Face Recognition Code Document

This instruction document is for all the face recognition related code, where we have codes that can take your picture and save it, a system for creating face encode from the saved photo, and codes for detecting faces from Camera and Images. In this project, we used the face_recognition and OpenCV module.

The Face Recognition library is widely known around the web for being the world's most straightforward facial recognition API for Python and the command line. The best of all is that you won't need to pay a dime for it, the project is open-source, so if you have some development knowledge and you can build a library from scratch, you will surely know how to work with this library.

OpenCV (Open Source Computer Vision Library) is an open-source computer vision library with bindings for C++, Python, and Java. It is used for a wide range of applications, including medical image analysis, stitching street view images, surveillance video, detecting and recognizing faces, tracking moving objects, extracting 3D models, and much more. OpenCV can take advantage of multi-core processing and features GPU acceleration for real-time operation.

OpenCV-Python is a library of Python bindings designed to solve computer vision problems. OpenCV-Python uses of NumPy, a highly optimized library for numerical operations with a MATLAB-style syntax. All the OpenCV array structures are converted to and from NumPy arrays.

Note: This project is developed on a Linux System (Ubuntu), so it is advisable to use Linux System.

System Set-Up

There are a couple of Python packages that need to validate before running any above program in this folder. Please go to the Installation Documents folder and follow all the steps.

Pre-Requisites

- A system running on Windows/Ubuntu APP/Ubuntu OS
- A user account with sudo/administration privileges
- Access to a terminal window/command-line

Before continuing with this tutorial, make sure you are logged in as root or a user with sudo or administration privileges and completed all the necessary steps from the Installation Documents folder**.**

In this tutorial, we will explain all the Face Recognition Codes that are written in Python3 on Ubuntu. We have some codes related to Face Recognition as follows:

1. Capture_Picture_Main.py
    I. Capture_Picture_Save.py
2. Face_Encoding.py

3. Face_Detection_Camera.py
4. Face_Detection_Image.py

```
somak@VivoBook:~/Robotic-Greeter/Development_Code/Face_Recognition_Code$
somak@VivoBook:~/Robotic-Greeter/Development_Code/Face_Recognition_Code$ ls -lrt
total 988
-rw-rw-r-- 1 somak somak 930127 Jun 11 21:07 haarcascade_frontalface_default.xml
drwxrwxr-x 5 somak somak   4096 Jun 12 00:00 Sample_Images
-rw-rw-r-- 1 somak somak   6281 Jun 18 21:11 README.rst
-rwxrwxr-x 1 somak somak   9172 Jun 26 01:36 Face_Encoding.py
-rwxrwxr-x 1 somak somak   7725 Jun 26 01:36 Face_Detection_Image.py
-rwxrwxr-x 1 somak somak   8030 Jun 26 02:13 Face_Detection_Camera.py
-rwxrwxr-x 1 somak somak   9893 Jun 26 02:20 Capture_Picture_Main.py
-rwxrwxr-x 1 somak somak  12352 Jun 26 02:20 Capture_Picture_Save.py
drwxrwxr-x 3 somak somak   4096 Jun 27 22:06 Dataset
-rw-rw-r-- 1 somak somak   2285 Jun 27 22:10 encodings.pickle
somak@VivoBook:~/Robotic-Greeter/Development_Code/Face_Recognition_Code$
```

Some of the codes in this folder has one function called **process_parameter_set(),** which contains the different parameter value. Depending on the parameter value, the program will perform another process. Hence it is essential to validate this function and all the parameters before running any codes.

```
def process_parameter_set():
    """
```

Face Recognition Code

Face Recognition codes written in Python as follow:

1. Capture_Picture_Main.py

Capture_Picture_Main.py program will be called from Main_Process.py. It will receive one input argument (Unique ID) from the primary process, and based on the input argument, it will call to Capture_Picture_Save.py for taking and saving the picture. This process is a looping process that asks the user if he want to take multiple photos and keep it.

Also, you can run this program as a stand-alone program. In this case, the Unique ID will be 0. The concept of Unique ID in this program is because this process needs to save the user images with the same Unique ID as present in the table. So that next time, if the user comes in front of the Robotic Greeter, it can detect the faces with the Unique ID. And with the same Unique ID, it can search the user details from the table.

