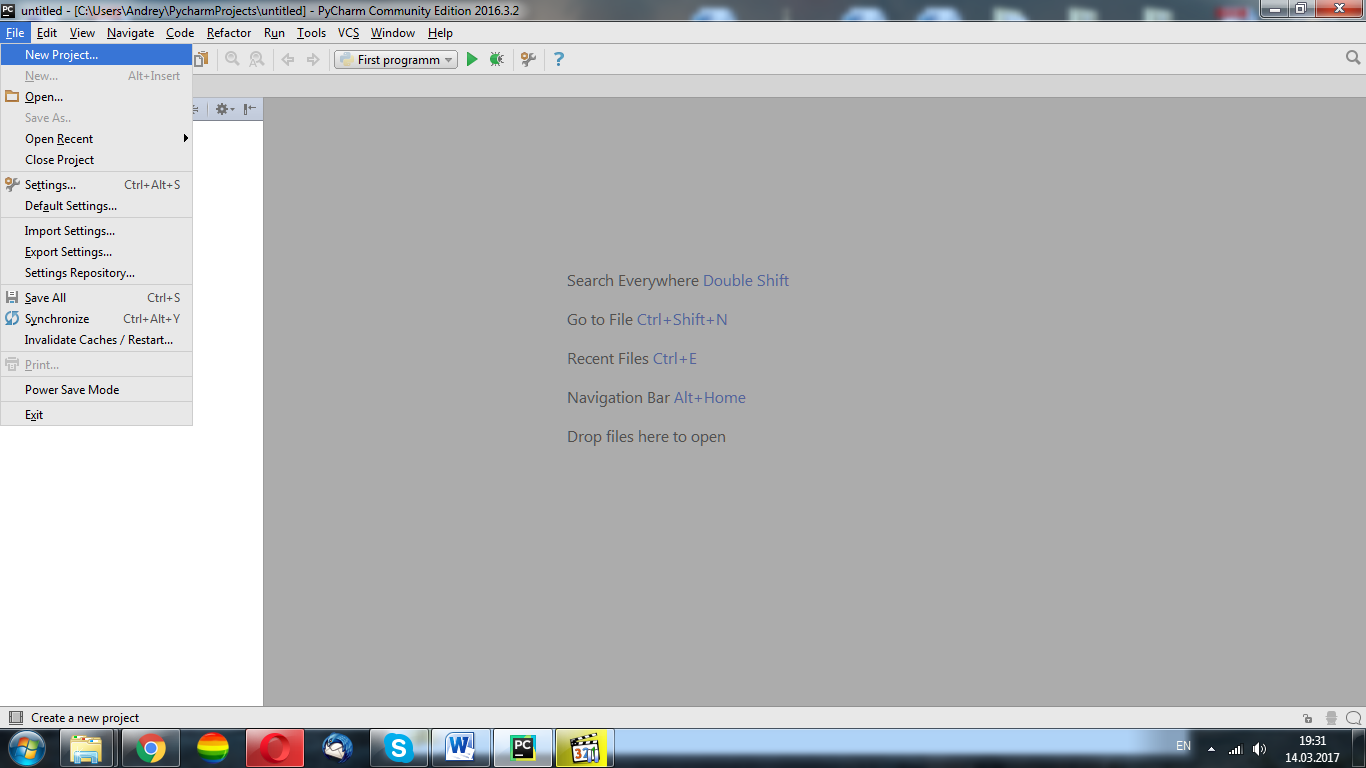
Урок № 2!

Сегодня мы научимся рисовать различные фигуры и рисовать с помощью мышки.

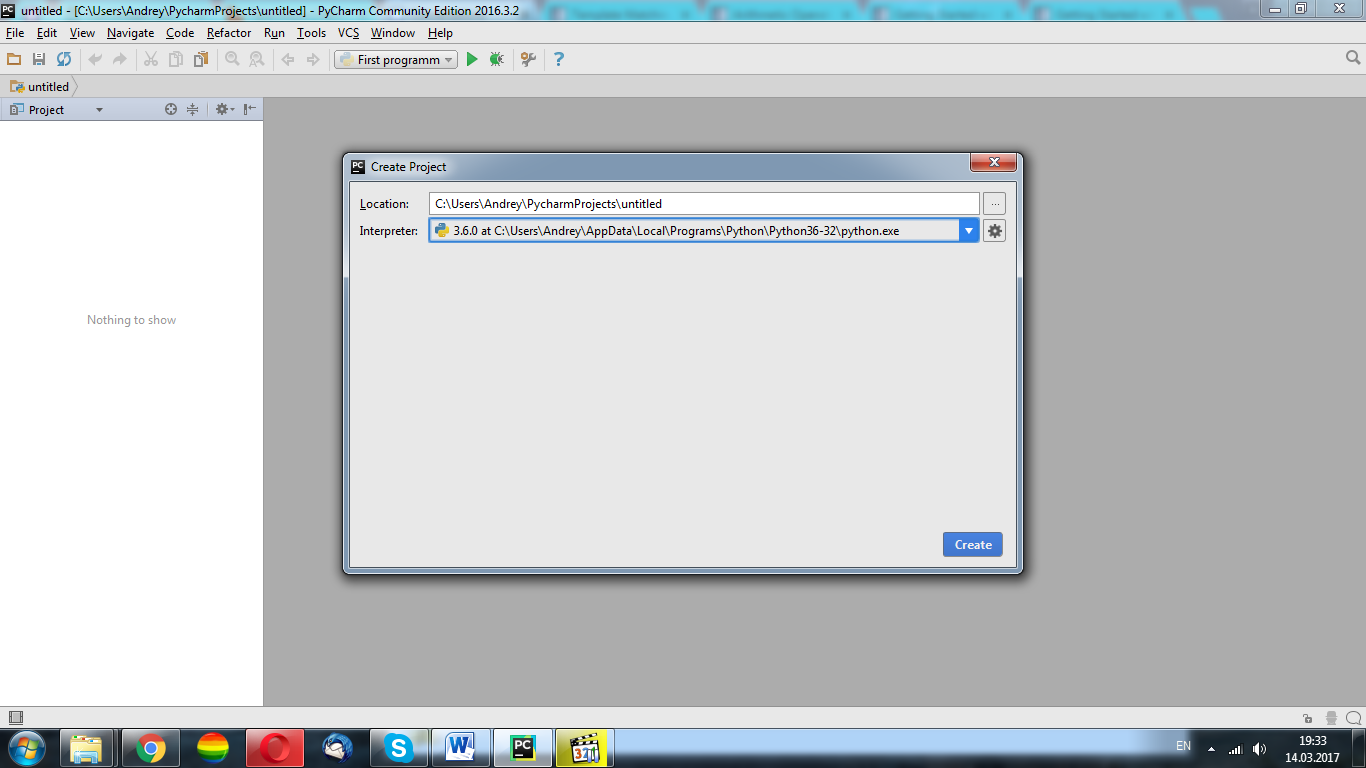
Запустите PyCharm Community Edition

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.

