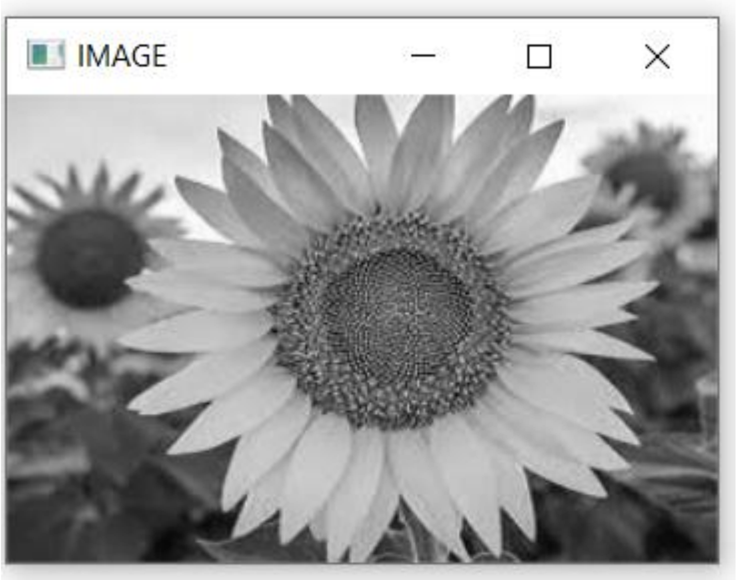
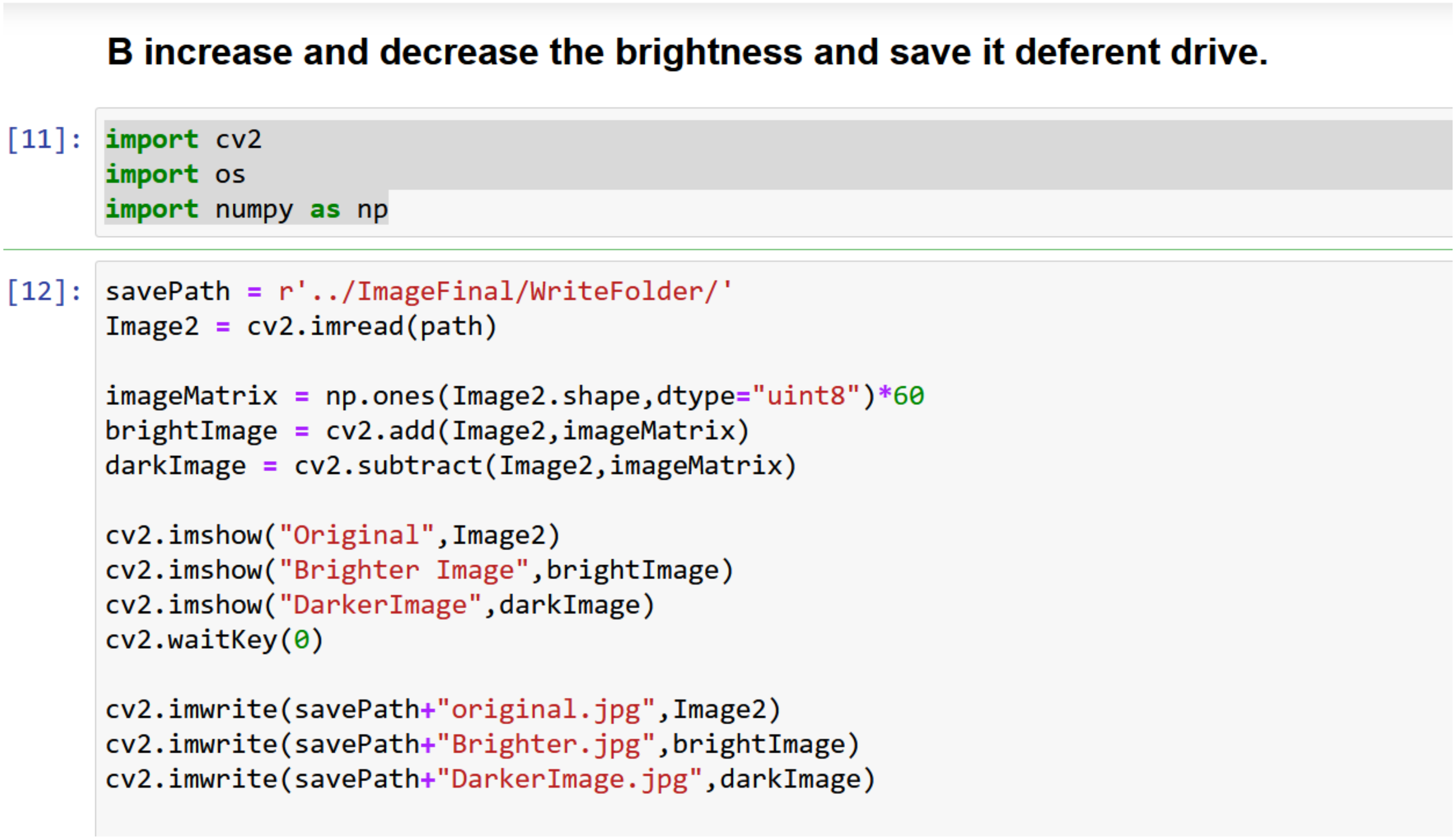
1. **Write a python program using OpenCV library and perform the following with respect to an image**

a. Reading Gray-Scale Image

```
import cv2
import matplotlib.pyplot as plt
path = r'./Images/Image1.jpg'
image1 = cv2.imread(path,0)
cv2.imshow("IMAGE",image1)
cv2.waitKey(0)
```

OUTPUT:

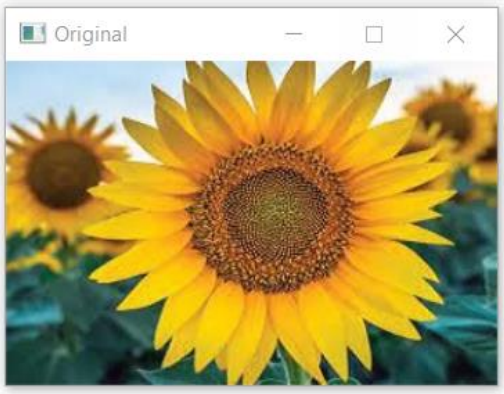


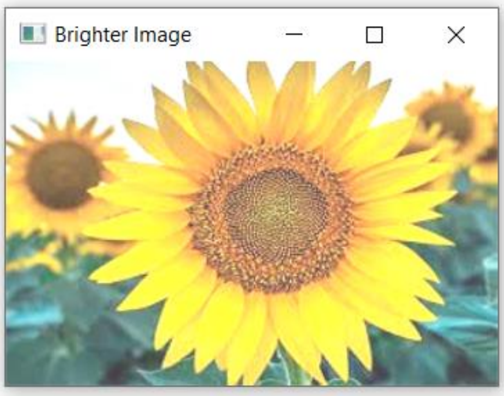


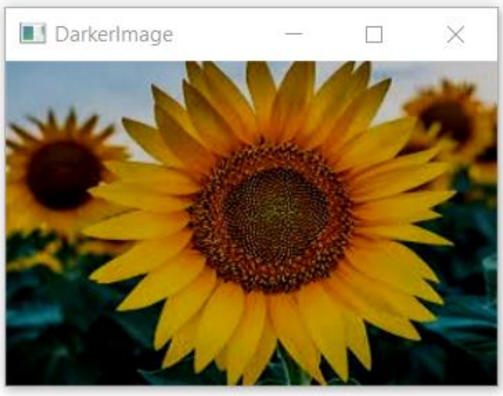
B increase and decrease the brightness and save it deferent drive.

[11]:
```
import cv2
import os
import numpy as np
```

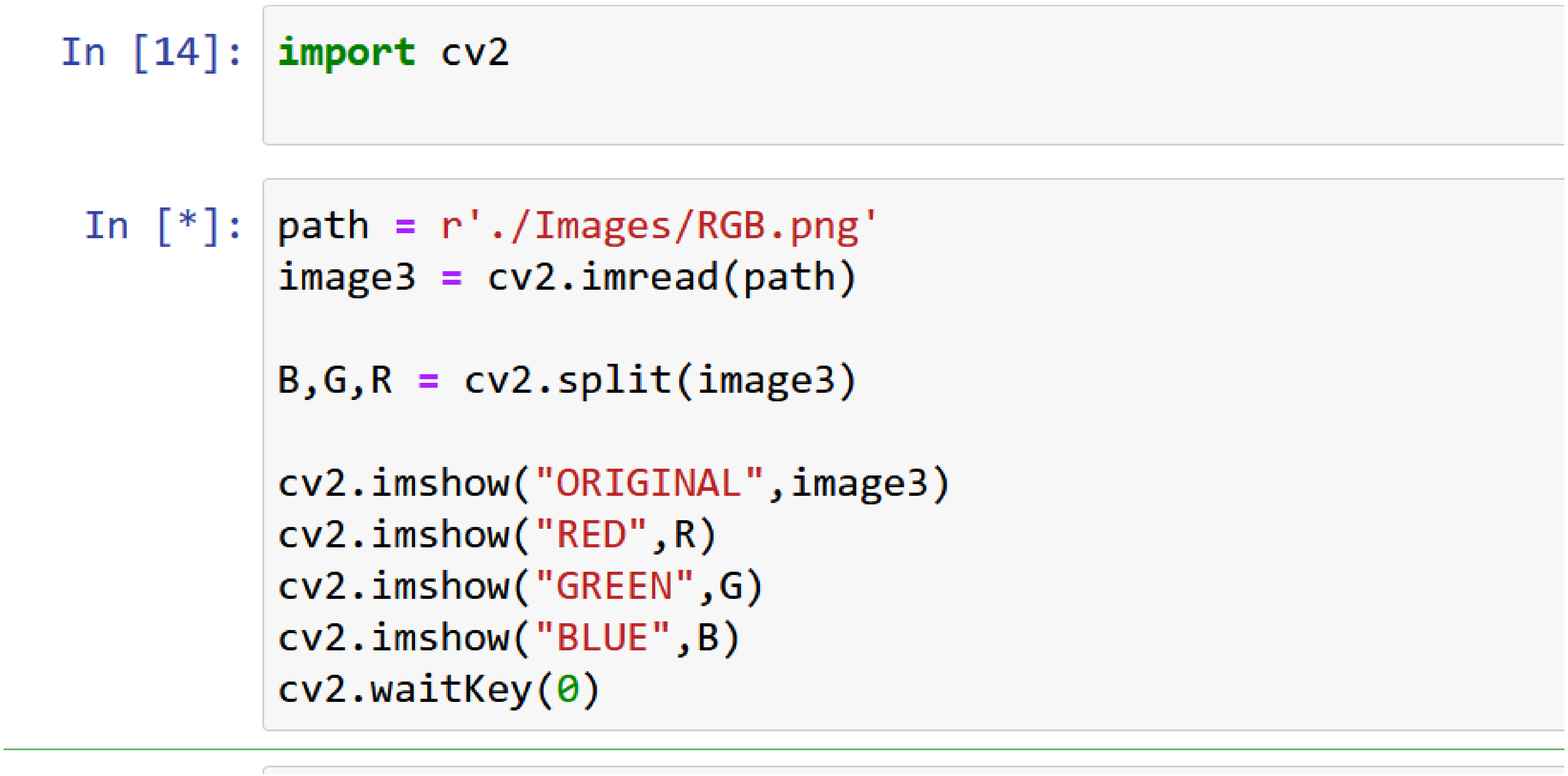
[12]:
```
savePath = r'../ImageFinal/WriteFolder/'
Image2 = cv2.imread(path)

imageMatrix = np.ones(Image2.shape,dtype="uint8")*60
brightImage = cv2.add(Image2,imageMatrix)
darkImage = cv2.subtract(Image2,imageMatrix)

cv2.imshow("Original",Image2)
cv2.imshow("Brighter Image",brightImage)
cv2.imshow("DarkerImage",darkImage)
cv2.waitKey(0)

cv2.imwrite(savePath+"original.jpg",Image2)
cv2.imwrite(savePath+"Brighter.jpg",brightImage)
cv2.imwrite(savePath+"DarkerImage.jpg",darkImage)
```





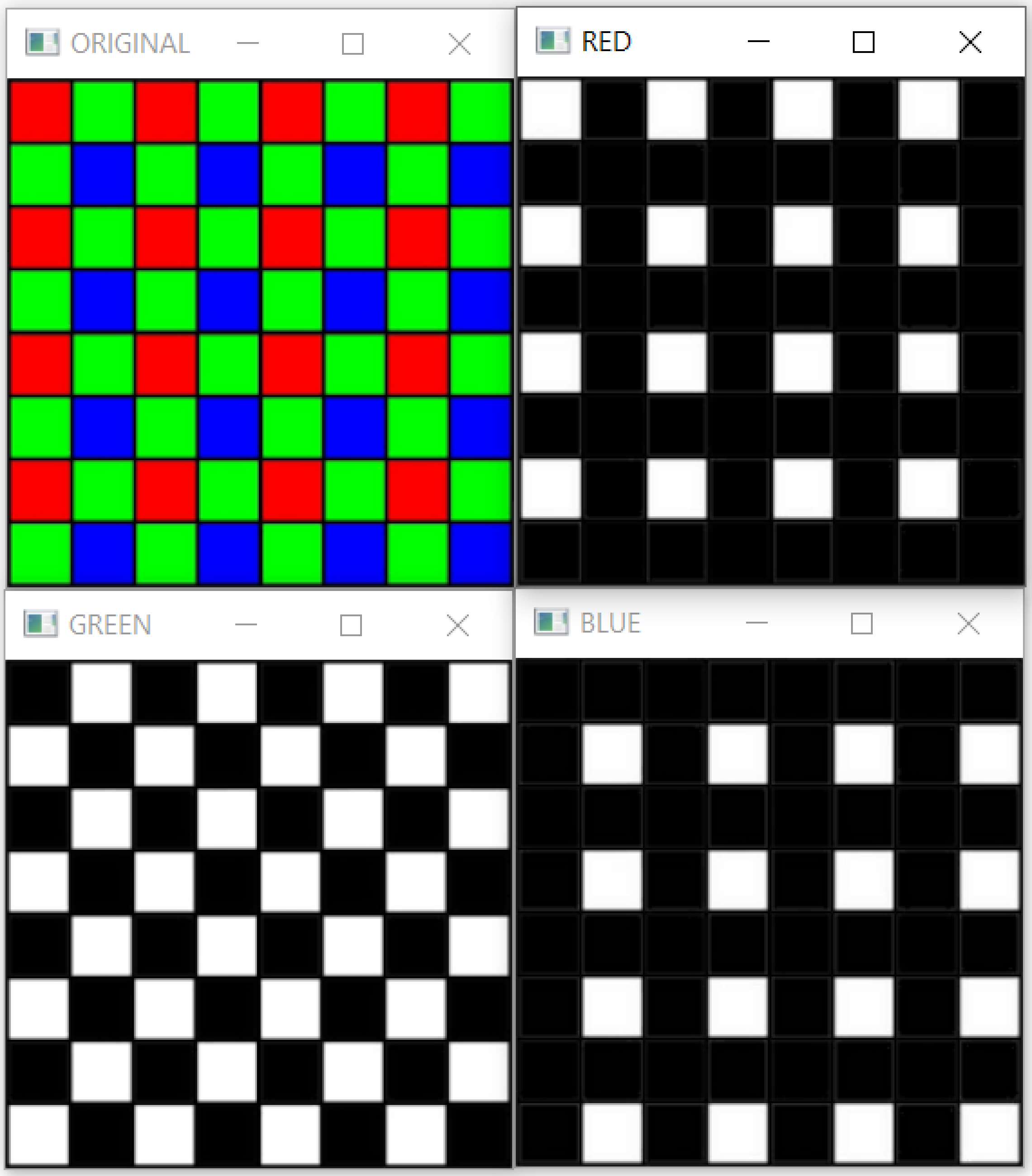


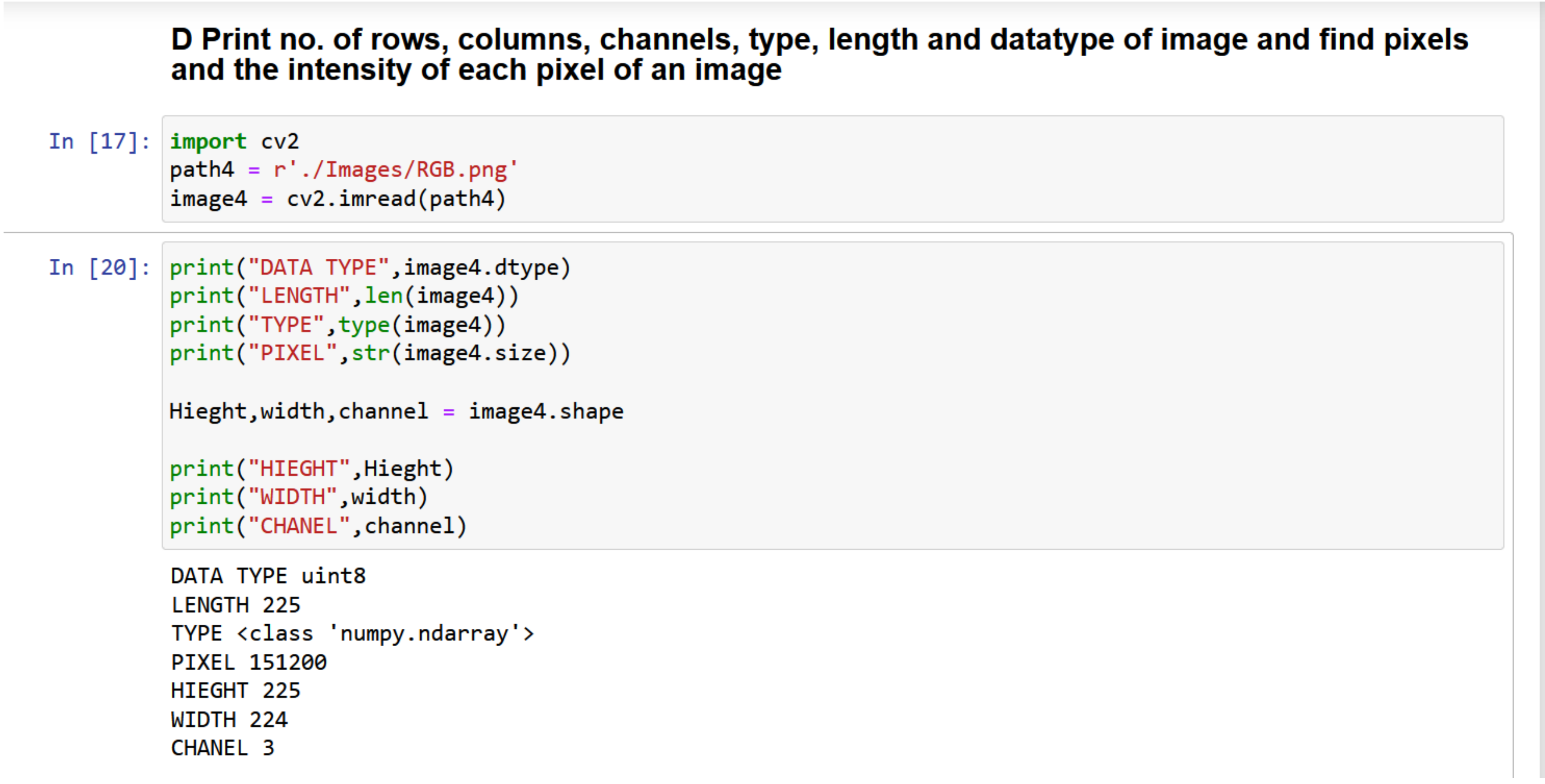
c. Split the image into three channel (R, G,B).



```
In [14]: import cv2
```

```
In [*]: path = r'./Images/RGB.png'
        image3 = cv2.imread(path)

        B,G,R = cv2.split(image3)

        cv2.imshow("ORIGINAL",image3)
        cv2.imshow("RED",R)
        cv2.imshow("GREEN",G)
        cv2.imshow("BLUE",B)
        cv2.waitKey(0)
```





D Print no. of rows, columns, channels, type, length and datatype of image and find pixels and the intensity of each pixel of an image

In [17]:

```
import cv2
path4 = r'./Images/RGB.png'
image4 = cv2.imread(path4)
```

In [20]:

```
print("DATA TYPE",image4.dtype)
print("LENGTH",len(image4))
print("TYPE",type(image4))
print("PIXEL",str(image4.size))

Hieght,width,channel = image4.shape

print("HIEGHT",Hieght)
print("WIDTH",width)
print("CHANEL",channel)
```

```
DATA TYPE uint8
LENGTH 225
TYPE <class 'numpy.ndarray'>
PIXEL 151200
HIEGHT 225
WIDTH 224
CHANEL 3
```

PROGRAM 2

Write a python program to perform the following with respect to an image using OpenCV library

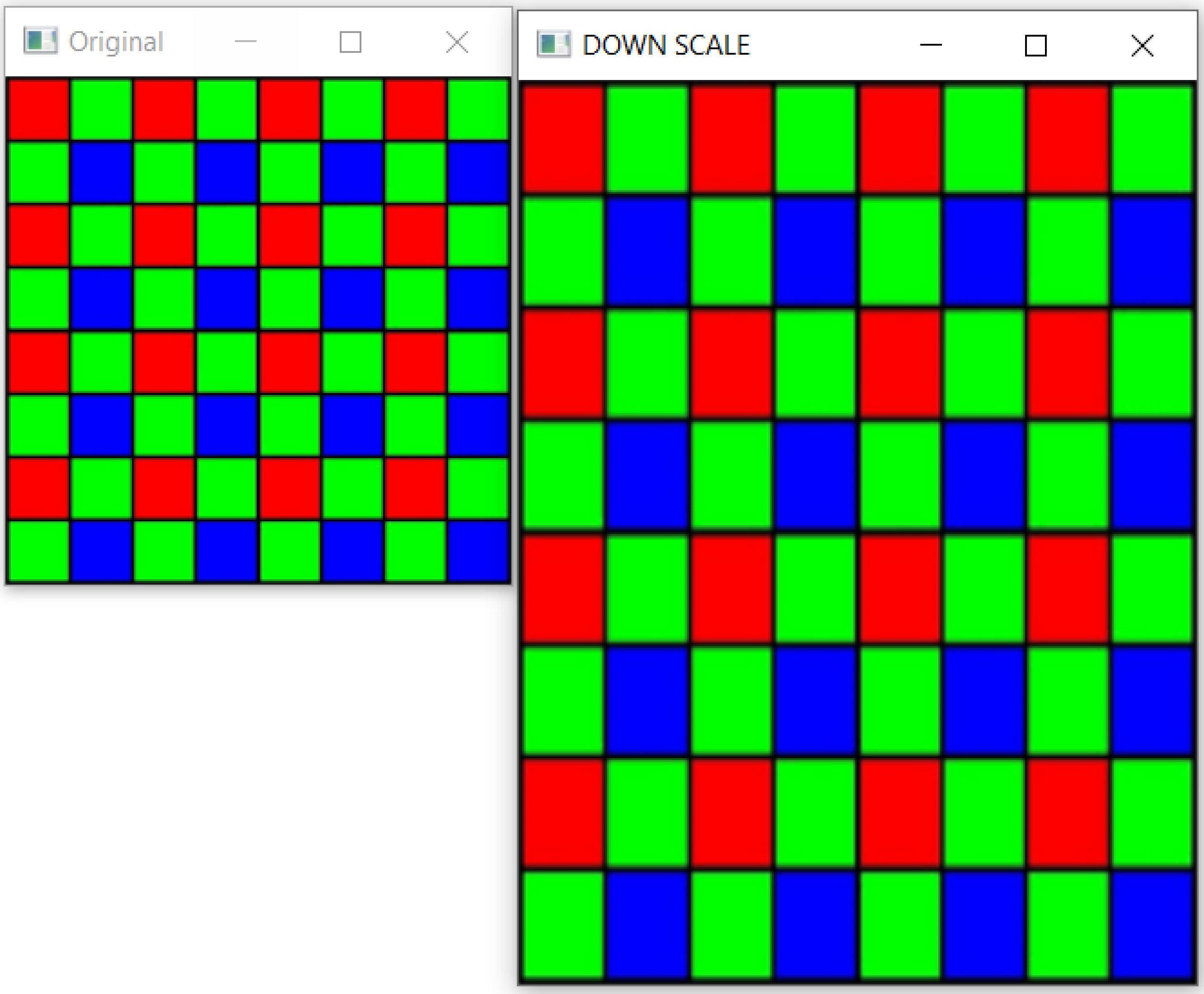
a) Read a colour image, convert it into grey scale image. Increase and decrease the contrast and display all the three images and save.

