Assignment # 1

DIP

Name	:	Arshad Habib
Roll No	:	17I-0208
Section	:	A
Submitted To	:	Dr. Akhtar Jamil
Submission Date	:	10-10-2021

Question 1: Perform following tasks.

a) Read the given image and convert it to grayscale image.
b) Use intensity slicing to separate each object. For each object count its pixels and display their count inside the object as shown in the following sample output. For this step, you are not allowed to use any Library or Package. You must use loops to reach each pixel and perform the required processing.
c) Perform intensity slicing using OpenCV and compare the results with your implementation done in b) part.

Solution:

a)

First of all importing numpy, math and matplotlib libraries as

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import math as m #importing libraries
```

Reading image from path and printing image shape

```
path = r'F:\Study\Fall 2021\DIP\Assignments\1\Assignment#1\img.jpg' #image path
img = cv2.imread(path,cv2.IMREAD_COLOR) #reading image by opencv
img.shape #printing image size
```

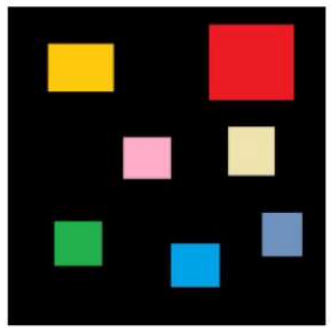
Output:

```
[2]: (300, 300, 3)
```

After that I am converting BGR image to RGB image by using openCV function.

```
img=cv2.cvtColor(img,cv2.COLOR_BGR2RGB) #converting to RGB
plt.imshow(img) #output
plt.axis('off')
plt.show()
```

Output:



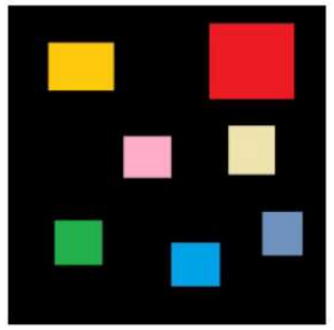
Then I made a function which converts a RGB image into grayscale image by applying grayscale formula to each pixel in nested loops.

```
def rgb_to_grayscale(image): #function to covert to gray scale
    newimage=np.zeros((image.shape[0],image.shape[0]),np.uint8) #making an empty image to original image size using numpy
    for i in range(len(newimage)):
        for j in range(len(newimage[i])): #nested loops
          newimage[i][j]=image[i][j][0]*0.3+image[i][j][1]*0.59+image[i][j][2]*0.11 #formula to convert RGB image to Grayscale Image
    return newimage    #returning grey scale image
```

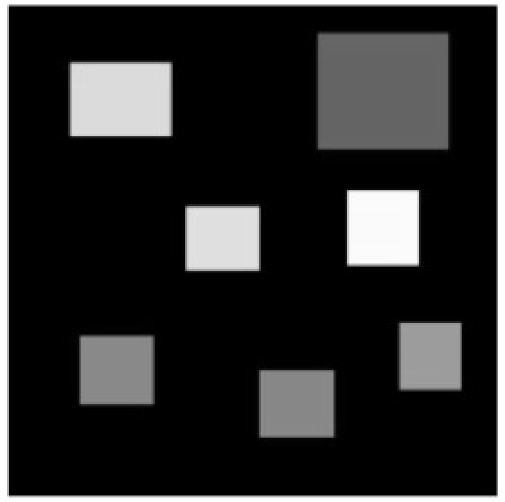
Now I am calling the above function to convert RGB image to grayscale image. First I am making a new empty image of size of original image and then copying original image to new image.

```
img1=np.zeros((img.shape[0],img.shape[0],3),np.uint8) #making an empty image to original image size using numpy
img1[:,:,0]=img[:,:,0] #copying image to new variable
img1[:,:,1]=img[:,:,1] #copying image to new variable
img1[:,:,2]=img[:,:,2] #copying image to new variable
greyimage=rgb_to_grayscale(img1) #calling grey scale function defined above
plt.imshow(greyimage,cmap='gray') #showing greyscale image
plt.axis('off')
plt.show()
```

Input:



Final Output:



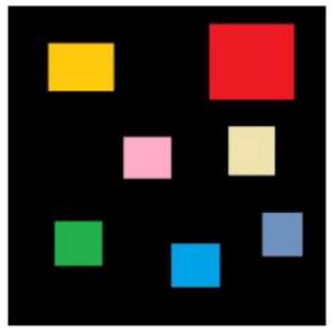
b)

1st Object:

First object is of red color. First of all I am cropping red object out of original image and then checking the pixel values.

```
plt.imshow(img[25:80,190:260]) #extracting red object from original image by cropping
plt.axis('off')
plt.show()
img[25:80,190:260] #pixel values of red object
```

Input:



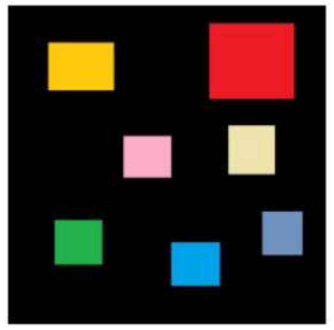
Output:

```
array([[[201,  44,  53],
        [255,  13,  27],
        [237,  28,  34],
        ...,
        [234,  29,  36],
        [231,  30,  40],
        [234,  28,  38]],
```

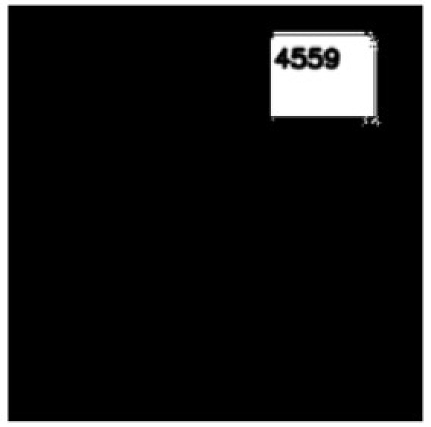
Now I am putting the Red pixel values in list1 array, Green pixel values in list2 array and Blue pixel values in list3 array. In nested loop, I am comparing the pixel values of original image with the above lists. If comparison is true, the same pixel of a new image is made white and all other pixels of that new image are made black. Meantime pix variable is counting the number of pixels in the red object. By using openCV puttext() function, the number of pixels are shown inside the white region.

