Pabna University of Science and Technology

Department of
Computer Science and Engineering

Faculty of
Engineering and Technology

Lab Report On

Course Code: CSE 4106

Course Title: Digital Image Processing Sessional.

Submitted by:

Name: Md. Habibul Islam

Roll: 200124

Session 2019-20.

4th year 1st semester

Dept. Of CSE,

Pabna University of Science and Technology.

Submitted to:

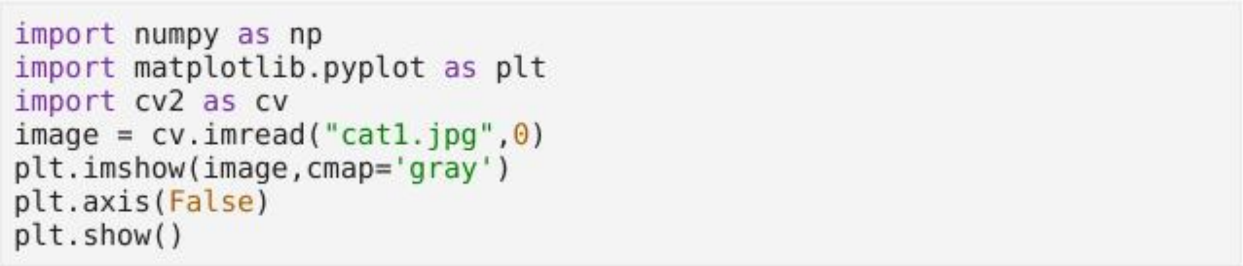
Md. Shafiul Azam

Associate Professor,

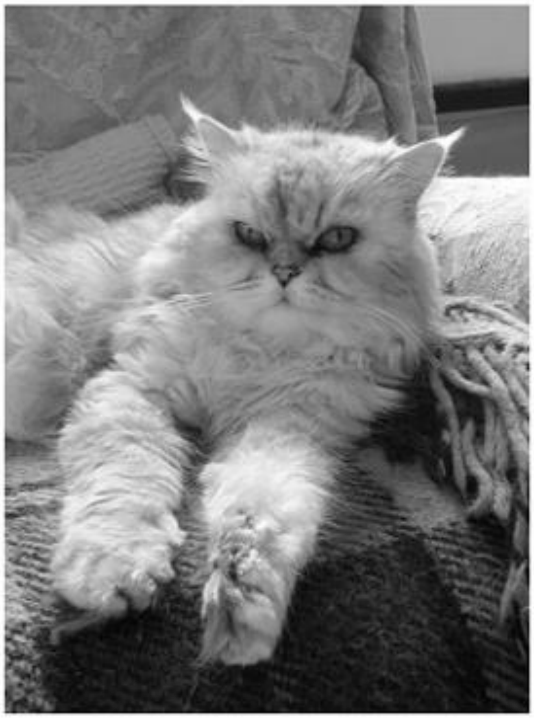
Department of CSE,

Pabna University of Science and Technology, Pabna, Bangladesh.

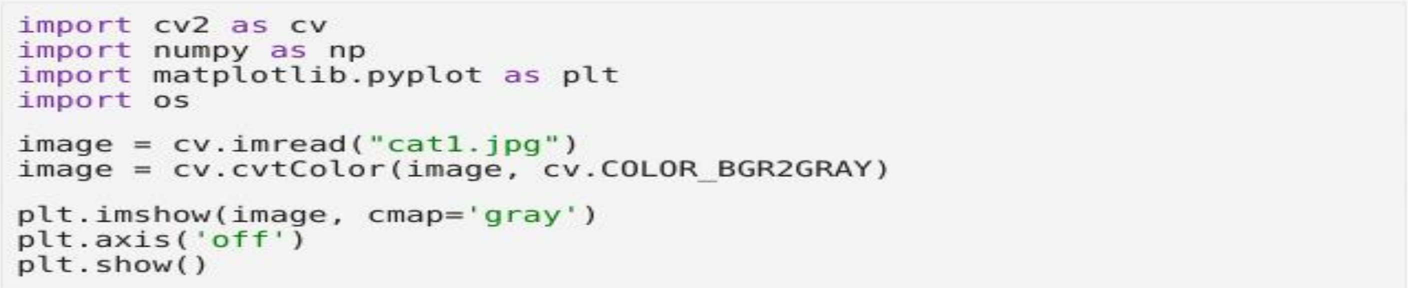
Date of Submission: January 11, 2025.



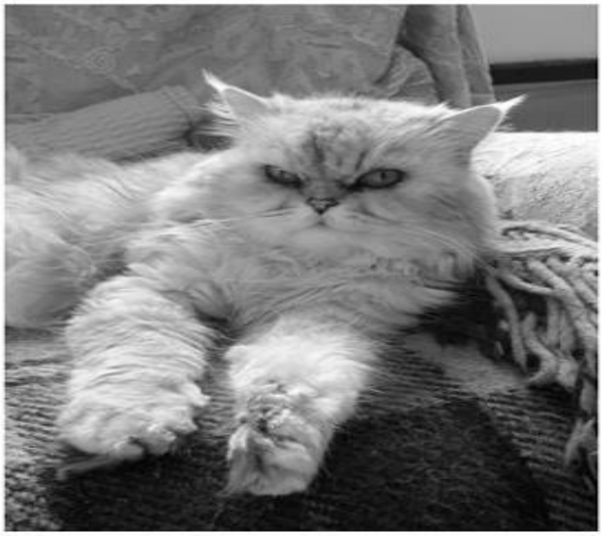
```
import numpy as np
import matplotlib.pyplot as plt
import cv2 as cv
image = cv.imread("cat1.jpg",0)
plt.imshow(image,cmap='gray')
plt.axis(False)
plt.show()
```

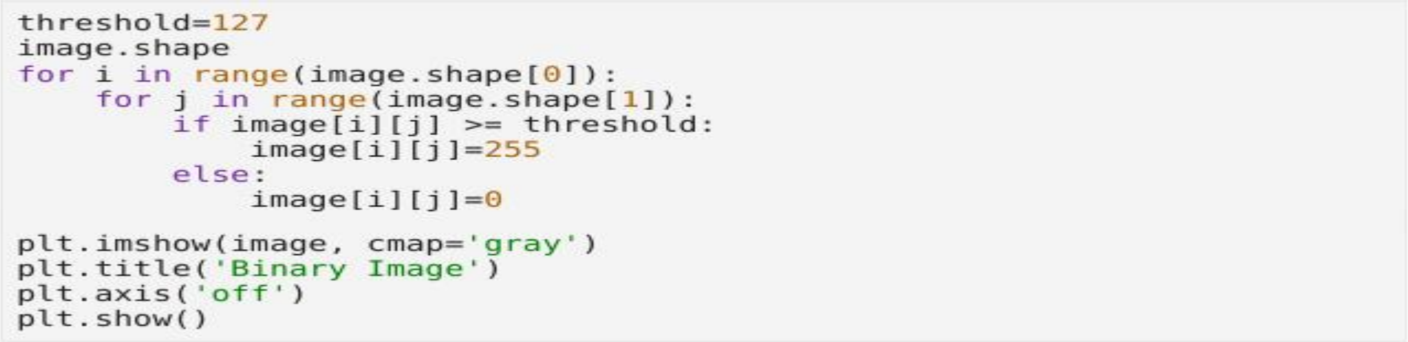


Binary Image

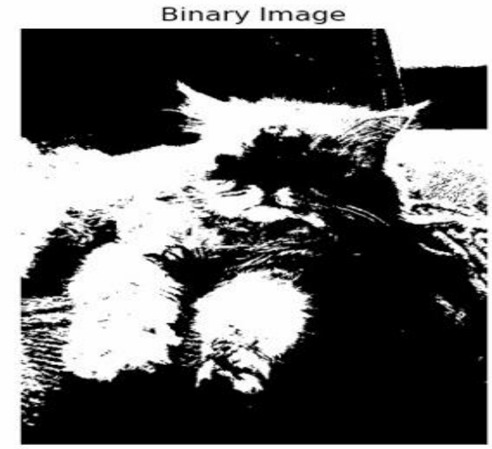


```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
import os

image = cv.imread("cat1.jpg")
image = cv.cvtColor(image, cv.COLOR_BGR2GRAY)

plt.imshow(image, cmap='gray')
plt.axis('off')
plt.show()
```

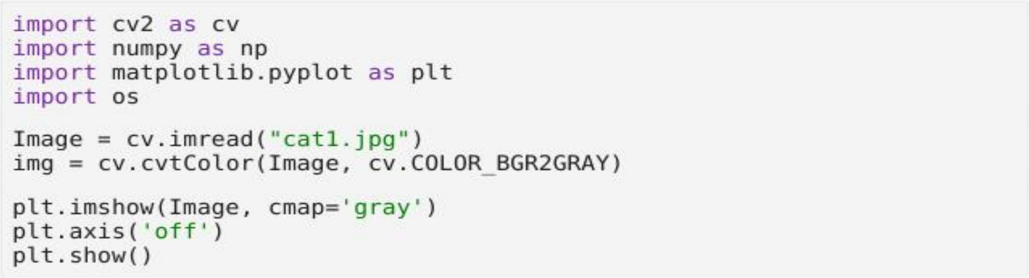




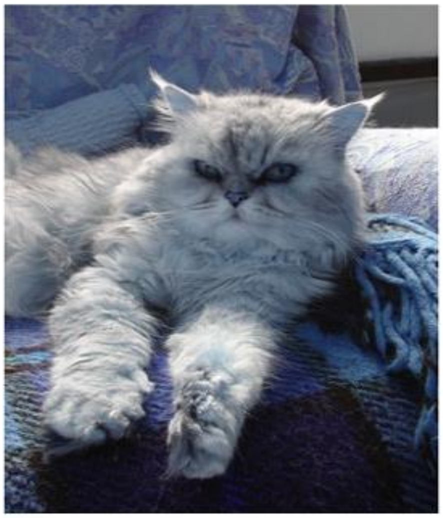
```
threshold=127
image.shape
for i in range(image.shape[0]):
    for j in range(image.shape[1]):
        if image[i][j] >= threshold:
            image[i][j]=255
        else:
            image[i][j]=0

plt.imshow(image, cmap='gray')
plt.title('Binary Image')
plt.axis('off')
plt.show()
```

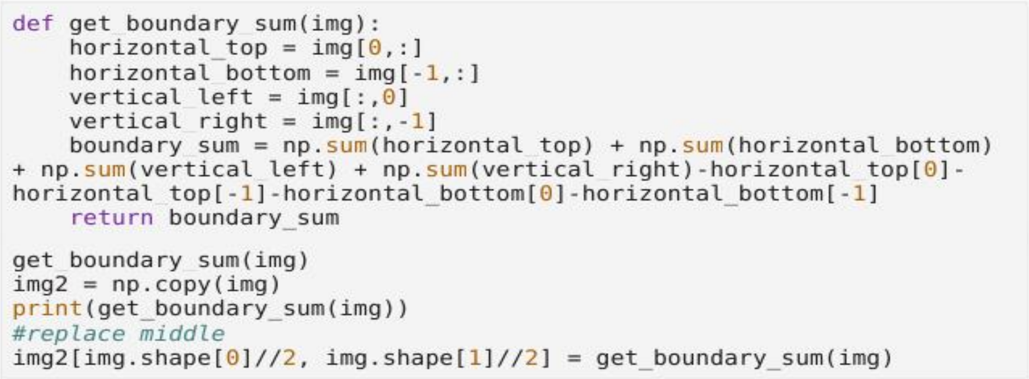


Boundary sum

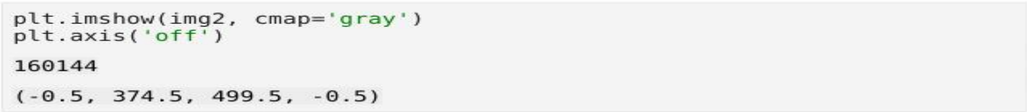


```python
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
import os

Image = cv.imread("cat1.jpg")
img = cv.cvtColor(Image, cv.COLOR_BGR2GRAY)

plt.imshow(Image, cmap='gray')
plt.axis('off')
plt.show()
```



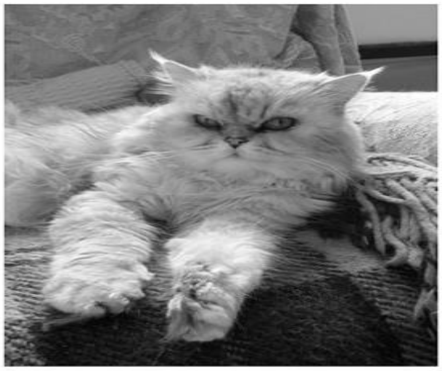


```python
def get_boundary_sum(img):
    horizontal_top = img[0,:]
    horizontal_bottom = img[-1,:]
    vertical_left = img[:,0]
    vertical_right = img[:,-1]
    boundary_sum = np.sum(horizontal_top) + np.sum(horizontal_bottom)
+ np.sum(vertical_left) + np.sum(vertical_right)-horizontal_top[0]-
horizontal_top[-1]-horizontal_bottom[0]-horizontal_bottom[-1]
    return boundary_sum

get_boundary_sum(img)
img2 = np.copy(img)
print(get_boundary_sum(img))
#replace middle
img2[img.shape[0]//2, img.shape[1]//2] = get_boundary_sum(img)
```

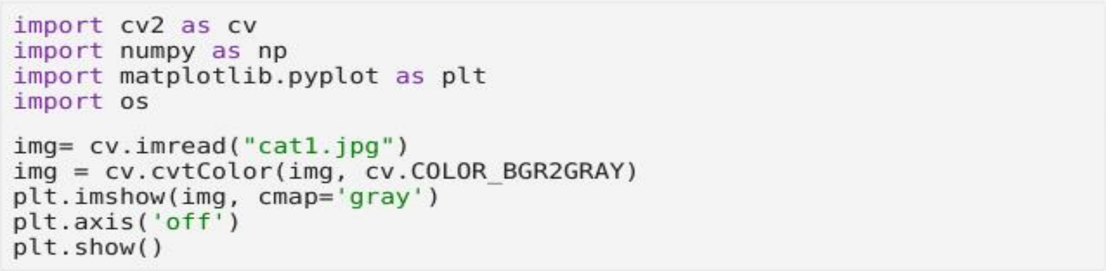


```python
plt.imshow(img2, cmap='gray')
plt.axis('off')
```

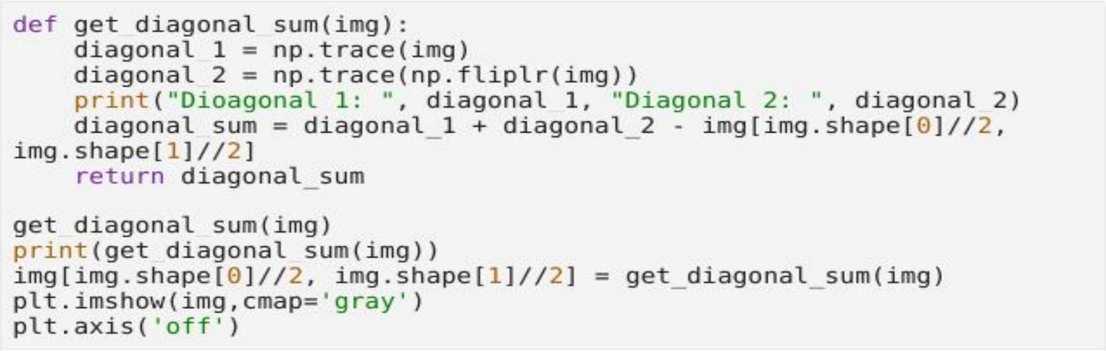
```
160144
(-0.5, 374.5, 499.5, -0.5)
```



Diagonal Sum

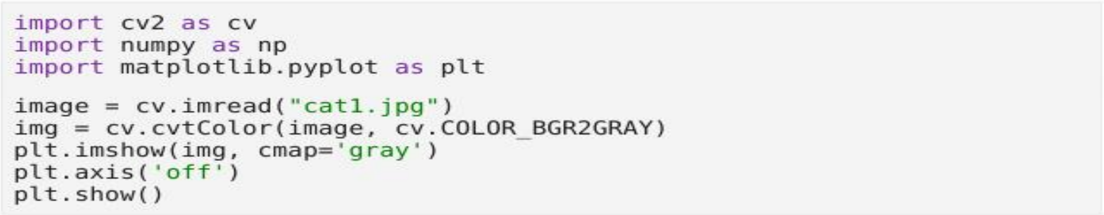


```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
import os

img= cv.imread("cat1.jpg")
img = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
plt.imshow(img, cmap='gray')
plt.axis('off')
plt.show()
```

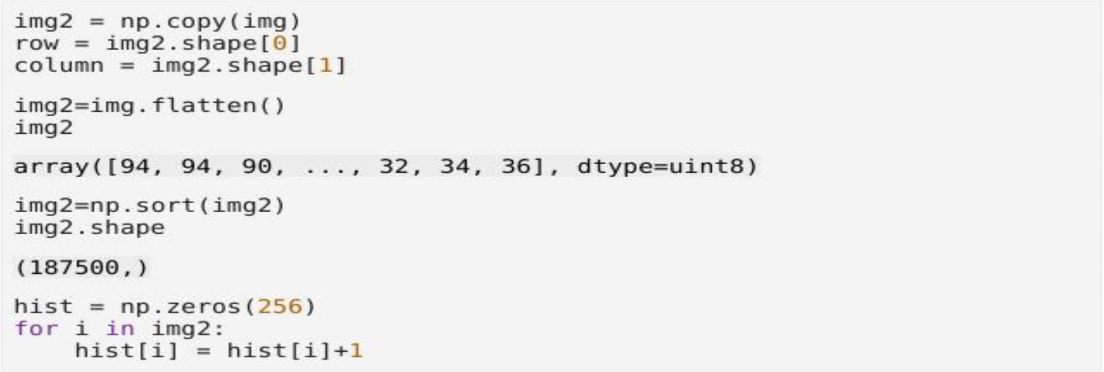


```
def get_diagonal_sum(img):
    diagonal_1 = np.trace(img)
    diagonal_2 = np.trace(np.fliplr(img))
    print("Dioagonal 1: ", diagonal_1, "Diagonal 2: ", diagonal_2)
    diagonal_sum = diagonal_1 + diagonal_2 - img[img.shape[0]//2,
img.shape[1]//2]
    return diagonal_sum

get_diagonal_sum(img)
print(get_diagonal_sum(img))
img[img.shape[0]//2, img.shape[1]//2] = get_diagonal_sum(img)
plt.imshow(img,cmap='gray')
plt.axis('off')
```

