jupyter lab

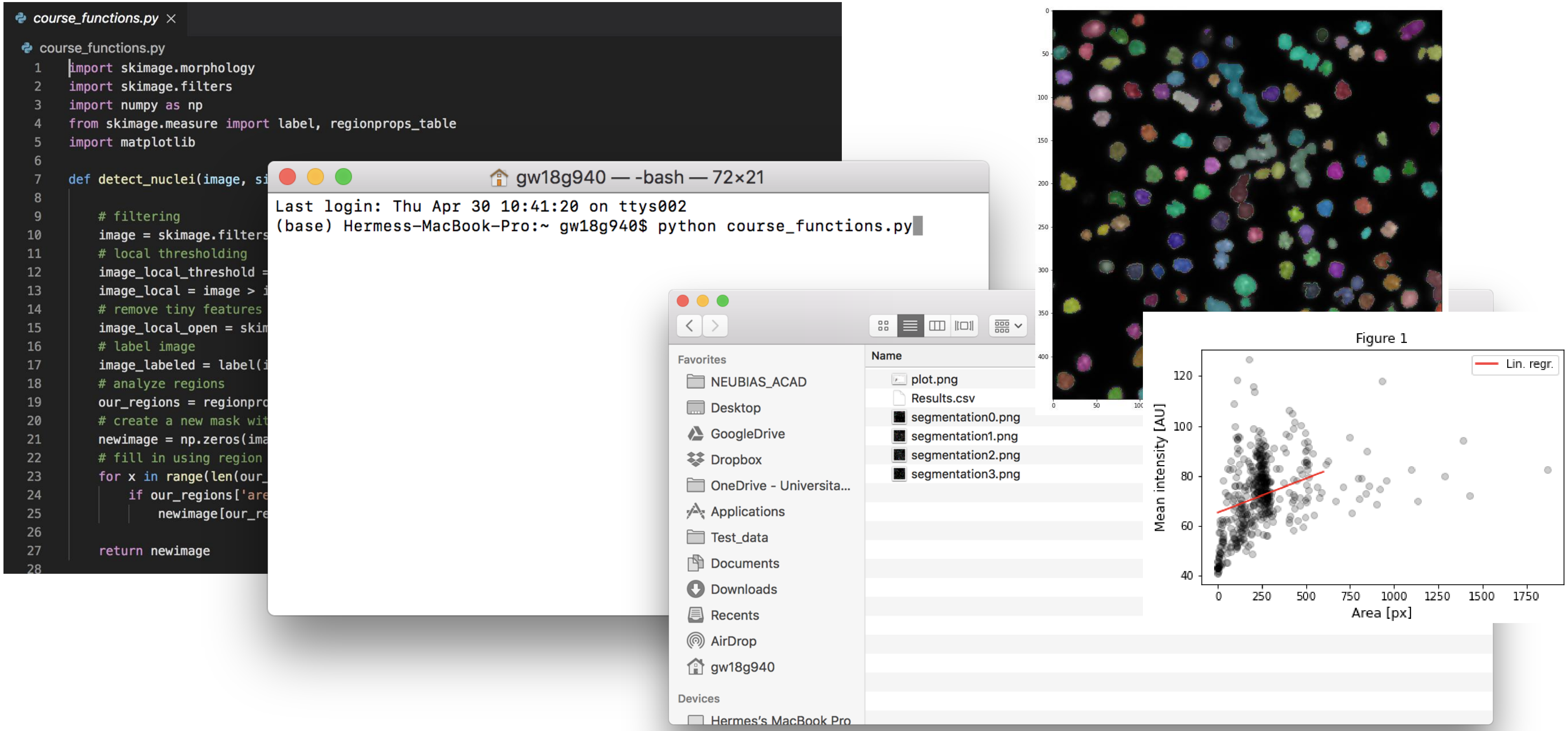
Marcelo Leomil Zoccoler

With materials from Robert Haase (Pol TU Dresden) and Guillaume Witz (Universität Bern)

