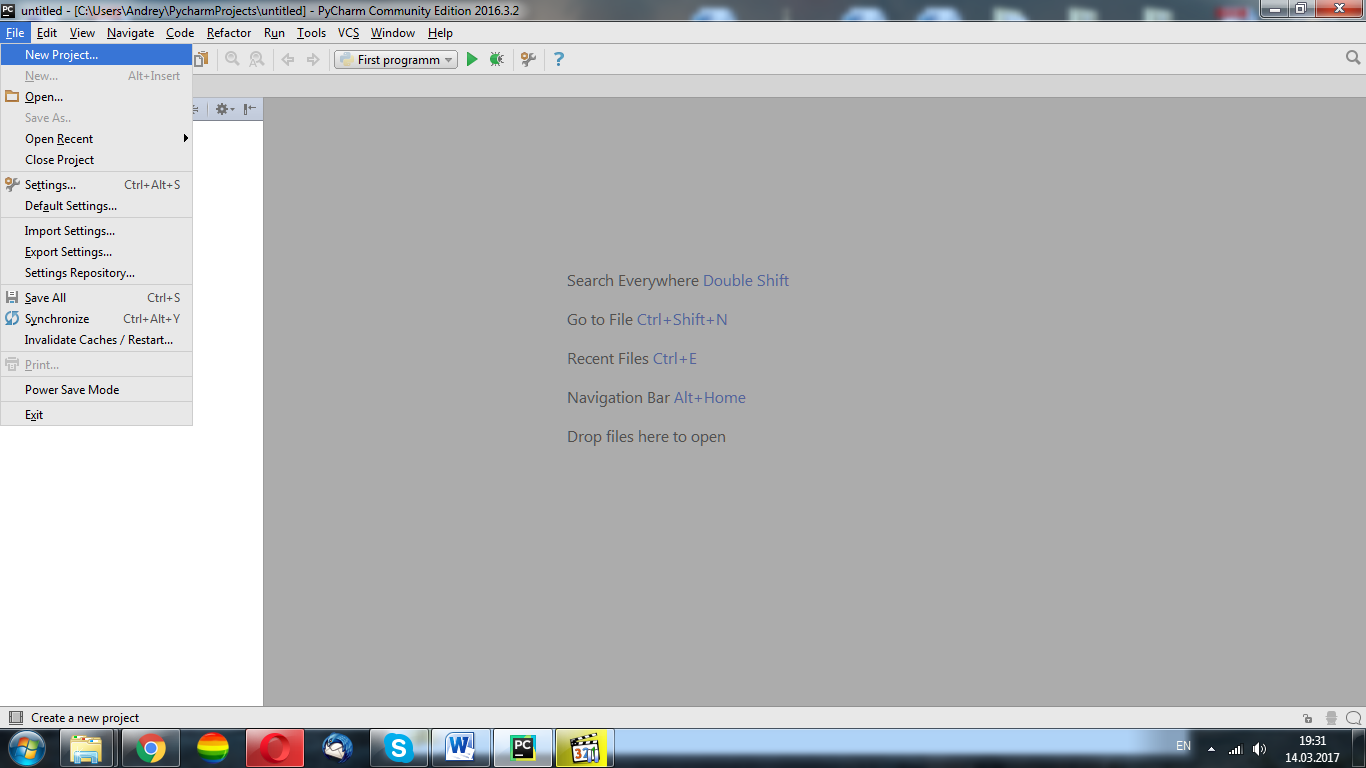
Урок № 3!

Сегодня мы освоим базовые операции с изображениями.

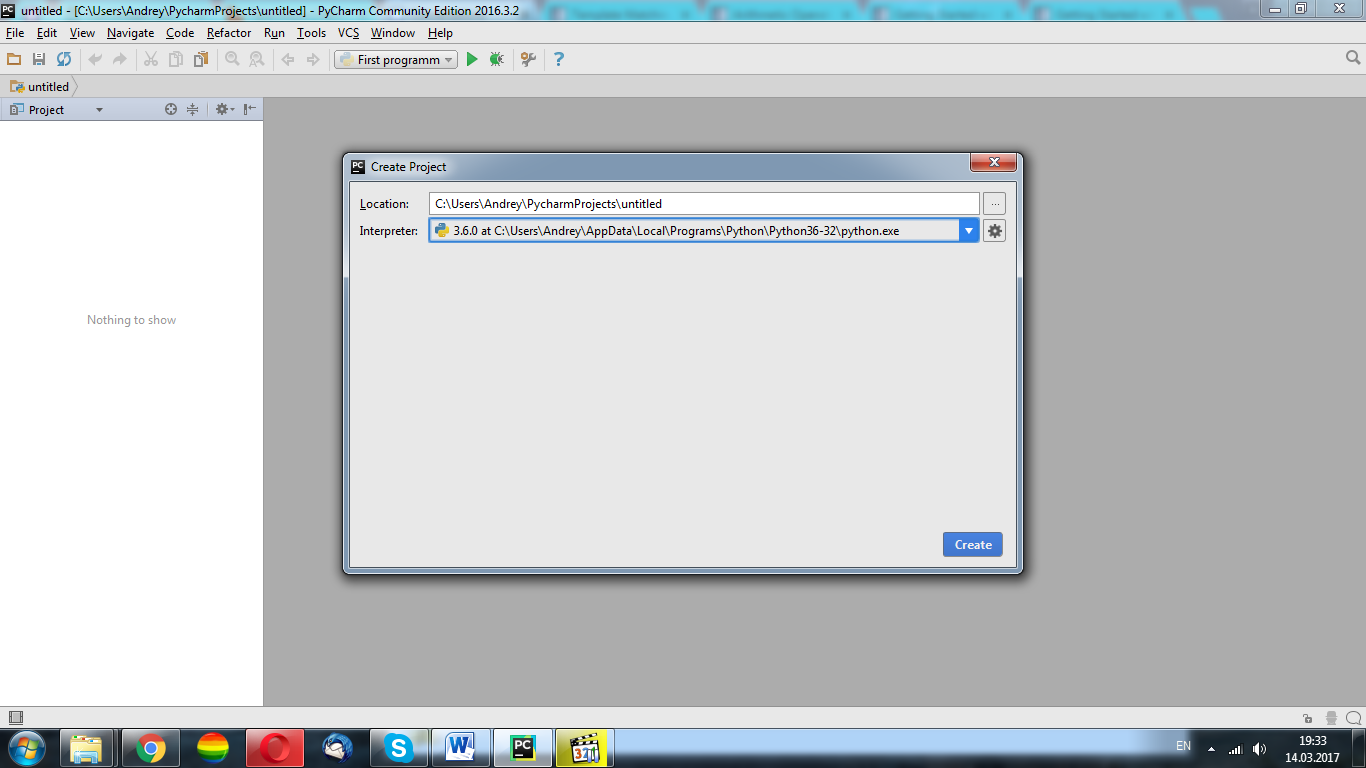
Запустите PyCharm Community Edition

Для начала нам нужно создать проект. Нажмите на вкладку File и выберите первый пункт New Project



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

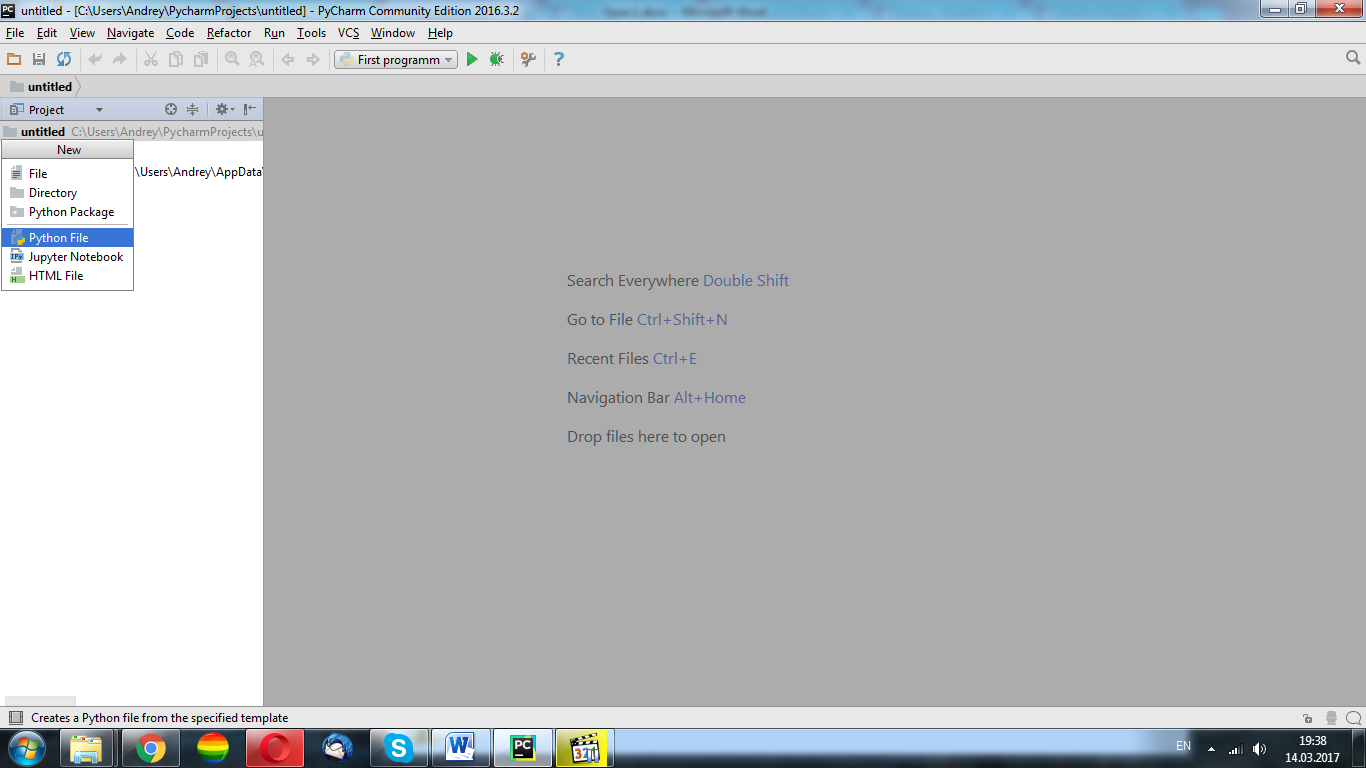
Во второй строке версия python. Выберите последнюю версию 3.6. Нажмите Create



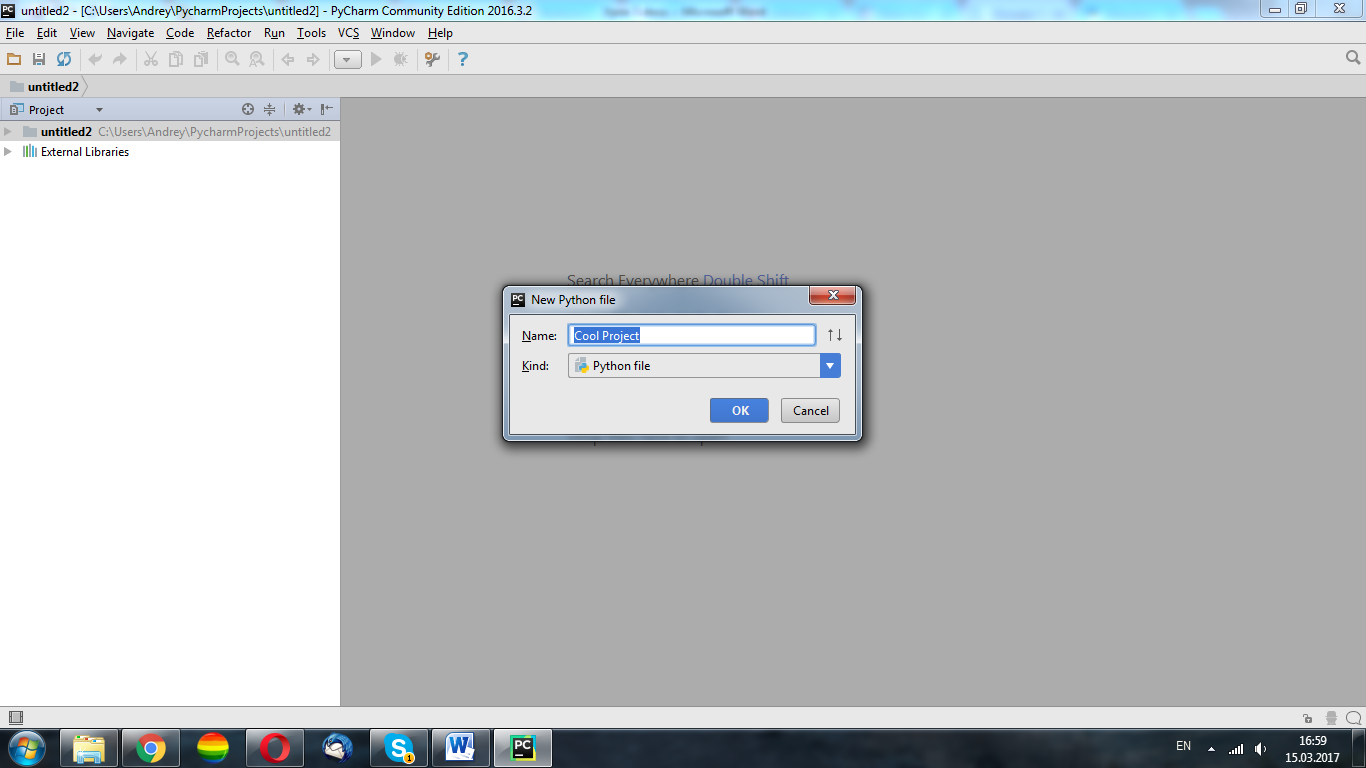
Далее создадим файл, в котором будем работать. Во вкладке File нажмите New…



