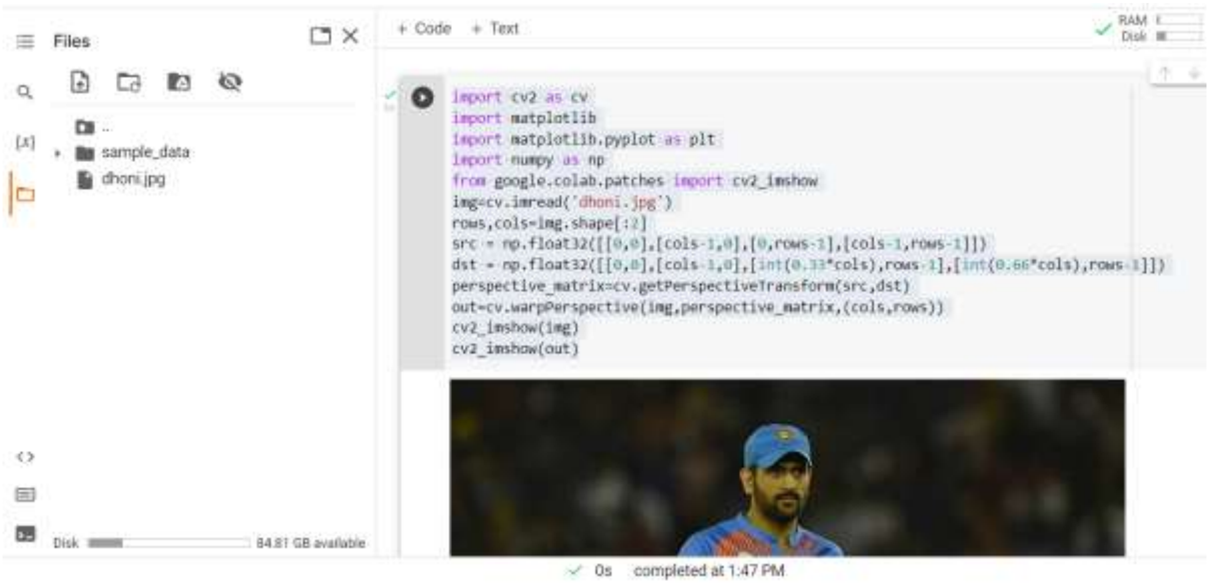
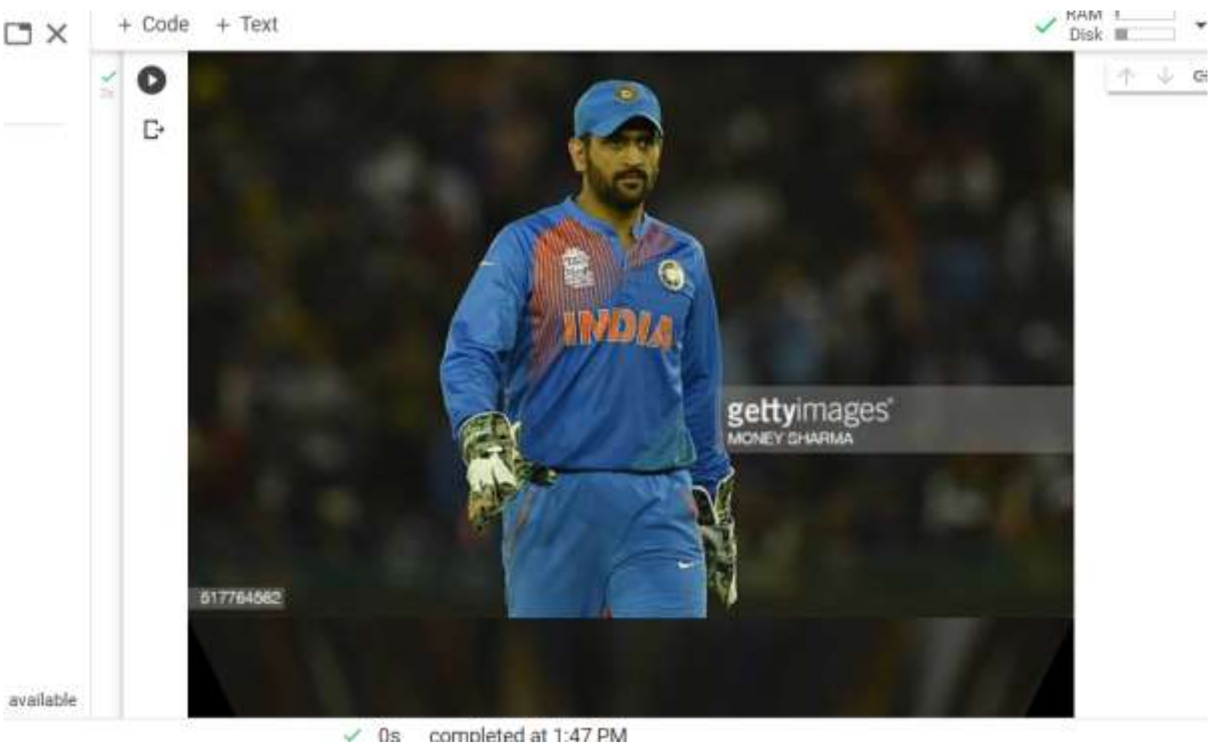
02_GAURAV BANE

```python
import cv2 as cv
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
from google.colab.patches import cv2_imshow
img=cv.imread('dhoni.jpg')
rows,cols=img.shape[:2]
src = np.float32([[0,0],[cols-1,0],[0,rows-1],[cols-1,rows-1]])
dst = np.float32([[0,0],[cols-1,0],[int(0.33*cols),rows-
1],[int(0.66*cols),rows-1]])
perspective_matrix=cv.getPerspectiveTransform(src,dst)
out=cv.warpPerspective(img,perspective_matrix,(cols,rows))
cv2_imshow(img)
cv2_imshow(out)
```

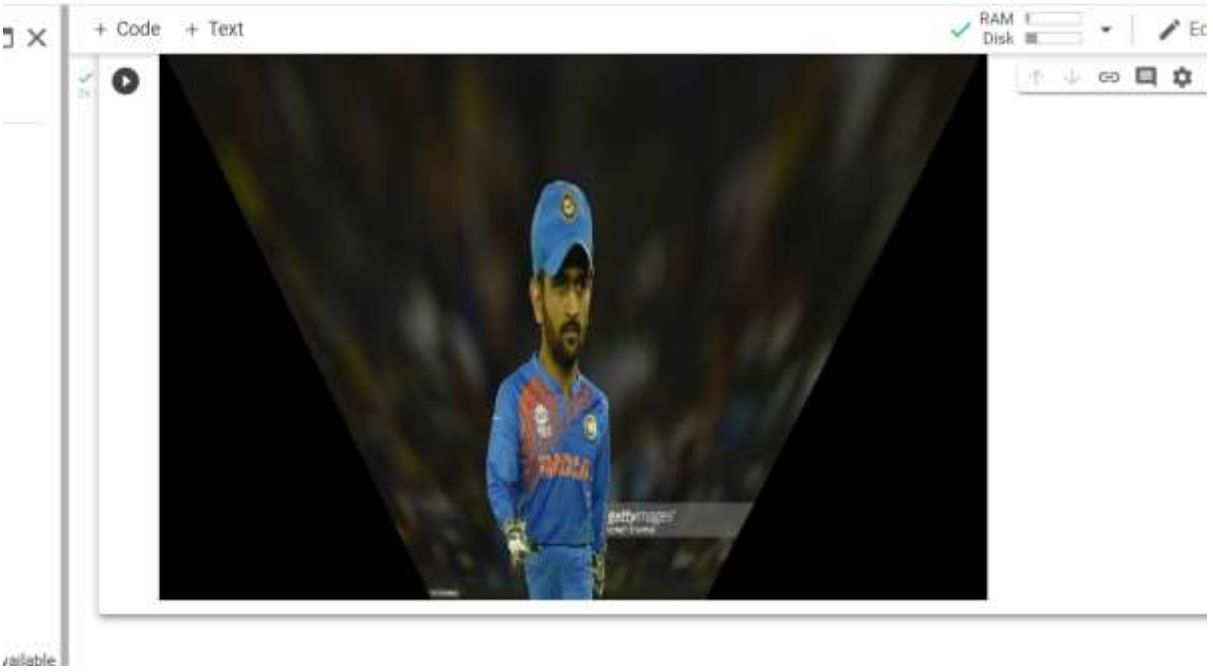


```python
import cv2 as cv
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
from google.colab.patches import cv2_imshow
img=cv.imread('dhoni.jpg')
rows,cols=img.shape[:2]
src = np.float32([[0,0],[cols-1,0],[0,rows-1],[cols-1,rows-1]])
dst = np.float32([[0,0],[cols-1,0],[int(0.33*cols),rows-1],[int(0.66*cols),rows-1]])
perspective_matrix=cv.getPerspectiveTransform(src,dst)
out=cv.warpPerspective(img,perspective_matrix,(cols,rows))
cv2_imshow(img)
cv2_imshow(out)
```

