About this Webinar

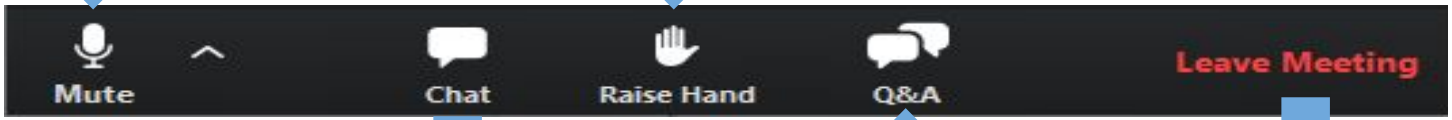
Your microphone is muted

There will be no oral questions,
unless the speaker/panelists enable them

No video

Raising hand may be monitored by the speaker
for quickly addressing the audience.

Zoom control panel when in a webinar (Participant view)



Do <u>not</u> use the *chat*

Use the Q&A Button instead.
Questions are managed
and answered by panelists.

Leaving and coming back
will reset the Q&As panel

All questions will be answered live if possible,
Answers will be made available on image.sc Forum

This webinar will be recorded and
uploaded to Youtube NEUBIAS Channel

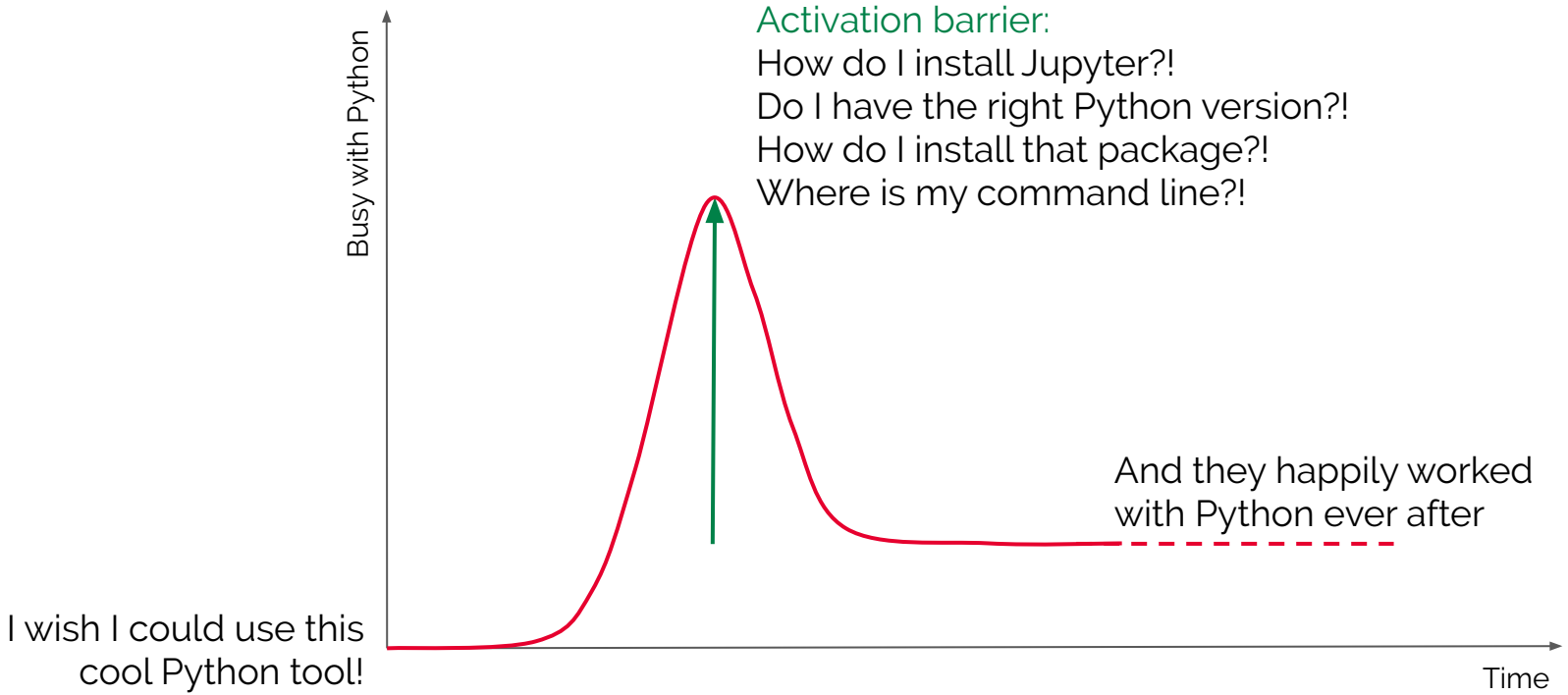
NEUBIAS Academy @Home:

Interactive Bioimage Analysis with Python and Jupyter

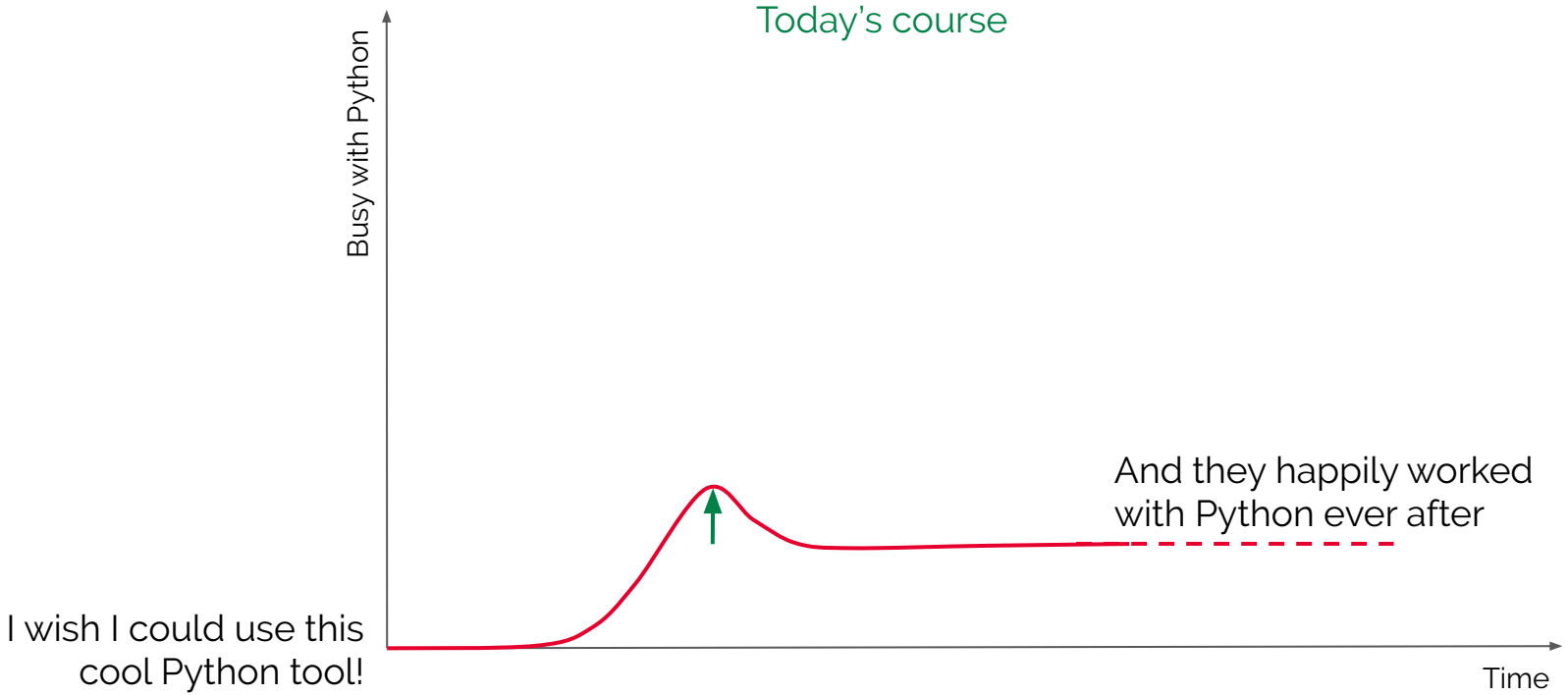
Guillaume Witz
Microscopy Imaging Center, Science IT Support
Bern University

ScITS
Science IT Support

The Python activation barrier



The Python activation barrier



Course structure

Webinar 1: Introduction to Jupyter notebooks and Python tools for bioimage analysis. Presentation of self-learning material, Q&A

One week self-learning (no download, no installation required!)

Webinar 2: Answer to common questions. Presentation of selected notebooks, Q&A

Q&A Moderators:

Mykhailo Vladymyrov, TKI, Bern University

Cédric Vonesch, ScITS, Bern University

Dominik Kutra, EMBL

Course material

All the course material available on GitHub:

https://github.com/guiwitz/neubias_academy_biapy

This presentation available here:

https://bit.ly/neubias_biapy

The Python bioimage analysis ecosystem

... A **programming language** and a way to execute it ...

The Python bioimage analysis ecosystem

... A **programming language** and a way to execute it ...

... **Specific tools** for our scientific task (packages) ...

aicsimageio, ipyvolume, napari, ipywidgets, trackpy, cellpose, stardist

The Python bioimage analysis ecosystem

... A **programming language** and a way to execute it ...

... **Specific tools** for our scientific task (packages) ...

aicsimageio, ipyvolume, napari, ipywidgets, trackpy, cellpose, stardist

... A solution to **interact** with code and data ...

The Python bioimage analysis ecosystem

... A **programming language** and a way to execute it ...

... **Specific tools** for our scientific task (packages) ...

aicsimageio, ipyvolume, napari, ipywidgets, trackpy, cellpose, stardist

... A solution to **interact** with code and data ...

... **Infrastructure** to install and run the code

Jupyter notebooks

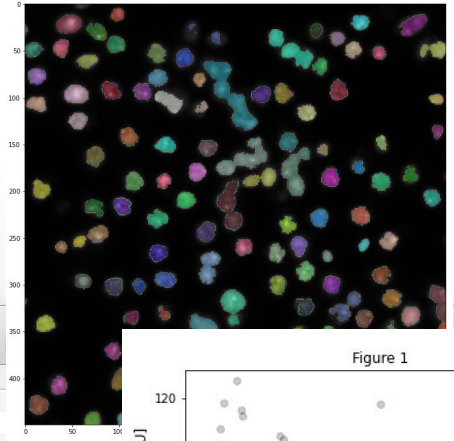
“Classic” software vs. notebooks

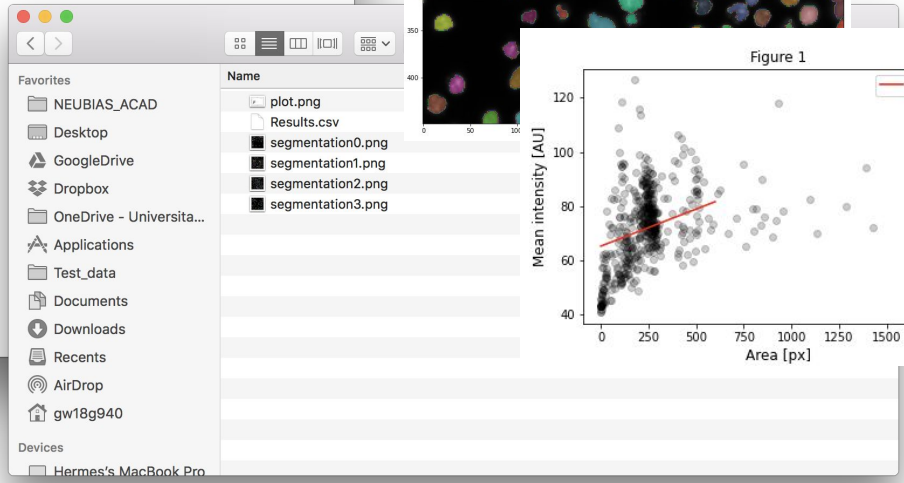
course_functions.py

```
import skimage.morphology
import skimage.filters
import numpy as np
from skimage.measure import label, regionprops_table
import matplotlib

def detect_nuclei(image, si

    # filtering
    image = skimage.filters
    # local thresholding
    image_local_threshold =
    image_local = image > i
    # remove tiny features
    image_local_open = skim
    # label image
    image_labeled = label(i
    # analyze regions
    our_regions = regionpro
    # create a new mask wit
    newimage = np.zeros(ima
    # fill in using region
    for x in range(len(our_
        if our_regions['are
            newimage[our_re

    return newimage
```

gw18g940 — -bash — 72×21

```
Last login: Thu Apr 30 10:41:20 on ttys002
(base) Hermess-MacBook-Pro:~ gw18g940$ python course_functions.py
```





“Classic” software vs. notebooks

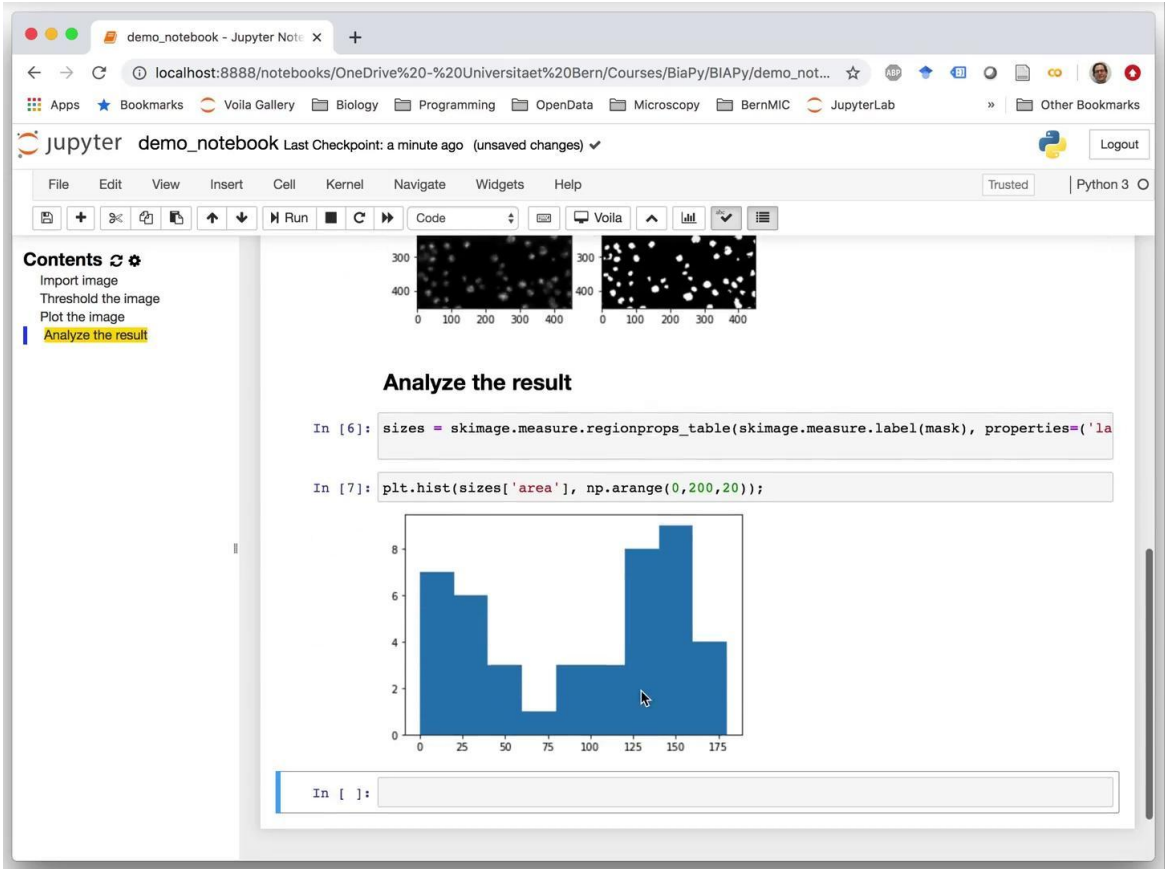
Code divided in parts: easy to investigate

