

About this Webinar



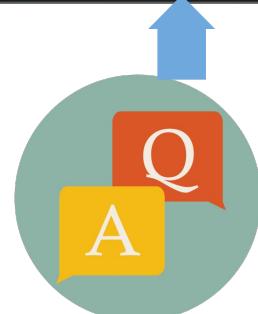
Your microphone is muted

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



Do not use the *chat*

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



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



No video

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

Zoom control panel when in a webinar (Participant view)

Leave Meeting



Leaving and coming back
will reset the Q&As panel

This webinar will be recorded and
uploaded to Youtube NEUBIAS Channel

NEUBIAS Academy @Home:

Interactive Bioimage Analysis with Python and Jupyter

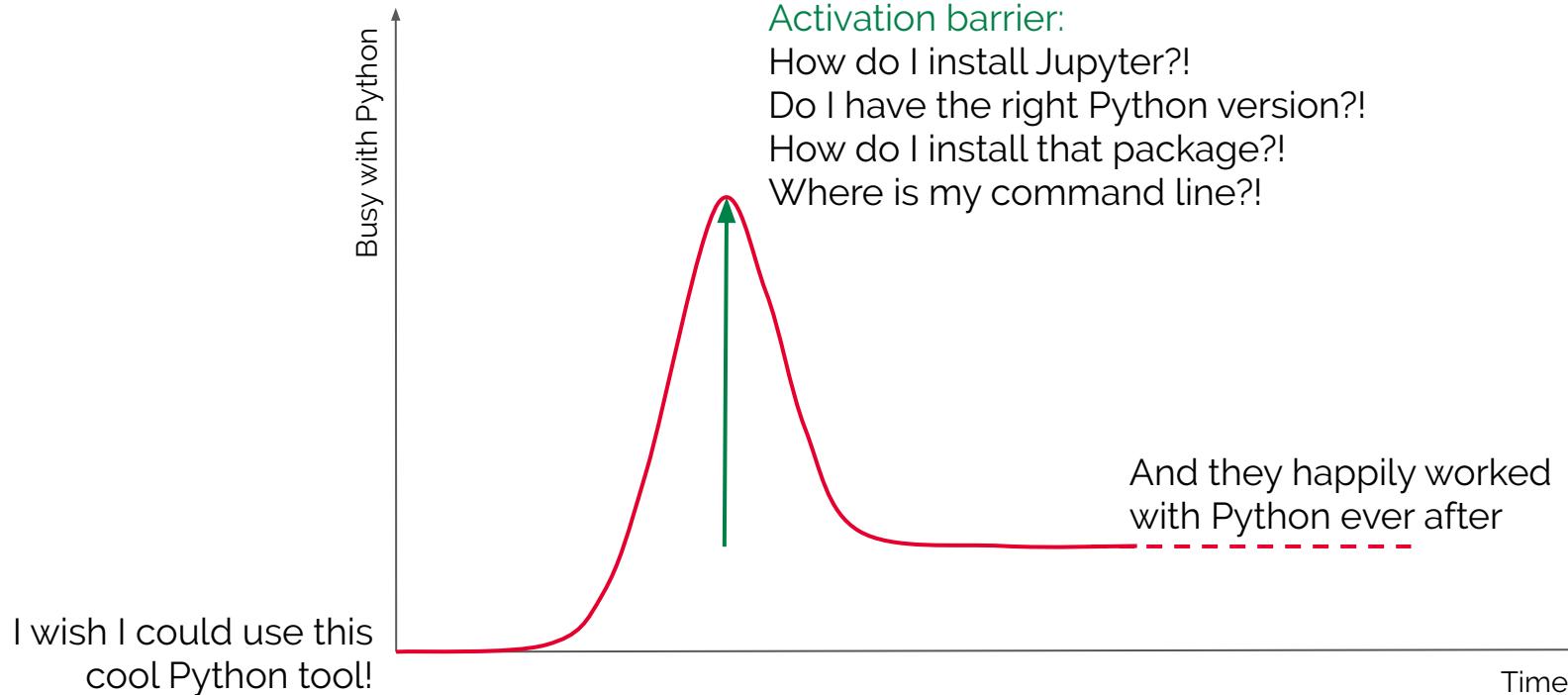
Guillaume Witz

Microscopy Imaging Center, Science IT Support
Bern University

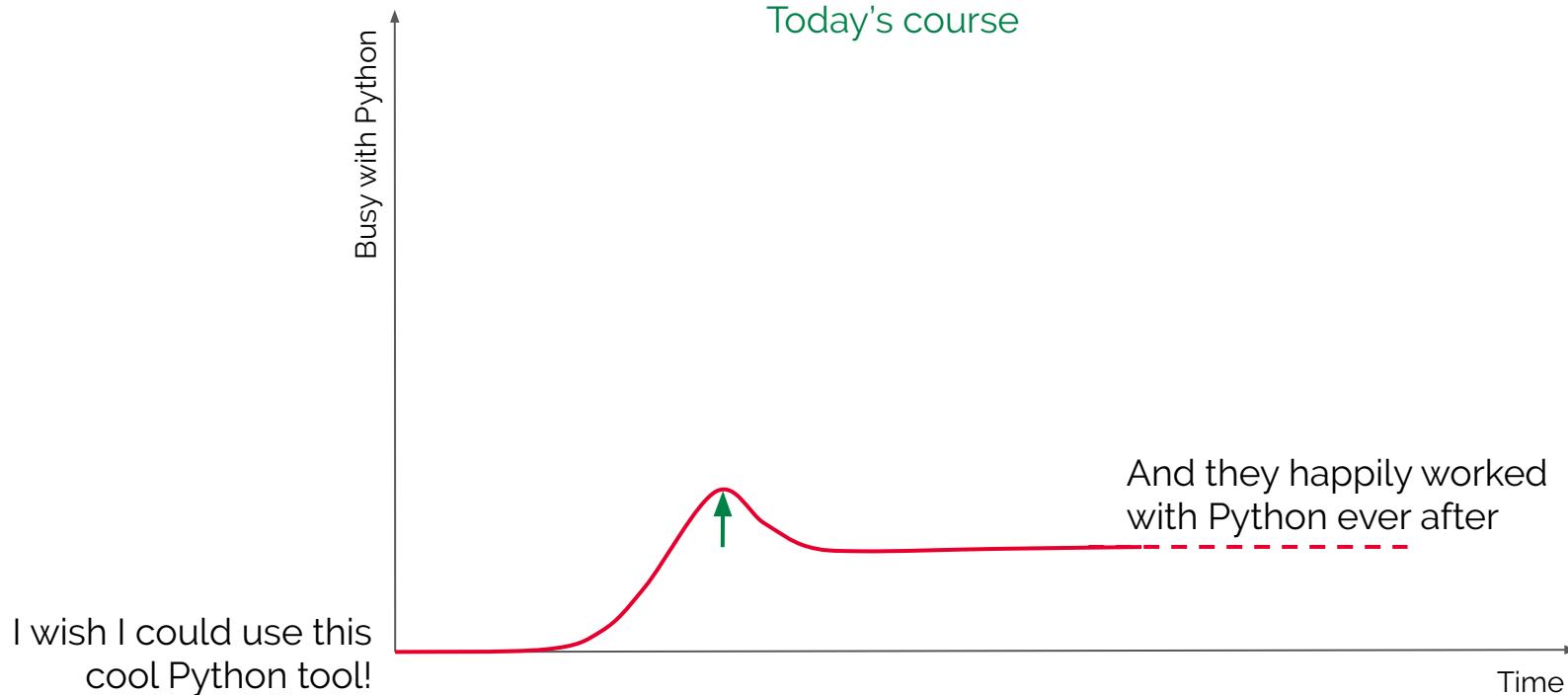


ScITS
Science IT Support

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



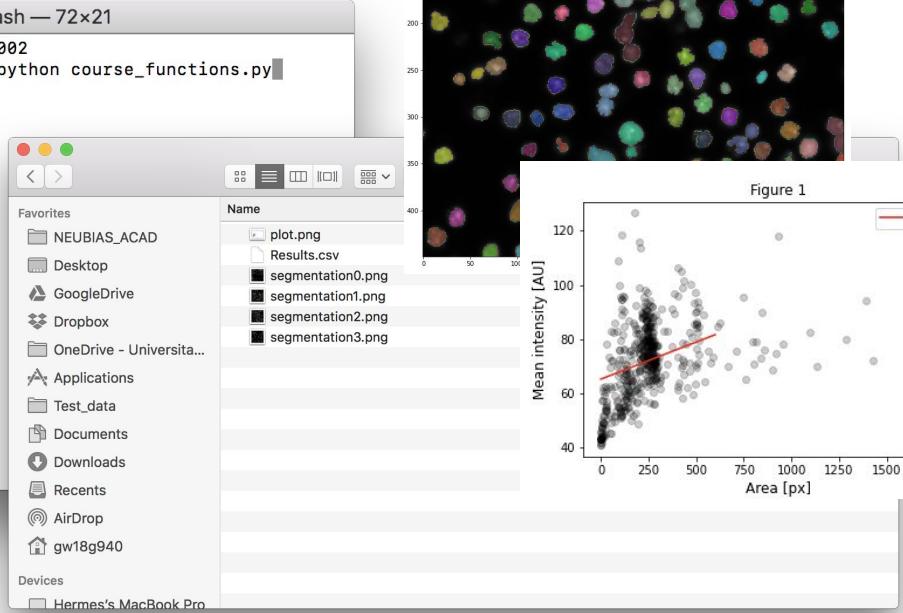
“Classic” software vs. notebooks

```
course_functions.py <input>
```

```
course_functions.py
1 import skimage.morphology
2 import skimage.filters
3 import numpy as np
4 from skimage.measure import label, regionprops_table
5 import matplotlib
6
7 def detect_nuclei(image, size_min=100, size_max=1000):
8     # filtering
9     image = skimage.filters.gaussian(image, sigma=3)
10    # local thresholding
11    image_local_threshold = np.percentile(image, 95)
12    image_local = image > image_local_threshold
13    # remove tiny features
14    image_local_open = skimage.morphology.remove_small_objects(
15        image_local, min_size=size_min)
16    # label image
17    image_labeled = label(image_local_open)
18    # analyze regions
19    our_regions = regionprops_table(
20        image_labeled, intensity_image=image,
21        include_size=True, include_intensity=True)
22    # create a new mask with the same dimensions as the input
23    newImage = np.zeros_like(image)
24    # fill in using regionprops table
25    for x in range(len(our_regions)):
26        if our_regions[x]['area'] > size_min and our_regions[x]['area'] < size_max:
27            newImage[our_regions[x].label] = our_regions[x].intensity
28
29    return newImage
```

gw18g940 — bash — 72x21

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



The desktop environment includes:

- A terminal window titled "gw18g940 — bash — 72x21" showing the command "python course_functions.py" was run.
- A file browser window showing a directory structure with files like "plot.png", "Results.csv", and several "segmentation" files.
- A scatter plot titled "Figure 1" showing "Mean intensity [AU]" on the y-axis versus "Area [px]" on the x-axis. A red line represents a linear regression fit.

“Classic” software vs. notebooks

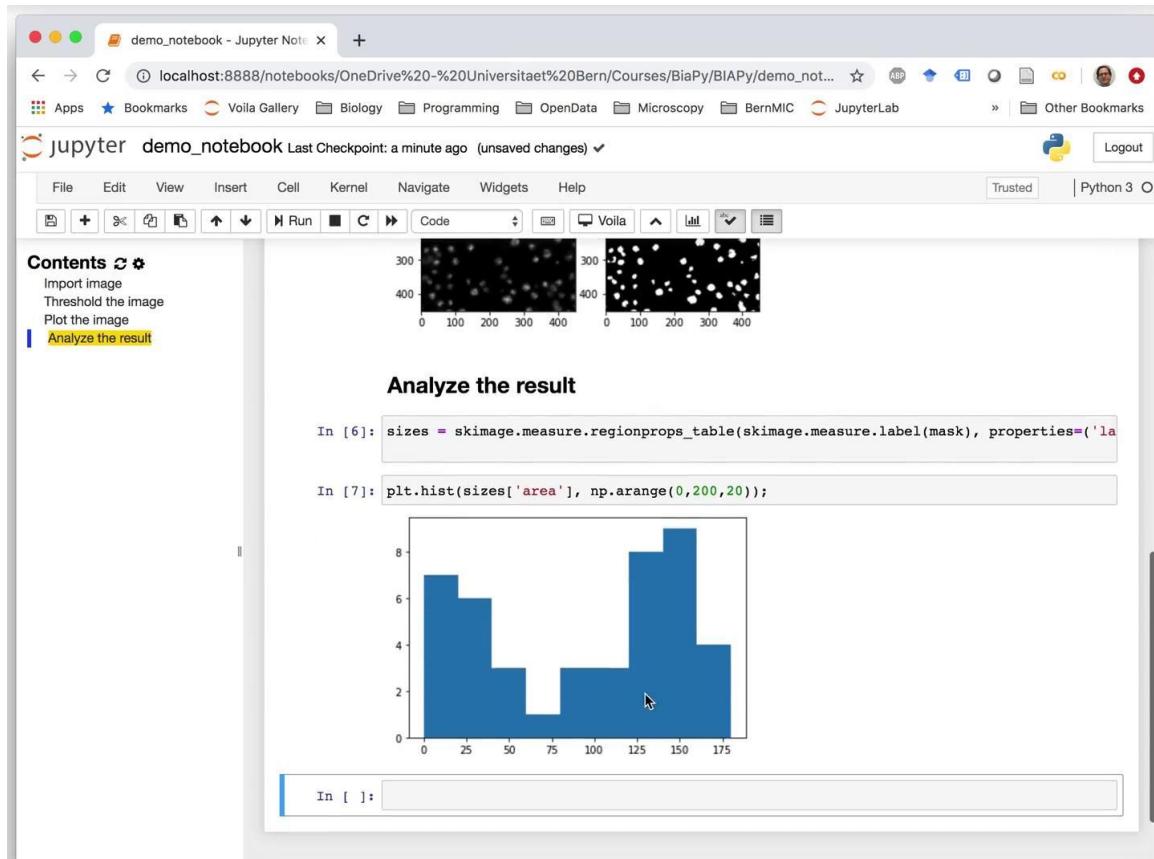
Code divided in parts: easy to investigate

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

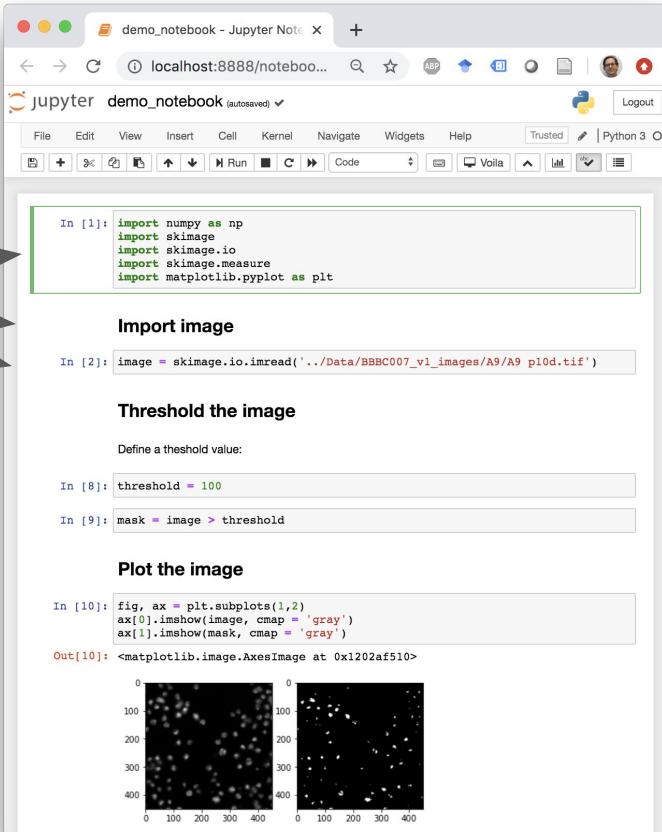
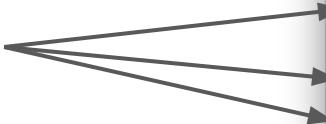


What is a Jupyter notebook?