August 2023

@zoccolermarcelo

“Classic” software vs. notebooks



“Classic” software vs. notebooks

Code divided in parts

Dynamic: easy to test

Rich output: processing + visualisation + analysis in one place

Code + formatted text: easy documentation

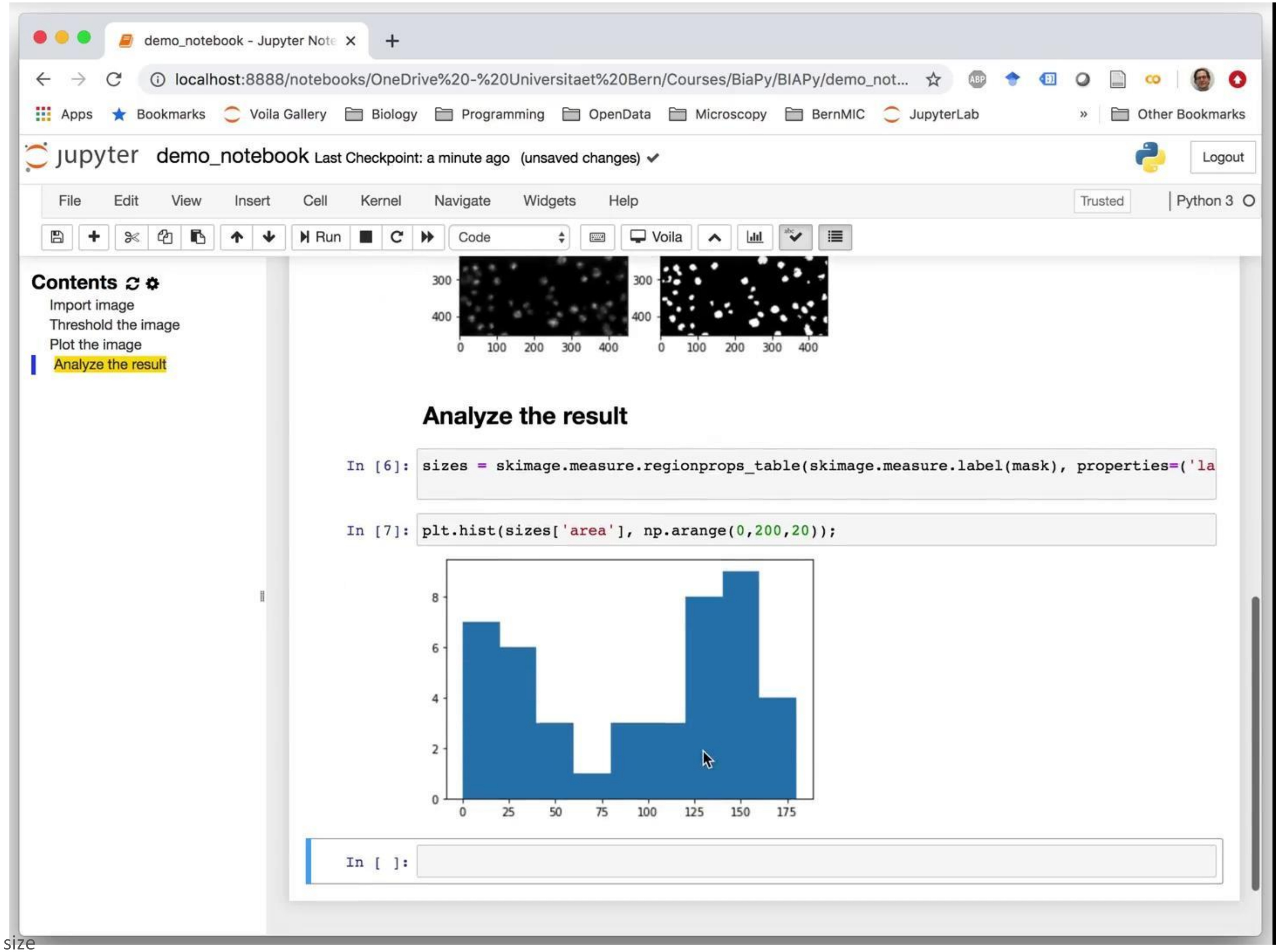


Illustration: load an image, threshold it, analyze object size

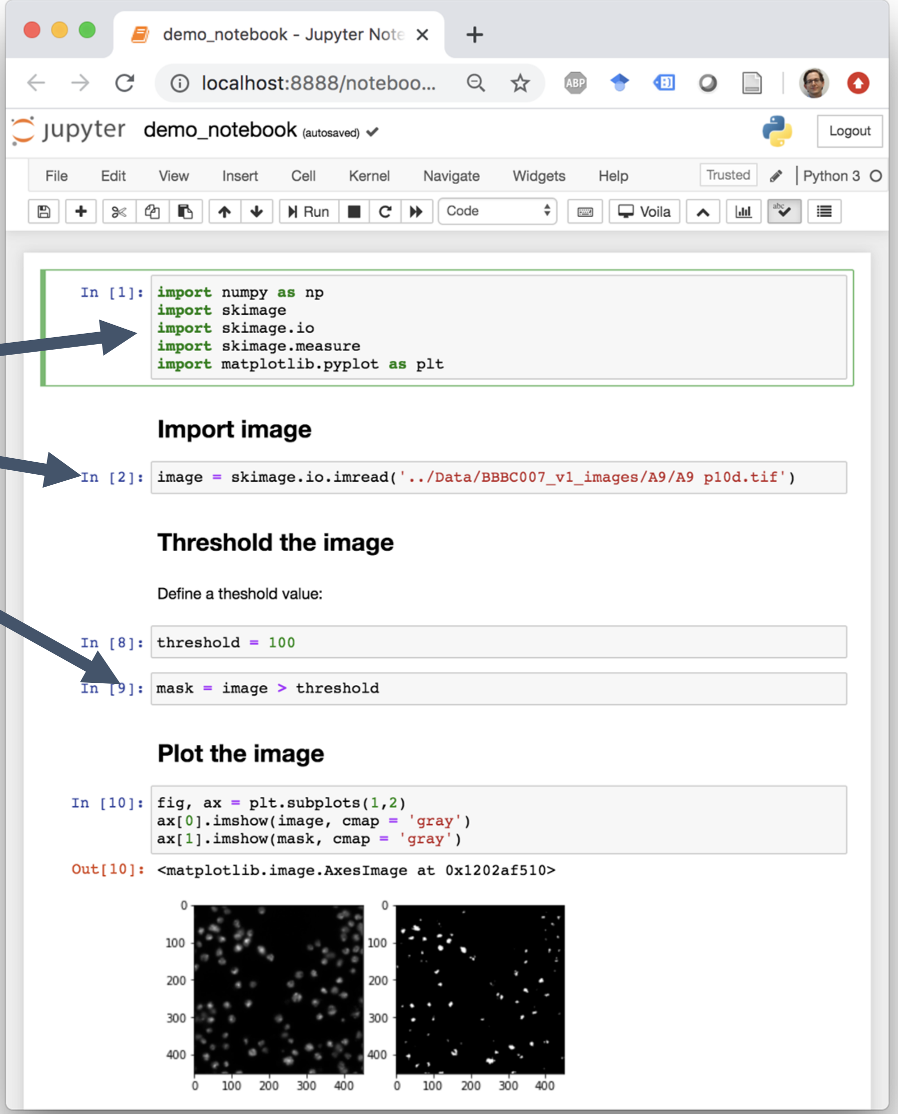
@zoccolermarcelo

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?

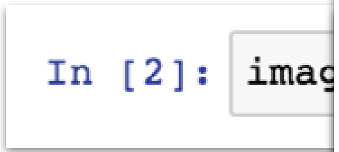
A text file (easily sent around)

