In [1]:
```
#Project  => face Recongnition.... by (YOGESH BAGHEL)
#Python script to capture images from web cam video streame...
#Face Recongnize to store in array When the same face recongnize it's give
# the name of person"
```

In [2]:
```
#===================(Require Modules are..) ==================================
#opencv => To recognize from image, web cam, video etc
#numpy => array to store the face data"
```

In [9]:
```
#STEP-1 => Read and show video streame and capture images.....



import cv2
import numpy as np

#intialise the camera
cap = cv2.VideoCapture(0)

#Face Detection
face_cascade = cv2.CascadeClassifier("haarccascade_frontalface_alt.xml")

    #CascadeClasifier method in cv2 module support the loading haar cascade file.
    #Haar Cascade is a machine learning-based approach where a lot of positive and negative images are
    #used to train the classifier. \n",
    #   => its capture the face image, and inside eye's images and mouth's images..
```

```
[ WARN:0@0.145] global /io/opencv/modules/videoio/src/cap_v4l.cpp (889) open VIDEOIO(V4L2:/dev/video0): can't open cam
era by index
```

In [ ]:
```
#=========================Step-3 => Flatten the largest face and store in numpy
arrray..=========================


```

```
#Create an empty list to store the coordinate of face...

face_data = []

#store that list at location.
data_path = "/Desktop/"

filename = input("Enter your name who scann the face =")

skip=0

while True:


#Read information from webcam
    ret,frame = cap.read()

    #Due to any reasn web cannot be capture the images i.e\n
    if ret==False:
        continue


     #STEP=2 => Detect face and Show Boundary Box (haarcascade)..


      #==================================(Step-2)=====================================

    faces = face_cascade.detectMultiScale(frame,1.3,5)


        #==================================(Step-3)=====================================
```

```
    # now, convert Image into grayscale....
                #WHY => Because OUr image has RGB has three diffrrent layer and gray hass one layer
                #        from 0-255, its reconginize each pixel very well...


    gray_frame = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)


       # if multiple faces are occur then your find largest face => {area of Rectangle} => sorted the
width and height
       # on the bases of area....

    faces = sorted(faces,key=lambda f:f[2] *f[3],reverse=True)
    #                                        index->    [0,1,2,3]
    # pick the last face (because its larger face of area [x,y,w,h] => f[2] * f[3]  => (w*h) area

    for face in faces:
        x,y,w,h=face
        cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,255),2)

    #now,Create a new window to display only face  =>  need to extract crop the face images

        offset =10
        face_section = frame[y-offset:y+h+offset, x-offset:x+w+offset]
        face_section = cv2.resize(face_section, (100,100))

        skip+=1
        if skip%10==0:
            face_data.append(face_section)
            print(len(face_data))
```

```
    cv2.imshow("Frame", frame)
    cv2.imshow("Face section",face_section)


   #==================================(Step-4)=====================================

face_data = np.asarray(face_data)
face_data = face_data.reshape((face_data.shape[0],-1))
print(face_data.shape)


#save this array data in at location
np.save(data_path+filename+'npy',face_data)
print("Data Successfully save at"+data_path+filename+'npy')
        # if person enter => A then => Desktop/A.npy
        # if person enter => B then => Desktop/B.npy
        # if person enter => C then => Desktop/C.npy


 #detectMultiScale() =>  Detect object of difrent sizes in input images.eturn a list of position
 #of rectangle on the face...

#    (x,y,w,h)
#     ----------
#     |        |
#     |        |       => Its has (x,y) coordinate for rectangle and width and height.
#     |        |       => in rectangle inside the face which size detect by detectMultiScale()..
#     |        |       => Ex= face1 ->(10,20,30,40) and face2 -> (20,50,30,40)
#     |        |
#     ----------
```

```
    #print(faces)

    # #Store every 10th face..
    # if(skip%10==0):
    #     pass

    # for face in faces:
    #     x,y,w,h=face
    #     cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,255),2)


        cv2.imshow("Frame",frame)
        #imshow() => used to add iamge in window..Itself adjust to size of images

  #====================================(Step-2) complete==================================================

#when the user press 'q' key then,break the program
        key_press = cv2.waitkey(1) & OxFF
        if key_press==ord('q'):
            break

    #ord('q') => return the Unicode code point of 'q'
    #cv2.waitkey(1) => return the 32-bits integer corresponding to the user press key
    # & OxFF => is bit mask which sets tthe 24-bits to zero (its use to check correspondin key)

    cap.release()
    cv2.destroyAllWindow()
    #=========================================STEP-1
COMPLETE============================================
```

In [ ]:

```
# Completed the Backend work.....
```