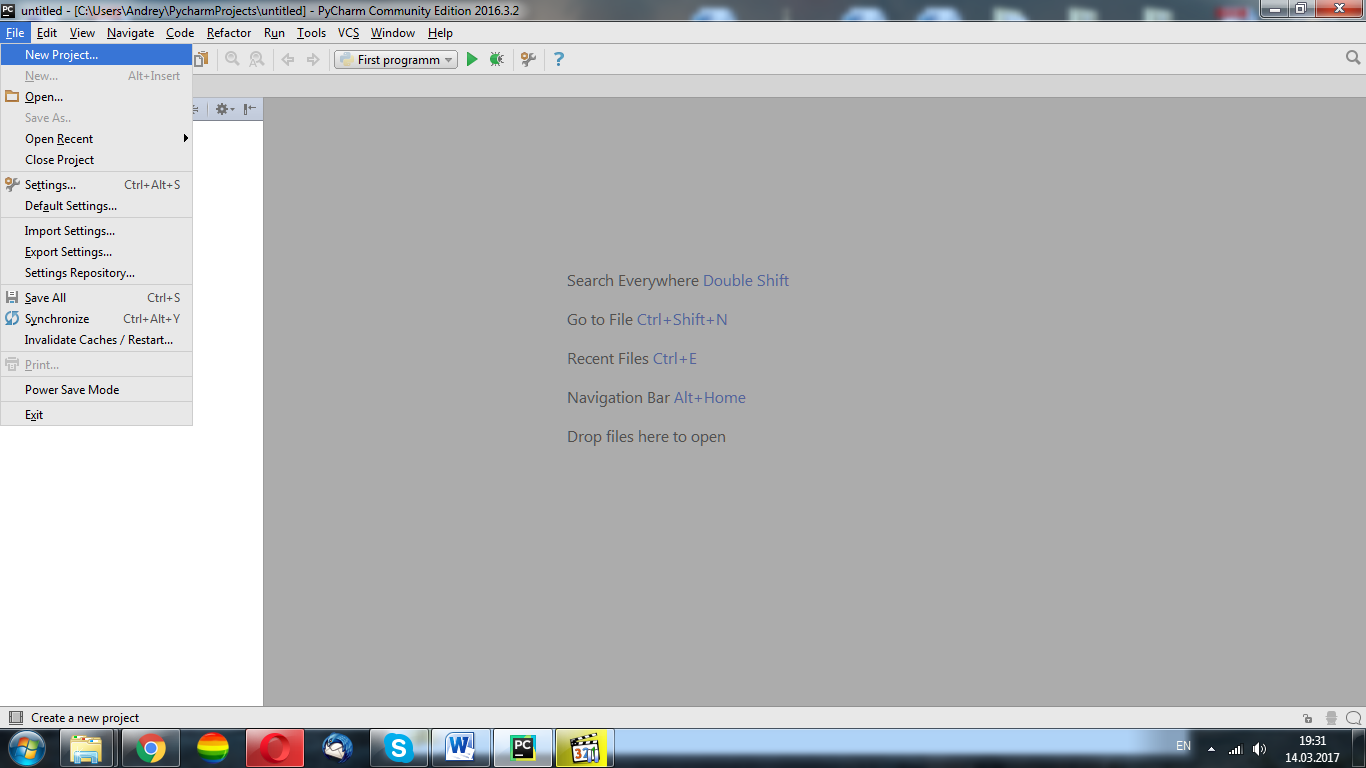
Урок № 8!

Последний урок. Детектор лиц, глаз, улыбки и не только.