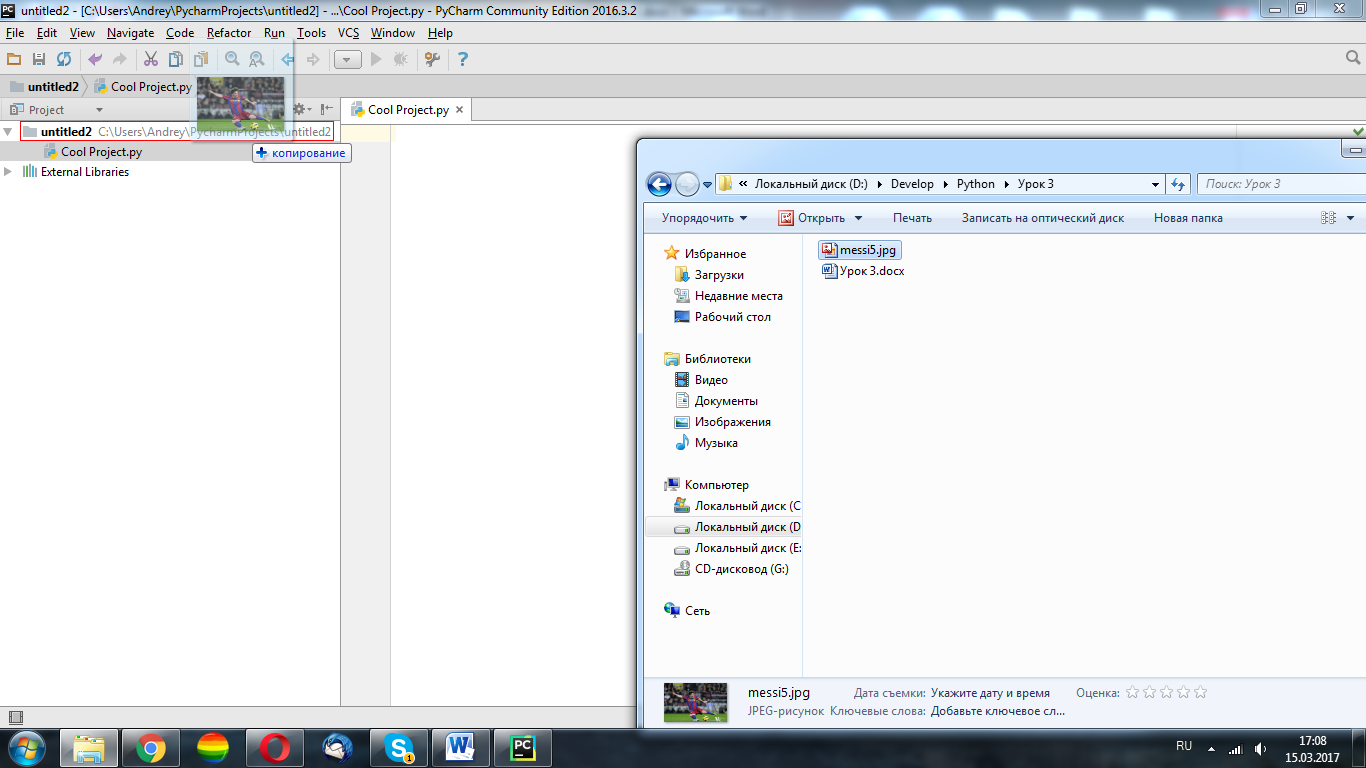
Python file



Назовем этот файл Cool Project



Для начала загрузим изображение. В папке с уроком есть картинка с названием «messi5», выделите ее и зажимая клавишу Ctrl перетащите в указанную область.



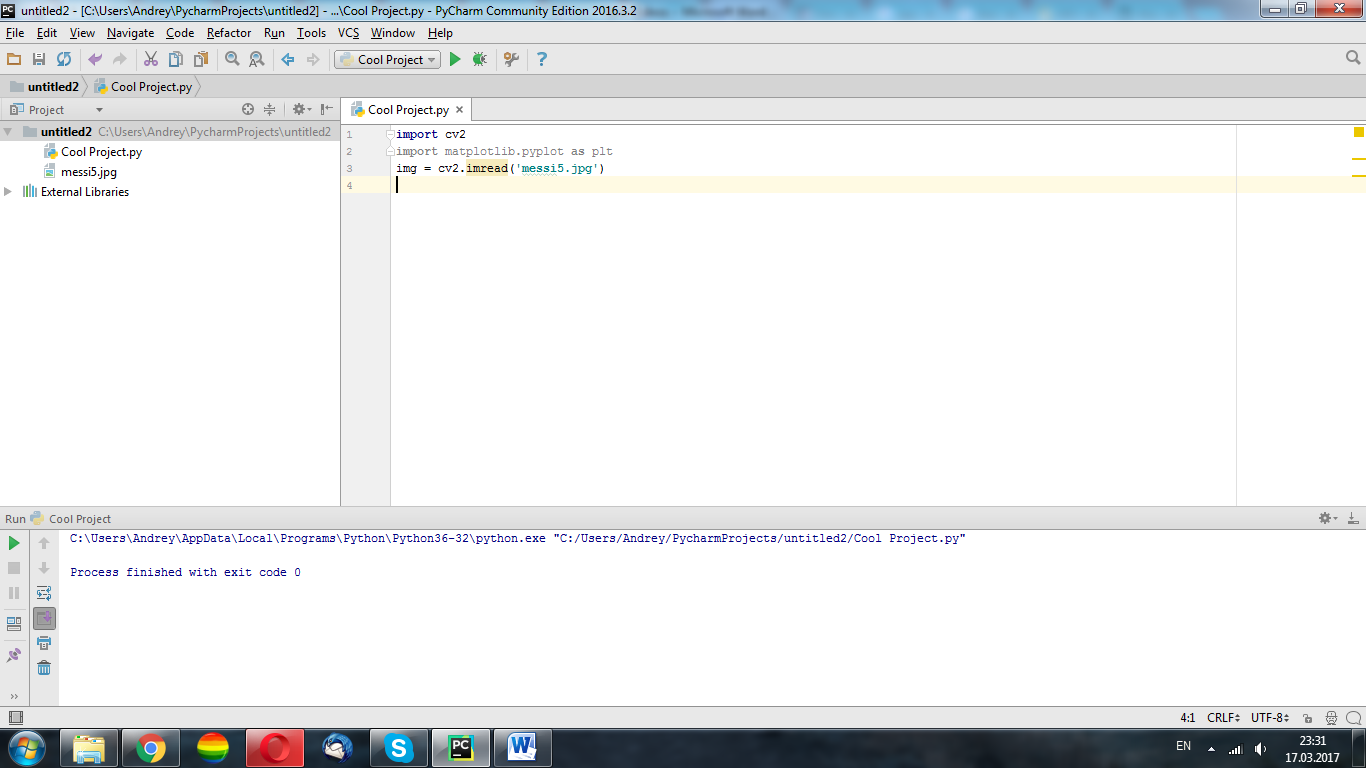
Начнем с копирования.

Иногда вам придется играть с определенной областью изображений. Для обнаружения глаз в изображениях сначала надо будет выполнить распознавание лица по изображению до тех пор, пока лицо не будет найдено, а затем выполнить поиск в области лица для глаз. Этот подход повышает точность (потому что глаза всегда на лицах) и производительность (потому что мы ищем небольшую область).

Скопируем мяч в другую область картинки.

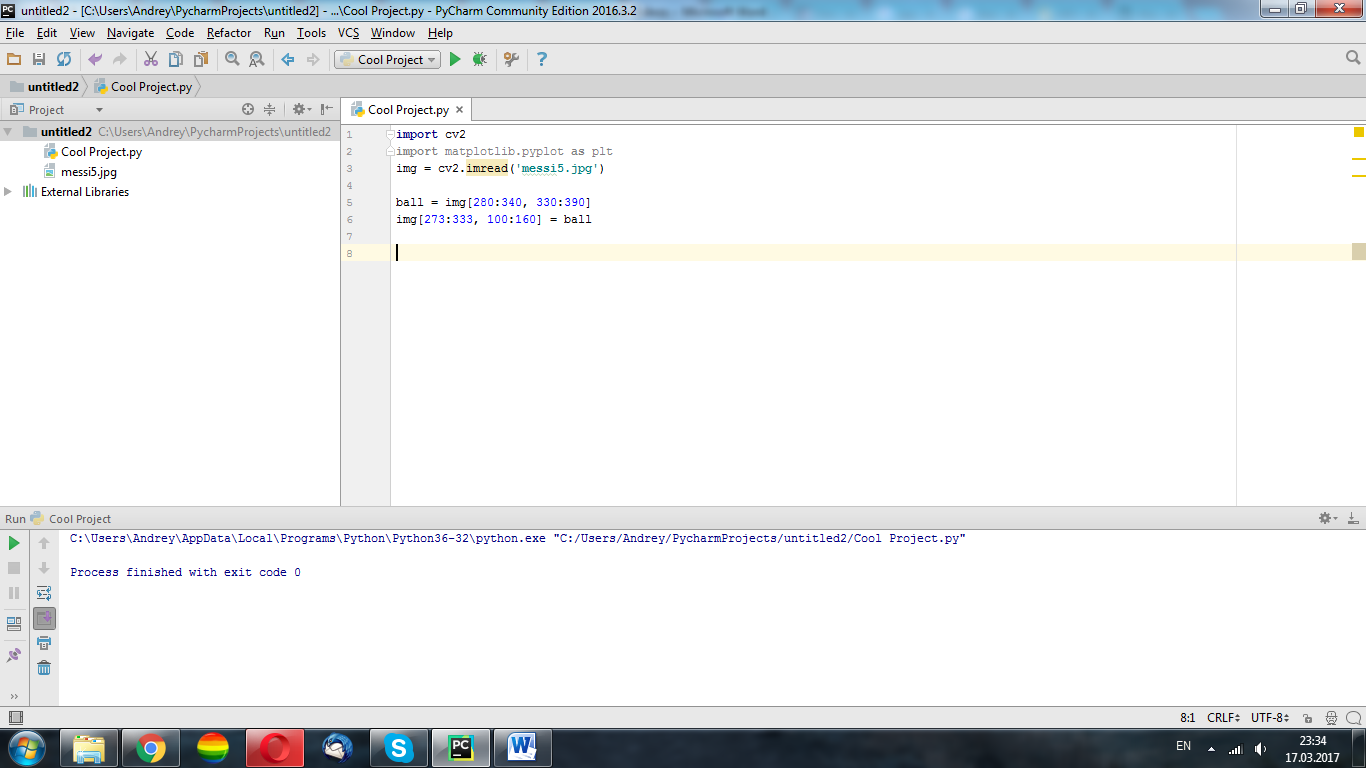
Начнем с подключения библиотеки cv2, matplotlib, и нашего изображения.

**import** cv2  
**import** matplotlib.pyplot **as** plt  
img = cv2.imread(**'messi5.jpg'**)



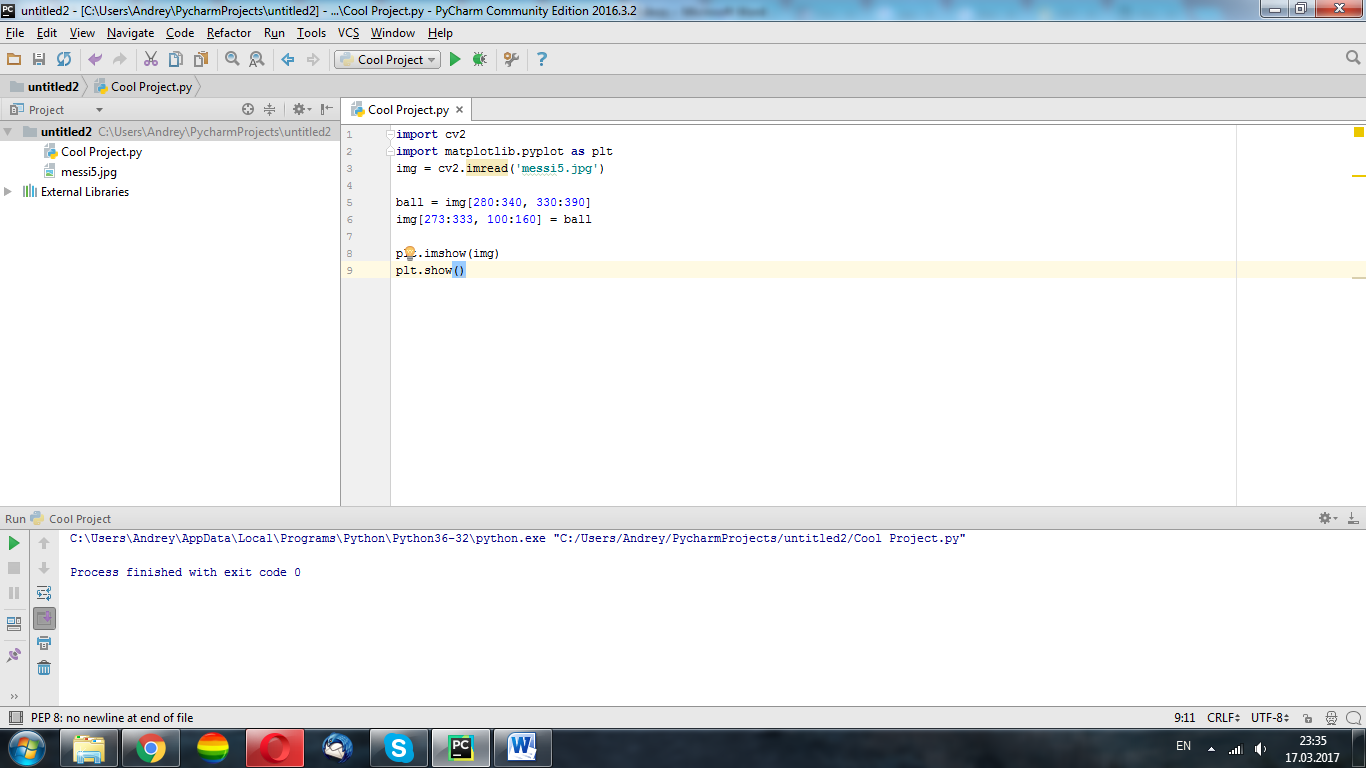
Укажем координаты мяча и координаты, в которые мы его хотим перенести

ball = img[280:340, 330:390]  
img[273:333, 100:160] = ball

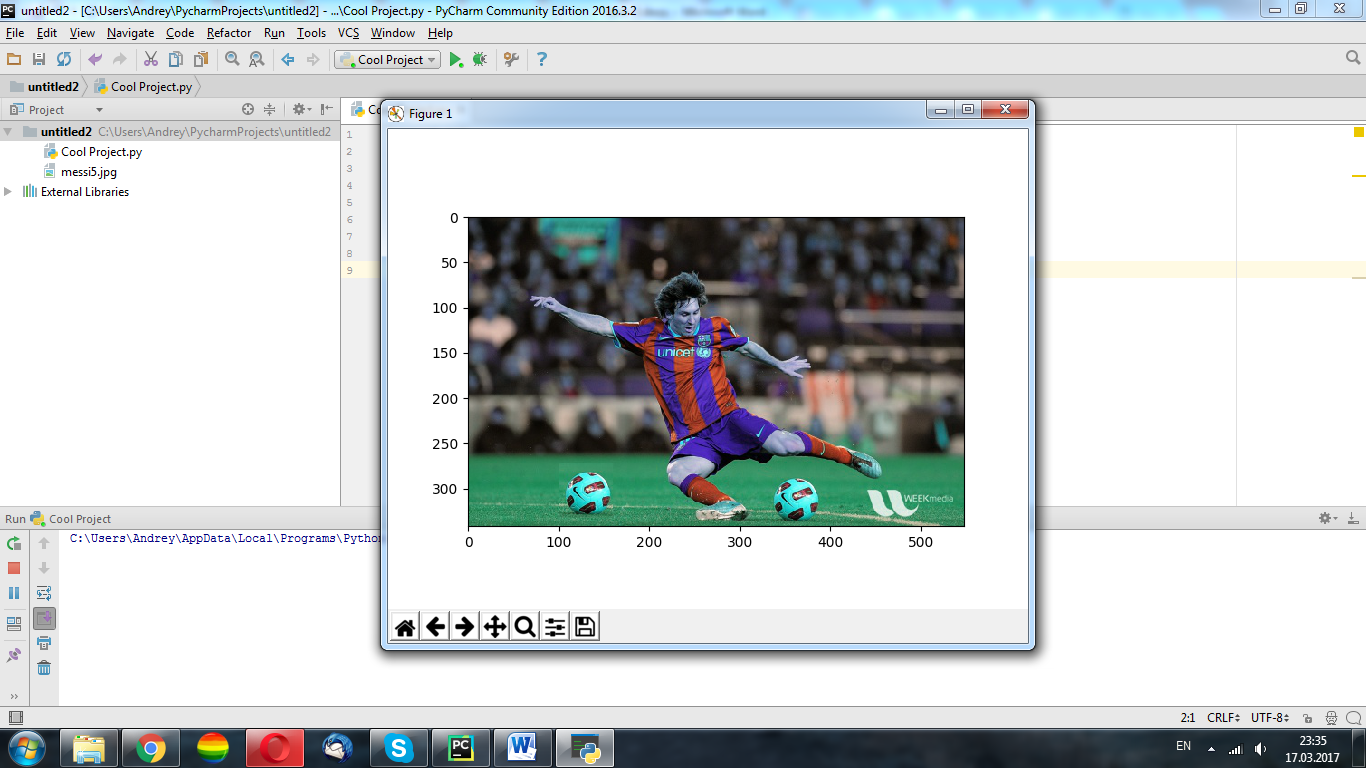


И выведем изображение

plt.imshow(img)  
plt.show()