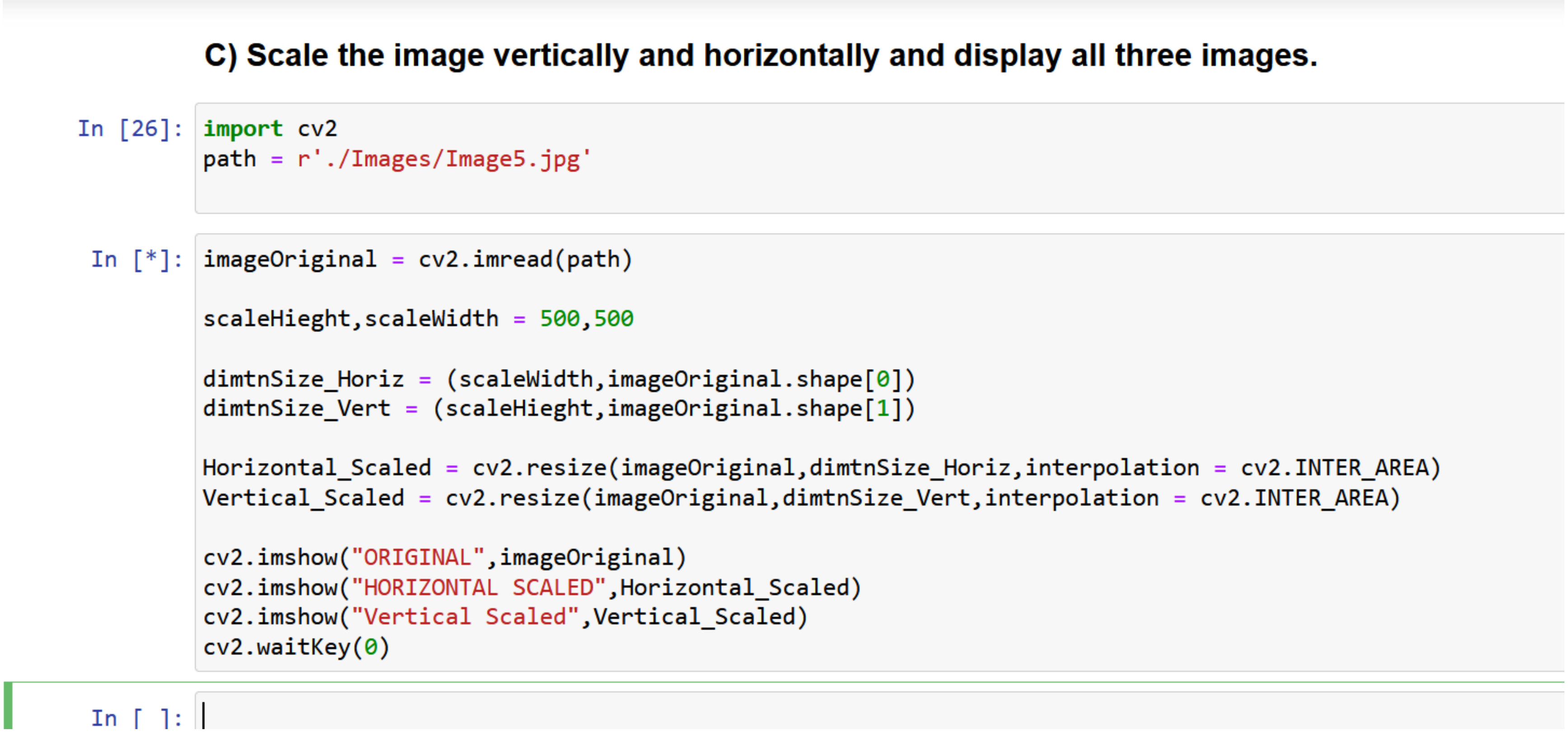
```
In [21]: # NOT COMPELETED
```

b) Downscale the image by resizing with custom height and width while not preserving the aspect ratio .

```
In [22]: import cv2
         path = r'./Images/Image22.jpg'
```

```
In [24]: imageOriginal = cv2.imread(path)

         downHgt,downWdth = 300,400
         downPoints = (downHgt,downWdth)

         downScaling = cv2.resize(imageOriginal,downPoints,interpolation = cv2.INTER_LINEAR)

         cv2.imshow("Original",imageOriginal)
         cv2.imshow("DOWN SCALE",downScaling)
         cv2.waitKey(0)

Out[24]: -1
```



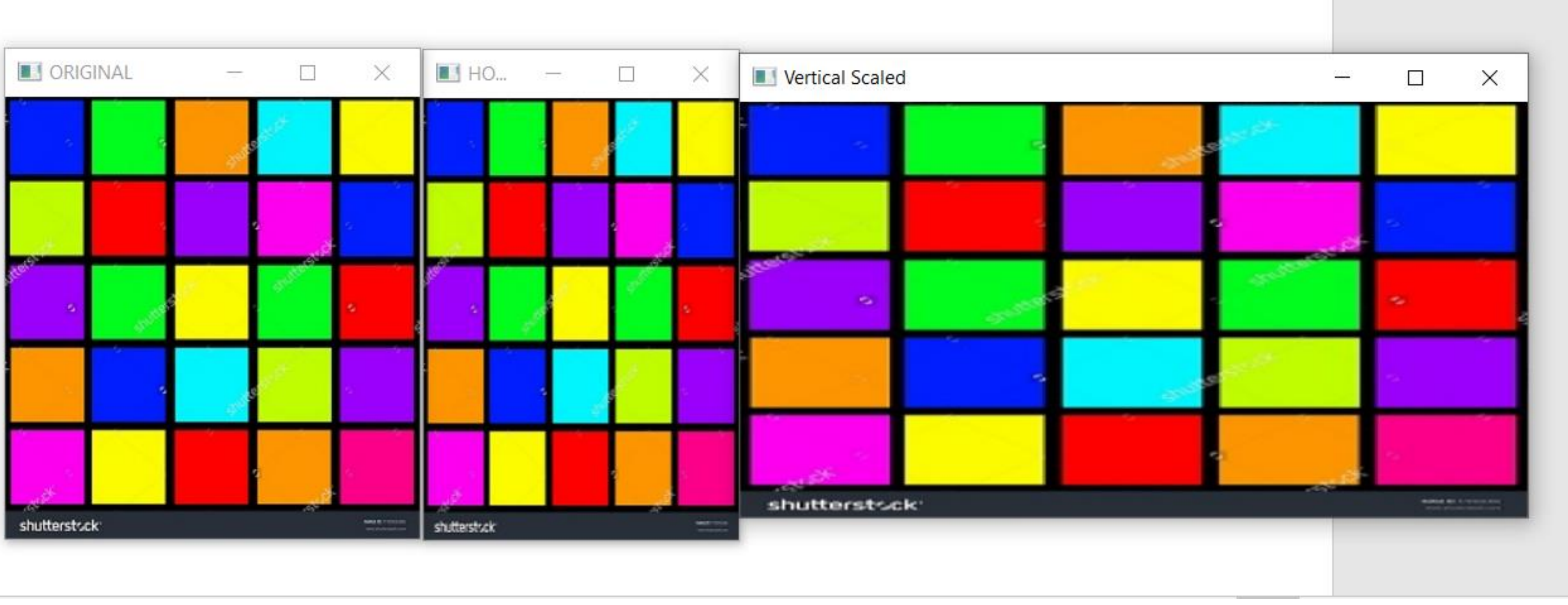


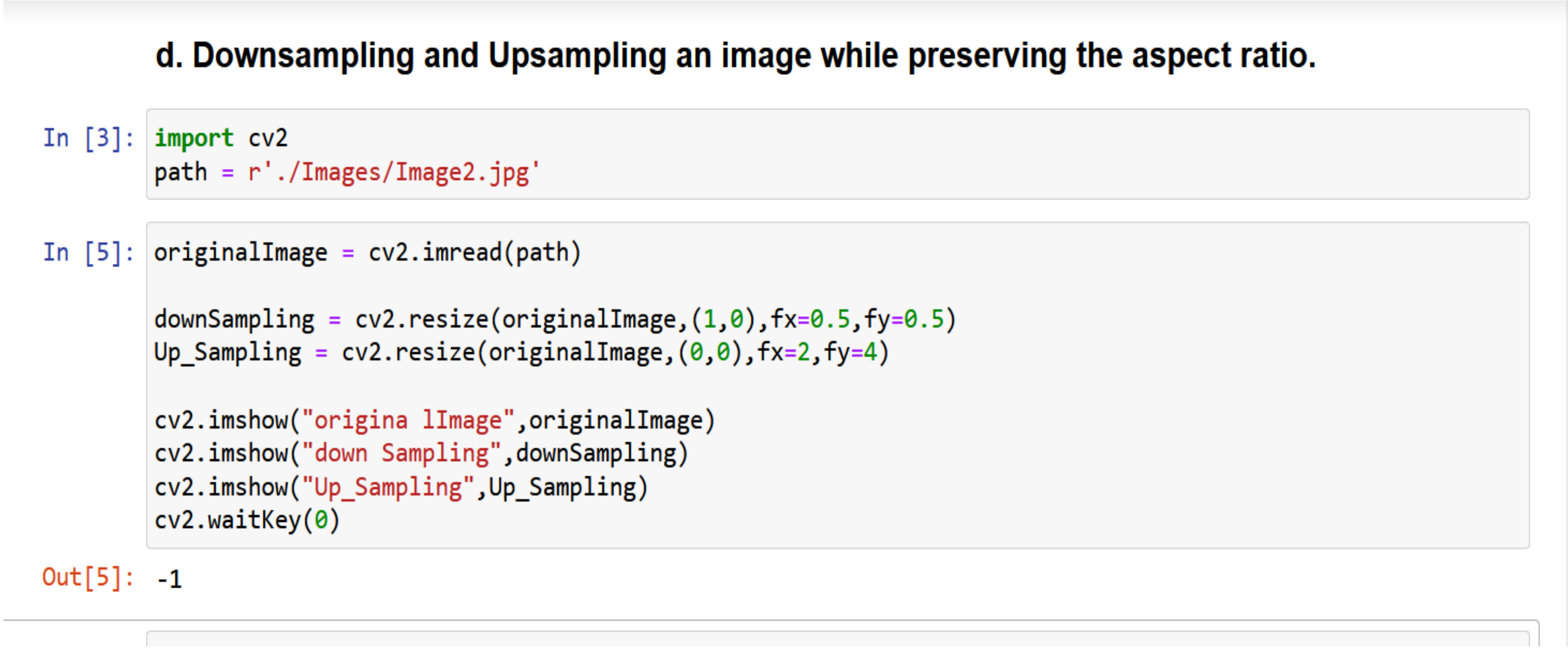
C) Scale the image vertically and horizontally and display all three images.

```
In [26]: import cv2
         path = r'./Images/Image5.jpg'
```

```
In [*]: imageOriginal = cv2.imread(path)

        scaleHieght,scaleWidth = 500,500

        dimtnSize_Horiz = (scaleWidth,imageOriginal.shape[0])
        dimtnSize_Vert = (scaleHieght,imageOriginal.shape[1])

        Horizontal_Scaled = cv2.resize(imageOriginal,dimtnSize_Horiz,interpolation = cv2.INTER_AREA)
        Vertical_Scaled = cv2.resize(imageOriginal,dimtnSize_Vert,interpolation = cv2.INTER_AREA)

        cv2.imshow("ORIGINAL",imageOriginal)
        cv2.imshow("HORIZONTAL SCALED",Horizontal_Scaled)
        cv2.imshow("Vertical Scaled",Vertical_Scaled)
        cv2.waitKey(0)
```

```
In [ ]: |
```

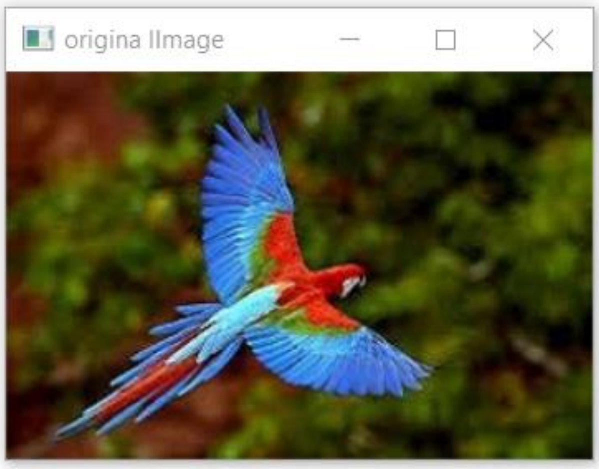


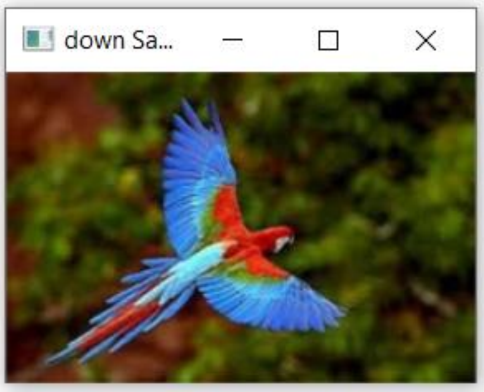


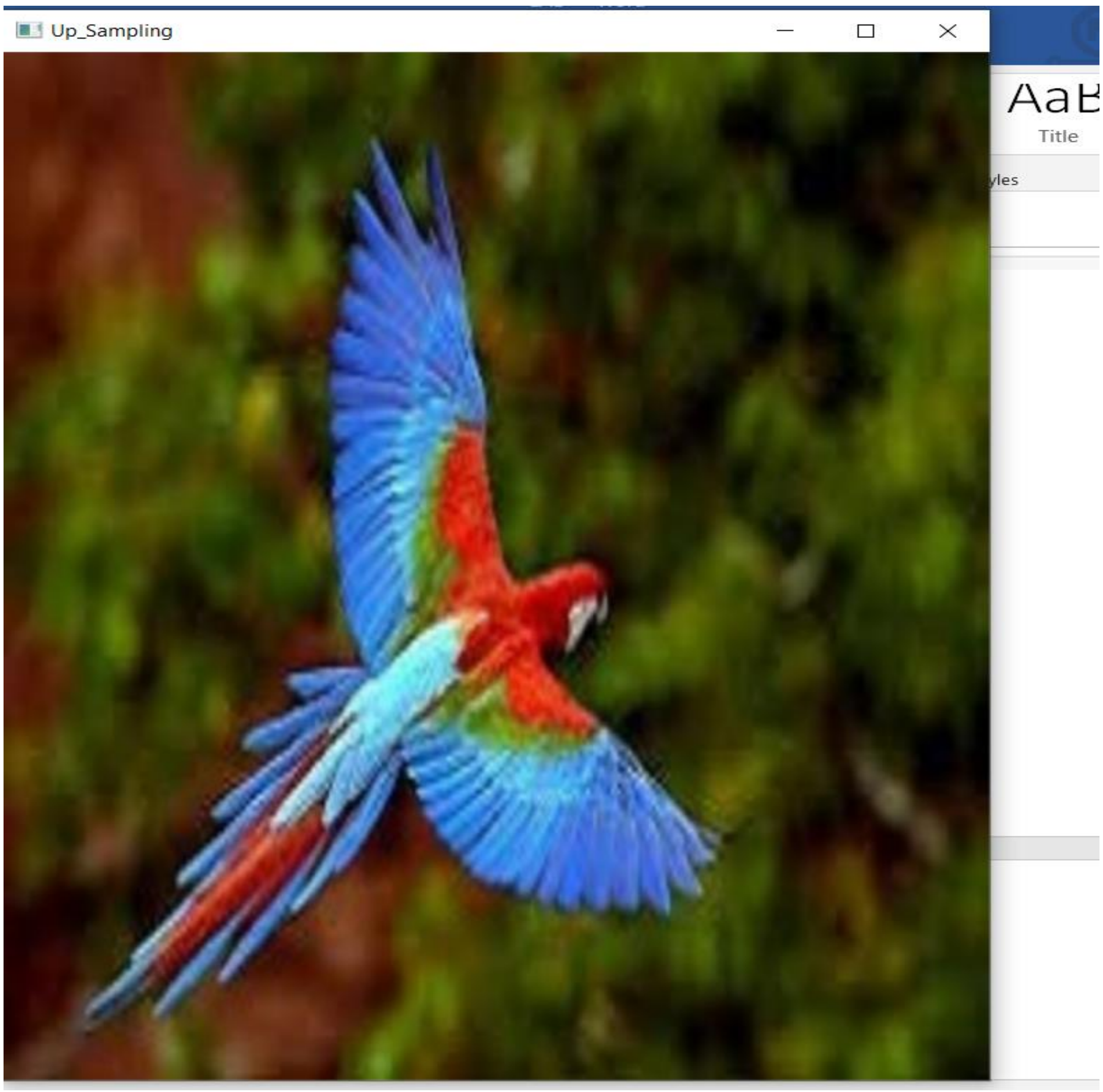
d. Downsampling and Upsampling an image while preserving the aspect ratio.

```
In [3]: import cv2
        path = r'./Images/Image2.jpg'
```

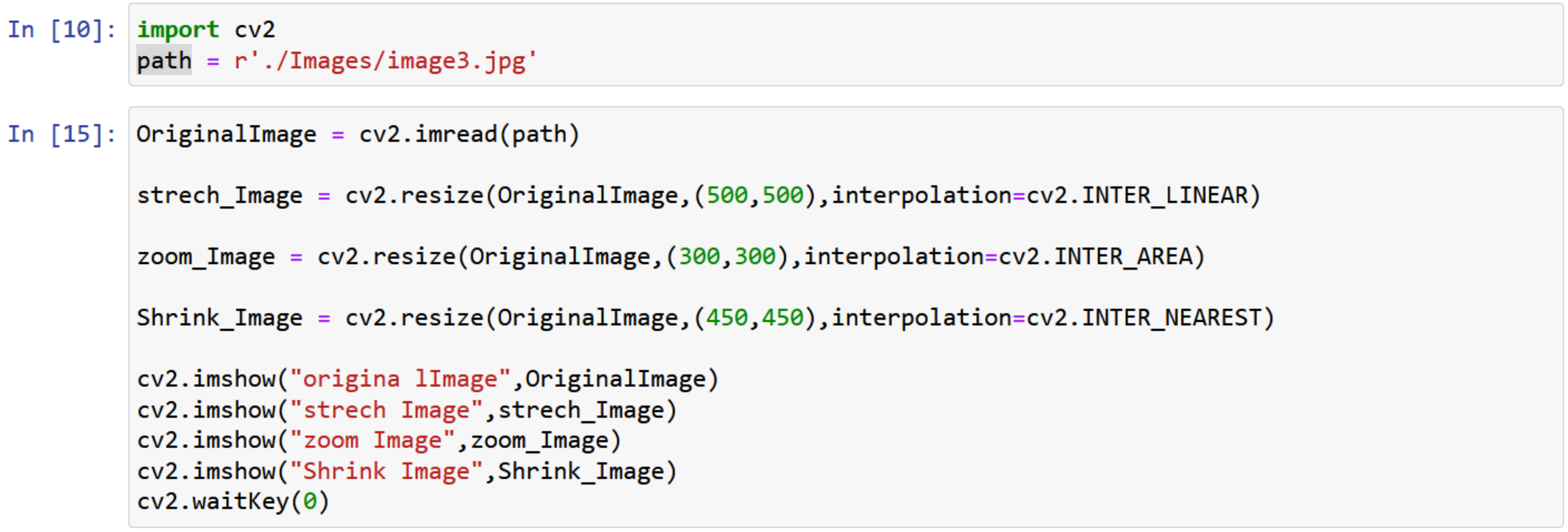
```
In [5]: originalImage = cv2.imread(path)

        downSampling = cv2.resize(originalImage,(1,0),fx=0.5,fy=0.5)
        Up_Sampling = cv2.resize(originalImage,(0,0),fx=2,fy=4)

        cv2.imshow("origina lImage",originalImage)
        cv2.imshow("down Sampling",downSampling)
        cv2.imshow("Up_Sampling",Up_Sampling)
        cv2.waitKey(0)

Out[5]: -1
```





