**Automatic Attendance System**

**Libraries used:**

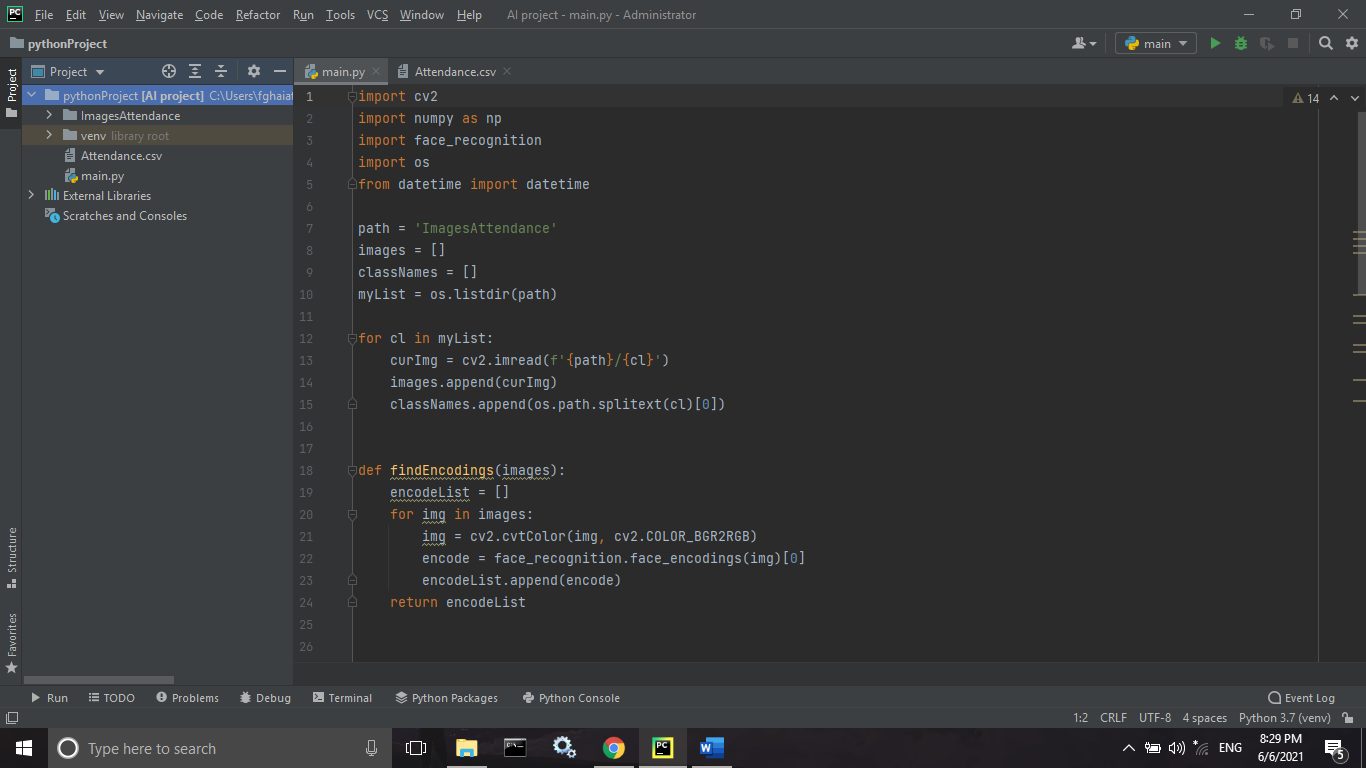
* **face\_recognition:** used for real-time face recognition from a webcam or a camera.
* **cv2:** used for image processing, video capture, and analysis (face features, face detection, and object detection)
* **os:** import images in a given folder at one, interacting with operating system, interacting with file system.
* **datetime:** supplies classes to work with date and time.
* **numpy:** working with multidimensional arrays.

