

# Python: File handling & Image Visualization

Robert Haase

Funded by



Bundesministerium  
für Bildung  
und Forschung

SACHSEN



Diese Maßnahme wird gefördert durch die Bundesregierung  
aufgrund eines Beschlusses des Deutschen Bundestages.  
Diese Maßnahme wird mitfinanziert durch Steuermittel auf  
der Grundlage des von den Abgeordneten des Sächsischen  
Landtags beschlossenen Haushaltes.



# Quiz: Recap

## What would this Python code result in?

```
files = ['a.tif', 'b.btif', 'c.tif', 'd.tif']  
  
files[1]
```

'a.tif'



'b.tif'



'c.tif'



'd.tif'



# Quiz: Recap

## What would this Python code result in?

```
files = ['a.tif', 'b.btif', 'c.tif', 'd.tif']  
  
files[-1]
```

'a.tif'



'b.tif'



'c.tif'



'd.tif'



# Working with files in folders

Key-skill when it comes to automating data analysis tasks

```
[2]: # define the location of the folder to go through
     directory = 'data_banana/'

     # get a list of files in that folder
     file_list = os.listdir(directory)

     file_list
```

It's just a  
Python list!

```
[2]: ['banana0002.tif',
      'banana0003.tif',
      'banana0004.tif',
      'banana0005.tif',
      'banana0006.tif',
      'banana0007.tif',

      'banana0026.tif',
      'image_source.txt']
```

There are not  
just images  
inside

# Filtering file lists

To focus on image data, we need to filter the file list

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

```
image_file_list
```

```
[3]: ['banana0002.tif',  
      'banana0003.tif',  
      'banana0004.tif',  
      'banana0005.tif',  
      'banana0006.tif',  
      'banana0007.tif',  
      'banana0026.tif']
```

A for-loop in  
a single line

# Filtering file lists

To focus on image data, we need to filter the file list

```
[4]: # go through all files in the folder
      for file in file_list:
          # if the filename is of a tif-image, print it out
          if file.endswith(".tif"):
              print(file)
```

banana0002.tif

banana0003.tif

banana0004.tif

banana0005.tif

banana0006.tif

banana0007.tif

banana0026.tif

# Filtering file lists

To focus on image data, we need to filter the file list

```
[5]: # go through all files in the folder
for file in file_list:
    # if the filename is of a tif-image, print it out
    if file.endswith(".tif"):
        print(file)

    # store the image
    image = imread(directory + file)

    # show the image
    stackview.imshow(image)
```



# Side note: comments

Your code will become longer...

Consider structuring it and putting comments and empty lines in between.



```
for file in file_list:
    if file.endswith(".tif"):
        print(file)
        image = imread(directory + file)
        stackview.imshow(image)
```



```
# go through all files in the folder
for file in file_list:
    # if the filename is of a tif-image, print it out
    if file.endswith(".tif"):
        print(file)

    # store the image
    image = imread(directory + file)

    # show the image
    stackview.imshow(image)
```



# Image visualization

In Python there are many ways to visualize image data.  
We focus on Jupyter Notebook compatible ways for now.

Before visualizing image data, get an idea about the dataset.

```
[2]: image = imread("../day2.1_image_segmentation/data/BMP4blastocystC3-cropped_resampled_8bit.tif")  
image.shape
```

```
[2]: (86, 396, 393)
```

Image.shape  
tells you the  
dimensions of  
an image

Not all image  
viewers  
support 3D  
data

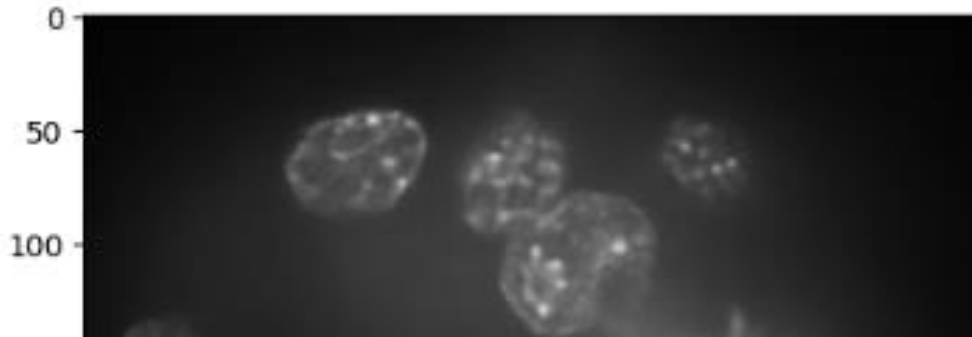
# Image visualization

scikit-image's `imshow()` (originating from matlab) can only visualize 2 dimensions.

```
from skimage.io import imread, imshow
```

```
[4]: center_slice = int(image.shape[0] / 2)
      imshow(image[center_slice])
```

```
[4]: <matplotlib.image.AxesImage at 0x2c9b9164820>
```



The first shape-dimension - 2 is the center plane in Z

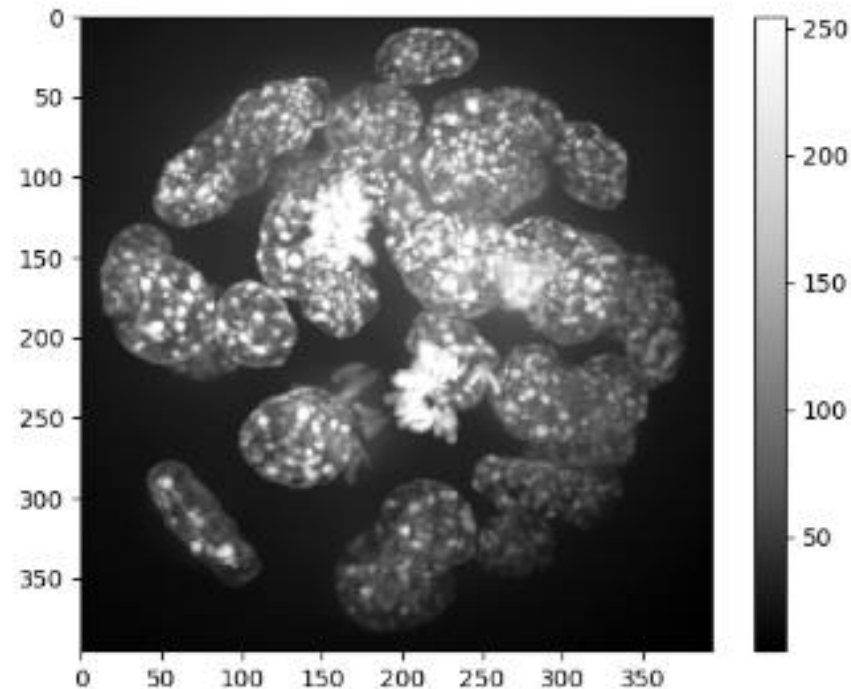
Image data in Python is commonly organized in Z-Y-X dimension order.

# Image visualization

Stackview can show 3D images (by applying a projection along Z)

```
import stackview
```

```
stackview.insight(image)
```



shape (86, 396, 393)