Dynamic: easy to test

Rich output: processing + visualisation + analysis in one place

Code + formatted text: easy documentation

Illustration: load an image, threshold it, analyze object size

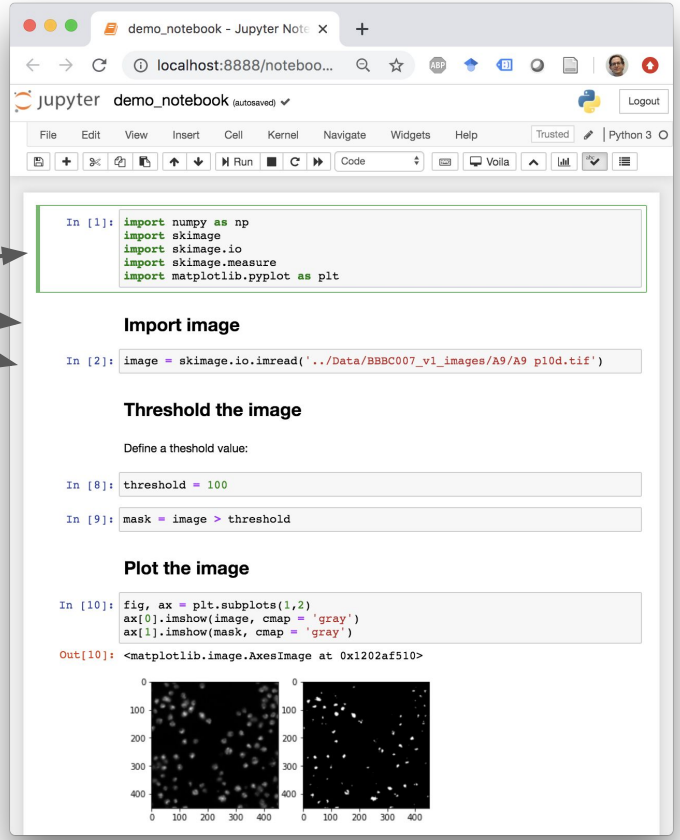


What is a Jupyter notebook?

A **text** file (easily sent around)

Rendered by Jupyter in the **browser**

Split into sections called **cells**



What is a Jupyter notebook?

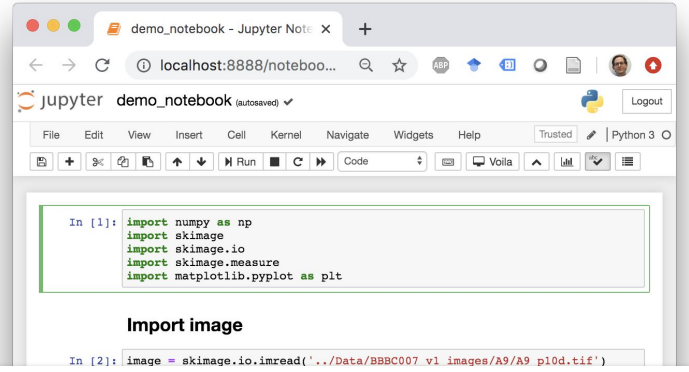
A **text** file (easily sent around)

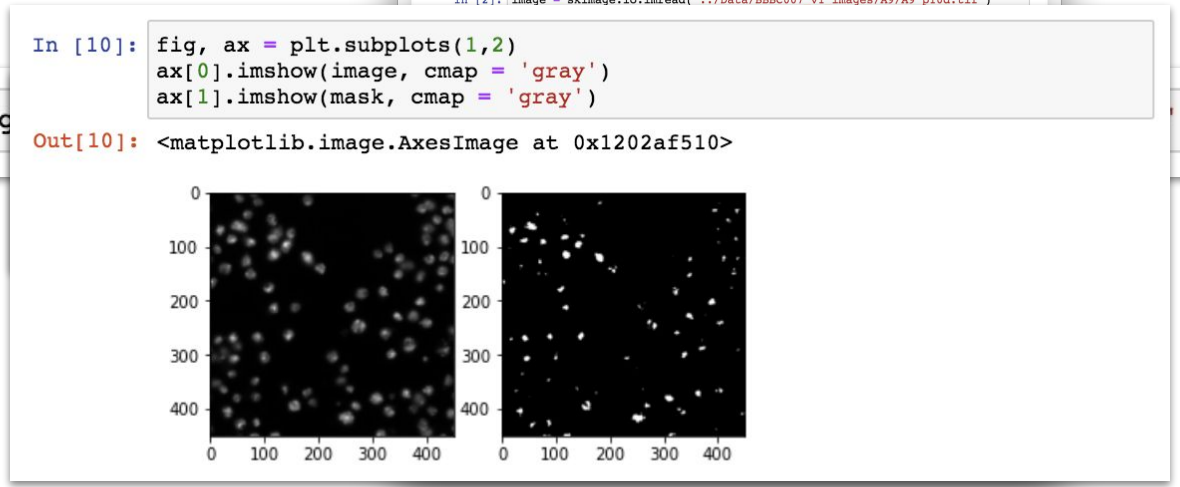
Rendered by Jupyter in the **browser**

Split into sections called **cells**

Cells can contain:

- Code
- Formatted text
- Rich output





Notebook cells

Cells can be executed individually

Variables accessible in the whole notebook

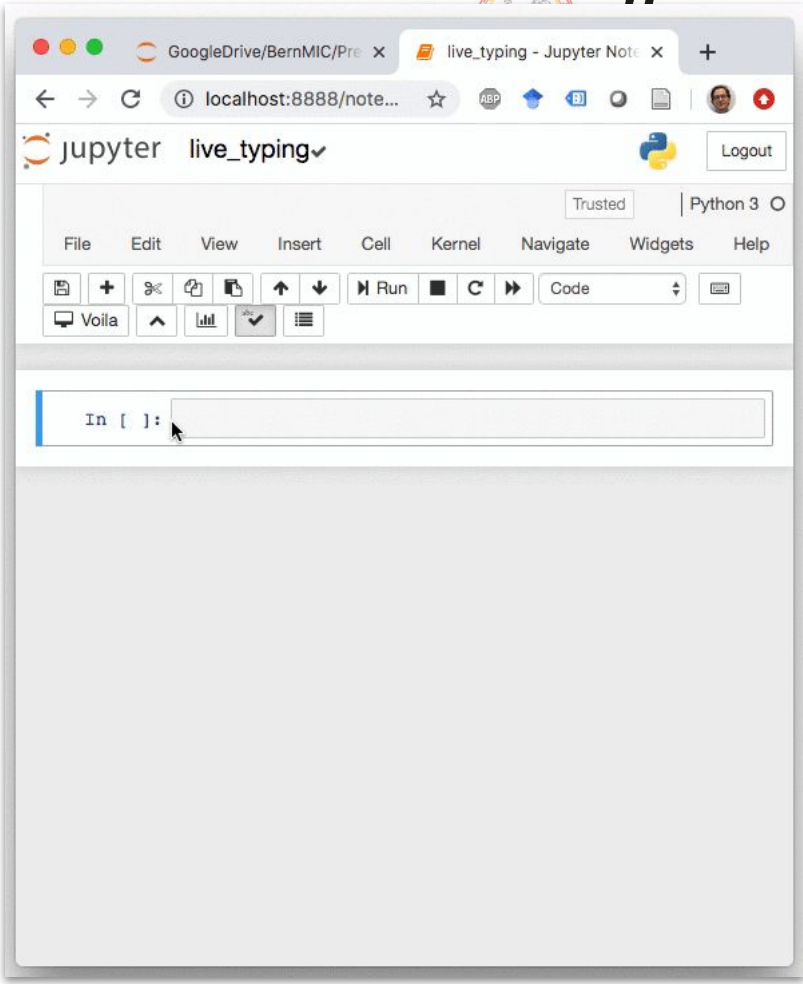
New cells can be added anywhere

Only the order of execution matters.

Good practice: Top-down order

The cell type can be switched from Code to Text

Possible to run all cells are part of a notebook



Notebook cells

Cells can be executed individually

Variables accessible in the whole notebook

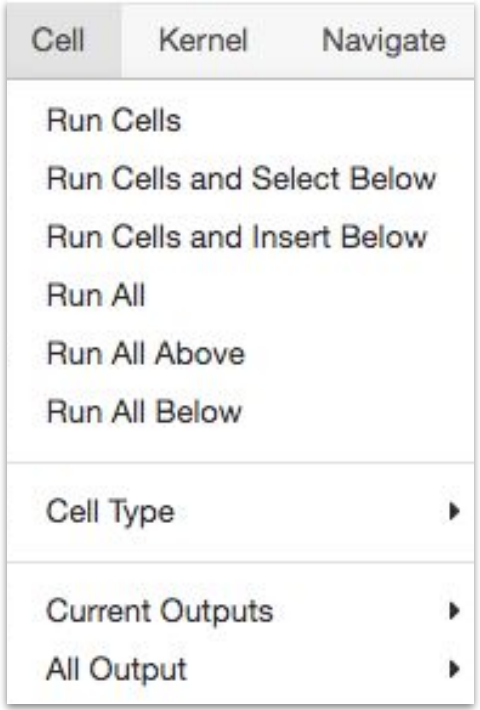
New cells can be added anywhere

Only the order of execution matters.

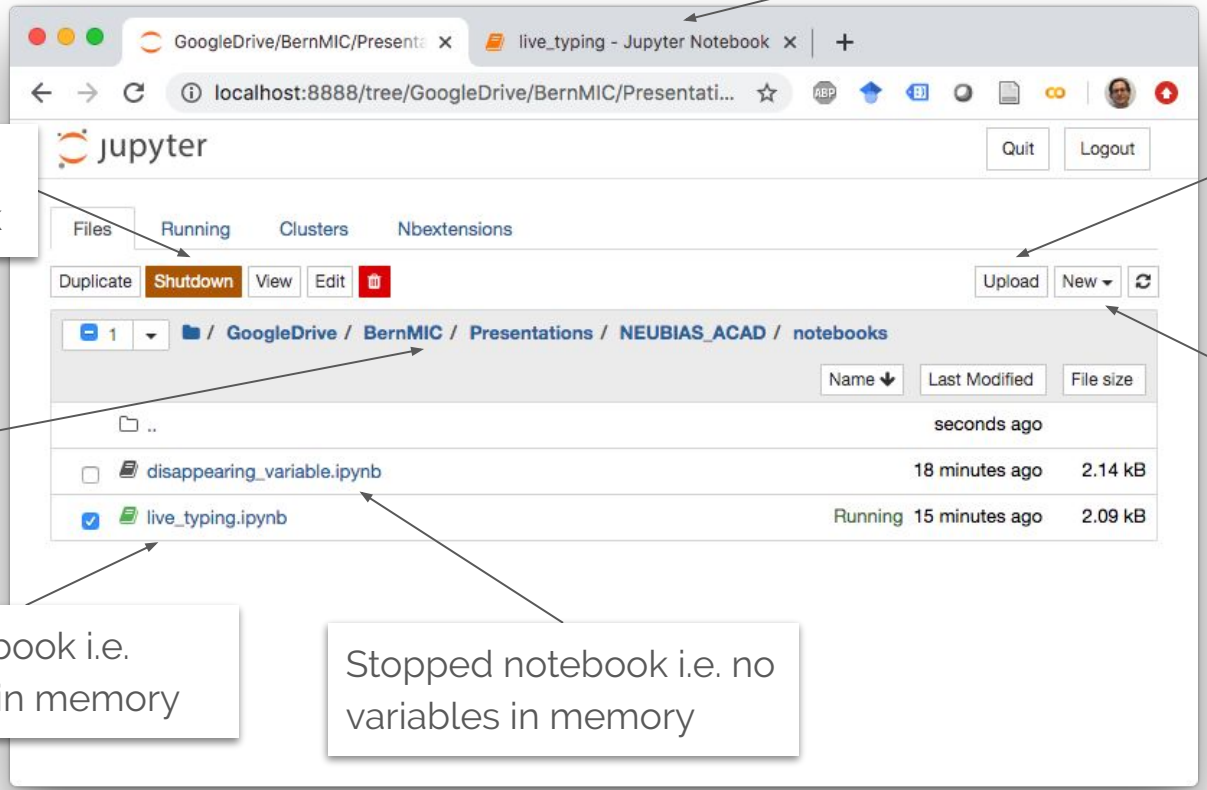
Good practice: Top-down order

The cell type can be switched from Code to Text

Possible to run all cells are part of a notebook



Jupyter browser



Open notebooks appear as browser tabs

Stop kernel of selected notebook

Upload files (e.g. data)