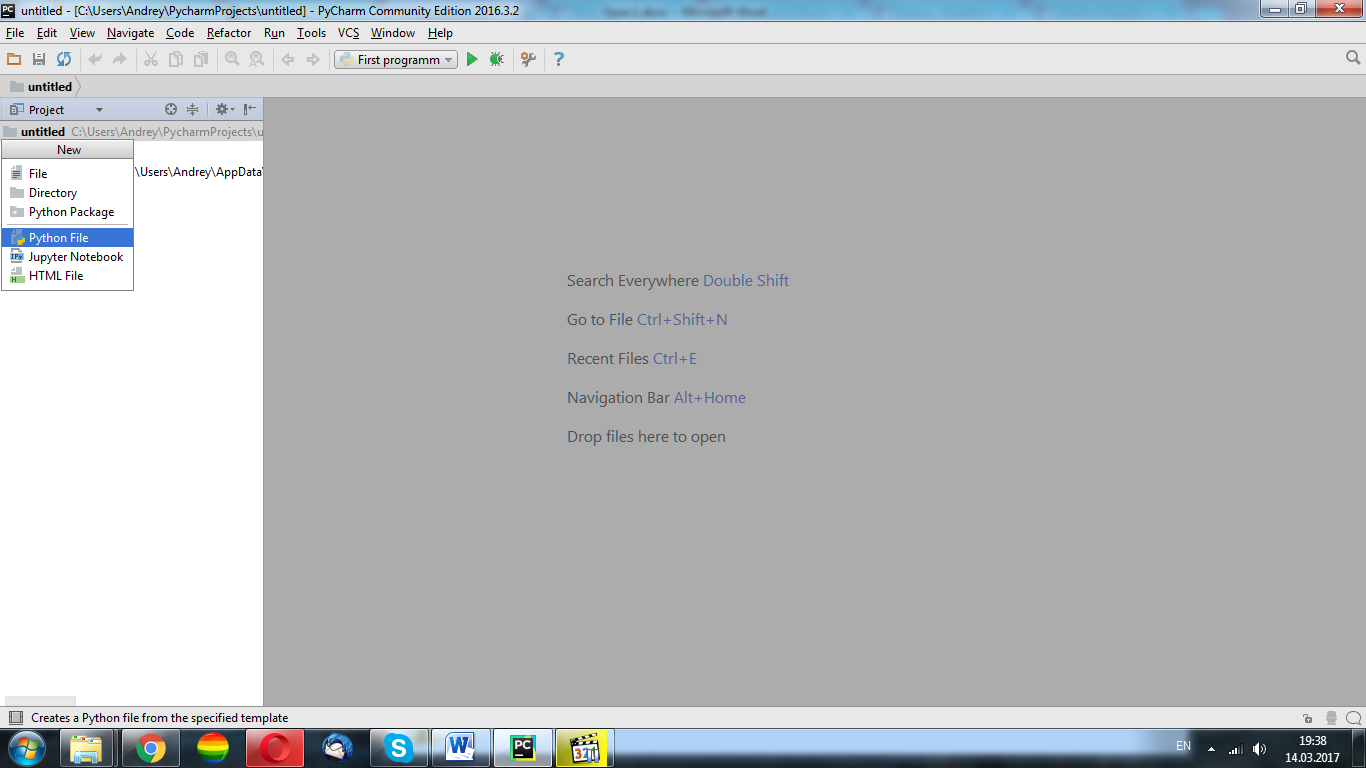
Во второй строке версия python. Выберите последнюю версию 3.6. Нажмите Create



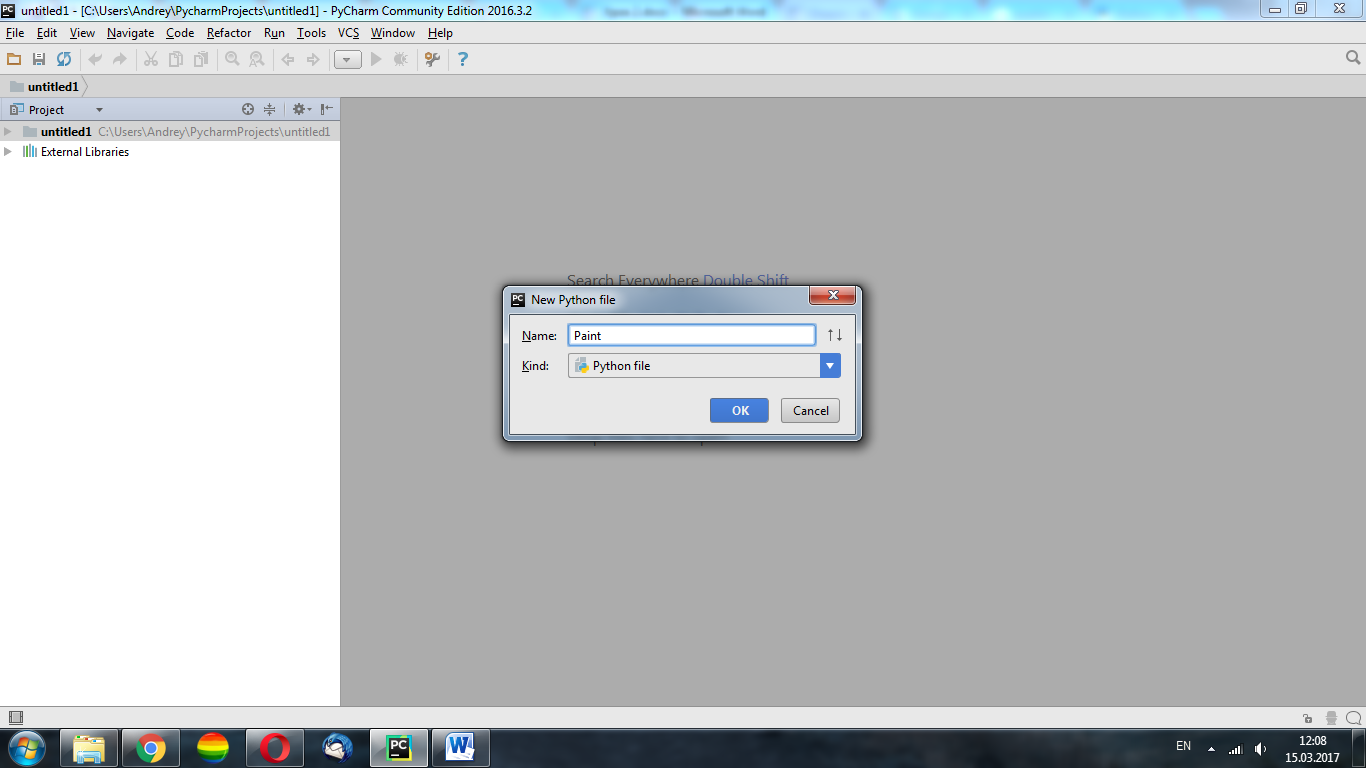
Далее создадим файл, в котором будем работать. Во вкладке File нажмите New…



