PoL Physics of Life TU Dresden

CENTER FOR SYSTEMS BIOLOGY DRESDEN

#OpenScience

#OpenSource

Good practice in scientific programming

Accessible

Reusable

Sustainable

Robert Haase

April 2023

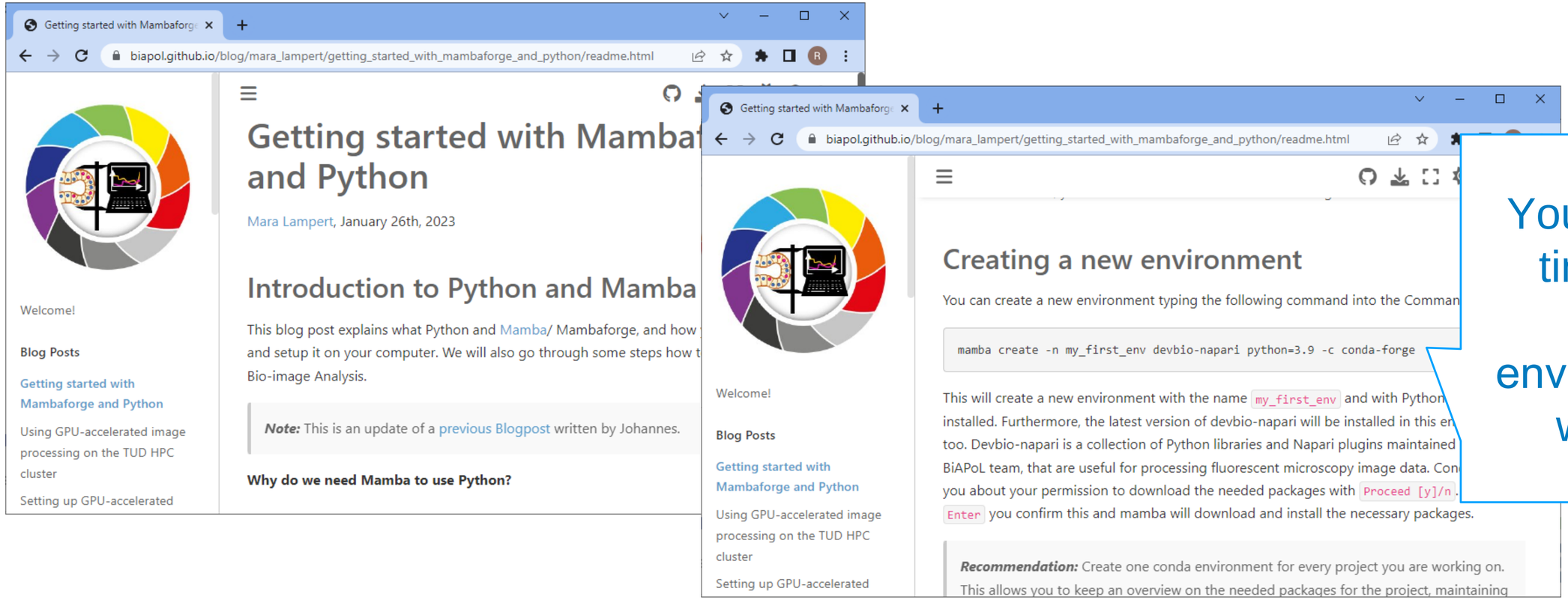
@haesleinhuepf

TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

Document what you use

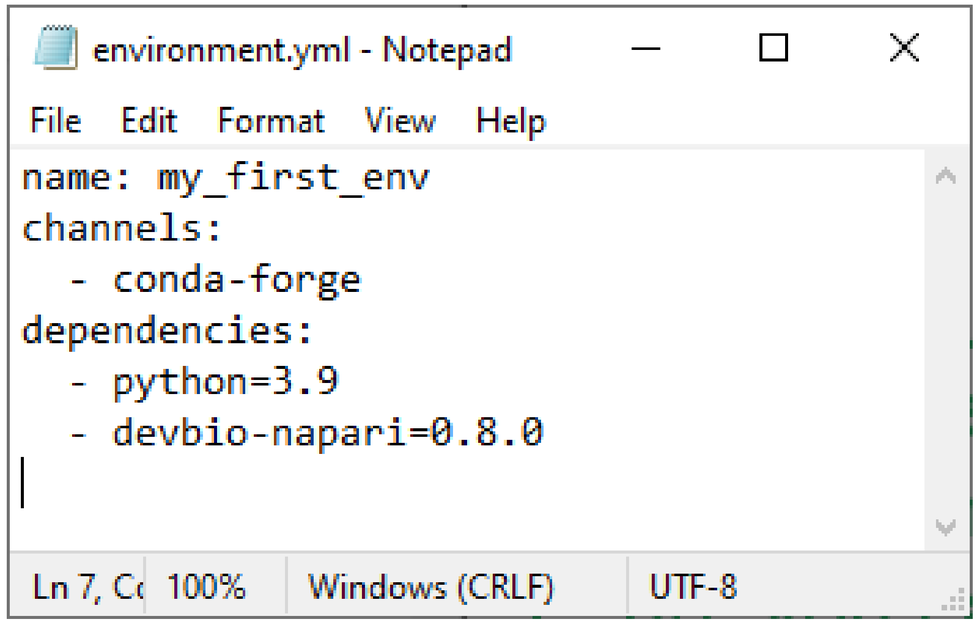
- Installation instructions enable reproducible science
- Not necessary as detailed as a blog-post



You should at any time be able to rebuild the environment you're working with.

Document what you use

Maintain a document with the dependencies (and versions) you need in your project!



In case your environment is screwed up, you can rebuild it any time.

```
conda env create -f environment.yml
```


https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html#creating-an-environment-from-an-environment-yml-file

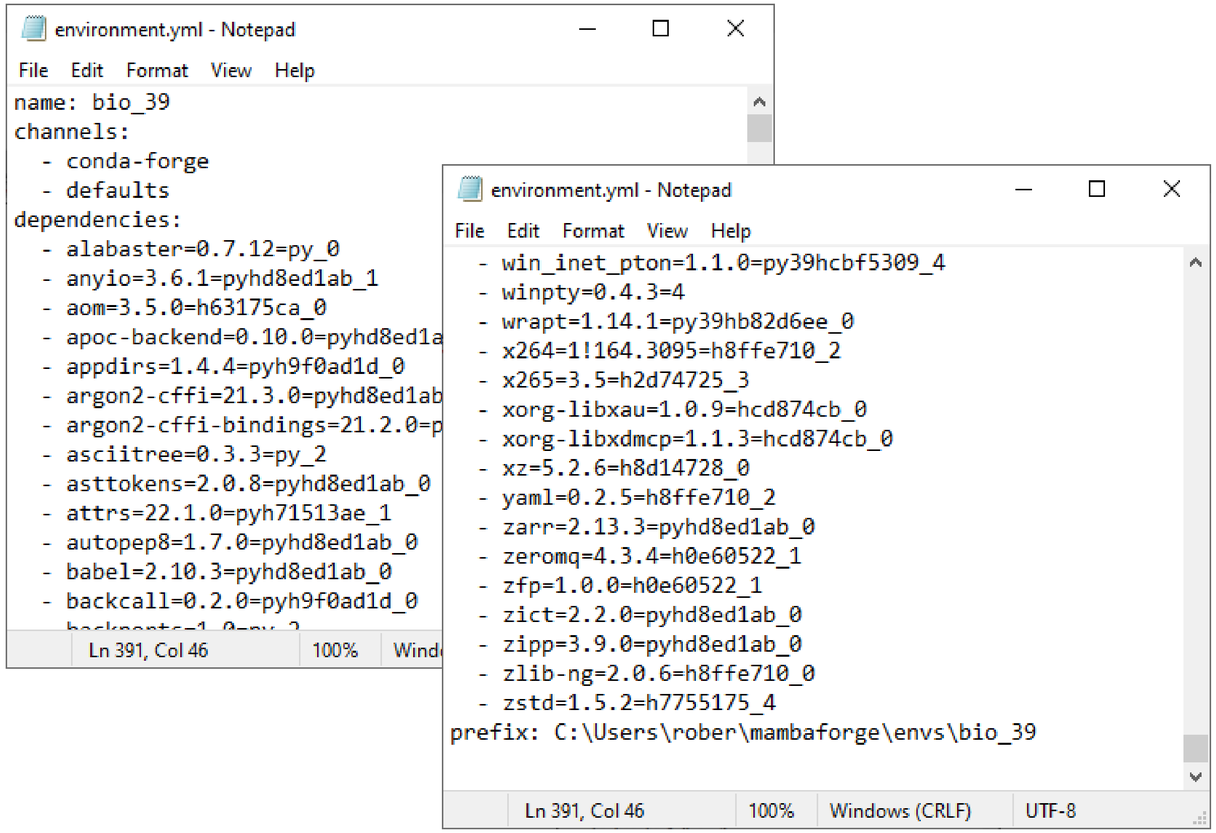
https://pip.pypa.io/en/stable/cli/pip_install/#examples

Document what you use

PoL Physics of Life TU Dresden

... the complete way.

```
conda env export > environment.yml
```



Excellent way to document all dependencies were *actually* used…

It is *questionable* if re-creating an environment from this yml file works.

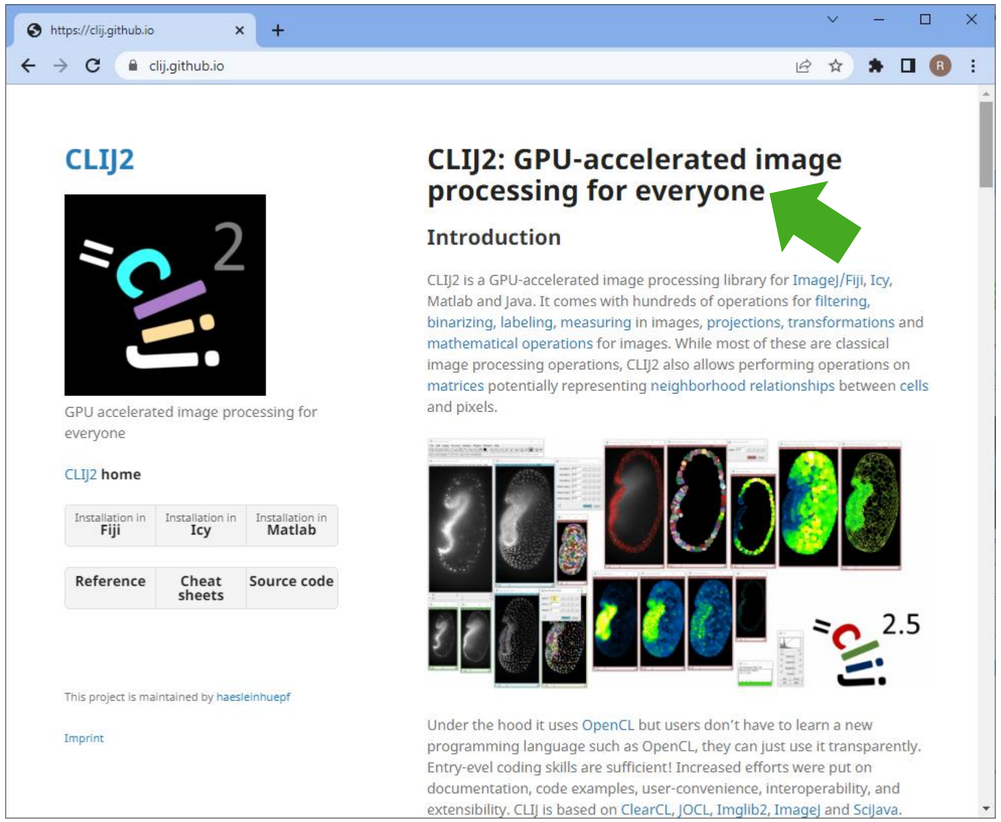
Software quality indicators

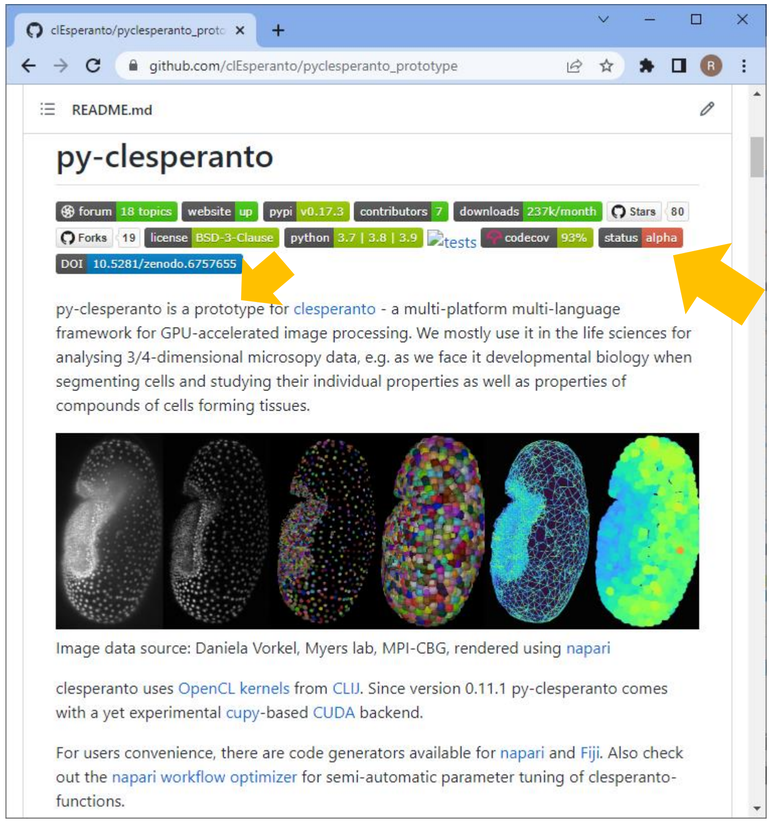
… to differentiate the good stuff from the bad.

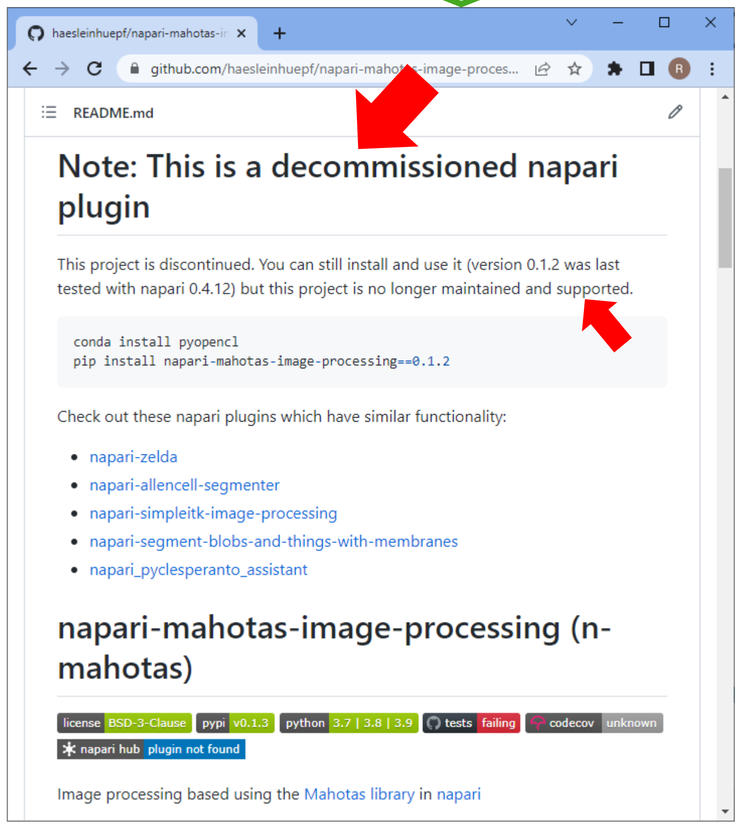
Target audience

- Documentation should tell who is the target audience and how far it is developed

Communication is key!



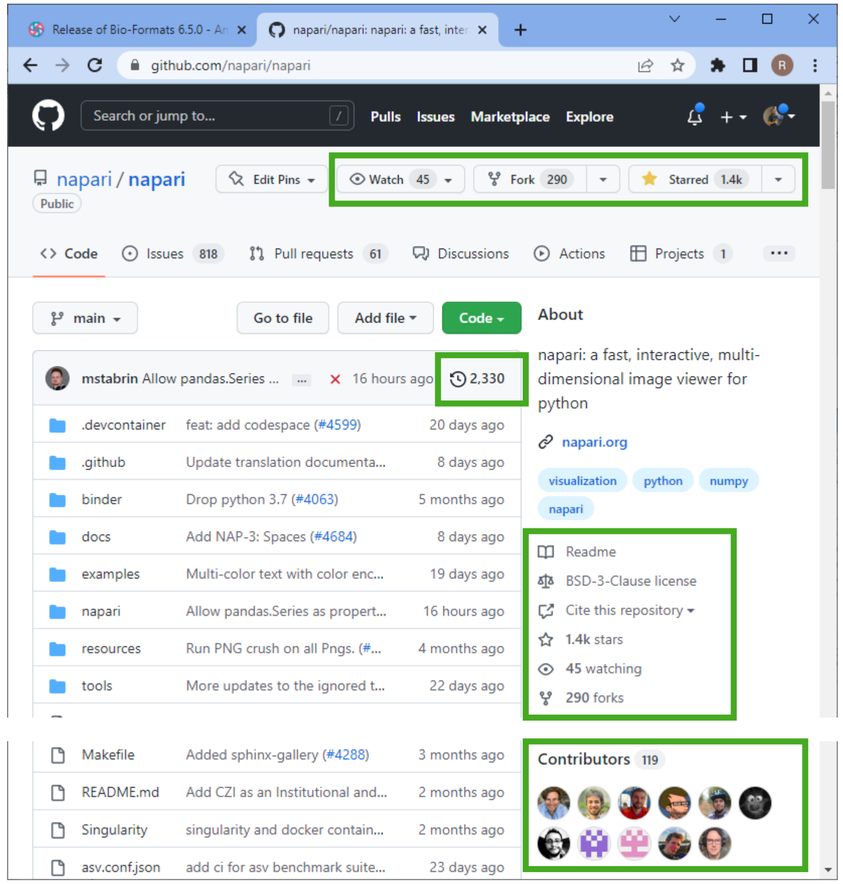




Software quality indicators

PoL Physics of Life TU Dresden

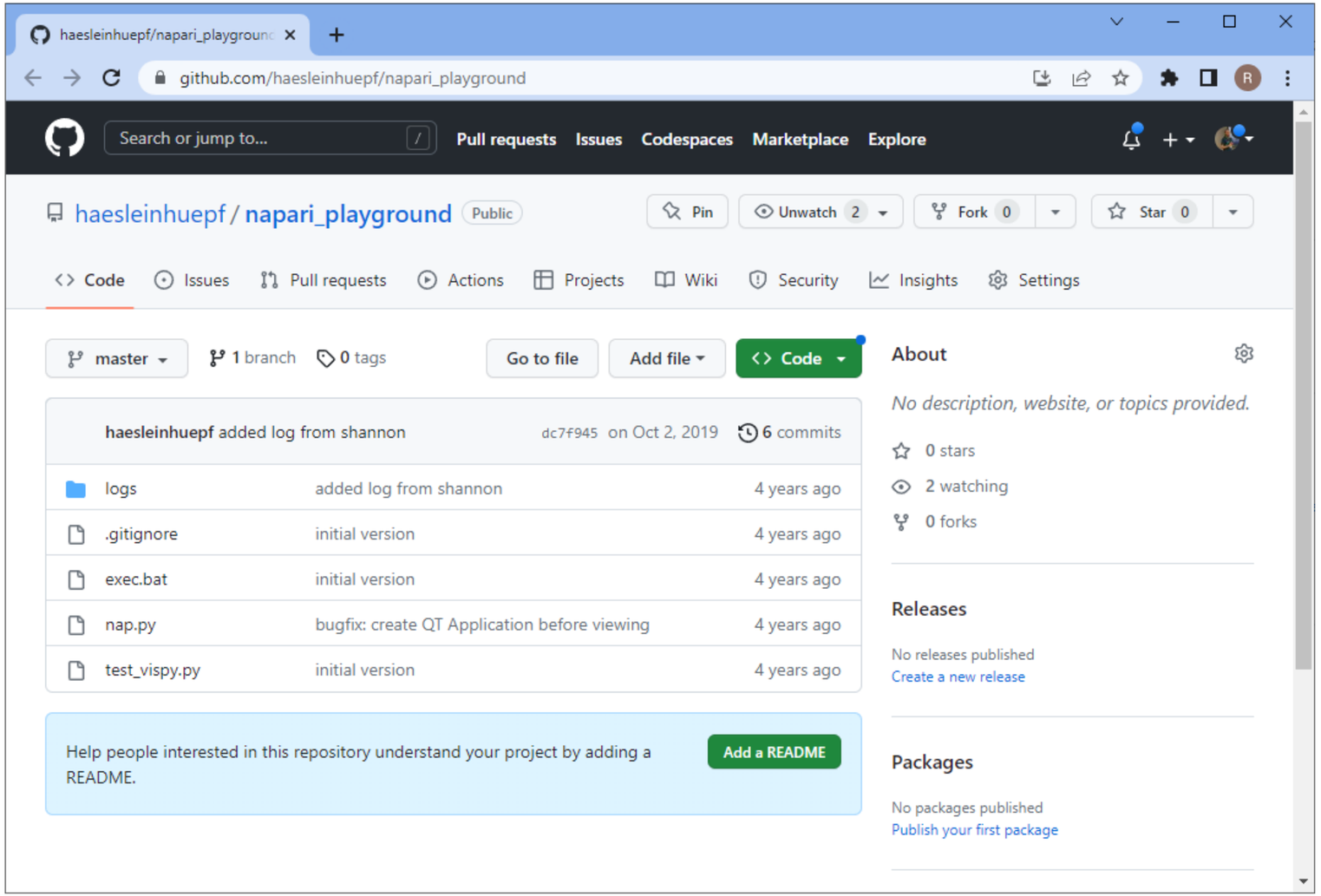
Visit the project's github or gitlab page and review indicators.



