Demo - Reconocimiento facial

Este proyecto está desarrollado bajo la **versión 3.8 de python** por medio del entorno de **Anaconda para Windows**, además se hará uso de la última versión de **openCV** en conjunto con el algoritmo **LPBH (Local Binary Pattern Histogram) (**<https://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html#local-binary-patterns-histograms>**)**  para la creación de un modelo para el reconocimiento facial.

**Requerimientos:**

Para el reconocimiento facial es necesario tener instalado lo siguiente:

* Python 3.X
* OpenCv
* NumPy

**Instalación del entorno de trabajo:**

* **Anaconda:** <https://docs.anaconda.com/anaconda/install/windows/>
* **OpenCV:** pip install opencv-python

pip install opencv-contrib-python

* **NumPy:** pip install numpy
* **Modelo Detección de rostros:** Descargar archivo "haarcascade\_frontalface\_alt.xml" desde https://github.com/opencv/opencv/tree/master/data/haarcascades

**ESTRUCTURA DEL PROYECTO**

Dentro del proyecto se encuentran 5 carpetas con la siguiente estructura:

* **Img\_train**: Almacena las imágenes conocidas con las que se entrena un modelo de reconocimiento facial, generando un dataset de 30 imágenes por usuario.
  + User.1.1.jpg
  + User.1.2.jpg
  + User.2.1.jpg
  + User.2.2.jpg
* **Video\_train**: (Opcional) se almacena los videos desde donde se desde extraer las imágenes conocidas para almacenar en el dataset **img\_train**.
  + Video\_01.mp4
  + Video\_02.mp4
* **Img\_test**: Aquí se almacenarán las imágenes que servirán de prueba para el recnocimiento facial.
  + Image\_01.jpg
* **Video\_test**: Aquí se almacenarán los videos de archivos que se ingresaron para su reconocimiento.
  + Video.mp4
* **Face\_model**: Aquí se almacena el modelo obtenido para el reconocimiento facial en un archivo con extensión yml.

Los archivos escritos en Python permiten:

* **Get\_images.py**: Realiza la captura de imágenes de los rostros obtenidos desde un archivo de video o webcam por cada persona con la que se desea entrenar el modelo.
* **get\_model.py:** Genera el modelo de reconocimiento facial y lo almacena en un archivo con extensión yml.
* Los archivos **recognize\_faces\_image.py**, **recognize\_faces\_video.py** y **recognize\_faces\_video\_file.py** contienen la lógica del reconocimiento facial.

**Flowchart**

Training

**FLOWCHART**

Dataset



Modelo de clasificación

Vector de características

Extracción de características



Video Test

Vector de características

Extracción de características



Predicción

**OUTPUT**

**EJECUCIÓN DEL PROYECTO**

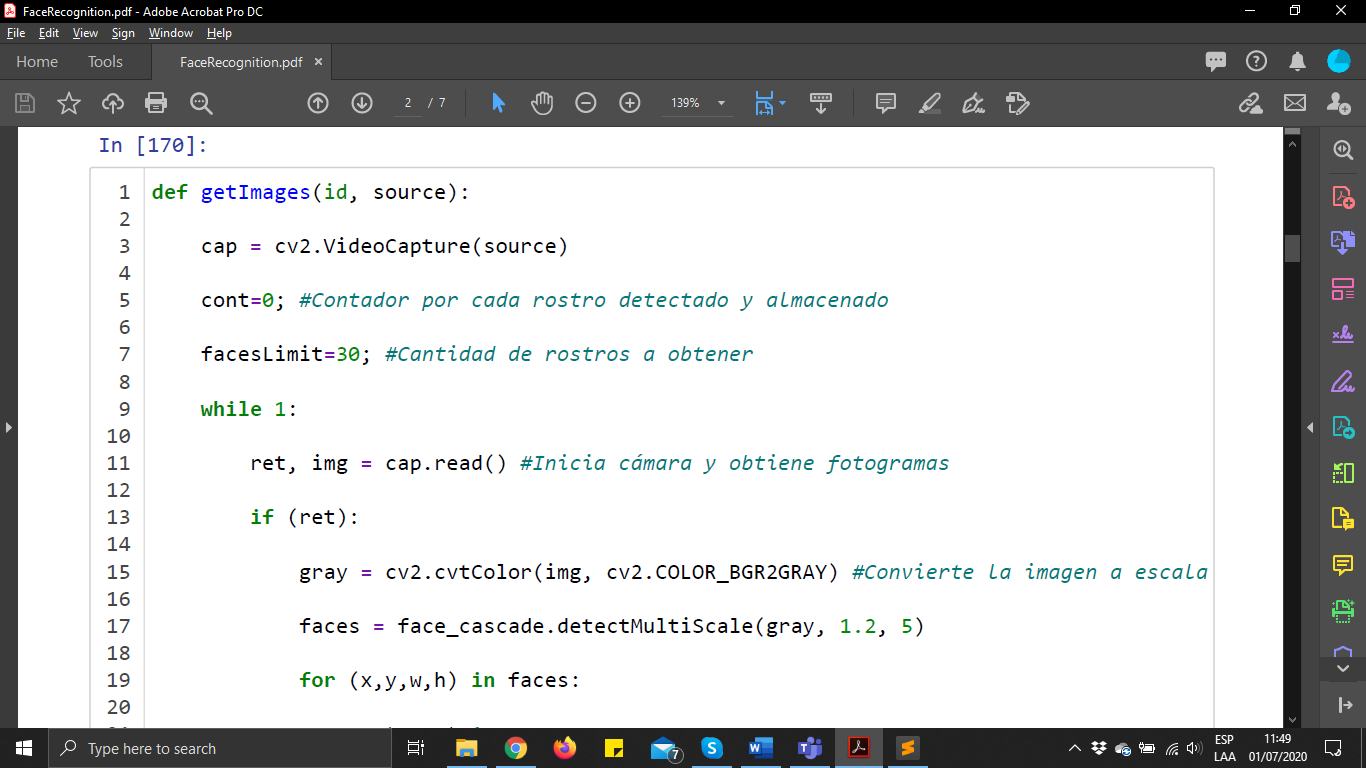
**Captura de imágenes para entrenamiento**

