

From Assistant to Notebooks

Till Korten



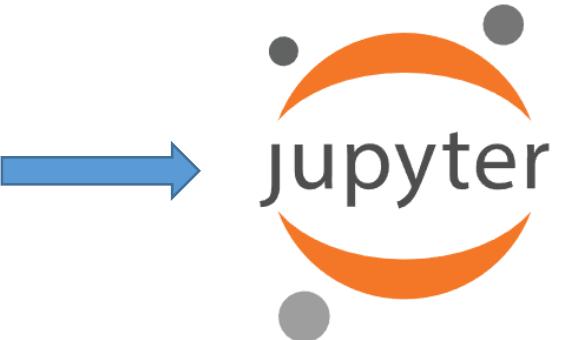
With material from: Robert Haase

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



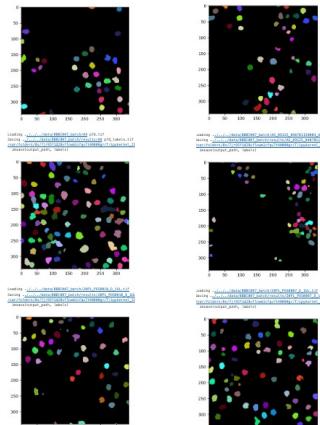
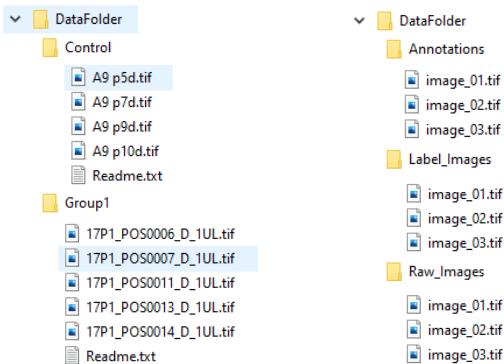
Outline

- Exporting workflows as notebooks
- Python programming basics
- Processing folders of images



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

Hello Robert Haase



Recap – image analysis workflows

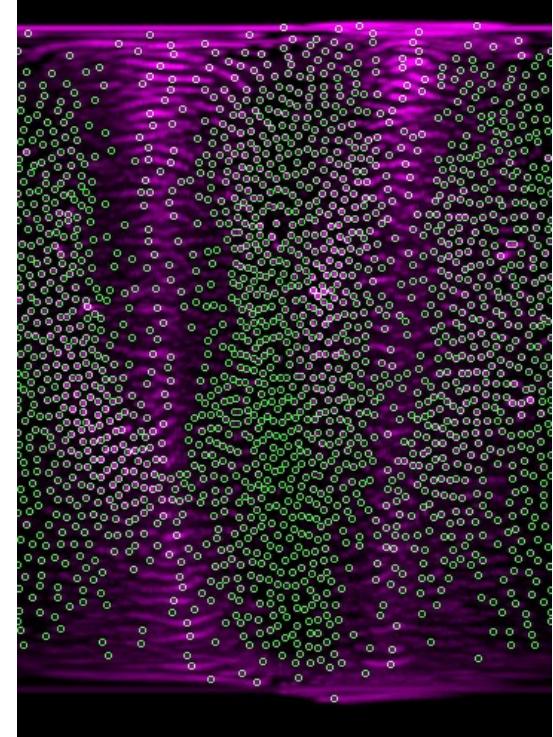
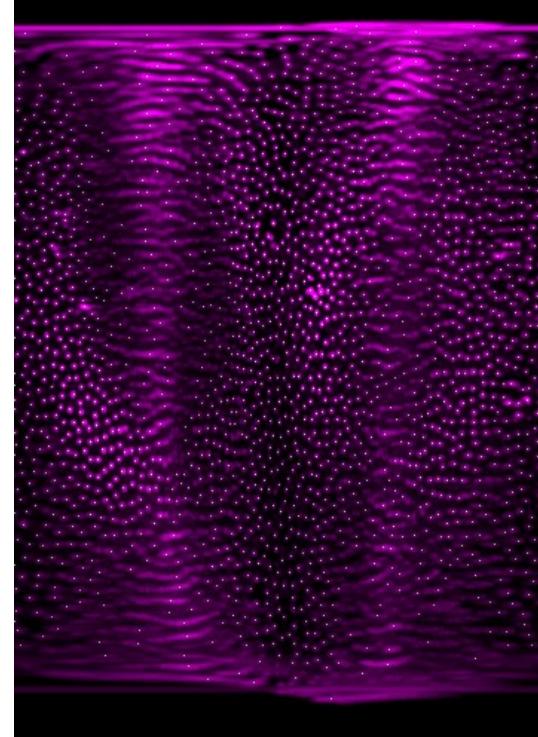
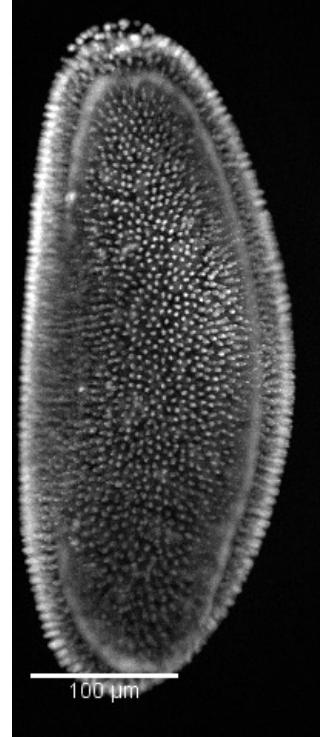
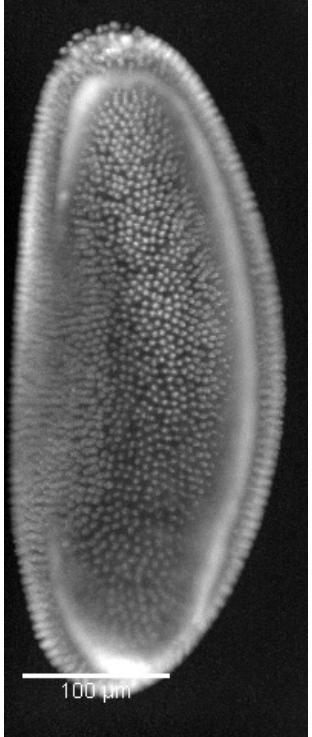
Load data

Preprocessing

Transformation

Segmentation

Save data



Why Notebooks?