Path to current folder

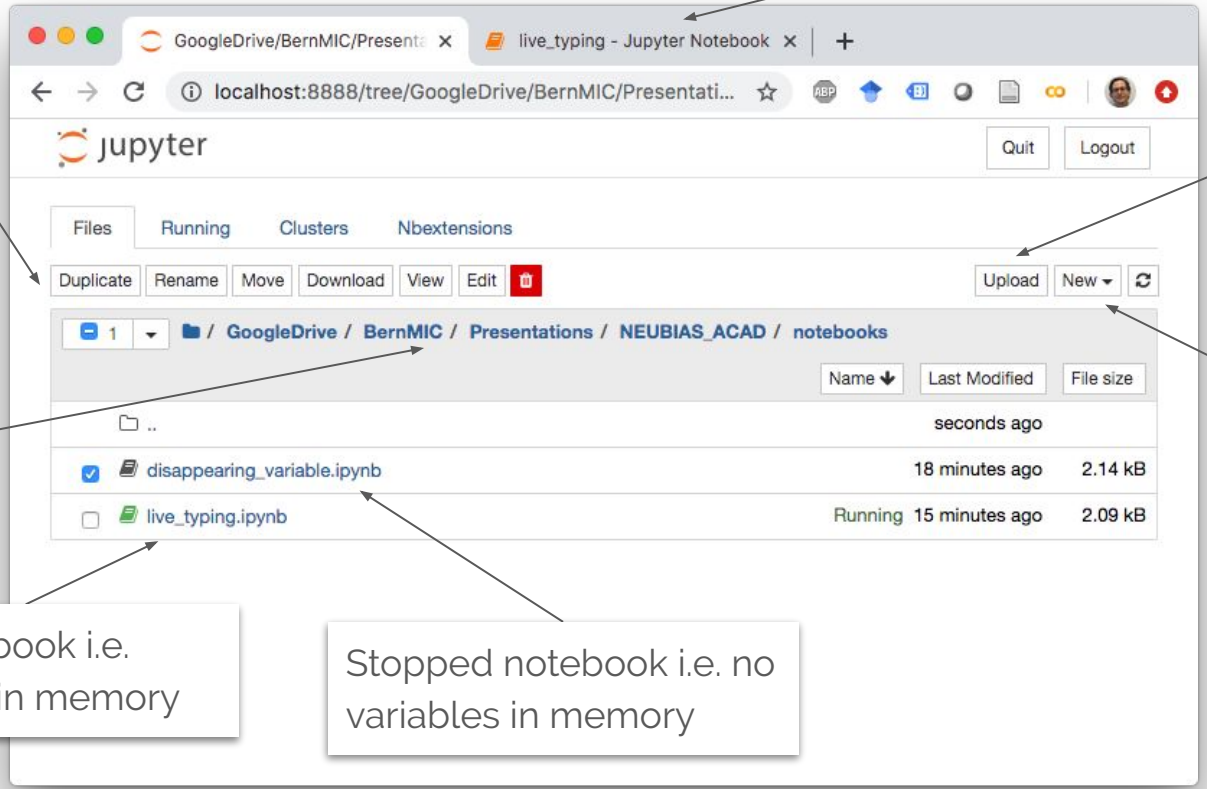
Create new notebook

Live notebook i.e. variables in memory

Stopped notebook i.e. no variables in memory

Jupyter browser

Open notebooks appear as browser tabs



Options for stopped notebook: duplication, download etc.

Upload files (e.g. data)

Path to current folder

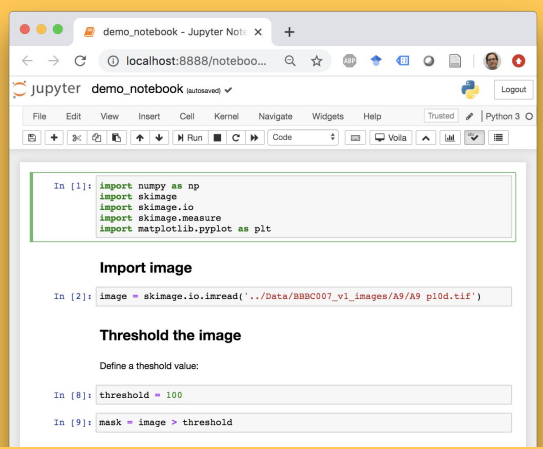
Create new notebook

Live notebook i.e. variables in memory

Stopped notebook i.e. no variables in memory

How does a notebook calculate?

Notebook displayed in **browser**



Sends computations

Sends result

Computations done by **kernel**

python™

Kernel can be **local** (laptop) or **remote** (cluster, Google etc,)

For you, the user, “where it runs” doesn’t affect the interface

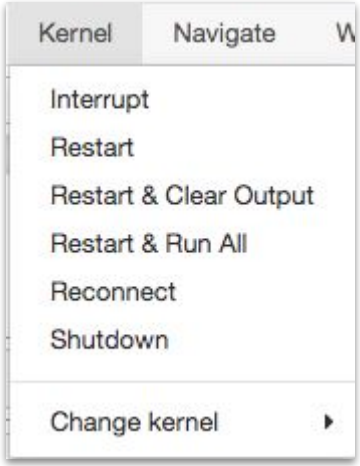
The notebook kernel

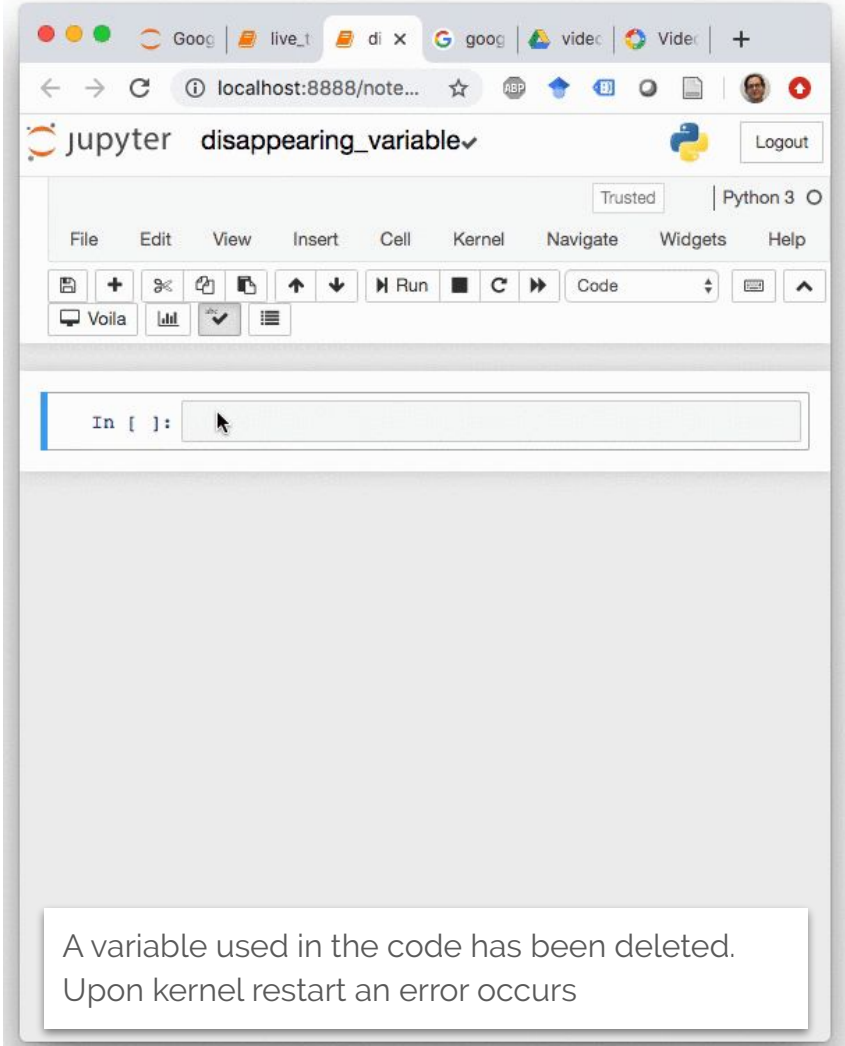
Notebook content does not depend on kernel

Variables conserved as long as kernel is ON

Kernel can be restarted (e.g to stop never-ending calculation)

Good practice: periodically restart kernel to avoid “strange” states





A variable used in the code has been deleted.
Upon kernel restart an error occurs

Sharing and running notebooks

GitHub

Sharing notebooks

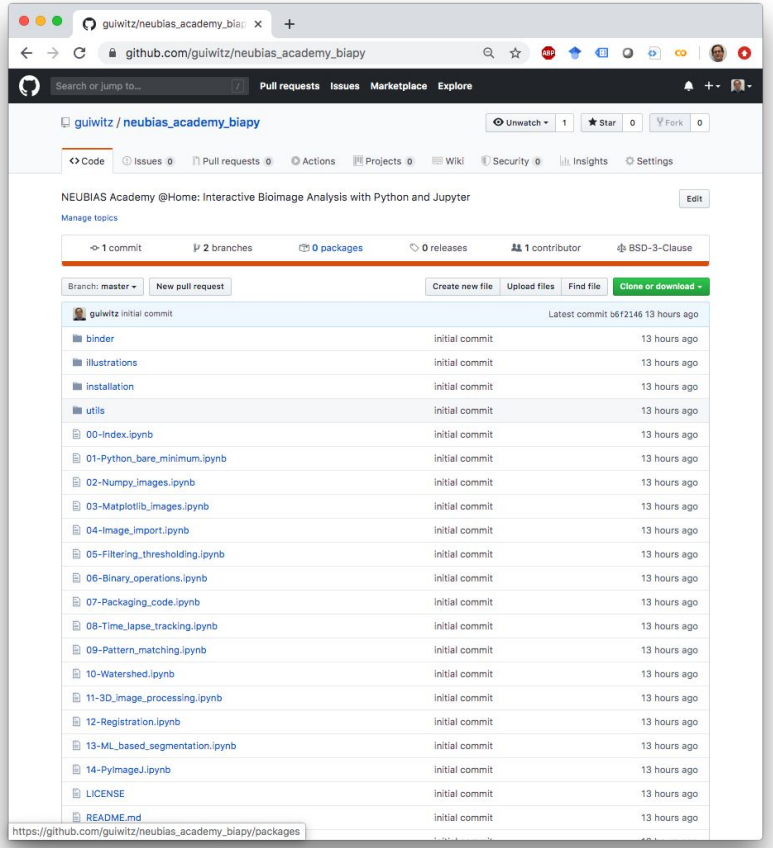
GitHub is a repository for code based on git

Projects are “folders” called a repositories

Public repositories are browsable:

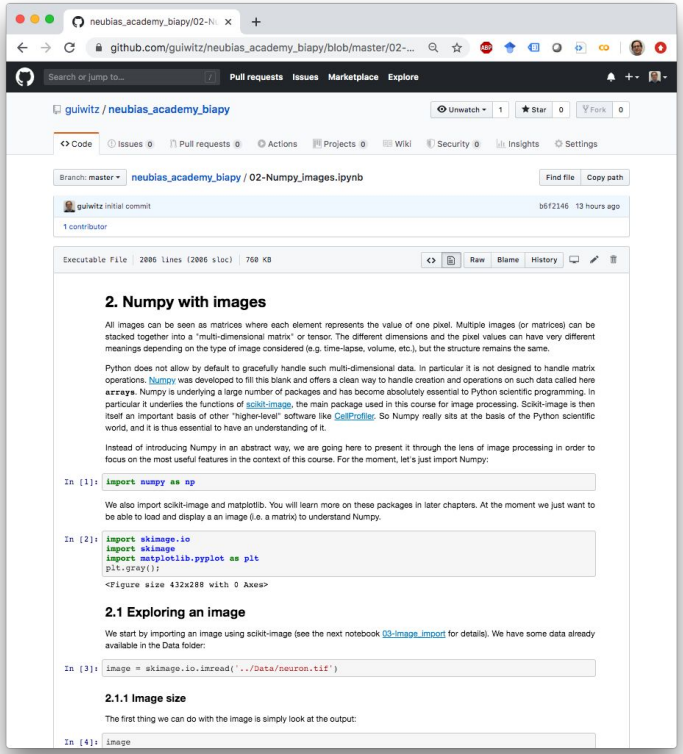
https://github.com/guiwitz/neubias_academy_biapy

GitHub is **many** more things than just a repository



Sharing notebooks statically

Notebooks are rendered **statically** on GitHub

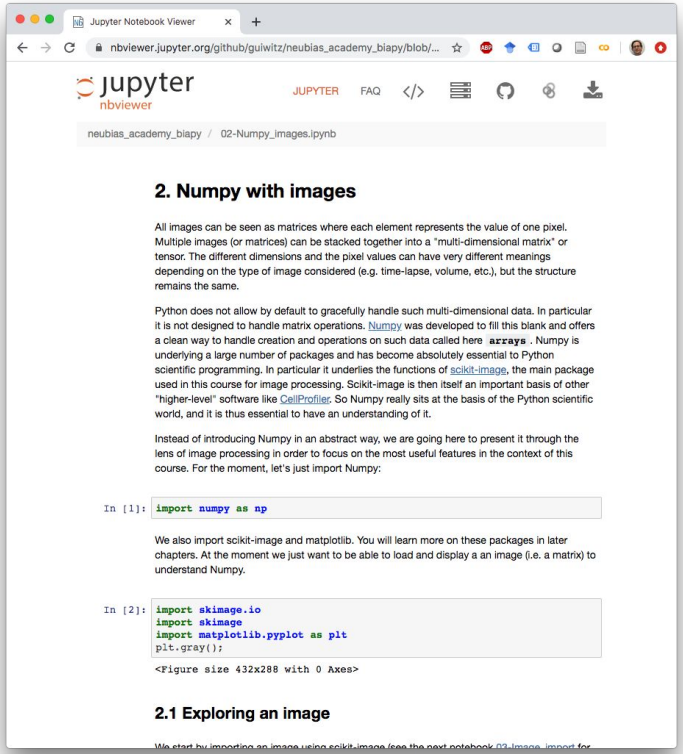


GitHub rendering of notebook

Sharing notebooks statically

Nbviewer (https://nbviewer.jupyter.org/) offers a more reliable rendering:

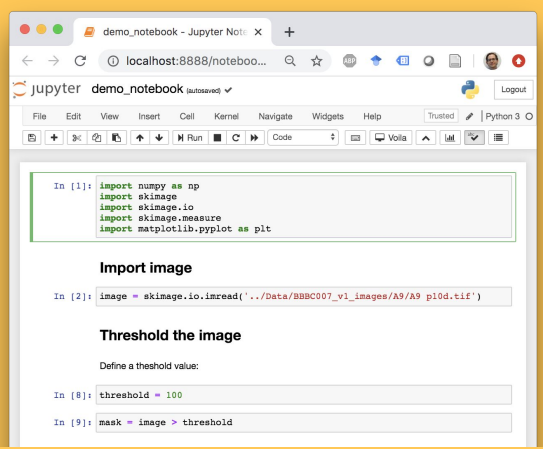
https://nbviewer.jupyter.org/github/guiwitz/neubias_academy_biapy/tree/master/



Nbviewer rendering

How does a notebook calculate?