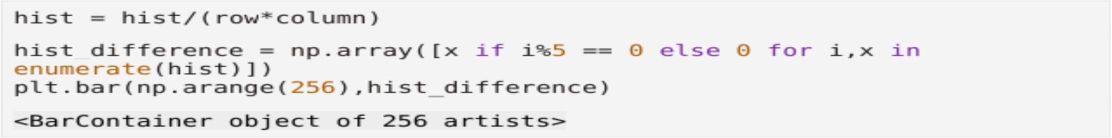
Histogram



```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt

image = cv.imread("cat1.jpg")
img = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
plt.imshow(img, cmap='gray')
plt.axis('off')
plt.show()
```



```
img2 = np.copy(img)
row = img2.shape[0]
column = img2.shape[1]

img2=img.flatten()
img2

array([94, 94, 90, ..., 32, 34, 36], dtype=uint8)

img2=np.sort(img2)
img2.shape

(187500,)

hist = np.zeros(256)
for i in img2:
    hist[i] = hist[i]+1
```



```
hist = hist/(row*column)

hist_difference = np.array([x if i%5 == 0 else 0 for i,x in
enumerate(hist)])
plt.bar(np.arange(256),hist_difference)

<BarContainer object of 256 artists>
```

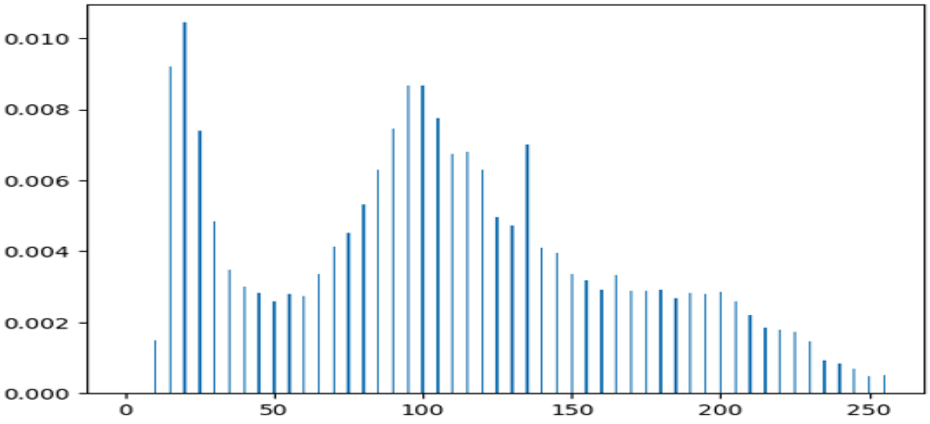
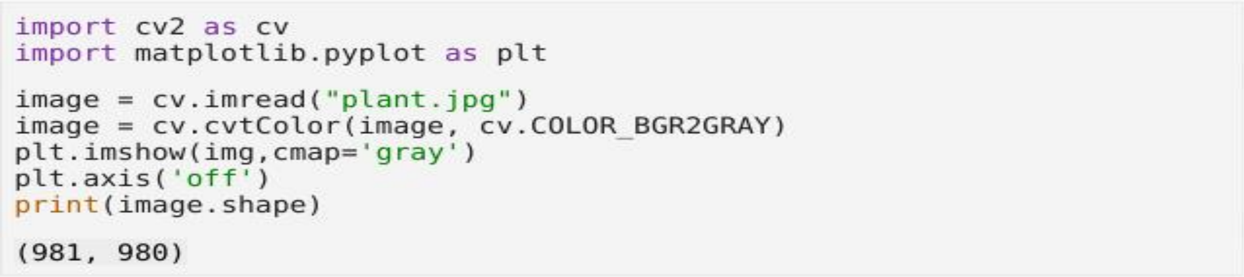
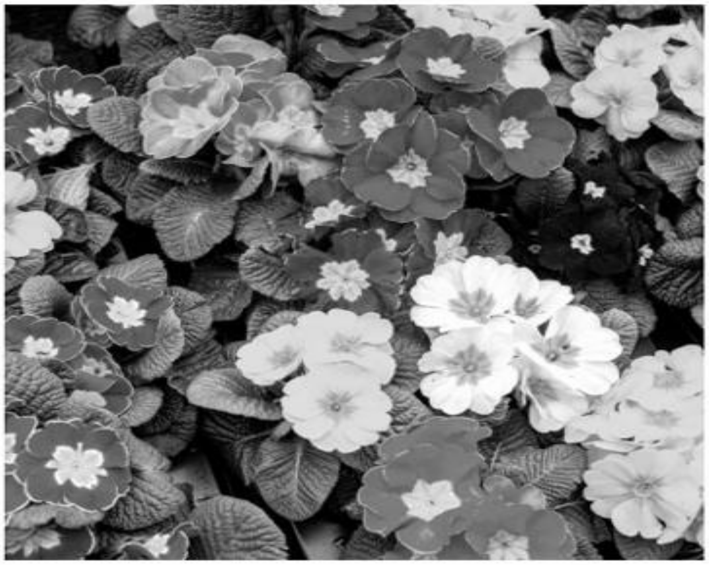
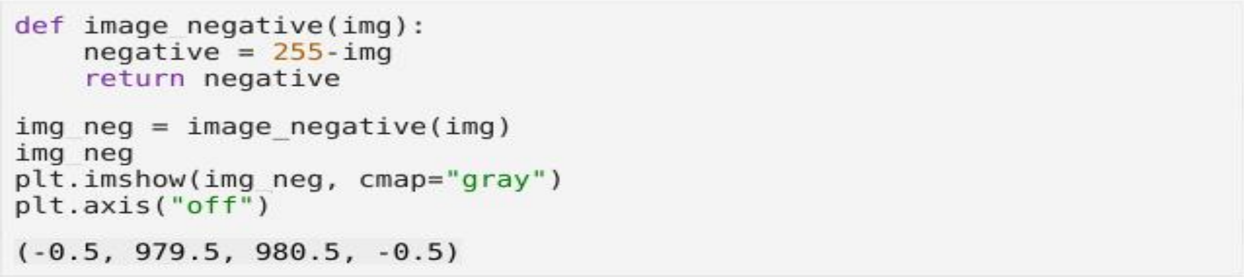


Image Negative

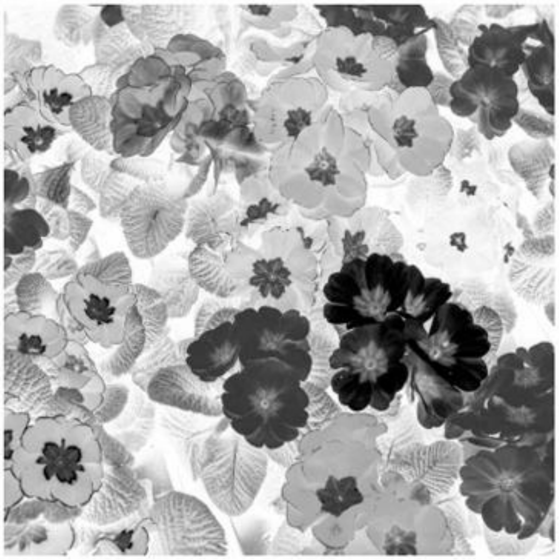


```
import cv2 as cv
import matplotlib.pyplot as plt

image = cv.imread("plant.jpg")
image = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
plt.imshow(img,cmap='gray')
plt.axis('off')
print(image.shape)

(981, 980)
```

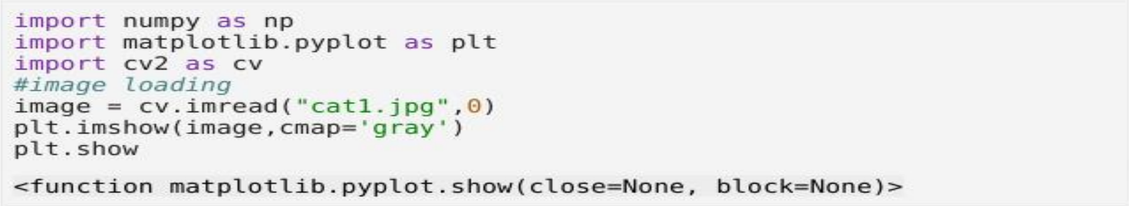




```
def image_negative(img):
    negative = 255-img
    return negative

img_neg = image_negative(img)
img_neg
plt.imshow(img_neg, cmap="gray")
plt.axis("off")

(-0.5, 979.5, 980.5, -0.5)
```

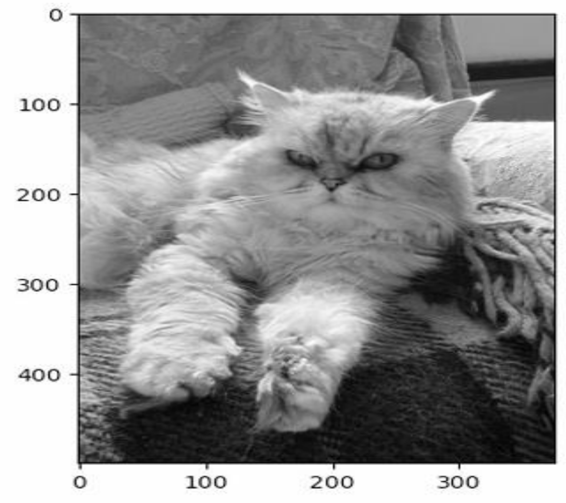


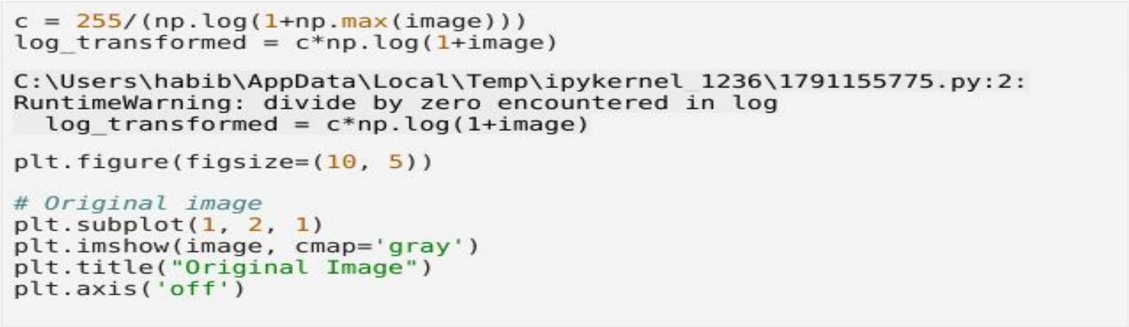
Log Transform



```
import numpy as np
import matplotlib.pyplot as plt
import cv2 as cv
#image loading
image = cv.imread("cat1.jpg",0)
plt.imshow(image,cmap='gray')
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```

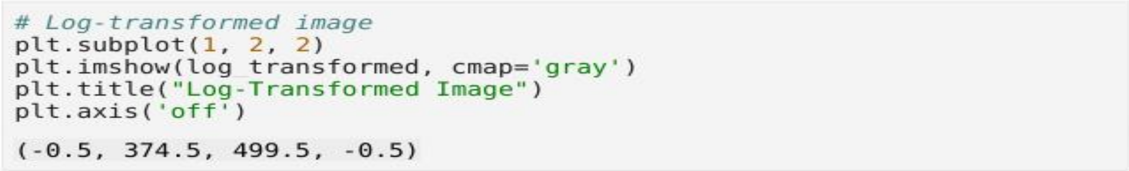




```
c = 255/(np.log(1+np.max(image)))
log_transformed = c*np.log(1+image)
```

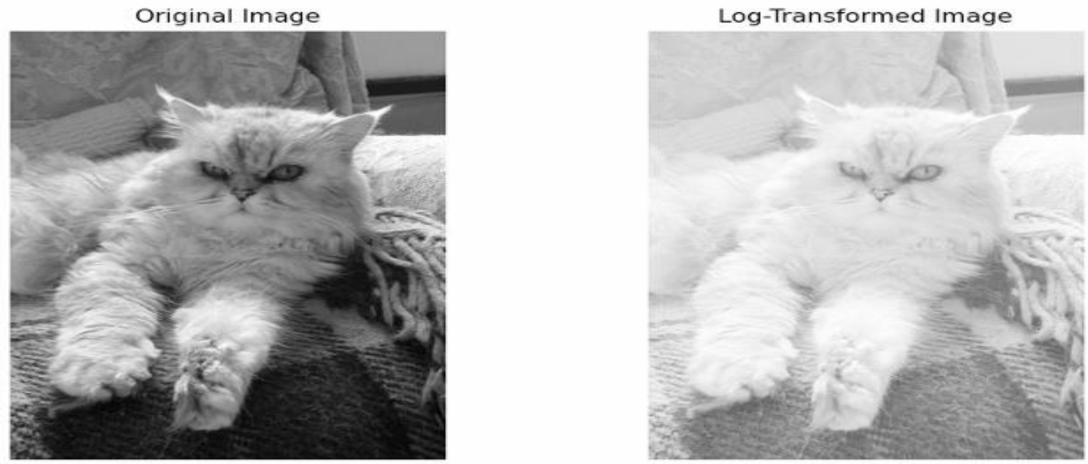
```
C:\Users\habib\AppData\Local\Temp\ipykernel_1236\1791155775.py:2:
RuntimeWarning: divide by zero encountered in log
  log_transformed = c*np.log(1+image)
```

```
plt.figure(figsize=(10, 5))

# Original image
plt.subplot(1, 2, 1)
plt.imshow(image, cmap='gray')
plt.title("Original Image")
plt.axis('off')
```

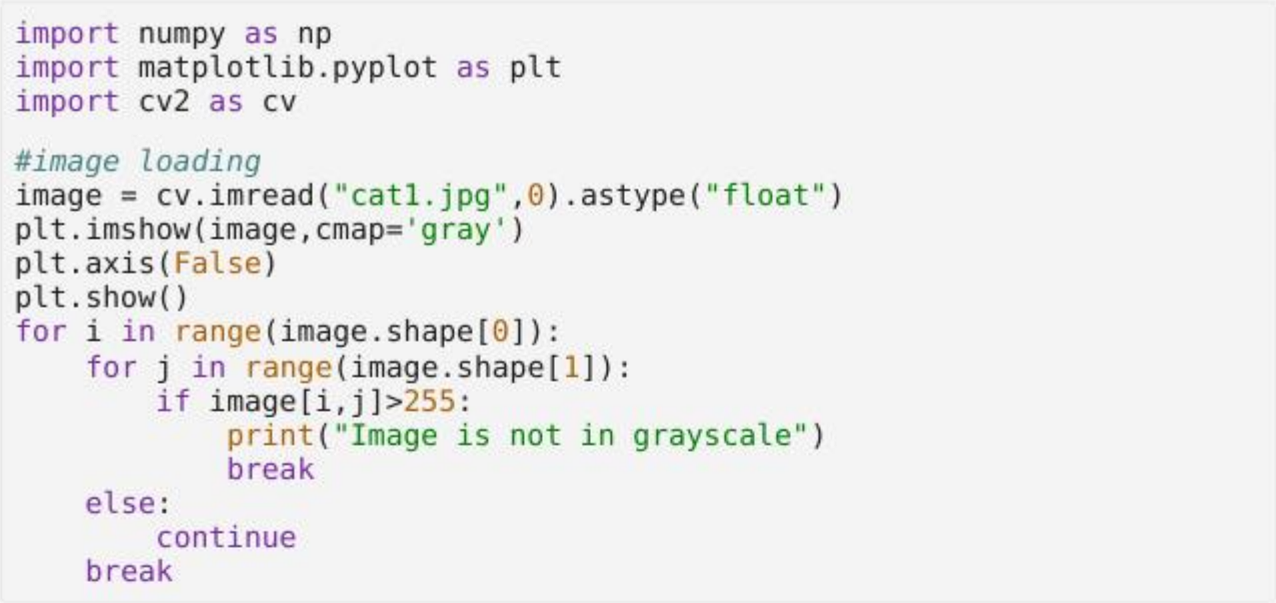