02_GAURAV BANE



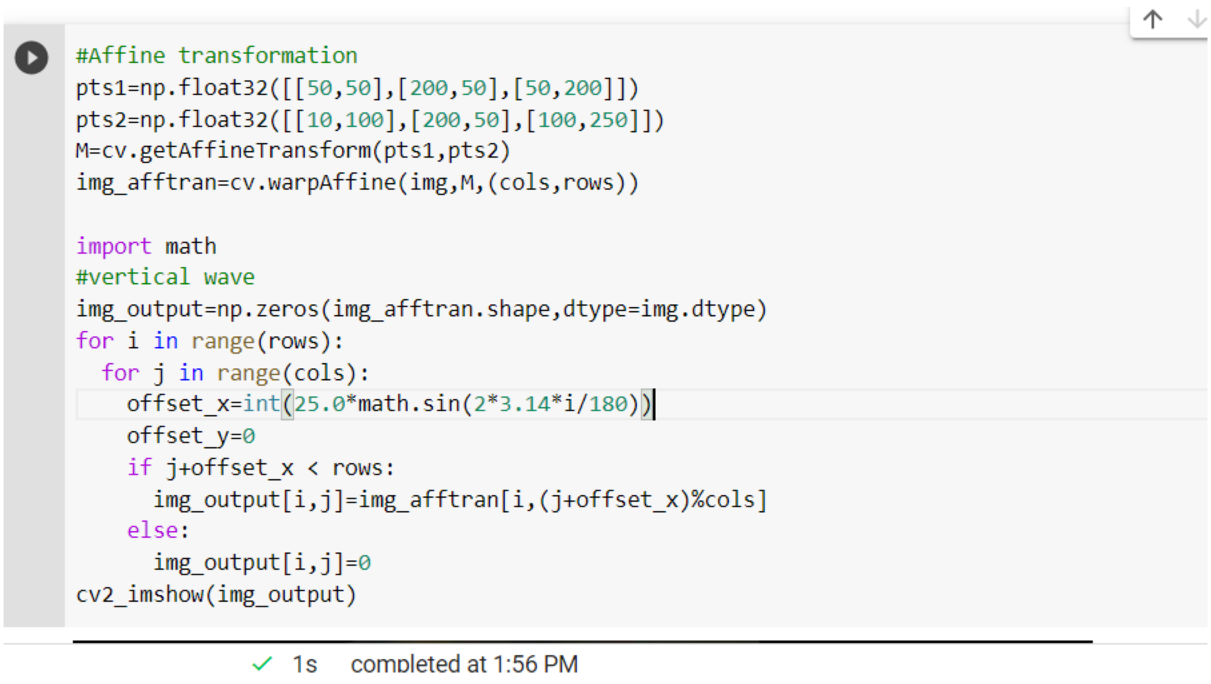
✓ 0s completed at 1:47 PM



```
#Affine transformation
pts1=np.float32([[50,50],[200,50],[50,200]])
pts2=np.float32([[10,100],[200,50],[100,250]])
M=cv.getAffineTransform(pts1,pts2)
img_afftran=cv.warpAffine(img,M,(cols,rows))

import math
#vertical wave
img_output=np.zeros(img_afftran.shape,dtype=img.dtype)
```

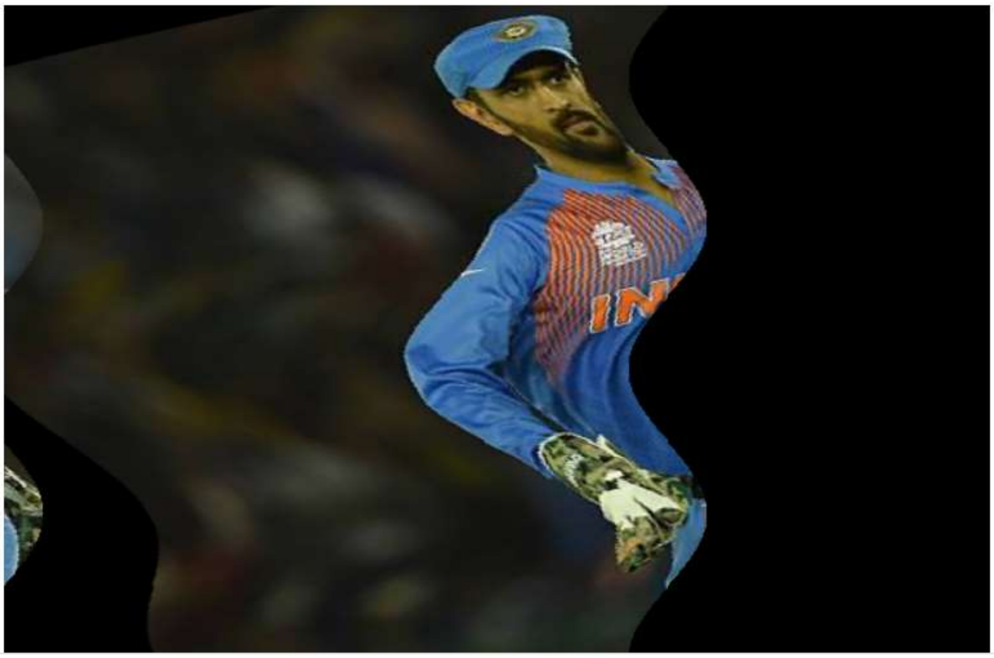
02_GAURAV BANE

```
for i in range(rows):
  for j in range(cols):
    offset_x=int(25.0*math.sin(2*3.14*i/180))
    offset_y=0
    if j+offset_x < rows:
      img_output[i,j]=img_afftran[i,(j+offset_x)%cols]
    else:
      img_output[i,j]=0
cv2_imshow(img_output)
```



```
#Affine transformation
pts1=np.float32([[50,50],[200,50],[50,200]])
pts2=np.float32([[10,100],[200,50],[100,250]])
M=cv.getAffineTransform(pts1,pts2)
img_afftran=cv.warpAffine(img,M,(cols,rows))

import math
#vertical wave
img_output=np.zeros(img_afftran.shape,dtype=img.dtype)
for i in range(rows):
  for j in range(cols):
    offset_x=int(25.0*math.sin(2*3.14*i/180))
    offset_y=0
    if j+offset_x < rows:
      img_output[i,j]=img_afftran[i,(j+offset_x)%cols]
    else:
      img_output[i,j]=0
cv2_imshow(img_output)
```