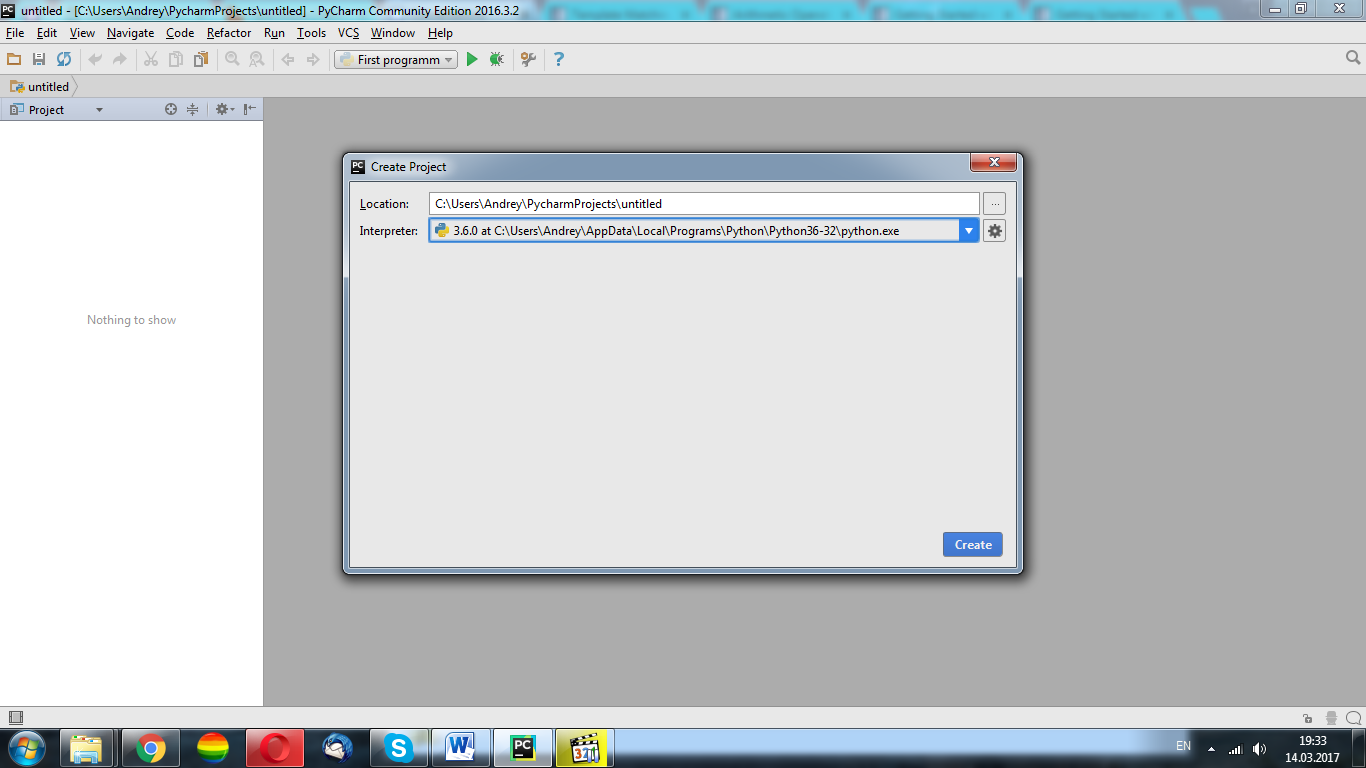
Запустите PyCharm Community Edition

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

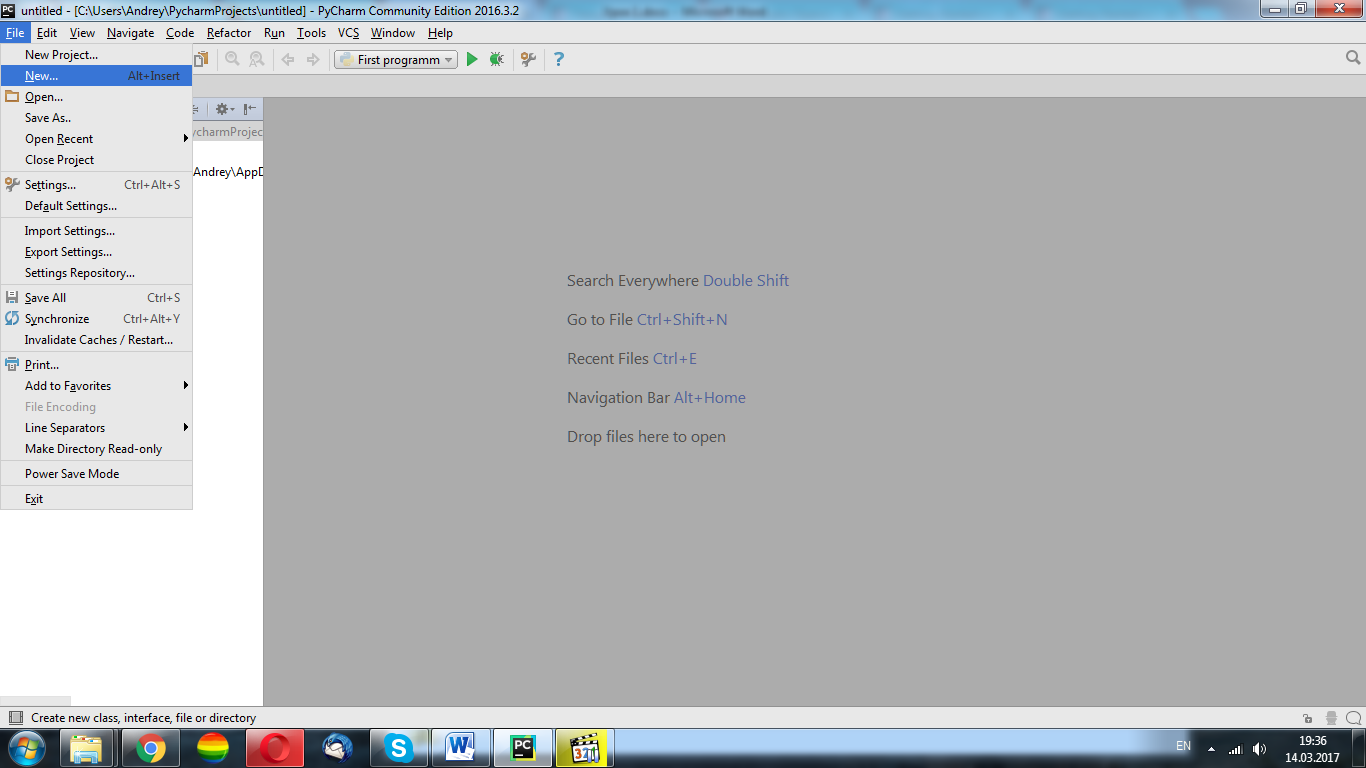


В следующем окне в первой строке записывается путь, где сохранится проект.

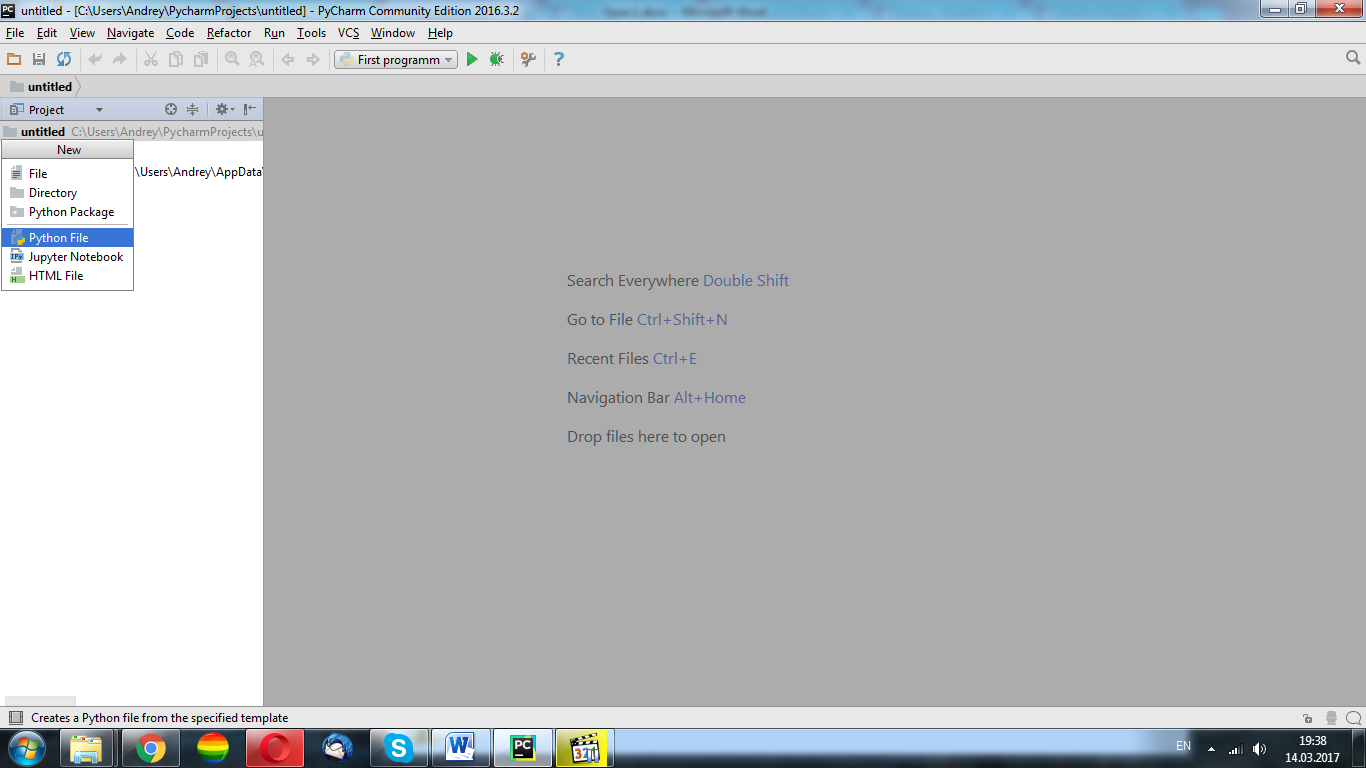
Во второй строке версия python. Выберите последнюю версию 3.6. Нажмите Create



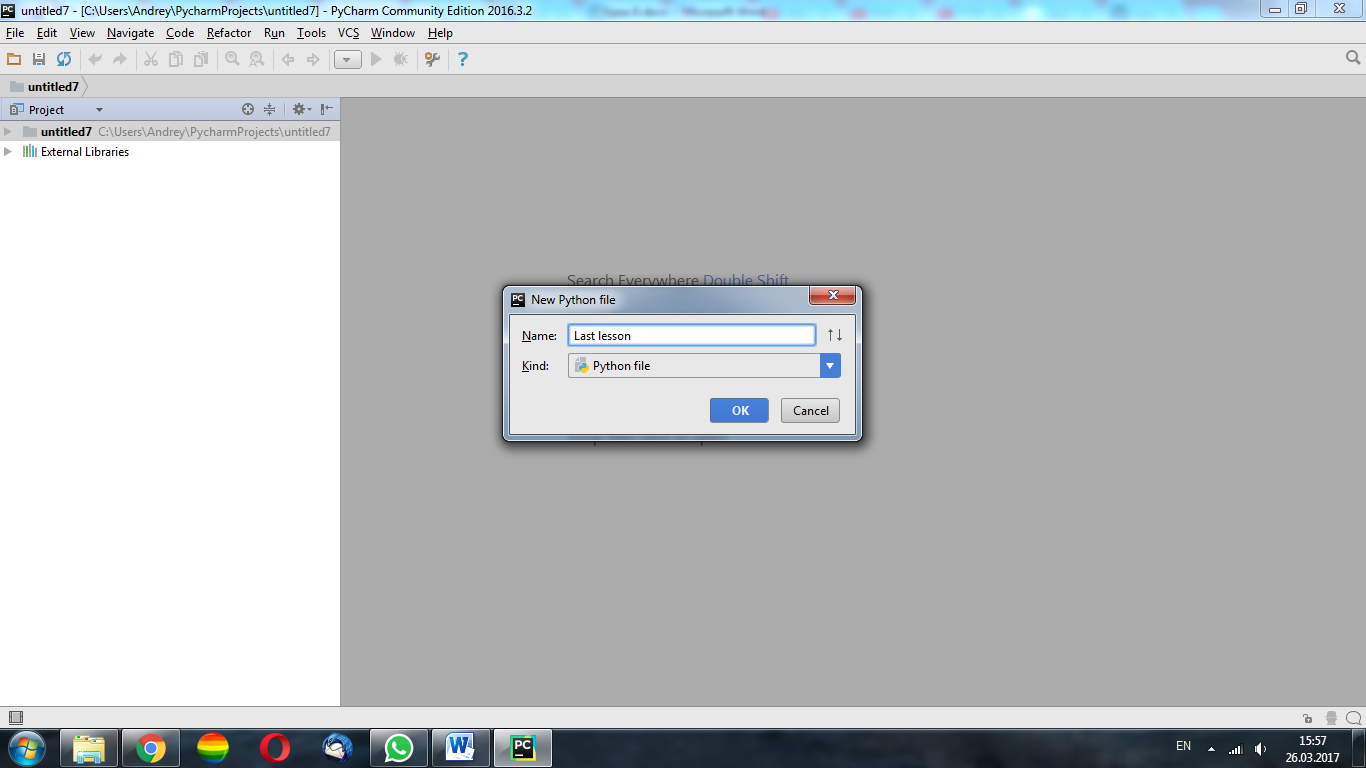
Далее создадим файл, в котором будем работать. Во вкладке File нажмите New…



