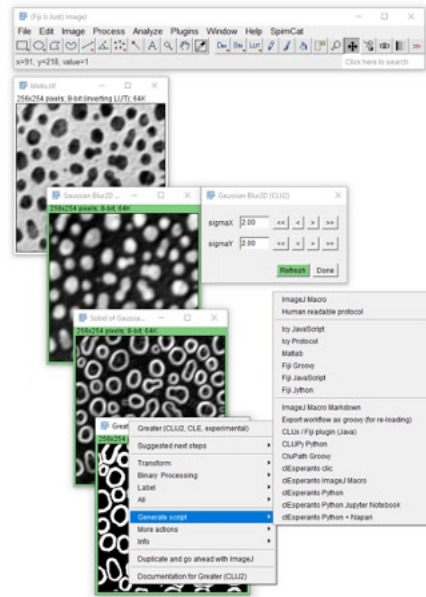


From Assistant to Notebooks

Till Korten



With material from: Robert Haase

- Follow this blog post to install python, mamba and devbio-napari:
<https://t.ly/OBXB>



- <https://t.ly/DVsyu>

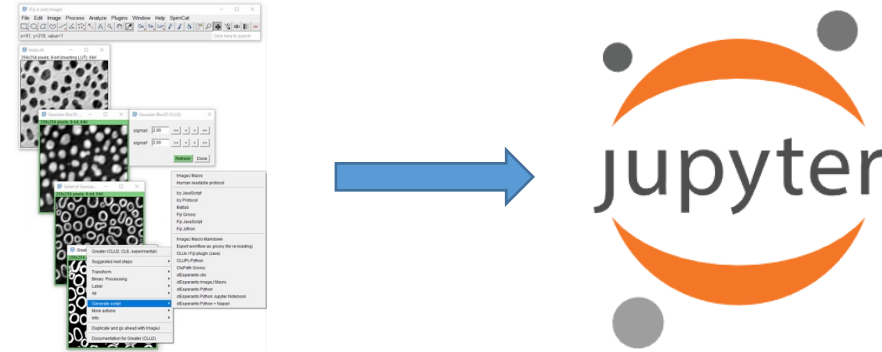


Download sample data and notebooks:

- <https://t.ly/qlma>



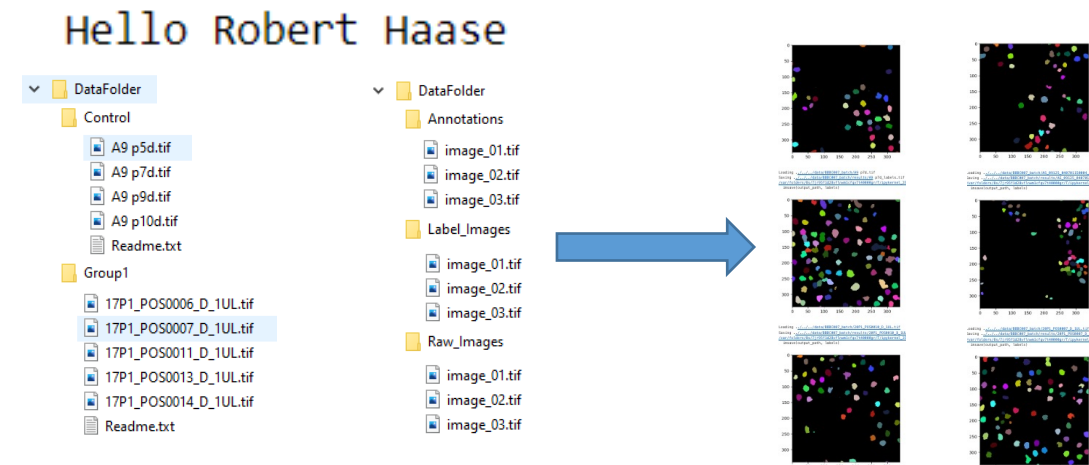
- Exporting workflows as notebooks



- Python programming basics

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

- Processing folders of images



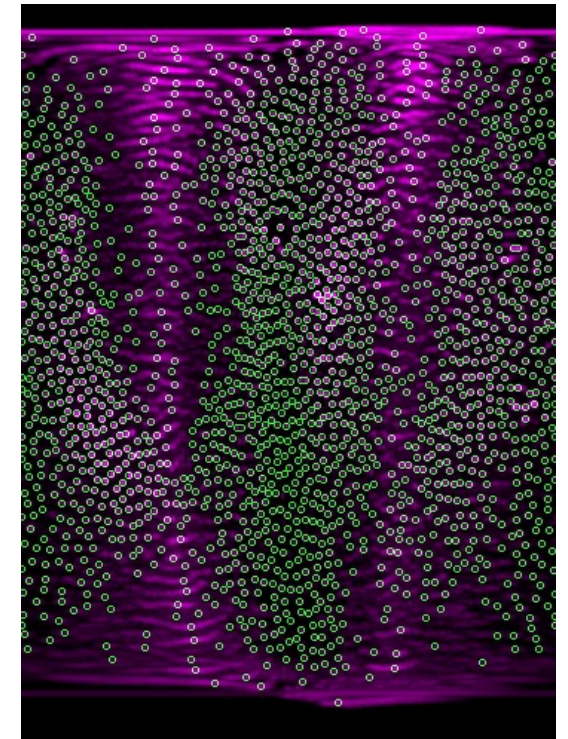
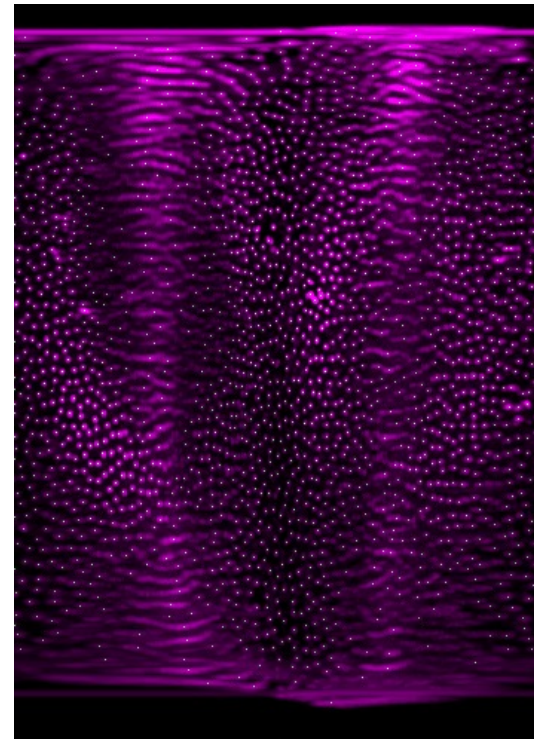
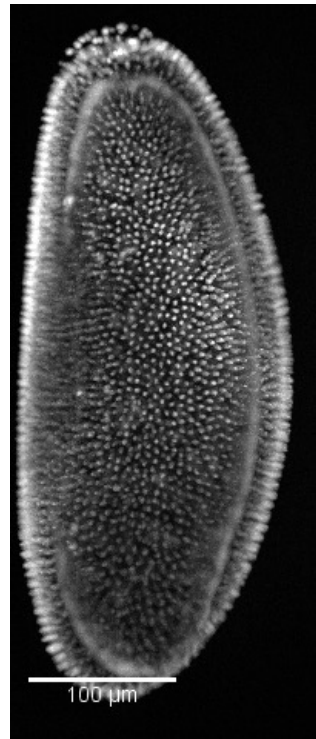
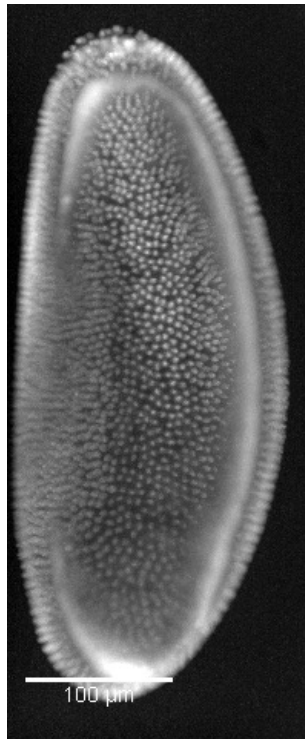
Load data

Preprocessing

Transformation

Segmentation

Save data



- A great way to document your analysis workflow
 - Understandable
 - Reproducible
 - Publishable
 - Reusable - don't forget to add a license: <https://choosealicense.com/>
- Helps automate your workflow (some programming required)
 - Saves time
 - Improves reproducibility and repeatability
 - Reduces bias (but you still need to be careful when you set up the workflow)

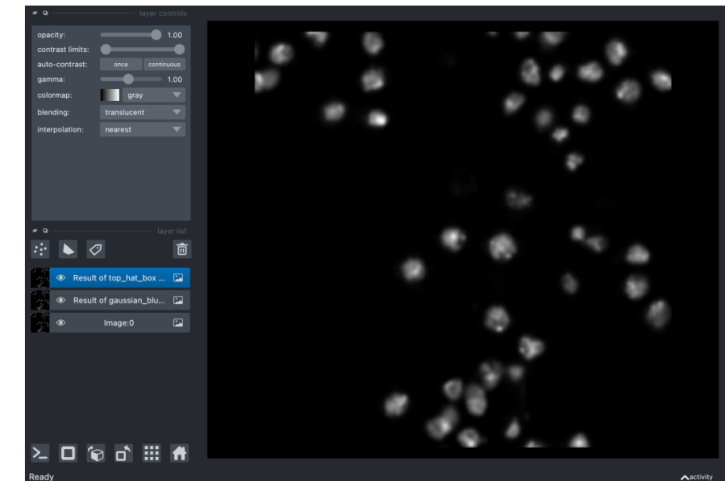
top hat box

Background was removed using a [top-hat algorithm](#) with a symmetric, box-shaped footprint of 10 by 10 pixels.

This algorithm treats objects that are significantly larger than the footprint as background and smaller objects as signal.

```
1 image2_thb = cle.top_hat_box(image1_gb, None, 10.0, 10.0)
2 viewer.add_image(image2_thb, name='Result of top_hat_box')
3 napari.utils.nbscreenshot(viewer)
```

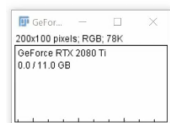
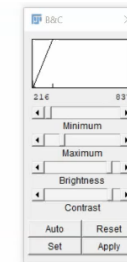
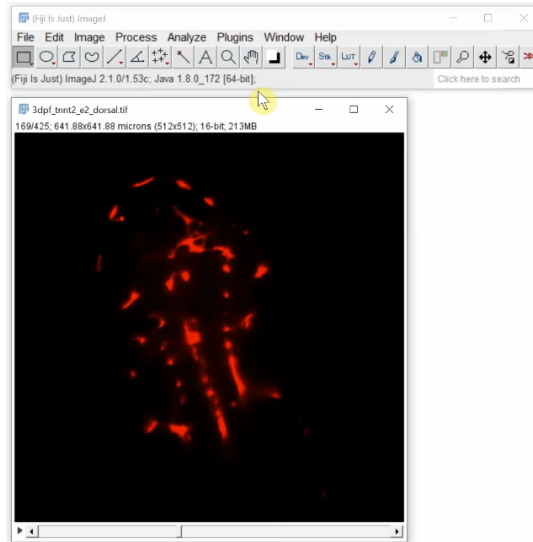
Python



threshold otsu binarization

The image was binarized using a threshold determined by using Otsu's method [1]