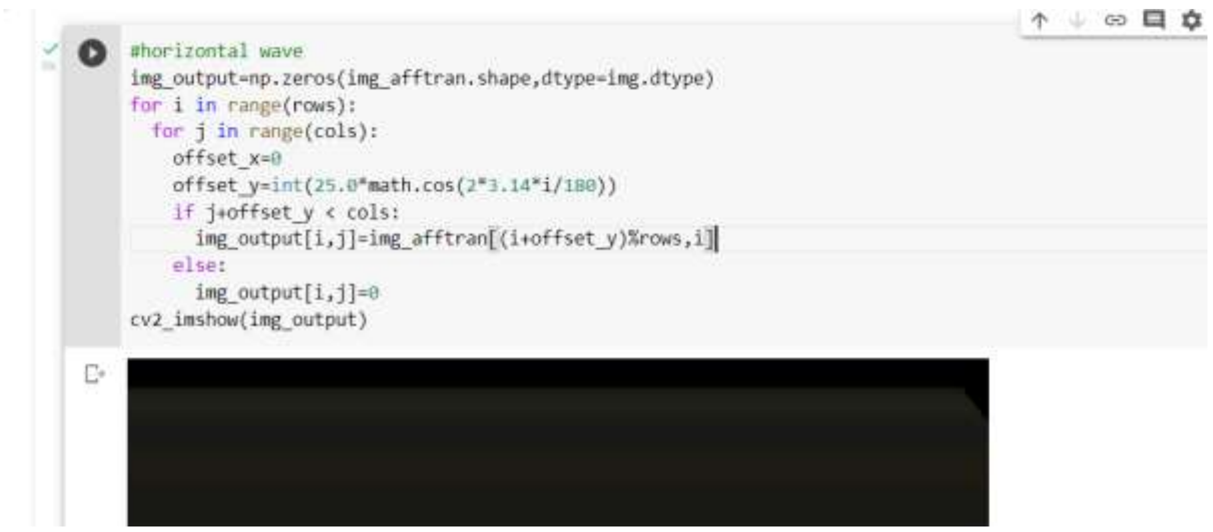
✓ 1s completed at 1:56 PM



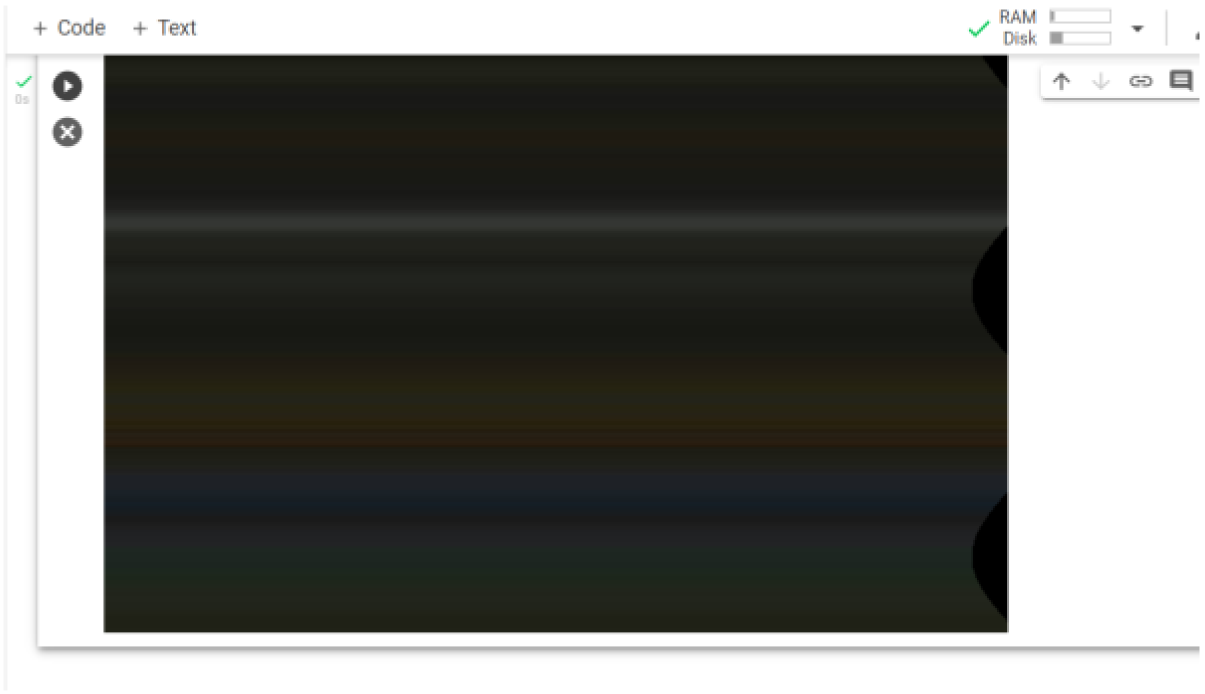
✓ 1s completed at 1:56 PM

02_GAURAV BANE

```
#horizontal wave
img_output=np.zeros(img_afftran.shape,dtype=img.dtype)
for i in range(rows):
  for j in range(cols):
    offset_x=0
    offset_y=int(25.0*math.cos(2*3.14*i/180))
    if j+offset_y < cols:
      img_output[i,j]=img_afftran[(i+offset_y)%rows,i]
    else:
      img_output[i,j]=0
cv2_imshow(img_output)
```

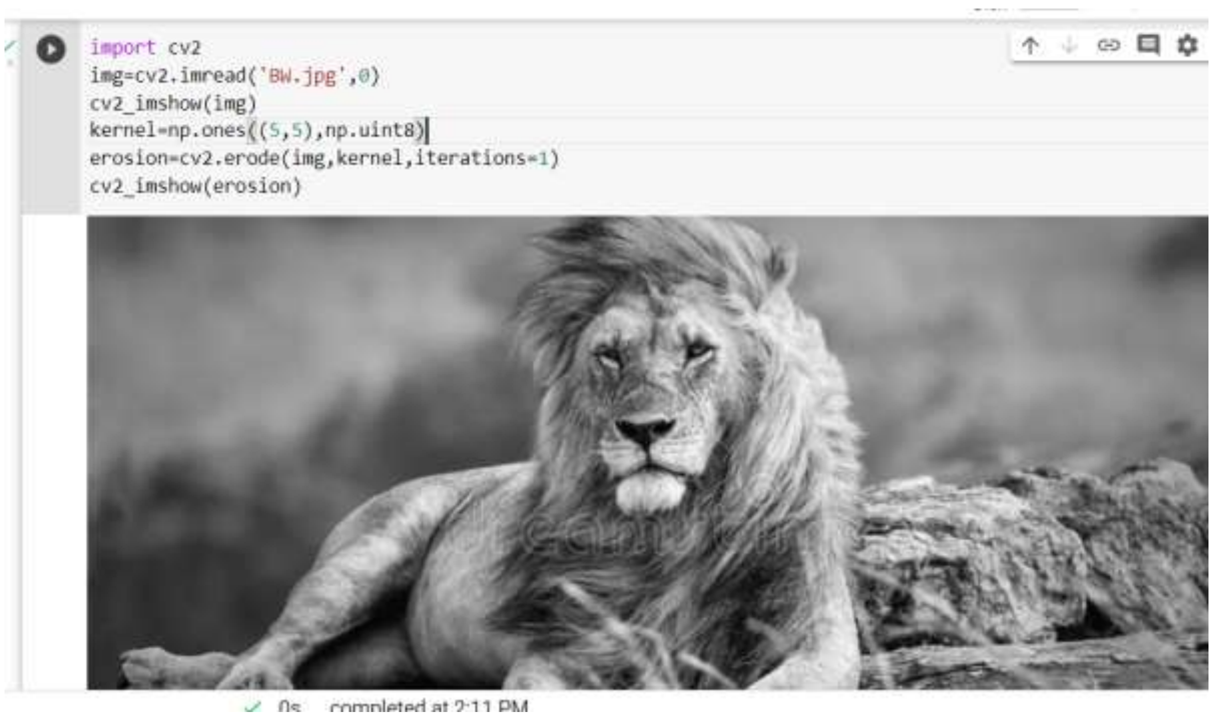


```
#horizontal wave
img_output=np.zeros(img_afftran.shape,dtype=img.dtype)
for i in range(rows):
  for j in range(cols):
    offset_x=0
    offset_y=int(25.0*math.cos(2*3.14*i/180))
    if j+offset_y < cols:
      img_output[i,j]=img_afftran[(i+offset_y)%rows,i]
    else:
      img_output[i,j]=0
cv2_imshow(img_output)
```



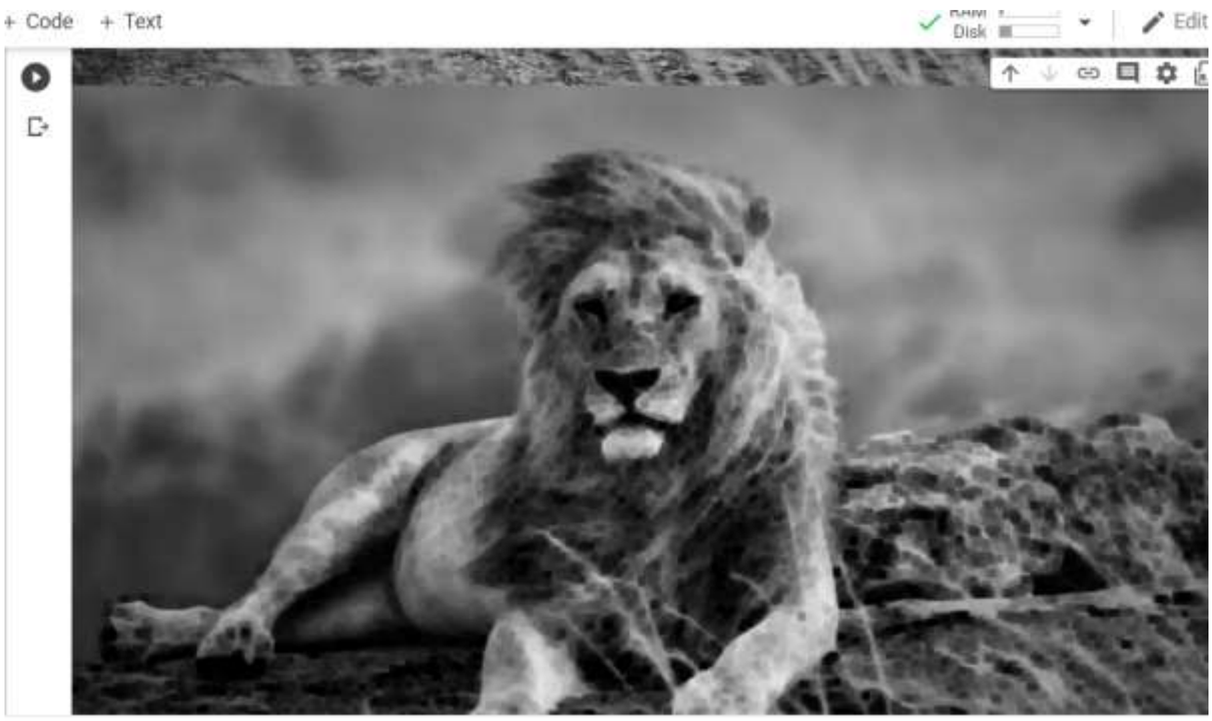
02_GAURAV BANE

```
import cv2
img=cv2.imread('BW.jpg',0)
cv2_imshow(img)
kernel=np.ones((5,5),np.uint8)
erosion=cv2.erode(img,kernel,iterations=1)
cv2_imshow(erosion)
```