Notebook displayed in **browser**



Sends computations

Sends result

Computations done by **kernel**

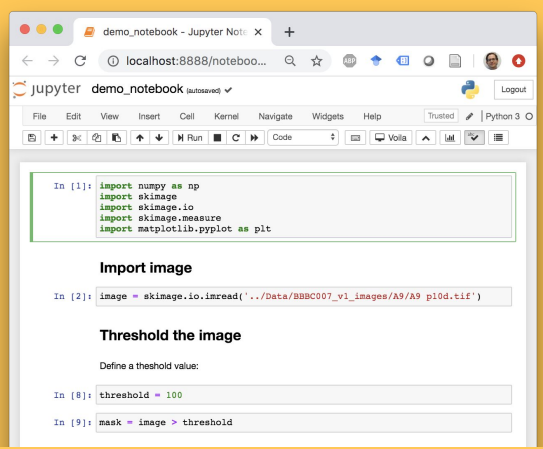
python™

Kernel can be **local** (laptop) or **remote** (cluster, Google etc,)

For you, the user, “where it runs” doesn’t affect the interface

How does a notebook calculate?

Notebook displayed in **browser**



Sends computations

Sends result

binder

CO

For you, the user, “where it runs” doesn’t affect the interface

Making notebooks interactive: binder

https://mybinder.org/

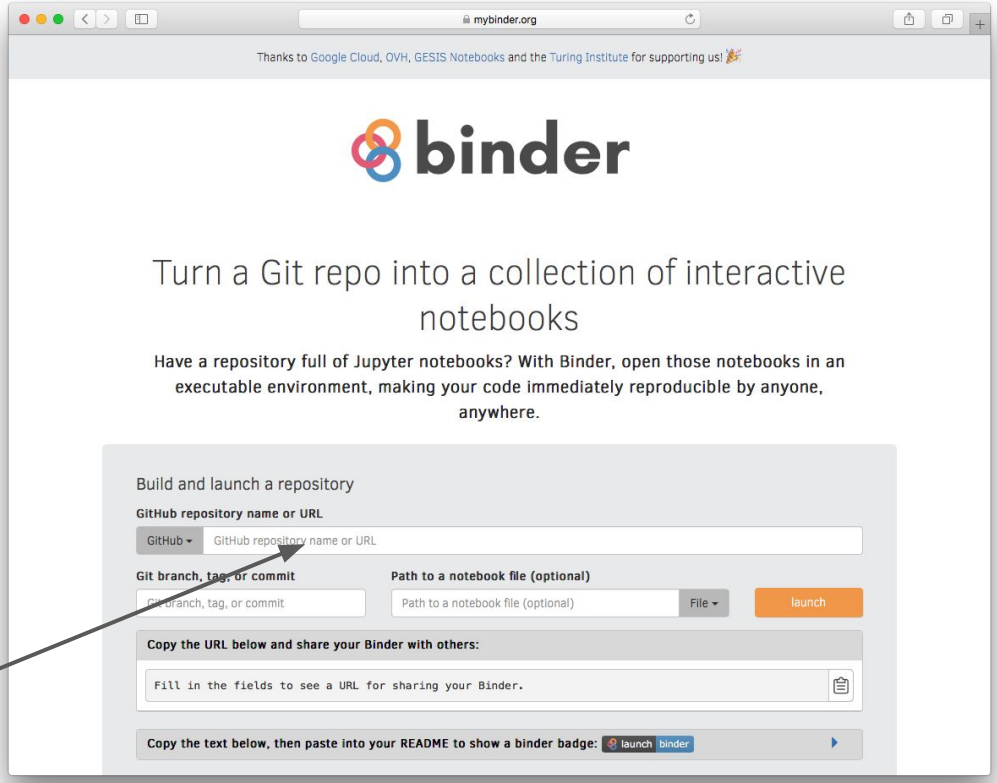
Creates remote Jupyter instance

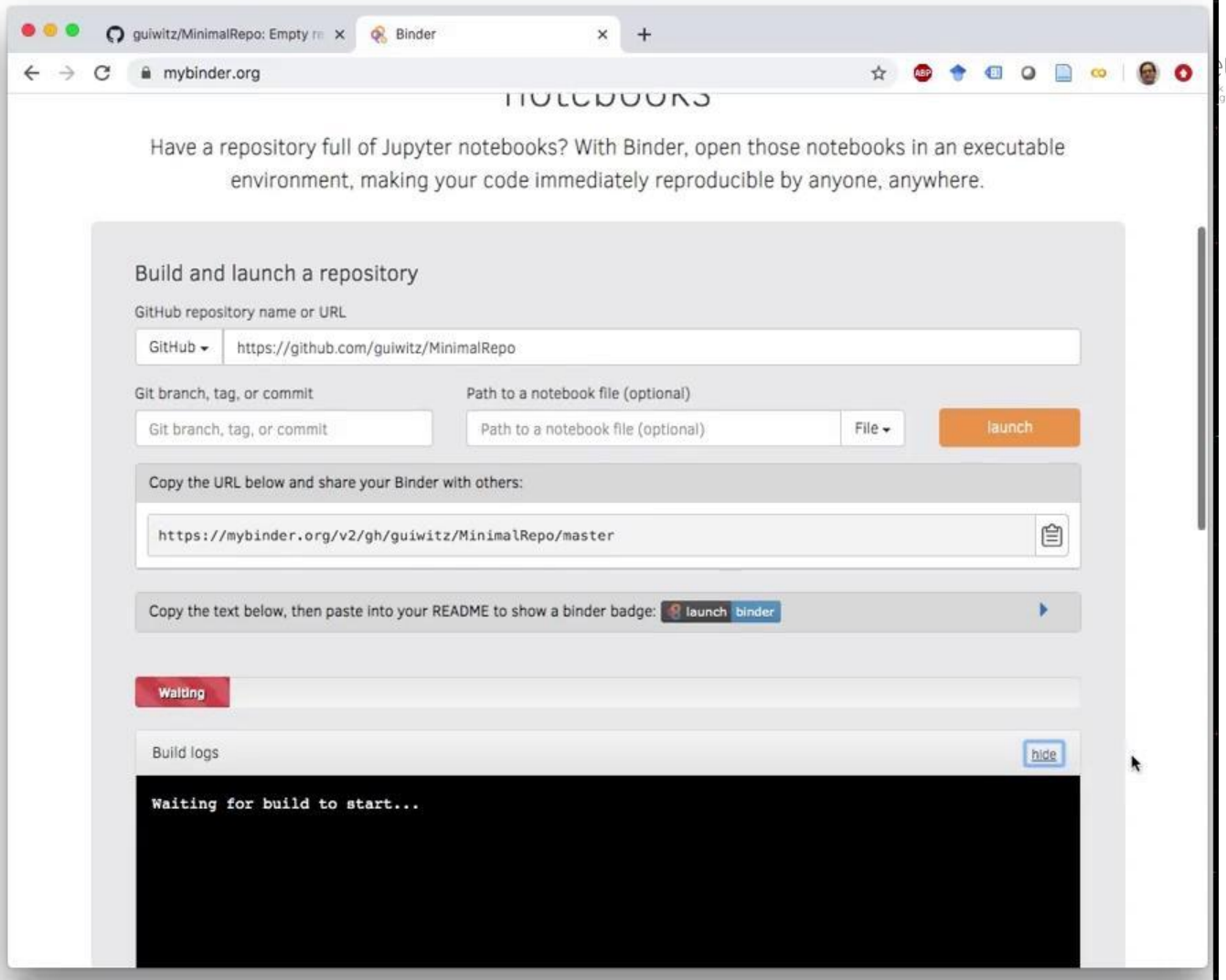
Copies content of Github repository

Opens Jupyter in the browser and **works exactly like a local Jupyter**

Try out with minimal repository (for speed):

https://github.com/guiwitz/MinimalRepo



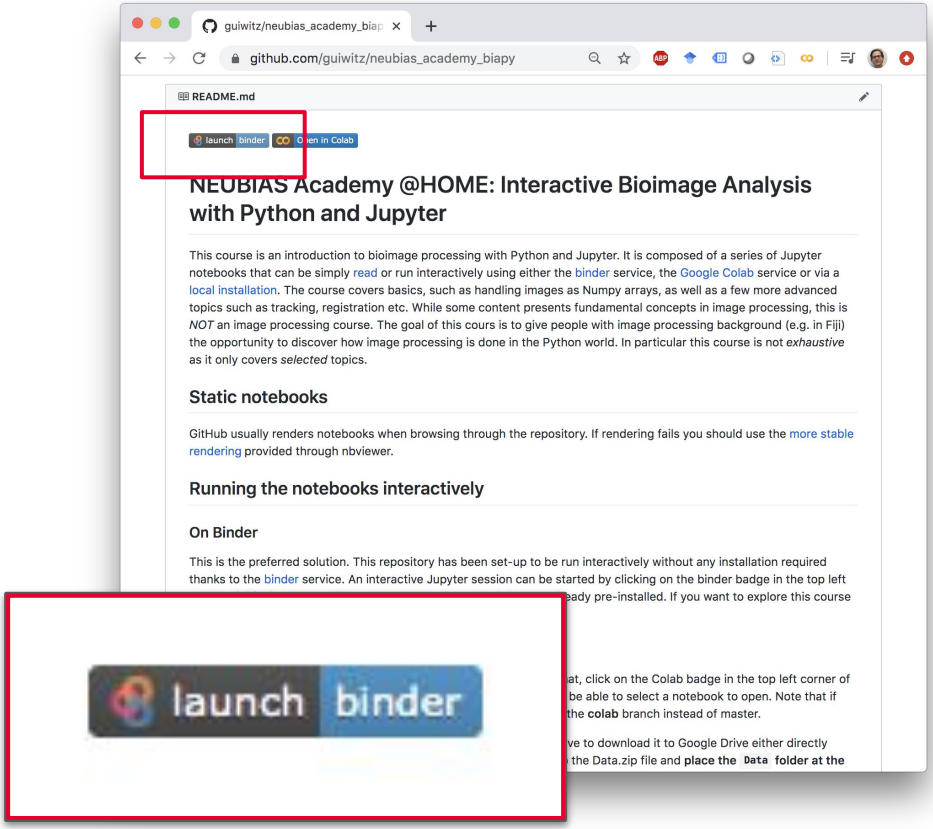

guiwitz/MinimalRepo: Empty re
Binder
mybinder.org
Have a repository full of Jupyter notebooks? With Binder, open those notebooks in an executable environment, making your code immediately reproducible by anyone, anywhere.
Build and launch a repository
GitHub repository name or URL
GitHub
https://github.com/guiwitz/MinimalRepo
Git branch, tag, or commit
Path to a notebook file (optional)
File
launch
Copy the URL below and share your Binder with others:
https://mybinder.org/v2/gh/guiwitz/MinimalRepo/master
Copy the text below, then paste into your README to show a binder badge:
launch binder
Waiting
Build logs
hide
Waiting for build to start...

Launching the course on binder

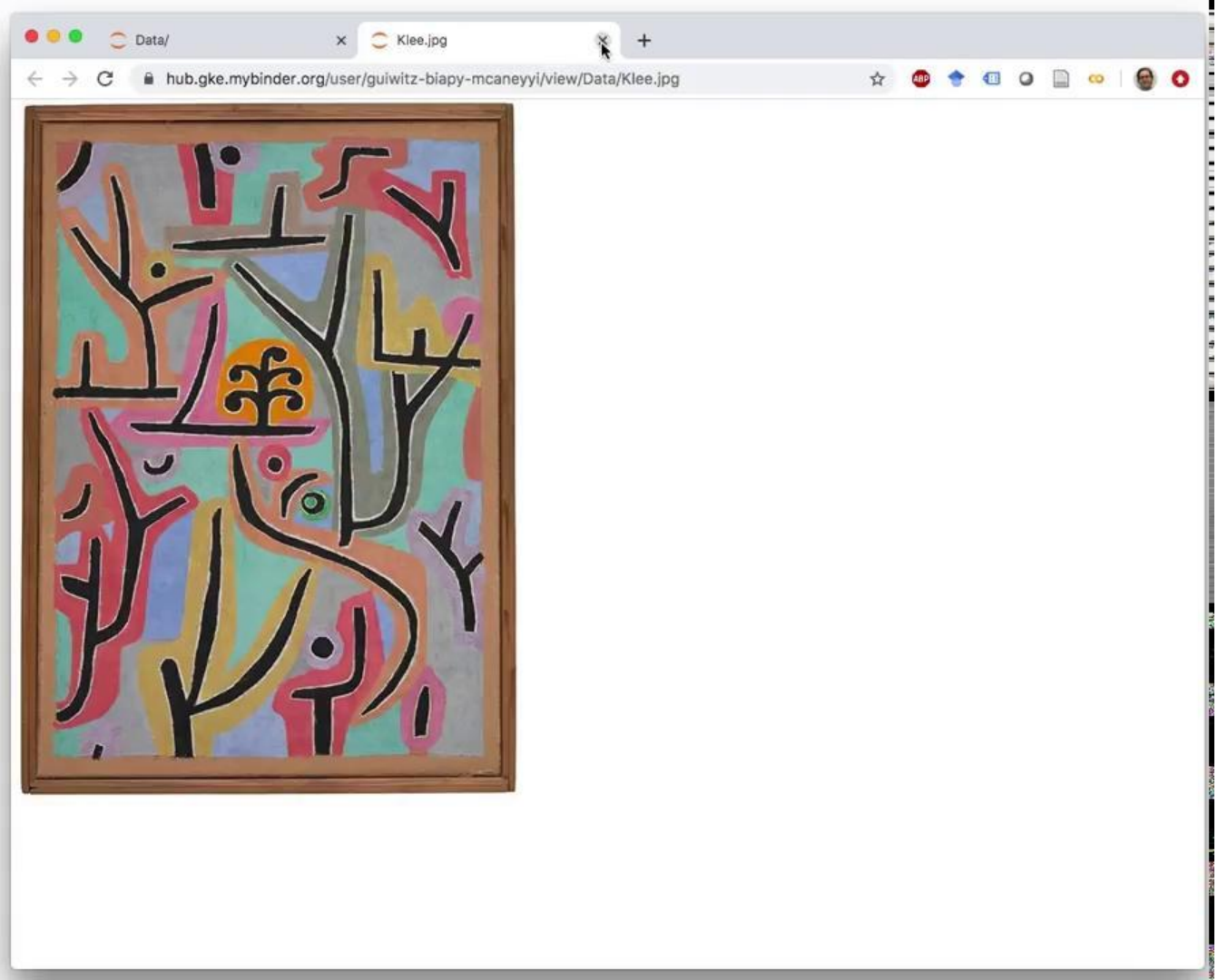
Use the “launch binder” button in the repository (replaces copying url to mybinder.org)

https://github.com/guiwitz/neubias_academy_biapy

First launch might take several minutes. Later launches are fast.