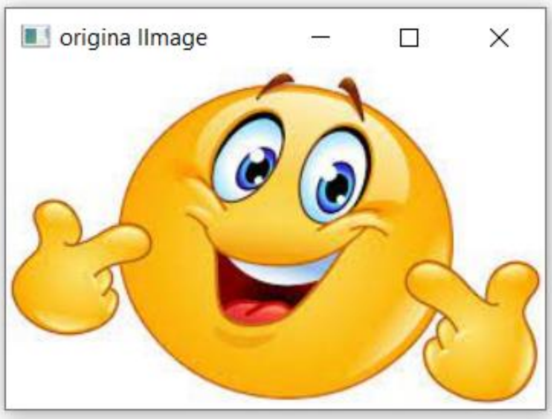


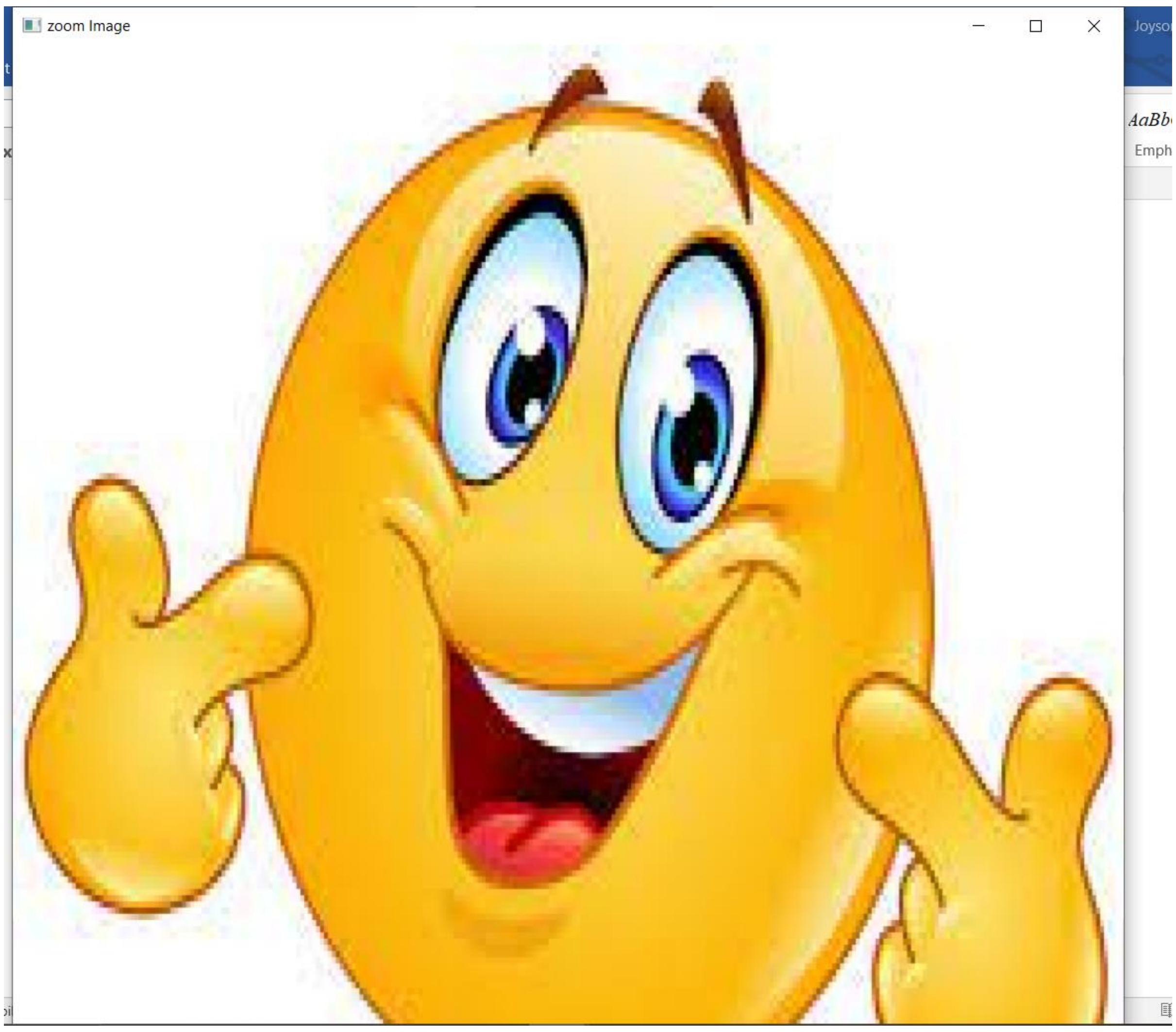
e. Resize the image by stretching, shrinking and zooming the same by using interpolation methods.

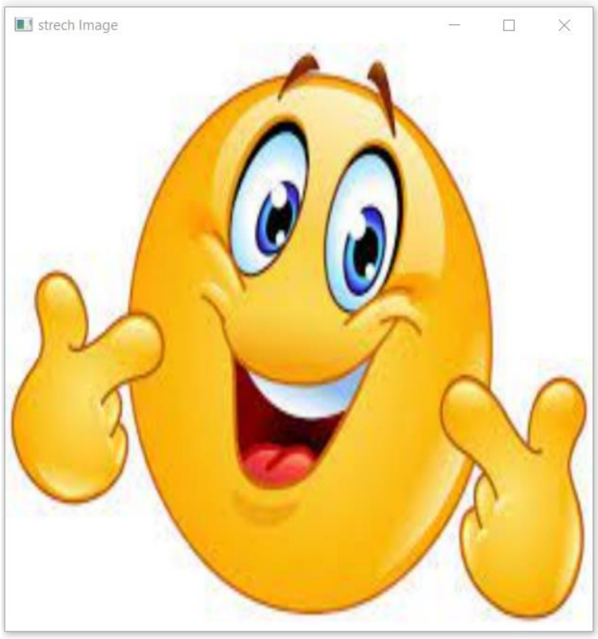


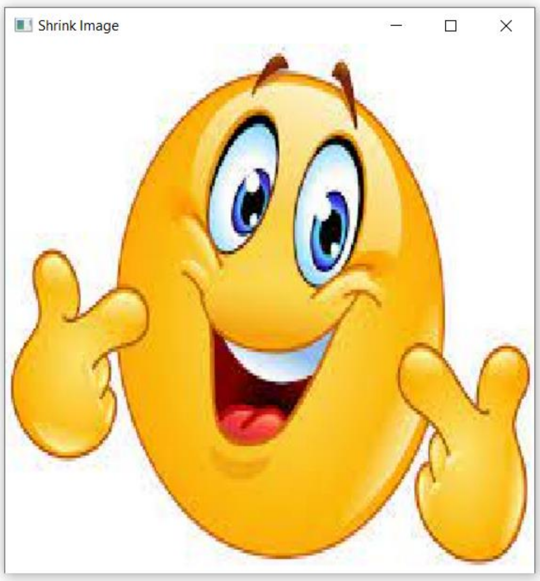
```
In [10]: import cv2
         path = r'./Images/image3.jpg'
```

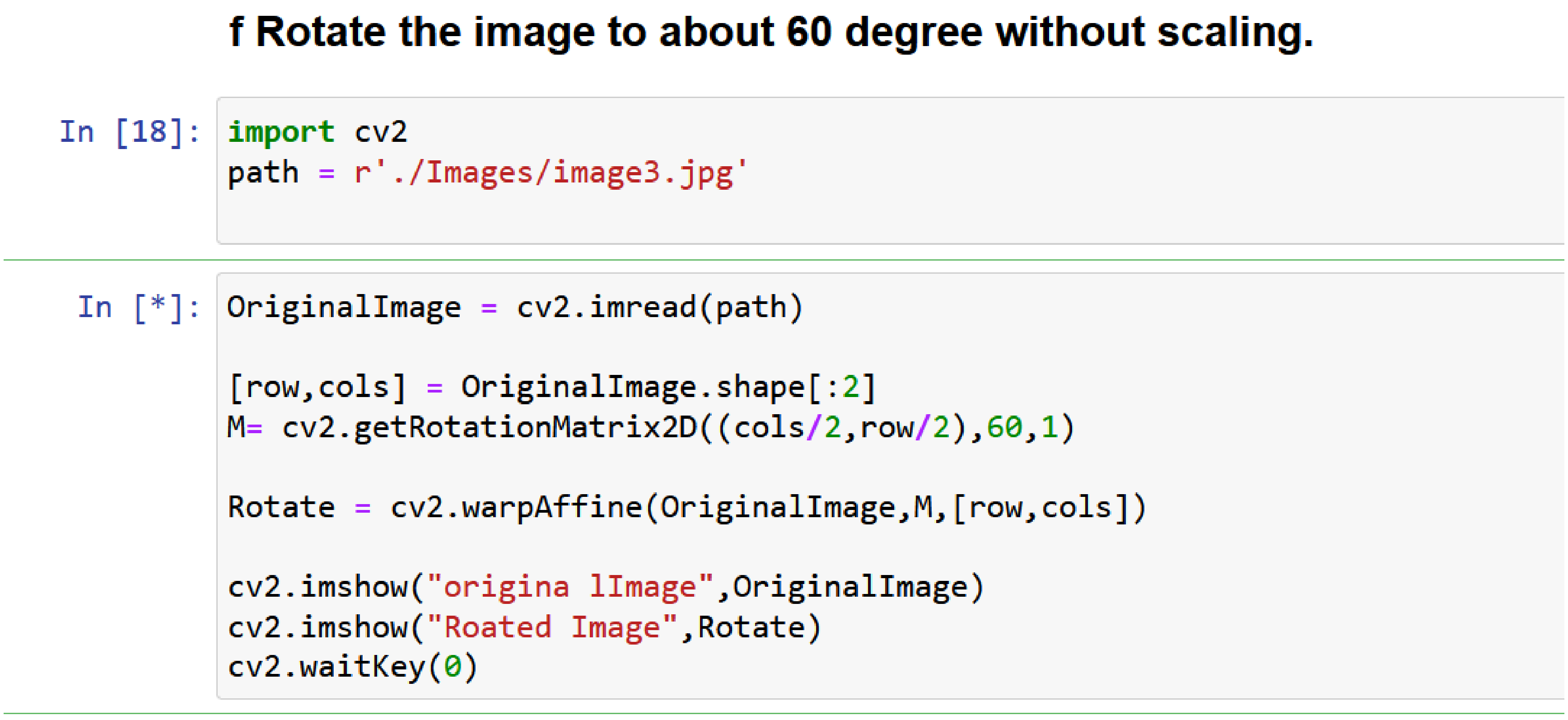
```
In [15]: OriginalImage = cv2.imread(path)

         strech_Image = cv2.resize(OriginalImage,(500,500),interpolation=cv2.INTER_LINEAR)

         zoom_Image = cv2.resize(OriginalImage,(300,300),interpolation=cv2.INTER_AREA)

         Shrink_Image = cv2.resize(OriginalImage,(450,450),interpolation=cv2.INTER_NEAREST)

         cv2.imshow("origina lImage",OriginalImage)
         cv2.imshow("strech Image",strech_Image)
         cv2.imshow("zoom Image",zoom_Image)
         cv2.imshow("Shrink Image",Shrink_Image)
         cv2.waitKey(0)
```







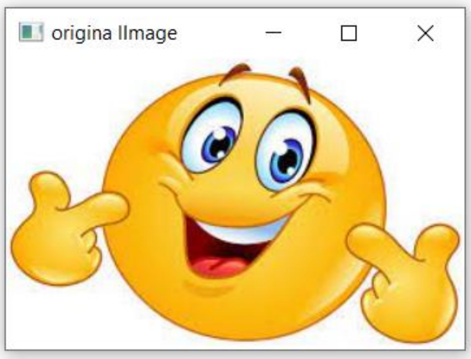


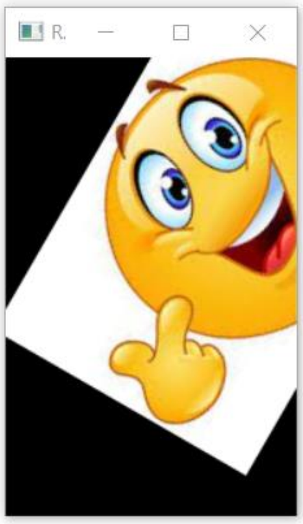


f Rotate the image to about 60 degree without scaling.

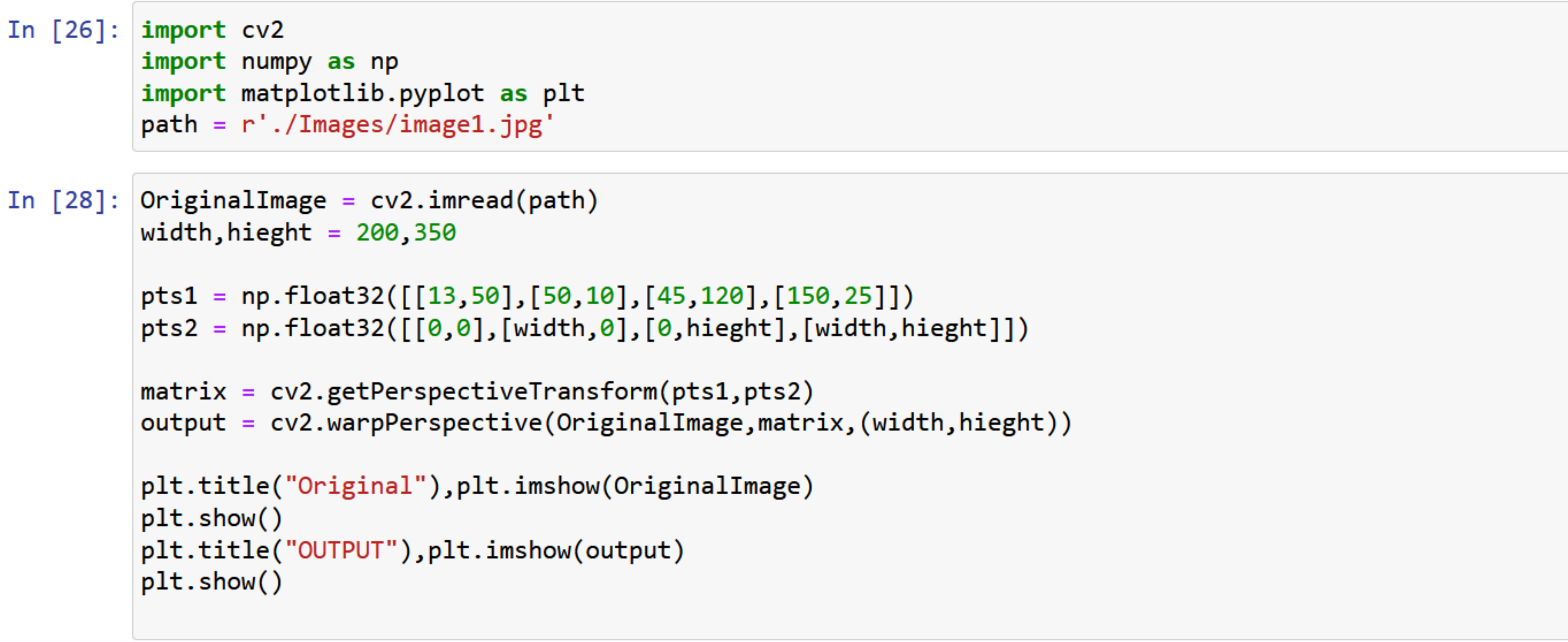
In [18]:
```
import cv2
path = r'./Images/image3.jpg'
```

In [*]:
```
OriginalImage = cv2.imread(path)

[row,cols] = OriginalImage.shape[:2]
M= cv2.getRotationMatrix2D((cols/2,row/2),60,1)

Rotate = cv2.warpAffine(OriginalImage,M,[row,cols])

cv2.imshow("origina lImage",OriginalImage)
cv2.imshow("Roated Image",Rotate)
cv2.waitKey(0)
```



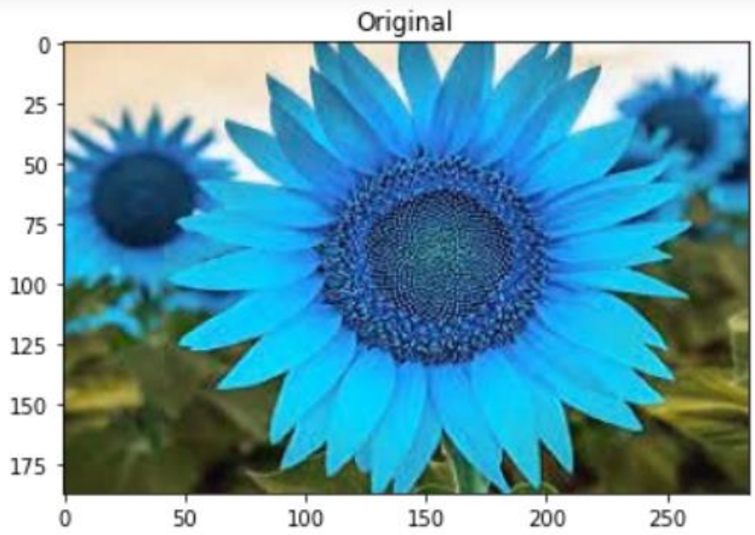


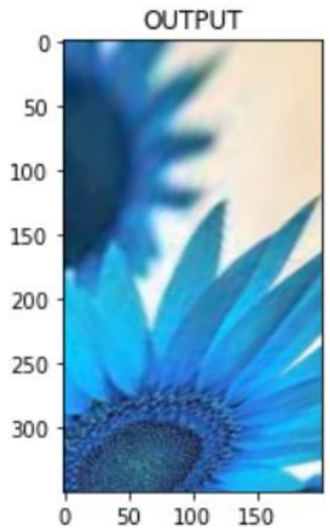
g) Get a better information about an image by changing the perspective of an image

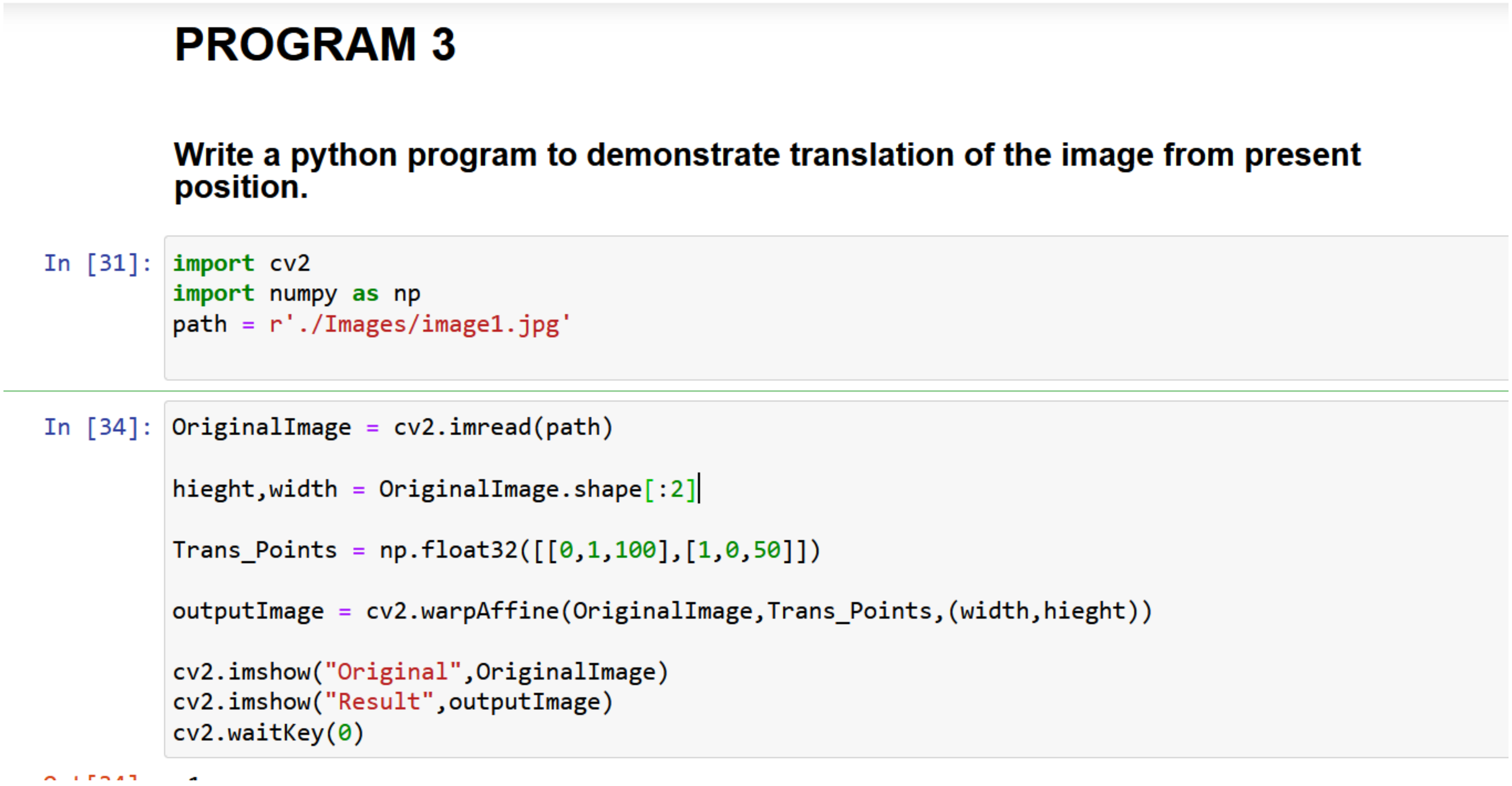


```
In [26]: import cv2
         import numpy as np
         import matplotlib.pyplot as plt
         path = r'./Images/image1.jpg'
```

```
In [28]: OriginalImage = cv2.imread(path)
         width,hieght = 200,350

         pts1 = np.float32([[13,50],[50,10],[45,120],[150,25]])
         pts2 = np.float32([[0,0],[width,0],[0,hieght],[width,hieght]])

         matrix = cv2.getPerspectiveTransform(pts1,pts2)
         output = cv2.warpPerspective(OriginalImage,matrix,(width,hieght))

         plt.title("Original"),plt.imshow(OriginalImage)
         plt.show()
         plt.title("OUTPUT"),plt.imshow(output)
         plt.show()
```





