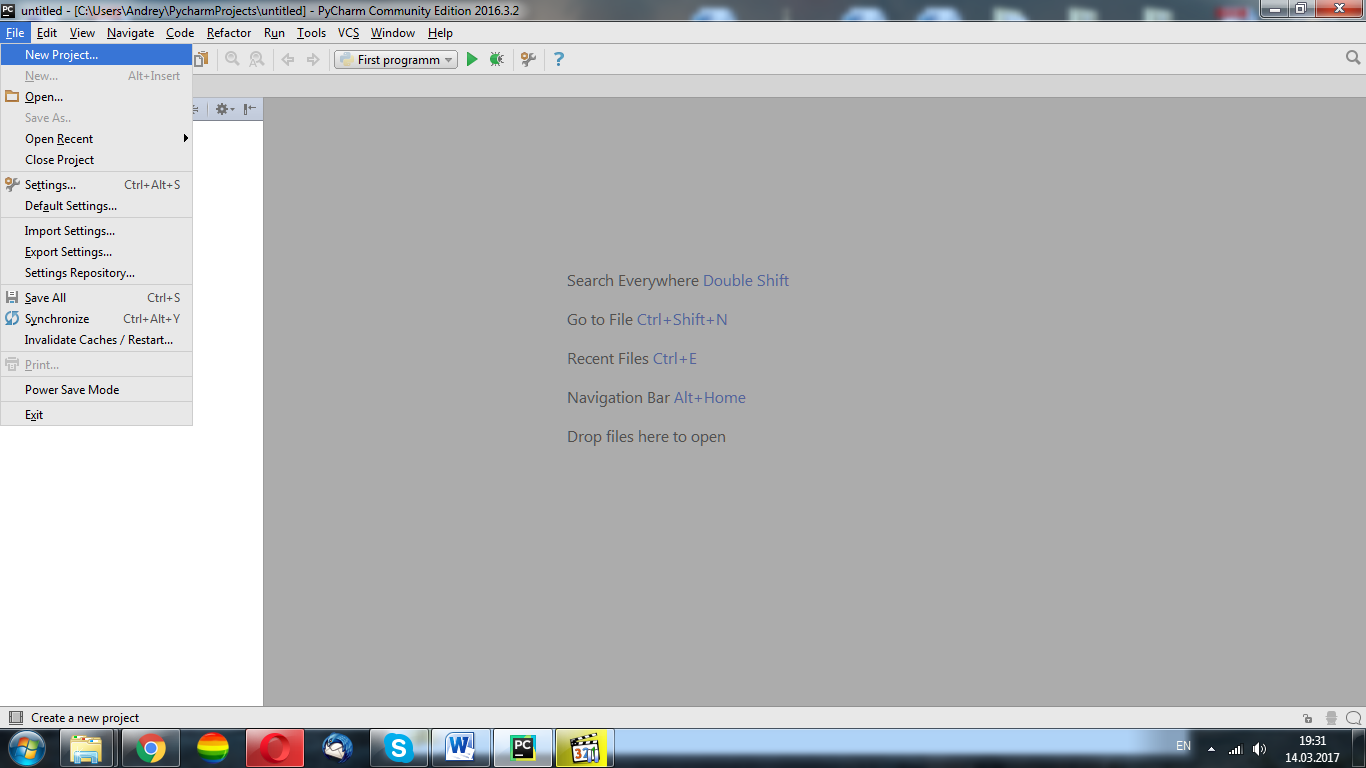
Урок № 7!

Поиск по шаблону.

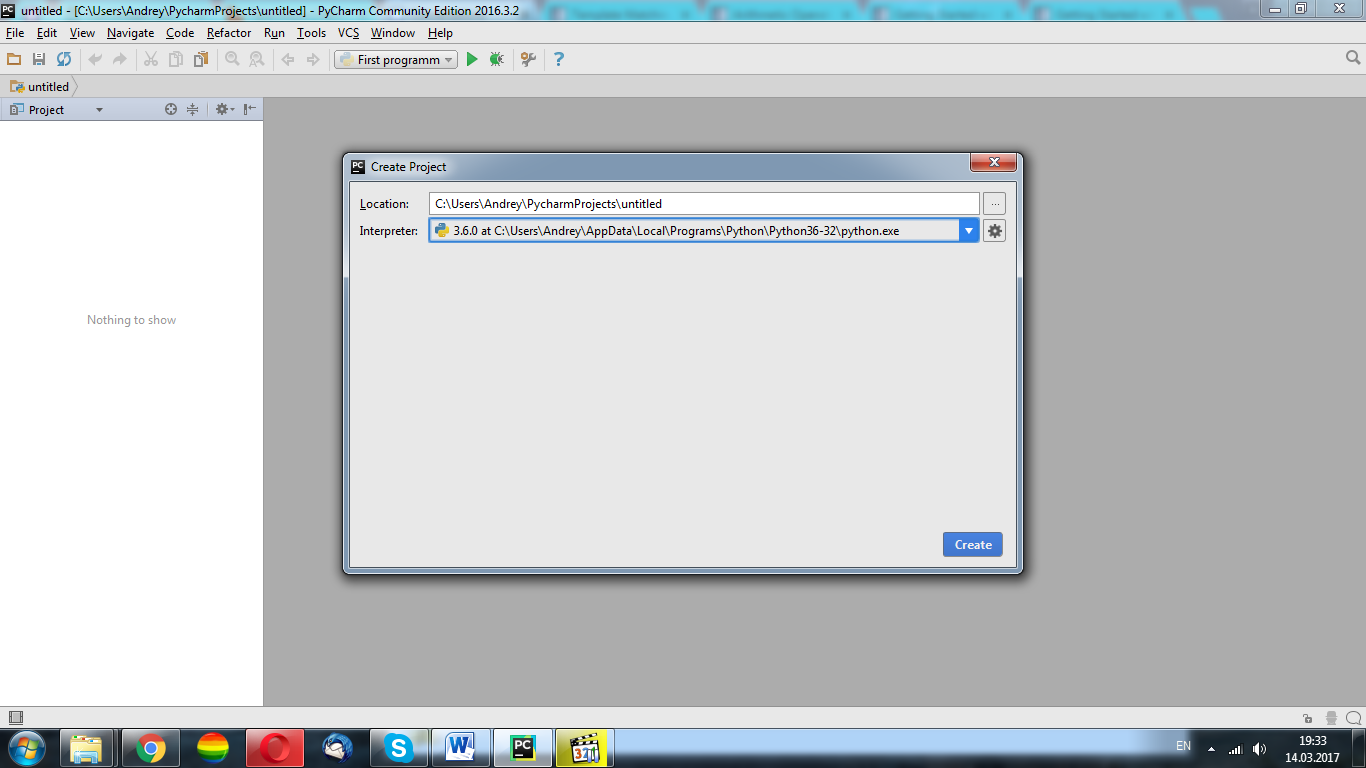
Запустите PyCharm Community Edition

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



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

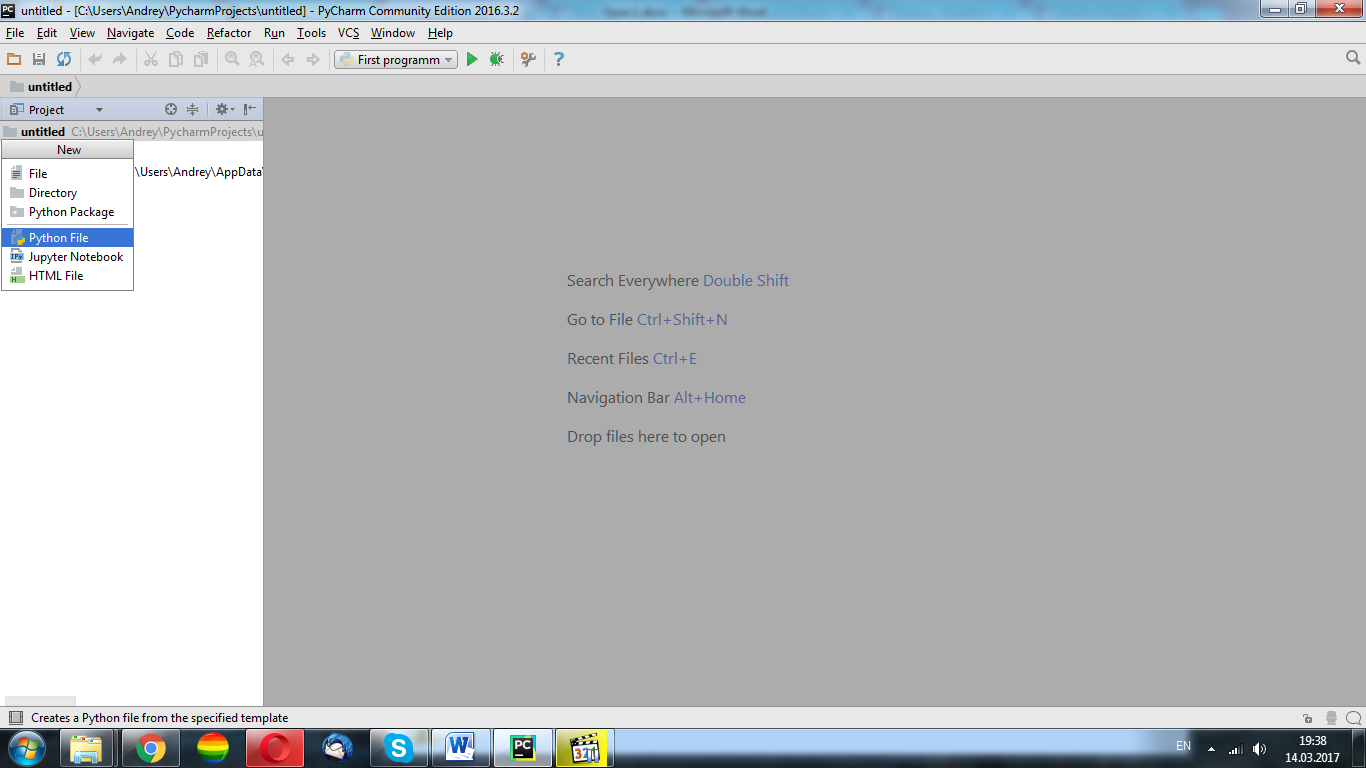
Во второй строке версия python. Выберите последнюю версию 3.6. Нажмите Create