**Programs used:**

* Cmake
* Pycharm

**Codes Functionalities:**

**1.Importing images:**

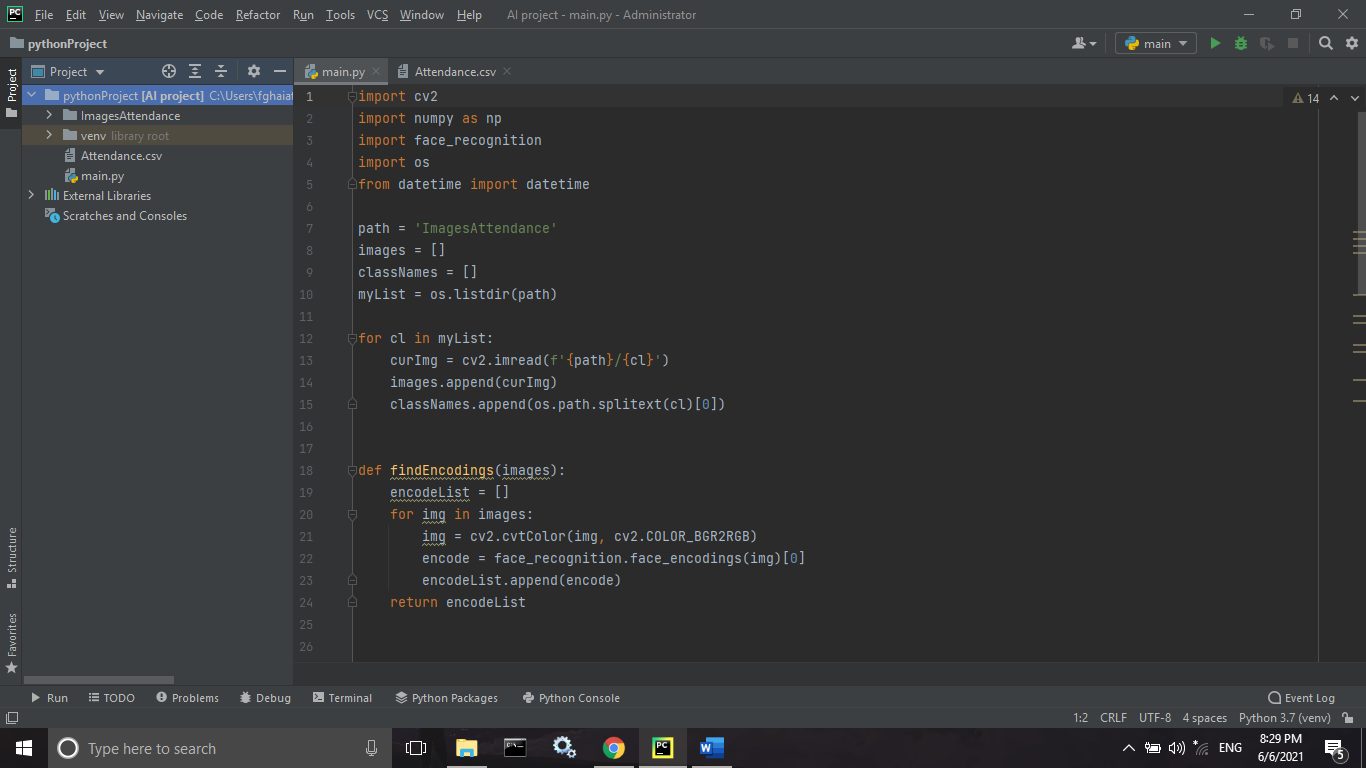


We have multiple images, importing them individually can become messy.

Therefore, we will write a script to import all images in a given folder at once.

For this we will need the os library so we will import that first.

We will store all the images in one list and their names in another.

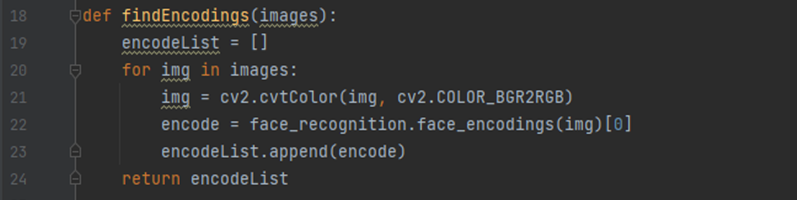
****

Now that we have a list of images we can iterate through those and create a corresponding encoded list for known faces.

To do this we will create a function.

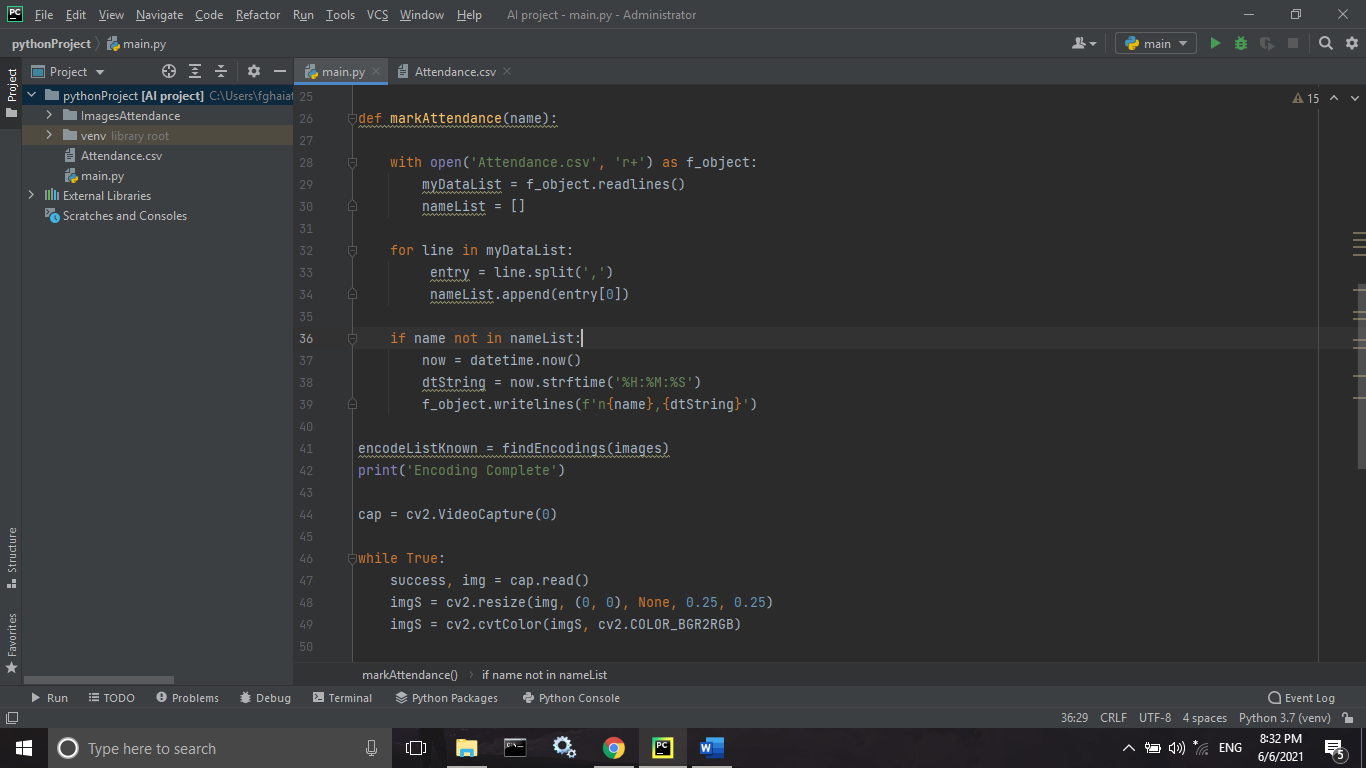
We will first convert it into RGB and then find its encoding using the face\_encodings() function.

Then we will append each encoding to our list.