```
# Log-transformed image
plt.subplot(1, 2, 2)
plt.imshow(log_transformed, cmap='gray')
plt.title("Log-Transformed Image")
plt.axis('off')
```

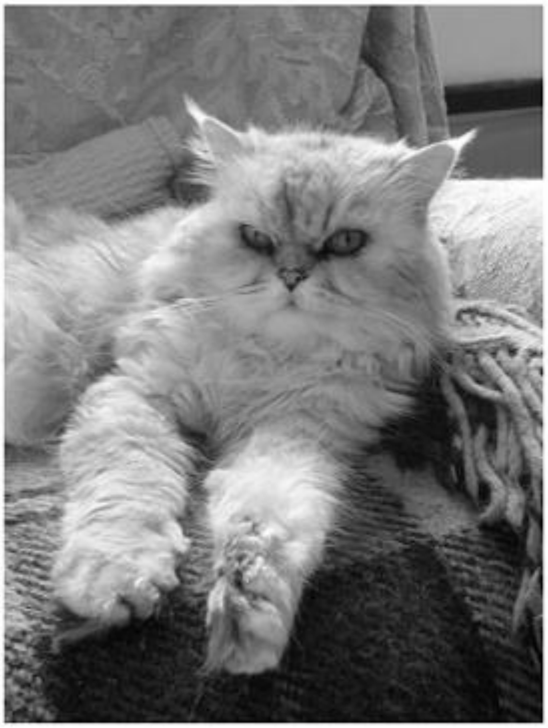
```
(-0.5, 374.5, 499.5, -0.5)
```

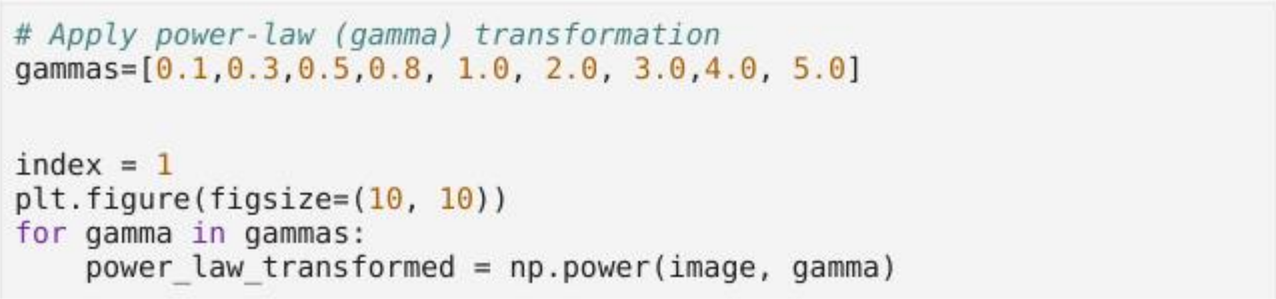


Power-law Transform

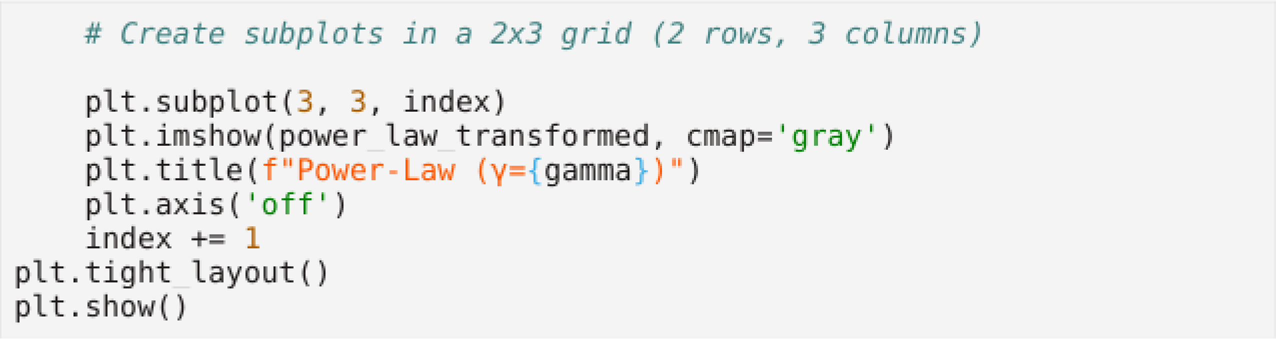


```python
import numpy as np
import matplotlib.pyplot as plt
import cv2 as cv

#image loading
image = cv.imread("cat1.jpg",0).astype("float")
plt.imshow(image,cmap='gray')
plt.axis(False)
plt.show()
for i in range(image.shape[0]):
    for j in range(image.shape[1]):
        if image[i,j]>255:
            print("Image is not in grayscale")
            break
    else:
        continue
    break
```

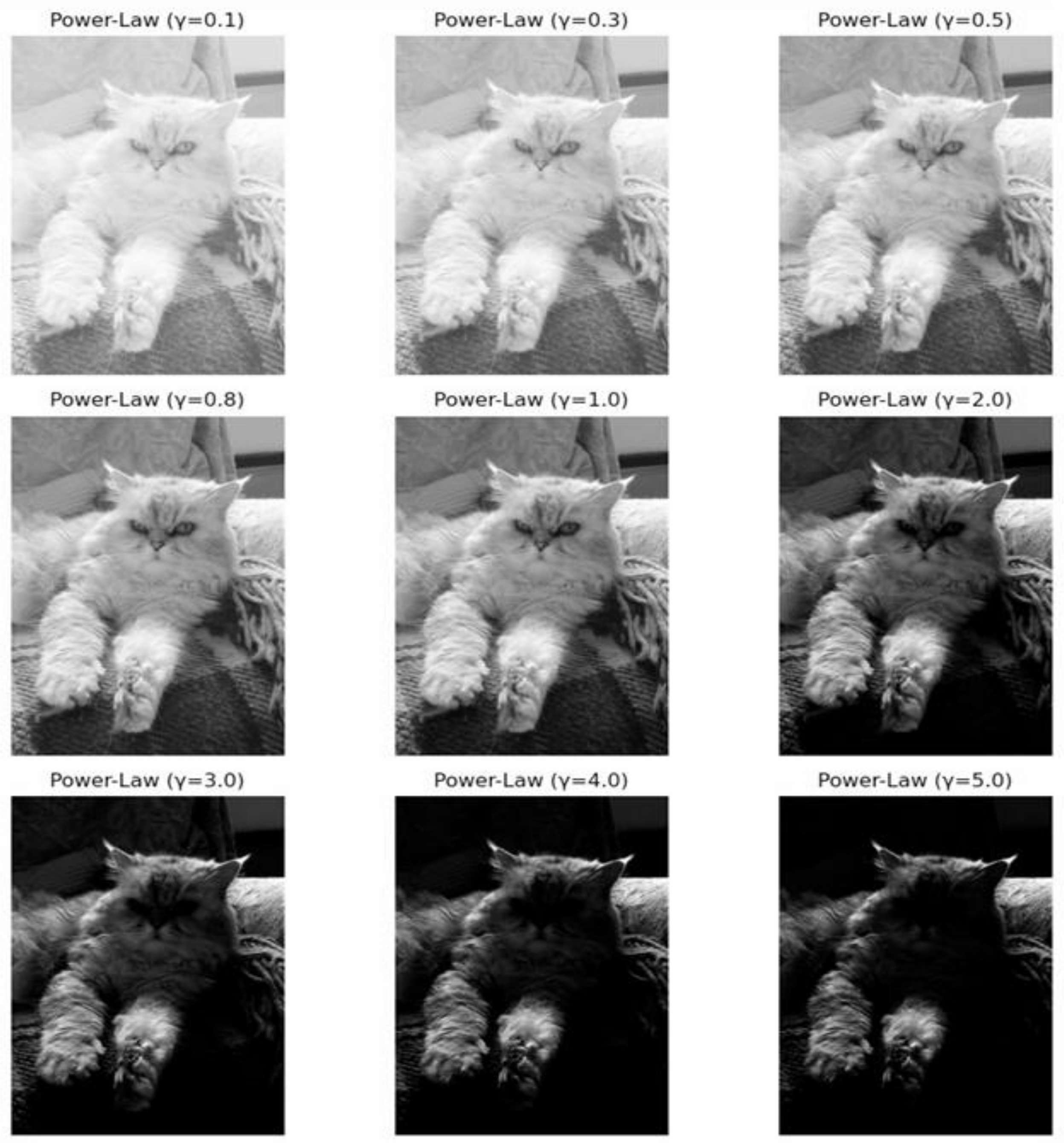




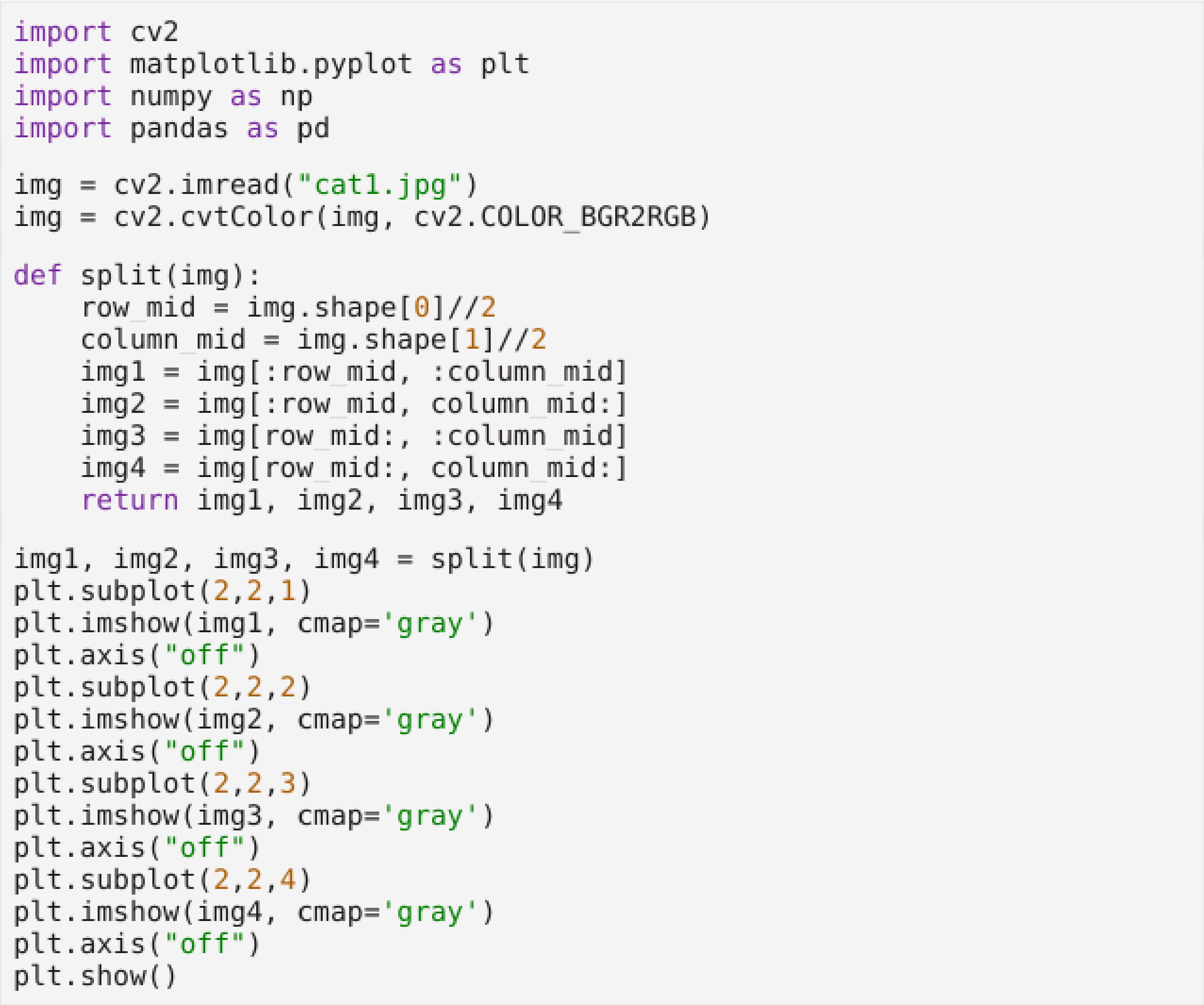
```python
# Apply power-law (gamma) transformation
gammas=[0.1,0.3,0.5,0.8, 1.0, 2.0, 3.0,4.0, 5.0]


index = 1
plt.figure(figsize=(10, 10))
for gamma in gammas:
    power_law_transformed = np.power(image, gamma)
```



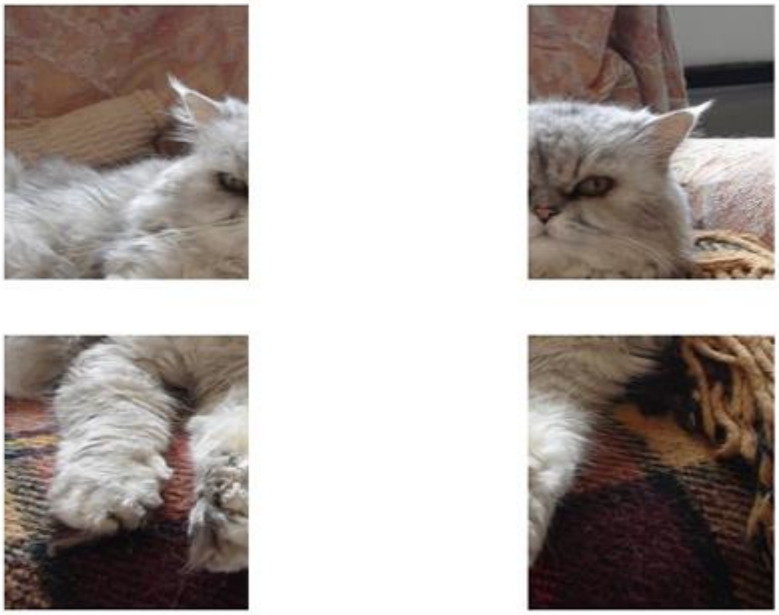
```
    # Create subplots in a 2x3 grid (2 rows, 3 columns)

    plt.subplot(3, 3, index)
    plt.imshow(power_law_transformed, cmap='gray')
    plt.title(f"Power-Law (γ={gamma})")
    plt.axis('off')
    index += 1
plt.tight_layout()
plt.show()
```



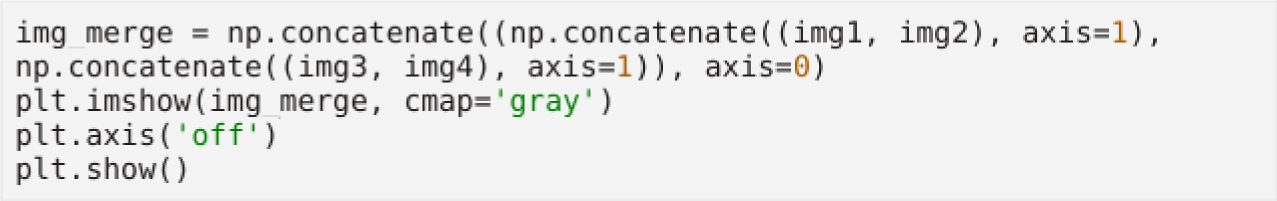
Split Image



```
import cv2
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

img = cv2.imread("cat1.jpg")
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

def split(img):
    row_mid = img.shape[0]//2
    column_mid = img.shape[1]//2
    img1 = img[:row_mid, :column_mid]
    img2 = img[:row_mid, column_mid:]
    img3 = img[row_mid:, :column_mid]
    img4 = img[row_mid:, column_mid:]
    return img1, img2, img3, img4

img1, img2, img3, img4 = split(img)
plt.subplot(2,2,1)
plt.imshow(img1, cmap='gray')
plt.axis("off")
plt.subplot(2,2,2)
plt.imshow(img2, cmap='gray')
plt.axis("off")
plt.subplot(2,2,3)
plt.imshow(img3, cmap='gray')
plt.axis("off")
plt.subplot(2,2,4)
plt.imshow(img4, cmap='gray')
plt.axis("off")
plt.show()
```

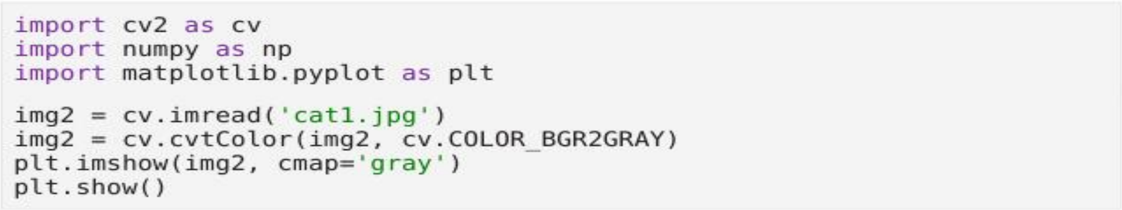


Merge Image

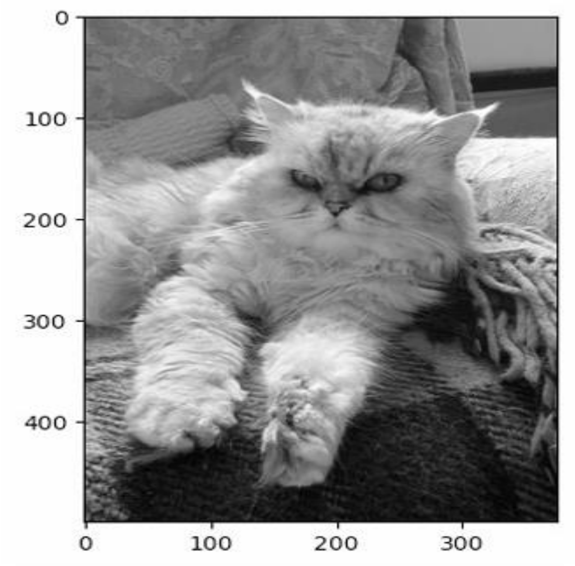


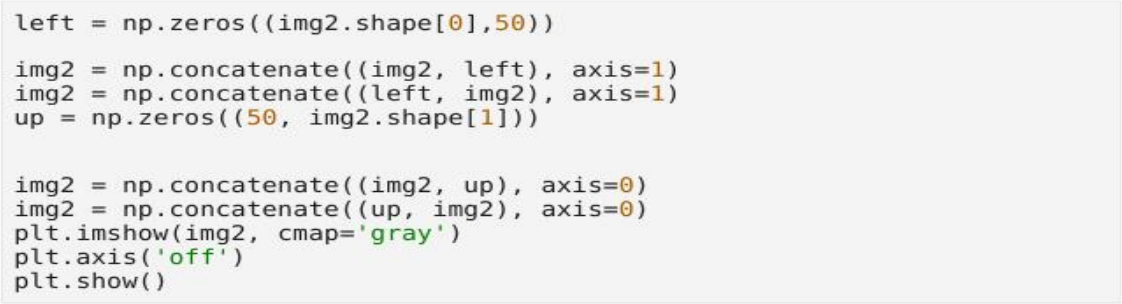
```
img_merge = np.concatenate((np.concatenate((img1, img2), axis=1),
np.concatenate((img3, img4), axis=1)), axis=0)
plt.imshow(img_merge, cmap='gray')
plt.axis('off')
plt.show()
```

Padding on Image

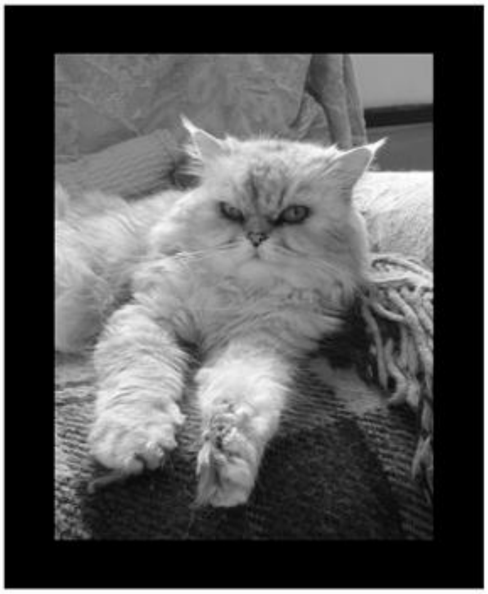