- 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)

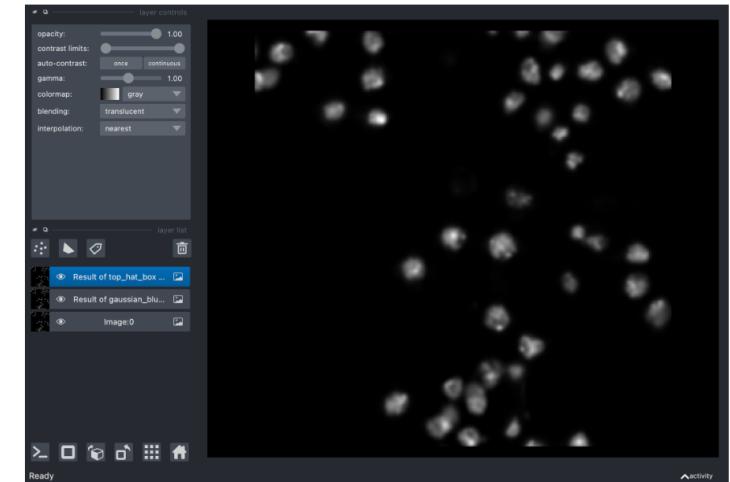
top hat box

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)
```

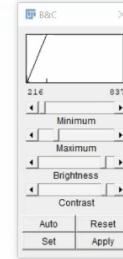
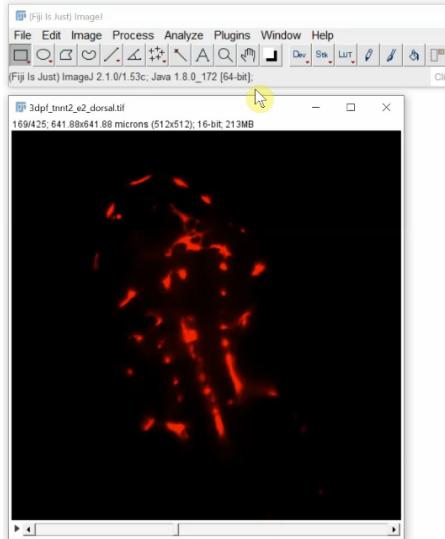
Python



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!

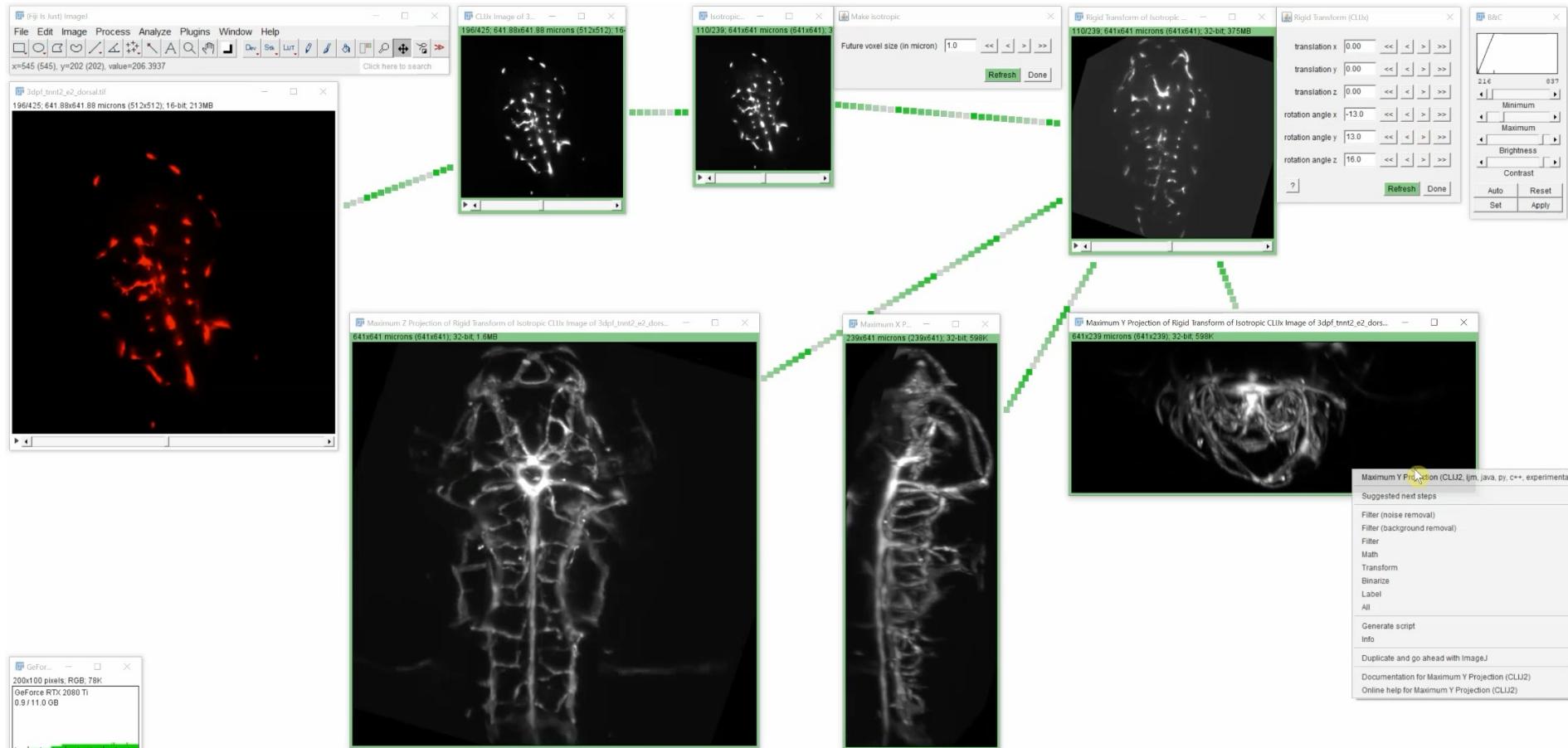


Elisabeth Kugler
@KuglerElisabeth

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!



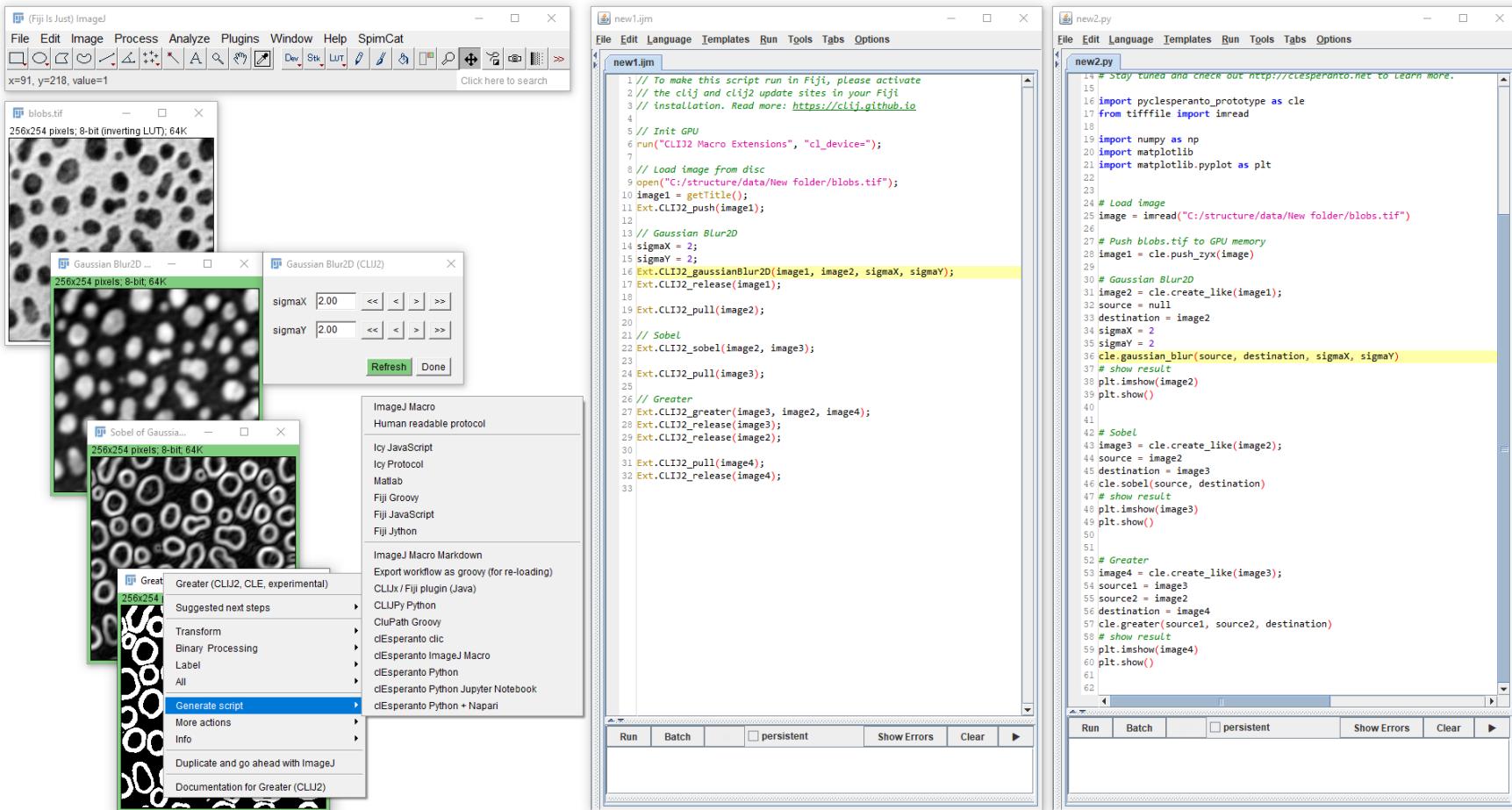