```
img4=np.zeros((img1.shape[0],img1.shape[0],3),np.uint8) #making an empty image to original image size using numpy
img4[:,:,0]=img1[:,:,0] #copying image to new variable
img4[:,:,1]=img1[:,:,1] #copying image to new variable
img4[:,:,2]=img1[:,:,2] #copying image to new variable
img5=np.zeros((img4.shape[0],img4.shape[0],3),np.uint8) #making an empty image to original image size using numpy
pix=0 #variable to count total number of pixels in an object ie red object
row=1 #row number where red object detection started
column=1 #column number where red object detection started
check=0 #variable to only put values in row, column variables once in loop
list1=img4[25:80,190:260,0] #intensities of Red ie first channel pixels from original image specific range
list2=img4[25:80,190:260,1] #intensities of Green ie secong channel pixels from original image specific range
list3=img4[25:80,190:260,2] #intensities of Blue ie third channel pixels from original image specific range
for i in range(len(img4)):
    for j in range(len(img4[i])): #nested loop
        if ((img4[i,j,0] in list1) and (img4[i,j,1] in list2) and (img4[i,j,2] in list3)): #comparing original image red pixel values with list1 and green pixel 
values with list2 and blue pixel values with list3
            img5[i,j,0]=255 #if comparison is true then make that pixel white in copied image
            img5[i,j,1]=255 #if comparison is true then make that pixel white in copied image
            img5[i,j,2]=255 #if comparison is true then make that pixel white in copied image
            pix=pix+1 #if comparison is true then counting number of pixels
            if(check==0): #if check variabe is 0 then enter the if condition
                row=i #row number where red object detection started
                column=j #column number where red object detection started
                check=1 #after this, if condition will never run again
        else:
            img5[i,j,0]=0 #if comparison is fales then make that pixel black in copied image
            img5[i,j,1]=0 #if comparison is fales then make that pixel black in copied image
            img5[i,j,2]=0 #if comparison is fales then make that pixel black in copied image
pix1=str(pix) #converting pix variable to string
font = cv2.FONT_HERSHEY_SIMPLEX #font type to show number of pixel
cv2.putText(img5,pix1,(column,row+25), font, 0.6, (0, 0, 0), 2, cv2.LINE_AA) #opecv function to number of pixels inside the white region in img5, img5 contains 
identified object
plt.imshow(img5) #showing img5 after object identification
plt.axis('off')
plt.show()
```

Input:



Output:

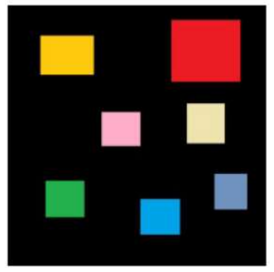


2nd Object:

First object is of green color. First of all I am cropping green object out of original image and then checking the pixel values.

```
#above code comments are valid line by line in this code also
plt.imshow(img4[208:240,48:88])
plt.axis('off')
plt.show()
img4[208:240,48:80]
```

Input:



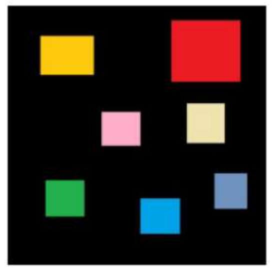
Output:

```
array([[[ 35, 177,  77],
        [ 35, 177,  77],
        [ 35, 177,  77],
        ...,
        [ 35, 177,  77],
        [ 35, 177,  77],
        [ 35, 177,  77]],
```

Now I am putting the Red pixel values in list1 array, Green pixel values in list2 array and Blue pixel values in list3 array. In nested loop, I am comparing the pixel values of original image with the above lists. If comparison is true, the same pixel of a new image is made white and all other pixels of that new image are made black. Meantime pix variable is counting the number of pixels in the red object. By using openCV puttext() function, the number of pixels are shown inside the white region.

```
#above code comments are valid line by line in this code also
img6=np.zeros((img4.shape[0],img4.shape[0],3),np.uint8)
pix=0
row=1
column=1
check=0
list1=img4[208:240,48:80,0]
list2=img4[208:240,48:80,1]
list3=img4[208:240,48:80,2]
for i in range(len(img4)):
    for j in range(len(img4[i])):
        if ((img4[i,j,0] in list1) and (img4[i,j,1] in list2) and (img4[i,j,2] in list3)):
            img6[i,j,0]=255
            img6[i,j,1]=255
            img6[i,j,2]=255
            pix=pix+1
            if(check==0):
                row=i
                column=j
                check=1
        else:
            img6[i,j,0]=0
            img6[i,j,1]=0
            img6[i,j,2]=0
pix1=str(pix)
font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(img6,pix1,(column,row+20), font, 0.6, (0, 0, 255), 2, cv2.LINE_AA)
plt.imshow(img6)
plt.axis('off')
plt.show()
```

Input:



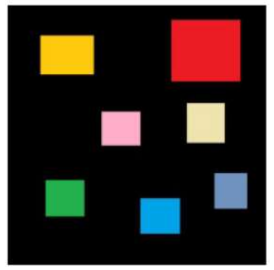
Output:

3rd Object:

Third object is of sky blue color. First of all I am cropping sky blue object out of original image and then checking the pixel values.

```
#above code comments are valid line by line in this code also
plt.imshow(img4[230:260,160:190])
plt.axis('off')
plt.show()
img4[230:260,160:190]
```

Input:



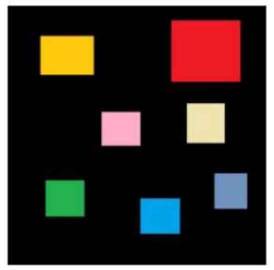
Output:

```
array([[[  0, 163, 232],
        [  0, 163, 232],
        [  0, 163, 232],
        ...,
        [  0, 163, 232],
        [  0, 163, 232],
        [  0, 163, 232]],
```

Now I am putting the Red pixel values in list1 array, Green pixel values in list2 array and Blue pixel values in list3 array. In nested loop, I am comparing the pixel values of original image with the above lists. If comparison is true, the same pixel of a new image is made white and all other pixels of that new image are made black. Meantime pix variable is counting the number of pixels in the red object. By using openCV puttext() function, the number of pixels are shown inside the white region.