```python
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt

img2 = cv.imread('cat1.jpg')
img2 = cv.cvtColor(img2, cv.COLOR_BGR2GRAY)
plt.imshow(img2, cmap='gray')
plt.show()
```

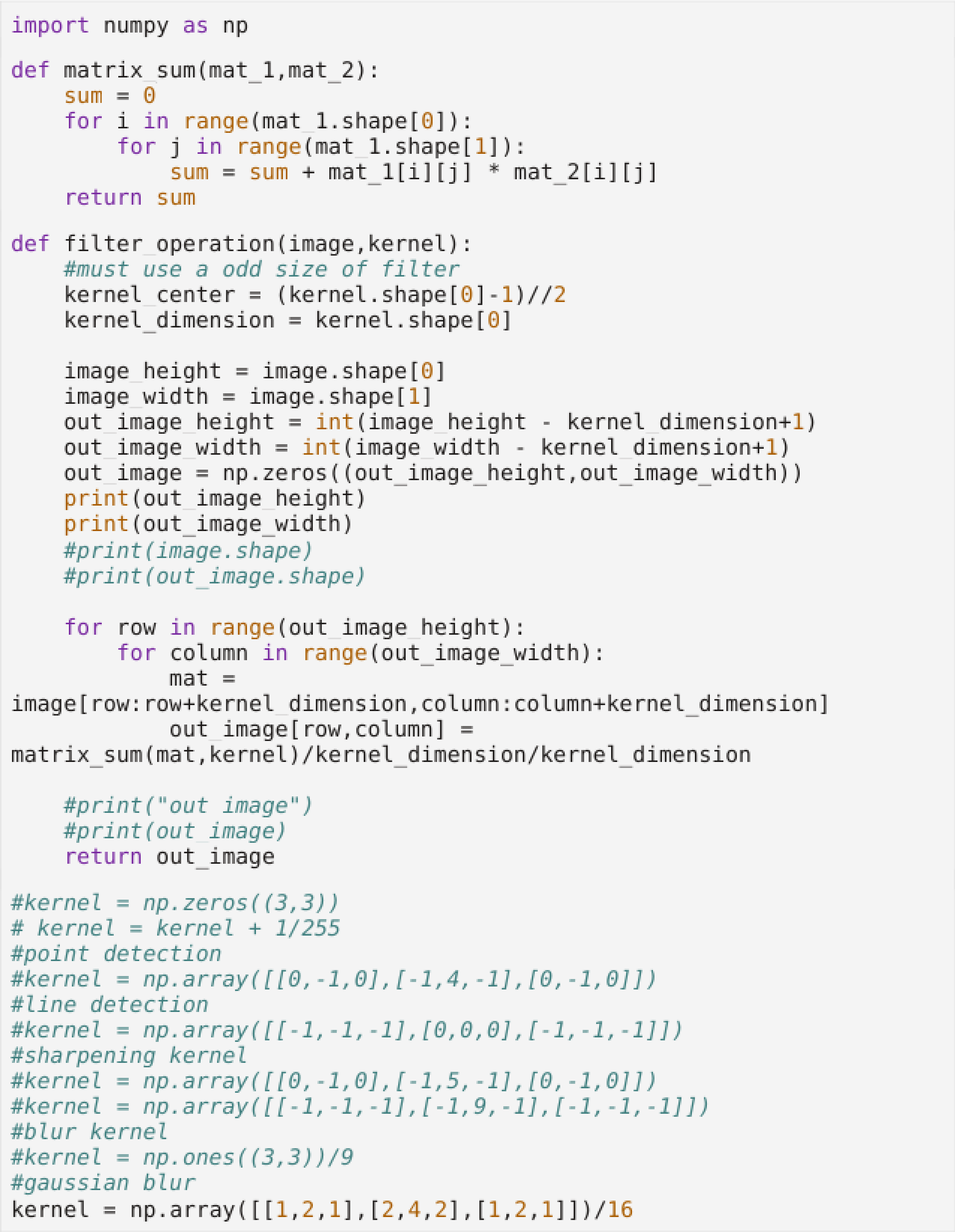




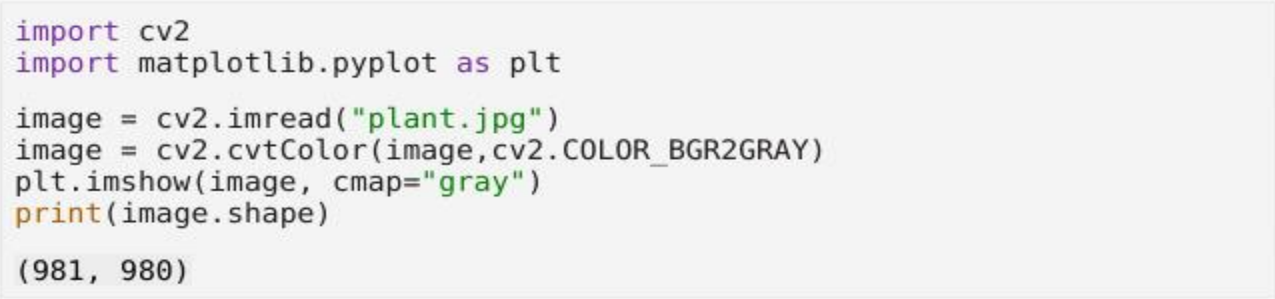
```python
left = np.zeros((img2.shape[0],50))

img2 = np.concatenate((img2, left), axis=1)
img2 = np.concatenate((left, img2), axis=1)
up = np.zeros((50, img2.shape[1]))


img2 = np.concatenate((img2, up), axis=0)
img2 = np.concatenate((up, img2), axis=0)
plt.imshow(img2, cmap='gray')
plt.axis('off')
plt.show()
```



Blurring Image

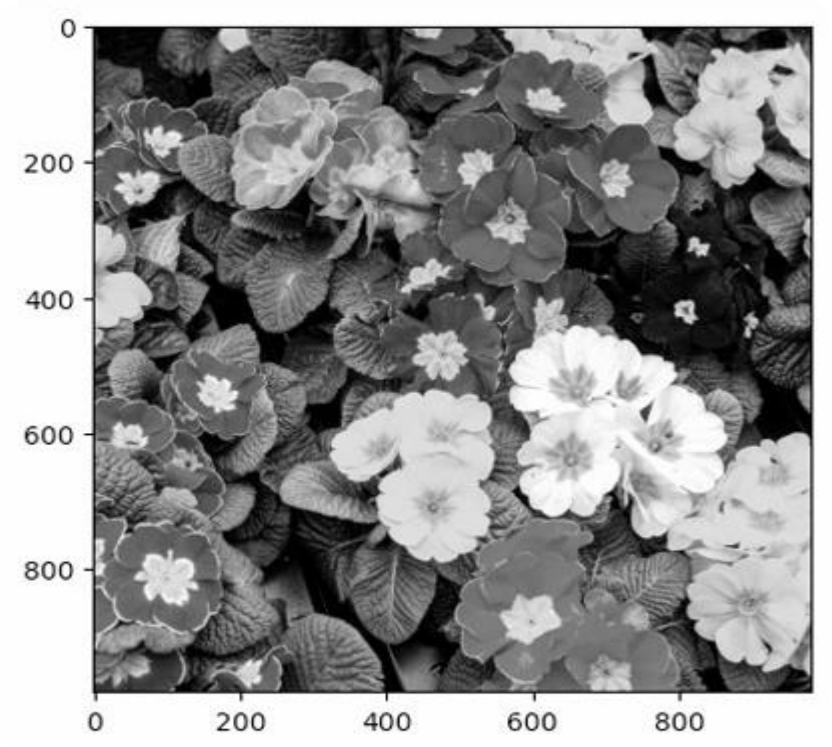


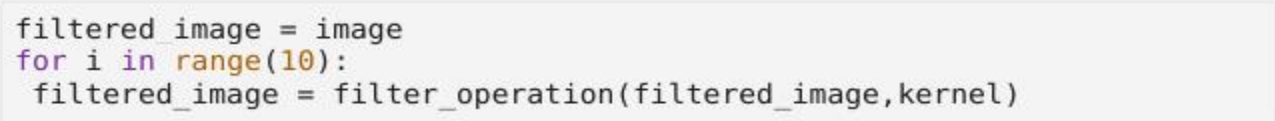
```
import numpy as np

def matrix_sum(mat_1,mat_2):
    sum = 0
    for i in range(mat_1.shape[0]):
        for j in range(mat_1.shape[1]):
            sum = sum + mat_1[i][j] * mat_2[i][j]
    return sum

def filter_operation(image,kernel):
    #must use a odd size of filter
    kernel_center = (kernel.shape[0]-1)//2
    kernel_dimension = kernel.shape[0]

    image_height = image.shape[0]
    image_width = image.shape[1]
    out_image_height = int(image_height - kernel_dimension+1)
    out_image_width = int(image_width - kernel_dimension+1)
    out_image = np.zeros((out_image_height,out_image_width))
    print(out_image_height)
    print(out_image_width)
    #print(image.shape)
    #print(out_image.shape)

    for row in range(out_image_height):
        for column in range(out_image_width):
            mat =
image[row:row+kernel_dimension,column:column+kernel_dimension]
            out_image[row,column] =
matrix_sum(mat,kernel)/kernel_dimension/kernel_dimension

    #print("out image")
    #print(out_image)
    return out_image

#kernel = np.zeros((3,3))
# kernel = kernel + 1/255
#point detection
#kernel = np.array([[0,-1,0],[-1,4,-1],[0,-1,0]])
#line detection
#kernel = np.array([[-1,-1,-1],[0,0,0],[-1,-1,-1]])
#sharpening kernel
#kernel = np.array([[0,-1,0],[-1,5,-1],[0,-1,0]])
#kernel = np.array([[-1,-1,-1],[-1,9,-1],[-1,-1,-1]])
#blur kernel
#kernel = np.ones((3,3))/9
#gaussian blur
kernel = np.array([[1,2,1],[2,4,2],[1,2,1]])/16
```



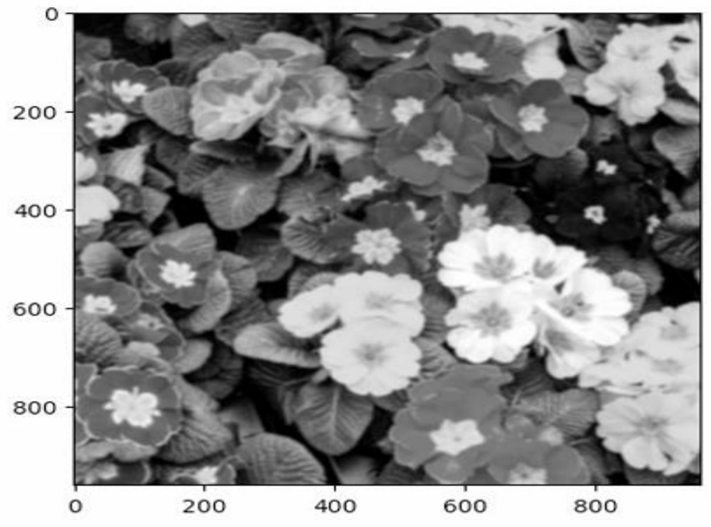
```
import cv2
import matplotlib.pyplot as plt

image = cv2.imread("plant.jpg")
image = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
plt.imshow(image, cmap="gray")
print(image.shape)
```

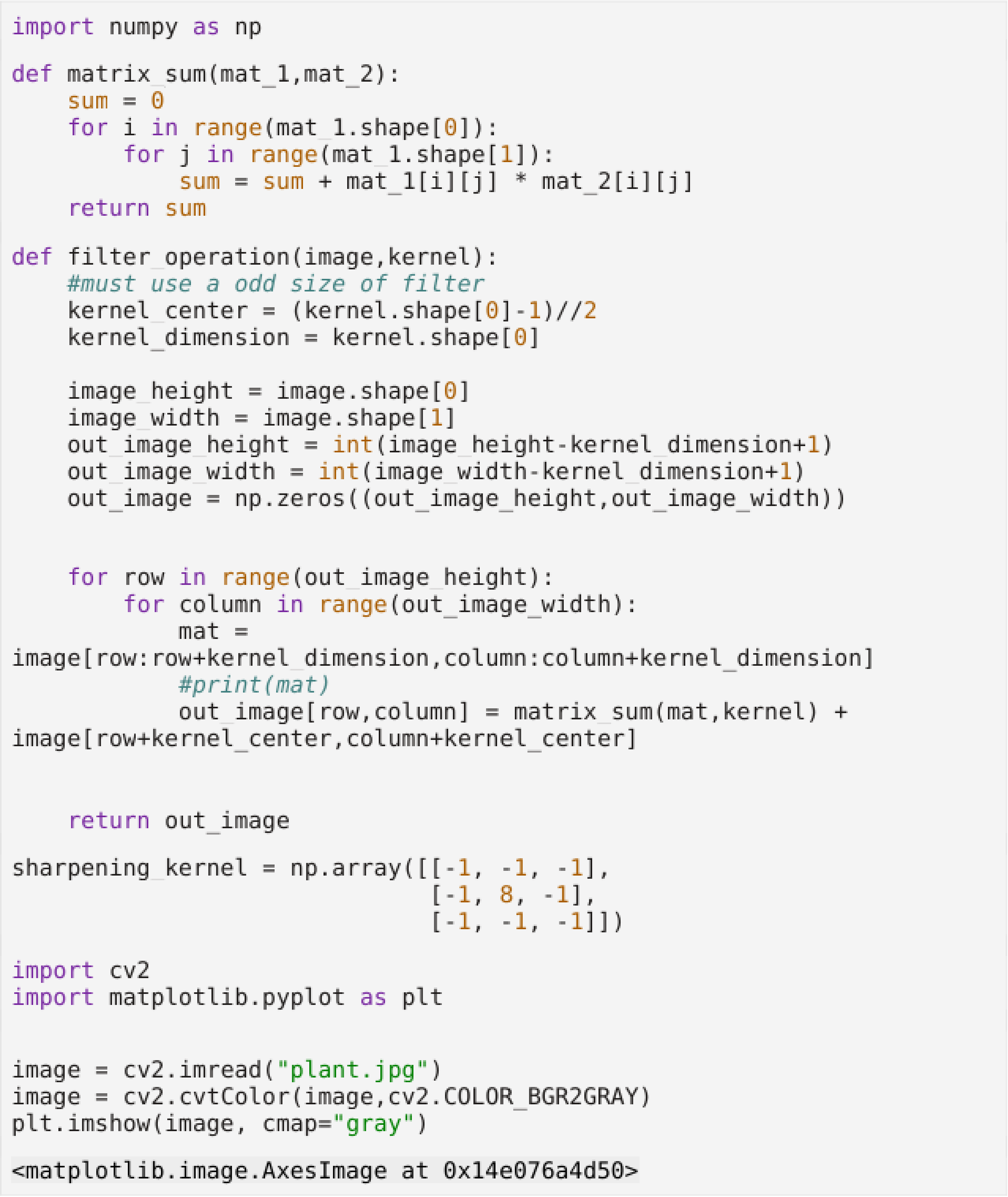
```
(981, 980)
```



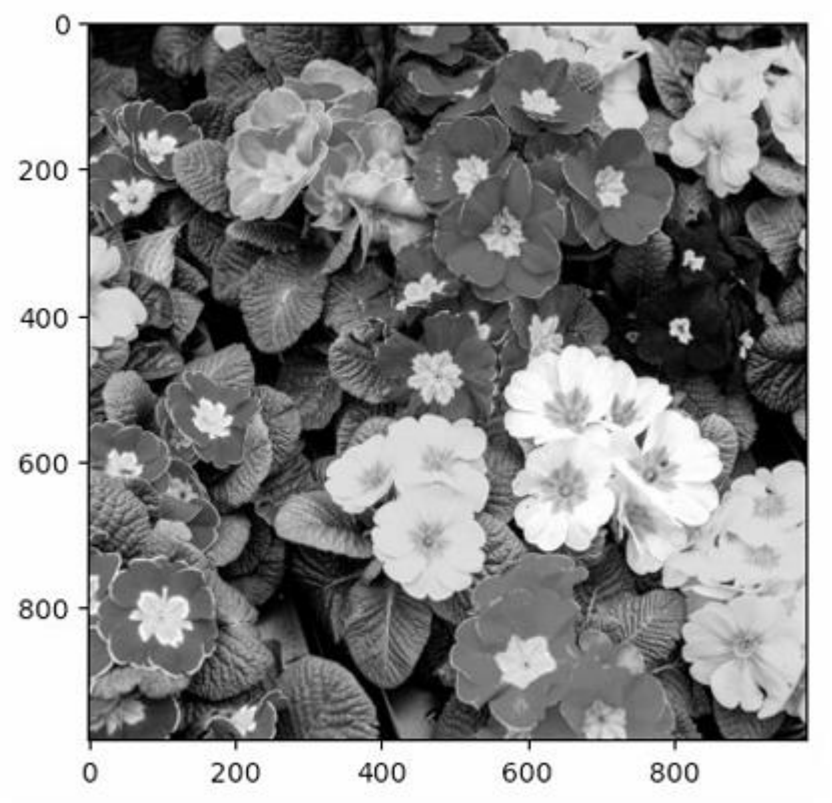


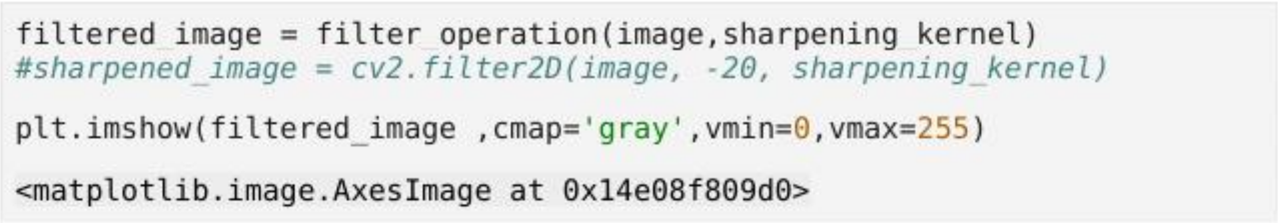
```
filtered_image = image
for i in range(10):
 filtered_image = filter_operation(filtered_image,kernel)
```



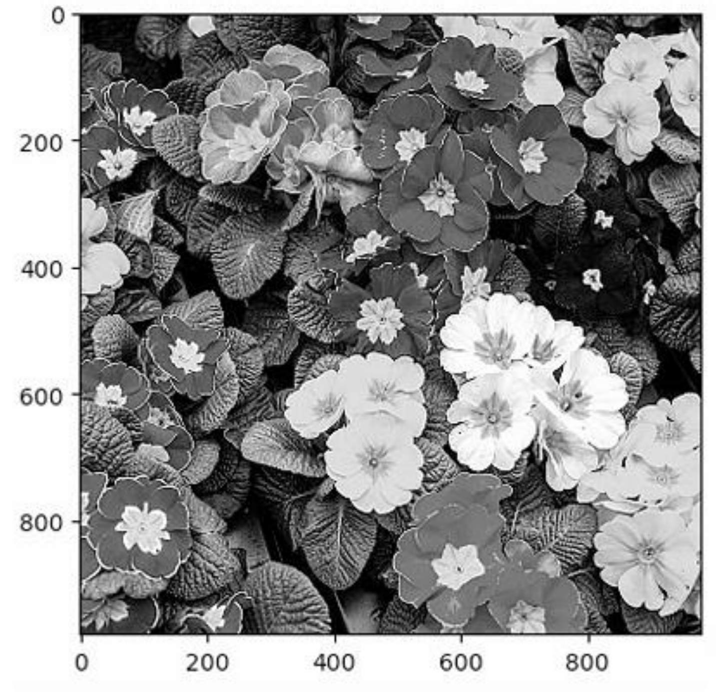