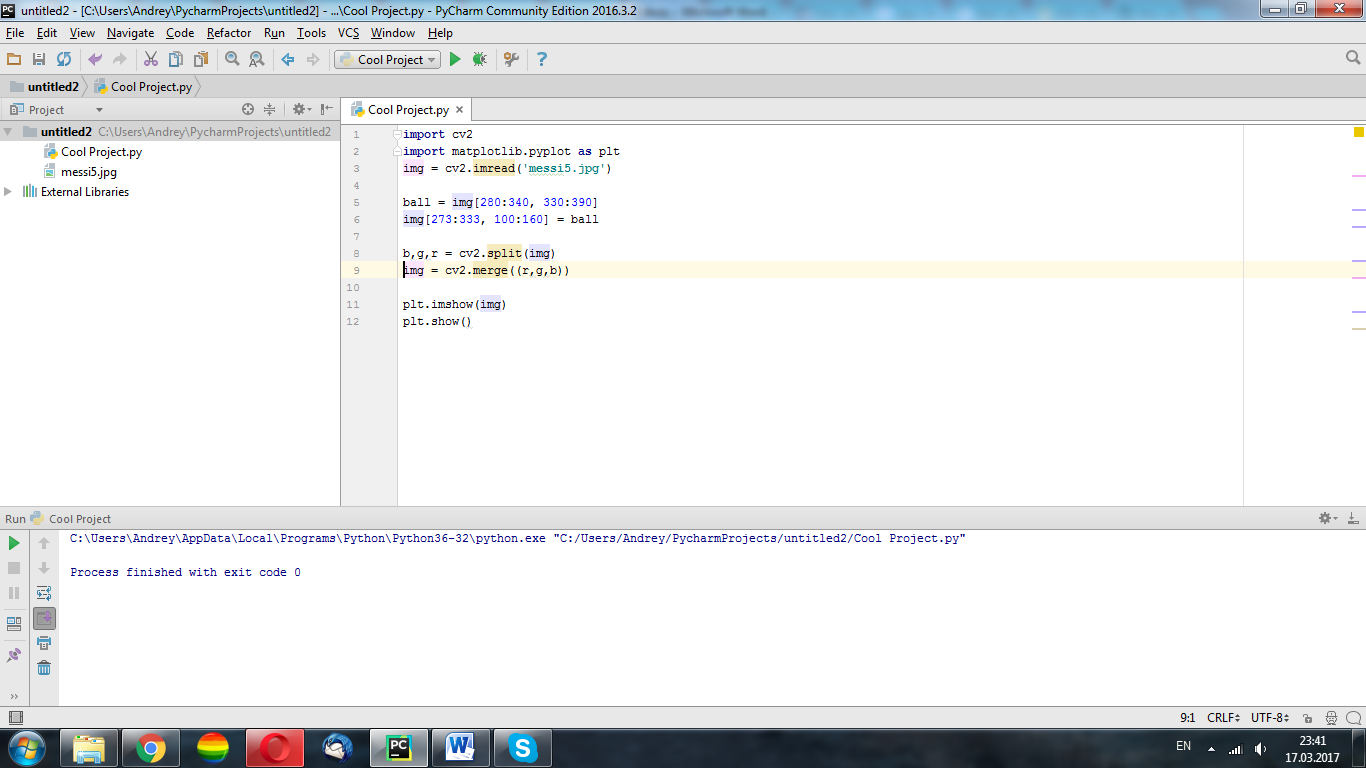
Результат:



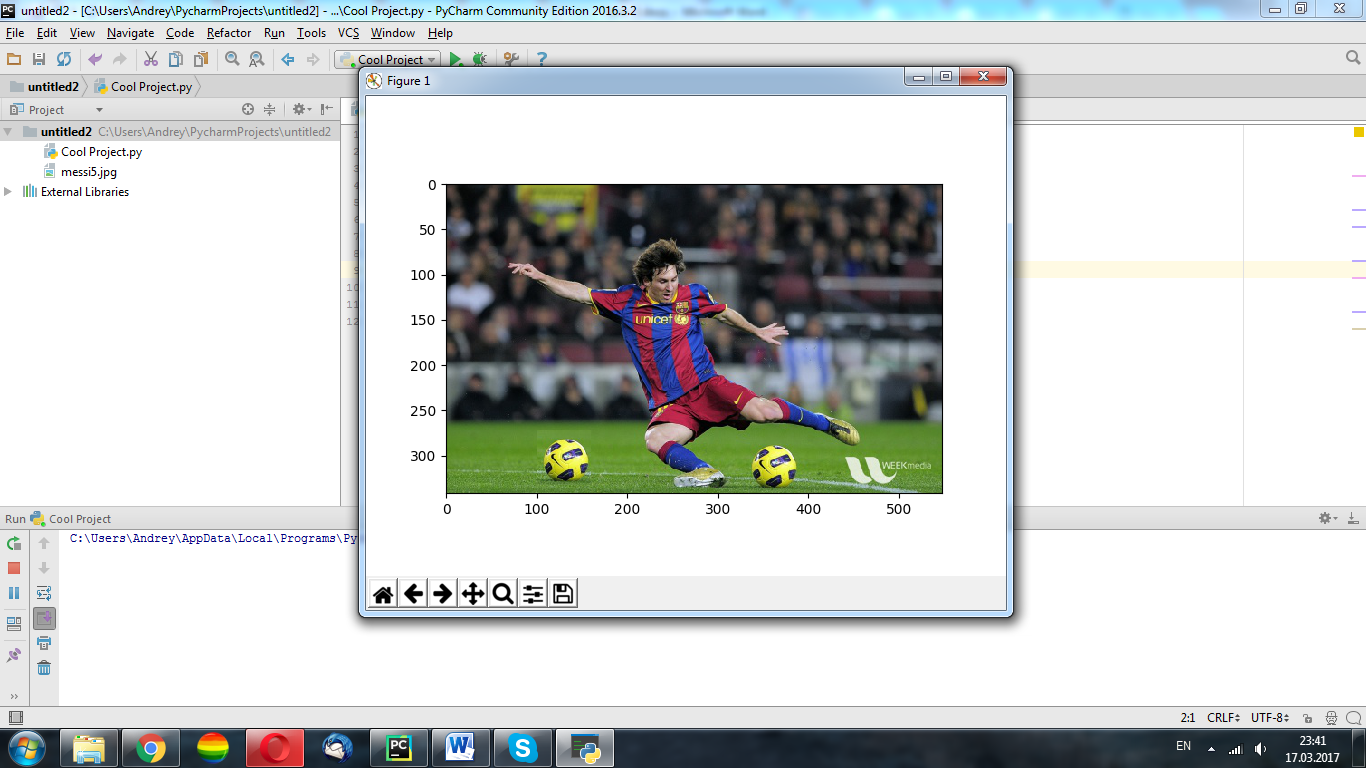
Библиотека, с которой мы работаем, меняет цветовую схему с rgb на bgr. Мы можем разложить его и собрать в нужном нам порядке

С помощью функции split мы разложим цветовую схему на bgr, а с помощью функции megre соберем ее обратно в привычный нам rgb

b,g,r = cv2.split(img)  
  
img = cv2.merge((r,g,b))



И вот что мы имеем теперь:



Можете скопировать мячик или голову в любое место на картинке.

Рамки.

Если вы хотите создать рамку вокруг изображения, что-то вроде фоторамки, вы можете использовать функцию cv2.copyMakeBorder (). Но он имеет больше приложений для операции свертки, нулевого заполнения и т.д. Эта функция принимает следующие аргументы:

Src - входное изображение

Верхняя, нижняя, левая, правая - ширина рамки в количестве пикселей в соответствующих направлениях

BorderType - Флаг, определяющий, какую границу добавить. Это могут быть следующие типы:

Cv2.BORDER\_CONSTANT - добавляет постоянную цветную рамку. Значение должно быть указано как следующий аргумент.

Cv2.BORDER\_REFLECT - Граница будет зеркальным отражением элементов границы, например: fedcba | abcdefgh | hgfedcb

Cv2.BORDER\_REFLECT\_101 или cv2.BORDER\_DEFAULT - То же, что и выше, но с небольшими изменениями, например: gfedcb | abcdefgh | gfedcba

Cv2.BORDER\_REPLICATE - последний элемент повторяется повсюду, например: aaaaaa | abcdefgh | hhhhhhh

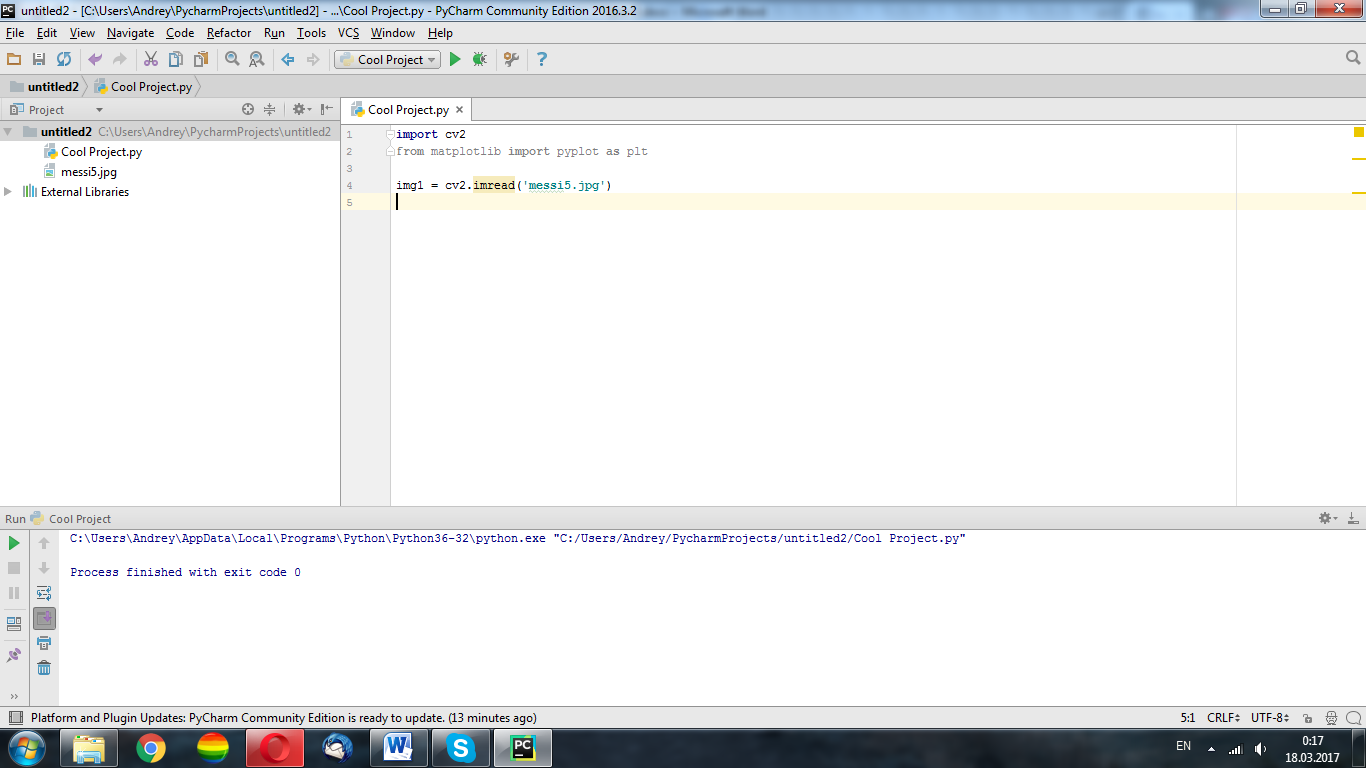
Cv2.BORDER\_WRAP - Не могу объяснить, это будет выглядеть так: cdefgh | abcdefgh | abcdefg

Value - Цвет границы, если тип границы - cv2.BORDER\_CONSTANT

Далее мы напишем код который, демонстрирующий все эти типы границ для лучшего понимания:

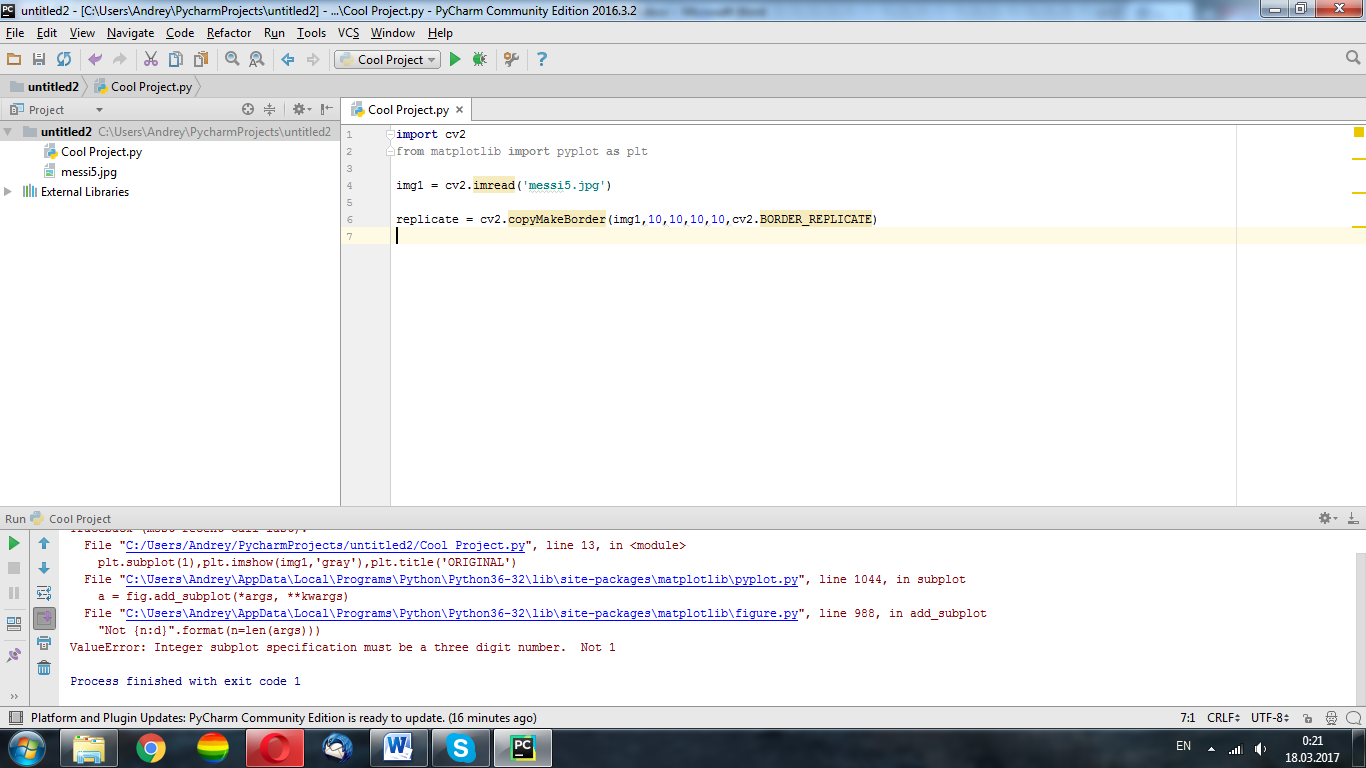
Оставьте только верхние 3 строчки к переменной img добавьте 1.

**import** cv2  
**from** matplotlib **import** pyplot **as** plt  
  
img1 = cv2.imread(**'messi5.jpg'**)



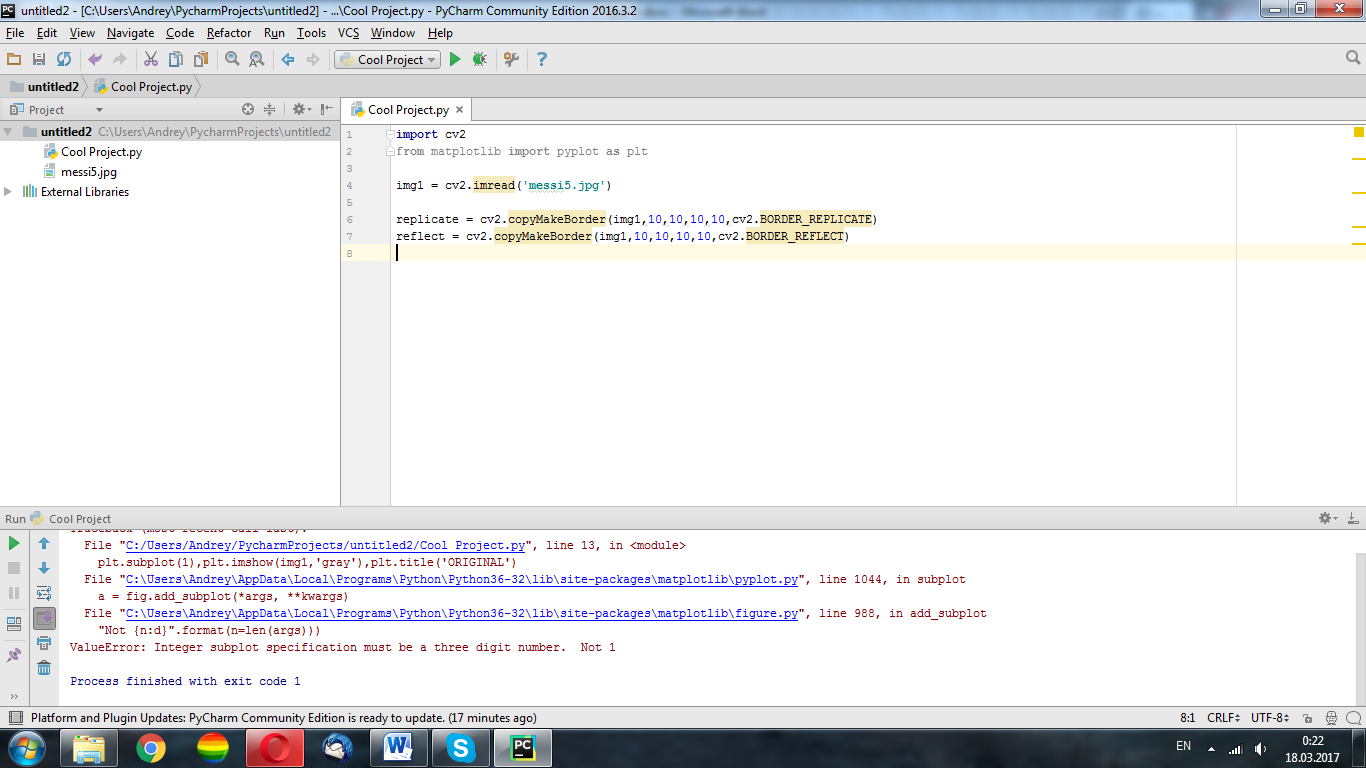
И по функциям, которые мы описывали выше, будем делать рамки вокруг нашего изображения. Начнем с replicate (рамка будет в 10 пикселей)

replicate = cv2.copyMakeBorder(img1,10,10,10,10,cv2.BORDER\_REPLICATE)



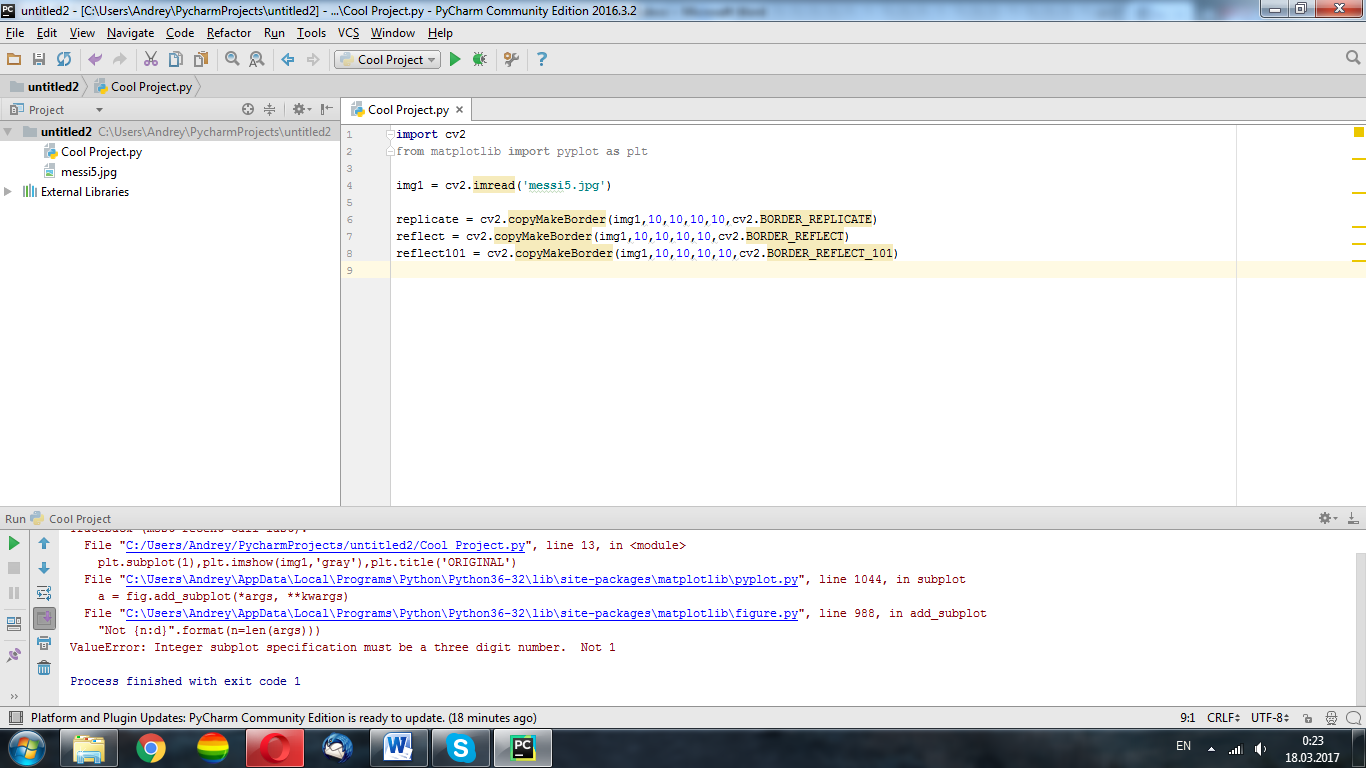
Далее reflect

reflect = cv2.copyMakeBorder(img1,10,10,10,10,cv2.BORDER\_REFLECT)



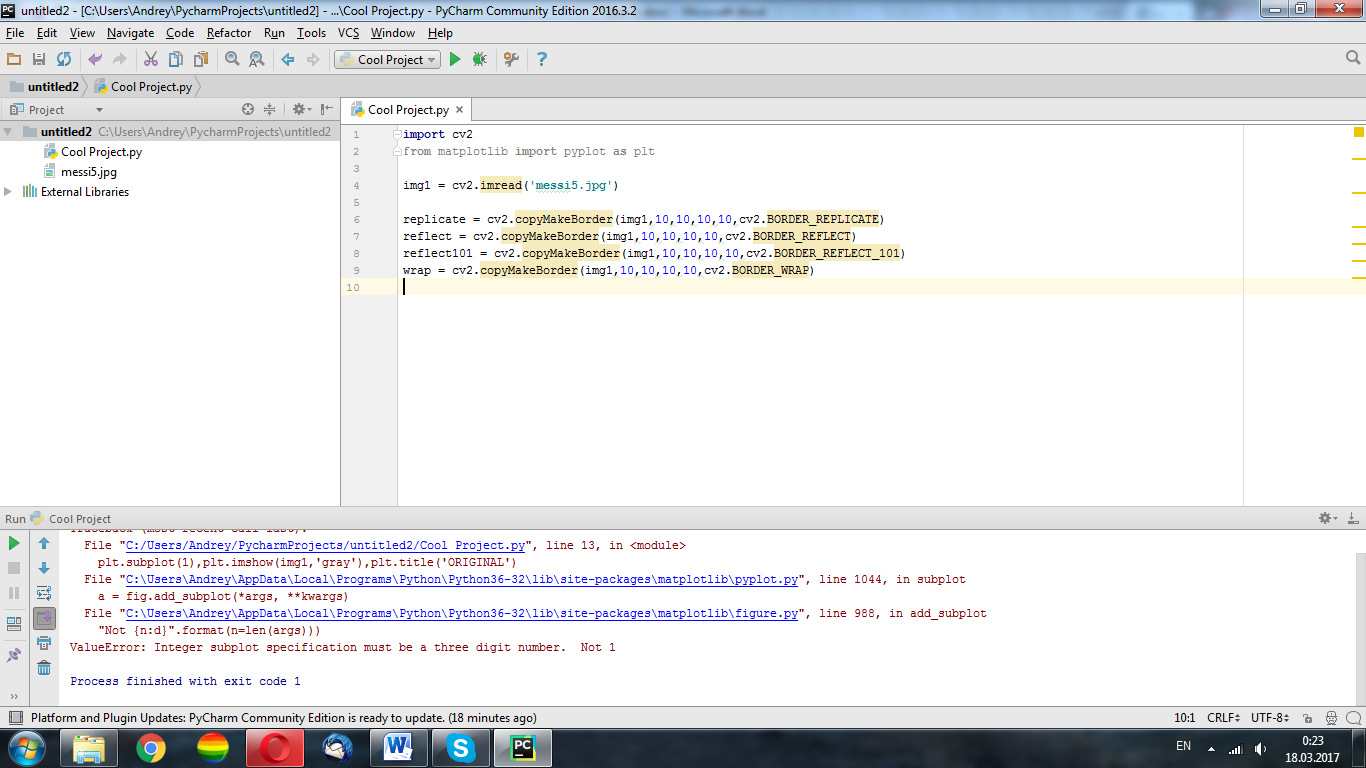
Reflect101

reflect101 = cv2.copyMakeBorder(img1,10,10,10,10,cv2.BORDER\_REFLECT\_101)