Special thanks to
Elisabeth
Kugler!



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#.X8DCRGj7Q2w>

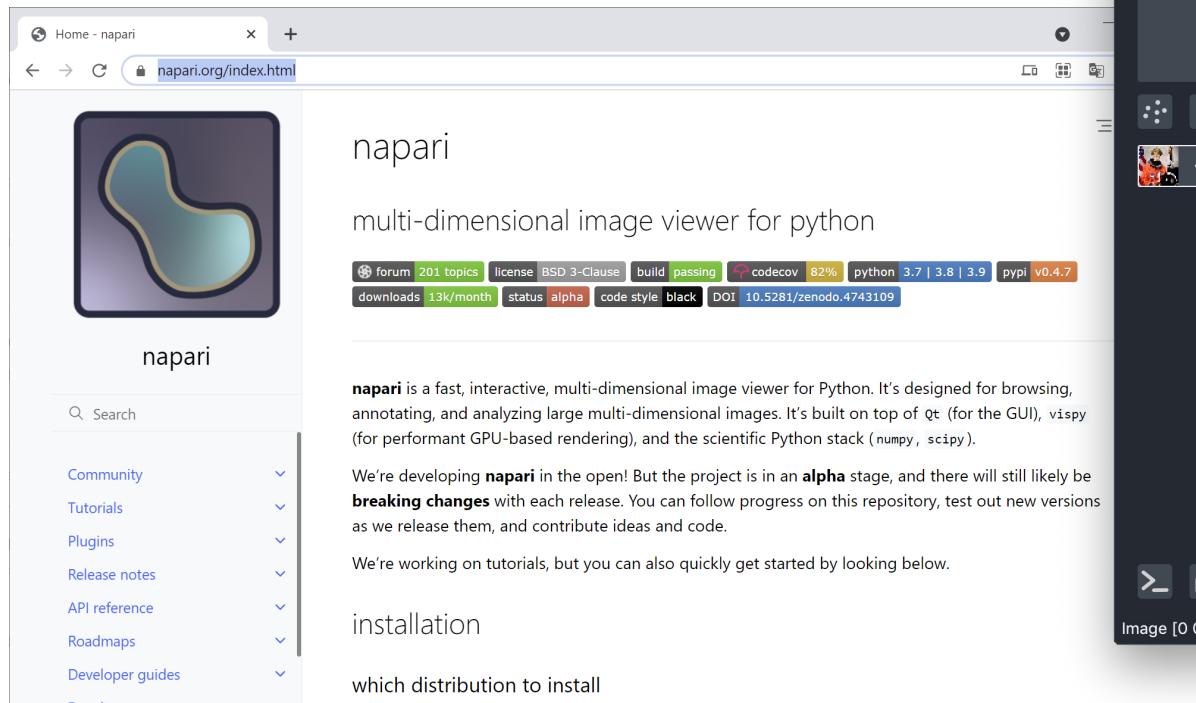
Choosing a programming language



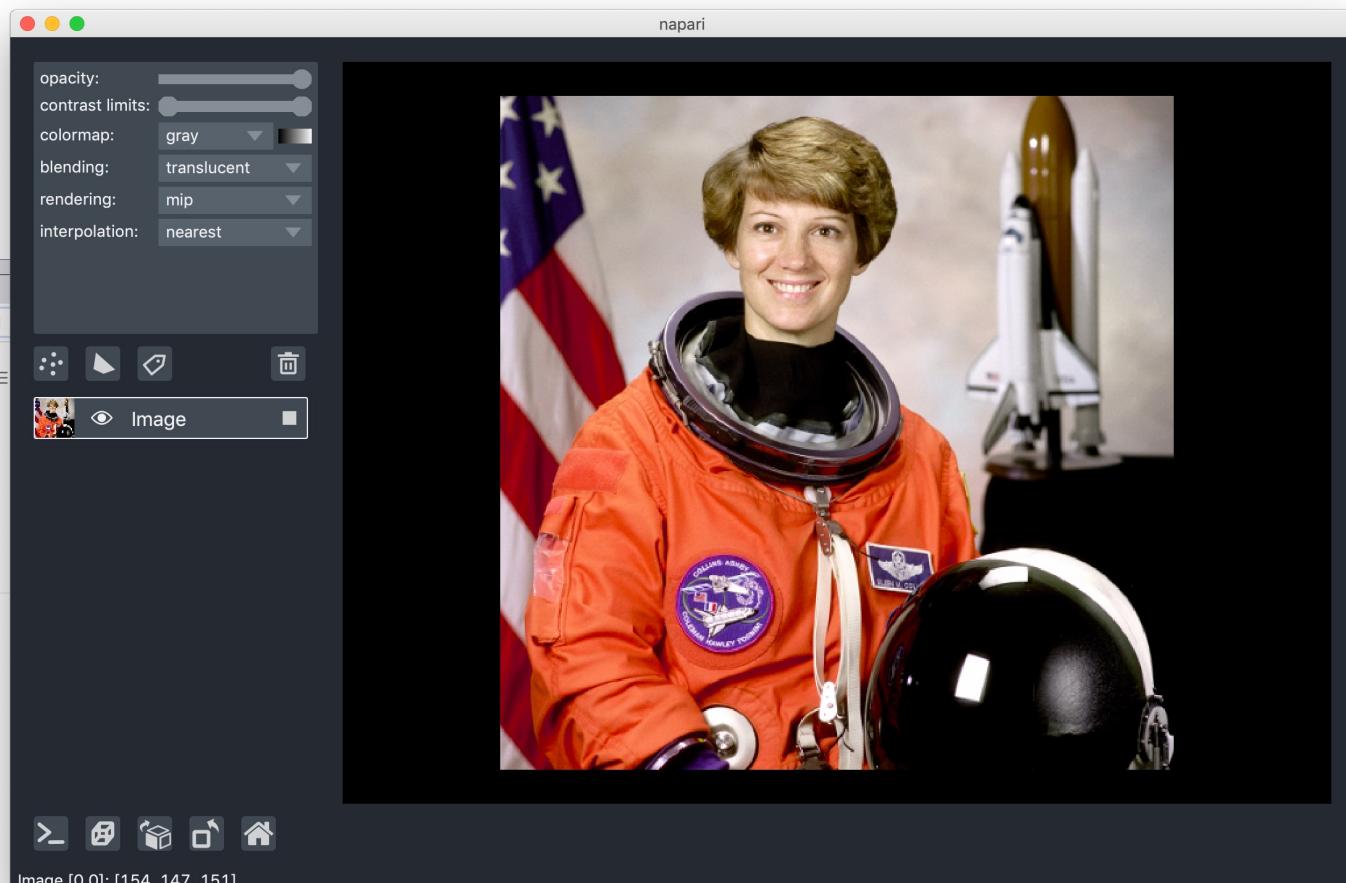
- Do you have access to existing scripts from others (and does the license allow you to use them)?
- Do you need other tools/packages (e.g. a deep learning python package)?
- Personal preference

Napari: 3D viewer for Python

- Multi-dimensional image viewer in python