Special thanks to the mybinder.org team for increasing usage limits for this course!



neubias network of european bioimage analysts

UNIVERSITÄT BERN

Complete course data available

All packages pre-installed

Ready to go !

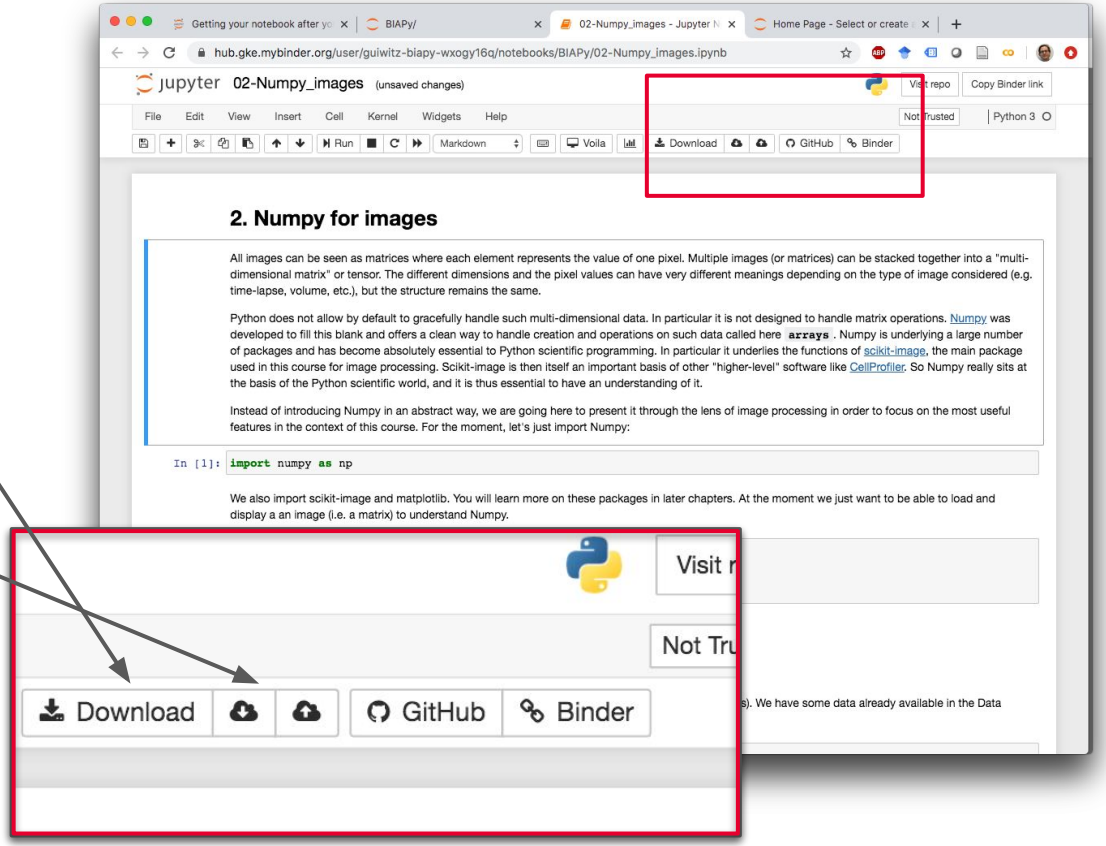
Binder sessions

Short sessions: stop after a few minutes inactivity

Download your modified notebook OR

Save/load it to/from browser storage

Does not work with “external” software (e.g. napari)



Making notebooks interactive: Colab

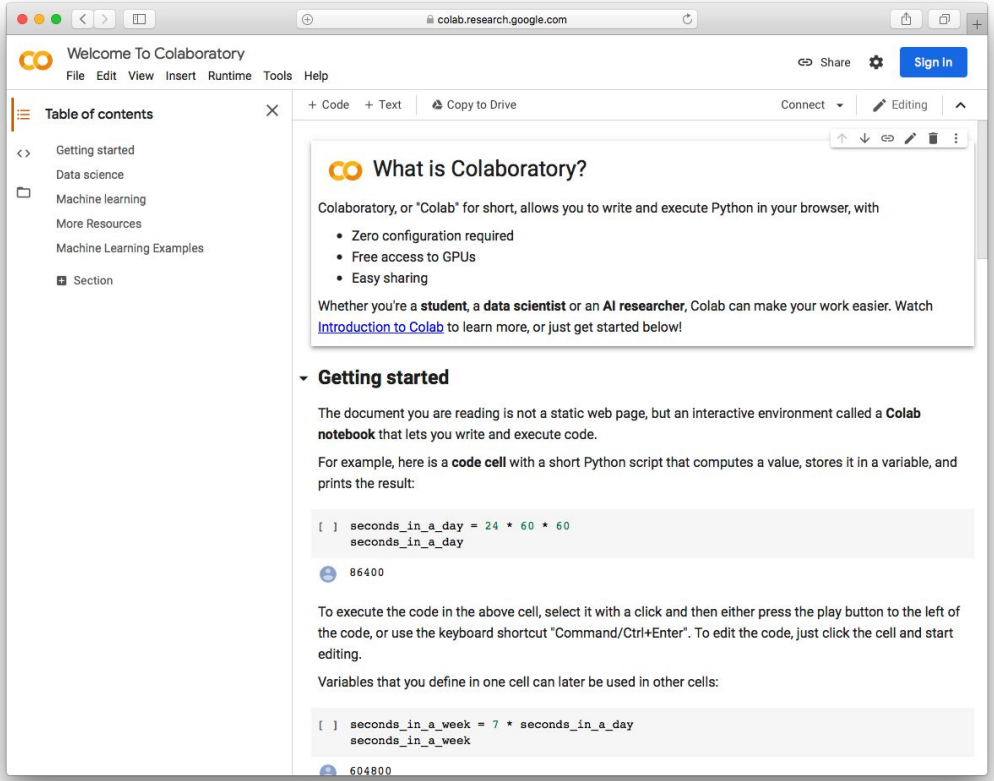
Google's version of Jupyter

Same basic principles, different layout

Kernels run on Google infrastructure for free

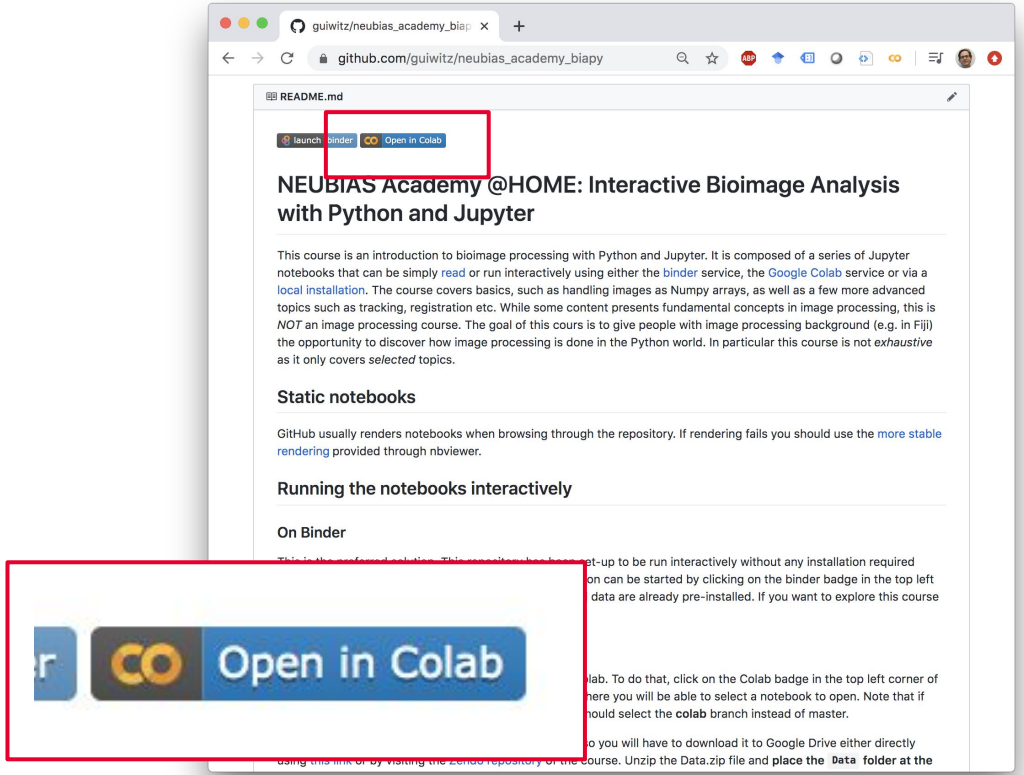
Opens any notebook on Github

https://colab.research.google.com/



Accessing course notebooks

Click on the “Open in Colab” in the repository



Accessing course notebooks

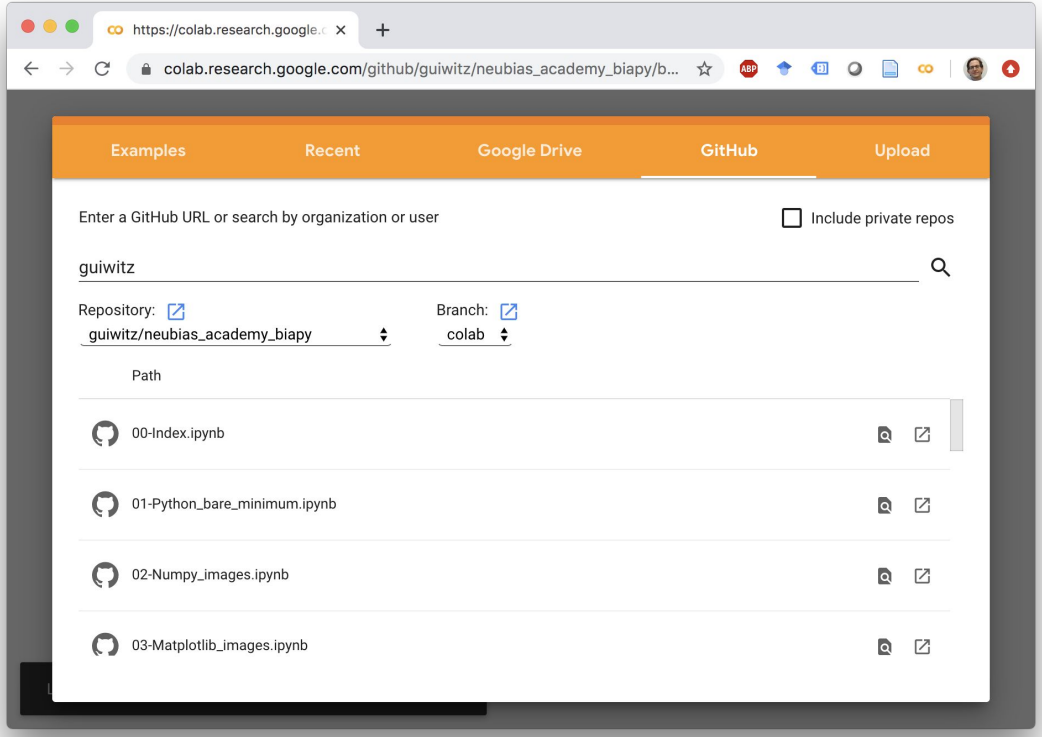
Click on the “Open in Colab” in the repository