I) Capture_Picture_Save.py

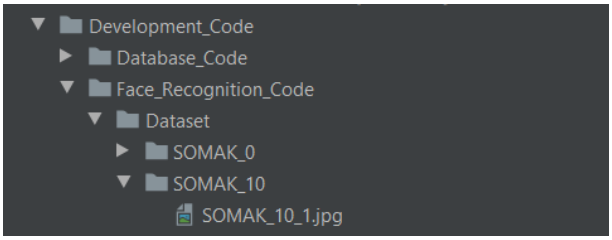
Capture_Picture_Save.py program will be called from Capture_Picture_Main.py. It will receive the Unique ID from the parent program. It will first take a picture of the person, and later it will prompt a pop-up screen where the user can enter the name. Now, this program will merge the name with Unique ID and will save it into the Dataset directory.

First, this program will create a new folder inside the Dataset directory (If there are no same folder present) and save the picture inside the newly created folder. If the users want to take

multiple pictures, then this program will automatically increase the instance ID (for the first picture, instance ID will be 001)

This program should save the picture into the following directory:

Dataset/XXX_UniqueID/XXX_UniqueID_YYY.jpg (XXX - Person Name, YYY – Instance ID)



This code has one function called **process_parameter_set(),** which contains the parameter of the Dataset directory. It is essential to check the Dataset directory present before running this program. If the Dataset directory is not present then, please create this directory first.

```
dataset_path = 'Dataset'
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

return dataset_path, face_cascade
```

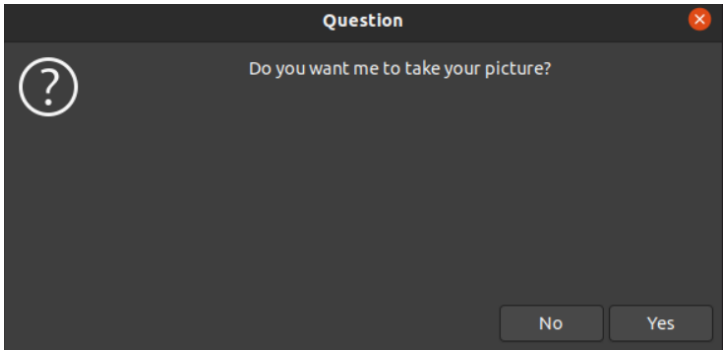
Execution:

- To run this program, run the below command from your command prompt.

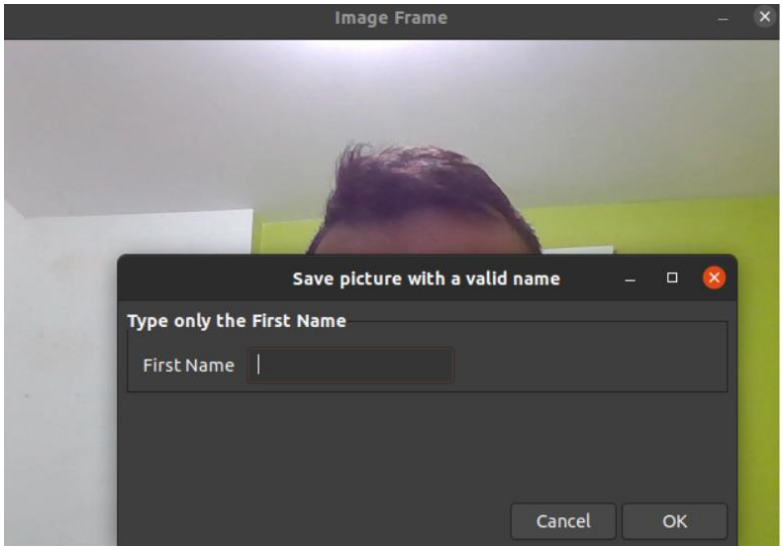
```
$ python3 Capture_Picture_Main.py
```

```
somak@VivoBook:~/Robotic-Greeter/Development_Code/Face_Recognition_Code$ python3 Capture_Picture_Main.py
Starting program : Capture_Picture_Main.py - at : 20:46:41 on : 29/06/2020
Processing Capture_Picture_Main.py from main process.
Processing Capture_Picture_Main.py stand alone
Calling program : Capture_Picture_Save.py......
Calling program : Capture_Picture_Save.py......
ERROR : I cannot take your picture currently - inside process_calling function.
Ending program : Capture_Picture_Main.py - at : 20:48:51 on : 29/06/2020
somak@VivoBook:~/Robotic-Greeter/Development_Code/Face_Recognition_Code$
```

