***OpenCV*:**

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning library. This was built to give a common structure for computer vision applications and to influence the use of machine interpretation in the commercial products like facial recognition which is extensively use in today’s world. OpenCV has more than 2,500 optimized algorithms. These algorithm can be used to detect and identify faces, objects, track moving objects in a video or in an image, follow eye movements etc.

OpenCV has been written in C++ and has templated interface that work with STL containers (Standard Template Library: it is a set of C++ template classes to provide common programming data structure and functions like list, array, stack etc). OpenCV has C++, Python, Java and Matlab interfaces and supports Windows, Linux, Mac OS and also Android

**Cascade:**

A Haar Cascade is a classifier which is used to detect the object for which it has been trained for. The Haar cascade is trained by superimposing the positive image over a set of images. This type of training is generally done on a server and on various stages. Better results are obtained by using high quality images and increasing the amount of stages for which the classifier is trained for. This cascade makes it easier to build a model. One just needs to pre-define the Haar cascade which are available on [github](https://github.com/opencv/opencv/tree/master/data/haarcascades). Moreover one can make their own cascade file.

**So, Let’s Start :**

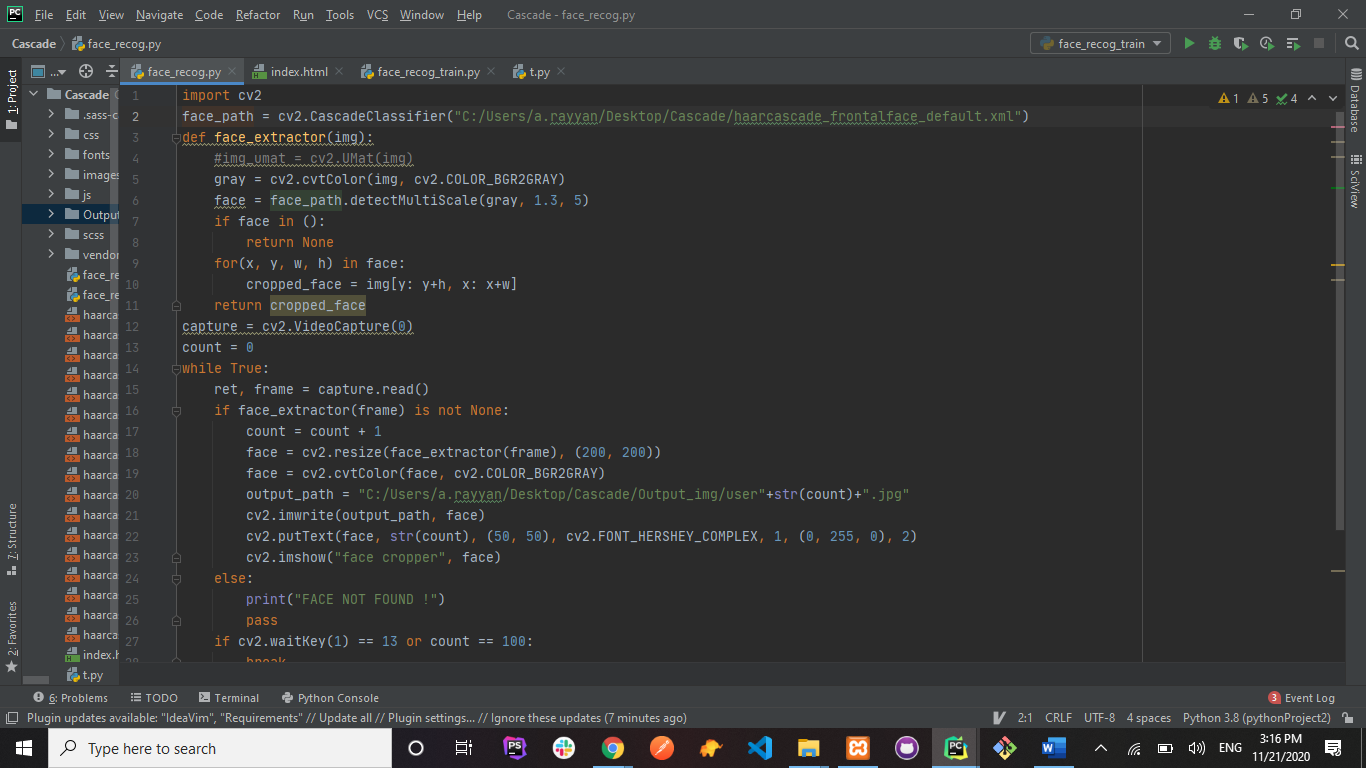
Before starting this, my suggestion is to break this project into different parts or do it in steps so that it becomes easy to understand. Therefore, I have divided this into 2 major parts so that it becomes easy and hassle free.

**First Part: (Data Collection)**

The first and the major part is to collect samples of data on which this model needs to be trained.