Python file

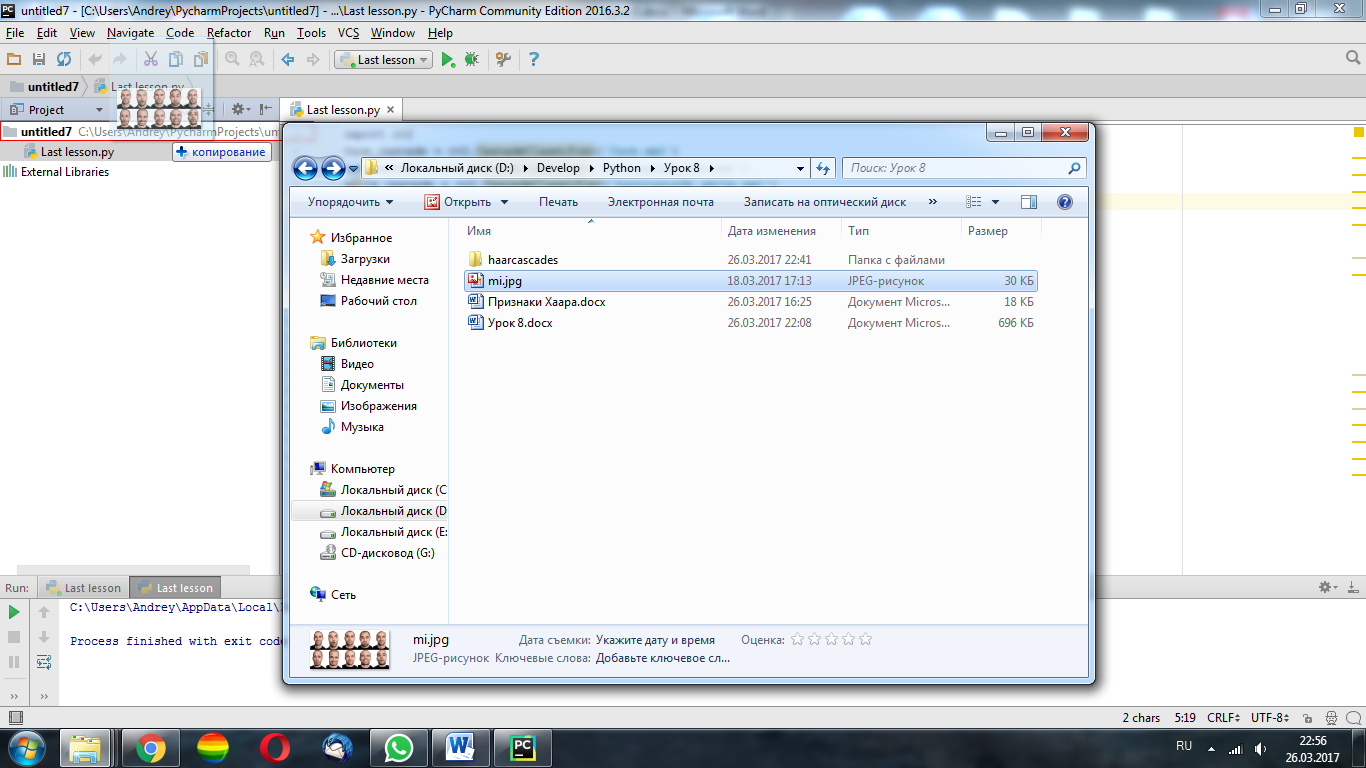


Назовем этот файл last lesson

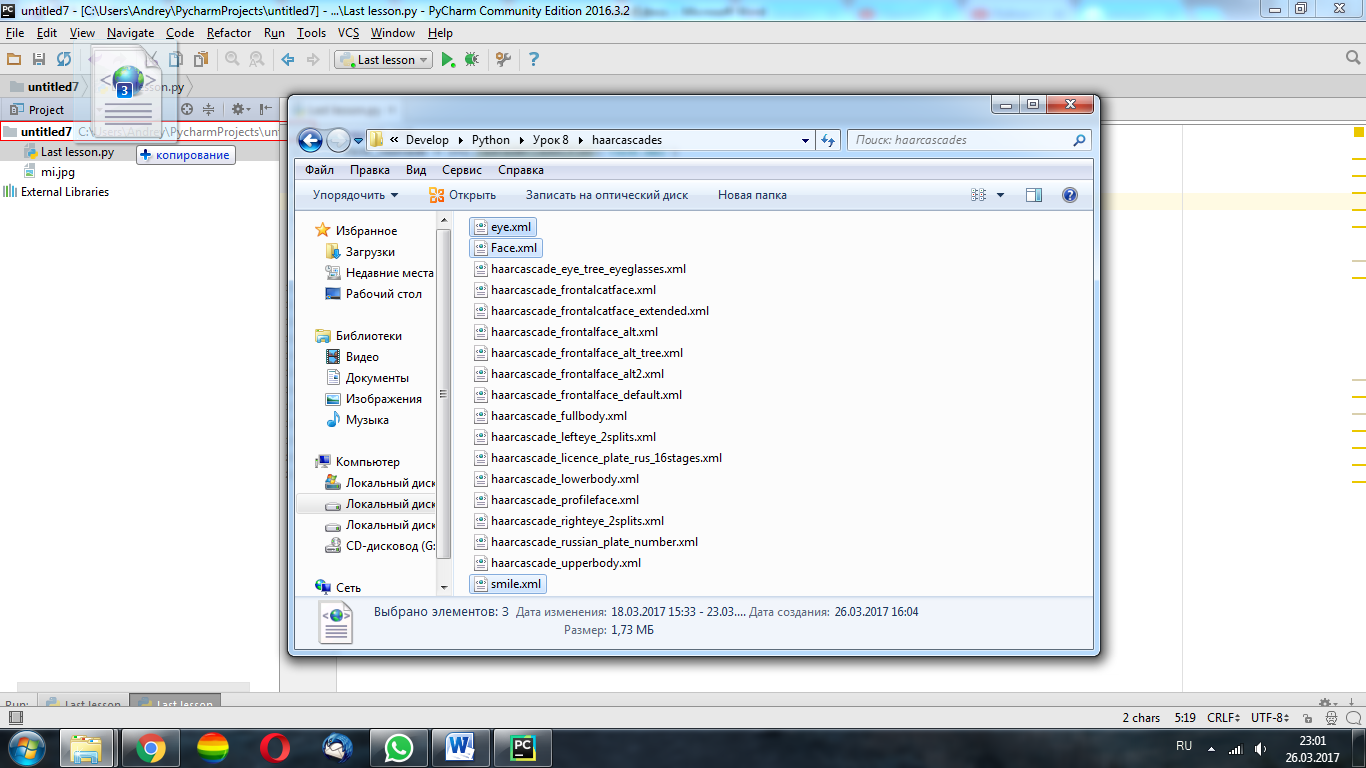


Подключите библиотеку cv2

Начнем с копирования необходимых для работы файлов. Из папки с уроком возьмите картинку «mi»



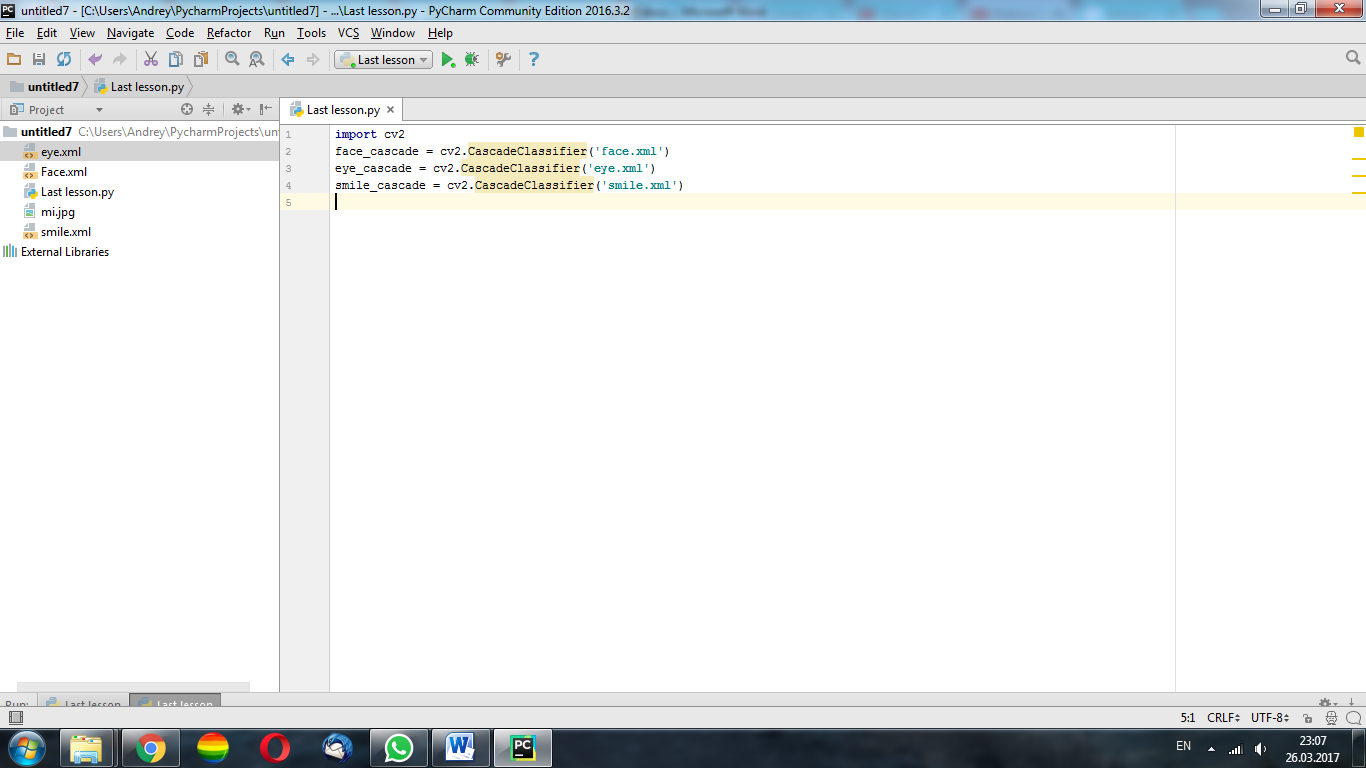
Далее из папки «haarcascades» скопируйте три файла, которые помогут библиотеке находить лица, глаза и улыбки на изображении. Названия файлов: Face, smile и eye. Выделите их и зажимая клавишу ctrl перенесите в папку с проектом