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



Python file



Назовем этот файл Matching



Поиск по шаблону

Чтобы найти объекты в изображении с помощью сопоставления с шаблонами

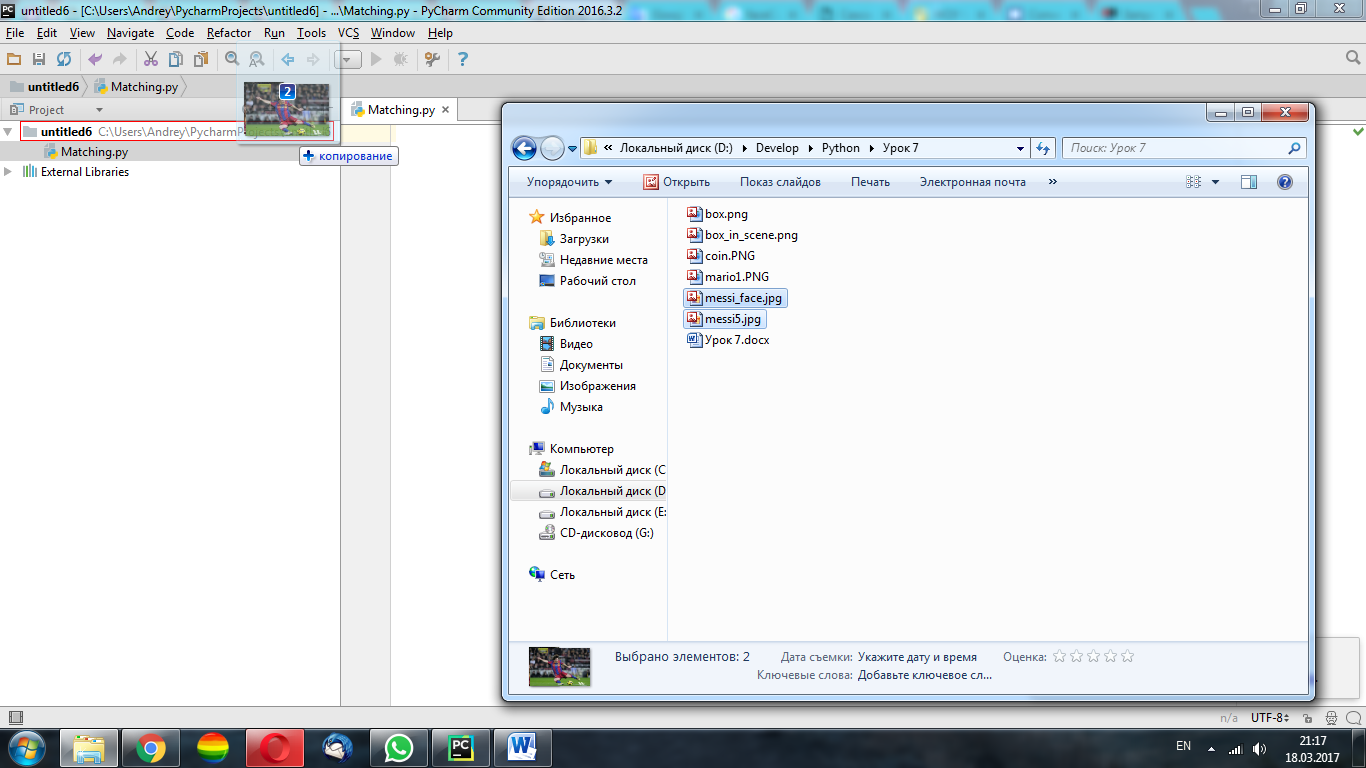
нам понадобятся следующие функции: cv2.matchTemplate (), cv2.minMaxLoc ()

Теория

Подбор шаблонов - это метод поиска и нахождения местоположения изображения шаблона на увеличенном изображении. OpenCV поставляется с функцией cv2.matchTemplate () для этой цели. Он просто сдвигает изображение шаблона по входному изображению (как в 2D свертке) и сравнивает шаблон и патч входного изображения под изображением шаблона. Он возвращает изображение в оттенках серого, где каждый пиксель обозначает, насколько окрестности этого пикселя совпадают с шаблоном.

Если входное изображение имеет размер (WxH), а шаблонное изображение имеет размер (wxh), выходное изображение будет иметь размер (W-w + 1, H-h + 1). Как только мы получим результат, то сможем использовать функцию cv2.minMaxLoc (), чтобы определить, где находится максимальное / минимальное значение. Можно взять его как верхний левый угол прямоугольника и взять (w, h) как ширину и высоту прямоугольника. Этот прямоугольник является нашей областью шаблона.

Загрузим два изображения messi\_face и messi5 для первого примера.

