ANA-522-OL1 Spring 2022
Mod02 Week03 HW: NumPy
Due: Sunday January 30th at midnight

In digital imaging, a pixel is the smallest item of information in an image. Pixels are arranged in a 2-dimensional grid, represented using squares (or array cells). Each pixel is a sample of an original image, where more samples typically provide more-accurate representations of the original. Each pixel has typically three color components, red, green, and blue. The intensity of each color is represented with value in the range of [0,255]. See RGB color model or Color Picker for details.

We are going to use NumPy array to store pixels, so that we can create images by creating arrays, save pixels in array cells, and manipulate arrays for image processing.

[ ]:
```
## We are creating a simple image of 300x300 pixels using NumPy array
## the value of each pixel is a color configuration of [Red, Green, Blue]
## so we can see the array shape is 300x300x3 ( think of it as a cubic )
import numpy as np

width = 300
height = 300
array = np.zeros([height, width, 3], dtype=np.uint8)

print(array.shape)
```

[ ]:
```
## To see the image for the first time
## every pixel is 0, therefore there is no image to see but black
## (0,0,0): black , (255,255,255): white

import matplotlib.pyplot as plt
# show no axis
plt.axis('off')

#
# Display image from array
#
plt.imshow(array)
```

[ ]:
```
## To change current color configuration from all zeros to the ones we want
## for example, having red, green, blue row-wise strips
```

```
## each row strip is with 100x300 pixels of its color by slicing the original␣
 ↪array

# RGB configuration
array[:100,:] = [255, 0, 0]
array[100:200,:] = [0, 255, 0]
array[200:300,:] = [0, 0, 255]

import matplotlib.pyplot as plt
# show no axis
plt.axis('off')

#
# Display image from array
#
plt.imshow(array)
```

[ ]:
```
## If needed, we can save the image as a file in PNG ( Portable Network Graphics␣
 ↪) format
## for instance with filename RBG_row_strip.png in the following
## (once the cell run, see the file created and place in the current folder )
## (double clock on the filename in the folder will display the image in a new␣
 ↪browser page)

import matplotlib.image as mpimg

#save img directly from array
mpimg.imsave('RBG_row_strip.png',array)
```

[ ]:
```
## Similar, we can load an image from the file folder with the help of mpimg
## The image file absolute location on the JupyterLab server is at '/home/ANA522/
 ↪mod02/flower.png'
## Once loaded and displayed, the shape of the arrray is of 1300 x 1800 x 3

flower_src = mpimg.imread('/home/ANA522/mod02/flower.png') #this statement load␣
 ↪the flower.png from the server

# show dimension
print(flower_src.shape)

#display the image
plt.axis('off')
plt.imshow(flower_src)
```

[ ]:
```
## We can also examine the configuration of each image pixels stored in the array
```

```
print(type(flower_src))
print(flower_src.dtype)
print(flower_src.shape)

# and all cells of the array
print(flower_src)

# Notice that the sample picture (flower.png)
# has the RGB configuration range [0,1] by percentage
# in float32 data type
```

NumPy_HW questions start from here

There are 3 problems of image processing using NumPy

Please note that to solve all problems in this assignment, only NumPy functions are needed. DO NOT USE any image modules or image processing functions in your solution. We are practicing NumPy only.

Please write statements to manually process the image NumPy array to generate the targeted image results.

Image cropping

Cropping is the removal of unwanted outer areas from a photographic or illustrated image. The process usually consists of the removal of some of the peripheral areas of an image to remove extraneous trash from the picture.

0.0.1 Problem 01:

The original flower.png is 1300x1800. Use the image array to focue on a square area of 1100x1100 and make that a new array to create a new image that is the main area of the flower

Image padding

Padding is the space between an image or cell contents and its outside border. Padding goes completely around the contents: top, bottom, right, and left sides.

0.0.2 Problem 02:

Save the cropped flower image of pixels 1100x1100 (from Problem01), and use it to add apadding. Expand the original array to include padding area, width 100, around the picture.
The finished image with padding has pixels of 1300x1300. You get to choose a color for the padding.

0.0.3 Problem 03:

Rotate or flip the original flower.png of 1300x1800 to any direction.

0.0.4 Optional problem:

Grayscale the original flower.png of 1300x1800