```
#above code comments are valid line by line in this code also
img7=np.zeros((img4.shape[0],img4.shape[0],3),np.uint8)
pix=0
row=1
column=1
check=0
list1=img4[230:260,160:190,0]
list2=img4[230:260,160:190,1]
list3=img4[230:260,160:190,2]
for i in range(len(img4)):
    for j in range(len(img4[i])):
        if ((img4[i,j,0] in list1) and (img4[i,j,1] in list2) and (img4[i,j,2] in list3)):
            img7[i,j,0]=255
            img7[i,j,1]=255
            img7[i,j,2]=255
            pix=pix+1
            if(check==0):
                row=i
                column=j
                check=1
        else:
            img7[i,j,0]=0
            img7[i,j,1]=0
            img7[i,j,2]=0
pix1=str(pix)
font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(img7,pix1,(column,row+20), font, 0.6, (0, 0, 255), 2, cv2.LINE_AA)
plt.imshow(img7)
plt.axis('off')
plt.show()
```

Input:



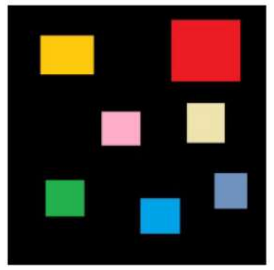
Output:

4th Object:

Fourth object is of pink color. First of all I am cropping pink object out of original image and then checking the pixel values.

```
#above code comments are valid line by line in this code also
plt.imshow(img4[130:160,110:150])
plt.axis('off')
plt.show()
img4[130:160,110:150]
```

Input:



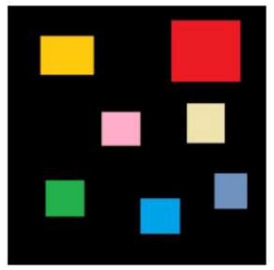
Output:

```
array([[[249, 177, 201],
        [255, 172, 201],
        [254, 174, 201],
        ...,
        [255, 172, 201],
        [255, 173, 201],
        [251, 175, 201]],
```

Now I am putting the Red pixel values in list1 array, Green pixel values in list2 array and Blue pixel values in list3 array. In nested loop, I am comparing the pixel values of original image with the above lists. If comparison is true, the same pixel of a new image is made white and all other pixels of that new image are made black. Meantime pix variable is counting the number of pixels in the red object. By using openCV puttext() function, the number of pixels are shown inside the white region.

```
#above code comments are valid line by line in this code also
img8=np.zeros((img4.shape[0],img4.shape[0],3),np.uint8)
pix=0
row=1
column=1
check=0
list1=img4[130:160,110:150,0]
list2=img4[130:160,110:150,1]
list3=img4[130:160,110:150,2]
for i in range(len(img4)):
    for j in range(len(img4[i])):
        if ((img4[i,j,0] in list1) and (img4[i,j,1] in list2) and (img4[i,j,2] in list3)):
            img8[i,j,0]=255
            img8[i,j,1]=255
            img8[i,j,2]=255
            pix=pix+1
            if(check==0):
                row=i
                column=j
                check=1
        else:
            img8[i,j,0]=0
            img8[i,j,1]=0
            img8[i,j,2]=0
pix1=str(pix)
font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(img8,pix1,(column-34,row+20), font, 0.6, (0, 0, 255), 2, cv2.LINE_AA)
plt.imshow(img8)
plt.axis('off')
plt.show()
```

Input:



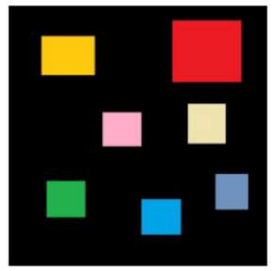
Output:

5th Object:

Fifth object is of skin color. First of all I am cropping skin object out of original image and then checking the pixel values.

```
#above code comments are valid line by line in this code also
plt.imshow(img4[120:150,210:250])
plt.axis('off')
plt.show()
img4[120:150,210:250]
```

Input:



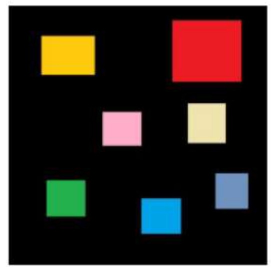
Output:

```
array([[[239, 228, 172],
        [239, 228, 172],
        [239, 228, 172],
        ...,
        [238, 228, 175],
        [238, 228, 177],
        [239, 228, 172]],
```

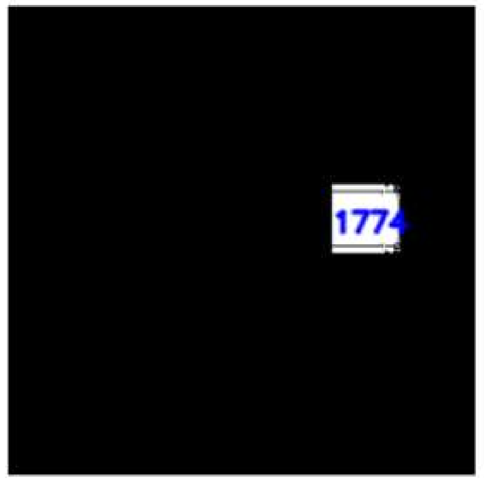
Now I am putting the Red pixel values in list1 array, Green pixel values in list2 array and Blue pixel values in list3 array. In nested loop, I am comparing the pixel values of original image with the above lists. If comparison is true, the same pixel of a new image is made white and all other pixels of that new image are made black. Meantime pix variable is counting the number of pixels in the red object. By using openCV puttext() function, the number of pixels are shown inside the white region.

```
#above code comments are valid line by line in this code also
img9=np.zeros((img4.shape[0],img4.shape[0],3),np.uint8)
pix=0
row=1
column=1
check=0
list1=img4[120:150,210:250,0]
list2=img4[120:150,210:250,1]
list3=img4[120:150,210:250,2]
for i in range(len(img4)):
    for j in range(len(img4[i])):
        if ((img4[i,j,0] in list1) and (img4[i,j,1] in list2) and (img4[i,j,2] in list3)):
            img9[i,j,0]=255
            img9[i,j,1]=255
            img9[i,j,2]=255
            pix=pix+1
            if(check==0):
                row=i
                column=j
                check=1
        else:
            img9[i,j,0]=0
            img9[i,j,1]=0
            img9[i,j,2]=0
pix1=str(pix)
font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(img9,pix1,(column,row+30), font, 0.6, (0, 0, 255), 2, cv2.LINE_AA)
plt.imshow(img9)
plt.axis('off')
plt.show()
```

Input:



Output:

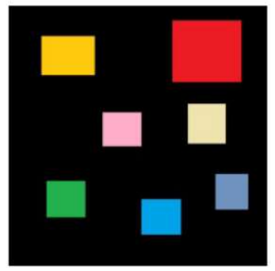


6th Object:

Sixth object is of light blue color. First of all I am cropping light blue object out of original image and then checking the pixel values.