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

Rendered by Jupyter in the **browser**

Split into sections called **cells**



The screenshot shows a Jupyter Notebook interface running in a web browser. The title bar indicates it's a demo notebook. The interface is organized into cells:

- In [1]:** Python code to import numpy, skimage, skimage.io, skimage.measure, and matplotlib.pyplot.
- Import image**: A section where the user imports an image from a local directory.
- In [2]:** The imported image variable.
- Threshold the image**: A section where the user defines a threshold value.
- In [8]:** The defined threshold value (100).
- In [9]:** The resulting binary mask.
- Plot the image**: A section where the user plots the original image and the thresholded mask side-by-side.
- In [10]:** Python code to create a subplot of the image and mask.
- Out[10]:** The resulting plot showing two grayscale images side-by-side, with the left image showing bright spots on a dark background and the right image showing a binary mask.

What is a Jupyter notebook?

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

Rendered by Jupyter in the **browser**

Split into sections called **cells**

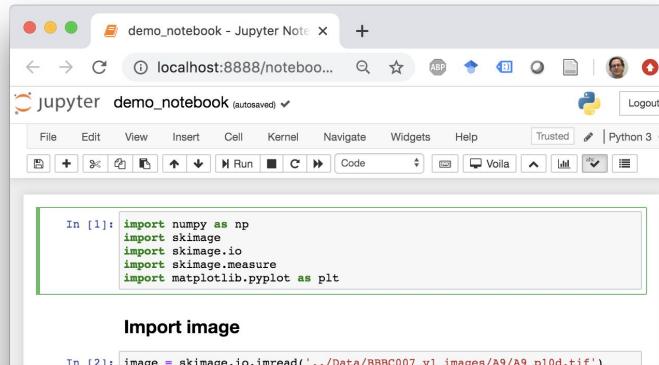
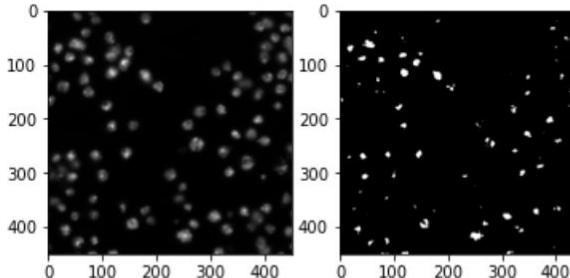
Cells can contain:

- Code
- Formatted text
- Rich output

In [2]: 

```
In [10]: fig, ax = plt.subplots(1,2)
ax[0].imshow(image, cmap = 'gray')
ax[1].imshow(mask, cmap = 'gray')

Out[10]: <matplotlib.image.AxesImage at 0x1202af510>
```



A screenshot of a Jupyter Notebook interface in a web browser. The title bar says "demo_notebook - Jupyter Notebook". The menu bar includes File, Edit, View, Insert, Cell, Kernel, Navigate, Widgets, Help, Trusted, and Python 3. Below the menu is a toolbar with various icons. The main area has two code cells. The first cell, "In [1]", contains Python code to import numpy, skimage, skimage.io, skimage.measure, and matplotlib.pyplot. The second cell, "In [2]", imports skimage.io and reads an image from a specified path. The output of cell 1 is a text message "Import image". The output of cell 2 is the memory address of the AxesImage object.

Notebook cells

Cells can be executed individually

Variables accessible in the whole notebook

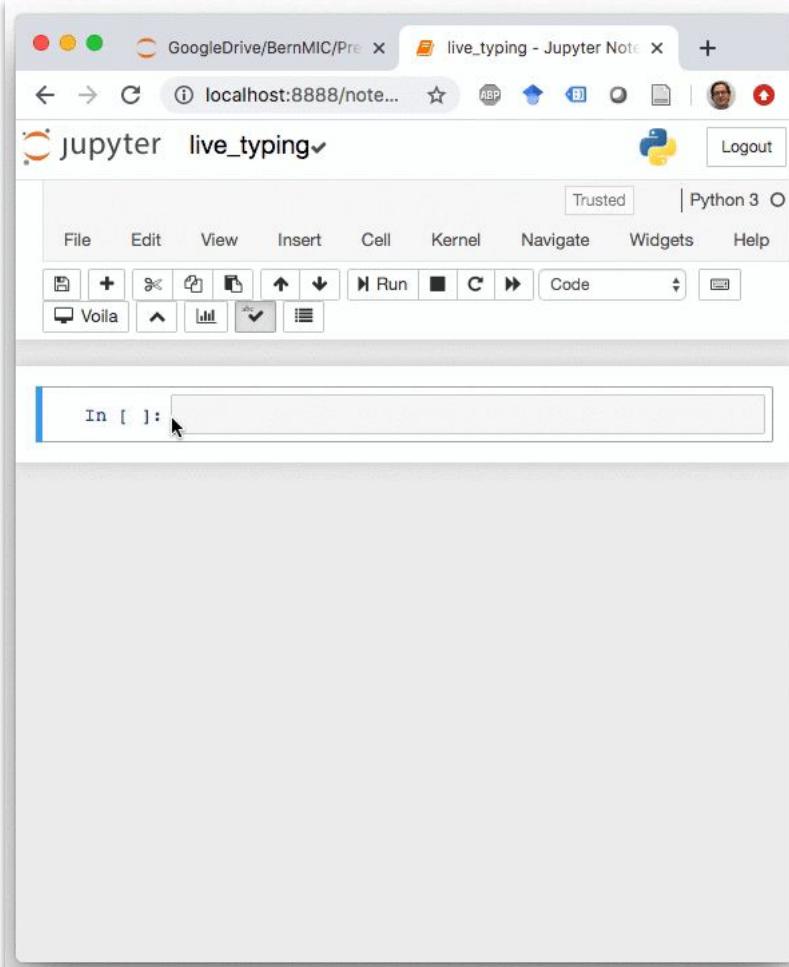
New cells can be added anywhere

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

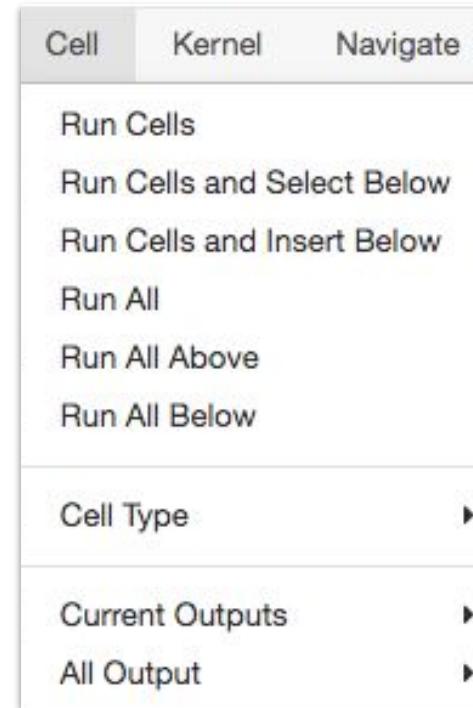
New cells can be added anywhere

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

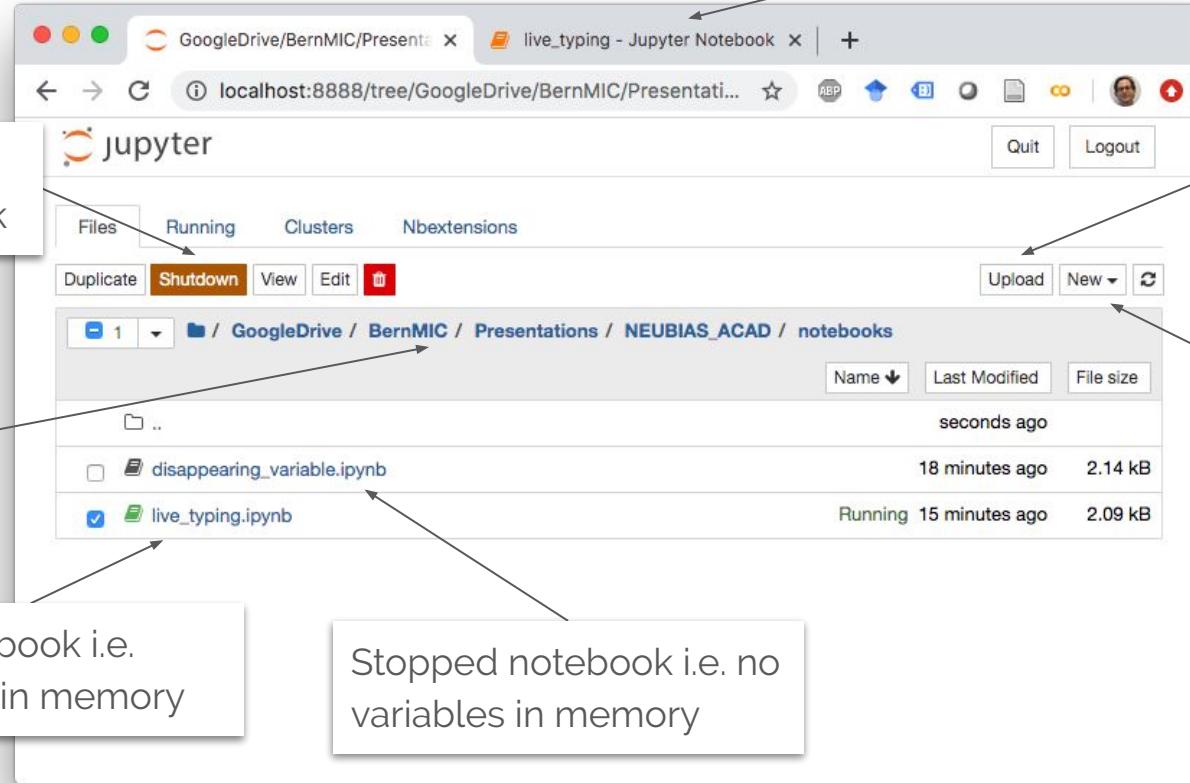
Path to current
folder

Live notebook i.e.
variables in memory

Stopped notebook i.e. no
variables in memory

Upload files
(e.g. data)

Create new
notebook

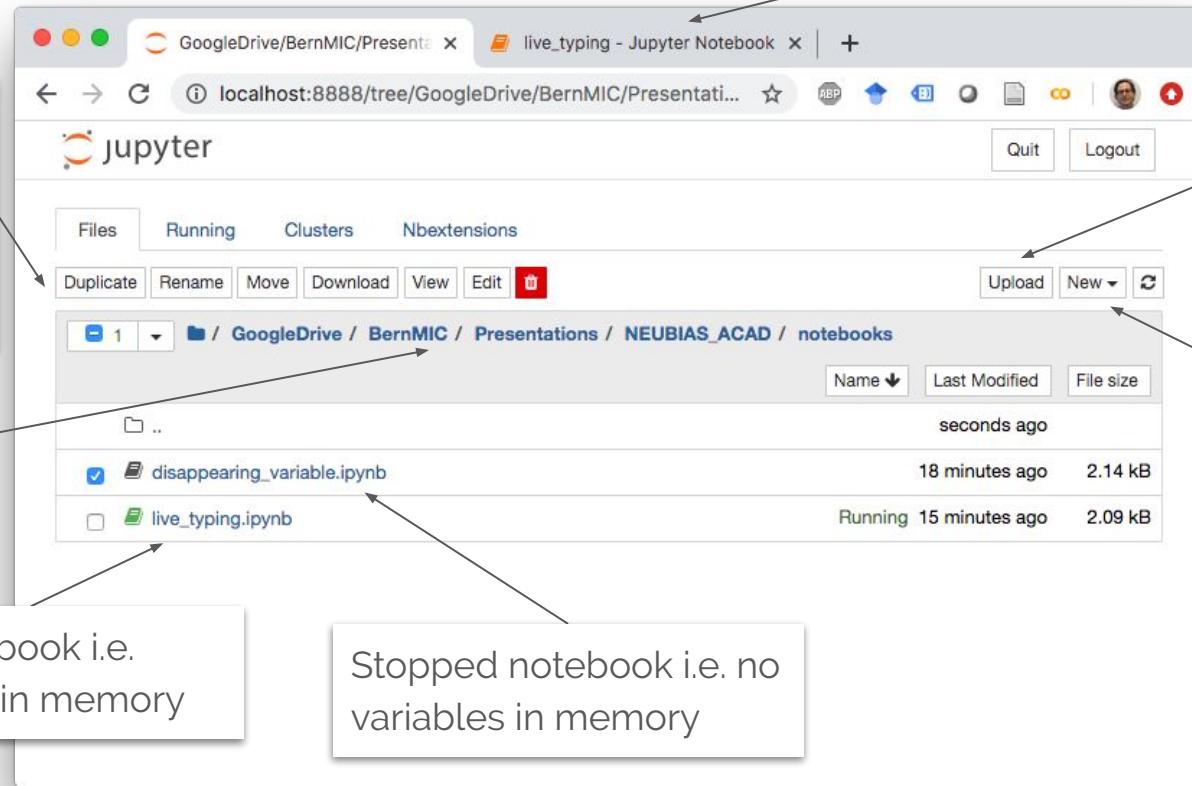


Jupyter browser

Options for stopped notebook:
duplication,
download etc.

