

# Working with file paths in python

Till Korten

With material from  
Marcelo Leomil Zoccoler

December 2022

# Use relative paths whenever possible

- Read images by providing a path to `skimage.io.imread` (or a similar 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

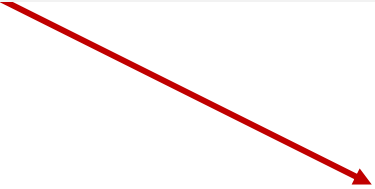
- Backslash ('\') is a special character in python strings
- When pasting Windows paths, this may lead to errors

```
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)

```
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.exists()
```

```
True
```

# 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'),
```

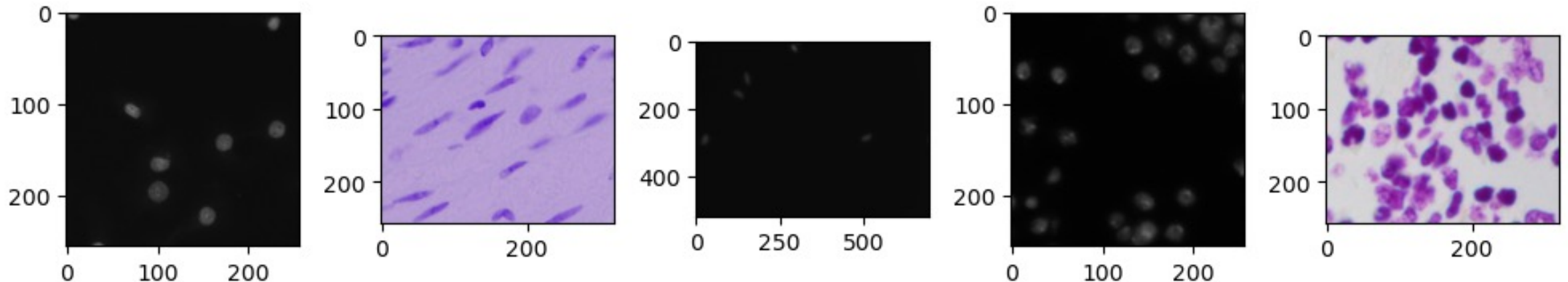
# For example, we plot all images in a folder:

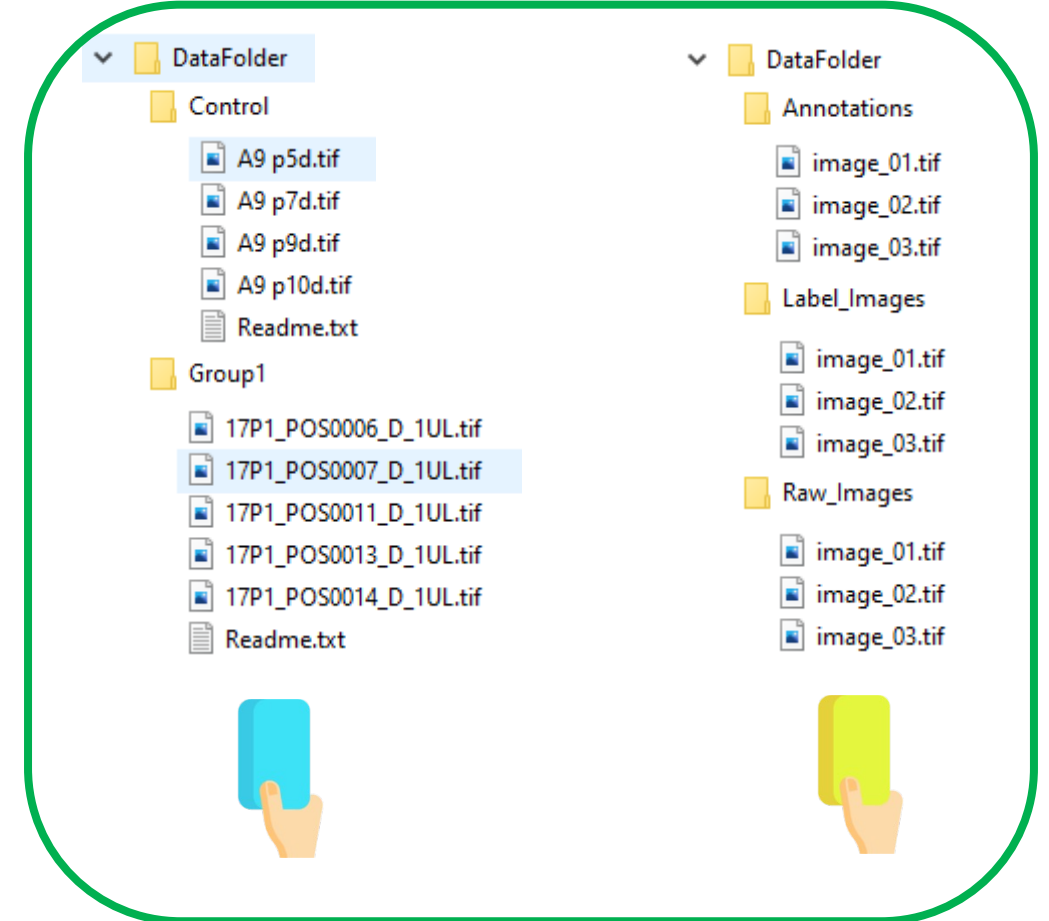
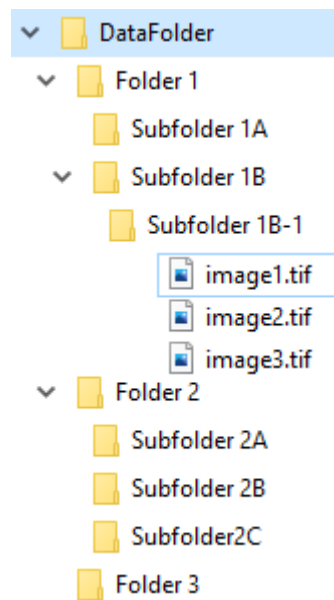
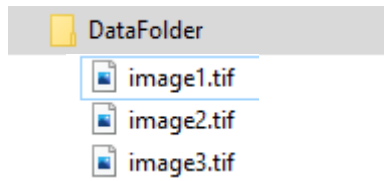
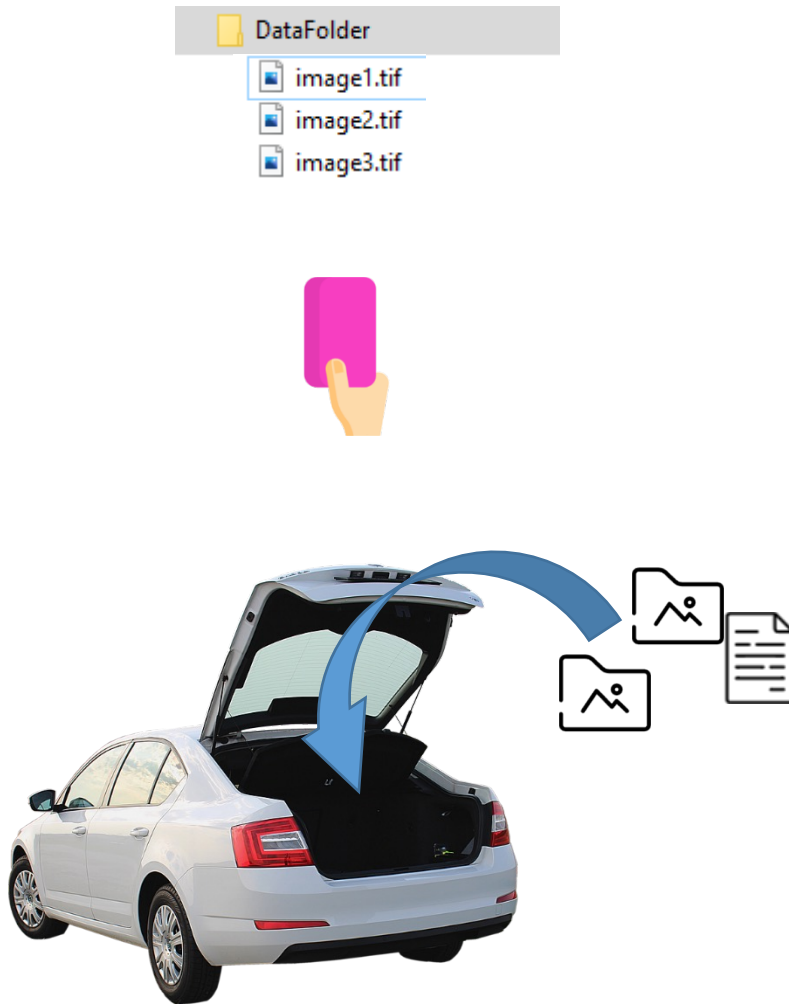
```
#create a list of all png image paths
image_path_list = list(data_path.glob('*.*png'))

# 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()
```





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