- **Stars**: People like software, similarly to tweets on Twitter
- **Watching**: People receive updates for new releases
- **Forks**: People made a copy of the code, e.g. to contribute to the project
- **Contributors**: People who contributed to the code
- **Commits**: Changes to the code

Bad example

- No readme / documentation
- No license / copyright statement
- No stars / users (?)
- Not maintained
  (last update 4 years ago)
- bus factor = 1



Software quality indicators

PoL Physics of Life TU Dresden

Visit the project's github or gitlab page and review indicators.

docs passing | tests passing | coverage 56% | pypi package 2.1.0

downloads/month 11k | python 3 | license BSD-3-Clause | contributors 25 | website up

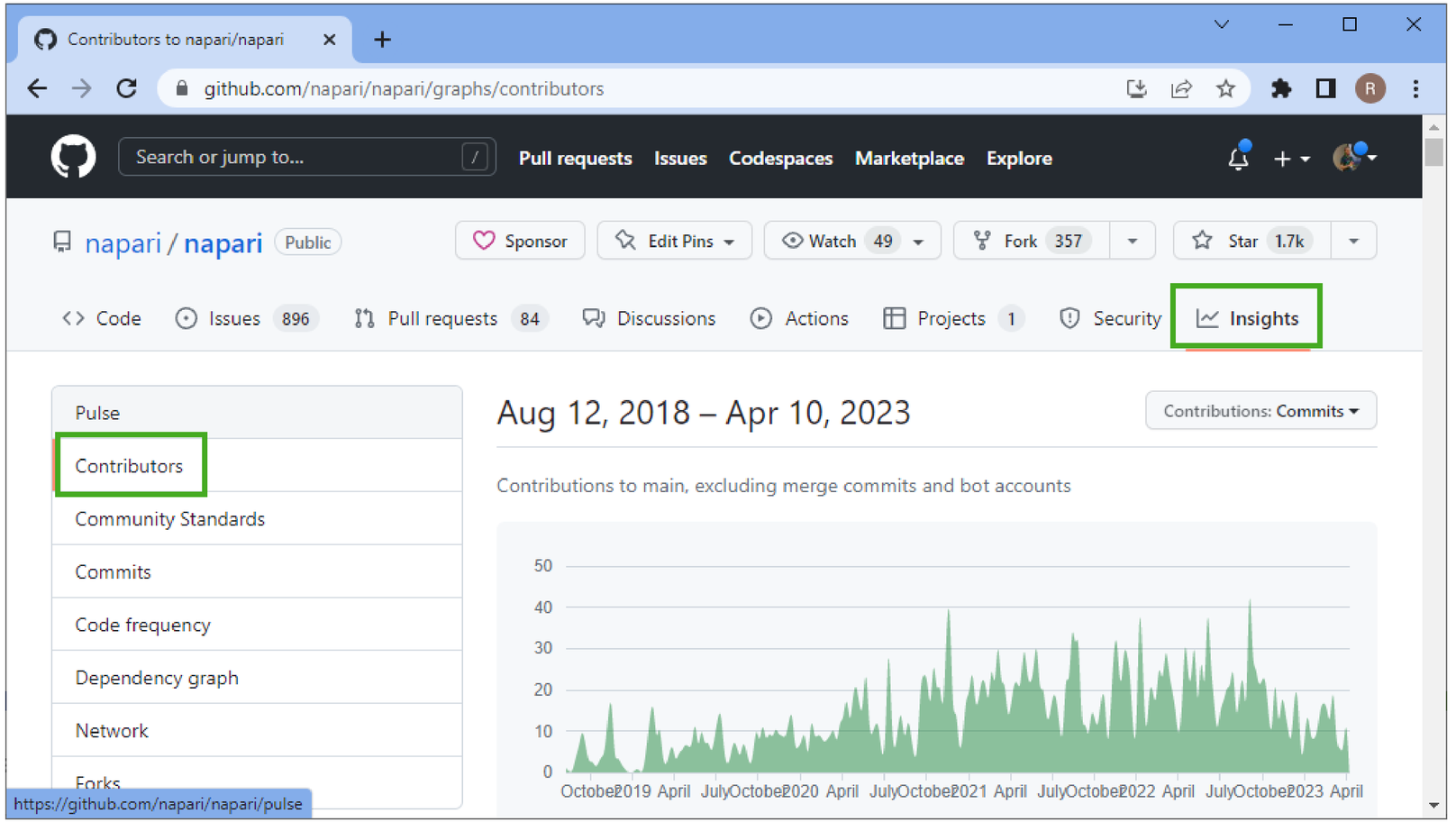
forum 131 topics | repo size 95.2 MB | Stars 641 | Forks 221

Note, github badges cannot be *deserved*. Developers put them there

Image source: Adapted from https://www.pexels.com/photo/shallow-focus-photo-of-two-persons-wearing-military-uniform-2859046/

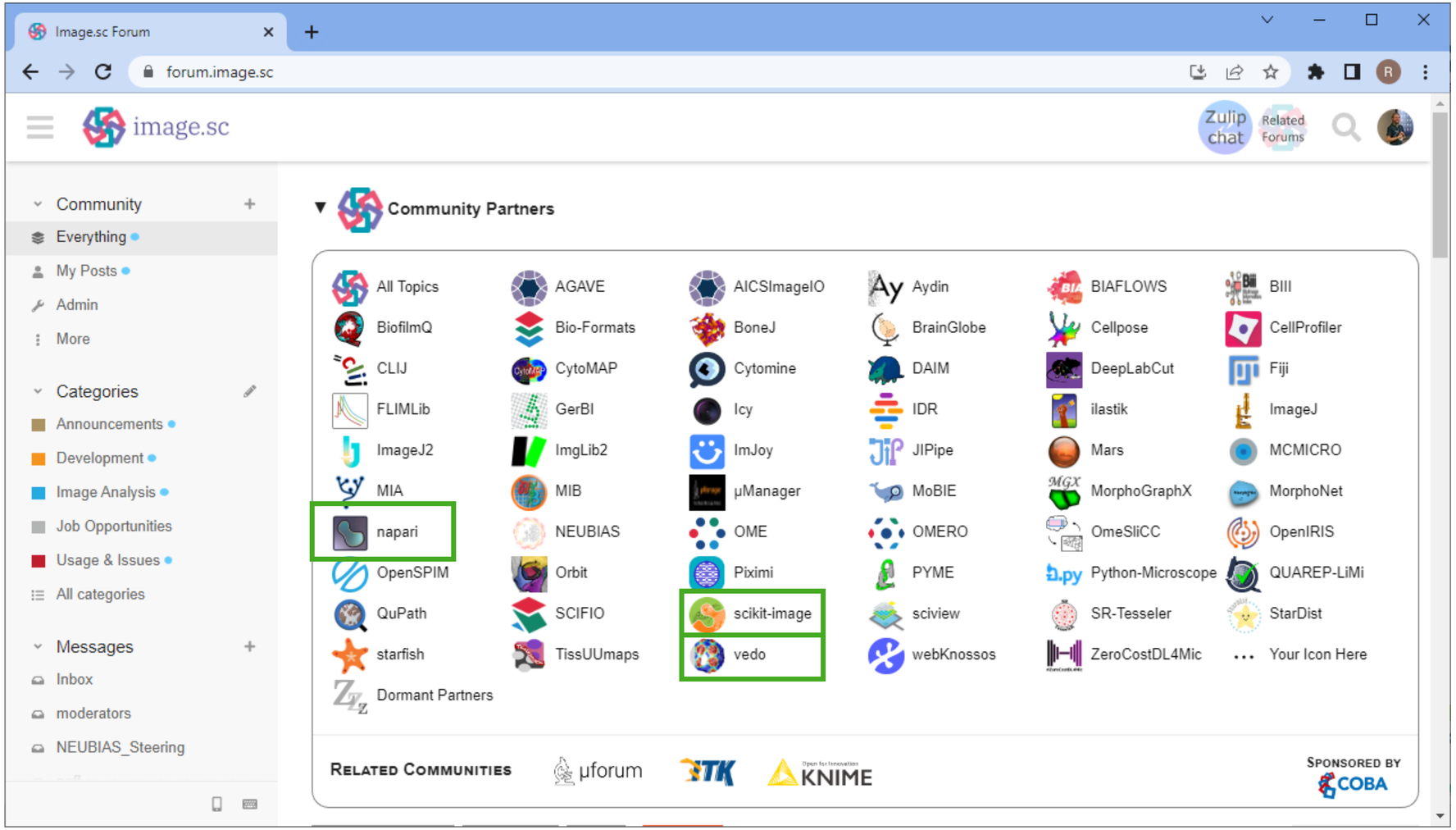
Software quality indicators

Visit the project's github or gitlab page and review indicators.



Software quality indicators

- Community actively involved

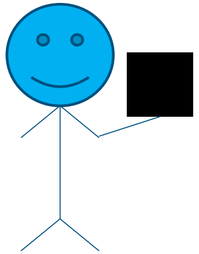


Openness of software / projects

PoL Physics of Life TU Dresden

Choose your project's level wisely, and communicate it clearly

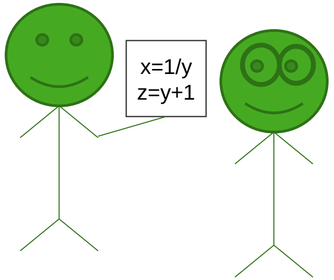
Closed source



- Open to collaborations
- "Black box"
- Compiled code (e.g. C/C++)
- Good for protecting intellectual properties ($$$)

Hardware device drivers

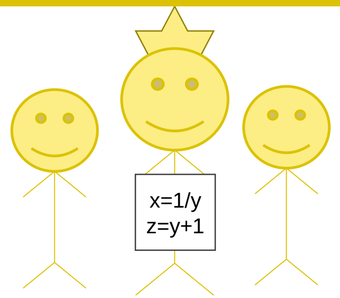
Open source



- Code available to read
- Not necessarily executable code
- No maintenance / support efforts

Custom image analysis scripts

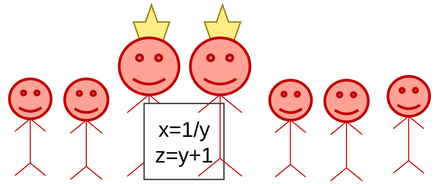
Benevolent dictatorship



- Open to contributions
- Single maintainer, often overwhelmed
- Efficient decision making
- Bus factor ≈1

TrackMate, SNT, MorpholibJ, CLIJ

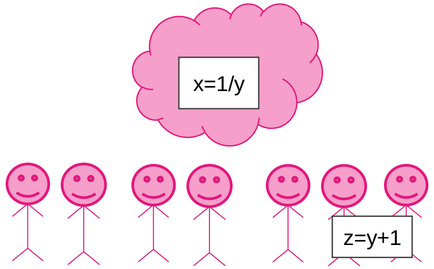
Community driven



- Open to contributions
- Partially democratic
- Board of maintainers (core developers)
- Long-winded decision making

scikit-image, scipy, OpenCL

Openly extensible



- Openly extensible; without maintainers involved
- Partially community driven

ImageJ, Python, numpy

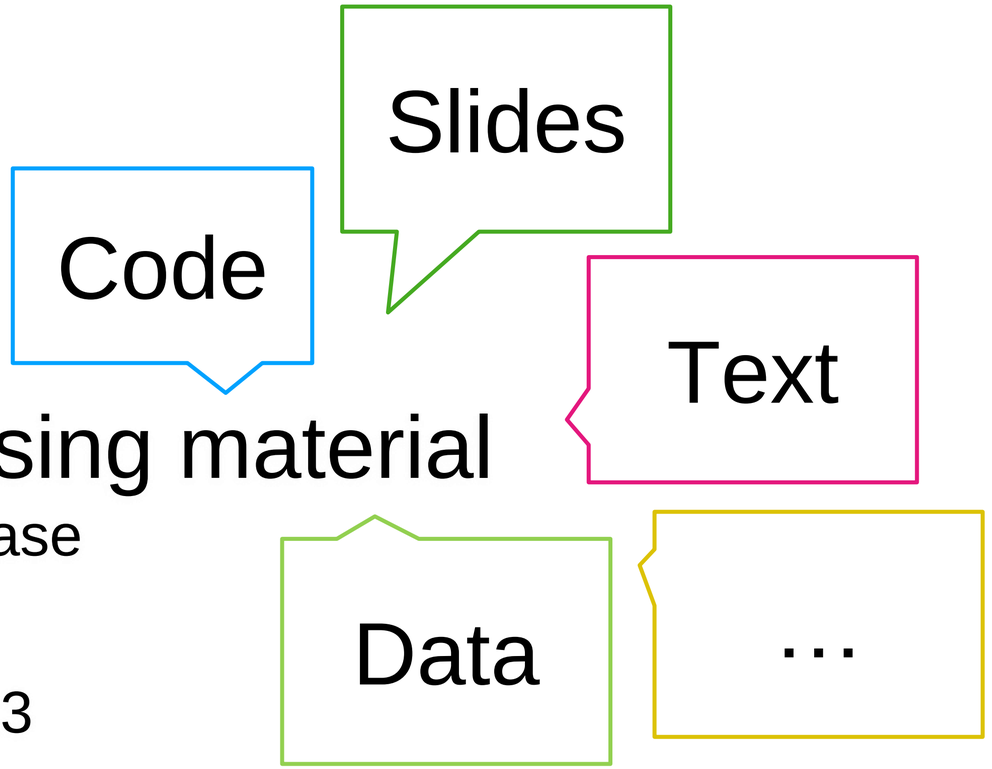
Take home message

When using [open-source] software, make sure

- it’s maintained
- used by others
- supported by an active community
- well-documented