Path to current folder

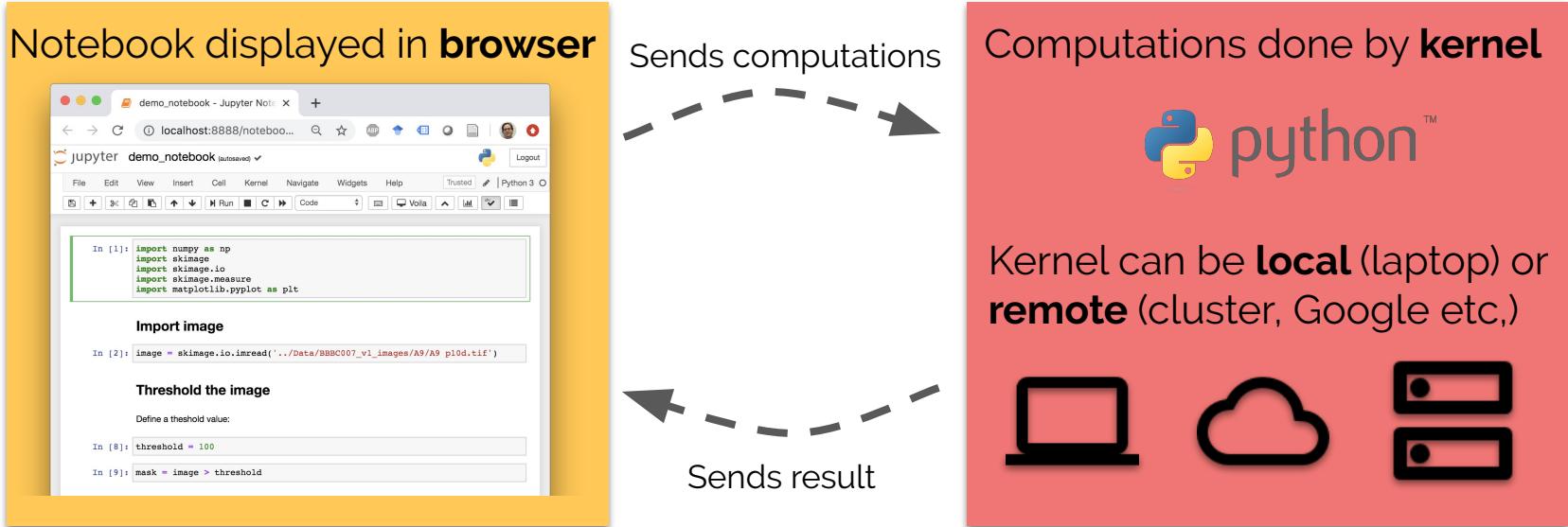
Open notebooks appear as browser tabs



Live notebook i.e.
variables in memory

Stopped notebook i.e. no
variables in memory

How does a notebook calculate?



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

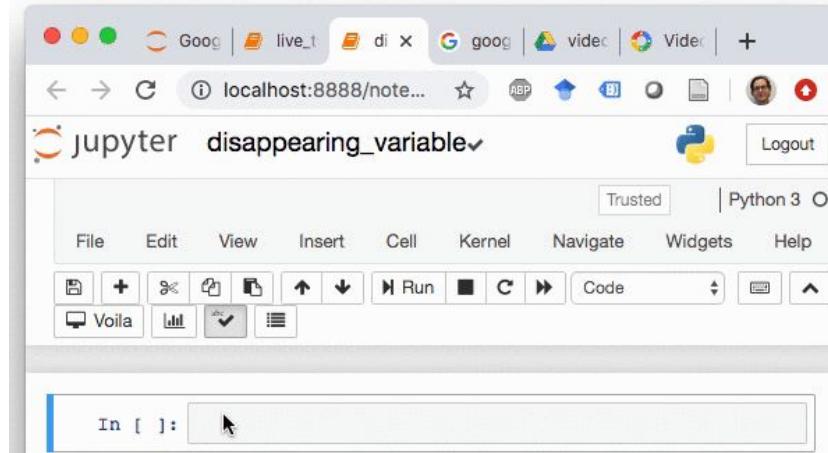
The notebook kernel

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



Sharing and running notebooks



Sharing notebooks

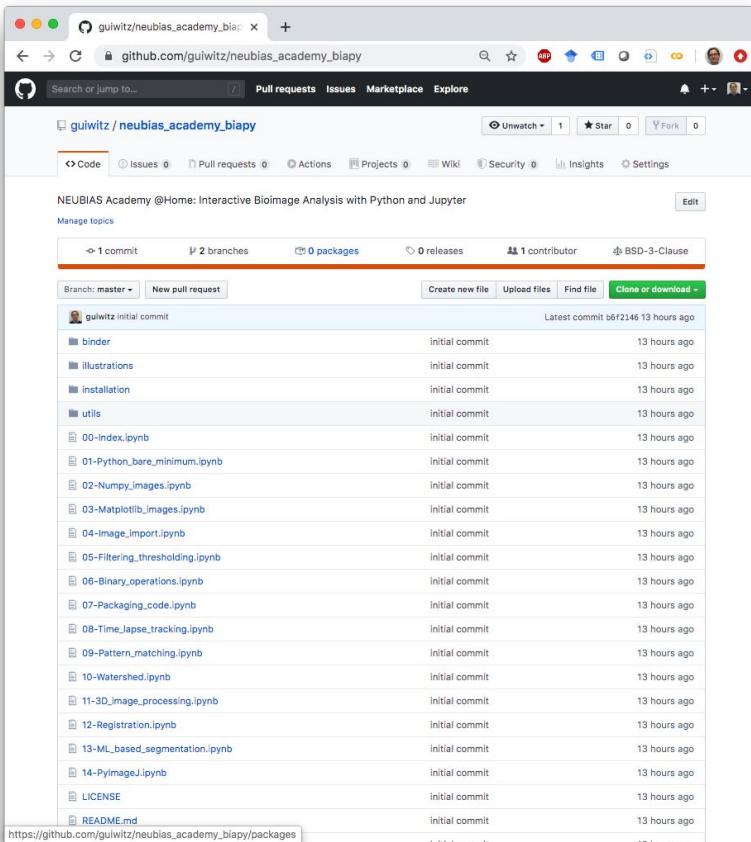
GitHub is a repository for code based on git

Projects are “folders” called a repositories

Public repositories are browsable:

https://github.com/quiwitz/neubias_academy_biapy

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

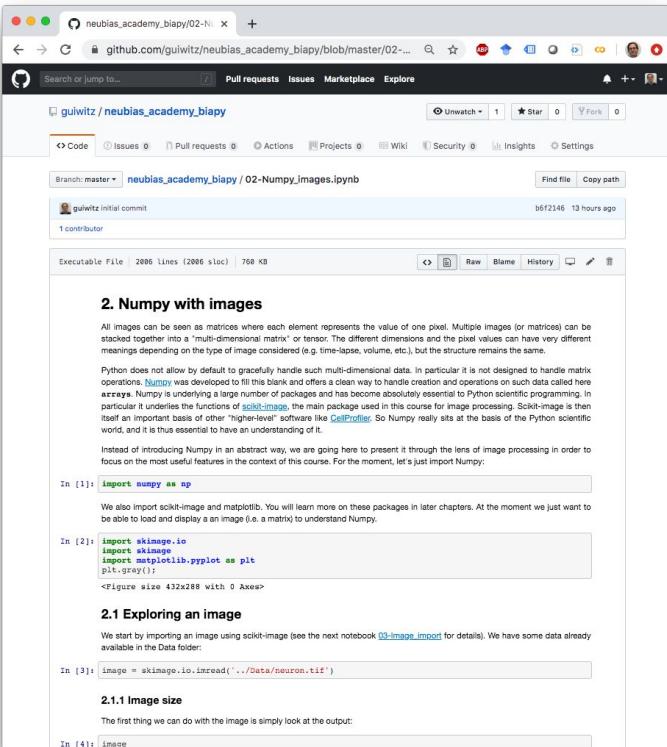


The screenshot shows a GitHub repository page for "neubias_academy_biapy". The repository has 1 commit, 2 branches, 0 packages, 0 releases, and 1 contributor. The master branch is selected. The repository contains several Jupyter notebooks (ipynb files) and other files like LICENSE and README.md. All files were committed 13 hours ago.

File	Type	Commit	Age
guiwitz initial commit		Initial commit	13 hours ago
binder		initial commit	13 hours ago
illustrations		initial commit	13 hours ago
installation		initial commit	13 hours ago
utils		initial commit	13 hours ago
00-Index.ipynb	ipynb	initial commit	13 hours ago
01-Python_bare_minimum.ipynb	ipynb	initial commit	13 hours ago
02-Numpy_images.ipynb	ipynb	initial commit	13 hours ago
03-Matplotlib_images.ipynb	ipynb	initial commit	13 hours ago
04-Image_import.ipynb	ipynb	initial commit	13 hours ago
05-Filtering_thresholding.ipynb	ipynb	initial commit	13 hours ago
06-Binary_operations.ipynb	ipynb	initial commit	13 hours ago
07-Packaging_code.ipynb	ipynb	initial commit	13 hours ago
08-Time_lapse_tracking.ipynb	ipynb	initial commit	13 hours ago
09-Pattern_matching.ipynb	ipynb	initial commit	13 hours ago
10-Watershed.ipynb	ipynb	initial commit	13 hours ago
11-3D_image_processing.ipynb	ipynb	initial commit	13 hours ago
12-Registration.ipynb	ipynb	initial commit	13 hours ago
13-ML_based_segmentation.ipynb	ipynb	initial commit	13 hours ago
14-PyImageJ.ipynb	ipynb	initial commit	13 hours ago
LICENSE		initial commit	13 hours ago
README.md		initial commit	13 hours ago

Sharing notebooks statically

Notebooks are rendered **statically** on GitHub



The screenshot shows a GitHub repository page for 'neubias_academy_biapy / 02-Numpy_images.ipynb'. The page displays the contents of the notebook, including code cells and their outputs. The notebook content includes sections like '2. Numpy with images' and '2.1 Exploring an image', with code snippets such as:

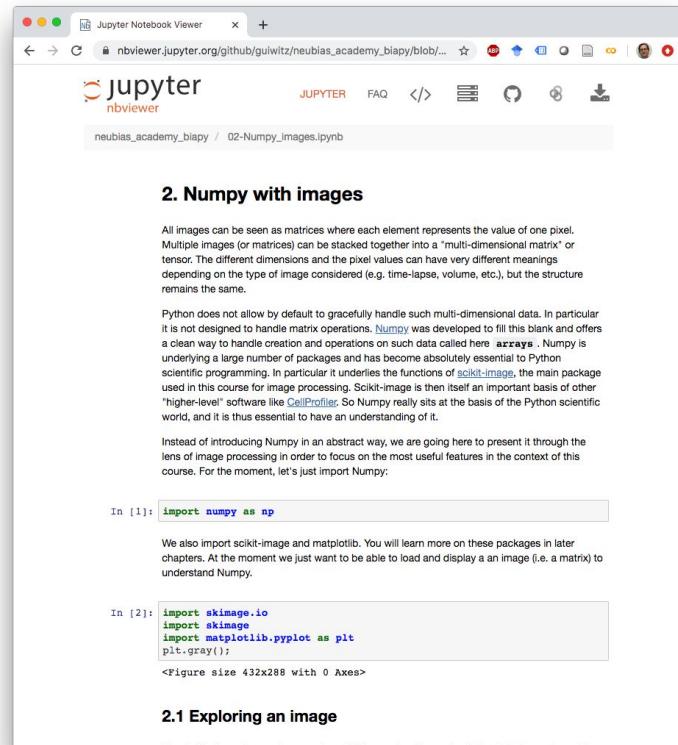
```
In [1]: import numpy as np
In [2]: import skimage.io
import skimage
import matplotlib.pyplot as plt
plt.gray()
<Figure size 432x288 with 0 Axes>
In [3]: image = skimage.io.imread('../Data/neuron.tif')
In [4]: image
```

GitHub rendering of notebook

Sharing notebooks statically

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

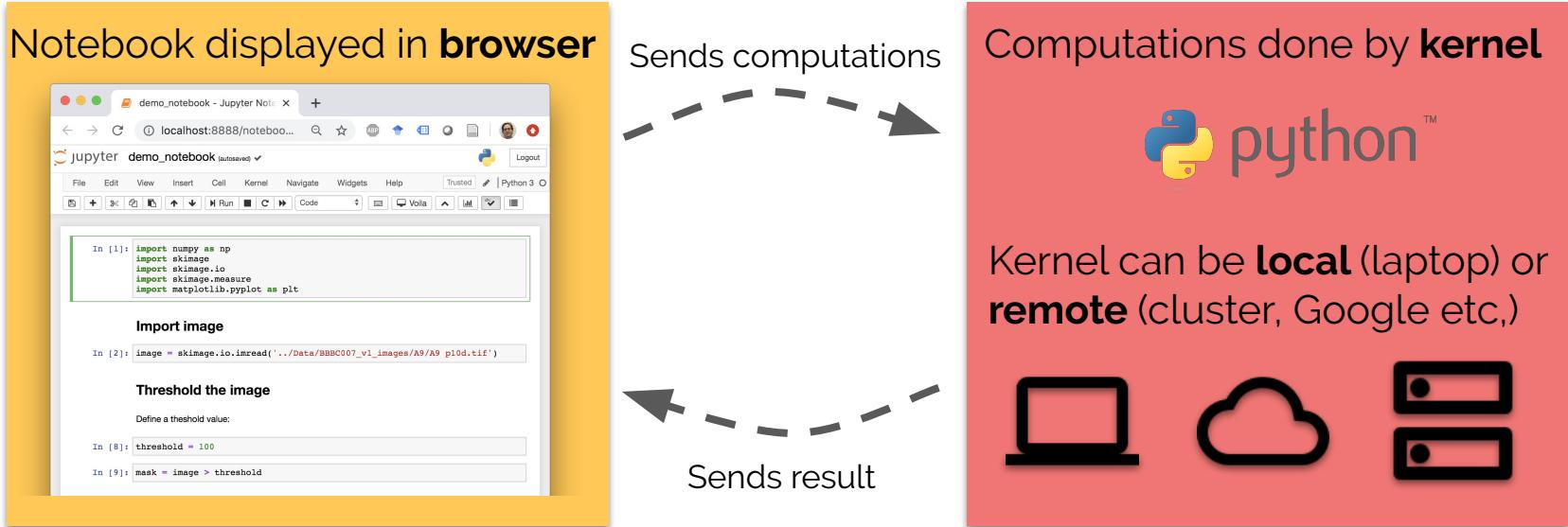
https://nbviewer.jupyter.org/github/quiwitz/neubias_academy_bappy/tree/master/



The screenshot shows a browser window titled "Jupyter Notebook Viewer" displaying a static rendered version of a Jupyter notebook. The page has a header with the "jupyter nbviewer" logo and navigation links for "JUPYTER", "FAQ", and "blob". Below the header, the title "neubias_academy_bappy / 02-Numpy_images.ipynb" is visible. The main content area is titled "2. Numpy with images". It contains text explaining that images can be seen as matrices and that Numpy provides a way to handle them. It also mentions Scikit-image as a main package used for image processing. A code cell in In [1] shows the import statement "import numpy as np". Another code cell in In [2] shows imports for "skimage", "skimage", and "matplotlib.pyplot", followed by a call to "plt.gray()". The rendered output of this cell is described as "<Figure size 432x288 with 0 Axes>". At the bottom, there is a note about importing an image using scikit-image.