- Make x, y, z – projections match biological axes



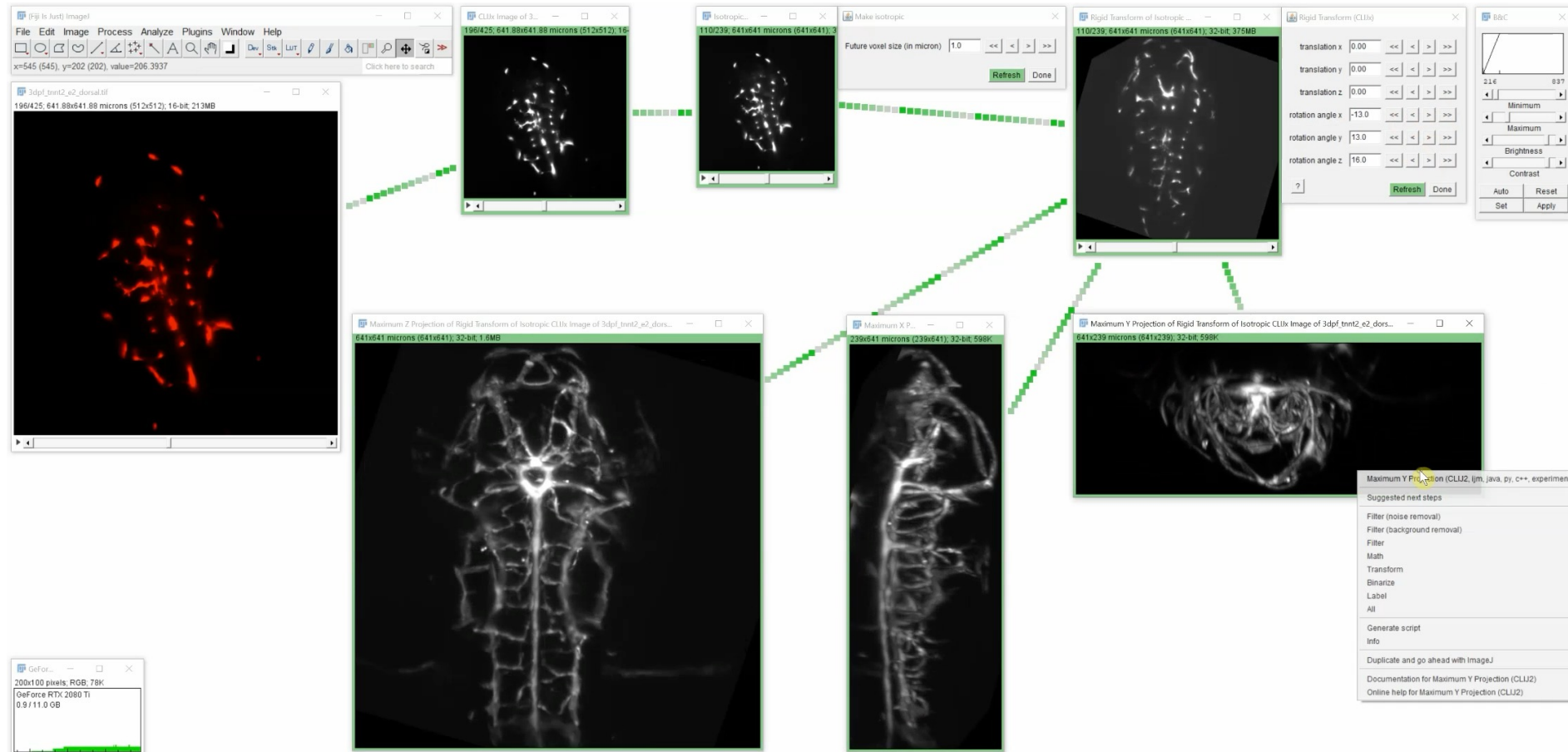
Special
thanks to
Elisabeth
Kugler!



Elisabeth Kugler
@KuglerElisabeth

Image data source: Elisabeth Kugler; labs of Tim Chico and Paul Armitage, The University of Sheffield (UK)" <https://zenodo.org/record/4204839#.X8DCRGj7Q2w>

- After setting up the workflow, generate code!



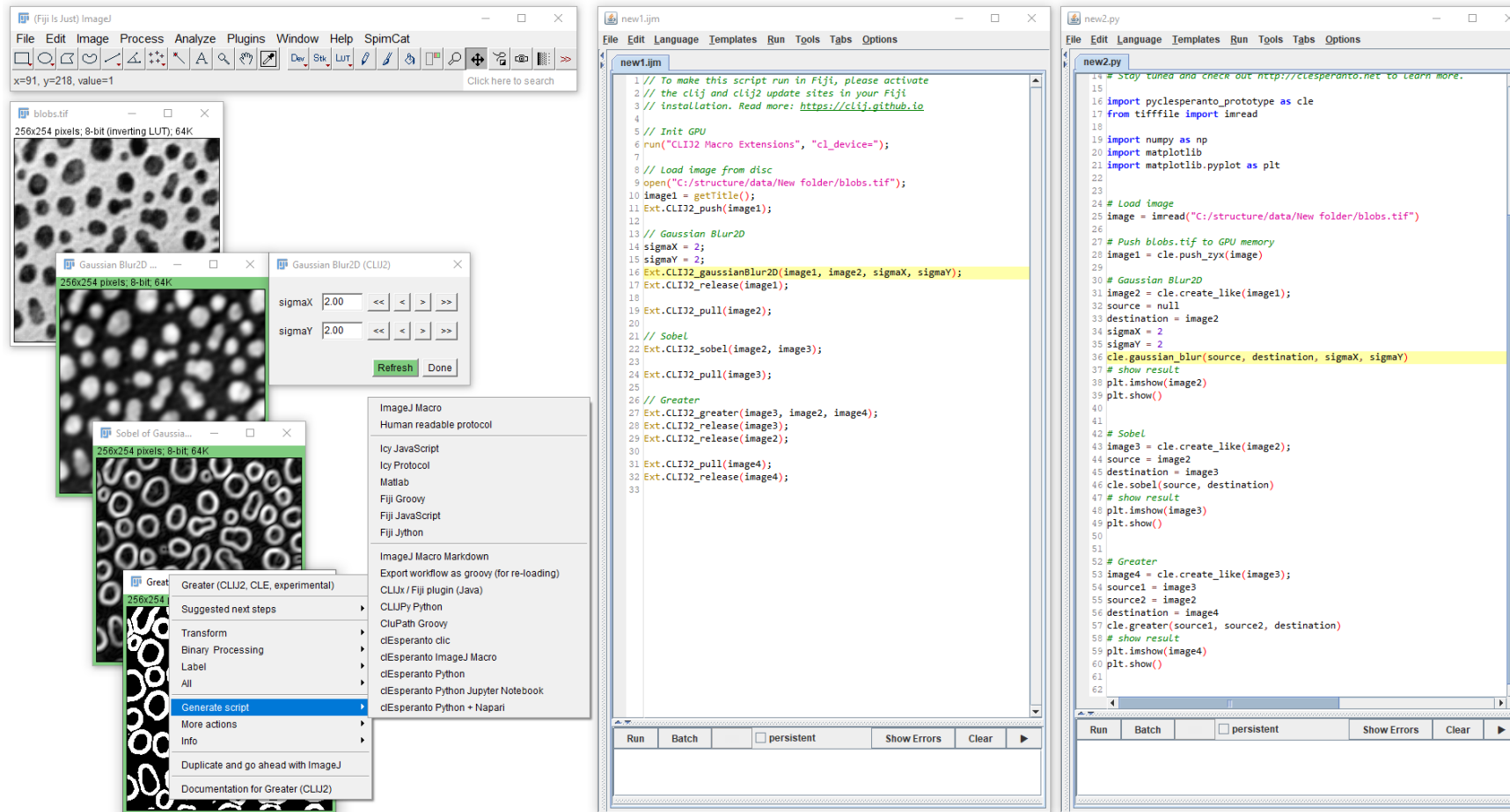
Special
thanks to
Elisabeth
Kugler!