PoL Physics of Life TU Dresden

CENTER FOR SYSTEMS BIOLOGY DRESDEN



Sharing and licensing material

Robert Haase

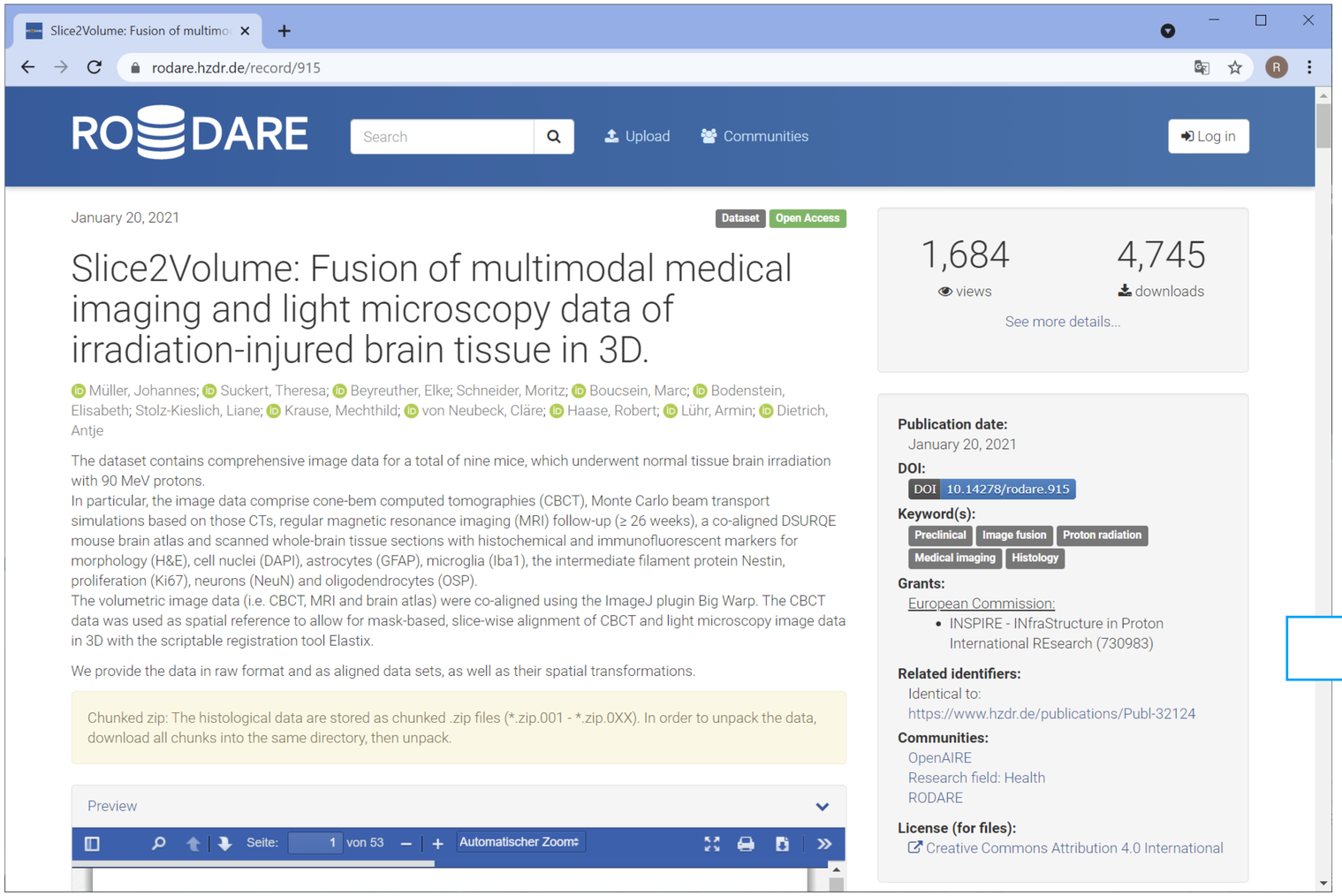
April 2023

@haesleinhuepf

TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

Use cases: Data



Unique datasets

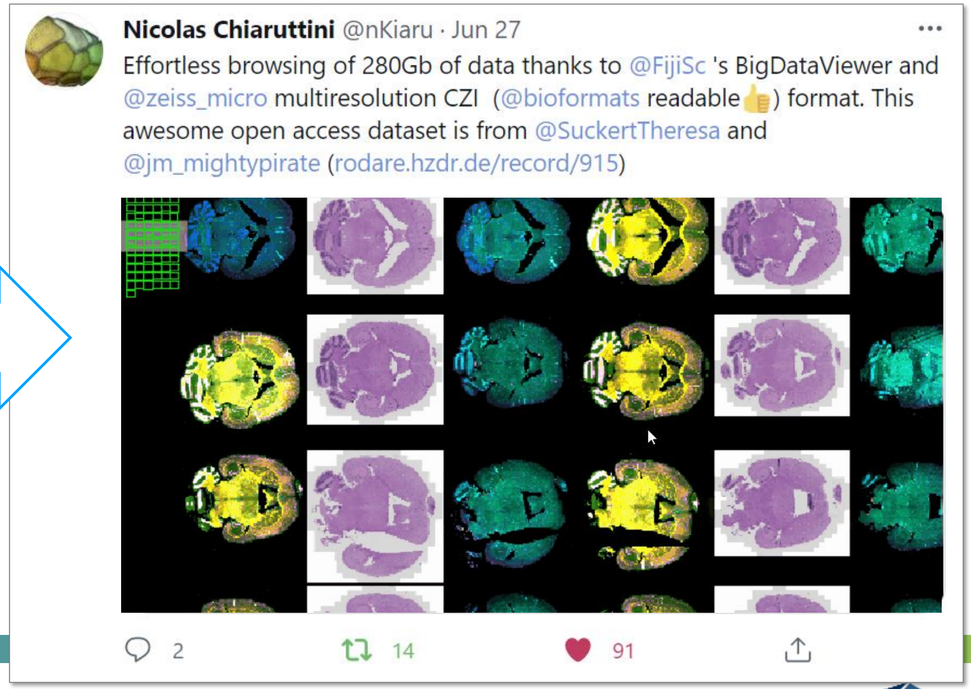
Valuable for biologists

Valuable for software developers

Institutional servers / services

https://idr.openmicroscopy.org/

https://zenodo.org



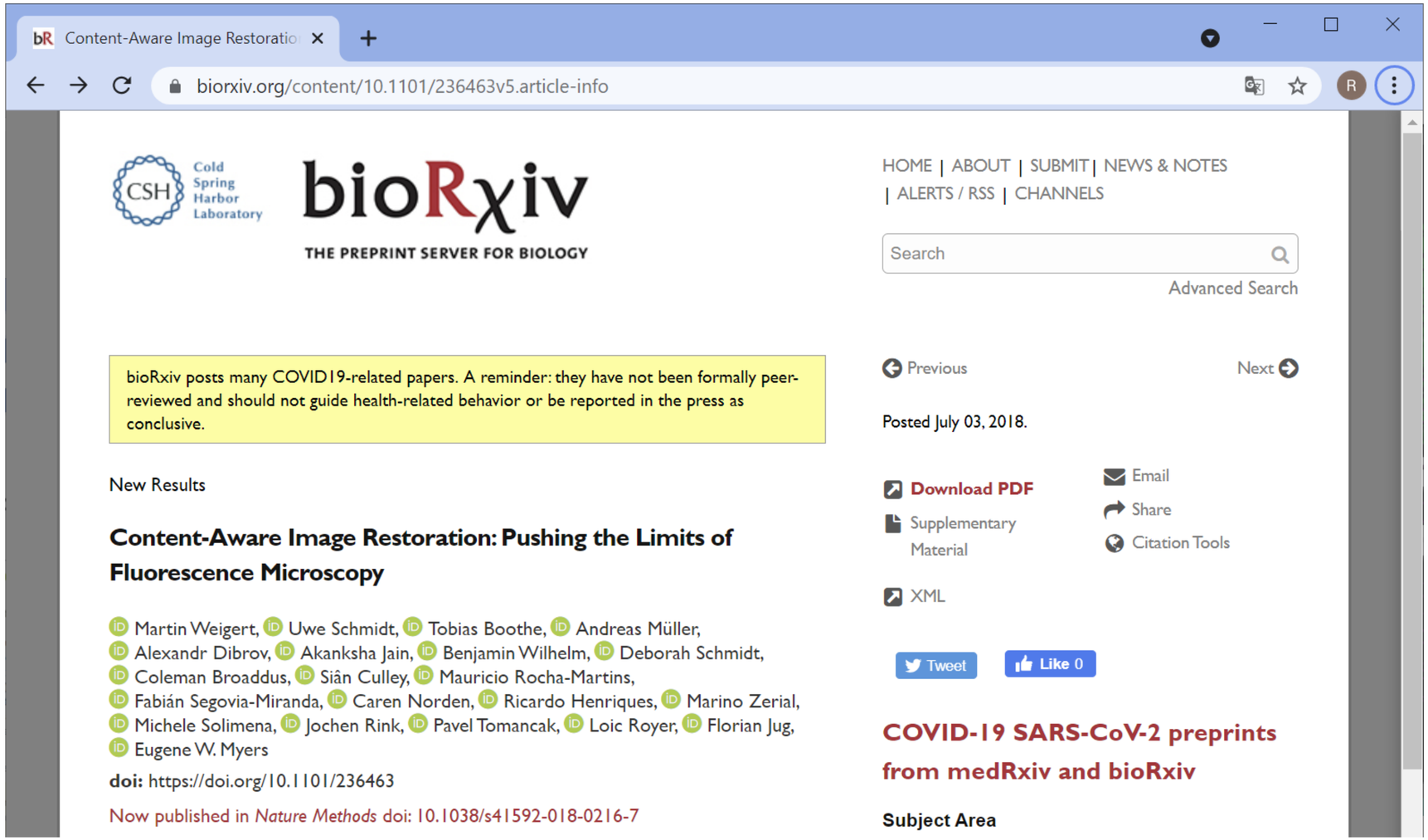
https://rodare.hzdr.de/record/915

https://twitter.com/nKiaru/status/1409194004219142148?s=20

Use cases: Manuscripts

PoL Physics of Life TU Dresden

- Preprints
  - —Accessible / reusable
  - —https://arxiv.org/
  - —https://biorxiv.org/
  - —https://medrxiv.org/
- Journals



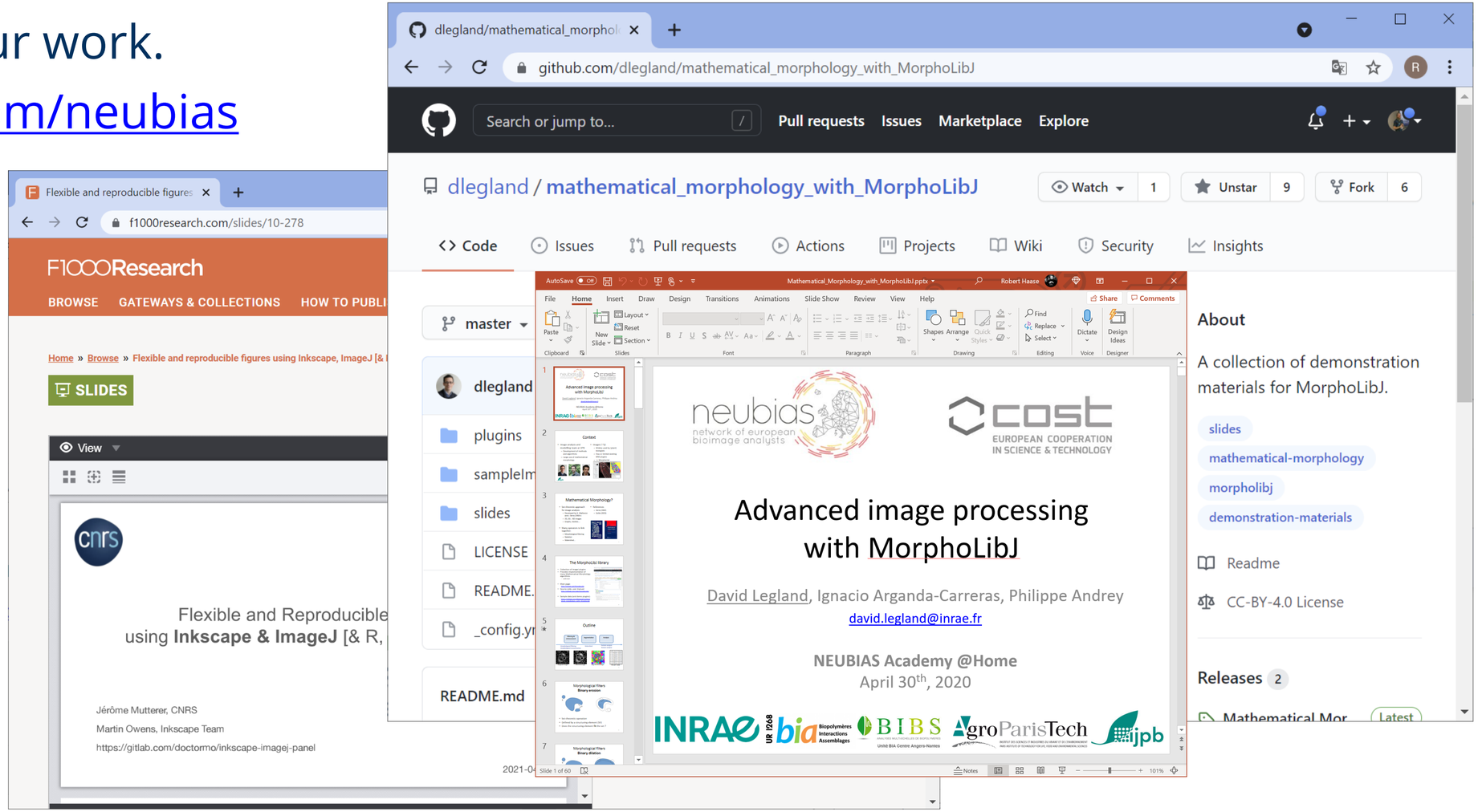
Use cases: Teaching material

Re-using, advertising your work.

https://f1000research.com/neubias

https://figshare.com

https://github.com



Use cases: Figures

Share efforts

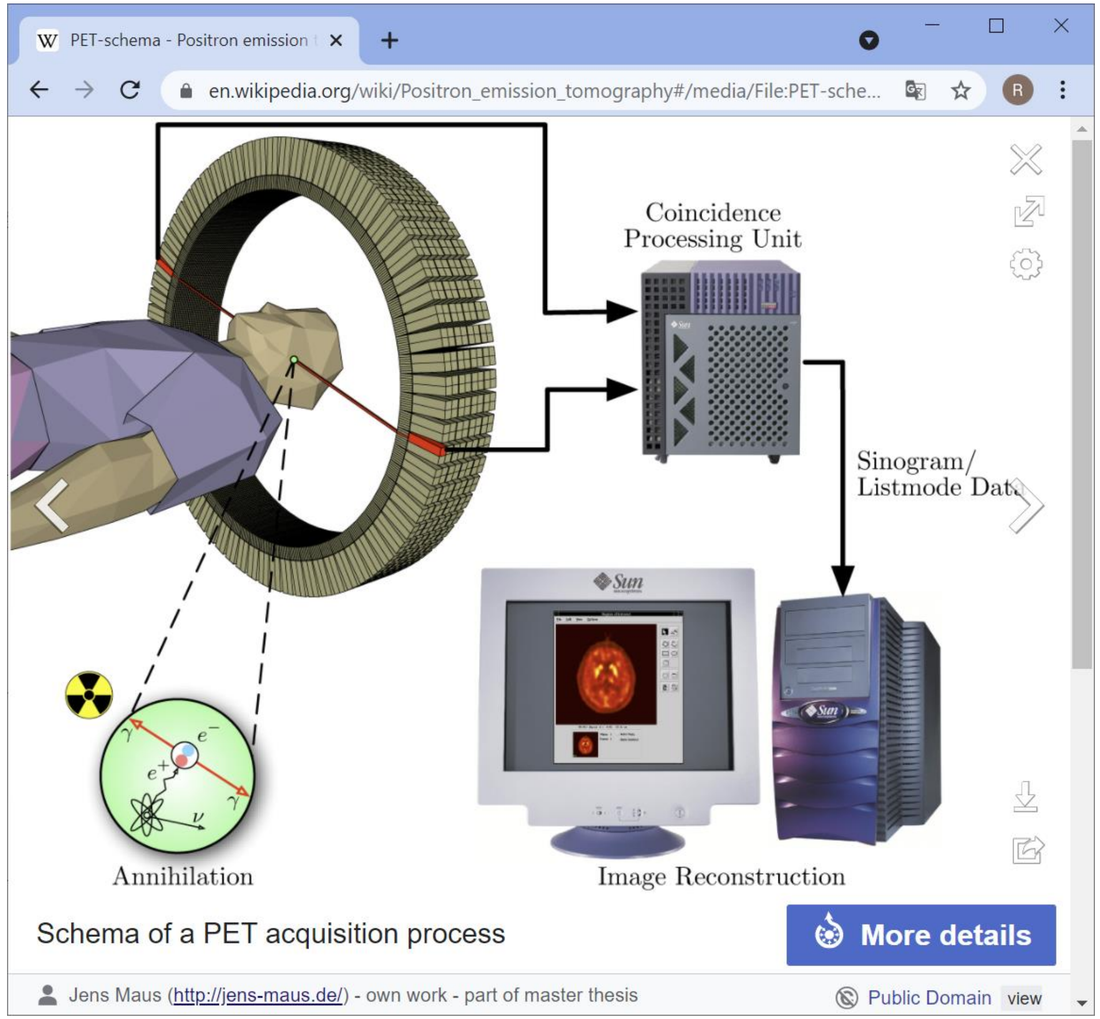
—Talk about each others' work

Advertise your work

… because our work is often publicly funded

https://commons.wikimedia.org/wiki/Main_Page

https://figshare.com

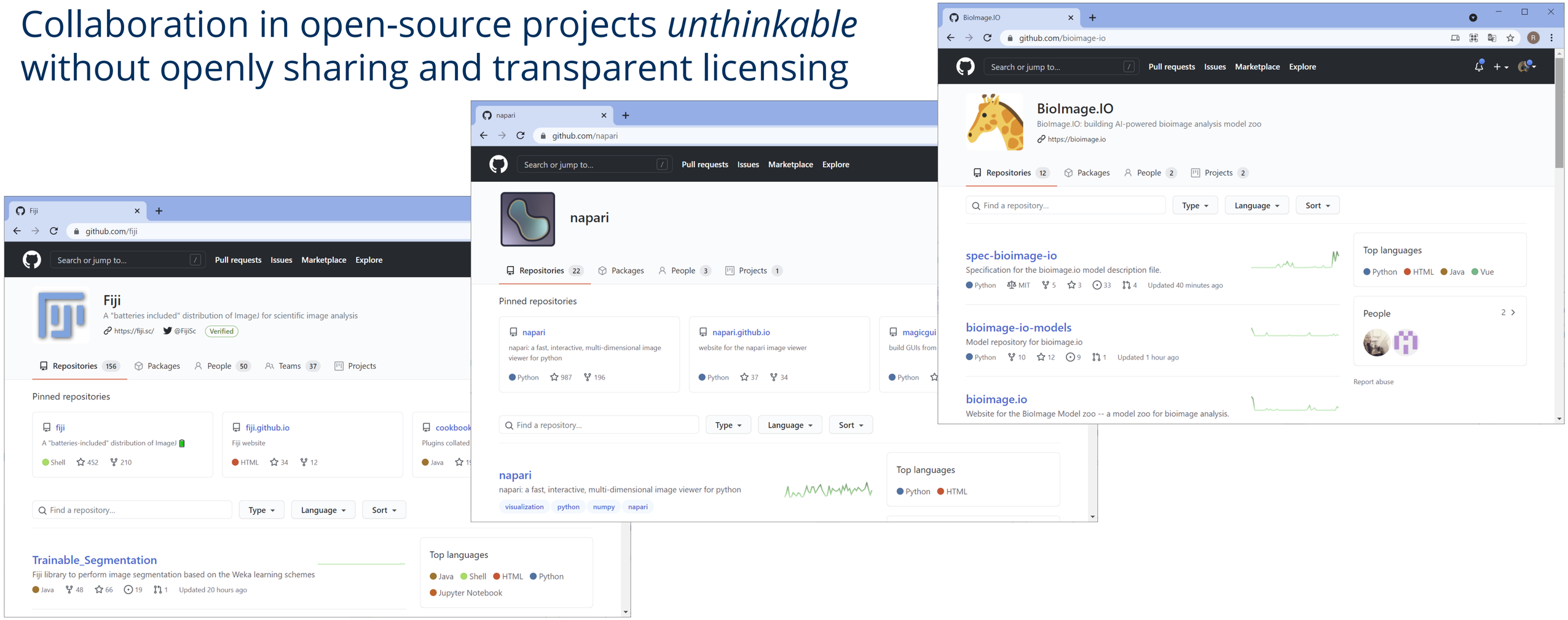


https://en.wikipedia.org/wiki/Positron_emission_tomography#/media/File:PET-schema.png

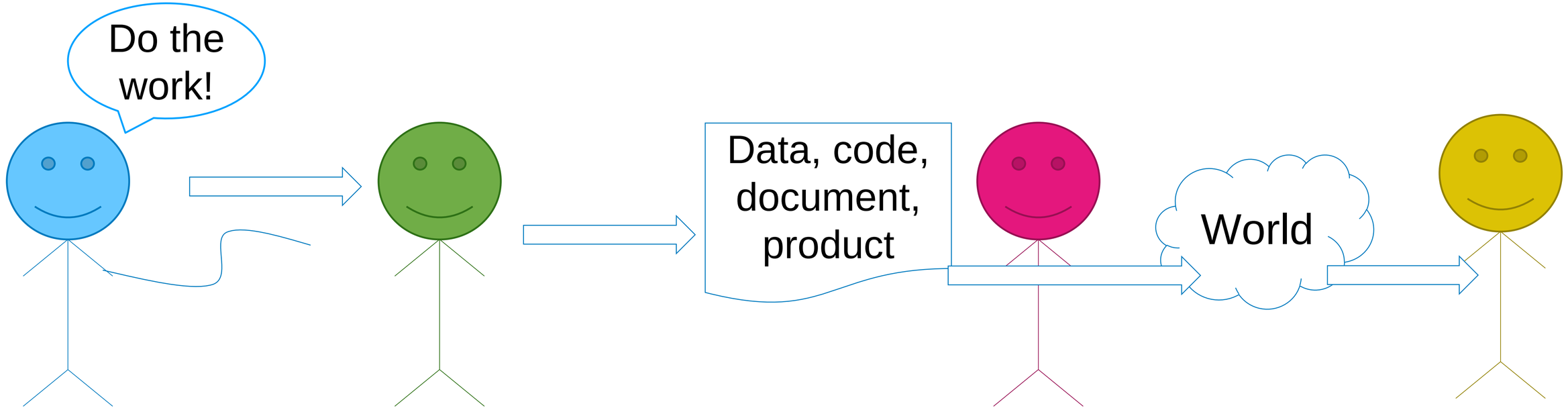
Use cases: Code

PoL Physics of Life TU Dresden

Collaboration in open-source projects *unthinkable* without openly sharing and transparent licensing



Terminology



Copyright holder
E.g. employer of the author

Author
Creator of the work

Publisher
The person who made the work available

Licensee
The person who uses / reuses the material

Acknowledgement, attribution, fees, …

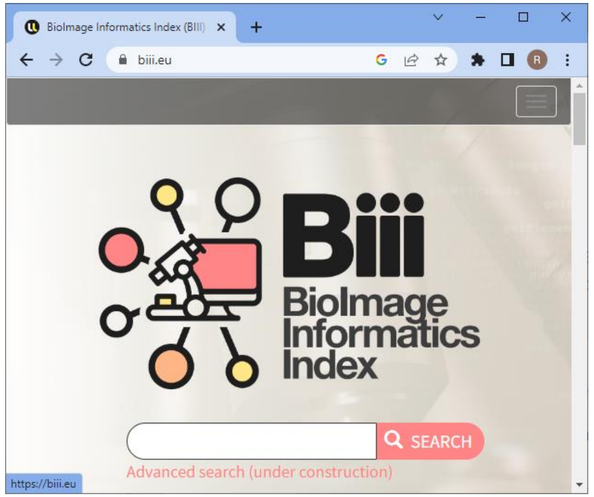
FAIR principles

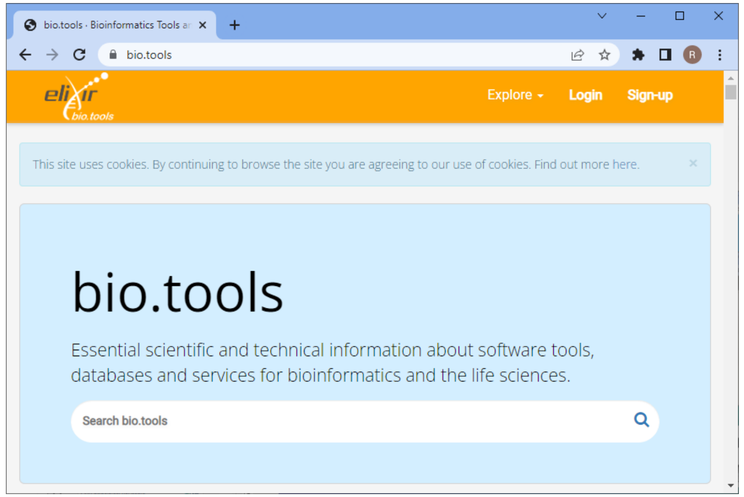
- Findable
- Accessible
- Interoperable
- Reusable

