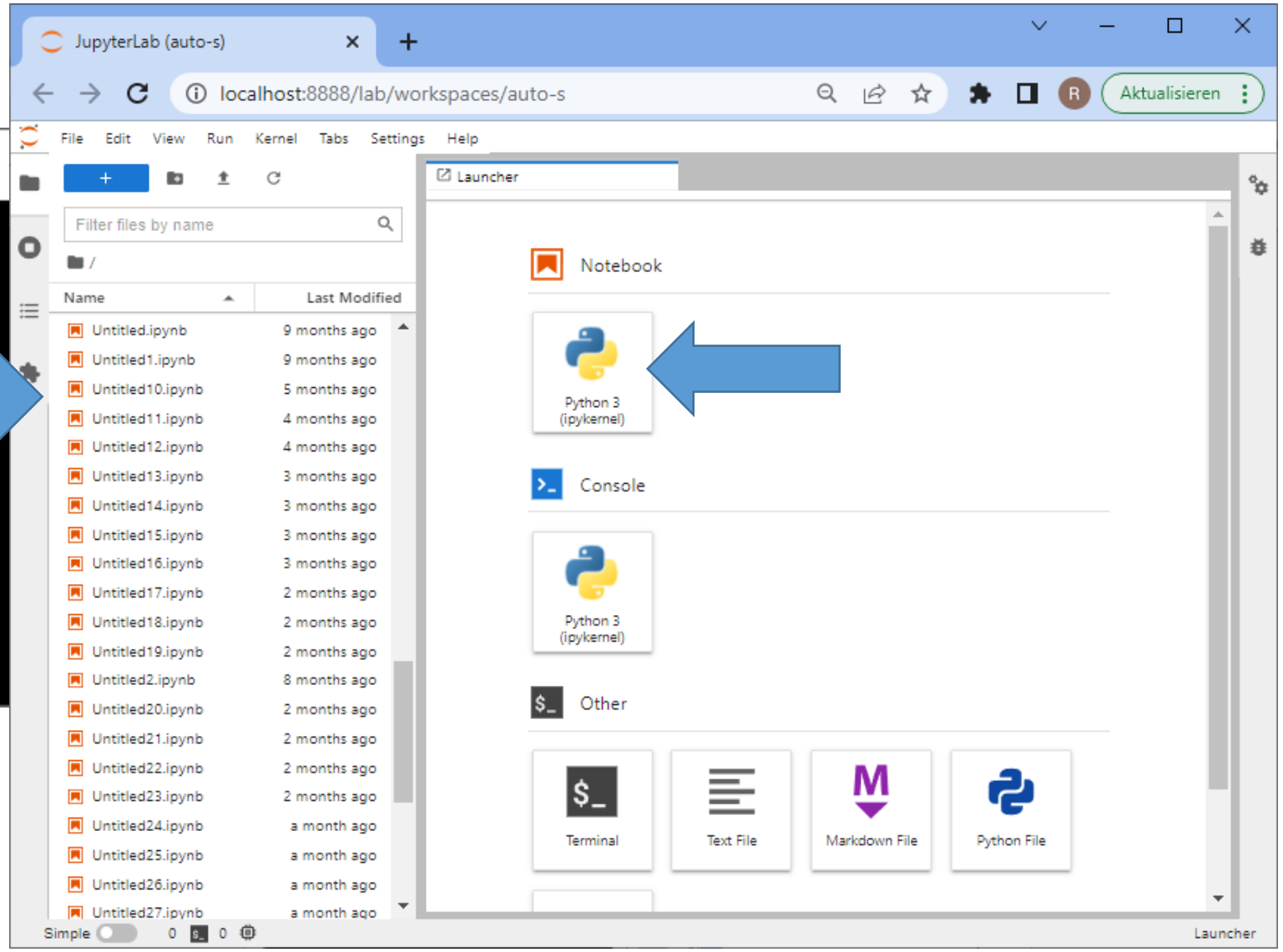




Jupyter lab

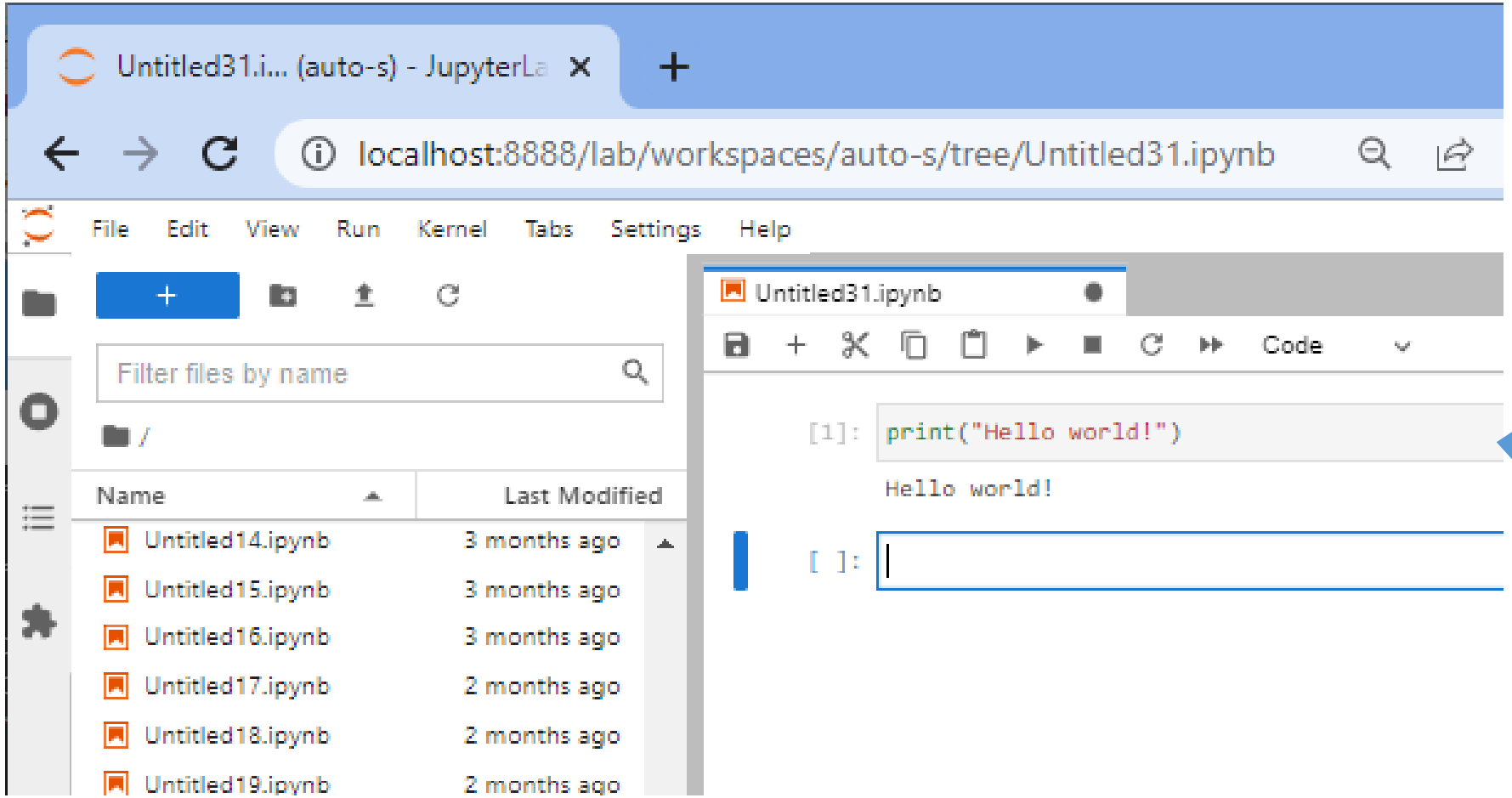
- Start Jupyter lab from the folder you want to work in