Login if not already done

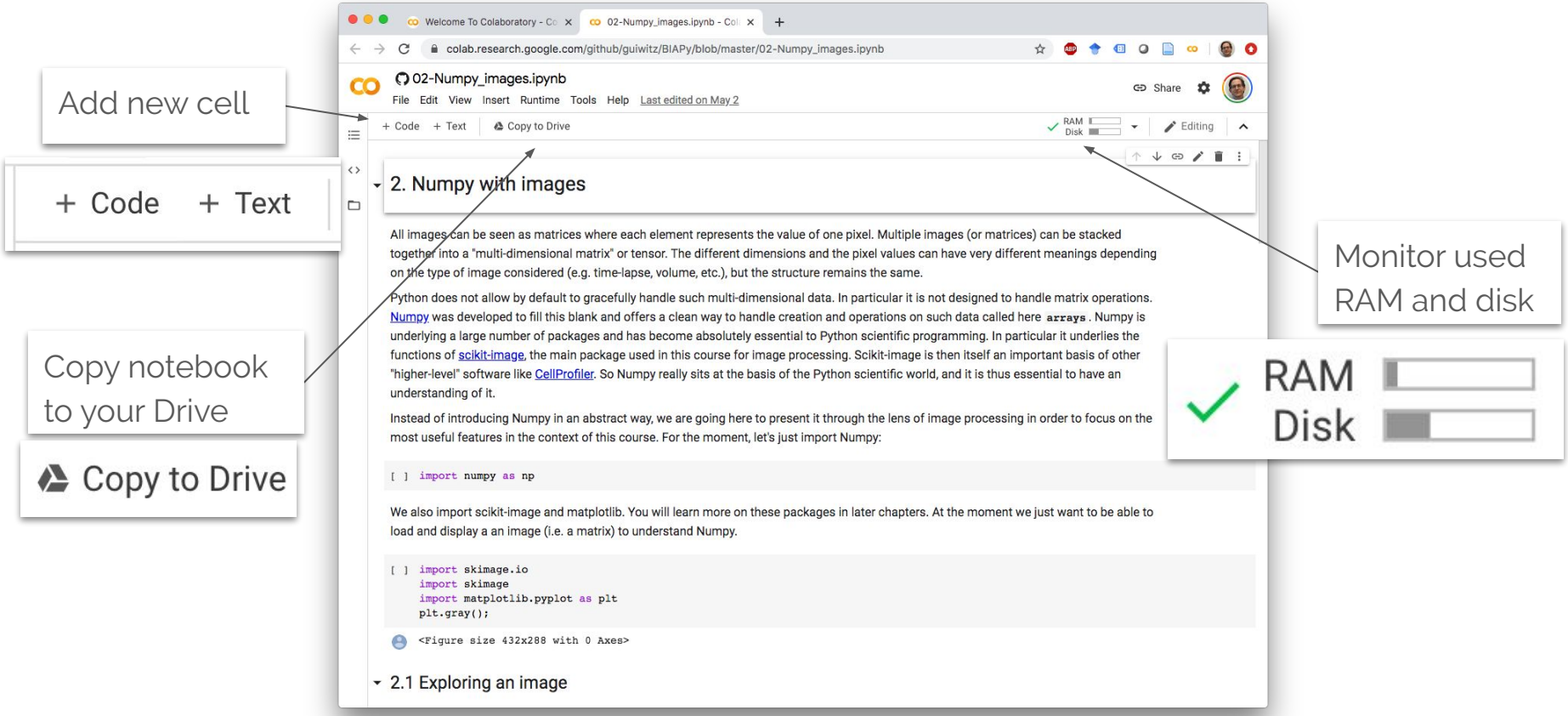
Select a notebook to run

Save the notebook to Google Drive to keep modifications

Open notebook from Google Drive directly next time



A Colab notebook



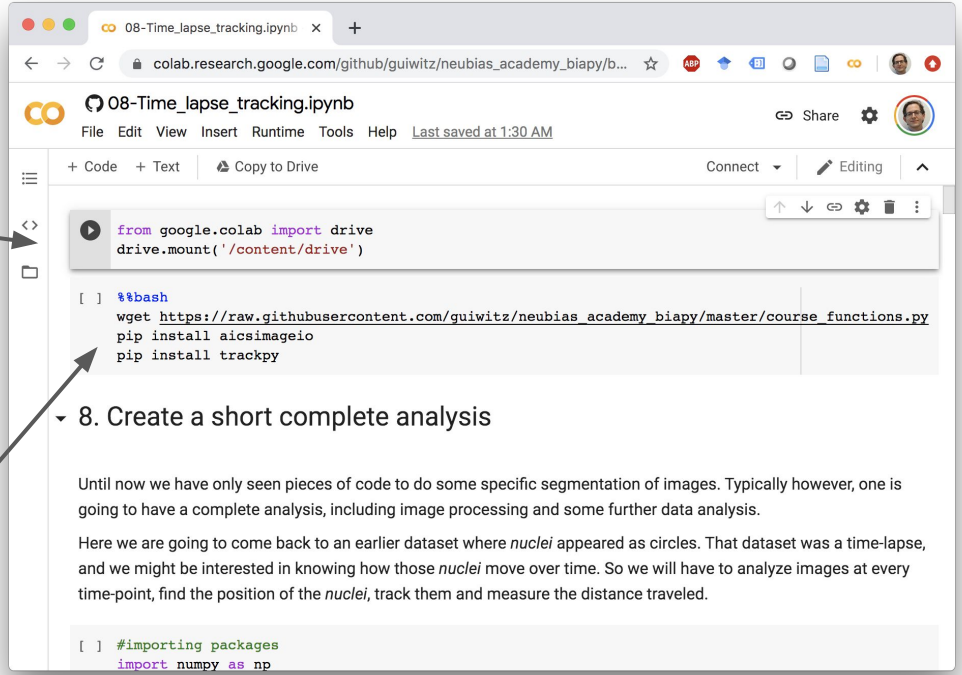
Colab sessions

Sessions time-out after max 12h

Data accessible through Google Drive (for details, see https://github.com/guiwitz/neubias_academy_biapy#On-Colab)

Common packages pre-installed

Additional packages need to be installed in each notebook



Installing packages

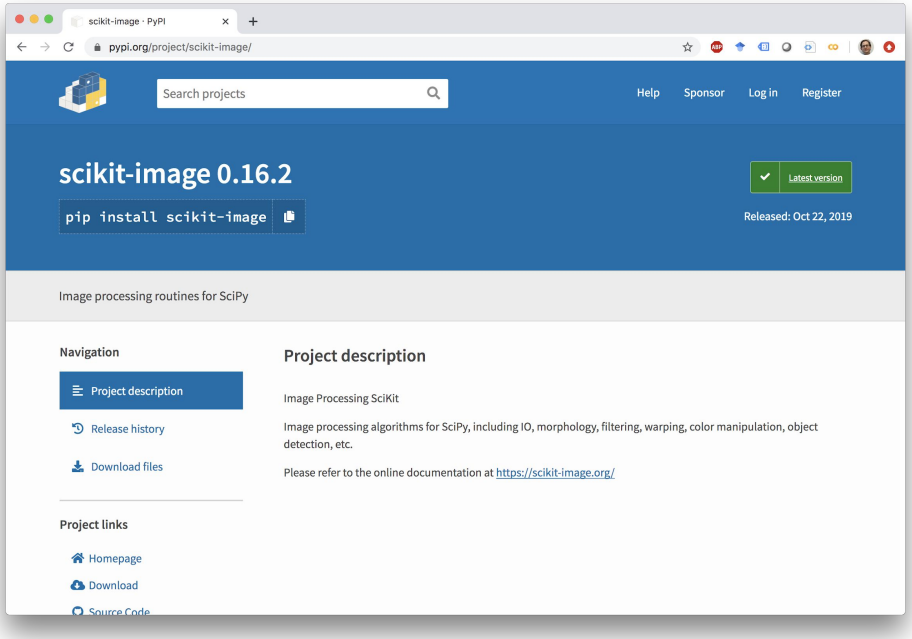
Installing packages: Pip

Pip **i**nstalls **p**ackages Pip

Widely used package manager

Installation via Command line OR Notebook : `pip install mypackage`

Pip installs the package AND its dependencies, i.e. other packages it depends on.



Example: install scikit-image from Jupyter

Use **!** sign in notebook for command line actions:

```
!pip install scikit-image
```

Install multiple packages at once:

```
!pip install scikit-image pandas
```

Install specific versions:

```
!pip install scikit-image==1.15
```

Install packages: conda

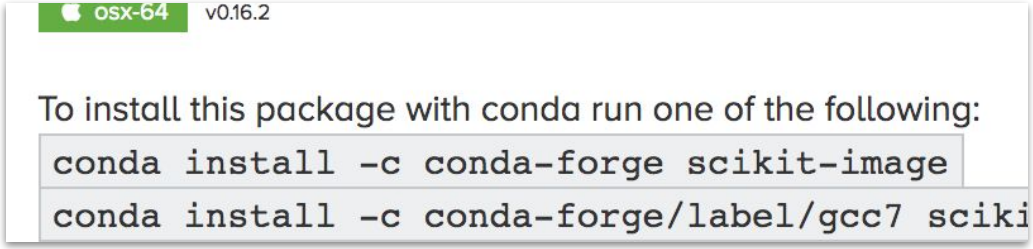
Conda also installs packages. E.g.:

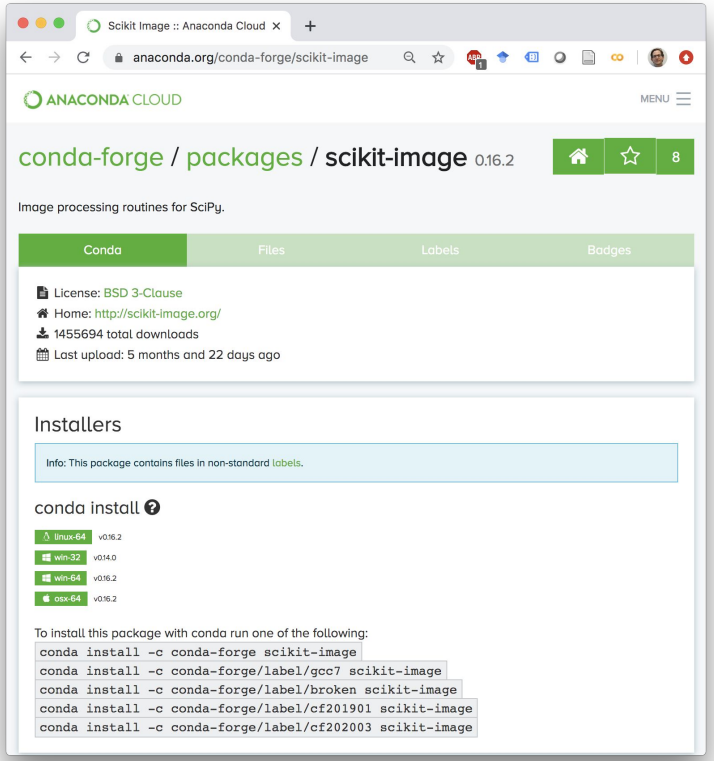
```
conda install -c conda-forge scikit-image
```

Looks for “best” version combinations

Not limited to Python (e.g. ffmpeg, CUDA etc.)

Create isolated environments for each project





Conda user-interface: Anaconda navigator

Install packages and create environments via a user-interface

Not just for Jupyter: Rstudio, Spyder etc. included

Available on all OS

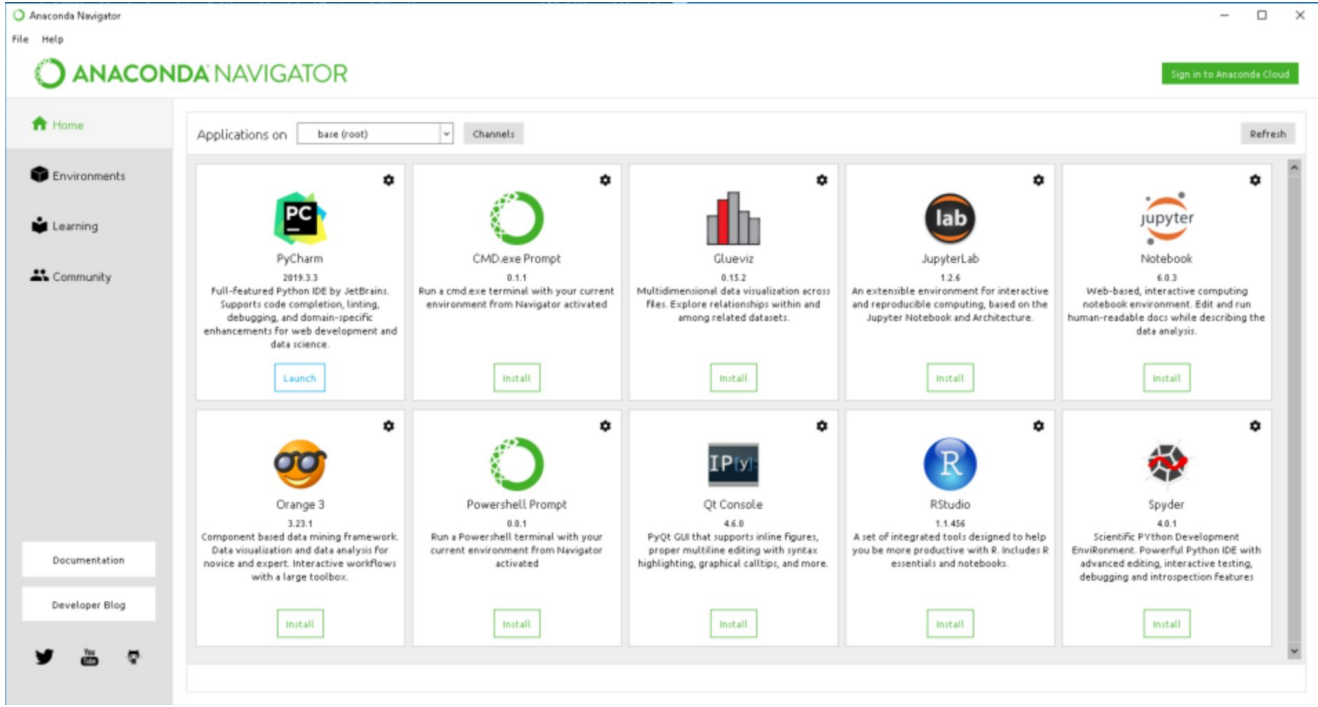
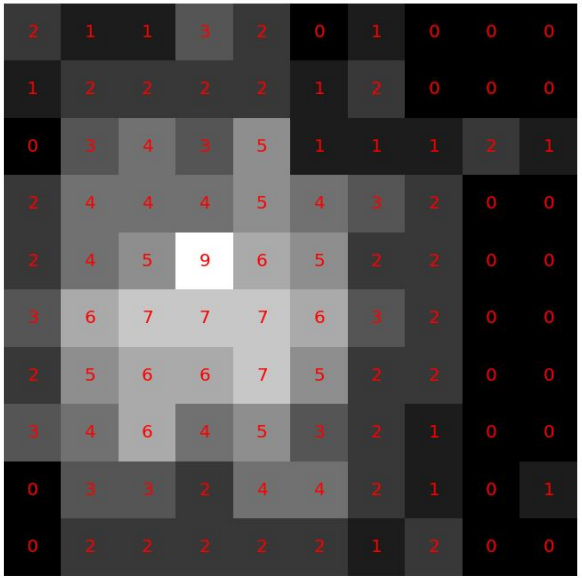


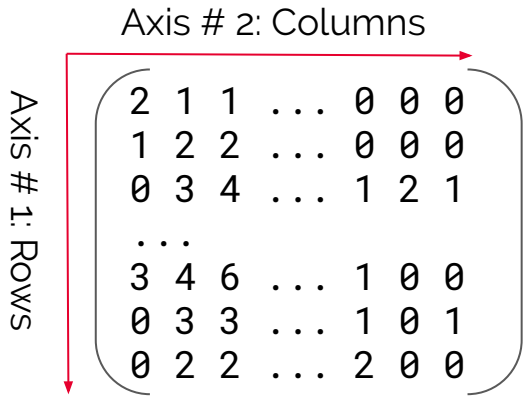
Image processing

aicsimageio, ipyvolume, napari, ipywidgets, trackpy, cellpose, stardist

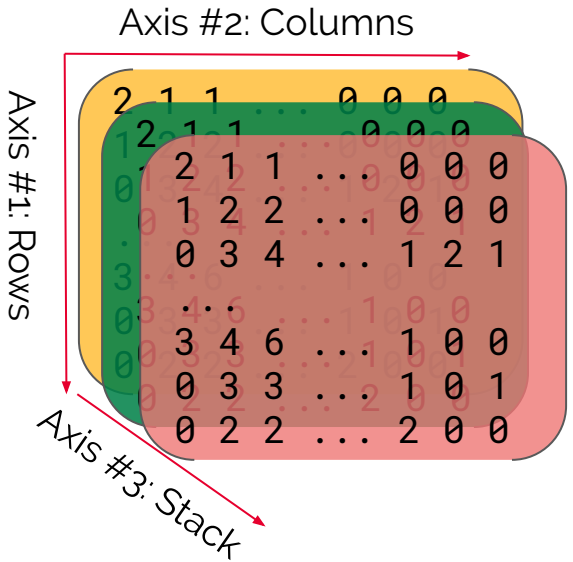
The array objects: Numpy



An image is just a matrix of numbers



Numpy 2D array



Numpy 3D array

In Scientific Python, almost all computations are done on Numpy arrays

Computations on Numpy arrays

Indexing:

```
my_array =
[[2 1 1]
 [1 2 2]
 [0 3 3]]
```

Extract single "pixel"
Row = 0, Column = 2

```
my_array[0,2]
->
1
```

Extract complete "pixel" row
Row = 1, Column = All (:)

```
my_array[1,:]
->
[1 2 2]
```

