Lab_01_21-27_19_DS

February 2, 2022

Name: Saurabh S. Ramteke
Roll No: 21-27-19
M.Tech: Data Science

Obj: To practise openCV packages and get used to its modules for usage in computer vision applications.

Import Dependencies

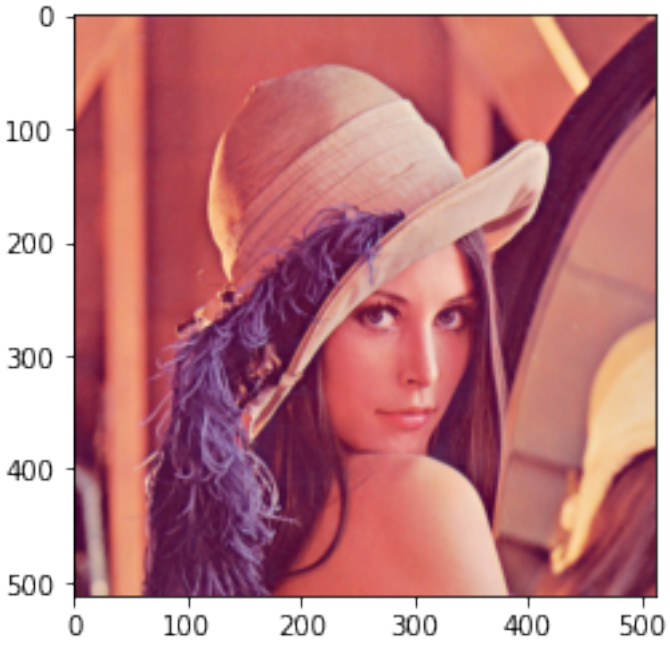
[13]:
```
import cv2
import matplotlib.pyplot as plt
import numpy as np

%matplotlib inline
```

Read File using matplotlib

[2]:
```
image = plt.imread("Lenna.png")
plt.imshow(image)
print(f"Dimensions: {image.shape}")
# cv2.imshow("Lenna", image)
# cv2.waitKey(0)
```

```
Dimensions: (512, 512, 3)
```

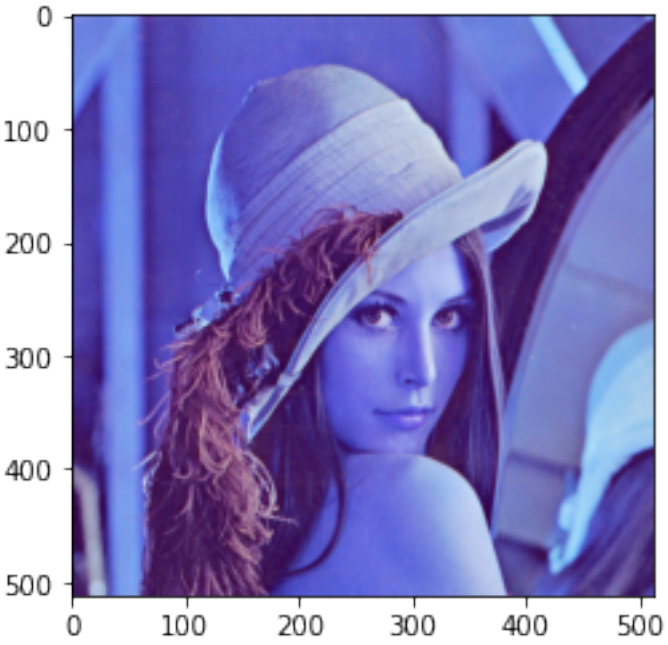


We can use plt or cv2 for display of the image, but in jupyter lab cv2 will not display the image inline, and hence we required to add another line waitKey(0) to make it stay their until we close the window.

Read file using openCV package

[3]:
```
image2 = cv2.imread("Lenna.png")
plt.imshow(image2)
print(f"Dimensions: {image2.shape}")
# cv2.imshow("Lenna", image2)
# cv2.waitKey(0)
```

```
Dimensions: (512, 512, 3)
```