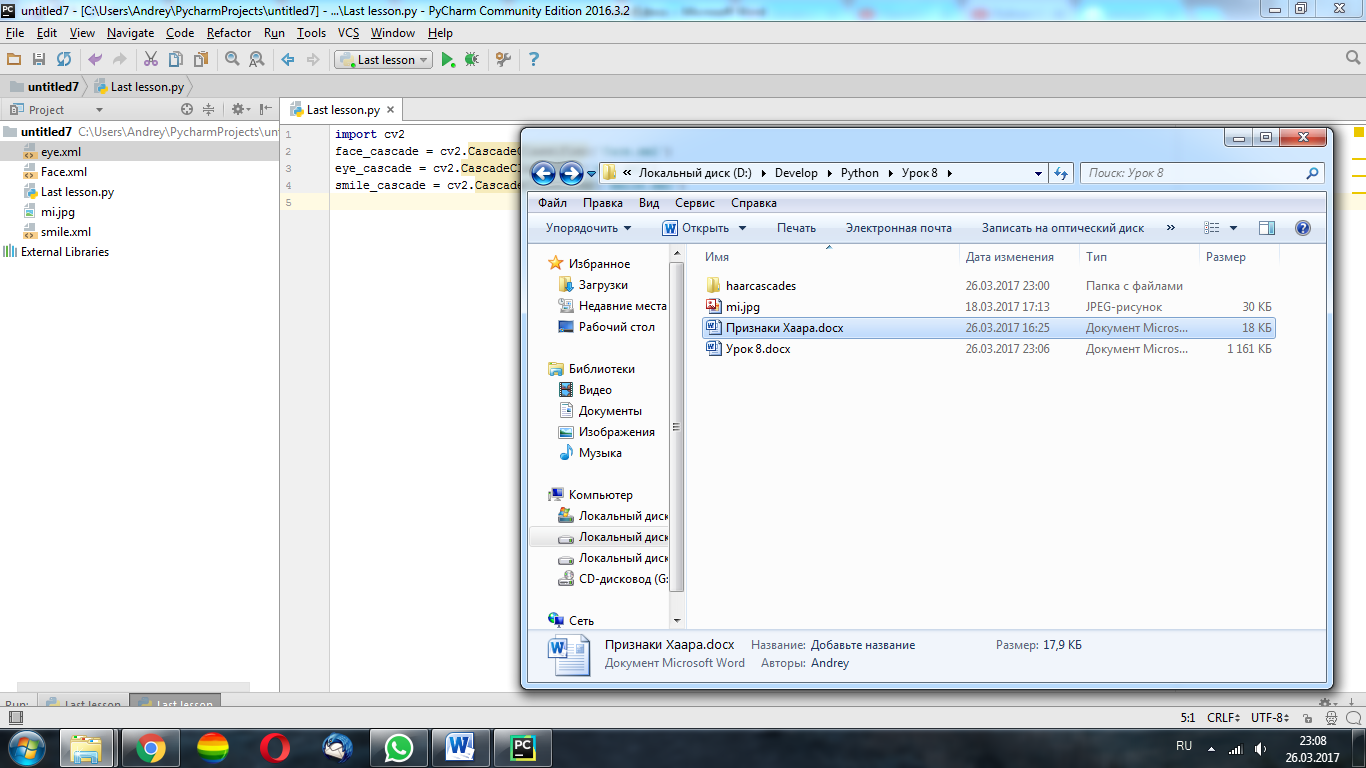
С остальными файлами, вы можете поэкспериментировать сами. В этой папке есть файлы, которые помогут найти: тело, верхнюю или только нижнюю часть тела, правый глаз, левый глаз и тд.

Теперь создаем Хаар каскад для каждого из добавленных файлов

face\_cascade = cv2.CascadeClassifier(**'face.xml'**)  
eye\_cascade = cv2.CascadeClassifier(**'eye.xml'**)  
smile\_cascade = cv2.CascadeClassifier(**'smile.xml'**)

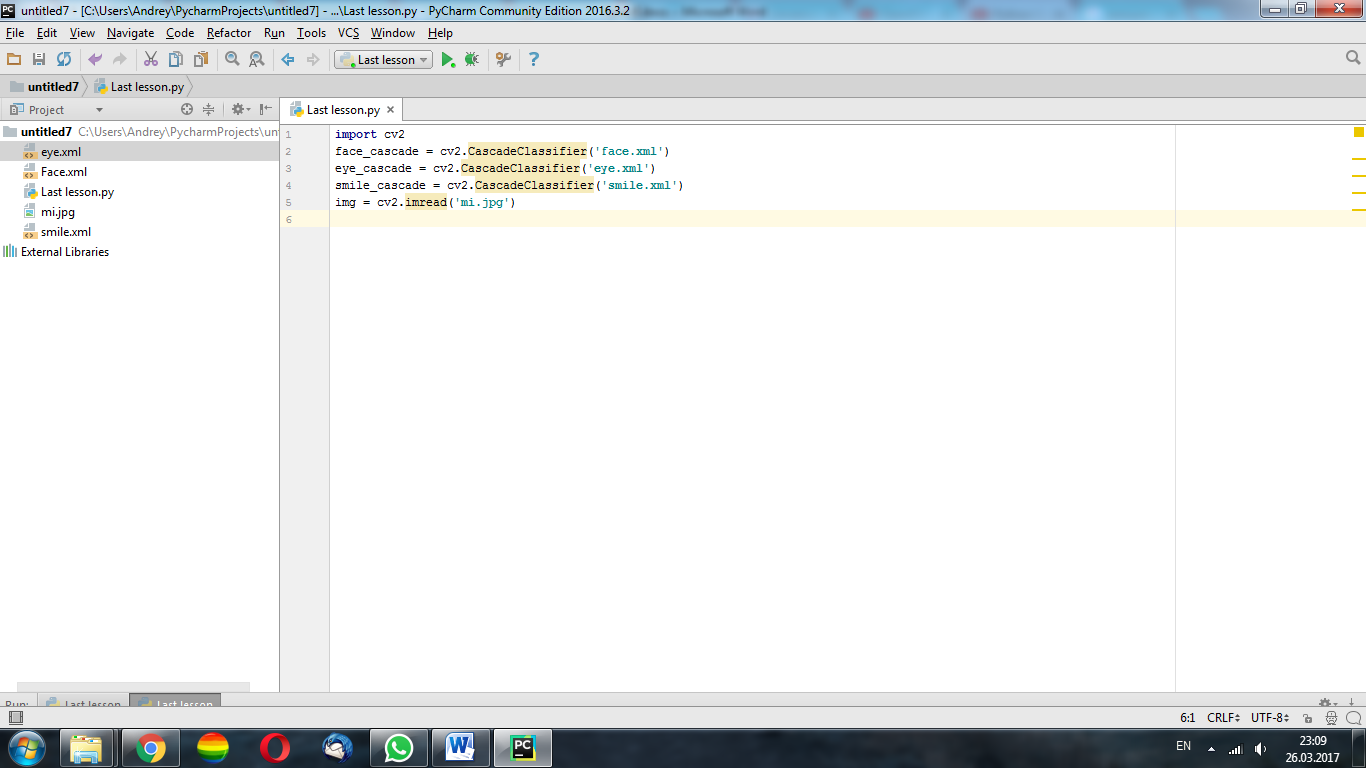


Почитать о том, что такое Хаар вы можете в файле «Признаки Хаара»



