In [3]:
```
import face_recognition as fr
```

In [4]:
```
print(dir(fr))
```

```
['__author__', '__builtins__', '__cached__', '__doc__', '__email__', '__f
ile__', '__loader__', '__name__', '__package__', '__path__', '__spec__',
'__version__', 'api', 'batch_face_locations', 'compare_faces', 'face_dist
ance', 'face_encodings', 'face_landmarks', 'face_locations', 'load_image_
file']
```

In [5]:
```
#Load Image
my_image = fr.load_image_file(r"C:\Users\PC\Desktop\satpal.jpeg")
your_image = fr.load_image_file(r"C:\Users\PC\Desktop\ravi.jpeg")

#Image Encondings
my_image_encoding = fr.face_encodings(my_image)[0]
your_image_encoding = fr.face_encodings(your_image)[0]

#Compare Images
result = fr.compare_faces([my_image_encoding],my_image_encoding)

if(result[0]==True):
    print("It's me")
else:
    print("It's not me")
```

```
It's me
```

In [6]:
```
#Load Image
my_image = fr.load_image_file(r"C:\Users\PC\Desktop\satpal.jpeg")
your_image = fr.load_image_file(r"C:\Users\PC\Desktop\ravi.jpeg")

#Image Encondings
my_image_encoding = fr.face_encodings(my_image)[0]
your_image_encoding = fr.face_encodings(your_image)[0]

#Compare Images
result = fr.compare_faces([my_image_encoding],your_image_encoding)

if(result[0]==True):
    print("It's me")
else:
    print("It's not me")
```

```
It's not me
```

In [8]:
```
import cv2
# Load the cascade
face_cascade = cv2.CascadeClassifier(r'C:\Users\PC\Desktop\haarcascade_fron

# Read the input image
img = cv2.imread(r'C:\Users\PC\Desktop\test.jpg')

# Convert into grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Detect faces
faces = face_cascade.detectMultiScale(gray, 1.1, 4)

# Draw rectangle around the faces
for (x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x + w, y + h), (0, 0, 255), 4)
# Display the output
cv2.imshow('img', img)
cv2.waitKey()
```

Out[8]: -1

In [9]:
```
import cv2
# Load the cascade
face_cascade = cv2.CascadeClassifier(r'C:\Users\PC\Desktop\haarcascade_eye.

# Read the input image
img = cv2.imread(r'C:\Users\PC\Desktop\test.jpg')

# Convert into grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Detect faces
faces = face_cascade.detectMultiScale(gray, 1.1, 4)

# Draw rectangle around the faces
for (x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x + w, y + h), (0, 0, 255), 4)
# Display the output
cv2.imshow('img', img)
cv2.waitKey()
```

Out[9]: -1

In [13]:

```
import cv2

face_cascade = cv2.CascadeClassifier(r'C:\Users\PC\Desktop\haarcascade_fron

# capture frames from a camera
cap = cv2.VideoCapture(0)

# loop runs if capturing has been initialized.
while 1:

    # reads frames from a camera
    ret, img = cap.read()

    # convert to gray scale of each frames
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # Detects faces of different sizes in the input image
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)

    for (x,y,w,h) in faces:
        # To draw a rectangle in a face
        cv2.rectangle(img,(x,y),(x+w,y+h),(255,255,0),2)

    # Display an image in a window
    cv2.imshow('img',img)

    # Wait for Esc key to stop
    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break

# Close the window
cap.release()

# De-allocate any associated memory usage
cv2.destroyAllWindows()
```

In [ ]: