```
from google.colab import files
uploaded = files.upload()
```

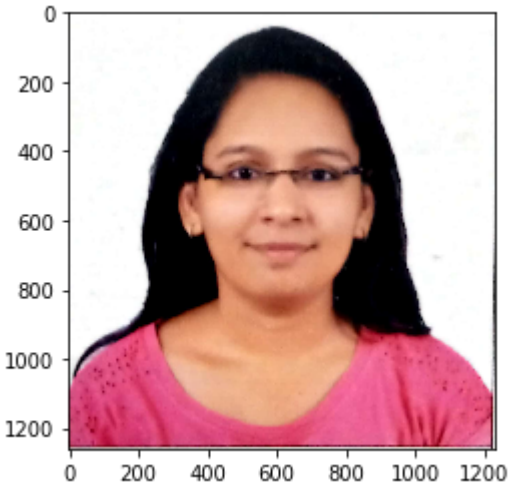
Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving juhipic.jpg to juhipic.jpg

TASK 1 Reading, Writing and Displaying Images

```
import numpy as np
import matplotlib.pyplot as plt
import cv2
%matplotlib inline

#reading the image

image = cv2.imread('juhipic.jpg')
image = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
#plotting the image
plt.imshow(image)

#saving image
cv2.imwrite('test_write.jpg',image)
```

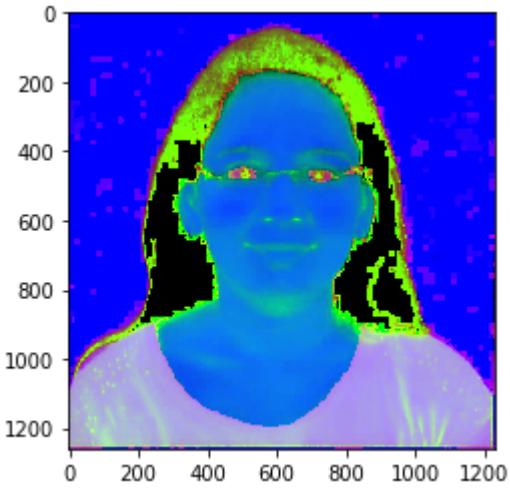
True



TASK 2 Changing Color Spaces

```
#import the required libraries
import numpy as np
import matplotlib.pyplot as plt
import cv2
%matplotlib inline
image = cv2.imread('juhipic.jpg')
#converting image to Gray scale
gray_image = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
#plotting the grayscale image
plt.imshow(gray_image)
#converting image to HSV format
hsv_image = cv2.cvtColor(image,cv2.COLOR_BGR2HSV)
#plotting the HSV image
plt.imshow(hsv_image)
```

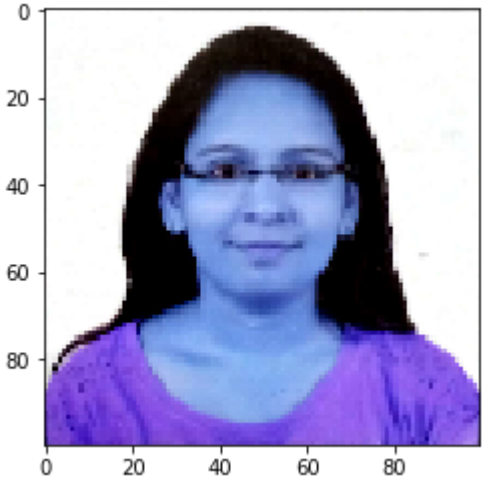
```
<matplotlib.image.AxesImage at 0x7f7dcb9de438>
```



TASK 3 Resizing Images

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
#reading the image
image = cv2.imread('juhipic.jpg')
#converting image to size (100,100,3)
smaller_image = cv2.resize(image,(100,100),interpolation=cv2.INTER_NEAREST)
#plot the resized image
plt.imshow(smaller_image)
```

```
<matplotlib.image.AxesImage at 0x7f7dca14f710>
```

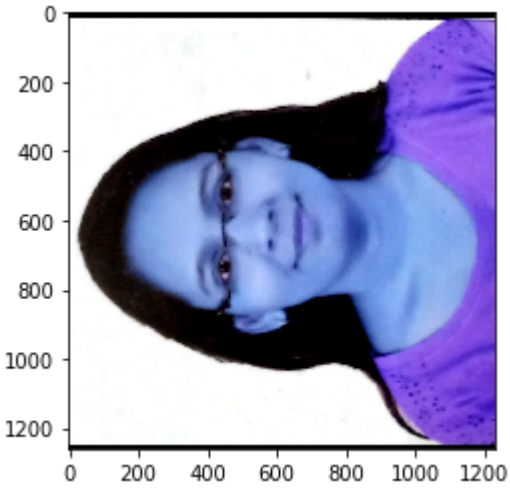


TASK 4 Image Rotation