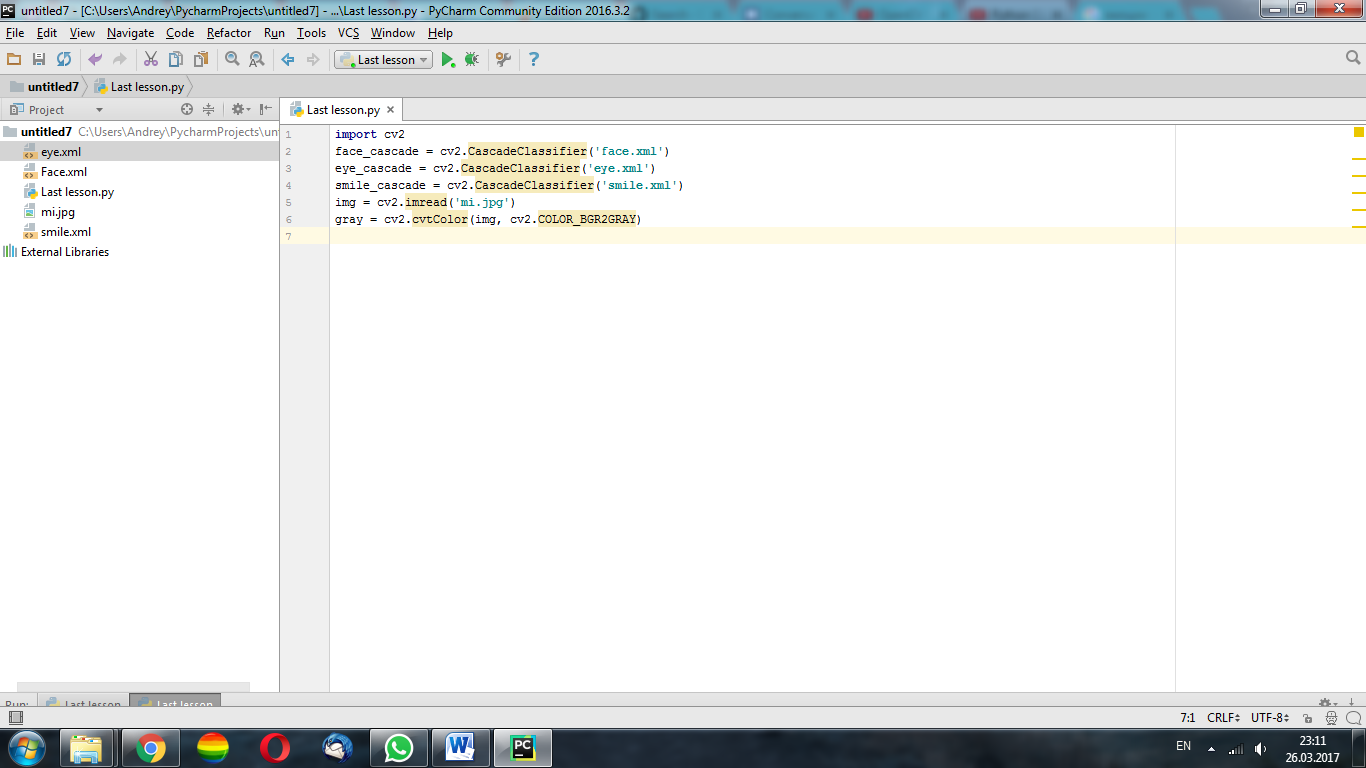
Далее читаем нашу картинку

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



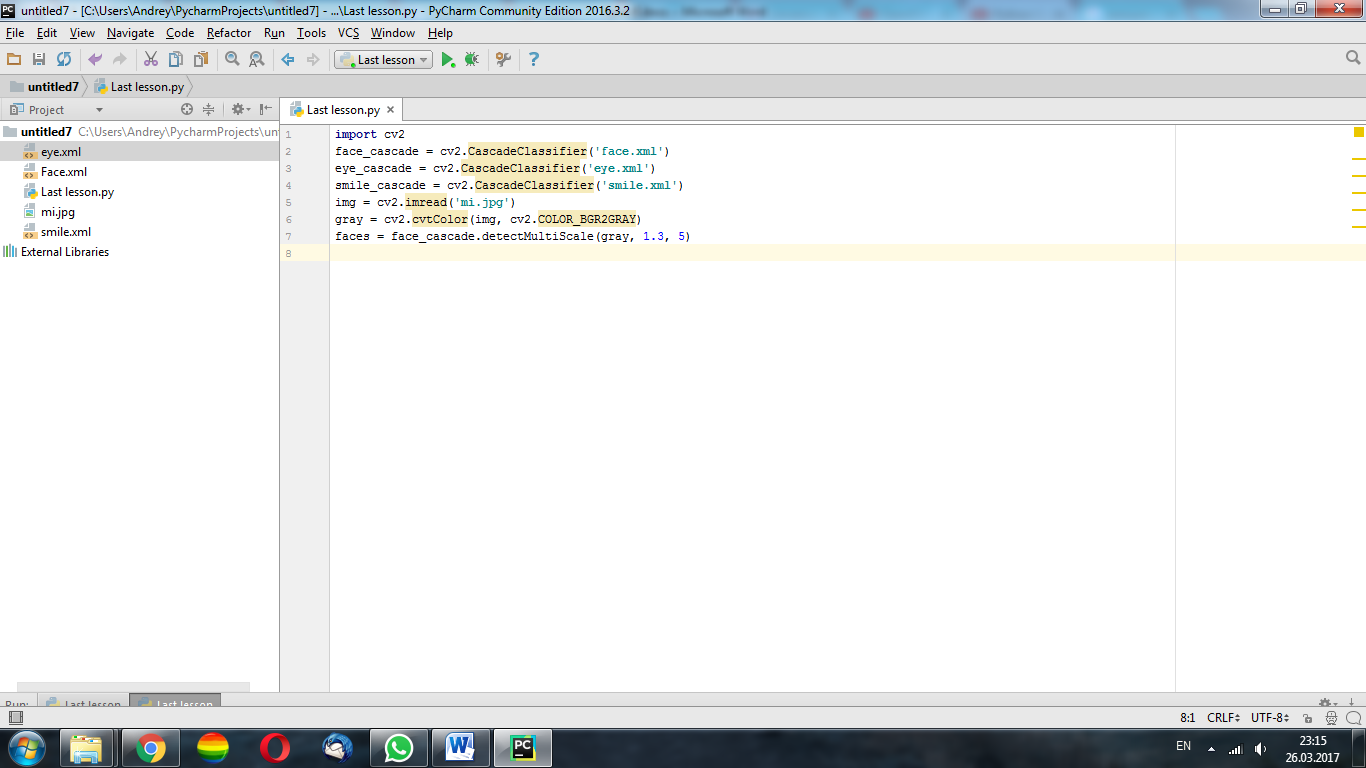
Переводим его в серые оттенки

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



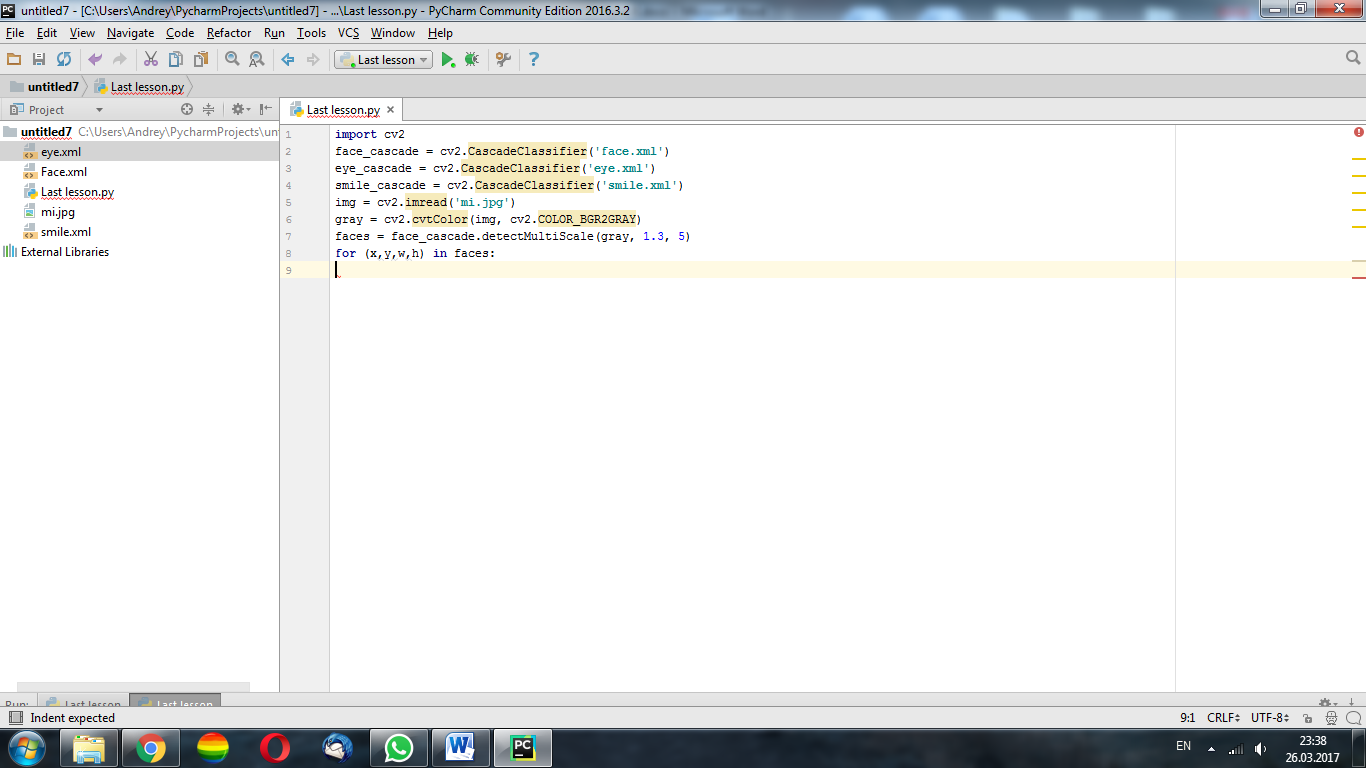
Используем метод detectMultiScale. Первым параметром «gray» мы передаем изображение в серый оттенках, далее процент масштабирования и интенсивность обнаружения лиц.