El primer paso es generar el dataset para el modelo de reconocimiento facial, para el se propone el archivo **get\_images.py**, donde se tiene la posibilidad de extraer las imágenes desde la webcam o un archivo de video por cada rostro de la persona.

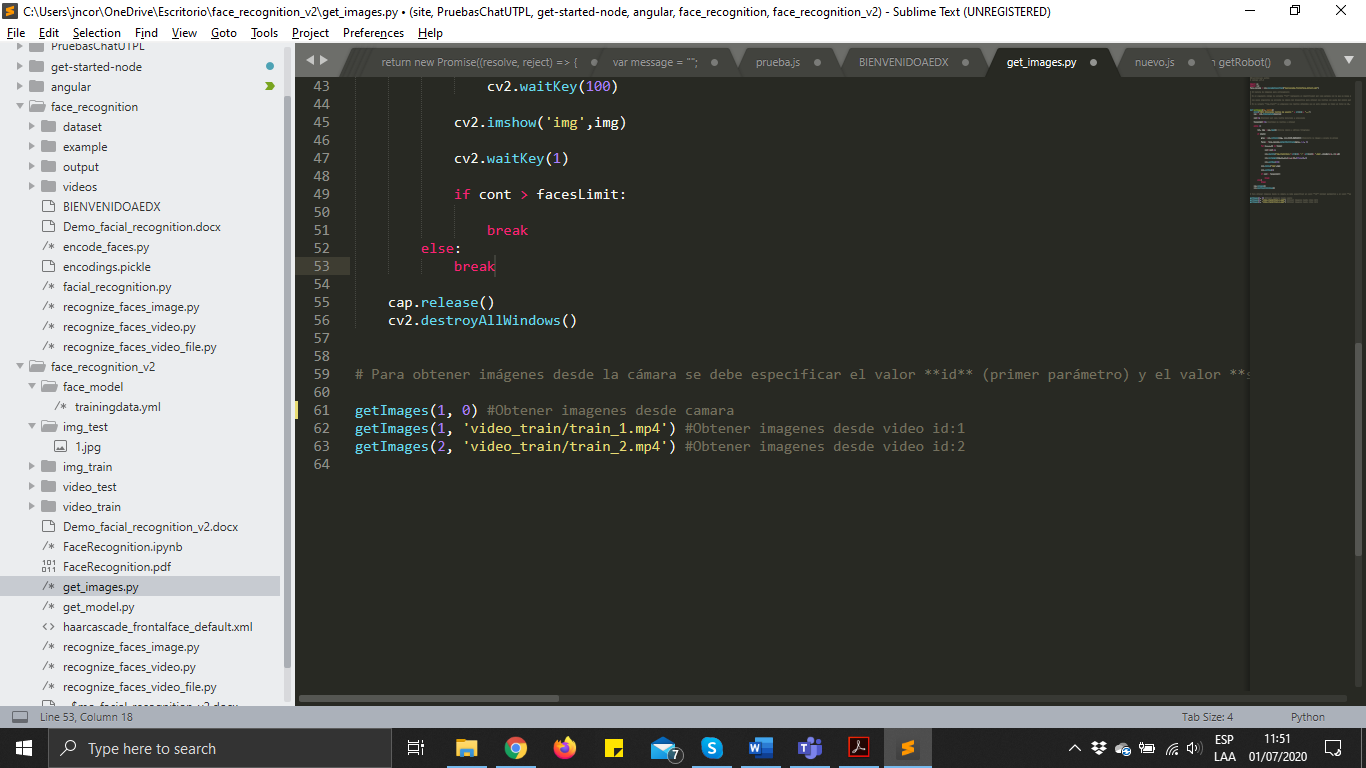
Para ello, se debe considerar, que dentro del código la variable **id** representa el identificador por cada persona con la que se desea entrenar el modelo, este debe incrementar por cada nuevo rostro.