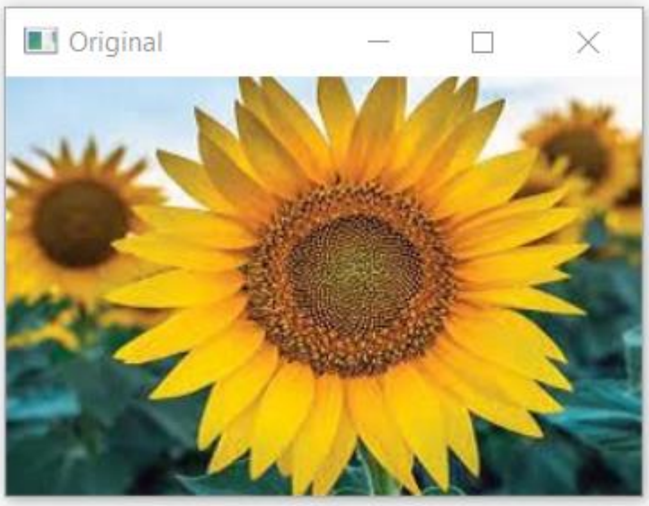


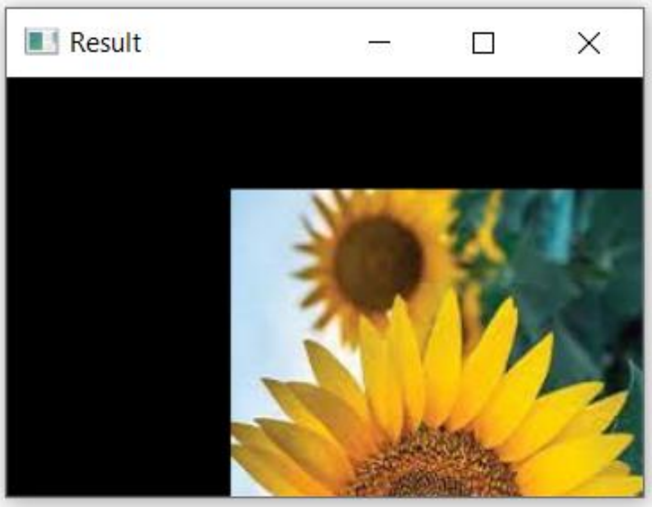
PROGRAM 3

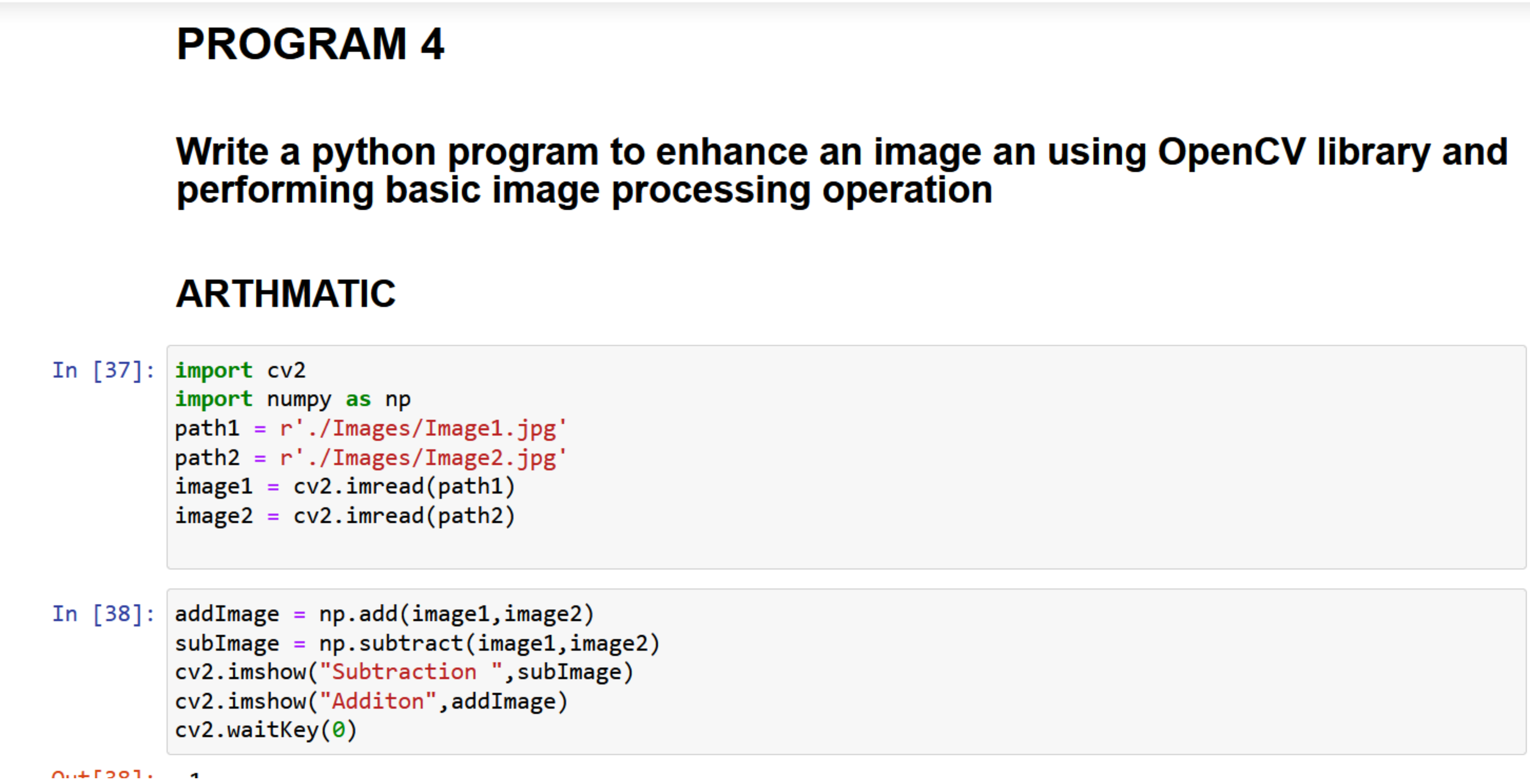
Write a python program to demonstrate translation of the image from present position.

```
In [31]: import cv2
         import numpy as np
         path = r'./Images/image1.jpg'
```

```
In [34]: OriginalImage = cv2.imread(path)

         hieght,width = OriginalImage.shape[:2]

         Trans_Points = np.float32([[0,1,100],[1,0,50]])

         outputImage = cv2.warpAffine(OriginalImage,Trans_Points,(width,hieght))

         cv2.imshow("Original",OriginalImage)
         cv2.imshow("Result",outputImage)
         cv2.waitKey(0)

Out[34]: 1
```







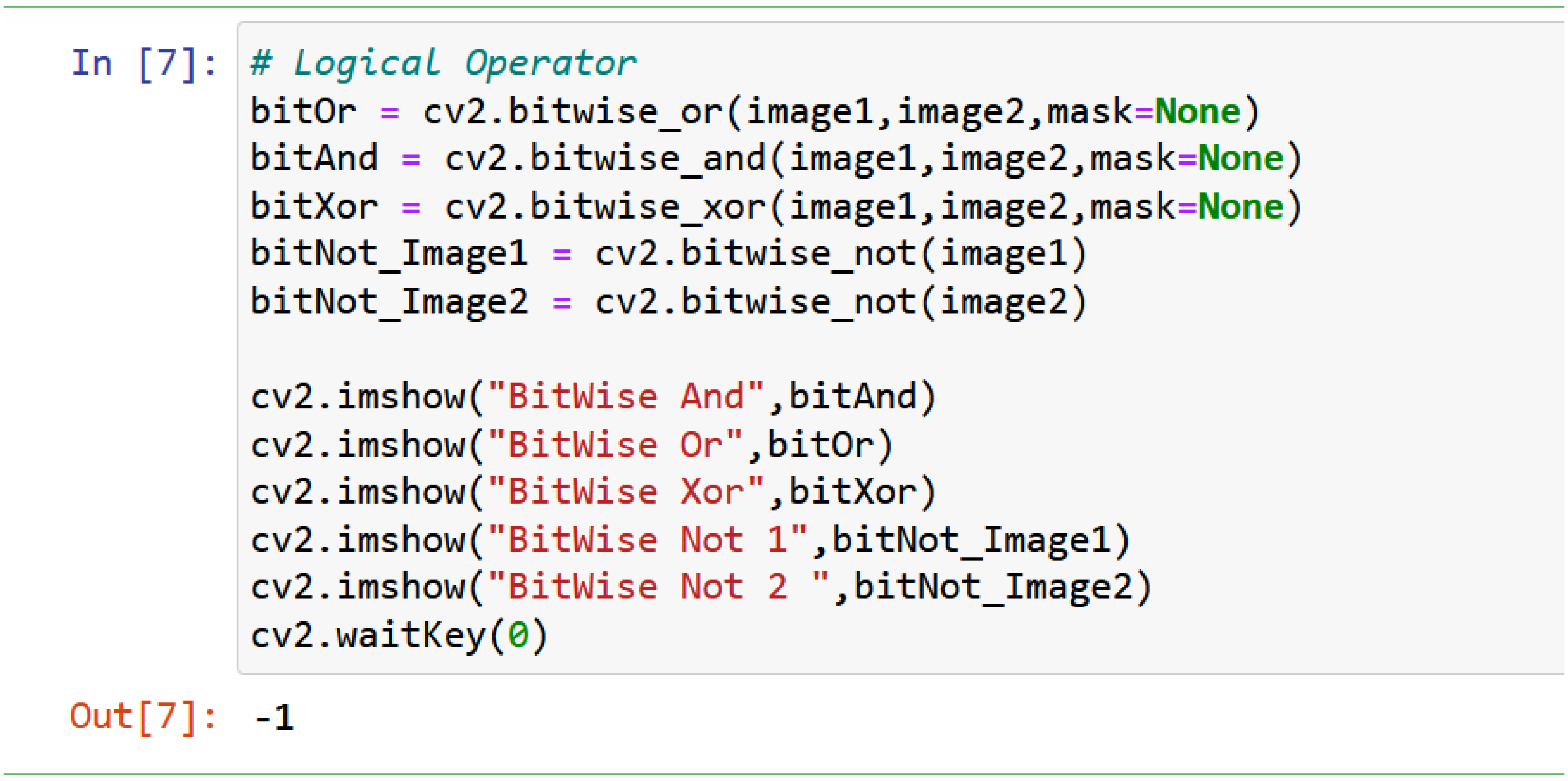
PROGRAM 4

Write a python program to enhance an image an using OpenCV library and performing basic image processing operation

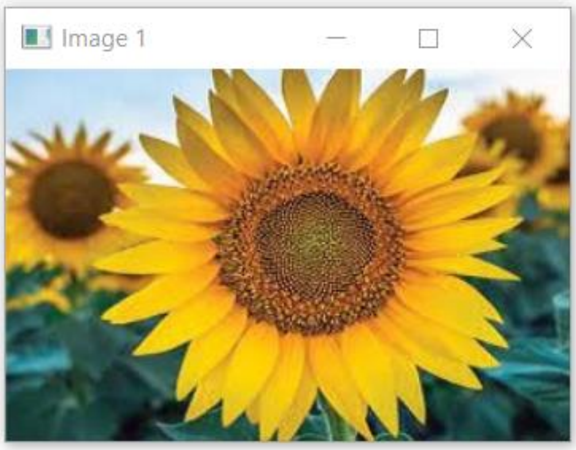
ARTHMATIC

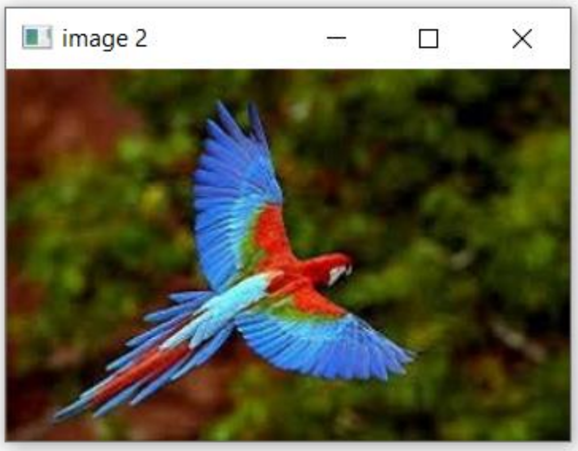
```
In [37]: import cv2
         import numpy as np
         path1 = r'./Images/Image1.jpg'
         path2 = r'./Images/Image2.jpg'
         image1 = cv2.imread(path1)
         image2 = cv2.imread(path2)
```

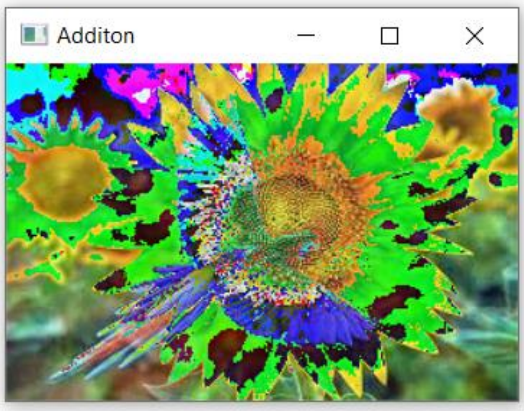
```
In [38]: addImage = np.add(image1,image2)
         subImage = np.subtract(image1,image2)
         cv2.imshow("Subtraction ",subImage)
         cv2.imshow("Additon",addImage)
         cv2.waitKey(0)

Out[38]: 1
```

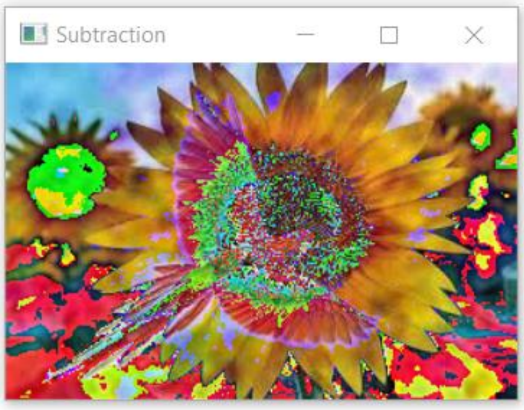


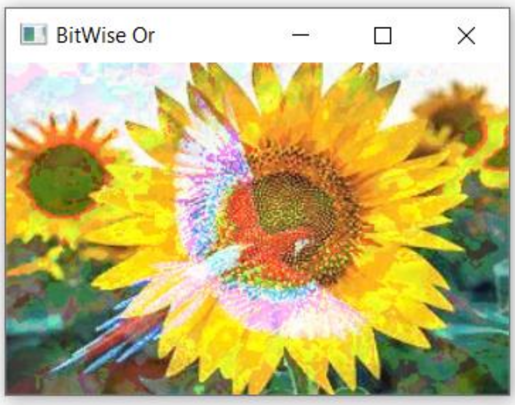
```
In [7]: # Logical Operator
        bitOr = cv2.bitwise_or(image1,image2,mask=None)
        bitAnd = cv2.bitwise_and(image1,image2,mask=None)
        bitXor = cv2.bitwise_xor(image1,image2,mask=None)
        bitNot_Image1 = cv2.bitwise_not(image1)
        bitNot_Image2 = cv2.bitwise_not(image2)

        cv2.imshow("BitWise And",bitAnd)
        cv2.imshow("BitWise Or",bitOr)
        cv2.imshow("BitWise Xor",bitXor)
        cv2.imshow("BitWise Not 1",bitNot_Image1)
        cv2.imshow("BitWise Not 2 ",bitNot_Image2)
        cv2.waitKey(0)

Out[7]: -1
```

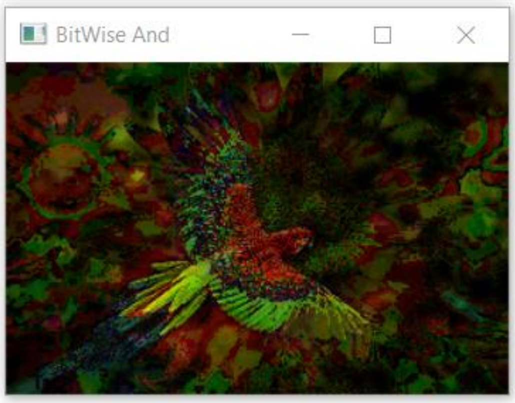


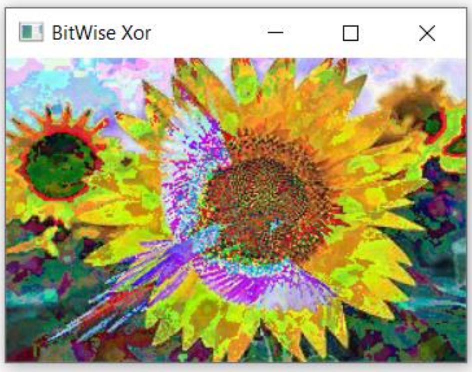


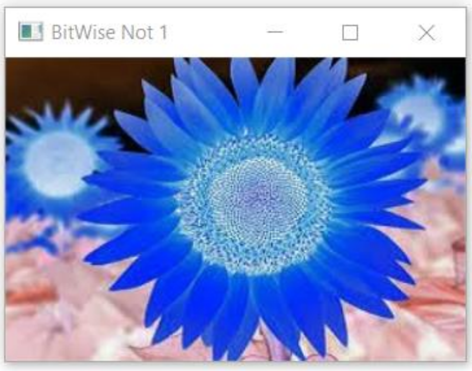


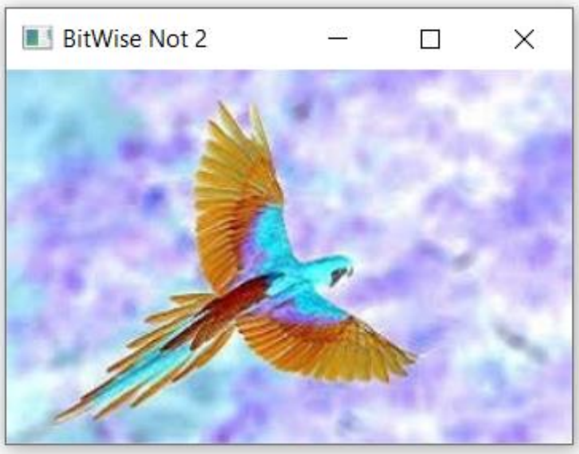











BitWise Not 2

PROGRAM 5

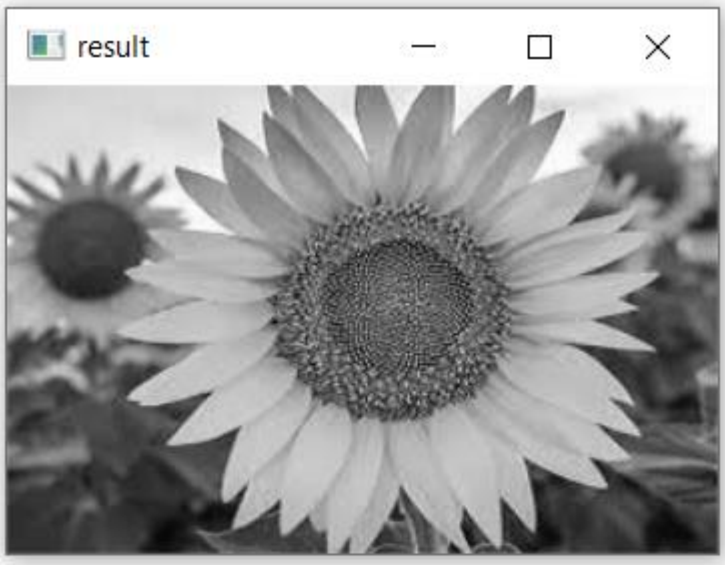
Write a python program to smooth(remove noise) an image using low pass filtering methods in frequency domain to perform the following

a) Read a gray scale image

```
In [1]: import cv2
        path = r'./Images/Image1.jpg'
```

```
In [2]: image = cv2.imread(path,0)

        cv2.imshow("Image",image)
        cv2.waitKey(0)
```

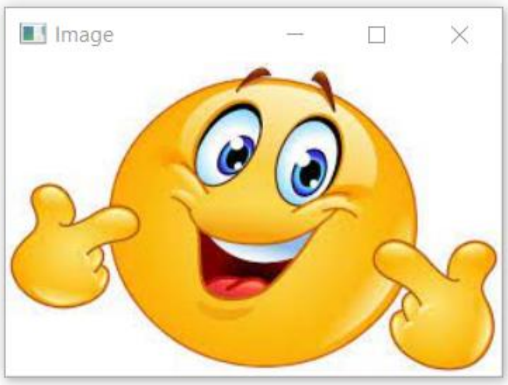
```
Out[2]: -1
```

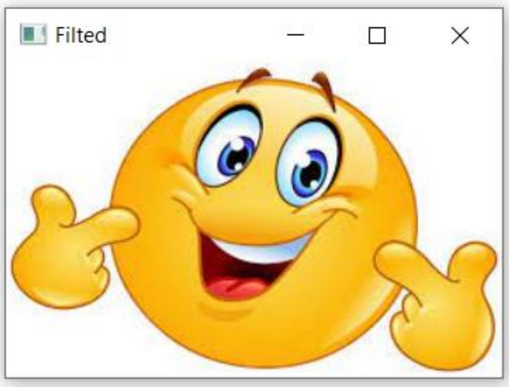


b) Apply Custom 2D-Convolution 3 X 3 Kernel, combine two images and display.

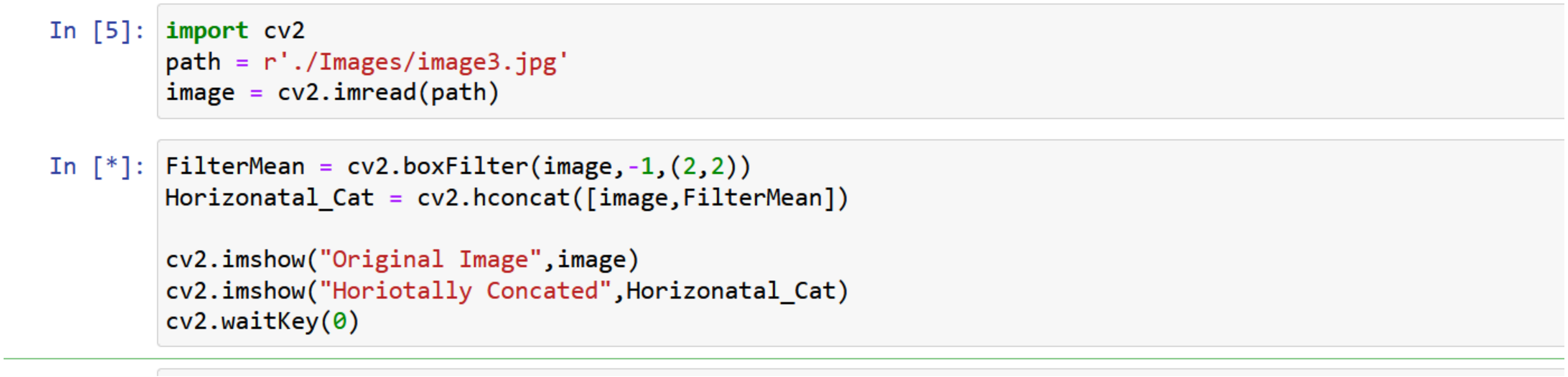
```
In [3]: import cv2
        import numpy as np
```

```
In [*]: path = r'./Images/image3.jpg'
        image = cv2.imread(path)
        kernel1 = np.array([[0,0,0],[0,1,0],[0,0,0]])

        identity = cv2.filter2D(src=image,ddepth=-1,kernel=kernel1)
        cv2.imshow("Image",image)
        cv2.imshow("Filted",identity)
        cv2.waitKey(0)
```



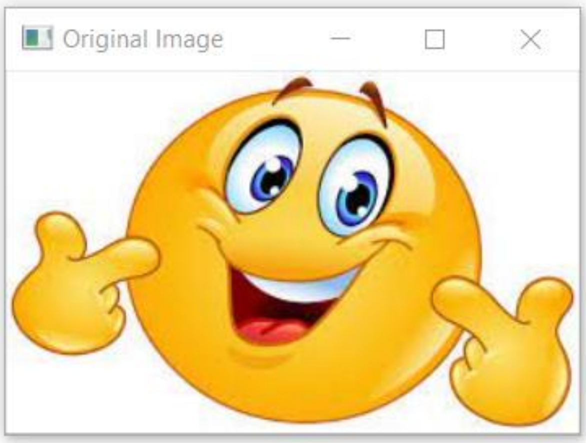


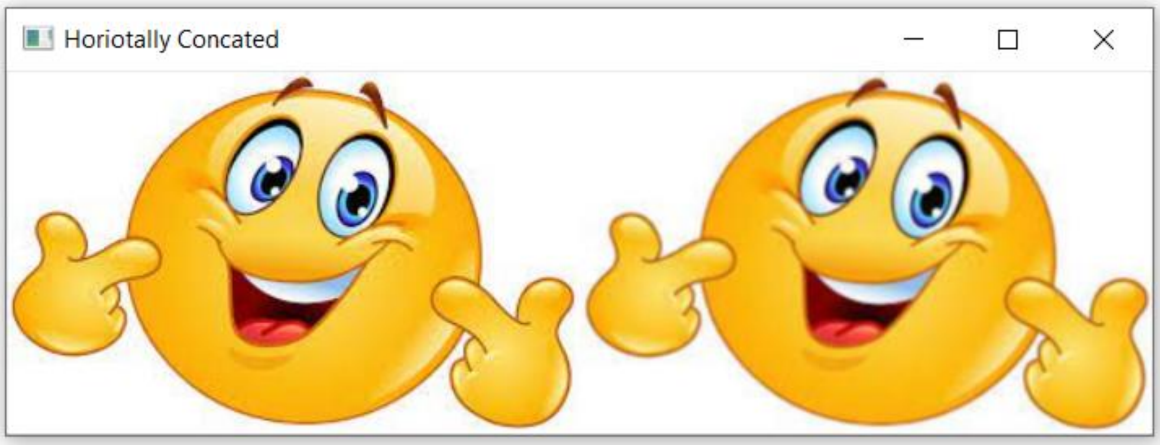
c) Apply Mean/Box/Average Filter, concatenate two images horizontally and display.