```
import numpy as np

def matrix_sum(mat_1,mat_2):
    sum = 0
    for i in range(mat_1.shape[0]):
        for j in range(mat_1.shape[1]):
            sum = sum + mat_1[i][j] * mat_2[i][j]
    return sum

def filter_operation(image,kernel):
    #must use a odd size of filter
    kernel_center = (kernel.shape[0]-1)//2
    kernel_dimension = kernel.shape[0]

    image_height = image.shape[0]
    image_width = image.shape[1]
    out_image_height = int(image_height-kernel_dimension+1)
    out_image_width = int(image_width-kernel_dimension+1)
    out_image = np.zeros((out_image_height,out_image_width))


    for row in range(out_image_height):
        for column in range(out_image_width):
            mat =
image[row:row+kernel_dimension,column:column+kernel_dimension]
            #print(mat)
            out_image[row,column] = matrix_sum(mat,kernel) +
image[row+kernel_center,column+kernel_center]


    return out_image

sharpening_kernel = np.array([[-1, -1, -1],
                              [-1, 8, -1],
                              [-1, -1, -1]])

import cv2
import matplotlib.pyplot as plt


image = cv2.imread("plant.jpg")
image = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
plt.imshow(image, cmap="gray")

<matplotlib.image.AxesImage at 0x14e076a4d50>
```

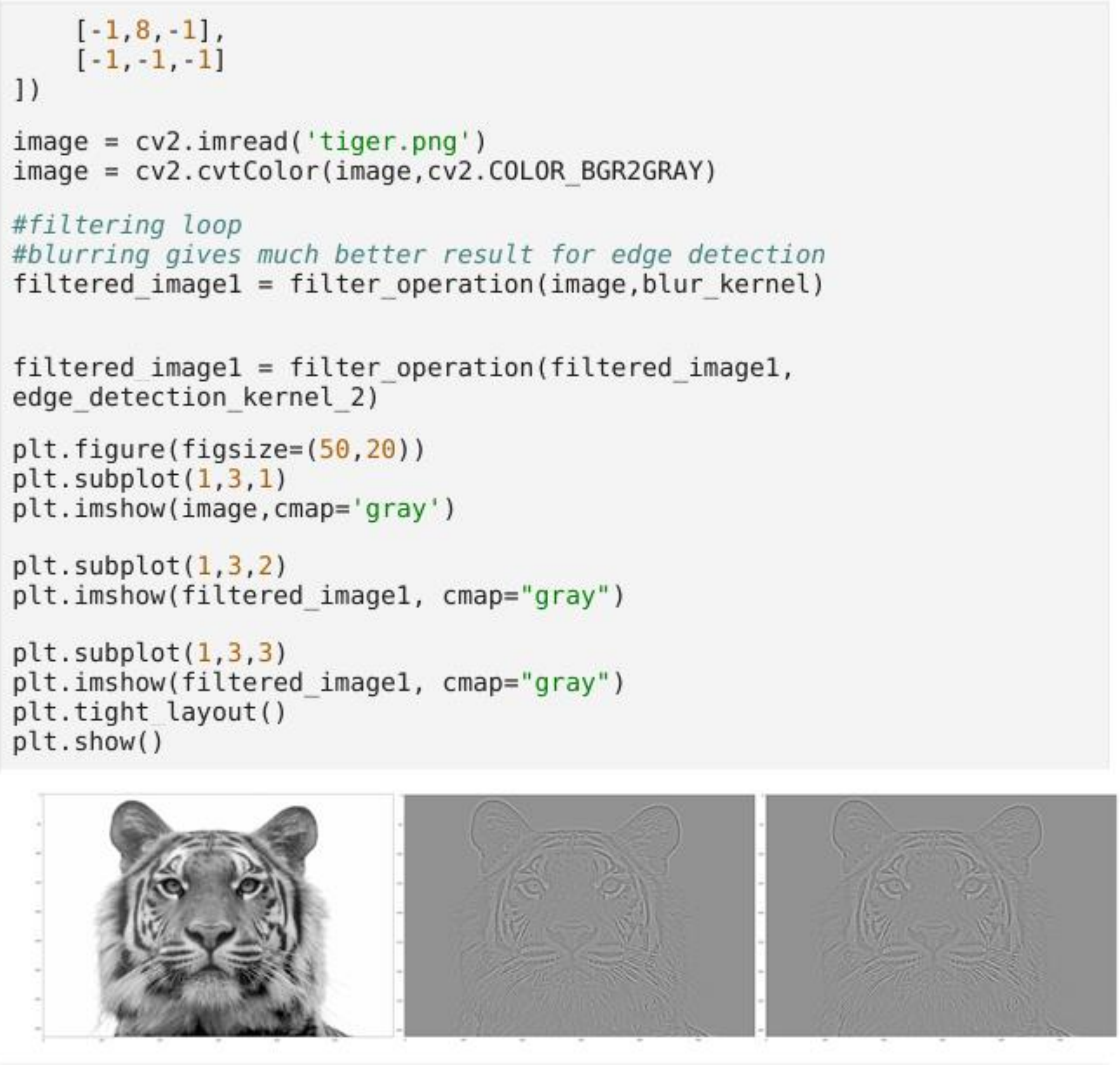




```
filtered_image = filter_operation(image,sharpening_kernel)
#sharpened_image = cv2.filter2D(image, -20, sharpening_kernel)

plt.imshow(filtered_image ,cmap='gray',vmin=0,vmax=255)

<matplotlib.image.AxesImage at 0x14e08f809d0>
```

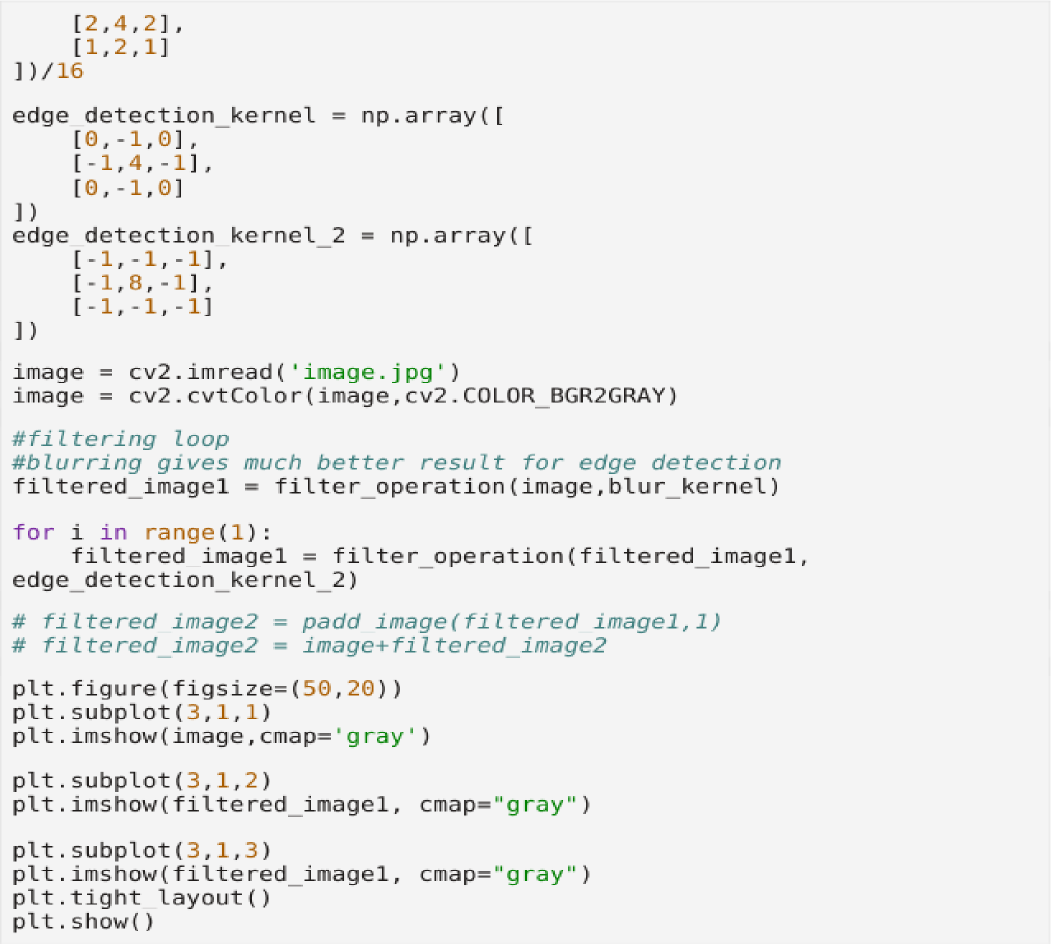


```
#filtering funtion
import numpy as np
import cv2
import matplotlib.pyplot as plt

def matrix_sum(mat_1,mat_2):
    sum = 0
    for i in range(mat_1.shape[0]):
        for j in range(mat_1.shape[1]):
            sum = sum + mat_1[i][j] * mat_2[i][j]
    return sum

def filter_operation(image,kernel):
    #must use a odd size of filter
    kernel_center = (kernel.shape[0]-1)//2
    kernel_dimension = kernel.shape[0]
    image_height = image.shape[0]
    image_width = image.shape[1]
    out_image_height = int(image_height-2*kernel_center)
    out_image_width = int(image_width-2*kernel_center)
    out_image = np.zeros((out_image_height,out_image_width))
    #print(image.shape)
    #print(out_image.shape)

    for row in range(out_image_height):
        for column in range(out_image_width):
            mat = 
image[row:row+kernel_dimension,column:column+kernel_dimension]
            #print(mat)
            out_image[row,column] = 
matrix_sum(mat,kernel)/kernel_dimension/kernel_dimension

    #print("out image")
    #print(out_image)
    return out_image

#blur kernel
blur_kernel = np.array([
    [1,2,1],
    [2,4,2],
    [1,2,1]
])/16

edge_detection_kernel = np.array([
    [0,-1,0],
    [-1,4,-1],
    [0,-1,0]
])
edge_detection_kernel_2 = np.array([
    [-1,-1,-1],
```

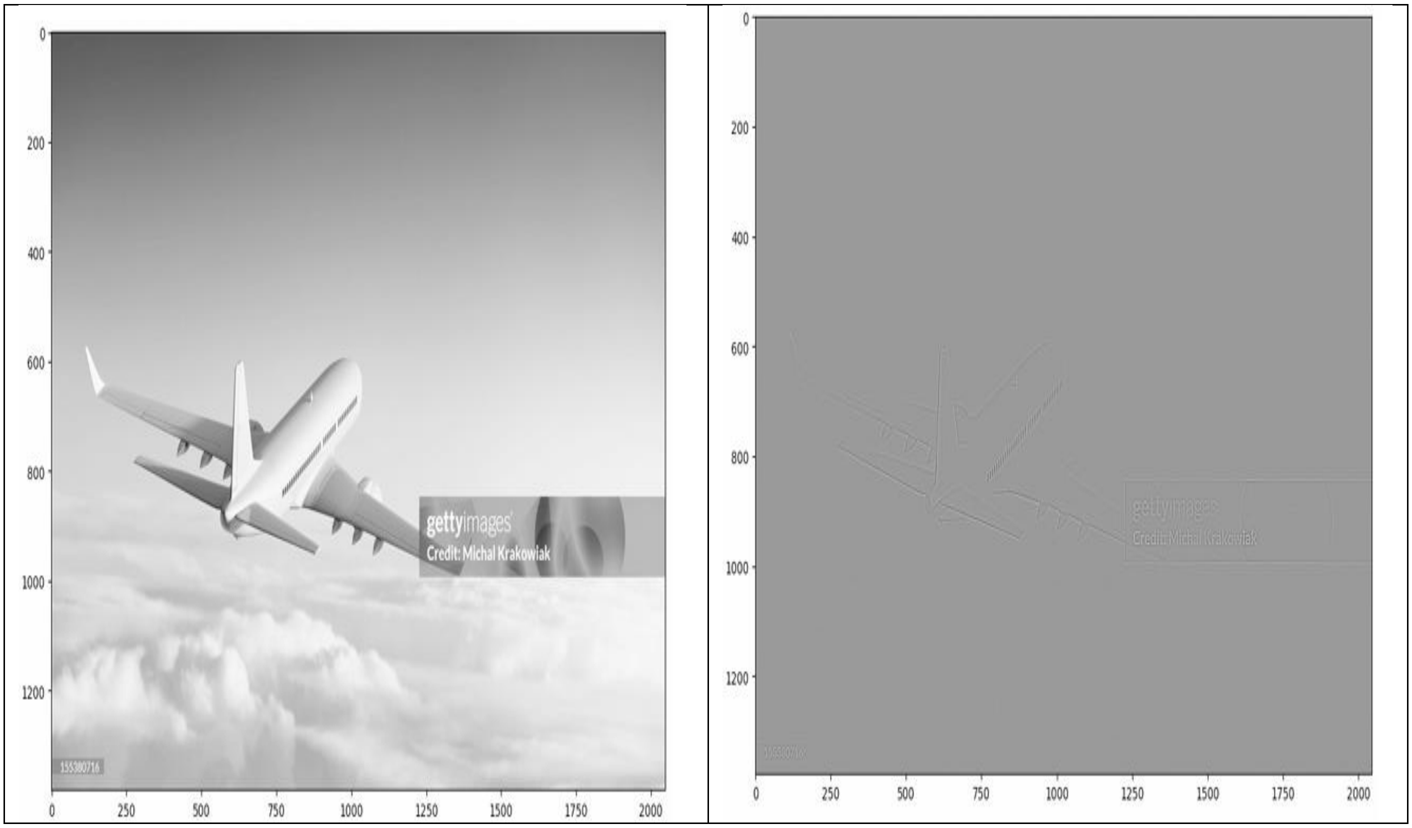


```
    [-1,8,-1],
    [-1,-1,-1]
])

image = cv2.imread('tiger.png')
image = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)

#filtering loop
#blurring gives much better result for edge detection
filtered_image1 = filter_operation(image,blur_kernel)


filtered_image1 = filter_operation(filtered_image1,
edge_detection_kernel_2)

plt.figure(figsize=(50,20))
plt.subplot(1,3,1)
plt.imshow(image,cmap='gray')

plt.subplot(1,3,2)
plt.imshow(filtered_image1, cmap="gray")

plt.subplot(1,3,3)
plt.imshow(filtered_image1, cmap="gray")
plt.tight_layout()
plt.show()
```

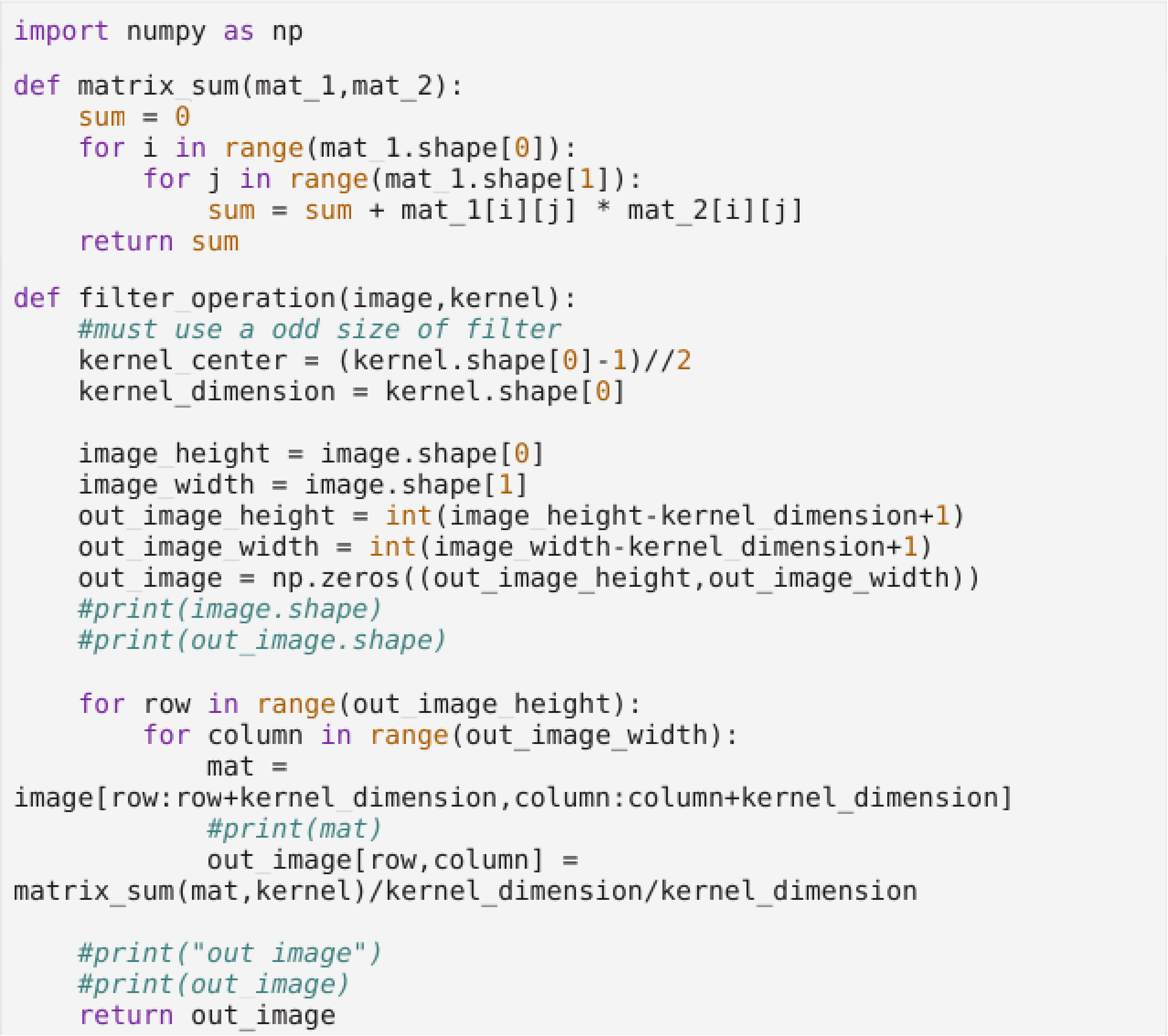
```
#filtering funtion
import numpy as np
import cv2
import matplotlib.pyplot as plt

def matrix_sum(mat_1,mat_2):
    sum = 0
    for i in range(mat_1.shape[0]):
        for j in range(mat_1.shape[1]):
            sum = sum + mat_1[i][j] * mat_2[i][j]
    return sum
def filter_operation(image,kernel):
    #must use a odd size of filter
    kernel_center = (kernel.shape[0]-1)//2
    kernel_dimension = kernel.shape[0]
    image_height = image.shape[0]
    image_width = image.shape[1]
    out_image_height = int(image_height-2*kernel_center)
    out_image_width = int(image_width-2*kernel_center)
    out_image = np.zeros((out_image_height,out_image_width))
    #print(image.shape)
    #print(out_image.shape)

    for row in range(out_image_height):
        for column in range(out_image_width):
            mat =
image[row:row+kernel_dimension,column:column+kernel_dimension]
            #print(mat)
            out_image[row,column] =
matrix_sum(mat,kernel)/kernel_dimension/kernel_dimension