Los pasos siguientes se enciende la cámara del dispositivo para obtener los rostros con ayuda del modelo para la detección de los mismo, que se encuentra en la variable: **face\_cascade**, para mejorar la precisión la imagen se convierte a escala de grises y se configuran los parámetros para la detección de rostros en **1.1**, este factor controla el re-escalado de la imagen que es de gran importancia para detectar rostros según su tamaño en la imagen.

En la carpeta **img\_train** se almacenan los rostros obtenidos que, en este ejemplo, se toman un total de 30, el nombre de cada archivo contiene el formato **User.1.1** el primer valor es el de la variable id y el siguiente un incremental por las 30 imágenes que se van a generar.



Para obtener imágenes desde la cámara se debe especificar el valor **id** (primer parámetro) y el valor source (segundo parámetro), para el cual especificar **0** corresponde a usar la cámara, mientras que, si se especifica una ruta, se tomara las capturas desde un video.



Una vez configurado todo, ubicarse en el directorio del proyecto y ejecutar:

python get\_images.py

**Creación de modelo**

El siguiente archivo (**get\_model.py)** contiene el entrenamiento del modelo para el reconocimiento facial, se toman las imágenes generadas en la carpeta **img\_train** por cada persona y obtiene las características encontradas relacionándolas al **id** de cada imagen.

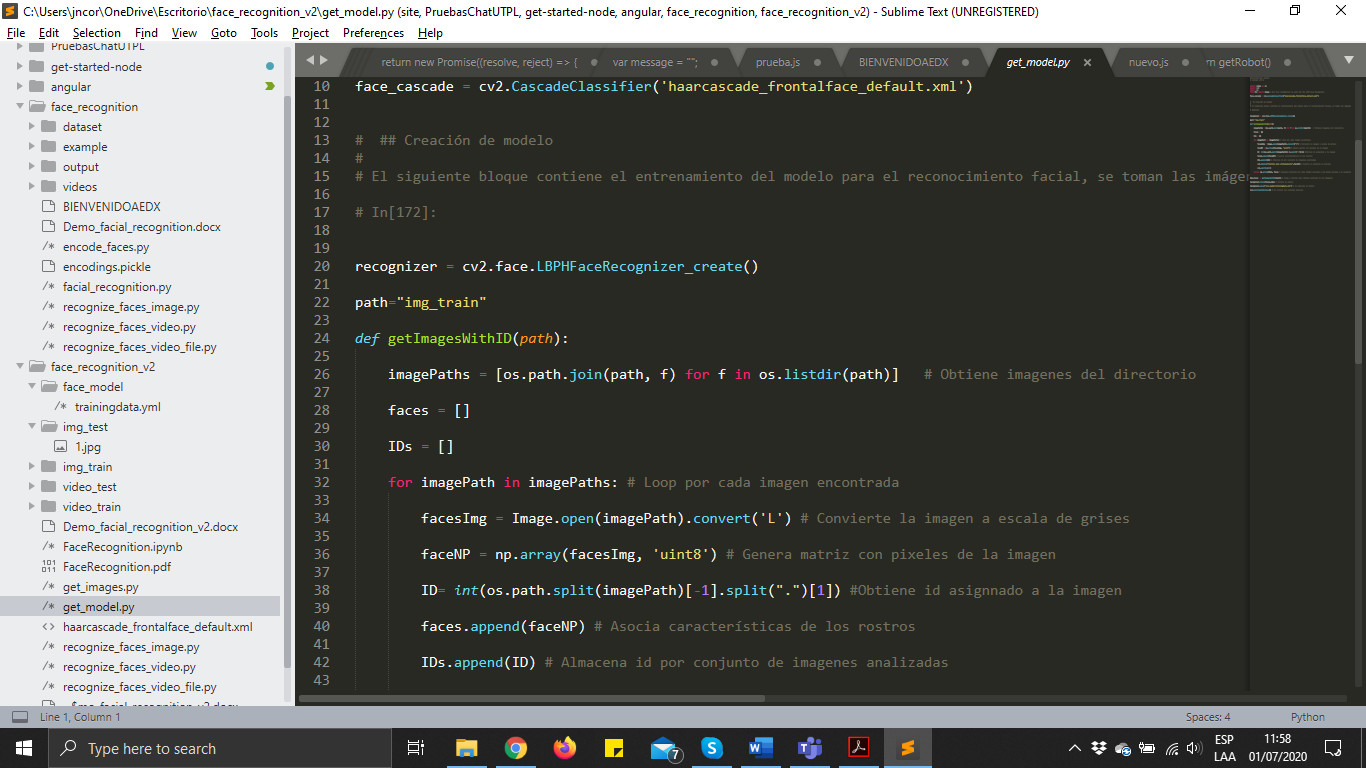
Se hace uso del algoritmo LBPH de OpenCV, para ello, importamos la librería de imágenes para python, la cual usaremos para el reconocimiento facial con LBPH.