```
plt.imshow(img4[200:235,240:270])
plt.axis('off')
plt.show()
img4[200:235,240:270]
```

Input:



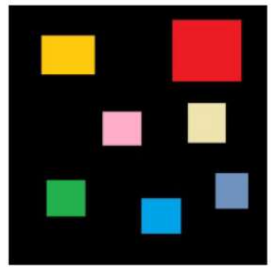
Output:

```
array([[[112, 146, 191],
        [112, 146, 191],
        [112, 146, 191],
        ...,
        [112, 146, 191],
        [112, 146, 191],
        [112, 146, 191]],
```

Now I am putting the Red pixel values in list1 array, Green pixel values in list2 array and Blue pixel values in list3 array. In nested loop, I am comparing the pixel values of original image with the above lists. If comparison is true, the same pixel of a new image is made white and all other pixels of that new image are made black. Meantime pix variable is counting the number of pixels in the red object. By using openCV puttext() function, the number of pixels are shown inside the white region.

```
img10=np.zeros((img4.shape[0],img4.shape[0],3),np.uint8)
pix=0
row=1
column=1
check=0
list1=img4[200:235,240:270,0]
list2=img4[200:235,240:270,1]
list3=img4[200:235,240:270,2]
for i in range(len(img4)):
    for j in range(len(img4[i])):
        if ((img4[i,j,0] in list1) and (img4[i,j,1] in list2) and (img4[i,j,2] in list3)):
            img10[i,j,0]=255
            img10[i,j,1]=255
            img10[i,j,2]=255
            pix=pix+1
            if(check==0):
                row=i
                column=j
                check=1
        else:
            img10[i,j,0]=0
            img10[i,j,1]=0
            img10[i,j,2]=0
pix1=str(pix)
font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(img10,pix1,(column-40,row+20), font, 0.6, (0, 0, 255), 2, cv2.LINE_AA)
plt.imshow(img10)
plt.axis('off')
plt.show()
```

Input:



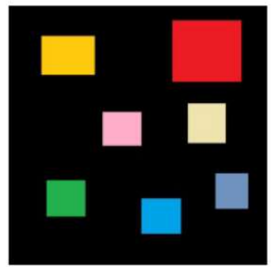
Output:

7th Object:

Seventh object is of yellow color. First of all I am cropping yellow object out of original image and then checking the pixel values.

```
#above code comments are valid line by line in this code also
plt.imshow(img4[40:70,40:100])
plt.axis('off')
plt.show()
img4[40:60,40:70]
```

Input:



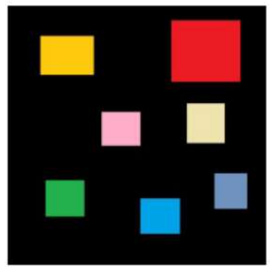
Output:

```
array([[[255, 200,   1],
        [255, 198,  31],
        [253, 199,  31],
        ...,
        [253, 202,  15],
        [253, 202,  15],
        [253, 202,  15]],
```

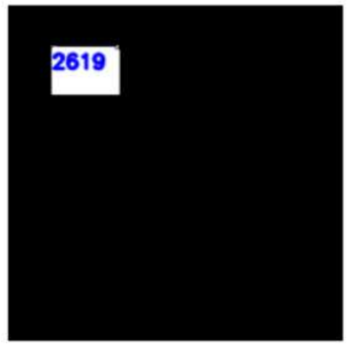
Now I am putting the Red pixel values in list1 array, Green pixel values in list2 array and Blue pixel values in list3 array. In nested loop, I am comparing the pixel values of original image with the above lists. If comparison is true, the same pixel of a new image is made white and all other pixels of that new image are made black. Meantime pix variable is counting the number of pixels in the red object. By using openCV puttext() function, the number of pixels are shown inside the white region.

```
#above code comments are valid line by line in this code also
img11=np.zeros((img4.shape[0],img4.shape[0],3),np.uint8)
pix=0
row=1
column=1
check=0
list1=img4[40:70,40:100,0]
list2=img4[40:70,40:100,1]
list3=img4[40:70,40:100,2]
for i in range(len(img4)):
    for j in range(len(img4[i])):
        if ((img4[i,j,0] in list1) and (img4[i,j,1] in list2) and (img4[i,j,2] in list3)):
            img11[i,j,0]=255
            img11[i,j,1]=255
            img11[i,j,2]=255
            pix=pix+1
            if(check==0):
                row=i
                column=j
                check=1
        else:
            img11[i,j,0]=0
            img11[i,j,1]=0
            img11[i,j,2]=0
pix1=str(pix)
font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(img11,pix1,(column,row+20), font, 0.6, (0, 0, 255), 2, cv2.LINE_AA)
plt.imshow(img11)
plt.axis('off')
plt.show()
```

Input:



Output:



c)

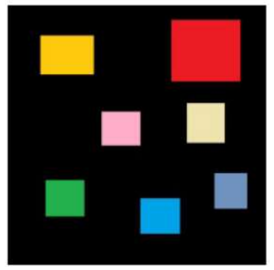
Now, I will do the same work using openCV intensity slicing. From B part, I already know the pixel range of each channel in the original image of each object. I this method I will only check if each channel pixel lies in specific range or not. If it lies in specific range then make this pixel white otherwise make it black.

1st Object:

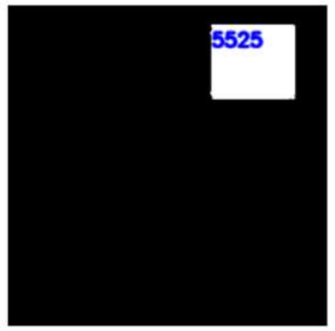
First Object is of red color. I already know the pixel ranges of red object. I will only check if each channel pixel lies in specific range or not. If it lies in specific range then make this pixel white otherwise make it black. . Meantime pix variable is counting the number of pixels in the red object. By using openCV puttext() function, the number of pixels are shown inside the white region.