```
#importing the required libraries
import numpy as np
import cv2
import matplotlib.pyplot as plt
%matplotlib inline
image = cv2.imread('juhipic.jpg')
```

```
rows,cols = image.shape[:2]
#(col/2,rows/2) is the center of rotation for the image
# M is the cordinates of the center
M = cv2.getRotationMatrix2D((cols/2,rows/2),90,1)
dst = cv2.warpAffine(image,M,(cols,rows))
plt.imshow(dst)
```

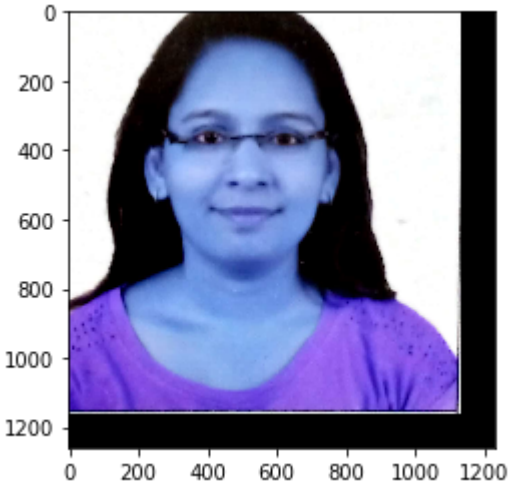
```
<matplotlib.image.AxesImage at 0x7f7dca11fe10>
```



TASK 5 Image Translation

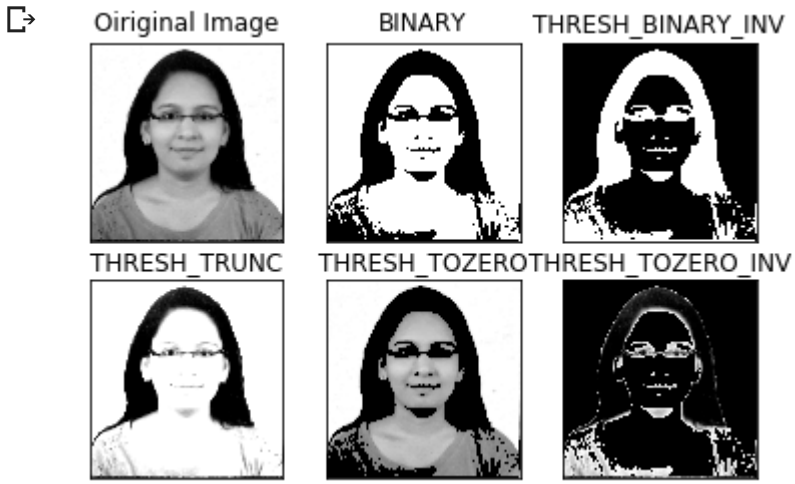
```
#importing the required libraries
import numpy as np
import cv2
import matplotlib.pyplot as plt
%matplotlib inline
#reading the image
image = cv2.imread('juhipic.jpg')
#shifting the image 100 pixels in both dimensions
M = np.float32([[1,0,-100],[0,1,-100]])
dst = cv2.warpAffine(image,M,(cols,rows))
plt.imshow(dst)
```

```
<matplotlib.image.AxesImage at 0x7f7dca085320>
```



TASK 6 Simple Image Thresholding

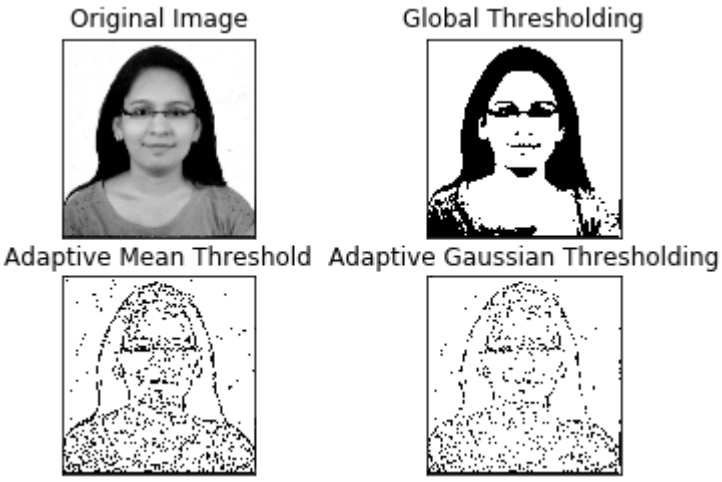
```
#importing the required libraries
import numpy as np
import cv2
import matplotlib.pyplot as plt
%matplotlib inline

