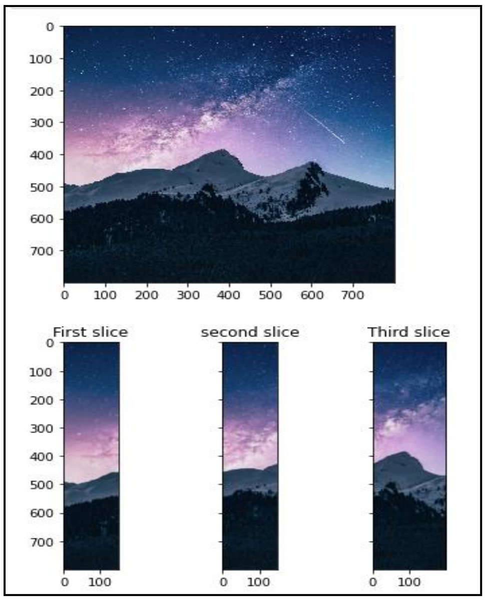
1.Program to perform :

a)Image slicing.

```
from skimage.io import imshow,imread
import matplotlib.pyplot as plt
img1=imread('night.jpg')
imshow(img1)

fig,ax=plt.subplots(1,3,figsize=(6,4), sharey=True)
ax[0].imshow(img1[:, 0:150])
ax[0].set_title('First slice')

ax[1].imshow(img1[:, 150:300])
ax[1].set_title('second slice')

ax[2].imshow(img1[:, 300:500])
ax[2].set_title('Third slice');
```

Output:



2.Develop a program to perform:

i) Sampling of an image (up and down sampling)

ii) Median filtering

iii) Average filtering

iv) Interpolation

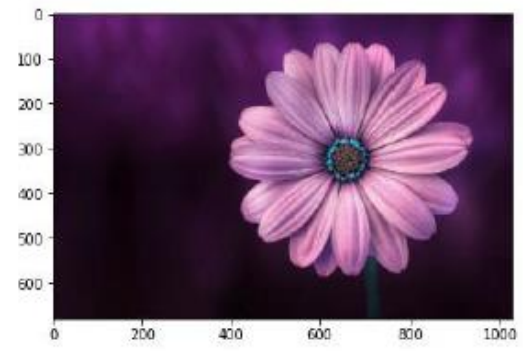
v) Quantization

CODE:

i)Sampling of an image (up and down sampling)

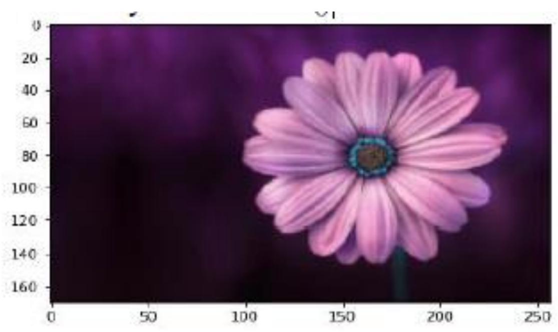
UP SAMPLING:

```
import cv2
from matplotlib import pyplot as plt
image = cv2.imread('flower1.jpg')
cv2.imshow("image before pyrUp: ",image)
image1 = cv2.pyrUp(image)
#cv2.imshow("image after pyrUp: ", image)
cv2.imshow('UpSample', image1)
plt.imshow(image1)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



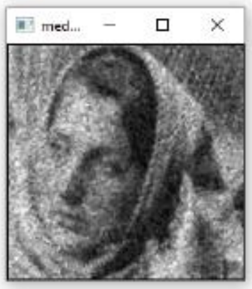
DOWN SAMPLING:

```
import cv2
from matplotlib import pyplot as plt
image = cv2.imread('flower1.jpg')
cv2.imshow("image before pyrDown: ",image)
image1 = cv2.pyrDown(image)
#cv2.imshow("image after pyrDown: ", image1)
cv2.imshow('DownSample', image1)
plt.imshow(image1)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



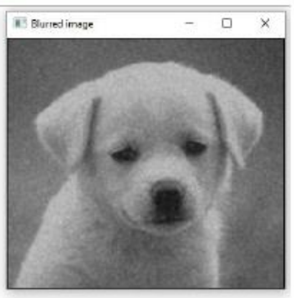
ii)MEDIAN FILTERING

```
import cv2
import numpy as np
img_noisy1=cv2.imread("filter.png",0)
m,n=img_noisy1.shape
img_new1=np.zeros([m,n])
for i in range(1,m-1):
    for j in range(1,n-1):
        temp=[img_noisy1[i-1,j-1],
                img_noisy1[i-1,j],
                img_noisy1[i-1,j+1],
                img_noisy1[i,j-1],
                img_noisy1[i,j],
                  img_noisy1[i,j+1],
              img_noisy1[i+1,j-1],
            img_noisy1[i+1,j],

           img_noisy1[i+1,j+1]]
        temp=sorted(temp)
        img_new1[i,j]=temp[4]
        img_new1=img_new1.astype(np.uint8)
cv2.imshow("MEDIAN FILTERED IMAGE",img_new1)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



iii)AVERAGE FILTERING

```
import cv2
import numpy as np
img=cv2.imread("filter.png",0)
m,n=img.shape
mask=np.ones([3,3],dtype=int)
mask=mask/9
img_new=np.zeros([m,n])
for i in range(1,m-1):
     for j in range(1,n-1):
        temp=img[i-1, j-1]*mask[0, 0]+img[i-1, j]*mask[0, 1]+img[i-1, j + 1]*mask[0, 2]+img[i, j-1]*mask[1, 0]+ img[i, j]*mask[1, 1]+img[i, j + 1]*mask[1, 2]+img[i + 1, j-1]*mask[2, 0]+img[i + 1, j]*mask[2, 1]+img[i + 1, j + 1]*mask[2, 2]
        img_new[i, j]= temp
img_new=img_new.astype(np.uint8)
cv2.imshow("Blurred image",img_new)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

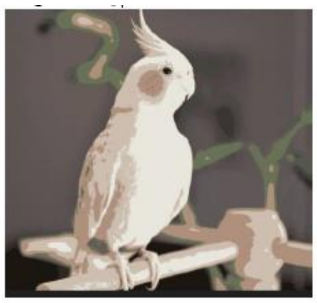


iv)Interpolation:

```
import cv2
import numpy as np
image = cv2.imread('night.jpg')
nearest=cv2.resize(image,None,fx=25,fy=25,interpolation=cv2.INTER_NEAREST)
cv2.imshow('Nearest',nearest)
image2 = cv2.imread('night.jpg')
linear=cv2.resize(image2,None,fx=5,fy=5,interpolation=cv2.INTER_LINEAR)
cv2.imshow('LINEAR',linear)
image3 = cv2.imread('night.jpg')
bicubic=cv2.resize(image3,None,fx=5,fy=5,interpolation=cv2.INTER_CUBIC)
cv2.imshow('BICUBIC',bicubic)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

v)Quantization:

```
import cv2
from PIL import Image
image = Image.open('bird3.jpg')
img=image.quantize(19)
img.show()
```

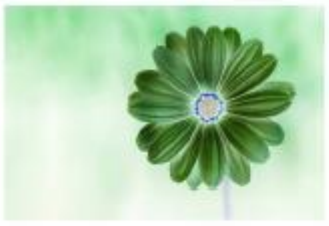


3.Write a program to perform basic image data analysis using intensity transformation:
a) Image negative b) Log transformation c) Gamma correction
<u>CODE:</u>

```
!pip install imageio
%matplotlib inline
import imageio
import matplotlib.pyplot as plt
import warnings
import matplotlib.cbook
warnings.filterwarnings("ignore",category=matplotlib.cbook.mplDeprecation)
pic=imageio.imread('flower1.jpg')
plt.figure(figsize=(6,6))
plt.imshow(pic)
plt.axis('off')
```

#NEGATING IMAGE

```
negative=255-pic
plt.figure(figsize=(6,6))
plt.imshow(negative)
        plt.axis('off')
```

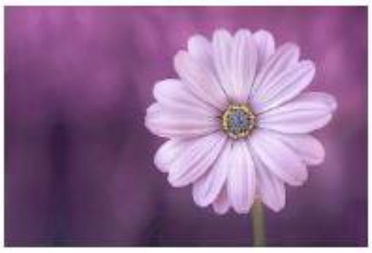


#LOG TRANSFORM

```
%matplotlib inline
import imageio
import numpy as np
import matplotlib.pyplot as plt
pic=imageio.imread('flower1.jpg')
gray=lambda rgb:np.dot(rgb[...,:3],[0.299,0.587,0.114])
gray=gray(pic)
max_=np.max(gray)
def log_transform():
        return(255/np.log(1+max_))*np.log(1+gray)
plt.figure(figsize=(5,5))
plt.imshow(log_transform(),cmap=plt.get_cmap(name='gray'))
plt.axis('off')
```

#GAMMA CORRECTION