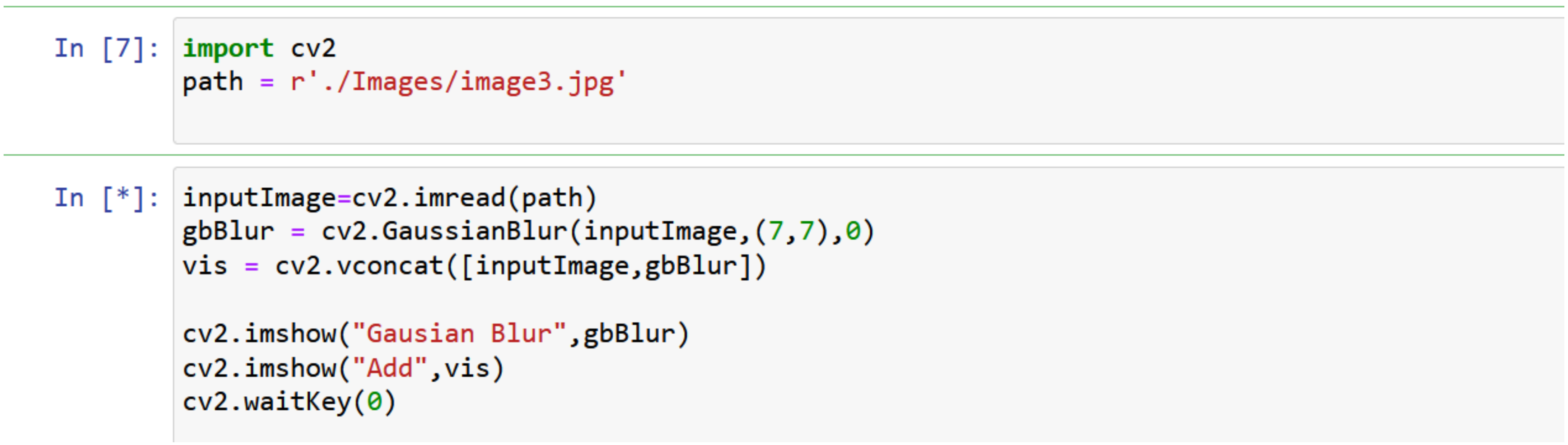
```
In [5]: import cv2
        path = r'./Images/image3.jpg'
        image = cv2.imread(path)
```

```
In [*]: FilterMean = cv2.boxFilter(image,-1,(2,2))
        Horizonatal_Cat = cv2.hconcat([image,FilterMean])

        cv2.imshow("Original Image",image)
        cv2.imshow("Horiotally Concated",Horizonatal_Cat)
        cv2.waitKey(0)
```



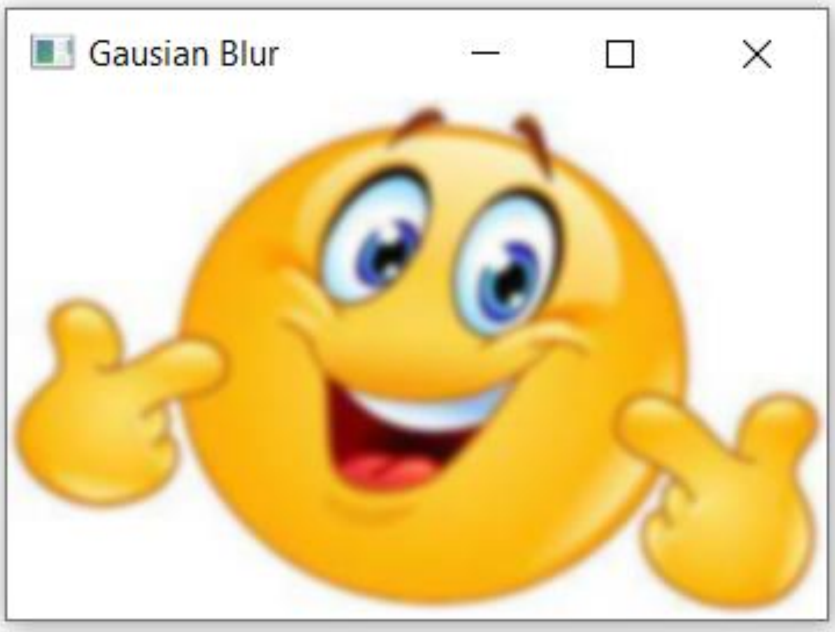


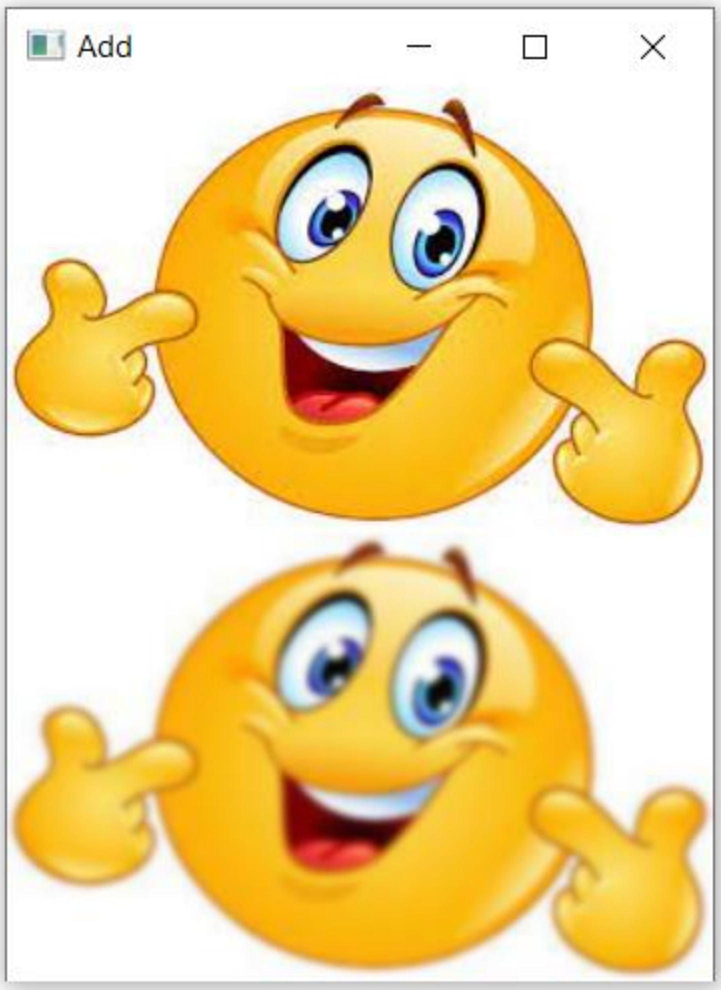
d) Apply Gaussian blurring, concatenate two images vertically and display.



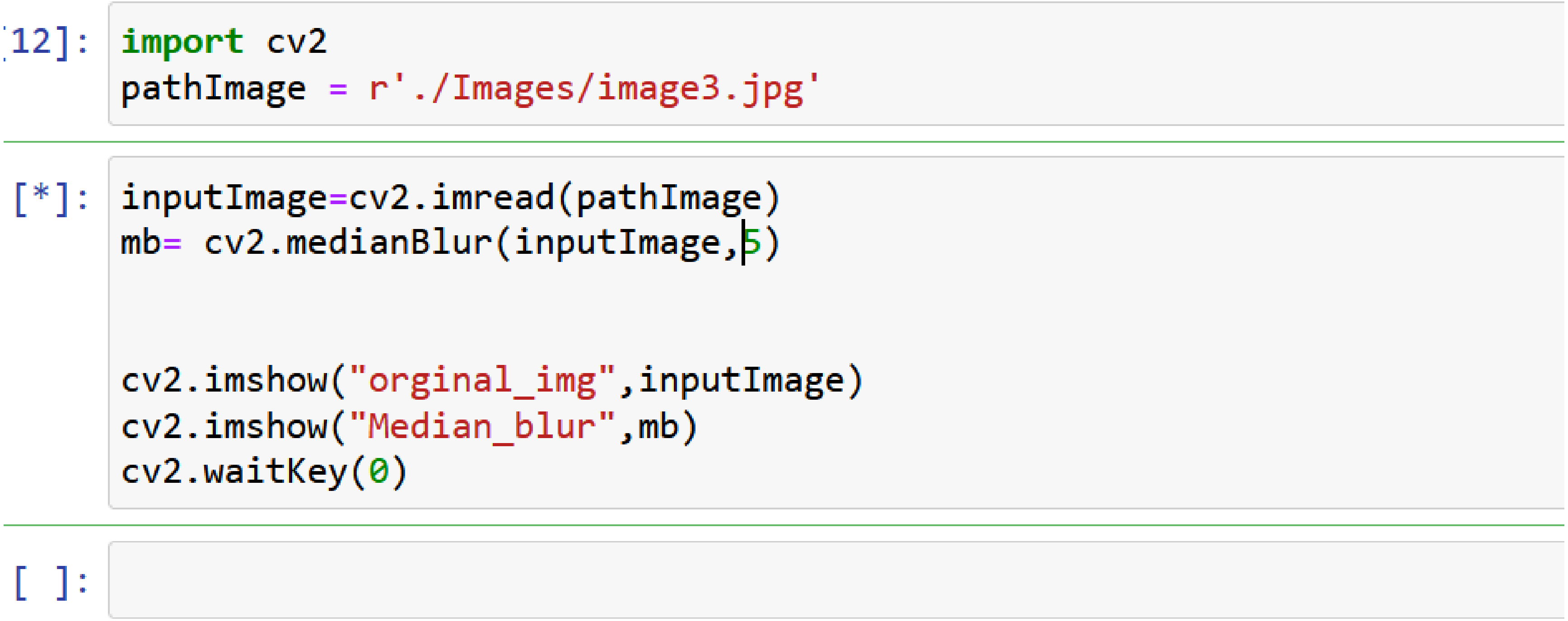
```
In [7]: import cv2
        path = r'./Images/image3.jpg'
```

```
In [*]: inputImage=cv2.imread(path)
        gbBlur = cv2.GaussianBlur(inputImage,(7,7),0)
        vis = cv2.vconcat([inputImage,gbBlur])

        cv2.imshow("Gausian Blur",gbBlur)
        cv2.imshow("Add",vis)
        cv2.waitKey(0)
```





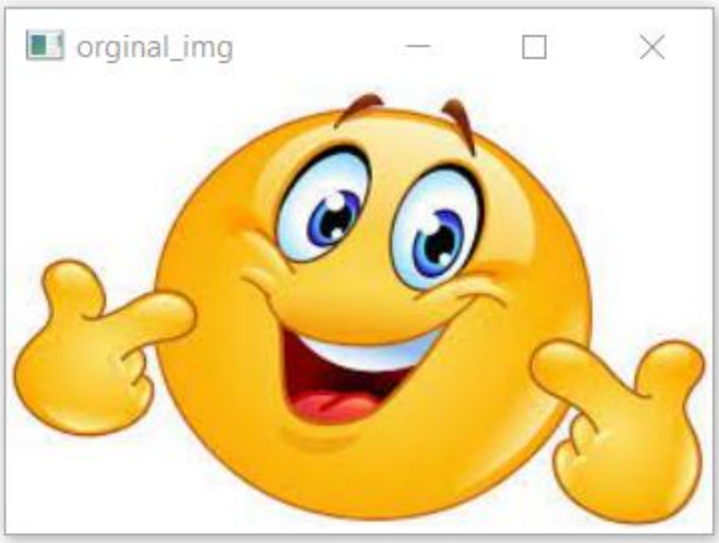
e) Apply Median blurring and display the image

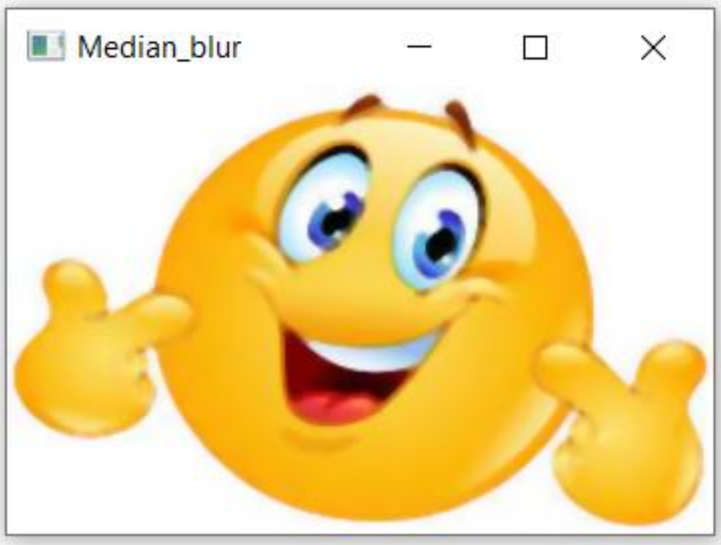


```
12]: import cv2
     pathImage = r'./Images/image3.jpg'
```

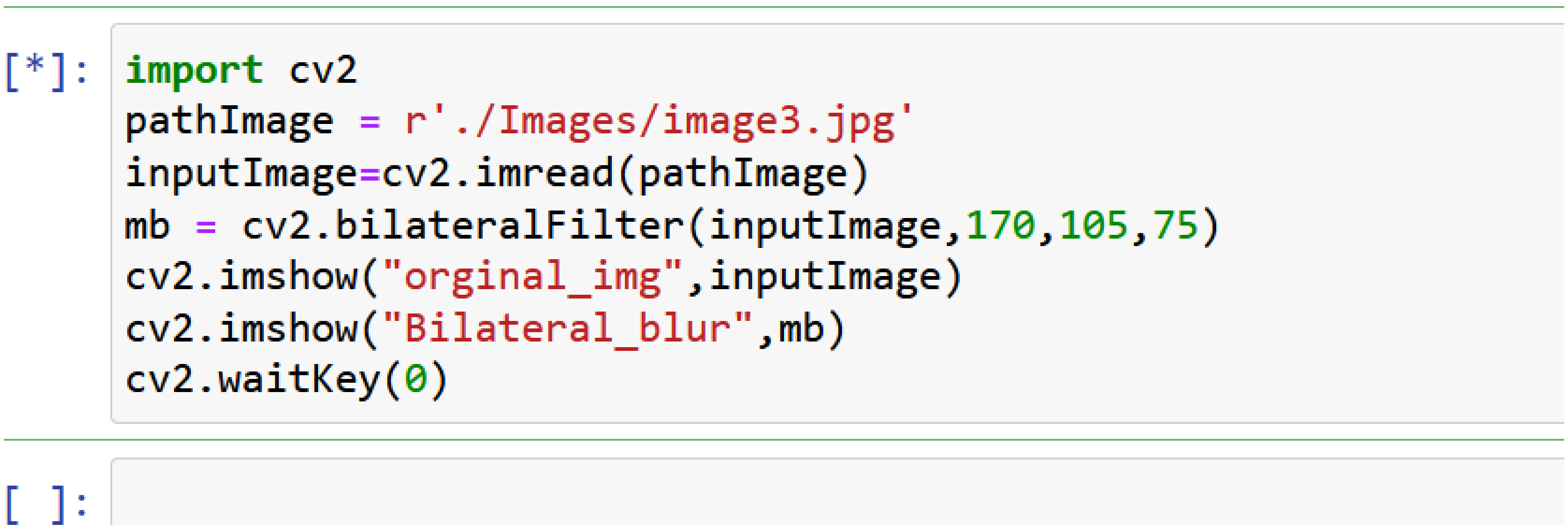
```
[*]: inputImage=cv2.imread(pathImage)
     mb= cv2.medianBlur(inputImage,5)


     cv2.imshow("orginal_img",inputImage)
     cv2.imshow("Median_blur",mb)
     cv2.waitKey(0)
```

```
[ ]:
```

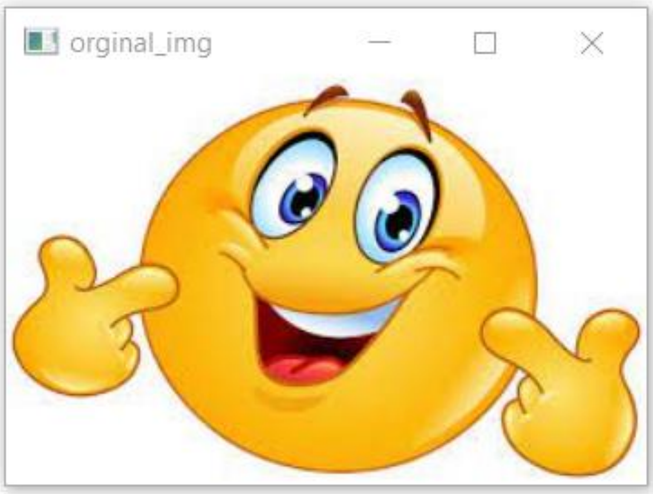


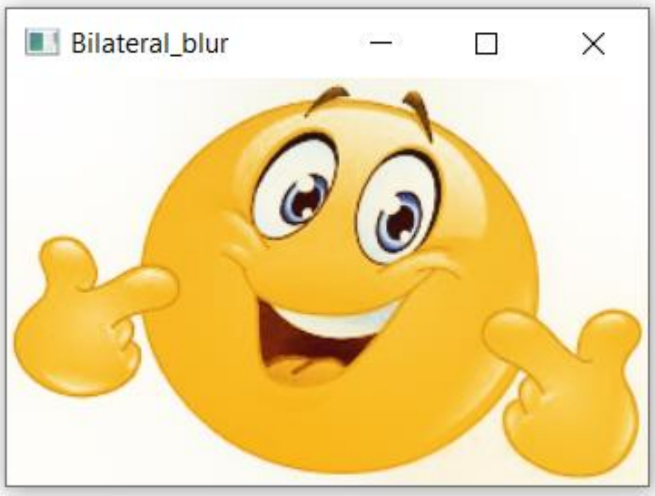


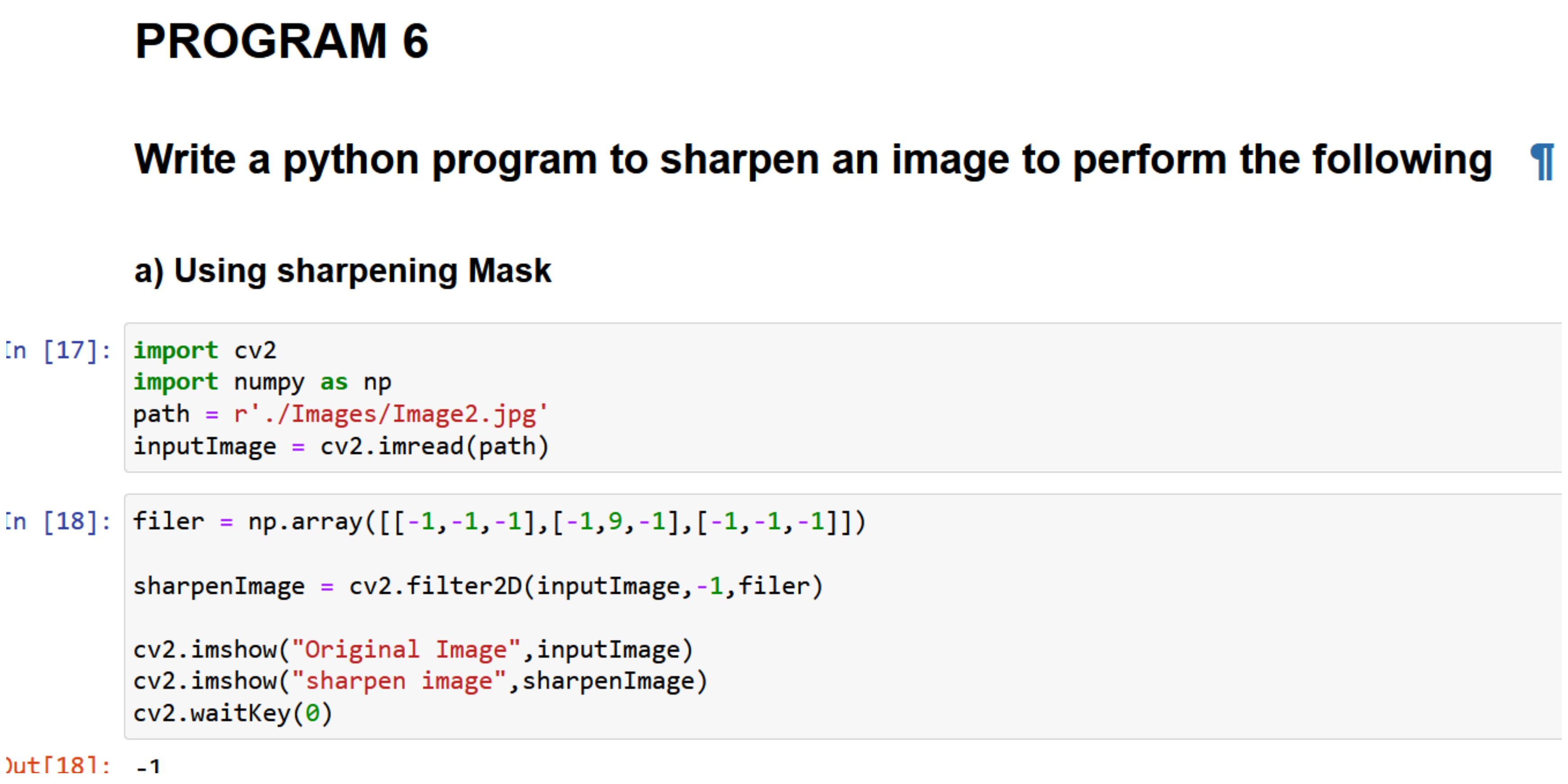
f) Apply Bilateral blurring and display the image



```
[*]: import cv2
     pathImage = r'./Images/image3.jpg'
     inputImage=cv2.imread(pathImage)
     mb = cv2.bilateralFilter(inputImage,170,105,75)
     cv2.imshow("orginal_img",inputImage)
     cv2.imshow("Bilateral_blur",mb)
     cv2.waitKey(0)

[ ]:
```







