

Problem 1 Image operations and vectorization (1pt)

Vector operations using numpy can offer a significant speedup over doing an operation iteratively on an image. The problem below will demonstrate the time it takes for both approaches to change the color of quadrants of an image.

The problem reads an image "Lenna.png" that you will find in the assignment folder. Two functions are then provided as different approaches for doing an operation on the image.

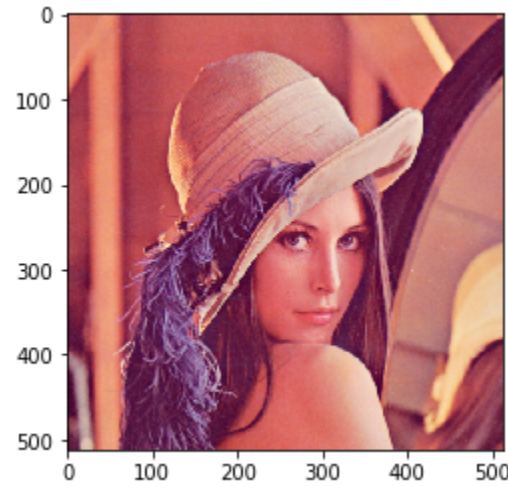
Your task is to follow through the code and fill in the "piazza" function using instructions on Piazza.

```
In [7]: import numpy as np
import matplotlib.pyplot as plt
import copy
import time

img = plt.imread('Lenna.png')          # read a JPEG image
print("Image shape", img.shape)        # print image size and color depth

plt.imshow(img)                        # displaying the original image
plt.show()
```

Image shape (512, 512, 3)



```
In [8]: def iterative(img):

    image = copy.deepcopy(img)          # create a copy of the image matrix
    for x in range(image.shape[0]):
        for y in range(image.shape[1]):
            if x < image.shape[0]/2 and y < image.shape[1]/2:
                image[x,y] = image[x,y] * [0,1,1]    #removing the red channel
            elif x > image.shape[0]/2 and y < image.shape[1]/2:
                image[x,y] = image[x,y] * [1,0,1]    #removing the green channel
            elif x < image.shape[0]/2 and y > image.shape[1]/2:
                image[x,y] = image[x,y] * [1,1,0]    #removing the blue channel
            else:
                pass
    return image

def vectorized(img):

    image = copy.deepcopy(img)
    a = int(image.shape[0]/2)
    b = int(image.shape[1]/2)
    image[:a,:b] = image[:a,:b]*[0,1,1]
    image[a,:b] = image[a,:b]*[1,0,1]
    image[:a,b:] = image[:a,b:]*[1,1,0]

    return image
```

```
In [9]: ## The code for this problem is posted on Piazza. Sign up for the course if you have not. Then find
## the function definition included in the post 'Welcome to CSE252A' to complete this problem.
## This is the only cell you need to edit for this problem.
def piazza():
    start = time.time()
    image_iterative = iterative(img)
    end = time.time()
    print("Iterative method took {0} seconds".format(end-start))
    start = time.time()
    image_vectorized = vectorized(img)
    end = time.time()
    print("Vectorized method took {0} seconds".format(end-start))
    return image_iterative, image_vectorized

# Run the function
image_iterative, image_vectorized = piazza()
```

Iterative method took 2.140733242034912 seconds
Vectorized method took 0.018352031707763672 seconds

```
In [10]: # Plotting the results in sepearate subplots

plt.subplot(1, 3, 1) # create (1x3) subplots, indexing from 1
plt.imshow(img)      # original image

plt.subplot(1, 3, 2)
plt.imshow(image_iterative)

plt.subplot(1, 3, 3)
plt.imshow(image_vectorized)

plt.show()           #displays the subplots

plt.imsave("multicolor_Lenna.png",image_vectorized)    #Saving an image
```