Nbviewer rendering

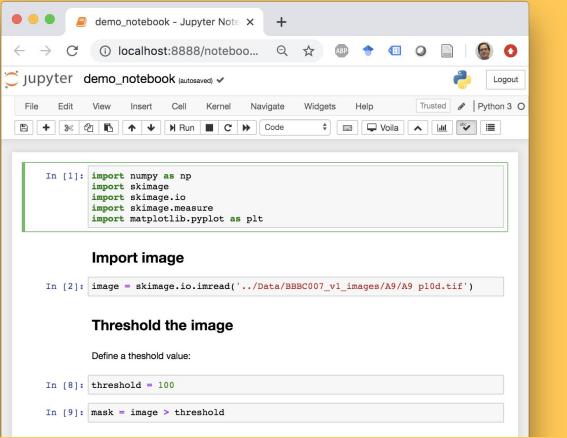
How does a notebook calculate?



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

How does a notebook calculate?

Notebook displayed in **browser**



The screenshot shows a Jupyter Notebook interface within a browser. The code in the cells is:

```
In [1]: import numpy as np
import skimage
import skimage.io
import skimage.measure
import matplotlib.pyplot as plt

Import image

In [2]: image = skimage.io.imread('../Data/BBBC007_v1_images/A9/A9_p10d.tif')

Threshold the image

Define a threshold value:

In [8]: threshold = 100

In [9]: mask = image > threshold
```

Sends computations

Sends result



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

Making notebooks interactive: binder

<https://mybinder.org/>

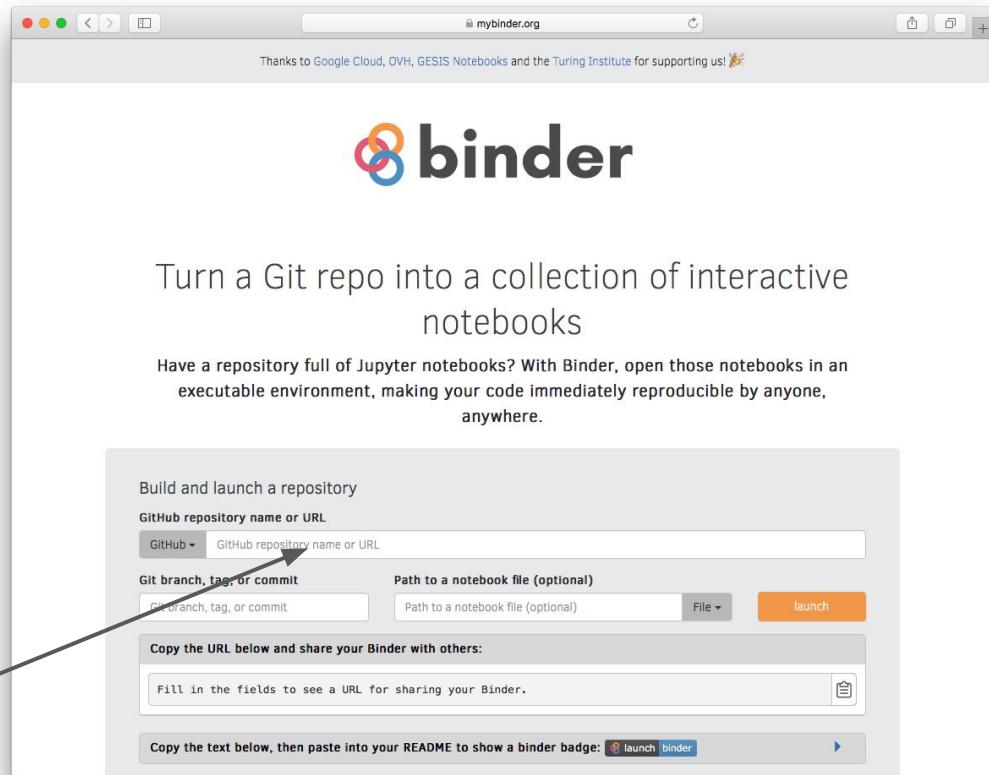
Creates remote Jupyter instance

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 repository

Binder

← → C mybinder.org

JUPYTERBOOKS

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)

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

Waiting for build to start...

hide

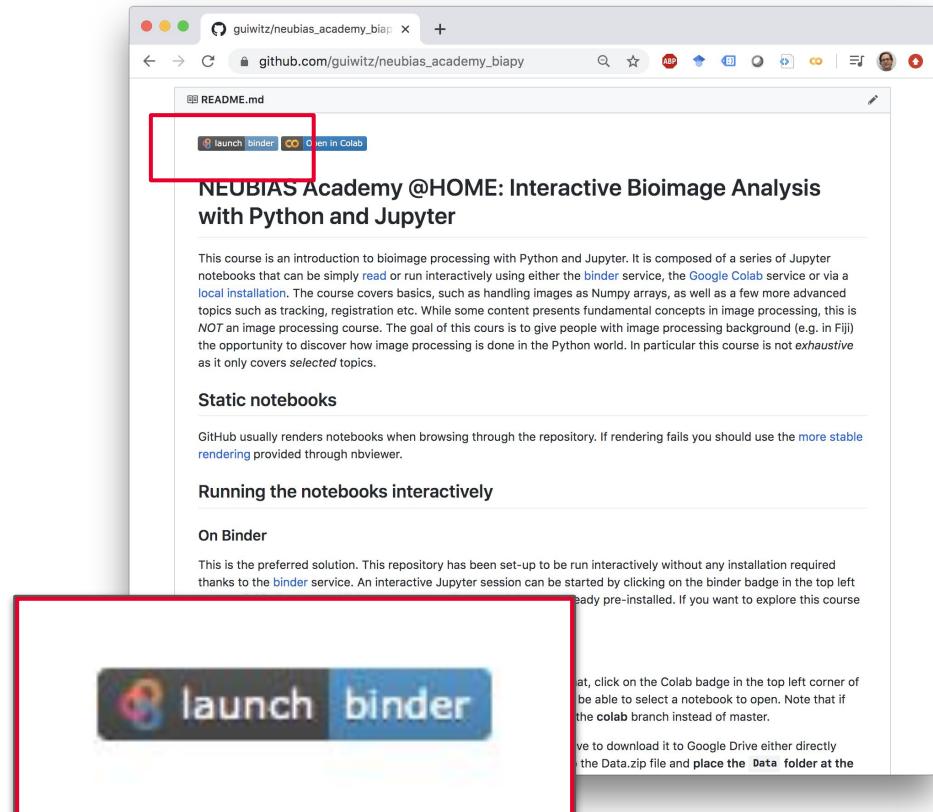
Launching the course on binder

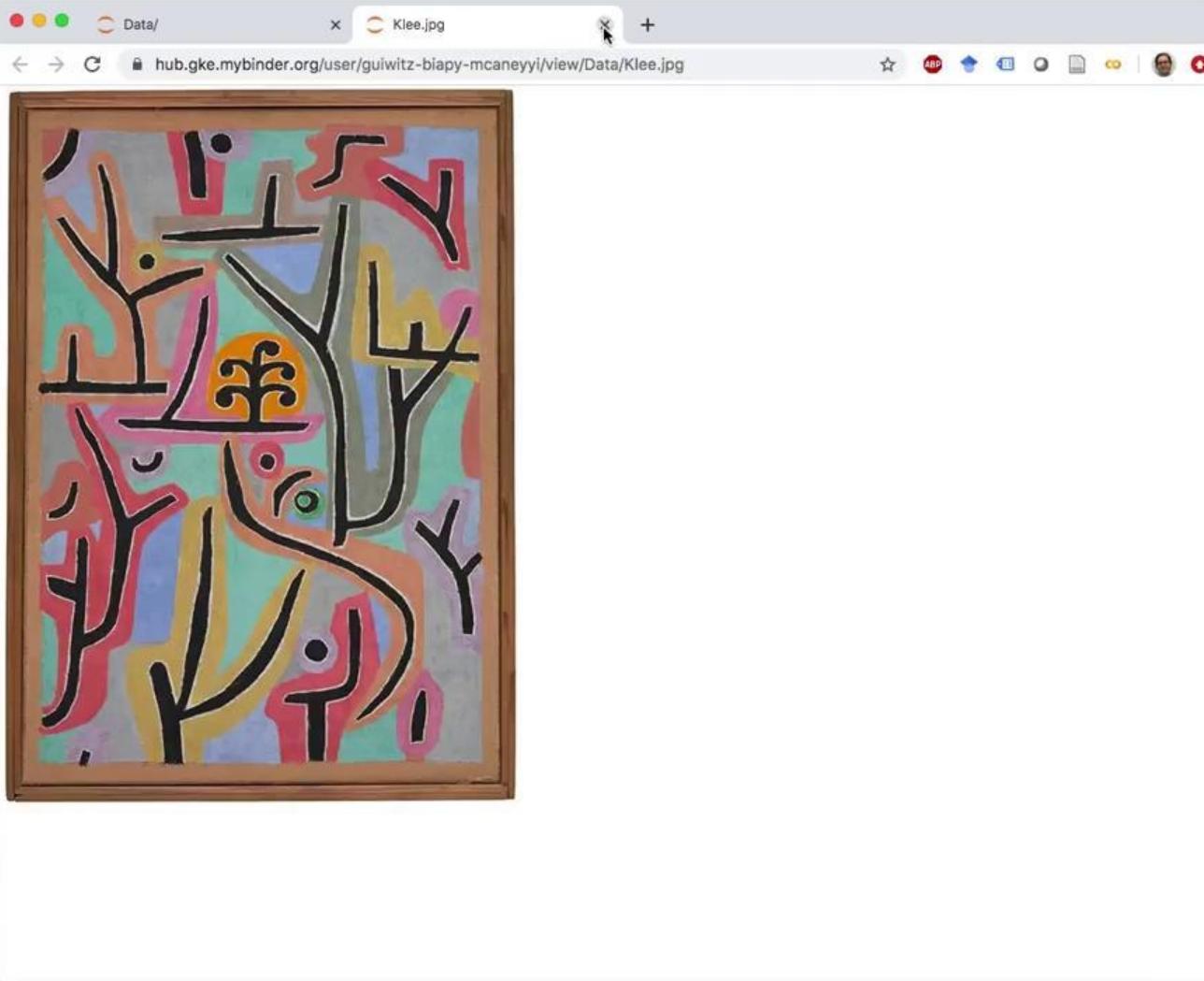
Use the “launch binder” button in the repository (replaces copying url to mybinder.org)

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!





Complete course
data available

All packages
pre-installed

Ready to go !

