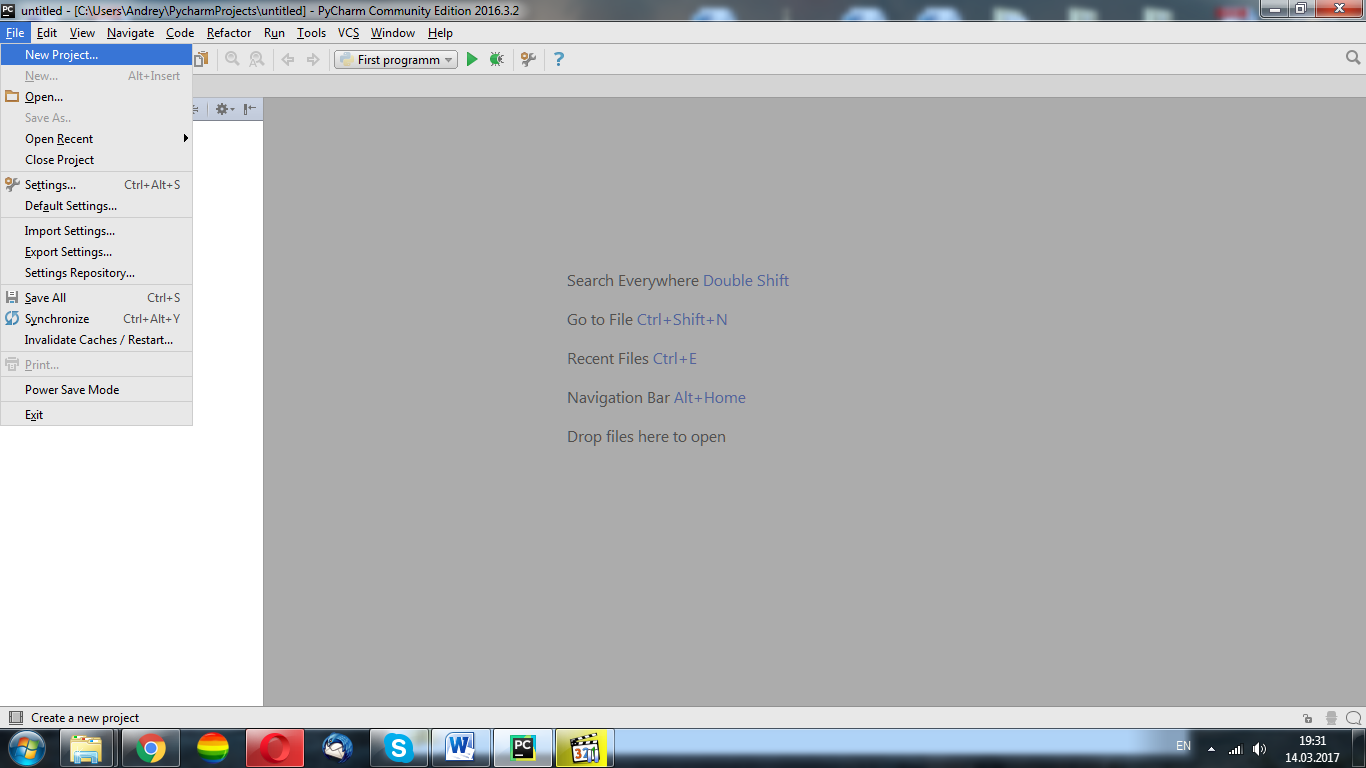
Урок № 6!

Система обнаружения лиц.

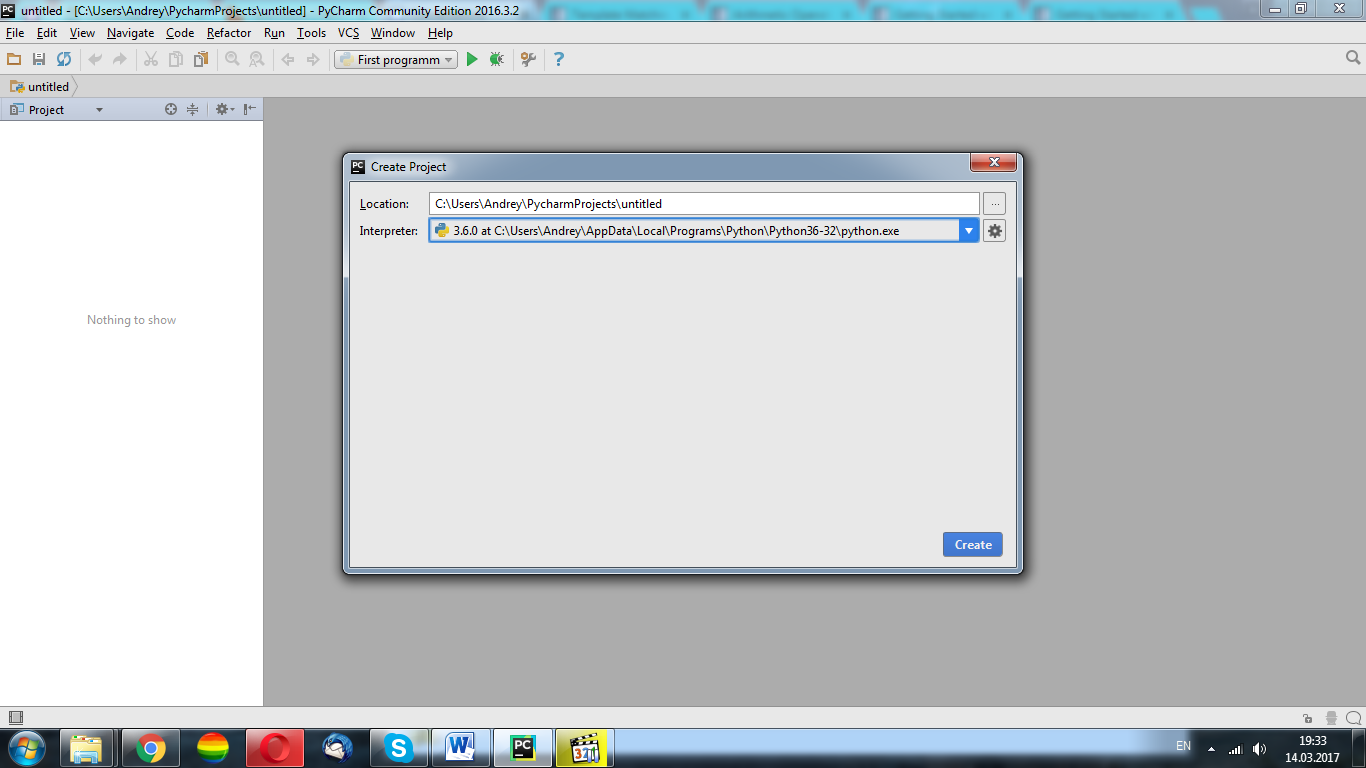
Запустите PyCharm Community Edition

Создайте проект. Нажмите на вкладку File и выберите первый пункт New Project



В следующем окне в первой строке записывается путь, где сохранится проект.

