PoL
Physics of Life
TU Dresden

CENTER FOR
SYSTEMS BIOLOGY
DRESDEN

Working with file paths in Python

Marcelo Leomil Zoccoler

With material from

Till Korten

August 2023

@zoccolermarcelo

Use relative paths whenever possible

PoL Physics of Life TU Dresden

- Read images by providing a path to skimage.io.imread (or a similar reader function)
- The path can be a relative path

```
from skimage.io import imread

image = imread('../../data/blobs.tif')
```

Relative path to image

- or an absolute path.

```
from skimage.io import imread

image = imread('C:\data\blobs.tif')
```

Absolute path to image

- If you keep your script in a subfolder, the relative path also works on your collaborator's computer
- That is why we use relative paths throughout the teaching material

Backslashes often cause problems on Windows

- Backslash (‘\’) is a special character in python strings
- When pasting Windows paths, this may lead to errors

```
from skimage.io import imread

image = imread(‘C:\data\blobs.tif')
```

OSError: [Errno 22] Invalid argument:

- Add a lowercase ‘r’ before path to fix that

```
from skimage.io import imread

image = imread(r‘C:\data\blobs.tif')
```

This ’r’ tells python to interpret the string as “raw string literal”
No characters with special meaning are interpreted here

Reading Proprietary File Formats with Python

PoL
Physics of Life
TU Dresden

- Sometimes images may be stored in specific proprietary file formats, like '.czi', '.lif', '.nd2', '.ims', etc
- In Fiji, Bioformats usually takes care of opening many different proprietary file formats
- Currently in python, there are a couple solutions to open them:
  - Install a python package capable of opening a specific file format. Example (look for specific package in PyPI, https://pypi.org/):

    conda install –c conda-forge czifile

  - Use AICSImageIO package or its equivalent napari plugin to open them in napari:
    - https://github.com/AllenCellModeling/aicsimageio#aicsimageio
    - https://github.com/AllenCellModeling/napari-aicsimageio#napari-aicsimageio

Reading Multidimentional Images in Python

PoL Physics of Life TU Dresden

- 2D images

```
image.shape
```

```
(254, 256)
```

- 3D images

```
image.shape
```

```
(36, 254, 256)
```

- Multichannel images

```
image.shape
```

```
(36, 254, 256)
```

```
channel1 = image[:,:,0]
channel2 = image[:,:,1]
channel3 = image[:,:,2]
```

- Some libraries return an array in a standardized fashion

```
image.shape
```

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

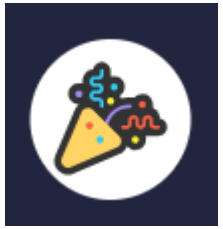
What dimension is this?

?

Tile

I don't know

Time



Slice

Check image metadata to know which dimension corresponds to what.

Check library documentation.

Use `pathlib` to work with paths

```
from pathlib import Path

data_path = Path('../../data/Folder_Structures/Project1_Car_Trunk’)
data_path
```

```
PosixPath('../../data/Folder_Structures/Project1_Car_Trunk')
```

- `data_path` is now an object that lets you do all kinds of useful stuff

```
data_path.name
```

```
‘Project1_Car_Trunk’
```

```
data_path.parent
```

```
PosixPath('../../data/Folder_Structures')
```

```
data_path / 'subdirectory/file.txt'
```

```
PosixPath('../../data/Folder_Structures/Project1_Car_Trunk/subdirectory/file.txt')
```

```
data_path.exists()
```

```
True
```


https://docs.python.org/3/library/pathlib.html

Pathlib also lets you loop over files


```
for path in data_path.iterdir():
    print(path.name)
```

```
0acd2c223d300ea55d0546797713851e818e5c697d073b7f4091b96ce0f3d2fe.png
0bf33d3db4282d918ec3da7112d0bf0427d4eafe74b3ee0bb419770eefe8d7d6.png
.DS_store.txt
```

```
for path in data_path.glob('*.png'):
    print(path.name)
```

```
0acd2c223d300ea55d0546797713851e818e5c697d073b7f4091b96ce0f3d2fe.png
0bf33d3db4282d918ec3da7112d0bf0427d4eafe74b3ee0bb419770eefe8d7d6.png
```