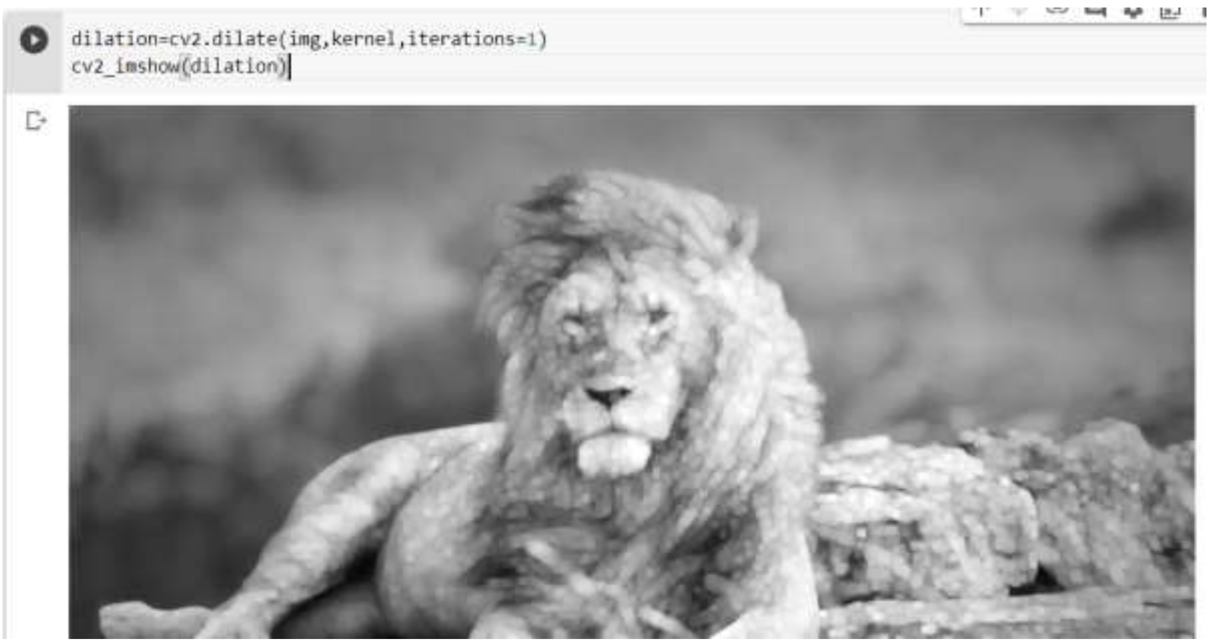
```
import cv2
img=cv2.imread('BW.jpg',0)
cv2_imshow(img)
kernel=np.ones((5,5),np.uint8)
erosion=cv2.erode(img,kernel,iterations=1)
cv2_imshow(erosion)
```

✓ 0s completed at 2:11 PM



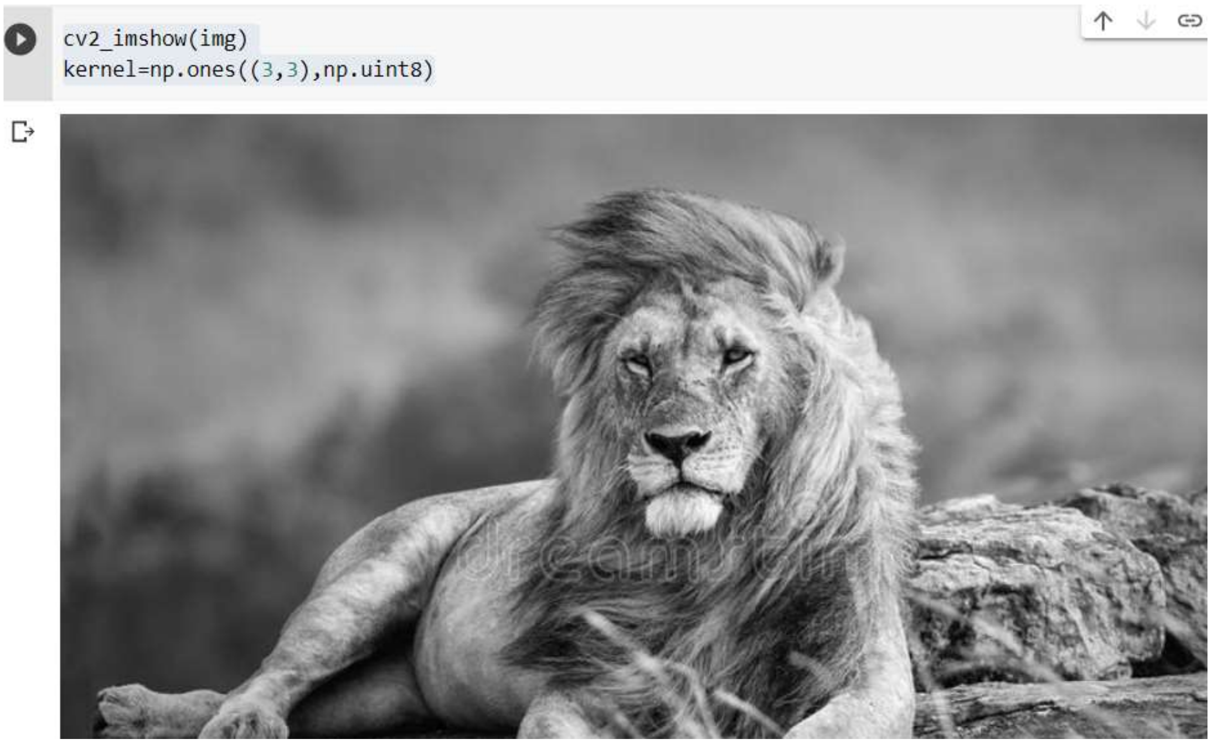
02_GAURAV BANE

```
dilation=cv2.dilate(img,kernel,iterations=1)
cv2_imshow(dilation)
```



```
dilation=cv2.dilate(img,kernel,iterations=1)
cv2_imshow(dilation)
```

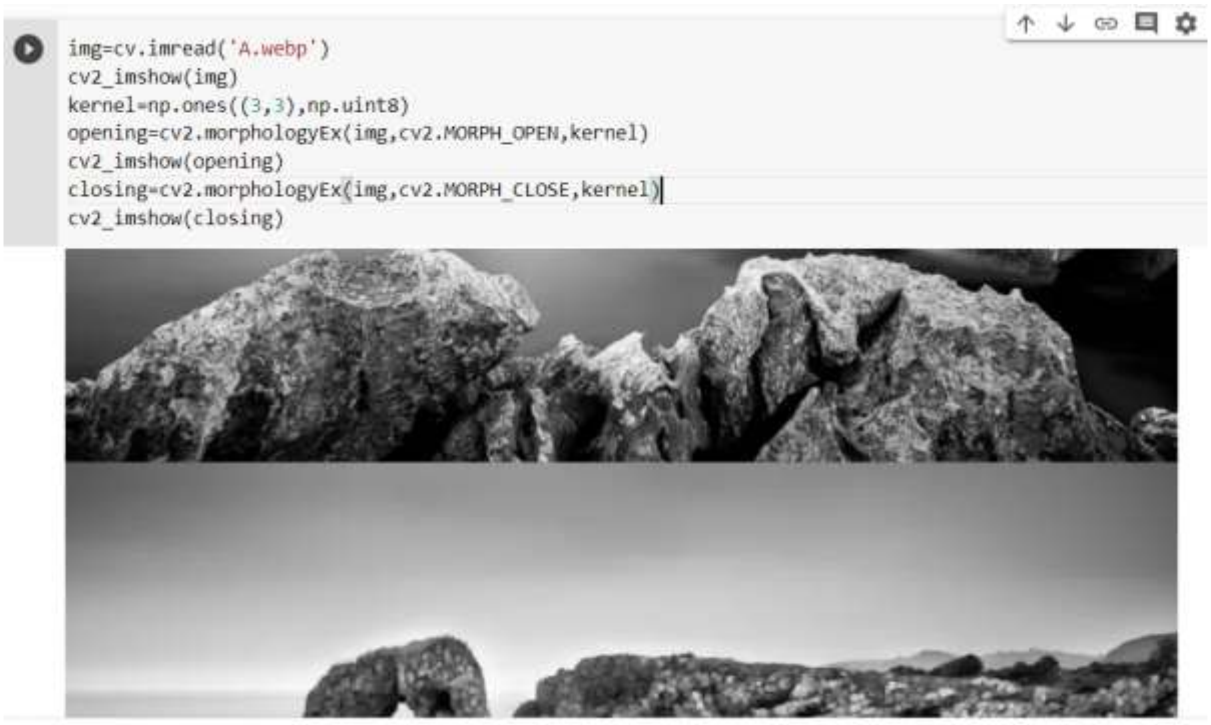
```
cv2_imshow(img)
kernel=np.ones((3,3),np.uint8)
```



```
cv2_imshow(img)
kernel=np.ones((3,3),np.uint8)
```