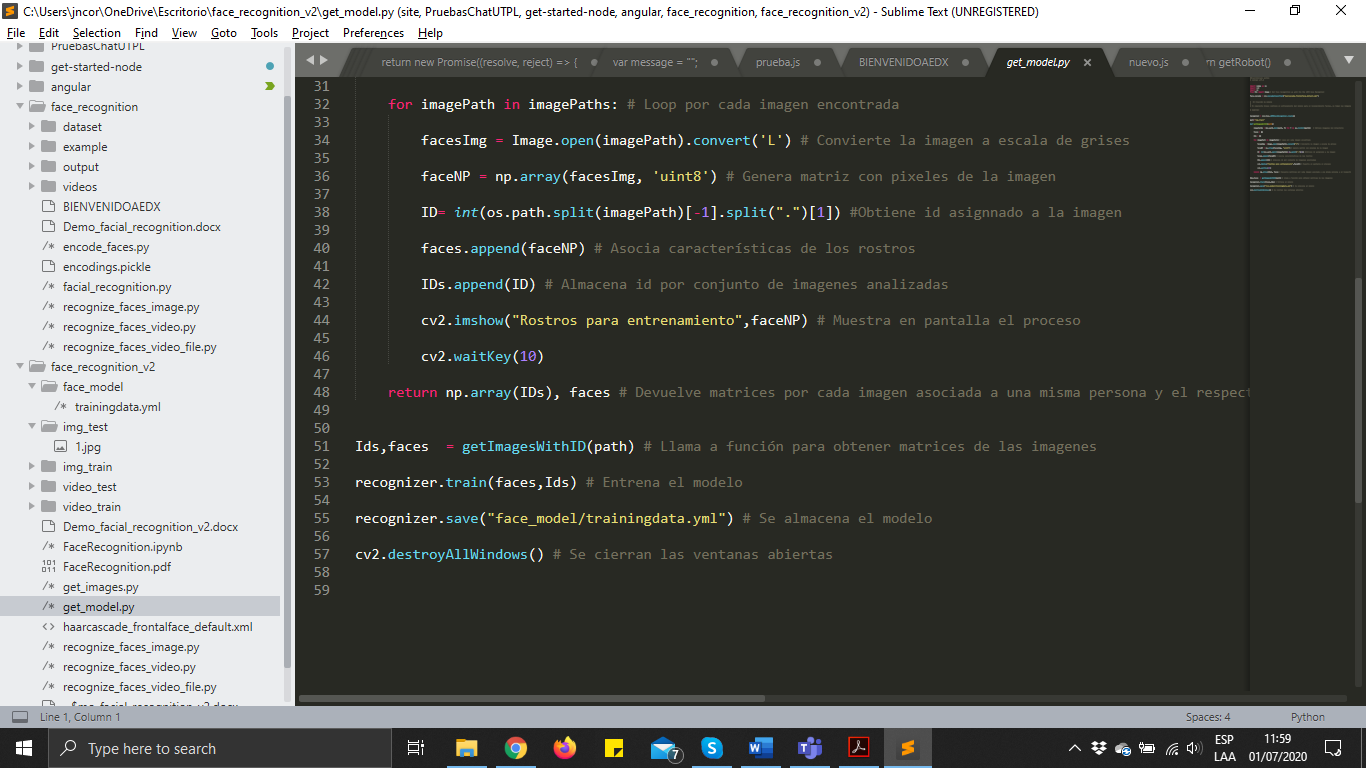
from PIL import Image

OpenCV cuenta con algunos clasificadores pre-entrenados para la detección del rostros, ojos, sonrisa, etc. Por lo que, se hará en esta ocasión, se hará uso del Clasificador **haarcascades frontal face**, el cual ha sido previamente almacenado en el directorio del proyecto. Para ello, cargamos el archivo en la variable **face\_cascade**