Python file



Назовем этот файл Paint



Рисовать геометрические фигуры мы будем с помощью различных функций: cv2.line (), cv2.circle (), cv2.rectangle (), cv2.ellipse (), cv2.putText () и т.д.

Во всех перечисленных выше функциях вы увидите некоторые общие аргументы, приведенные ниже:

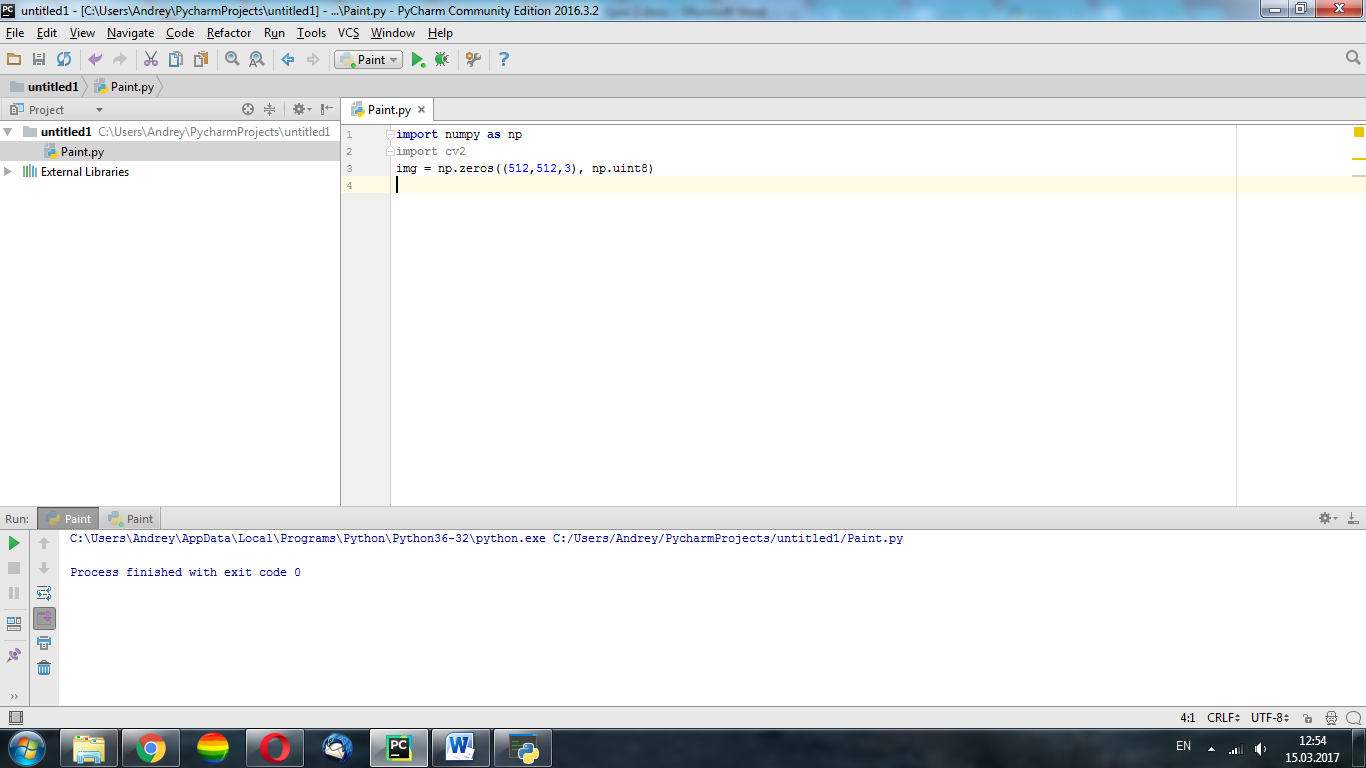
Img: изображение, в котором вы хотите нарисовать фигуры

Color: Цвет фигуры. Записывается в скобках для BGR, например: (255,0,0) для синего, (0,0, 255) для красного.

Толщина: толщина линии или круга и т.д. Если -1 передается для закрытых фигур, например кругов, он заполняет форму. Default thickness = 1