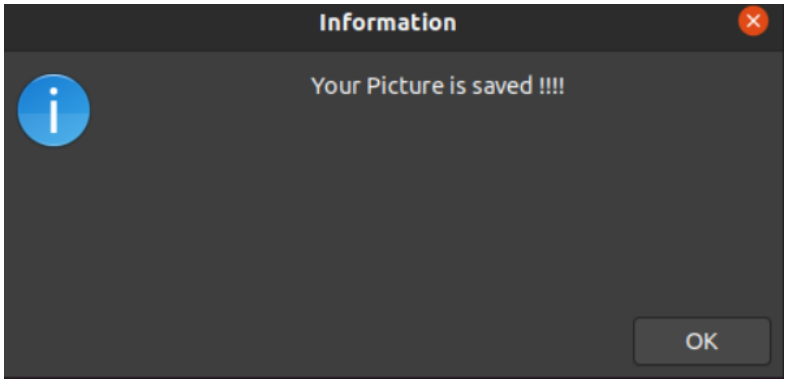
- Then this process will prompt a pop-up message as below. Click Yes to continue and No to exit.



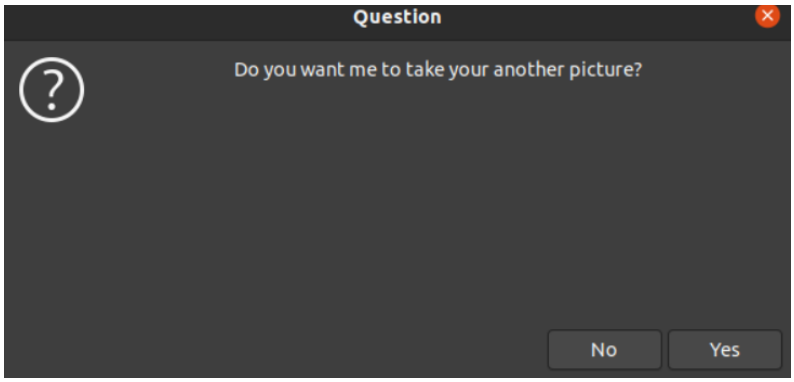
- Now, this process will automatically take your picture and will ask if you want to enter your name as below. Fill the name and click OK to continue.



- It will save your picture and will inform you. Click OK to continue.