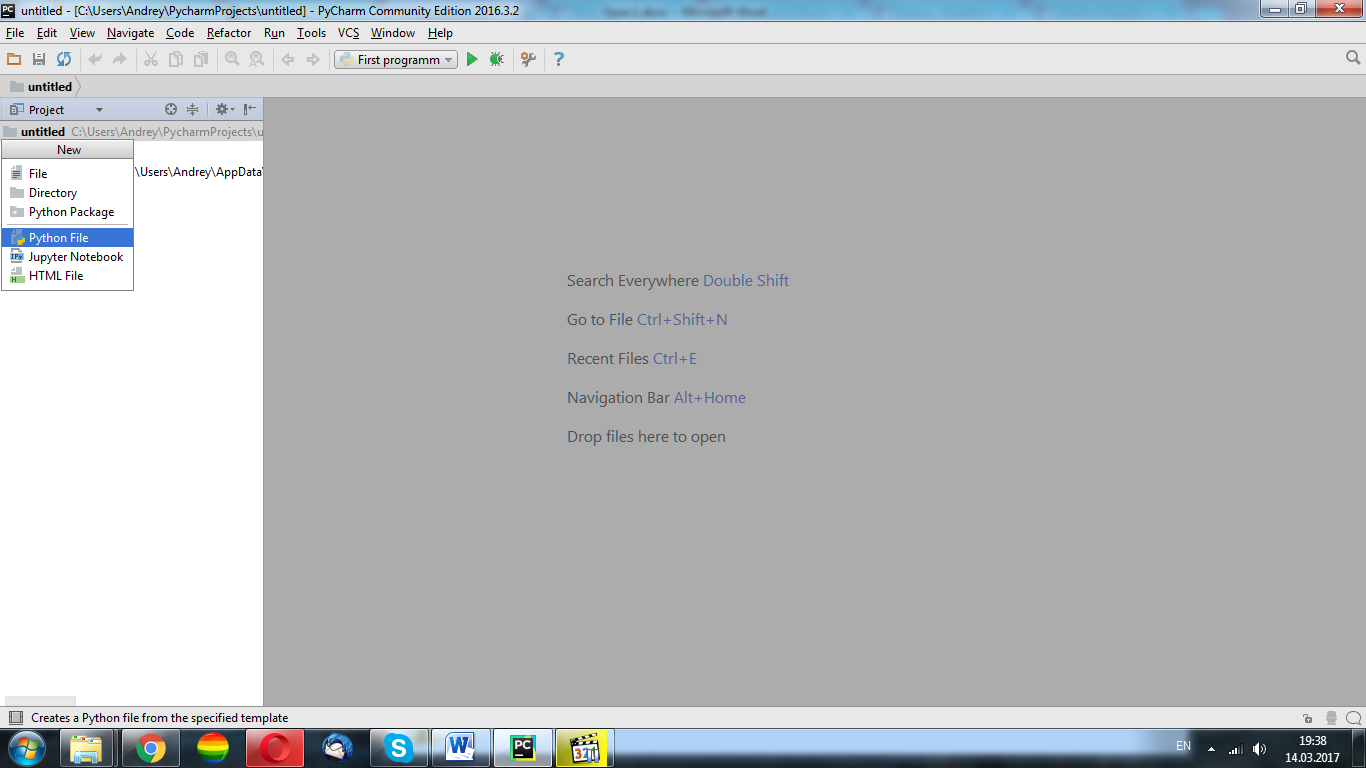
Во второй строке версия python. Выберите последнюю версию 3.6. Нажмите Create



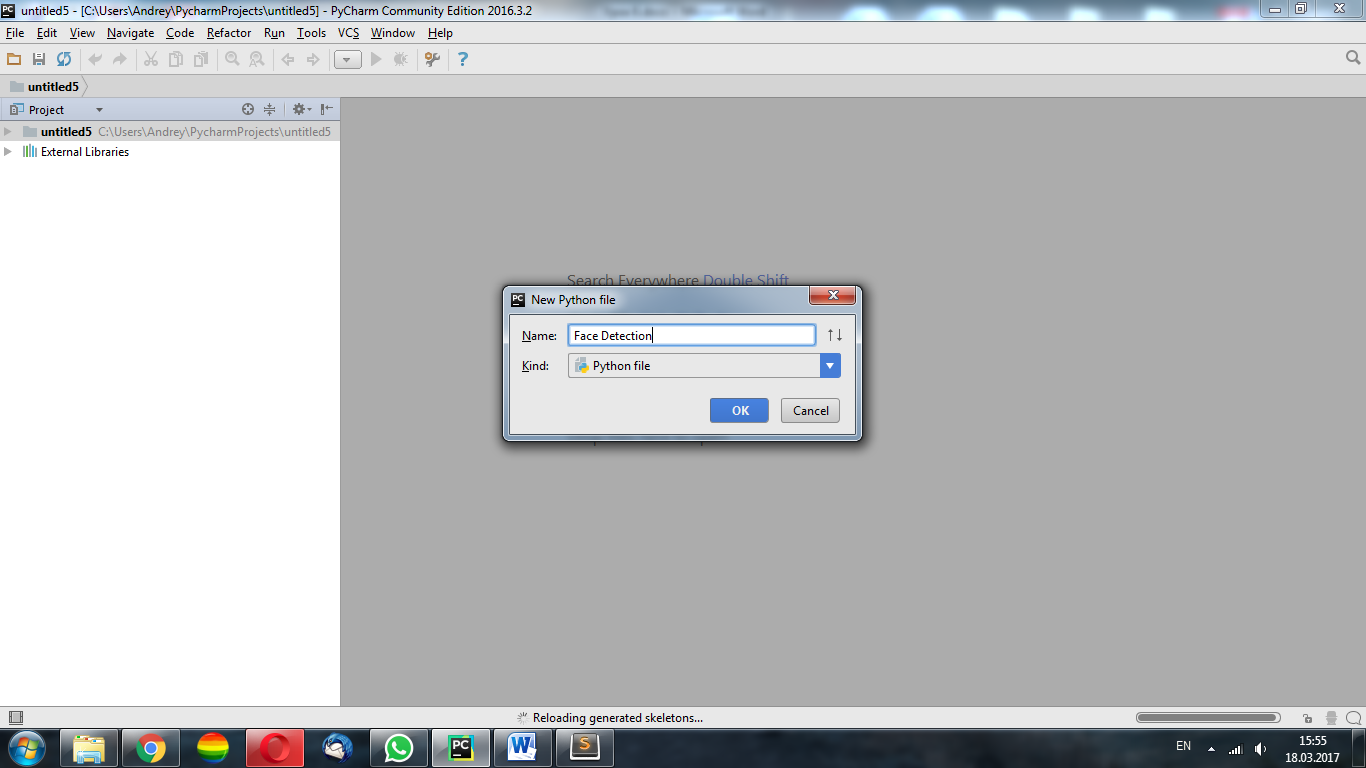
Далее создадим файл, в котором будем работать. Во вкладке File нажмите New…