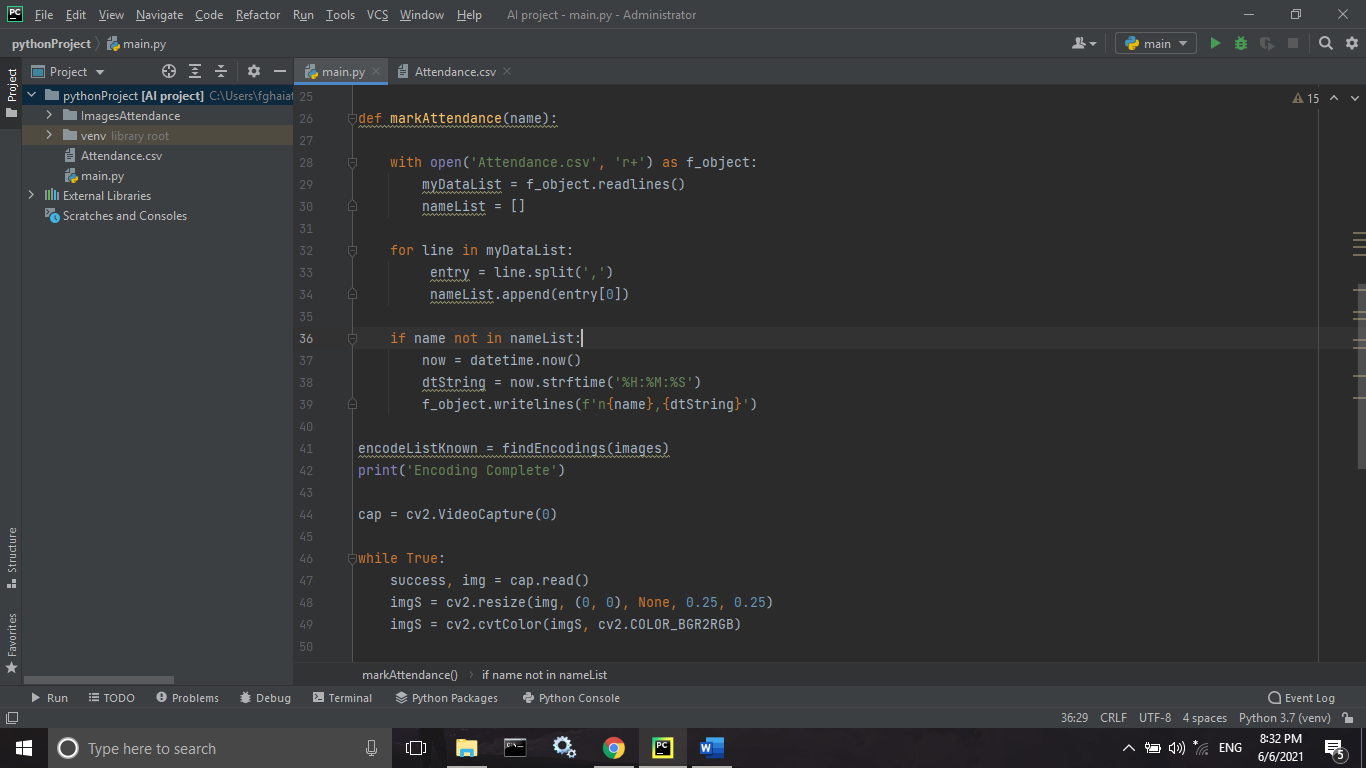


Now we can simply call this function with the images list as the input arguments.

**2.Webcame Images:**

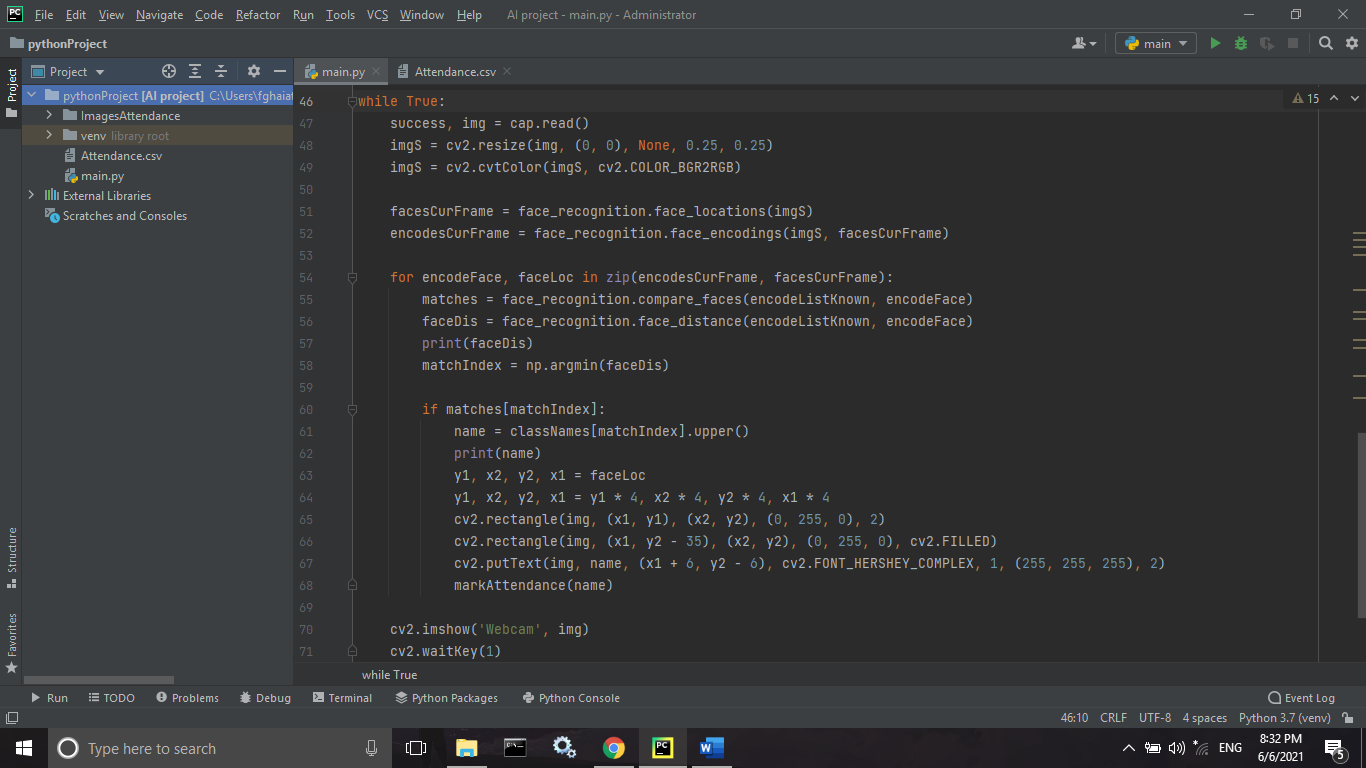


Now we must create a video capture object so that we can grab frames from the webcam.