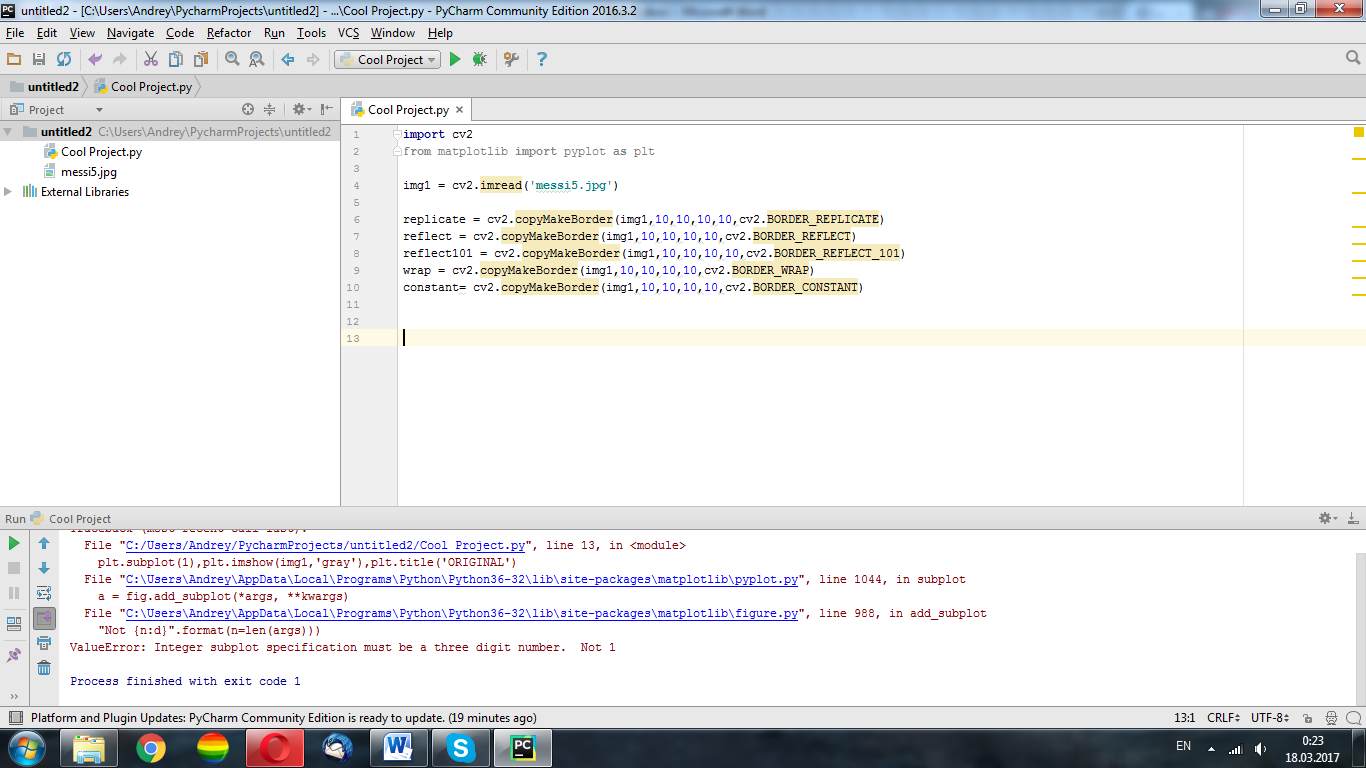
Wrap

wrap = cv2.copyMakeBorder(img1,10,10,10,10,cv2.BORDER\_WRAP)



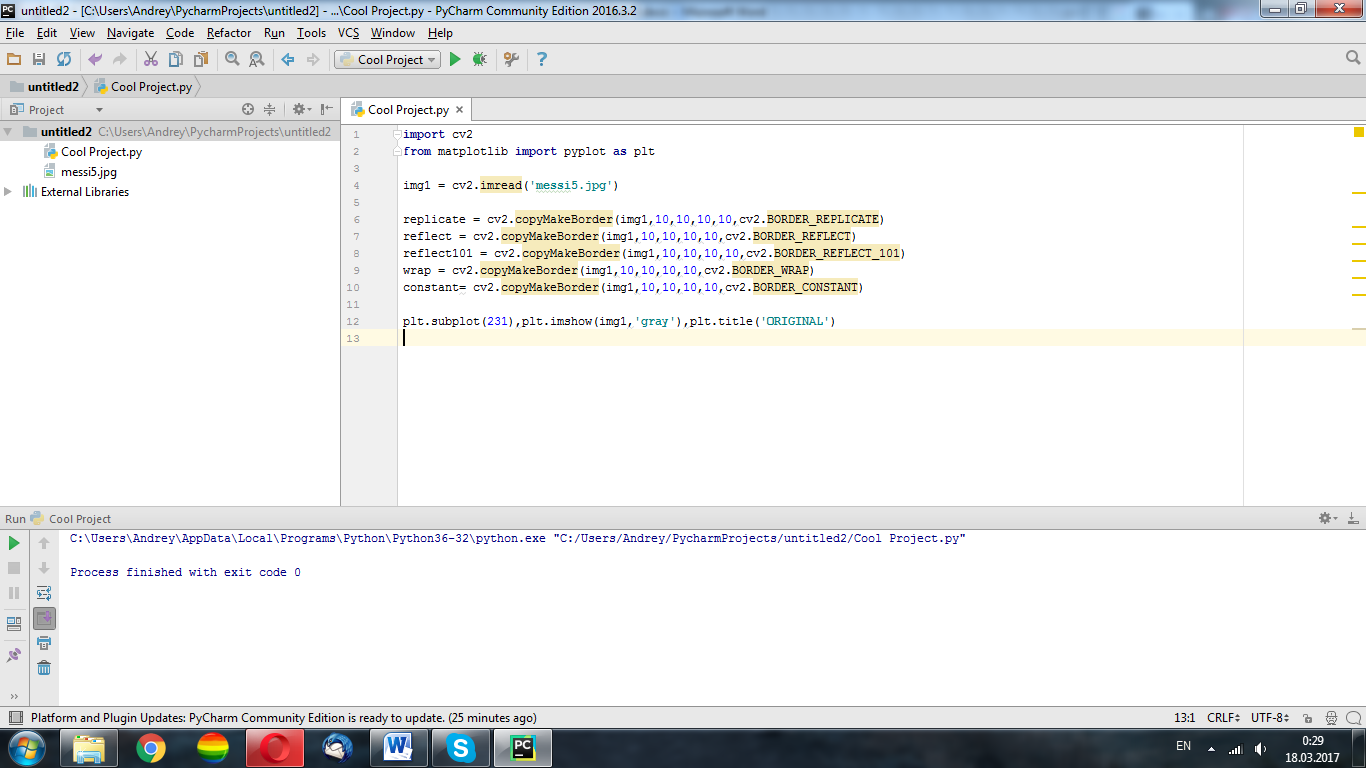
И constant

constant= cv2.copyMakeBorder(img1,10,10,10,10,cv2.BORDER\_CONSTANT)



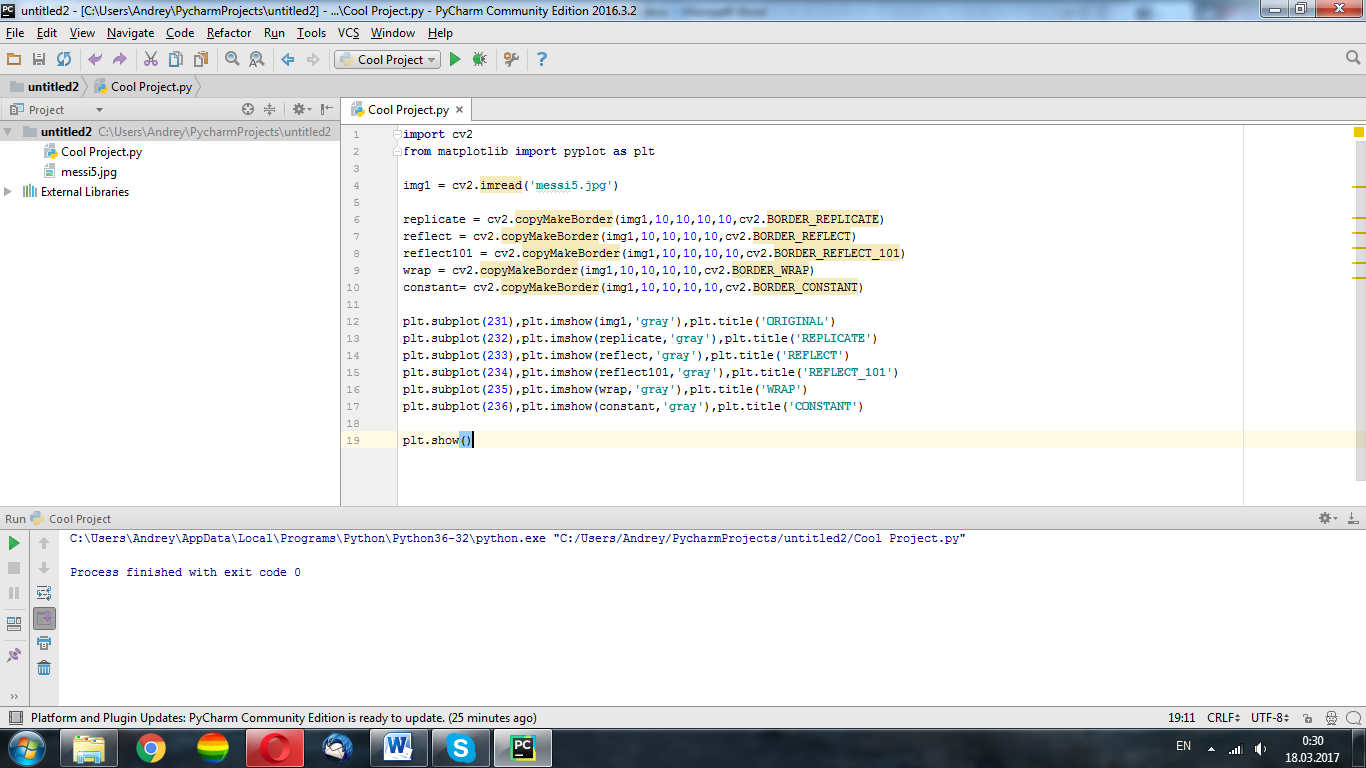
Теперь выведем все изображения. Нам понадобится функция plt.subplot для того чтобы выводить изображения в две строки и три столбца в одном окне.

plt.subplot(231),plt.imshow(img1,**'gray'**),plt.title(**'ORIGINAL'**)

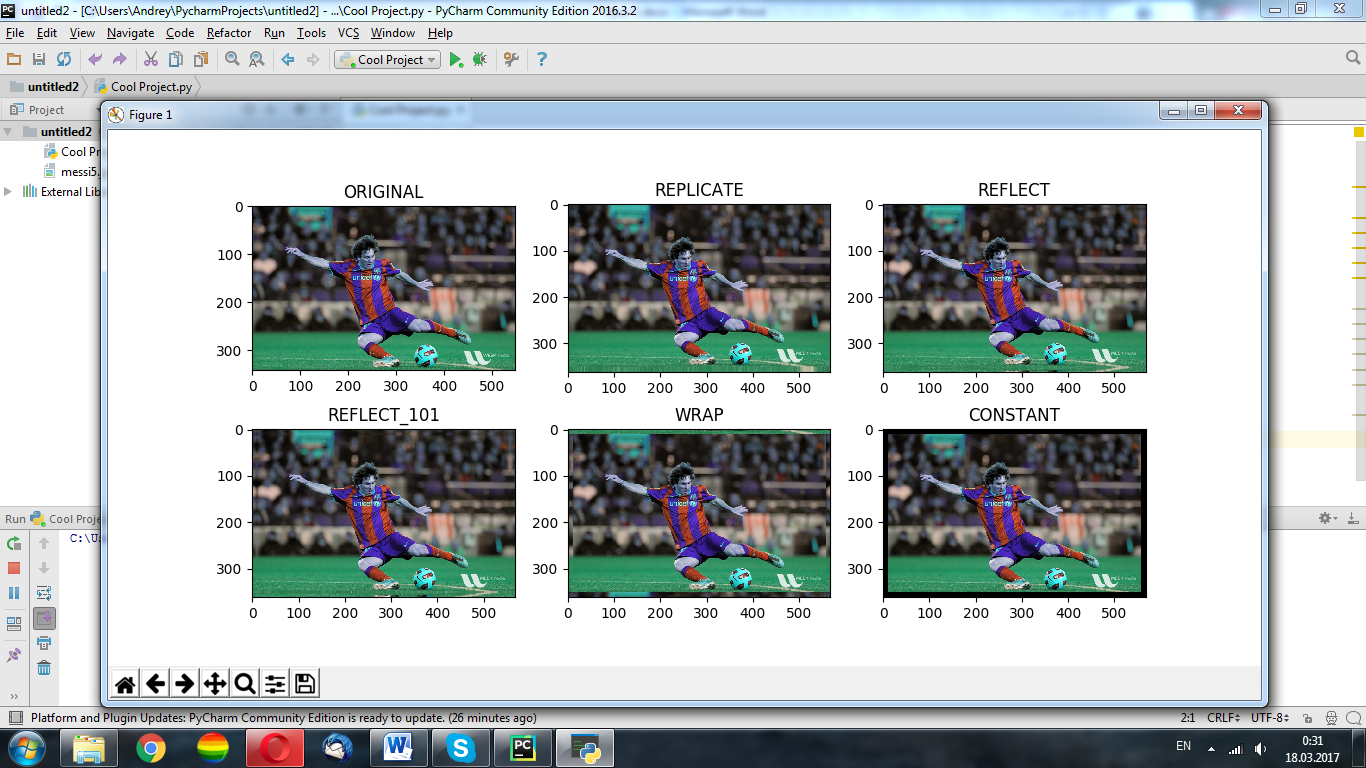


И далее все остальные со своими позициями и заголовками и выводом основного окна

plt.subplot(232),plt.imshow(replicate,**'gray'**),plt.title(**'REPLICATE'**)  
plt.subplot(233),plt.imshow(reflect,**'gray'**),plt.title(**'REFLECT'**)  
plt.subplot(234),plt.imshow(reflect101,**'gray'**),plt.title(**'REFLECT\_101'**)  
plt.subplot(235),plt.imshow(wrap,**'gray'**),plt.title(**'WRAP'**)  
plt.subplot(236),plt.imshow(constant,**'gray'**),plt.title(**'CONSTANT'**)  
  
plt.show()

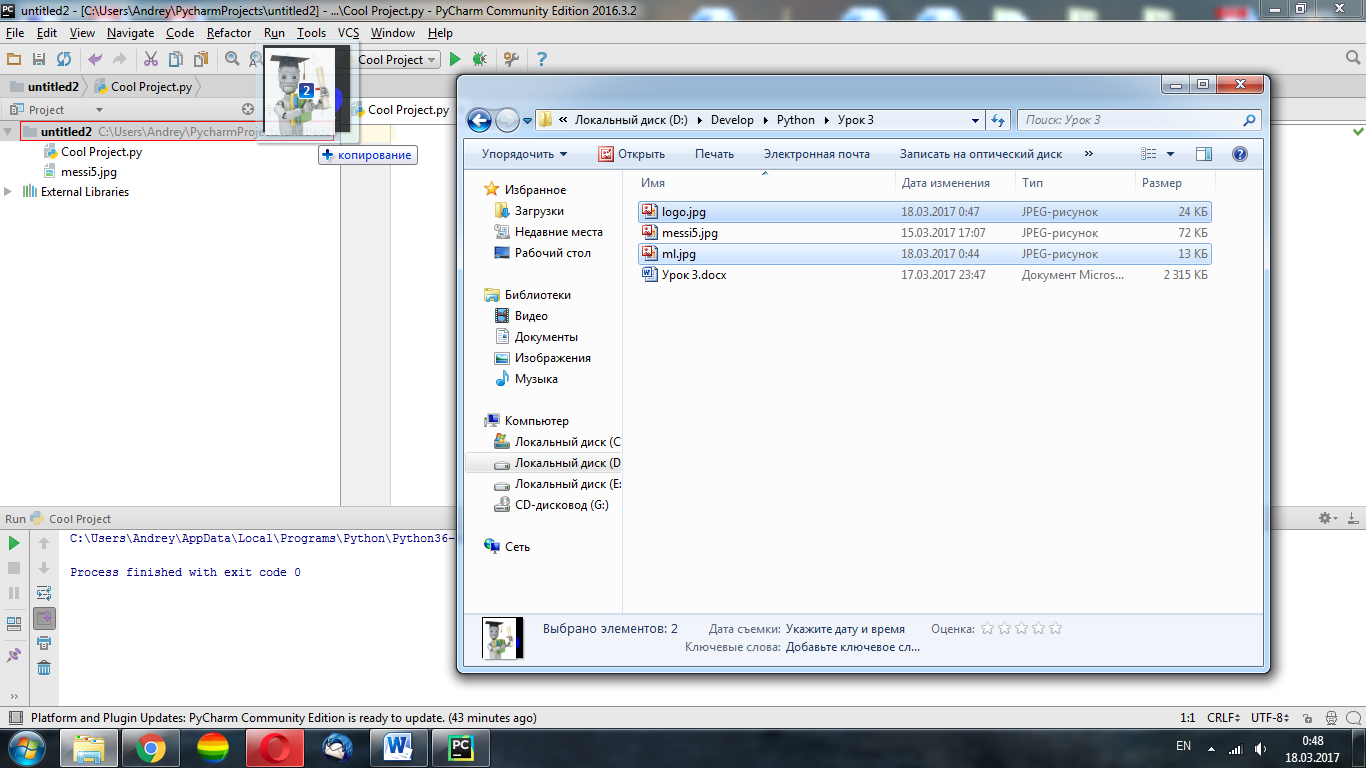


Результат. Обратите внимания на рамки. Можете сделать их большего размера



Перейдем к арифметическим операция с изображениями.

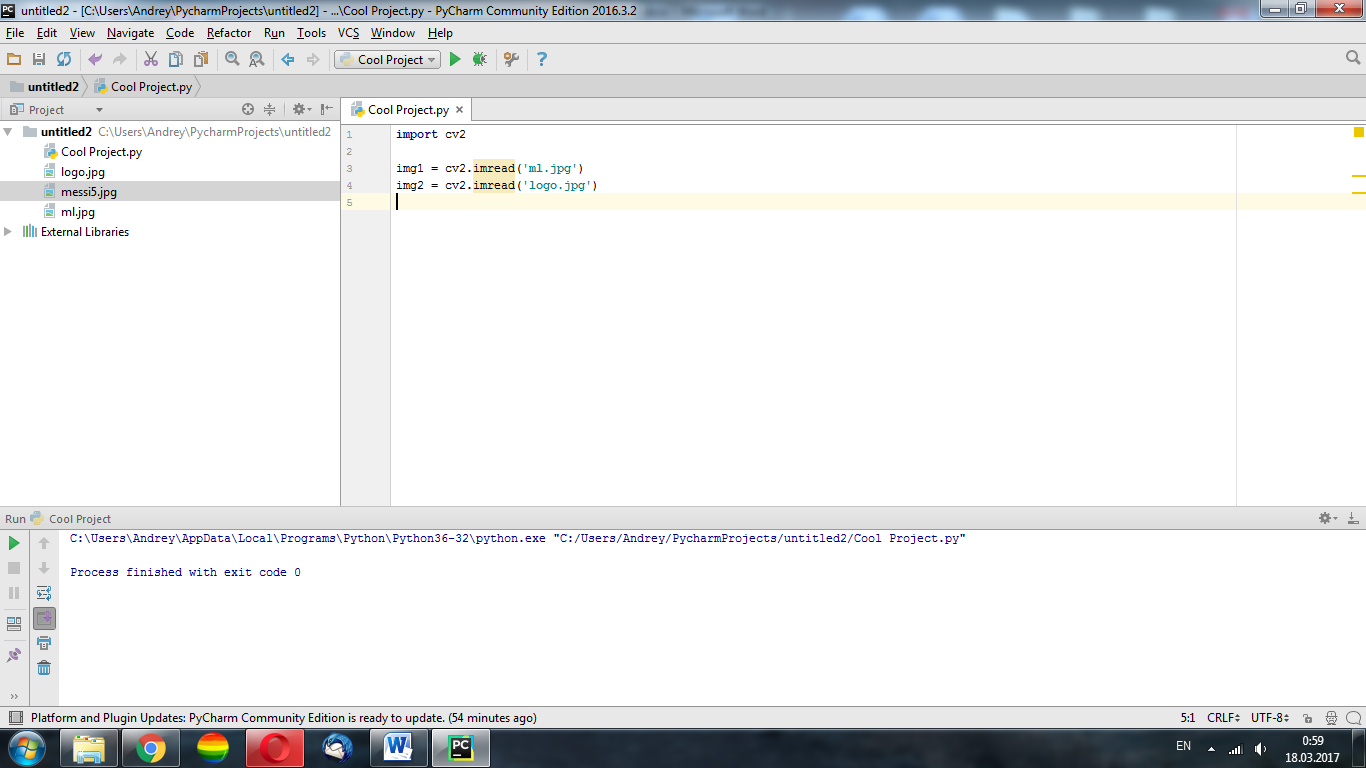
Можете удалить весь код и загрузить еще две картинки



Сложим два этих изображения.

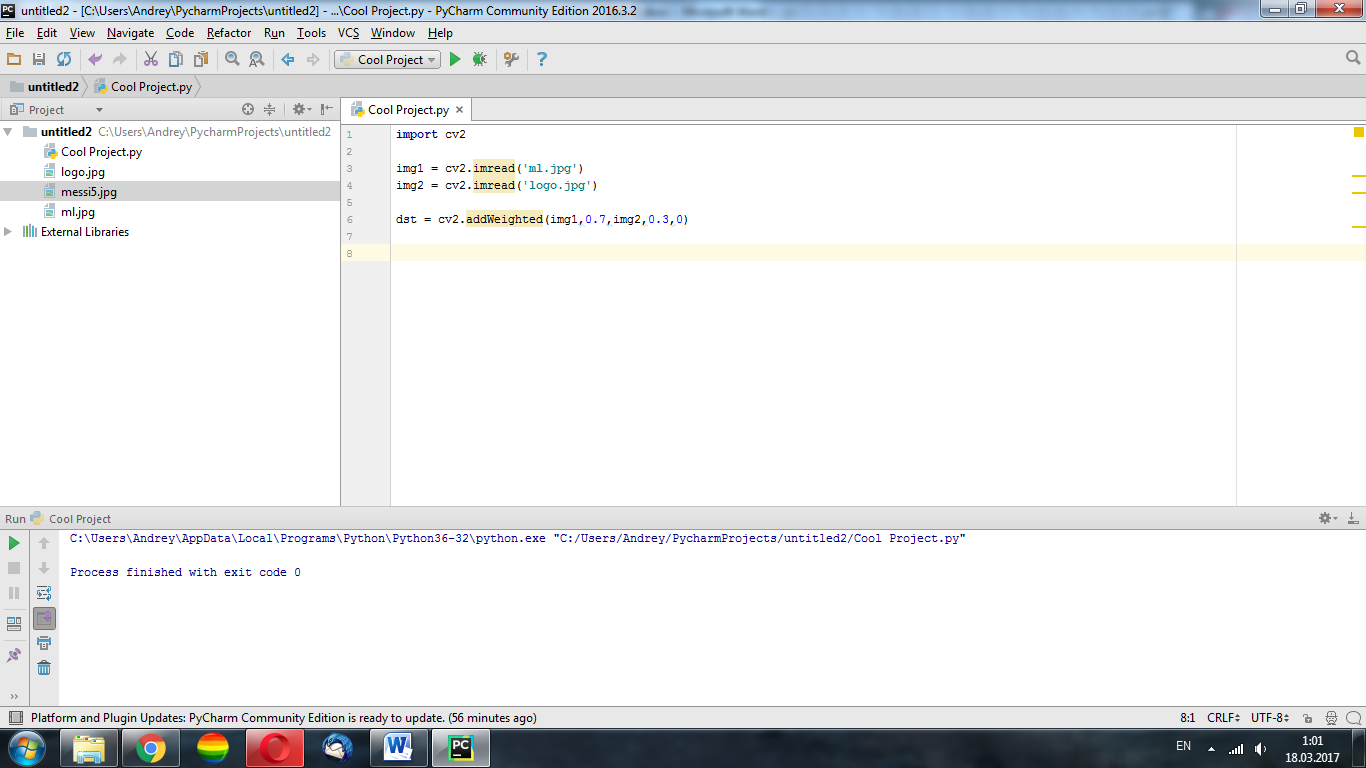
Для начала нам надо подключить библиотеку cv2 и наши изображения

**import** cv2  
  
img1 = cv2.imread(**'ml.jpg'**)  
img2 = cv2.imread(**'logo.jpg'**)



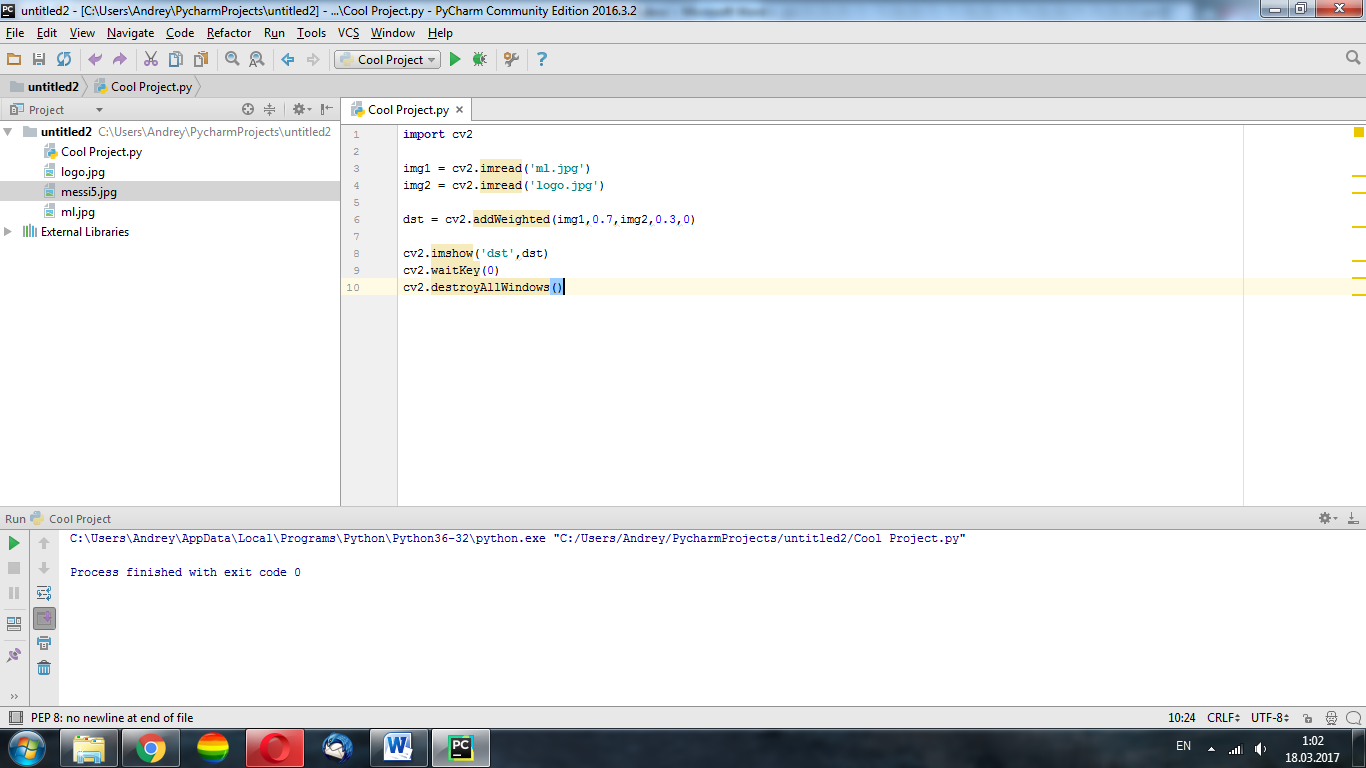
Изображения должны представлять одно целое, из первого мы возьмем 0.7, а из второго 0.3

dst = cv2.addWeighted(img1,0.7,img2,0.3,0)

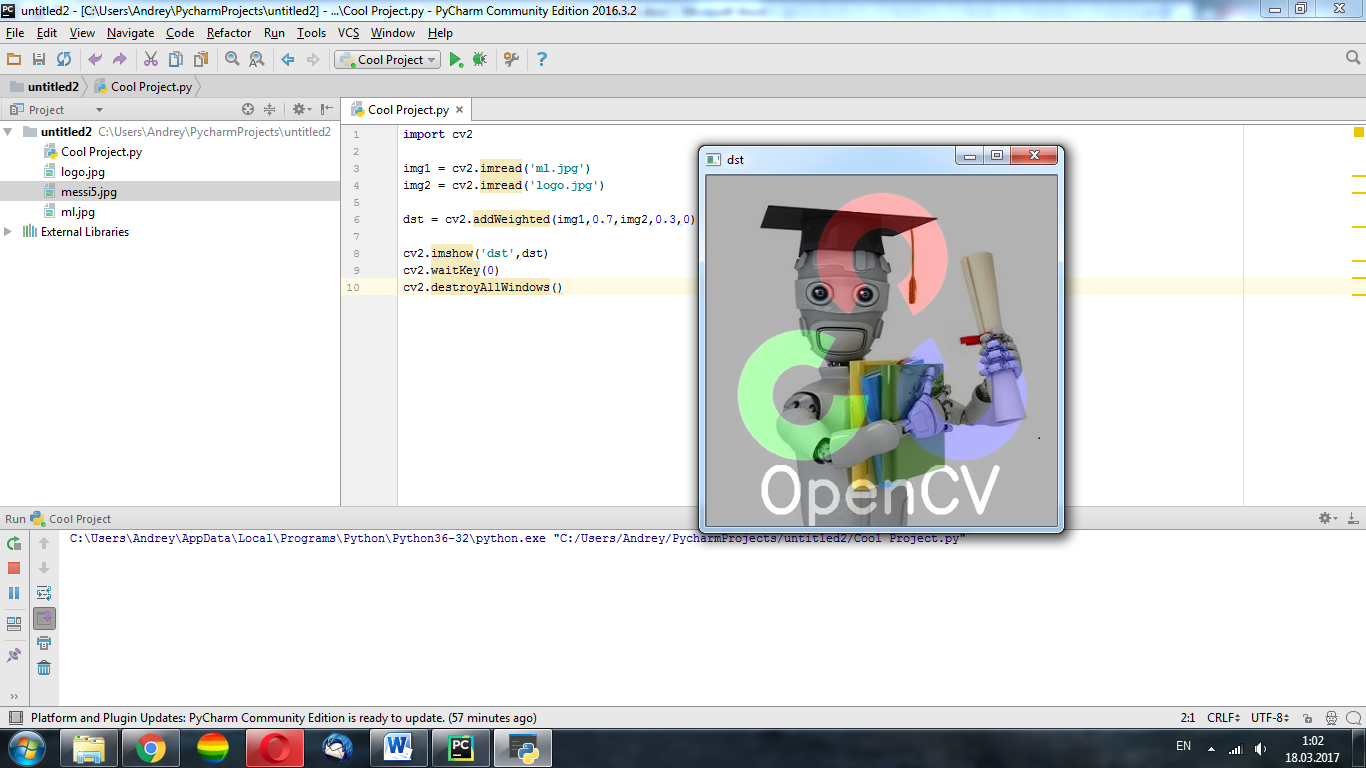


Выведем полученное изображение, поставим закрытие окна на любую клавишу.

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



Результат



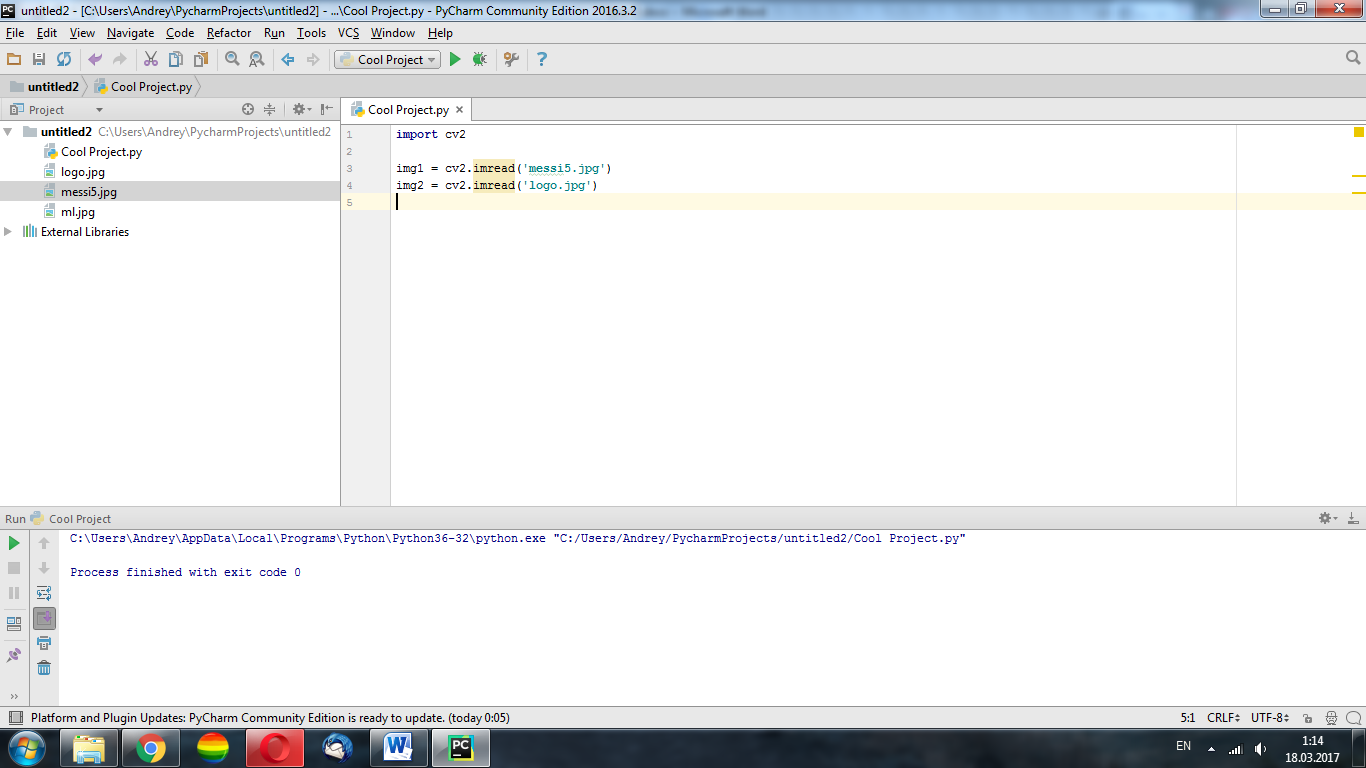
Побитовые операции

Я хочу разместить logo в изображении messi5. Если я просто добавлю его, то это изменит цвет. Если я это смешаю, я получу прозрачный эффект. Но я хочу, чтобы он был непрозрачным.

Можно переместить изображение с logo на messi5 с помощью побитовых операций.

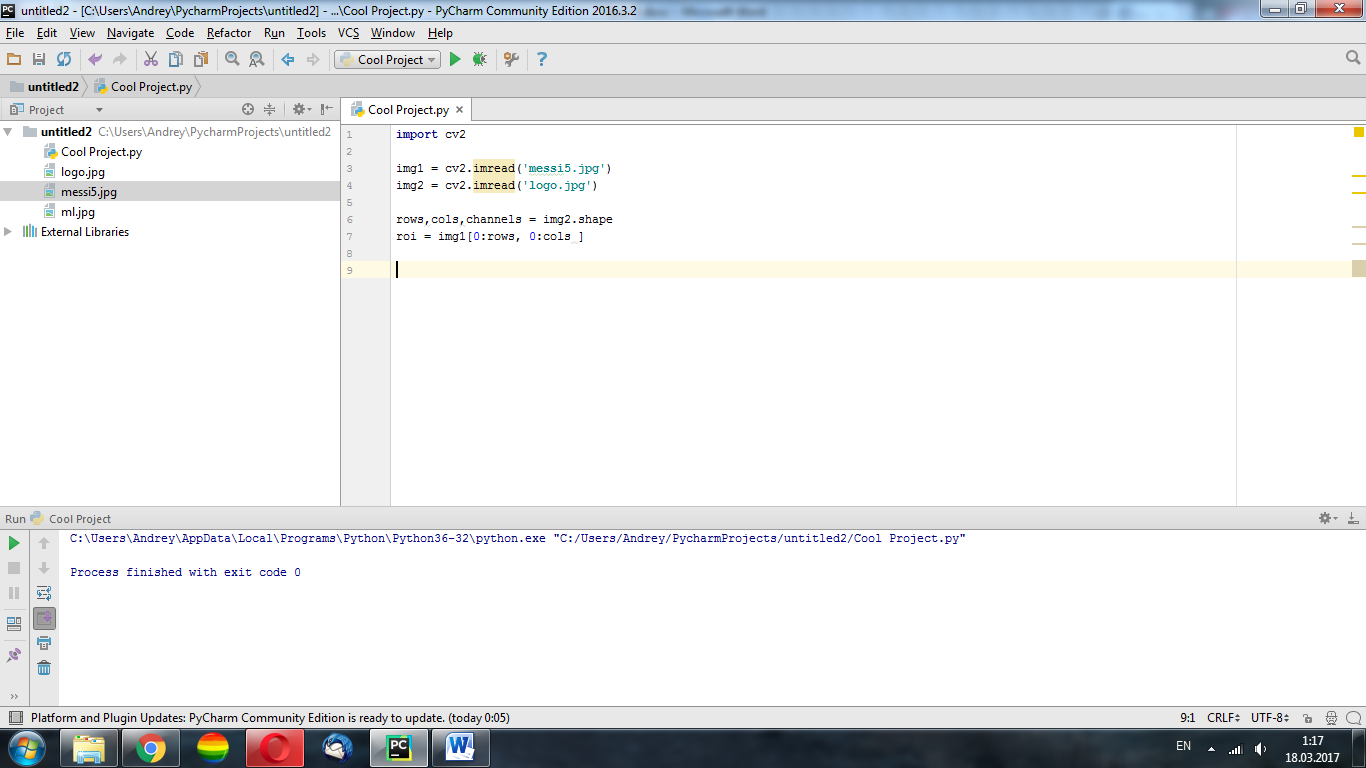
Подключим библиотеку cv2 и необходимые изображения

**import** cv2  
  
img1 = cv2.imread(**'messi5.jpg'**)  
img2 = cv2.imread(**'logo.jpg'**)



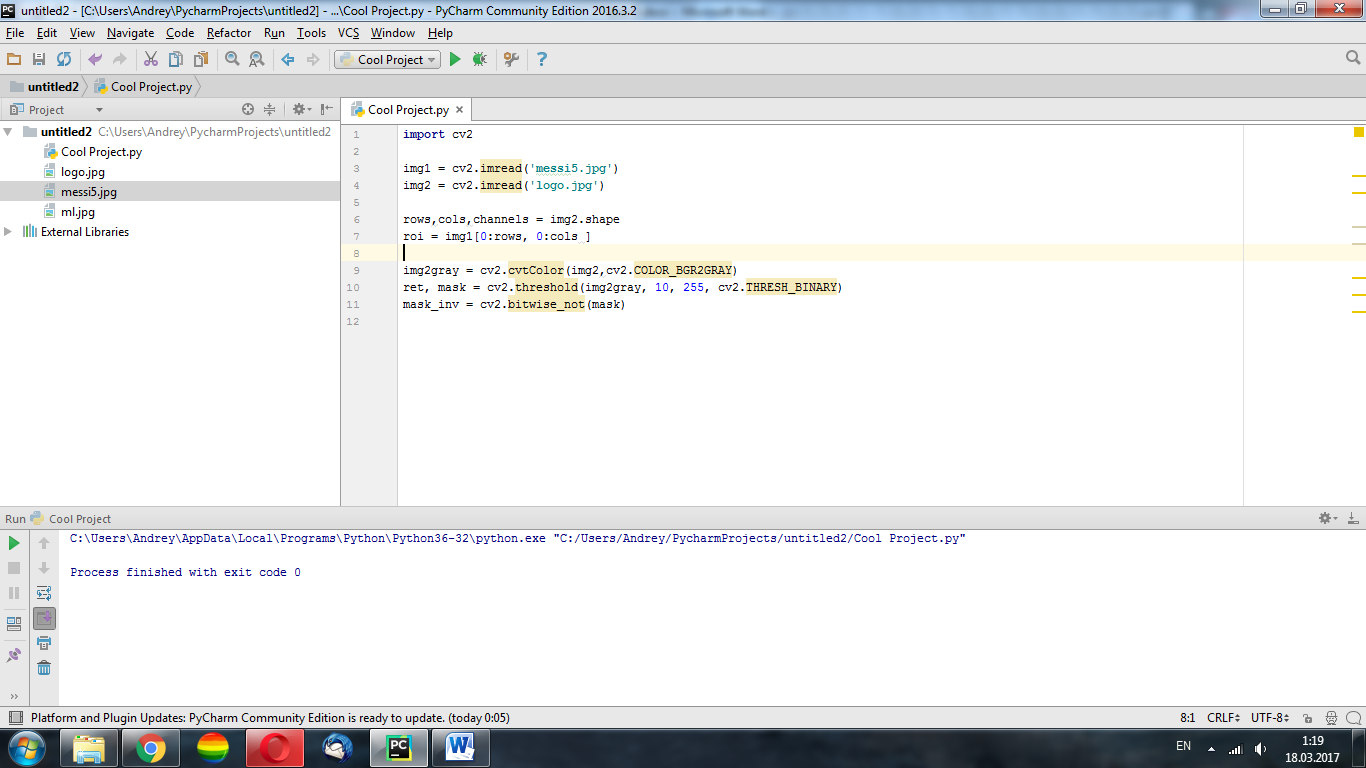
Разместим логотип с левой стороны

rows,cols,channels = img2.shape  
roi = img1[0:rows, 0:cols ]



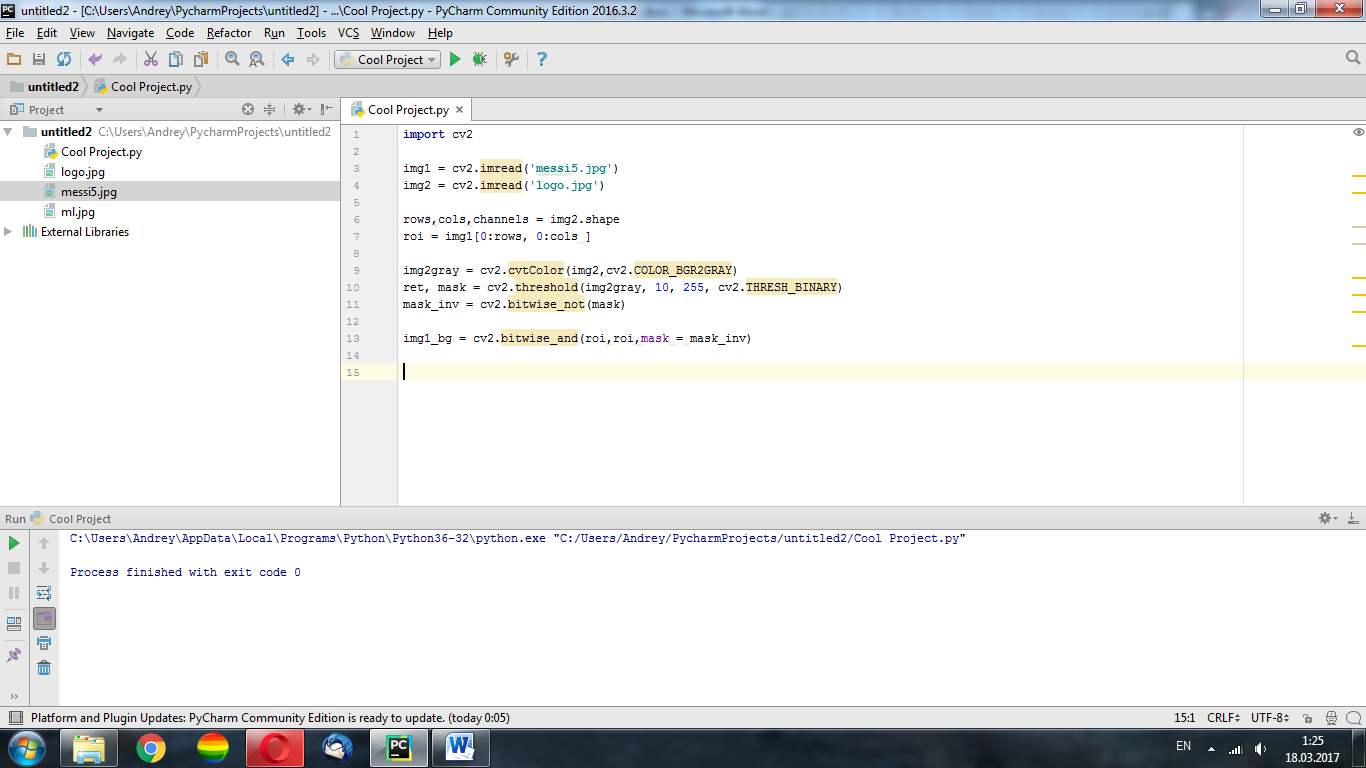
Создадим маску логотипа и его обратную маску

img2gray = cv2.cvtColor(img2,cv2.COLOR\_BGR2GRAY)  
ret, mask = cv2.threshold(img2gray, 10, 255, cv2.THRESH\_BINARY)  
mask\_inv = cv2.bitwise\_not(mask)