    #print("out_image")
    #print(out_image)
    return out_image

def padd_image(img2,n):
    left = np.zeros((img2.shape[0],n))
    left = left + 255
    img2 = np.concatenate((img2, left), axis=1)
    img2 = np.concatenate((left, img2), axis=1)

    up = np.zeros((n,img2.shape[1]))
    up = up + 255
    img2 = np.concatenate((img2, up), axis=0)
    img2 = np.concatenate((up, img2), axis=0)
    return img2

#blur_kernel
blur_kernel = np.array([
    [1,2,1],
```



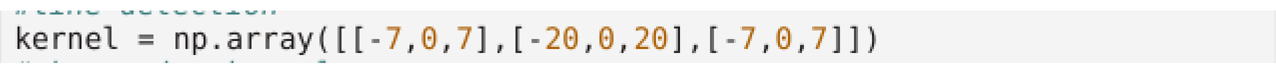
```
    [2,4,2],
    [1,2,1]
])/16

edge_detection_kernel = np.array([
    [0,-1,0],
    [-1,4,-1],
    [0,-1,0]
])
edge_detection_kernel_2 = np.array([
    [-1,-1,-1],
    [-1,8,-1],
    [-1,-1,-1]
])

image = cv2.imread('image.jpg')
image = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)

#filtering loop
#blurring gives much better result for edge detection
filtered_image1 = filter_operation(image,blur_kernel)

for i in range(1):
    filtered_image1 = filter_operation(filtered_image1,
edge_detection_kernel_2)

# filtered_image2 = padd_image(filtered_image1,1)
# filtered_image2 = image+filtered_image2

plt.figure(figsize=(50,20))
plt.subplot(3,1,1)
plt.imshow(image,cmap='gray')

plt.subplot(3,1,2)
plt.imshow(filtered_image1, cmap="gray")

plt.subplot(3,1,3)
plt.imshow(filtered_image1, cmap="gray")
plt.tight_layout()
plt.show()
```



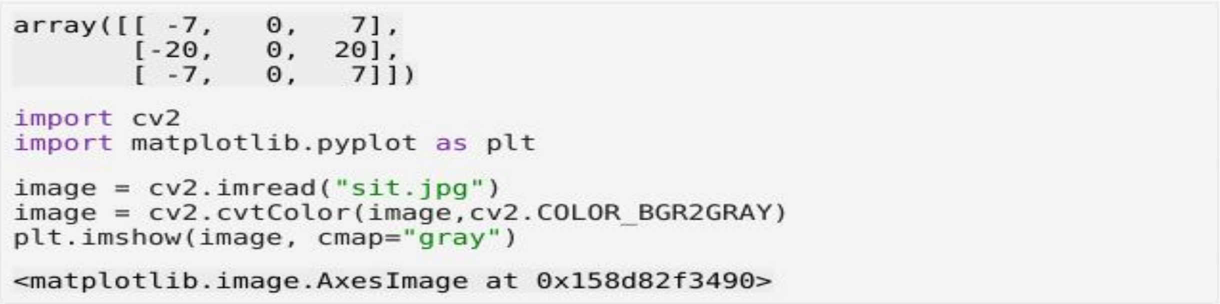
Point detection



```
import numpy as np

def matrix_sum(mat_1,mat_2):
    sum = 0
    for i in range(mat_1.shape[0]):
        for j in range(mat_1.shape[1]):
            sum = sum + mat_1[i][j] * mat_2[i][j]
    return sum

def filter_operation(image,kernel):
    #must use a odd size of filter
    kernel_center = (kernel.shape[0]-1)//2
    kernel_dimension = kernel.shape[0]

    image_height = image.shape[0]
    image_width = image.shape[1]
    out_image_height = int(image_height-kernel_dimension+1)
    out_image_width = int(image_width-kernel_dimension+1)
    out_image = np.zeros((out_image_height,out_image_width))
    #print(image.shape)
    #print(out_image.shape)

    for row in range(out_image_height):
        for column in range(out_image_width):
            mat =
image[row:row+kernel_dimension,column:column+kernel_dimension]
            #print(mat)
            out_image[row,column] =
matrix_sum(mat,kernel)/kernel_dimension/kernel_dimension

    #print("out image")
    #print(out_image)
    return out_image
```



```
#line detection
kernel = np.array([[-7,0,7],[-20,0,20],[-7,0,7]])
```

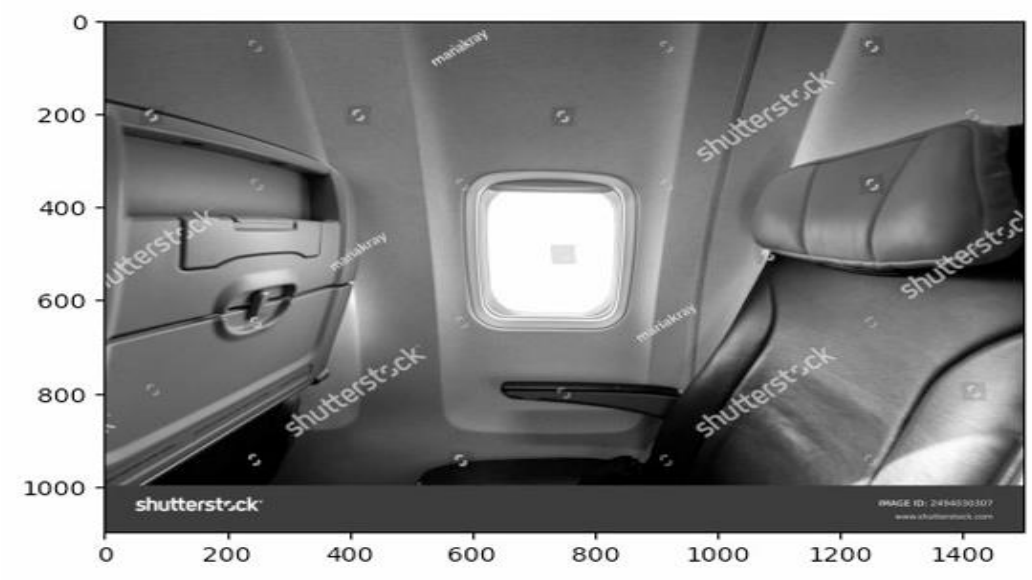


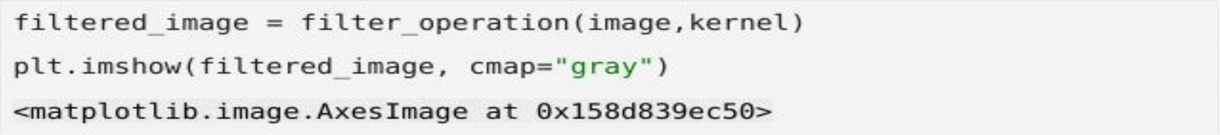
```
array([[ -7,   0,   7],
       [-20,   0,  20],
       [ -7,   0,   7]])
```

```python
import cv2
import matplotlib.pyplot as plt
```

```python
image = cv2.imread("sit.jpg")
image = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
plt.imshow(image, cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x158d82f3490>
```

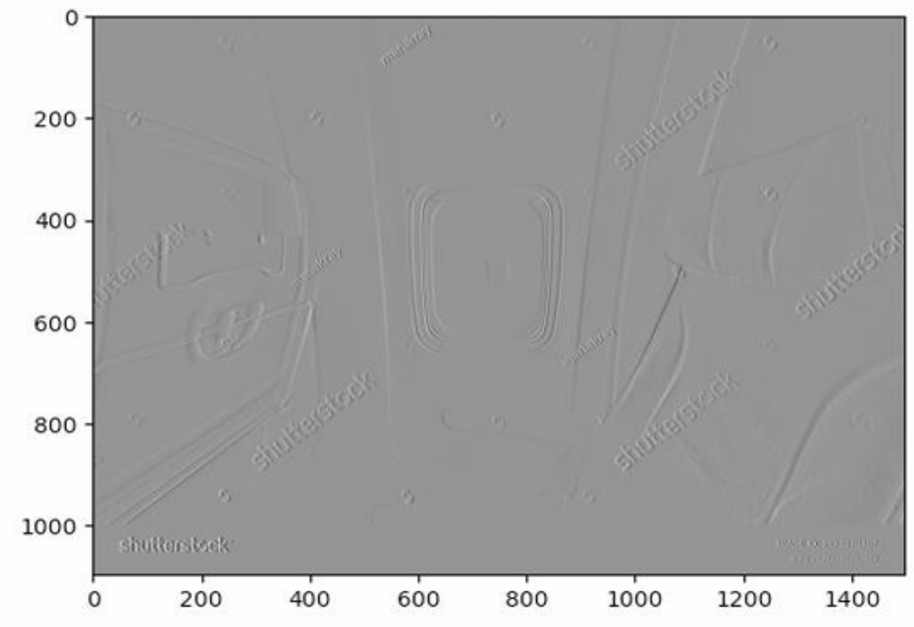


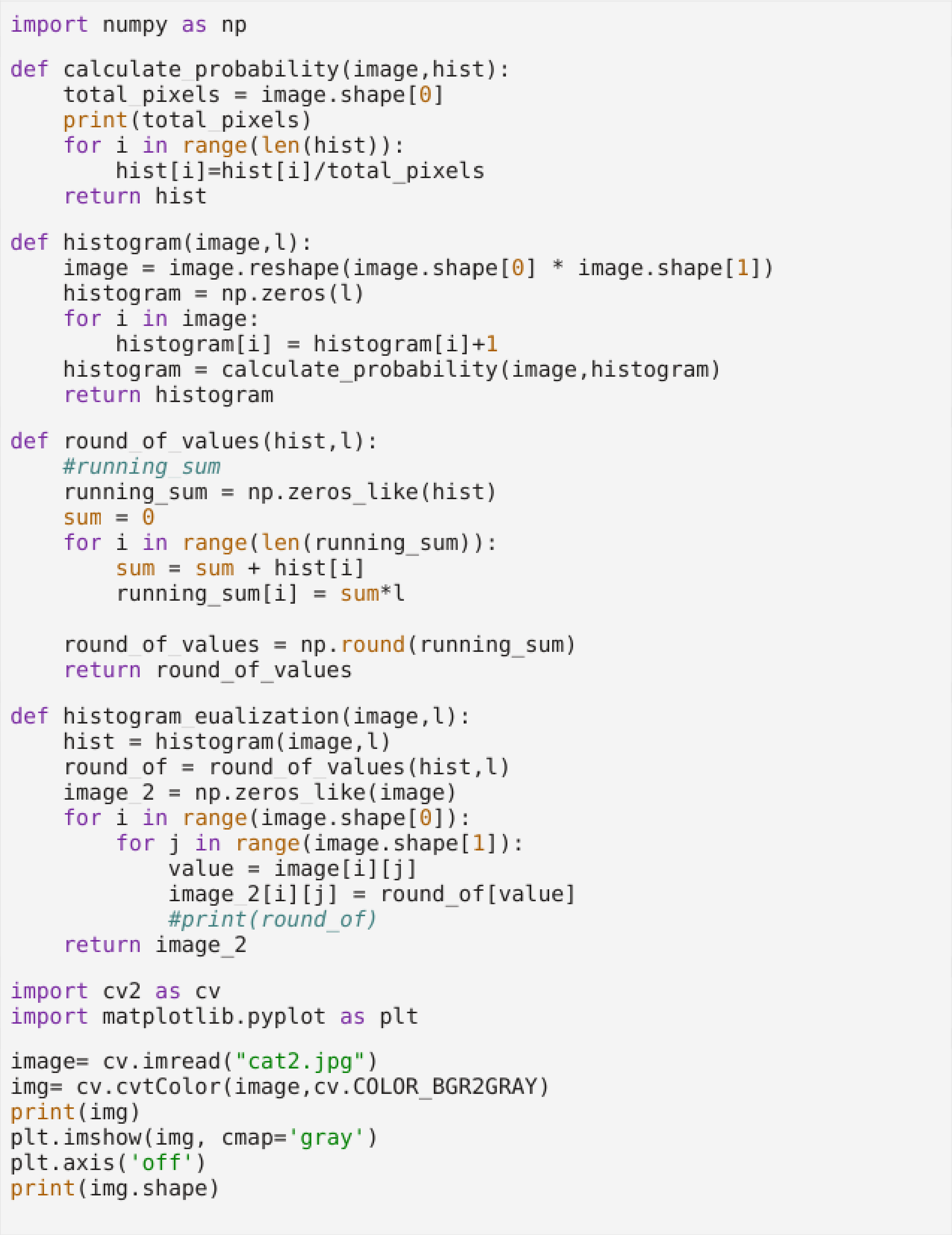


```python
filtered_image = filter_operation(image,kernel)
```

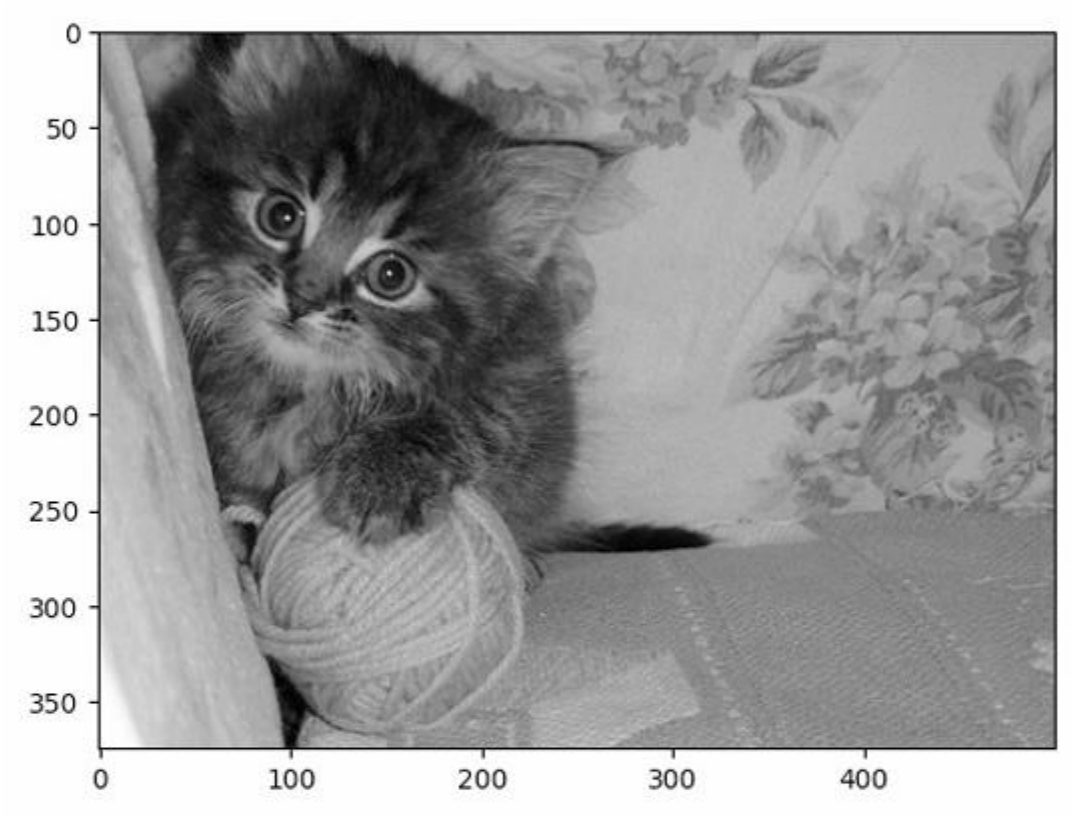
```python
plt.imshow(filtered_image, cmap="gray")
```

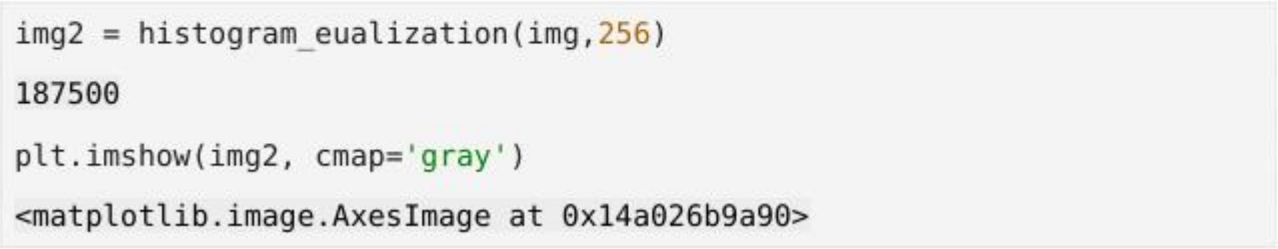
```
<matplotlib.image.AxesImage at 0x158d839ec50>
```





