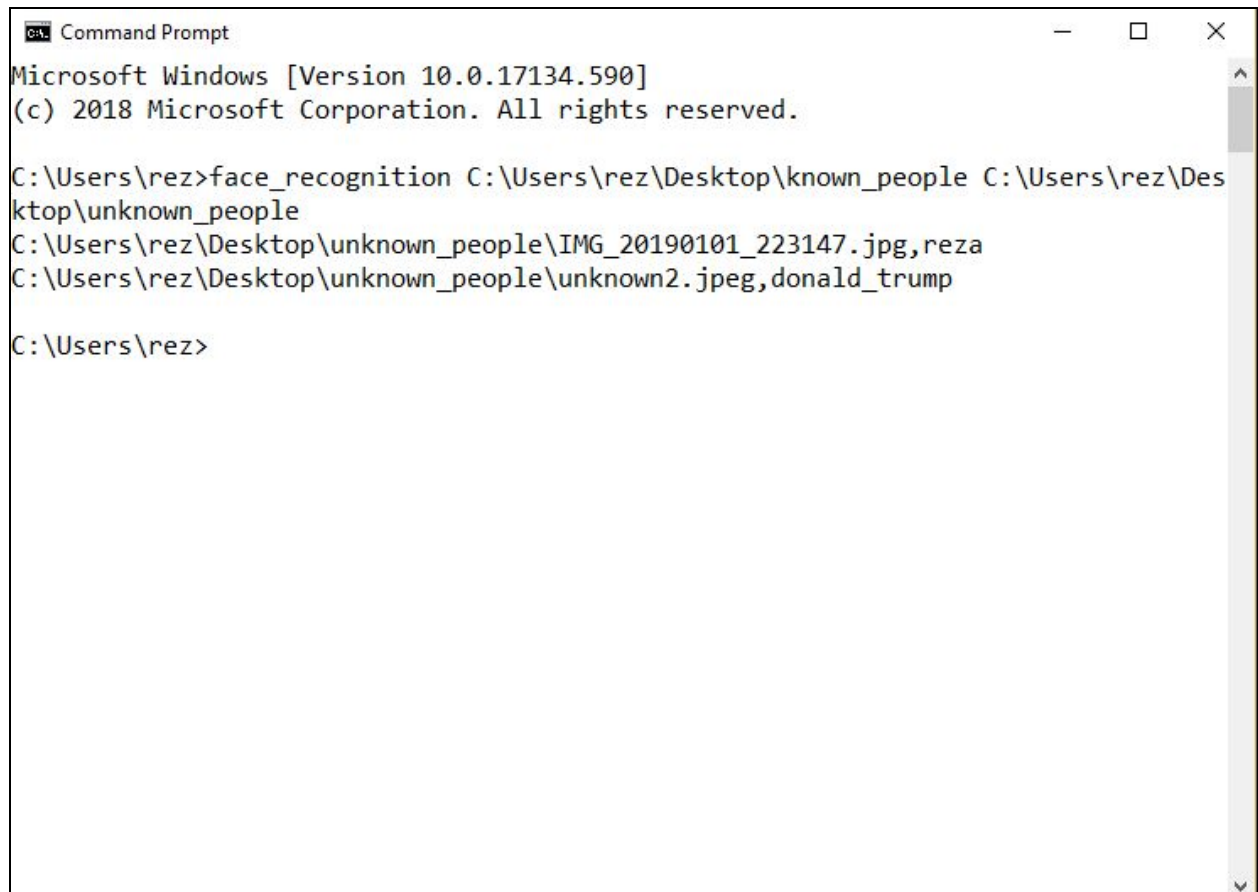


1. Running the command *face\_recognition* with the folder of known people (donald trump, reza) yields two results, with the recognition of the two unknown files, labeling each with names.