Combining arrays:
Pixel by pixel

$$\begin{pmatrix} 2 & 1 & 7 \\ 1 & 0 & 2 \\ 5 & 3 & 4 \end{pmatrix} * \begin{pmatrix} 3 & 1 & 1 \\ 2 & 5 & 2 \\ 0 & 3 & 1 \end{pmatrix} = \begin{pmatrix} 6 & 1 & 7 \\ 2 & 0 & 4 \\ 0 & 9 & 4 \end{pmatrix}$$

Mathematics on arrays:
Pixel by pixel

$$\text{np.cos}\left(\begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 2 \\ 0 & 3 & 3 \end{pmatrix}\right) = \begin{pmatrix} -0.42 & 0.54 & 0.54 \\ 0.54 & -0.42 & -0.42 \\ 1 & -0.99 & -0.99 \end{pmatrix}$$

Image processing in Python: scikit-image

"The project aims are: 1. To provide high quality, well-documented and easy-to-use implementations of common image processing algorithms…"[1]

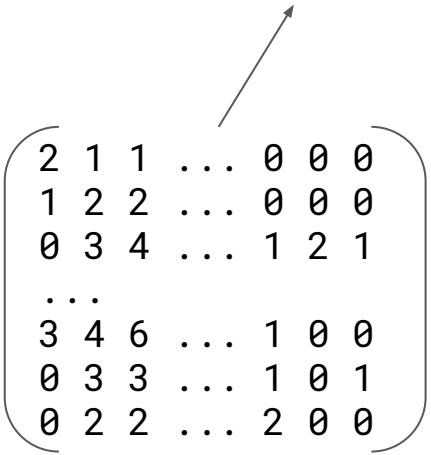
[1]Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu and the scikit-image contributors. scikit-image: Image processing in Python. PeerJ 2:e453 (2014)

Image processing in Python: scikit-image

"The project aims are: 1. To provide high quality, well-documented and easy-to-use implementations of common image processing algorithms..."[1]

Import your image

```
image = skimage.io.imread('myimage.tiff')
```



Numpy array

[1]Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu and the scikit-image contributors. scikit-image: Image processing in Python. PeerJ 2:e453 (2014)

Image processing in Python: scikit-image

"The project aims are: 1. To provide high quality, well-documented and easy-to-use implementations of common image processing algorithms..."[1]

Import your image

```
image = skimage.io.imread('myimage.tiff')
```

Gaussian filter your image

```
im_gauss = skimage.filters.gaussian(image, sigma=2, preserve_range=True)
```

$$\texttt{skimage.filters.gaussian}\left(\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 2 \\ 0 & 3 & 3 \end{bmatrix}\right) = \begin{bmatrix} 1.5 & 1.5 & 1.6 \\ 1.4 & 1.6 & 1.8 \\ 1.3 & 1.7 & 2.0 \end{bmatrix}$$

Numpy array Numpy array

[1]Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu and the scikit-image contributors. scikit-image: Image processing in Python. PeerJ 2:e453 (2014)

Image processing in Python: scikit-image

"The project aims are: 1. To provide high quality, well-documented and easy-to-use implementations of common image processing algorithms..."[1]

Import your image	`image = skimage.io.imread('myimage.tiff')`
Gaussian filter your image	`im_gauss = skimage.filters.gaussian(image, sigma=2, preserve_range=True)`
Rotate an image by 90˚	`skimage.transform.rotate(image, angle=90)`
Erode a binary mask with a disk	`skimage.morphology.binary_erosion(mask, selem=disk)`
Measure area and label of a labelled image	`skimage.measure.regionprops_table(labelled, properties=('label','area'))`
Detect a template in an image	`skimage.feature.match_template(image, template=T)`
Segment and image with active contours	`skimage.segmentation.active_contour(image, snake=init_contour)`

https://scikit-image.org/docs/stable/api/api.html

[1]Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu and the scikit-image contributors. scikit-image: Image processing in Python. PeerJ 2:e453 (2014)

Matplotlib for images

Oldest Python plotting library

Widely used, also basis of other libraries

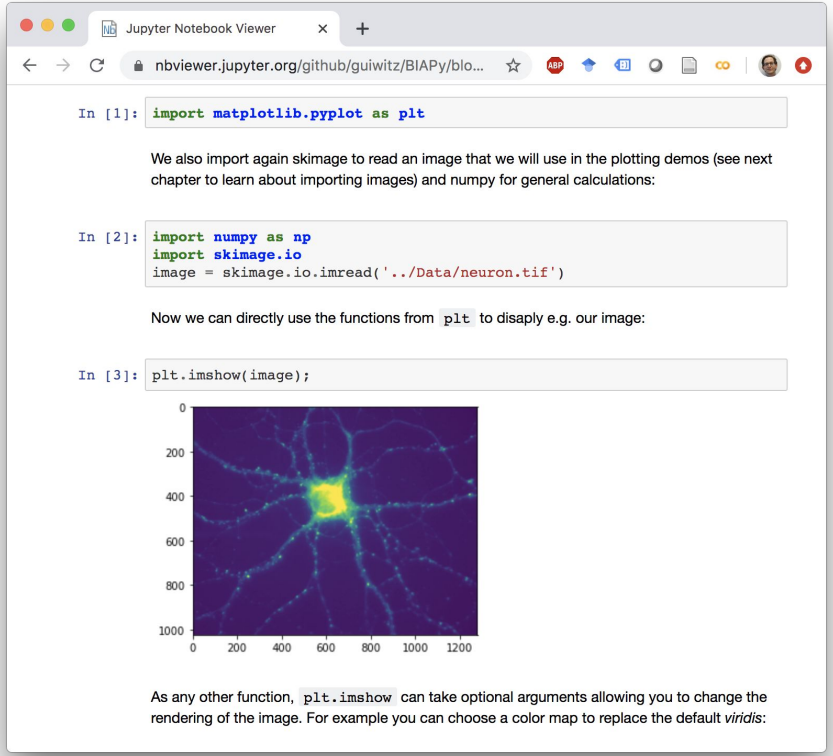
Works with Numpy arrays

Allows for extensive plot customization

Used in minimal way in this course

Easy solution for plotting images:

```
plt.imshow(my_2Dnumpy_array)
```



https://nbviewer.jupyter.org/github/guiwitz/neubias_academy_biapy/blob/master/03-Matplotlib_images.ipynb

Additional packages used in the course

napari: high-performance, user-friendly volume viewer (only for local installations): https://napari.org/

AICSImageIO: image importer from Allen Institute of Cell Science. Practical handling of metadata: https://allencellmodeling.github.io/aicsimageio/

Trackpy: very complete particle tracker: https://github.com/soft-matter/trackpy

Ipyvolume: in browser 3D volume rendering (only in Binder): https://github.com/maartenbreddels/ipyvolume

Ipywidgets: easy creation of interactive components in notebooks: https://ipywidgets.readthedocs.io/en/stable/

StarDist: ML based object segmentation: https://github.com/mpicbg-csbd/stardist (see this NEUBIAS course: https://www.youtube.com/watch?v=Amn_eHRGX5M)

Cellpose: generalistic ML based nuclei and cell segmentation: http://www.github.com/mouseland/cellpose

PyImageJ: Mixing Fiji and Python code in notebooks: https://github.com/imagej/pyimagej

napari

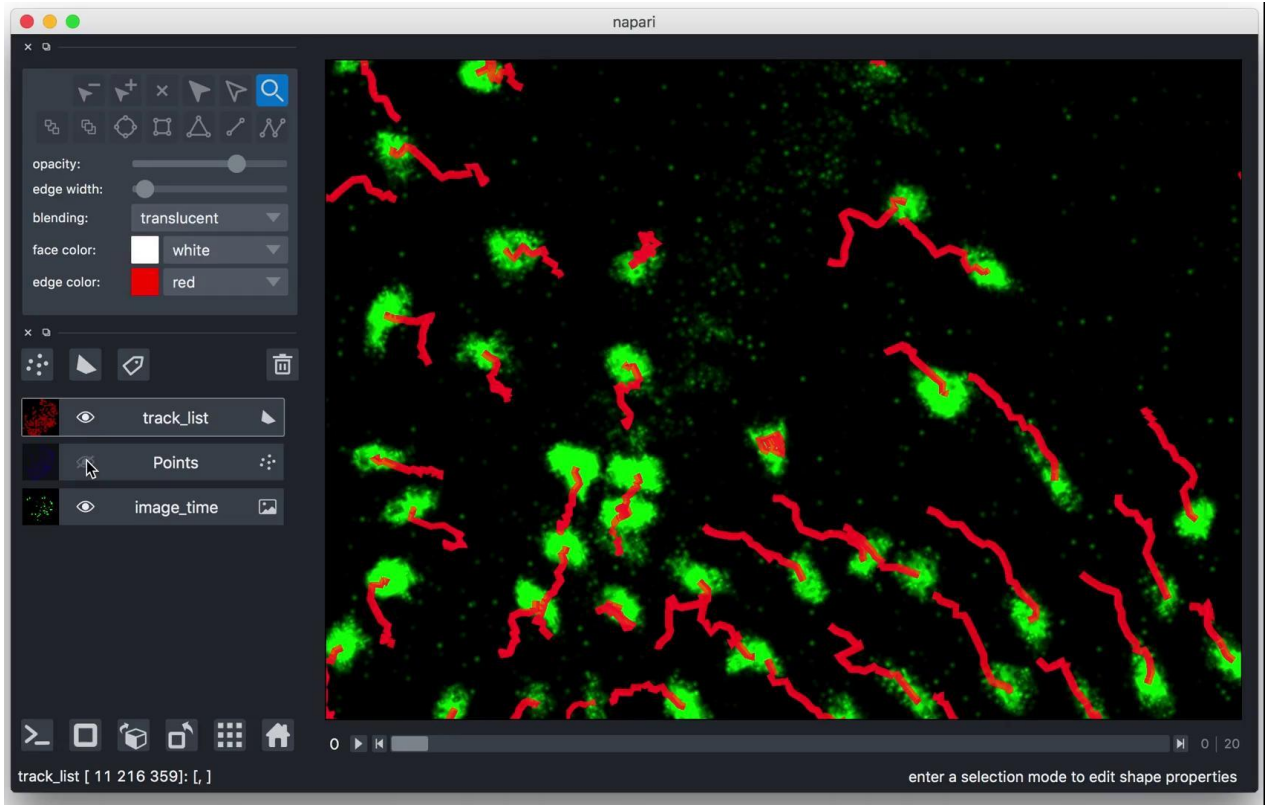
Handles 5D data

Volumetric rendering

Other objects: labels, shapes etc.

Easy to add custom interactions (selecting objects, drawing etc.)

Tracks made with trackpy



https://nbviewer.jupyter.org/github/guiwitz/neubias_academy_biapy/blob/master/11-3D_image_processing.ipynb

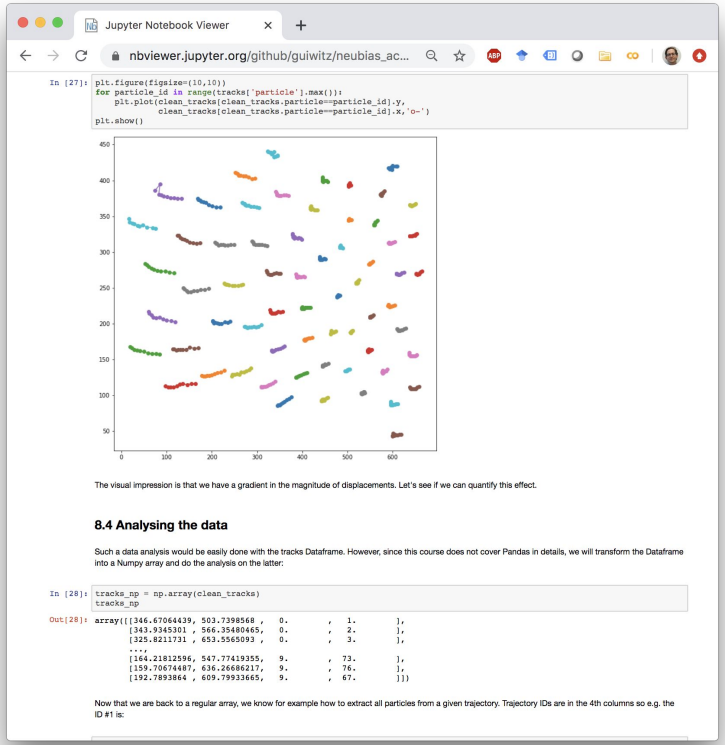
trackpy

Originally developed for particle tracking

User-friendly Pandas-based input/output

Also spot detection algorithms available

Multiple algorithms and settings to choose out of



https://nbviewer.jupyter.org/github/guiwitz/neubias_academy_biapy/blob/master/08-Time_lapse_tracking.ipynb

ipyvolume

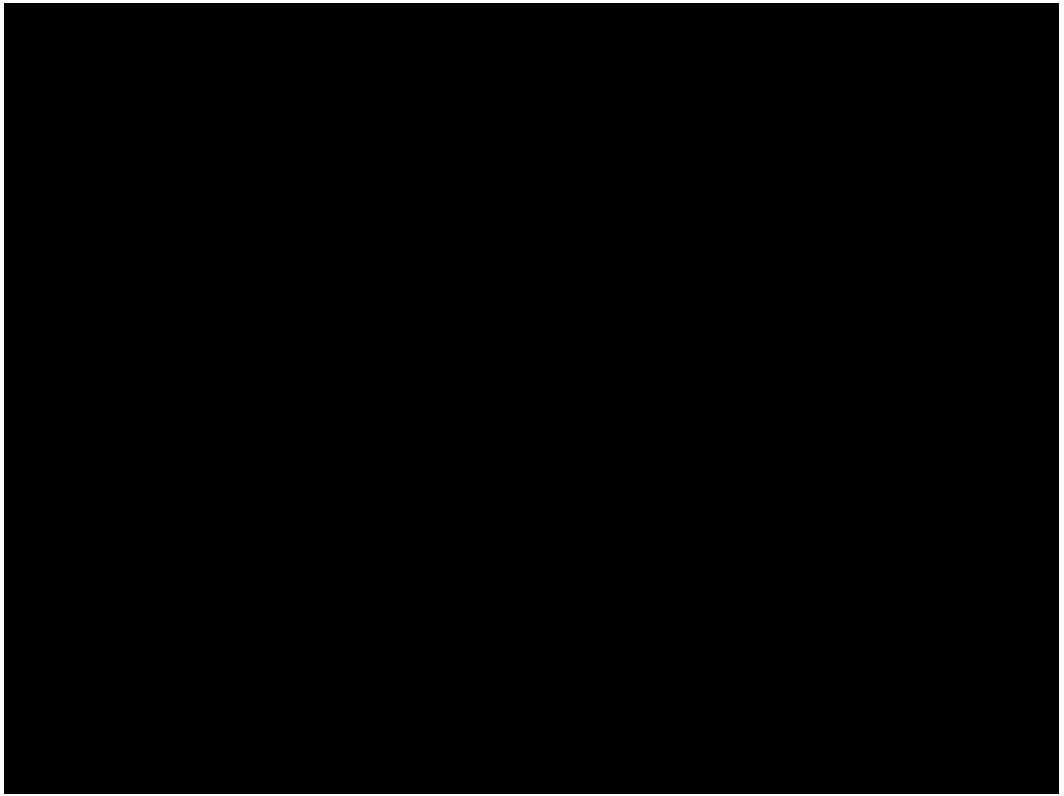
Purely browser based rendering

Volumetric rendering

Other objects: scatter plots, meshes etc.