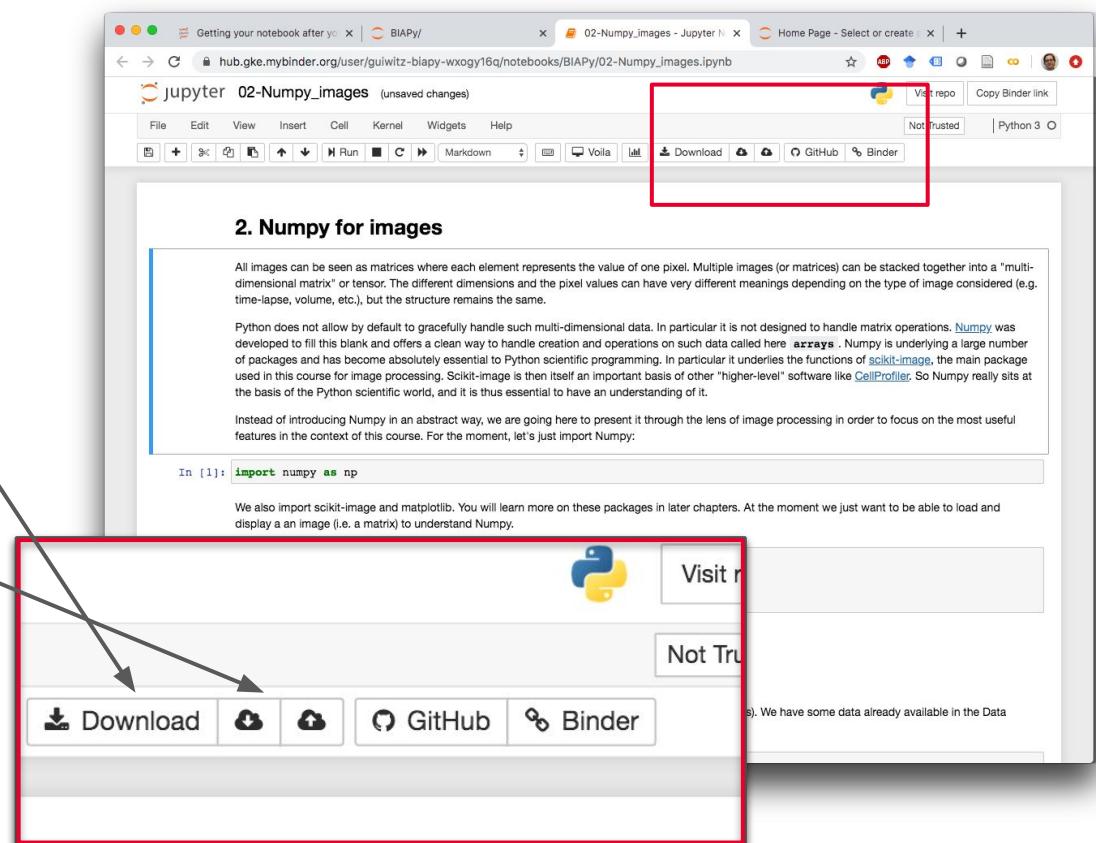
Binder sessions

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

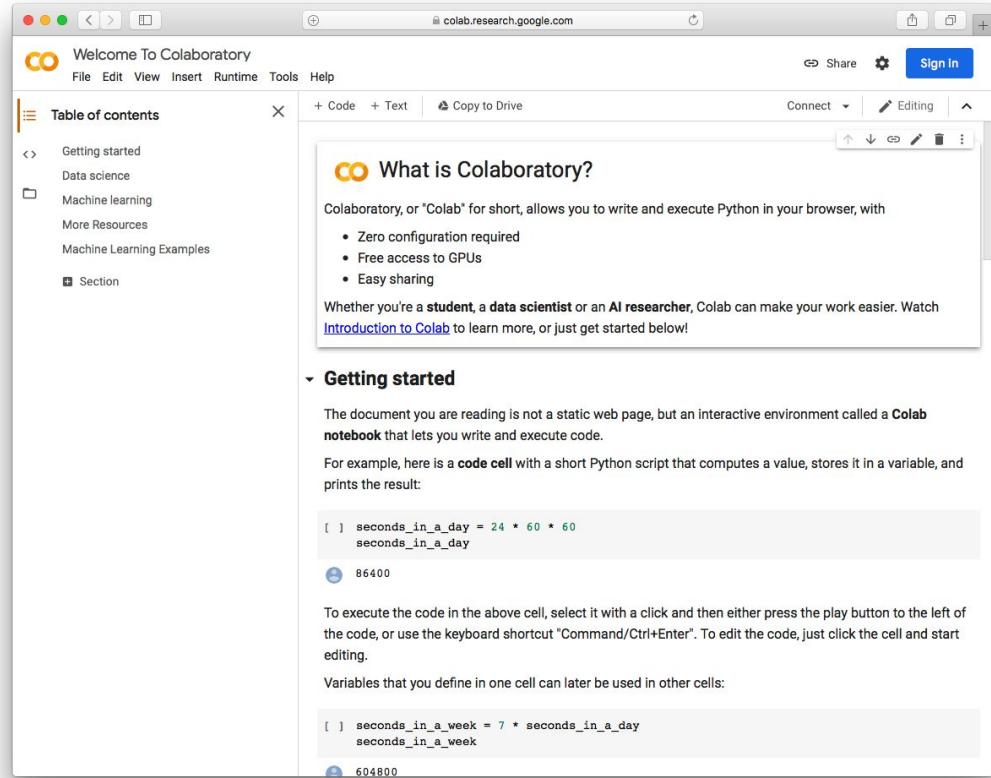
Google's version of Jupyter

Same basic principles, different layout

Kernels run on Google infrastructure for free

Opens any notebook on Github

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



The screenshot shows the Google Colaboratory interface. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. On the right, there are 'Share', 'Sign In' buttons, and a 'Connect' dropdown. Below the bar, there's a 'Table of contents' sidebar with sections like 'Getting started', 'Data science', 'Machine learning', 'More Resources', and 'Machine Learning Examples'. The main content area has a title 'What is Colaboratory?' with a brief description and a bulleted list: 'Zero configuration required', 'Free access to GPUs', and 'Easy sharing'. It also mentions that Colaboratory is suitable for students, data scientists, and AI researchers. A 'Getting started' section follows, explaining that it's an interactive environment for writing and executing code. It shows a code cell with Python code to calculate seconds in a day, which is then executed and the result (86400) is shown. Below this, instructions for executing code are given, along with a note about variable scope and another code cell for calculating seconds in a week.

```
[ ] seconds_in_a_day = 24 * 60 * 60
seconds_in_a_day
86400
```

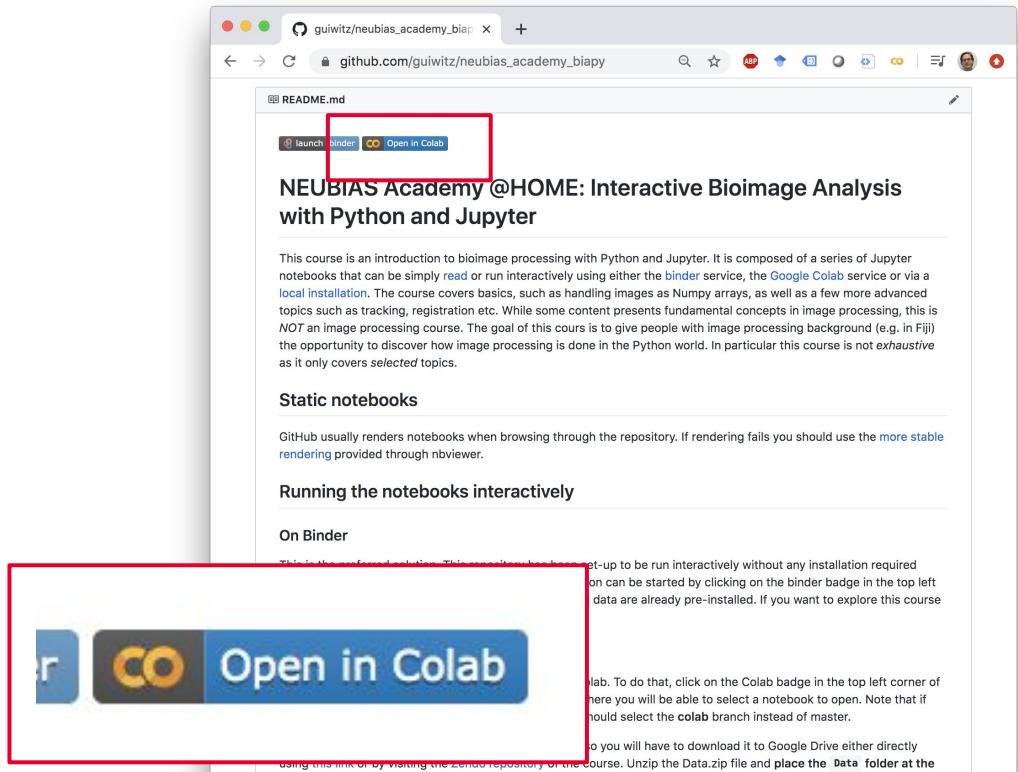
To execute the code in the above cell, select it with a click and then either press the play button to the left of the code, or use the keyboard shortcut "Command/Ctrl+Enter". To edit the code, just click the cell and start editing.

Variables that you define in one cell can later be used in other cells:

```
[ ] seconds_in_a_week = 7 * seconds_in_a_day
seconds_in_a_week
604800
```

Accessing course notebooks

Click on the “Open in Colab” in the repository



Accessing course notebooks

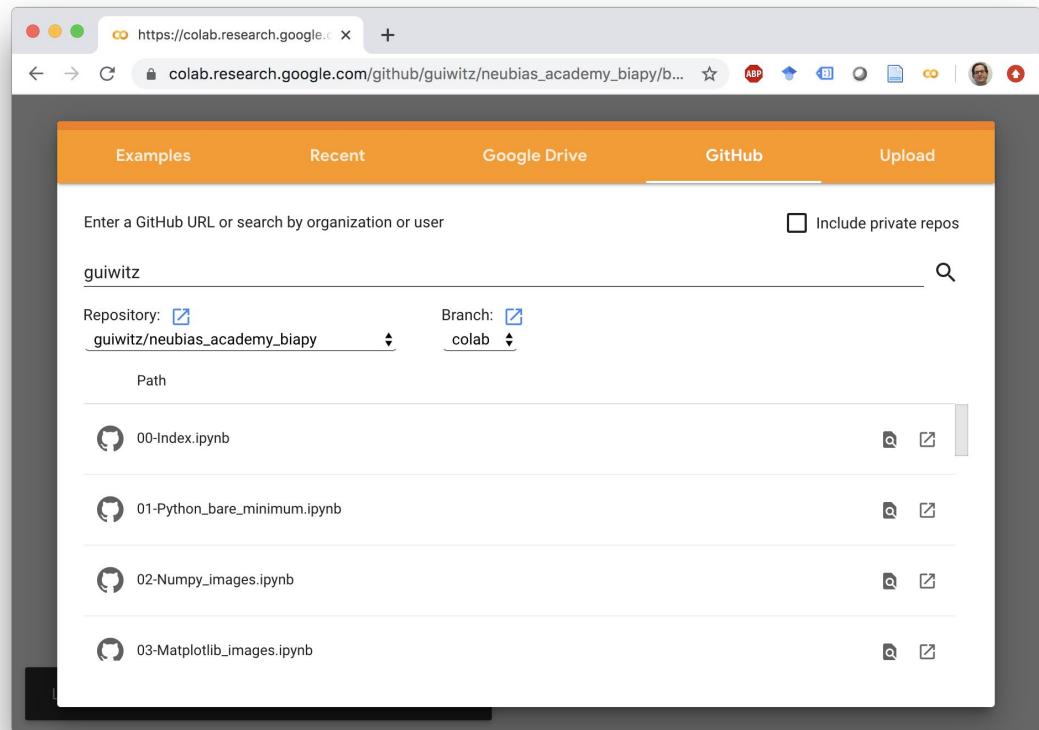
Click on the “Open in Colab” in the repository

Login if not already done

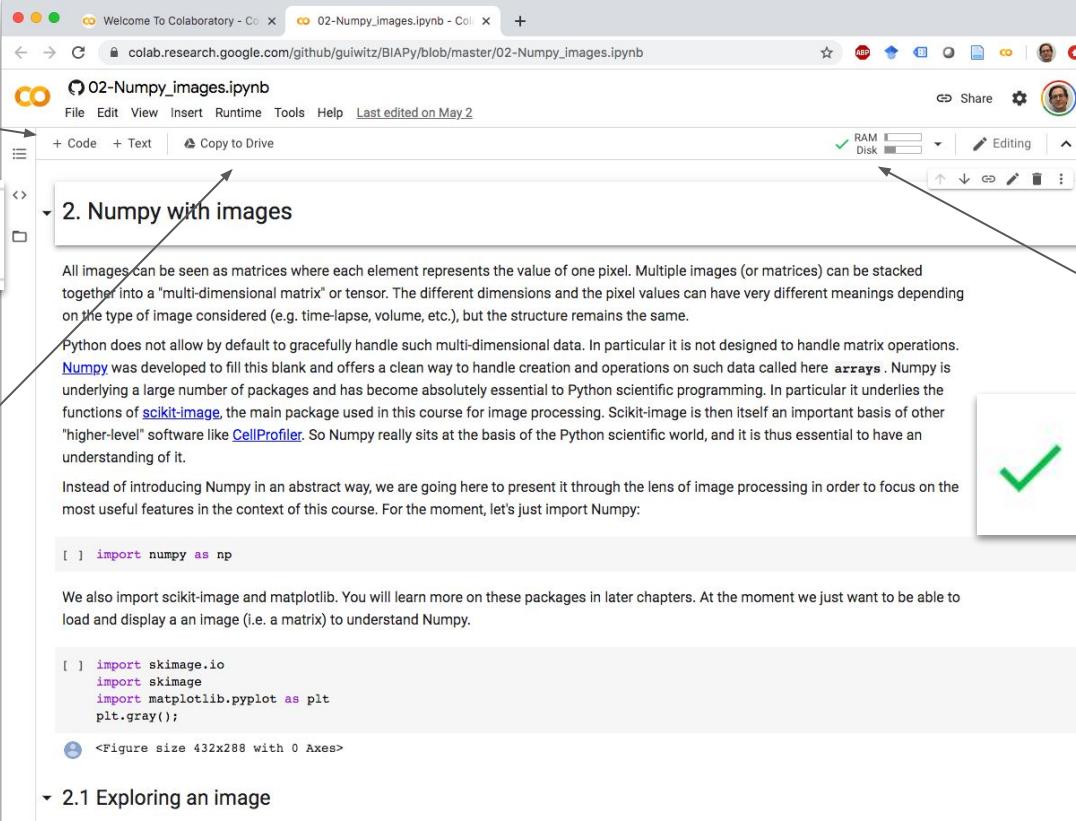
Select a notebook to run

Save the notebook to Google Drive to keep modifications

Open notebook from Google Drive directly next time



A Colab notebook



The screenshot shows a Google Colab notebook titled "02-Numpy_images.ipynb". The notebook has a single section titled "2. Numpy with images". The content of the section discusses how images can be represented as matrices and how Numpy provides a clean way to handle such data. It also mentions Scikit-image and matplotlib as supporting packages.