- <https://napari.org/>



The screenshot shows the official napari website at napari.org/index.html. The page features a large logo icon on the left, followed by the word "napari" and a subtitle "multi-dimensional image viewer for python". Below this is a navigation bar with links to "forum", "license", "build", "codecov", "python", "downloads", "status", "code style", "DOI", and "pypi". A sidebar on the left contains links for "Community", "Tutorials", "Plugins", "Release notes", "API reference", "Roadmaps", and "Developer guides". The main content area includes a brief description of what napari is, a note about its development stage, and sections for "installation" and "which distribution to install".



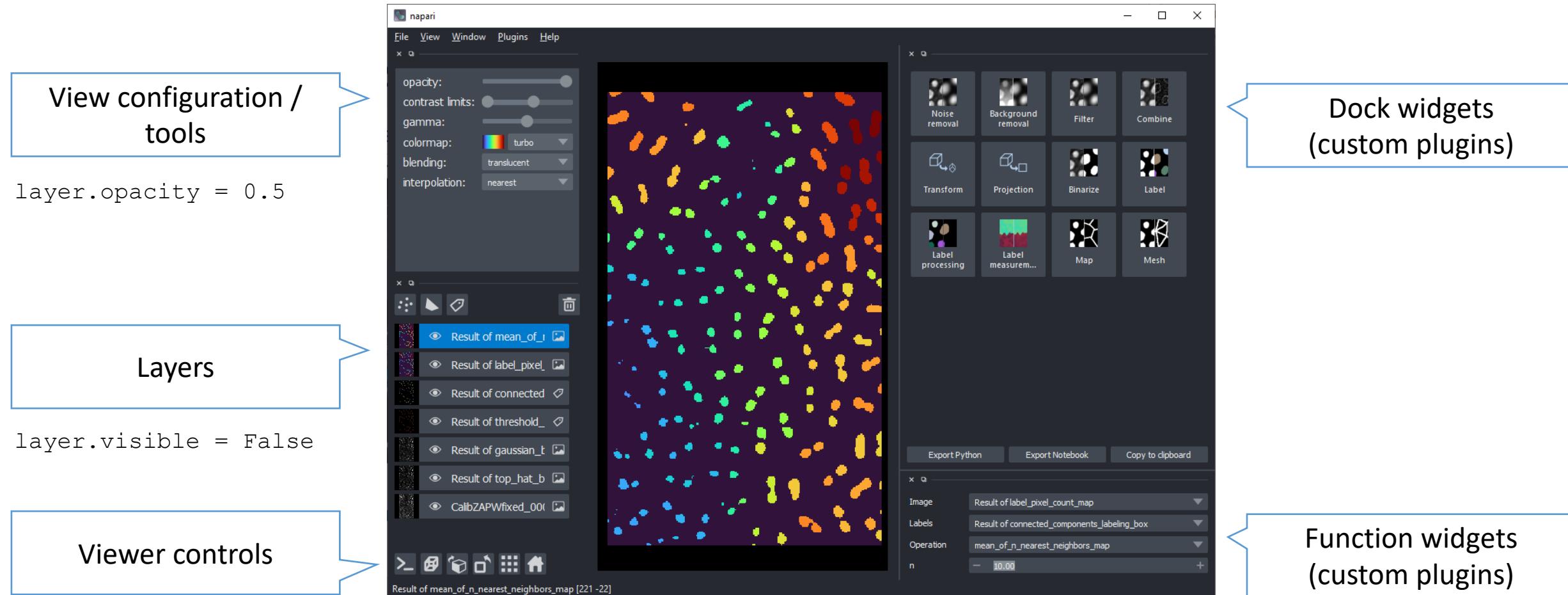
<https://napari.org/>

Napari: 3D viewer for Python

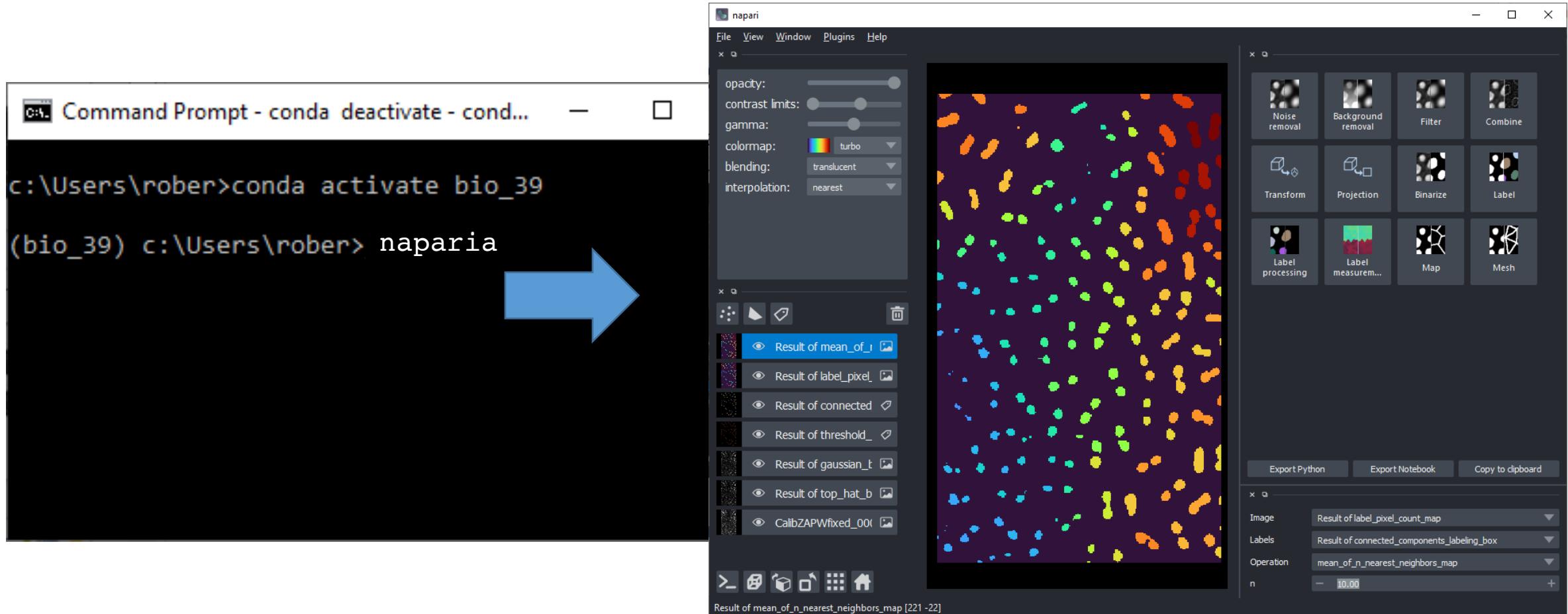


Image data source: Daniela Vorkel, Myers lab, MPI-CBGB/CSBD

Napari user interface



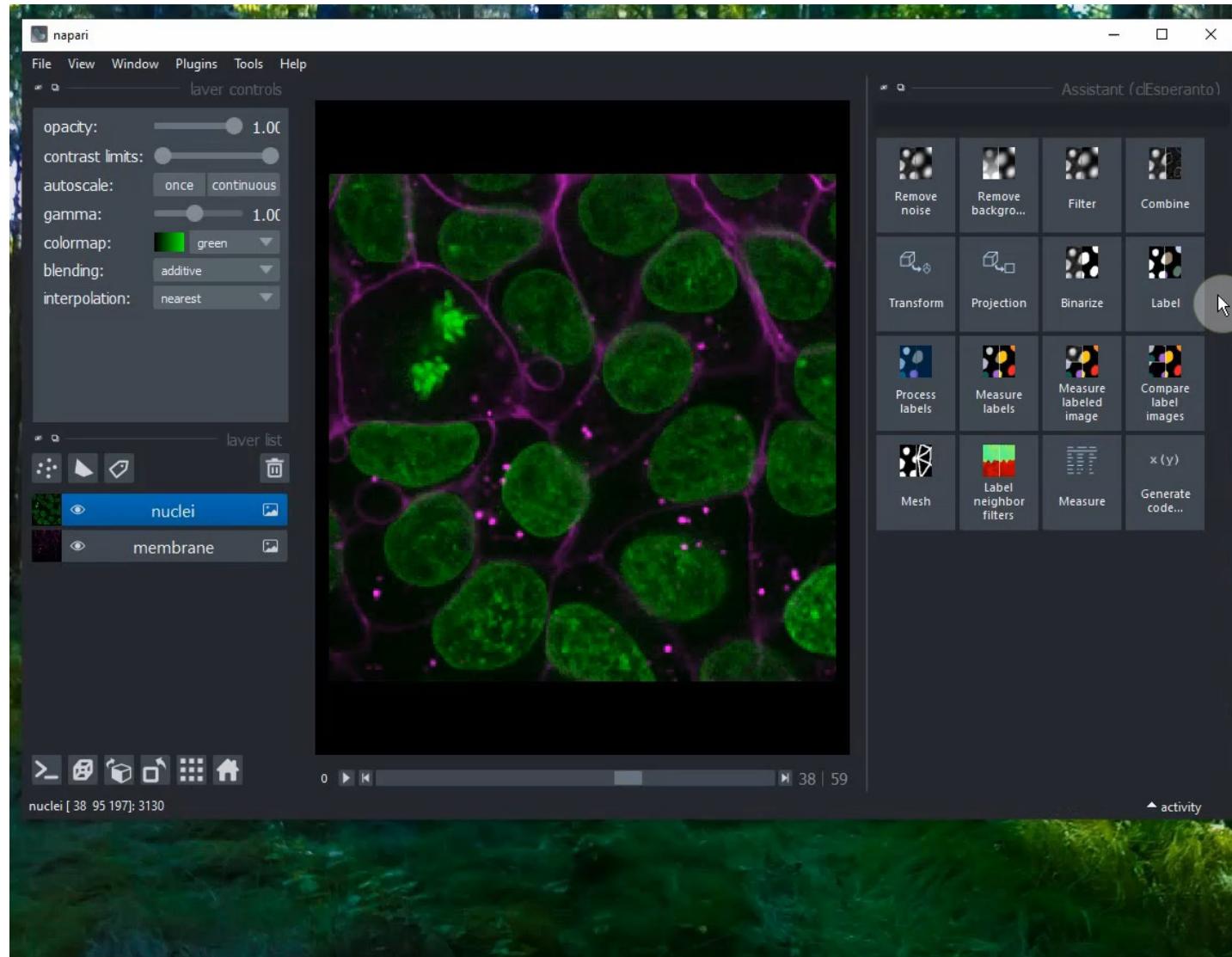
Open napari assistant