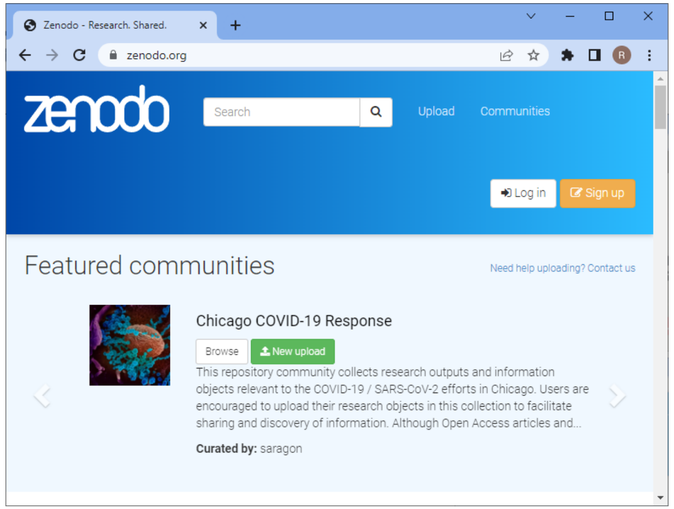
=> State-of-the-art Research Data Management (RDM)

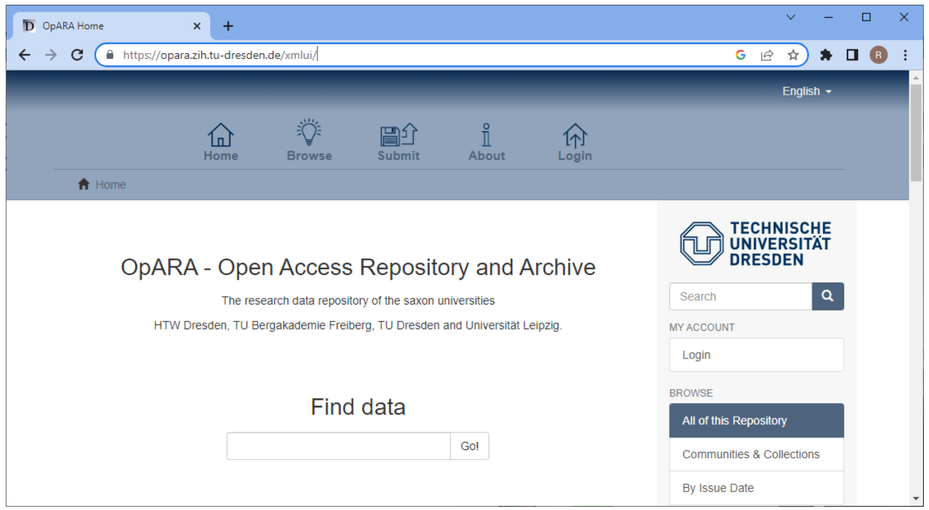
FAIR principles: Findable

- Research data / code / … can be found if it's listed in *repositories*
  - Preferably: global, public, field-specific repository
  - Alternative: institutional repository
- Findability can be improved through attaching
  - *meta data*
  - *unique digital object identifiers* (DOI)









Quiz: Digital object identifiers

PoL Physics of Life TU Dresden

Which of these is a *unique* digital object identifier?

https://twitter.com/haesleinhuepf/status/891596662782779392

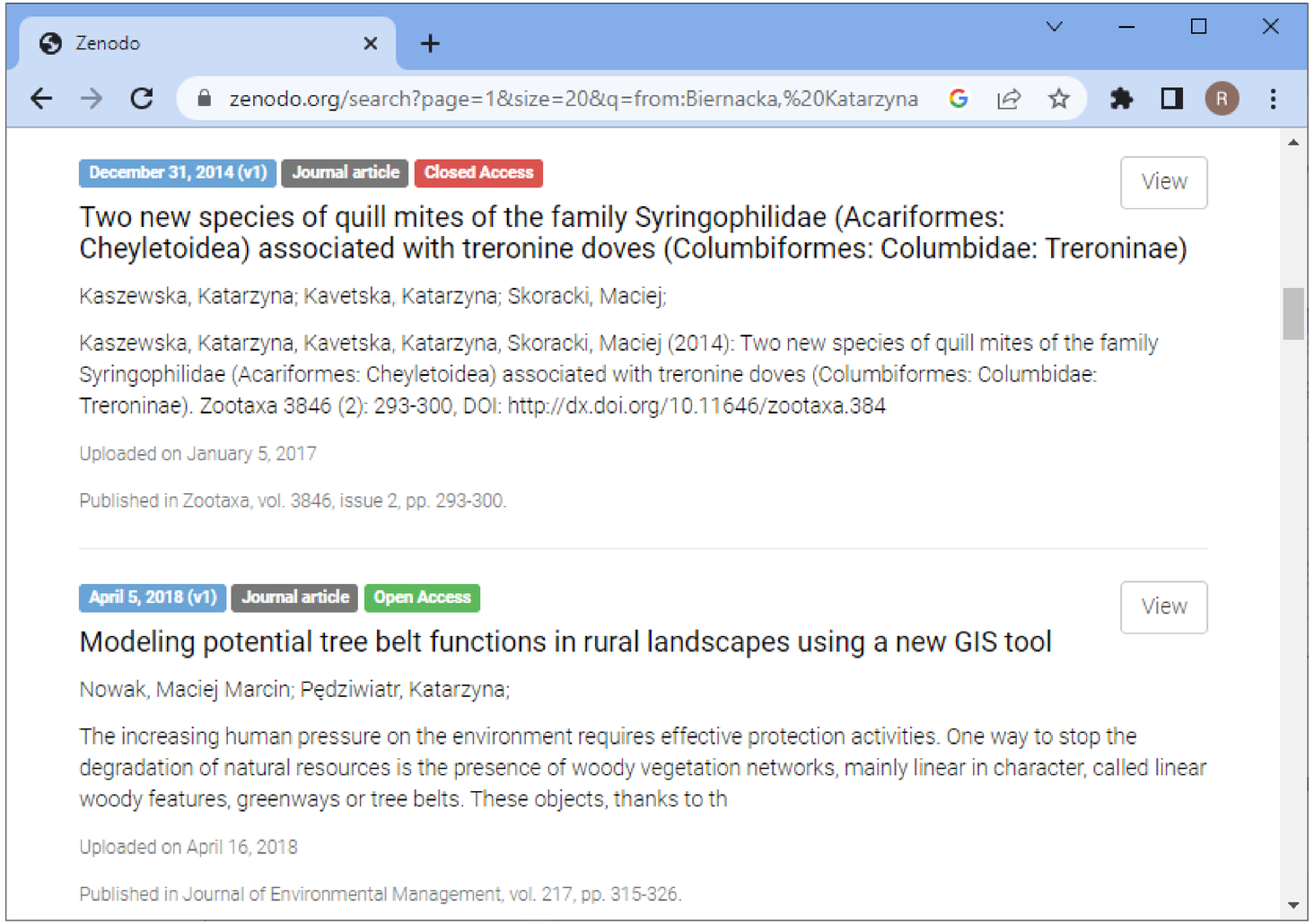
https://doi.org/10.5281/zenodo.28325

https://github.com/haesleinhuepf/devbio-napari

https://napari.org/

FAIR principles: Accessible

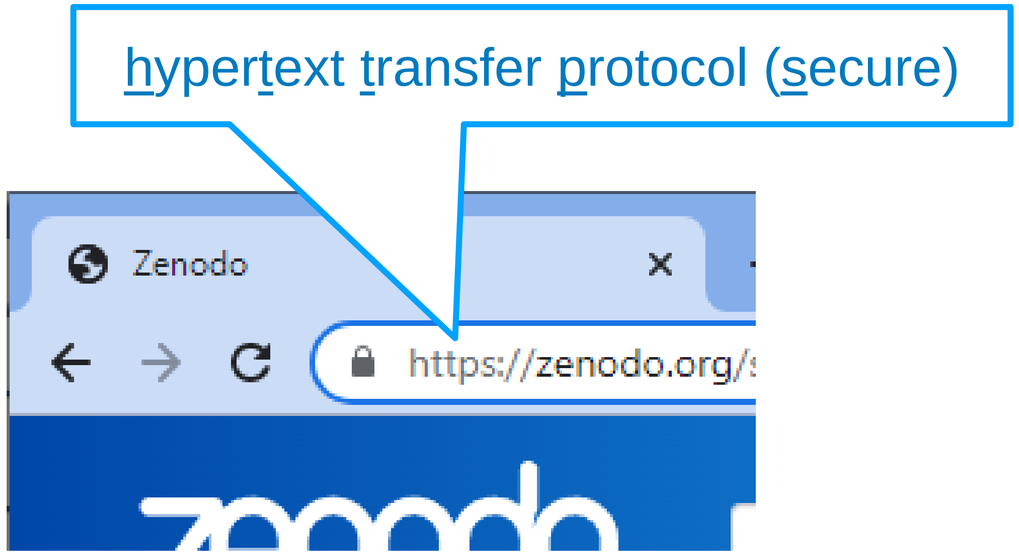
- Research data can be made accessible (after it was found by potential users)
  - Open Access is just one form of accessibility
  - Authentication enables other forms

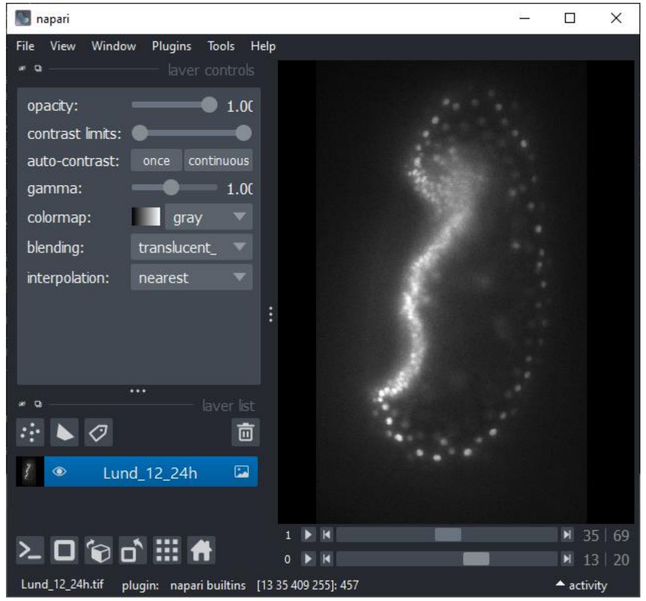


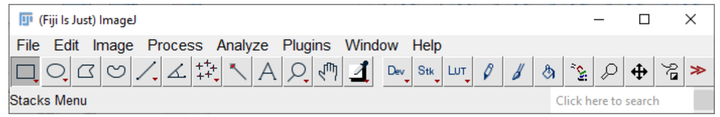
FAIR principles: Interoperable

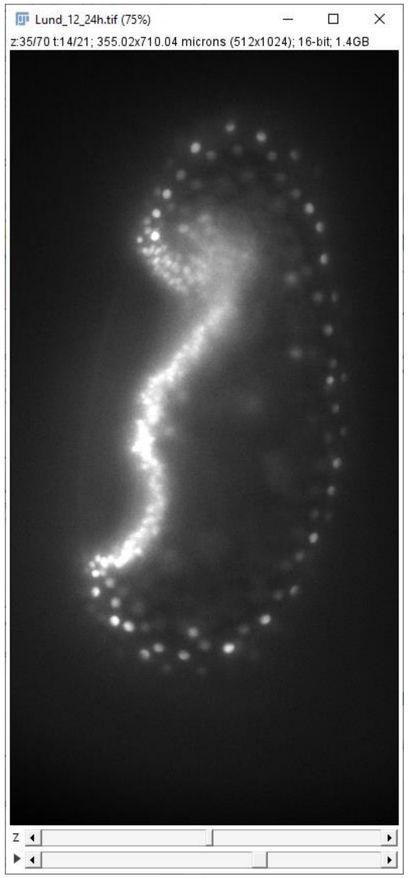
Data can be opened in multiple software through open and documented…

- file-formats
- protocols









FAIR principles: Reusable

Data can be reused if

- Other FAIR principles are fulfilled
- Data is properly licensed
  - copyright statement is given
  - No copyright statement means: You do not have the right to copy

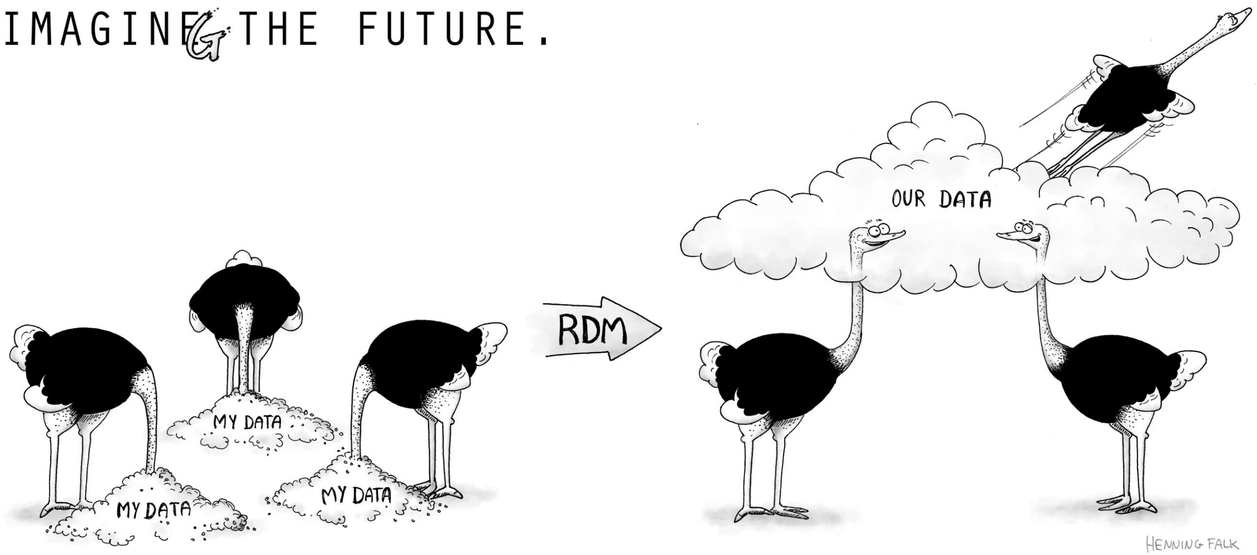
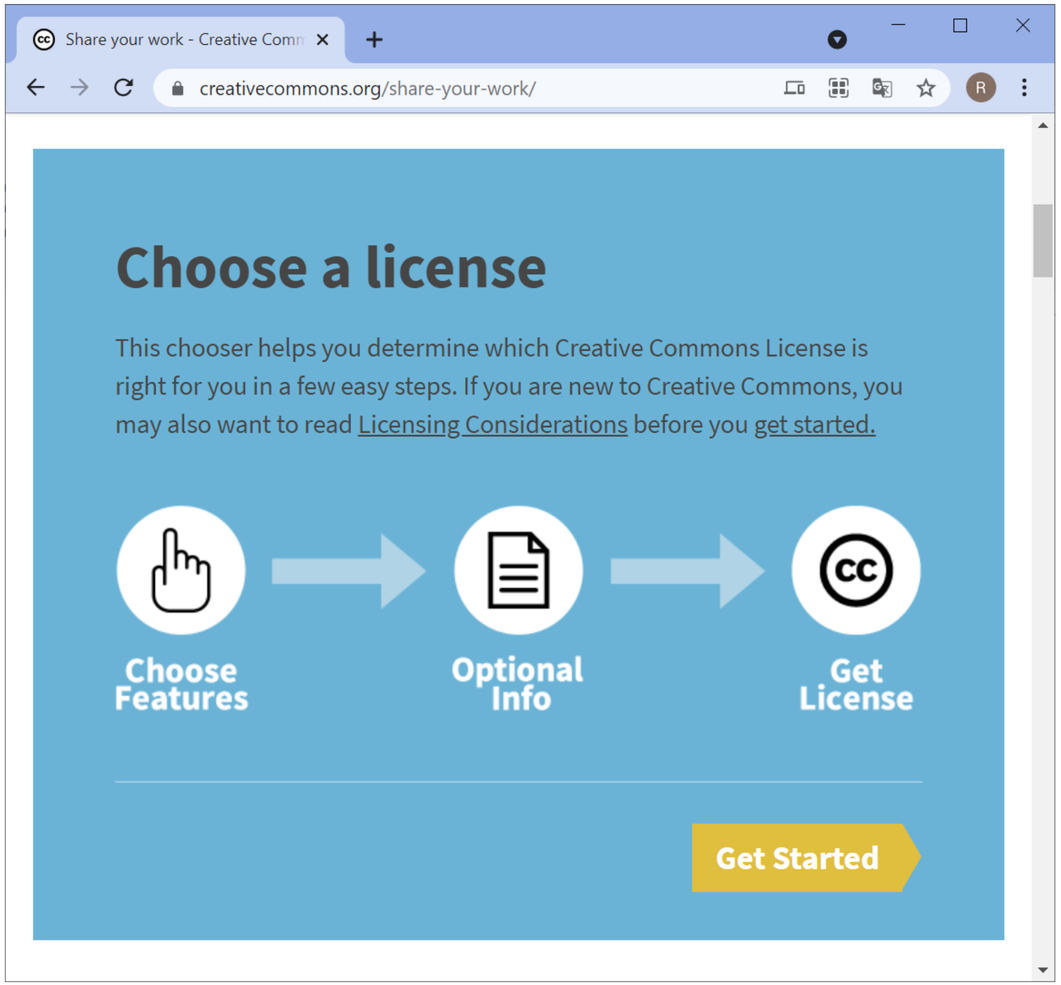


Image source: @DrHenningFalk, licensed CC-BY 4.0

Licensing: Creative Commons (CC)

Public domain (CC0)
Attribution International (CC-BY)
Attribution ShareAlike Int. (CC-BY-SA)
Attribution Non-Commercial Int. (CC-BY-NC)
Attribution NoDerivatives Int. (CC-BY-ND)

+ Combinations, e.g. CC-BY-NC-ND



Licensing: Creative Commons (CC)

Public domain (CC0)

Everyone can reuse without mentioning the source or author of the shared resource.

Public domain licenses cannot be revoked.

The author must own the right to copy (copyright) the resource.

—If you authored work as part of your job, you may not be the copyright holder. (check your employers' guidelines)

Employers don't like this one because you give away the rights to *exploit* you work.

Licensing: Creative Commons (CC)

"By attribution"

CC BY: This license allows reusers to distribute, remix, adapt, and build upon the material in any medium or format, so long as attribution is given to the creator. The license allows for commercial use.

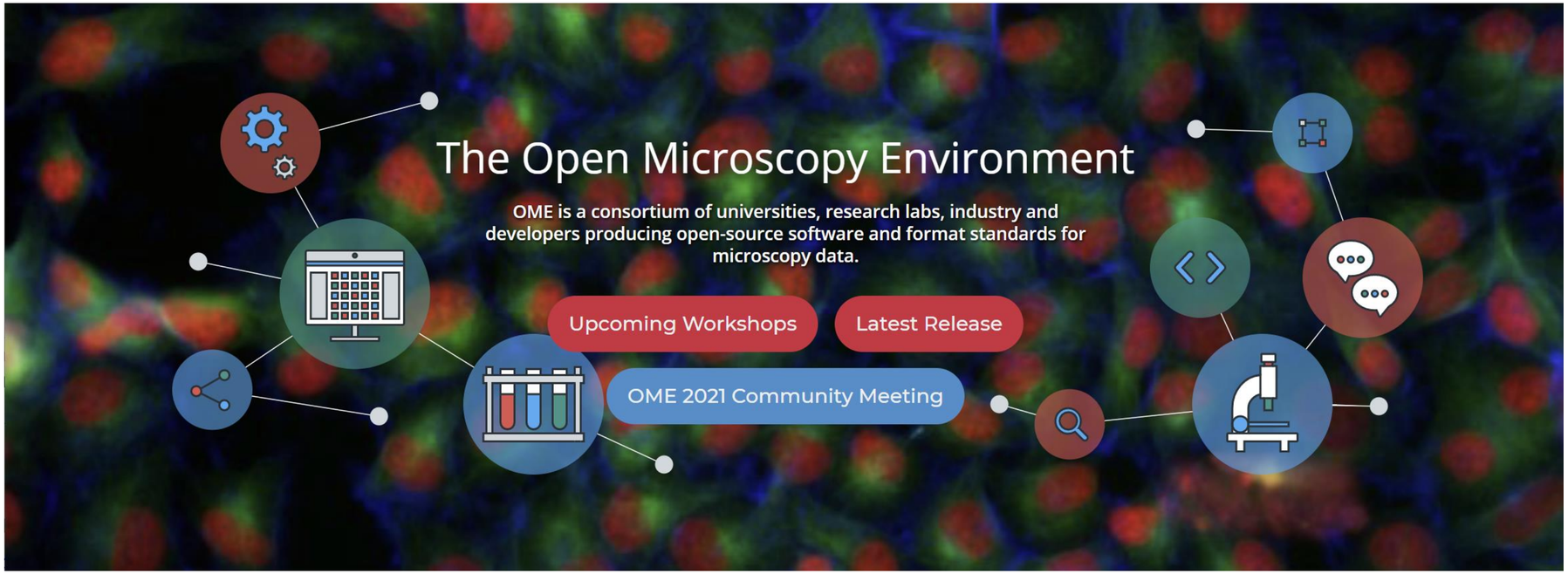
CC BY includes the following elements:

BY – Credit must be given to the creator

Licensing: Creative Commons (CC)

PoL Physics of Life TU Dresden

Example



You must put such a sentence and keep the link to CC-BY

Figure adapted from https://www.openmicroscopy.org/ licensed by University of Dundee & Open Microscopy Environment under Creative Commons Attribution 4.0 International License

Licensing: Creative Commons (CC)

CC BY-SA: This license allows reusers to distribute, remix, adapt, and build upon the material in any medium or format, so long as attribution is given to the creator. The license allows for commercial use. If you remix, adapt, or build upon the material, you must license the modified material under identical terms.