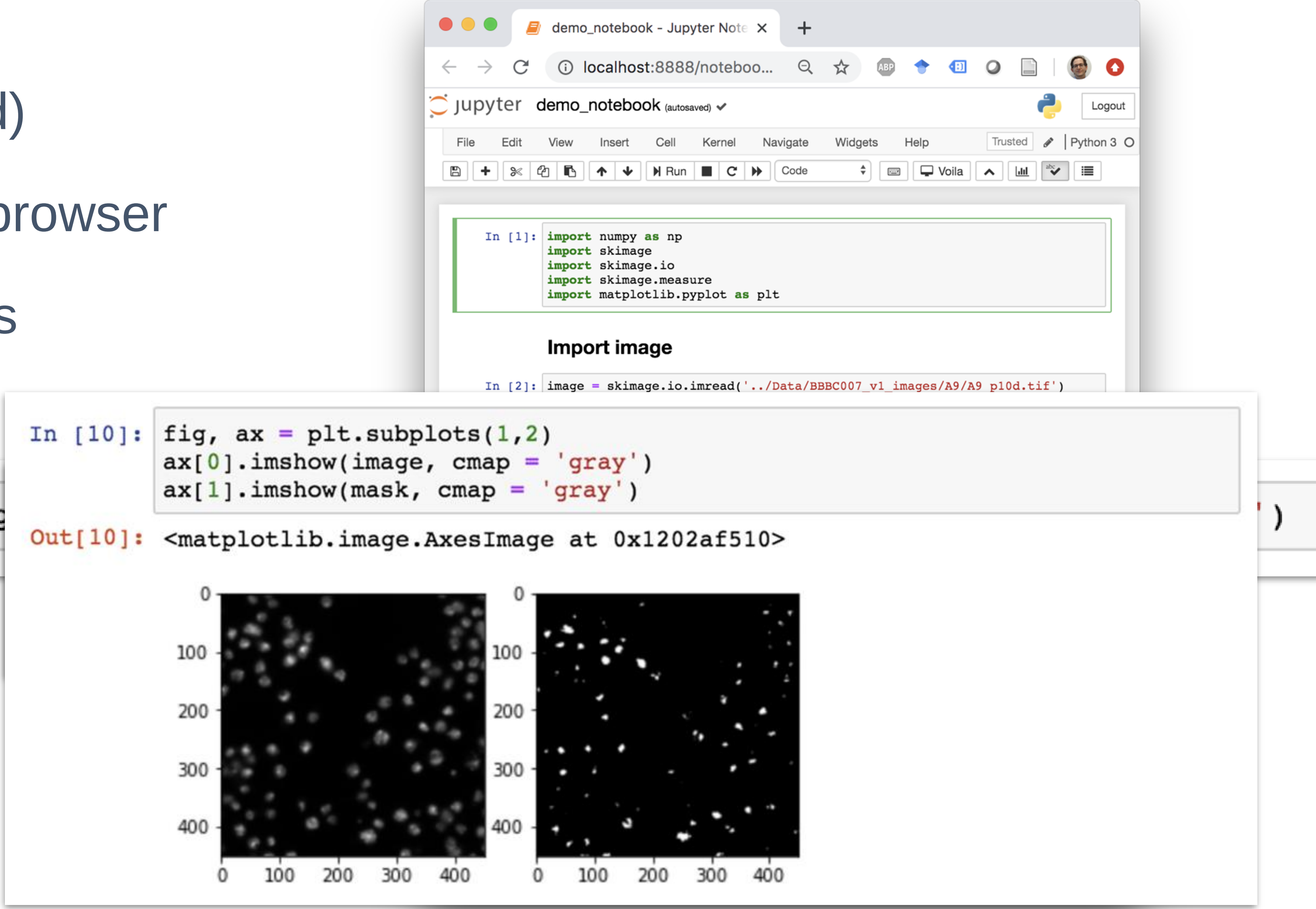
Rendered by Jupyter in the browser

Split into sections called cells

Cells can contain:

- Code
- Formatted text
- Rich output



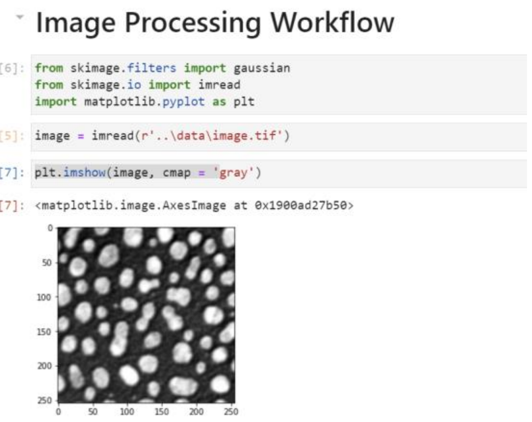


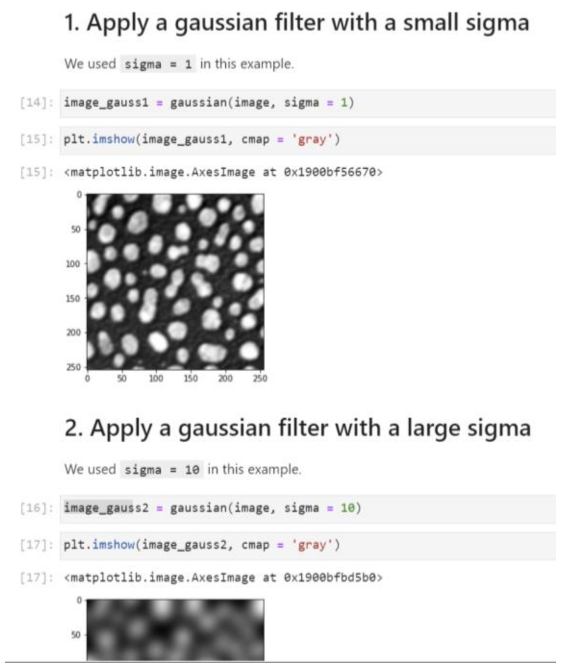
@zoccolermarcelo

Why using notebooks?

- Documenting (for your-(future)-self and for others) and enhanced reproducibility

"First, we applied a Gaussian filter from scikit-image with sigma = 1. Then, we applied another Gaussian filter to the original image with sigma = 10. After that, we subtracted the first result from the second...

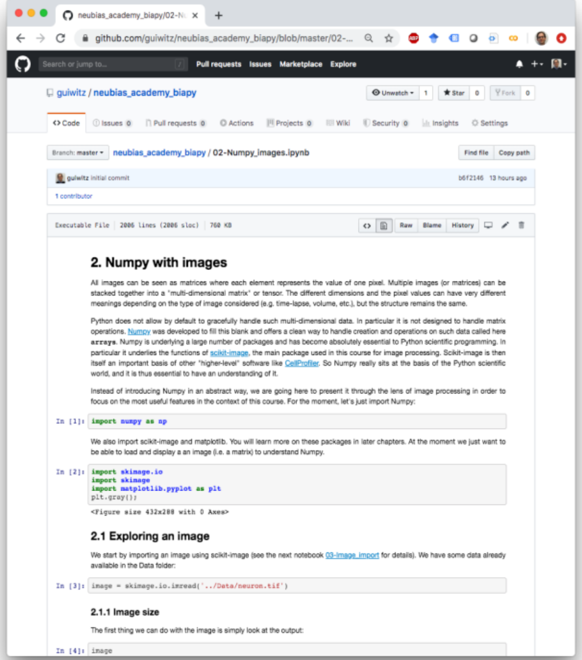




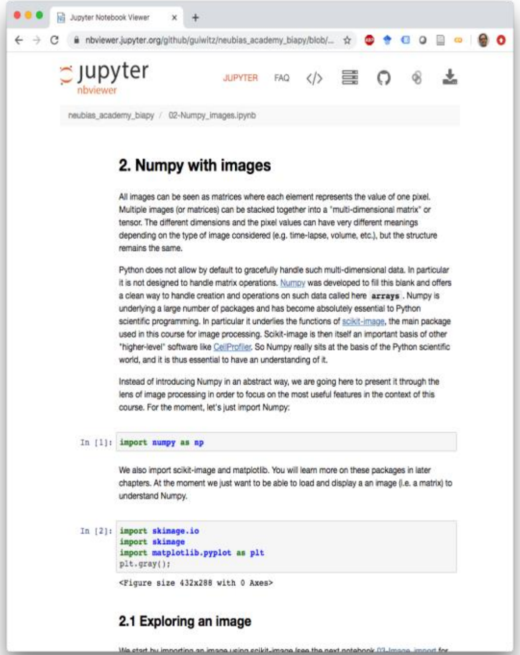
Why using notebooks?