- If you need the list of files multiple times, store it in a list

```
list(data_path.iterdir())
```

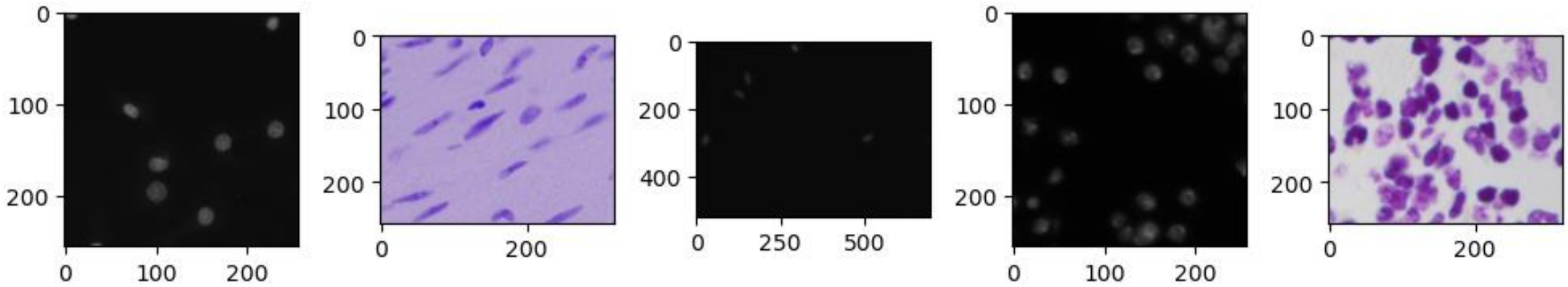
```
[PosixPath('../../data/Folder_Structures/Project1_Car_Trunk/0acd2c223d300ea55d.png'),
PosixPath('../../data/Folder_Structures/Project1_Car_Trunk/0bf33d3db4282d918ec.png'),
PosixPath('../../data/Folder_Structures/Project1_Car_Trunk/.DS_store.txt']
```

```
list(data_path.glob('*.png')
```

```
[PosixPath('../../data/Folder_Structures/Project1_Car_Trunk/0acd2c223d300ea55d.png'),
PosixPath('../../data/Folder_Structures/Project1_Car_Trunk/0bf33d3db4282d918ec.png'),
```

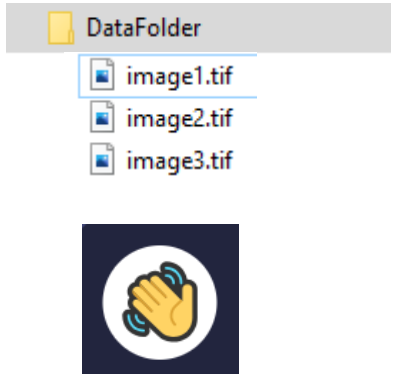
For example, we plot all images in a folder:

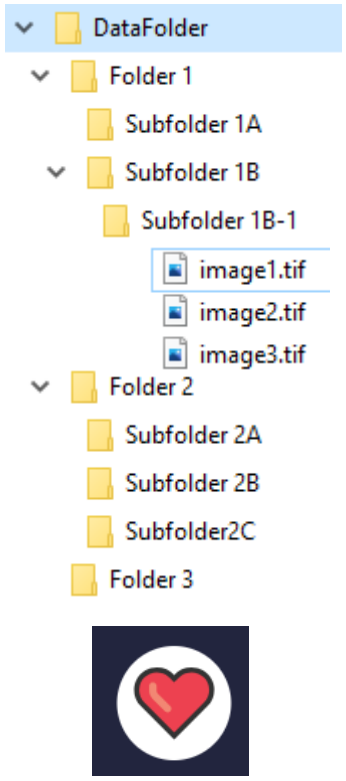
```
#create a list of all png image paths
image_path_list = list(data_path.glob('*.png'))

# first we use the list to determine the number of files with len(image_path_list),
so that we can create a figure with the appropriate number of subplots
fig, ax = plt.subplots(1, len(image_path_list), figsize=(10,3))

# Now we loop over the list to plot each image
for count, image_path in enumerate(image_path_list):
    image = imread(image_path)
    ax[count].imshow(image)

plt.tight_layout()
```