Elisabeth Kugler
@KuglerElisabeth

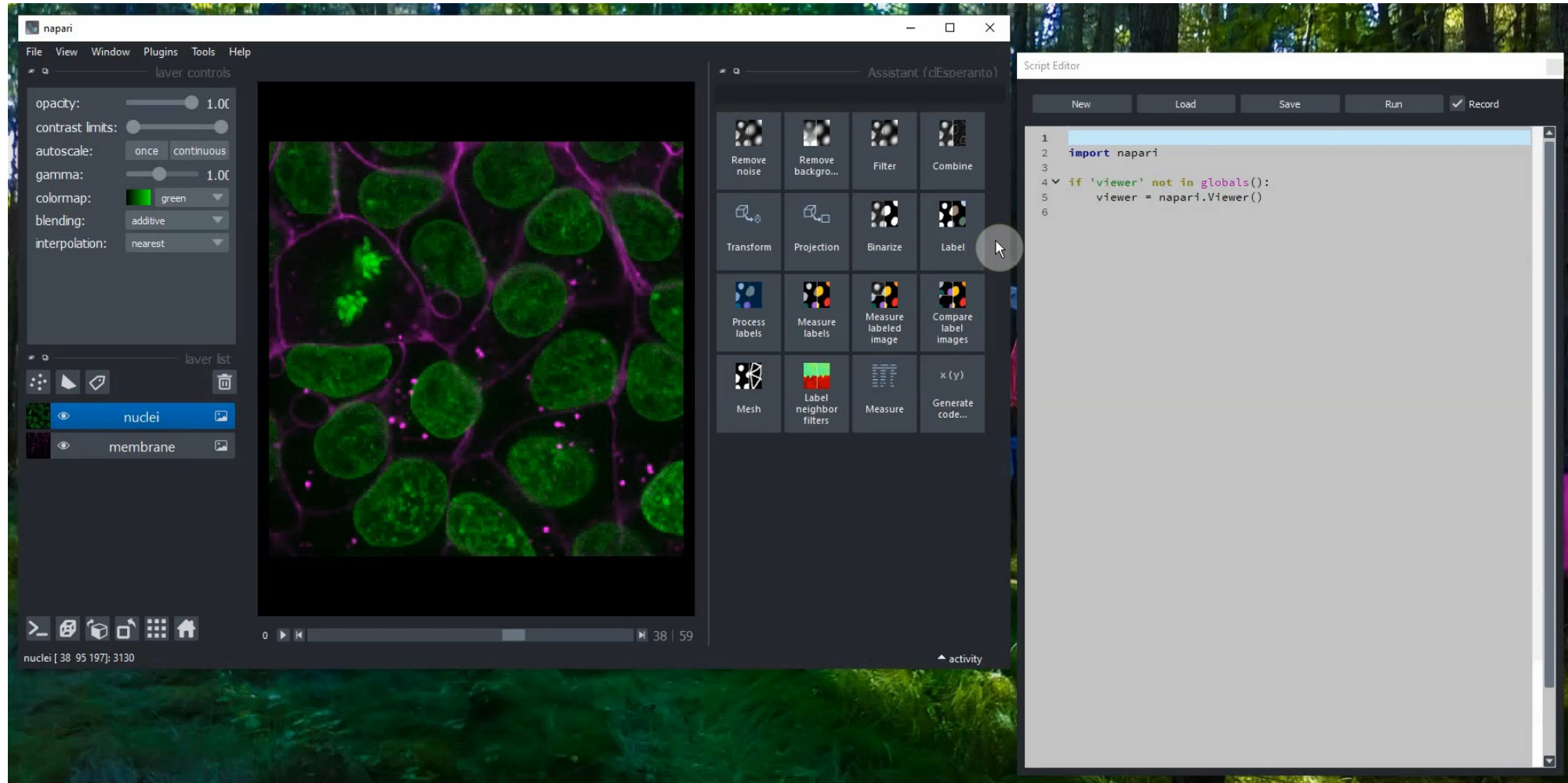
Image data source: Elisabeth Kugler; labs of Tim Chico and Paul Armitage, The University of Sheffield (UK)" <https://zenodo.org/record/4204839#.X8DCRGj7Q2u>