PoL Physics of Life TU Dresden

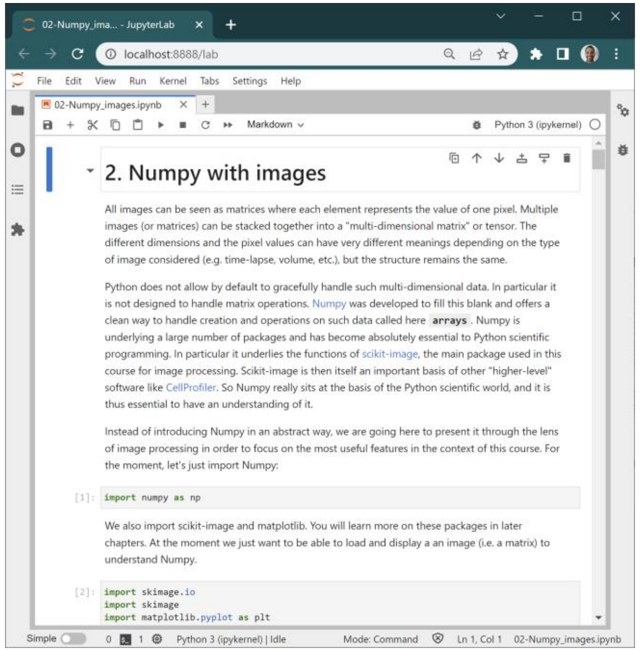
- Sharing
- - Send a single file with code, intermediary outputs and rich text explanations



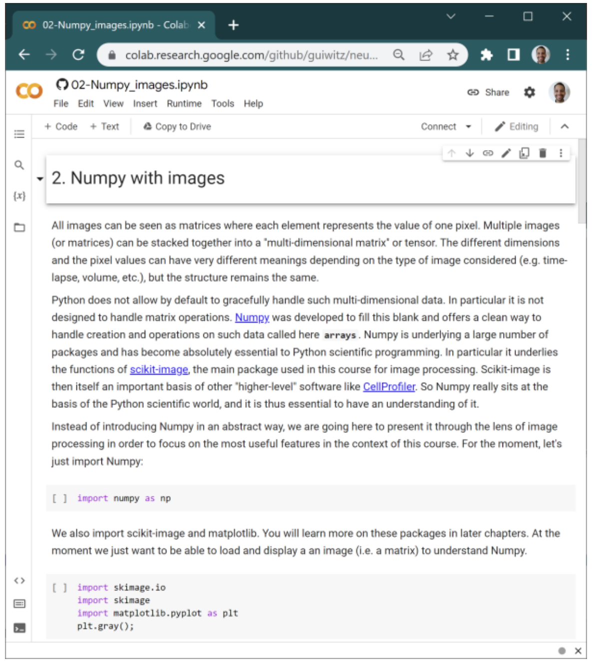
Github rendering



Nbviewer rendering



Jupyter lab (local or Binder)