**Steps:**

1. Defining Path : In this we first give the path of the file where the cascade file named “haarcascade\_frontalface\_default.xml” has been saved if you can download it from [github](https://github.com/opencv/opencv/tree/master/data/haarcascades). This cascade file will help your model classify your facial expression and will help in recognizing the face for which it has been trained for.
2. Face Extraction : In this we basically convert the color of the captured images to gray because for many applications based on image processing, color information doesn’t help us identify important edges or other features. Also, for learning image processing, its better to understand multichannel processing rather than starting with full color imaging and missing all important insight that can learned from single channel processing. Moreover, in grayscale images, the watershed algorithm is fairly easy to conceptualize because we can think of the two spatial dimensions and one brightness dimension as a 3D image with hills, valleys, catchment basins, ridges, etc.
3. Multi Scaling : After the color transition we use detectMultiScale(), this basically detects objects of different sizes in the input image and the detected objects are returned as list of rectangle. This method has three main parameters which involves image, scale factor and nearest neighbors.
4. Cropping Image : this is the very crucial part as in this we just crop the image so that we could get the face of the object.
5. Reading : Now after cropping the face we start reading. One thing that you must have noticed is that I used two variables to unpack the read function. It is because, this function always returns two values one Boolean and second the coordinates .
6. Resizing and saving the Output : The resize function here resizes the cropped image into the desired set of frames, also declaring the location of the file where the training data is going to be saved (“cv2.putText” is used to add text on the screen along with the video and “cv2.FONT\_HERSHEY\_COMPLEX” is the name of the font used just like Arial etc)
7. Wait Key : This is a method which is used to kill the process if any key is pressed it only takes ASCII values here 13 is an ASCII value.
8. Releasing and Destroying the windows : This is probably the most important thing to do because this will destroy all the windows which are open during the process of capturing the data and if you do not use this the camera window will not shut down and you will have to restart the system all over again and in some cases restart doesn’t even work.



The aforementioned code will collect 100 sample data and then store it in the predefined folder (output\_path).

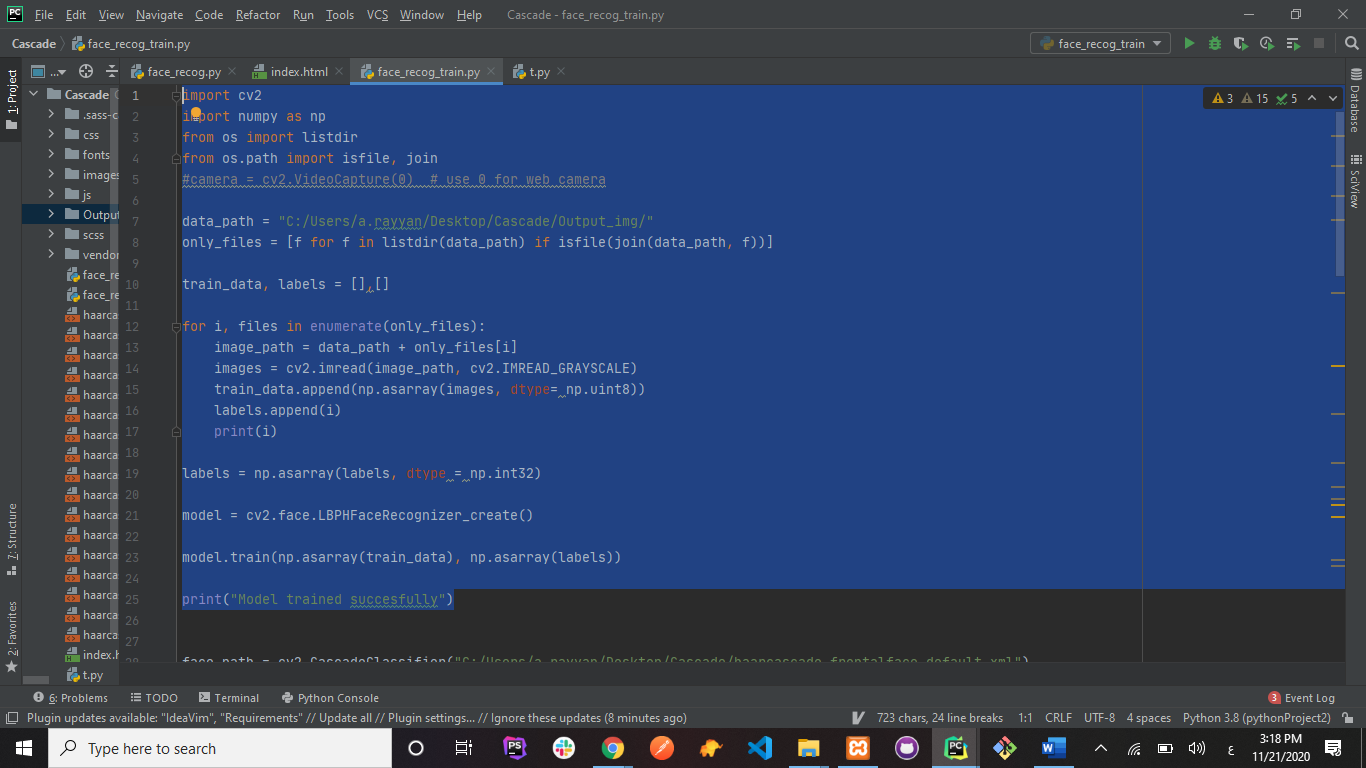
## ****Second Part (Training and Predicting the Data) 😊()****

**Training :**

In this part we will train the model on the data that we have collected and based on the training our model will predict (based on confidence), whether the user is the same user or its somebody else; same as any other face lock of a mobile phone.

Before beginning the training we first need to get the data from the output file and then convert it into grayscale. After this color transition we then append the data into train\_data.

The model which determines the face of the subject is already defined which is “cv2.face.LBPHFaceRecognizer\_create()”. Therefore we use this as our primary model for training the sample data.



**Predicting ()**

This is the last and final step. In this we use test our model and calculate the confidence and based on this confidence we our model will determine and predict whether the user is the same for which the data has been trained or not. Few steps of this part are similar to the steps listed in part one(Defining path, Face Extraction, Multi scaling and Resizing). The only thing which is different is calculating confidence. First we set a threshold value, if our model predicts and the prediction value comes out to be less than the threshold value then only the confidence is calculated and If the value of confidence turns out to be greater than 80 then only the message “UNLOCKED” will pop otherwise the device will remain locked for every other faces.