```
import imageio
import matplotlib.pyplot as plt
pic=imageio.imread('flower1.jpg')
gamma=2.2
gamma_correction=((pic/255)**(1/gamma))
plt.figure(figsize=(5,5))
plt.imshow(gamma_correction)
plt.axis('off')
```

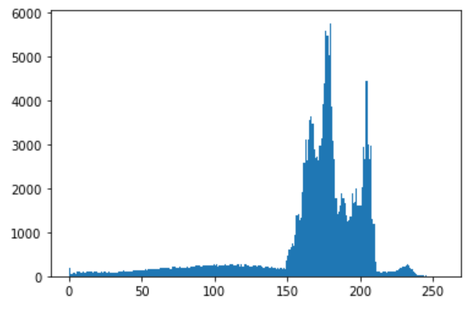


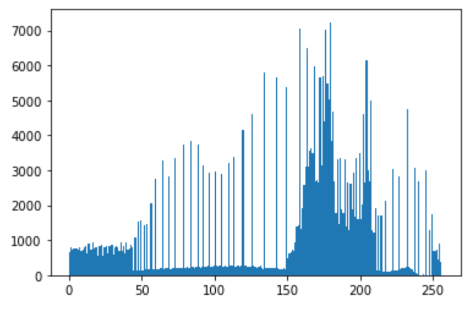
4.Write a program to perform histogram equalization of an image.

CODE:

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('king.jpg',0)
plt.hist(img.ravel(),256,[0,256])

plt.show()
plt.savefig('hist.png')

equ = cv2.equalizeHist(img)
res = np.hstack((img,equ))

cv2.imshow('Equalized Image',res)
cv2.imwrite('Equalized Image.png',res)

plt.hist(res.ravel(),256,[0,256])

plt.show()
plt.savefig('equal-hist.png')
```

Output:





```
<Figure size 432x288 with 0 Axes>
```

5.Develop a program to iterate through the folders and read all the image files and display its name respectively

CODE:

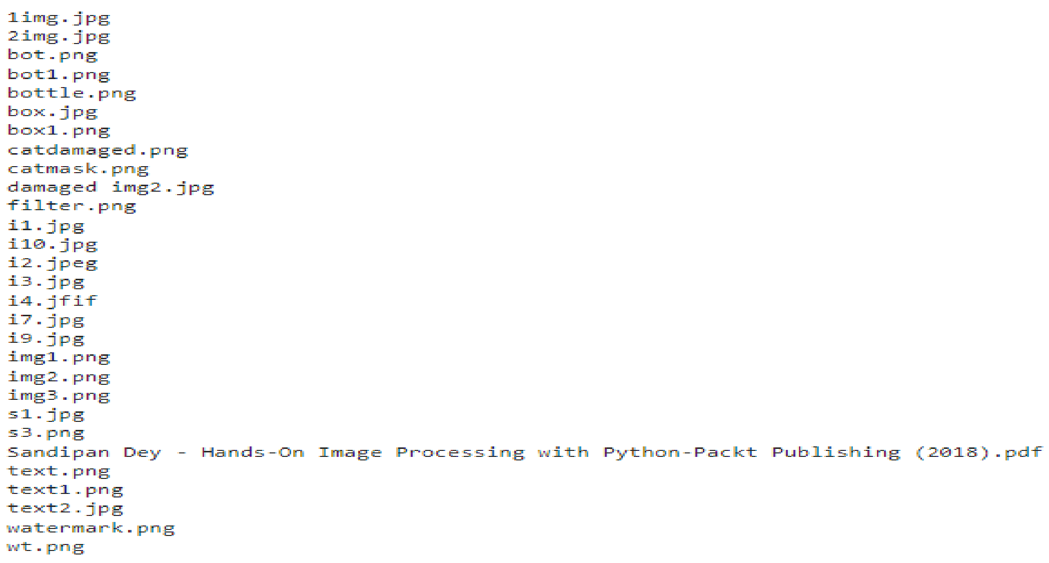
```
#Listing images that ends with .png
import os
from os import listdir

# get the path/directory
folder_dir = "D:/images"
for images in os.listdir(folder_dir):

    # check if the image ends with png
    if (images.endswith(".png")):
        print(images)
```

Output:

```
bot.png
bot1.png
bottle.png
box1.png
catdamaged.png
catmask.png
filter.png
img1.png
img2.png
img3.png
s3.png
text.png
text1.png
watermark.png
wt.png
```

```
#Listing all the images from the directory
import os
from os import listdir

# get the path or directory
folder_dir = "D:/images"
for images in os.listdir(folder_dir):
            print(images)
```