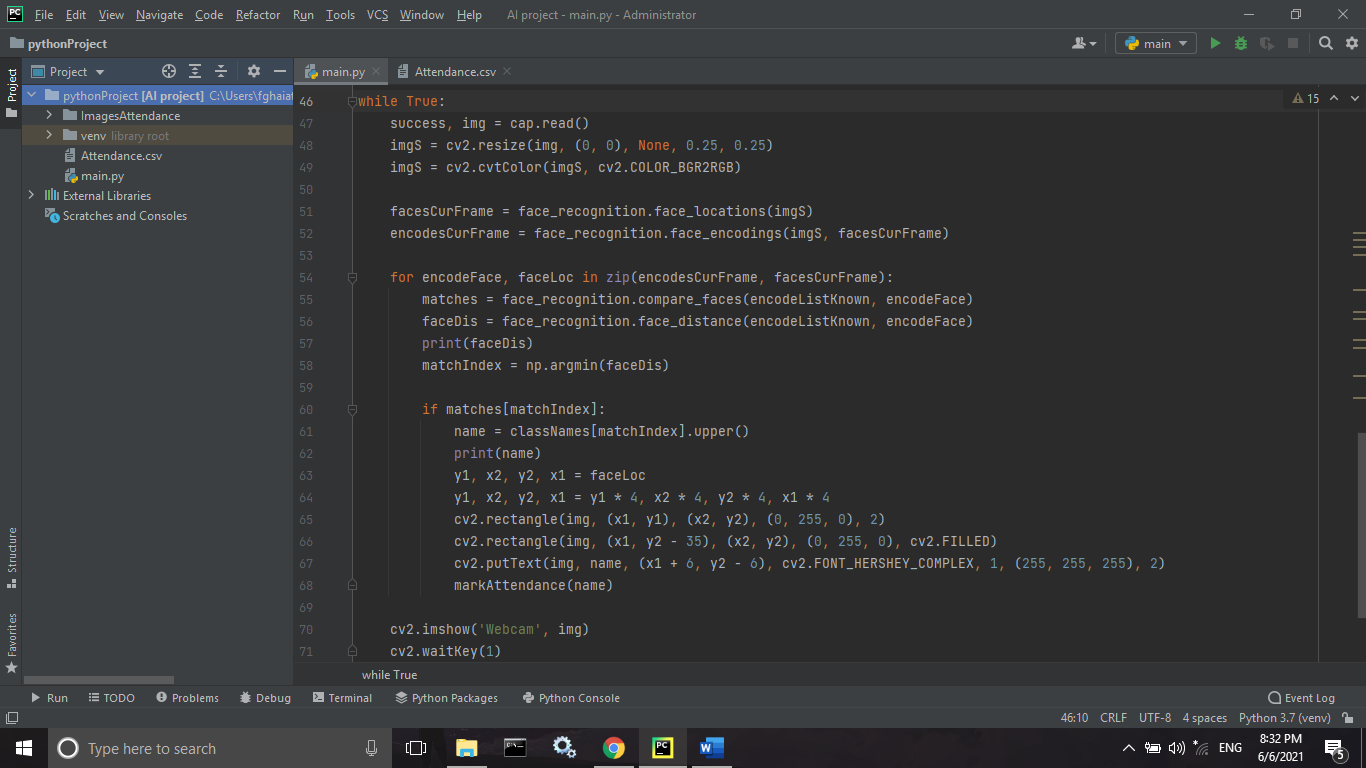
First we will read the image from the webcam and then resize it to quarter the size. This is done to increase the speed of the system. Even though the image being used is 1/4 th of the original, we will still use the original size while displaying. Next we will convert it to RGB.