faces = face\_cascade.detectMultiScale(gray, 1.3, 5)



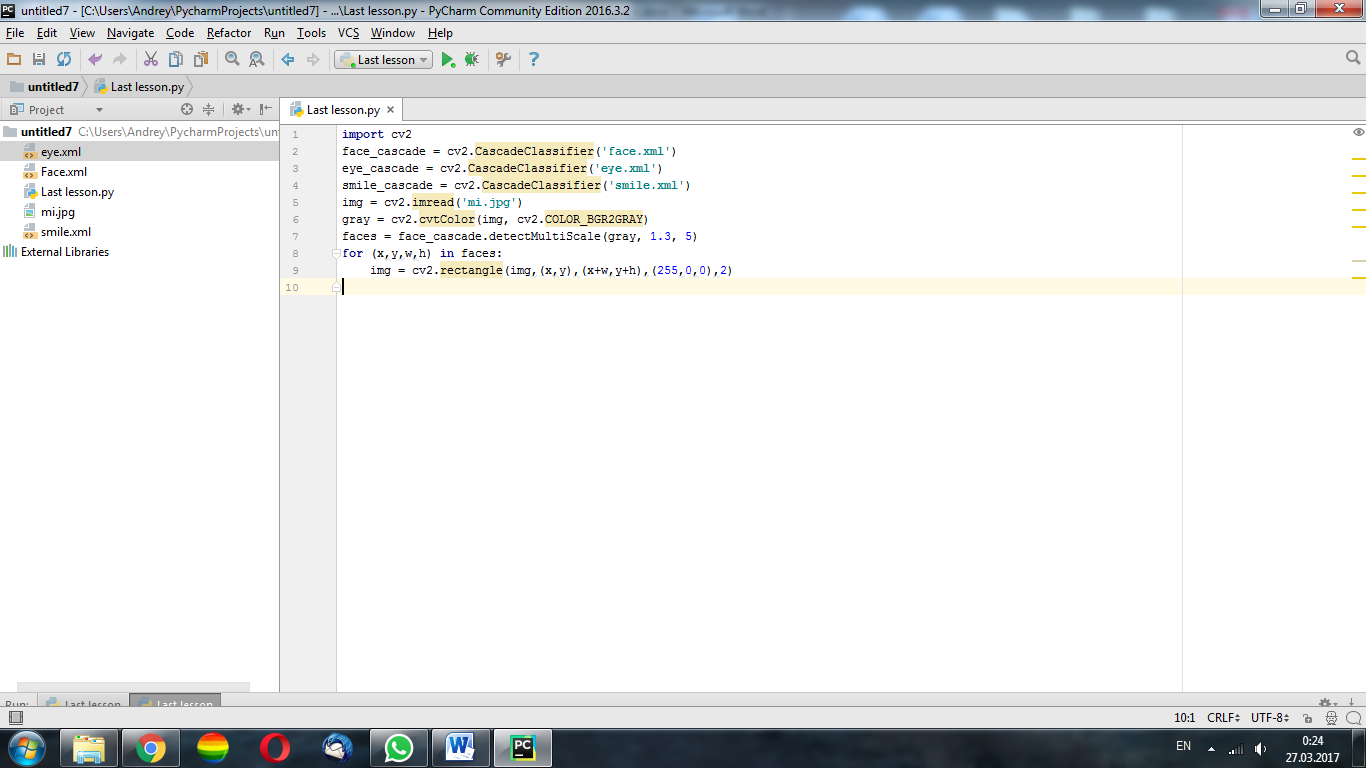
Создаем цикл с переменными для координат

**for** (x,y,w,h) **in** faces:



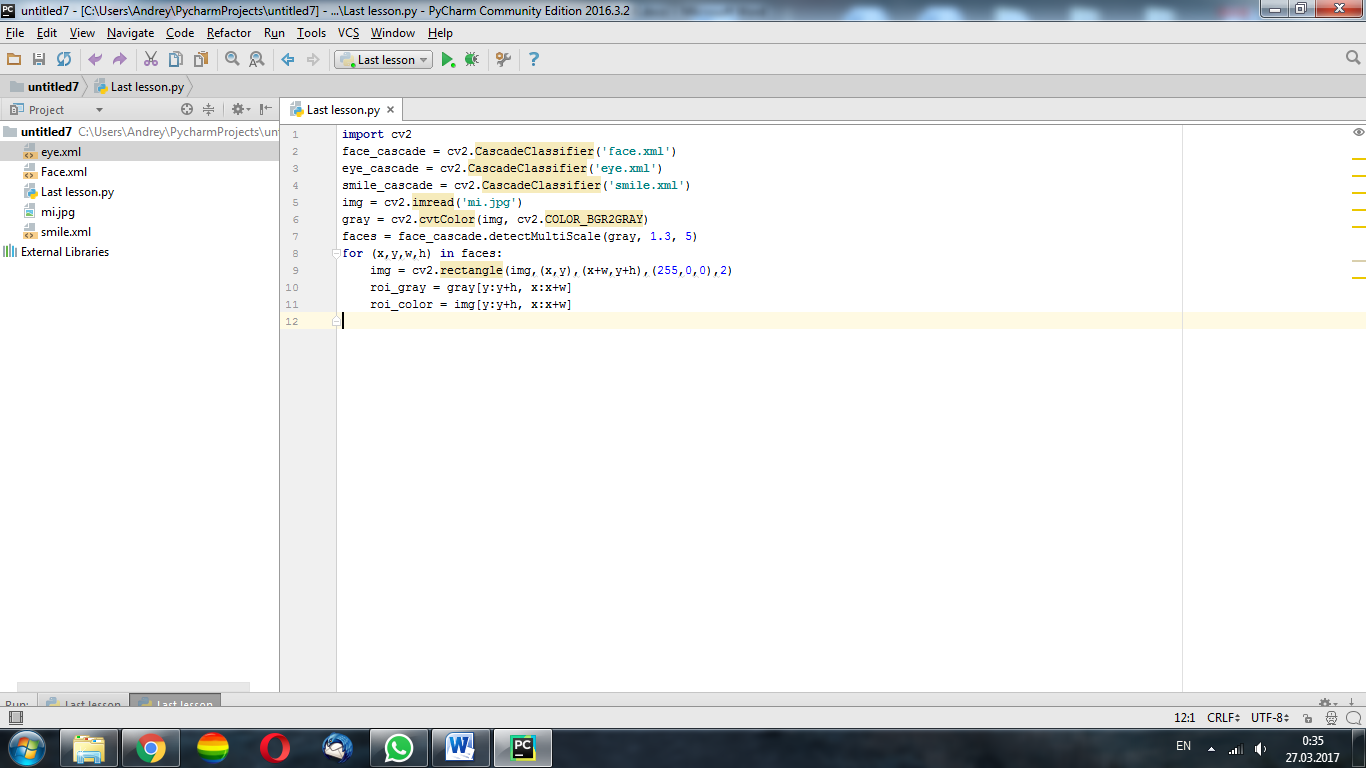
Передаем картинку и с помощью координат выделяем лица. 255.0.0 это синий цвет, далее идет толщина

img = cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)