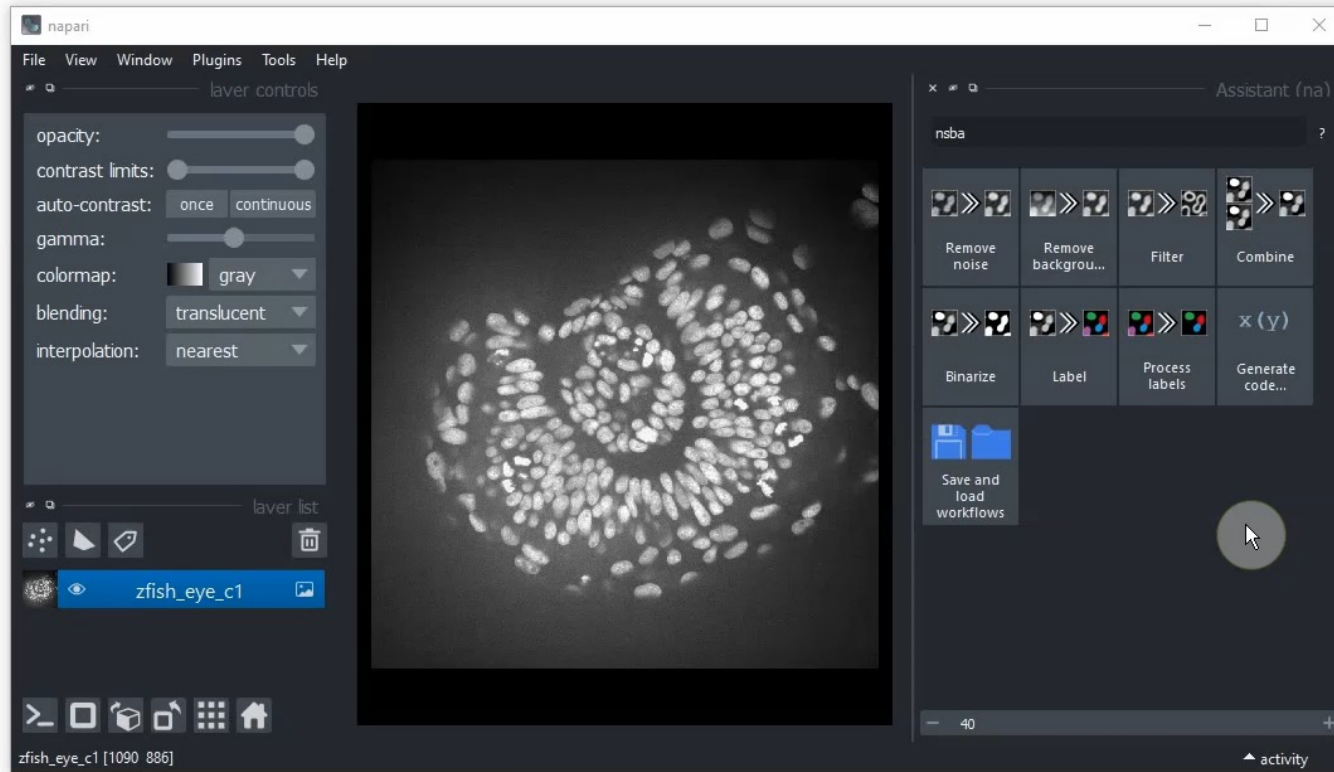
Choosing a programming language



- Do you have access to existing scripts from others (and does the license allow you to use them)?
- Do you need other tools/packages (e.g. a deep learning python package)?
- Personal preference

Create a workflow with Napari-Assistant



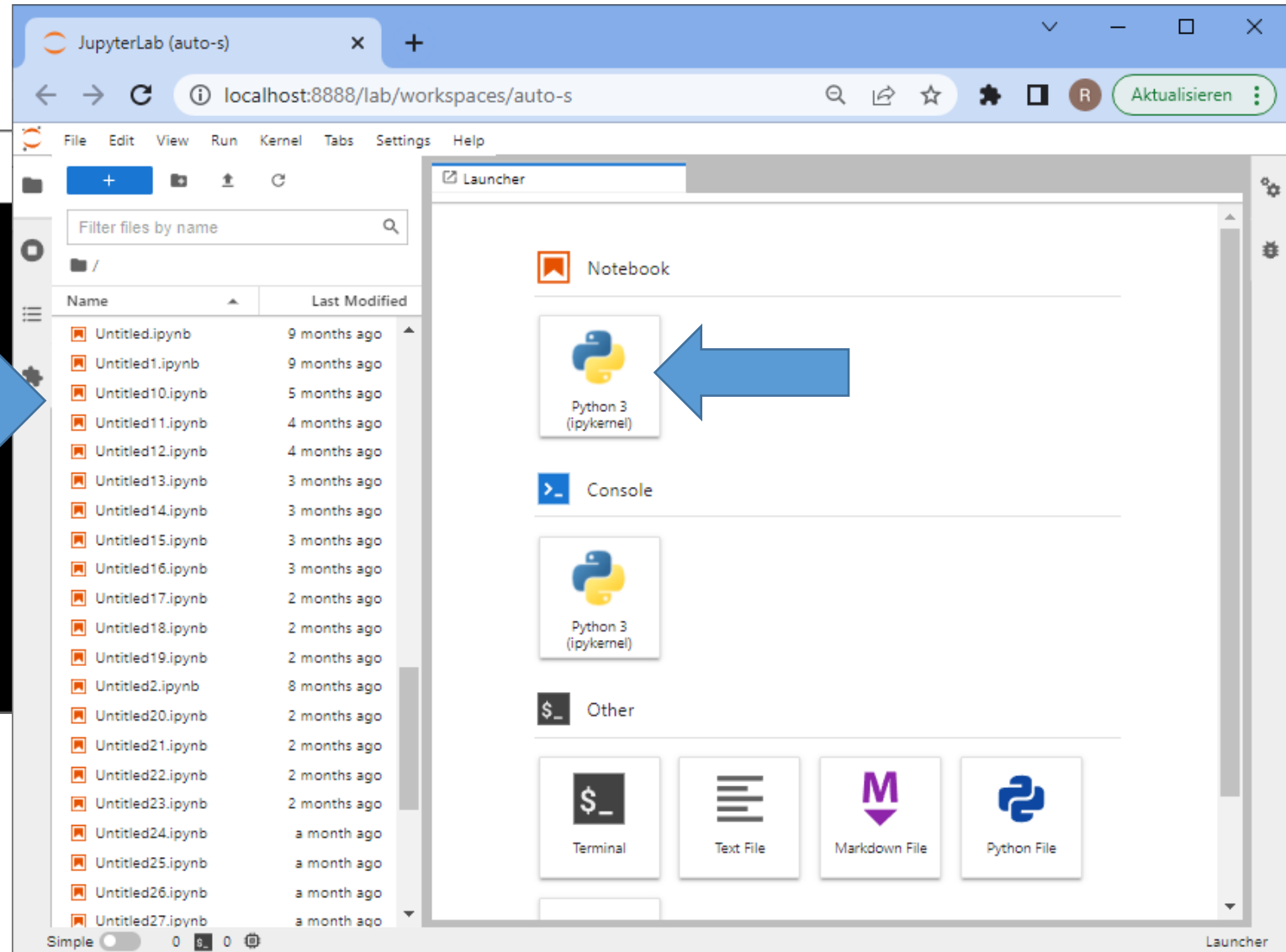
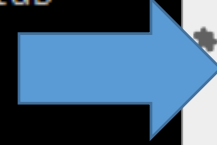