CC BY-SA includes the following elements:

BY – Credit must be given to the creator

SA – Adaptations must be shared under the same terms

"Restrictive" licensing

"Share alike"

Licensing: Creative Commons (CC)

CC BY-NC: This license allows reusers to distribute, remix, adapt, and build upon the material in any medium or format for noncommercial purposes only, and only so long as attribution is given to the creator.

It includes the following elements:

BY – Credit must be given to the creator

NC – Only noncommercial uses of the work are permitted

Licensing: Creative Commons (CC)

CC BY-ND: This license allows reusers to copy and distribute the material in any medium or format in unadapted form only, and only so long as attribution is given to the creator. The license allows for commercial use.

CC BY-ND includes the following elements:

BY – Credit must be given to the creator

ND – No derivatives or adaptations of the work are permitted

"Restrictive" licensing

Licensing: Creative Commons (CC)

CC BY-NC-ND: This license allows reusers to copy and distribute the material in any medium or format in unadapted form only, for noncommercial purposes only, and only so long as attribution is given to the creator.

CC BY-NC-ND includes the following elements:

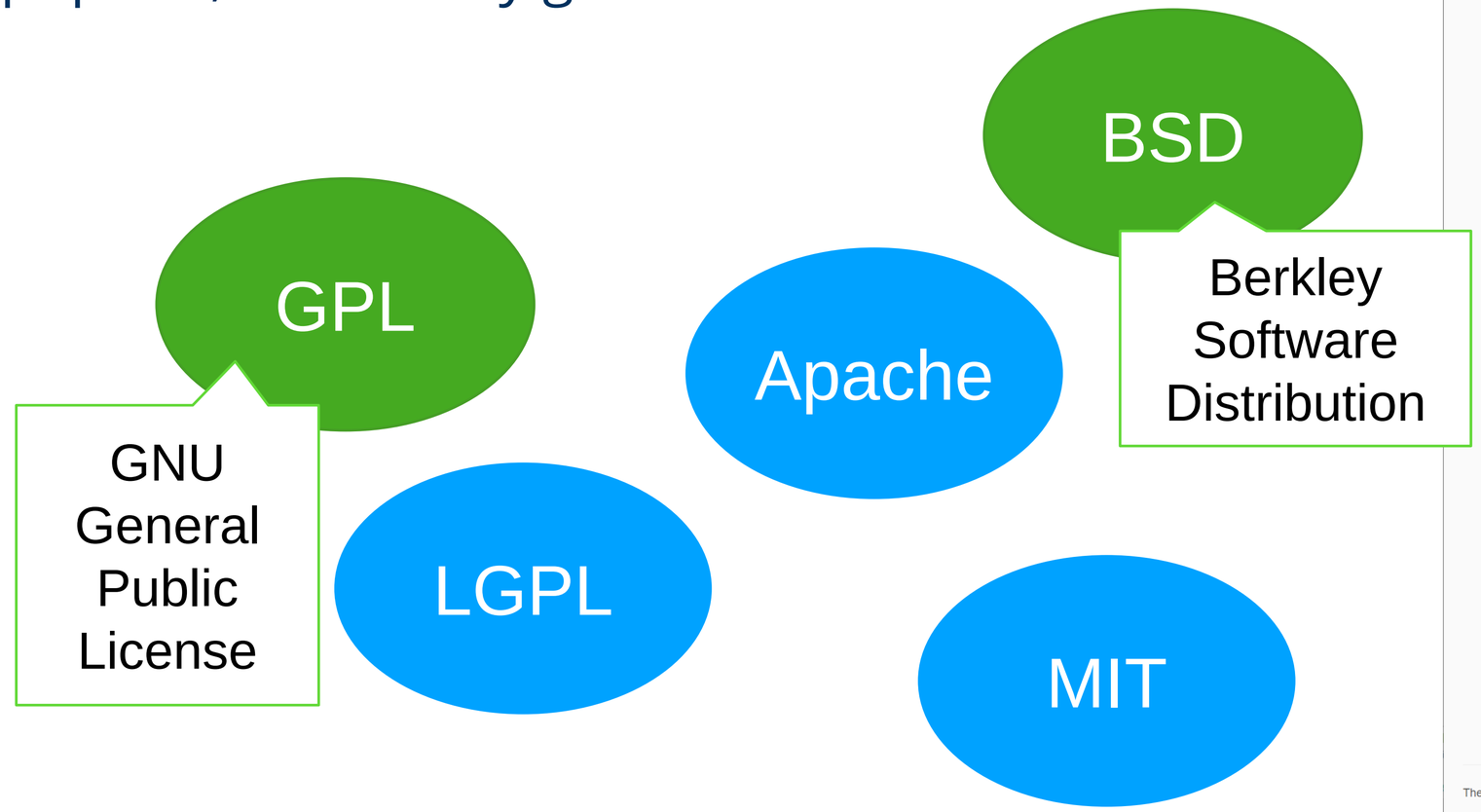
BY – Credit must be given to the creator

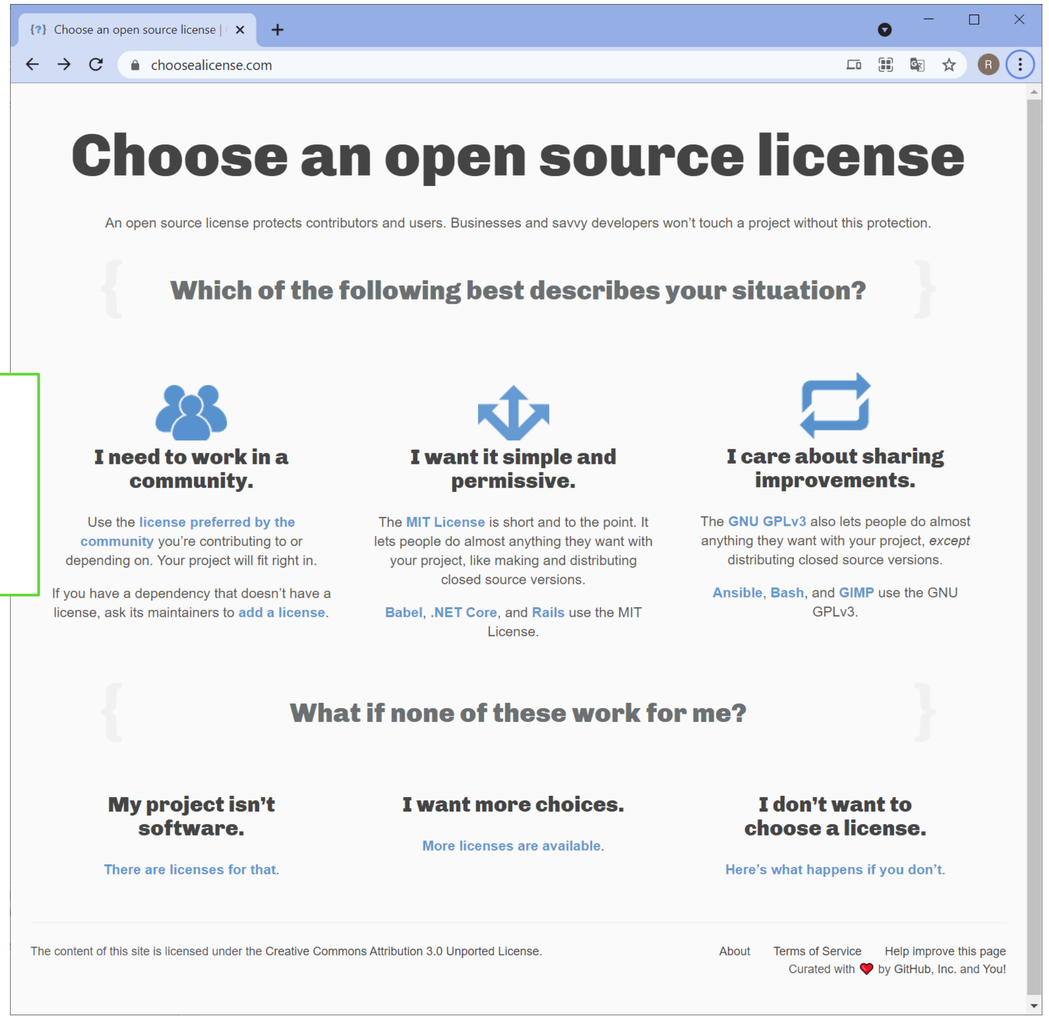
NC – Only noncommercial uses of the work are permitted

ND – No derivatives or adaptations of the work are permitted

Licensing software

In the software world, other licenses are more popular, historically grown.





Licensing Software: GPL

PoL Physics of Life TU Dresden

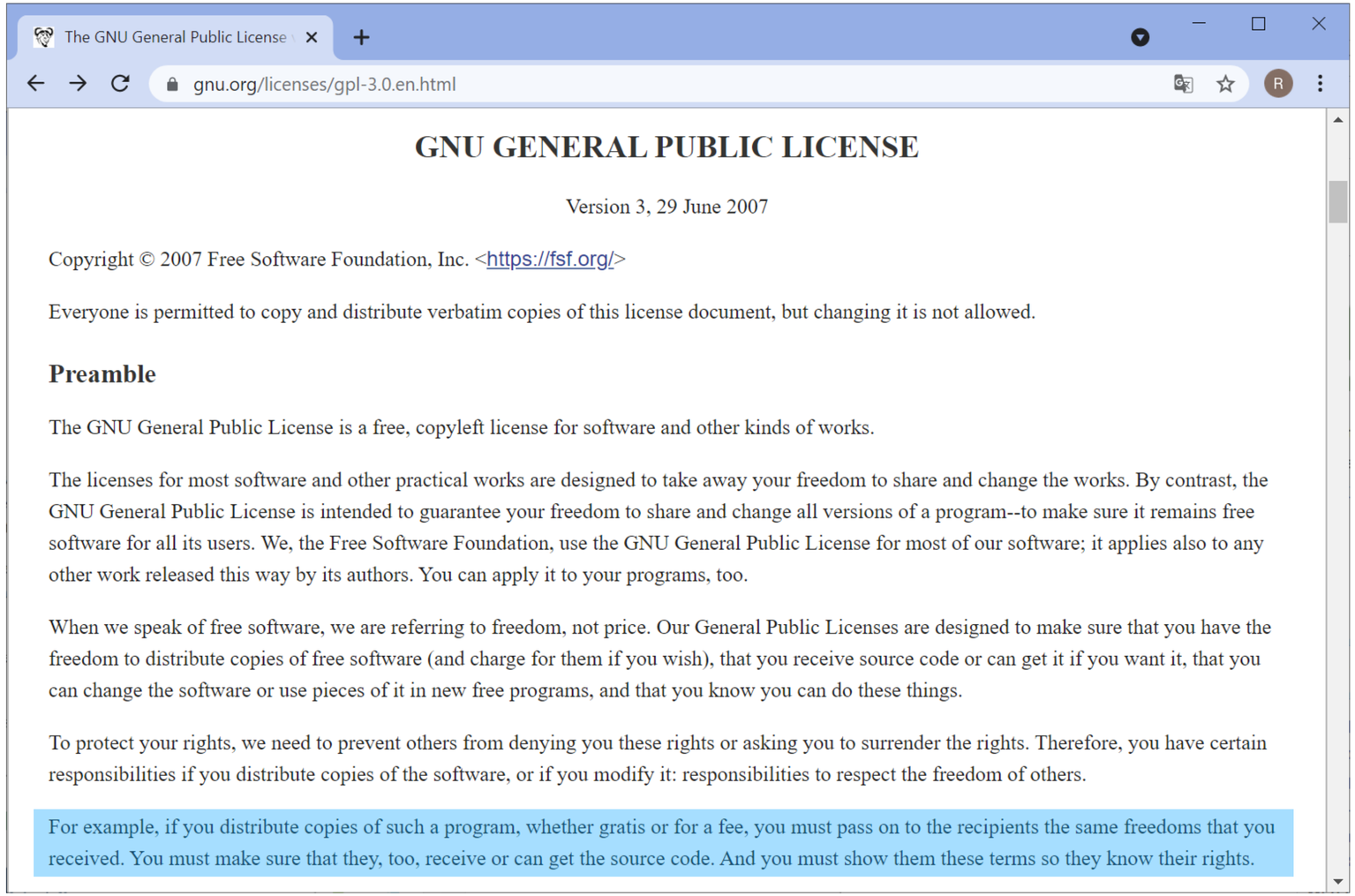
GPL

- Derivatives must also be GPL-licensed

"Restrictive" licensing

See also:

- Lesser General Public License (LGPL)
  - —Integrate LGPL-licensed code into not-LGPL-licensed code



https://www.gnu.org/licenses/gpl-3.0.en.html
https://en.wikipedia.org/wiki/GNU_General_Public_License#Copyleft

Licensing Software: BSD0

Copyright (C) [year] by [copyright holder] <[email]>

Permission to use, copy, modify, and/or distribute this software for any purpose with or without fee is hereby granted.

Similar to CC0 (public domain)

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE

Licensing Software: BSD0

Copyright (C) [year] by [copyright holder] <[email]>

Permission to use, copy, modify, and/or distribute this software for any purpose with or without fee is hereby granted.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE

Disclaimer

Whatever you do with it, we [the authors] are not liable

Relevant for [image] data analysis script-authors!

Licensing Software: BSD2

Copyright (c) <year>, <copyright holder>

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Similar to CC-BY

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE

Licensing Software: BSD3

Copyright <year> <copyright holder>

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Similar to CC-BY

3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

You must not use the copyright holder's name to endorse your derivative of the work.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE

That's also part of all CC-BY licenses

How/when to decide for a license

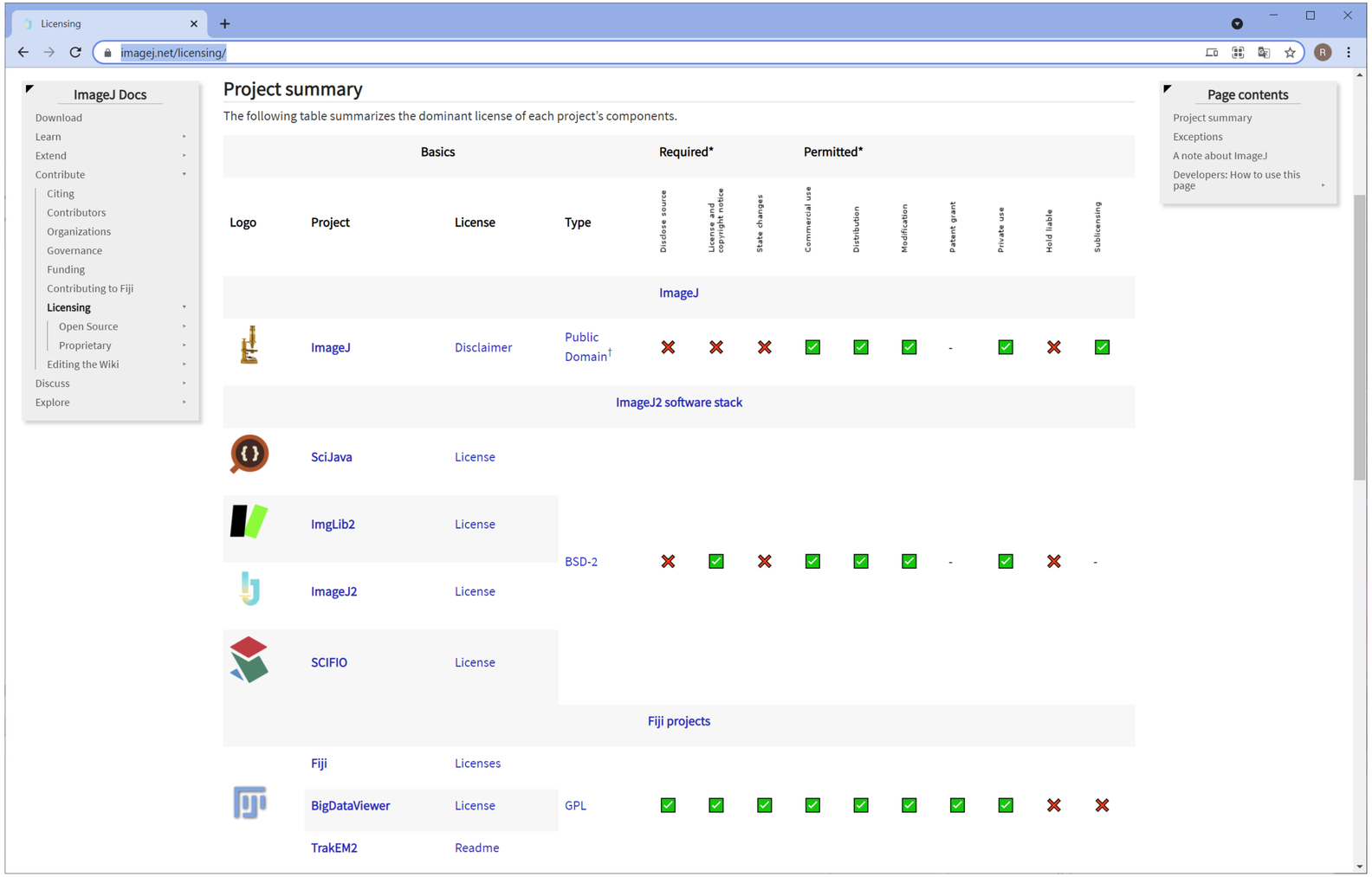
How?

- Choose a license compatible with your ecosystem

When?

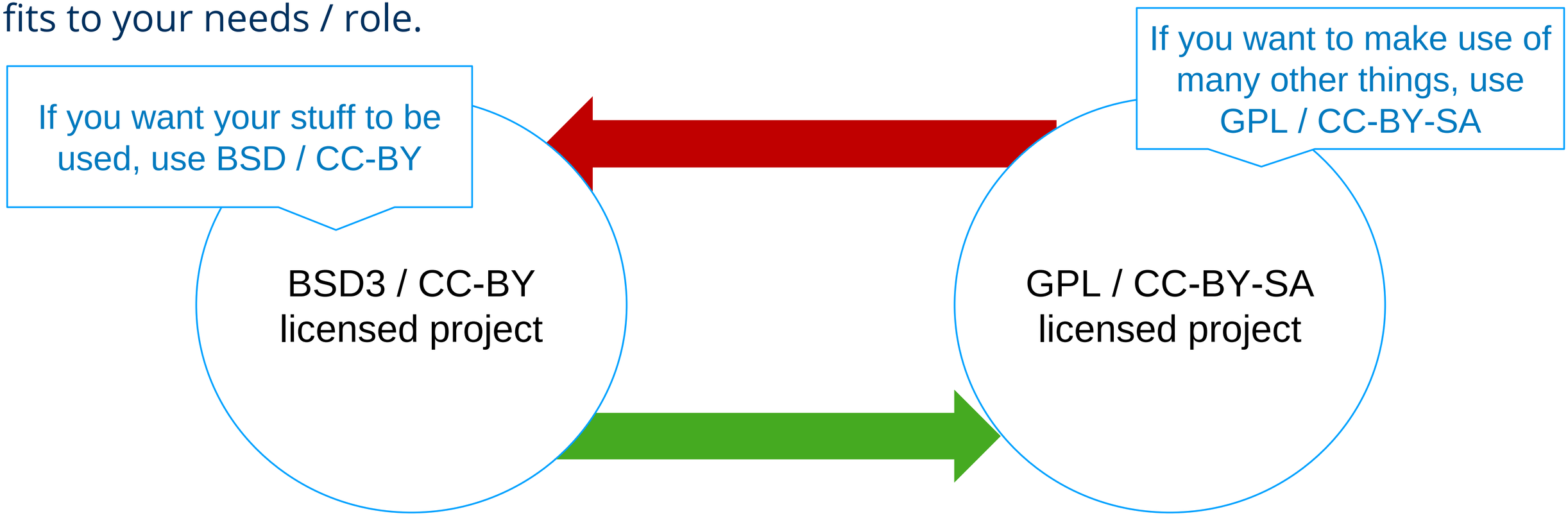
- When the project starts
- As early as possible
- Changing the license later may be hard.