Python file

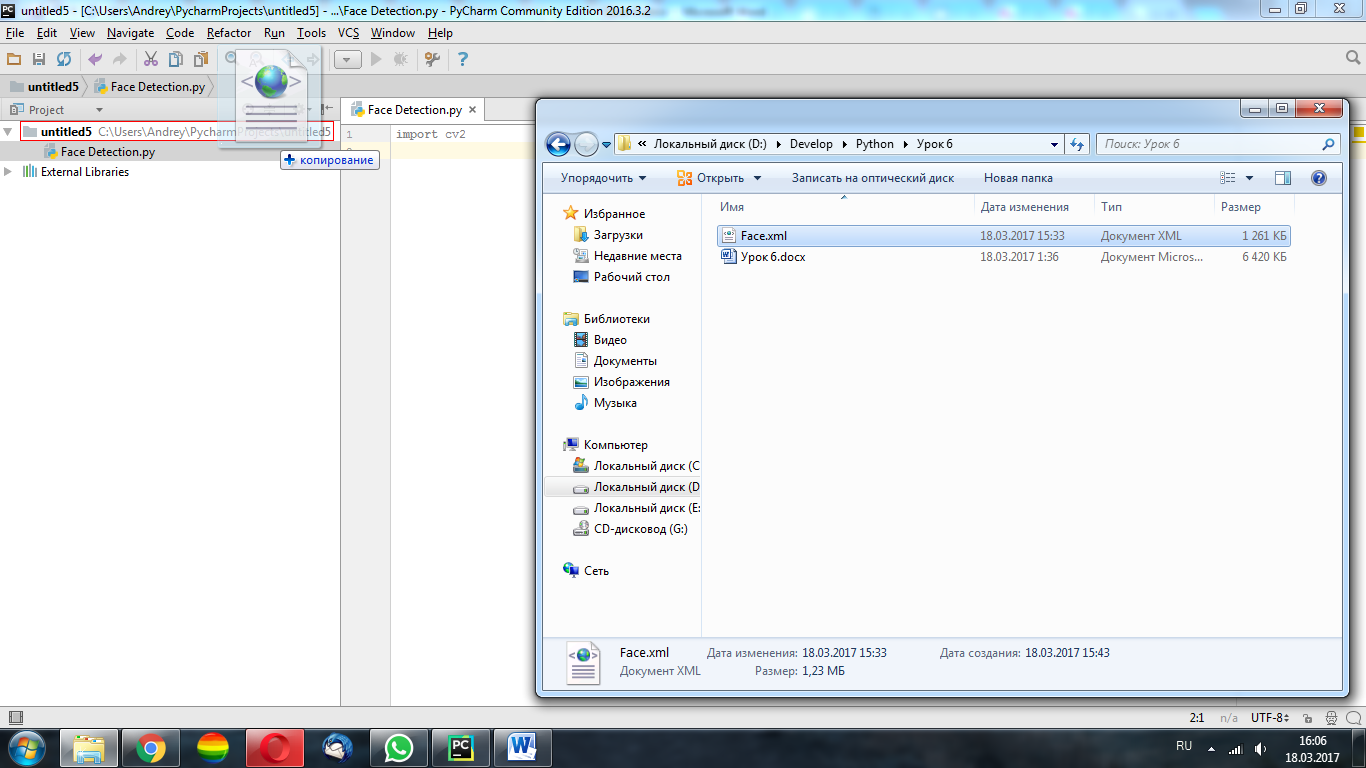


Назовем этот файл Face Detection

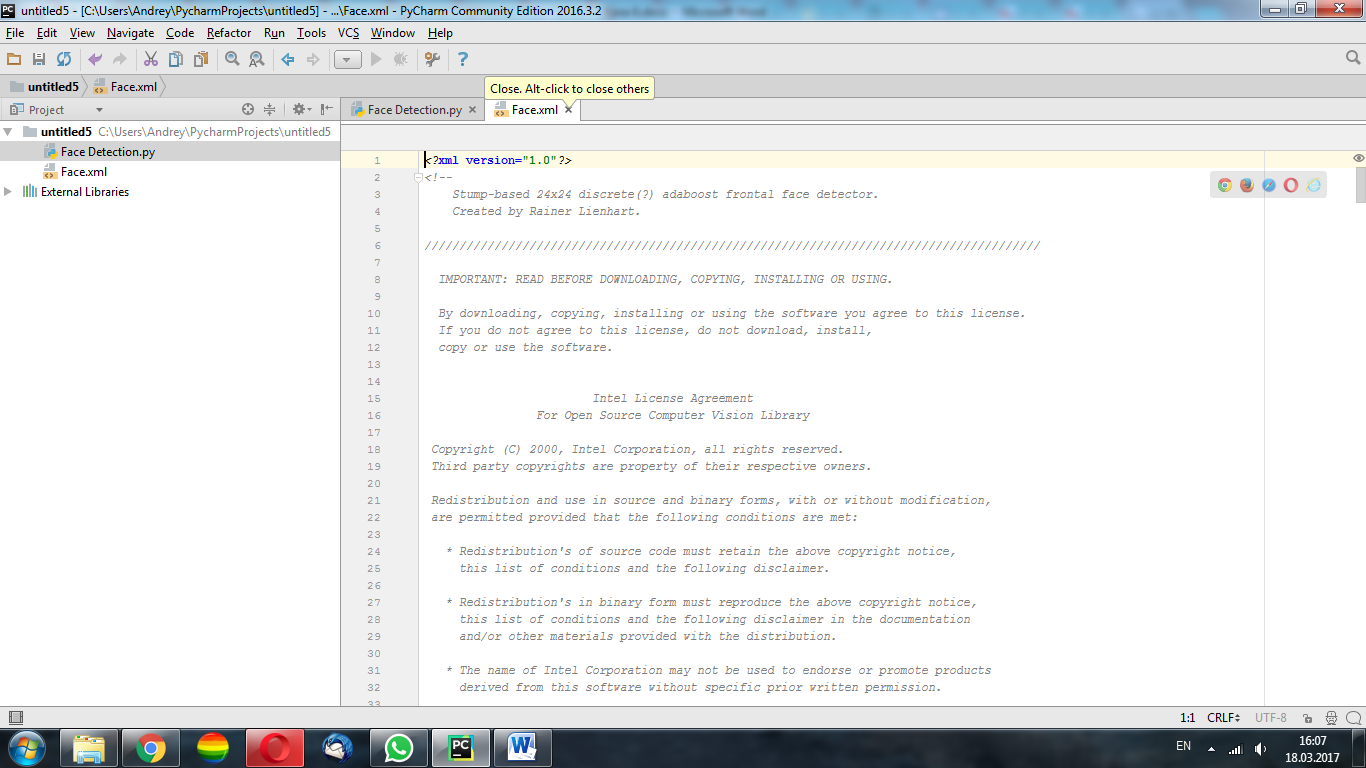


Мы будем использовать библиотеку OpenCV, она способна определить наличие предмета на снимке, например, лицо человека.

Начнем с подключения библиотеки cv2 и скопируем в папку с проектом из папки с уроком, файл Face.xml Это готовое решение, оно нам поможет с определением лиц.



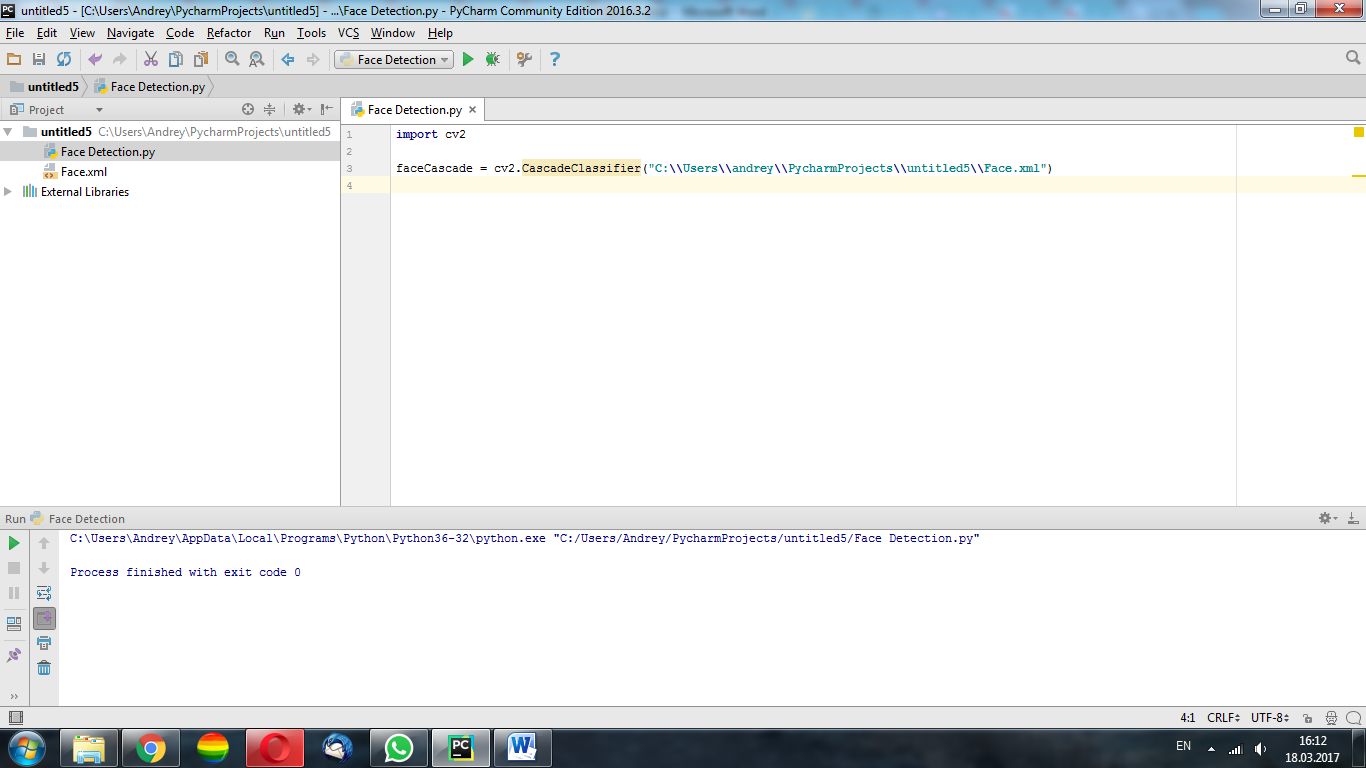
Открывшуюся вкладку можете закрыть



Далее нам нужно записать этот файл в переменную faceCascade используя функцию cv2.CascadeClassidier. В скобках необходимо указать полный адрес до этого файла

faceCascade = cv2.CascadeClassifier(**"C:\\Users\\andrey\\PycharmProjects\\untitled5\\Face.xml"**)