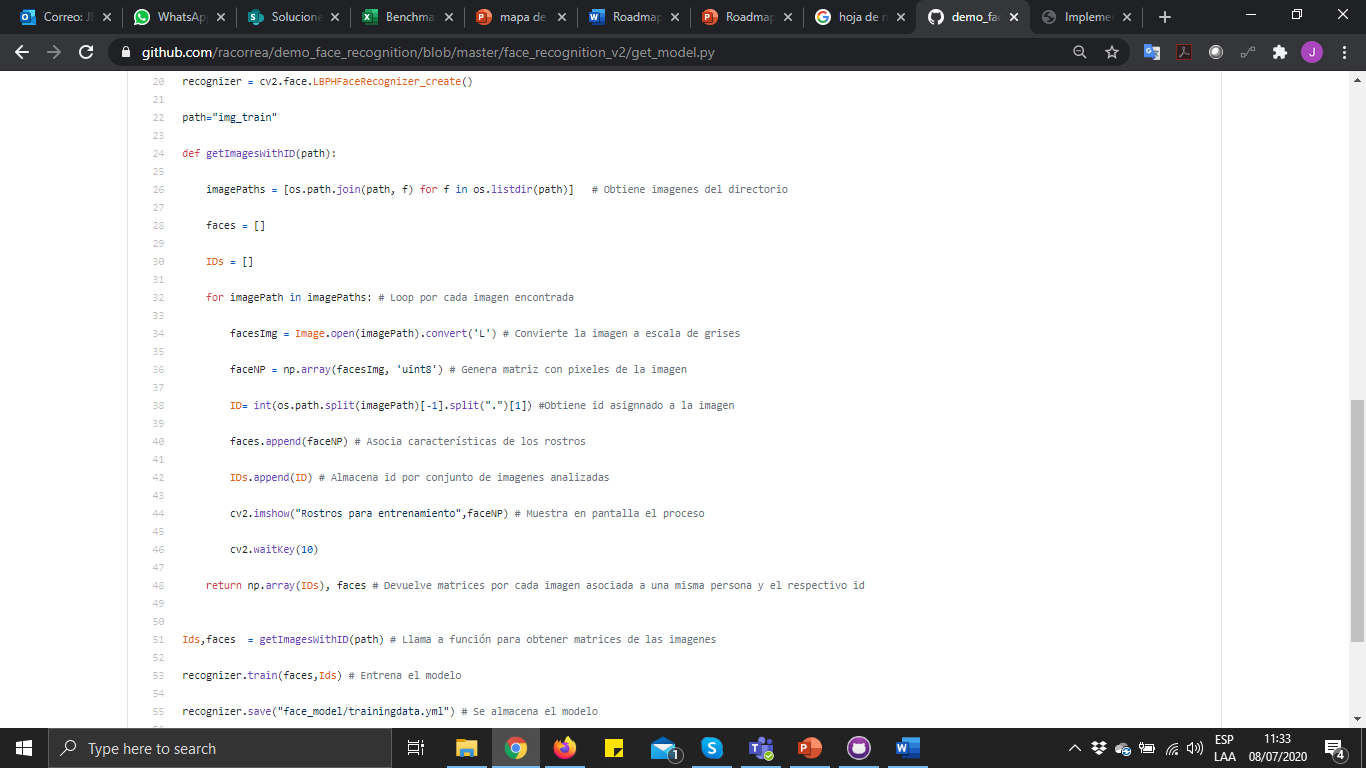
face\_cascade = cv2.CascadeClassifier('haarcascade\_frontalface\_alt.xml')

Para ello, configurar el path del dataset, en este caso “img\_train” y el path del archivo donde se almacenará el modelo LBPH para el reconocimiento facial.





**MODELO**



Función para obtener la matriz de cada rostros y el id asociado

Se entrena el modelo con los rostros y id obtenidos por cada imagen conocida

Una vez configurado todo, ubicarse en el directorio del proyecto y ejecutar:

python get\_model.py

**Prueba del modelo**

Para el reconocimiento facial, es posible probar el modelo bajo dos conceptos:

* De forma estática en una imagen (recognize\_faces\_image.py)
* De forma dinámica:
  + Webcam(recognize\_faces\_video.py)
  + Archivos de video(recognize\_faces\_video\_file.py)