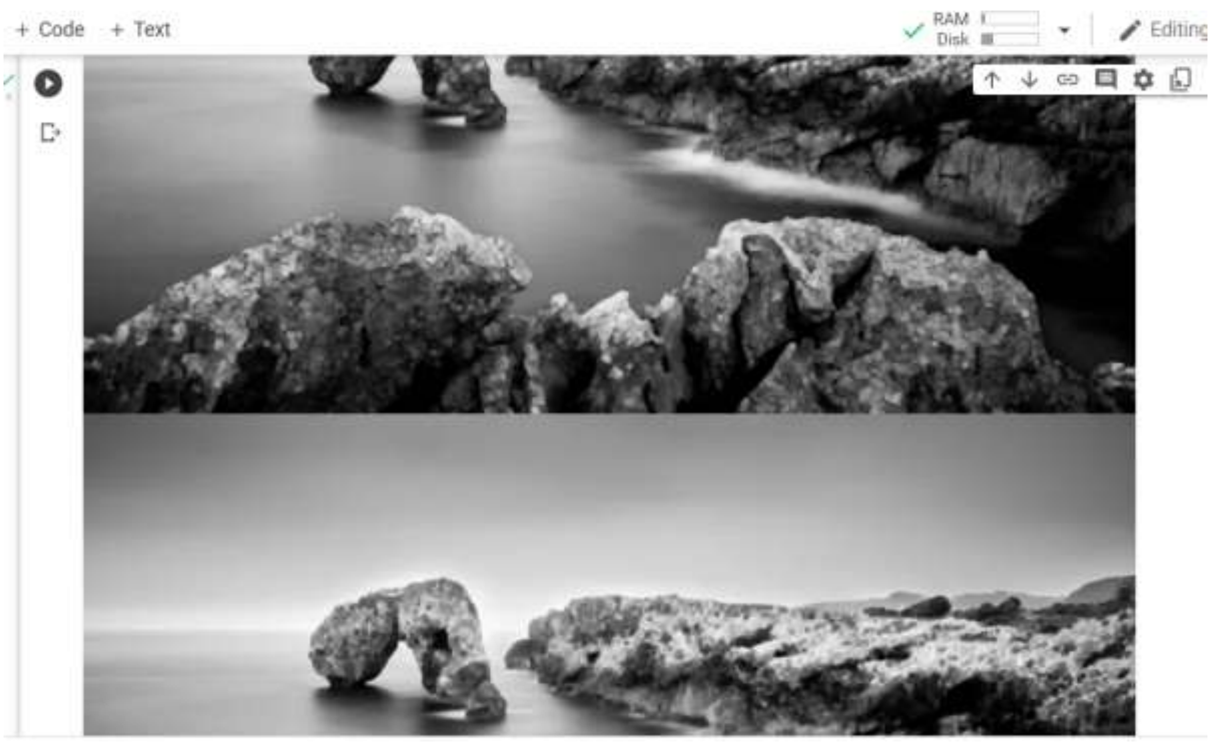
02_GAURAV BANE

```
img=cv.imread('A.webp')
cv2_imshow(img)
kernel=np.ones((3,3),np.uint8)
opening=cv2.morphologyEx(img,cv2.MORPH_OPEN,kernel)
cv2_imshow(opening)
closing=cv2.morphologyEx(img,cv2.MORPH_CLOSE,kernel)
cv2_imshow(closing)
```



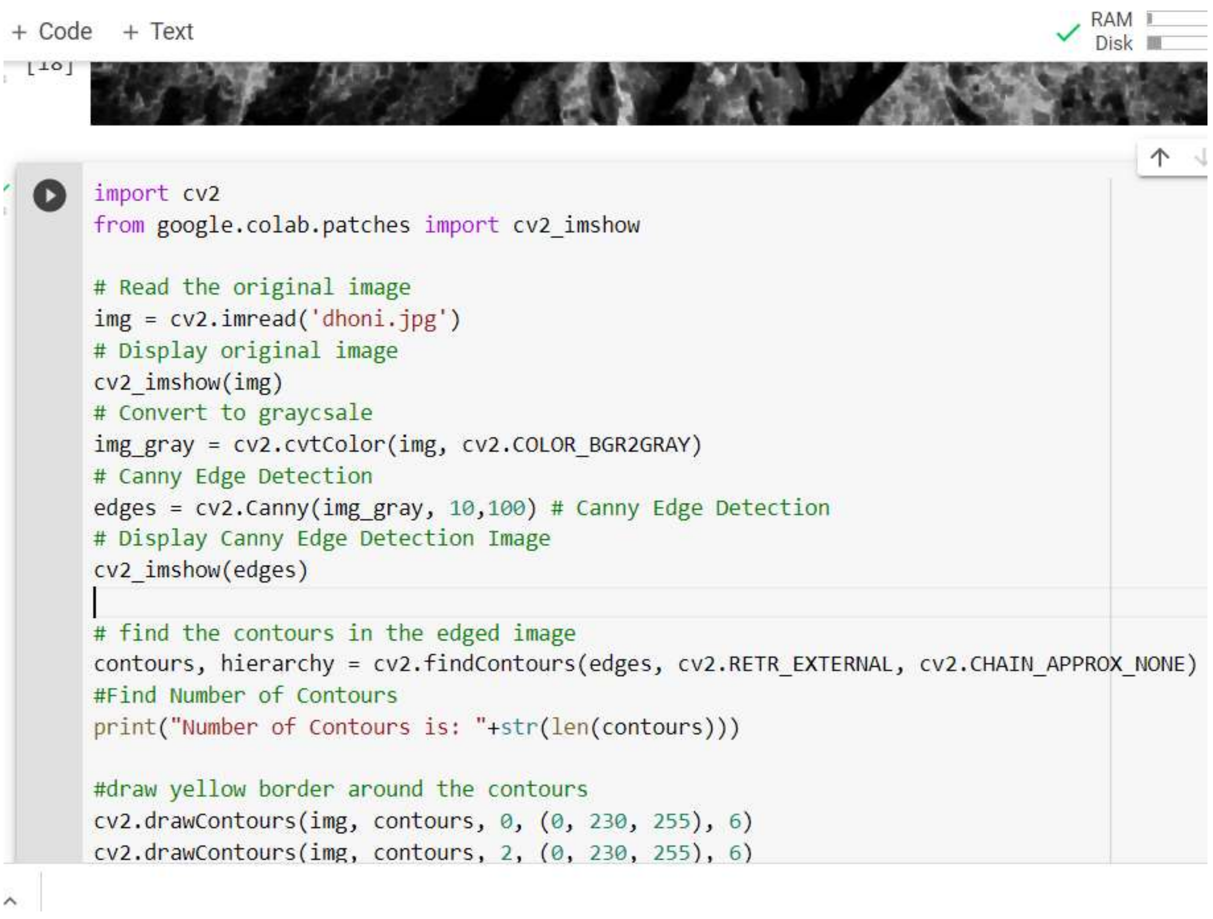
```
img=cv.imread('A.webp')
cv2_imshow(img)
kernel=np.ones((3,3),np.uint8)
opening=cv2.morphologyEx(img,cv2.MORPH_OPEN,kernel)
cv2_imshow(opening)
closing=cv2.morphologyEx(img,cv2.MORPH_CLOSE,kernel)
cv2_imshow(closing)
```

02_GAURAV BANE

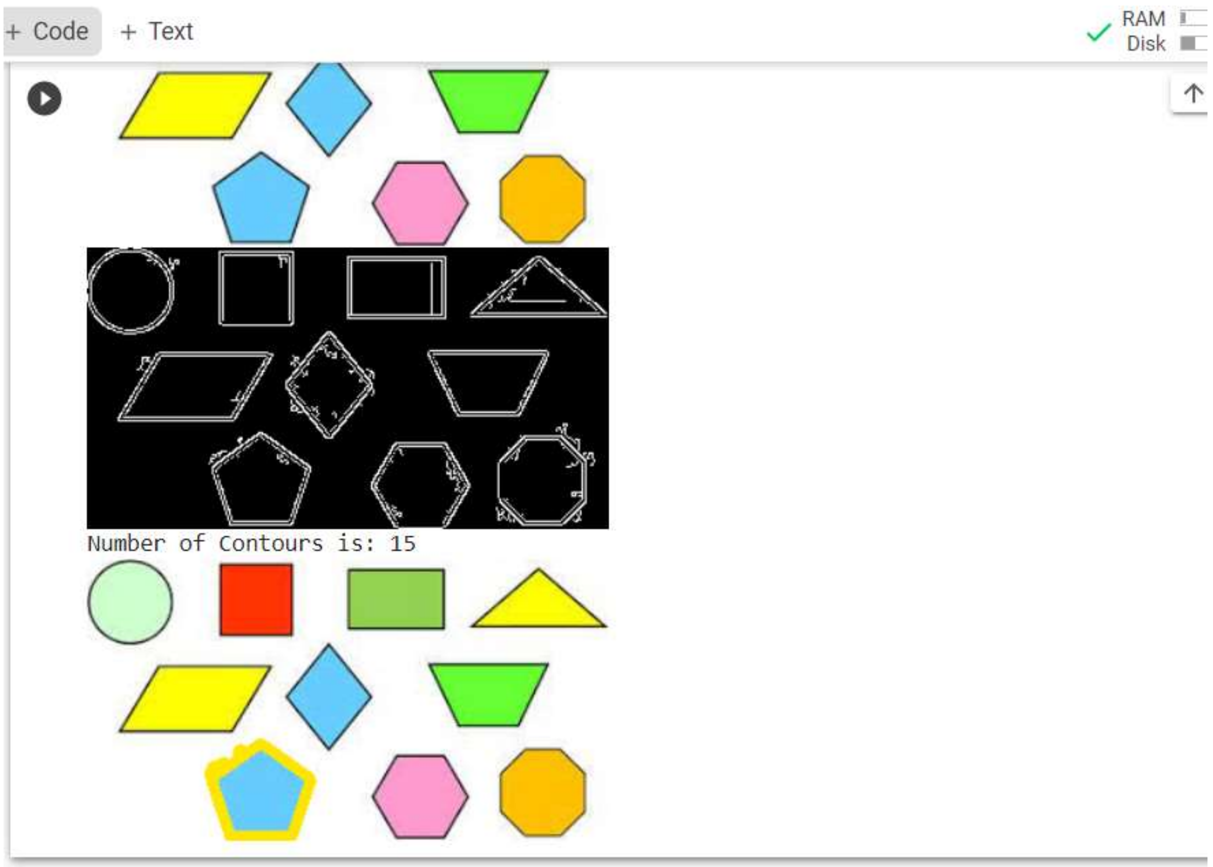


```
import cv2
from google.colab.patches import cv2_imshow

# Read the original image
img = cv2.imread('dhoni.jpg')
# Display original image
cv2_imshow(img)
# Convert to graycsale
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Canny Edge Detection
edges = cv2.Canny(img_gray, 10,100) # Canny Edge Detection
# Display Canny Edge Detection Image
cv2_imshow(edges)

# find the contours in the edged image
contours, hierarchy = cv2.findContours(edges, cv2.RETR_EXTERNAL, cv2.CH
AIN_APPROX_NONE)
#Find Number of Contours
print("Number of Contours is: "+str(len(contours)))

#draw yellow border around the contours
cv2.drawContours(img, contours, 0, (0, 230, 255), 6)
cv2.drawContours(img, contours, 2, (0, 230, 255), 6)

#show the image with Contours
cv2_imshow(img)
```