dimensions

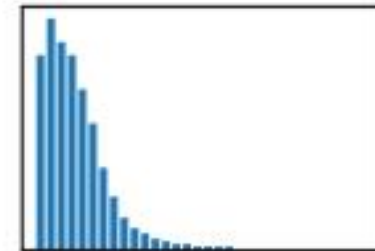
dtype uint8

size 12.8 MB

min 0

max 255

Basic statistics



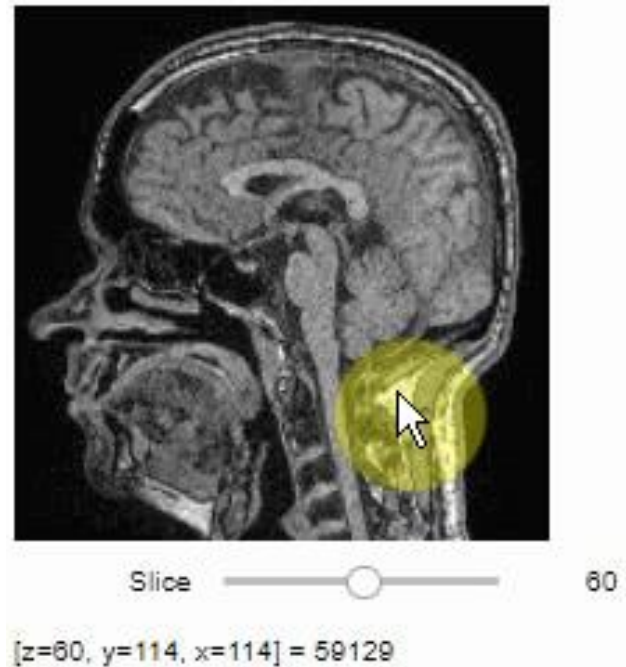
# Image visualization

Stackview also has some interactive tools

```
stackview.slice(image)
```

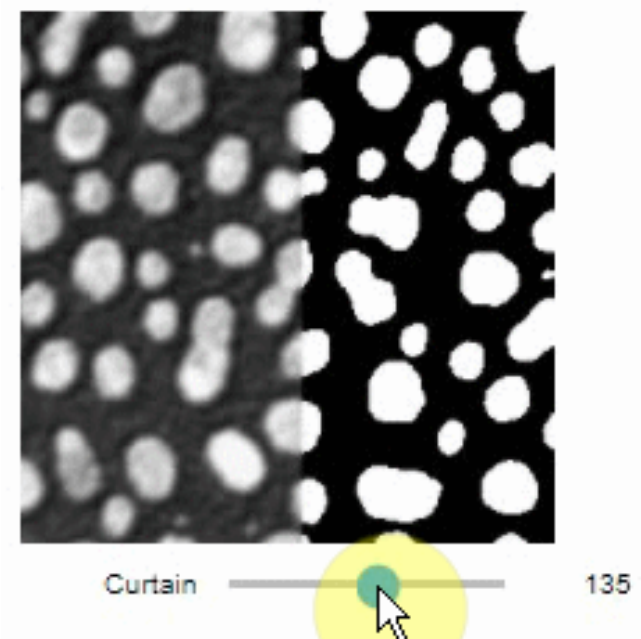


```
stackview.picker(image)
```



```
binary_image = image > 80
```

```
stackview.curtain(image, binary_image)
```



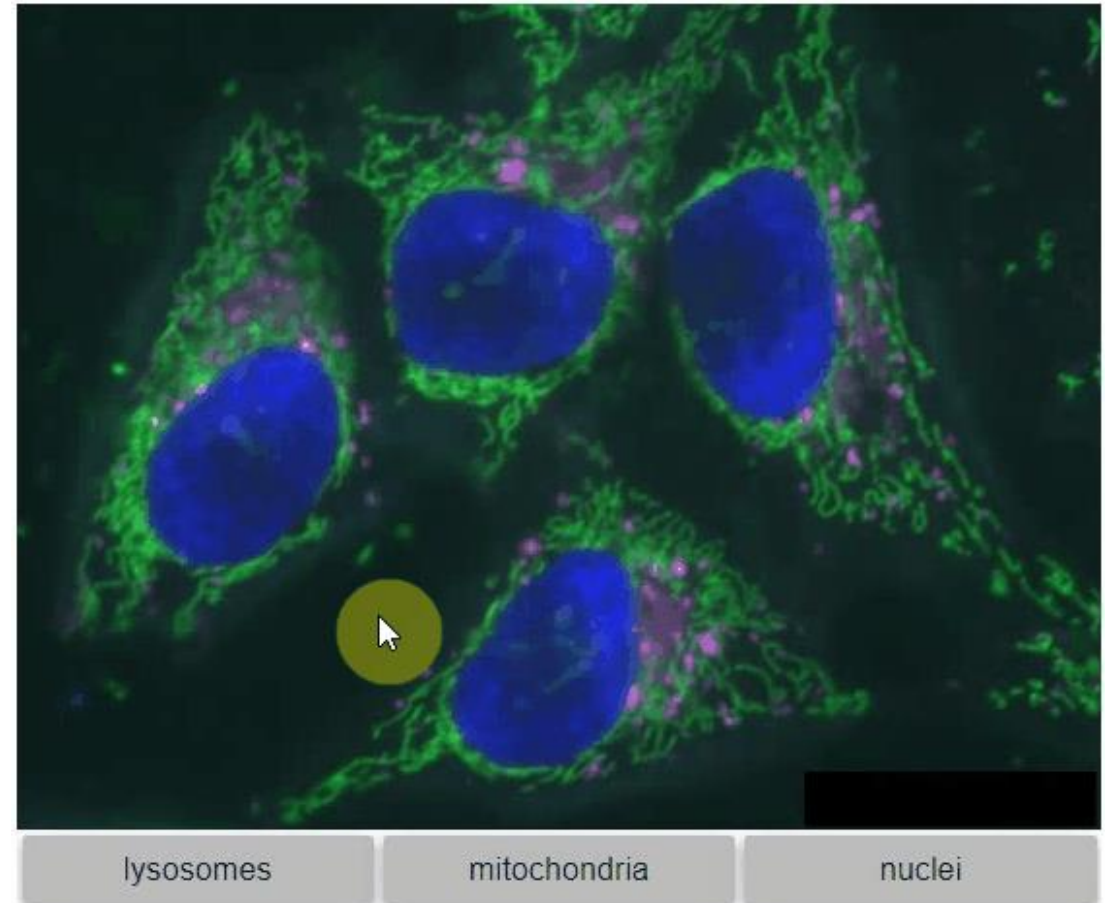
# Image visualization

Stackview also has some interactive tools

```
stackview.switch(  
    {  
        "Lysosomes": lysosomes_channel,  
        "Mitochondria": mitochondria_channel,  
        "Nuclei": nuclei_channel  
    },  
    colormap=[  
        "pure_magenta",  
        "pure_green",  
        "pure_blue"  
    ],  
    toggleable=True  
)
```

Identical but less readable:

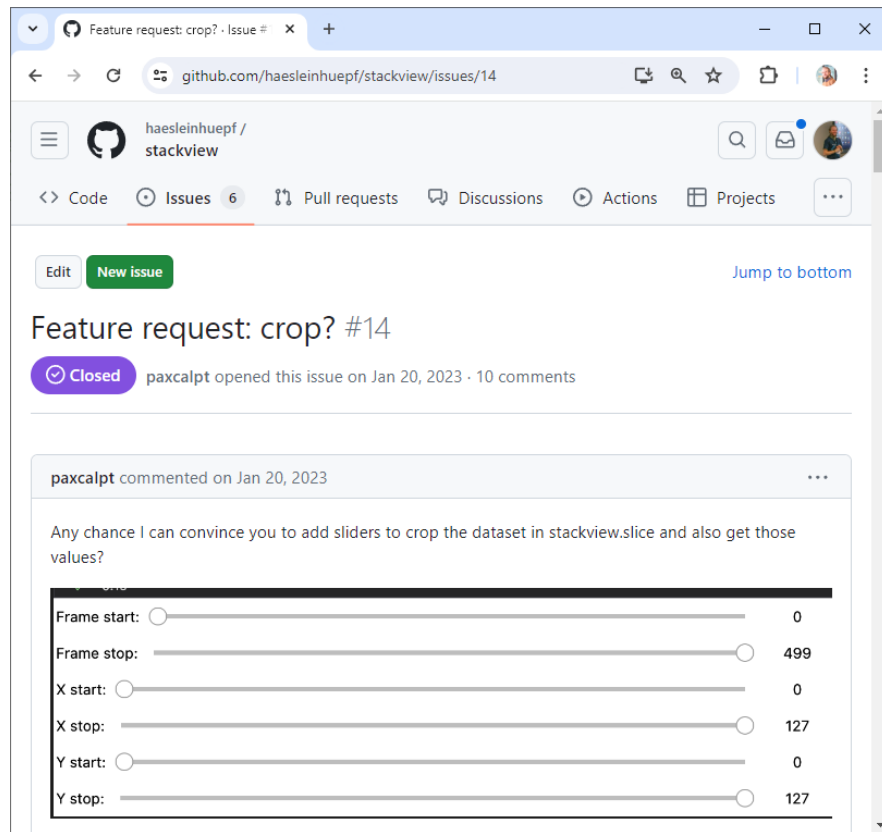
```
stackview.switch({"Lysosomes":lysosomes_channel,  
"Mitochondria":mitochondria_channel,"Nuclei":nuclei_channel},  
colormap=["pure_magenta", "pure_green", "pure_blue"], toggleable=True)
```