<https://github.com/haesleinhuepf/napari-assistant>

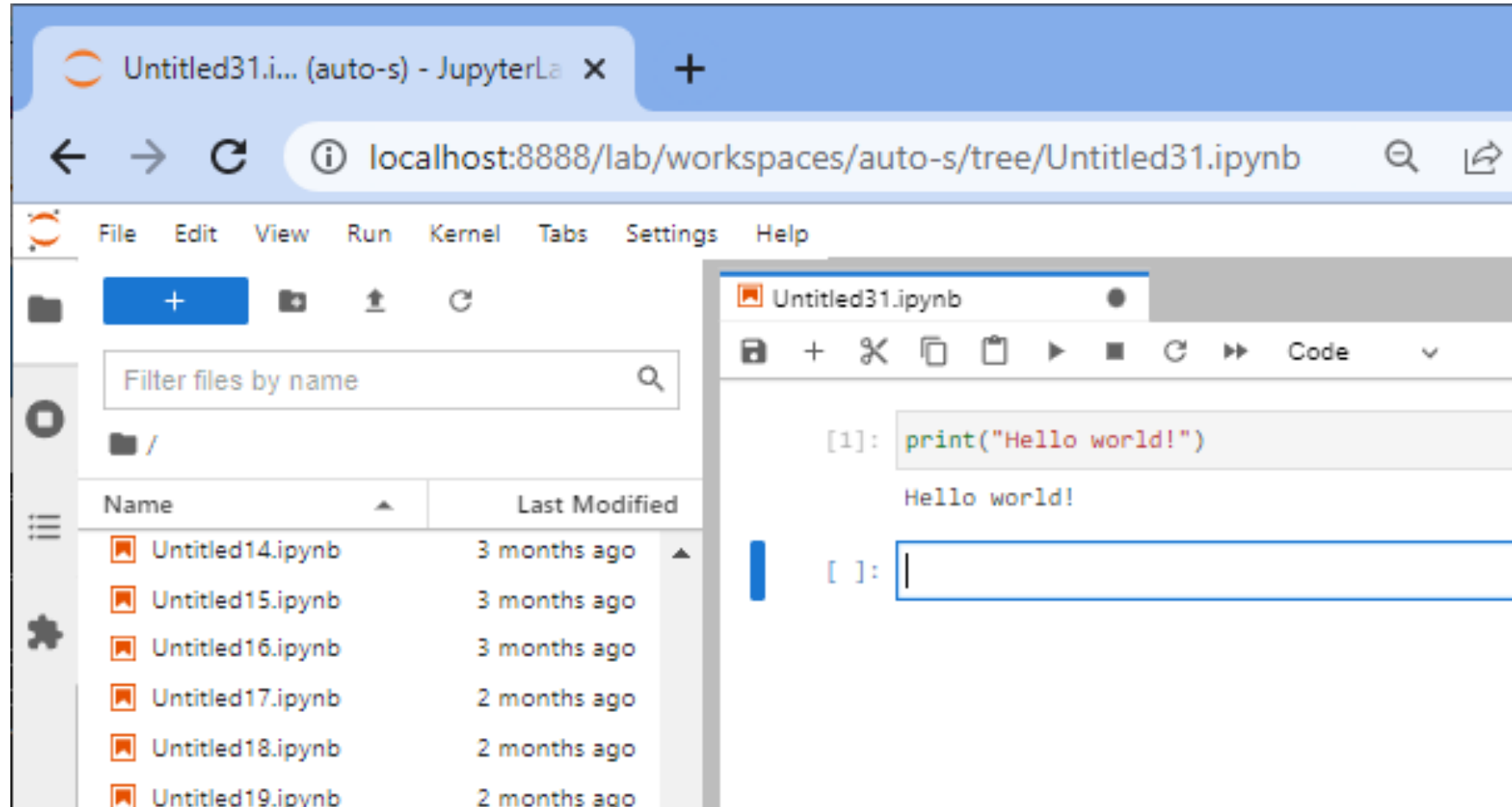
Image data source: Mauricio Rocha Martins, Norden lab, MPI CBG (now at IGC Oeiras)