Create a workflow with Napari-Assistant



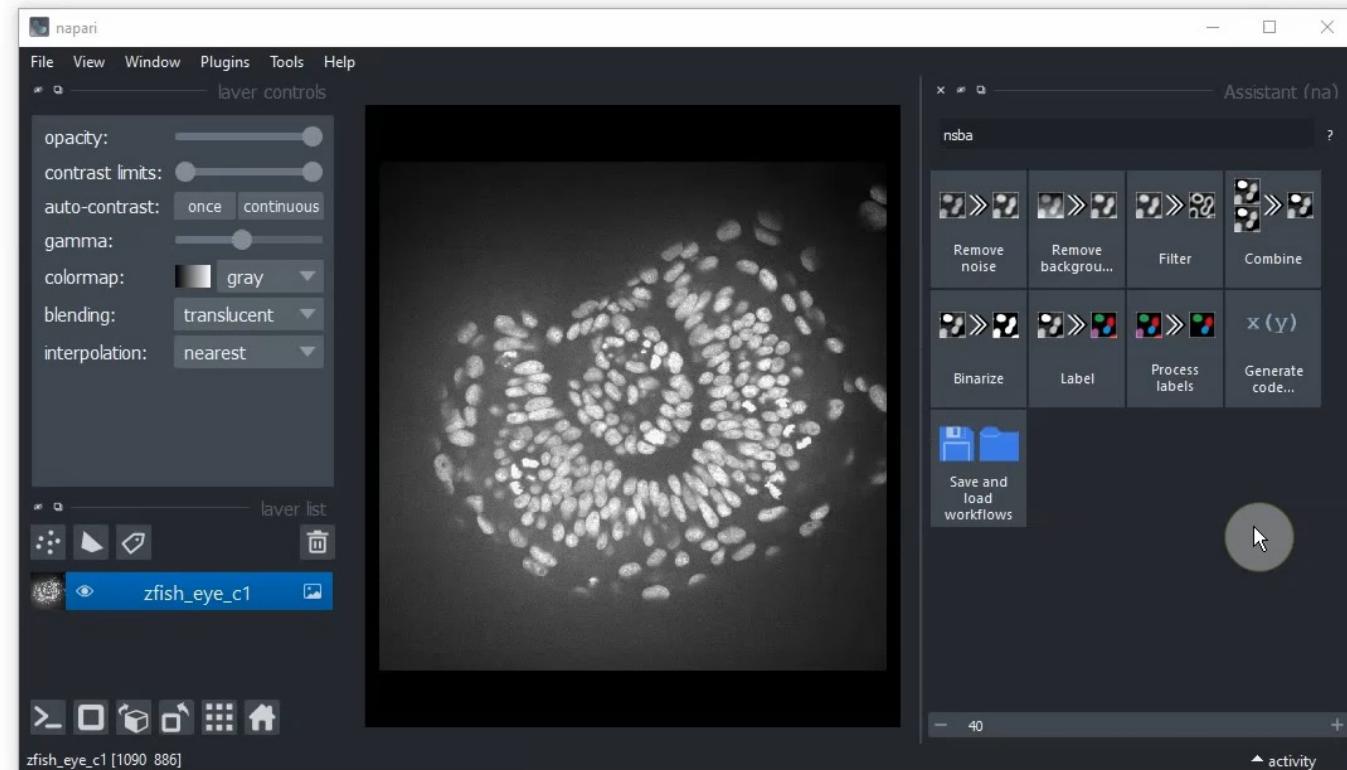
PoL
Physics of Life
TII Dresden



<https://www.napari-hub.org/plugins/napari-script-editor>

May 2023

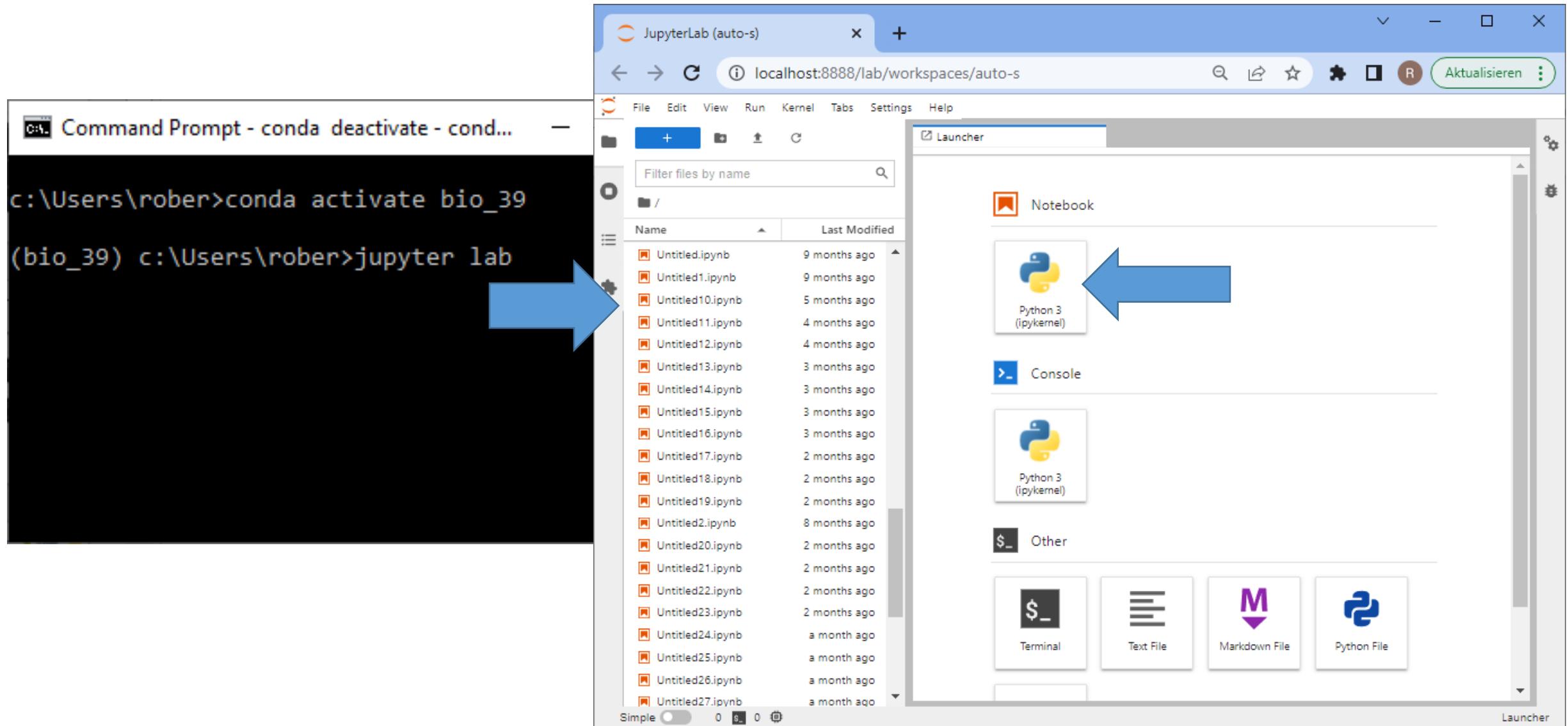
Export code to Jupyter Notebooks



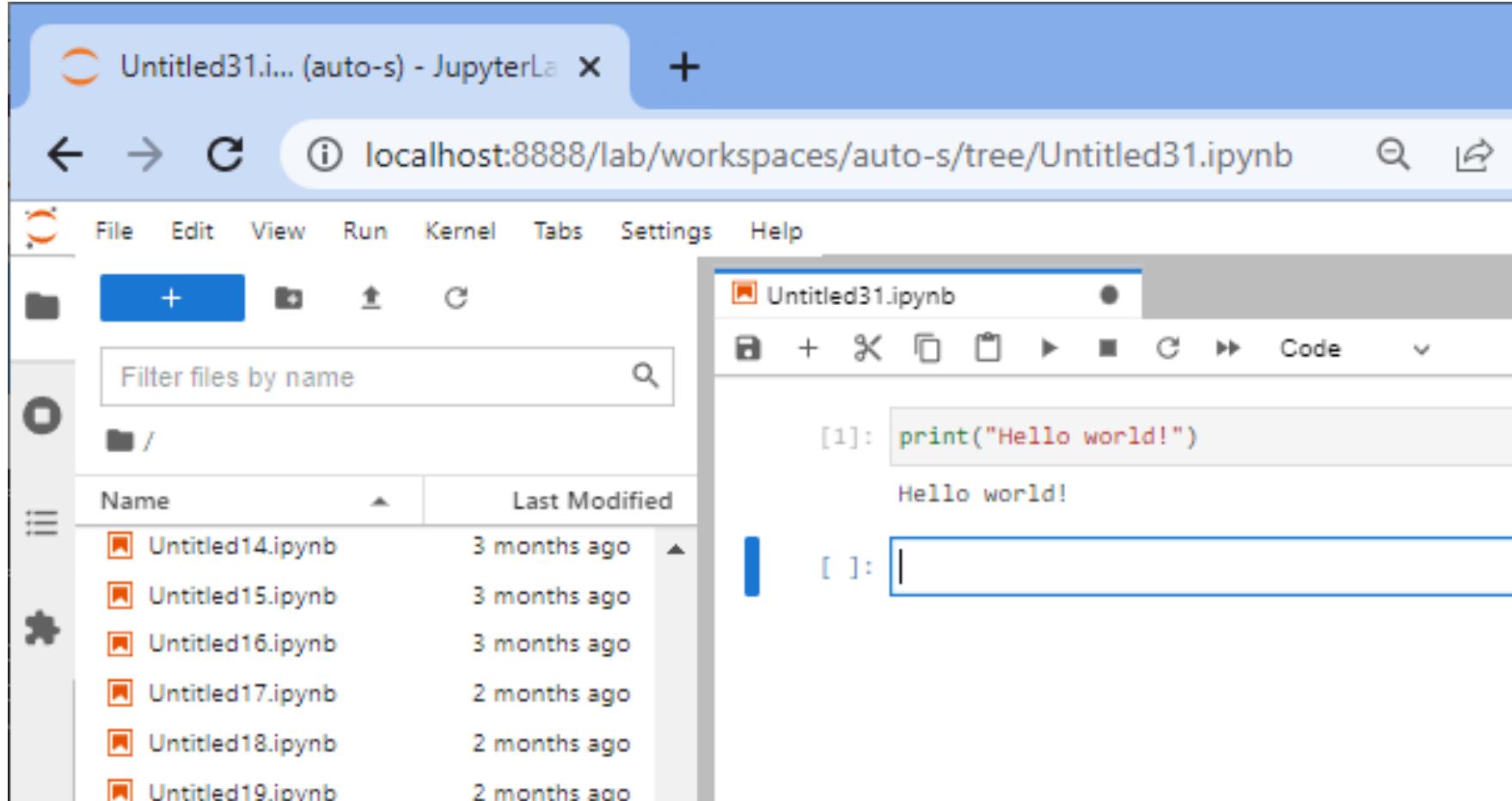
<https://github.com/haesleinhuepf/napari-assistant>

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

Open the notebook in Jupyter lab

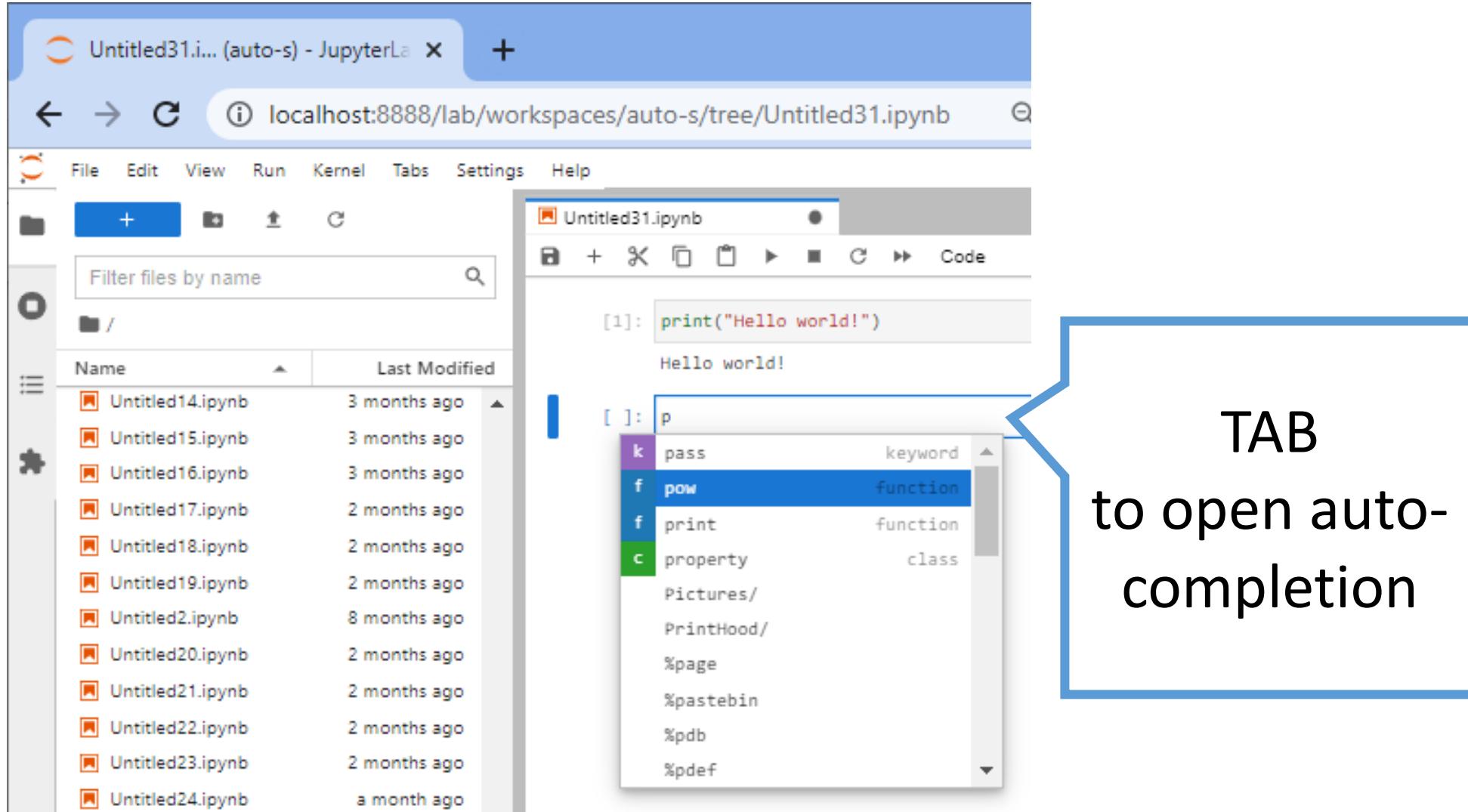