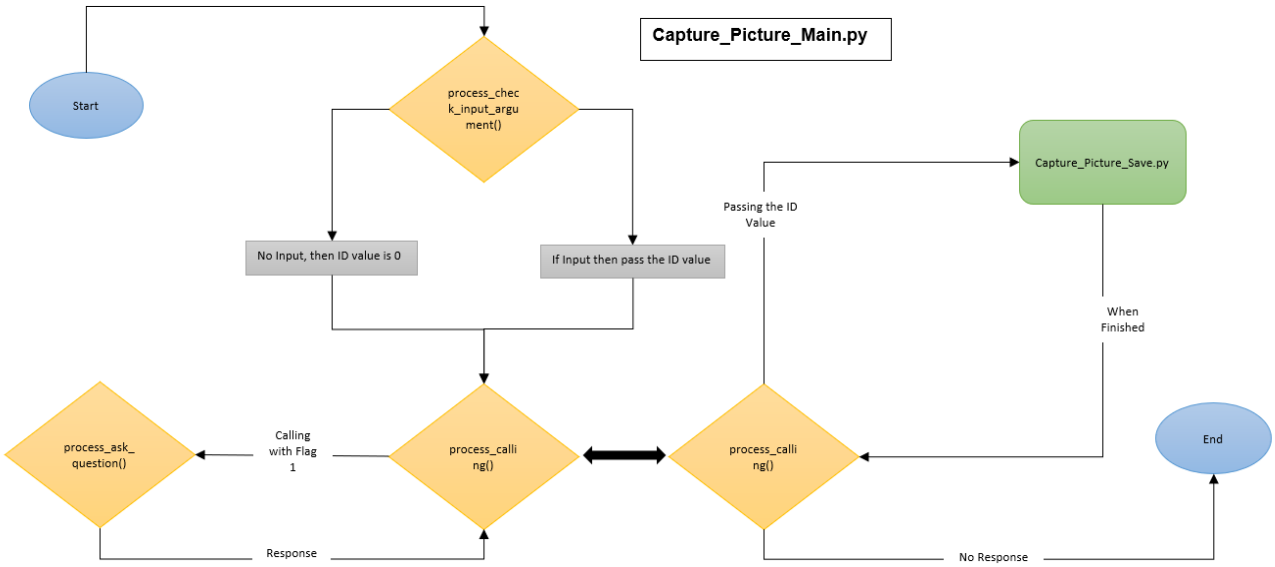


- At last, it will ask if you want to take another picture. To continue, click Yes or No to exit.



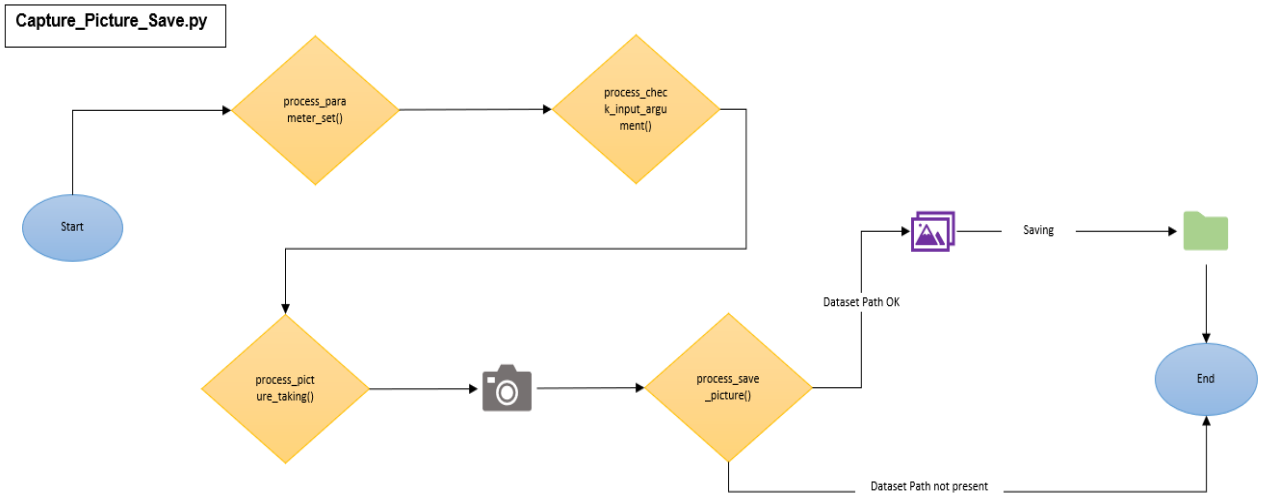
Flowchart

Following is the flowchart architecture of the Capture_Picture_Main.py program.



Flowchart

Following is the flowchart architecture of the Capture_Picture_Save.py program.



2. Face_Encoding.py

This program is the main program that will create an encoding—pickle file, which is essential for Face_Detection_Camera.py & Face_Detection_Image.py. Mainly, for all the application which needs to detect the faces from camera or images.

First, this program will check if the encoding. pickle file is already present or not in the same folder. If the file exists, then it will read all the old data (old coordinate data) from the encoding file and store it into one variable, later this program will check if any new images present in the Dataset directory or not. If the Dataset directory has any new pictures or old photos, then this program will read those photos and will create coordinate for new images. After creating all the coordinate of all the latest images, this program will merge all the old coordinates with new coordinates and will write into encoding—pickle file.

This code has one function called **process_parameter_set(),** which contains the parameter of the Dataset directory. From this directory, this process will validate new images. So, it is essential to look at this function before running this program.

```
image_path = 'Dataset'
detect_model = 'cnn'

return image_path, detect_model
```

You need to run separately since this program will not be going to call from any other program. This program will create an encoding file that is essential for any face detection program.