Путь к папке с проектом, вы можете видеть на скрине. В моем случае имя пользователя Andrey у вас может быть ваше имя или admin, а так же обратите внимание на название проекта



Подключим веб камеру

video\_capture = cv2.VideoCapture(0)

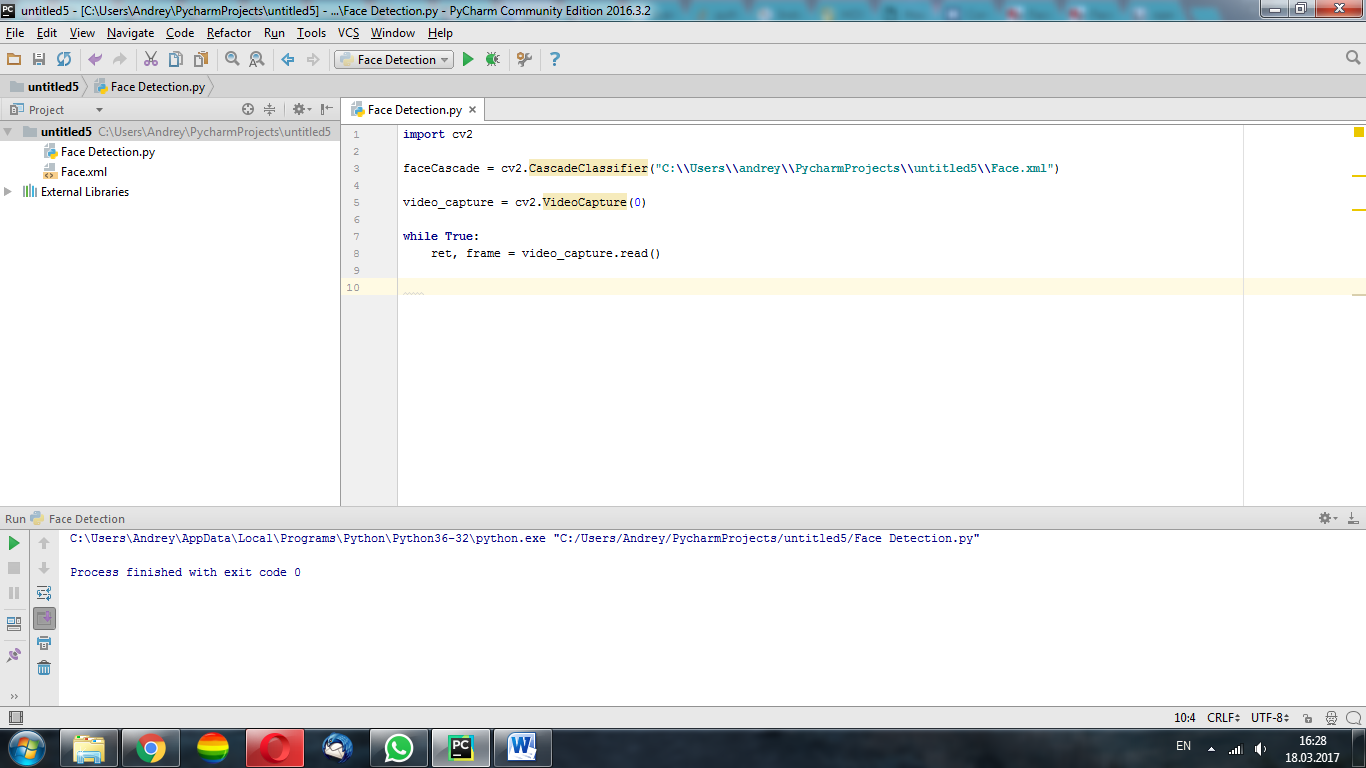


Откроем условие

**while True**:

Далее захват с камеры кадр за кадром

ret, frame = video\_capture.read()



Конвертируем полученное изображение в оттенки серого

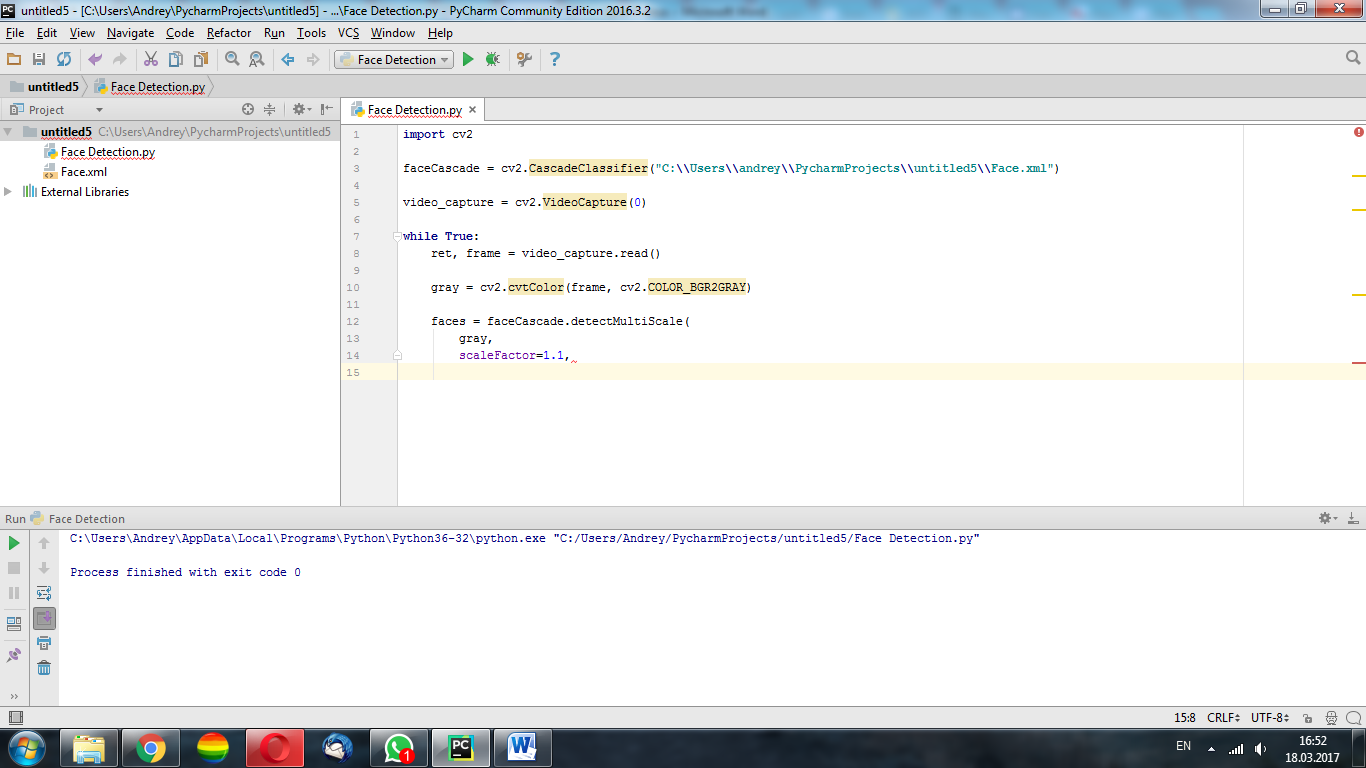
gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