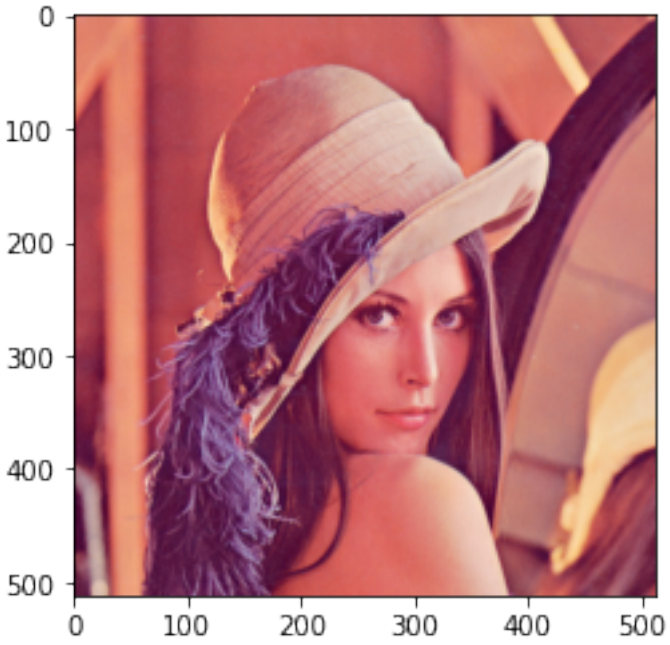


File imported using openCV reads the file to BGR format. Hence we need to convert it into RGB which is being used by most image compression tools nowadays.

Converting BGR -> RGB

[4]:
```
imageRGB = cv2.cvtColor(image2, cv2.COLOR_BGR2RGB)
plt.imshow(imageRGB)
print(f"Dimensions: {imageRGB.shape}")
```

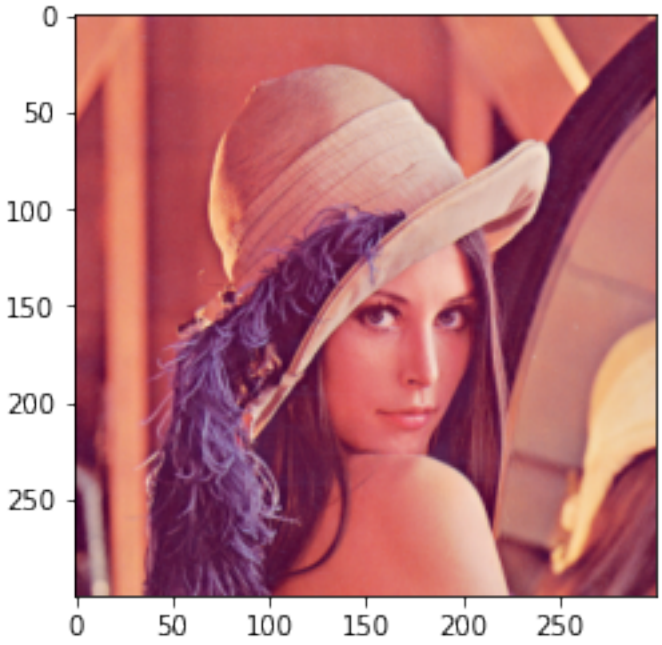
```
Dimensions: (512, 512, 3)
```



Resizing Image

[5]:
```
width, height = 300,300
image_resize = cv2.resize(imageRGB, (width, height))
plt.imshow(image_resize)
print(f"Dimensions: {image_resize.shape}")
```

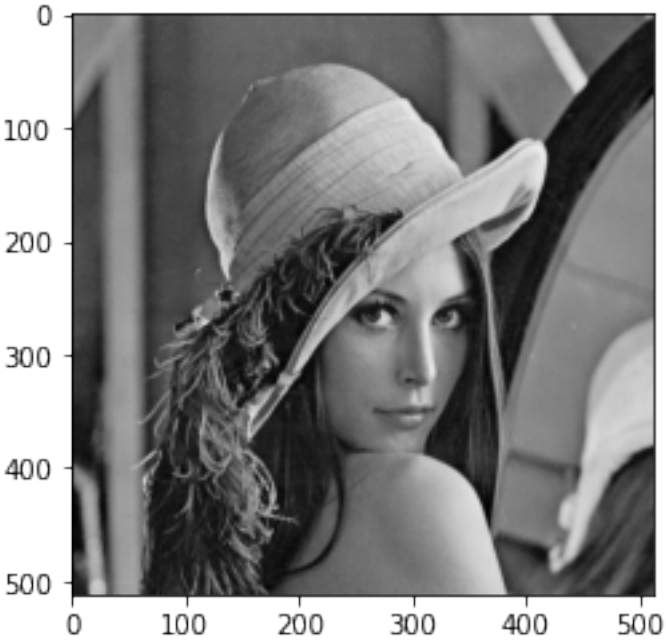
```
Dimensions: (300, 300, 3)
```



Converting Image to grayscale

[6]:
```
image_gray = cv2.cvtColor(image2, cv2.COLOR_RGB2GRAY)
plt.imshow(image_gray, cmap = 'gray')
```