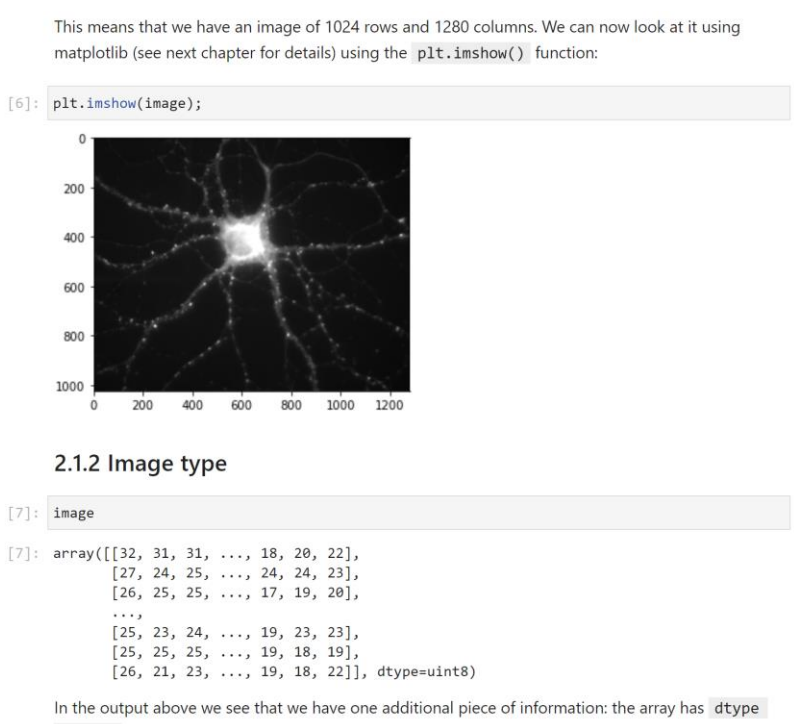


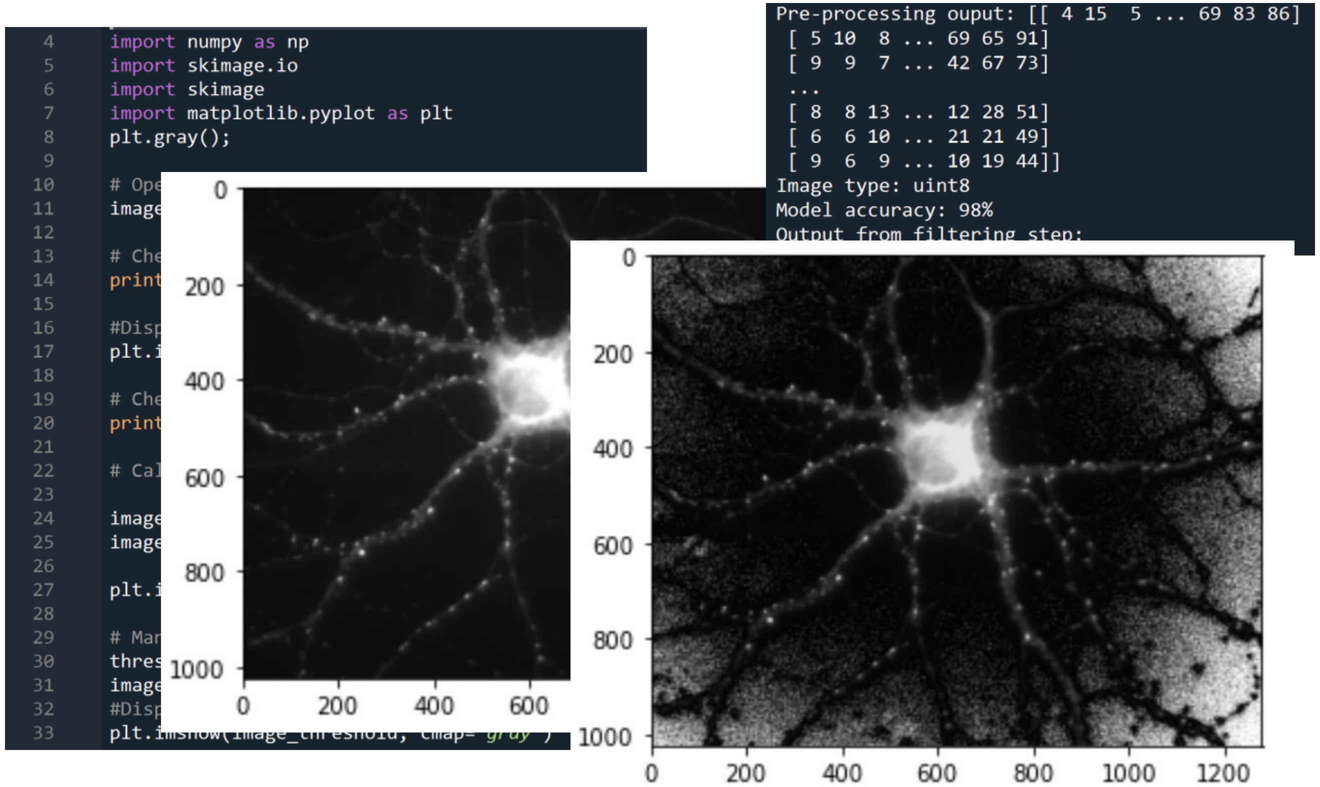
Google Colab

Why using notebooks?

- Teaching

- Small blocks of code with intermediary results are easier to understand than scripts that spit tons of outputs in sequence





Why using notebooks?

- Keep all the benefits from using code:

  - Batch processing

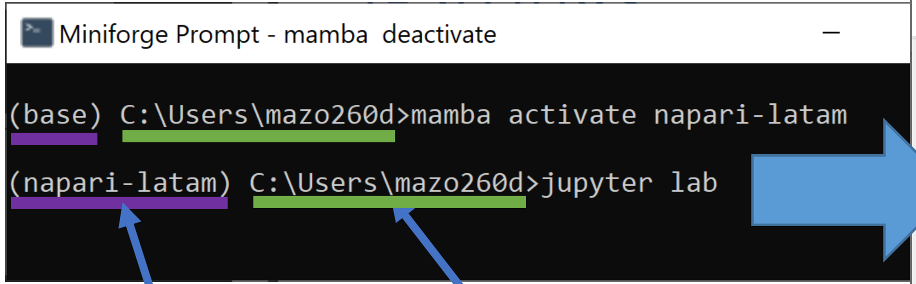
  - Running python functions/tools still unavailable as plugins

Why using notebooks with napari?

- Easy data interaction and visualization with napari:
    - Great for visualizing 3D (and more) data
    - Each processing step result can be displayed as a separate layer
- Data annotation