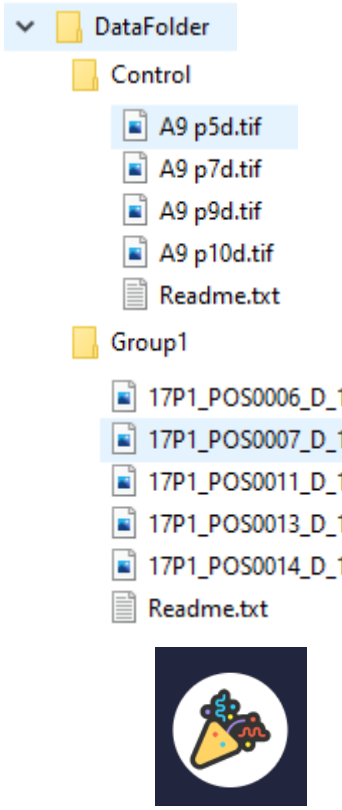


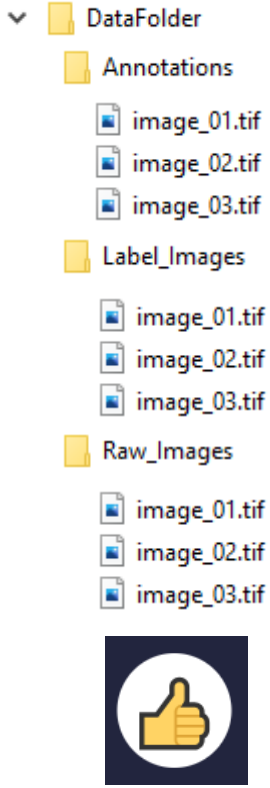
@zoccolermarcelo

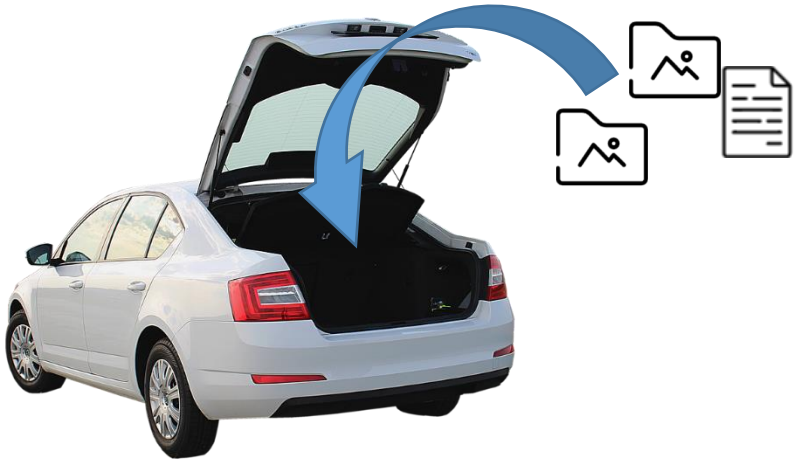
Folder Structures











A few general advice:

- Avoid too many levels
- Add “Readme” files as soon as you create a folder (you will forget later)
- Consider using a data management platform
- Talk to a data management experts to find the best structure to your needs

Image by eikira from Pixabay (https://pixabay.com/photos/trunk-automobile-vehicle-luggage-1478832/)

@zoccolermarcelo