Еще нам нужно задать параметры для определения лиц

faces = faceCascade.detectMultiScale(  
 gray,  
 scaleFactor=1.1,

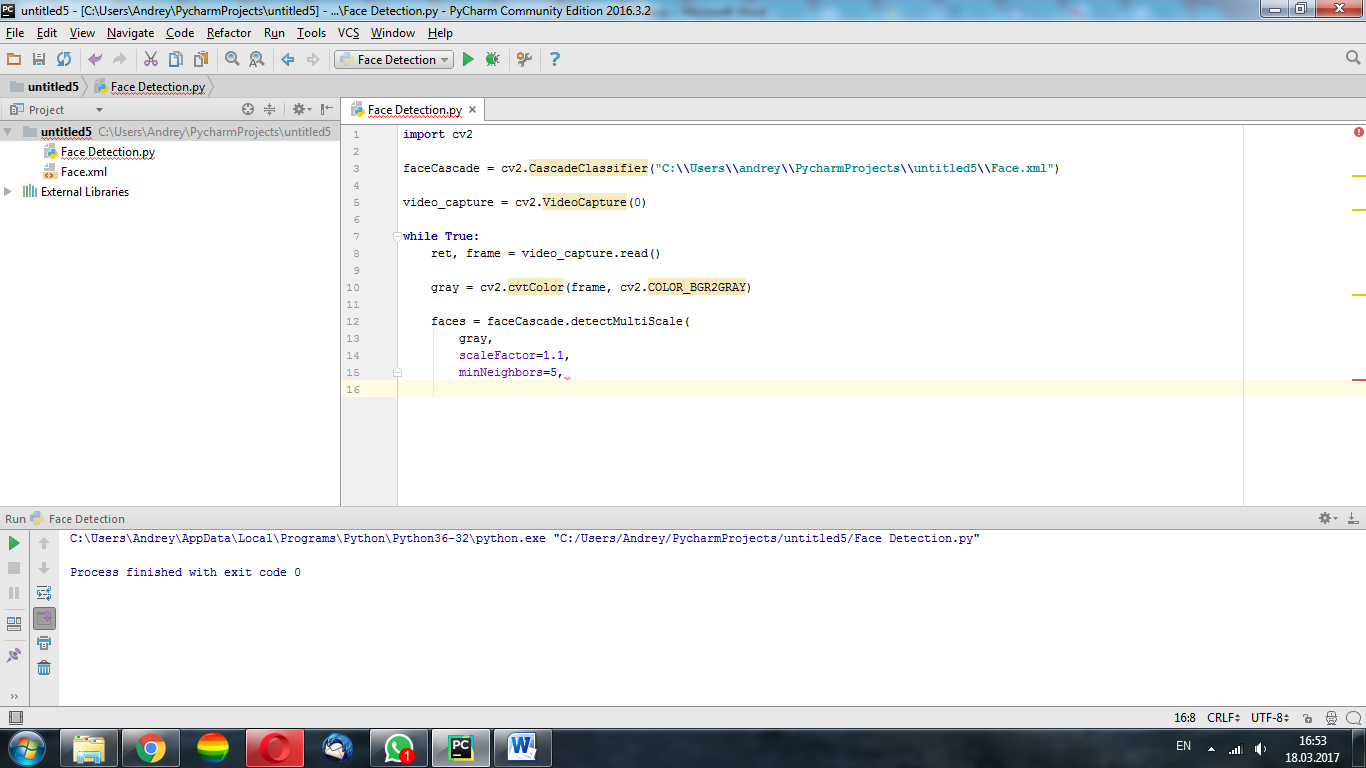
ScaleFactor - параметр, определяющий, насколько уменьшен размер изображения на каждой шкале изображения. В xml файле, который мы используем стандартные параметры немного меньше. Увеличим их на 10 %

scaleFactor=1.1,



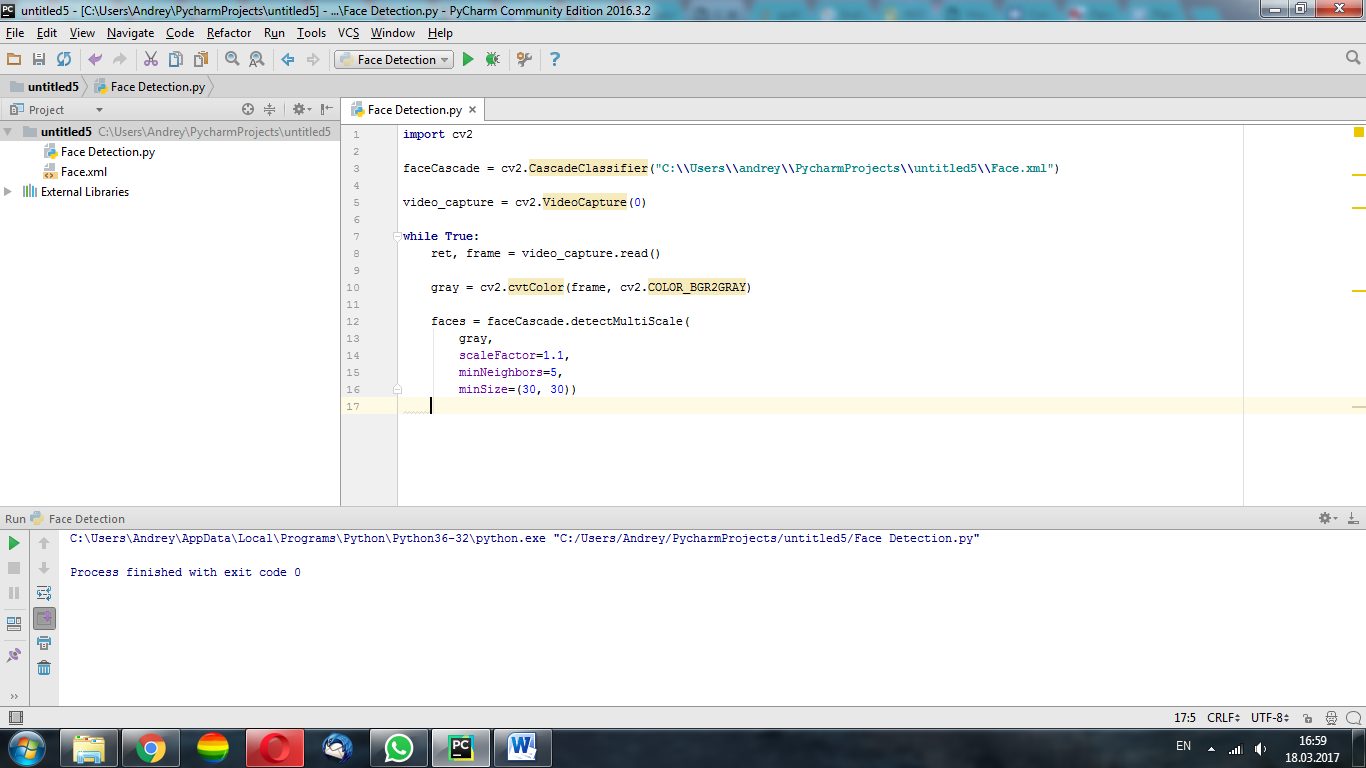
MinNeighbors - Параметр, указывающий, сколько соседей должно быть у каждого прямоугольника-кандидата, чтобы сохранить его. 5 для нас должно подойти.