Open the notebook in Jupyter lab

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

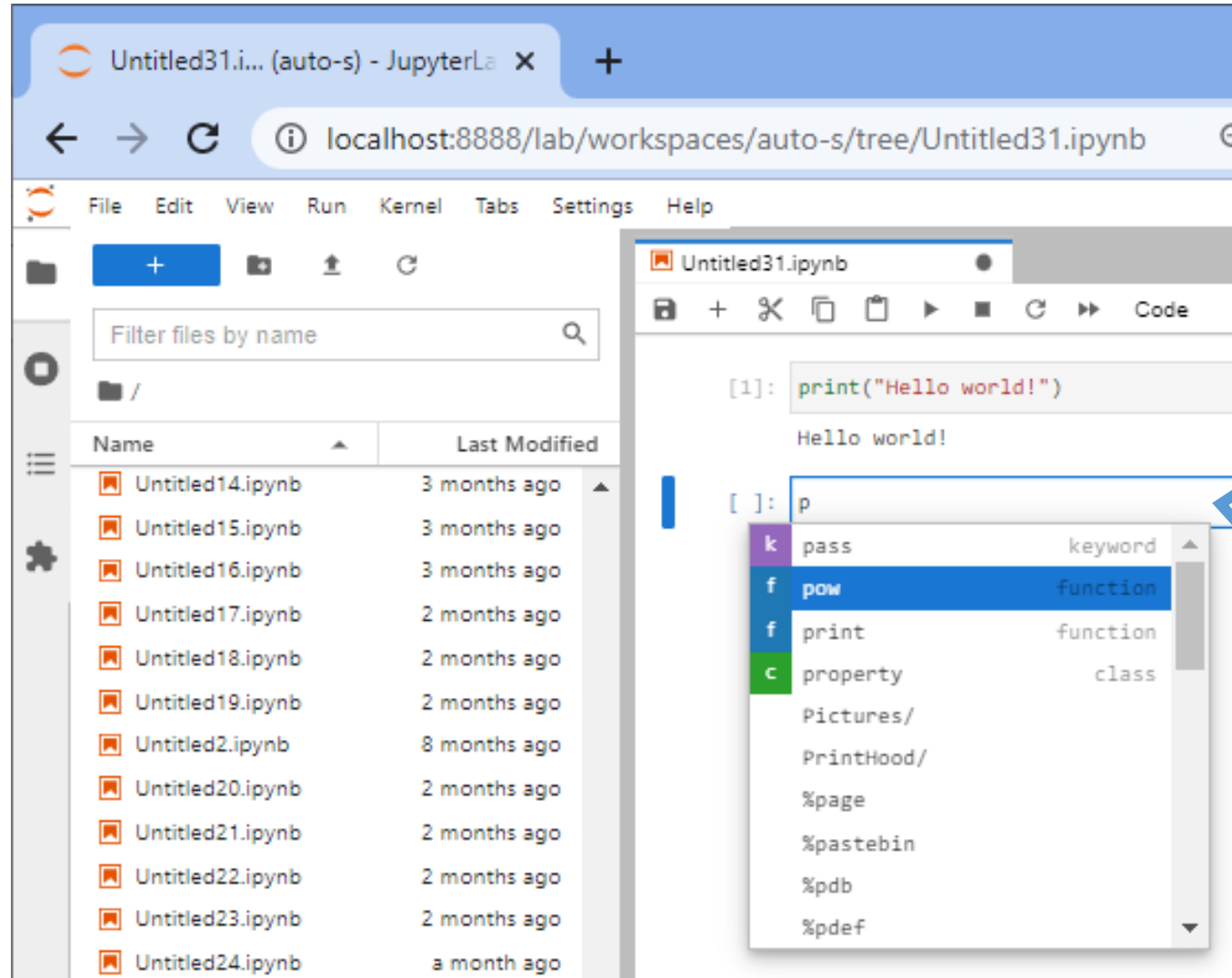


- Execute code cell-by-cell and see results instantaneously



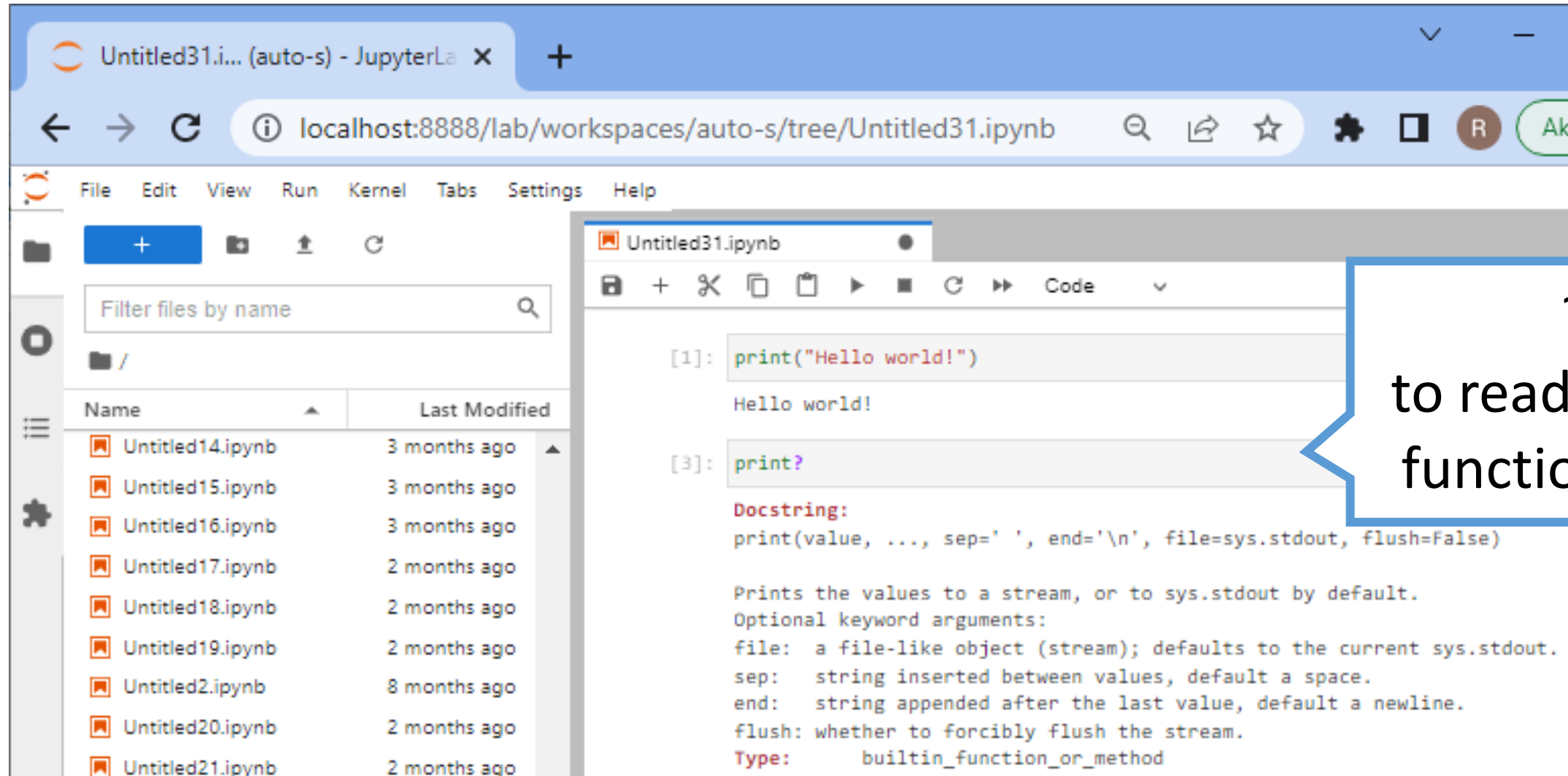
SHIFT + ENTER
to execute a
code cell

- Context-specific help, auto-completion



TAB
to open auto-
completion

- Help / “docstrings”



The screenshot shows the JupyterLab interface. On the left is a file browser with a search bar and a list of files. The main area on the right is the code editor for 'Untitled31.ipynb'. It shows two code cells. The first cell contains `print("Hello world!")