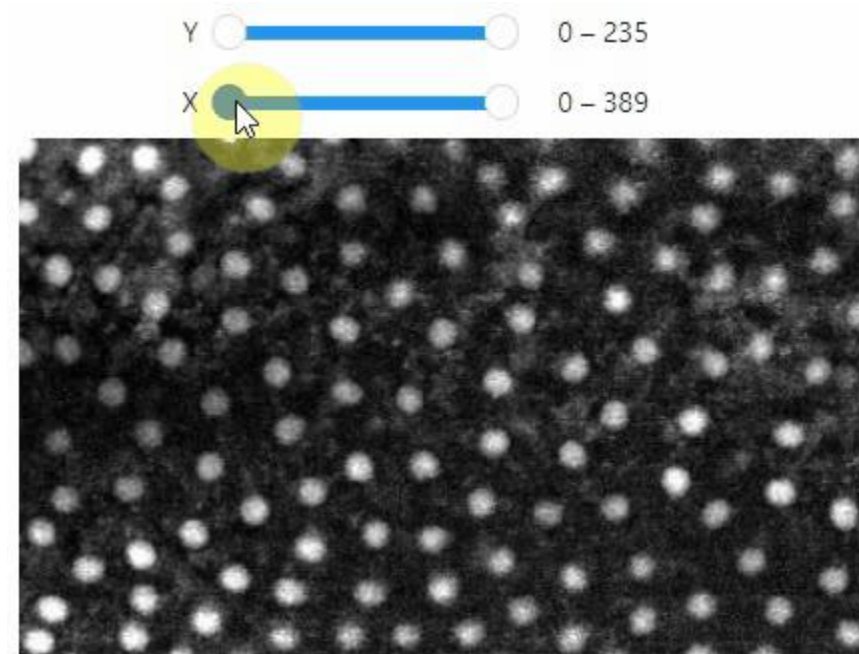


# Image Visualization

Stackview is developed for microscopists. If something doesn't work as you expect, communicate it :-)



```
crop_widget = stackview.crop(image_stack, continuous_update=True)
crop_widget
```



# Loading domain-specific file formats

Common Python libraries cannot open file formats such as OME-TIF, CZI, LIF, ND2... Solution: aicsimageio

From the Allen Institute for Cell Science

```
from aicsimageio import AICSImage
```

```
aics_image = AICSImage("data/EM_C_6_c0.ome.tif")  
aics_image
```

```
<AICSImage [Reader: OmeTiffReader, Image-is-in-Memory: False]>
```

# Loading domain-specific file formats

Aicsimageio supports dimension names and physical voxel sizes.

```
aics_image.shape
```

What's width, height,  
depth and time?

```
(1, 1, 256, 256, 256)
```

```
aics_image.dims
```

```
<Dimensions [T: 1, C: 1, Z: 256, Y: 256, X: 256]>
```

Key feature when  
working with big  
data!

Lazy loading (virtual stacks)

Allows processing files larger  
than computer memory.

```
np_image = aics_image.get_image_data("ZYX", T=0)  
np_image.shape
```

```
(256, 256, 256)
```



# Loading domain-specific file formats

Aicsimageio supports dimension names and physical voxel sizes.

```
aics_image.physical_pixel_sizes
```

```
PhysicalPixelSizes(Z=0.16784672897196262, Y=0.16776018346253663, X=0.16776018346253663)
```

```
def get_voxel_size_from_aics_image(aics_image):  
    return (aics_image.physical_pixel_sizes.Z,  
            aics_image.physical_pixel_sizes.Y,  
            aics_image.physical_pixel_sizes.X)
```

Such a helper-  
function may be  
different from  
project to project

```
get_voxel_size_from_aics_image(aics_image)
```

```
(0.16784672897196262, 0.16776018346253663, 0.16776018346253663)
```

# Loading domain-specific file formats

Common Python libraries cannot open file formats such as OME-TIF, CZI, LIF, ND2... Solution: aicsimageio

In case your file format is not supported, it gives hints what to install.

```
czi_image = AICSImage("data/PupalWing.czi")
czi_image.shape
```

```
czi_image = AICSImage("data/PupalWing.czi")
czi_image.shape
```

```
(1, 1, 80, 520, 692)
```

```
np_czi_image = czi_image.get_image_data("ZYX", T=0)
np_czi_image.shape
```

```
(80, 520, 692)
```

```
get_voxel_size_from_aics_image(czi_image)
```

```
(1.0, 0.20476190476190476, 0.20476190476190476)
```

```
Attempted file (C:/structure/code/BIDS-training-2024/day1.2_file_handling/data/PupalWing.czi) load with reader: aicsimageio.readers.czi_reader.CziReader failed with error: aicspylibczi is required for this reader. Install with `pip install 'aicspylibczi>=3.1.1' 'fsspec>=2022.7.1'`
```

```
Attempted file (C:/structure/code/BIDS-training-2024/day1.2_file_handling/data/PupalWing.czi) load with reader: aicsimageio.readers.bioformats_reader.BioformatsReader failed with error: bioformats_jar is required for this reader. Install with `pip install bioformats_jar` or `conda install bioformats_jar`
```

# Working with files in the cloud

Example nextcloud / owncloud

Install another Python library  
into your environment

```
[2]: import ipywidgets as widgets
import nextcloud_client
```

```
conda activate devbio-napari-env
pip install pyncclient
```

## Login-form

```
[3]: server_widget = widgets.Text(value='https://speicherwolke.uni-leipzig.de', description='Server')
username_widget = widgets.Text(description='Username:')
password_widget = widgets.Password(description='Password')

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

```
[3]:
```

Server	<input type="text" value="https://speicherwolke.uni-leipzig.d"/>
Username:	<input type="text"/>
Password	<input type="password"/>

## Actually logging in

```
[6]: ncc = nextcloud_client.Client(server_widget.value)
ncc.login(username_widget.value, password_widget.value)
```

# Working with files in the cloud

## Listing files in a remote folder

```
[7]: # enter a folder on the owncloud cloud
remote_folder = "/data/"

for f in ncc.list(remote_folder):
    print (f.path)

/data/blobs.tif
```

## Downloading a file

```
[8]: # enter the source file here
remote_source_file = '/data/blobs.tif'
# enter the destination
local_file = 'blobs.tif'

ncc.get_file(remote_path=remote_source_file,
             local_file=local_file)

[8]: True
```

## Uploading a file

```
[13]: ncc.put_file(remote_folder, local_file_to_upload)

[13]: True
```

## Listing files again

```
[14]: for f in ncc.list(remote_folder):
        print (f.path)