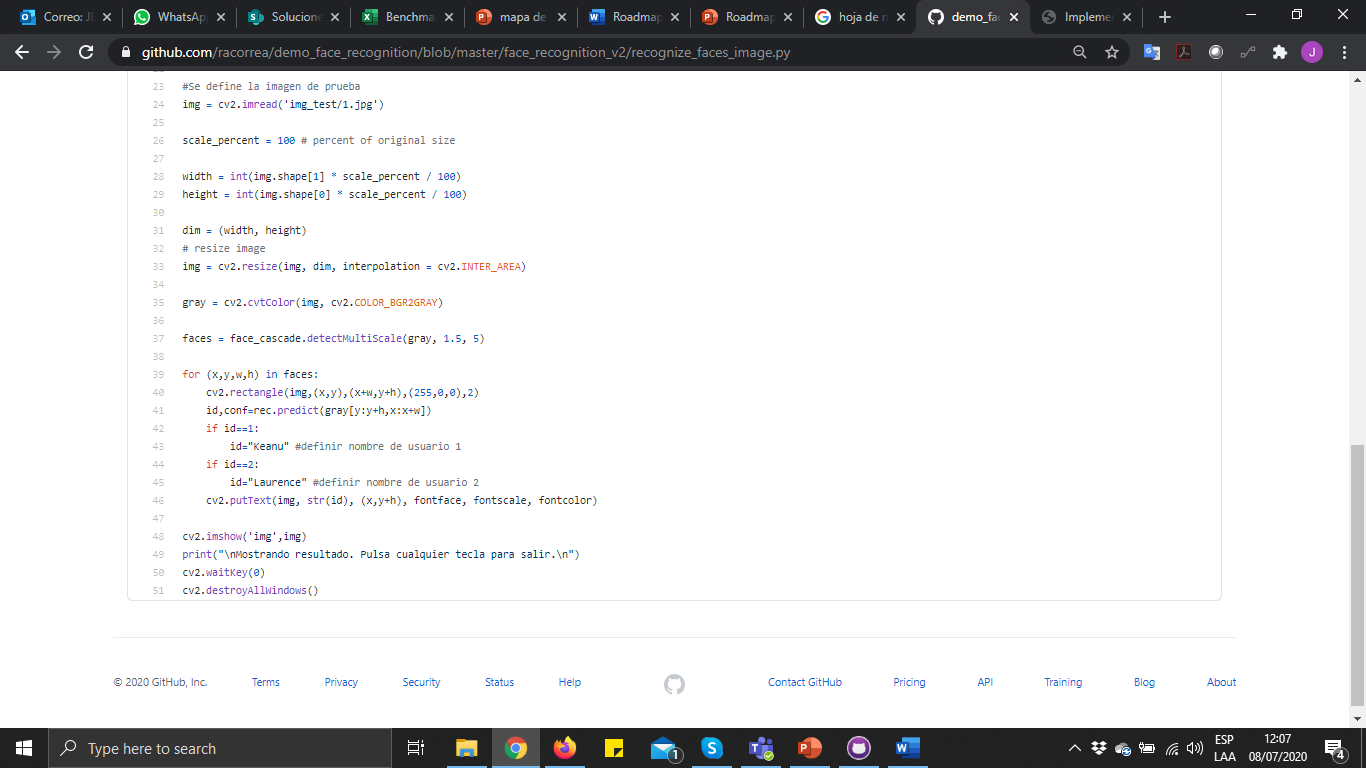
Inicialmente se establece el modelo del clasificador a utilizar, se crea la instancia para usar el algoritmo LBPH de OpenCV para el reconocimiento facial, y se define el archivo del modelo entrenado con los rostos conocidos.

face\_cascade = cv2.CascadeClassifier('haarcascade\_frontalface\_alt.xml')

rec = cv2.face.LBPHFaceRecognizer\_create();

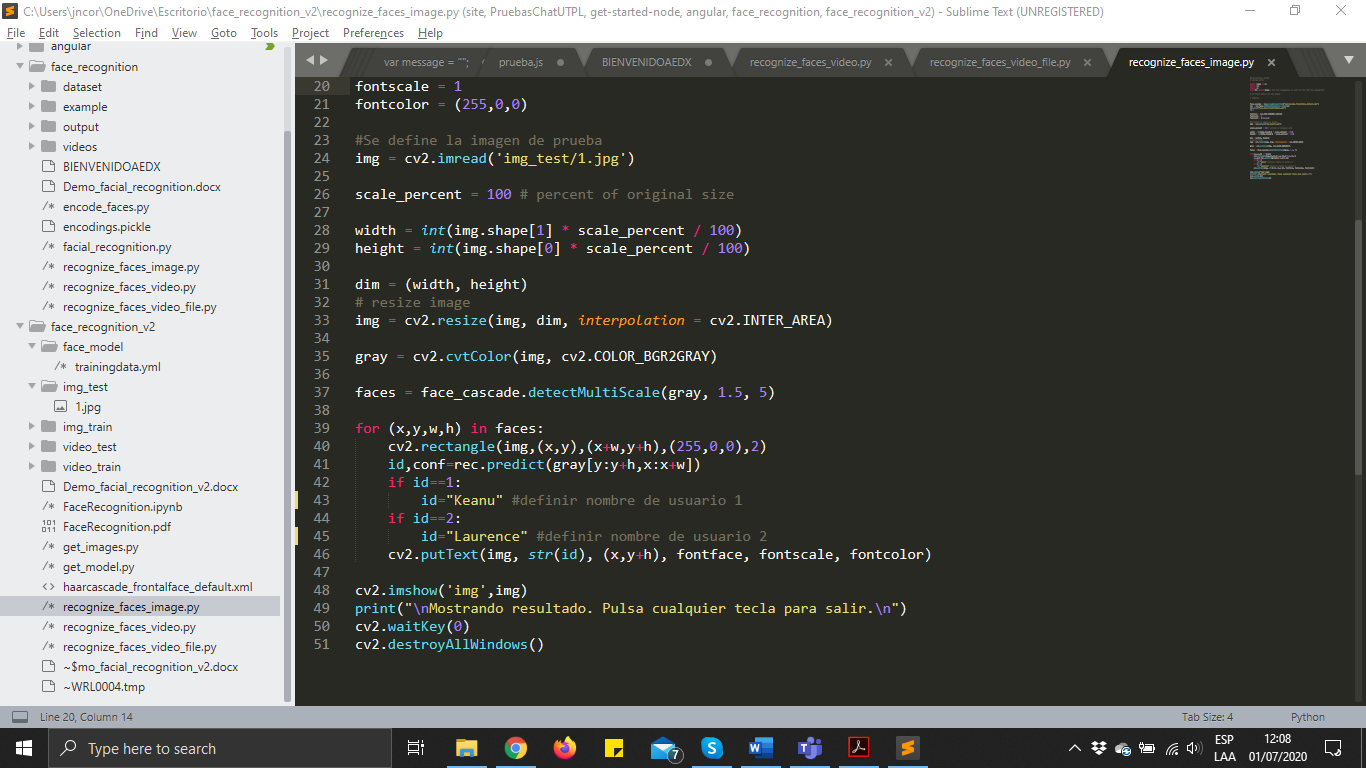
rec.read("face\_model/trainingdata.yml")