[6]: <matplotlib.image.AxesImage at 0x1b257e6be50>

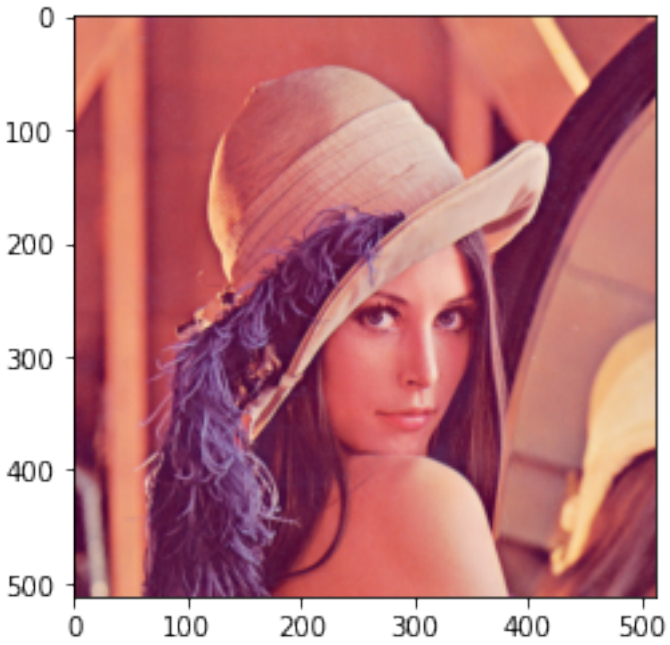


We use cmap = 'gray' because matplotlib has separate colormap. cmap stands for colormap and it's a colormap instance or registered colormap name (cmap will only work if c is an array of floats). Matplotlib colormaps are divided into the following categories: sequential, diverging, and qualitative

Copying the image

[7]:
```
image3 = np.copy(image)
plt.imshow(image3)
```

[7]:
```
<matplotlib.image.AxesImage at 0x1b257ecfa00>
```



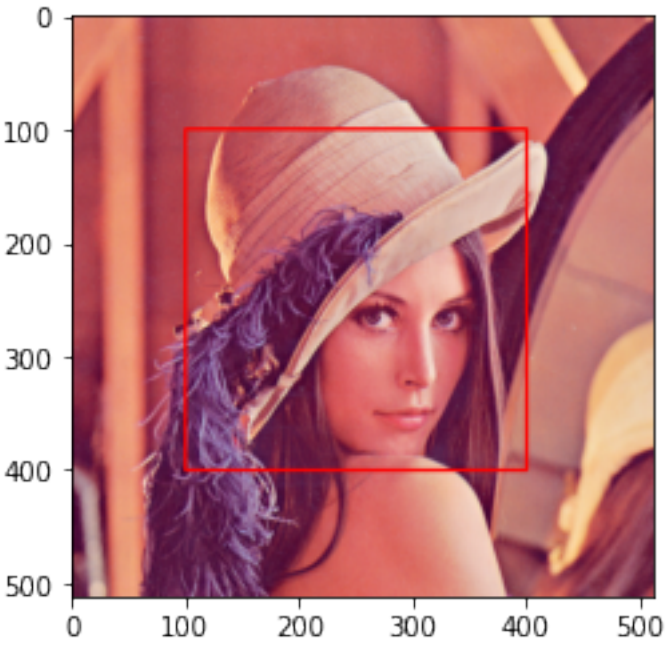
Adding rectangle on image

[8]:
```
image_copy = np.copy(imageRGB)

start_point = (100, 100)
end_point = (400, 400)
color = (255, 0, 0)
thickness = 2

image_rect = cv2.rectangle(imageRGB, start_point, end_point, color, thickness)
plt.imshow(image_rect)
```

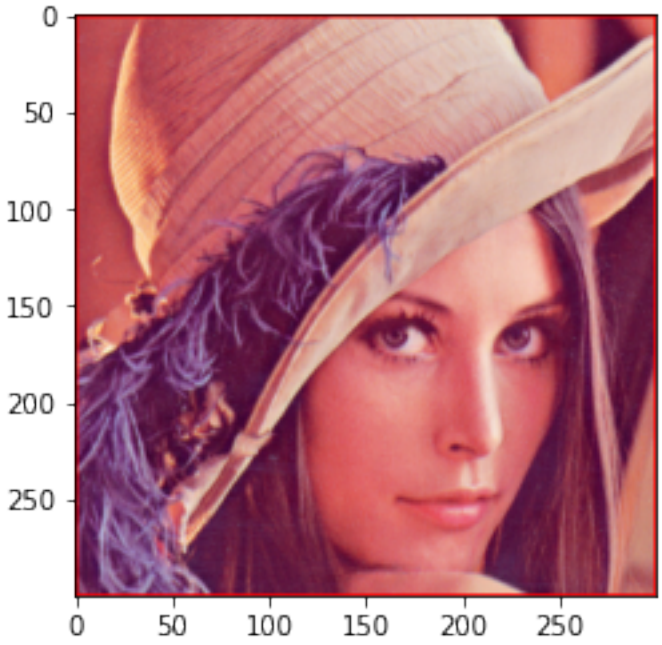
[8]: <matplotlib.image.AxesImage at 0x1b257f3f490>