/data/blobs.tif
/data/blobs_labels.tif
```

# Quiz: Large Language Models

How often do you use ChatGPT or similar AI-based tools?

monthly



weekly



daily



hourly



# Quiz: Large Language Models

What do you use ChatGPT and similar AI-based tools for?

Emails



Paperwork



Code

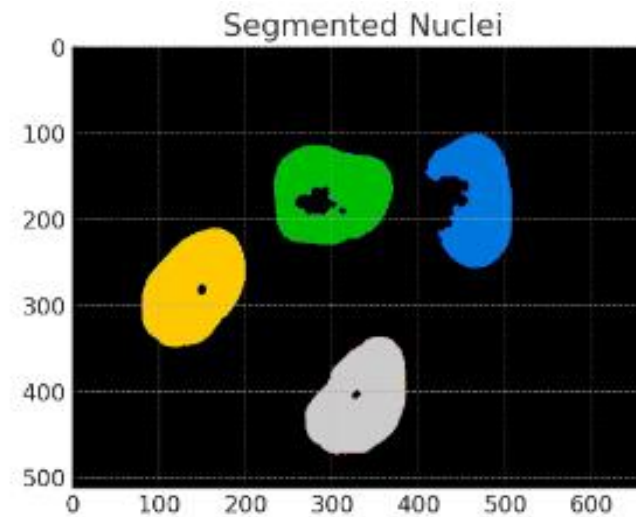
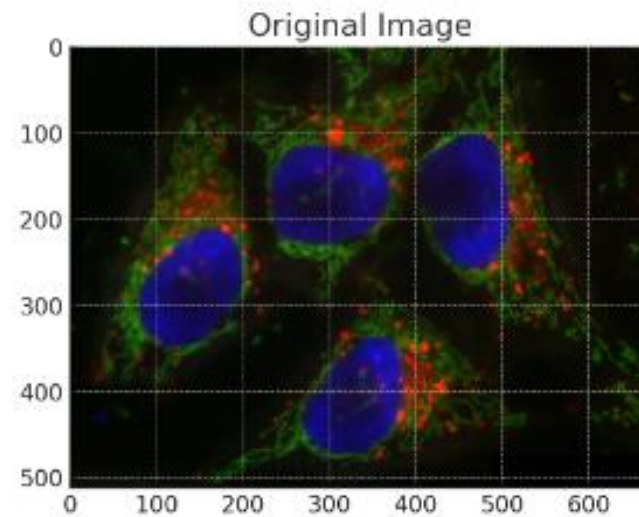


other



# Bio-image Analysis

My job ...



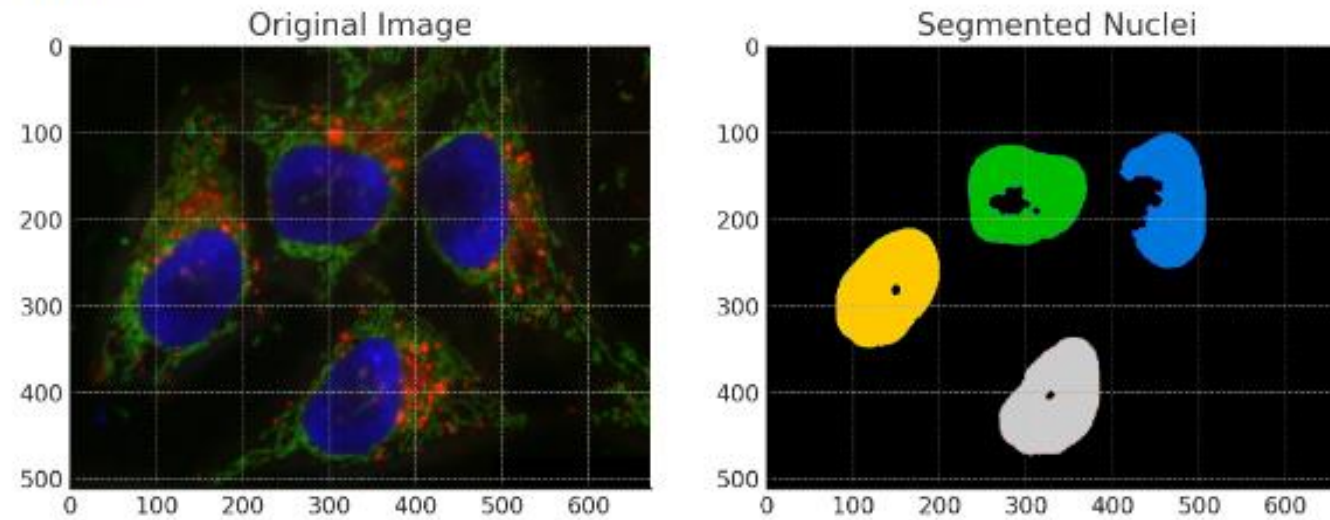


# Bio-image Analysis using Large-Language Models

My job is changing, since we have ChatGPT

Prompt:  
“Please segment  
the nuclei in the  
blue channel of  
the image”

ChatGPT



The segmentation of the nuclei has been performed on the blue channel of the image. The right side of the figure shows the segmented nuclei in different colors, each corresponding to a different nucleus in the field of view. Each color represents a unique label assigned to the individual nuclei detected in the image. [↗]

```
Analysis
from skimage import io, filters, measure, morphology
import matplotlib.pyplot as plt
import numpy as np

# Load the image
image = io.imread('/mnt/data/hela-cells-8bit.png')

# Since it's a color image, we'll just take the blue channel
blue_channel = image[:, :, 2]

# Threshold the image to get the binary image for nuclei
thresh = filters.threshold_otsu(blue_channel)
binary = blue_channel > thresh

# Remove small objects and perform morphological closing
cleaned = morphology.remove_small_objects(binary, 50)
closed = morphology.closing(cleaned, morphology.disk(3))

# Label the image
label_image = measure.label(closed)

# Plot the original image and the binary image
fig, ax = plt.subplots(1, 2, figsize=(12, 6))
ax[0].imshow(image)
ax[0].set_title('Original Image')
ax[1].imshow(label_image, cmap='nipy_spectral')
ax[1].set_title('Segmented Nuclei')
plt.show()
```



# Coding assistance: BiA-Bob

- Open-source alternative to OpenAI's Data Analyst
- No need to upload your [image] data
- Extensible through plugins

```
[1]: from bia_bob import bob  
     bob.__version__
```

```
[1]: '0.3.0'
```

```
[ ]:
```



With Christian  
Tischer (EMBL  
Heidelberg)  
@tischitischer

# Coding assistance: BiA-Bob

Use `%bob ...` to ask for small things.

In [2]:

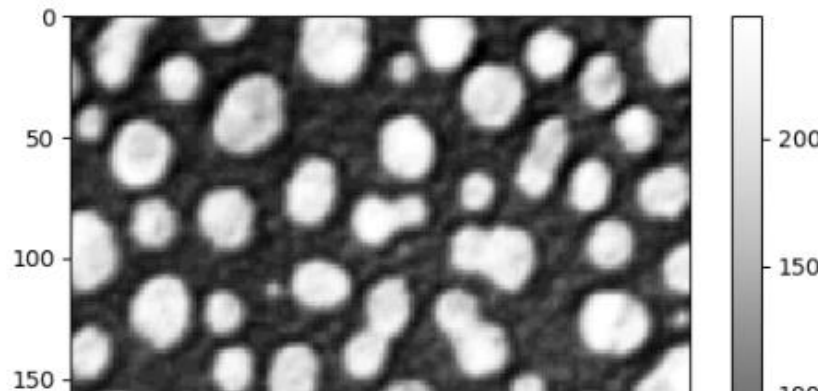
```
%bob Load the blobs.tif image file and show it.
```

In [3]:

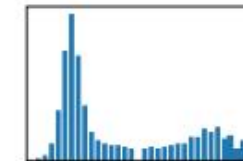
```
from skimage.io import imread
import stackview

image = imread('blobs.tif')
stackview.insight(image)
```

Out[3]:



shape	(254, 256)
dtype	uint8
size	63.5 kB
min	8
max	248



# Coding assistance: BiA-Bob

Use `%%bob ...` to ask for complex analysis tasks.

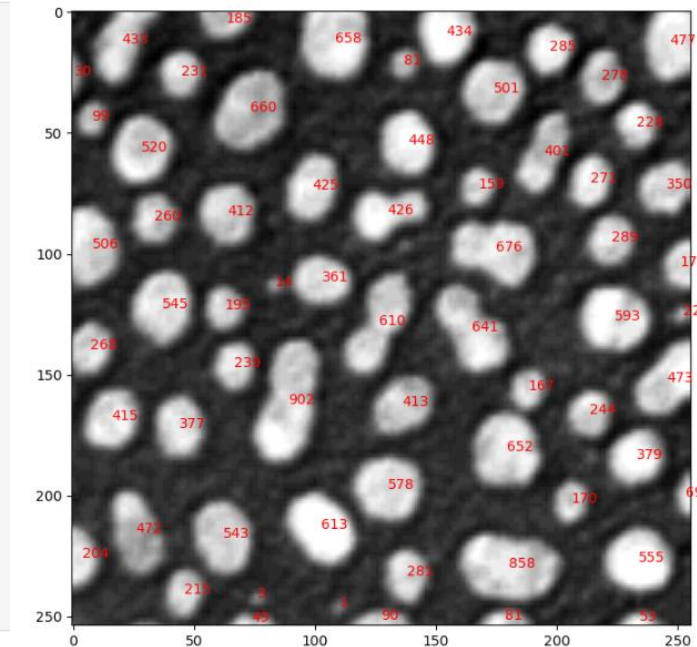
In [4]:

```
%%bob
```

```
Load the blobs.tif grayscale image from disc.  
Segment the nuclei-like objects in it using  
Otsu's method and connected component labeling.  
Measure area of the segmented labels.  
Print the area on top of the labels.
```

In [5]:

```
from skimage.io import imread  
from skimage.filters import threshold_otsu  
from skimage.measure import label, regionprops  
import matplotlib.pyplot as plt  
import stackview  
  
# Load the image  
image = imread('blobs.tif')  
  
# Apply Otsu's thresholding  
threshold = threshold_otsu(image)  
binary_image = image > threshold  
  
# Perform connected component labeling  
labels = label(binary_image)  
  
# Measure area of the segmented labels  
props = regionprops(labels)  
areas = [prop.area for prop in props]  
  
# Create an image with labels and area values on top  
fig, ax = plt.subplots(figsize=(8, 8))  
ax.imshow(image, cmap='gray')  
  
for prop in props:  
    y, x = prop.centroid  
    area = prop.area  
    ax.text(x, y, f"{area}", color='red')  
  
stackview.curtain(image, labels)
```



# Coding assistance: BiA-Bob

Use %%doc ... to ask for code explanations / comments

```
[1]: from bia_bob import doc
```

```
[ ]: %%doc  
from skimage.filters import threshold_otsu  
from skimage.measure import label  
threshold = threshold_otsu(image)  
binary = image > threshold  
labels = label(binary)
```

```
[ ]:
```

Click to add a cell.

# Coding assistance: BiA-Bob

Use `%%fix` ... to apply a bug-fix after an error happened

```
[ ]: from bia_bob import fix
```



```
[ ]: from skimage.io import imread  
image = imread(blobs.tif)
```

```
[ ]:
```

I

Click to add a cell.

# Coding assistance: BiA-Bob

- Bob adds a new cell to your code. If you ask it 3 times, it will make 3 cells.
- Clean up from time to time!

Click on the bar here and enter **dd**

```
[3]: %bob load blobs.tif and show it
```

We need to load an image named `blobs.tif` from the disk and display it.

```
[ ]: from skimage.io import imread
import stackview

# Load the image from disk
image = imread('blobs.tif')

# Display the image
stackview.insight(image)
```

```
[ ]: from skimage.io import imread
import stackview

# Load the image file
image = imread('blobs.tif')

# Display the image
stackview.insight(image)
```

```
[ ]: from skimage.io import imread
import stackview

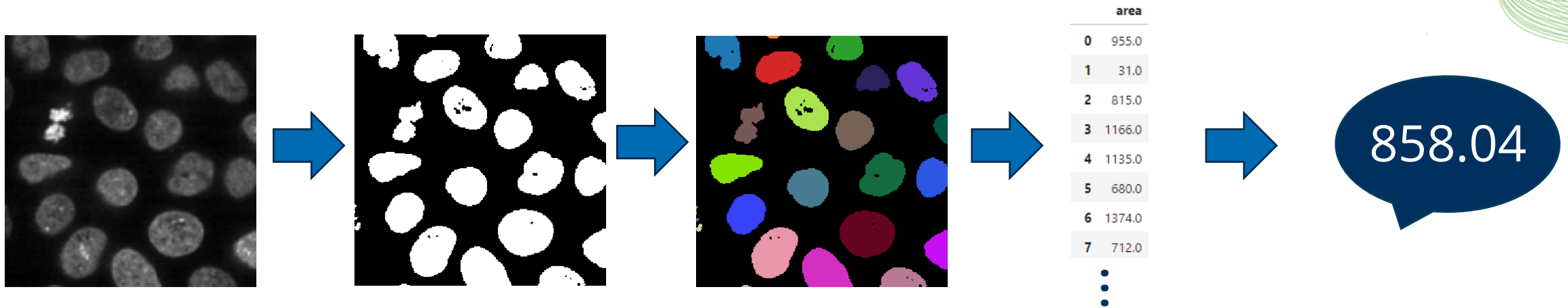
# Load image from disk
image_blobs = imread('blobs.tif')