La predicción, la realiza mediante la siguiente línea de código, desde la cual se obtiene el id del rostro y la confianza de detección:



Es necesario, además, definir como parámetros las entradas, según el caso:

* **Reconocimiento facial en Imagen:** Establecer la imagen sobre la cual se probará el modelo.

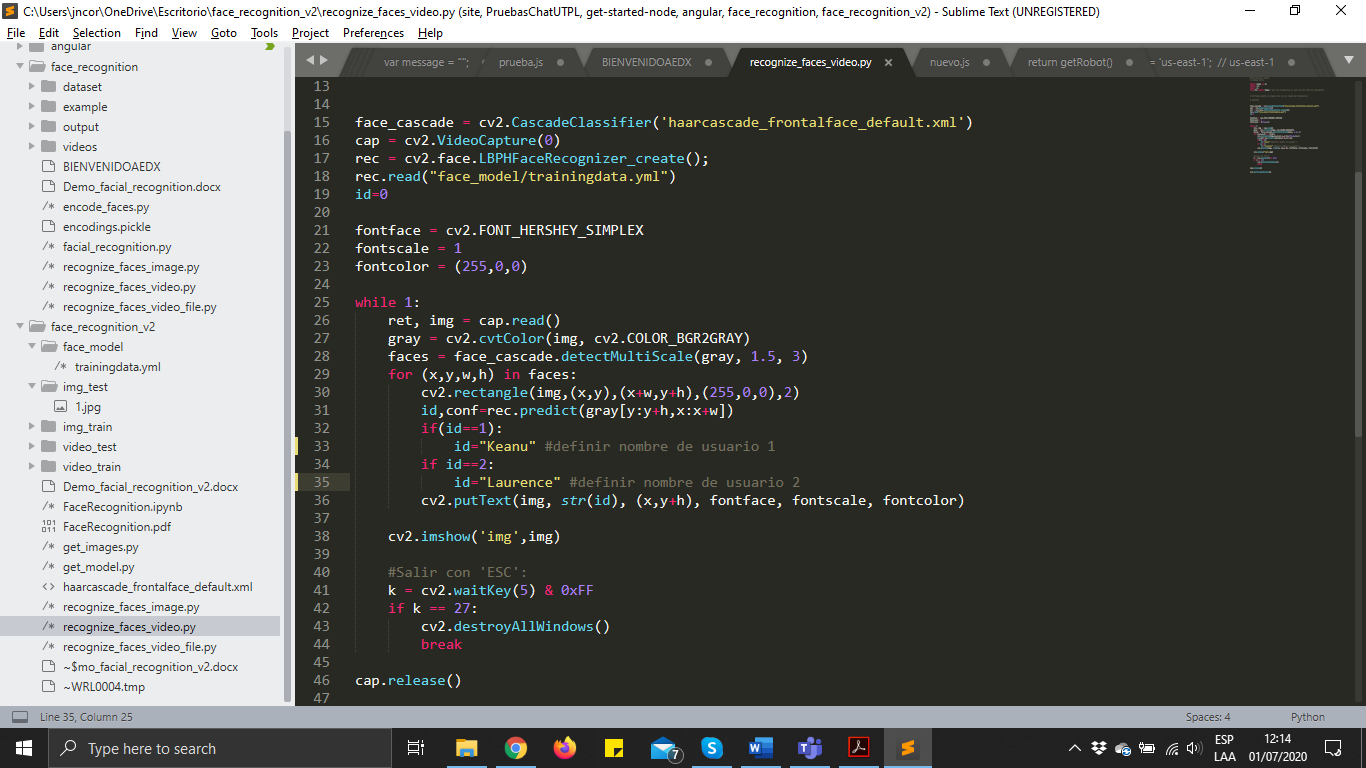


Definir los nombres correctos de las personas, según fueron entrenados.