minNeighbors=5,



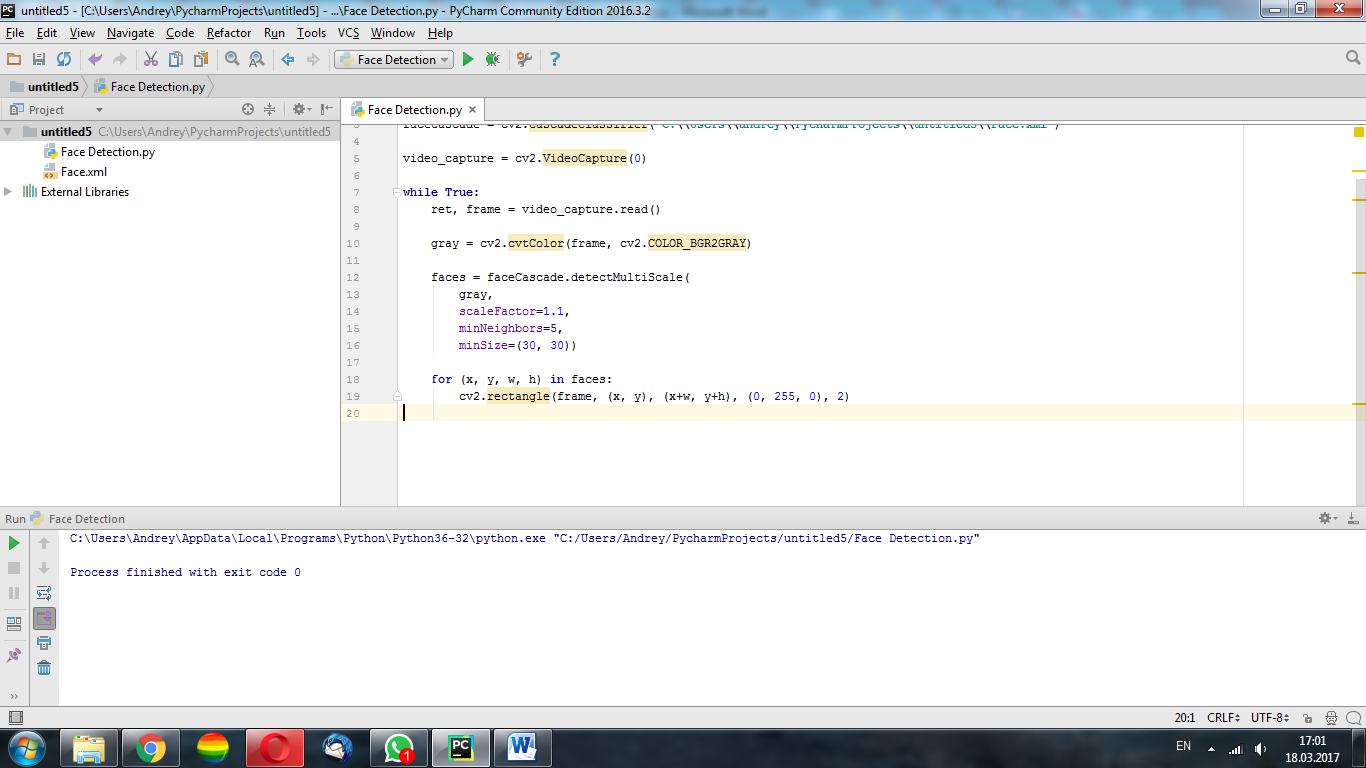
MinSize - Минимально возможный размер объекта. Объекты меньшего размера игнорируются.

minSize=(30, 30))



Далее нарисуем прямоугольник вокруг лица. Сделаем его зеленого цвета.

**for** (x, y, w, h) **in** faces:  
 cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)



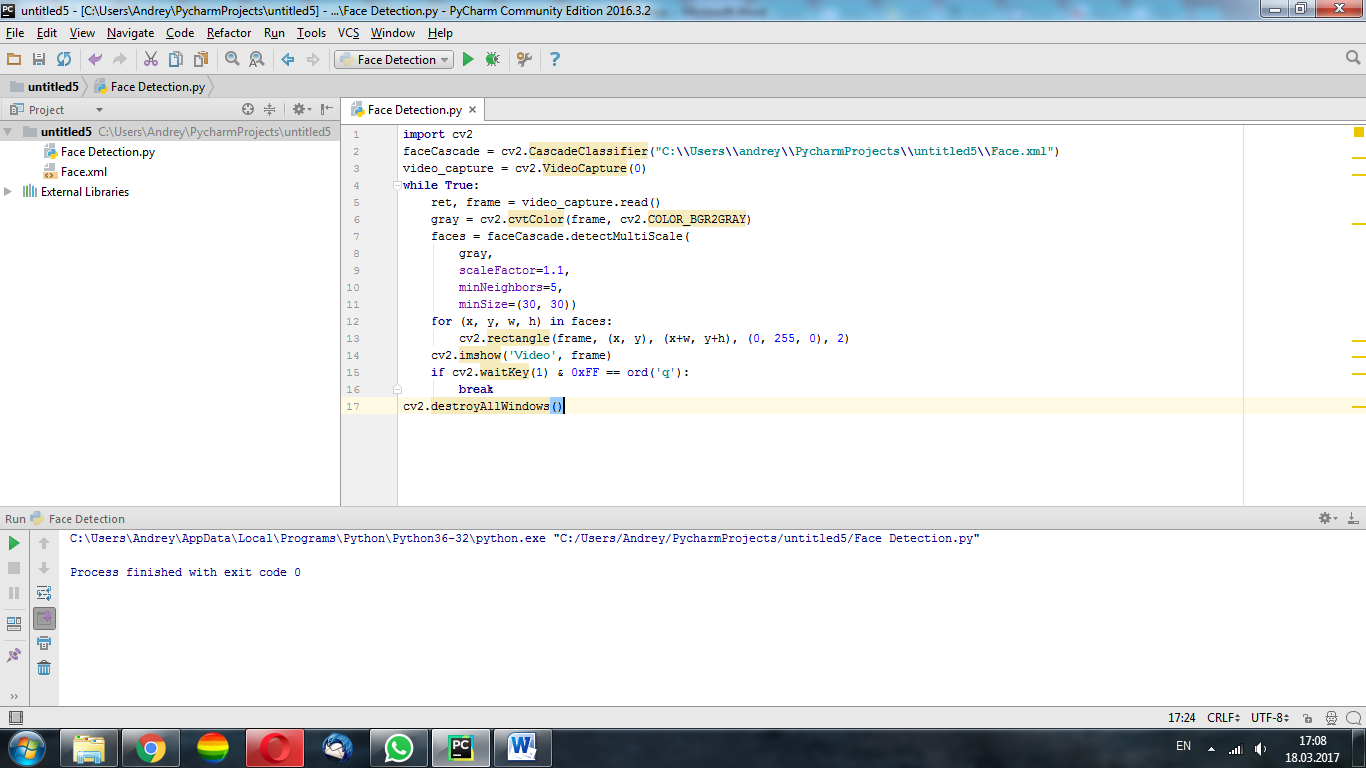
Выводим результат

cv2.imshow(**'Video'**, frame)