Execution:

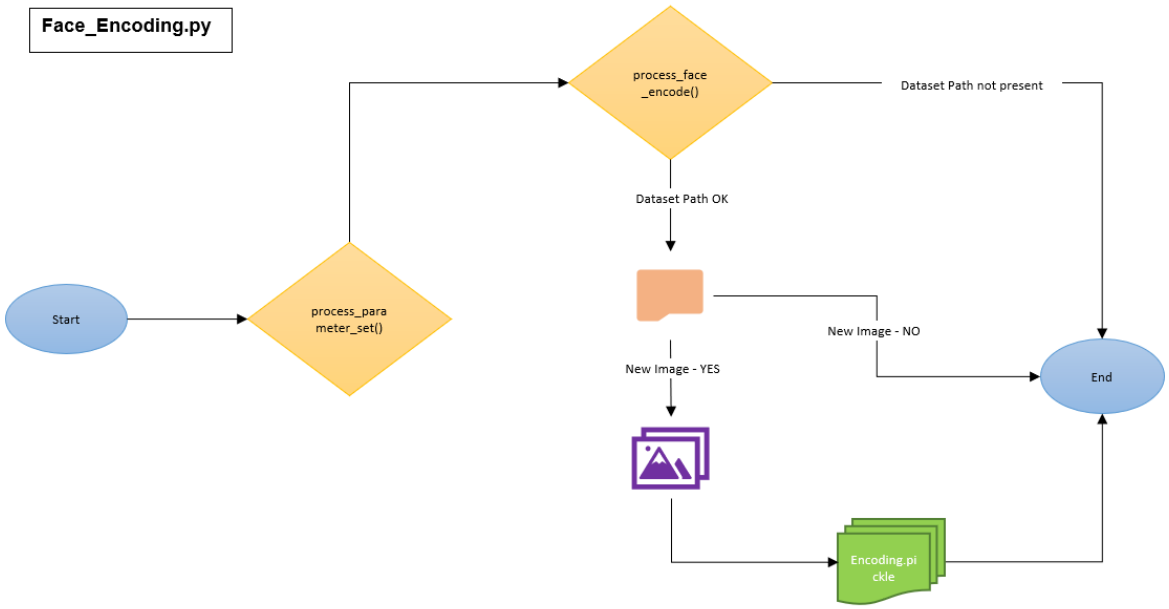
- To run this program, run the below command from your command prompt.

```
$ python3 Face_Encoding.py
```

```
somak@VivoBook:~/Robotic-Greeter/Development_Code/Face_Recognition_Code$ python3 Face_Encoding.py
Starting program : Face_Encoding.py - at : 20:43:48 on : 29/06/2020
Quantifying faces....
Processing image 1/3 - image data : Johnny Depp
Processing image 2/3 - image data : SOMAK_0
Processing image 3/3 - image data : SOMAK_0
Serializing encodings....
Reading from old encoding file....
Writing new data in encoding file....
Ending Program : Face_Encoding.py - at : 20:44:00 on : 29/06/2020
somak@VivoBook:~/Robotic-Greeter/Development_Code/Face_Recognition_Code$
```

Flowchart

Following is the flowchart architecture of the Face_Encoding.py program.



3. Face_Detection_Camera.py

This program will be called from Main_Process.py. It is the primary process of the face detection program from the real-time video camera.

First, this process will read all the coordinates present in the encoding.pickle file (If this file is not current, this program will exit) and store it into one variable (data variable). Later, it will open the video camera automatically and check if any faces are present in the real-time video or not. If it detects any faces in the video, then it will create a coordinate for the faces from the camera.

This process will then check if the newly created coordinate matches with any of the coordinates stored into the data variable. If any matches happen, this program will show the person's name in the video (also, it will create a square box over the face). If not, then it will show as Unknown.

This code has one function called **process_parameter_set(),** which contains the parameter of encoding_file and face_cascade. It is essential to validate encodings.pickle & haarcascade_frontalface_default.xml files are present in the same folder or not. To create encodings.pickle file, please execute Face_Encoding.py as mentioned in above, and haarcascade_frontalface_default.xml is vital to detect frontal face with OpenCV.

```
encoding_file = 'encodings.pickle'
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

return encoding_file, face_cascade
```

Execution:

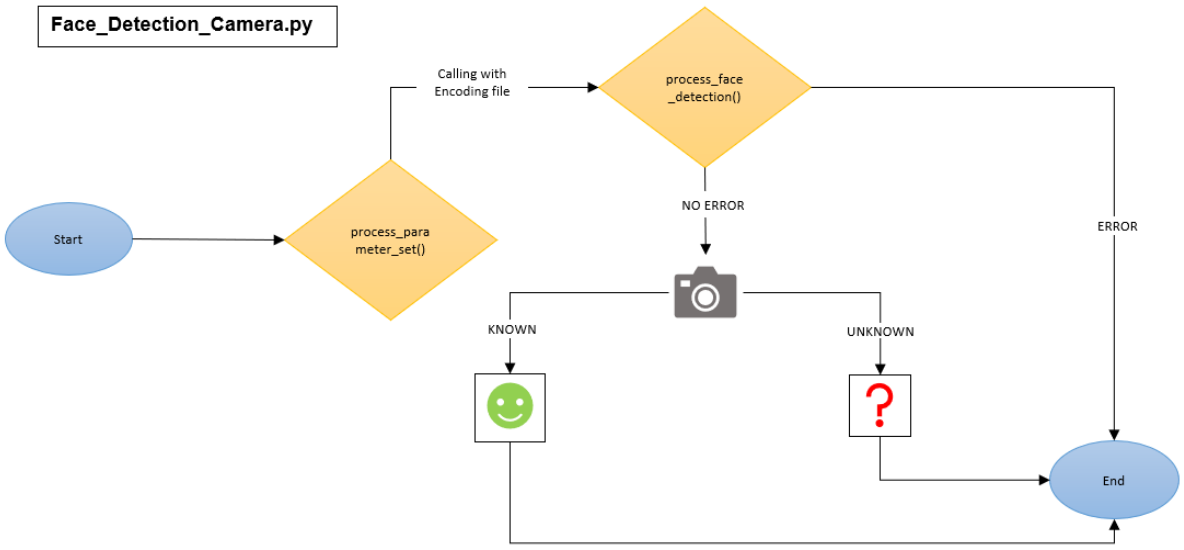
- To run this program, run the below command from your command prompt.

```
$ python3 Face_Detection_Camera.py
```

```
somak@VivoBook:~/Robotic-Greeter/Development_Code/Face_Recognition_Code$ python3 Face_Detection_Camera.py
Starting program : Face_Detection_Camera.py - at : 20:50:26 on : 29/06/2020
SOMAK_0
Ending program : Face_Detection_Camera.py - at : 20:50:32 on : 29/06/2020
somak@VivoBook:~/Robotic-Greeter/Development_Code/Face_Recognition_Code$
```

Flowchart

Following is the flowchart architecture of the Face_Detection_Camera.py program.



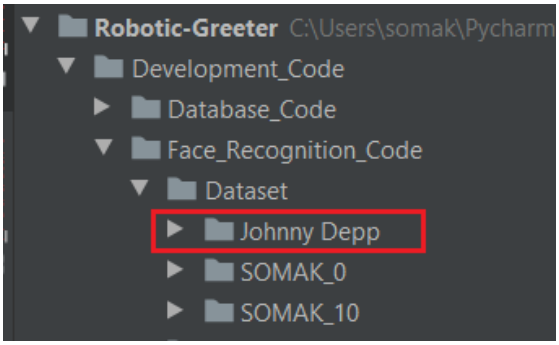
4. Face_Detection_Image.py

This program is a testing program of Face_Detection_Camera.py; this process is used to detect the faces from an image. This process works the same as the above program.

But to run this program, you need to do the follows:

a) First, take your two pictures from your machine or just download any two images of any person from Google.

b) Then, put one image into a Dataset directory/Image_Name_Directory. (Suppose if image name is ABC then first create a directory inside Dataset directory with name ABC and put the picture inside of ABC directory).



c) Run the Face_Encoding.py (This will create an encoding.pickle file with the image coordinate).

d) Then you can put another image inside of the Sample_Images directory. You can rename the picture as your own choice (But if the image name is ABC in the dataset directory, then

do not put the same name here). Then, go to **process_parameter_set(),** and change the image_path value.