` and has executed, showing 'Hello world!'. The second cell contains `print?` and has also executed, displaying the docstring for the `print` function.

File Browser:

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

Code Editor:

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

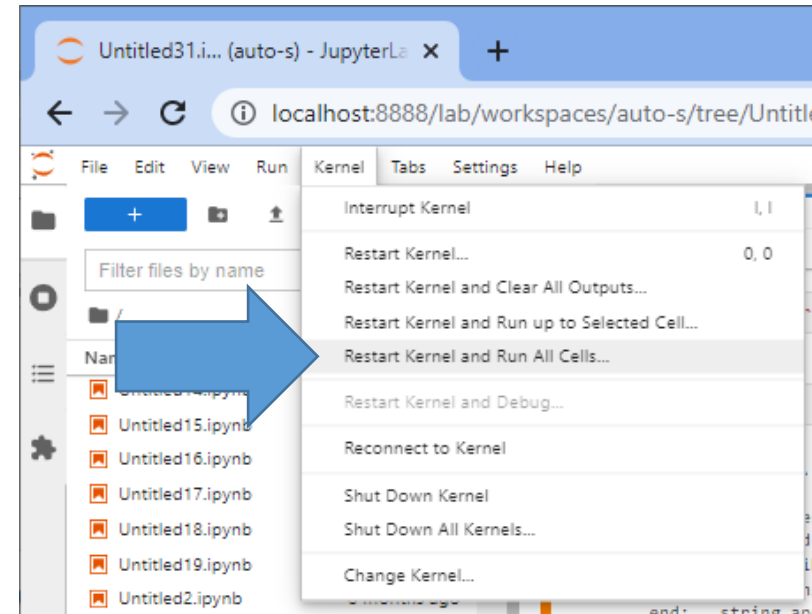
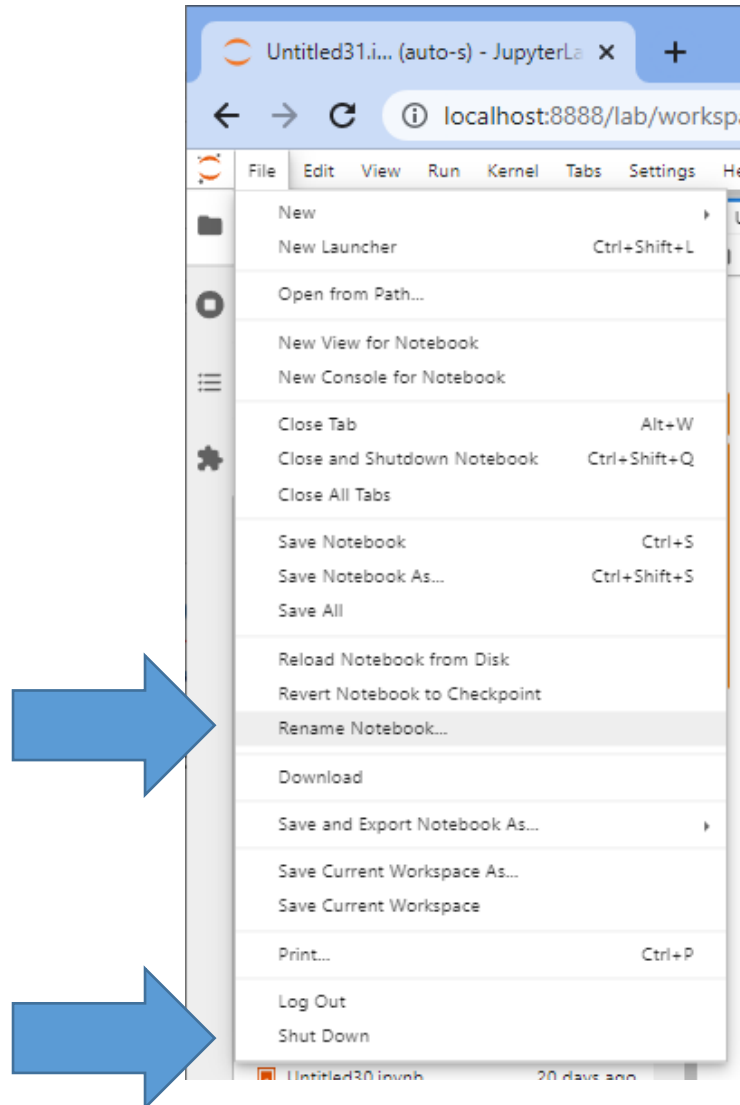
[3]: print?

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

- Saving / renaming / closing



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

- Example/exercise notebooks online: <https://t.ly/DVsyu>