Possible to add interactivity with ipywidgets

Export interactive figures as html!



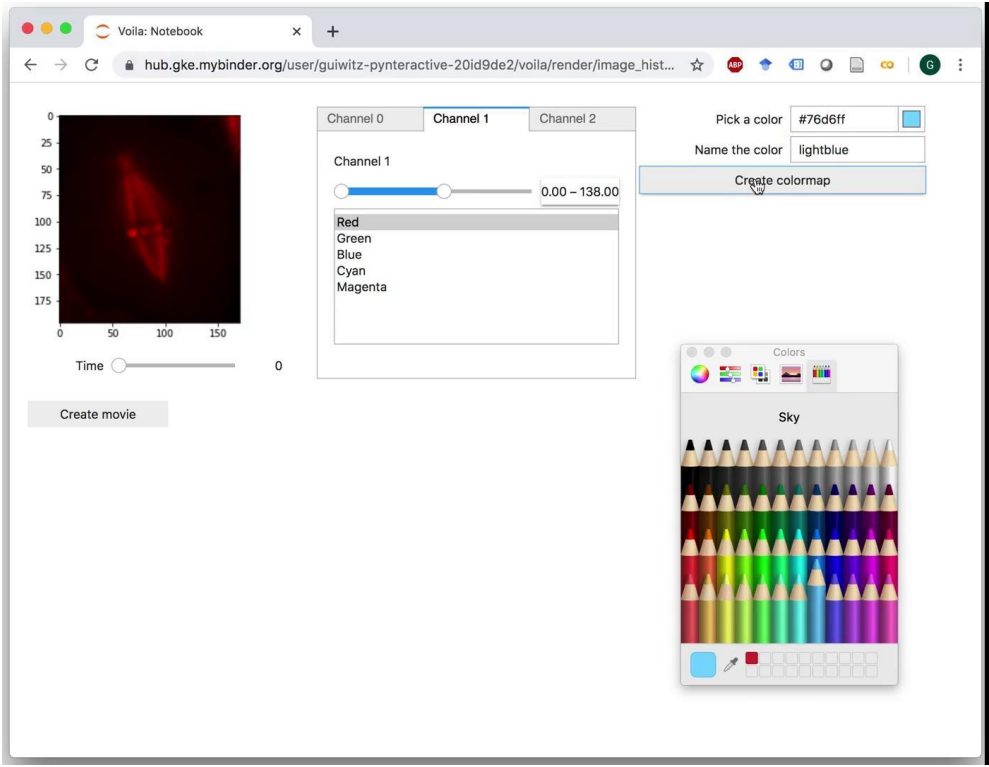
ipywidgets

Create interfaces in notebooks: buttons, sliders, etc.

Easy to connect widgets to actions

Possibility to create web-apps using voilà: https://github.com/voila-dashboards/voila

Used by other packages (ipyvolume)

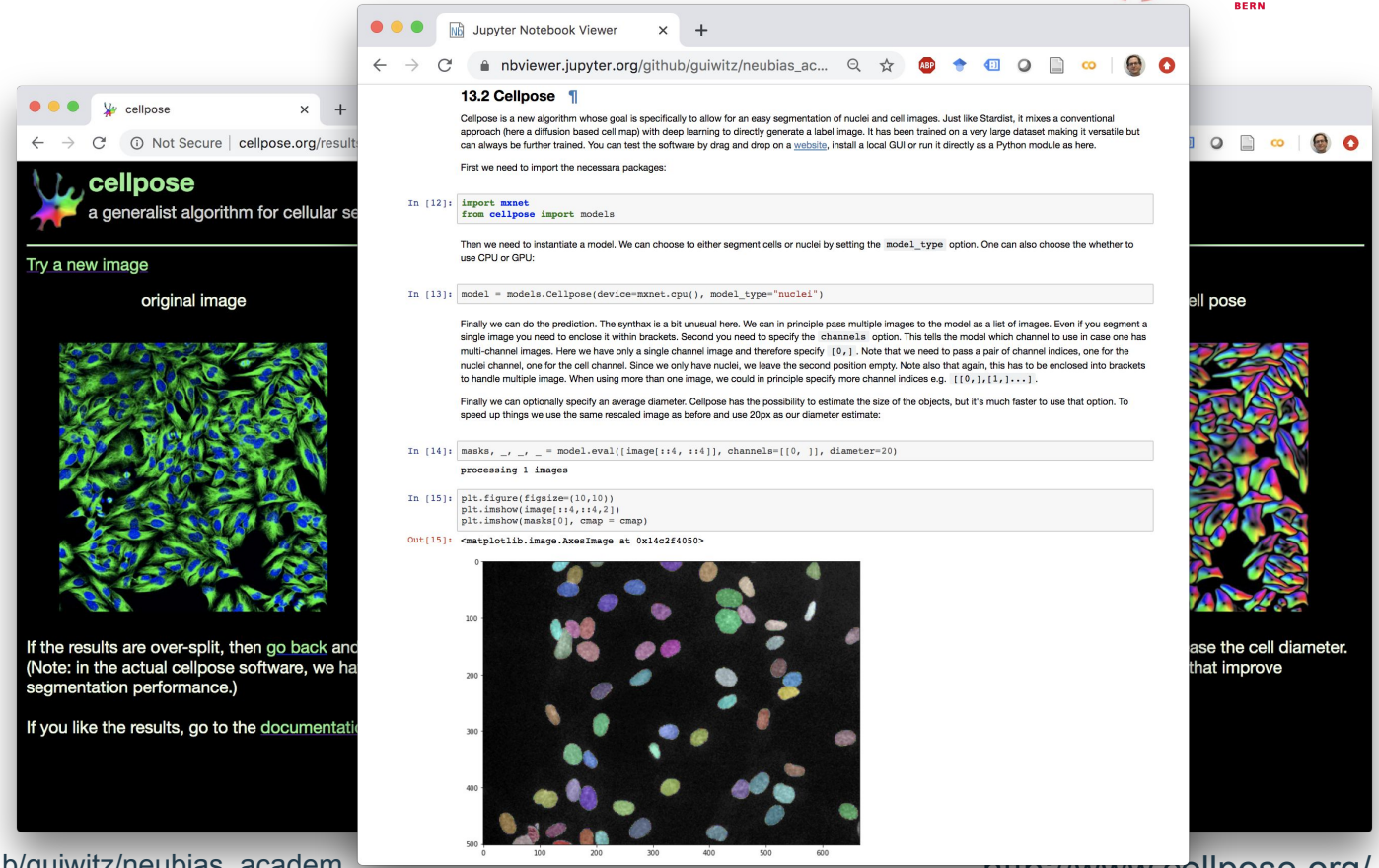


Cellpose

Pre-trained on large image database: very flexible

Predicts labels, not just masks

Can be used as Python package



https://nbviewer.jupyter.org/github/guiwitz/neubias_academy_biapy/blob/master/13-ML_based_segmentation.ipynb

http://www.cellpose.org/

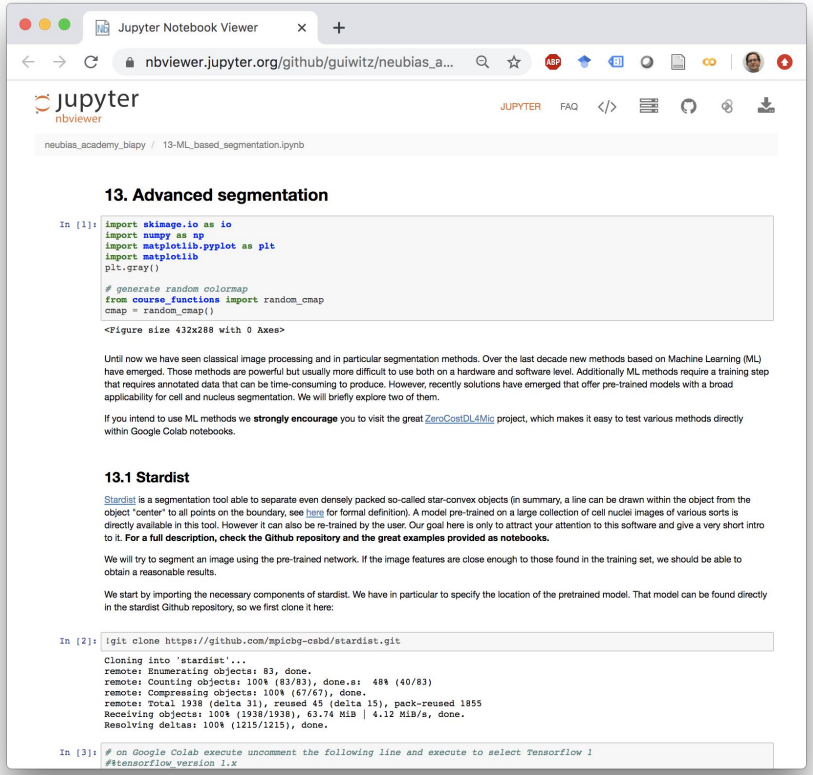
StarDist

Generalistic object segmentation tool

Segments star-convex objects (all contour points accessible from point in the object)

Predicts labels, not just masks

Can be re-trained for specific applications (for details see https://www.youtube.com/watch?v=Amn_eHRGX5M)



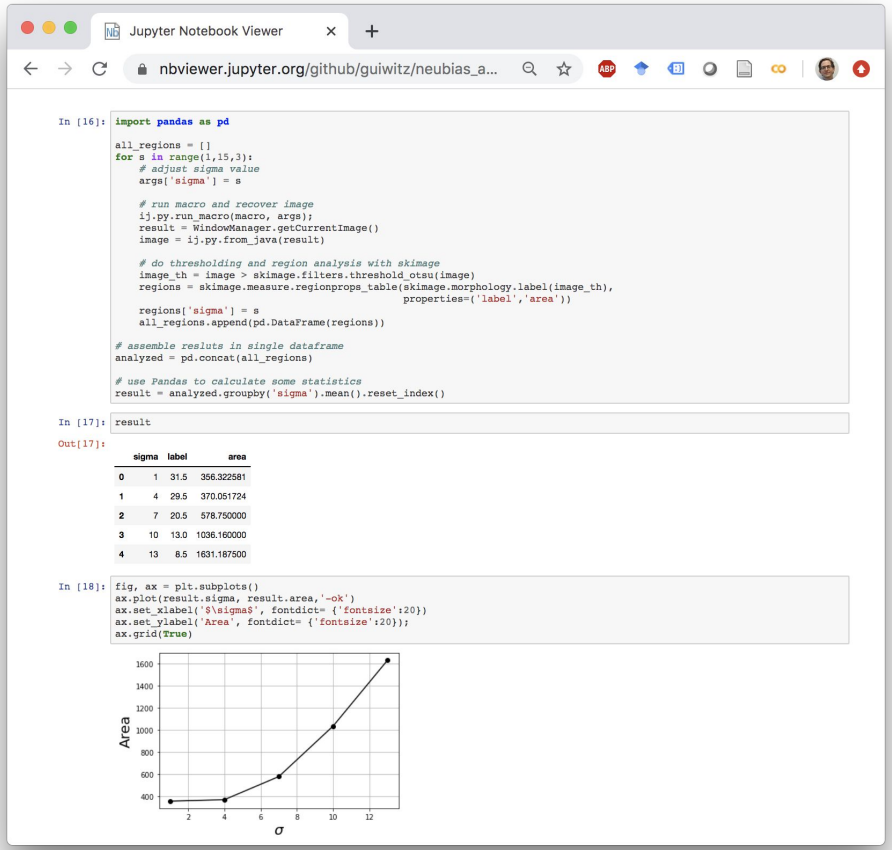
https://nbviewer.jupyter.org/github/guiwitz/neubias_academy_biapy/blob/master/13-ML_based_segmentation.ipynb

PyimageJ

Package allowing communication between Fiji and Python

Downloads Fiji or uses your local version (and plugins)

Re-use your macros within a more general Python pipeline



https://nbviewer.jupyter.org/github/guiwitz/neubias_academy_biapy/blob/master/14-PyImageJ.ipynb

Thanks for your attention, see you in a week!

A big thank you to all the developers of these open-source tools. Don't forget to cite them in your articles!

Also a big thank you to all the people publicly releasing their data!

Used to create this presentation

Color palette: https://coolors.co/e6002e-ffc857-ef7674-233d4d-008148

Icons: https://material.io/

Creating code for slides: https://romannurik.github.io/SlidesCodeHighlighter/