**3.Webcam Encoding:**

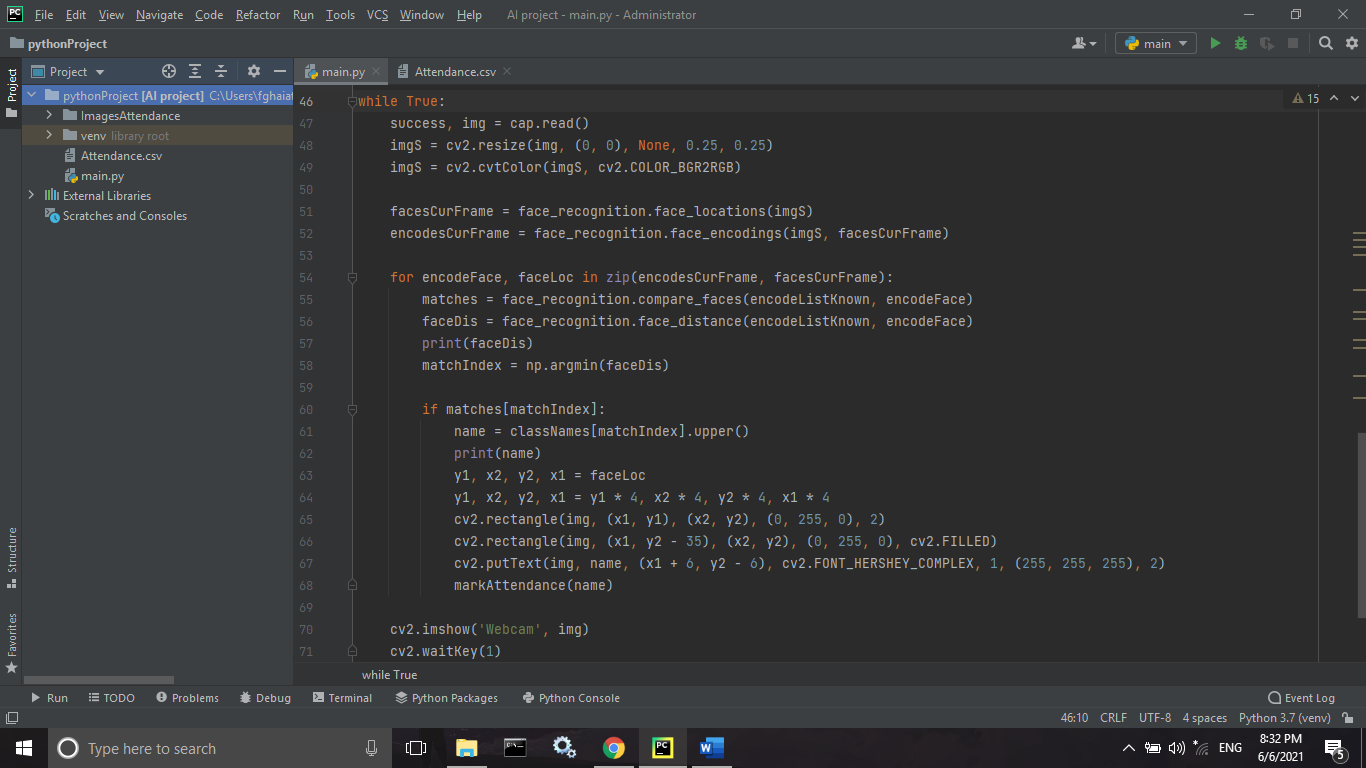
****

Once we have the webcam frame we will find all the faces in our image. The face\_locations function is used for this purpose. Later we will find the face\_encodings as well.