Создадим окно размером 512 на 512 и зальем его черным цветом

**import** numpy **as** np  
**import** cv2  
img = np.zeros((512,512,3), np.uint8)



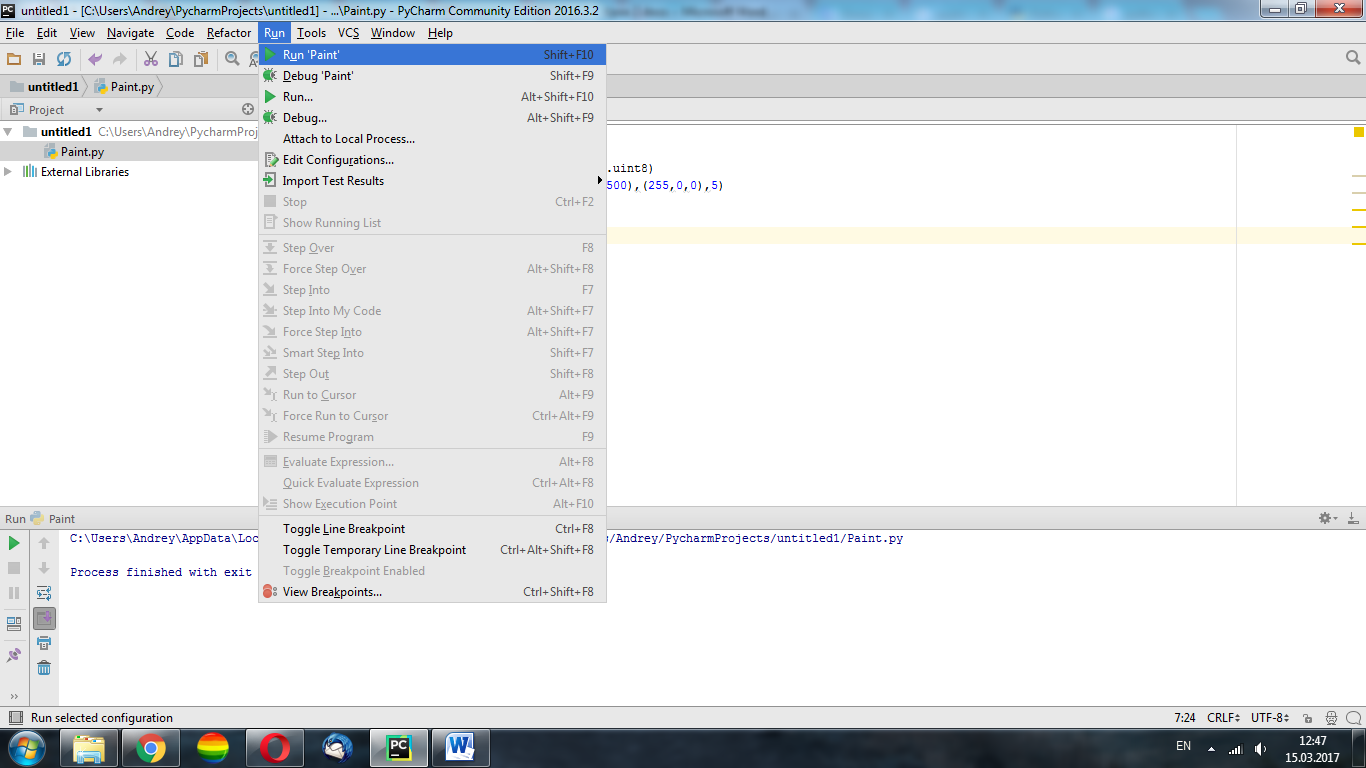
Нарисуем линию от левого верхнего до правого нижнего угла. Синего цвета и толщиной 5 пикселей

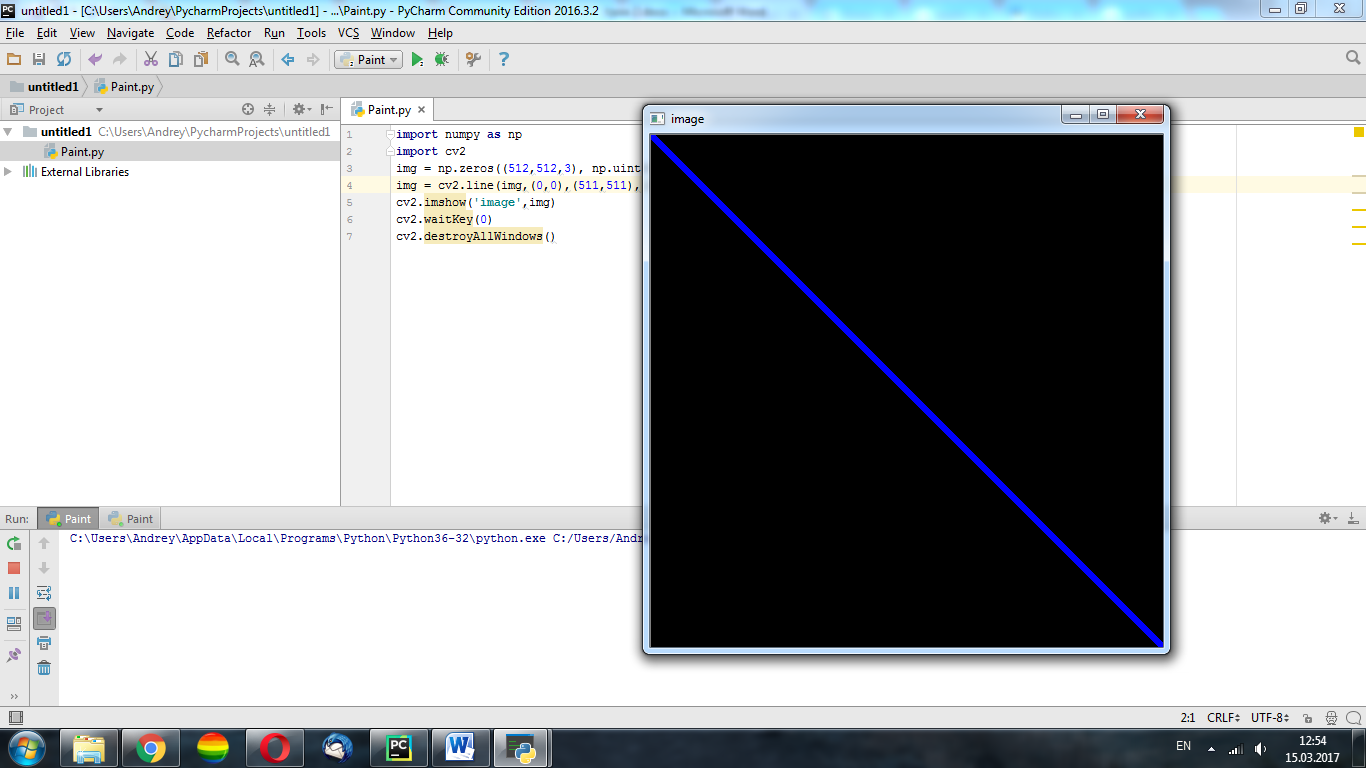
img = cv2.line(img,(0,0),(511,511),(255,0,0),5)