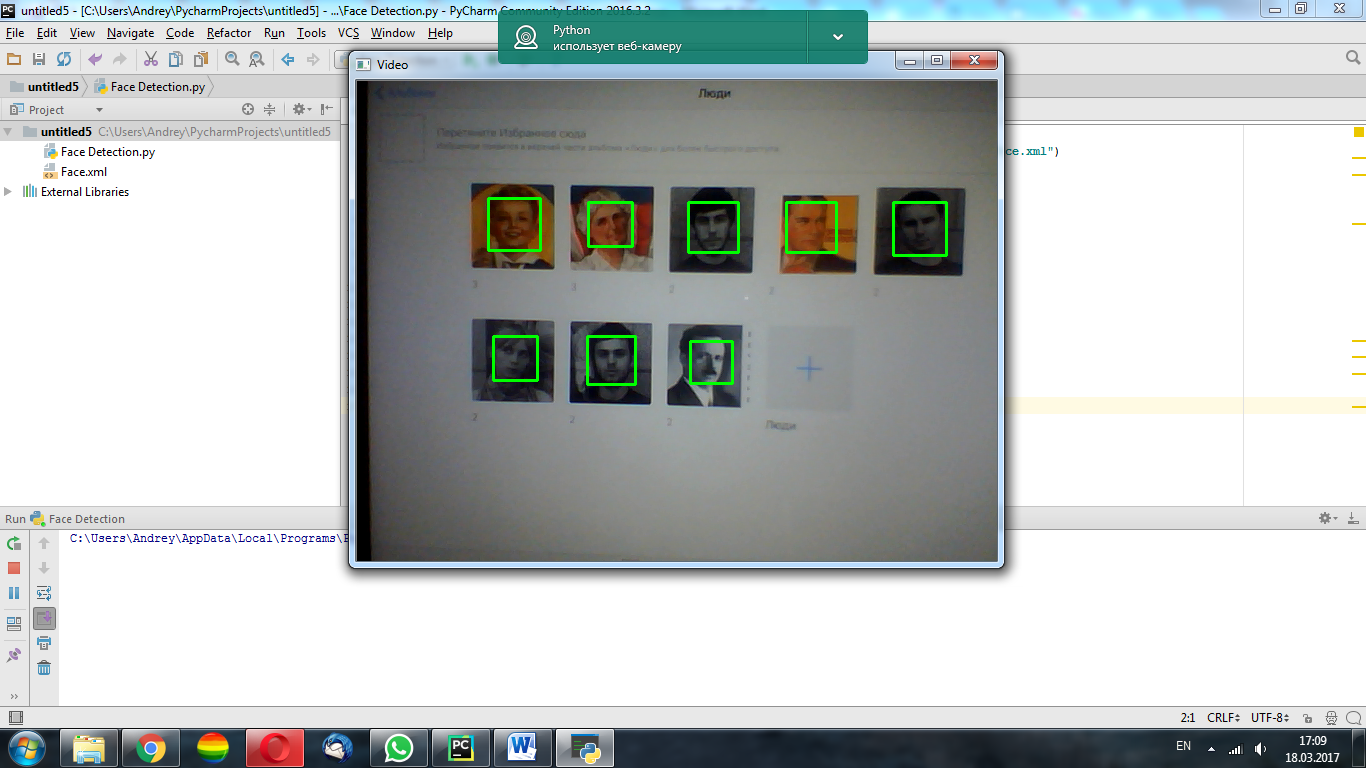


Поставим закрытие программы по нажатию на клавишу q

**if** cv2.waitKey(1) & 0xFF == ord(**'q'**):  
 **break**cv2.destroyAllWindows()

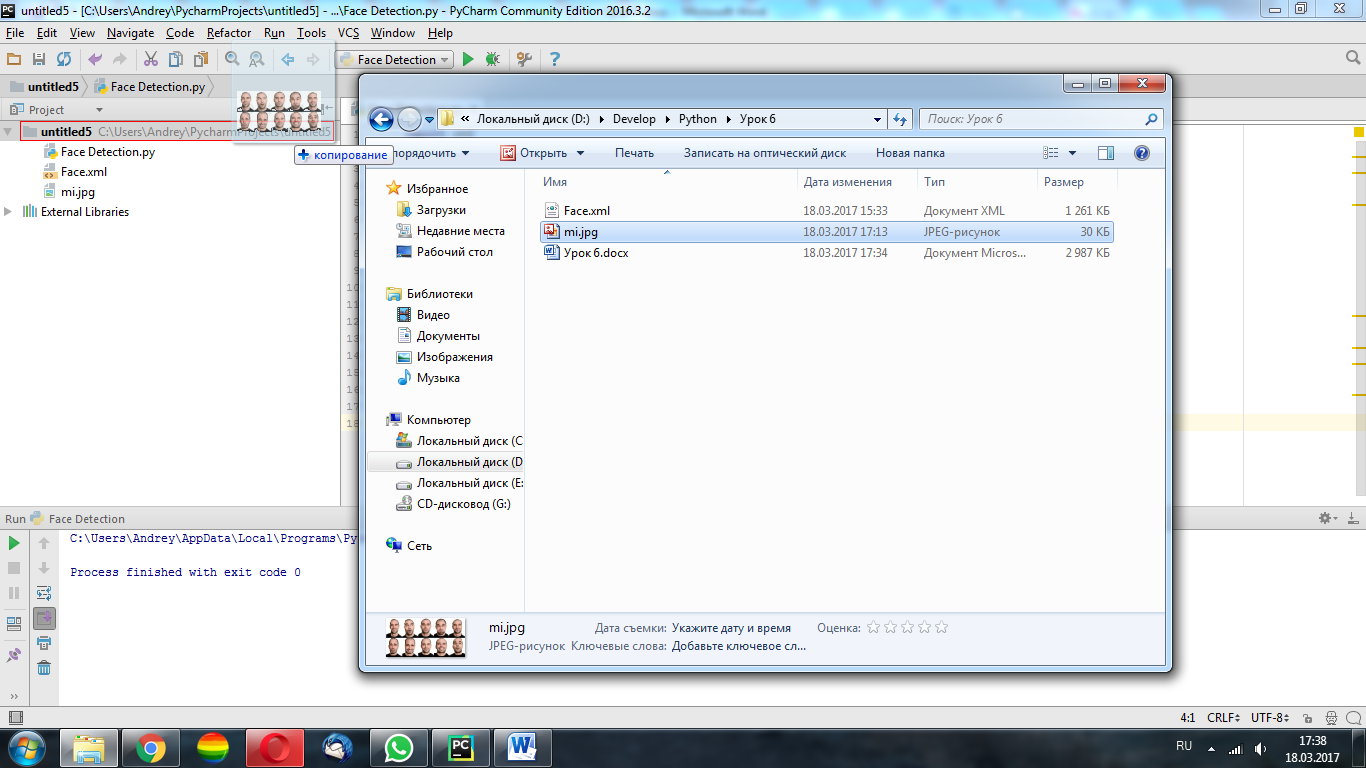


Финальный результат



Перейдем к обнаружению лиц по фото. Здесь система точно такая же.

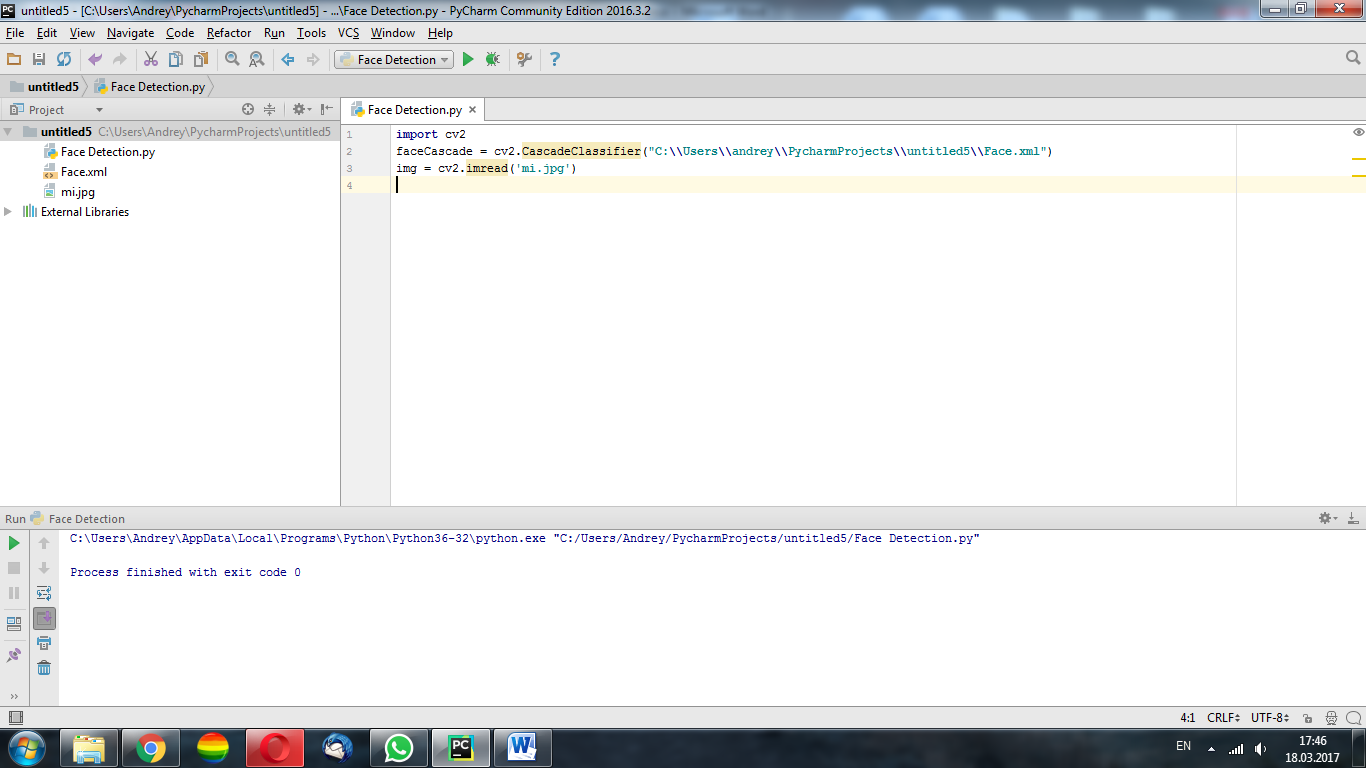
Начнем с загрузки фото с лицами. Картинка с названием mi в папке с уроком.



