# Exercises on digital image analysis with Python

This document contains some exercises you can do to get a feel for some digital image analysis with Python. It is intended to go along with a course on the subject matter, and you can read the corresponding lecture notes if you are unclear on how to proceed.

## Exercise 1: Reading and plotting an image

In this exercise you will read in an image and plot it.

- Import the `io` module from the `skimage` package
- Use `io.imread` to read the file `img/chest_006.jpg`, store the image data as a variable
- Import `matplotlib.pyplot` under the alias `plt`
- Use `plt.imshow` to display the image

## Exercise 2: A deuteranopi filter

In this exercise you will implement a *deuteranopi* filter by working with `numpy` array indexing. Deuteranopi, or "red-green color blindness" as it is often called, is a condition where it is hard to separate red and green colors.

- a) Read in the example image `img/color_wheel.png` and plot it
- b) Use indexing the pick out the red, green and blue color channels
- c) Now define a new array `mixed`, by adding `0.5*red + 0.5*green`.
- d) Now go back and replace both the red and the green channels with this mix, by writing
  - `colorwheel[:, :, 0] = mixed`
- e) Plot the modified colorwheel
- f) Run the same process through the `deuteranopia_test.jpg` image. You should be able to read the number in the image before you apply the filter, but not after.

## Exercise 3: Explore a `numpy` array

In this exercise you will load in a few images, and use the fact that it is a `numpy` array to explore the data a bit.

- a) Load in the image `img/brain_tumors/glioma4.jpg`
- b) Use `numpy` to check the data type of the pixels `.dtype`, the number of dimensions `.ndim`, the shape `.shape`..
- c) The number of dimensions is a bit strange on this image. Can you figure out why?
- d) Load in the image `img/chest_003.png` and check the same variables.
- e) Why is the number of dimensions larger for the brain tumor image than the chest x-ray?

Hint: The brain tumor image is a bit strange because it is a graytone image that is stored like a RGB color image. You can pull out the three channels to double check that the information stored in each is identical. Can you find a way to simplify the image?

## Exercise 4: Cropping an image using slicing

In this exercise you will crop an image using slicing of a numpy array. You can write your code from scratch, or fill into the skeleton below

- a) Read in the example image `skimage.data.chelsea`
- b) Check the image dimensions using `.shape`
- c) plot the image, what does it show?
- d) Use slicing to crop out a 64x64 pixel window of an eye
- e) Plot the eye as its own image

In [ ]:
```
from skimage import data
import matplotlib.pyplot as plt

# Read in example image
chelsea = data.chelsea()

# Check image dimensions
print(...)

# Plot example image
plt.imshow(...)
plt.show()

# Crop the image
x0 = ...
y0 = ...
width = ...

eye = chelsea[..., ...]

# Plot the cropped image
plt.imshow(...)
plt.show()
```

### Exercise 5: Reading a DICOM-image and working with tags

In this exercise you will use the `pydicom` package to read a DICOM-image, change its tag, plot the image data and save the modified file. You can read an image using `pydicom.dcmread`

- a) Read in the file `img/vannfantom.dcm`
- b) Read the `PatientName` tag, what does it say?
- c) Does the file have a `PatientID` tag? If so, what does it say?
- d) Delete both tags
- e) Plot the image data contained within the file (Hint: `.pixel_array` )

- f) Find the size of the image, the data type of the pixels and the average value of the entire image.
- g) Save the image with the saved tags under `results/vannfantom.dcm`

## Exercise 6: Batch-process images

In this exercise you will use `glob` to loop over and repeat operations.

The `img` folder contains a series of images called `chest000.png`, `chest001.png` and so on. You will now use a loop-combined with `glob` to loop over these and plot them out. Fill into the code below to do this:

In [ ]:
```
from skimage import io
import matplotlib.pyplot as plt
import glob

xray_files = sorted(glob.glob(...))

for xray in xray_files:
    img = io.imread(xray)

    plt.imshow(img)
    plt.title(xray)
    plt.show()
```

### Exercise 7: Segmenting a cell ¶

- a) Read in the example image `skimage.data.cell`
- b) Check the `shape` and `dtype` of the image
- c) Plot the image
- d) Use thresholding to segment out the cell itself
- e) How big is the cell (in the number of pixels)?

### Exercise 8: Finding average pixels in a water phantom

Let us return to the water phantom we had you read in earlier. You can find this image data as follows

In [ ]:
```
import pydicom as dcm

data = dcm.dcmread("img/vannfantom.dcm")
img = data.pixel_array
```

- a) Print out the pixel type, and the shape of this image.
- b) Plot the image
- c) Compute the average pixel intensity and the standard deviation ( `np.mean` and `np.std` )

- d) Use `filters.threshold_otsu` on the image to segment it, then compute again the mean and std of just the thresholded parts of the image

In this case the Otsu thresholded image gets the small gray segment at the top of the image. Let us instead create a circular mask towards the center. The code lines below do this:

In [ ]:
```
import numpy as np
from skimage import draw

mask = np.zeros(img.shape, dtype=bool)
rr, cc = draw.disk((256, 256), 20, shape=img.shape)
mask[rr, cc] = 1
```

- e) Read the code lines and *attempt* to understand what is happening.
- f) Plot the mask image
- g) Use this mask image to compute the mean and standard deviation.