```
img12 = np.zeros((img4.shape[0],img4.shape[0],3),np.uint8) #making an empty image to original image size using numpy
Rmin_range = 199 #Red Pixel Minimum value
Rmax_range = 256 #Red Pixel Maximum value, less operator is used in loop thats why 256 is written
Gmin_range = 12 #Green Pixel Minimum value
Gmax_range = 51 #Green Pixel Maximum value
Bmin_range = 23 #Blue Pixel Minimum value
Bmax_range = 54 #BLue Pixel Maximum value
row=1 #row number where red object detection started
column=1 #column number where red object detection started
check=0 #variable to only put values in row, column variables once in loop
pix=0 #variable to count total number of pixels in an object ie red object
for i in range(len(img4)):
    for j in range(len(img4[i])): #nested loops
        if ((img4[i,j,0]>Rmin_range and img4[i,j,0]<Rmax_range)  and (img4[i,j,1]>Gmin_range and img4[i,j,1]<Gmax_range) and (img4[i,j,2]>Bmin_range and
img4[i,j,2] < Bmax_range)): #if original pixel is in range of the above defined variables then condition is true
            img12[i,j,0] = 255 #if comparison is true then make that pixel white in copied image
            img12[i,j,1] = 255 #if comparison is true then make that pixel white in copied image
            img12[i,j,2] = 255 #if comparison is true then make that pixel white in copied image
            pix=pix+1 #if comparison is true then counting number of pixels
            if(check==0): #if check variabe is 0 then enter the if condition
                row=i #row number where red object detection started
                column=j #column number where red object detection started
                check=1 #after this, if condition will never run again
        else:
            img12[i,j,0] = 0 #if comparison is fales then make that pixel black in copied image
            img12[i,j,1] = 0 #if comparison is fales then make that pixel black in copied image
            img12[i,j,2] = 0 #if comparison is fales then make that pixel black in copied image
pix1=str(pix) #converting pix variable to string
font = cv2.FONT_HERSHEY_SIMPLEX #font type to show number of pixel
cv2.putText(img12,pix1,(column,row+20), font, 0.6, (0, 0, 255), 2, cv2.LINE_AA) #opecv function to number of pixels inside the white region in img12, img12
contains identified object
plt.imshow(img12) #showing img12 after object identification
plt.axis('off')
plt.show()
```

Input:



Output:



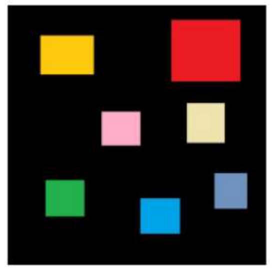
2nd Object:

First Object is of green color. I already know the pixel ranges of green object. I will only check if each channel pixel lies in specific range or not. If it lies in specific range then make this pixel white otherwise make it black. . Meantime pix variable is counting the number of pixels in the red object. By using openCV puttext() function, the number of pixels are shown inside the white region.

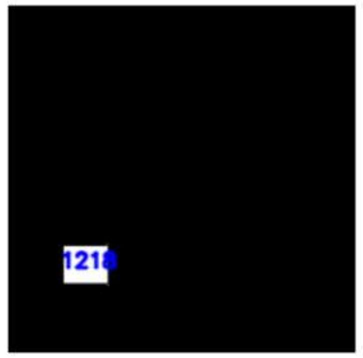
```
#above code comments are valid line by line in this code also
img13 = np.zeros((img4.shape[0],img4.shape[0],3),np.uint8)
Rmin_range = 30
Rmax_range = 40
Gmin_range = 175
Gmax_range = 180
Bmin_range = 75
Bmax_range = 80
row=1
column=1
check=0
pix=0
for i in range(len(img4)):
    for j in range(len(img4[i])):
        if ((img4[i,j,0]>Rmin_range and img4[i,j,0]<Rmax_range)  and (img4[i,j,1]>Gmin_range and img4[i,j,1]<Gmax_range) and (img4[i,j,2]>Bmin_range and
img4[i,j,2] < Bmax_range)):
            img13[i,j,0] = 255
            img13[i,j,1] = 255
            img13[i,j,2] = 255
            pix=pix+1
```

```
                if(check==0):
                    row=i
                    column=j
                    check=1
            else:
                img13[i,j,0] = 0
                img13[i,j,1] = 0
                img13[i,j,2] = 0
pix1=str(pix)
font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(img13,pix1,(column-40,row+20), font, 0.6, (0, 0, 255), 2, cv2.LINE_AA)
plt.imshow(img13)
plt.axis('off')
plt.show()
```

Input:



Output:



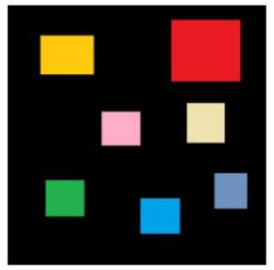
3rd Object:

Third Object is of sky blue color. I already know the pixel ranges of sky blue object. I will only check if each channel pixel lies in specific range or not. If it lies in specific range then make this pixel white otherwise make it black. . Meantime pix variable is counting the number of pixels in the red object. By using openCV puttext() function, the number of pixels are shown inside the white region.

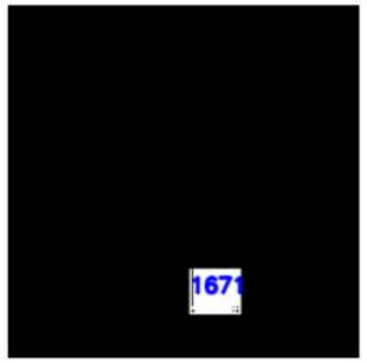
```
#above code comments are valid line by line in this code also
img14 = np.zeros((img4.shape[0],img4.shape[0],3),np.uint8)
Rmin_range = -5
Rmax_range = 10
Gmin_range = 150
Gmax_range = 256
Bmin_range = 200
```

```
Bmax_range = 256
row=1
column=1
check=0
pix=0
for i in range(len(img4)):
    for j in range(len(img4[i])):
        if ((img4[i,j,0]>Rmin_range and img4[i,j,0]<Rmax_range)  and (img4[i,j,1]>Gmin_range and img4[i,j,1]<Gmax_range) and (img4[i,j,2]>Bmin_range and
img4[i,j,2] < Bmax_range)):
            img14[i,j,0] = 255
            img14[i,j,1] = 255
            img14[i,j,2] = 255
            pix=pix+1
            if(check==0):
                row=i
                column=j
                check=1
        else:
            img14[i,j,0] = 0
            img14[i,j,1] = 0
            img14[i,j,2] = 0
pix1=str(pix)
font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(img14,pix1,(column,row+20), font, 0.6, (0, 0, 255), 2, cv2.LINE_AA)
plt.imshow(img14)
plt.axis('off')
plt.show()
```

Input:



Output:

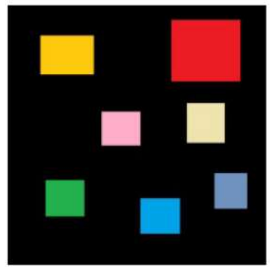


4th Object:

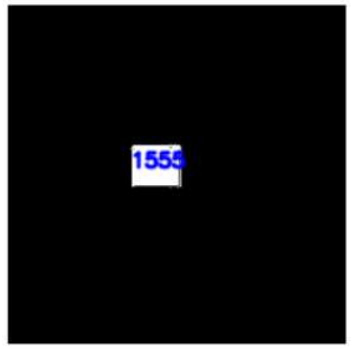
Fourth Object is of pink color. I already know the pixel ranges of pink object. I will only check if each channel pixel lies in specific range or not. If it lies in specific range then make this pixel white otherwise make it black. . Meantime pix variable is counting the number of pixels in the red object. By using openCV puttext() function, the number of pixels are shown inside the white region.

```
#above code comments are valid line by line in this code also
img15 = np.zeros((img4.shape[0],img4.shape[0],3),np.uint8)
Rmin_range = 239
Rmax_range = 256
Gmin_range = 170
Gmax_range = 189
Bmin_range = 195
Bmax_range = 210
row=1
column=1
check=0
pix=0
for i in range(len(img4)):
    for j in range(len(img4[i])):
        if ((img4[i,j,0]>Rmin_range and img4[i,j,0]<Rmax_range)  and (img4[i,j,1]>Gmin_range and img4[i,j,1]<Gmax_range) and (img4[i,j,2]>Bmin_range and
img4[i,j,2] < Bmax_range)):
            img15[i,j,0] = 255
            img15[i,j,1] = 255
            img15[i,j,2] = 255
            pix=pix+1
            if(check==0):
                row=i
                column=j
                check=1
        else:
            img15[i,j,0] = 0
            img15[i,j,1] = 0
            img15[i,j,2] = 0
pix1=str(pix)
font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(img15,pix1,(column-35,row+20), font, 0.6, (0, 0, 255), 2, cv2.LINE_AA)
plt.imshow(img15)
plt.axis('off')
plt.show()
```

Input:



Output:

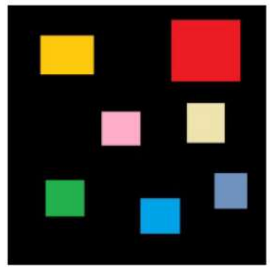


5th Object:

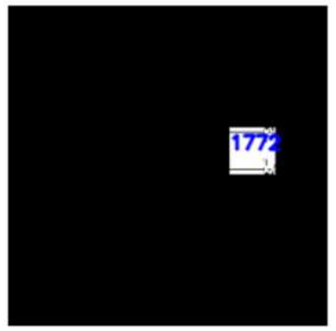
Fifth Object is of skin color. I already know the pixel ranges of skin object. I will only check if each channel pixel lies in specific range or not. If it lies in specific range then make this pixel white otherwise make it black. . Meantime pix variable is counting the number of pixels in the red object. By using openCV puttext() function, the number of pixels are shown inside the white region.

```
#above code comments are valid line by line in this code also
img16 = np.zeros((img4.shape[0],img4.shape[0],3),np.uint8)
Rmin_range = 235
Rmax_range = 245
Gmin_range = 225
Gmax_range = 235
Bmin_range = 168
Bmax_range = 185
row=1
column=1
check=0
pix=0
for i in range(len(img4)):
    for j in range(len(img4[i])):
        if ((img4[i,j,0]>Rmin_range and img4[i,j,0]<Rmax_range)  and (img4[i,j,1]>Gmin_range and img4[i,j,1]<Gmax_range) and (img4[i,j,2]>Bmin_range and 
img4[i,j,2] < Bmax_range)):
            img16[i,j,0] = 255
            img16[i,j,1] = 255
            img16[i,j,2] = 255
            pix=pix+1
            if(check==0):
                row=i
                column=j
                check=1
        else:
            img16[i,j,0] = 0
            img16[i,j,1] = 0
            img16[i,j,2] = 0
pix1=str(pix)
font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(img16,pix1,(column,row+20), font, 0.6, (0, 0, 255), 2, cv2.LINE_AA)
plt.imshow(img16)
plt.axis('off')
plt.show()
```

Input:



Output:

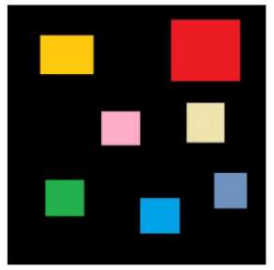


6th Object:

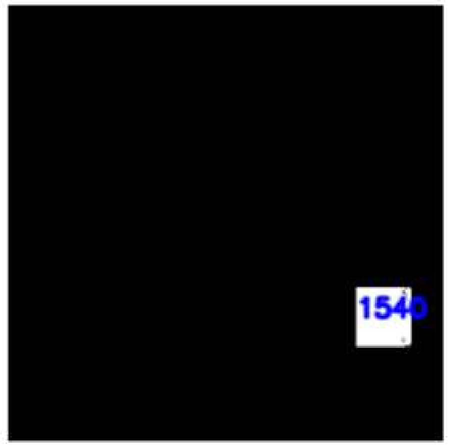
Sixth Object is of light blue color. I already know the pixel ranges of light object. I will only check if each channel pixel lies in specific range or not. If it lies in specific range then make this pixel white otherwise make it black. . Meantime pix variable is counting the number of pixels in the red object. By using openCV puttext() function, the number of pixels are shown inside the white region.