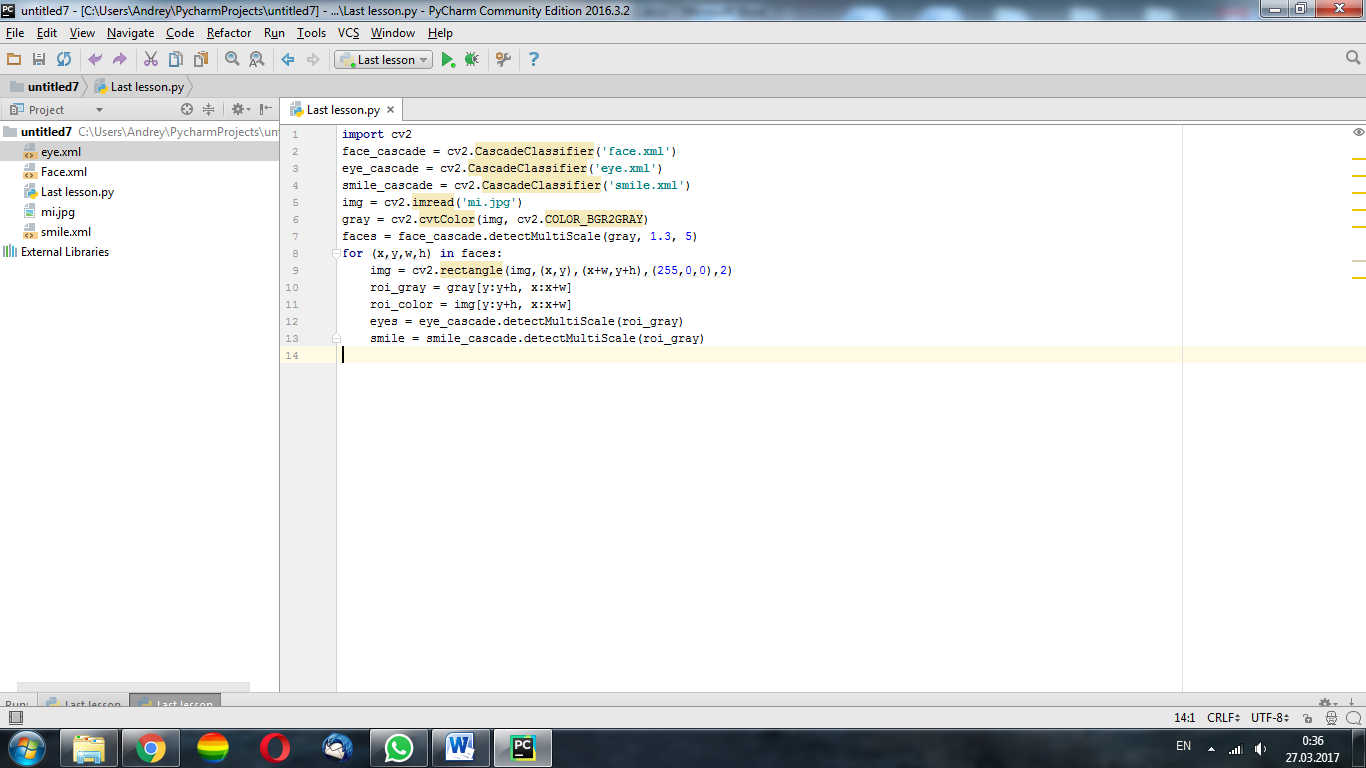
Еще нам нужны переменные, которые будут передавать координаты для улыбки и глаз в оттенках серого и такие же координаты для картинки, которую мы будем выводить

roi\_gray = gray[y:y+h, x:x+w]  
roi\_color = img[y:y+h, x:x+w]



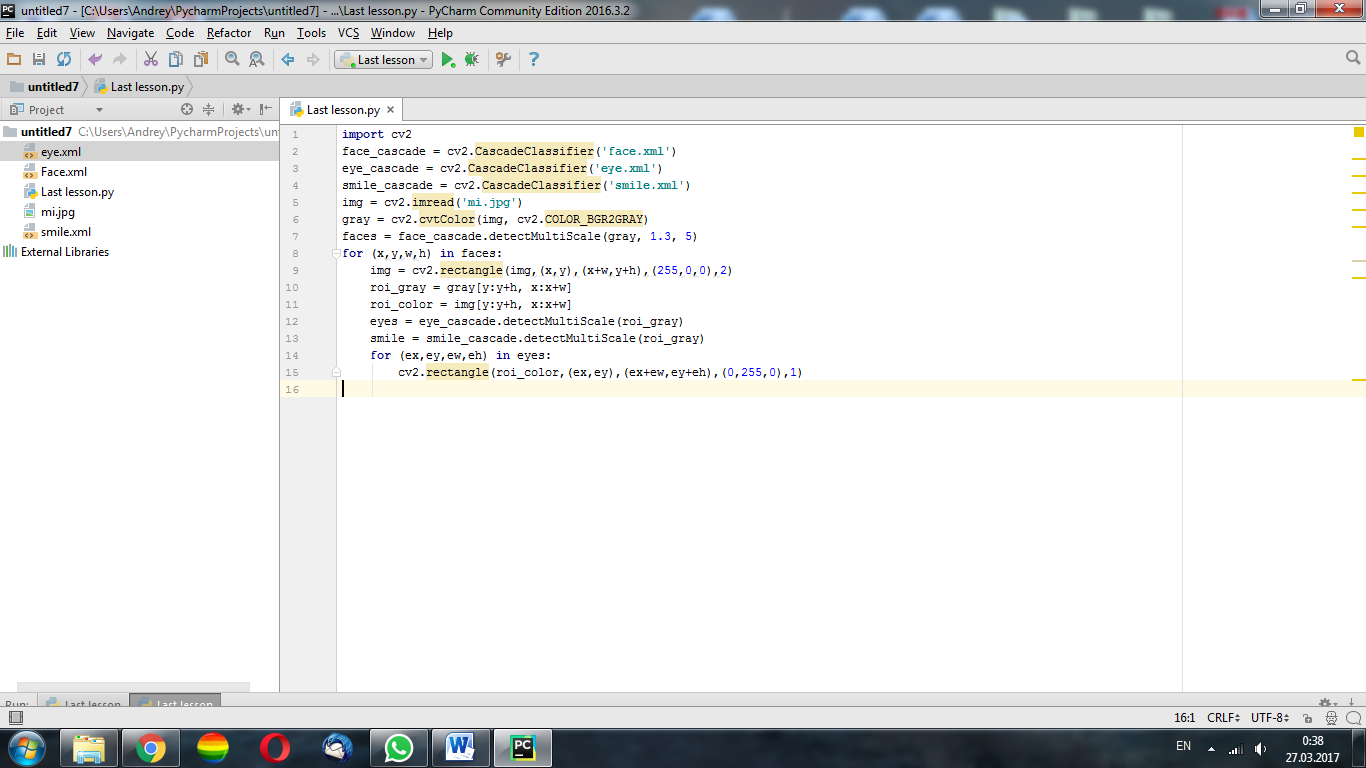
Далее используем метод detectMulriScale с заданными параметрами для переменных eyes и smile

eyes = eye\_cascade.detectMultiScale(roi\_gray)  
smile = smile\_cascade.detectMultiScale(roi\_gray)



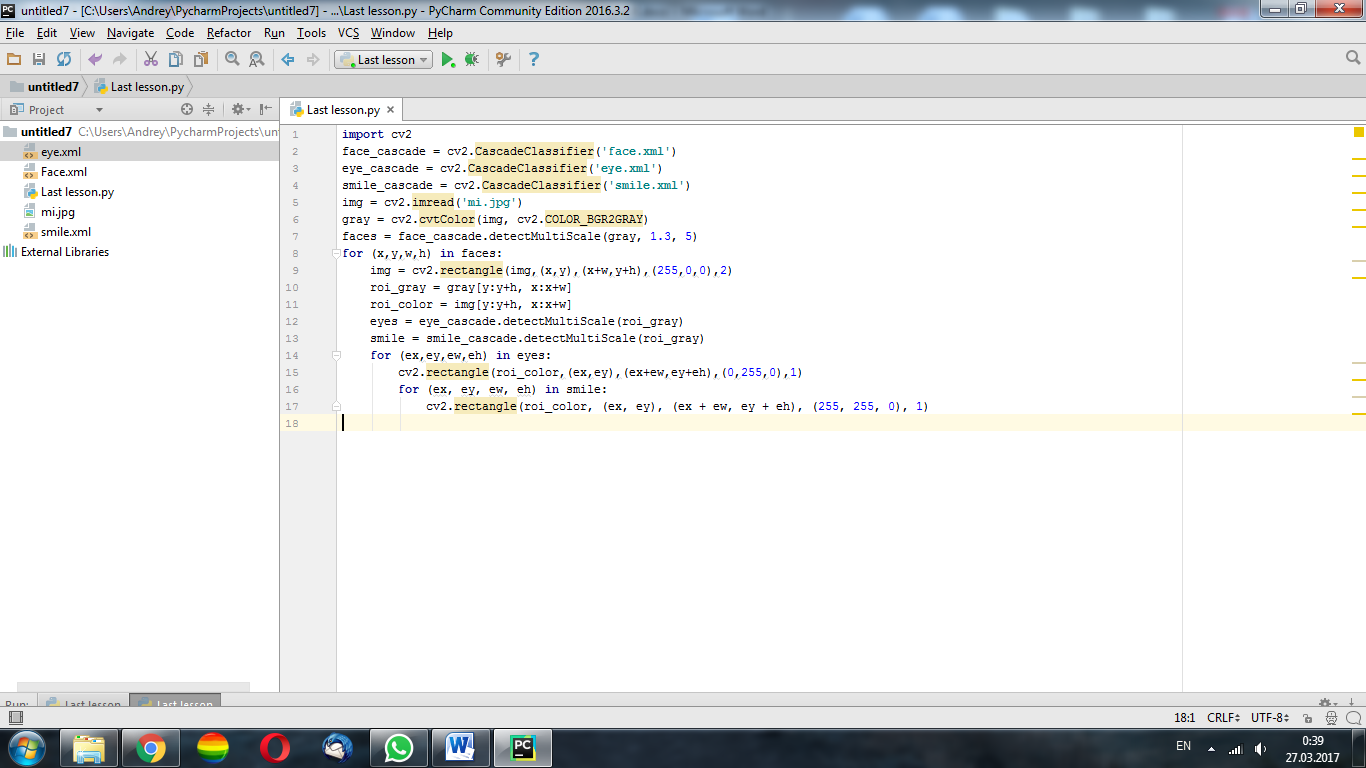
Через цикл выделяем в квадрат области вокруг глаз, зеленым цветом.