Выведем окно для просмотра (назовем его Image) и поставим закрытие этого окна по нажатию любой клавиши

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

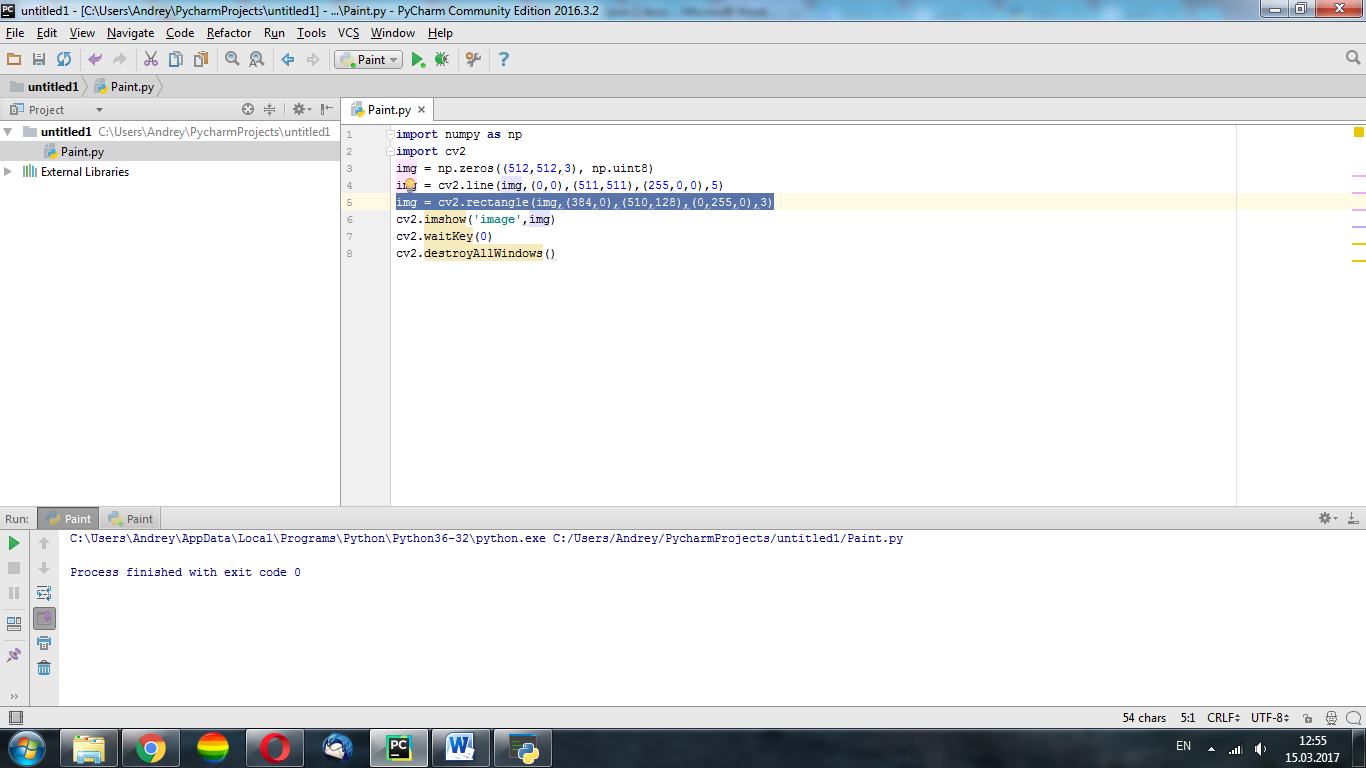
Запустите программу

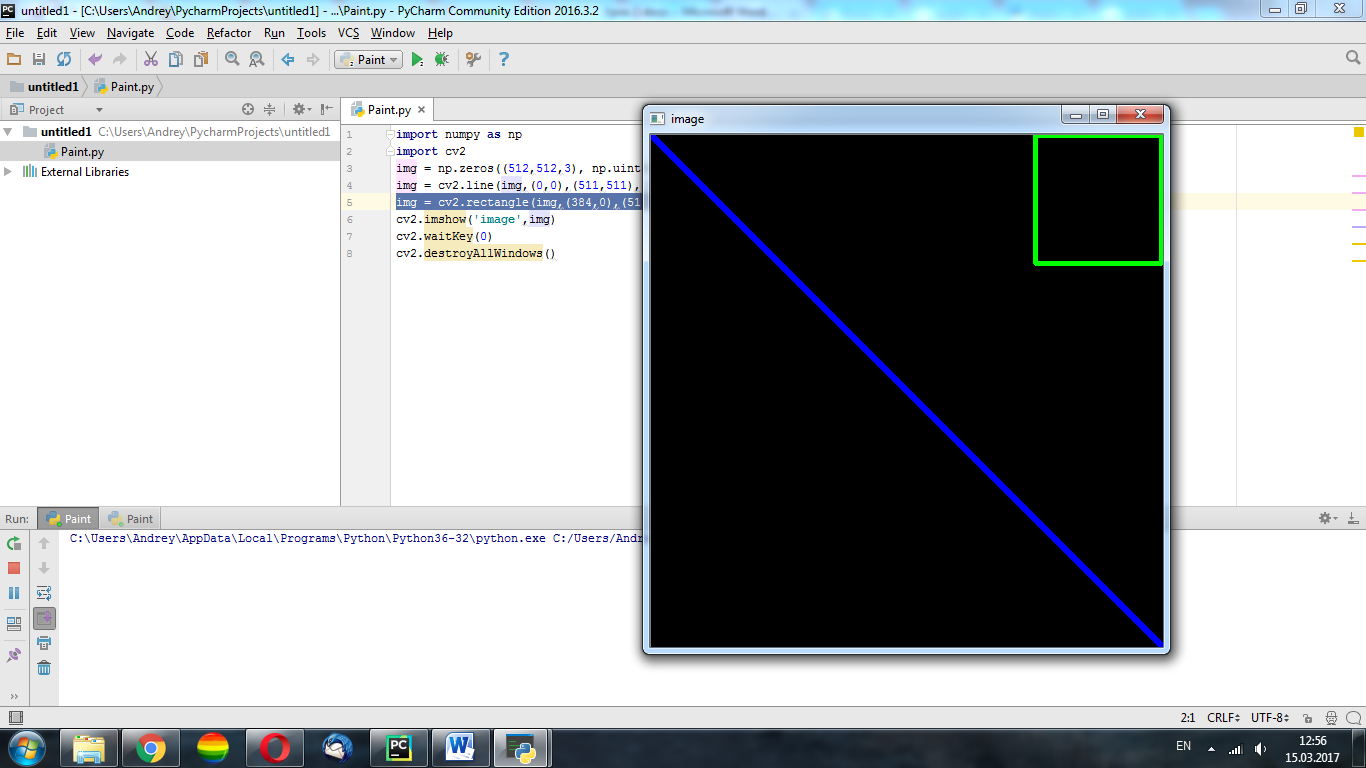




Чтобы нарисовать прямоугольник, нам нужен верхний левый угол и нижний правый угол прямоугольника. На этот раз мы нарисуем зеленый прямоугольник с толщиной 3рх в верхнем правом углу изображения.