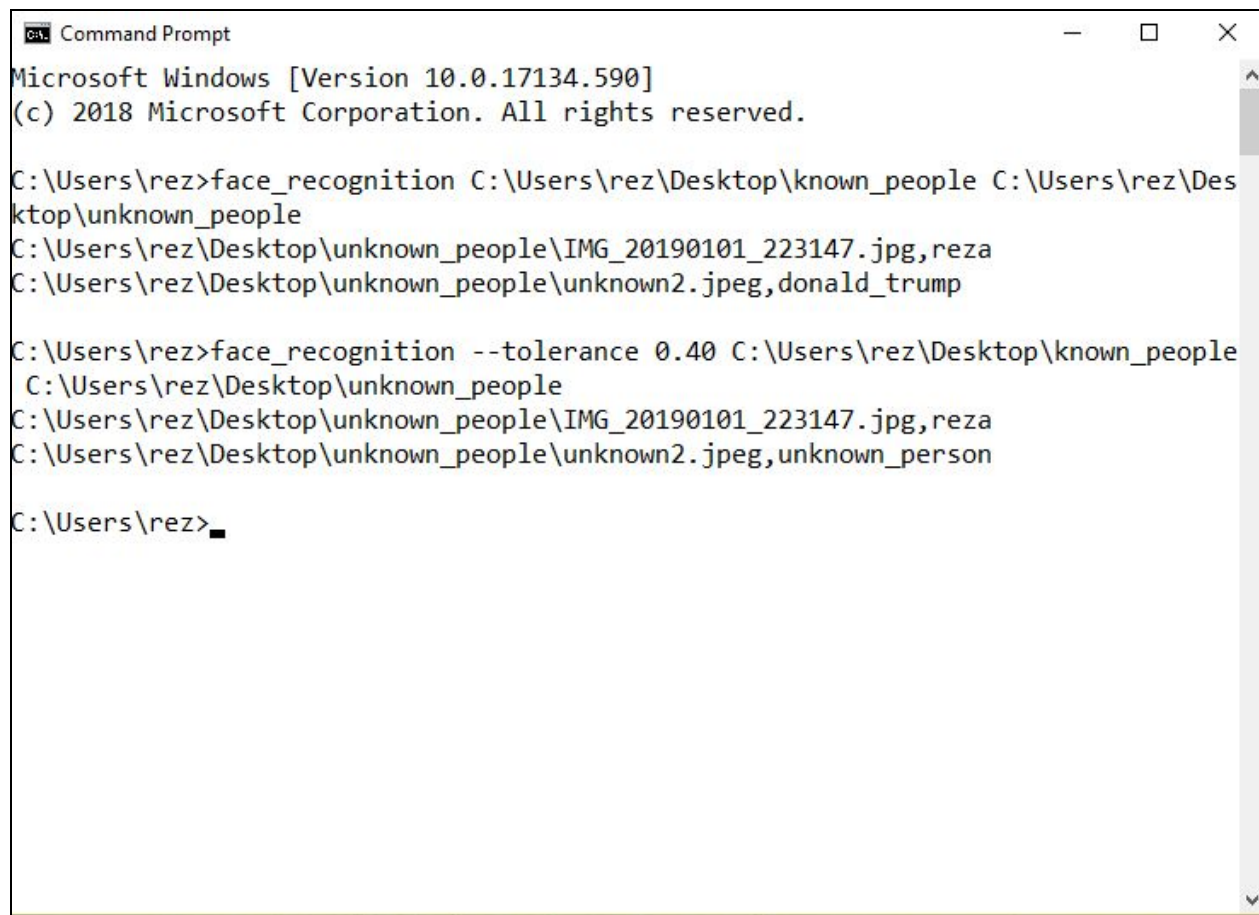


```
Command Prompt
Microsoft Windows [Version 10.0.17134.590]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\rez>face_recognition C:\Users\rez\Desktop\known_people C:\Users\rez\Desktop\unknown_people
C:\Users\rez\Desktop\unknown_people\IMG_20190101_223147.jpg,reza
C:\Users\rez\Desktop\unknown_people\unknown2.jpeg,donald_trump

C:\Users\rez>
```

Running the command with *tolerance 0.4* yields a different result and in this case, does not recognize trump. This is because 0.4 makes face comparison much more strict.