Оставьте первые две строчки кода.

Подключим нашу картинку

img = cv2.imread(**'mi.jpg'**)



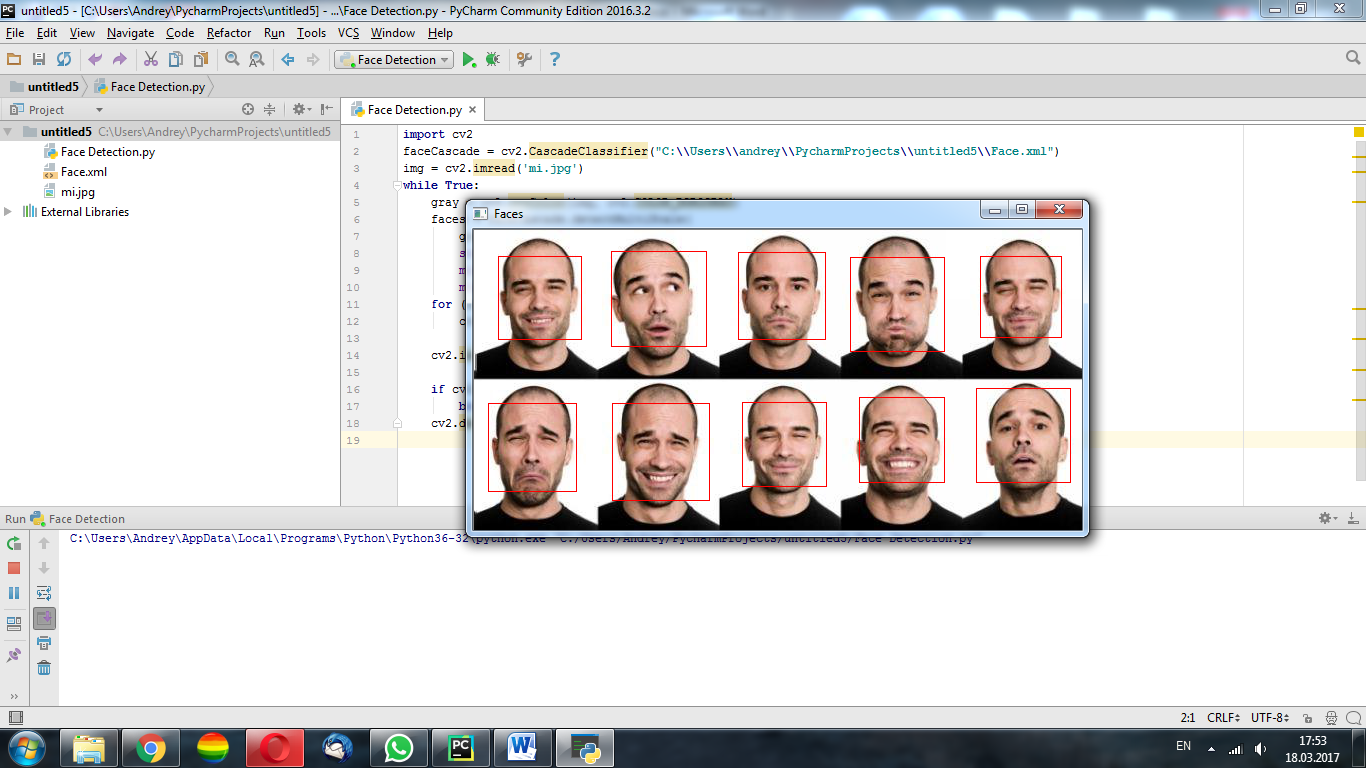
Остальные данные точно такие же, только для переменной img

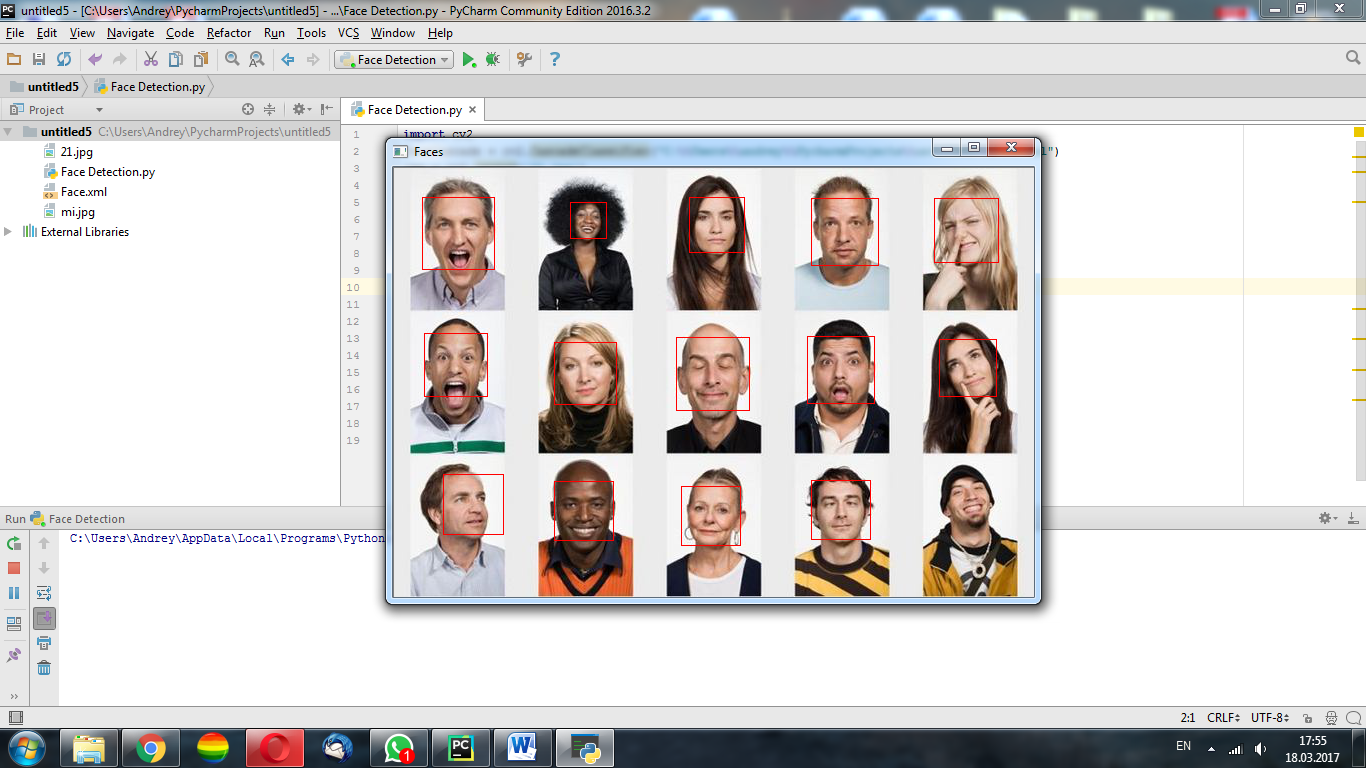
**while True**:  
 gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
 faces = faceCascade.detectMultiScale(  
 gray,  
 scaleFactor=1.1,  
 minNeighbors=5,  
 minSize=(30, 30))  
 **for** (x, y, w, h) **in** faces:  
 cv2.rectangle(img, (x, y), (x + w, y + h), (0, 0, 255), 1)  
  
 cv2.imshow(**"Faces"**, img)

и обратите внимание, что я поставил рамку толщиной в 1 пиксель и сделал ее красного цвета.



Финальный результат:





Загрузите свои картинки, фотографии, можно хоть целый клип.