

ELE510 Image Processing with robot vision: LAB - Test the environment

Purpose: This jupyter notebook is just for you to test that the environment is set and ready for the various assignments.

The following package are necessary for the assignments:

- [opencv](#) "OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library" - [Documentation](#)
- [numpy](#) "The fundamental package for scientific computing with Python" - [Documentation](#)
- [matplotlib](#) "A comprehensive library for creating static, animated, and interactive visualizations in Python" - [Documentation](#)

Numpy

NumPy is the fundamental package for scientific computing in Python. It provides routines for fast operations on arrays.

If the following cell returns you an error: ModuleNotFoundError: No module named 'np' Open the Anaconda Prompt and run the following: pip install numpy

```
In [4]: import numpy as np # It is common to import **numpy** under the briefer name **np**
```

```
In [5]: m = np.matrix('3 2; 3 4') # create a matrix
```

```
In [6]: # print the matrix  
print(m)
```

```
[[3 2]  
 [3 4]]
```

```
In [7]: # print the shape of the matrix  
print(m.shape)
```

```
(2, 2)
```

The shape of the matrix should be: (2,2)

```
In [8]: # print the following values: m(0,0), m(0,1), m(1,0)
```

```
print(m[0,0])  
print(m[0,1])  
print(m[1,0])
```

```
3  
2  
3
```

The answer should be: 3, 2, 3

OpenCV

OpenCV-Python is a library of Python bindings designed to solve computer vision problems.

If the following cell returns you an error: ModuleNotFoundError: No module named 'cv2' Open the Anaconda Prompt and run the following: pip install opencv-python

```
In [9]: # import the opencv-python module  
import cv2  
import os
```

Import an image

```
In [10]: flag = cv2.COLOR_BGR2RGB
```

```
# Read the image 'preikestolen.jpg' located in the 'images/' folder with cv2.  
# Example:  
img_file = os.path.join(os.getcwd(), 'images', 'preikestolen.jpg')  
img = cv2.imread('./images/preikestolen.jpg', flag)
```

```
#img = ...
```

```
# OpenCV uses BGR as its default colour order for images, matplotlib uses RGB.  
# When you display an image loaded with OpenCV in matplotlib the channels will be back to front.
```

```
img = cv2.cvtColor(img, flag)
```

```
In [11]: img.shape[2]
```

```
3
```

Print the dimension and other information of the image

```
In [12]: # Display the height, width, number of channels of the image  
height = img.shape[0]  
width = img.shape[1]  
channels = img.shape[2]
```

```
print('Image Dimension      : ', img.shape)  
print('Image Height         : ', height)  
print('Image Width          : ', width)  
print('Number of Channels   : ', channels)
```

```
# The answer shoud be like this:
```

```
# Image Dimension      : (1064, 1600, 3)  
# Image Height         : 1064  
# Image Width          : 1600  
# Number of Channels   : 3
```

```
Image Dimension      : (1064, 1600, 3)  
Image Height         : 1064  
Image Width          : 1600  
Number of Channels   : 3
```

Matplotlib

Matplotlib is a Python 2D plotting library.

If the following cell returns you an error: ModuleNotFoundError: No module named 'plt' Open the Anaconda Prompt and run the following: pip install matplotlib

```
In [13]: # We use the matplotlib submodule **pyplot**. Following a widely used convention, we use the 'plt' alias  
import matplotlib.pyplot as plt
```

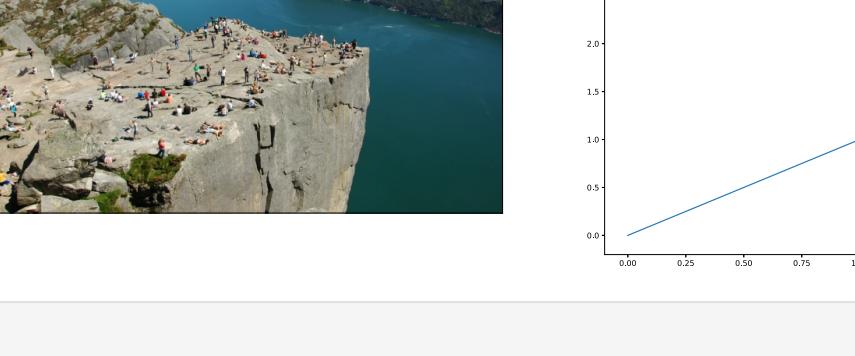
Display the imported image

```
In [14]: plt.imshow(img)  
plt.xticks([]), plt.yticks([]) # Hides the graph ticks and x / y axis  
plt.show()
```



Display the image in greyscale

```
In [15]: flag = cv2.IMREAD_GRAYSCALE  
grey_img = cv2.imread(img_file, flag)  
  
plt.imshow(grey_img, cmap='gray', vmin=0, vmax=255)  
plt.xticks([]), plt.yticks([]) # Hides the graph ticks and x / y axis  
plt.show()
```



```
In [16]: xs = np.array([0, 1, 2]) # Set x-axis values  
f = xs**2 # Set the corresponding y values
```

```
plt.plot(xs, f) # Create a plot  
plt.show() # Display the plot
```



```
In [17]: # Display the previous plot and the preikestolen images (color and grayscale) together in the same row.  
# Hint: use plt.subplot function to show them together
```

```
plt.figure(figsize=(40, 10))  
fig, axes = plt.subplots(1, 2)  
fig.suptitle('Preikestolen image processing')  
axes[0].imshow(img)  
axes[1].imshow(grey_img, cmap='gray', vmin=0, vmax=255)  
axes[0].axis('off')  
axes[1].axis('off')
```

```
(-0.5, 1599.5, 1063.5)
```

```
<Figure size 2880x720 with 0 Axes>
```

Preikestolen image processing

```
In [18]: plt.figure(figsize=(40,10))
```

```
plt.subplot(1,3,1) # == nrows, ncols, index
```

```
plt.plot(xs)
```

```
plt.subplot(1,3,2)
```

```
plt.imshow(img)
```

```
plt.xticks([]), plt.yticks([])
```

```
plt.subplot(1,3,3)
```

```
plt.imshow(grey_img, cmap='gray', vmin=0, vmax=255)
```

```
plt.xticks([]), plt.yticks([])
```

```
plt.show()
```

```
In [19]: plt.figure(figsize=(40,10))
```

```
plt.subplot(1,3,1) # == nrows, ncols, index
```

```
plt.plot(xs)
```

```
plt.subplot(1,3,2)
```

```
plt.imshow(img)
```

```
plt.xticks([]), plt.yticks([])
```

```
plt.subplot(1,3,3)
```

```
plt.imshow(grey_img, cmap='gray', vmin=0, vmax=255)
```

```
plt.xticks([]), plt.yticks([])
```

```
plt.show()
```



```
In [ ]:
```