- Beware: Your employer might be the copyright holder. They have the final word on how to license, publish and make accessible your work!



Permissive versus restrictive

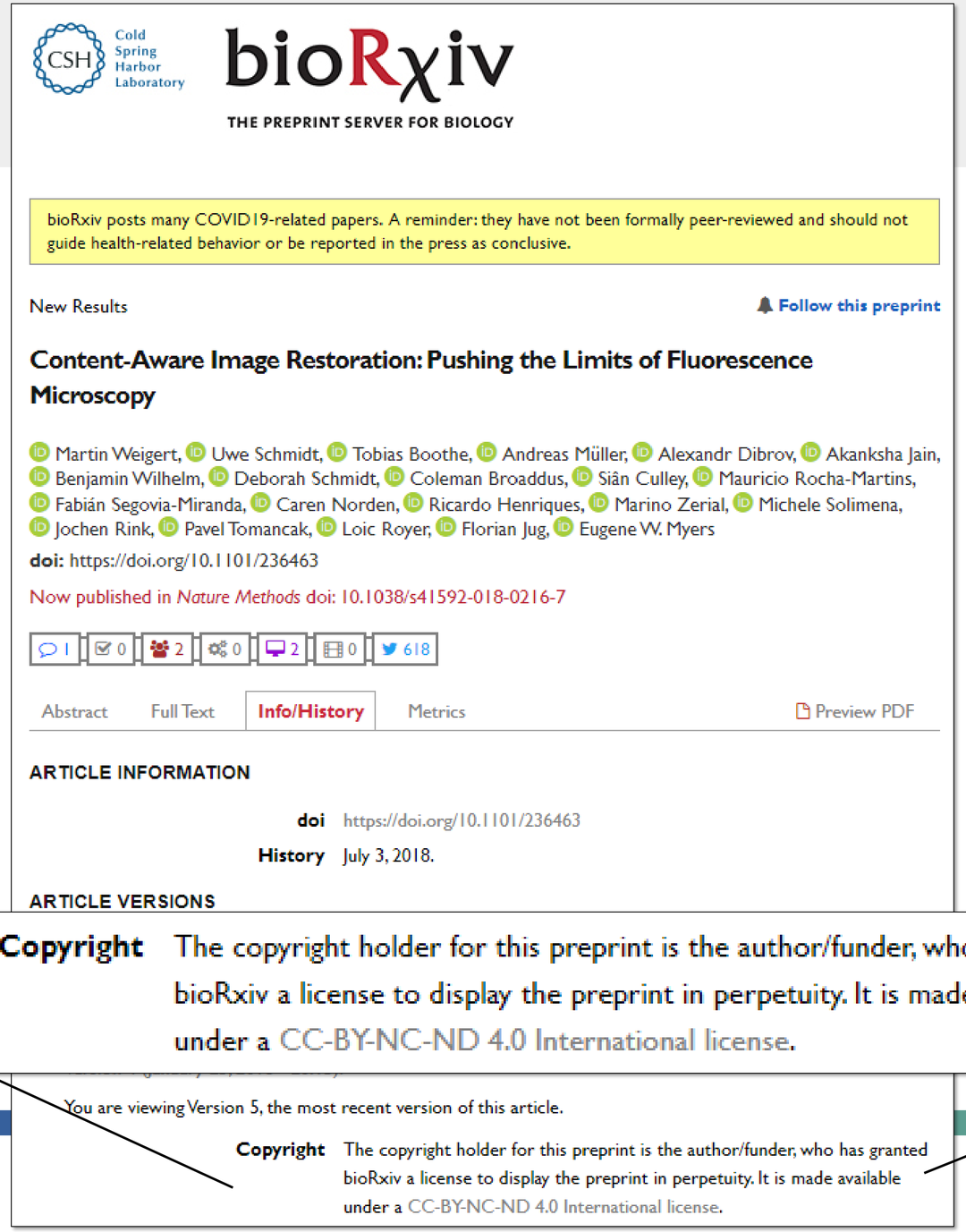
Use a license which
is compatible to other projects you're collaborating with and
fits to your needs / role.



Quiz

May I use one of the Figures from this preprint?

May I download and redistribute this preprint to students of a course for free?



Cold Spring Harbor Laboratory

bioRxiv

THE PREPRINT SERVER FOR BIOLOGY

bioRxiv posts many COVID19-related papers. A reminder: they have not been formally peer-reviewed and should not guide health-related behavior or be reported in the press as conclusive.

New Results

Follow this preprint

Content-Aware Image Restoration: Pushing the Limits of Fluorescence Microscopy

Martin Weigert, Uwe Schmidt, Tobias Boothe, Andreas Müller, Alexandr Dibrov, Akanksha Jain, Benjamin Wilhelm, Deborah Schmidt, Coleman Broaddus, Siân Culley, Mauricio Rocha-Martins, Fabián Segovia-Miranda, Caren Norden, Ricardo Henriques, Marino Zerial, Michele Solimena, Jochen Rink, Pavel Tomancak, Loic Royer, Florian Jug, Eugene W. Myers

doi: https://doi.org/10.1101/236463

Now published in *Nature Methods* doi: 10.1038/s41592-018-0216-7

1 0 2 0 2 0 618

Abstract Full Text Info/History Metrics Preview PDF

ARTICLE INFORMATION

doi https://doi.org/10.1101/236463

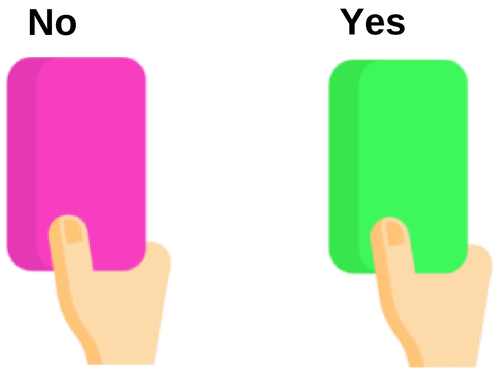
History July 3, 2018.

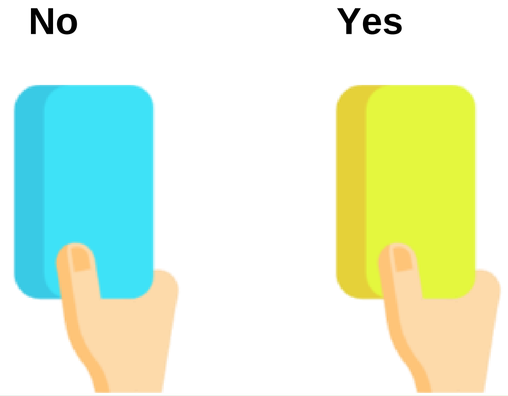
ARTICLE VERSIONS

Copyright The copyright holder for this preprint is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under a CC-BY-NC-ND 4.0 International license.

You are viewing Version 5, the most recent version of this article.

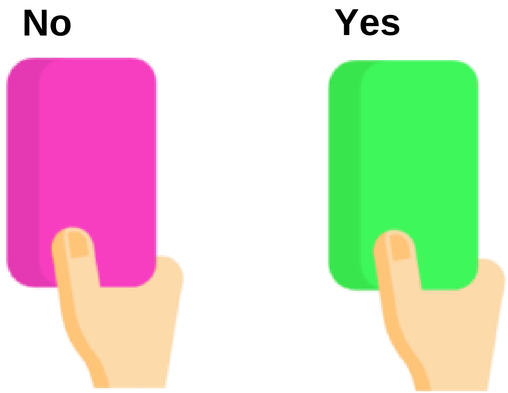
Copyright The copyright holder for this preprint is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under a CC-BY-NC-ND 4.0 International license.

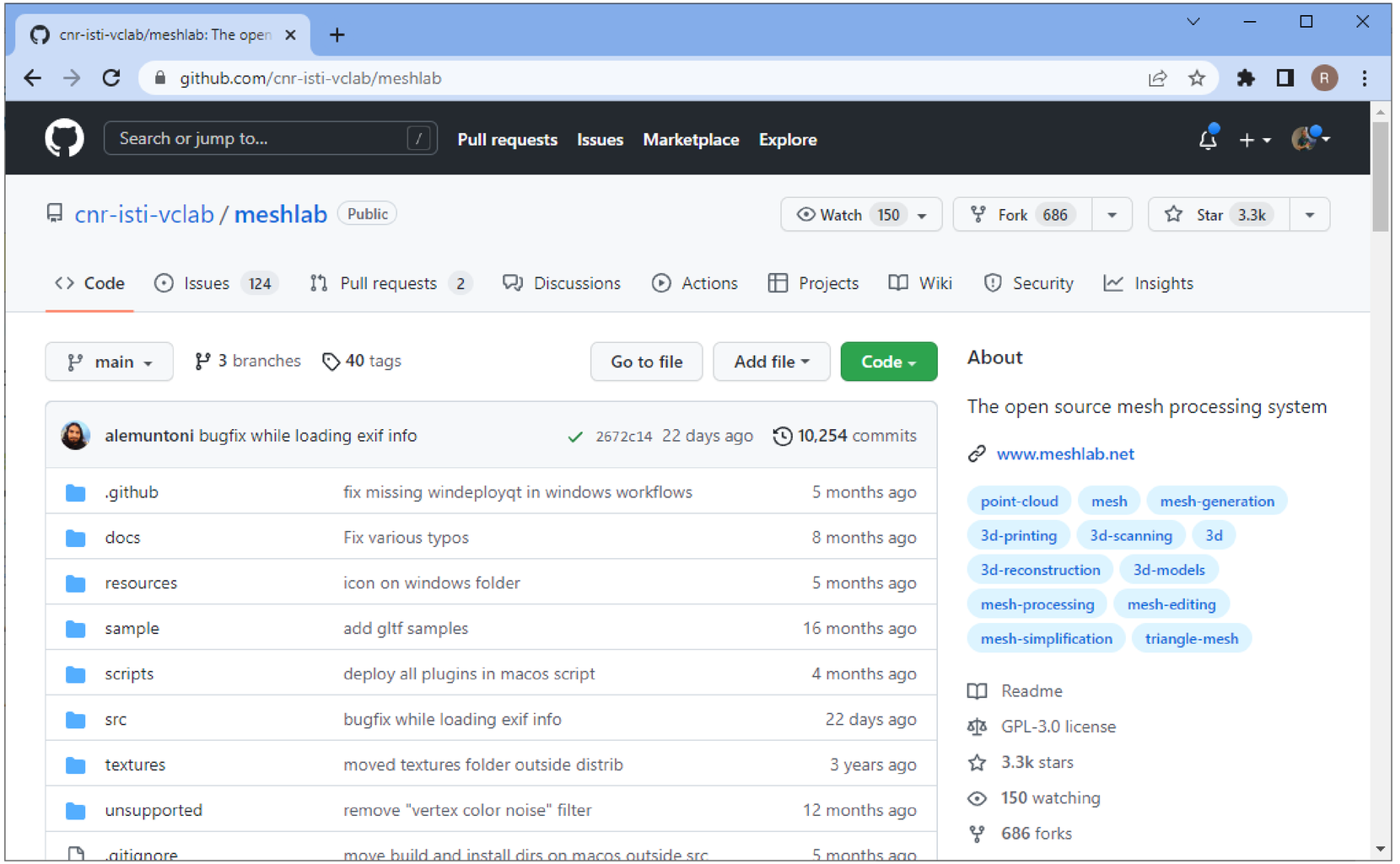




Quiz

May I reuse code from this repository in my own BSD-licensed work?



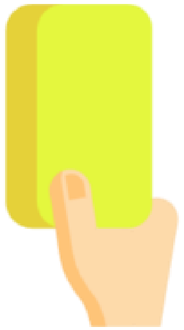


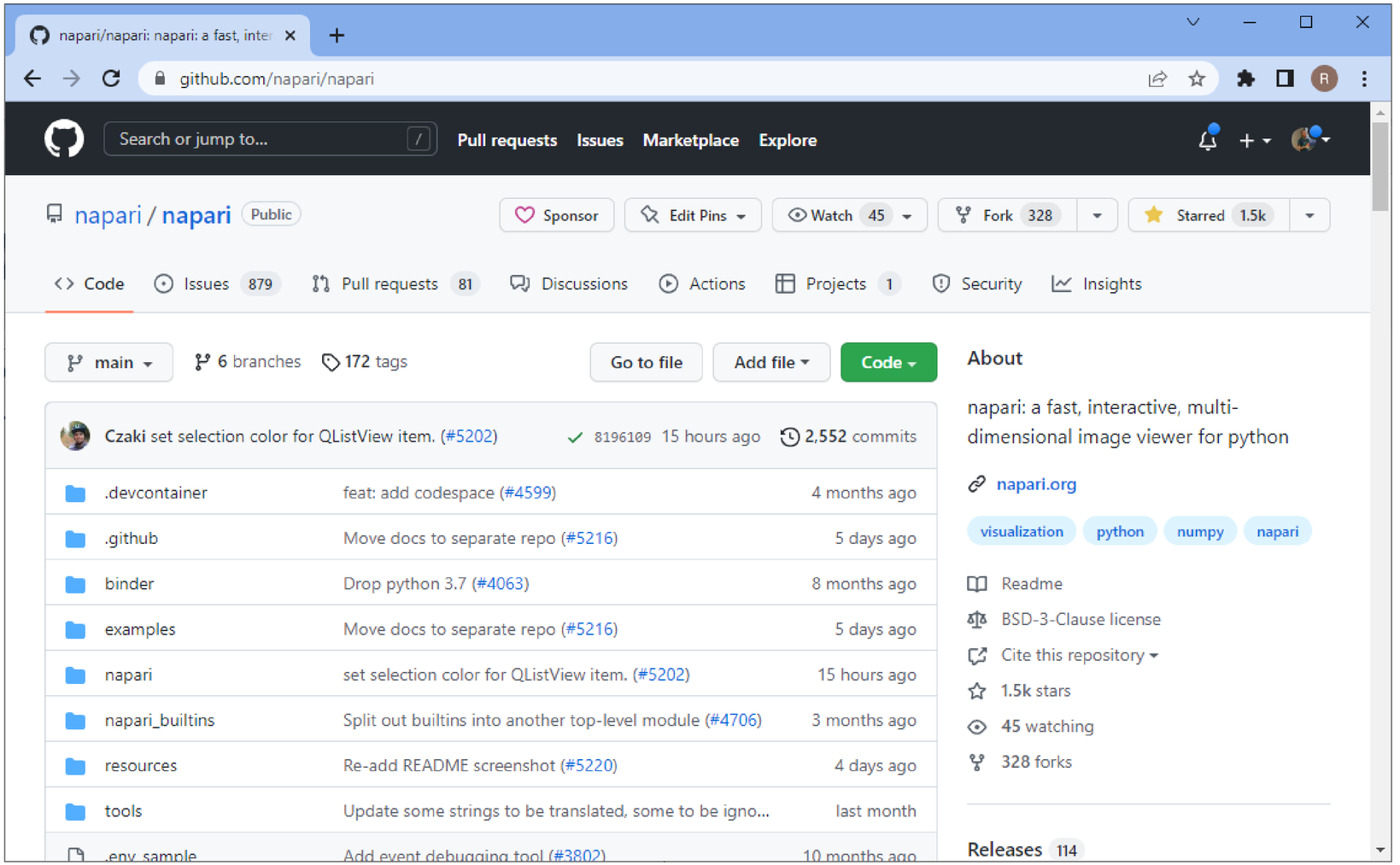
Quiz

May I reuse code from this repository in my own GPL-licensed work?

No

Yes





Take home message

If you share material (openly or not)

license it

and it’ll be harder to steal it

PoL Physics of Life TU Dresden

CENTER FOR SYSTEMS BIOLOGY DRESDEN

Python Algorithms: Conditions, loops, functions and custom libraries

Robert Haase

April 2023

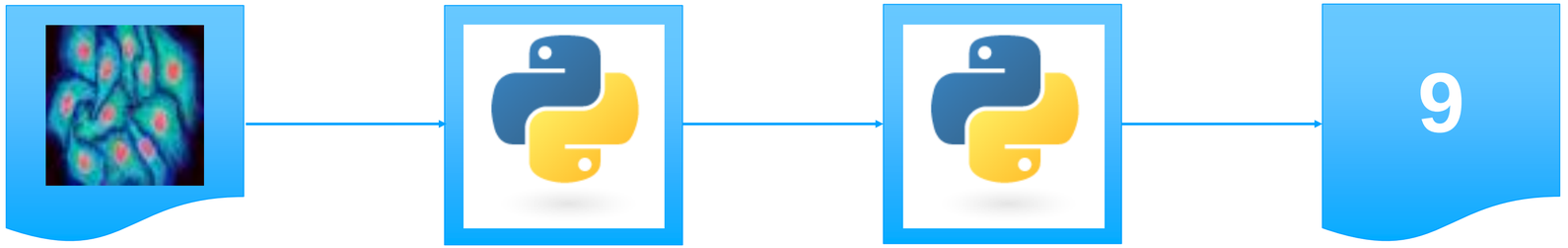
@haesleinhuepf

TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

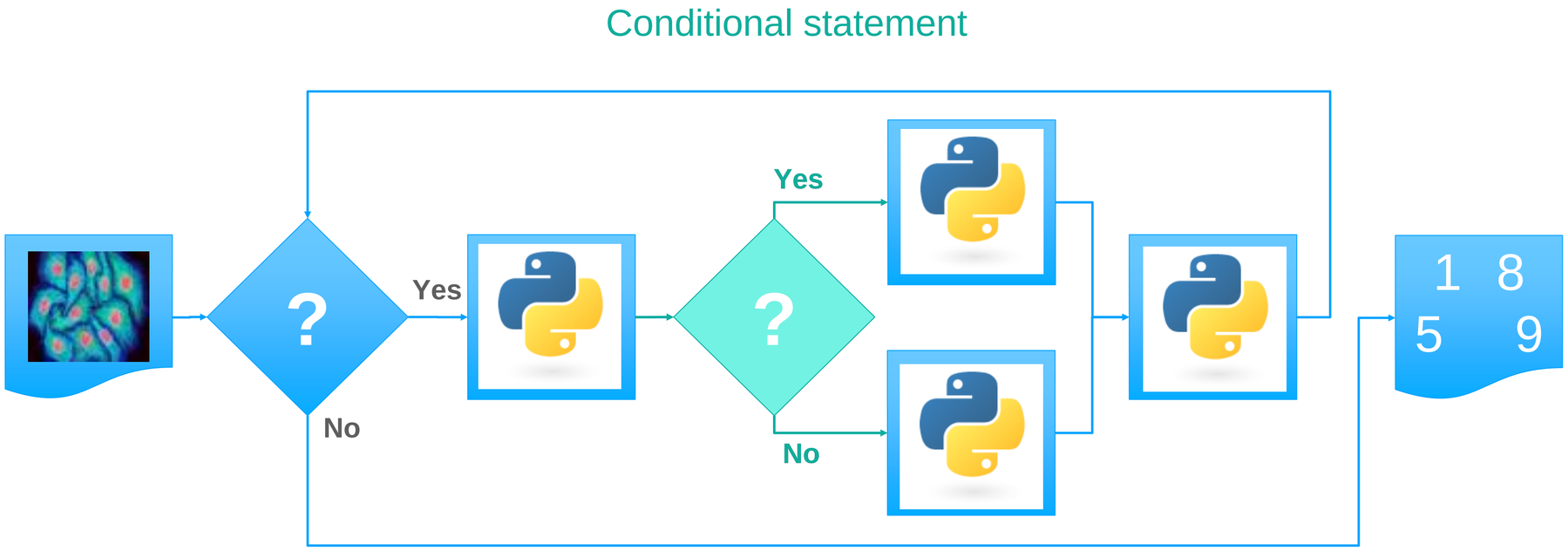
Conditions

Data science workflows *rarely* look like this



Conditions

Data science workflows *rather* look like this



Conditions

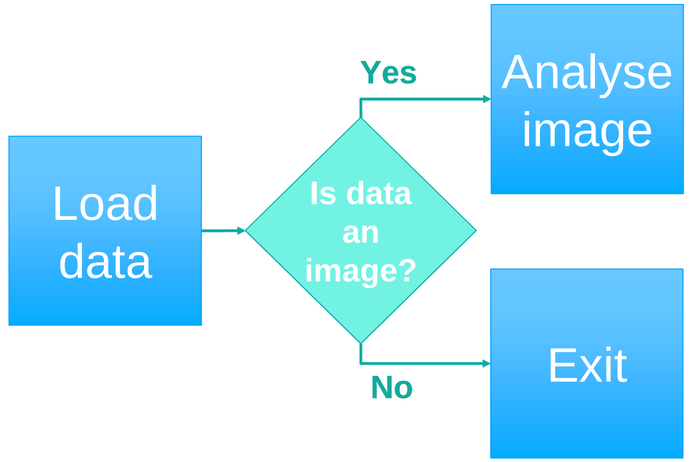
Conditional statements can be used to

— Check if pre-requisites are met

— Check if data has the right format

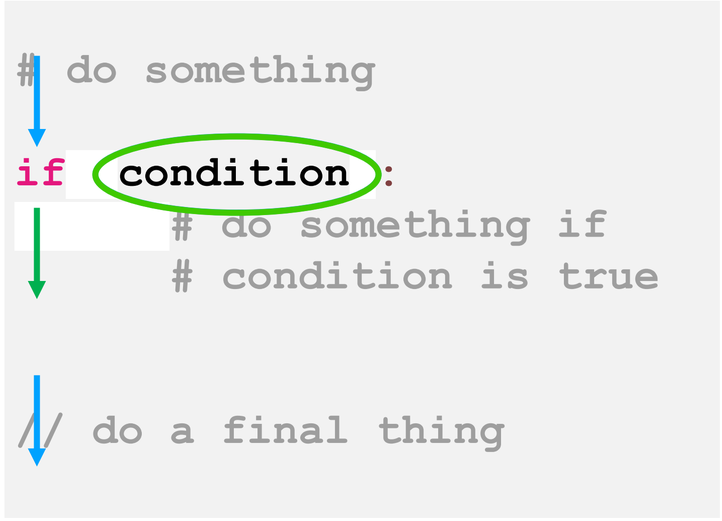
— Check if processing results are within an expected range