# Display the image
stackview.insight(image_blobs)
```



# Benchmarking LLMs for Bio-image Analysis

Use case: segment the image and measure the average area of objects.



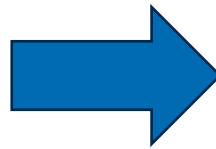
Unit-test pass-rate (n=10):

	reference	gpt-4-turbo-2024-04-09	Claude-3-opus-20240229	gpt-4-1106-preview	gpt-3.5-turbo-1106	gemini-pro	codellama
workflow_segmentation_measurement_summary	1.0	0.9	1.0	0.8	0.5	0.5	0.1

# Benchmarking LLMs for Bio-image Analysis

Use-case: correlation matrix

	a	b	c	d	e
0	1.600000	0.100000	1.600000	1.700000	1.700000
1	2.300000	0.200000	2.300000	2.400000	2.400000
2	2.600000	0.300000	2.600000	2.400000	2.400000
3	3.700000	0.300000	3.700000	3.600000	3.600000
4	3.400000	0.400000	3.400000	3.500000	3.500000
5	3.900000	0.400000	3.900000	3.900000	3.900000
6	4.300000	0.400000	4.300000	4.400000	4.400000
7	4.300000	0.500000	4.300000	4.200000	4.200000
8	4.000000	0.500000	4.000000	4.100000	4.100000
9	5.100000	0.500000	5.100000	5.000000	5.000000
10	5.200000	0.600000	5.200000	5.100000	5.100000
11	5.300000	0.600000	5.300000	5.400000	5.400000
12	5.500000	0.600000	5.400000	5.600000	5.600000



	a	b	c	d	e
a	1.000000	0.949504	0.999775	0.995800	0.995800
b	0.949504	1.000000	0.949594	0.946039	0.946039
c	0.999775	0.949594	1.000000	0.995001	0.995001
d	0.995800	0.946039	0.995001	1.000000	1.000000
e	0.995800	0.946039	0.995001	1.000000	1.000000

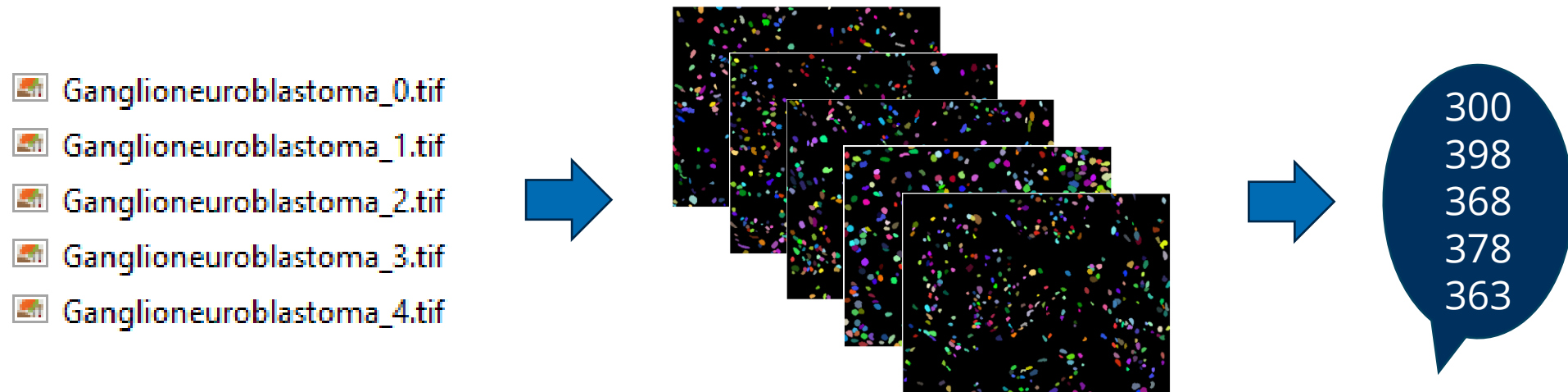
Unit-test pass-rate (n=10):

	reference	gpt-4-turbo-2024-04-09	Claude-3-opus-20240229	gpt-4-1106-preview	gpt-3.5-turbo-1106	gemini-pro	codellama
pair_wise_correlation_matrix	1.0	1.0	1.0	0.9	1.0	0.5	0.1



# Benchmarking LLMs for Bio-image Analysis

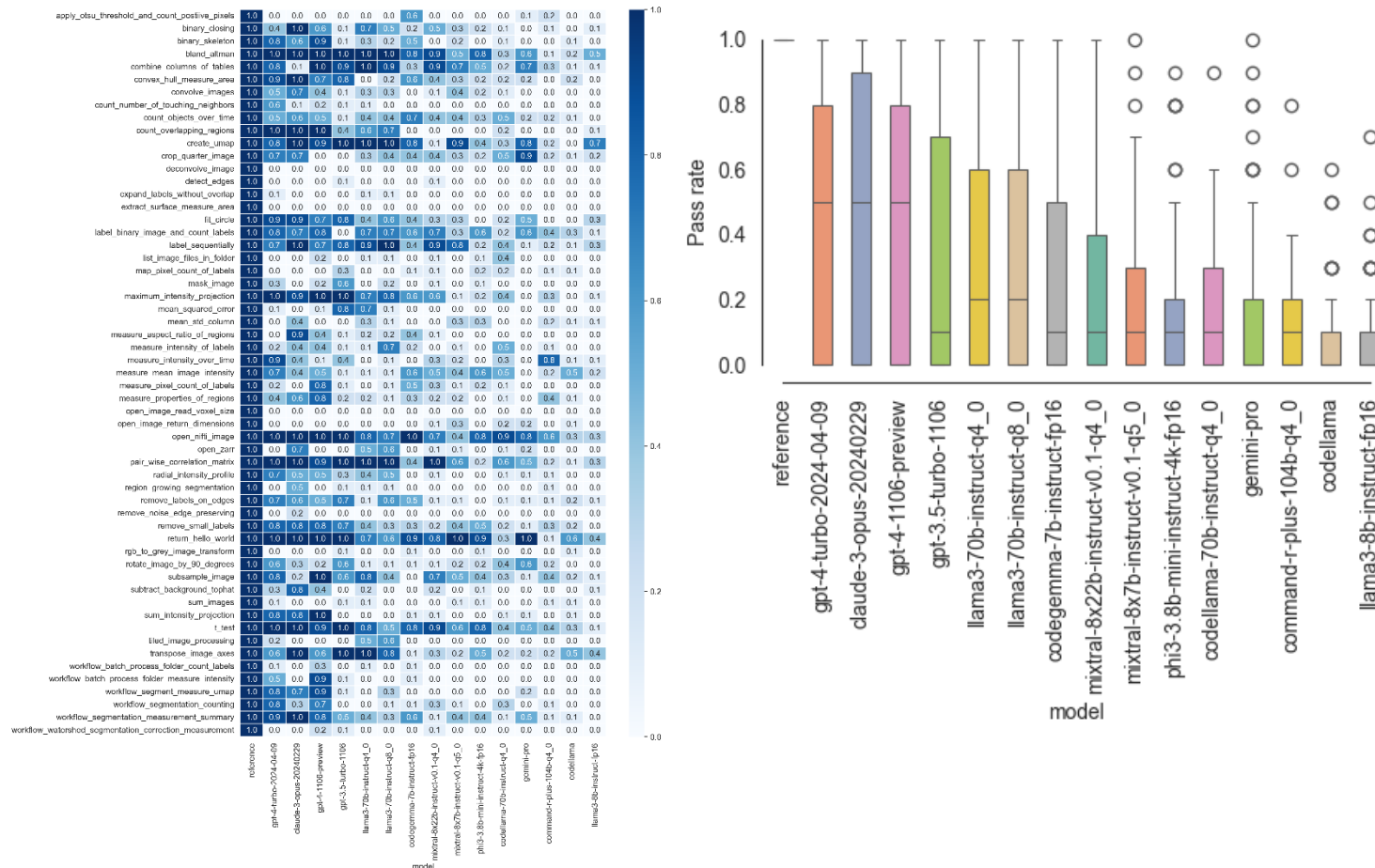
Use case: Count segmented objects in a folder of segmentation results.





Unit-test pass-rate (n=10):

	reference	gpt-4-turbo-2024-04-09	Claude-3-opus-20240229	gpt-4-1106-preview	gpt-3.5-turbo-1106	gemini-pro	codellama
workflow_batch_process_folder_count_labels	1.0	0.1	0.0	0.3	0.0	0.0	0.0

Summary: 57 use-cases (yet), 15 LLMs (yet), n=10









THE PREPRINT SERVER FOR BIOLOGY

New Results








[Follow this preprint](#)

## Benchmarking Large Language Models for Bio-Image Analysis Code Generation

 Robert Haase,  Christian Tischer,  Jean-Karim Hériché,  Nico Scherf

doi: <https://doi.org/10.1101/2024.04.19.590278>

This article is a preprint and has not been certified by peer review [what does this mean?].


 0  0  0  0  0  0  8

Abstract

Full Text

Info/History

Metrics

 Preview PDF

# Coding assistance: BiA-Bob

Keep your feed on the ground with *Bob*. Bob can do crazy things, but you are responsible for what it does with your data.

Do not enter personal / private information. What you enter will be sent to the server of an american company.

# Exercises

## Robert Haase

Funded by



Bundesministerium  
für Bildung  
und Forschung

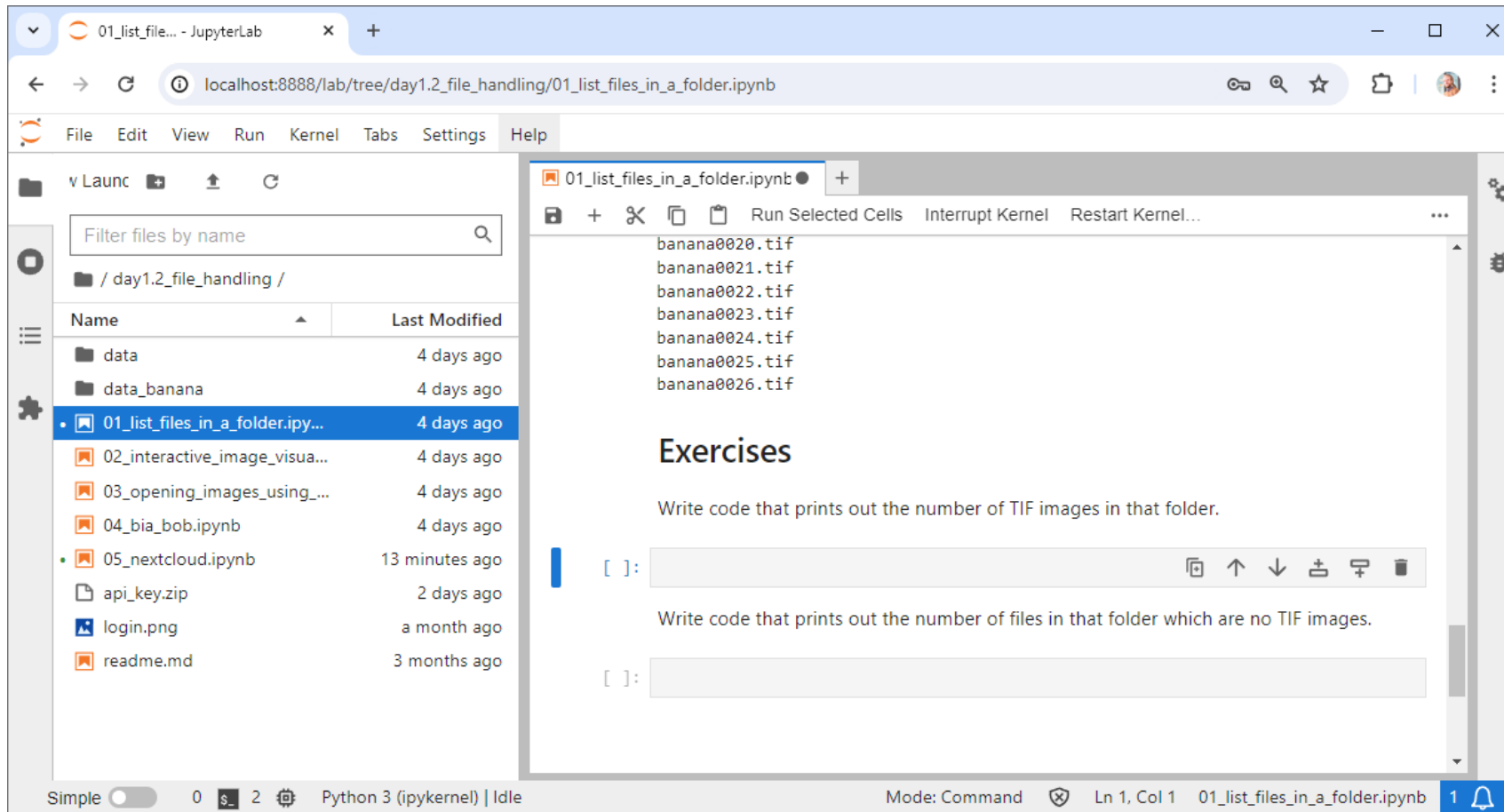
SACHSEN



Diese Maßnahme wird gefördert durch die Bundesregierung  
aufgrund eines Beschlusses des Deutschen Bundestages.  
Diese Maßnahme wird mitfinanziert durch Steuermittel auf  
der Grundlage des von den Abgeordneten des Sächsischen  
Landtags beschlossenen Haushaltes.

# Exercise: File lists and folder

Apply your knowledge about Python lists to list of files.



The screenshot shows a JupyterLab environment. On the left, a file browser displays the directory structure of a project. The main pane on the right shows a Jupyter notebook with a list of files and two exercise prompts.

**File Browser (Left Pane):**

- Filter files by name
- Directory: / day1.2\_file\_handling /
- Table with columns: Name, Last Modified
- Files and folders listed:

Name	Last Modified
data	4 days ago
data_banana	4 days ago
01_list_files_in_a_folder.ipynb	4 days ago
02_interactive_image_visua...	4 days ago
03_opening_images_using_...	4 days ago
04_bia_bob.ipynb	4 days ago
05_nextcloud.ipynb	13 minutes ago
api_key.zip	2 days ago
login.png	a month ago
readme.md	3 months ago

**Code Editor (Right Pane):**

01\_list\_files\_in\_a\_folder.ipynb

```
banana0020.tif  
banana0021.tif  
banana0022.tif  
banana0023.tif  
banana0024.tif  
banana0025.tif  
banana0026.tif
```

### Exercises

Write code that prints out the number of TIF images in that folder.

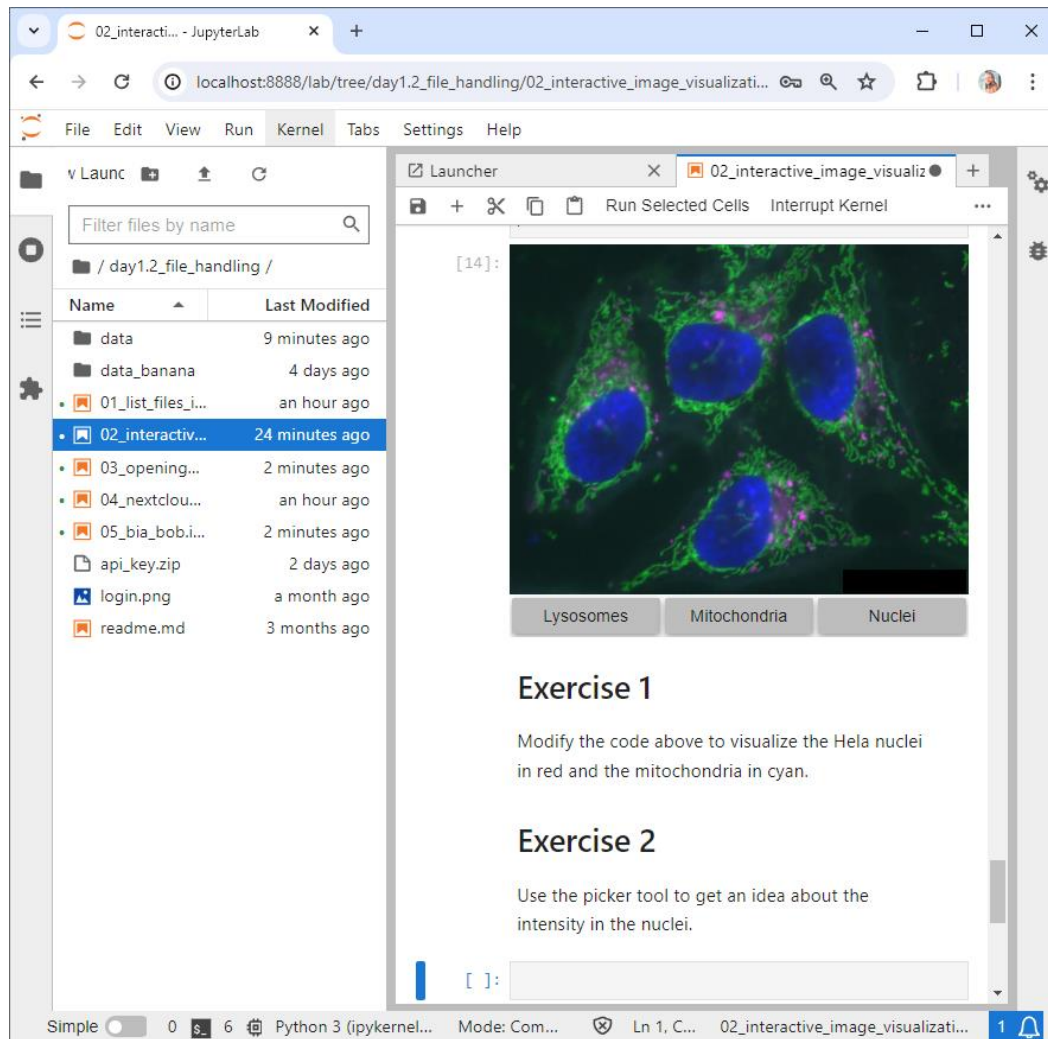
[ ]:

Write code that prints out the number of files in that folder which are no TIF images.

[ ]:



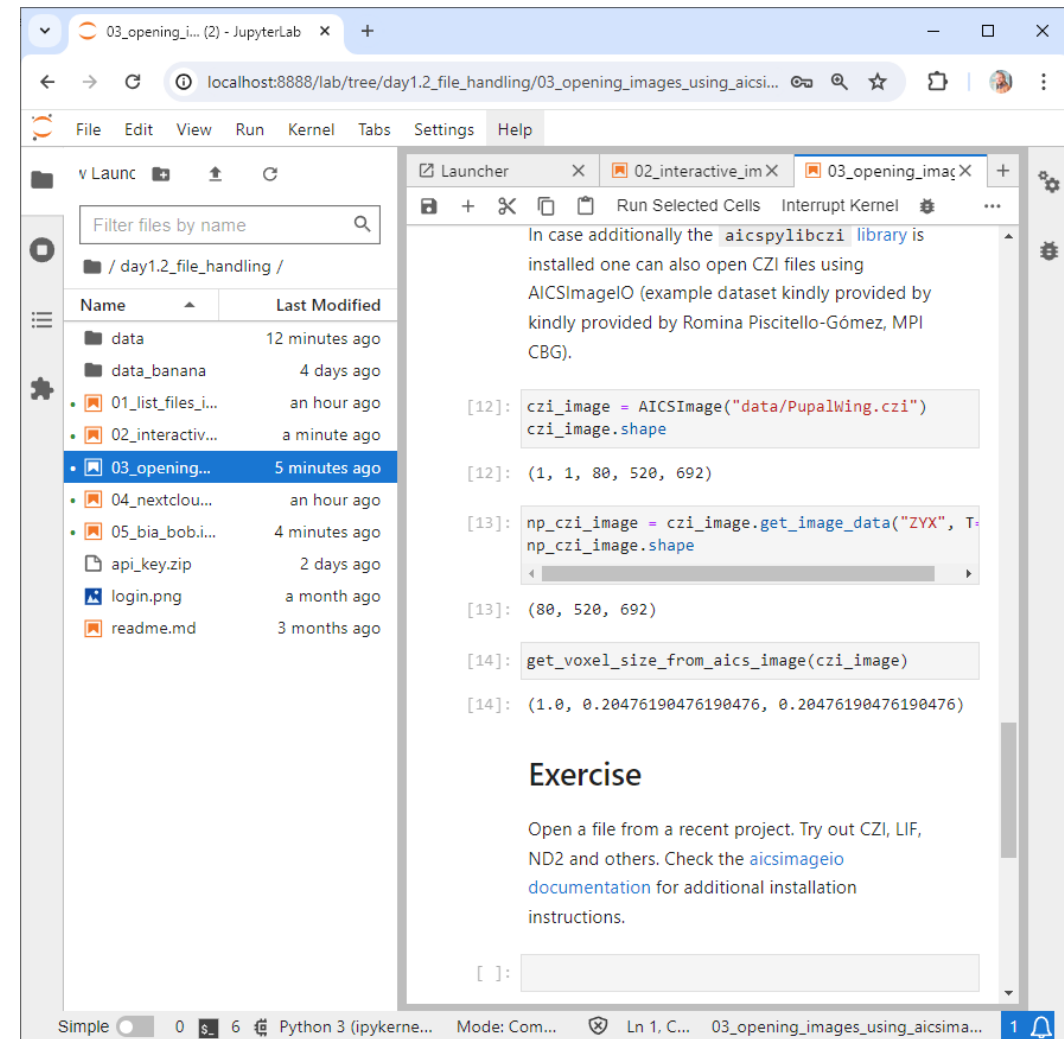
# Exercise: Loading and visualizing image files



The screenshot shows a JupyterLab window titled "02\_interacti... - JupyterLab". The browser address bar indicates the URL: `localhost:8888/lab/tree/day1.2_file_handling/02_interactive_image_visualizati...`. The left sidebar displays a file explorer for the directory `/ day1.2_file_handling /`. The file list includes:

Name	Last Modified
data	9 minutes ago
data_banana	4 days ago
01_list_files_i...	an hour ago
02_interactiv...	24 minutes ago
03_opening...	2 minutes ago
04_nextclou...	an hour ago
05_bia-bob.i...	2 minutes ago
api_key.zip	2 days ago
login.png	a month ago
readme.md	3 months ago

The main area shows a visualization of a cell image with three distinct regions labeled: "Lysosomes", "Mitochondria", and "Nuclei". Below the image, the text "Exercise 1" is followed by the instruction: "Modify the code above to visualize the Hela nuclei in red and the mitochondria in cyan." Below that, "Exercise 2" is followed by the instruction: "Use the picker tool to get an idea about the intensity in the nuclei."



The screenshot shows a JupyterLab window titled "03\_opening\_i... (2) - JupyterLab". The browser address bar indicates the URL: `localhost:8888/lab/tree/day1.2_file_handling/03_opening_images_using_aicsi...`. The left sidebar displays a file explorer for the directory `/ day1.2_file_handling /`. The file list includes:

Name	Last Modified
data	12 minutes ago
data_banana	4 days ago
01_list_files_i...	an hour ago
02_interactiv...	a minute ago
03_opening...	5 minutes ago
04_nextclou...	an hour ago
05_bia-bob.i...	4 minutes ago
api_key.zip	2 days ago
login.png	a month ago
readme.md	3 months ago

The main area shows a code cell with the following text:

In case additionally the `aicspylibczi` library is installed one can also open CZI files using AICSImageIO (example dataset kindly provided by kindly provided by Romina Piscitello-Gómez, MPI CBG).