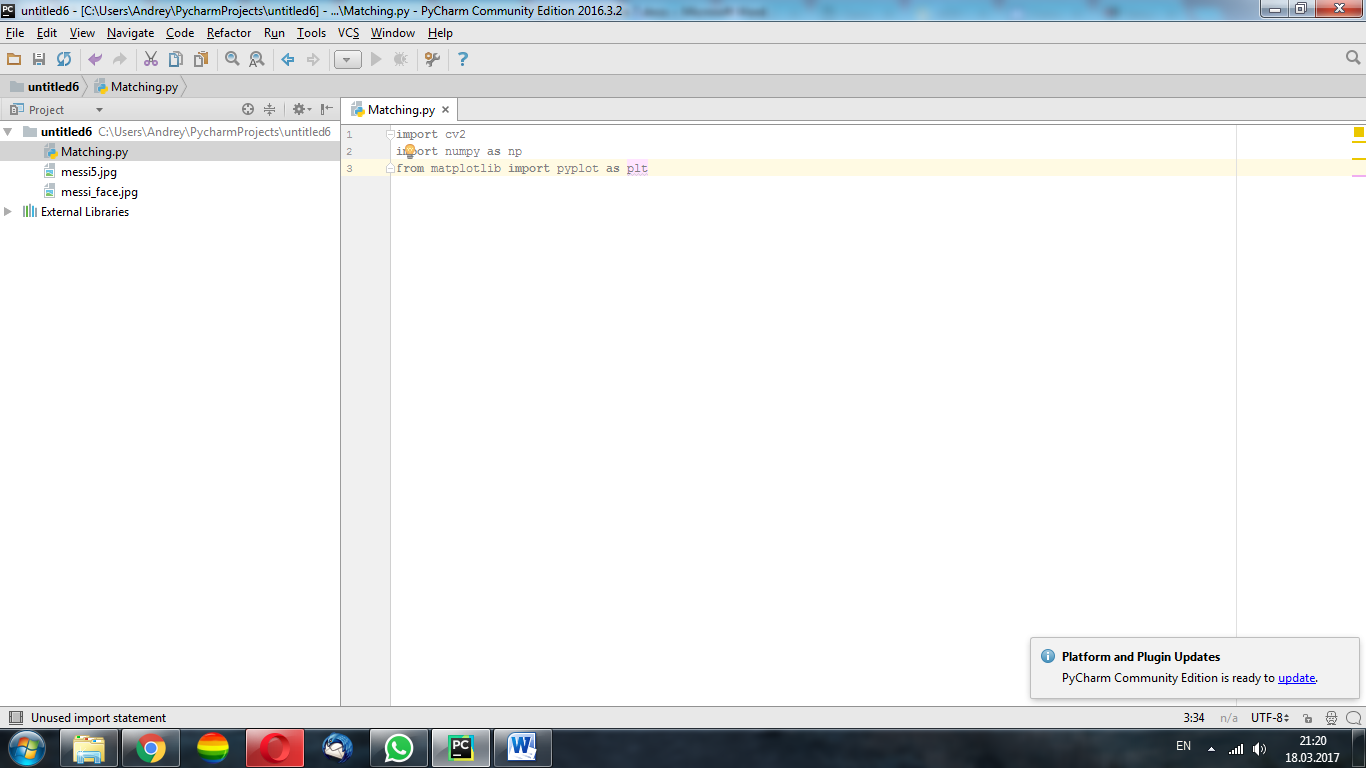


Подключим библиотеки cv2, numpy и pyplot

**import** cv2

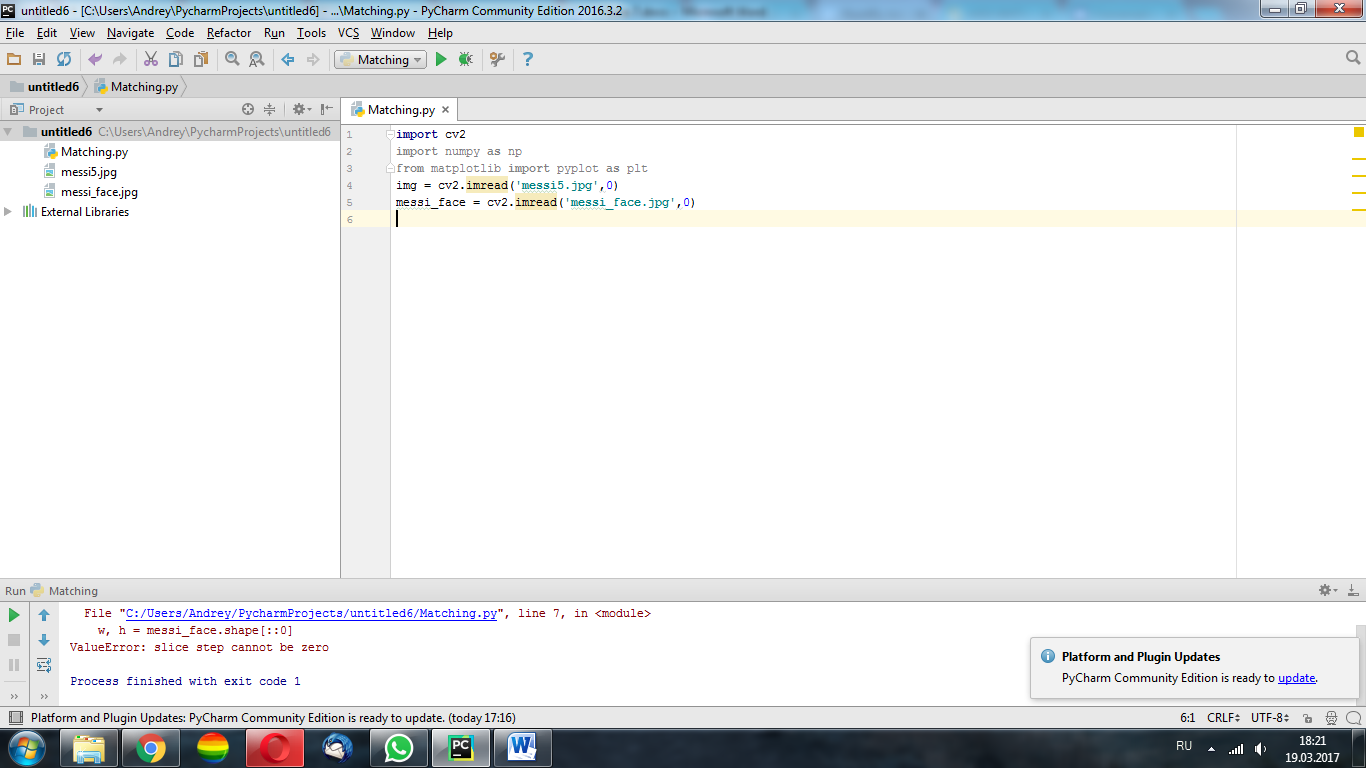
**import** numpy **as** np

**from** matplotlib **import** pyplot **as** plt



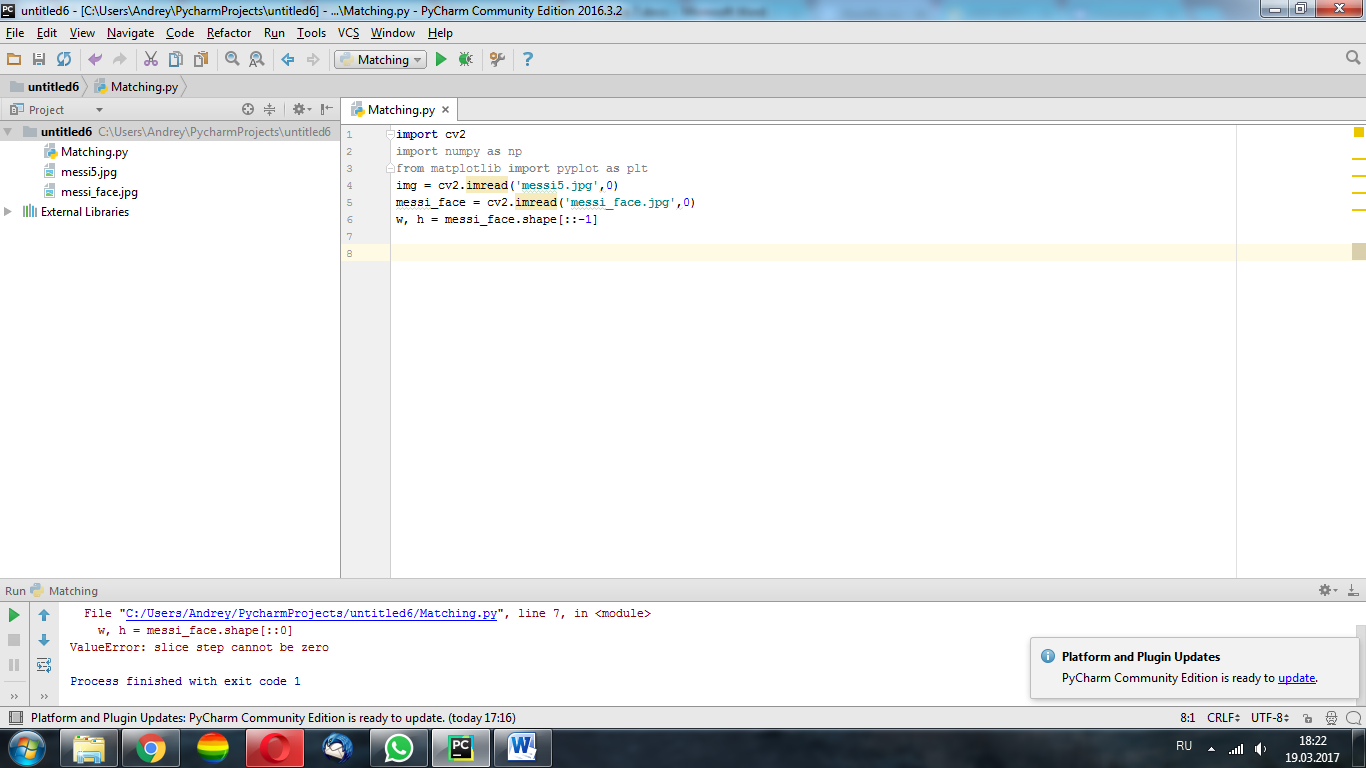
Зададим значения переменным img и messie\_face для чтения нашей картинки и шаблона (в сером цвете)

img = cv2.imread(**'messi5.jpg'**,0)  
messi\_face = cv2.imread(**'messi\_face.jpg'**,0)



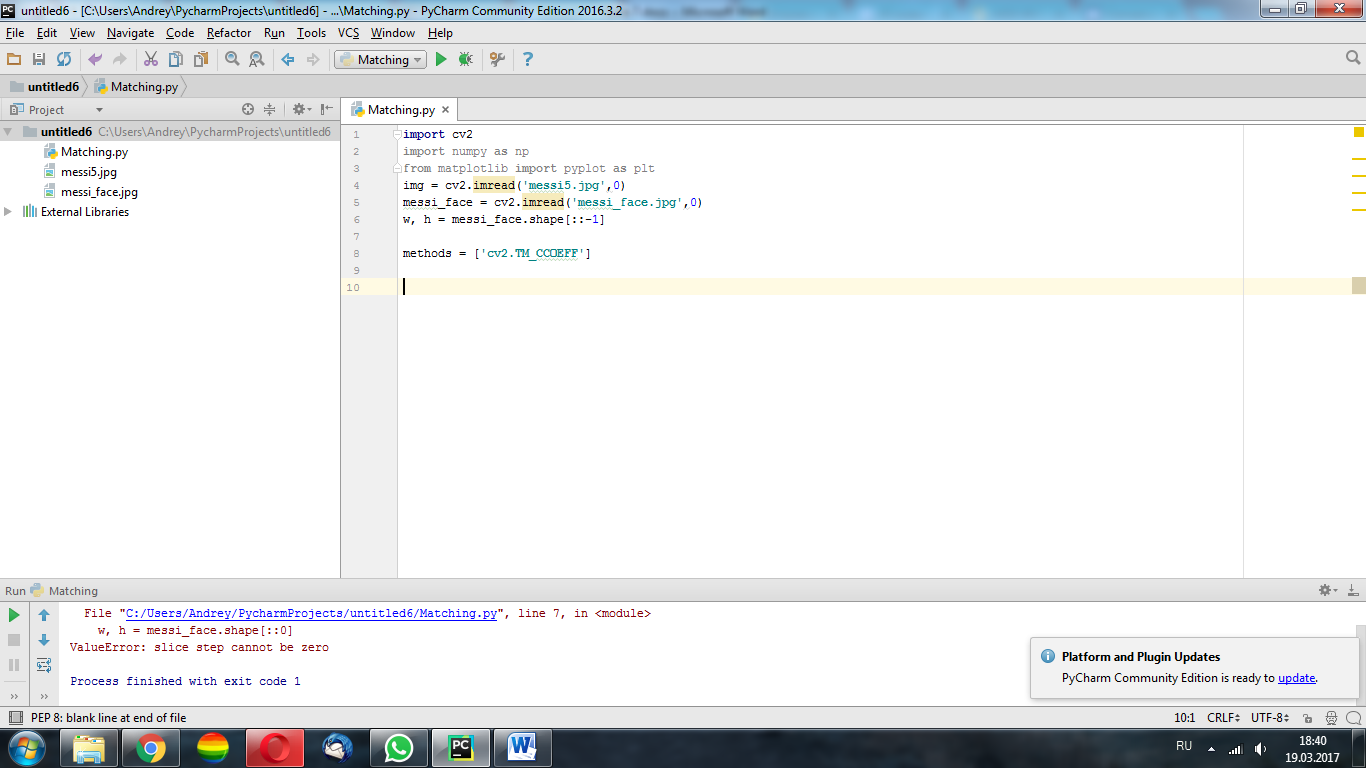
Для переменных w и h зададим массив с формой для шаблона

w, h = messi\_face.shape[::-1]



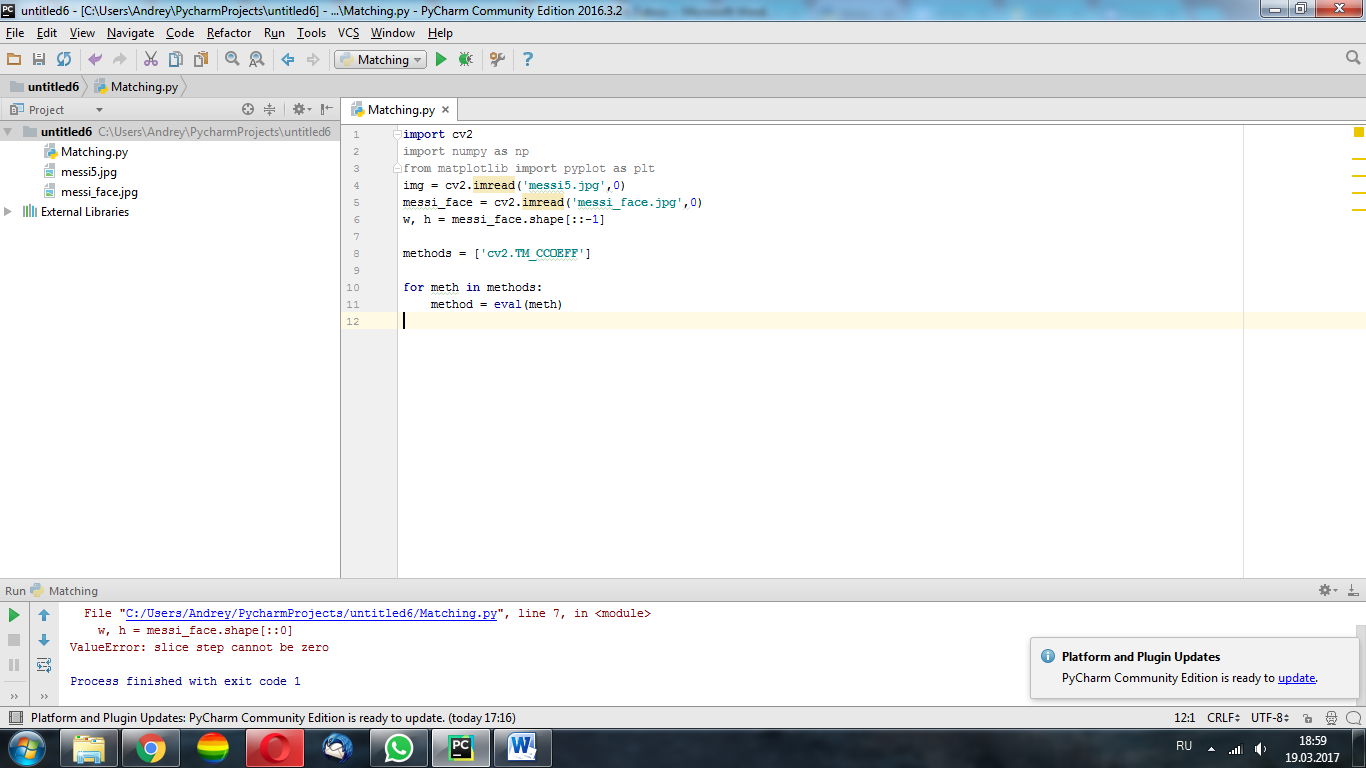
Возьмем один из методов для сравнения шаблона с картинкой.

methods = [**'cv2.TM\_CCOEFF'**]