02_GAURAV BANE



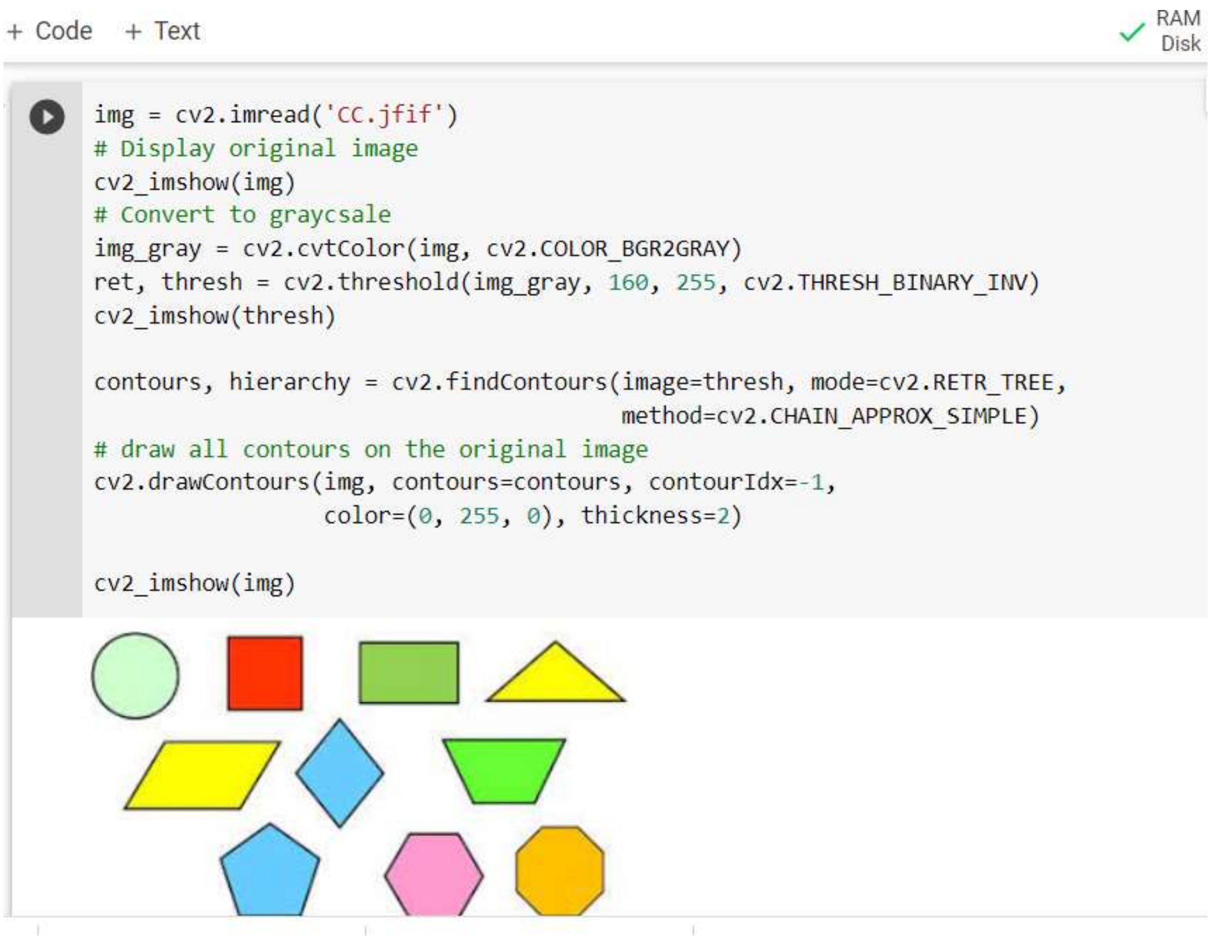
```
import cv2
from google.colab.patches import cv2_imshow

# Read the original image
img = cv2.imread('dhoni.jpg')
# Display original image
cv2_imshow(img)
# Convert to graycsale
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Canny Edge Detection
edges = cv2.Canny(img_gray, 10,100) # Canny Edge Detection
# Display Canny Edge Detection Image
cv2_imshow(edges)

# find the contours in the edged image
contours, hierarchy = cv2.findContours(edges, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
#Find Number of Contours
print("Number of Contours is: "+str(len(contours)))

#draw yellow border around the contours
cv2.drawContours(img, contours, 0, (0, 230, 255), 6)
cv2.drawContours(img, contours, 2, (0, 230, 255), 6)
```



```
Number of Contours is: 15
```

02_GAURAV BANE

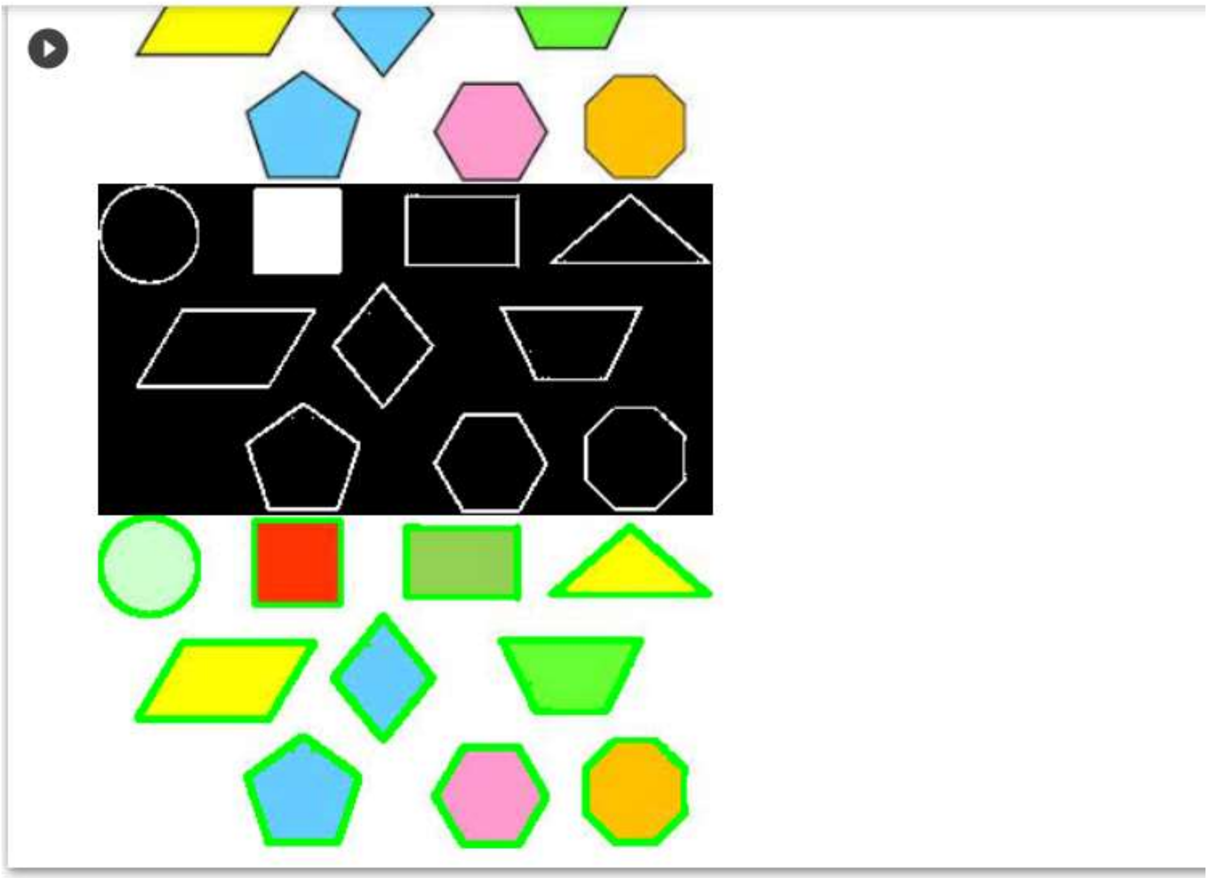
```
img = cv2.imread('CC.jfif')
# Display original image
cv2_imshow(img)
# Convert to graycsale
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
ret, thresh = cv2.threshold(img_gray, 255, 255, cv2.THRESH_BINARY_INV)
cv2_imshow(thresh)

contours, hierarchy = cv2.findContours(image=thresh, mode=cv2.RETR_TREE
,
                                       method=cv2.CHAIN_APPROX_SIMPLE)
# draw all contours on the original image
cv2.drawContours(img, contours=contours, contourIdx=-1,
                 color=(0, 255, 0), thickness=2, lineType=cv2.LINE_AA)

cv2_imshow(img)
```



```
img = cv2.imread('CC.jfif')
# Display original image
cv2_imshow(img)
# Convert to graycsale
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
ret, thresh = cv2.threshold(img_gray, 160, 255, cv2.THRESH_BINARY_INV)
cv2_imshow(thresh)

contours, hierarchy = cv2.findContours(image=thresh, mode=cv2.RETR_TREE,
                                       method=cv2.CHAIN_APPROX_SIMPLE)
# draw all contours on the original image
cv2.drawContours(img, contours=contours, contourIdx=-1,
                 color=(0, 255, 0), thickness=2)

cv2_imshow(img)
```