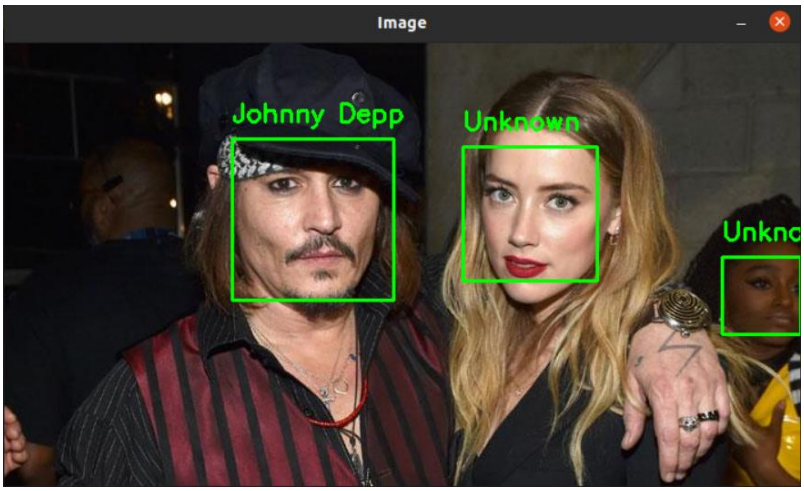
```
    encoding_file = 'encodings.pickle'
    image_path = 'Sample_Images/image1.jpg'
    detect_model = 'cnn'

    return encoding_file, image_path, detect_model
```

e) At last, run this program.

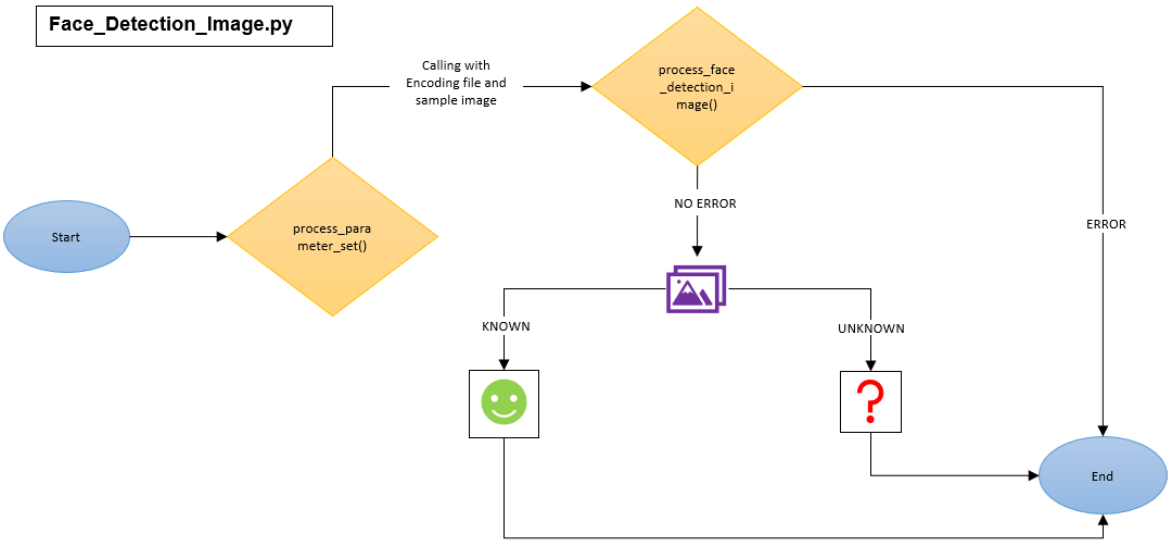
```
$ python3 Face_Detection_Image.py
```

```
somak@VivoBook:~/Robotic-Greeter/Development_Code/Face_Recognition_Code$
somak@VivoBook:~/Robotic-Greeter/Development_Code/Face_Recognition_Code$ python3 Face_Detection_Image.py
Starting program : Face_Detection_Image.py - at : 20:44:51 on : 29/06/2020
Ending program : Face_Detection_Image.py - at : 20:45:21 on : 29/06/2020
somak@VivoBook:~/Robotic-Greeter/Development_Code/Face_Recognition_Code$
```



Flowchart

Following is the flowchart architecture of the Face_Detection_Image.py program.


Face_Detection_Image.py
Calling with Encoding file and sample image
process_face _detection_i mage()
Start
process_para meter_set()
NO ERROR
ERROR
KNOWN
UNKNOWN
End