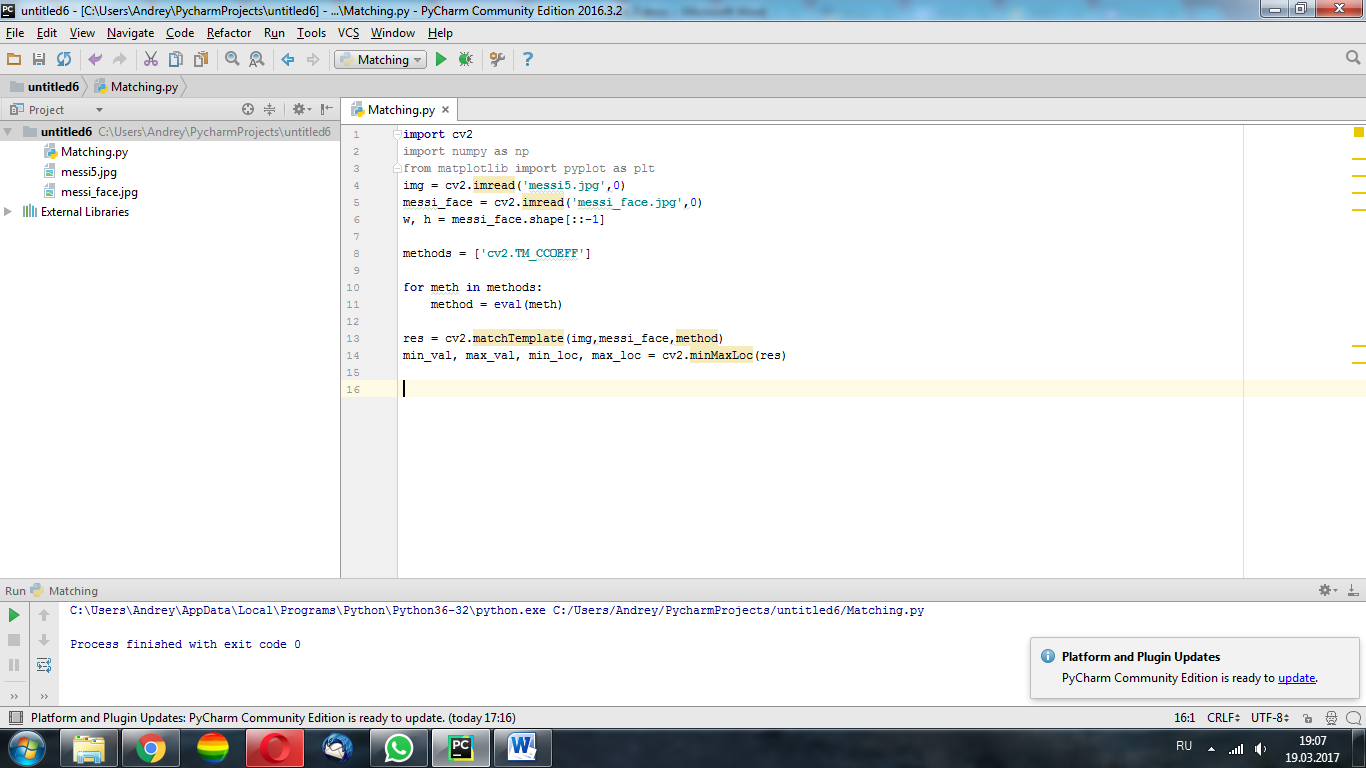
Откроем цикл для meth с функцией eval (превращающую строковые объекты в код)

**for** meth **in** methods:  
 method = eval(meth)



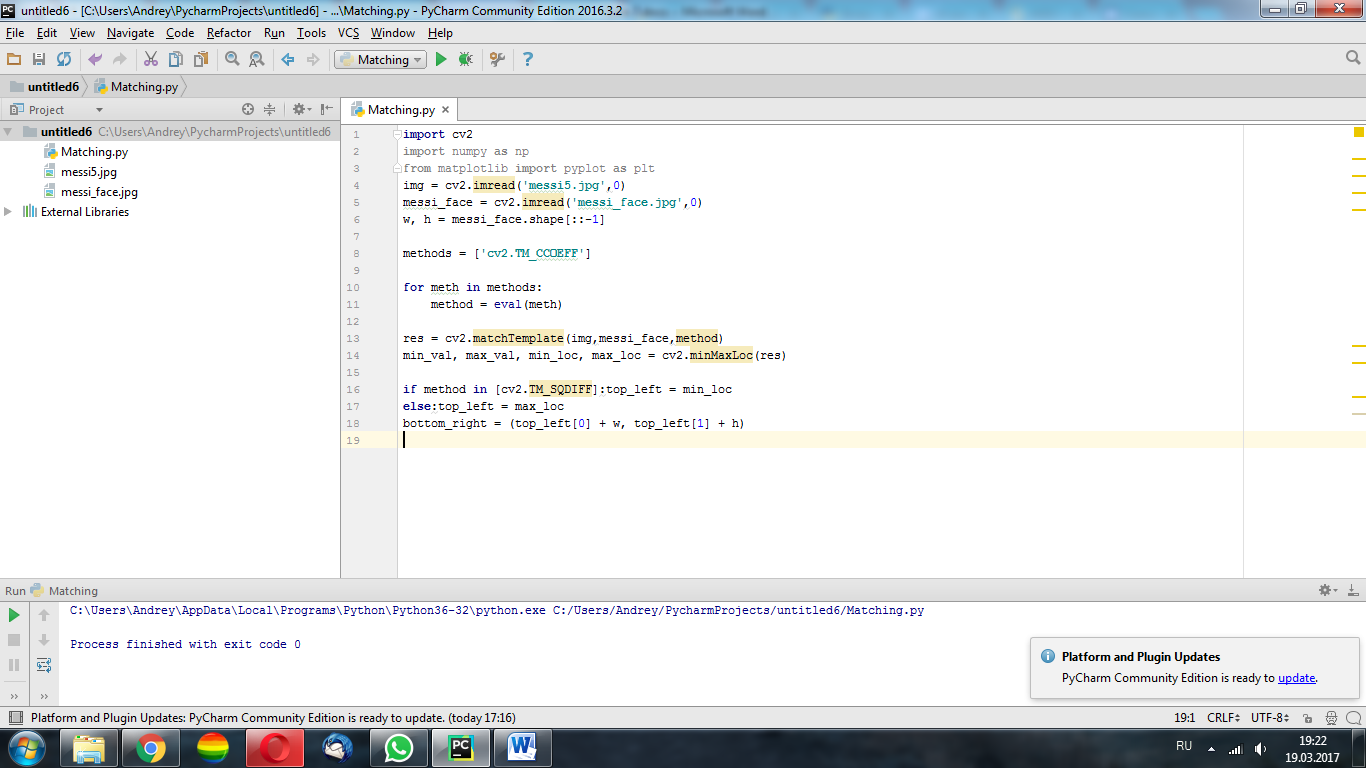
Подключаем функцию cv2.matchTampate для поиска соответствий и подключаем функцию cv2.minMaxLoc для определения, где находится максимальное / минимальное значение.

res = cv2.matchTemplate(img,messi\_face,method)  
min\_val, max\_val, min\_loc, max\_loc = cv2.minMaxLoc(res)



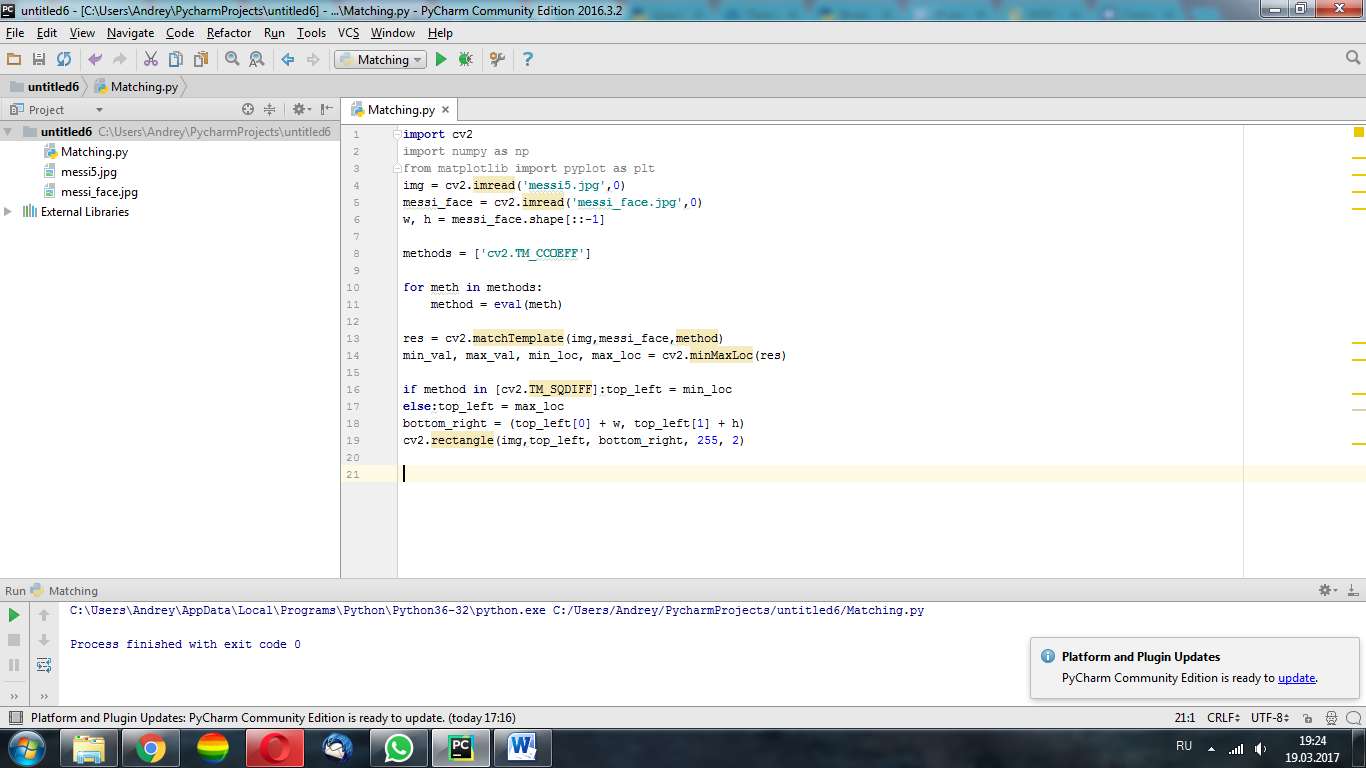
И как описывалось в теории, мы будем просто сдвигать наш шаблон и сравнивать с изображением под ним

**if** method **in** [cv2.TM\_SQDIFF]:top\_left = min\_loc  
**else**:top\_left = max\_loc  
bottom\_right = (top\_left[0] + w, top\_left[1] + h)