```
#above code comments are valid line by line in this code also
img17 = np.zeros((img4.shape[0],img4.shape[0],3),np.uint8)
Rmin_range = 105
Rmax_range = 125
Gmin_range = 140
Gmax_range = 150
Bmin_range = 150
Bmax_range = 256
row=1
column=1
check=0
pix=0
for i in range(len(img4)):
    for j in range(len(img4[i])):
        if ((img4[i,j,0]>Rmin_range and img4[i,j,0]<Rmax_range)  and (img4[i,j,1]>Gmin_range and img4[i,j,1]<Gmax_range) and (img4[i,j,2]>Bmin_range and 
img4[i,j,2] < Bmax_range)):
            img17[i,j,0] = 255
            img17[i,j,1] = 255
            img17[i,j,2] = 255
            pix=pix+1
            if(check==0):
                row=i
                column=j
                check=1
        else:
            img17[i,j,0] = 0
            img17[i,j,1] = 0
            img17[i,j,2] = 0
pix1=str(pix)
font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(img17,pix1,(column,row+20), font, 0.6, (0, 0, 255), 2, cv2.LINE_AA)
plt.imshow(img17)
plt.axis('off')
plt.show()
```

Input:



Output:



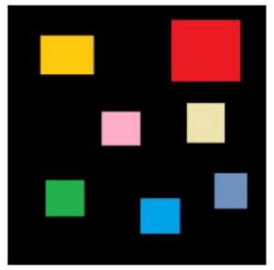
7th Object:

Seventh Object is of yellow color. I already know the pixel ranges of yellow object. I will only check if each channel pixel lies in specific range or not. If it lies in specific range then make this pixel white otherwise make it black. . Meantime pix variable is counting the number of pixels in the red object. By using openCV puttext() function, the number of pixels are shown inside the white region.

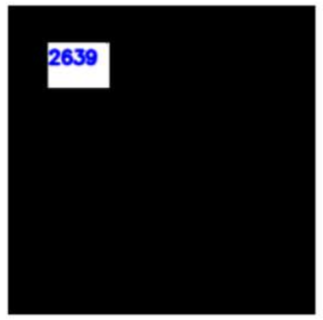
```
#above code comments are valid line by line in this code also
img18 = np.zeros((img4.shape[0],img4.shape[0],3),np.uint8)
Rmin_range = 200
Rmax_range = 256
Gmin_range = 190
Gmax_range = 210
```

```
Bmin_range = -1
Bmax_range = 40
row=1
column=1
check=0
pix=0
for i in range(len(img4)):
    for j in range(len(img4[i])):
        if ((img4[i,j,0]>Rmin_range and img4[i,j,0]<Rmax_range)  and (img4[i,j,1]>Gmin_range and img4[i,j,1]<Gmax_range) and (img4[i,j,2]>Bmin_range and 
img4[i,j,2] < Bmax_range)):
            img18[i,j,0] = 255
            img18[i,j,1] = 255
            img18[i,j,2] = 255
            pix=pix+1
            if(check==0):
                row=i
                column=j
                check=1
        else:
            img18[i,j,0] = 0
            img18[i,j,1] = 0
            img18[i,j,2] = 0
pix1=str(pix)
font = cv2.FONT_HERSHEY_SIMPLEX
cv2.putText(img18,pix1,(column,row+20), font, 0.6, (0, 0, 255), 2, cv2.LINE_AA)
plt.imshow(img18)
plt.axis('off')
plt.show()
```

Input:



Output:



Question 2:

a) Read the given image and convert it to grayscale image.
b) Use thresholding to obtain only hand
c) Find the tallest figure in the image and draw a circle on that finger's tip. A sample input image and a sample output is shown in the Figure 2

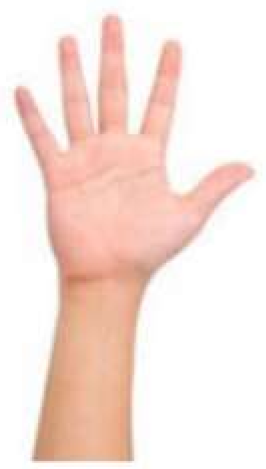
Solution:

a)

Reading image and converting it to RGB format from BGR.

```
path = r'F:\Study\Fall 2021\DIP\Assignments\1\Assignment#1\img2.jpg' #image path
img = cv2.imread(path,cv2.IMREAD_COLOR) #reading image by opencv
img=cv2.cvtColor(img,cv2.COLOR_BGR2RGB) #converting to RGB
plt.imshow(img) #output
plt.axis('off')
plt.show()
```

Output:



Now calling the above function to convert it to grayscale image

```
img1=np.zeros((img.shape[0],img.shape[1],3),np.uint8) #making an empty image to original image size using numpy
img1[:,:,0]=img[:,:,0] #copying image to new variable
img1[:,:,1]=img[:,:,1] #copying image to new variable
img1[:,:,2]=img[:,:,2] #copying image to new variable
greyimage=rgb_to_grayscale(img1) #calling grey scale function defined above
plt.imshow(greyimage,cmap='gray') #showing greyscale image
plt.axis('off')
plt.show()
```

Output:

Question 3:

Read any grayscale image and perform following operations with various range of possible values for each method and show their outputs. Also explain the effects of each variable such as C and gamma on the output of the image

- Log and Inverse Log transform
- Power law nth power and nth root
- Power Law transformation

Solution:

Reading grayscale image and printing it.

```
path1 = r'F:\Study\Fall 2021\DIP\Assignments\1\Assignment#1\img3.png' #image path
img30 = cv2.imread(path1,cv2.IMREAD_GRAYSCALE) #reading image by opencv
plt.imshow(img30, cmap='gray') #output
plt.axis('off')
plt.show()
```

Output:

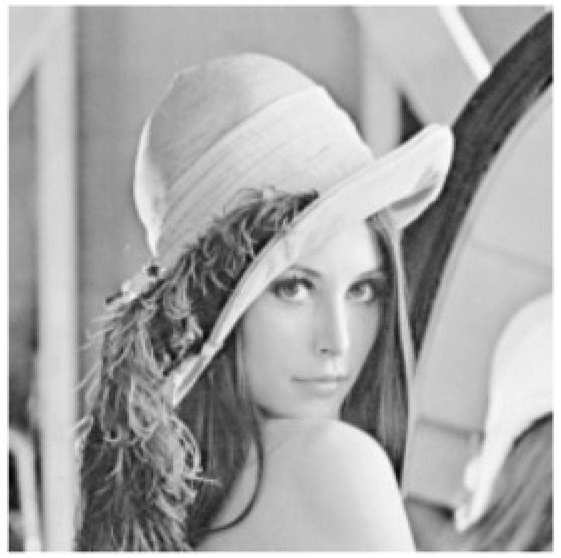
Log Transformation:

This method makes the darker pixel brighter and brighter pixel a little bit brighter.