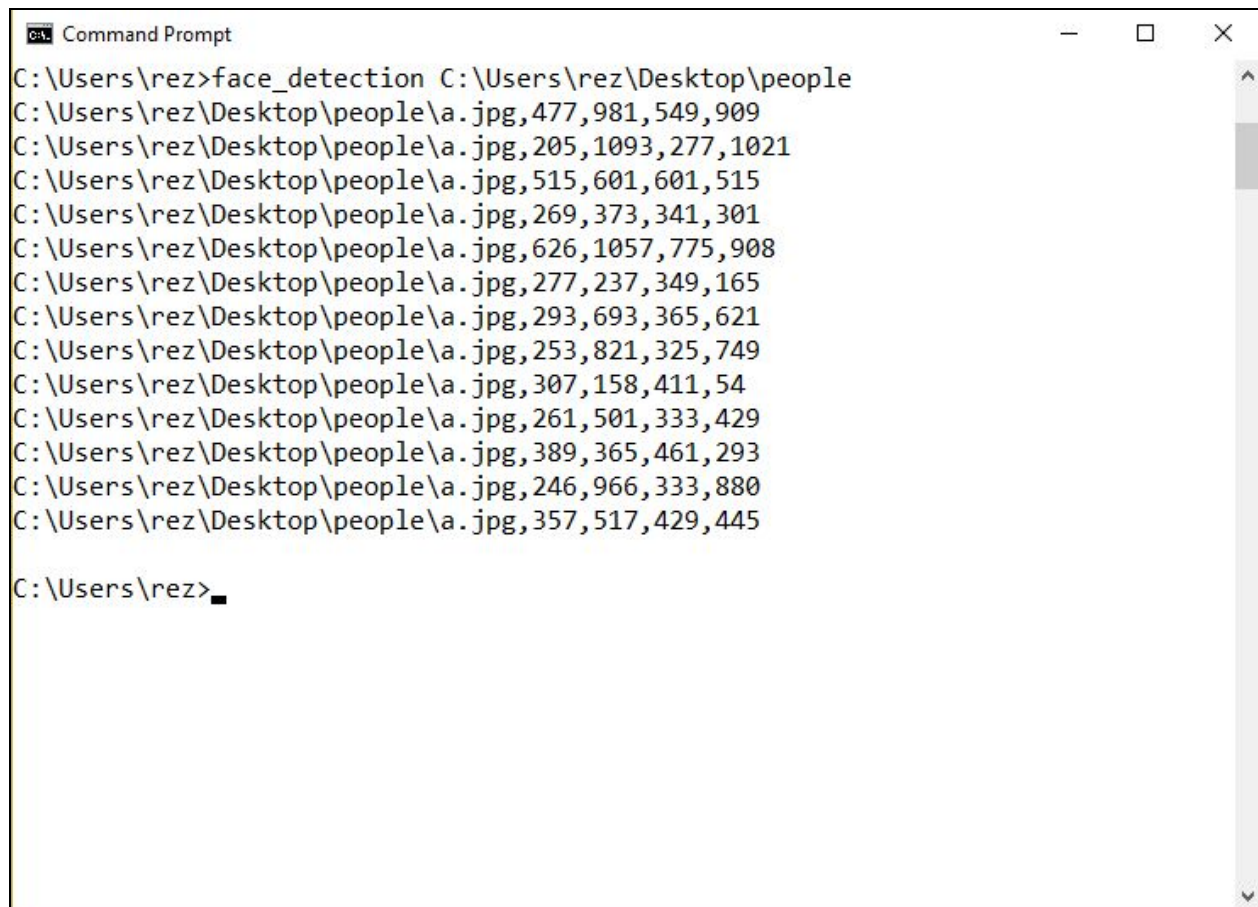
```
Command Prompt
Microsoft Windows [Version 10.0.17134.590]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\rez>face_recognition C:\Users\rez\Desktop\known_people C:\Users\rez\Desktop\unknown_people
C:\Users\rez\Desktop\unknown_people\IMG_20190101_223147.jpg,reza
C:\Users\rez\Desktop\unknown_people\unknown2.jpeg,donald_trump

C:\Users\rez>face_recognition --tolerance 0.40 C:\Users\rez\Desktop\known_people C:\Users\rez\Desktop\unknown_people
C:\Users\rez\Desktop\unknown_people\IMG_20190101_223147.jpg,reza
C:\Users\rez\Desktop\unknown_people\unknown2.jpeg,unknown_person

C:\Users\rez>
```

2. Running the command `face_detection` with a folder containing an image full of people yields multiple results, each being the coordinates of a detected face in the photograph.