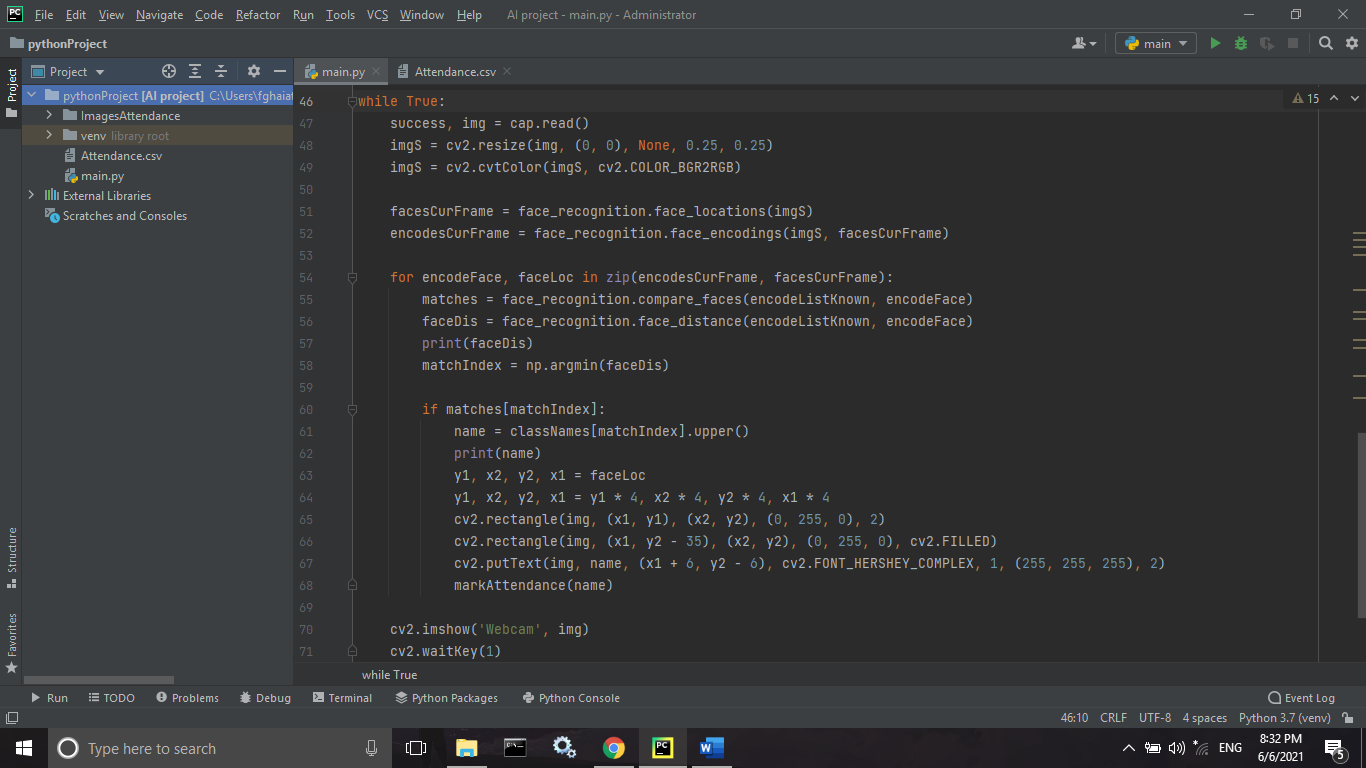
**4.Finding Matches:**

****

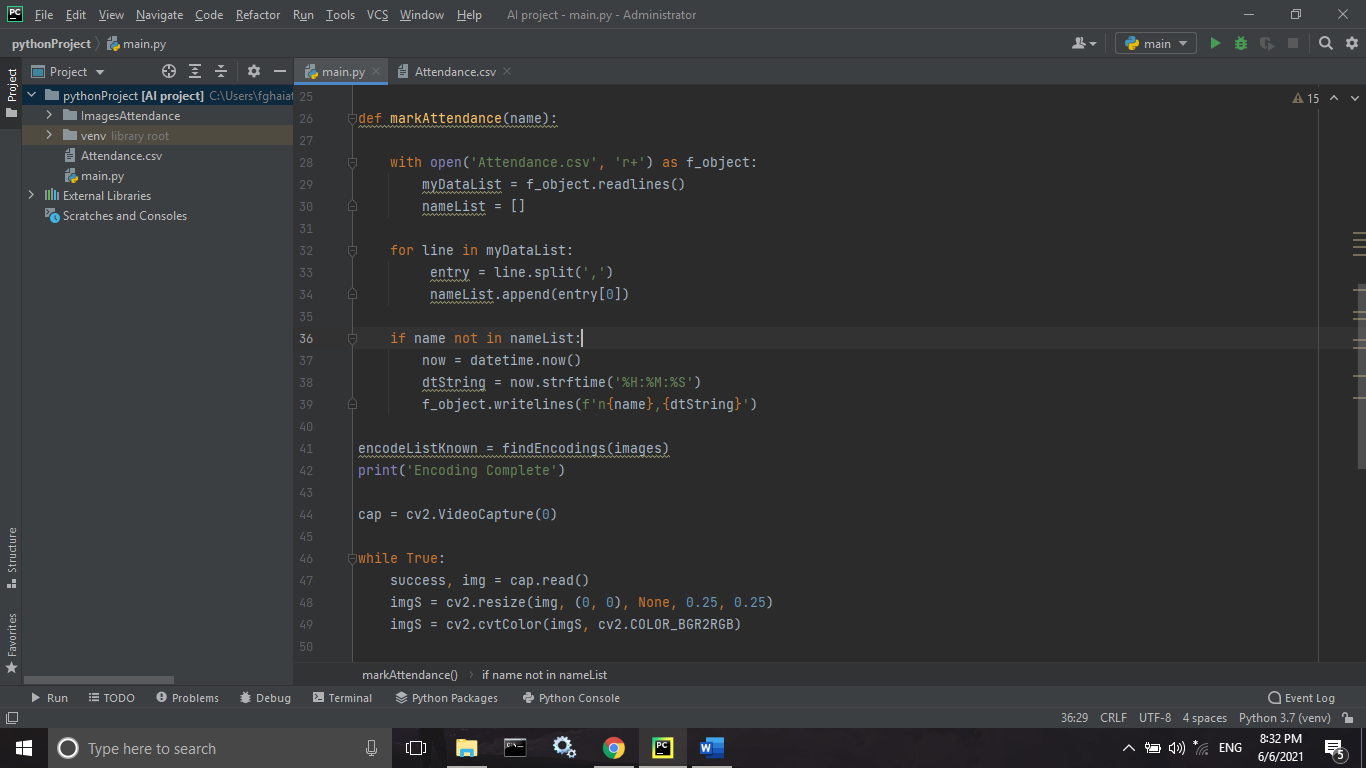
Now we can match the current face encodings to our known faces encoding list to find the matches. We will also compute the distance. This is done to find the best match in case more than one face is detected at a time.