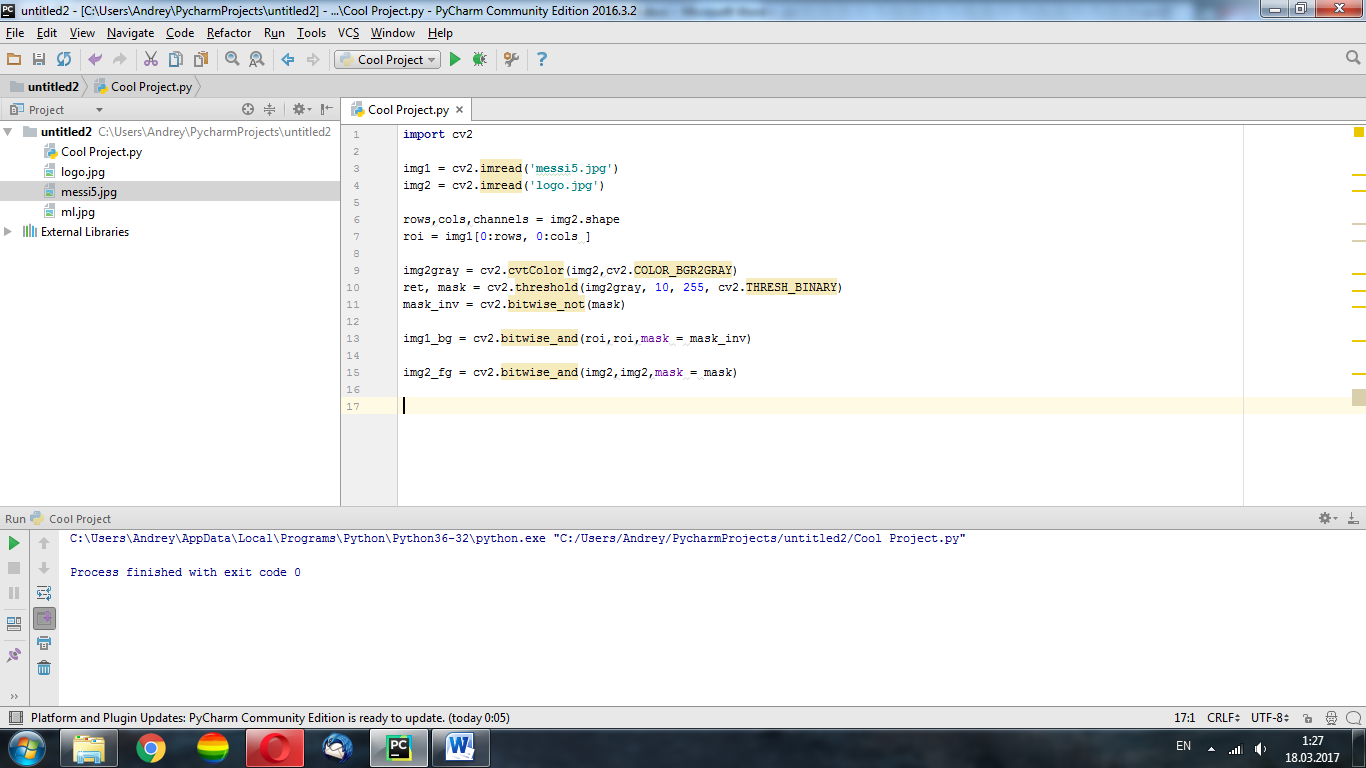
Мы можем отключить черную область логотипа

img1\_bg = cv2.bitwise\_and(roi,roi,mask = mask\_inv)



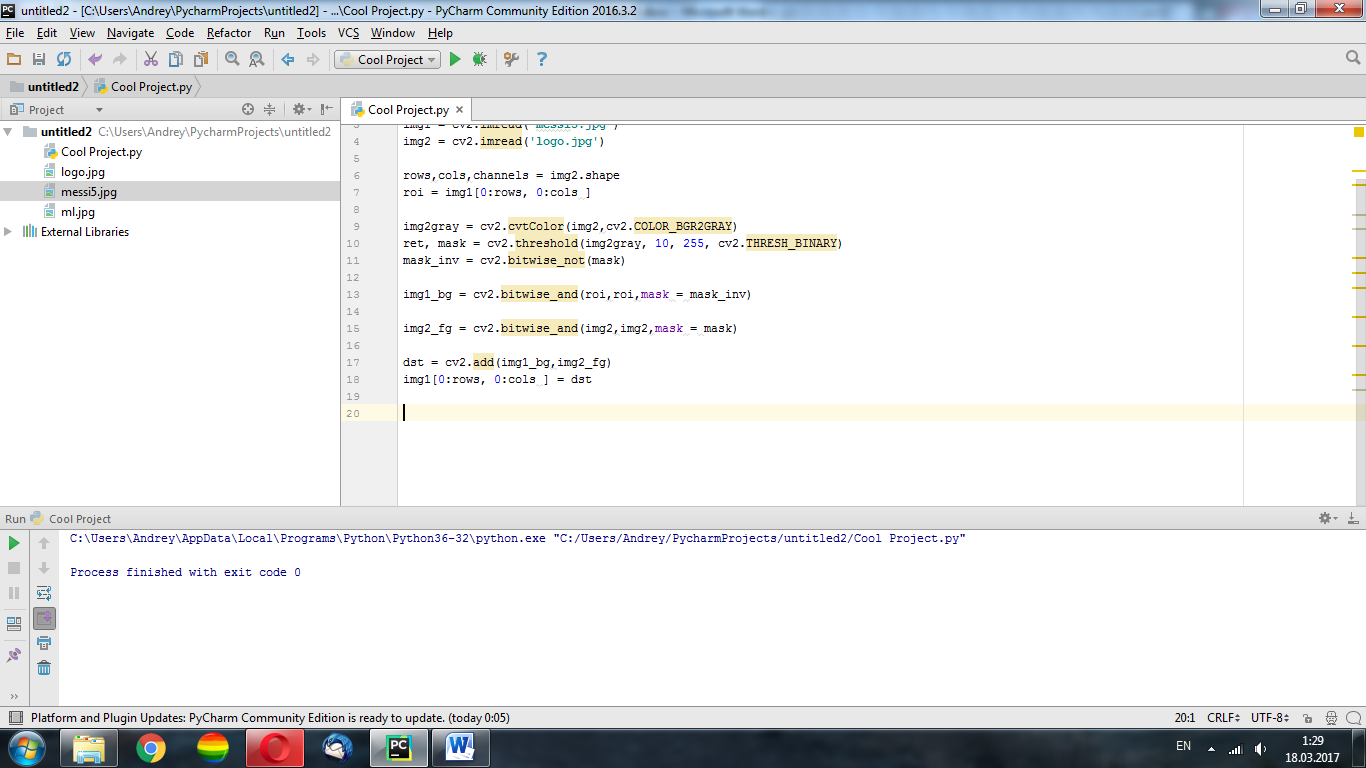
И взять только области с изображением логотипа

img2\_fg = cv2.bitwise\_and(img2,img2,mask = mask)



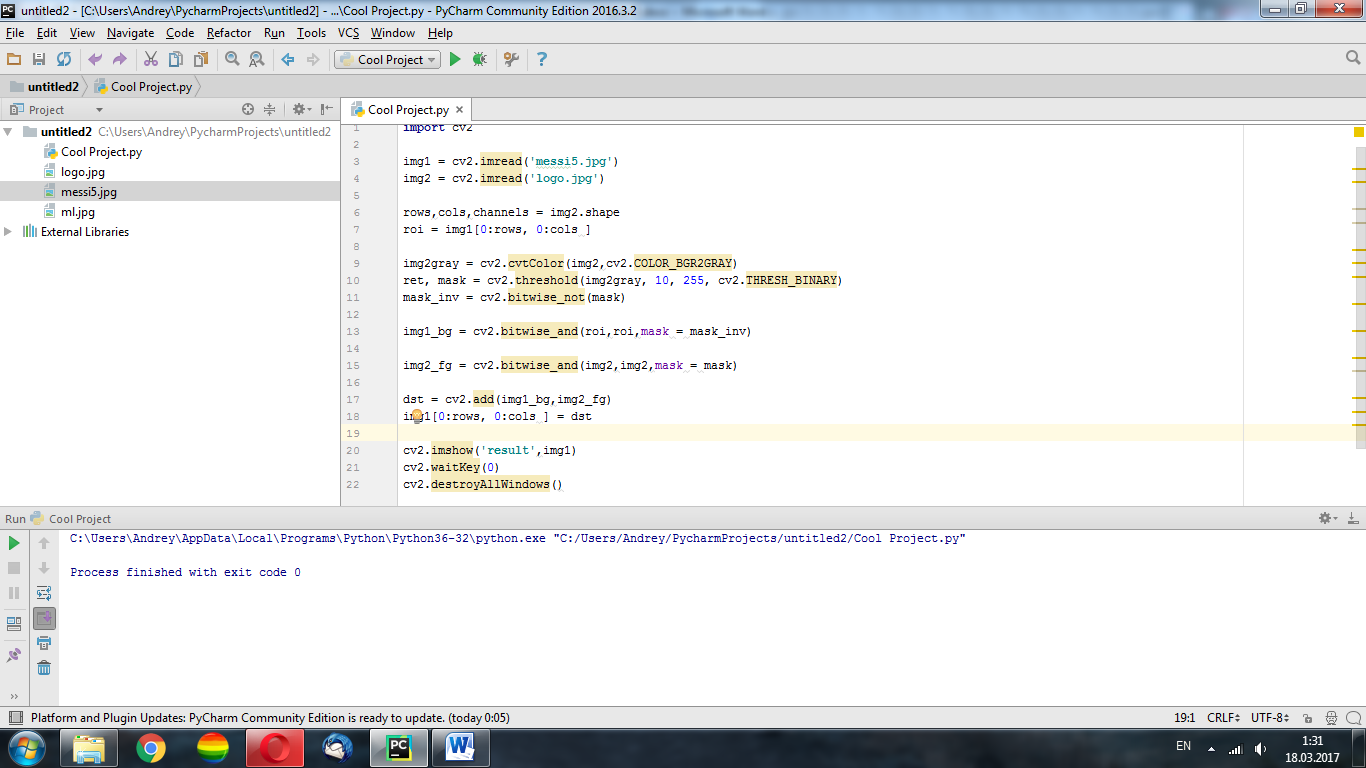
И сложим полученные результаты

dst = cv2.add(img1\_bg,img2\_fg)  
img1[0:rows, 0:cols ] = dst

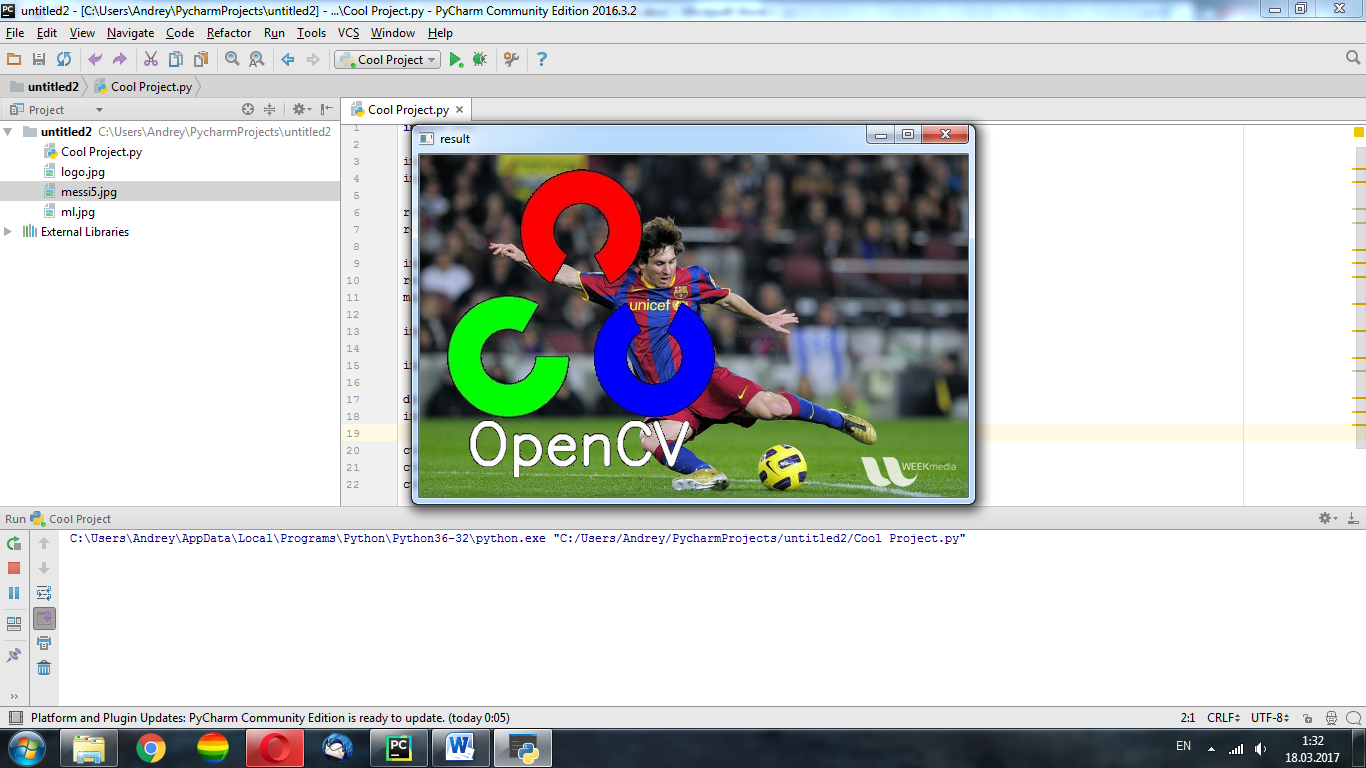


Выведем полученный результат и поставим закрытие окна по нажатию любой клавиши на клавиатуре

cv2.imshow(**'result'**,img1)  
cv2.waitKey(0)  
cv2.destroyAllWindows()



Получился довольно аккуратный результат



Задание: Создайте слайд-шоу из изображений с плавным переходом между ними с помощью функции cv2.addWeighted