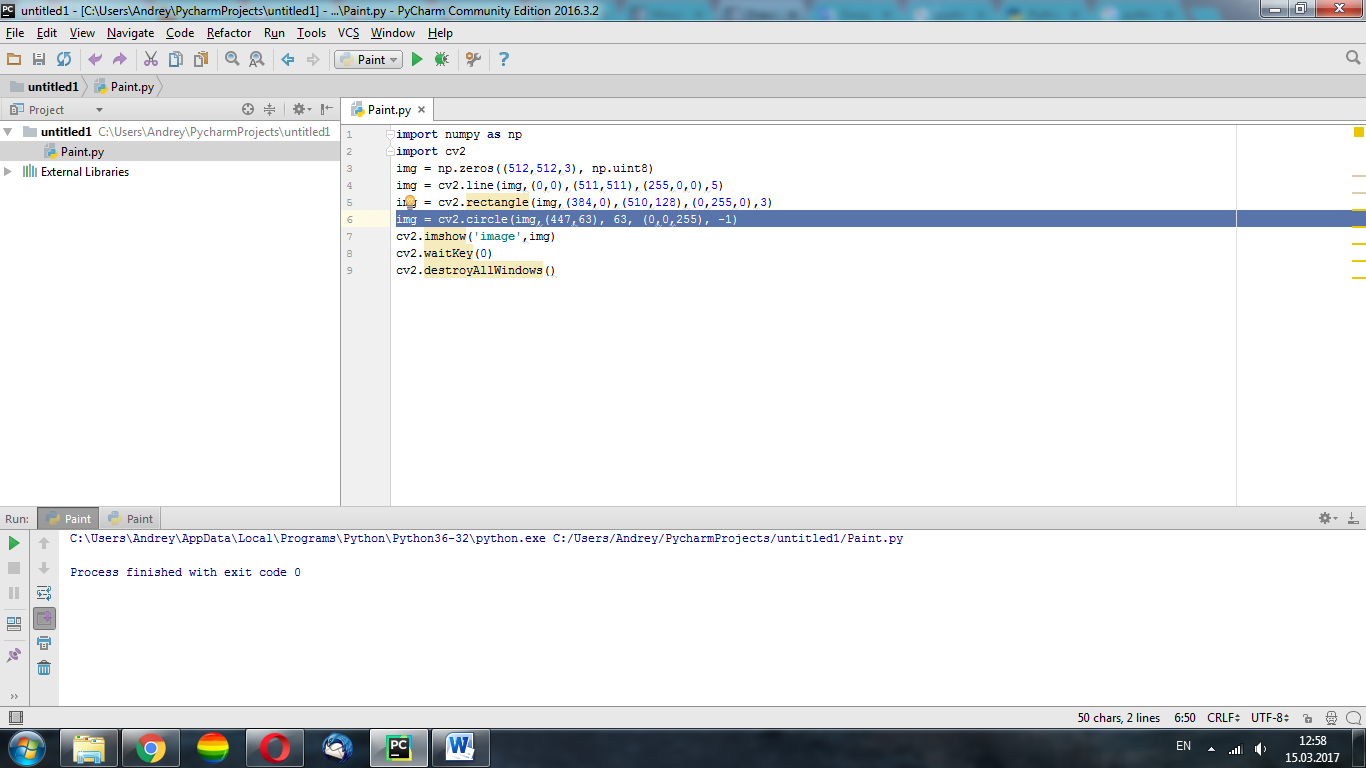
img **=** cv2**.**rectangle(img,(384,0),(510,128),(0,255,0),3)