Code Cell:

```
[ ] import numpy as np
```

We also import scikit-image and matplotlib. You will learn more on these packages in later chapters. At the moment we just want to be able to load and display an image (i.e. a matrix) to understand Numpy.

```
[ ] import skimage
import skimage
import matplotlib.pyplot as plt
plt.gray();
```

<Figure size 432x288 with 0 Axes>

Section 2.1 Exploring an image

Add new cell

+ Code + Text

Copy notebook
to your Drive

 Copy to Drive

Monitor used
RAM and disk



Colab sessions

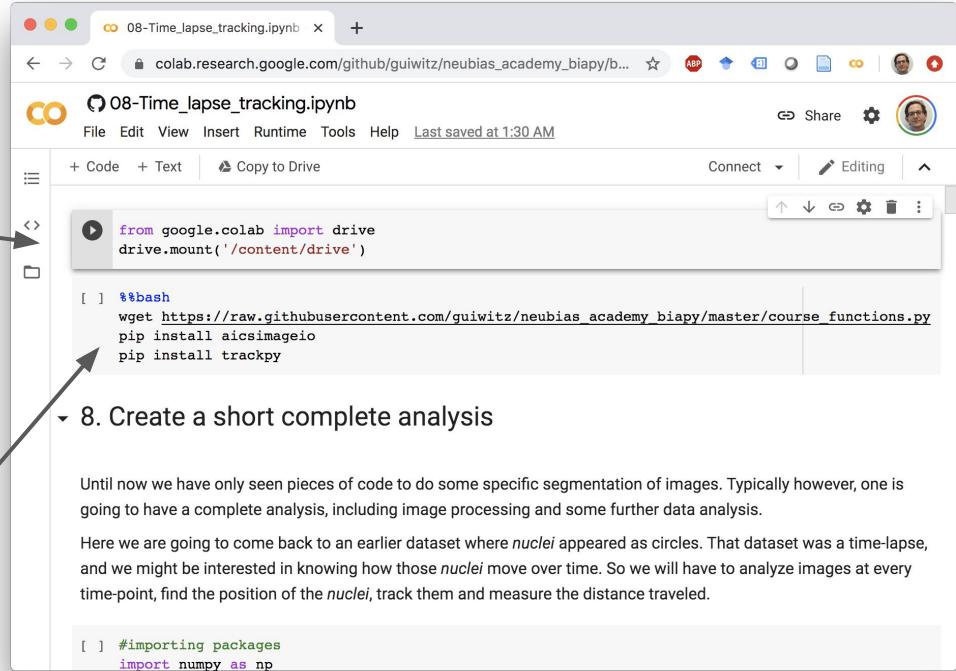
Sessions time-out after max 12h

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



08-Time_lapse_tracking.ipynb

File Edit View Insert Runtime Tools Help Last saved at 1:30 AM

+ Code + Text Copy to Drive Connect Editing

```
from google.colab import drive
drive.mount('/content/drive')

%%bash
wget https://raw.githubusercontent.com/guiwitz/neubias_academy_biapy/master/course_functions.py
pip install aicsimageio
pip install trackpy
```

8. Create a short complete analysis

Until now we have only seen pieces of code to do some specific segmentation of images. Typically however, one is going to have a complete analysis, including image processing and some further data analysis.

Here we are going to come back to an earlier dataset where *nuclei* appeared as circles. That dataset was a time-lapse, and we might be interested in knowing how those *nuclei* move over time. So we will have to analyze images at every time-point, find the position of the *nuclei*, track them and measure the distance traveled.

```
#importing packages
import numpy as np
```

Installing packages



CONDA

Installing packages: Pip

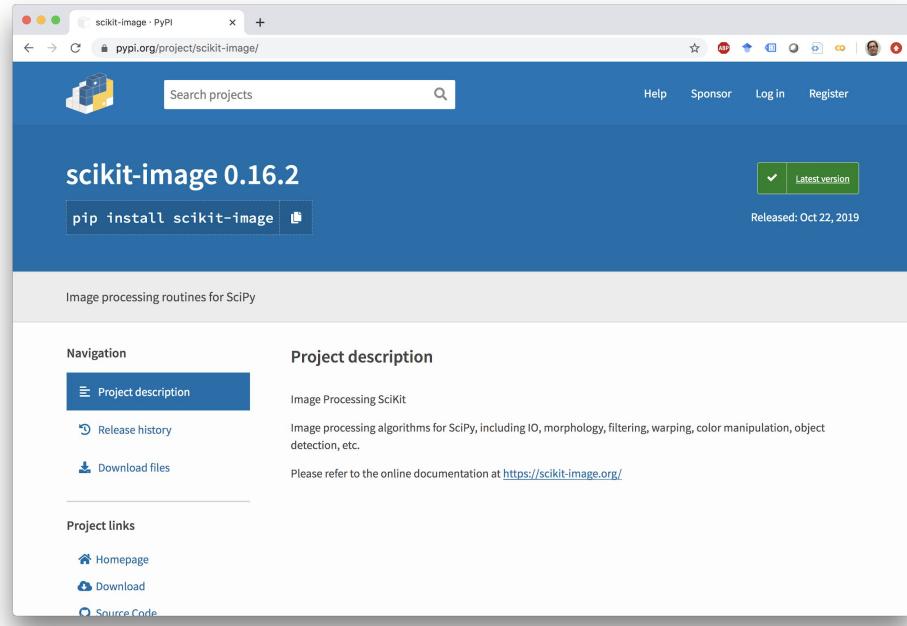


Pip installs packages Pip

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.



The screenshot shows the PyPI (Python Package Index) website. The URL in the address bar is pypi.org/project/scikit-image/. The main header features the Python logo and a search bar labeled "Search projects". Below the header, the project title "scikit-image 0.16.2" is displayed, along with a green button containing a checkmark and the text "Latest version". To the right of the button, the release date "Released: Oct 22, 2019" is shown. The central content area is titled "Image processing routines for SciPy". On the left, there's a sidebar with "Navigation" links: "Project description" (which is currently selected and highlighted in blue), "Release history", and "Download files". On the right, the "Project description" section includes a brief description: "Image Processing SciKit", a list of features: "Image processing algorithms for SciPy, including IO, morphology, filtering, warping, color manipulation, object detection, etc.", and a note: "Please refer to the online documentation at <https://scikit-image.org/>".

Example: install scikit-image from Jupyter

Use ! sign in notebook for command line actions:

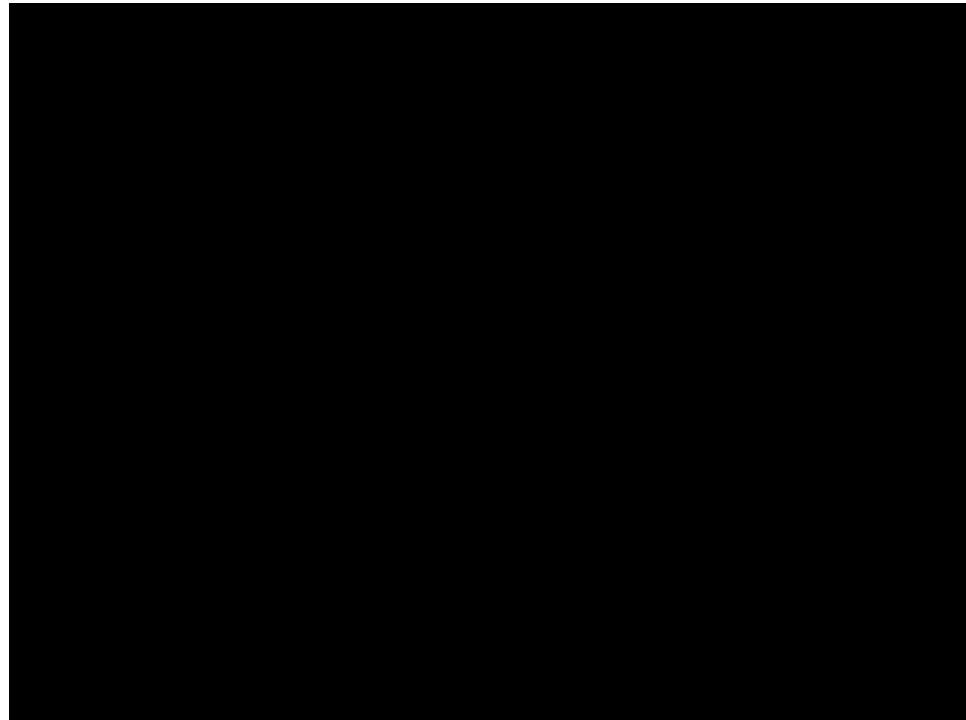
```
!pip install scikit-image
```

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.)

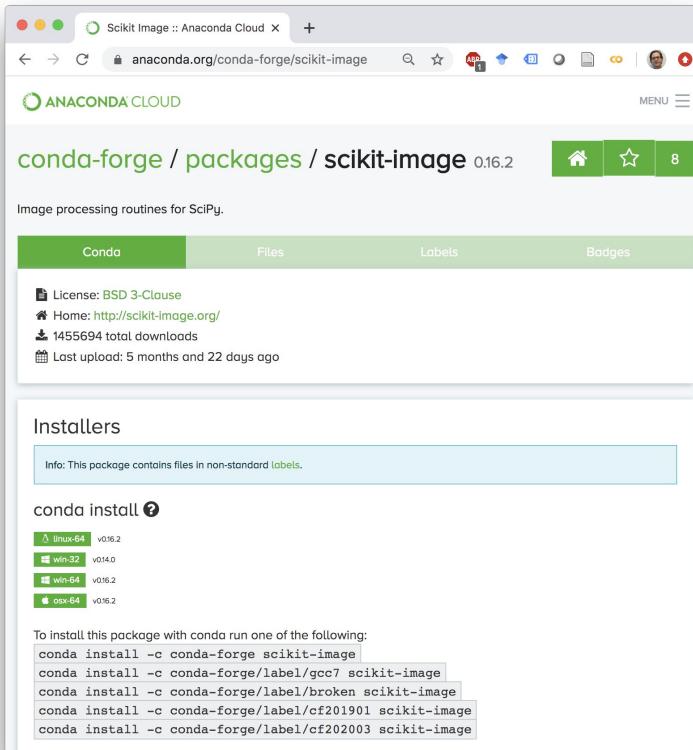
Create isolated environments for each project



osx-64 v0.16.2

To install this package with conda run one of the following:

```
conda install -c conda-forge scikit-image
conda install -c conda-forge/label/gcc7 sciki
```



Scikit Image :: Anaconda Cloud

anaconda.org/conda-forge/scikit-image

ANACONDA CLOUD

conda-forge / packages / scikit-image 0.16.2

Image processing routines for SciPy.

Conda Files Labels Badges

License: BSD 3-Clause
Home: <http://scikit-image.org/>
1455694 total downloads
Last upload: 5 months and 22 days ago

Installers

Info: This package contains files in non-standard labels.

conda install

linux-64	v0.16.2
win-32	v0.14.0
win-64	v0.16.2
osx-64	v0.16.2

To install this package with conda run one of the following:
conda install -c conda-forge scikit-image
conda install -c conda-forge/label/gcc7 scikit-image
conda install -c conda-forge/label/broken scikit-image
conda install -c conda-forge/label/cf201901 scikit-image
conda install -c conda-forge/label/cf202003 scikit-image

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

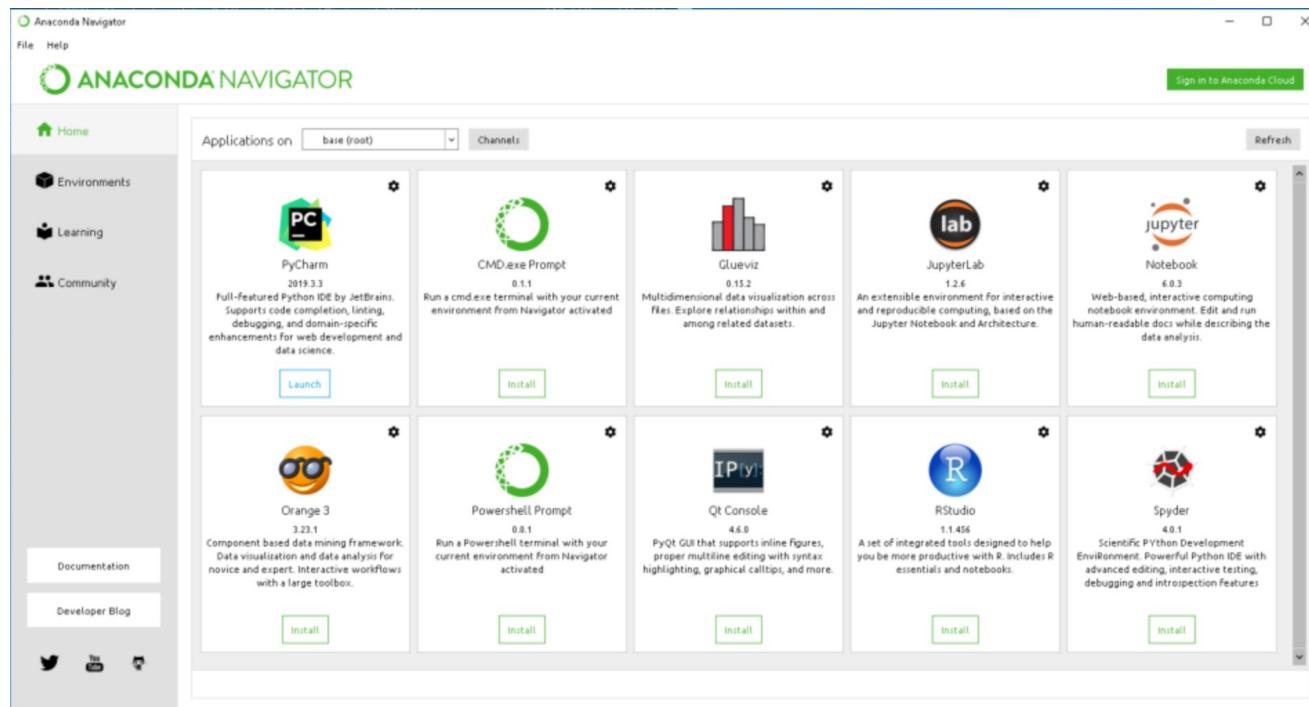
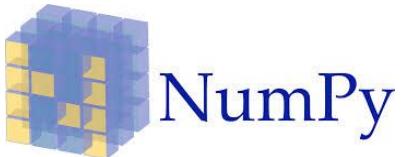