— Check for errors



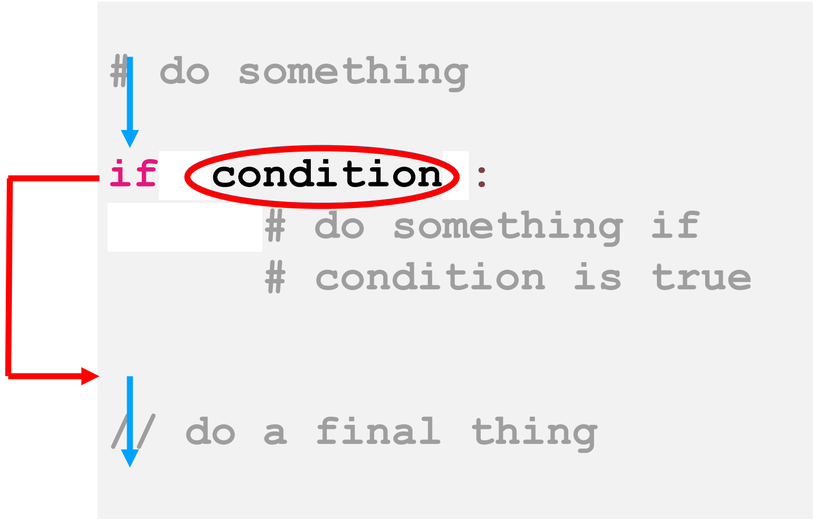
If-statement

Depending on a condition, some lines of code are executed or not.



If-statement

Depending on a condition, some lines of code are executed or not.



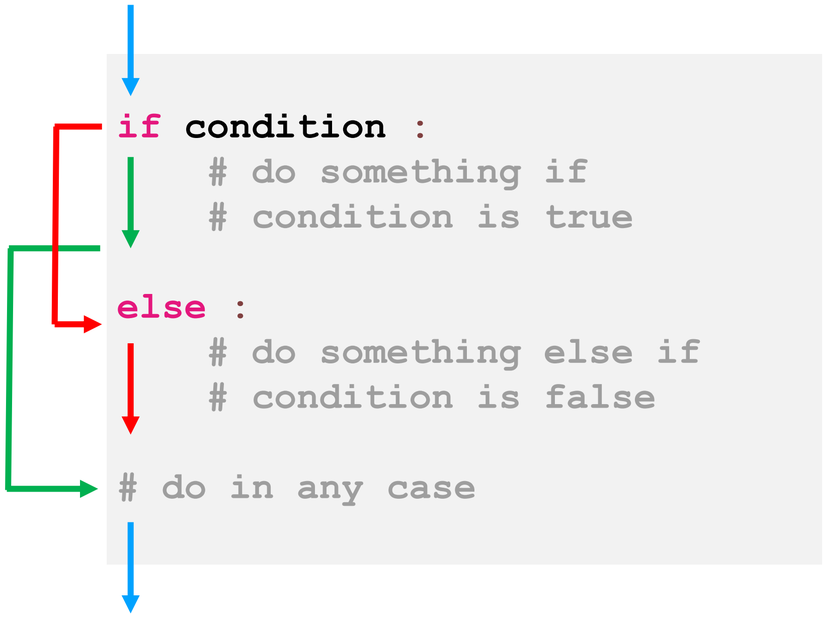
If-statement

The **if** / **elif** / **else** statement allows to program alternatives.

Depending on conditions, only one block is computed

Indentation is used to mark where a block starts and ends.

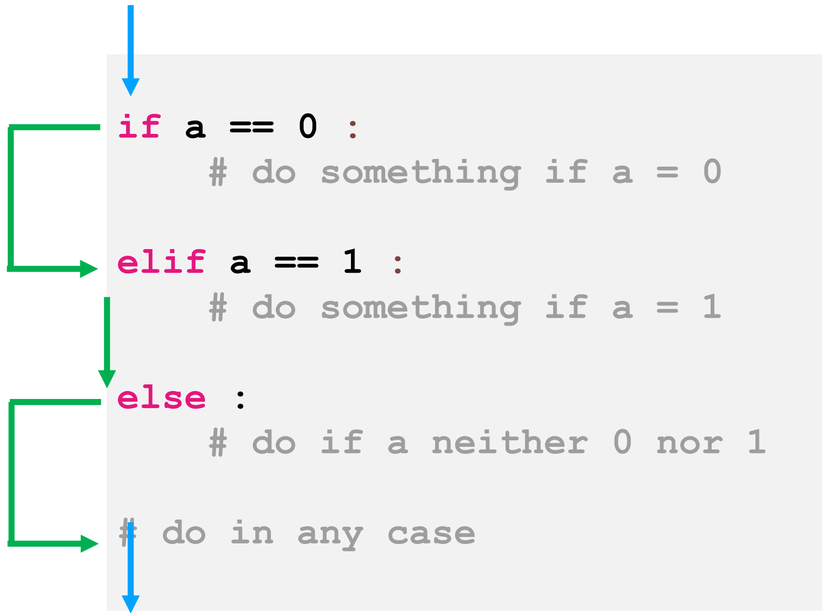
Indentation helps reading blocks,



```
if condition :
    # do something if
    # condition is true

else :
    # do something else if
    # condition is false

# do in any case
```



```
if a == 0 :
    # do something if a = 0

elif a == 1 :
    # do something if a = 1

else :
    # do if a neither 0 nor 1

# do in any case
```

If-statement

PoL Physics of Life TU Dresden

Comparison operators always have True (1) or False (0) as results.

```
# initialise program
quality = 99.5

# evaluate result
if quality > 99.9 :
    print("Everything is fine.")
else :
    print("We need to improve!")
```

```
In [1]: a = 4

        if a = 5:
            print("Hello world")

  File "<ipython-input-1-13fb587c9332>", line 3
    if a = 5:
         ^
SyntaxError: invalid syntax
```

Note: These are two equal signs!

Operator	Description	Example
<, <=	smaller than, smaller or equal to	a < b
>, >=	greater than, greater or equal to	a > b
==	equal to	a == b
!=	not equal to	a != 1

Combined conditions

PoL
Physics of Life
TU Dresden

Logic operators always take conditions as operands and result in a condition.

- **and**
- **or**
- **not**

Also combined conditions can be either True (1) or False (0).

```
# initialise program
quality = 99.9
age = 3

if quality >= 99.9 and age > 5 :
    print("The item is ok.")
```

```
# initialise program
quality = 99.9


if not quality < 99.9 :
    print("The item is ok.")
```

Conditions with arrays

PoL Physics of Life TU Dresden

Checking contents of lists can be done intuitively using the `in` statement

```
# initialise program
my_list = [1, 5, 7, 8]
item = 3

if item in my_list :
    print("The item is in the list.")
else :
    print("There is no", item, "in", my_list )
```

```
# initialise program
my_list = [1, 5, 7, 8]
item = 3

if item not in my_list :
    print("There is no", item, "in", my_list )
else :
    print("The item is in the list.")
```

Readable code

- Every command belongs on its own line
- Insert empty lines to separate important processing steps
- Put spaces between operators and operands, because:

  This is easier to read thanthat,orisnt'it?

- Indent every conditional block (if/else) using the TAB key

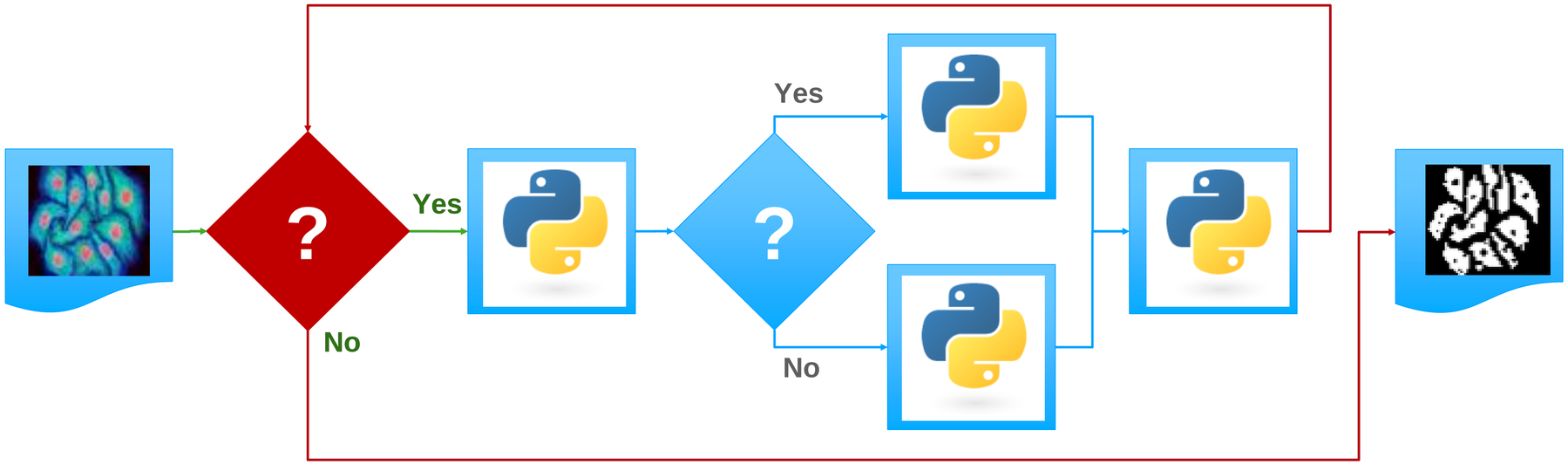
```
# initialise program
a = 5
b = 3
c = 8

# execute algorithm
d = (a + b) / c

# evaluate result
if a == 5 :
    print("Yin")
else :
    print("Yang")
```

Loops

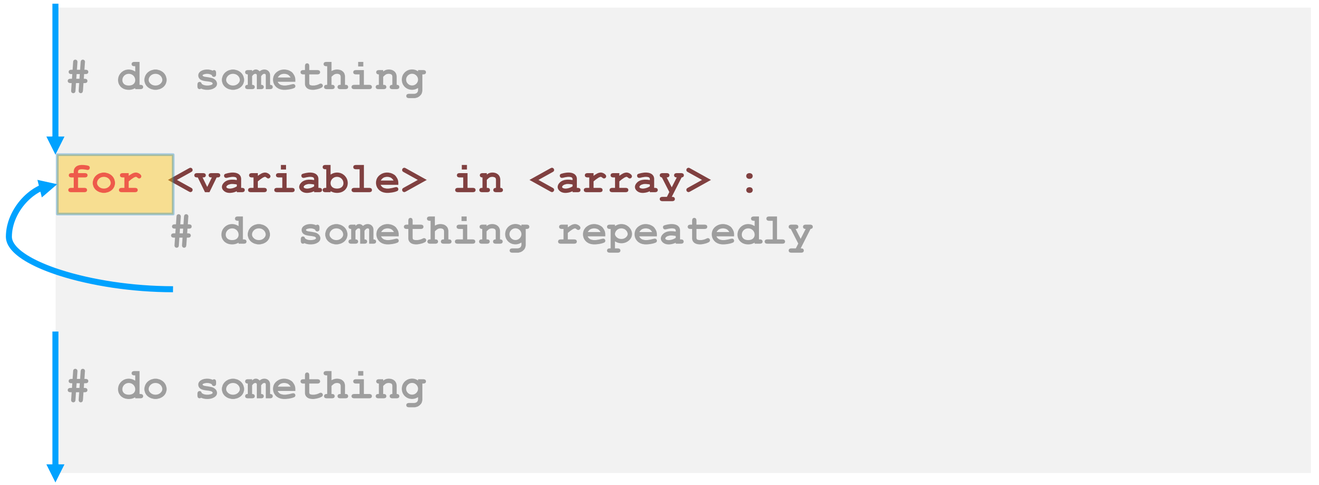
To repeat actions, you run code in loops



Loop statement

Loops

The `for` statement allows us to execute some lines of code *for* several times, typically for all items in an array-like thing (lists, tuples, images)



For-loops

Example list : `range(start, stop, step)`

```
for i in range(0, 5):
    print(i)
```

```
0
1
2
3
4
```

```
animal_set = ["Cat", "Dog", "Mouse"]

for animal in animal_set:
    print(animal)
```

```
Cat
Dog
Mouse
```

For-loops

- Indentation *means* combining operations to a block

```
for i in range(0, 5):
print(i)
```

```
  File "<ipython-input-15-59c457ae0ac9>", line 3
    print(i)
        ^
IndentationError: expected an indented block
```

Don’t forget to indent!

- Colon necessary

Don’t forget the colon!

```
for i in range(0, 5)
    print(i)
```

```
  File "<ipython-input-13-23157c0ed137>", line 2
    for i in range(0, 5)
                       ^
SyntaxError: invalid syntax
```

Generating arrays within for-loops

PoL Physics of Life TU Dresden

There is a long and a short way for creating arrays with numbers.

```python
# we start with an empty list
numbers = []

# and add elements
for i in range(0, 5):
    numbers.append(i * 2)

print(numbers)
```

```python
numbers = [i * 2 for i in range(0, 5)]

print(numbers)
```

```
[0, 2, 4, 6, 8]
```

Generating arrays within for-loops

Also a combination with the if-statement is possible

```
# we start with an empty list
numbers = []

# and add elements
for i in range(0, 5):
    # check if the number is odd
    if i % 2:
        numbers.append(i * 2)

print(numbers)
```

```
[2, 6]
```

```
numbers = [i * 2 for i in range(0, 5) if i % 2]

print(numbers)
```

```
[2, 6]
```

Listing files in a folder

PoL Physics of Life TU Dresden

Common use-case: do something with all *image* files in a folder

```
for file in file_list:
    if file.endswith(".tif"):
        print(file)
```

```
banana0002.tif
banana0003.tif
banana0004.tif
banana0005.tif
banana0006.tif
```

```
image_file_list = [file for file in file_list if file.endswith(".tif")]

image_file_list
```

```
['banana0002.tif',
 'banana0003.tif',
 'banana0004.tif',
 'banana0005.tif',
 'banana0006.tif',
```

While-loops

While loops keep executing indented code as long as a condition is met:

```
number = 1024

while (number > 1):
    number = number / 2
    print(number)
```

Works the same as with the if statement

```
512.0
256.0
128.0
64.0
32.0
16.0
8.0
4.0
2.0
1.0
```

Executing loops

Using the break statement, you can leave a loop

```
number = 1024

while (True):
    number = number / 2
    print(number)

    if number < 1:
        break
```

```
512.0
256.0
128.0
64.0
32.0
16.0
8.0
4.0
2.0
1.0
0.5
```

Skipping iterations

PoL Physics of Life TU Dresden

The `continue` statement allows to skip iterations

```
for i in range(0, 10):
    if i >= 3 and i <= 6:
        continue
    print(i)
```

```
0
1
2
7
8
9
```

Functions

- In case repetitive tasks appear that cannot be handled in a loop, custom functions are the way to go.
- Functions allow to re-use code in different contexts.
- Indentation is crucial.
- Functions must be defined before called

Definition:

```
def sum_numbers(a, b):
    result = a + b
    return result
```

name (parameters)

body

return statement (optional)

Call:

```
c = sum_numbers(4, 5)
print(c)
```

```
9
```

```
sum_numbers(5, 6)
```

```
11
```

```
sum_numbers(3, 4)
```

```
7
```

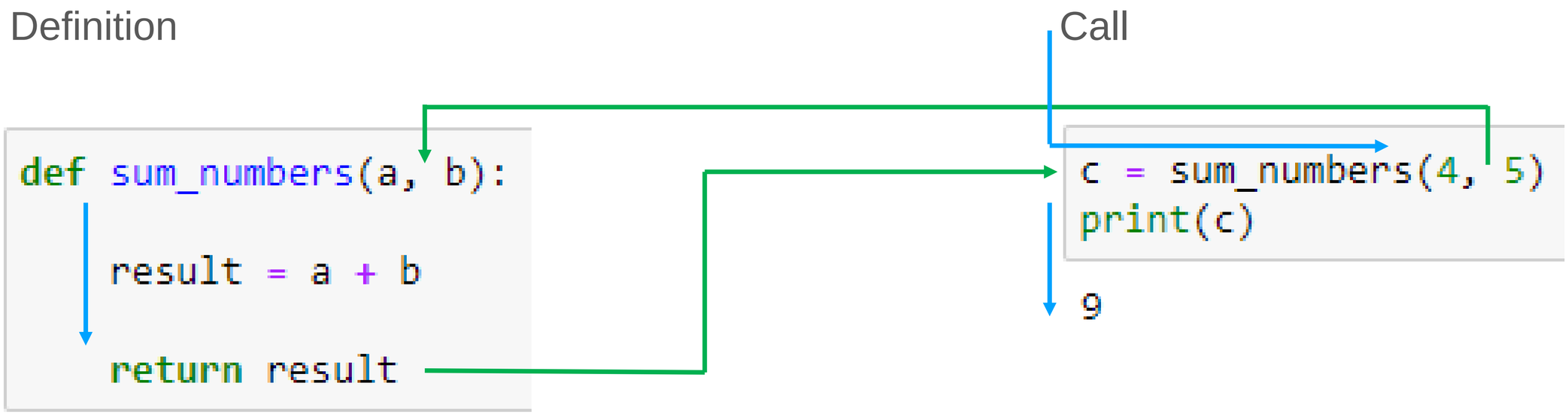
Functions

In case repetitive tasks appear that cannot be handled in a loop, custom functions are the way to go.

Functions allow to re-use code in different contexts.

Indentation is crucial.

Functions must be defined before called



Functions

Document your functions to keep track of what they do.