- Execute code cell-by-cell and see results instantaneously



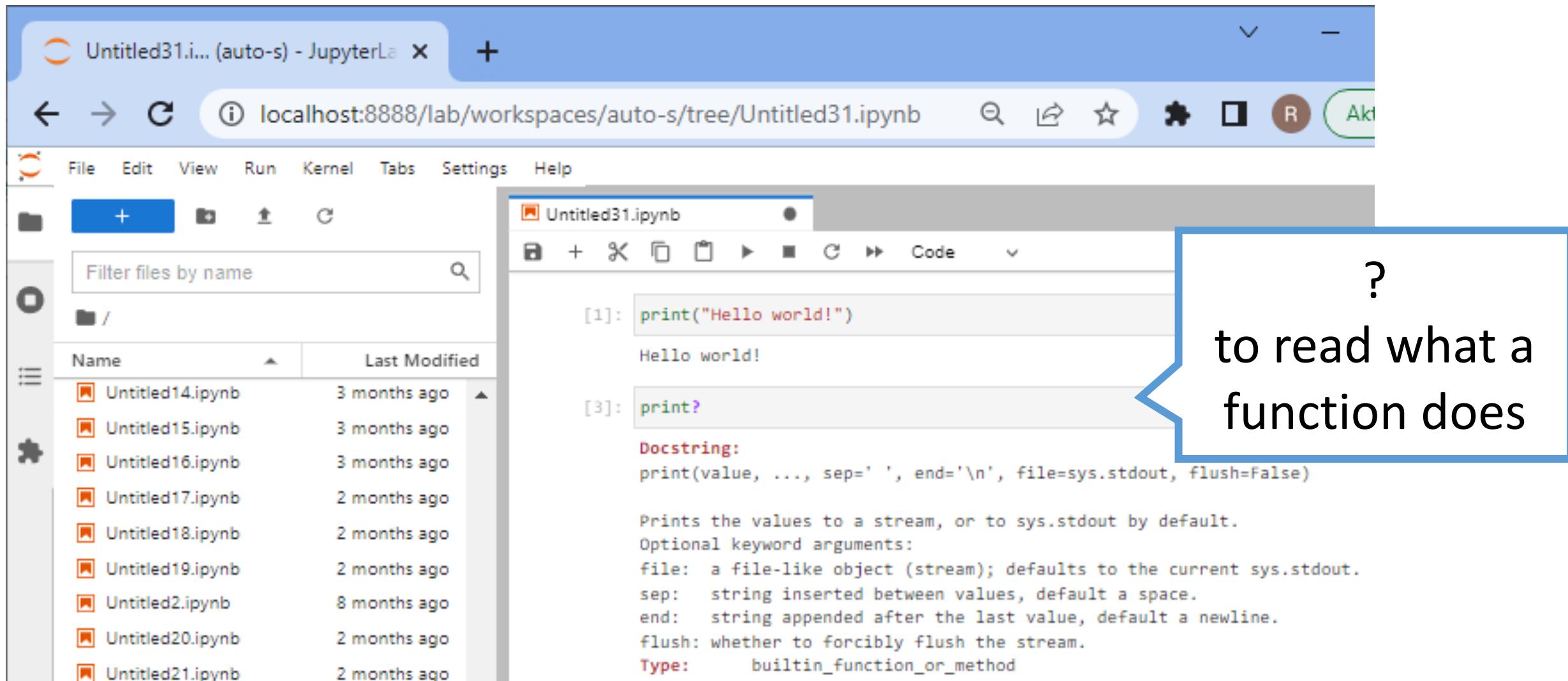
SHIFT + ENTER
to execute a
code cell

- Context-specific help, auto-completion



TAB
to open auto-
completion

- Help / “docstrings”



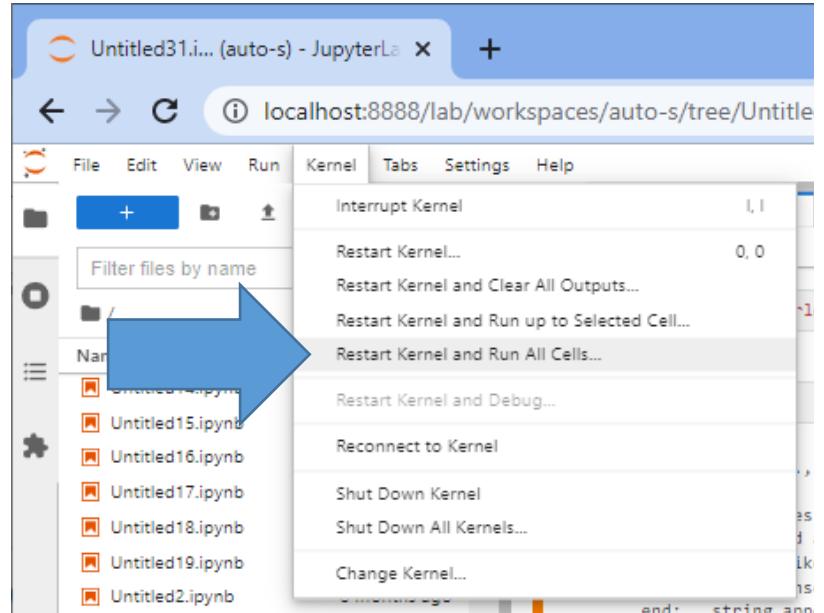
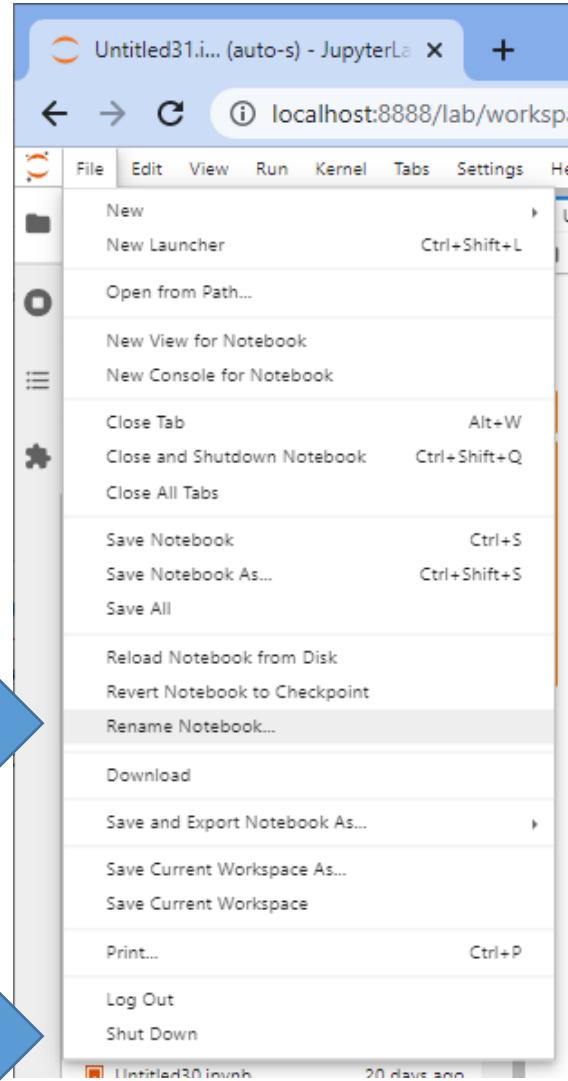
The screenshot shows the Jupyter Lab interface. On the left is a file browser with a sidebar containing icons for file operations like creating (+), deleting, and moving files. A search bar labeled "Filter files by name" is also present. The main area displays a notebook titled "Untitled31.ipynb". In the first cell, the command `print("Hello world!")` is run, outputting "Hello world!". In the third cell, the command `print?` is run, which displays the function's docstring:

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

A blue callout box points from the question mark "?" to the word "Docstring" in the docstring output.

?
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>