**for** (ex,ey,ew,eh) **in** eyes:  
 cv2.rectangle(roi\_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),1)



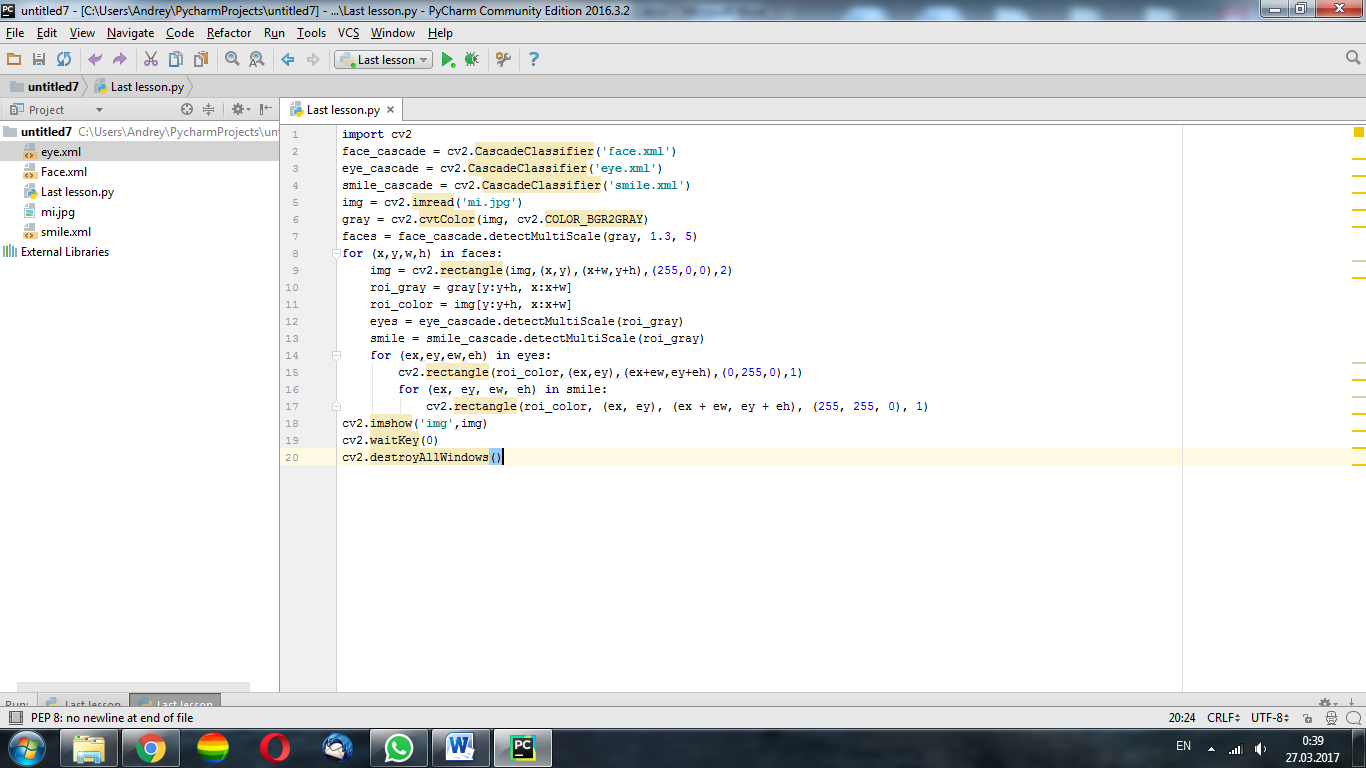
И точно так же для улыбки. На этот раз я поставил бирюзовый цвет

**for** (ex, ey, ew, eh) **in** smile:  
 cv2.rectangle(roi\_color, (ex, ey), (ex + ew, ey + eh), (255, 255, 0), 1)

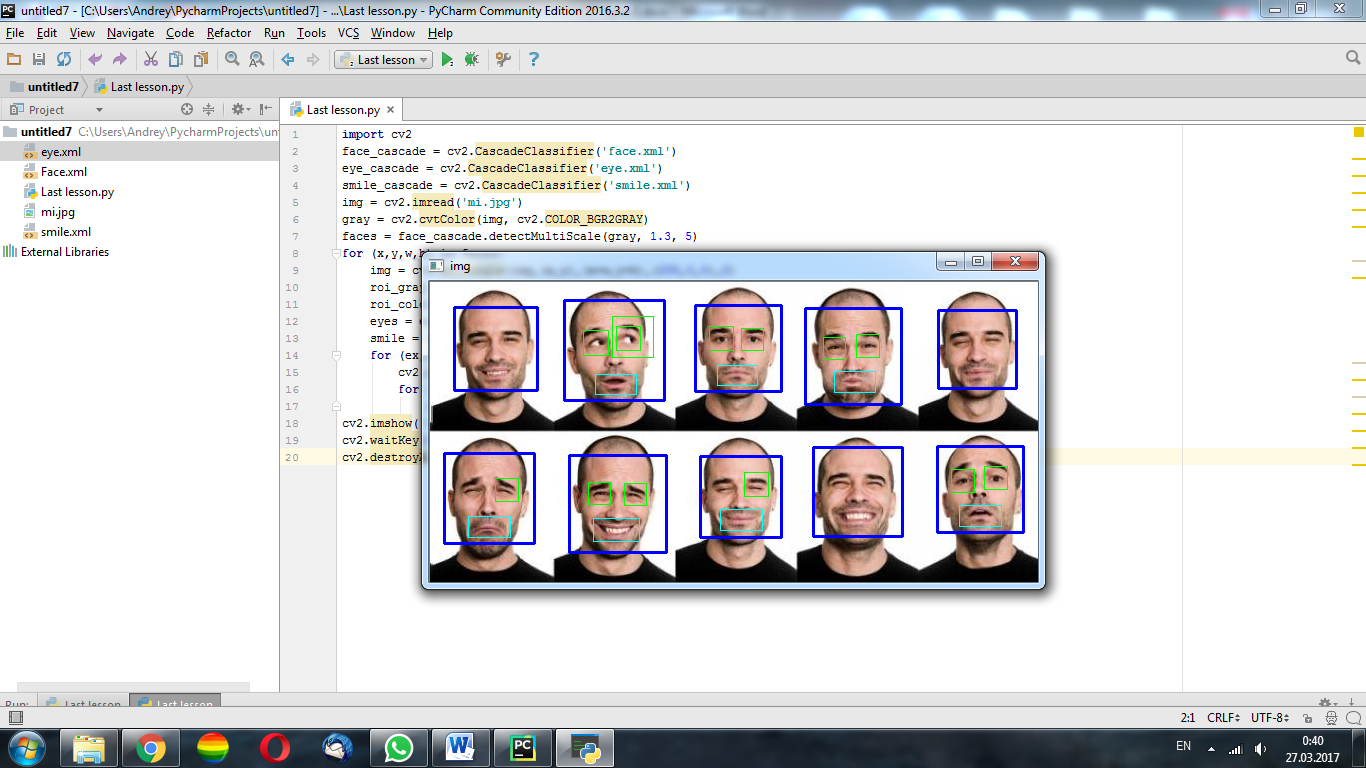


Выводим окно с картинкой и ставим ее закрытие по нажатии любой клавиши на клавиатуре

cv2.imshow(**'img'**,img)  
cv2.waitKey(0)  
cv2.destroyAllWindows()



Запускаем программу! Результат неидеальный, но это работает!



Задания для самостоятельной работы:

Основываясь на полученных с предыдущих уроков знаниях, попробуйте осуществить выделение только голубых глаз (изображение есть в папке «готовые примеры»).

Разберите работы, которые находятся в папке «готовые примеры», попытайтесь понять, как они работают.

Можете совмещать рисование и вывод изображения с веб камеры.