Output:



```
1img.jpg
2img.jpg
bot.png
bot1.png
bottle.png
box.jpg
box1.png
catdamaged.png
catmask.png
damaged img2.jpg
filter.png
i1.jpg
i10.jpg
i2.jpeg
i3.jpg
i4.jfif
i7.jpg
i9.jpg
img1.png
img2.png
img3.png
s1.jpg
s3.png
Sandipan Dey - Hands-On Image Processing with Python-Packt Publishing (2018).pdf
text.png
text1.png
text2.jpg
watermark.png
wt.png
```

6.Write a program to develop montage of images.

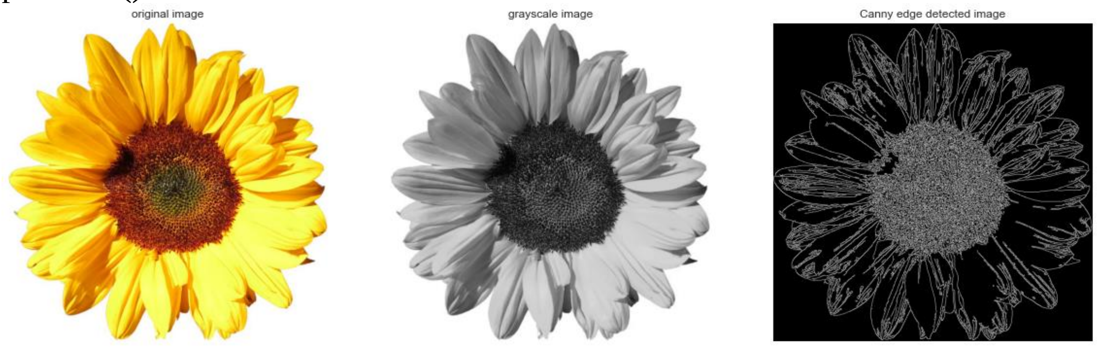
<u>CODE:</u>

```
import skimage.io
import skimage.util

a = skimage.io.imread('flower1.jpg')
print(a.shape)
# (225, 400, 3)

b = a // 2
c = a // 3
d=a//4
m = skimage.util.montage([a, b, c,d], multichannel=True)
print(m.shape)
# (450, 800, 3)

skimage.io.imsave('skimage_montage_default.jpg', m)
```

7.Develop a program to perform various edge detection programs.

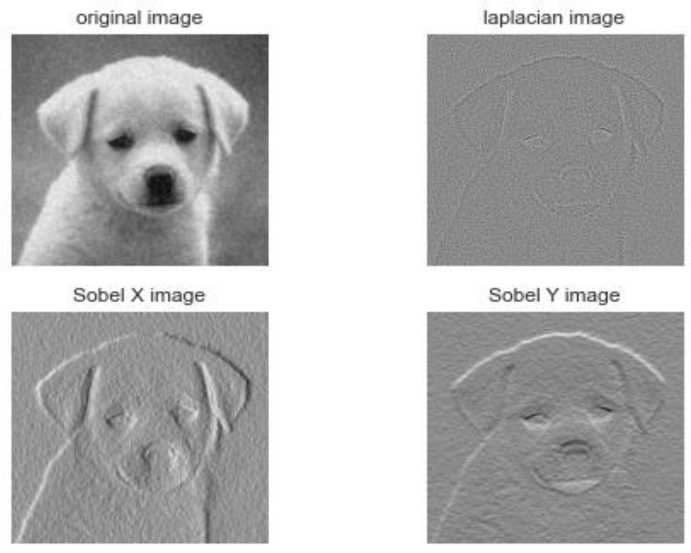
<u>CODE:</u>

```
#canny edge detection
import cv2
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('seaborn')

image1=cv2.imread("flower.jpg")
image1=cv2.cvtColor(image1,cv2.COLOR_BGR2RGB)

gray_image=cv2.cvtColor(image1,cv2.COLOR_BGR2GRAY)

edged_image=cv2.Canny(gray_image,threshold1=20,threshold2=100)

plt.figure(figsize=(20,20))
plt.subplot(1,3,1)
plt.imshow(image1,cmap='gray')
plt.title('original image')
plt.axis('off')

plt.subplot(1,3,2)
plt.imshow(gray_image,cmap='gray')
plt.title('grayscale image')
plt.axis('off')

plt.subplot(1,3,3)
plt.imshow(edged_image,cmap='gray')
plt.title('Canny edge detected image')
plt.axis('off')
plt.show()
```