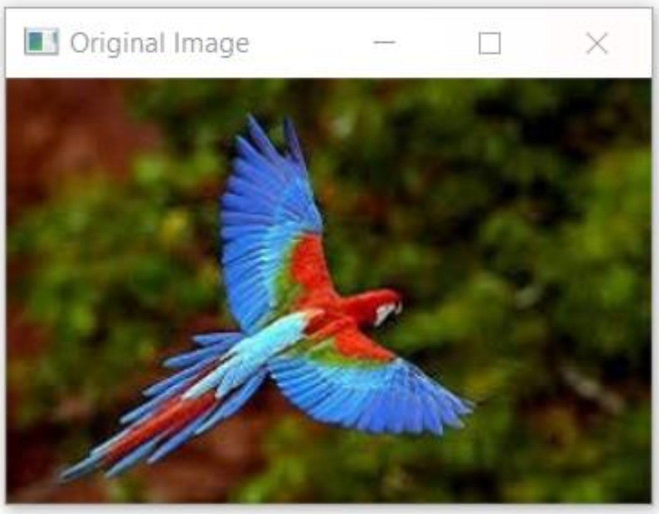
PROGRAM 6

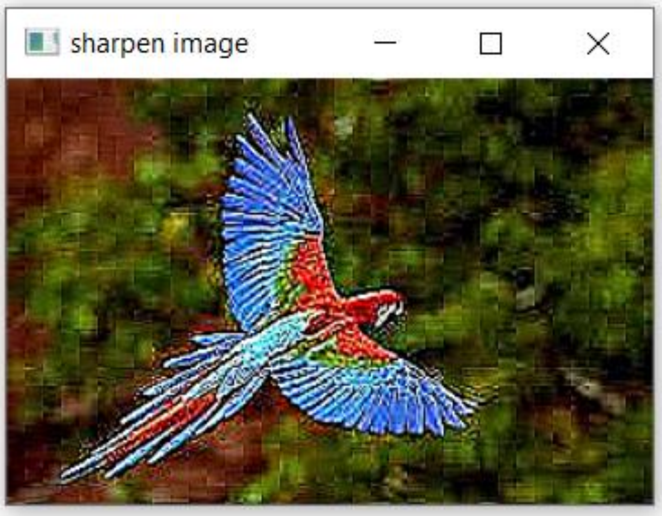
Write a python program to sharpen an image to perform the following ¶

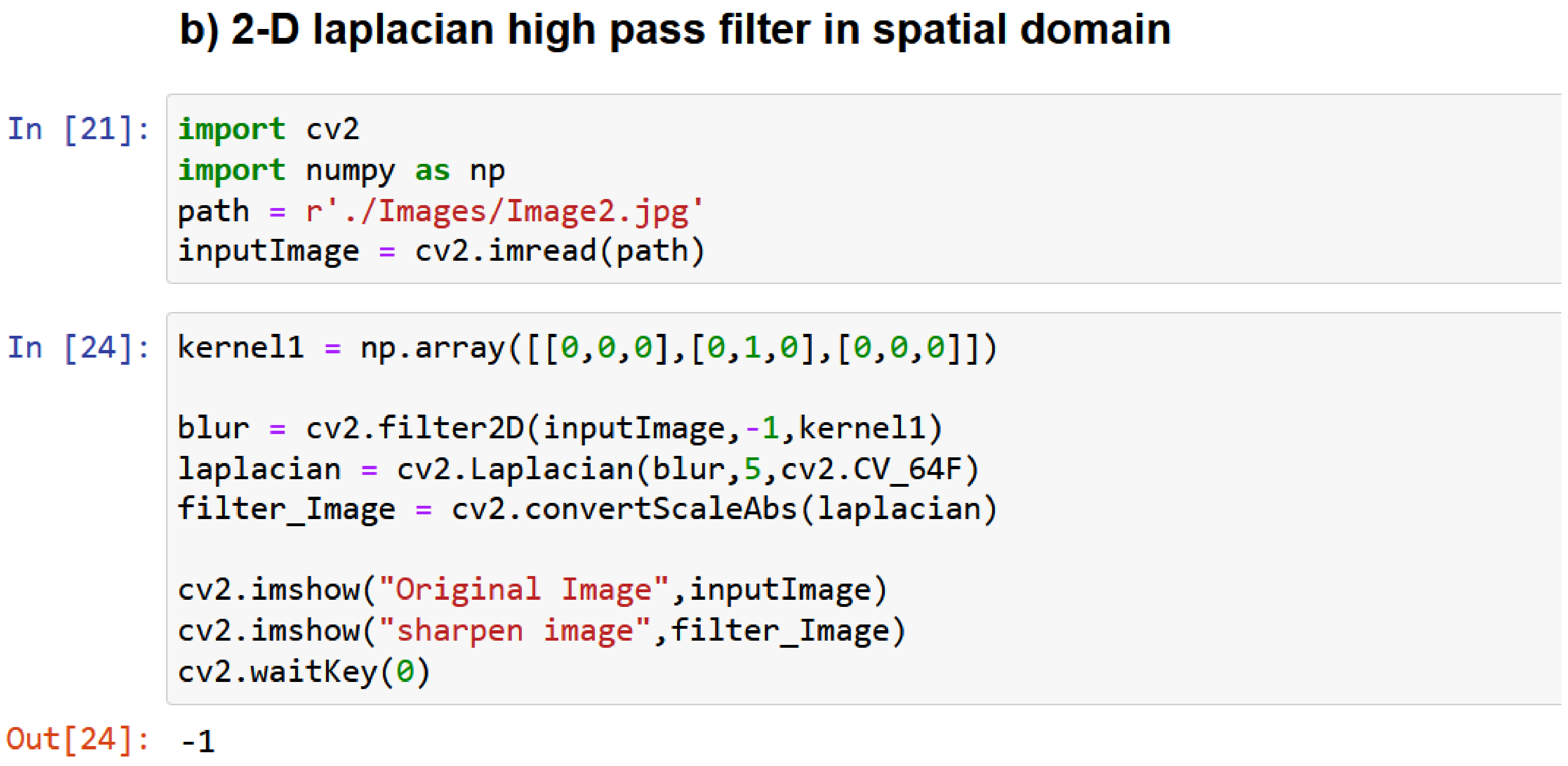
a) Using sharpening Mask

```
In [17]: import cv2
         import numpy as np
         path = r'./Images/Image2.jpg'
         inputImage = cv2.imread(path)
```

```
In [18]: filer = np.array([[-1,-1,-1],[-1,9,-1],[-1,-1,-1]])

         sharpenImage = cv2.filter2D(inputImage,-1,filer)

         cv2.imshow("Original Image",inputImage)
         cv2.imshow("sharpen image",sharpenImage)
         cv2.waitKey(0)

Out[18]: -1
```



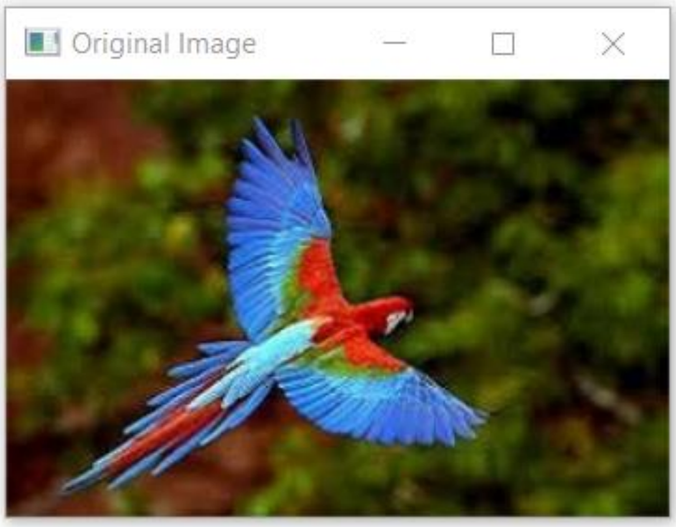


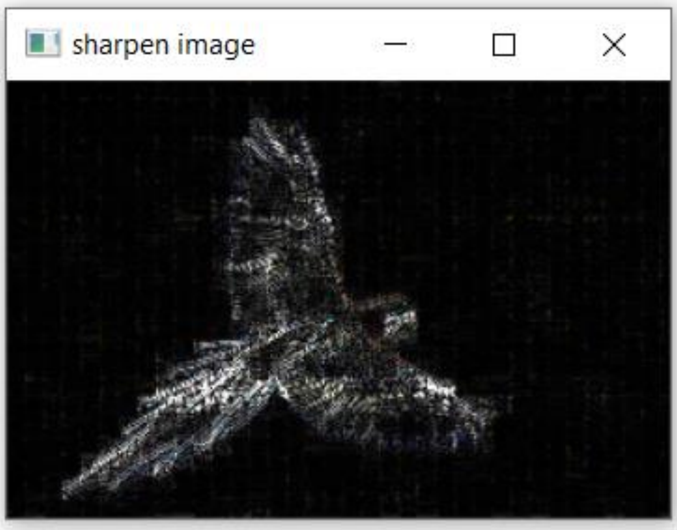


b) 2-D laplacian high pass filter in spatial domain

```
In [21]: import cv2
         import numpy as np
         path = r'./Images/Image2.jpg'
         inputImage = cv2.imread(path)
```

```
In [24]: kernel1 = np.array([[0,0,0],[0,1,0],[0,0,0]])

         blur = cv2.filter2D(inputImage,-1,kernel1)
         laplacian = cv2.Laplacian(blur,5,cv2.CV_64F)
         filter_Image = cv2.convertScaleAbs(laplacian)

         cv2.imshow("Original Image",inputImage)
         cv2.imshow("sharpen image",filter_Image)
         cv2.waitKey(0)

Out[24]: -1
```



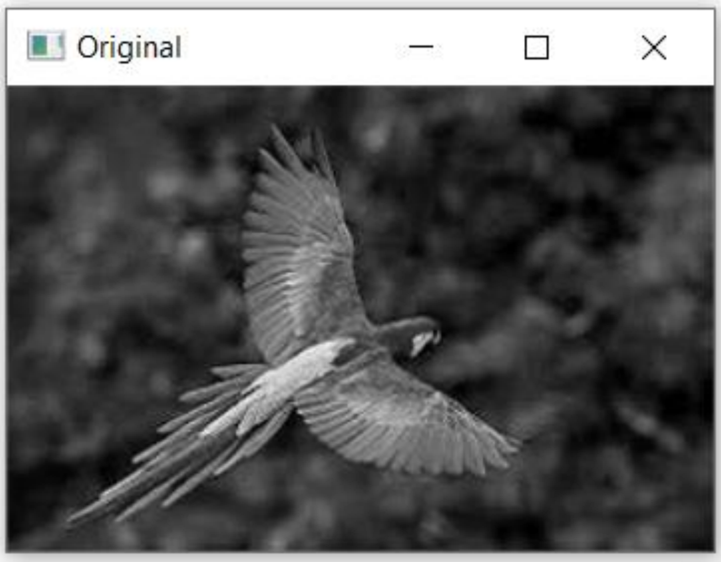


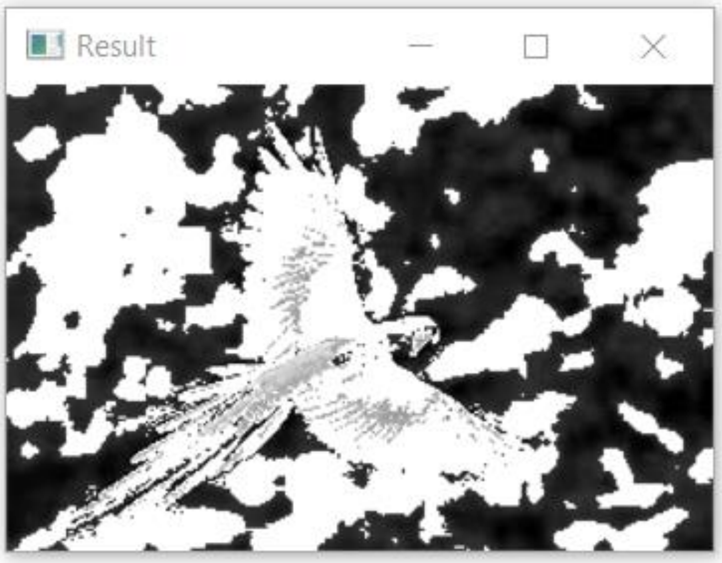
7) Write a python program for gray level slicing with or without background.

with background.

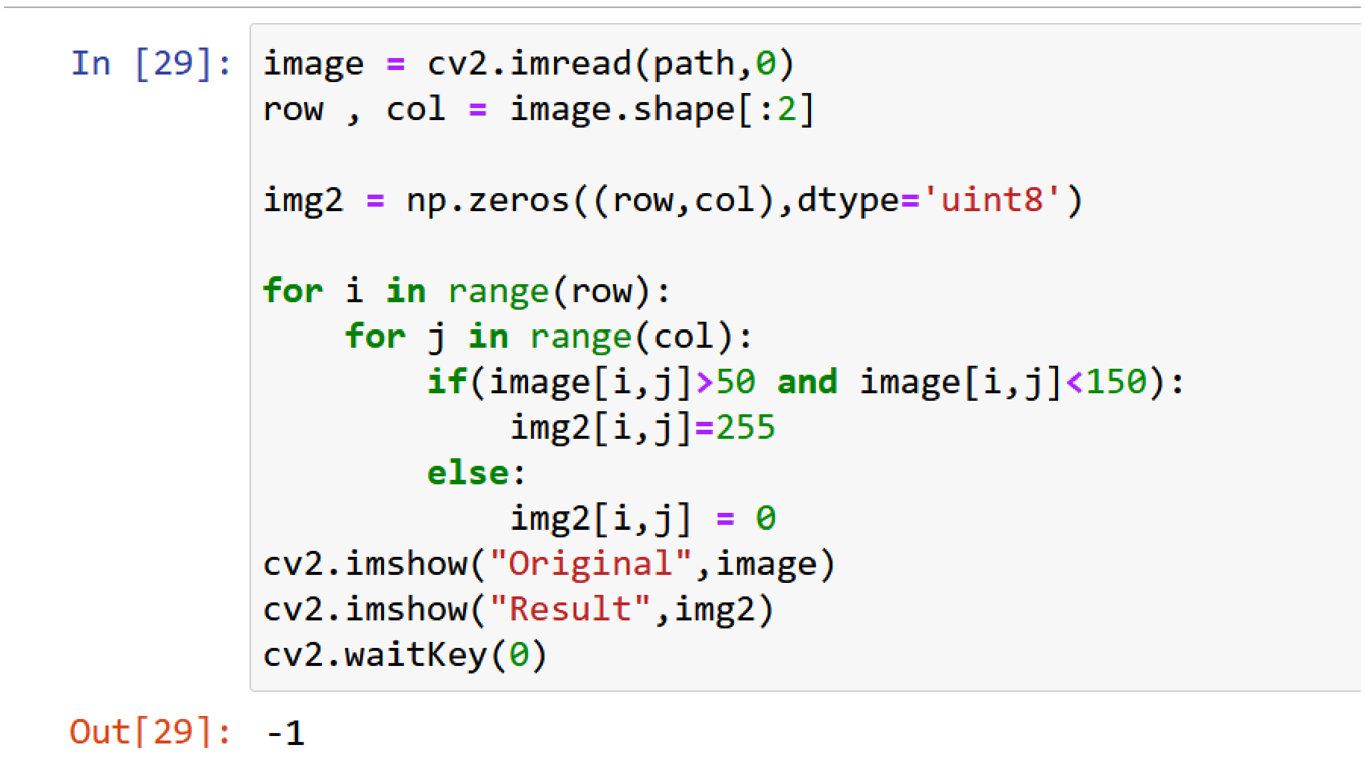
```
In [26]: import numpy as np
         import cv2
         path = r'./Images/Image2.jpg'
```

```
In [28]: image = cv2.imread(path,0)
         row , col = image.shape[:2]

         img1 = np.zeros((row,col),dtype='uint8')

         for i in range(row):
             for j in range(col):
                 if(image[i,j]>50 and image[i,j]<150):
                     img1[i,j]=255
                 else:
                     img1[i,j] = image[i,j]
         cv2.imshow("Original",image)
         cv2.imshow("Result",img1)
         cv2.waitKey(0)
```

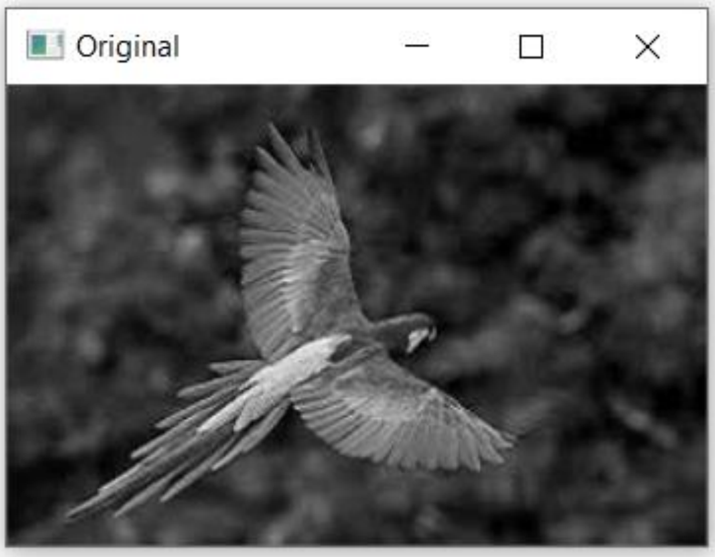


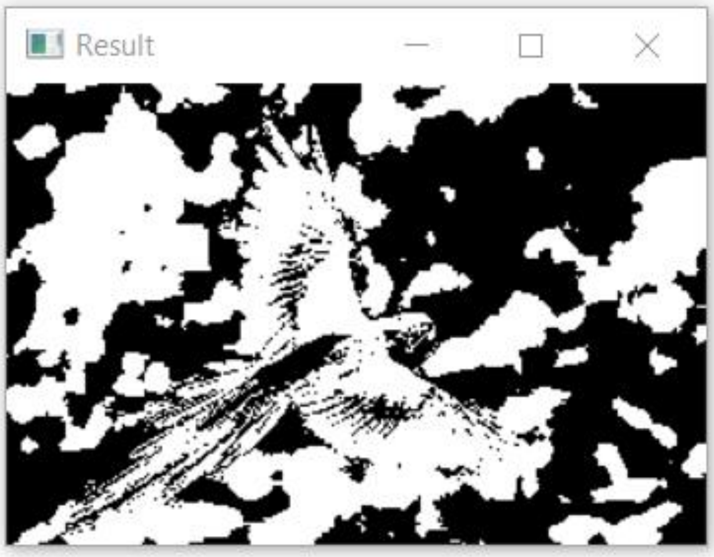


WITHOUT BACK-GROUND

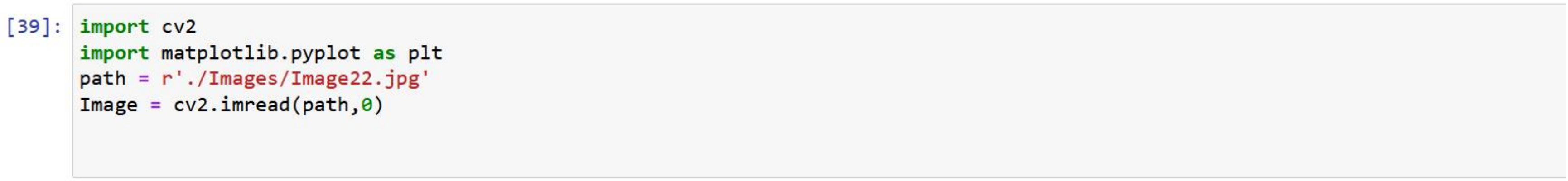


```
In [29]: image = cv2.imread(path,0)
         row , col = image.shape[:2]

         img2 = np.zeros((row,col),dtype='uint8')

         for i in range(row):
             for j in range(col):
                 if(image[i,j]>50 and image[i,j]<150):
                     img2[i,j]=255
                 else:
                     img2[i,j] = 0
         cv2.imshow("Original",image)
         cv2.imshow("Result",img2)
         cv2.waitKey(0)

Out[29]: -1
```

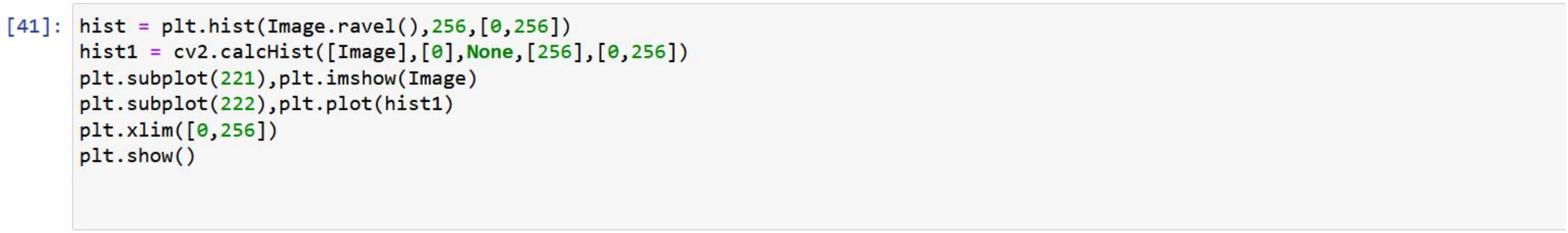




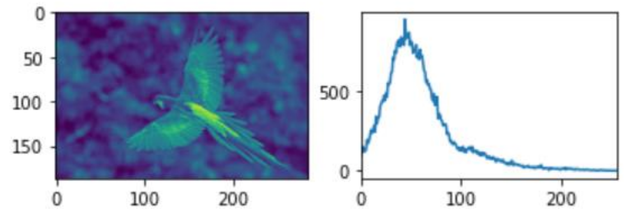
8) Analyze a colour and gray scale image using histogram and interpret the same and display both histogram.



```
[39]: import cv2
      import matplotlib.pyplot as plt
      path = r'./Images/Image22.jpg'
      Image = cv2.imread(path,0)
```

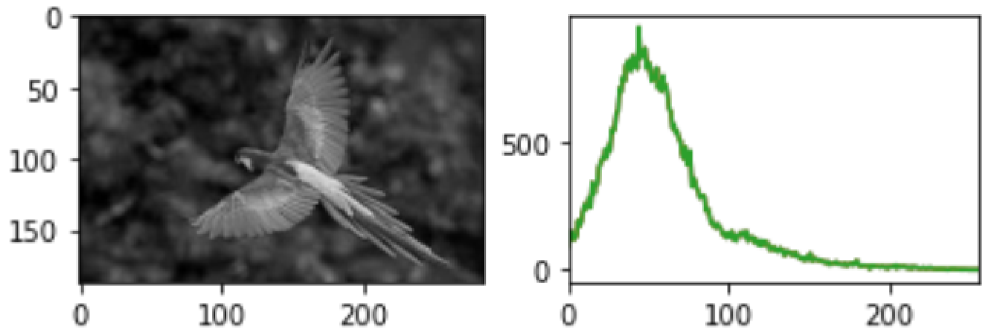


```
[41]: hist = plt.hist(Image.ravel(),256,[0,256])
      hist1 = cv2.calcHist([Image],[0],None,[256],[0,256])
      plt.subplot(221),plt.imshow(Image)
      plt.subplot(222),plt.plot(hist1)
      plt.xlim([0,256])
      plt.show()
```



COLOR IMAGE

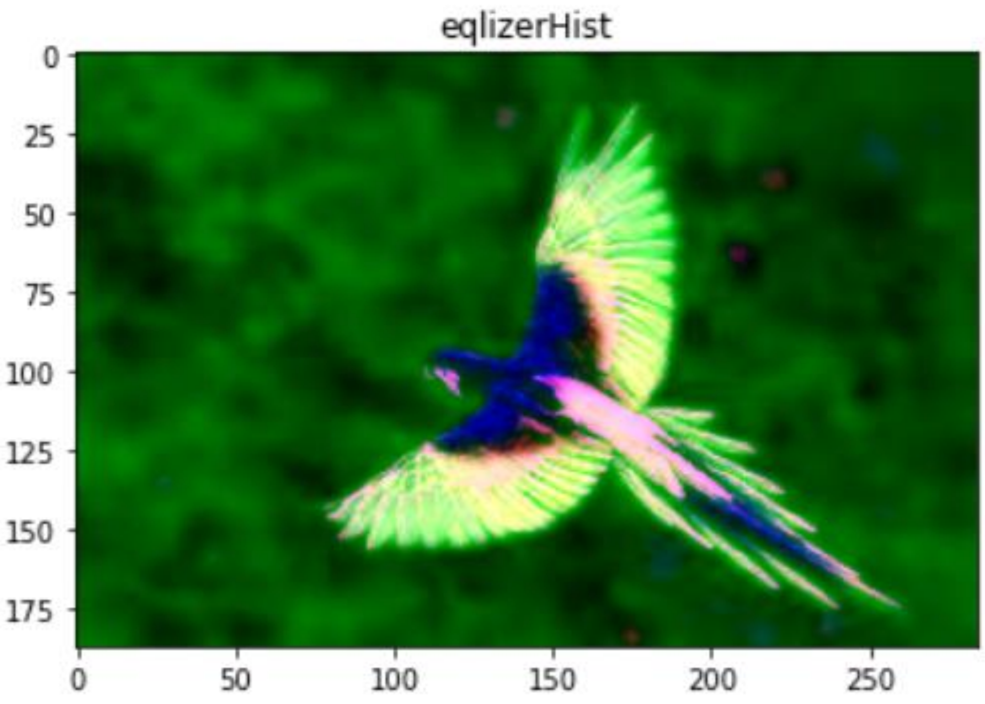
In [51]:
```
Image1 = cv2.imread(path,0)
Image22 = cv2.cvtColor(Image1,cv2.COLOR_GRAY2BGR)
hist = plt.hist(Image.ravel(),256,[0,256])
hist1 = cv2.calcHist([Image22],[0],None,[256],[0,256])
hist2 = cv2.calcHist([Image22],[1],None,[256],[0,256])
hist3 = cv2.calcHist([Image22],[2],None,[256],[0,256])
plt.subplot(221),plt.imshow(Image22)
plt.subplot(222),plt.plot(hist1),plt.plot(hist2),plt.plot(hist3)
plt.xlim([0,256])
plt.show()
```



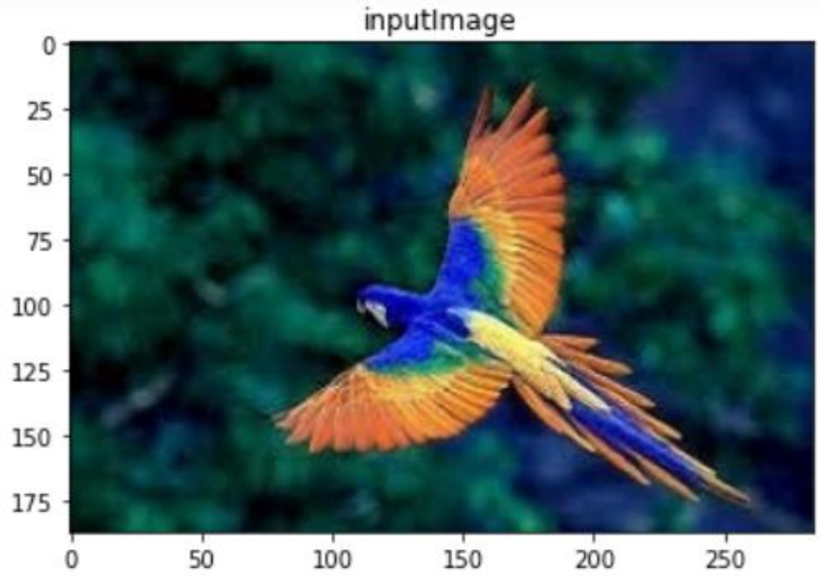
9 Write a python program for enhance the brightness and contrast using histogram equalization.

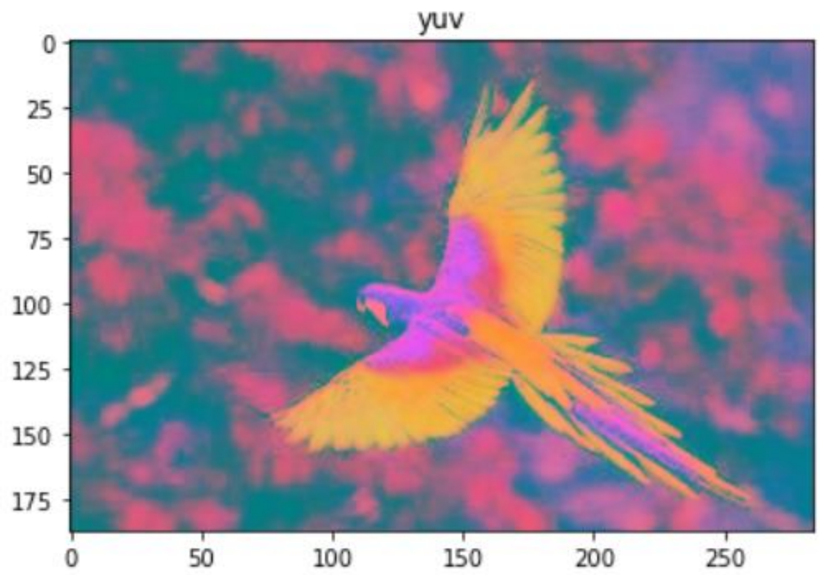
In [52]:
```
import cv2
path = r'./Images/Image22.jpg'
inputImage = cv2.imread(path)
```

In [54]:
```
yuv = cv2.cvtColor(inputImage,cv2.COLOR_BGR2YUV)
yuv[:,:,0] = cv2.equalizeHist(yuv[:,:,0])
eqlizerHist = cv2.cvtColor(inputImage,cv2.COLOR_YUV2BGR)
imagess=[inputImage,yuv,eqlizerHist]
title = ['inputImage','yuv','eqlizerHist']

for i in range(0,3):
    plt.title(title[i])
    plt.imshow(imagess[i])
    plt.show()
```



In [ ]:


inputImage
0
25
50
75
100
125
150
175
0
50
100
150
200
250


yuv
0
25
50
75
100
125
150
175
0
50
100
150
200
250

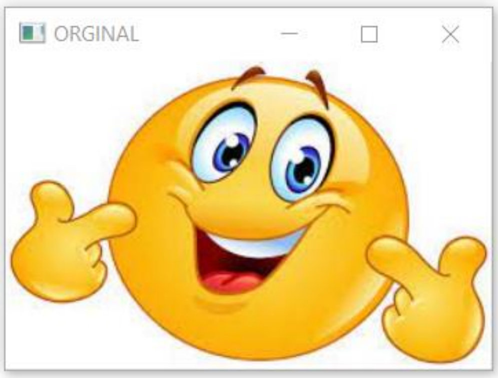
11. Demonstrate morphological operations in an input image and display the same. ¶

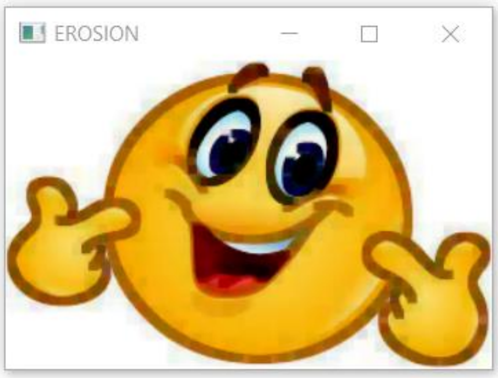
In [1]:

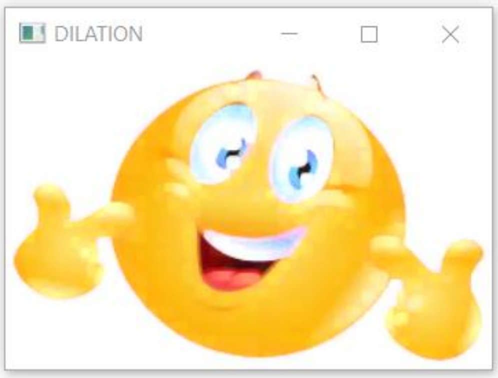
```
import cv2
import numpy as np
path = r'./Images/image3.jpg'
```

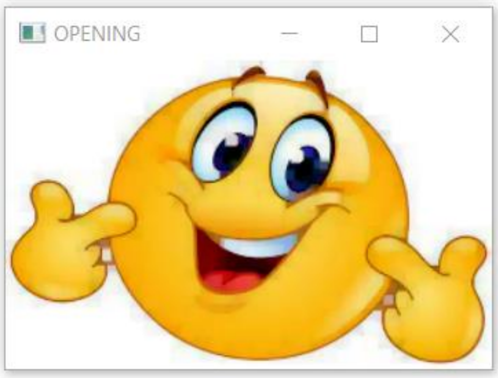
In [*]:

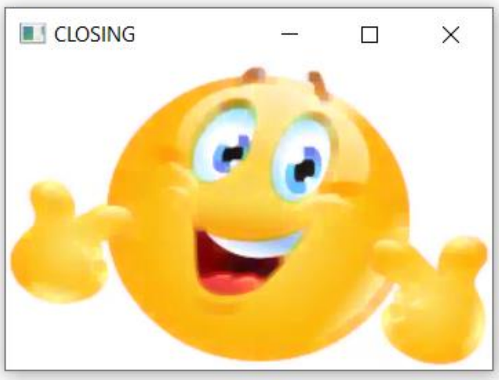
```
imageInput = cv2.imread(path)
kernel = np.ones((5,5),np.uint8)

imag_erode = cv2.erode(imageInput,kernel,iterations=1)
imag_dilate = cv2.dilate(imageInput,kernel,iterations=1)

opening = cv2.morphologyEx(imageInput,cv2.MORPH_OPEN,kernel,iterations=1)
closing = cv2.morphologyEx(imageInput,cv2.MORPH_CLOSE,kernel,iterations=1)

cv2.imshow("ORGINAL",imageInput)
cv2.imshow("EROSION",imag_erode)
cv2.imshow("DILATION",imag_dilate)
cv2.imshow("OPENING",opening)
cv2.imshow("CLOSING",closing)
cv2.waitKey(0)
```

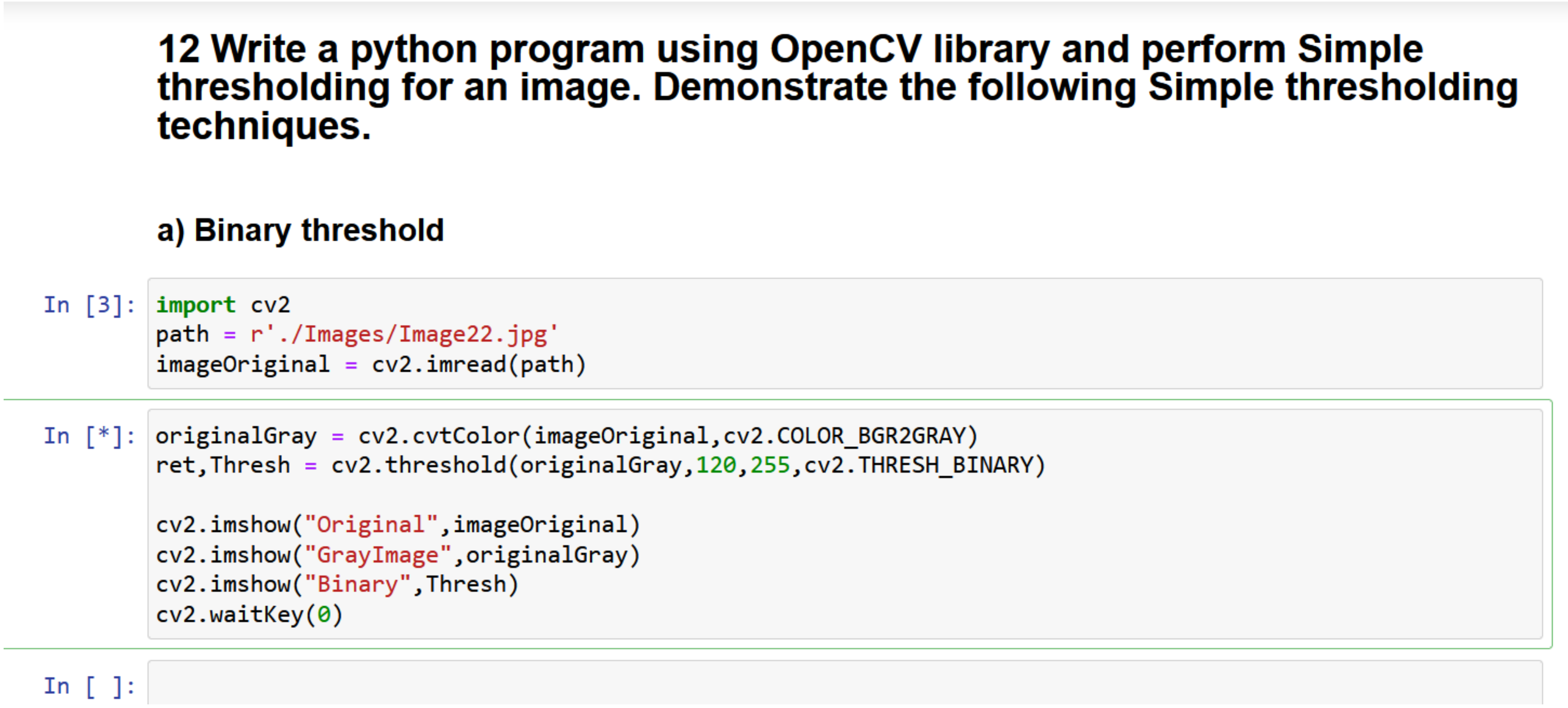












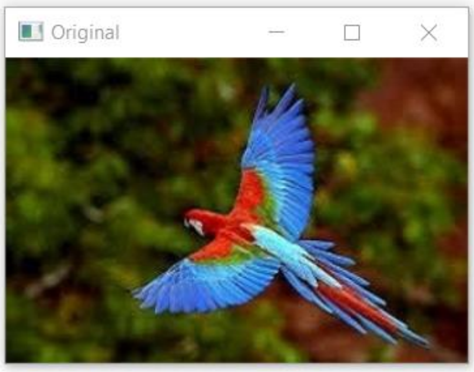
12 Write a python program using OpenCV library and perform Simple thresholding for an image. Demonstrate the following Simple thresholding techniques.

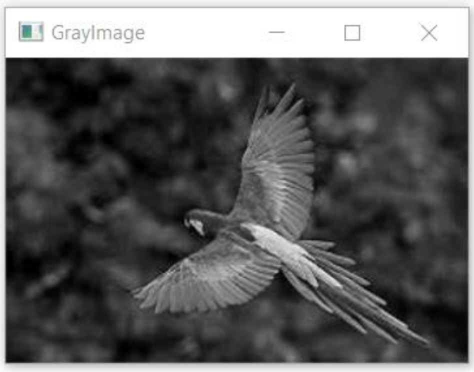
a) Binary threshold

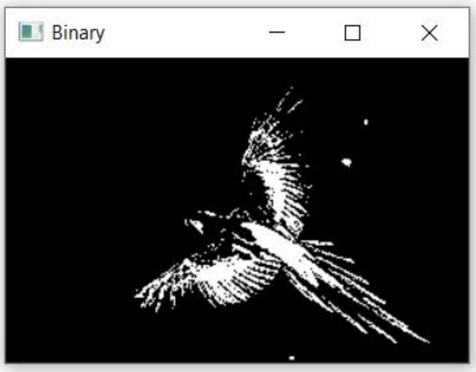
```
In [3]: import cv2
        path = r'./Images/Image22.jpg'
        imageOriginal = cv2.imread(path)
```

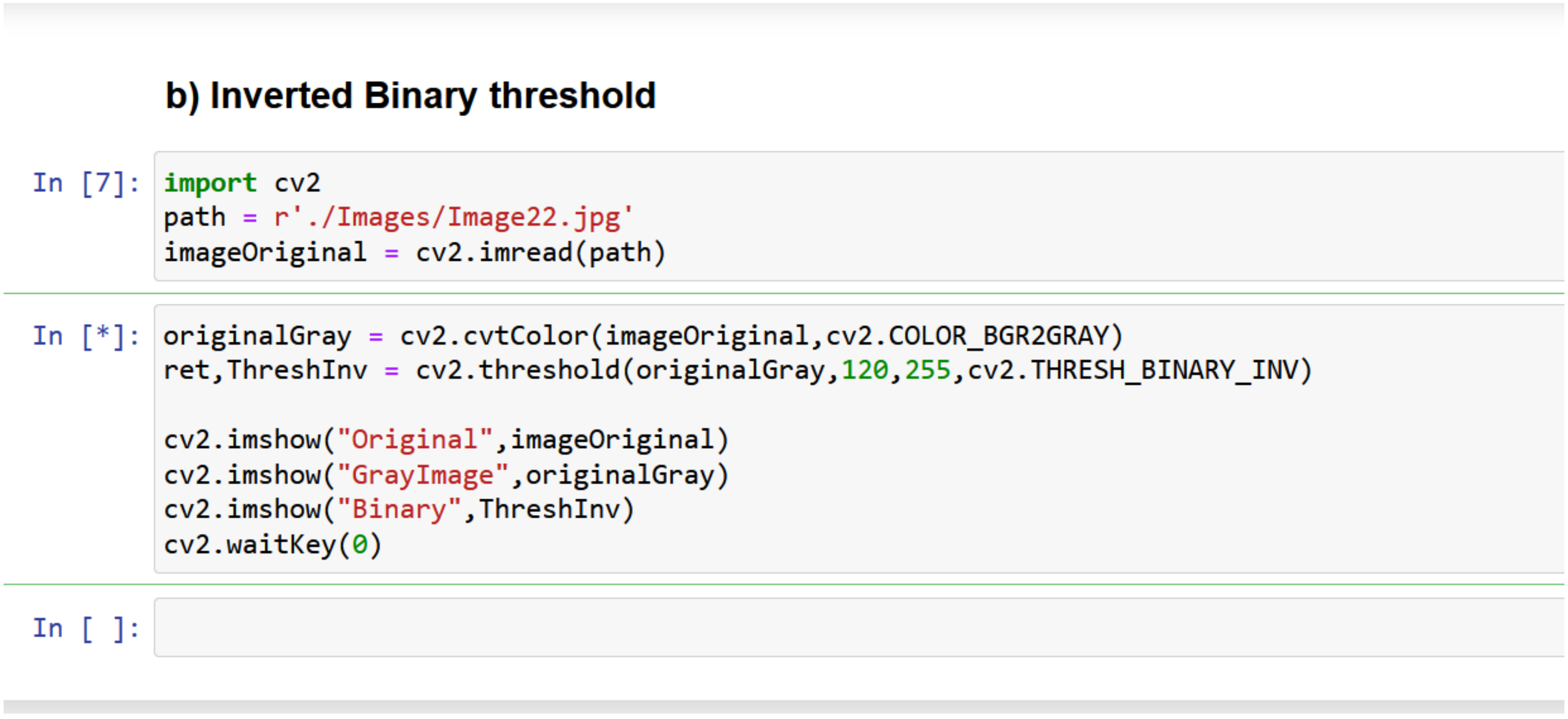
```
In [*]: originalGray = cv2.cvtColor(imageOriginal,cv2.COLOR_BGR2GRAY)
        ret,Thresh = cv2.threshold(originalGray,120,255,cv2.THRESH_BINARY)

        cv2.imshow("Original",imageOriginal)
        cv2.imshow("GrayImage",originalGray)
        cv2.imshow("Binary",Thresh)
        cv2.waitKey(0)
```

```
In [ ]:
```