Jupyter lab

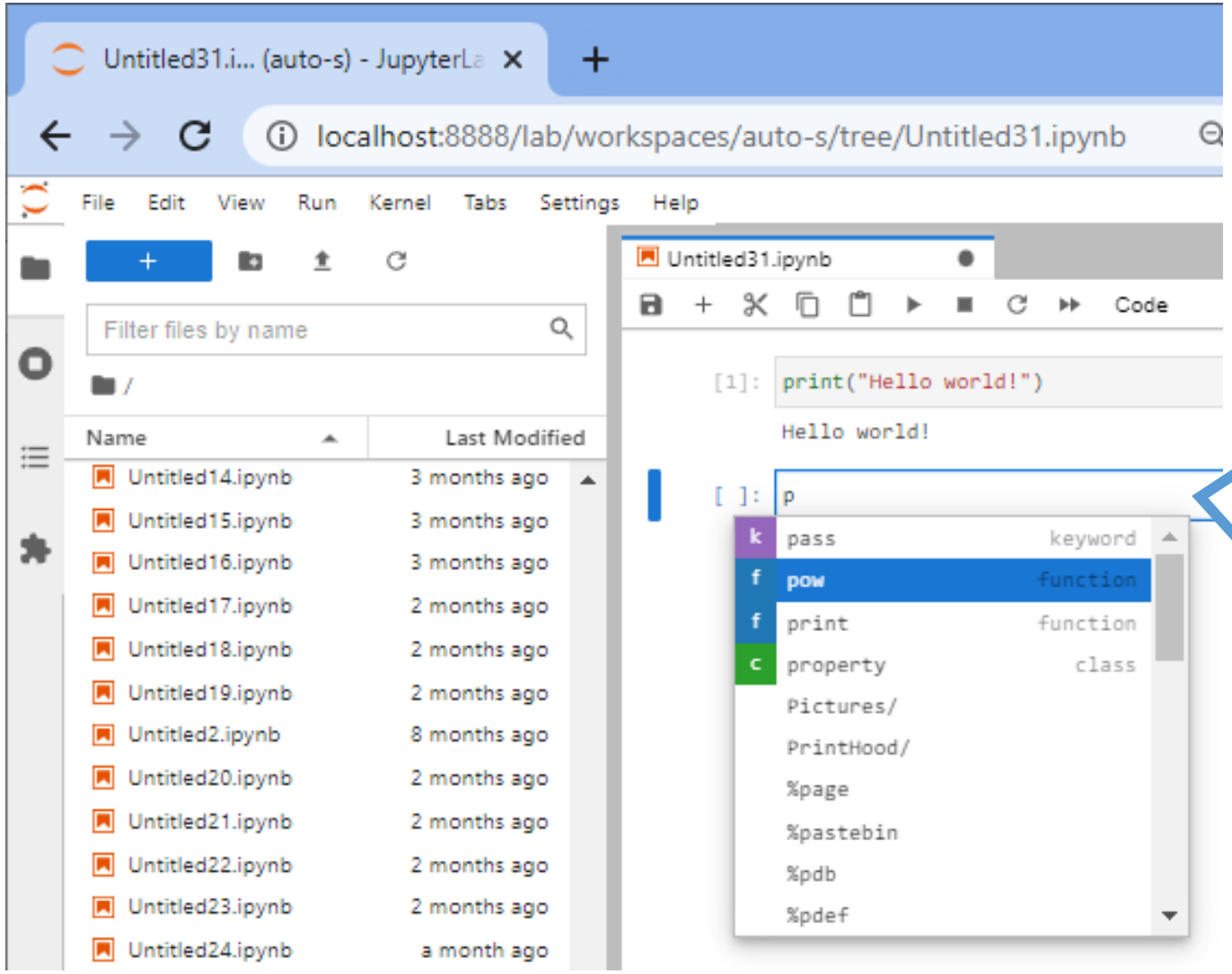
- Execute code cell-by-cell and see results instantaneously