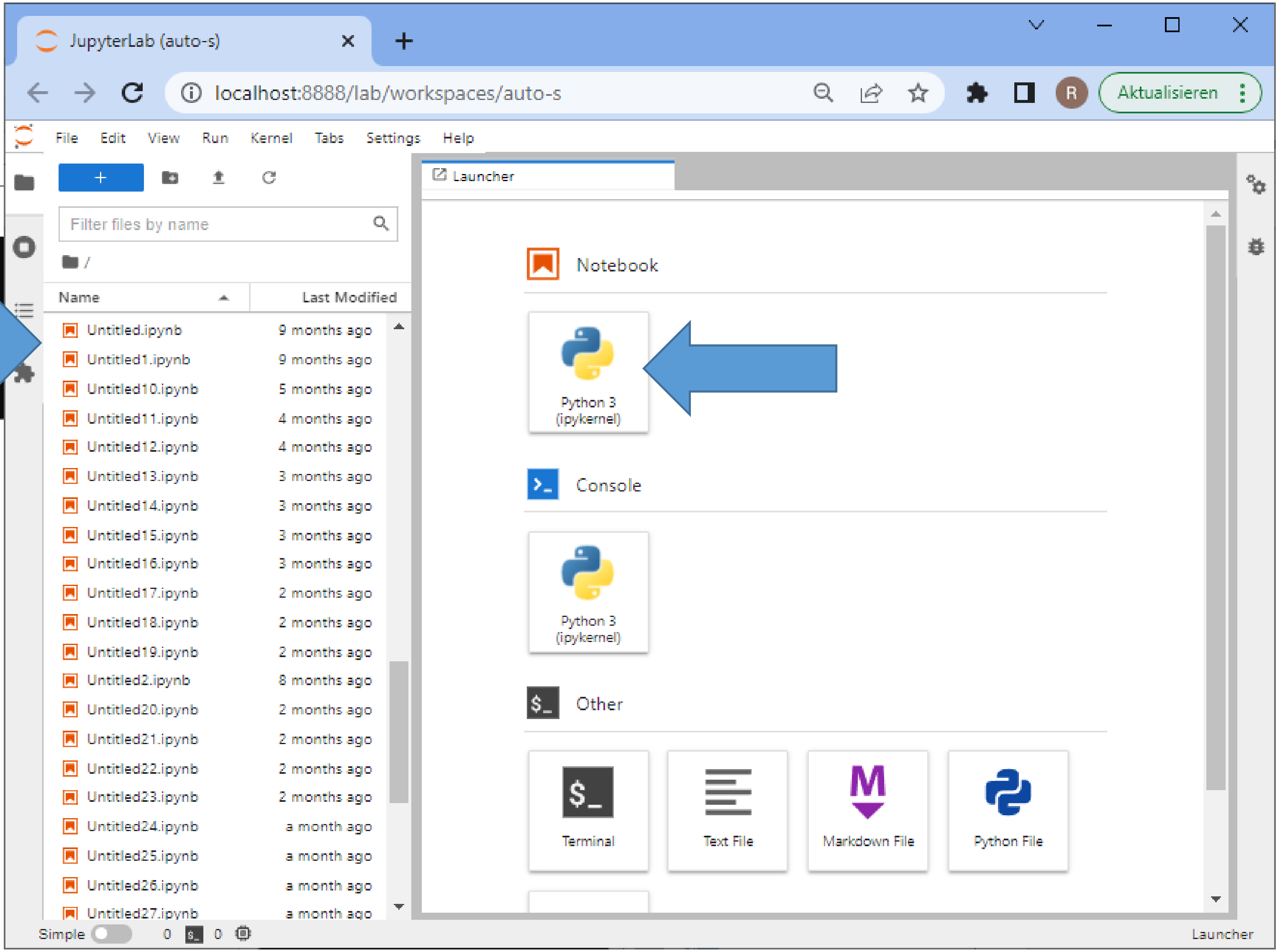
Jupyter lab

- Our programming environment for this course



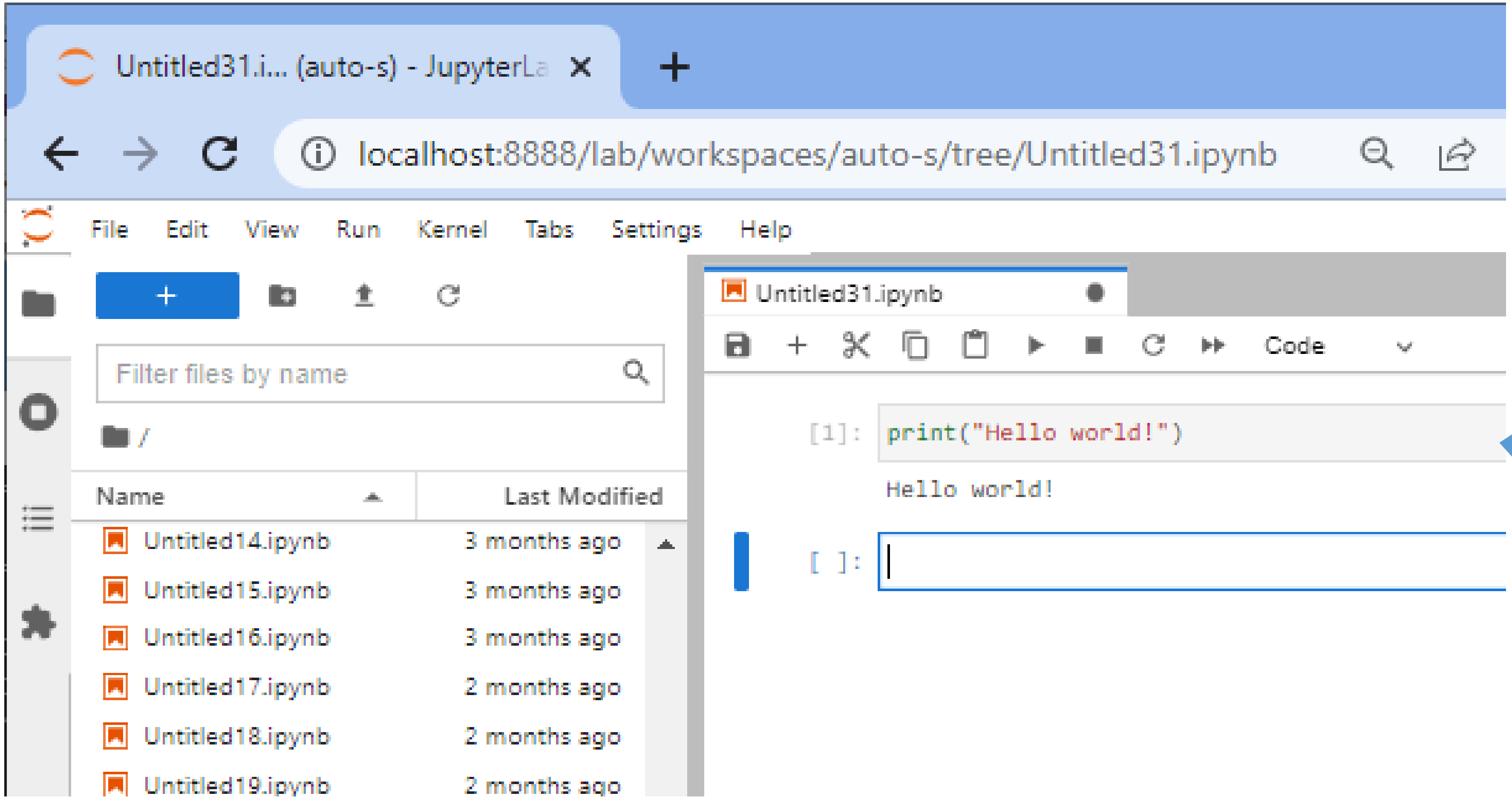
Current environment

Current path



Jupyter lab

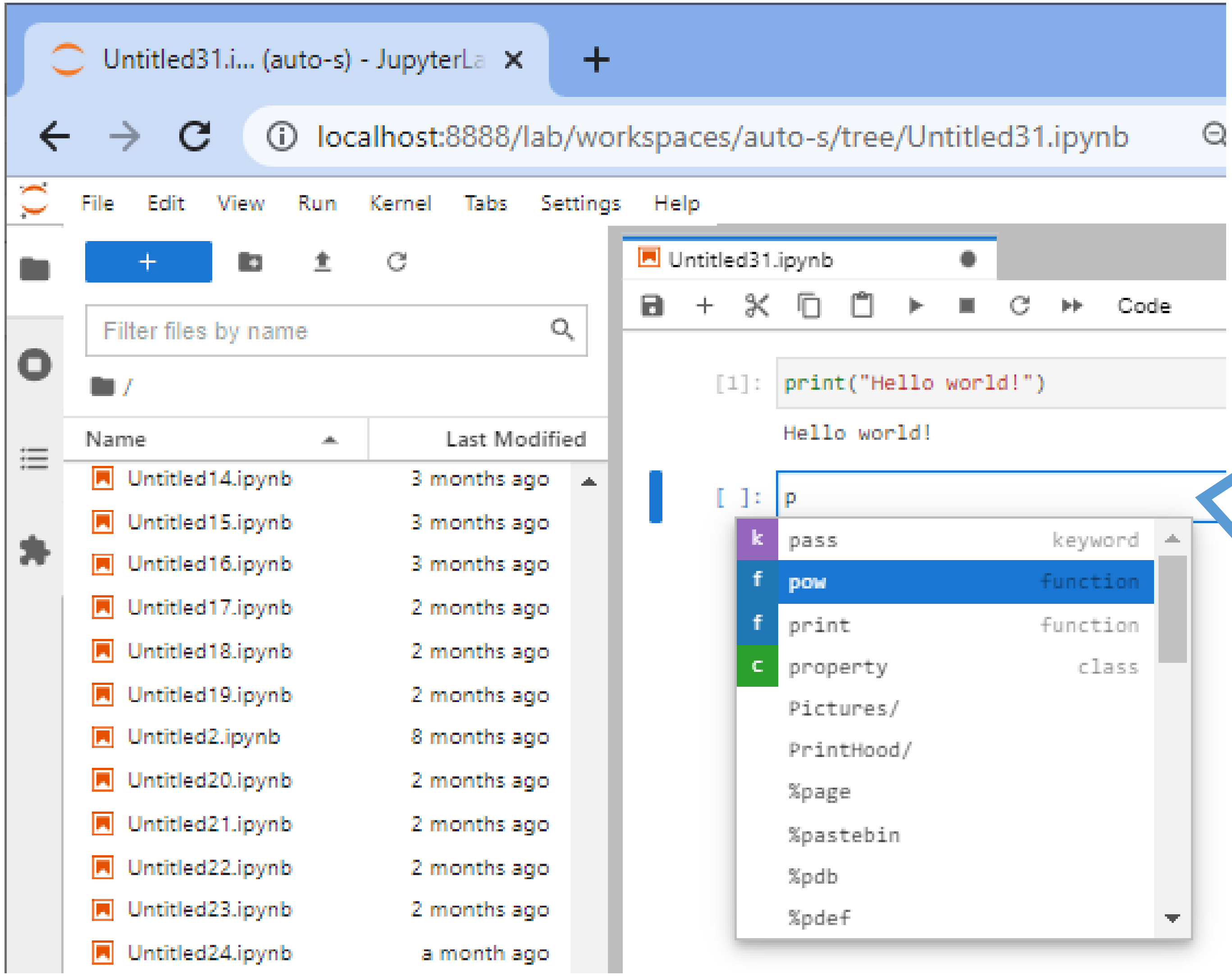