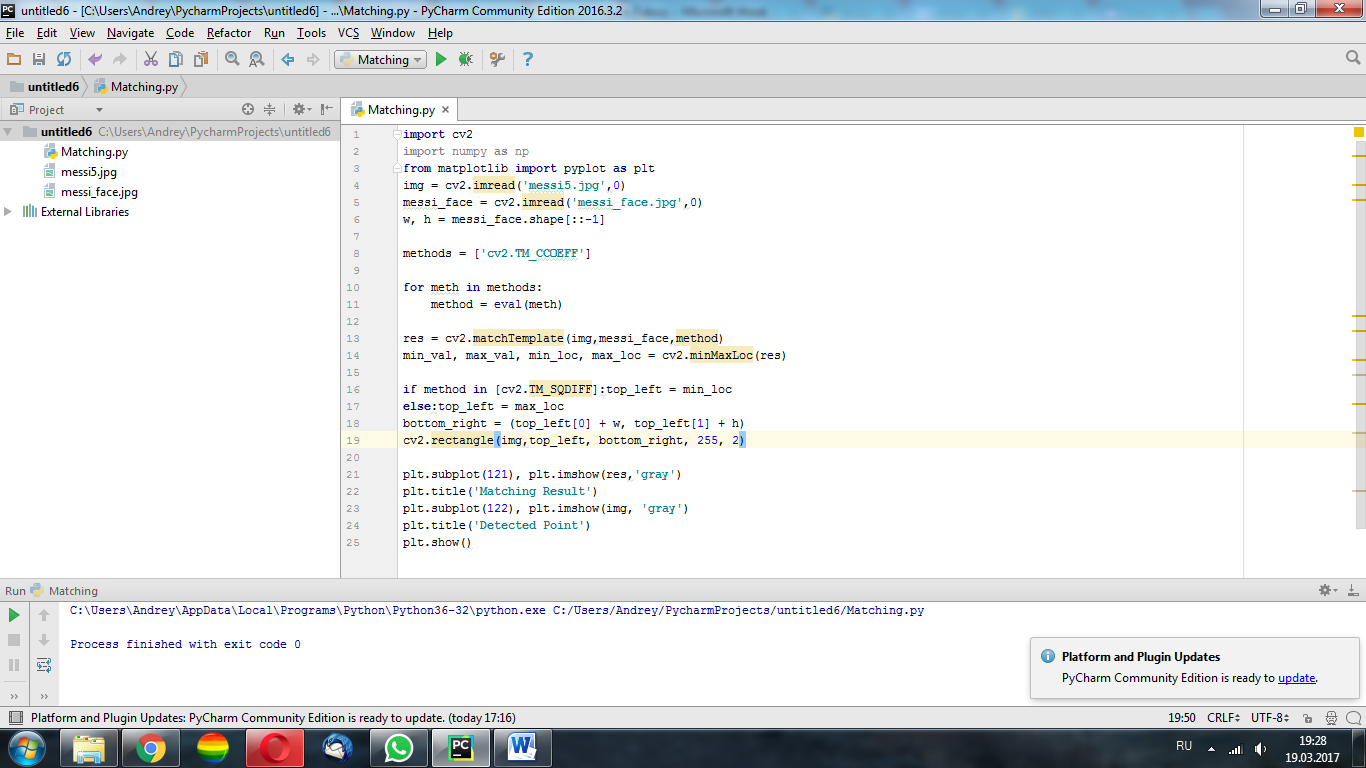
И при соответствии шаблона с изображением под ним подсветим его прямоугольником

cv2.rectangle(img,top\_left, bottom\_right, 255, 2)

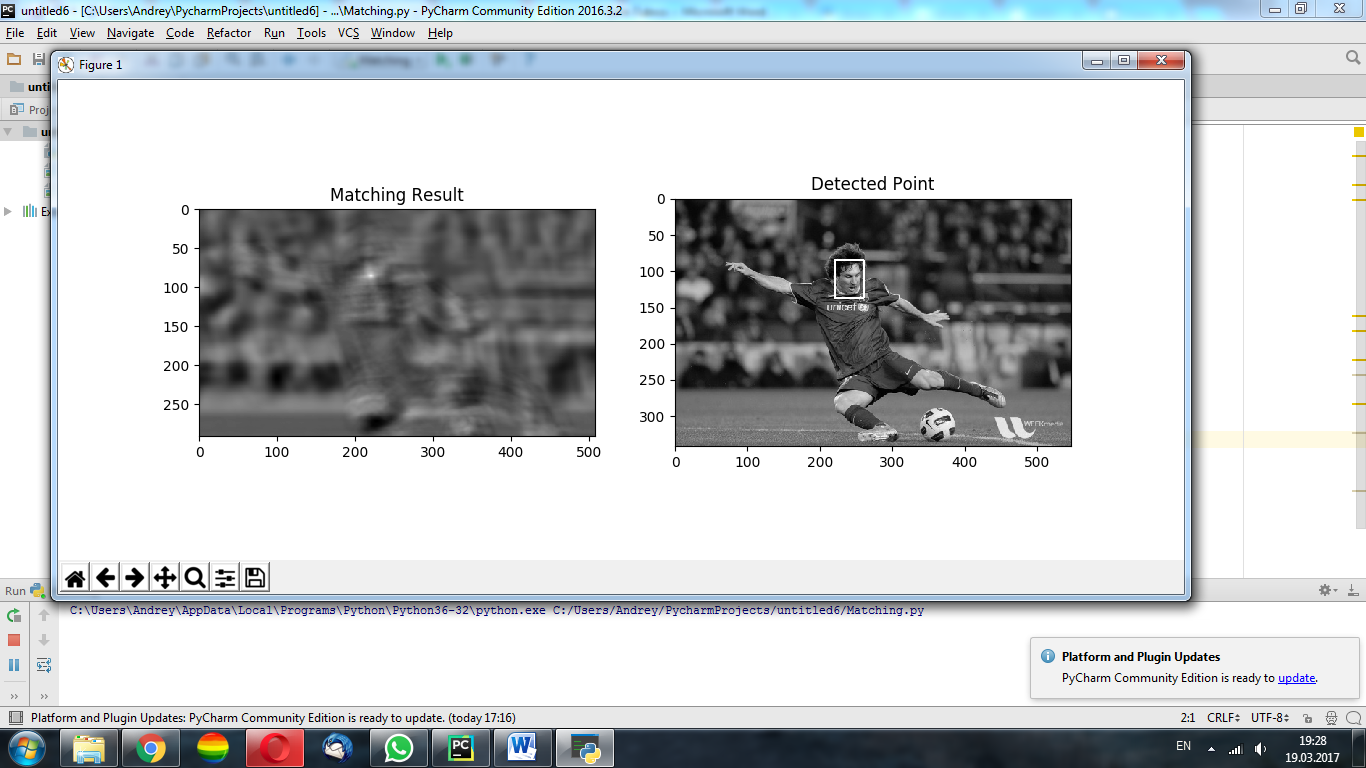


И выводим изображения с подписями, в сером цвете и само окно программы

plt.subplot(121), plt.imshow(res,**'gray'**)  
plt.title(**'Matching Result'**)  
plt.subplot(122), plt.imshow(img, **'gray'**)  
plt.title(**'Detected Point'**)  
plt.show()



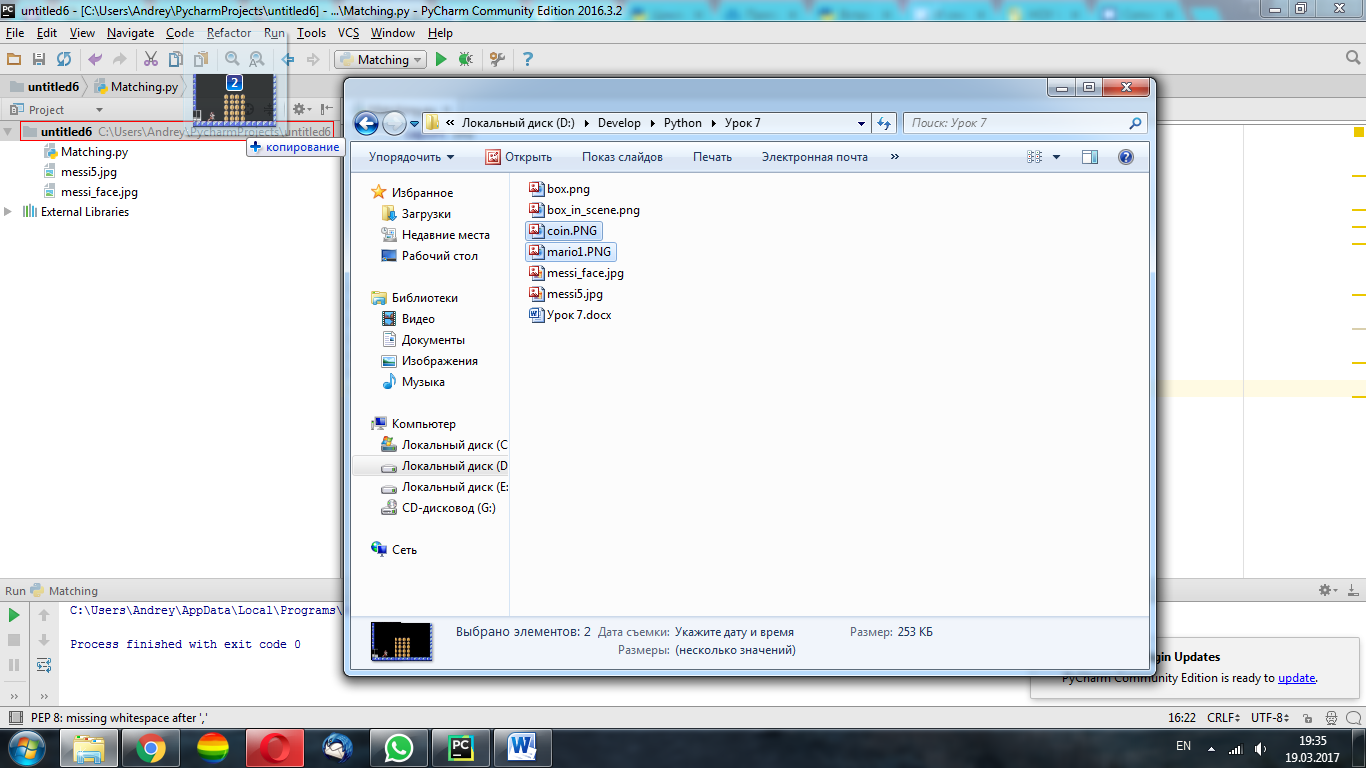
Слева отображается результат сопоставления, а справа найденная область.



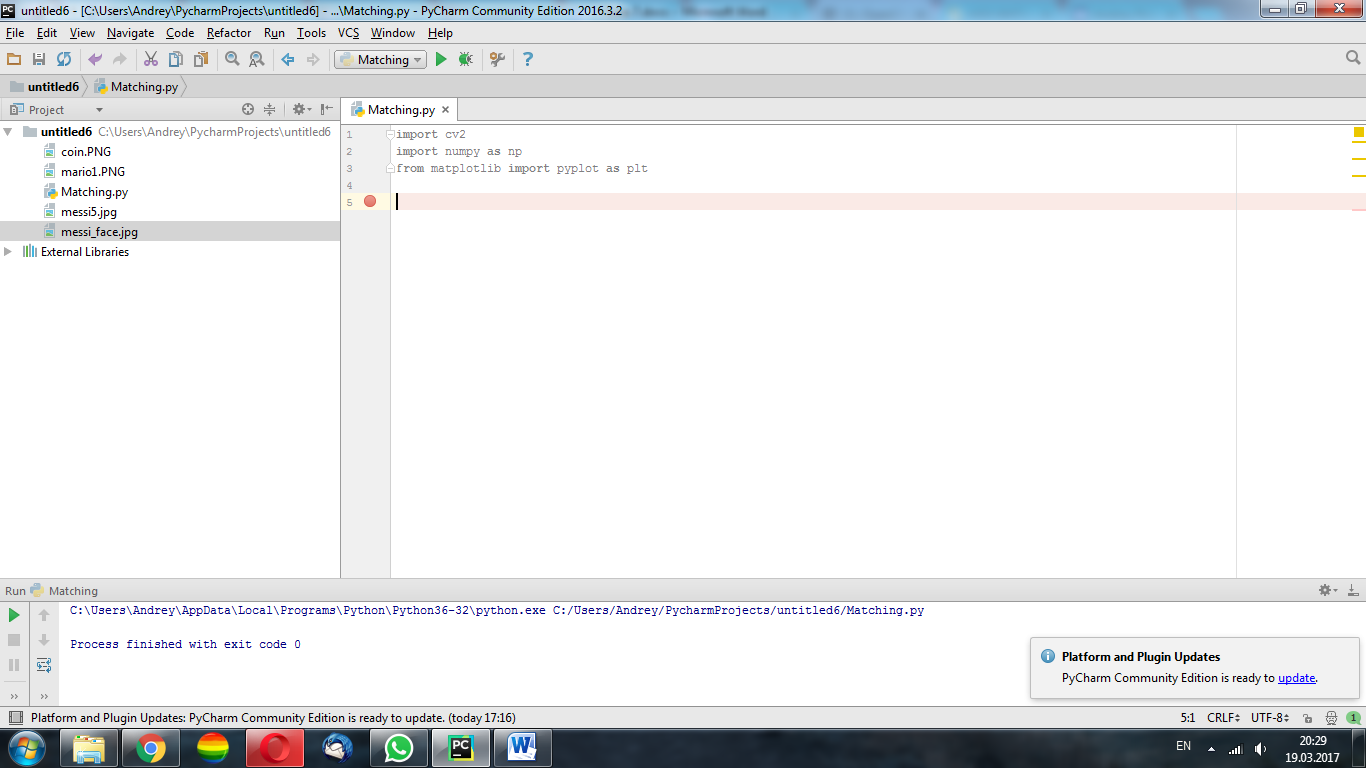
Совмещение шаблона с несколькими объектами.

До этого мы работали с шаблоном, который мог встретиться только один раз на изображении. Если у нас имеется несколько одинаковых обьектов на изображении, то функция cv2.minMaxLoc () не сможет определить их всех. В этом случае мы будем использовать пороговое значение (threshold)

Начнем с загрузки картинок. Выберите две с названиями mario1 и coin

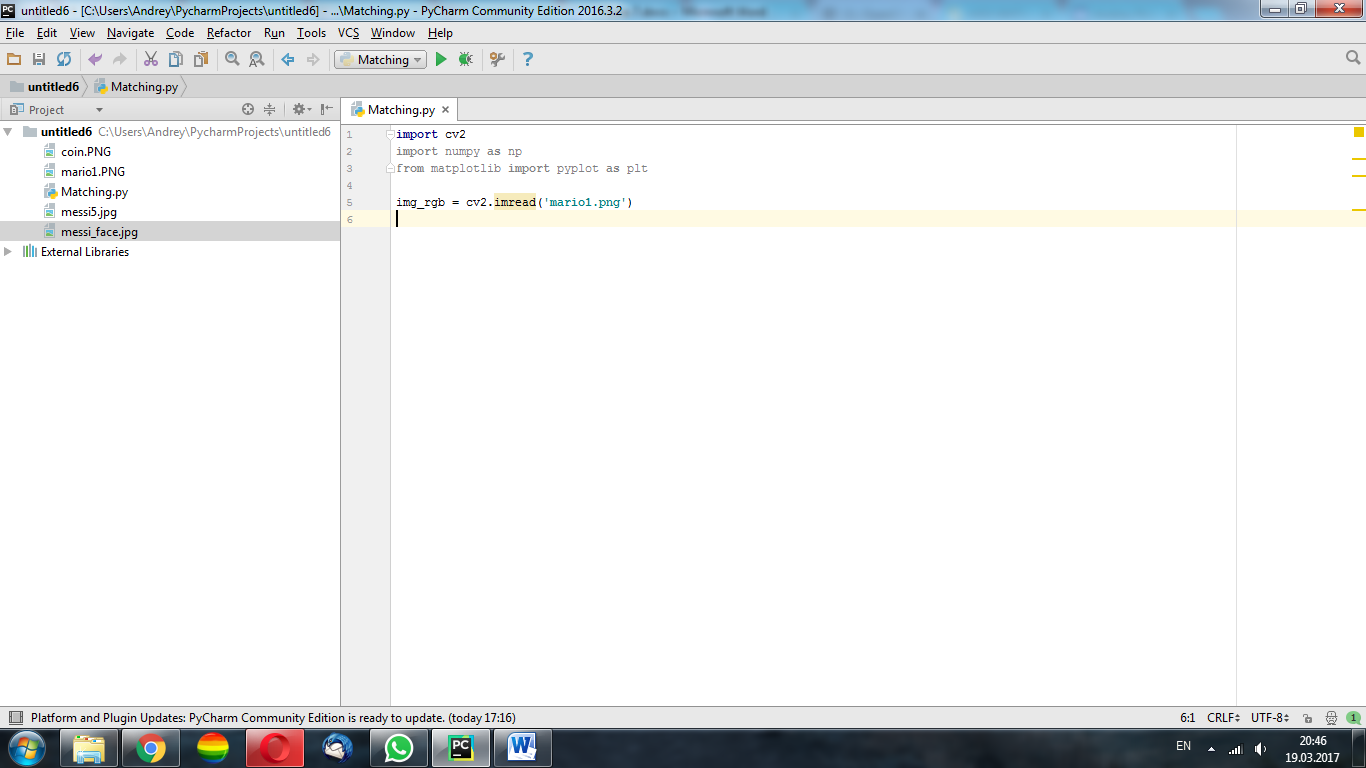


Оставьте только 3 верхние строчки кода с библиотеками



Зададим значения переменной img\_rgb для чтения картинки mario1.png.

img\_rgb = cv2.imread(**'mario1.png'**)

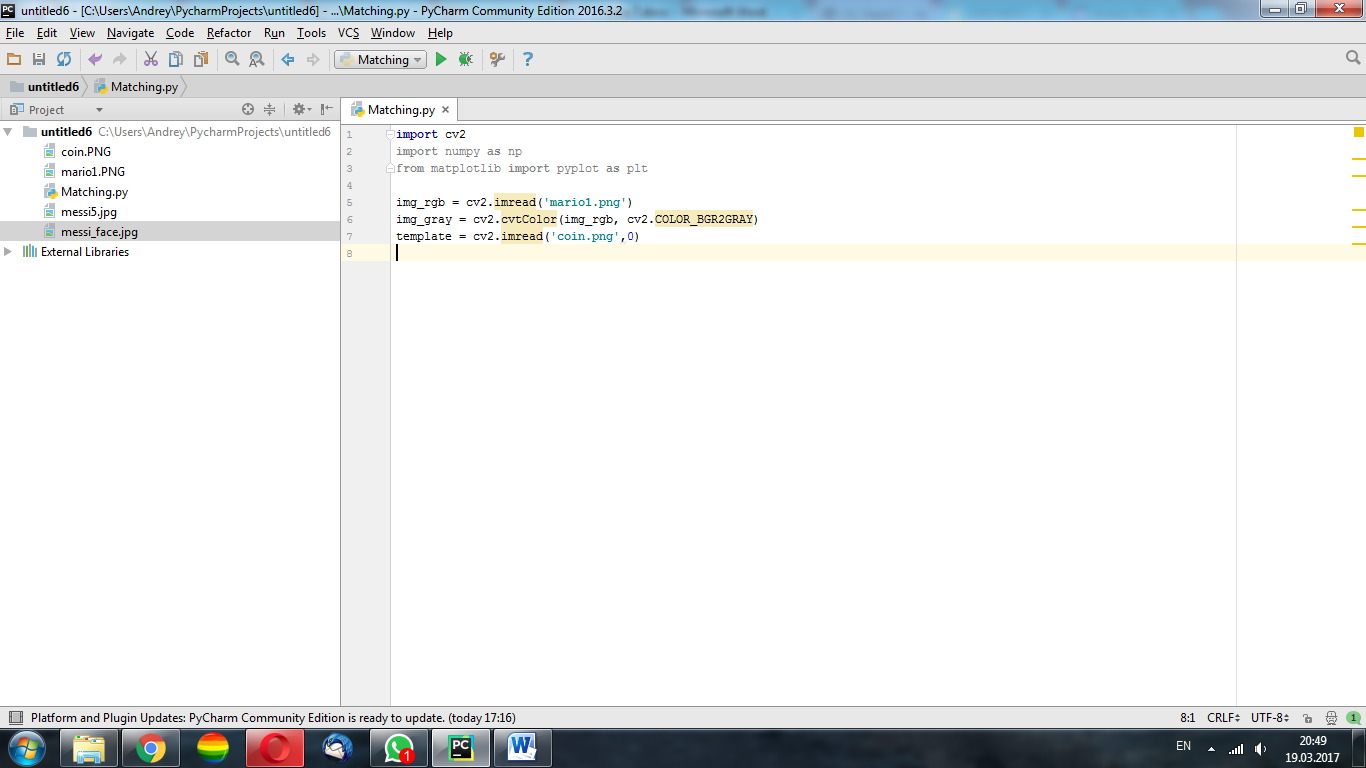


В переменную img\_gray запишем функцию cv2.cvtColor для преобразования цвета, с флагом cv2.COLOR\_BGR2GRAY

img\_gray = cv2.cvtColor(img\_rgb, cv2.COLOR\_BGR2GRAY)

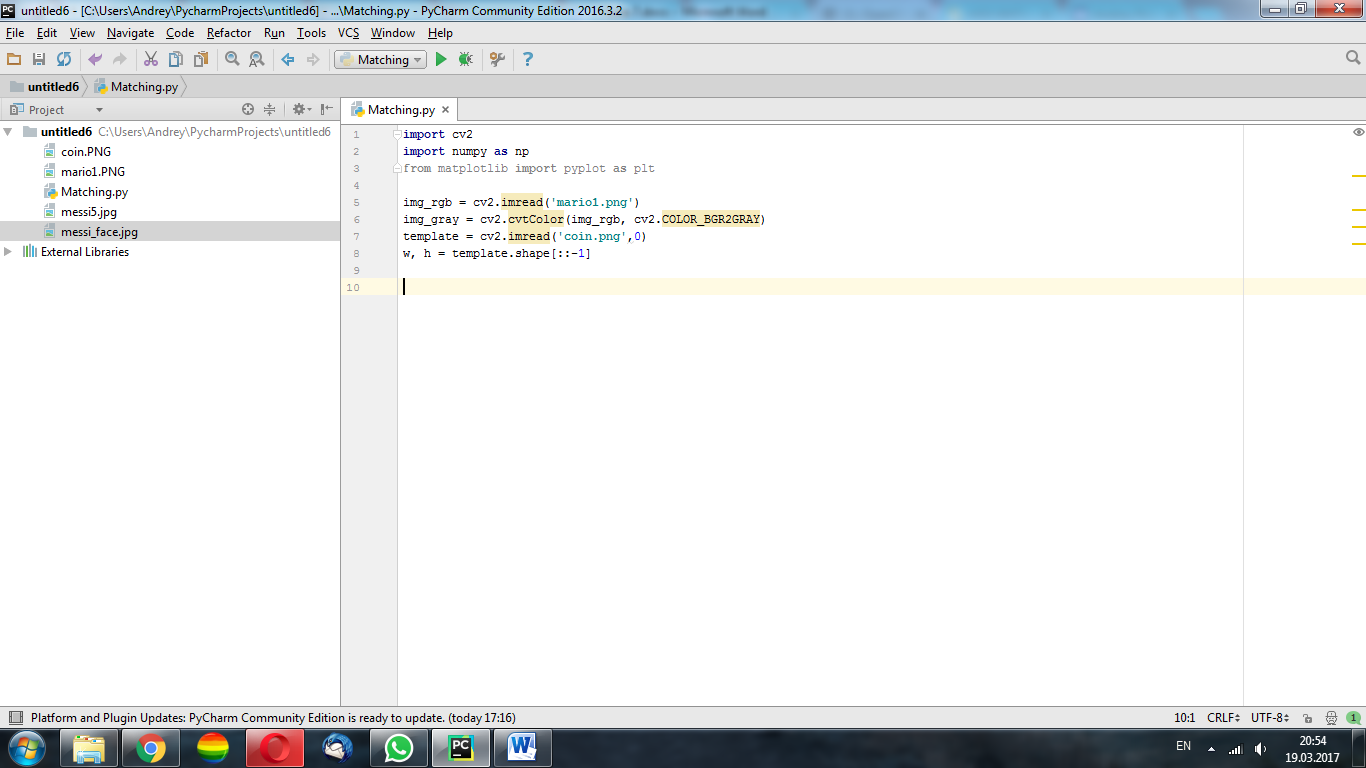
И зададим значение переменной template для чтения шаблона в сером цвете

template = cv2.imread(**'coin.png'**,0)



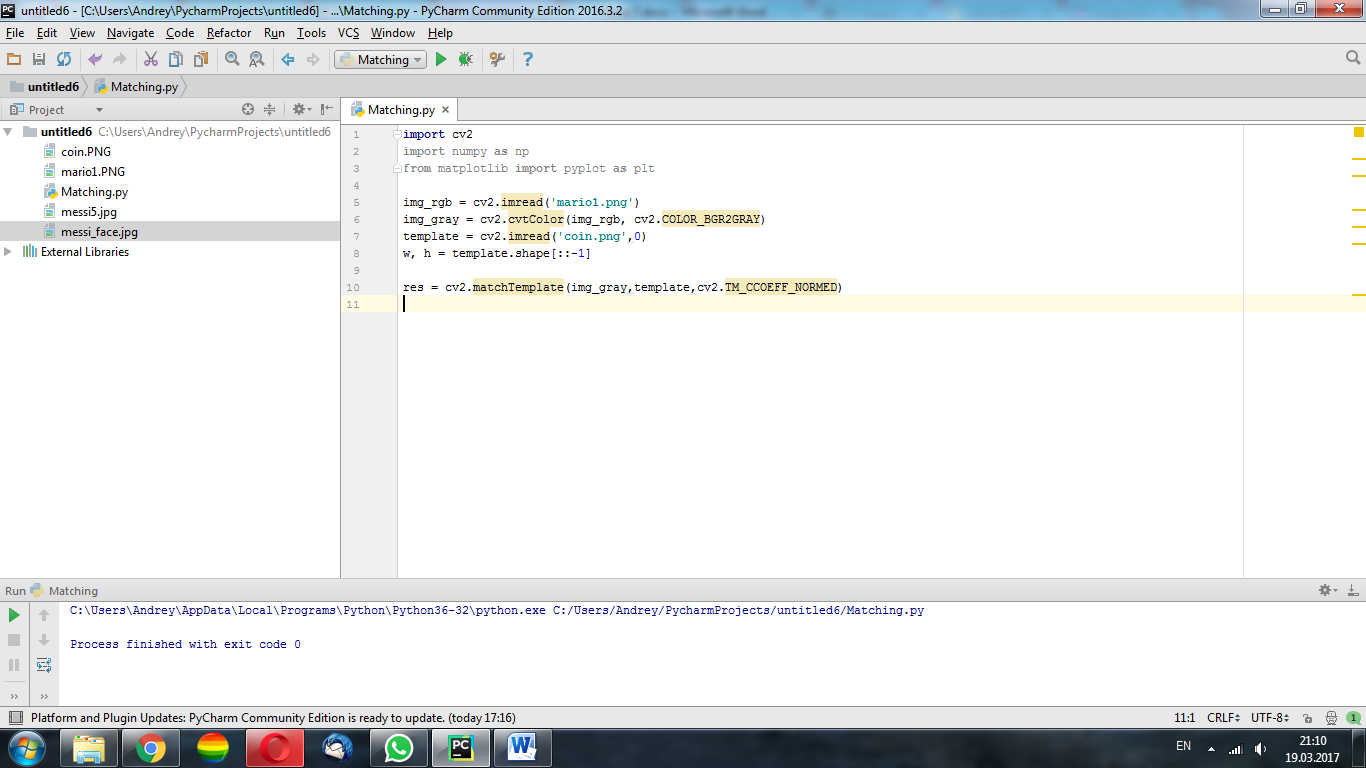
Для переменных w и h зададим массив с формой для шаблона

w, h = template.shape[::-1]



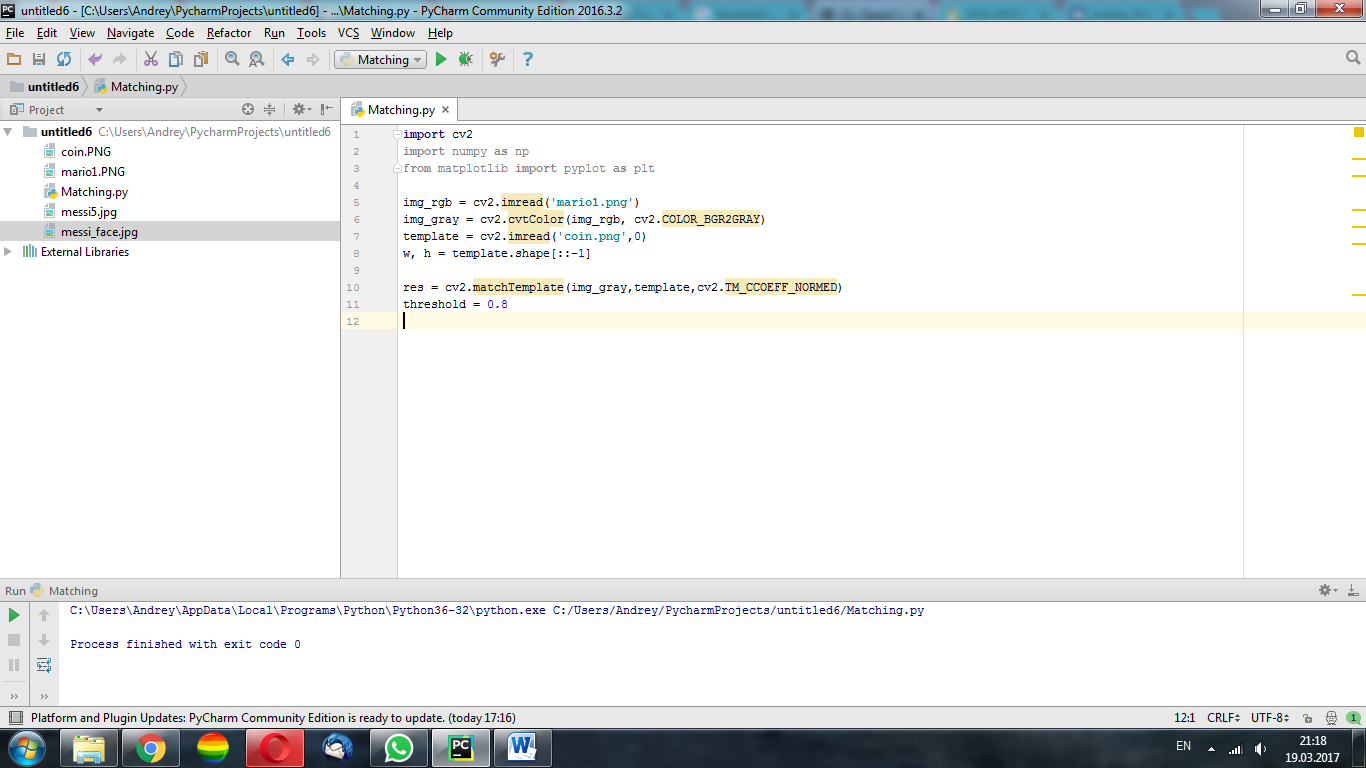
Для сравнения картинки с шаблоном и поиска соответствий, мы будем использовать функции cv2.matchTemplate и Tm\_ccoeff\_normed для переменных img\_gray и template

res = cv2.matchTemplate(img\_gray,template,cv2.TM\_CCOEFF\_NORMED)



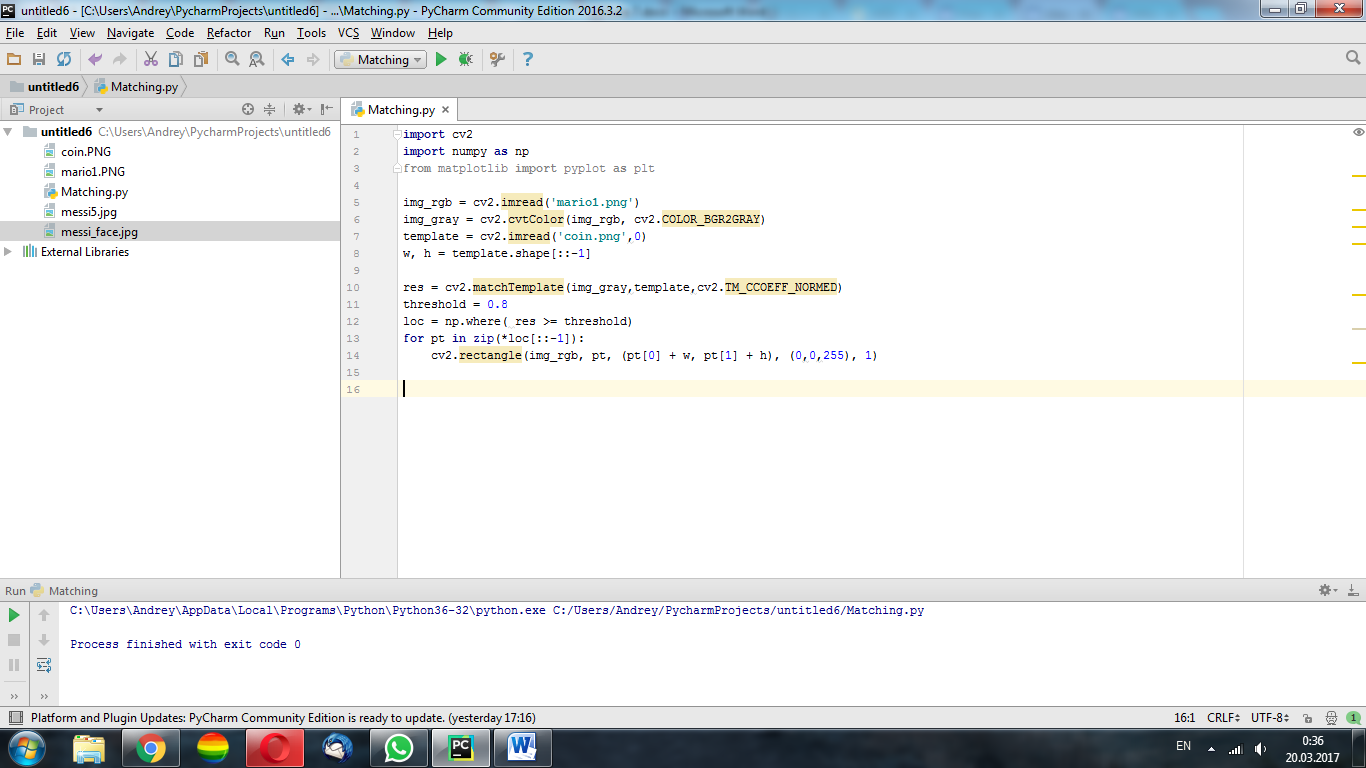
Пороговое значение для нашей картинки поставим 0.8

threshold = 0.8



Для выделения объектов схожих с нашим элементом зададим переменную loc, которая будет проверять найденные области в ответчик с пороговым значением, и цикл который будет выделять найденные области в прямоугольники (красного цвета с толщиной в 1пиксель) в соответствии с пороговым значением

loc = np.where( res >= threshold)  
**for** pt **in** zip(\*loc[::-1]):  
 cv2.rectangle(img\_rgb, pt, (pt[0] + w, pt[1] + h), (0,0,255), 1)

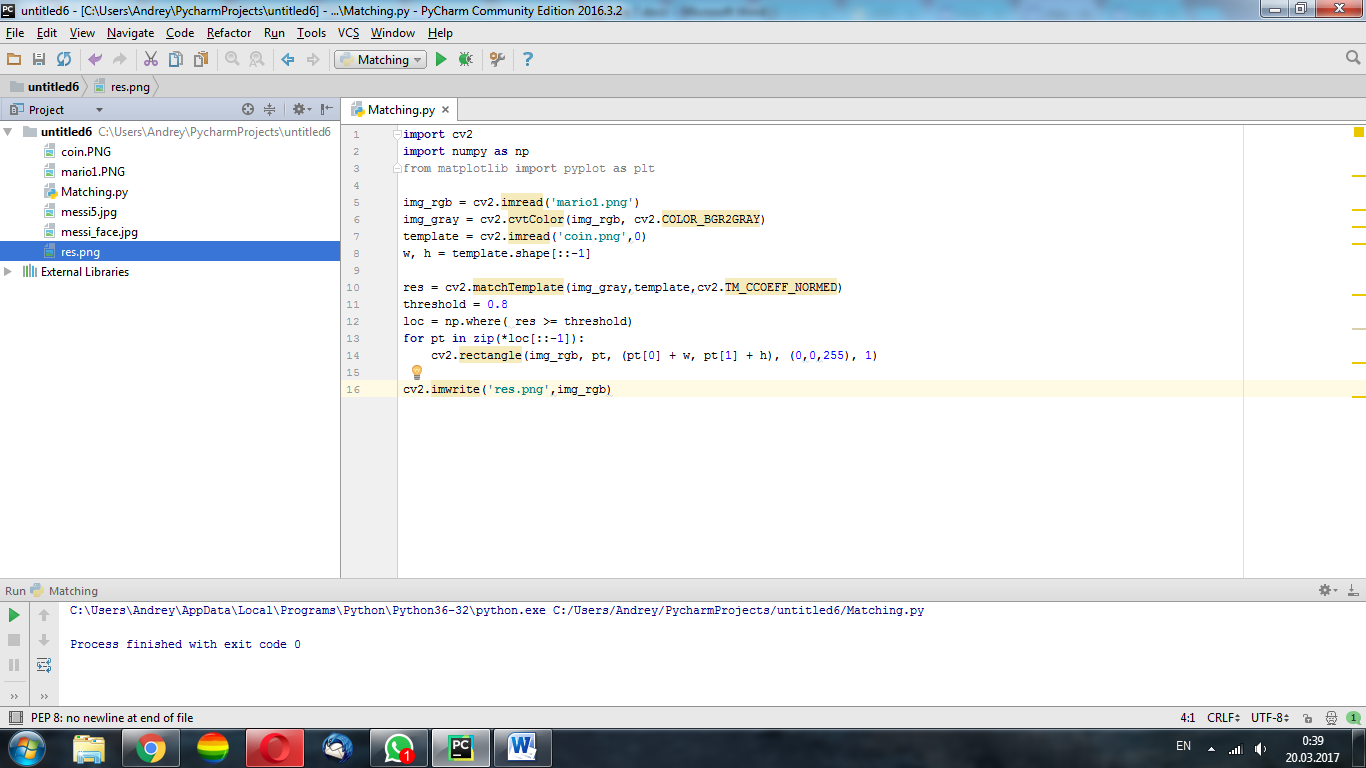


Далее будем записывать полученный результат в новый файл, а не показывать его в окне как обычно.

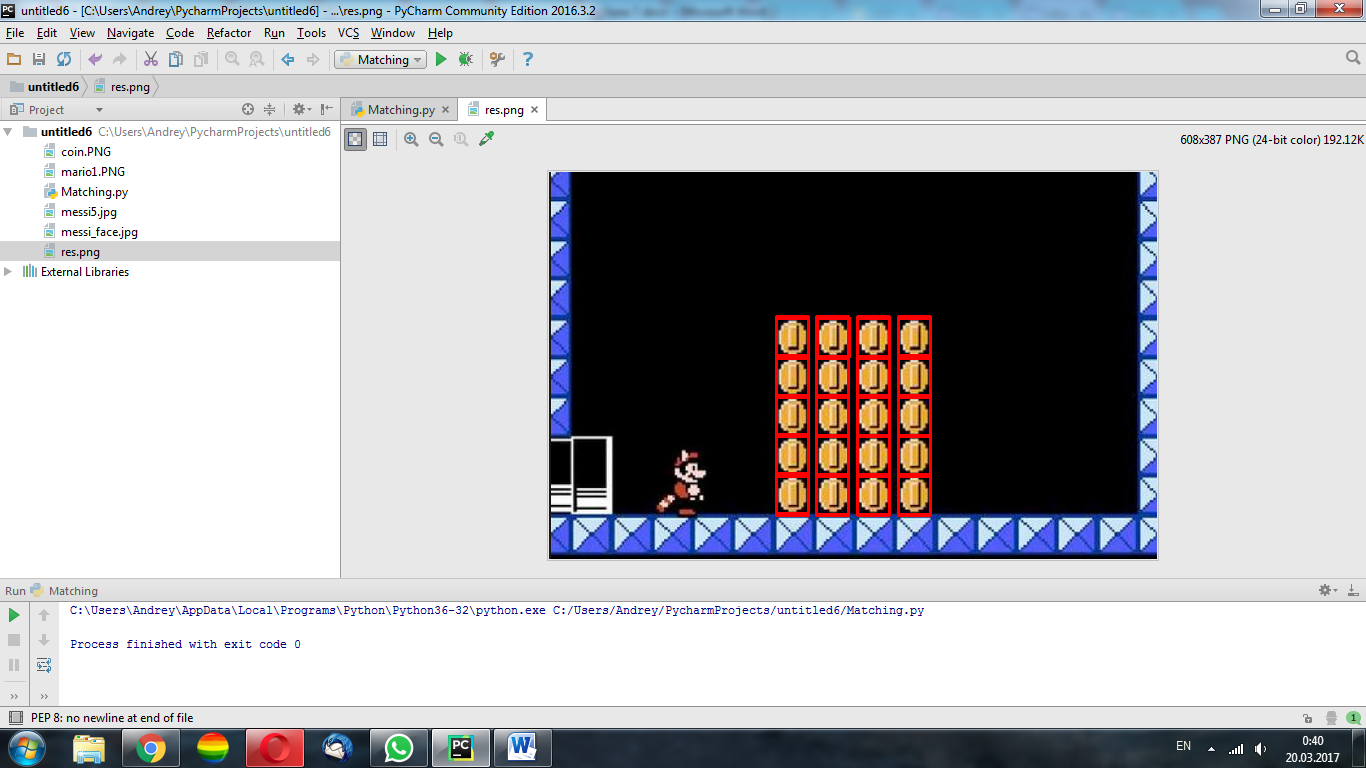
cv2.imwrite(**'res.png'**,img\_rgb)



Теперь, если вы запустите программу, то у вас создастся новый файл res.png кликните по нему два раза



Итоговый результат:



Можете загрузить свои картинки и шаблоны для них.