SHIFT + ENTER to execute a code cell

Jupyter lab

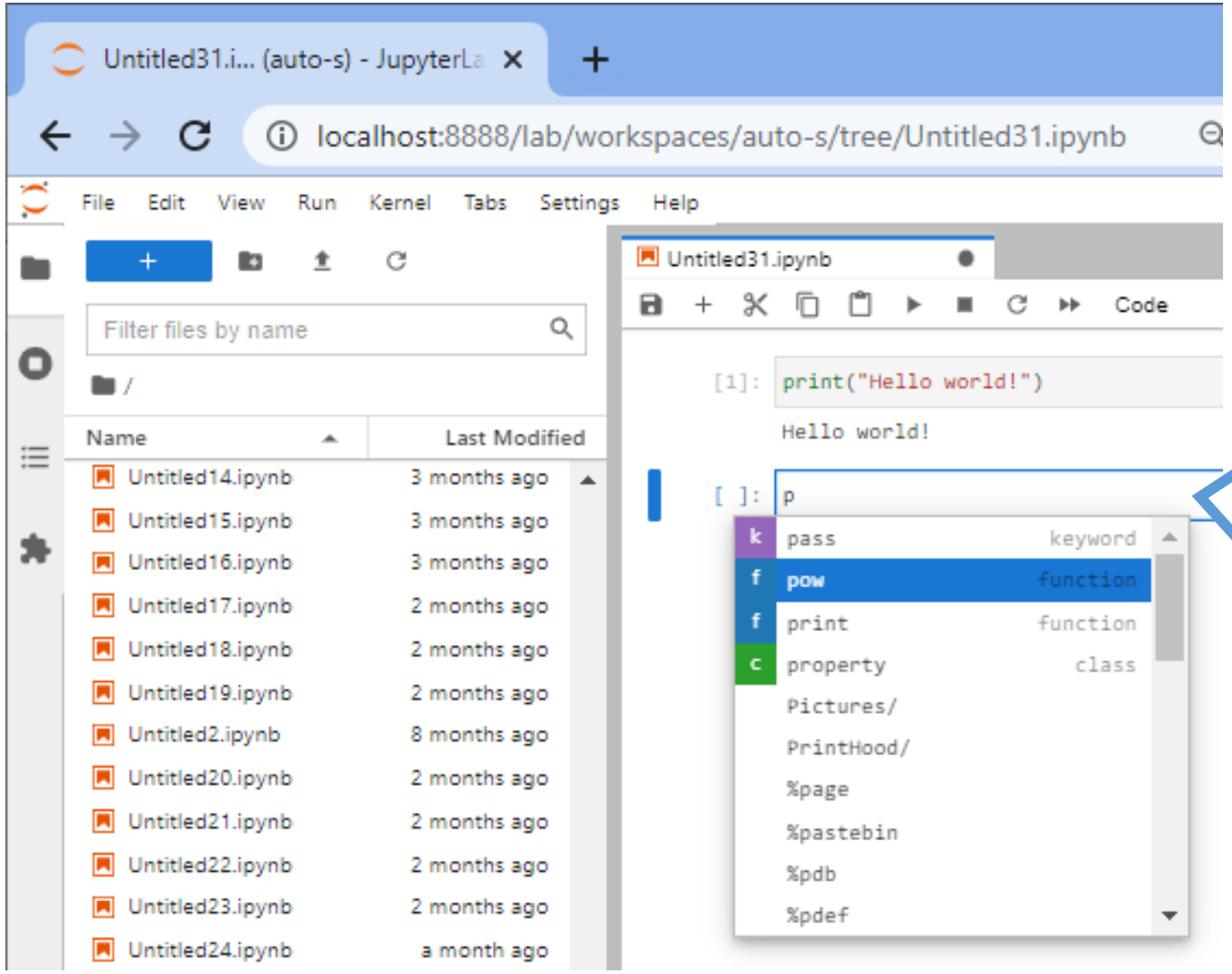
- Context-specific help, auto-completion



TAB
to open auto-completion

Jupyter lab

- Context-specific help, auto-completion



TAB
to open auto-completion

Jupyter lab

- Help / “docstrings”