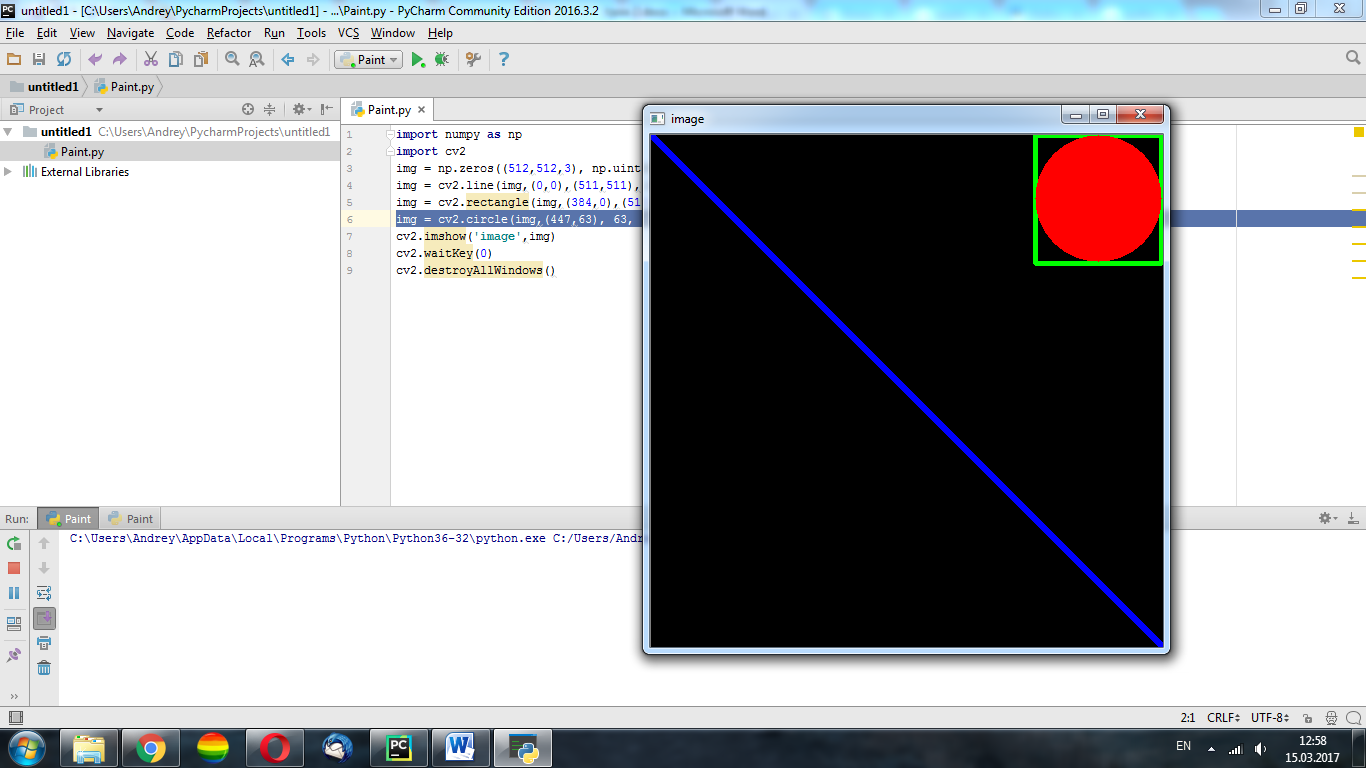




Чтобы нарисовать круг, нам нужны его координаты центра и радиус. Мы нарисуем круг внутри прямоугольника, нарисованного выше. Зальем его красным цветом (0,0,255) -1

img **=** cv2**.**circle(img,(447,63), 63, (0,0,255), **-**1)