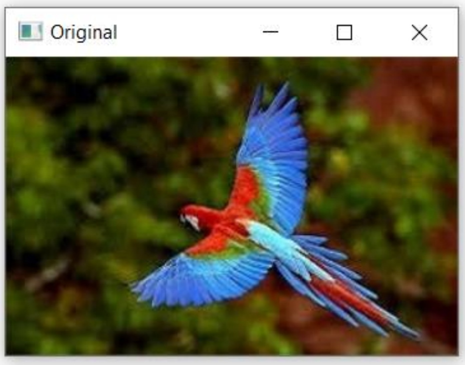


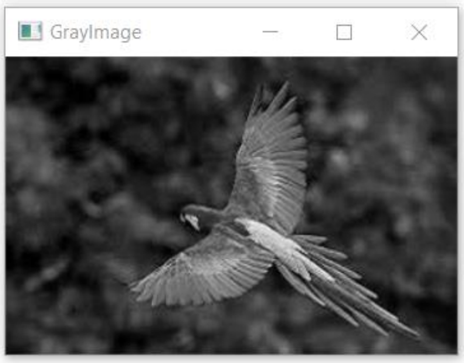
b) Inverted Binary threshold

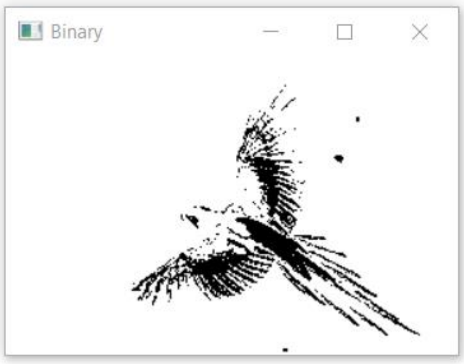
```
In [7]: import cv2
        path = r'./Images/Image22.jpg'
        imageOriginal = cv2.imread(path)
```

```
In [*]: originalGray = cv2.cvtColor(imageOriginal,cv2.COLOR_BGR2GRAY)
        ret,ThreshInv = cv2.threshold(originalGray,120,255,cv2.THRESH_BINARY_INV)

        cv2.imshow("Original",imageOriginal)
        cv2.imshow("GrayImage",originalGray)
        cv2.imshow("Binary",ThreshInv)
        cv2.waitKey(0)
```

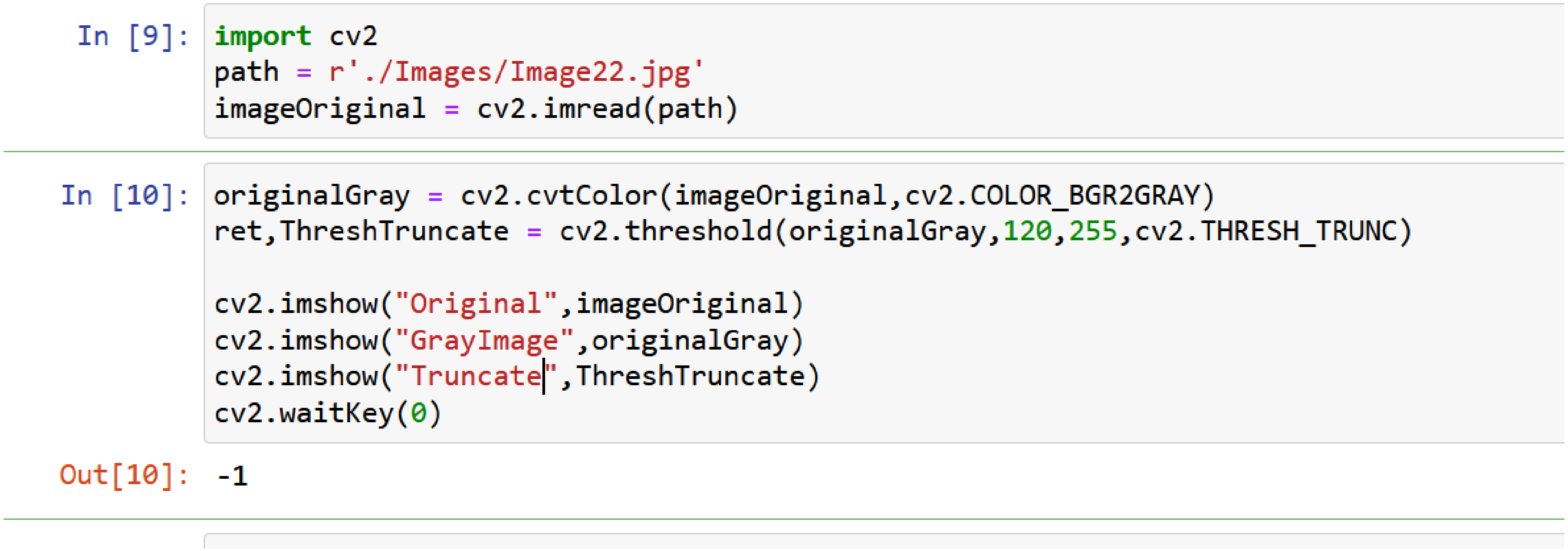
```
In [ ]:
```





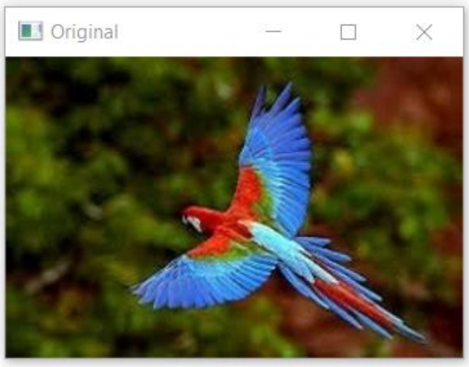


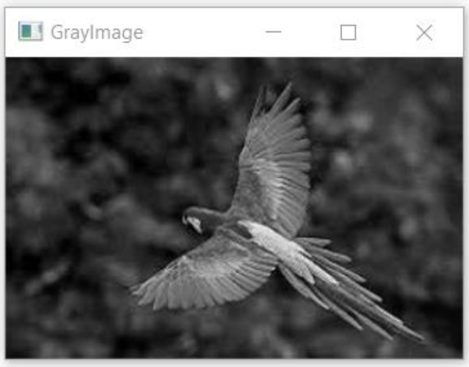
c. Threshold truncate

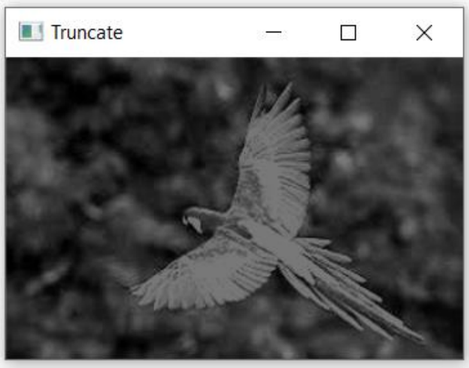


```
In [9]:  import cv2
         path = r'./Images/Image22.jpg'
         imageOriginal = cv2.imread(path)
```

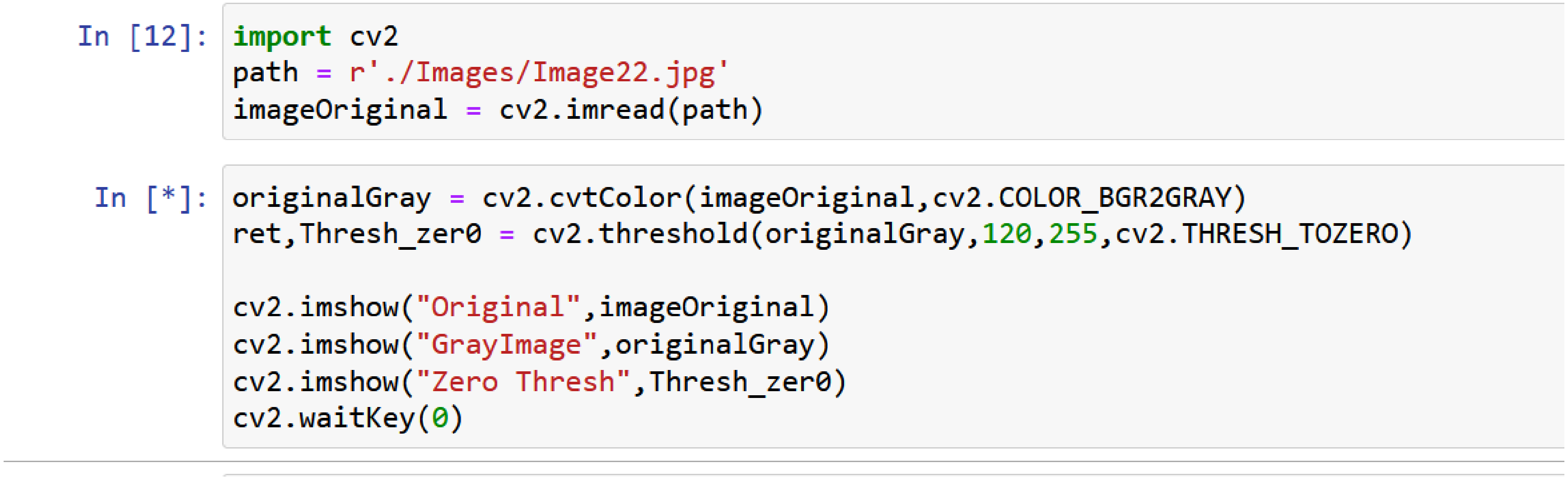
```
In [10]: originalGray = cv2.cvtColor(imageOriginal,cv2.COLOR_BGR2GRAY)
         ret,ThreshTruncate = cv2.threshold(originalGray,120,255,cv2.THRESH_TRUNC)

         cv2.imshow("Original",imageOriginal)
         cv2.imshow("GrayImage",originalGray)
         cv2.imshow("Truncate",ThreshTruncate)
         cv2.waitKey(0)

Out[10]: -1
```





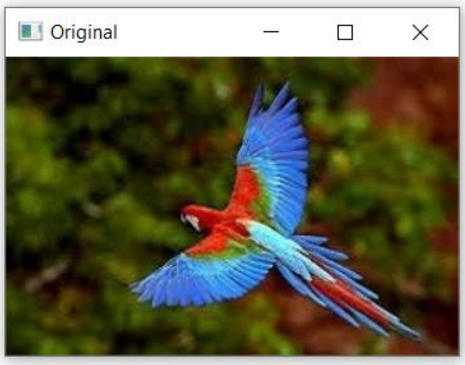


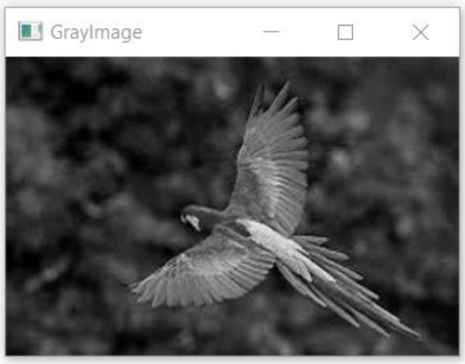
d. Threshold to zero

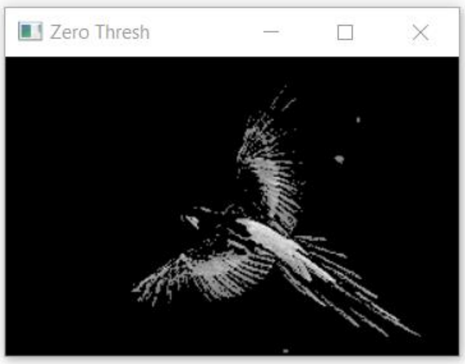


```
In [12]: import cv2
         path = r'./Images/Image22.jpg'
         imageOriginal = cv2.imread(path)
```

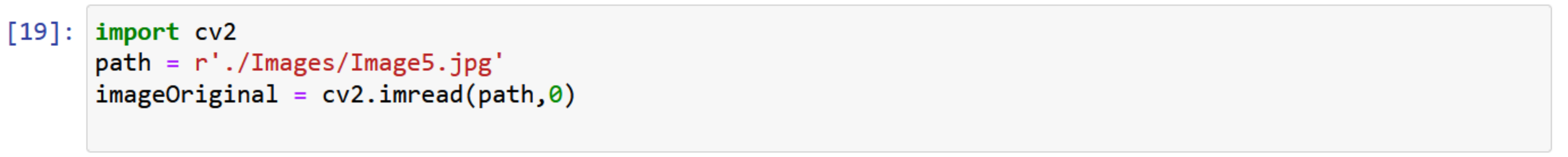
```
In [*]:  originalGray = cv2.cvtColor(imageOriginal,cv2.COLOR_BGR2GRAY)
         ret,Thresh_zer0 = cv2.threshold(originalGray,120,255,cv2.THRESH_TOZERO)

         cv2.imshow("Original",imageOriginal)
         cv2.imshow("GrayImage",originalGray)
         cv2.imshow("Zero Thresh",Thresh_zer0)
         cv2.waitKey(0)
```



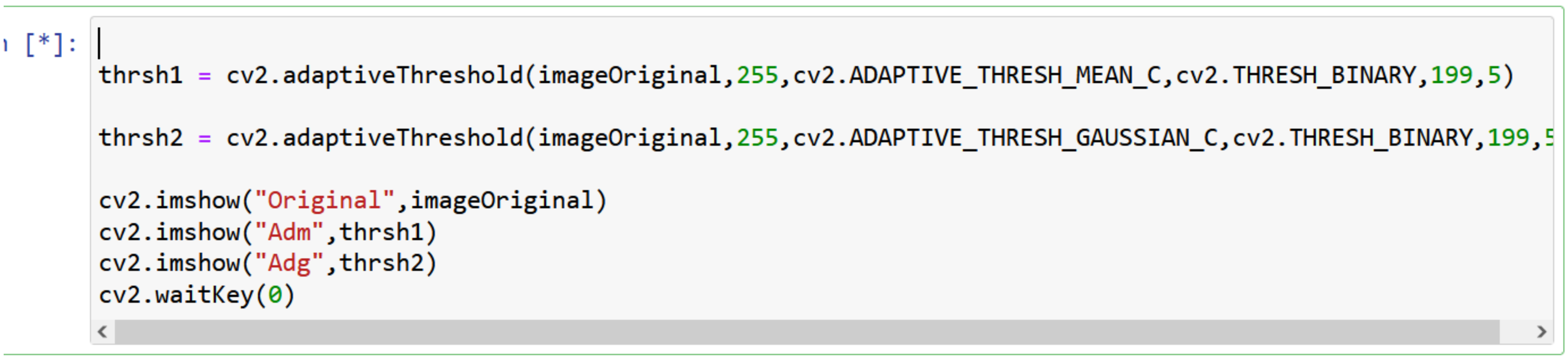




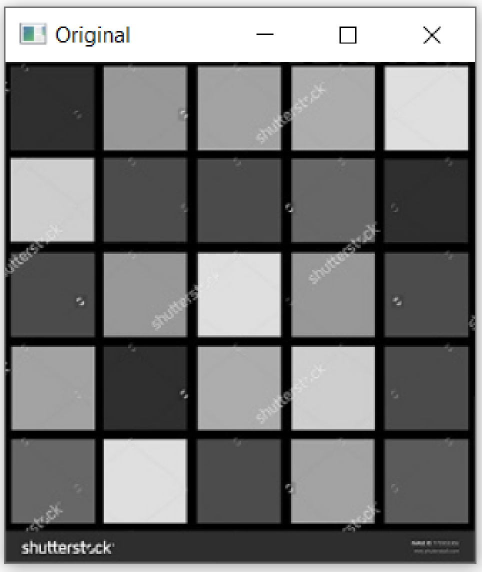
13. Demonstrate Adaptive thresholding in an input image. Use both the Adaptive thresholding methods

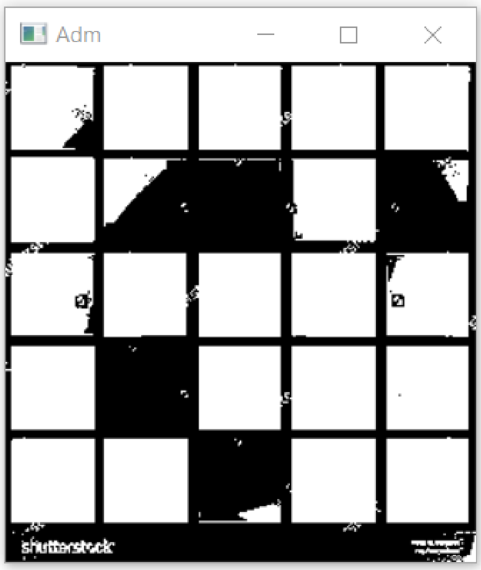


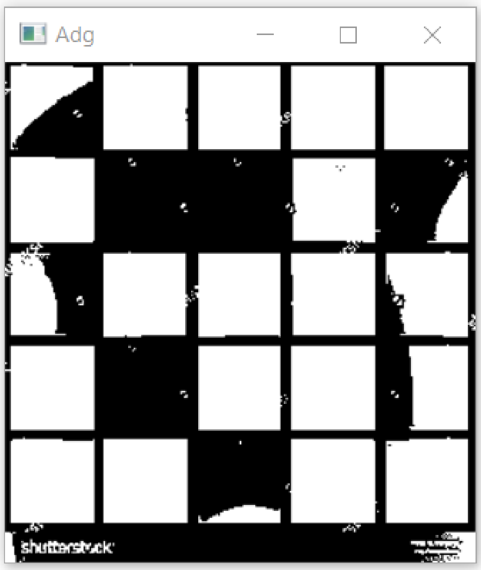
```
[19]: import cv2
      path = r'./Images/Image5.jpg'
      imageOriginal = cv2.imread(path,0)
```



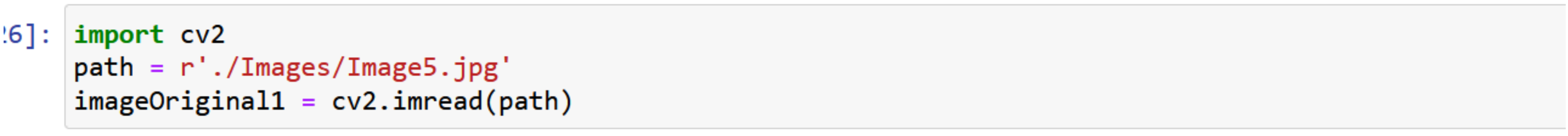
```
[*]: 
     thrsh1 = cv2.adaptiveThreshold(imageOriginal,255,cv2.ADAPTIVE_THRESH_MEAN_C,cv2.THRESH_BINARY,199,5)

     thrsh2 = cv2.adaptiveThreshold(imageOriginal,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY,199,5

     cv2.imshow("Original",imageOriginal)
     cv2.imshow("Adm",thrsh1)
     cv2.imshow("Adg",thrsh2)
     cv2.waitKey(0)
```



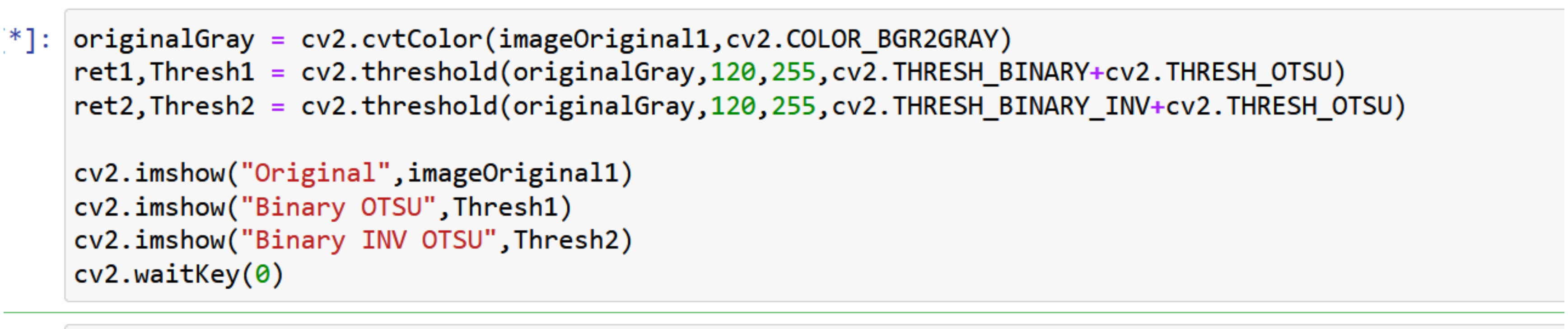




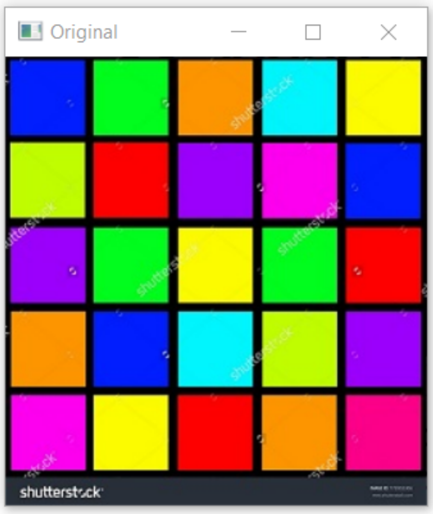
14. Apply Otsu thresholding in an input image with binary thresholding and Inverted binary threshold.

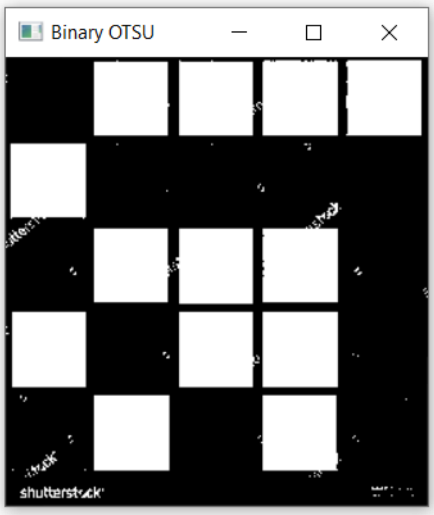


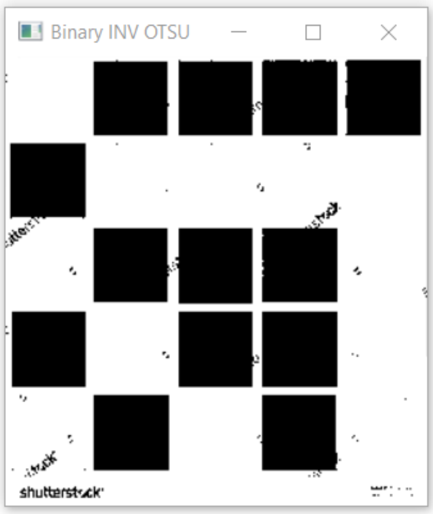
```
6]: import cv2
    path = r'./Images/Image5.jpg'
    imageOriginal1 = cv2.imread(path)
```



```
*]: originalGray = cv2.cvtColor(imageOriginal1,cv2.COLOR_BGR2GRAY)
    ret1,Thresh1 = cv2.threshold(originalGray,120,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
    ret2,Thresh2 = cv2.threshold(originalGray,120,255,cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)

    cv2.imshow("Original",imageOriginal1)
    cv2.imshow("Binary OTSU",Thresh1)
    cv2.imshow("Binary INV OTSU",Thresh2)
    cv2.waitKey(0)
```


Original
shutterstock


Binary OTSU
shutterstock


Binary INV OTSU
shutterstock