#here 0 means that the image is loaded in gray scale format
gray_image = cv2.imread('juhipic.jpg',0)

ret,thresh_binary = cv2.threshold(gray_image,127,255,cv2.THRESH_BINARY)
ret,thresh_binary_inv = cv2.threshold(gray_image,127,255,cv2.THRESH_BINARY_INV)
ret,thresh_trunc = cv2.threshold(gray_image,127,255,cv2.THRESH_TRUNC)
ret,thresh_tozero = cv2.threshold(gray_image,127,255,cv2.THRESH_TOZERO)
ret,thresh_tozero_inv = cv2.threshold(gray_image,127,255,cv2.THRESH_TOZERO_INV)

#DISPLAYING THE DIFFERENT THRESHOLDING STYLES
names = ['Oiriginal Image','BINARY','THRESH_BINARY_INV','THRESH_TRUNC','THRESH_TOZERO','THRESH_TOZER
images = gray_image,thresh_binary,thresh_binary_inv,thresh_trunc,thresh_tozero,thresh_tozero_inv

for i in range(6):
    plt.subplot(2,3,i+1),plt.imshow(images[i],'gray')
    plt.title(names[i])
    plt.xticks([]),plt.yticks([])

plt.show()
```



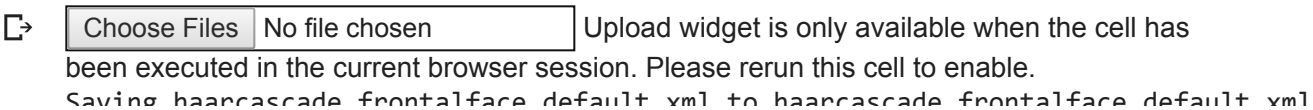
TASK 7 Adaptive Thresholding

```
#import the libraries
import numpy as np
import matplotlib.pyplot as plt
import cv2
%matplotlib inline

#ADAPTIVE THRESHOLDING
gray_image = cv2.imread('juhipic.jpg',0)

ret,thresh_global = cv2.threshold(gray_image,127,255,cv2.THRESH_BINARY)
#here 11 is the pixel neighbourhood that is used to calculate the threshold value
thresh_mean = cv2.adaptiveThreshold(gray_image,255,cv2.ADAPTIVE_THRESH_MEAN_C,cv2.THRESH_BINARY,11,2

thresh_gaussian = cv2.adaptiveThreshold(gray_image,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BIN

names = ['Original Image','Global Thresholding','Adaptive Mean Threshold','Adaptive Gaussian Thresho
images = [gray_image,thresh_global,thresh_mean,thresh_gaussian]

for i in range(4):
```

```
    plt.subplot(2,2,i+1),plt.imshow(images[i],'gray')
    plt.title(names[i])
    plt.xticks([]),plt.yticks([])

plt.show()
```

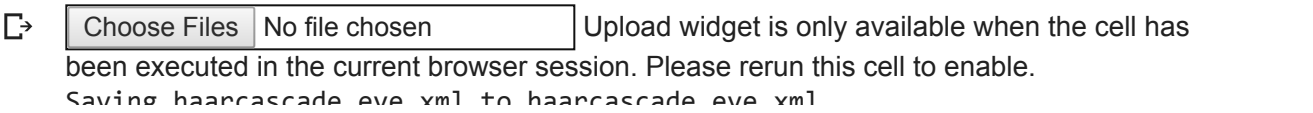



```
from google.colab import files
uploaded = files.upload()
```



Saving haarcascade_frontalface_default.xml to haarcascade_frontalface_default.xml

```
from google.colab import files
uploaded = files.upload()
```



Saving haarcascade_eye.xml to haarcascade_eye.xml

TASK 8 Face Detection

```
#import required libraries
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt
%matplotlib inline

#load the classifiers downloaded
face_cascade = cv.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv.CascadeClassifier('haarcascade_eye.xml')
#read the image and convert to grayscale format
img = cv.imread('juhipic.jpg')
gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
#calculate coordinates
faces = face_cascade.detectMultiScale(gray, 1.1, 4)
for (x,y,w,h) in faces:
    cv.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = img[y:y+h, x:x+w]
```

```
        eyes = eye_cascade.detectMultiScale(roi_gray)
        #draw bounding boxes around detected features
        for (ex,ey,ew,eh) in eyes:
            cv.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)
    #plot the image
    plt.imshow(img)
    k = cv2.waitKey(0)
    if k == 27:         # wait for ESC key to exit
        cv2.destroyAllWindows()
    elif k == ord('s'): # wait for 's' key to save and exit
        cv2.imwrite('face_detection.jpg',img)
        cv2.destroyAllWindows()
```