```
[12]: czi_image = AICSImage("data/Pupalwing.czi")
      czi_image.shape

[12]: (1, 1, 80, 520, 692)

[13]: np_czi_image = czi_image.get_image_data("ZYX", T=
      np_czi_image.shape

[13]: (80, 520, 692)

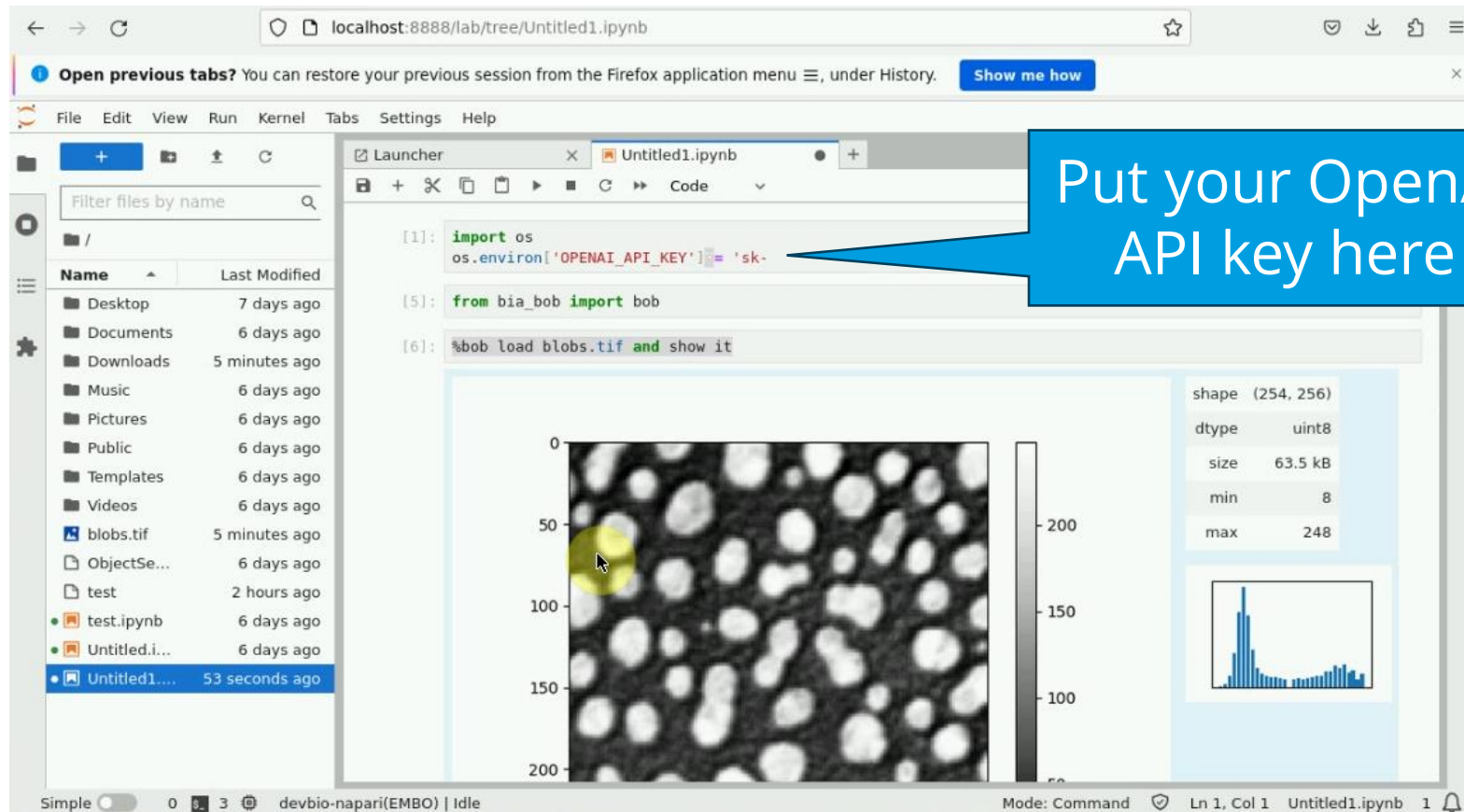
[14]: get_voxel_size_from_aics_image(czi_image)

[14]: (1.0, 0.20476190476190476, 0.20476190476190476)

Below the code, the text "Exercise" is followed by the instruction: "Open a file from a recent project. Try out CZI, LIF, ND2 and others. Check the aicsimageio documentation for additional installation instructions."


```

# Optional exercise: BiA-Bob



The screenshot shows a JupyterLab environment running on localhost:8888. The left sidebar contains a file browser with a search bar and a list of files and folders. The main area is divided into a code editor and a viewer. The code editor shows three cells of Python code:

```
[1]: import os
os.environ['OPENAI_API_KEY'] = 'sk-'

[5]: from bia_bob import bob

[6]: %bob load blobs.tif and show it
```

A blue callout box points to the second line of code, containing the text: "Put your OpenAI API key here".

The viewer displays a grayscale image of a textured surface (blobs.tif) with a color bar on the right. A yellow circle highlights a specific region of interest on the image. To the right of the image, a metadata panel shows the following information:

shape	(254, 256)
dtype	uint8
size	63.5 kB
min	8
max	248

Below the metadata panel is a small histogram of the image data.