- Execute code cell-by-cell and see results instantaneously



SHIFT + ENTER to execute a code cell

Jupyter lab

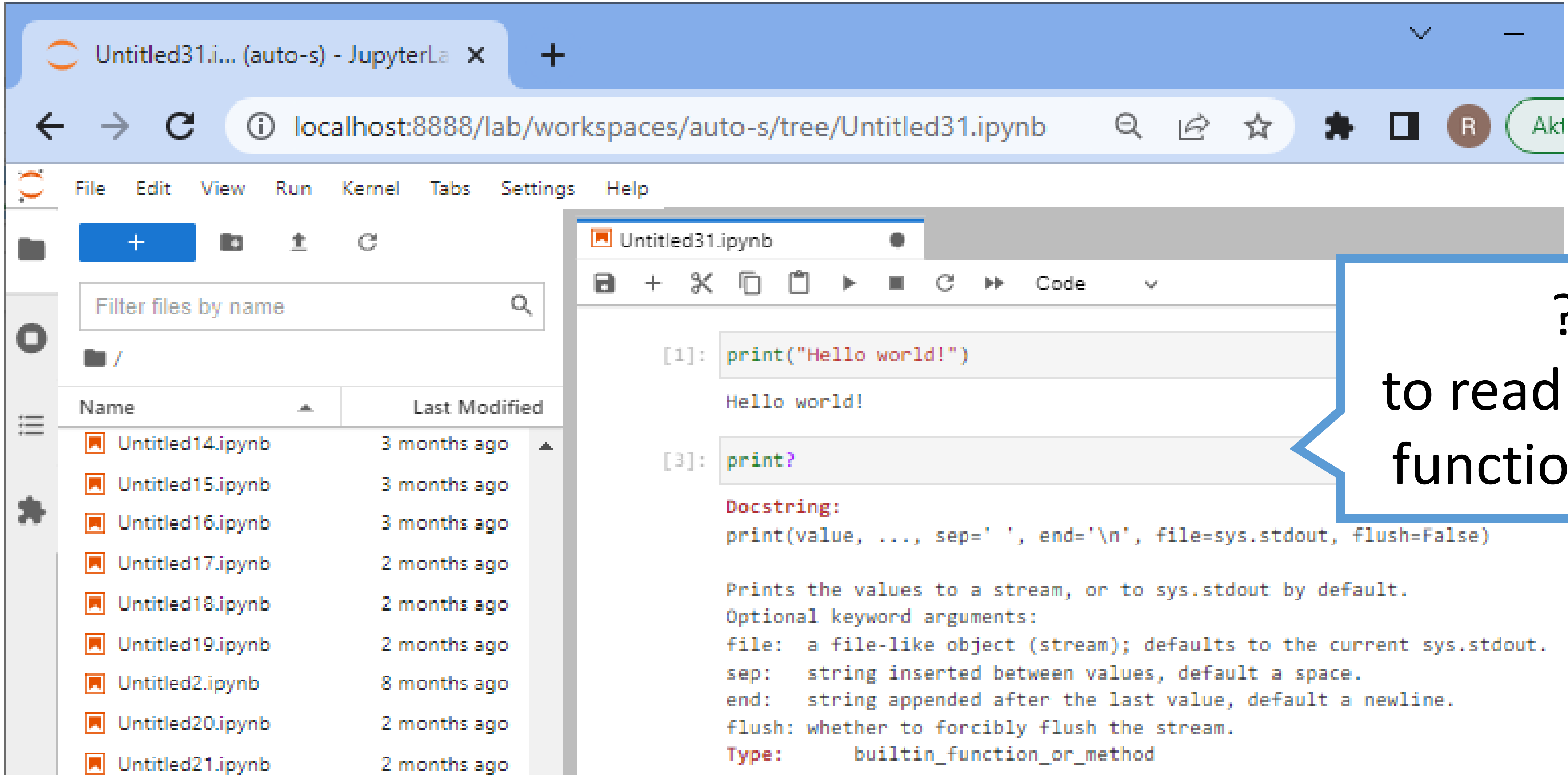
- Context-specific help, auto-completion