```
import numpy as np

def calculate_probability(image,hist):
    total_pixels = image.shape[0]
    print(total_pixels)
    for i in range(len(hist)):
        hist[i]=hist[i]/total_pixels
    return hist

def histogram(image,l):
    image = image.reshape(image.shape[0] * image.shape[1])
    histogram = np.zeros(l)
    for i in image:
        histogram[i] = histogram[i]+1
    histogram = calculate_probability(image,histogram)
    return histogram

def round_of_values(hist,l):
    #running_sum
    running_sum = np.zeros_like(hist)
    sum = 0
    for i in range(len(running_sum)):
        sum = sum + hist[i]
        running_sum[i] = sum*l

    round_of_values = np.round(running_sum)
    return round_of_values

def histogram_eualization(image,l):
    hist = histogram(image,l)
    round_of = round_of_values(hist,l)
    image_2 = np.zeros_like(image)
    for i in range(image.shape[0]):
        for j in range(image.shape[1]):
            value = image[i][j]
            image_2[i][j] = round_of[value]
            #print(round_of)
    return image_2

import cv2 as cv
import matplotlib.pyplot as plt

image= cv.imread("cat2.jpg")
img= cv.cvtColor(image,cv.COLOR_BGR2GRAY)
print(img)
plt.imshow(img, cmap='gray')
plt.axis('off')
print(img.shape)
```



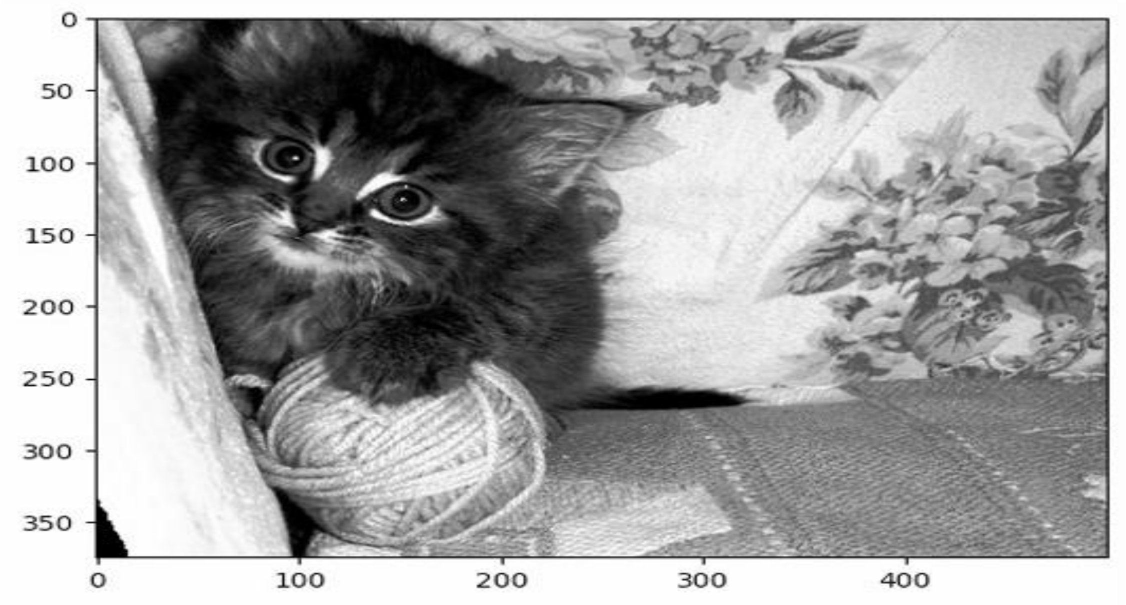


```
img2 = histogram_eualization(img,256)
```

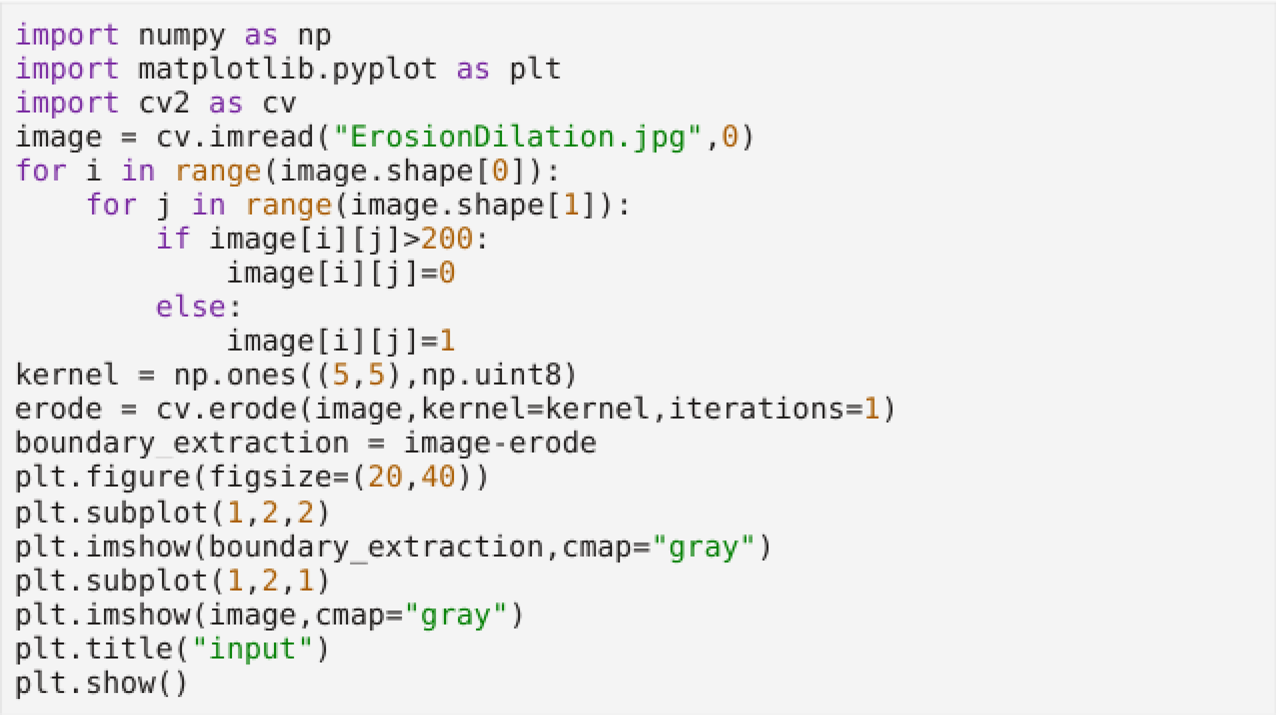
```
187500
```