Untitled31.i... (auto-s) - JupyterLa

localhost:8888/lab/workspaces/auto-s/tree/Untitled31.ipynb

File Edit View Run Kernel Tabs Settings Help

Filter files by name

/

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

Untitled31.ipynb

Code

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

```
Hello world!
```

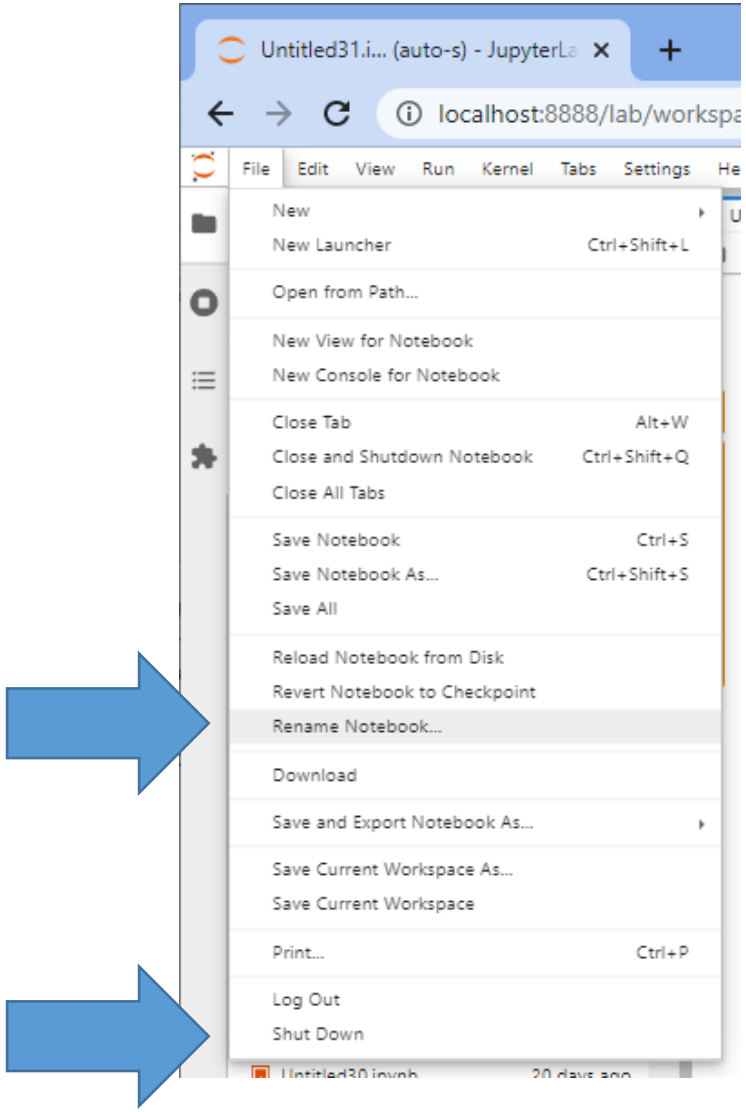
[3]:
```
print?
```

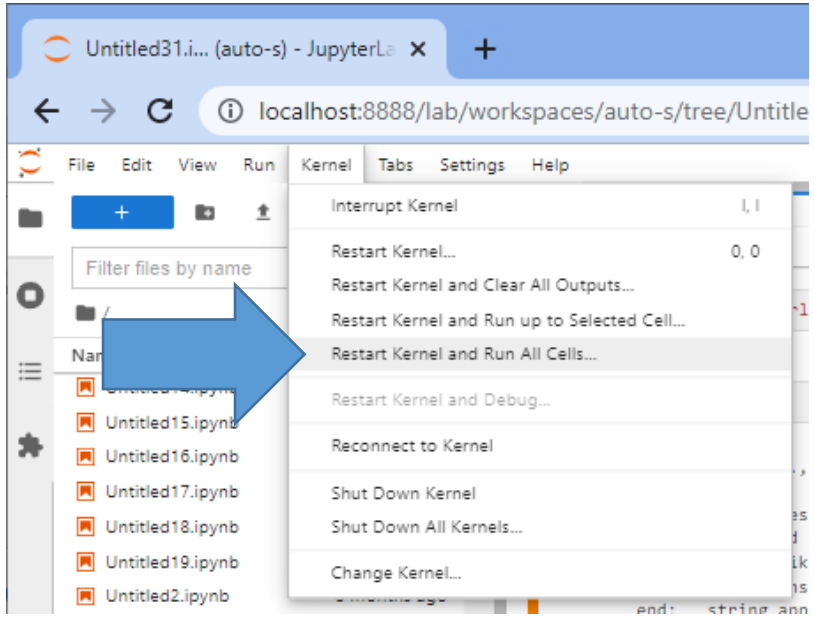
```
Docstring:
print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

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
```

?
to read what a function does

Jupyter lab

- Saving / renaming / closing





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

Exercise: Basic python

- https://github.com/BiAPoL/Bio-image_Analysis_with_Python