****

Once we have the list of face distances, we can find the minimum one, as this would be the best match.

****

Now based on the index value we can determine the name of the person and display it on the original Image and send the name to the attendance.

**5.** **Marking Attendance:**

Lastly, we are going to add the automated attendance code. By writing a function that requires only one input which is the name of the user.

First, we open our Attendance file which is in csv format. Then we read all the lines and iterate through each line using a for loop.

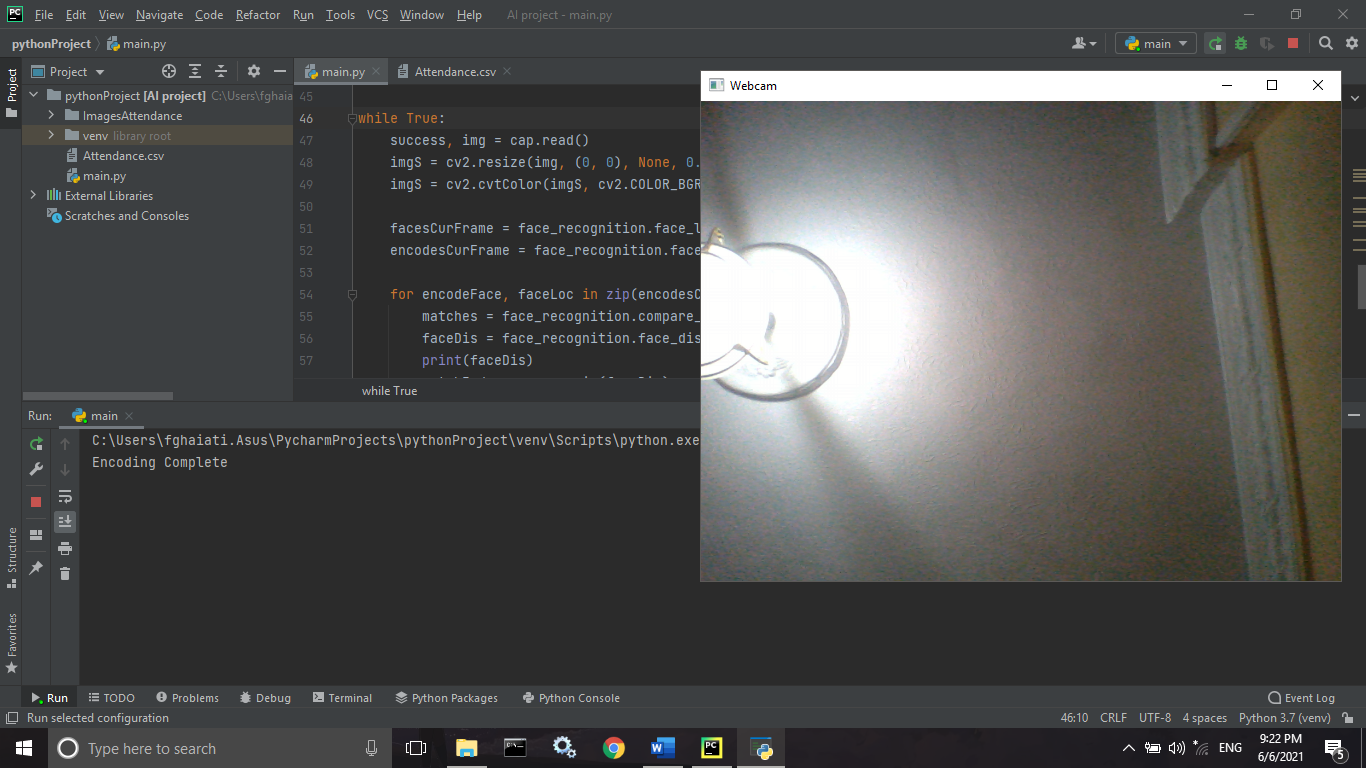
Next, we can split using comma ‘,’. This will allow us to get the first element which is the name of the user.

If the user in the camera already has an entry in the file, then nothing will happen.

On the other hand, if the user is new then the name of the user along with the current time stamp will be stored using the datetime class in the date time package to get the current time.

**Implementation of the Project**

As we run the project it will automatically open the web cam and detects the face and adding it to the attendance.



Then it will print the name of the person and will send their name and time to the csv file