02_GAURAV BANE

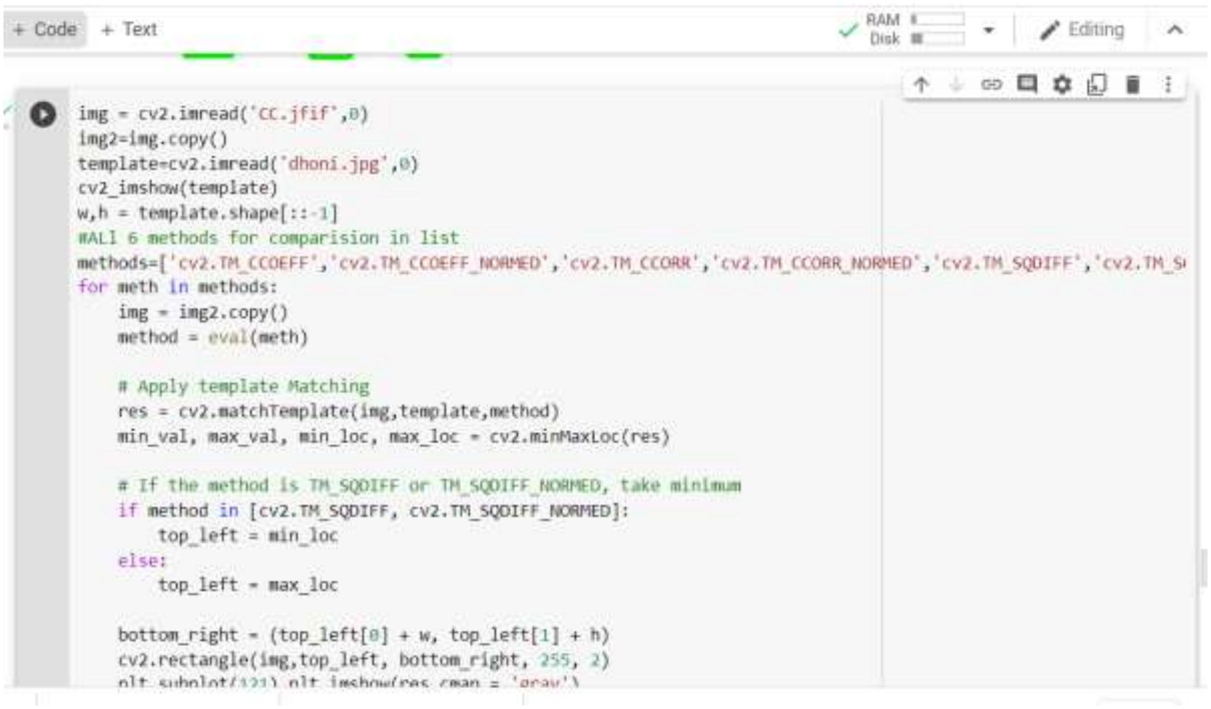
+ Code + Text



```
img = cv2.imread('CC.jfif',0)
img2=img.copy()
template=cv2.imread('dhoni.jpg',0)
cv2_imshow(template)
w,h = template.shape[::-1]
#ALl 6 methods for comparision in list
methods=['cv2.TM_CCOEFF','cv2.TM_CCOEFF_NORMED','cv2.TM_CCORR','cv2.TM_
CCORR_NORMED','cv2.TM_SQDIFF','cv2.TM_SQDIFF_NORMED']
for meth in methods:
    img = img2.copy()
    method = eval(meth)

    # Apply template Matching
    res = cv2.matchTemplate(img,template,method)
    min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)

    # If the method is TM_SQDIFF or TM_SQDIFF_NORMED, take minimum
    if method in [cv2.TM_SQDIFF, cv2.TM_SQDIFF_NORMED]:
        top_left = min_loc
    else:
        top_left = max_loc
```

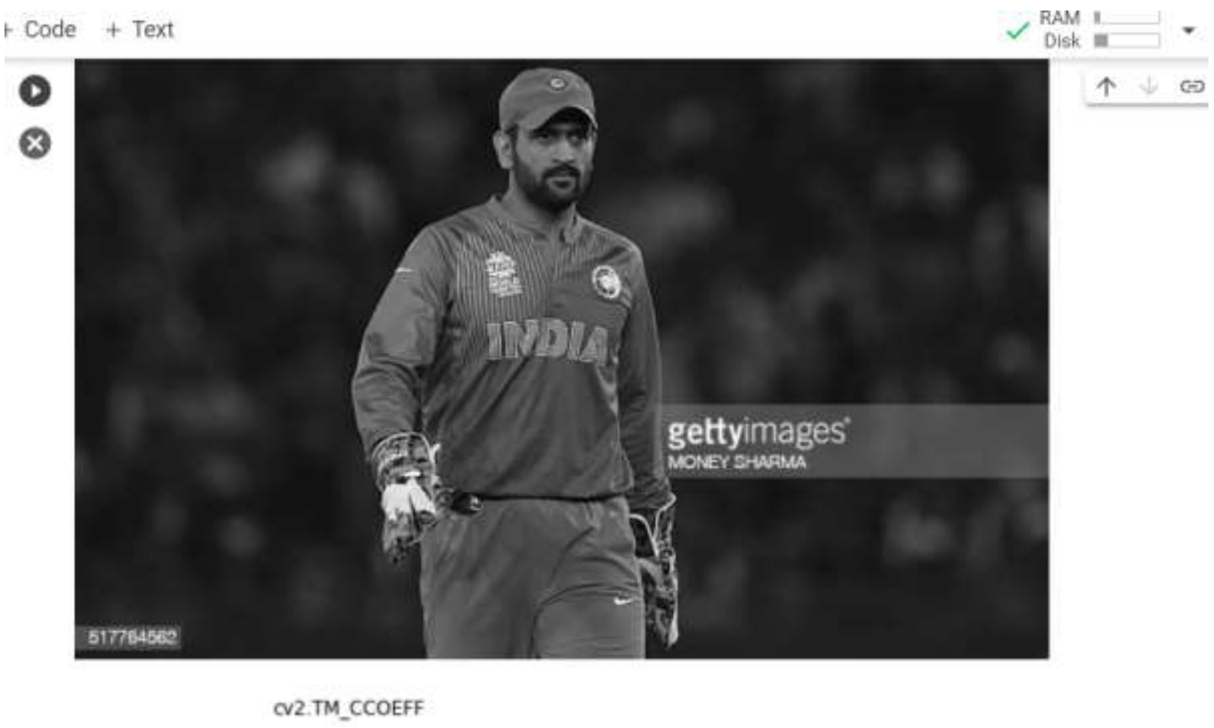
02_GAURAV BANE

```
bottom_right = (top_left[0] + w, top_left[1] + h)
cv2.rectangle(img,top_left, bottom_right, 255, 2)
plt.subplot(121),plt.imshow(res,cmap = 'gray')
plt.title('Matching Result'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(img,cmap = 'gray')

plt.title('Detected Point'), plt.xticks([]), plt.yticks([])
plt.suptitle(meth)
print("  ")
plt.show()
```