```
img31=np.zeros((img30.shape[0],img30.shape[1]),np.uint8) #making an empty image to original image size using numpy
img31[:,:]=img30[:,:] #copying image to new variable
c=255/(np.log(1+255)) #setting the value of constant
for i in range(len(img30)):
    for j in range(len(img30[i])): #nested loop
        img31[i][j]=c*m.log(1+img30[i][j]) #applying log on each pixel
plt.imshow(img31, cmap='gray') #output
plt.axis('off')
plt.show()
```

Input:

Output:



Inverse Log Transformation:

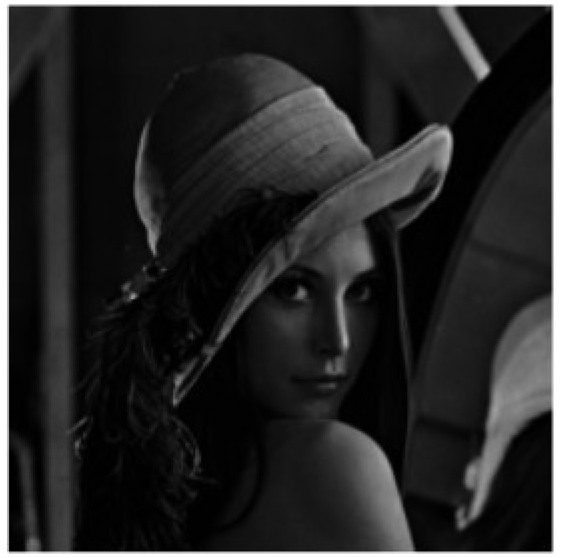
This method makes the darker pixel a little darker and brighter pixels into dark pixels.

```
img32=np.zeros((img30.shape[0],img30.shape[1]),np.uint8) #making an empty image to original image size using numpy
c=255/(np.log(1+255)) #setting the value of constant
```

```
for i in range(len(img30)):
    for j in range(len(img30[i])): #nested loop
        img32[i][j]=np.exp(img30[i][j]**1/c)-1 #applying inverse log on each pixel
plt.imshow(img32, cmap='gray') #output
plt.axis('off')
plt.show()
```

Input:

Output:



Power Law Transformations:

(i) Nth Power Transformation

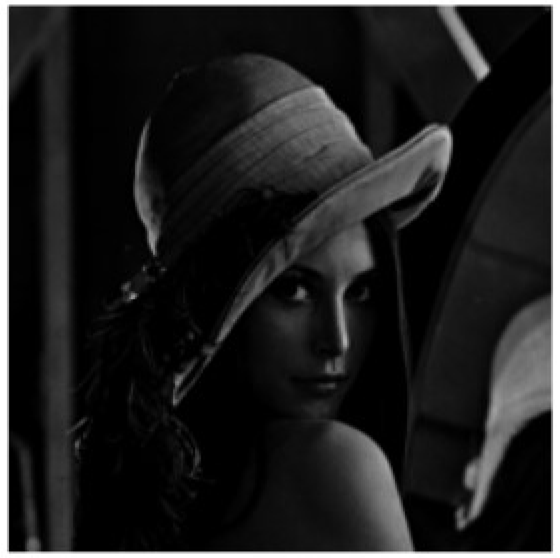
This method converts image into brighter image and darker image depending on value of gamme r.

```
img33=np.zeros((img30.shape[0],img30.shape[1]),np.uint8) #making an empty image to original image size using numpy
#img32[:,:]=img30[:,:] #copying image to new variable
for i in range(len(img30)):
    for j in range(len(img30[i])):
        img33[i][j]=255*((img30[i][j]/255)**5) #applying nth power on each pixel,5 is the power value
plt.imshow(img33, cmap='gray') #output
plt.axis('off')
plt.show()
```

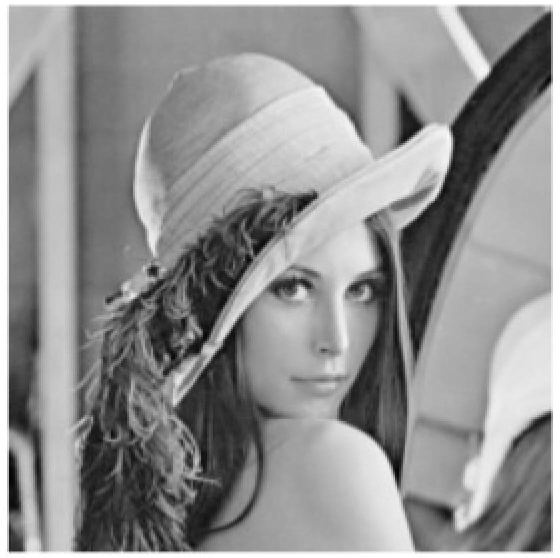
Input:

Output:

Gamma = 5 (darker result)



Gamma = 0.5 (brighter result)



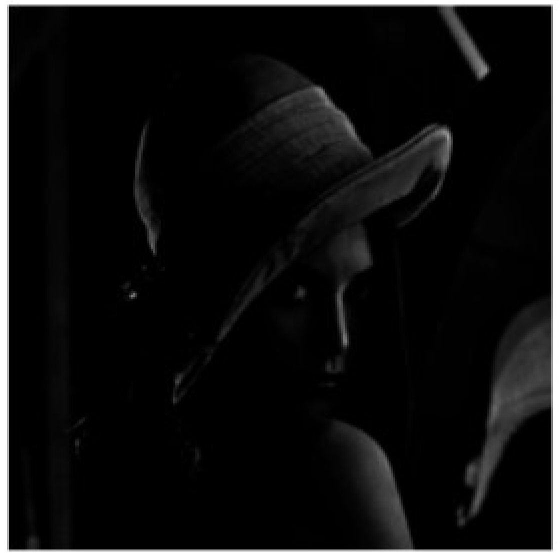
(ii) Nth Root Transformation

This method converts image into darker image and brighter image depending on value of gamme r.

```
img34=np.zeros((img30.shape[0],img30.shape[1]),np.uint8) #making an empty image to original image size using numpy
#img32[:,:]=img30[:,:] #copying image to new variable
for i in range(len(img30)):
    for j in range(len(img30[i])):
        img34[i][j]=255*np.power((img30[i][j]/255),(1/0.1)) #appllying nth root on each pixel, here 0.1 is the root value
plt.imshow(img34, cmap='gray') #output
plt.axis('off')
plt.show()
```

Input:

Output:
Gamma = 0.1 (darker result)



Gamma = 2 (brighter result)