Image processing



NumPy

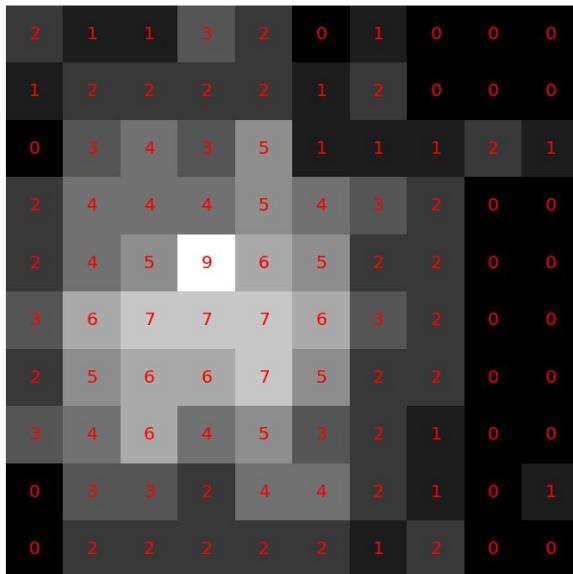
matplotlib



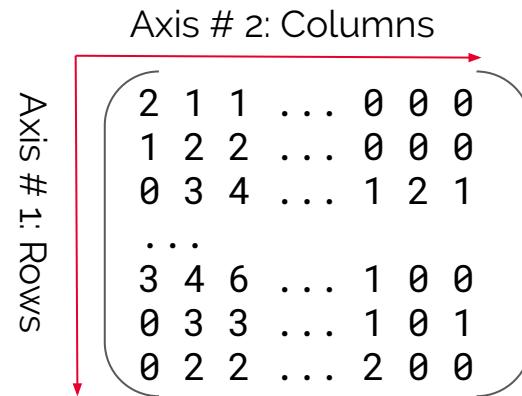
scikit-image
image processing in python

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

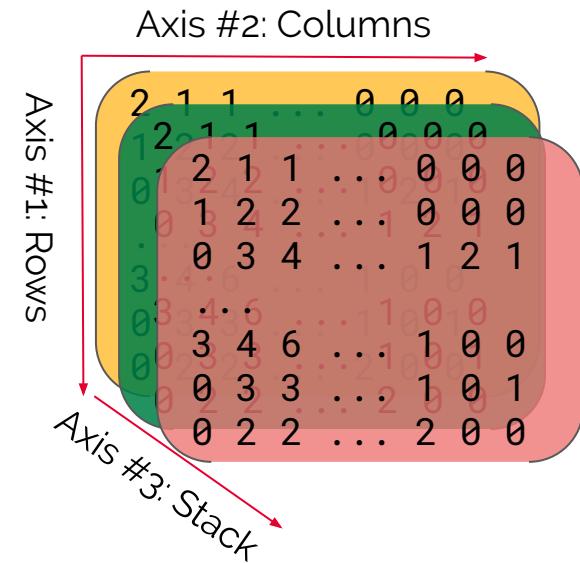
The array objects: Numpy



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:

$$\text{my_array} = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 2 \\ 0 & 3 & 3 \end{pmatrix}$$

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

$$\text{my_array}[0, 2] \rightarrow 1$$

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

$$\text{my_array}[1, :] \rightarrow \begin{pmatrix} 1 & 2 & 2 \end{pmatrix}$$

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

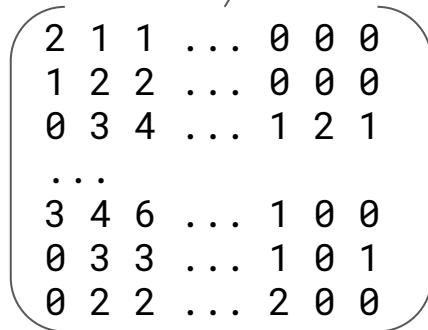
¹Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu and the scikit-image contributors. scikit-image: Image processing in Python. PeerJ 2:e453 (2014)

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.tif')
```



2	1	1	...	0	0	0
1	2	2	...	0	0	0
0	3	4	...	1	2	1
...						
3	4	6	...	1	0	0
0	3	3	...	1	0	1
0	2	2	...	2	0	0

Numpy array

¹Stéfan 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)
```

skimage.filters.gaussian

$$\begin{pmatrix} \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 2 \\ 0 & 3 & 3 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 1.5 & 1.5 & 1.6 \\ 1.4 & 1.6 & 1.8 \\ 1.3 & 1.7 & 2.0 \end{pmatrix}$$

Numpy array
Numpy array

¹Stéfan 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>

¹Stéfan 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

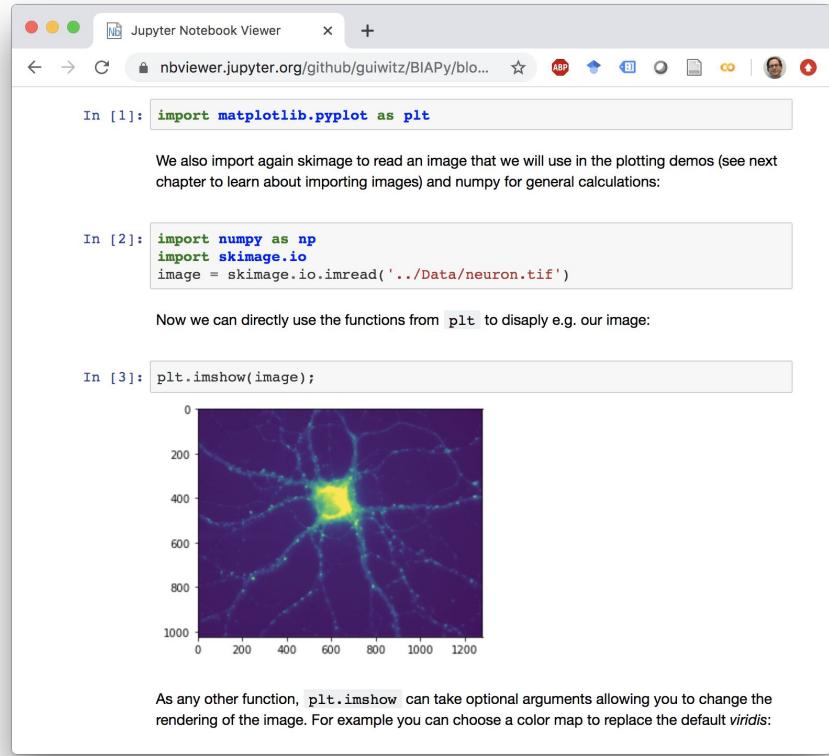
Works with Numpy arrays

Allows for extensive plot customization

Used in minimal way in this course

Easy solution for plotting images:

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



The screenshot shows a Jupyter Notebook interface with three code cells and their corresponding outputs.

- In [1]:** `import matplotlib.pyplot as plt`
- In [2]:** `import numpy as np`
`import skimage.io`
`image = skimage.io.imread('../Data/neuron.tif')`
- In [3]:** `plt.imshow(image);`

The output of In [3] is a grayscale image of a neuron cell body with processes extending outwards, centered against a dark background. The axes range from 0 to 1000.

We also import again skimage to read an image that we will use in the plotting demos (see next chapter to learn about importing images) and numpy for general calculations:

Now we can directly use the functions from `plt` to display e.g. our image:

As any other function, `plt.imshow` can take optional arguments allowing you to change the rendering of the image. For example you can choose a color map to replace the default `viridis`:

https://nbviewer.jupyter.org/github/guiwitz/neubias_academy_biaply/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>

PylImageJ: Mixing Fiji and Python code in notebooks: <https://github.com/imagej/pyimagej>

napari

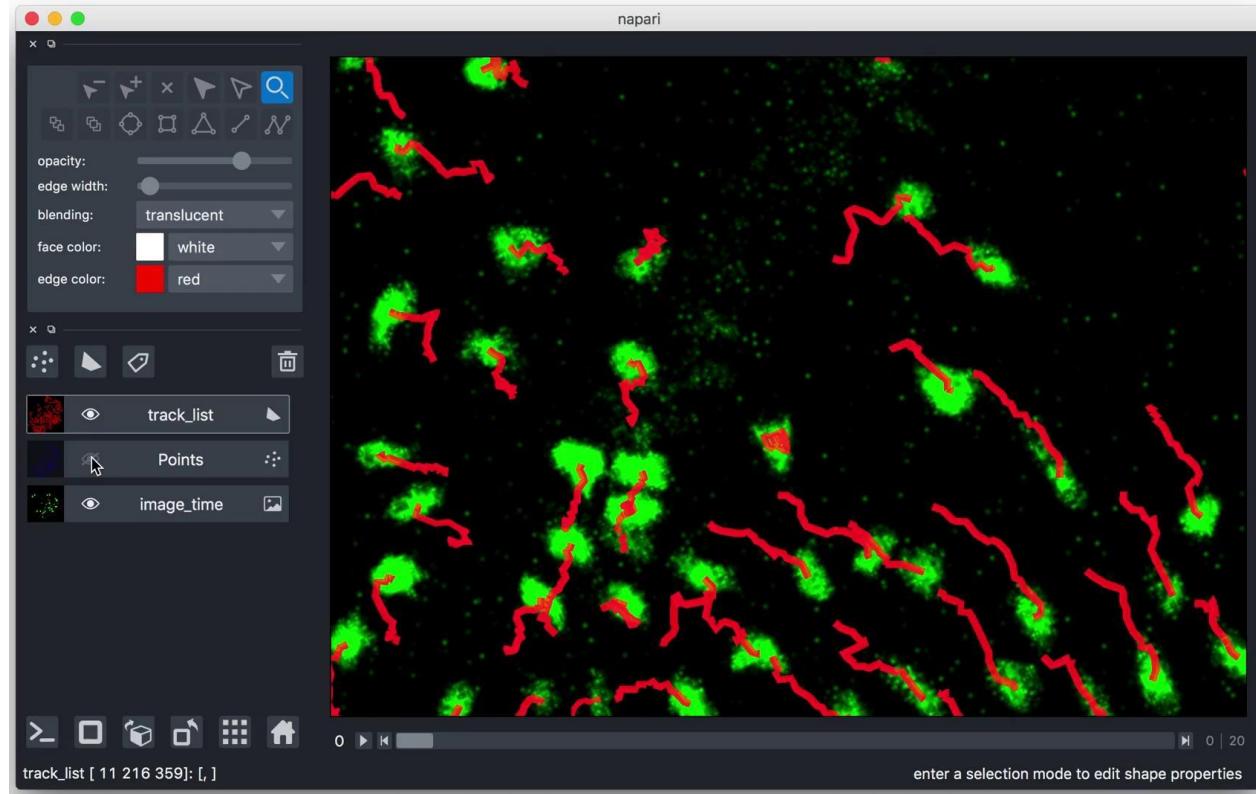
Handles 5D data

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_bappy/blob/master/11-3D_image_processing.ipynb

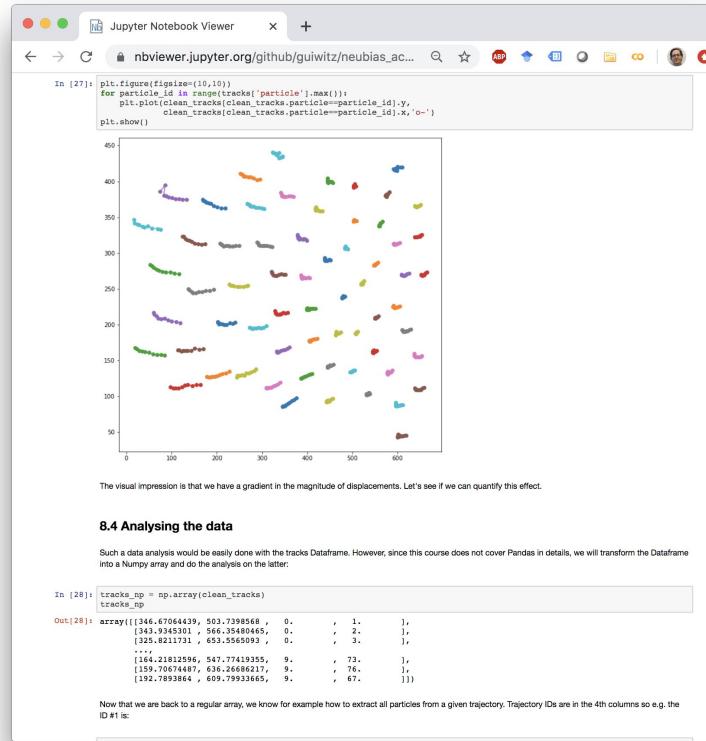
trackpy

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

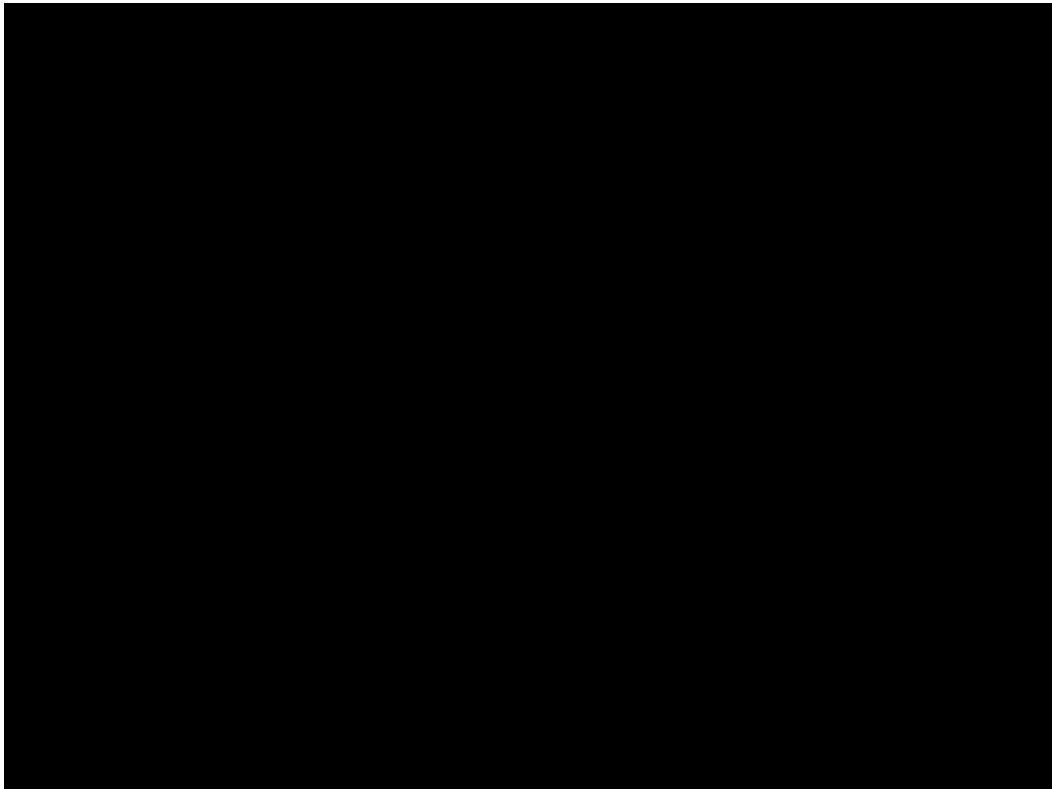
Purely browser based rendering

Volumetric rendering

Other objects: scatter plots,
meshes etc.