```
plt.imshow(img2, cmap='gray')
```

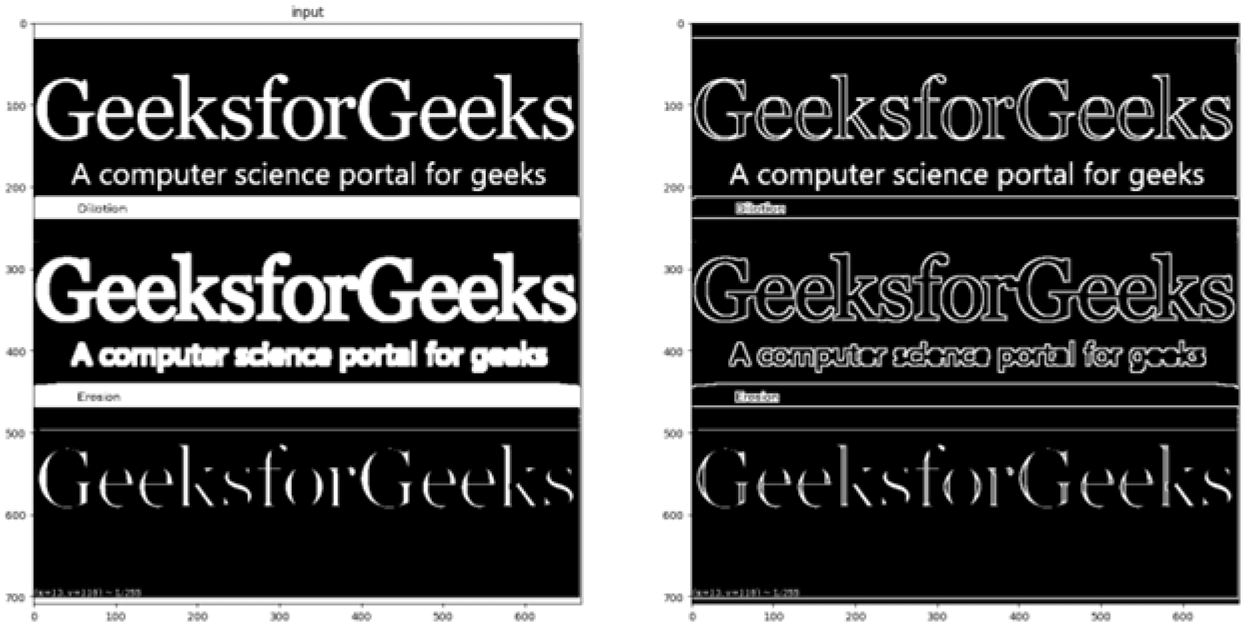
```
<matplotlib.image.AxesImage at 0x14a026b9a90>
```



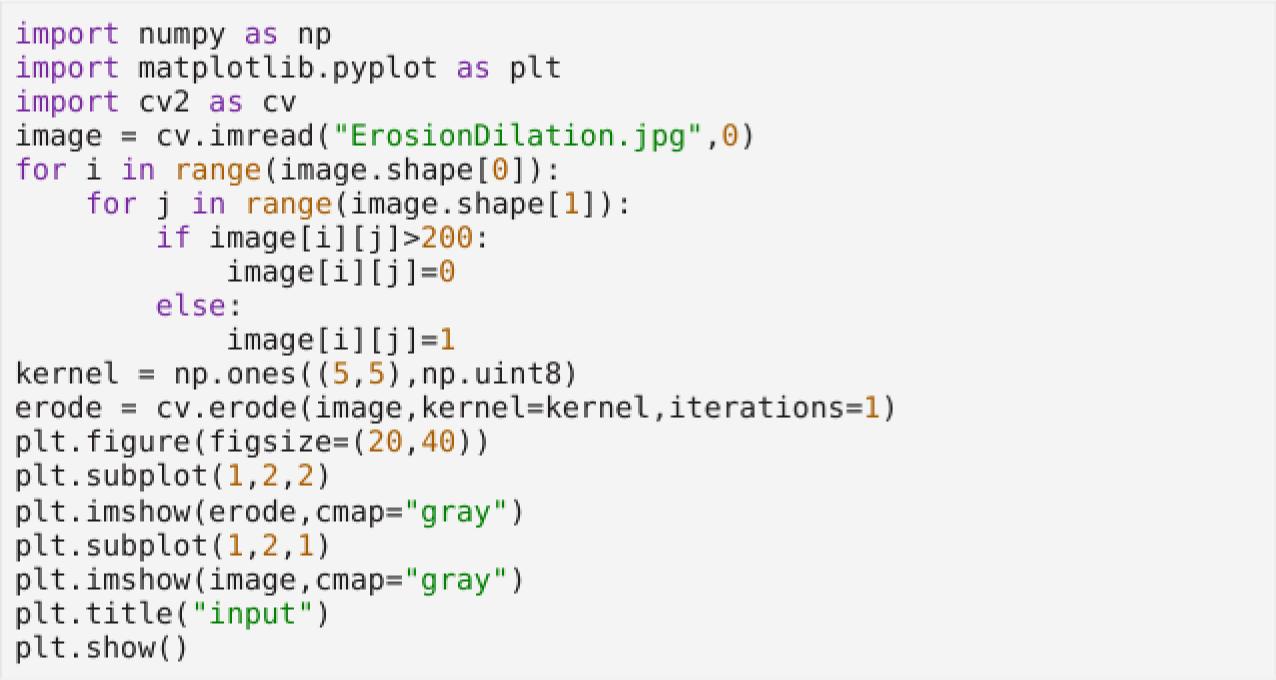
Boundary Extraction



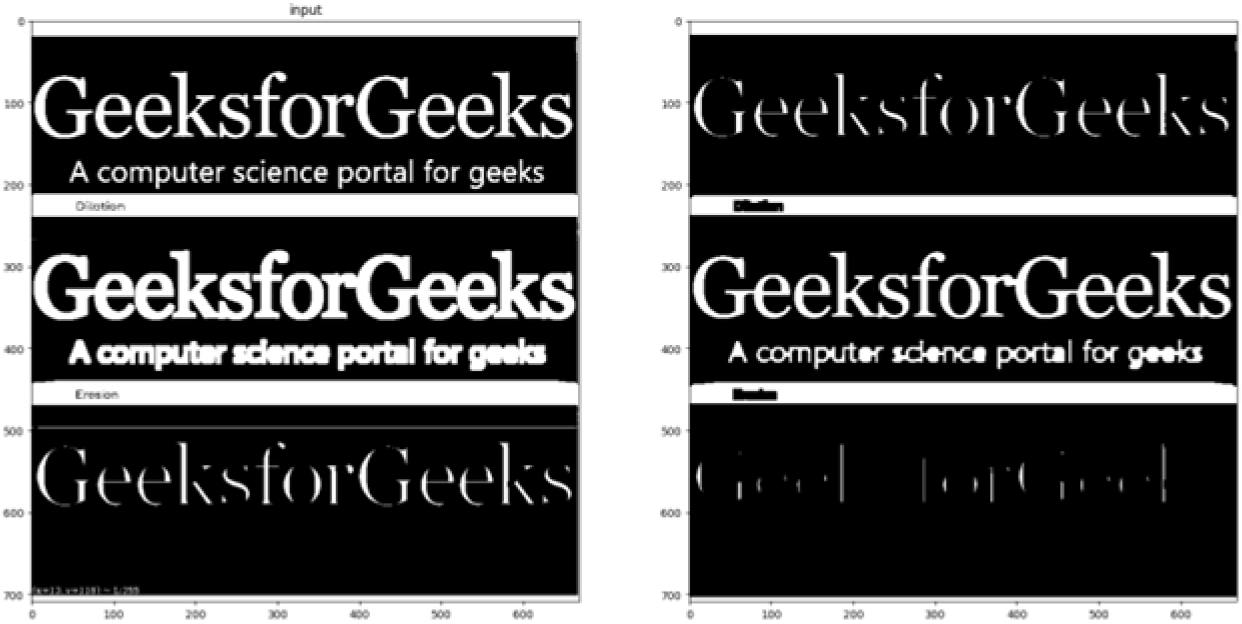
```
import numpy as np
import matplotlib.pyplot as plt
import cv2 as cv
image = cv.imread("ErosionDilation.jpg",0)
for i in range(image.shape[0]):
    for j in range(image.shape[1]):
        if image[i][j]>200:
            image[i][j]=0
        else:
            image[i][j]=1
kernel = np.ones((5,5),np.uint8)
erode = cv.erode(image,kernel=kernel,iterations=1)
boundary_extraction = image-erode
plt.figure(figsize=(20,40))
plt.subplot(1,2,2)
plt.imshow(boundary_extraction,cmap="gray")
plt.subplot(1,2,1)
plt.imshow(image,cmap="gray")
plt.title("input")
plt.show()
```



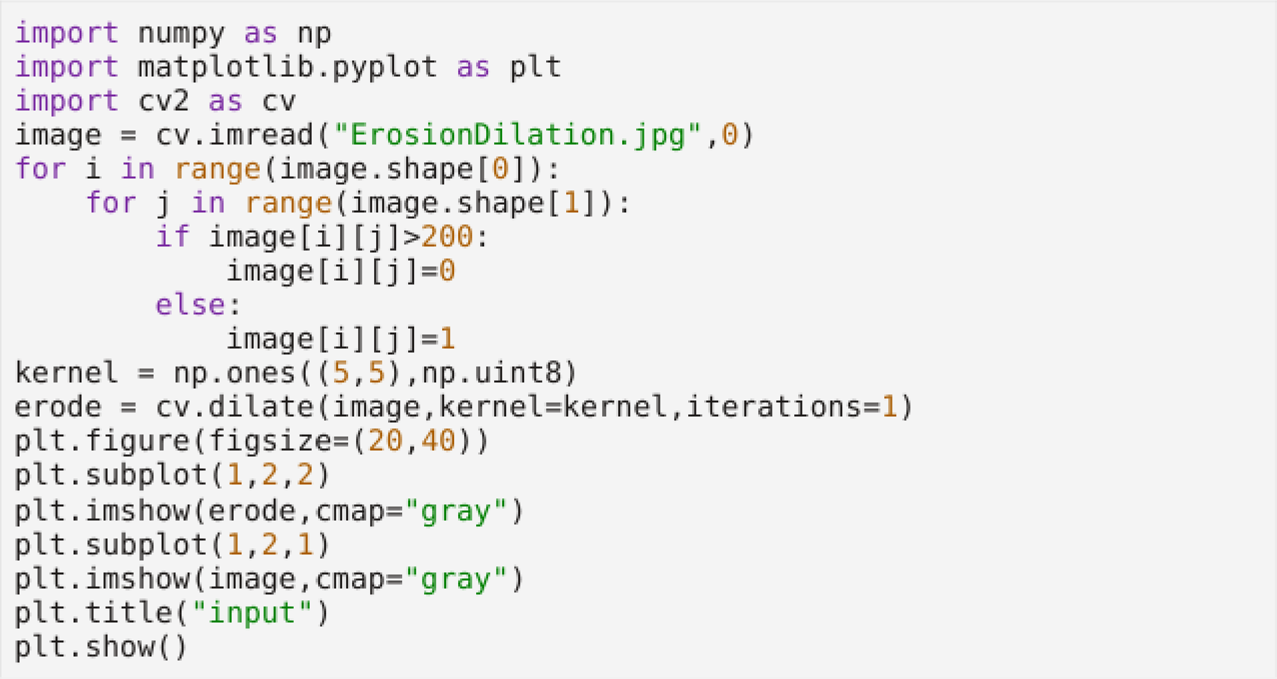
Erosion



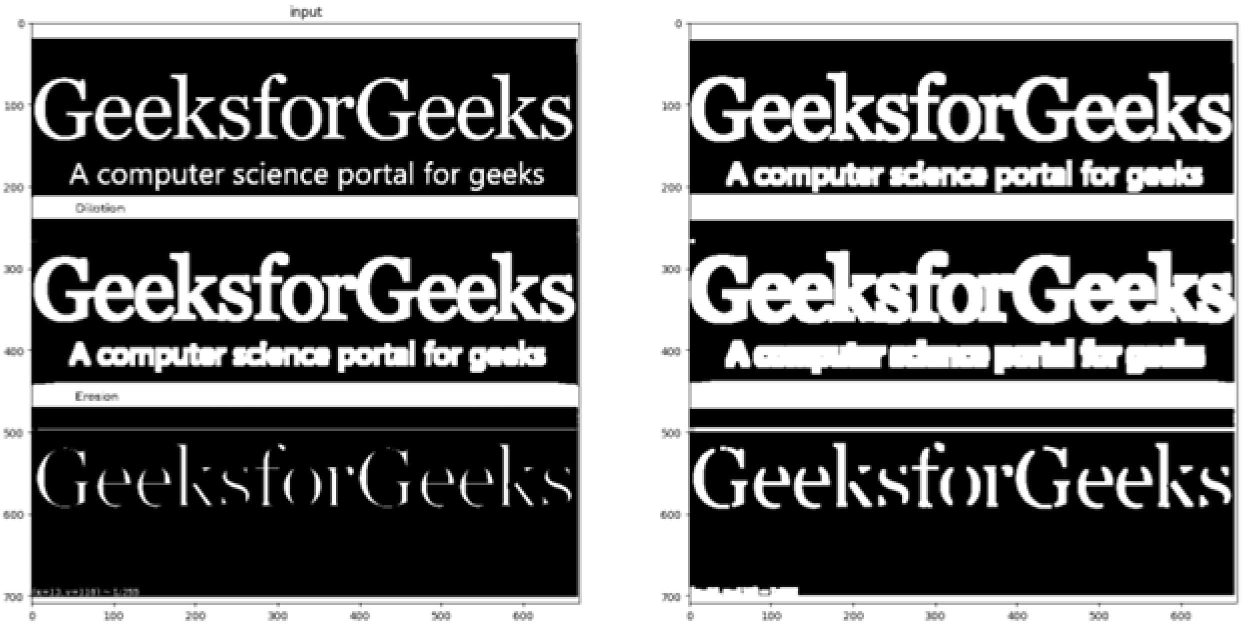
```python
import numpy as np
import matplotlib.pyplot as plt
import cv2 as cv
image = cv.imread("ErosionDilation.jpg",0)
for i in range(image.shape[0]):
    for j in range(image.shape[1]):
        if image[i][j]>200:
            image[i][j]=0
        else:
            image[i][j]=1
kernel = np.ones((5,5),np.uint8)
erode = cv.erode(image,kernel=kernel,iterations=1)
plt.figure(figsize=(20,40))
plt.subplot(1,2,2)
plt.imshow(erode,cmap="gray")
plt.subplot(1,2,1)
plt.imshow(image,cmap="gray")
plt.title("input")
plt.show()
```



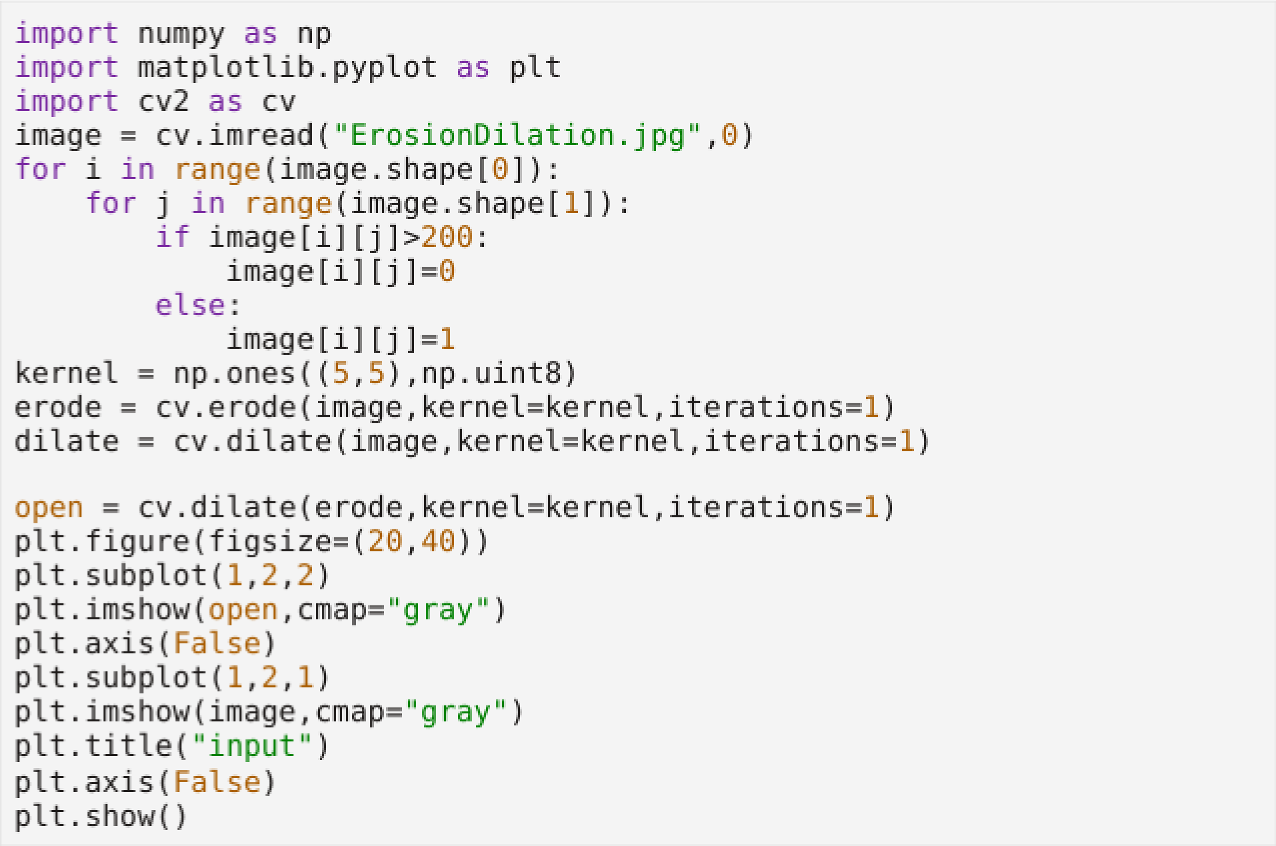
Dialation



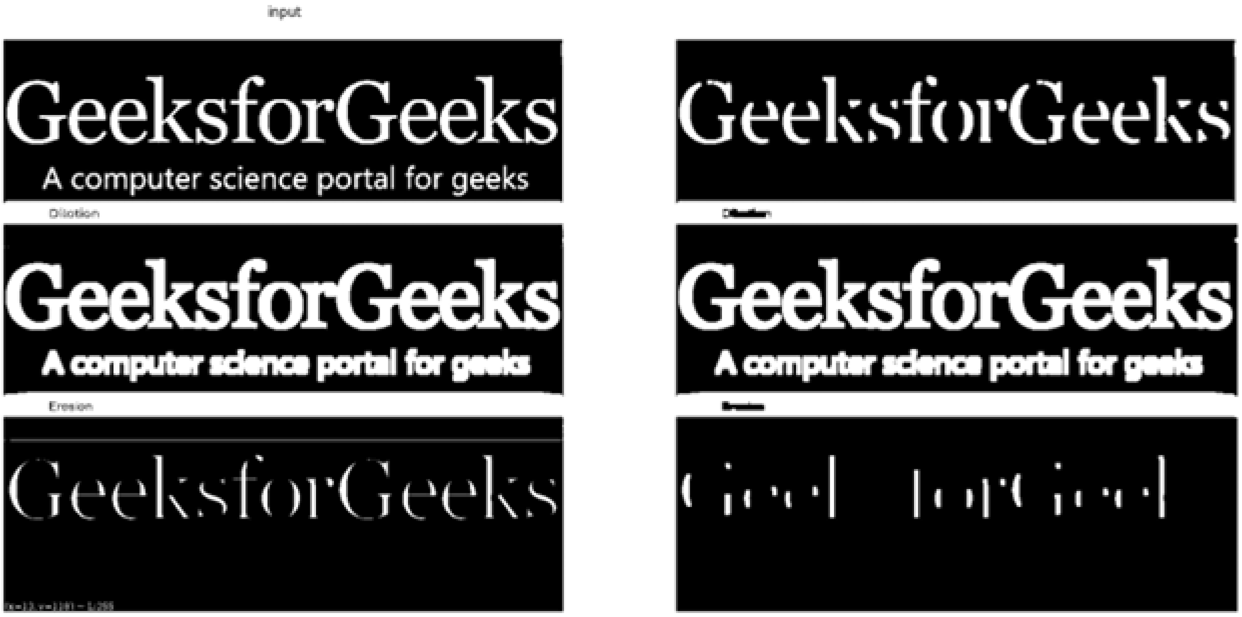
```
import numpy as np
import matplotlib.pyplot as plt
import cv2 as cv
image = cv.imread("ErosionDilation.jpg",0)
for i in range(image.shape[0]):
    for j in range(image.shape[1]):
        if image[i][j]>200:
            image[i][j]=0
        else:
            image[i][j]=1
kernel = np.ones((5,5),np.uint8)
erode = cv.dilate(image,kernel=kernel,iterations=1)
plt.figure(figsize=(20,40))
plt.subplot(1,2,2)
plt.imshow(erode,cmap="gray")
plt.subplot(1,2,1)
plt.imshow(image,cmap="gray")
plt.title("input")
plt.show()
```



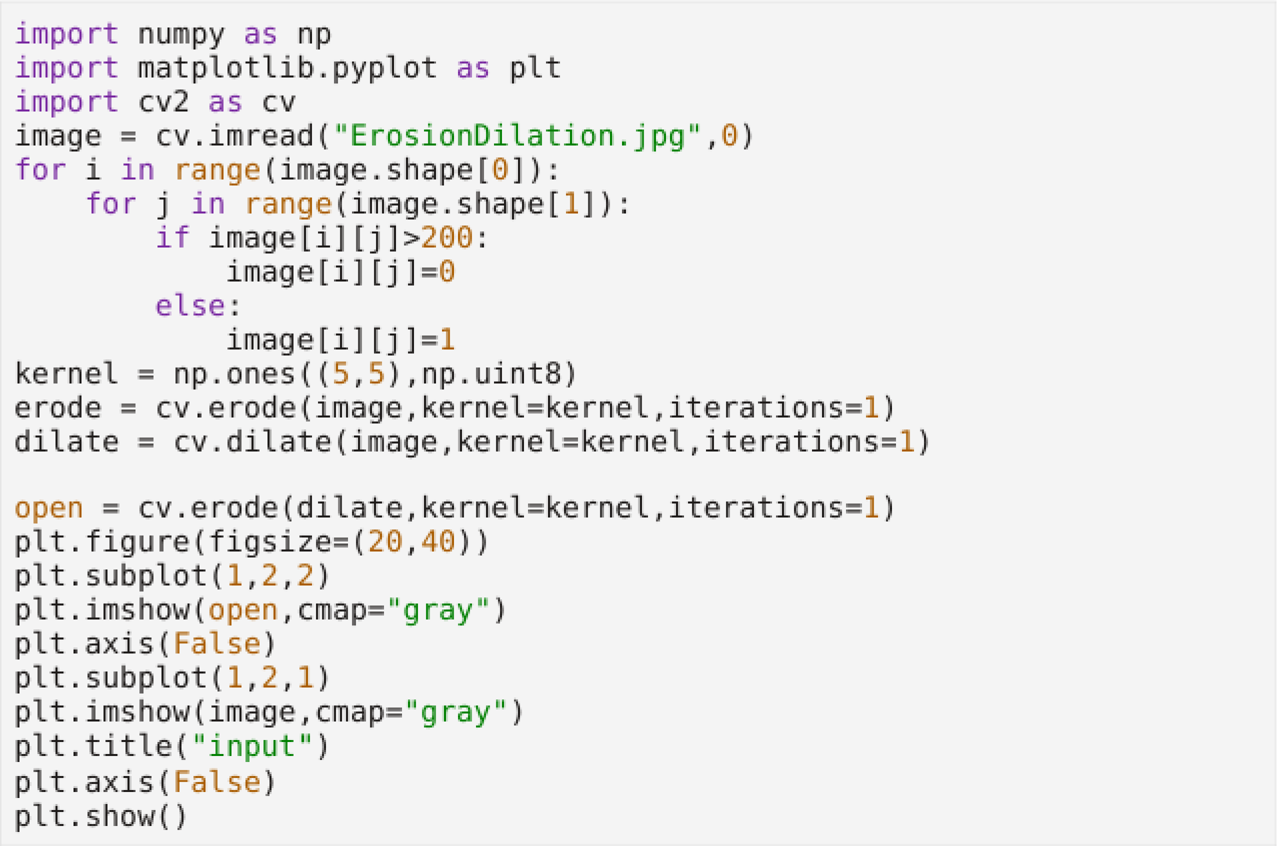
Opening



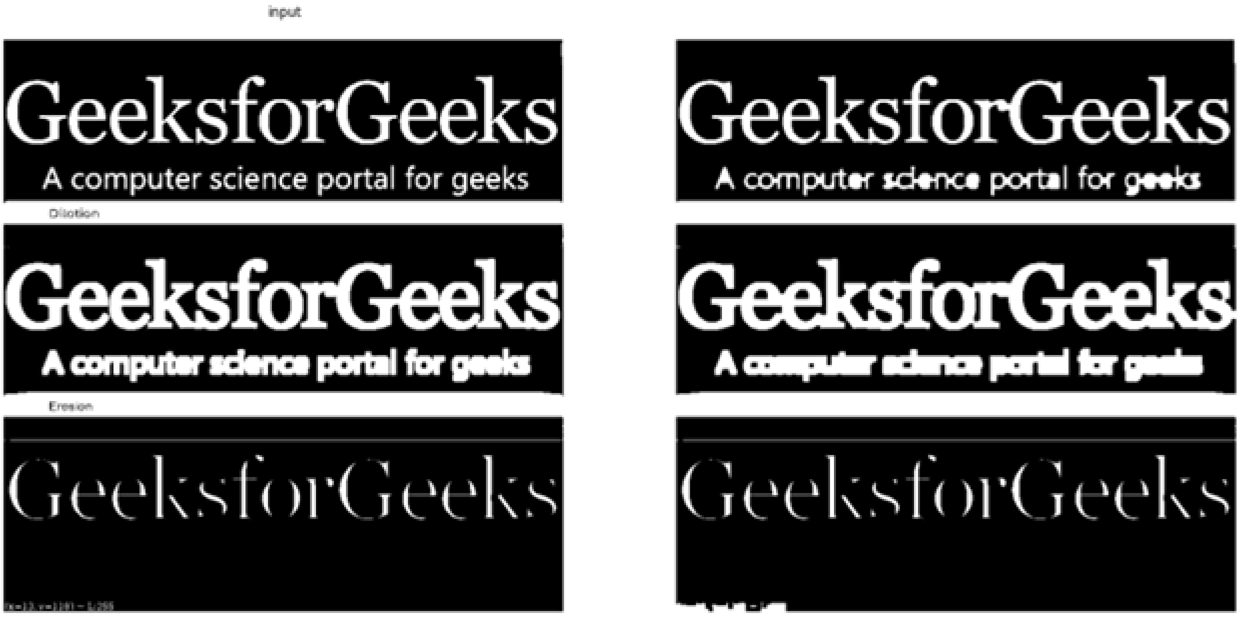
```
import numpy as np
import matplotlib.pyplot as plt
import cv2 as cv
image = cv.imread("ErosionDilation.jpg",0)
for i in range(image.shape[0]):
    for j in range(image.shape[1]):
        if image[i][j]>200:
            image[i][j]=0
        else:
            image[i][j]=1
kernel = np.ones((5,5),np.uint8)
erode = cv.erode(image,kernel=kernel,iterations=1)
dilate = cv.dilate(image,kernel=kernel,iterations=1)

open = cv.dilate(erode,kernel=kernel,iterations=1)
plt.figure(figsize=(20,40))
plt.subplot(1,2,2)
plt.imshow(open,cmap="gray")
plt.axis(False)
plt.subplot(1,2,1)
plt.imshow(image,cmap="gray")
plt.title("input")
plt.axis(False)
plt.show()
```


input
GeeksforGeeks
A computer science portal for geeks
GeeksforGeeks
Dilation
GeeksforGeeks
A computer science portal for geeks
Dilation
GeeksforGeeks
A computer science portal for geeks
Erosion
GeeksforGeeks
Erosion

Closing



```
import numpy as np
import matplotlib.pyplot as plt
import cv2 as cv
image = cv.imread("ErosionDilation.jpg",0)
for i in range(image.shape[0]):
    for j in range(image.shape[1]):
        if image[i][j]>200:
            image[i][j]=0
        else:
            image[i][j]=1
kernel = np.ones((5,5),np.uint8)
erode = cv.erode(image,kernel=kernel,iterations=1)
dilate = cv.dilate(image,kernel=kernel,iterations=1)

open = cv.erode(dilate,kernel=kernel,iterations=1)
plt.figure(figsize=(20,40))
plt.subplot(1,2,2)
plt.imshow(open,cmap="gray")
plt.axis(False)
plt.subplot(1,2,1)
plt.imshow(image,cmap="gray")
plt.title("input")
plt.axis(False)
plt.show()
```


input
GeeksforGeeks
A computer science portal for geeks
Dilation
Erosion