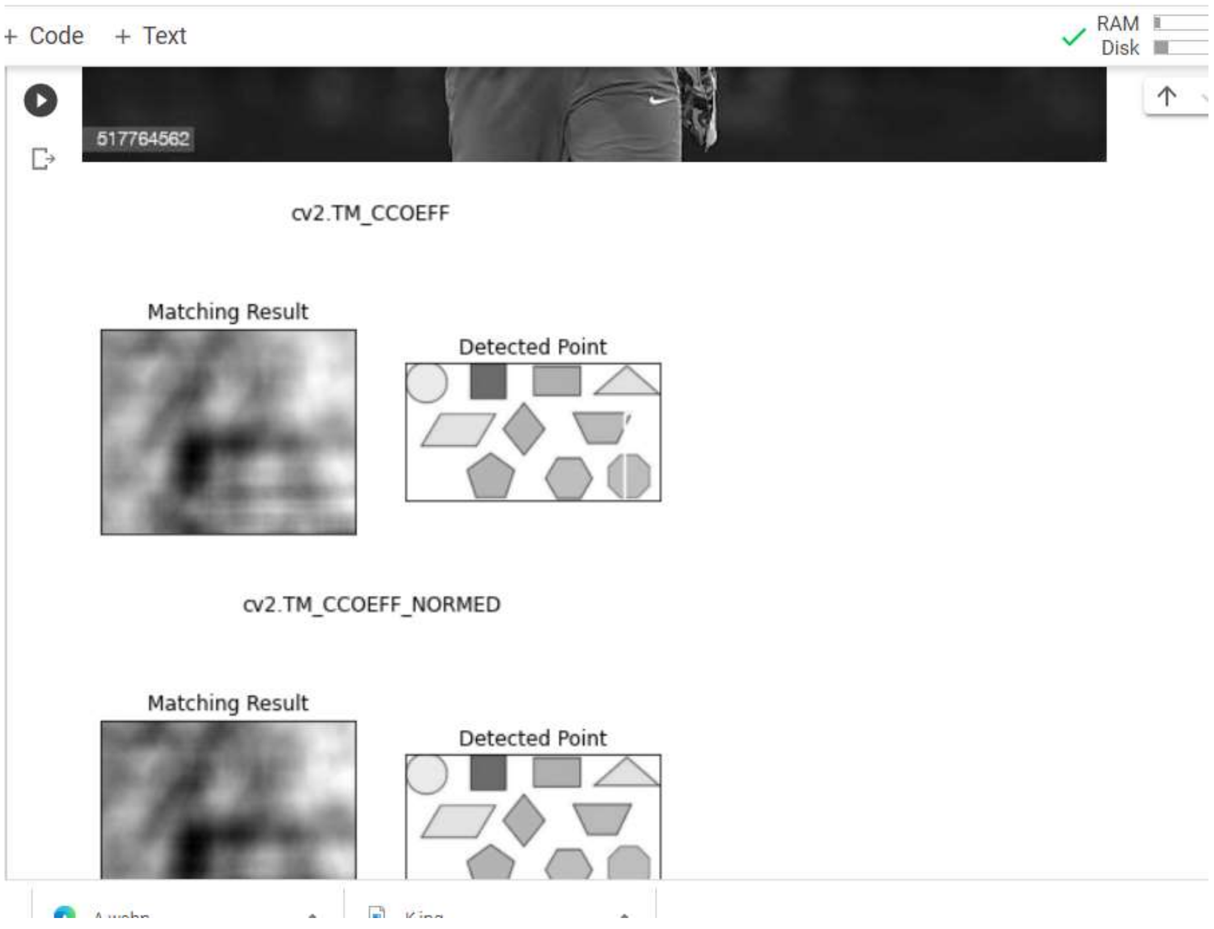


```
img = cv2.imread('CC.jfif',0)
img2=img.copy()
template=cv2.imread('dhoni.jpg',0)
cv2_imshow(template)
w,h = template.shape[::-1]
#All 6 methods for comparision in list
methods=['cv2.TM_CCOEFF','cv2.TM_CCOEFF_NORMED','cv2.TM_CCORR','cv2.TM_CCORR_NORMED','cv2.TM_SQDIFF','cv2.TM_S
for meth in methods:
    img = img2.copy()
    method = eval(meth)

    # Apply template Matching
    res = cv2.matchTemplate(img,template,method)
    min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)

    # If the method is TM_SQDIFF or TM_SQDIFF_NORMED, take minimum
    if method in [cv2.TM_SQDIFF, cv2.TM_SQDIFF_NORMED]:
        top_left = min_loc
    else:
        top_left = max_loc

    bottom_right = (top_left[0] + w, top_left[1] + h)
    cv2.rectangle(img,top_left, bottom_right, 255, 2)
    plt.subplot(121),plt.imshow(res,cmap = 'gray')
```

02_GAURAV BANE



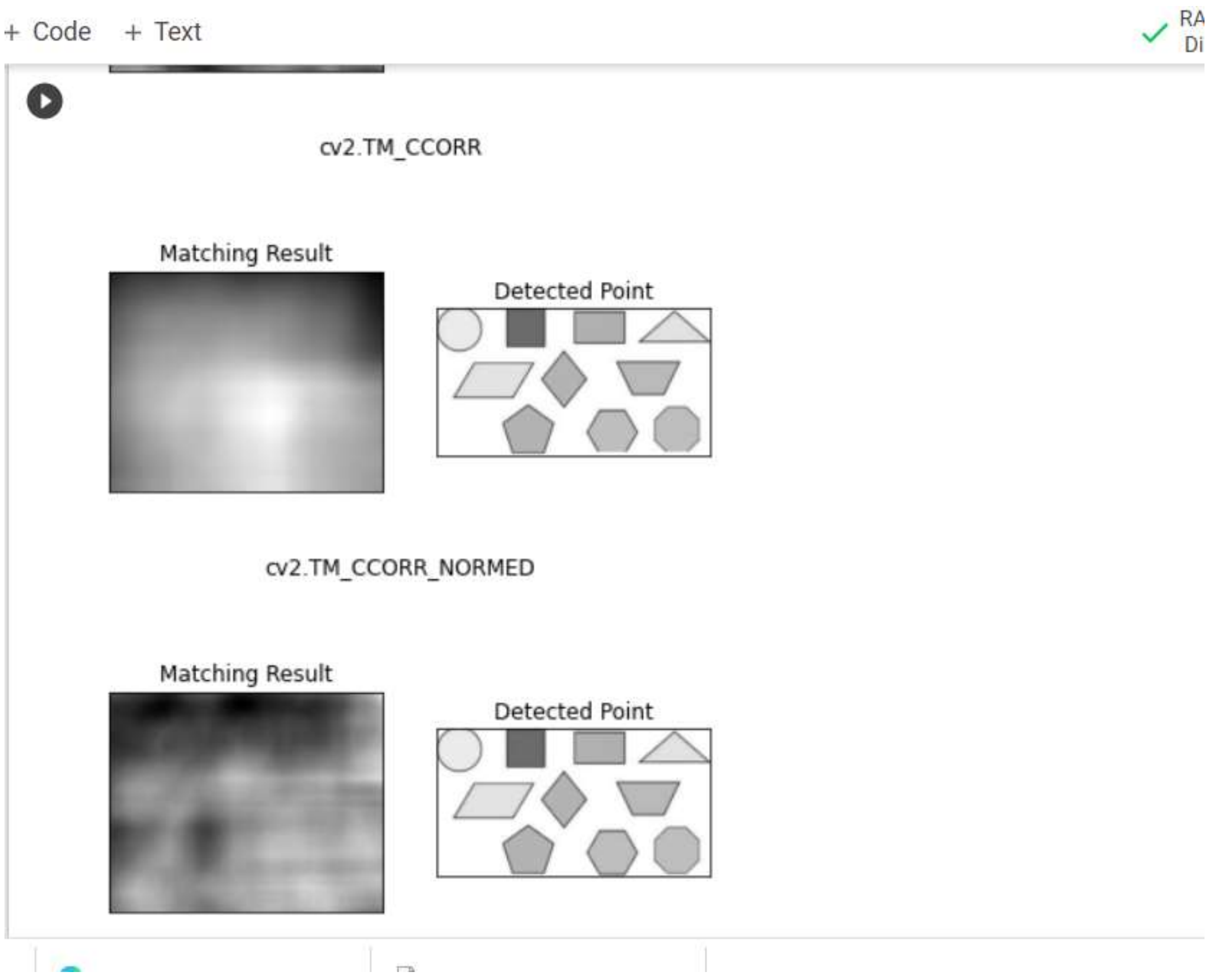
```
cv2.TM_CCOEFF
```



```
cv2.TM_CCOEFF
```

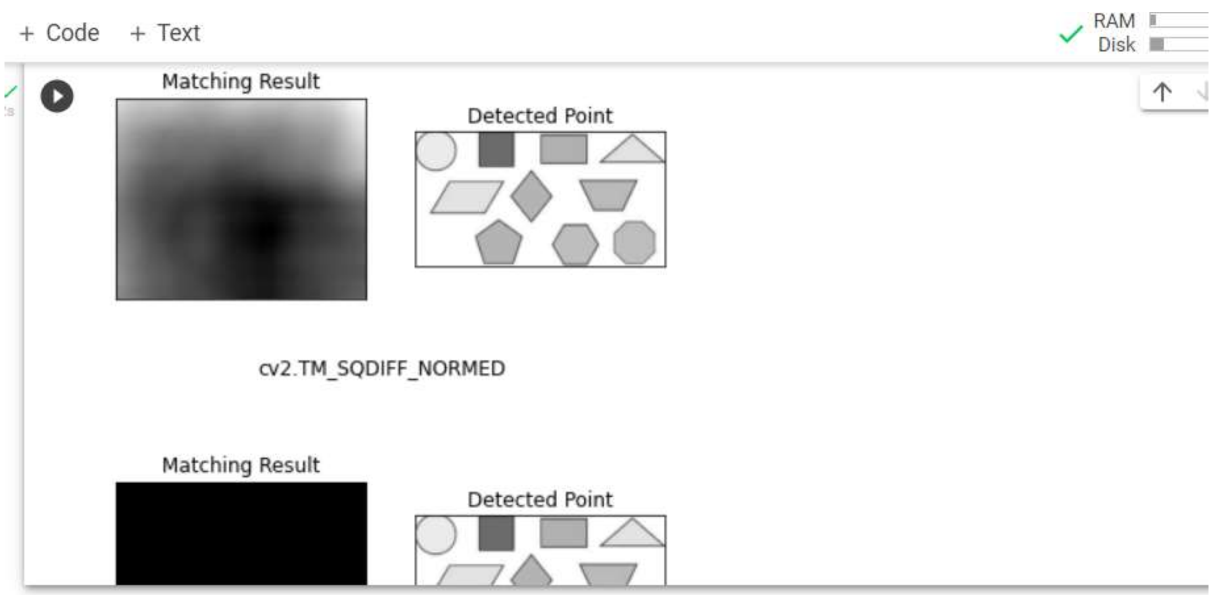
```
cv2.TM_CCOEFF_NORMED
```

02_GAURAV BANE



cv2.TM_CCORR

cv2.TM_CCORR_NORMED



cv2.TM_SQDIFF_NORMED