Possible to add interactivity with
ipywidgets

Export interactive figures as html!



ipywidgets

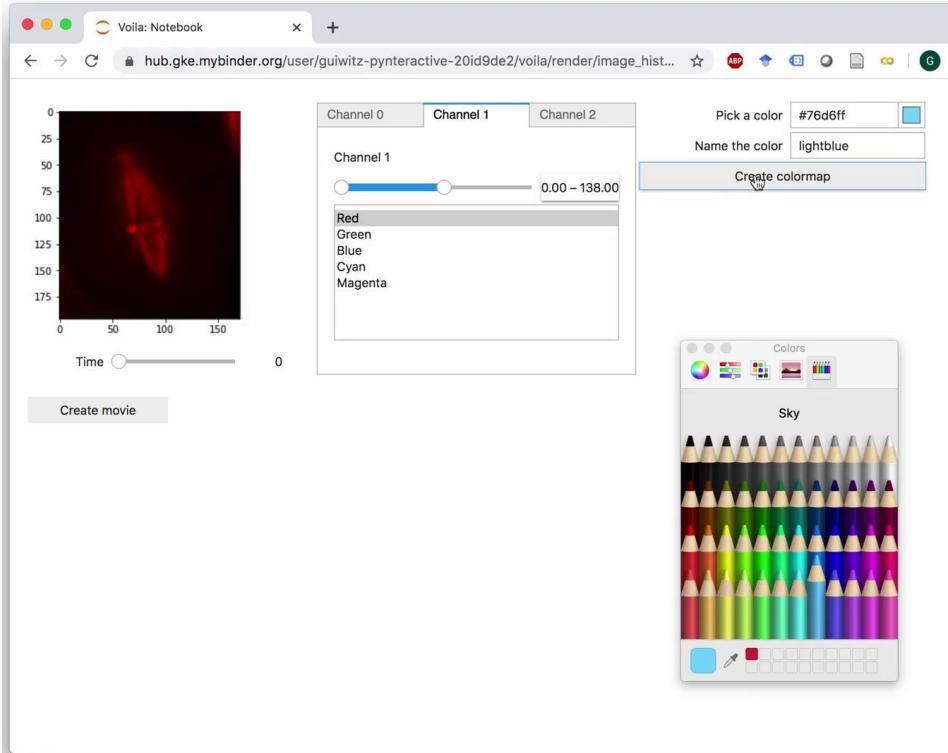
Create interfaces in notebooks:
buttons, sliders, etc.

Easy to connect widgets to actions

Possibility to create web-apps using
voilà:

<https://github.com/voila-dashboards/voila>

Used by other packages (ipyvolume)

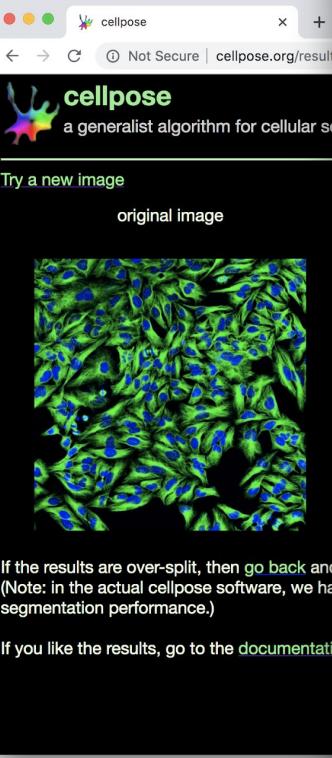


Cellpose

Pre-trained on
large image
database: very
flexible

Predicts labels,
not just masks

Can be used as
Python package



If the results are over-split, then go back and (Note: in the actual cellpose software, we have segmentation performance.)

If you like the results, go to the [documentation](#)

13.2 Cellpose [¶](#)

Cellpose is a new algorithm whose goal is specifically to allow for an easy segmentation of nuclei and cell images. Just like Stardist, it mixes a conventional approach (here a diffusion based cell map) with deep learning to directly generate a label image. It has been trained on a very large dataset making it versatile but can always be further trained. You can test the software by drag and drop on a [website](#), install a local GUI or run it directly as a Python module as here.

First we need to import the necessary packages:

```
In [12]: import mxnet
from cellpose import models
```

Then we need to instantiate a model. We can choose to either segment cells or nuclei by setting the `model_type` option. One can also choose the whether to use CPU or GPU:

```
In [13]: model = models.Cellpose(device=mxnet.cpu(), model_type="nuclei")
```

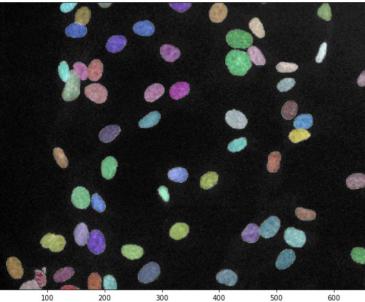
Finally we can do the prediction. The syntax is a bit unusual here. We can in principle pass multiple images to the model as a list of images. Even if you segment single image you need to enclose it within brackets. Second you need to specify the `channels` option. This tells the model which channel to use in case one has multi-channel images. Here we have only a single channel image and therefore specify `[0,]`. Note that we need to pass a pair of channel indices, one for the nuclei channel, one for the cell channel. Since we only have nuclei, we leave the second position empty. Note also that again, this has to be enclosed into brackets to handle multiple image. When using more than one image, we could in principle specify more channel indices e.g. `[[0,], [1,], ...]`.

Finally we can optionally specify an average diameter. Cellpose has the possibility to estimate the size of the objects, but it's much faster to use that option. To speed up things we use the same rescaled image as before and use 20px as our diameter estimate:

```
In [14]: masks, _, _ = model.eval([image[:14, :14]], channels=[[0, ]], diameter=20)
processing 1 images
```

```
In [15]: plt.figure(figsize=(10,10))
plt.imshow(image[:14,:14])
plt.imshow(masks[0], cmap = cmap)
```

```
Out[15]: <matplotlib.image.AxesImage at 0x14c2f4050>
```



use the cell diameter. that improve

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

<http://www.cellpose.org/>

StarDist

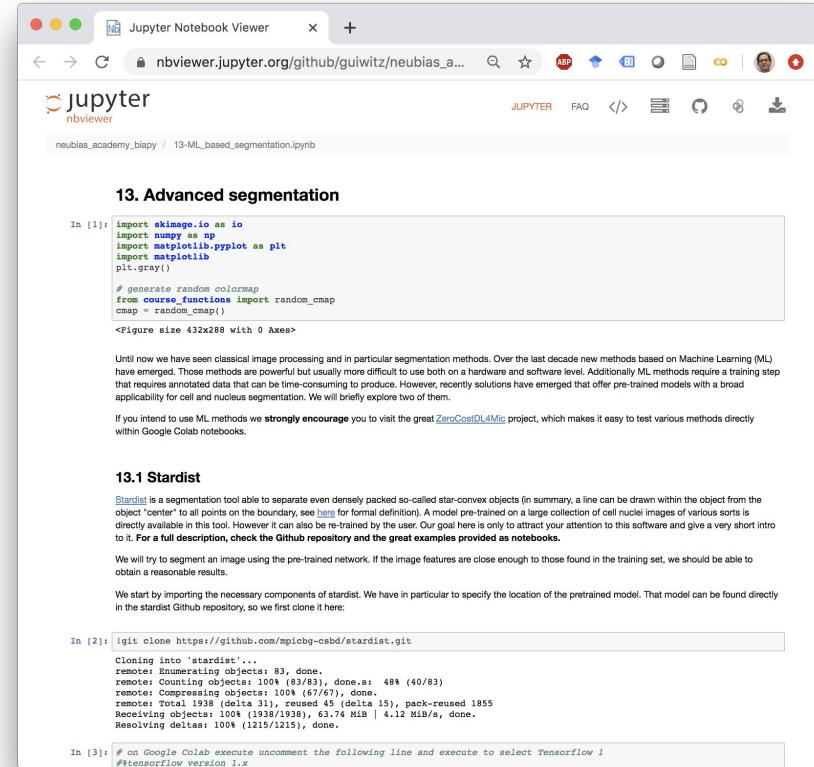
Generalistic object segmentation tool

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



The screenshot shows a Jupyter Notebook Viewer window. The title bar says "Jupyter Notebook Viewer". The URL in the address bar is "nbviewer.jupyter.org/github/guiwitz/neubias_academy_biap...". The page title is "jupyter nbviewer". Below it, it says "neubias_academy_biap... / 13-ML_based_segmentation.ipynb".

13. Advanced segmentation

```
In [1]: import skimage.io as io
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
plt.gray()

# generate random colormap
from course_functions import random_cmap
cmap = random_cmap()
```

<Figure size 432x288 with 0 Axes>

Until now we have seen classical image processing and in particular segmentation methods. Over the last decade new methods based on Machine Learning (ML) have emerged. Those methods are powerful but usually more difficult to use both on a hardware and software level. Additionally ML methods require a training step that requires annotated data that can be time-consuming to produce. However, recently solutions have emerged that offer pre-trained models with a broad applicability for cell and nucleus segmentation. We will briefly explore two of them.

If you intend to use ML methods we **strongly encourage** you to visit the great [ZeroCostDL4Mic](#) project, which makes it easy to test various methods directly within Google Colab notebooks.

13.1 Stardist

Stardist is a segmentation tool able to separate even densely packed so-called star-convex objects (in summary, a line can be drawn within the object from the object "center" to all points on the boundary, see [here](#) for formal definition). A model pre-trained on a large collection of cell nuclei images of various sorts is directly available in this tool. However it can also be re-trained by the user. Our goal here is only to attract your attention to this software and give a very short intro to it. For a full description, check the [Github repository](#) and the [great examples provided as notebooks](#).

We will try to segment an image using the pre-trained network. If the image features are close enough to those found in the training set, we should be able to obtain a reasonable results.

We start by importing the necessary components of stardist. We have in particular to specify the location of the pretrained model. That model can be found directly in the stardist Github repository, so we first clone it here:

```
In [2]: git clone https://github.com/mpicbg-csb/dstardist.git
```

```
Cloning into 'dstardist'...
remote: Enumerating objects: 83, done.
remote: Counting objects: 1000 (83/83), done. 48% (40/83)
remote: Compressing objects: 100% (67/67), done.
remote: Total 1938 (delta 31), reused 45 (delta 15), pack-reused 1855
Receiving objects: 100% (1938/1938), 63.74 MiB | 4.12 MiB/s, done.
Resolving deltas: 100% (1215/1215), done.
```

```
In [3]: # on Google Colab execute uncomment the following line and execute to select Tensorflow 1
#%tensorflow_version 1.x
```

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

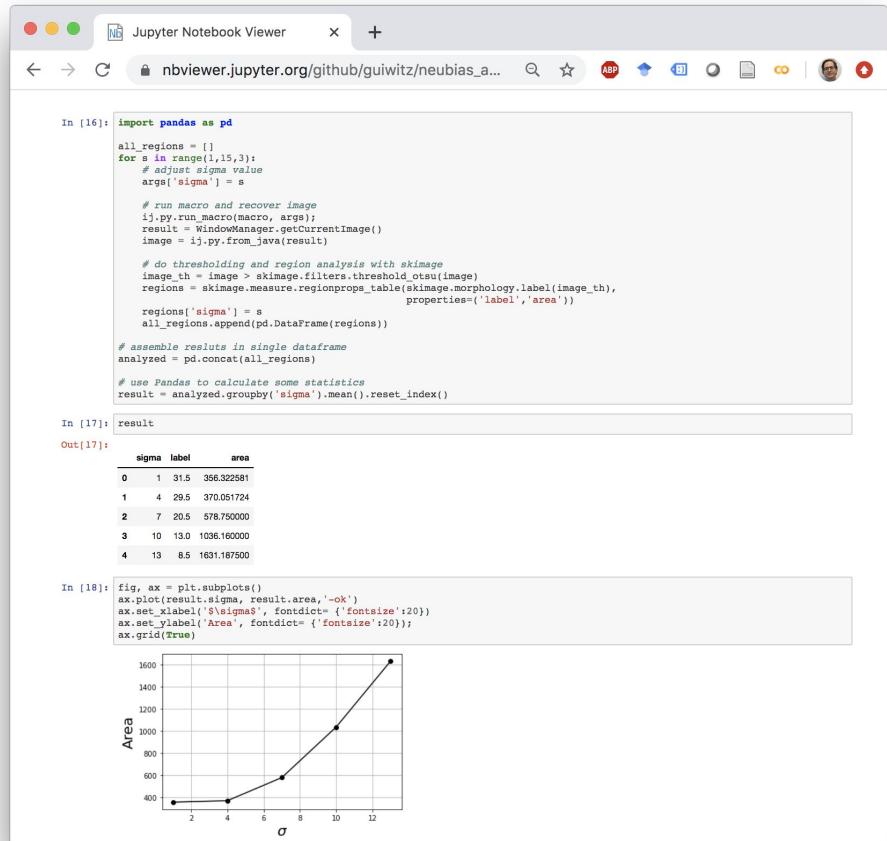
PyimageJ

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



In [16]:

```
import pandas as pd
all_regions = []
for s in range(1,15,3):
    # adjust sigma value
    args['sigma'] = s

    # run macro and recover image
    ij.py.run_macro(macro, args)
    result = WindowManager.getCurrentImage()
    image = ij.py.from_java(result)

    # do thresholding and region analysis with skimage
    image_th = image > skimage.filters.threshold_otsu(image)
    regions = skimage.measure.regionprops_table(skimage.morphology.label(image_th),
                                                properties=['label','area'])

    regions['sigma'] = s
    all_regions.append(pd.DataFrame(regions))

# assemble results in single dataframe
analyzed = pd.concat(all_regions)

# use Pandas to calculate some statistics
result = analyzed.groupby('sigma').mean().reset_index()
```

In [17]:

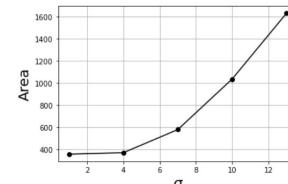
```
result
```

Out[17]:

	sigma	label	area
0	1	31.5	356.322581
1	4	29.5	370.051724
2	7	20.5	578.500000
3	10	13.0	1036.160000
4	13	8.5	1631.187500

In [18]:

```
fig, ax = plt.subplots()
ax.plot(result.sigma, result.area,'-ok')
ax.set_xlabel('$\sigma$@sigma$', fontdict= {'fontsize':20})
ax.set_ylabel('Area', fontdict= {'fontsize':20});
ax.grid(True)
```



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/>