Una vez configurado, ir al directorio y ejecutar:

python recognize\_faces\_image.py

* **Reconocimiento facial mediante webcam:** Definir **0** como entrada para la webcam. Presionar ESC para salir.

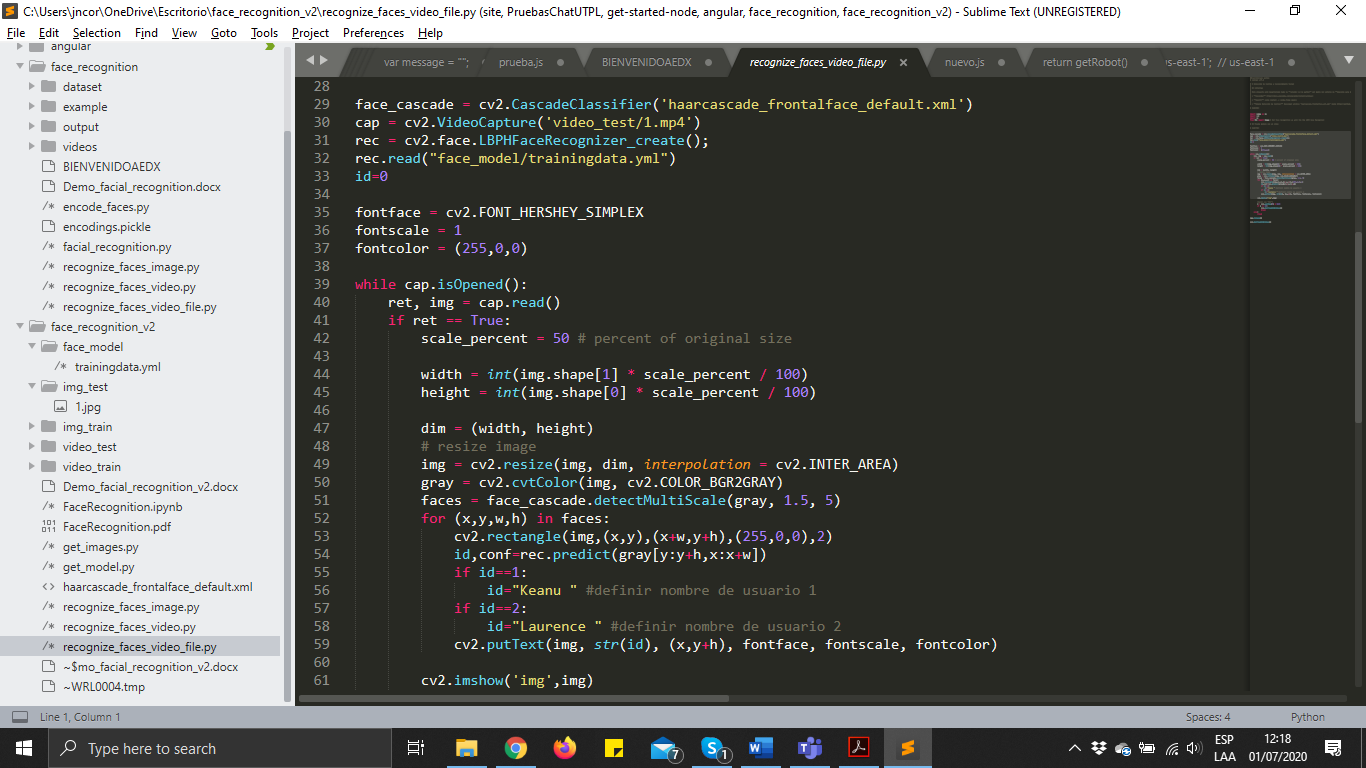


Definir los nombres correctos de las personas, según fueron entrenados.

Una vez configurado, ir al directorio y ejecutar:

python recognize\_faces\_video.py

* **Reconocimiento facial en archivos de video:** Establecer la ruta del video sobre la cual se probará el modelo.



Definir los nombres correctos de las personas, según fueron entrenados.

Una vez configurado, ir al directorio y ejecutar:

python recognize\_faces\_video\_file.py