Describe what the functions does and what the parameters are meant to be

```
def square(number):
    '''
    Squares a number by multiplying it with itself  and returns its result.
    '''

    return number * number
```

- You can then later print the *documentation* if you can't recall how a function works.

```
square?
```

```
Signature: square(number)
Docstring: Squares a number by multiplying it with itself and returns its
result.
```

Libraries

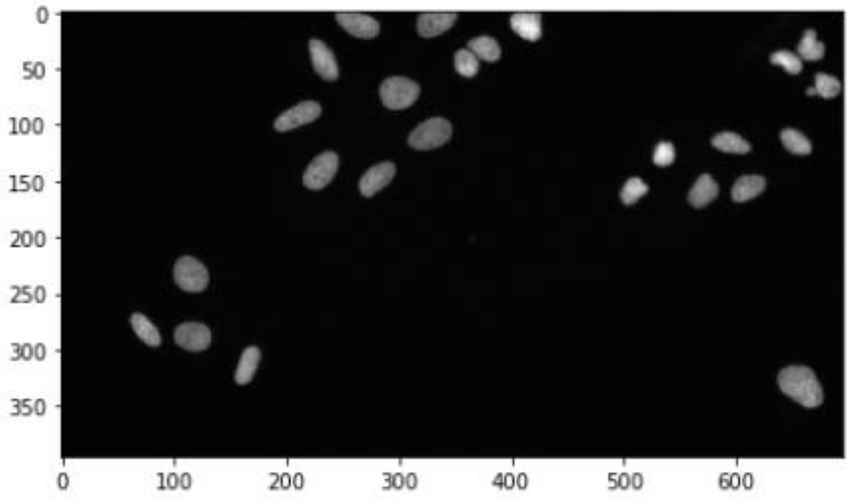
- The import statement allows to use functions provided by others.
- Commonly put at the beginning of a notebook or script to make sure everything is installed.

```
from skimage.io import imread
import matplotlib.pyplot as plt
import pyclesperanto_prototype as cle
```

After `cle` has been imported…

```
cle.imshow(input_crop, plot=axs[1])
```

… it can be used



Libraries

For re-using functions between notebooks / projects, use libraries. -> Sustainability

Simple python libraries are .py files containing multiple functions.

The import statement allows you to import python files from the same folder.

```
from my_library import square
```

```
square(5)
```

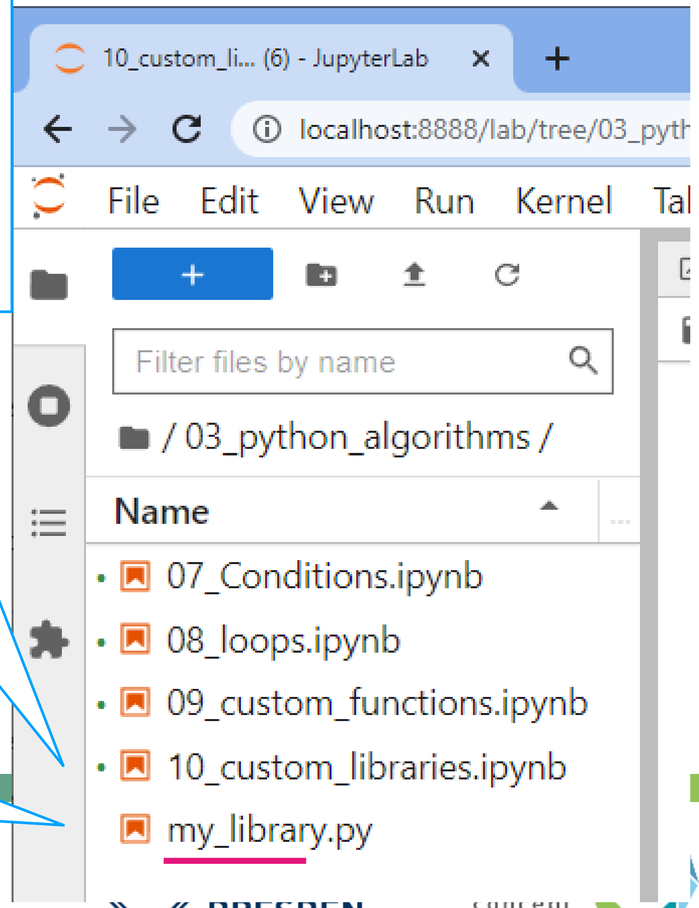
```
25
```

```
def wuzzle(number):
    '''
    The wuzzle function manipulates a number in a magic way and returns the result.
    '''

    import math
    return math.sqrt(number * 1.2)

def square(number):
    '''
    Squares a number by multiplying it with itself  and returns its result.
    '''

    return number * number
```



Outlook: The power of Python

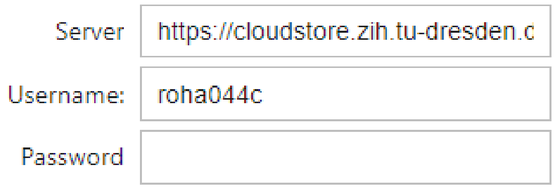
PoL Physics of Life TU Dresden

With Python, you can automate many tedious tasks. Example: Downloading files from the owncloud.

Build a login form

```
server_widget = widgets.Text(value='https://cloudstore.zih.tu-dresden.
username_widget = widgets.Text(description='Username:')
password_widget = widgets.Password(description='Password')

widgets.VBox([server_widget, username_widget, password_widget])
```



Log in

```
oc = owncloud.Client(server_widget.value)
oc.login(username_widget.value, password_widget.value)
```

NEVER save your password in Python code!

List all files in the owncloud

```
# enter a folder on the owncloud drive that exists.
remote_folder = "/"

for f in oc.list(remote_folder):
    print (f.path)
```

```
/BiAPoL/
/Documents/
/Nextcloud Manual.pdf
/Nextcloud intro.mp4
/Nextcloud.png
/Photos/
/Projects/
/Shared/
/Software/
```

Download a file

```
# enter the source file here
remote_source_file = '/Nextcloud Manual.pdf'
# enter the destination
local_file = 'Nextcloud Manual.pdf'

oc.get_file(remote_path=remote_source_file,
            local_file=local_file)
```

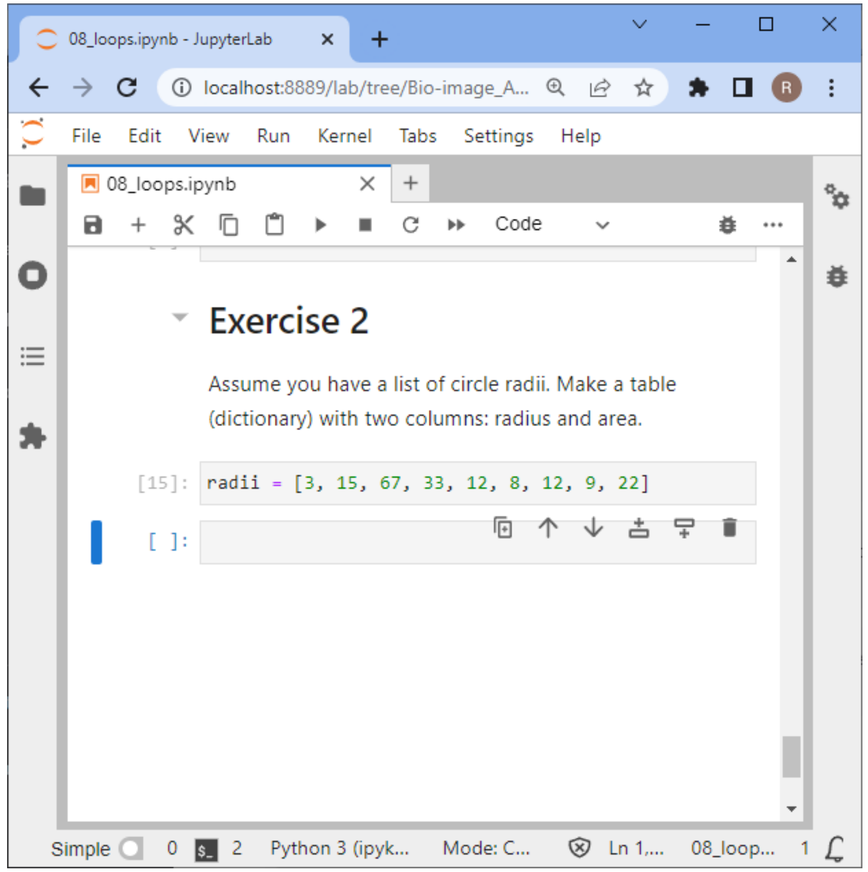
```
True
```

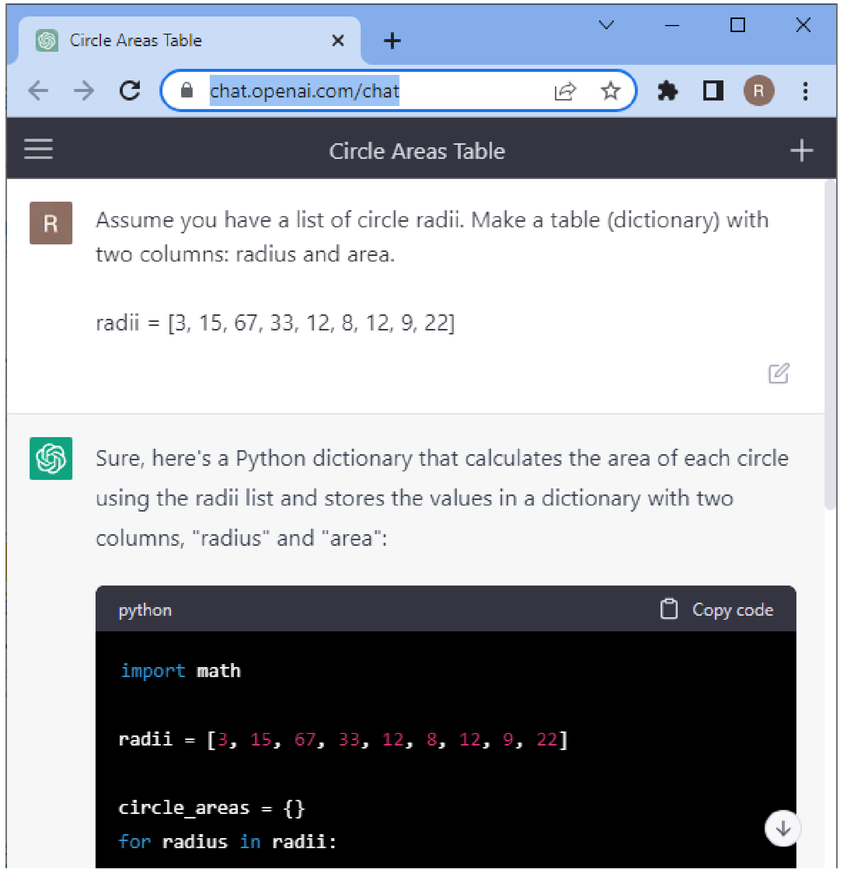
The PDF is now located in the same folder as this notebook.

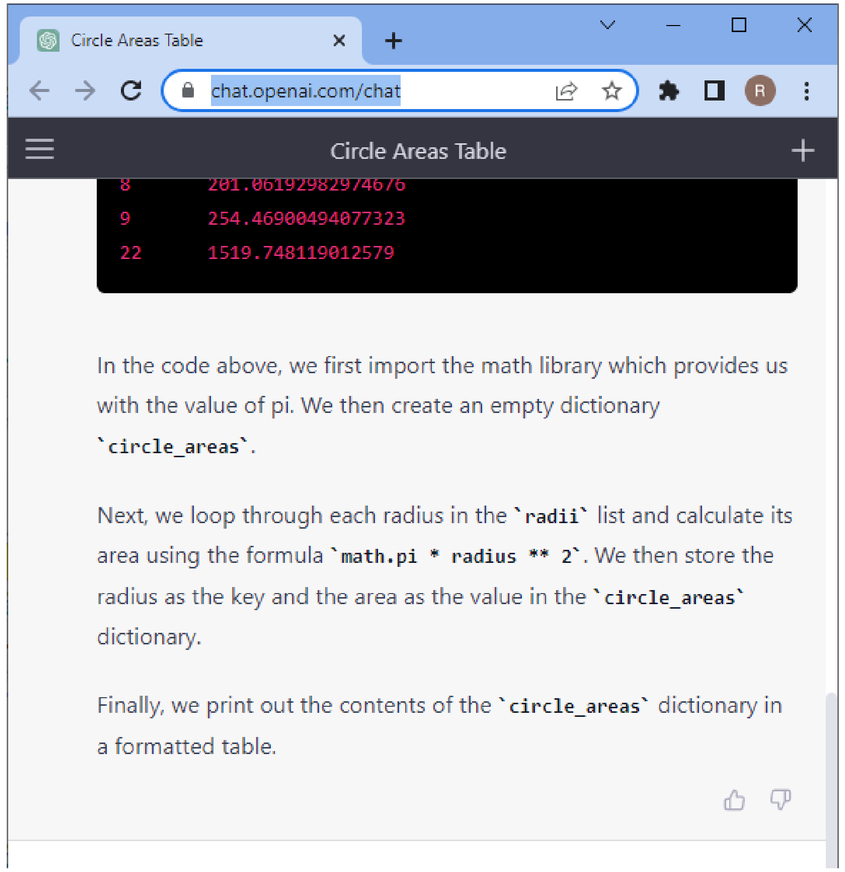
Take care: You can also DELETE all files in an owncloud folder using similar code

Outlook: The power of AI

Feel free to use artificial intelligence during the exercises. Play with it, learn how to *exploit it* best.
Consider: It *lies* from time to time and during the exam it can't help you.



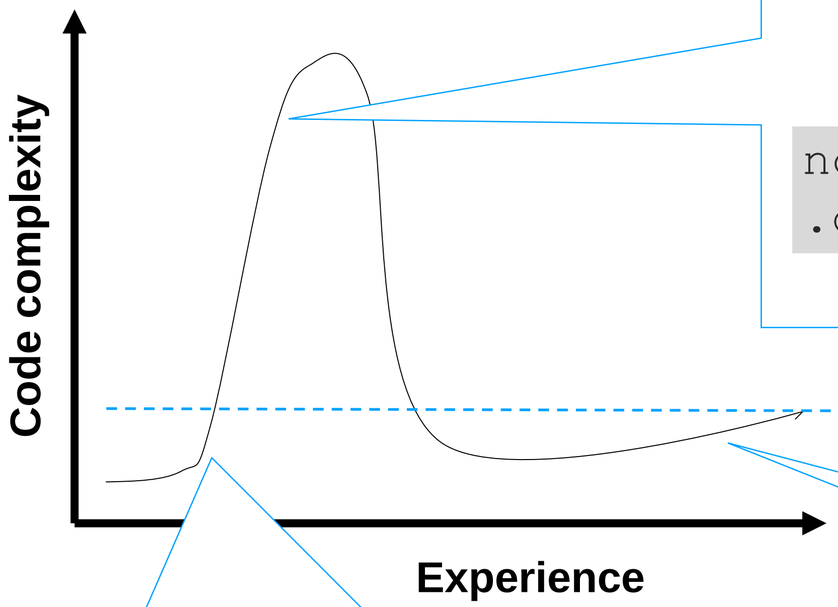




Software quality indicators

PoL Physics of Life TU Dresden

Interpretation of “High quality code” changes with experience



Jeey, I can write code like

```
non_square_numbers = len(set(range(1, 1001))
.difference(set(map(lambda x: x ** 2, range(1, 1001)))))
```

Good code lives below this line

Jeey, I can write code like

```
large_numbers = numbers[numbers > 100]
```

I prefer writing code like

```
large_numbers = numbers[numbers > 100]
```


https://towardsdatascience.com/how-to-write-high-quality-python-as-a-data-scientist-cde99f582675?gi=11843badbb14

https://en.wikipedia.org/wiki/Dunning%E2%80%93Kruger_effect

Summary

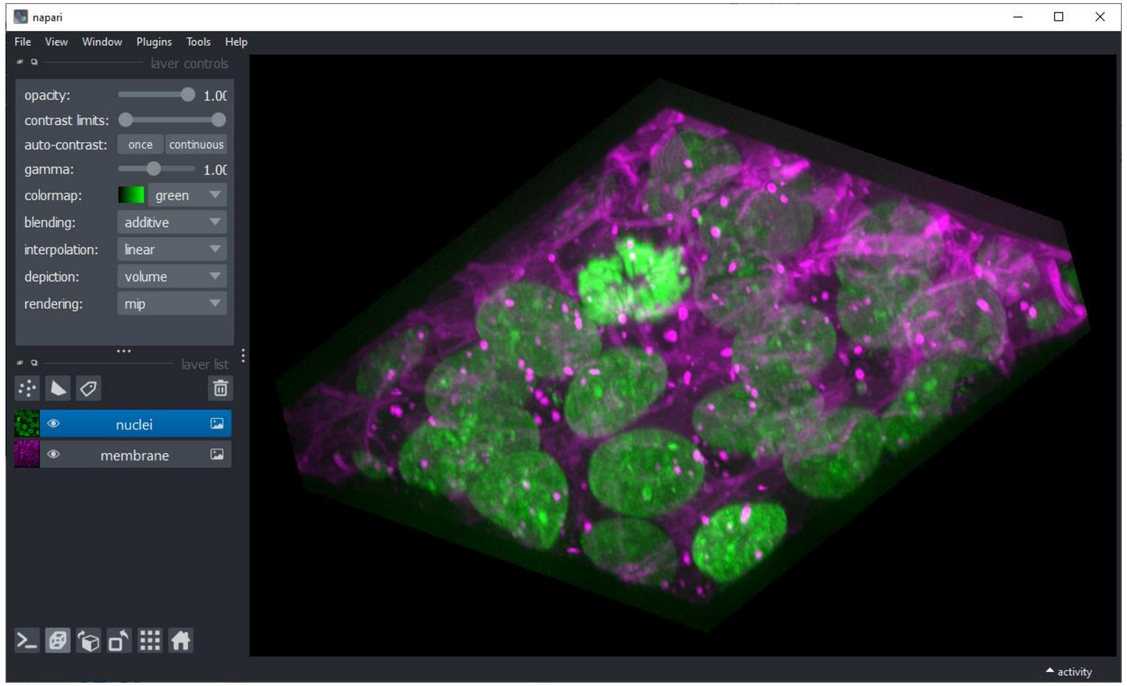
PoL Physics of Life TU Dresden

Today you learned:

- Document what *dependencies* you use!
- Sharing / licensing terminology
  - Levels of openness
  - Copyright holder / Author / Publisher / Licensee
  - FAIR principles (findable, accessible, interoperable, reusable)
- Python algorithms
  - Loops
  - Conditions
  - Functions
  - Libraries

Coming up next:

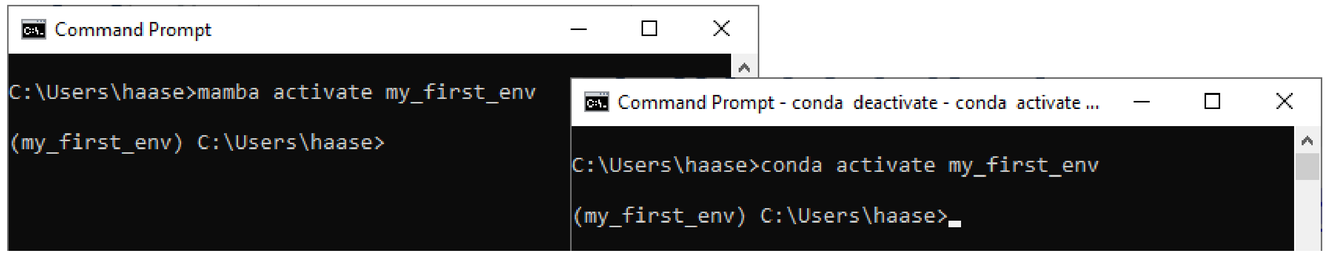
- Image processing
- Image filtering
- Napari



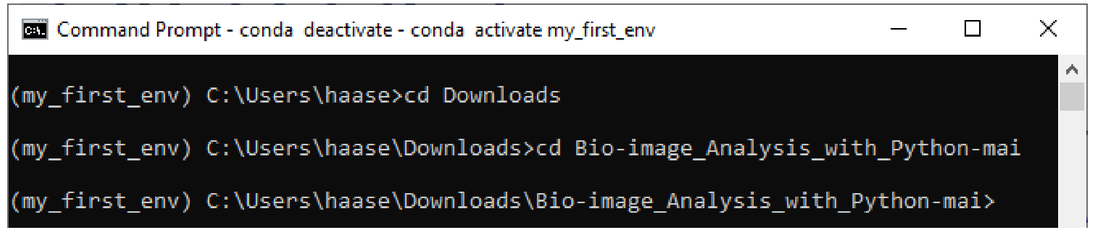
Exercises

- Check the `02_python_algorithms` folder online https://github.com/BiAPoL/Bio-image_Analysis_with_Python

1. Start up a terminal
2. Activate the environment using
   `conda activate my_first_env`



3. Use the `cd` command to navigate to the exercise folder



4. Start up `jupyter lab`