TAB
to open auto-completion

Jupyter lab

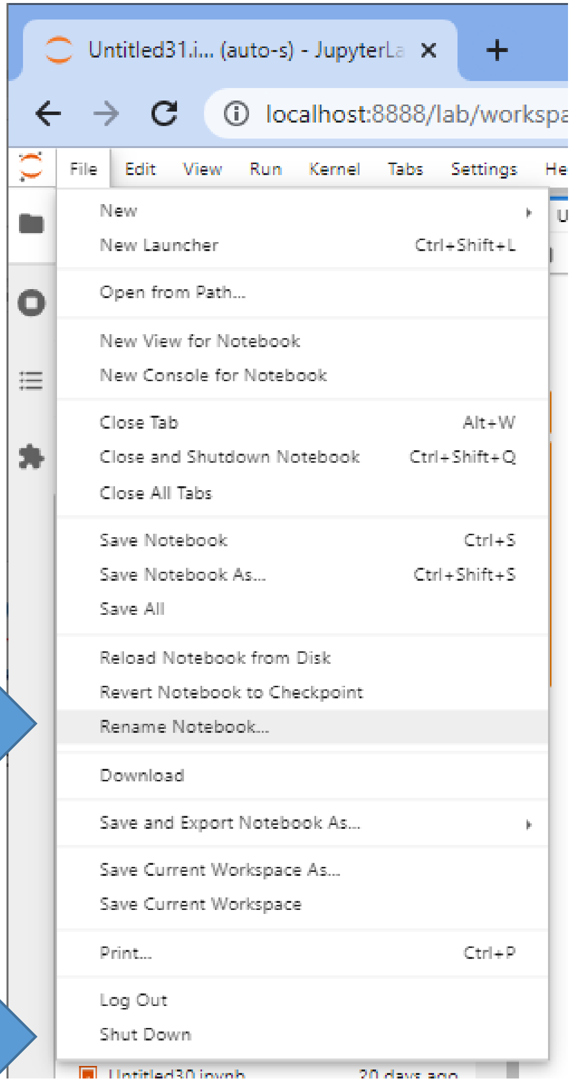
- Help / “docstrings”

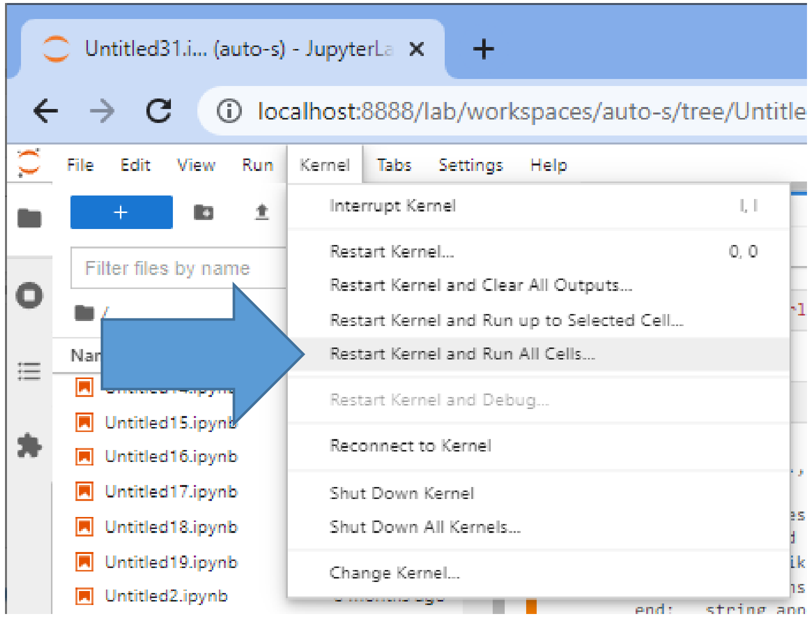


?
to read what a function does

Jupyter lab

- Saving / renaming / closing





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

Python programming basics

Marcelo Leomil Zoccoler

With materials from Robert Haase

August 2023

@zoccolermarcelo

Working with variables

- Variables can hold numeric values and you can do math with them

```
# initialize program
a = 5
b = 3

# run algorithm on given parameters
sum = a + b

# print out result
print (sum)
```

```
8
```

Mathematical operations

- Math commands supplement operators to be able to implement any form of calculations

- Power

```
pow(3, 2)
```

```
]: 9
```

- Absolute

```
abs(-8)
```

```
]: 8
```

- Rounding

```
round(4.6)
```

```
]: 5
```

Be careful with some of them!

```
round(4.5)
```

```
]: 4
```

https://en.wikipedia.org/wiki/Rounding#Round_half_to_even

Comments

Comments should contain additional information such as

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

```
#
# This program sums up two numbers.
#
# Usage:
# * Run it in Python 3.8
#
# Author: Robert Haase, PoL TUD
#         Robert.haase@tu-dresden.de
# April 2021

# initialise program
a = 1
b = 2.5

# run complicated algorithm
final_result = a + b

# print the final result
print( final_result )
```

Working with variables and string values

- Also strings as values for variables are supported

Single and double quotes allowed

```
firstname = "Robert"
lastname = 'Haase'

print("Hello " + firstname + " " + lastname)
```

```
Hello Robert Haase
```

f-Strings

- String formatting is made easy using f-strings.

```
f"This is an f-string. a's value is {a}. Doubling the value of a gives {2*a}."
```

```
"This is an f-string. a's value is 5. Doubling the value of a gives 10."
```

- Using f-strings, you can also call code from within a string. Take care of code readability!

```
f"The first_name variable contains {first_name.lower().count('r')} r letters."
```

```
'The first_name variable contains 2 r letters.'
```

Working with variables and string values

- Also strings as values for variables are supported
- When combining strings and numbers, you need to explicitly define what you want to do.

```
# mixing types

a = 5
b = "2"

print (a + b)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-4-51629e6a285f> in <module>
      4 b = "2"
      5
----> 6 print (a + b)

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

```
# mixing types

a = "5"
b = 2

print (a + b)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-5-85ae49867097> in <module>
      4 b = 2
      5
----> 6 print (a + b)

TypeError: can only concatenate str (not "int") to str
```

```
# mixing types to make numbers

a = 5
b = "2"

print (a + int(b))
```

```
7
```

```
# mixing types to make strings

a = "5"
b = 2

print (a + str(b))
```

```
52
```

- Conversion to a floating point number: float()