Crop Rectangle

[9]:
```
cropped_image = image_rect[100:400, 100:400]
plt.imshow(cropped_image)
```

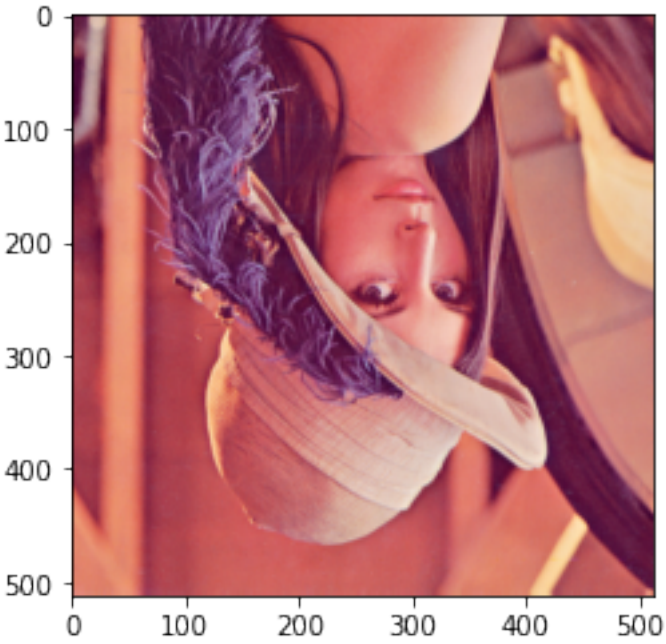
[9]: <matplotlib.image.AxesImage at 0x1b257fa1520>



Flip on Horizontal Axis, 0

[10]:
```
flip_on_Hori = cv2.flip(image_copy,0)
plt.imshow(flip_on_Hori)
```

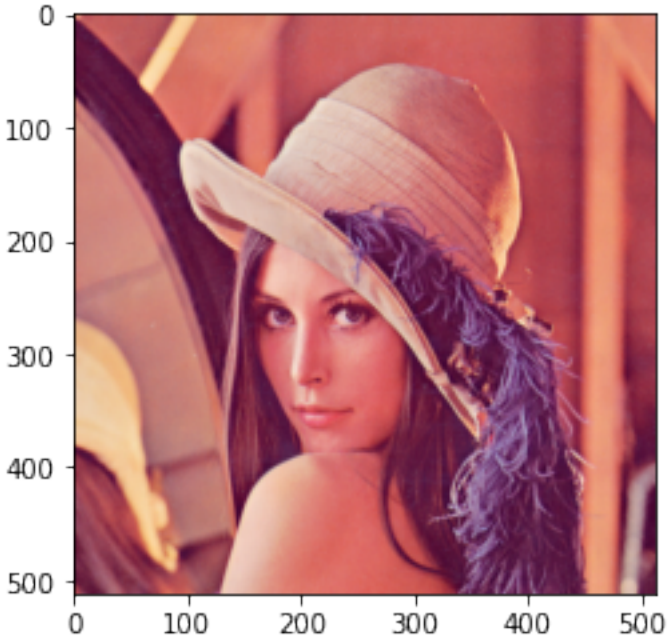
[10]: <matplotlib.image.AxesImage at 0x1b257ffceb0>



Flip on vertical Axis, 1

[11]:
```
flip_on_Vert = cv2.flip(image_copy,1)
plt.imshow(flip_on_Vert)
```

[11]:
```
<matplotlib.image.AxesImage at 0x1b258061fd0>
```



Write a function to create a white image size entered by the user and then create 4 boxes of Black, Blue, Green and Red respectively on each corner of the image. The size of the colored boxes should be 1/10th the size of the image. (HINT: the arrays of ones and zeros can be in more than 2 dimensions)

[15]:
```
def white_image(width, height):
    image_white = np.ones((width, height,3), dtype = np.uint8)
    image_white.fill(255)
    # Image is RGB

    x = int(width/10)
    y = int(height/10)

    black = cv2.rectangle(image_white, (0,0) ,(x,y), (0,0,0), -1 )
    red   = cv2.rectangle(image_white, (width, height), (width - x, height - y␣
 ↪), (200,0,0), -1)
```

```
    blue  = cv2.rectangle(image_white, (width, 0), (width - x, y), (0,0,200),␣
↪-1)
    green = cv2.rectangle(image_white, (0, height), (x, height -y), (0,200,0),␣
↪-1)
    plt.imshow(image_white)
    plt.xticks([]),plt.yticks([]) #HIdes the graph tics & axes

width  = int(input("Input the width of image: "))
height = int(input("Input the height of image: "))

white_image(width, height)
```

```
Input the width of image: 400
Input the height of image: 400
```