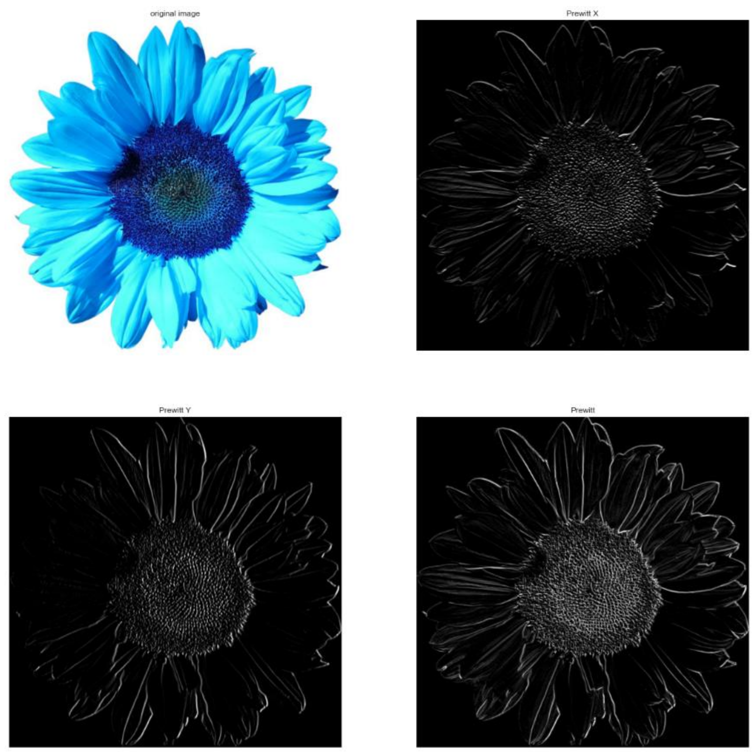


```
#laplacian and sobel edge detecting
import numpy as np
import matplotlib.pyplot as plt

image1=cv2.imread("noisy.png")
gray=cv2.cvtColor(image1,cv2.COLOR_BGR2GRAY)
```