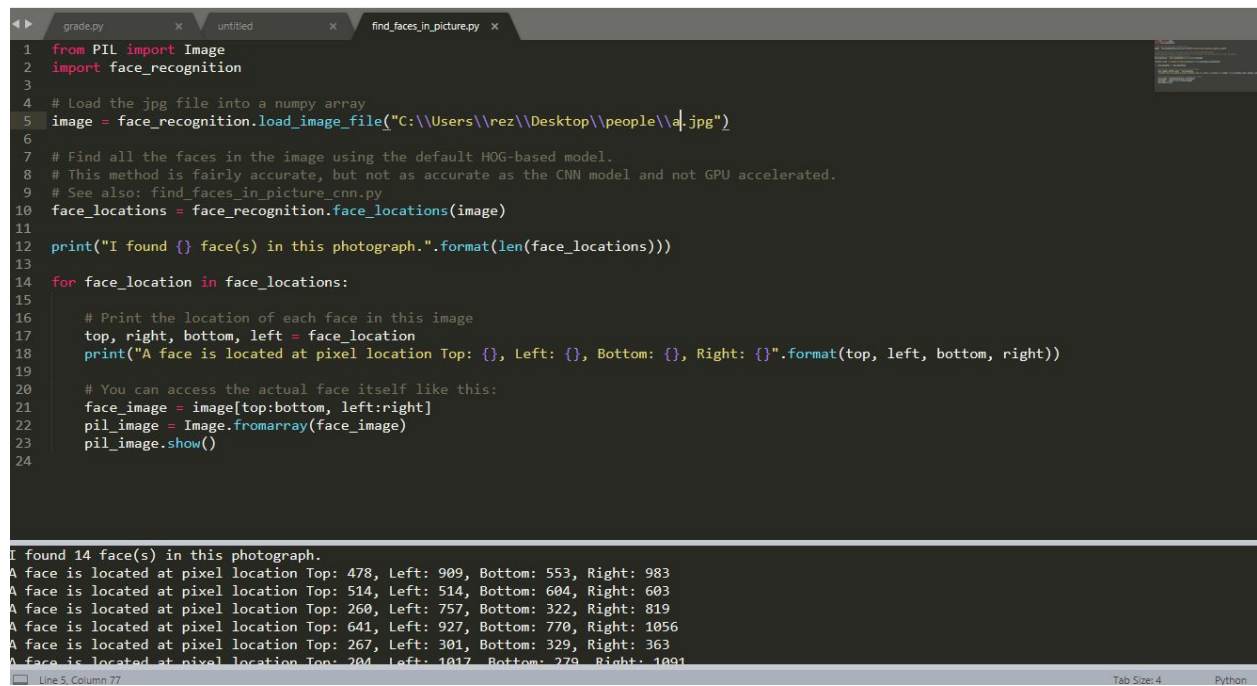


```
Command Prompt
C:\Users\rez>face_detection C:\Users\rez\Desktop\people
C:\Users\rez\Desktop\people\1.jpg,477,981,549,909
C:\Users\rez\Desktop\people\1.jpg,205,1093,277,1021
C:\Users\rez\Desktop\people\1.jpg,515,601,601,515
C:\Users\rez\Desktop\people\1.jpg,269,373,341,301
C:\Users\rez\Desktop\people\1.jpg,626,1057,775,908
C:\Users\rez\Desktop\people\1.jpg,277,237,349,165
C:\Users\rez\Desktop\people\1.jpg,293,693,365,621
C:\Users\rez\Desktop\people\1.jpg,253,821,325,749
C:\Users\rez\Desktop\people\1.jpg,307,158,411,54
C:\Users\rez\Desktop\people\1.jpg,261,501,333,429
C:\Users\rez\Desktop\people\1.jpg,389,365,461,293
C:\Users\rez\Desktop\people\1.jpg,246,966,333,880
C:\Users\rez\Desktop\people\1.jpg,357,517,429,445

C:\Users\rez>
```

3. By running `facerec_from_webcam_faster.py` from the command line you can run face recognition on webcam. Please look at [webcam.gif](#)

4. Running `find_faces_in_picture.py` in python. This yields 14 detected faces for `a.jpg`.



The screenshot shows a Python IDE with three tabs: `grade.py`, `untitled`, and `find_faces_in_picture.py`. The `find_faces_in_picture.py` tab is active, displaying the following code:

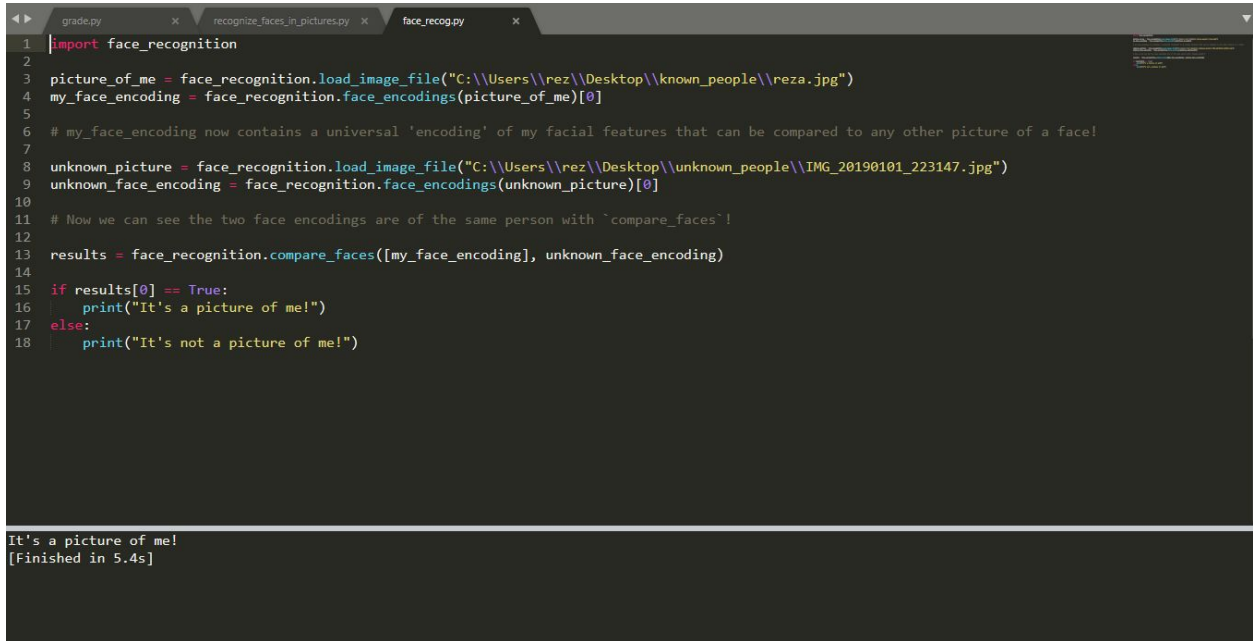
```
1 from PIL import Image
2 import face_recognition
3
4 # Load the jpg file into a numpy array
5 image = face_recognition.load_image_file("C:\\Users\\rez\\Desktop\\people\\a.jpg")
6
7 # Find all the faces in the image using the default HOG-based model.
8 # This method is fairly accurate, but not as accurate as the CNN model and not GPU accelerated.
9 # See also: find_faces_in_picture_cnn.py
10 face_locations = face_recognition.face_locations(image)
11
12 print("I found {} face(s) in this photograph.".format(len(face_locations)))
13
14 for face_location in face_locations:
15     # Print the location of each face in this image
16     top, right, bottom, left = face_location
17     print("A face is located at pixel location Top: {}, Left: {}, Bottom: {}, Right: {}".format(top, left, bottom, right))
18
19     # You can access the actual face itself like this:
20     face_image = image[top:bottom, left:right]
21     pil_image = Image.fromarray(face_image)
22     pil_image.show()
23
24
```

The output of the script is displayed in the console at the bottom of the IDE:

```
I found 14 face(s) in this photograph.
A face is located at pixel location Top: 478, Left: 909, Bottom: 553, Right: 983
A face is located at pixel location Top: 514, Left: 514, Bottom: 604, Right: 603
A face is located at pixel location Top: 260, Left: 757, Bottom: 322, Right: 819
A face is located at pixel location Top: 641, Left: 927, Bottom: 770, Right: 1056
A face is located at pixel location Top: 267, Left: 301, Bottom: 329, Right: 363
A face is located at pixel location Top: 284, Left: 1017, Bottom: 279, Right: 1091
```

The status bar at the bottom indicates "Line 5, Column 77" and "Tab Size: 4 Python".

Running face\_recog.py recognizes the face by yielding 'It is a picture of me' if the program finds the same person in the unknown file.



The image shows a screenshot of a Python IDE with three tabs: 'grade.py', 'recognize\_faces\_in\_pictures.py', and 'face\_recog.py'. The 'face\_recog.py' tab is active, displaying a Python script for face recognition. The script imports the 'face\_recognition' library, loads a known face image ('reza.jpg'), encodes it, loads an unknown face image ('IMG\_20190101\_223147.jpg'), encodes it, and then compares the two encodings. If they match, it prints 'It's a picture of me!'. The output window at the bottom shows the message 'It's a picture of me!' and '[Finished in 5.4s]'.

```
1 import face_recognition
2
3 picture_of_me = face_recognition.load_image_file("C:\\Users\\rez\\Desktop\\known_people\\reza.jpg")
4 my_face_encoding = face_recognition.face_encodings(picture_of_me)[0]
5
6 # my_face_encoding now contains a universal 'encoding' of my facial features that can be compared to any other picture of a face!
7
8 unknown_picture = face_recognition.load_image_file("C:\\Users\\rez\\Desktop\\unknown_people\\IMG_20190101_223147.jpg")
9 unknown_face_encoding = face_recognition.face_encodings(unknown_picture)[0]
10
11 # Now we can see the two face encodings are of the same person with 'compare_faces'!
12
13 results = face_recognition.compare_faces([my_face_encoding], unknown_face_encoding)
14
15 if results[0] == True:
16     print("It's a picture of me!")
17 else:
18     print("It's not a picture of me!")
```

It's a picture of me!  
[Finished in 5.4s]