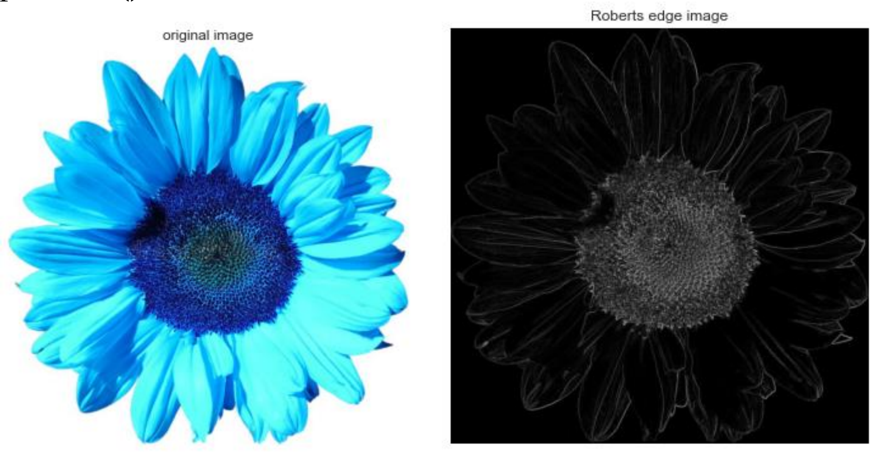
```
img=cv2.GaussianBlur(gray,(3,3),0)

laplacian=cv2.Laplacian(img,cv2.CV_64F)
sobelx=cv2.Sobel(img,cv2.CV_64F,1,0,ksize=5)
sobely=cv2.Sobel(img,cv2.CV_64F,0,1,ksize=5)

plt.subplot(2,2,1)
plt.imshow(img,cmap='gray')
plt.title('original image')
plt.xticks([])
plt.yticks([])

plt.subplot(2,2,2)
plt.imshow(laplacian,cmap='gray')
plt.title('laplacian image')
plt.xticks([])
plt.yticks([])

plt.subplot(2,2,3)
plt.imshow(sobelx,cmap='gray')
plt.title('Sobel X image')
plt.xticks([])
plt.yticks([])

plt.subplot(2,2,4)
plt.imshow(sobely,cmap='gray')
plt.title('Sobel Y image')
plt.xticks([])
plt.yticks([])

plt.show()
```



```
#edge detection using prewitt operator
import cv2
import numpy as np
import matplotlib.pyplot as plt

image1=cv2.imread("flower.jpg")
gray=cv2.cvtColor(image1,cv2.COLOR_BGR2GRAY)

img=cv2.GaussianBlur(gray,(3,3),0)

kernelx=np.array([[1,1,1],[0,0,0],[-1,-1,-1]])
kernely=np.array([[-1,0,1],[-1,0,1],[-1,0,1]])

img_prewittx=cv2.filter2D(img,-1,kernelx)
img_prewitty=cv2.filter2D(img,-1,kernely)

#cv2.imshow("Original image",image1)
#cv2.imshow("Prewitt X",img_prewittx)
#cv2.imshow("Prewitt Y",img_prewitty)
#cv2.imshow("Prewitt",img_prewittx+img_prewitty)
#cv2.waitKey(0)
#cv2.destroyAllWindows()

plt.figure(figsize=(20,20))
plt.subplot(2,2,1)
plt.imshow(image1,cmap='gray')
plt.title('original image')
plt.axis('off')

plt.subplot(2,2,2)
plt.imshow(img_prewittx,cmap='gray')
plt.title("Prewitt X")
plt.axis('off')
plt.subplot(2,2,3)
plt.imshow(img_prewitty,cmap='gray')
plt.title("Prewitt Y")
plt.axis('off')

plt.subplot(2,2,4)
plt.imshow(img_prewittx+img_prewitty,cmap='gray')
plt.title("Prewitt")
plt.axis('off')

plt.show()
```



```
#roberts edge detection
import cv2
import numpy as np
from scipy import ndimage
from matplotlib import pyplot as plt
roberts_cross_v=np.array([[1,0],[0,-1]])

roberts_cross_h=np.array([[0,1],[-1,0]])

img=cv2.imread("flower.jpg",0).astype('float64')
img/=255.0
vertical=ndimage.convolve(img,roberts_cross_v)
horizontal=ndimage.convolve(img,roberts_cross_h)

edged_img=np.sqrt(np.square(horizontal)+np.square(vertical))
edged_img*=255
cv2.imshow("Output image",edged_img)


cv2.waitKey(0)
cv2.destroyAllWindows()
plt.figure(figsize=(20,20))
plt.subplot(1,3,1)
plt.imshow(image1,cmap='gray')
plt.title('original image')
plt.axis('off')
plt.figure(figsize=(20,20))
plt.subplot(1,3,2)
plt.imshow(edged_img,cmap='gray')
plt.title('Roberts edge image')
plt.axis('off')
plt.show()
```



8.Develop a program to perform:
i)Global Thresholding
ii)Adaptive Thresholding
iii)Otsu thresholding using built in functions
CODE:

```
import cv2
import numpy as np

ret,th1 = cv2.threshold(blur,150,255,cv2.THRESH_BINARY)
cv2.imshow('Global', th1)
cv2.imwrite('Global.jpg',th1)

th2 =
cv2.adaptiveThreshold(blur,255,cv2.ADAPTIVE_THRESH_MEAN_C,cv2.THRESH
_BINARY,11,2)
cv2.imshow('Adaptive Mean', th2)
cv2.imwrite('AM.jpg',th2)


th3 =
cv2.adaptiveThreshold(blur,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.TH
RESH_BINARY,11,2)
cv2.imshow('Adaptive Gaussian', th3)
cv2.imwrite('AG.jpg',th3)
cv2.waitKey(0)
```

```
cv2.destroyAllWindows()
```

<u>#otsu thresholding</u>

```
import cv2
import numpy as np

img = cv2.imread('noisy.png', cv2.IMREAD_GRAYSCALE)
cv2.imshow('gray', img)
#cv2.imwrite('gray1.jpg',img)

blur = cv2.GaussianBlur(img,(7,7),0)
cv2.imshow('blur', img)
#cv2.imwrite('blur1.jpg',img)

x,threshold = cv2.threshold(blur, 200, 255, cv2.THRESH_BINARY)
cv2.imshow('Binary threshold', threshold)
#cv2.imwrite('binarythresh1.jpg',img)

ret2,th2 = cv2.threshold(blur,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
cv2.imshow('Otsus Thresholding', th2)
#cv2.imwrite('Otsus.jpg',img)

cv2.waitKey(0)
cv2.destroyAllWindows()
```