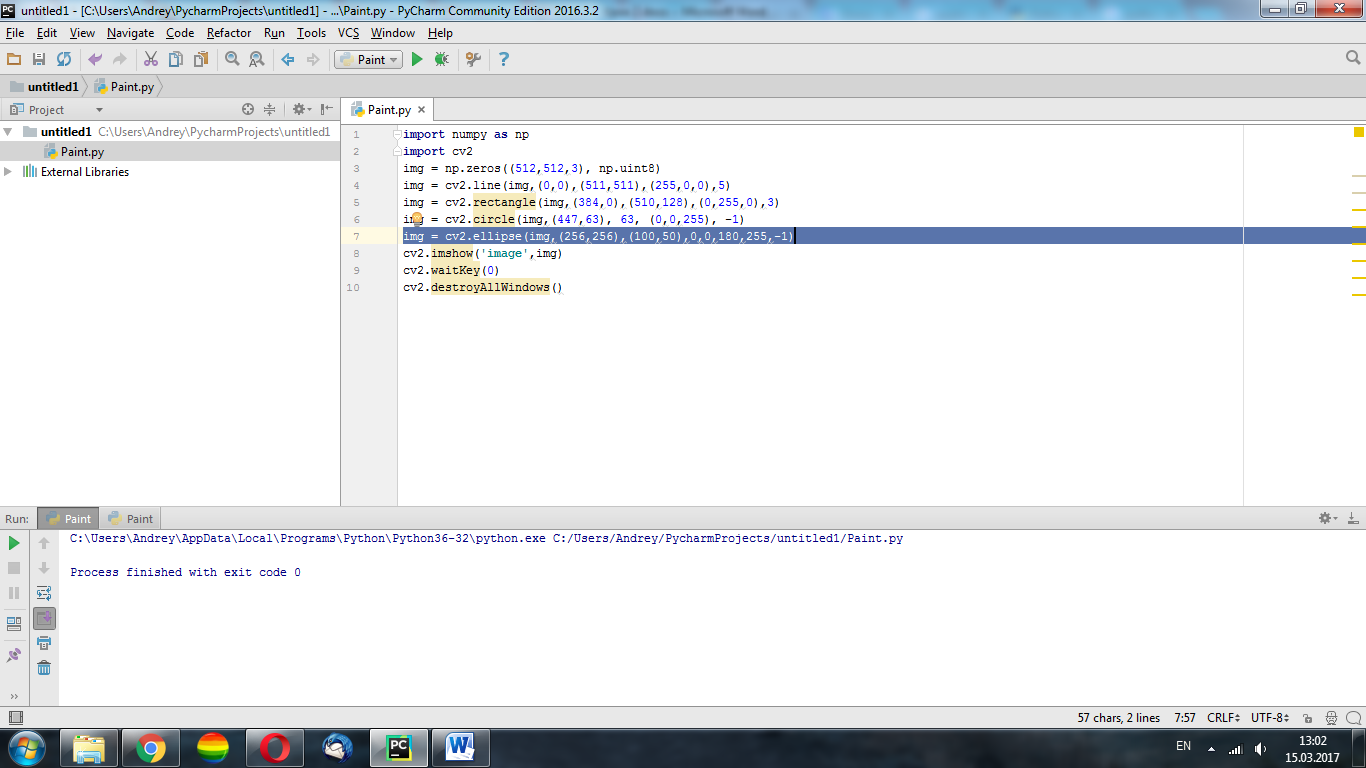


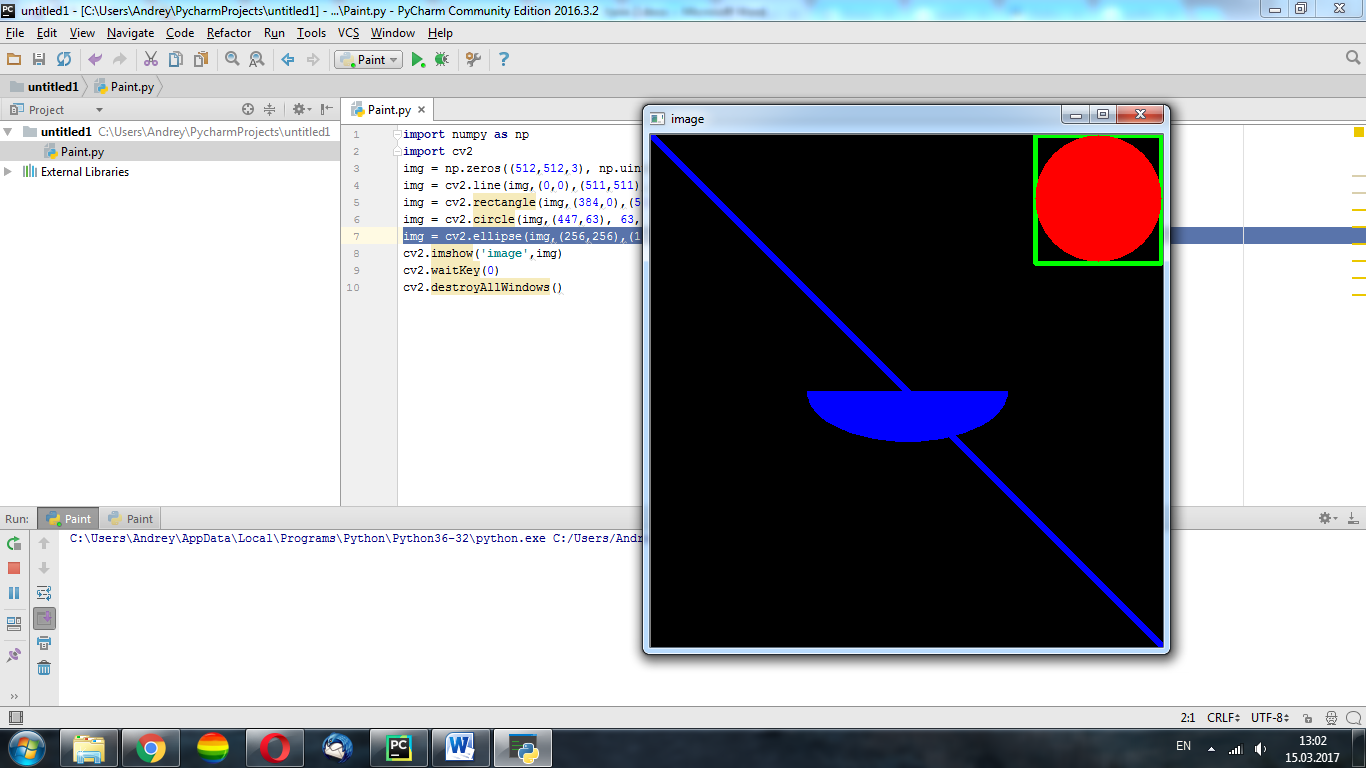


Чтобы нарисовать эллипс, нам необходимо передать несколько аргументов. Одним из аргументов является расположение центра (x, y). Следующий аргумент - длина осей (длина большой оси, малая длина оси). Angle - угол поворота эллипса в направлении против часовой стрелки. StartAngle и endAngle обозначает начало и конец дуги эллипса, измеренной по часовой стрелке от главной оси. Т.е. значения 0 и 360 дают полный эллипс.

По середине нашего рисунка, нарисуем элипс длиной 100рх и высотой 50 рх с дугой 180 градусов и зальем его синим цветом

img **=** cv2**.**ellipse(img,(256,256),(100,50),0,0,180,255,**-**1)



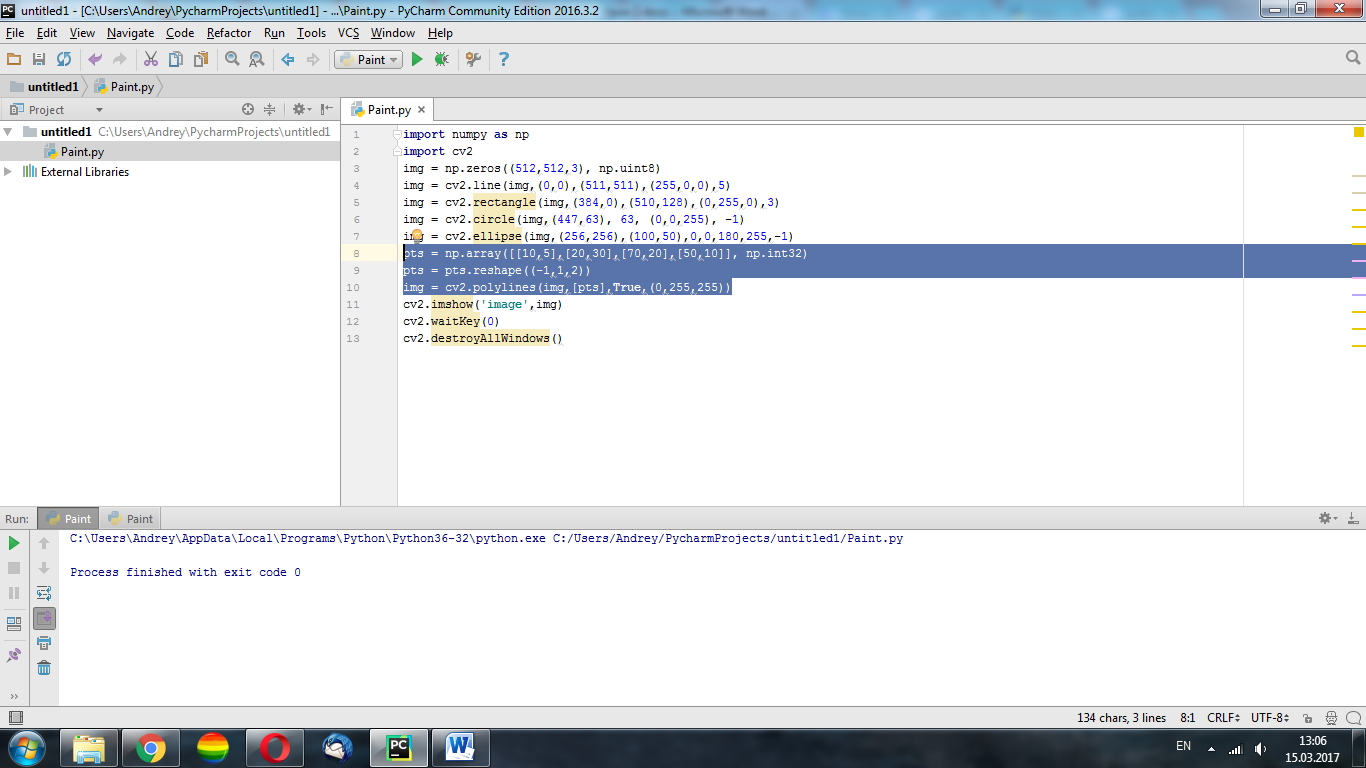


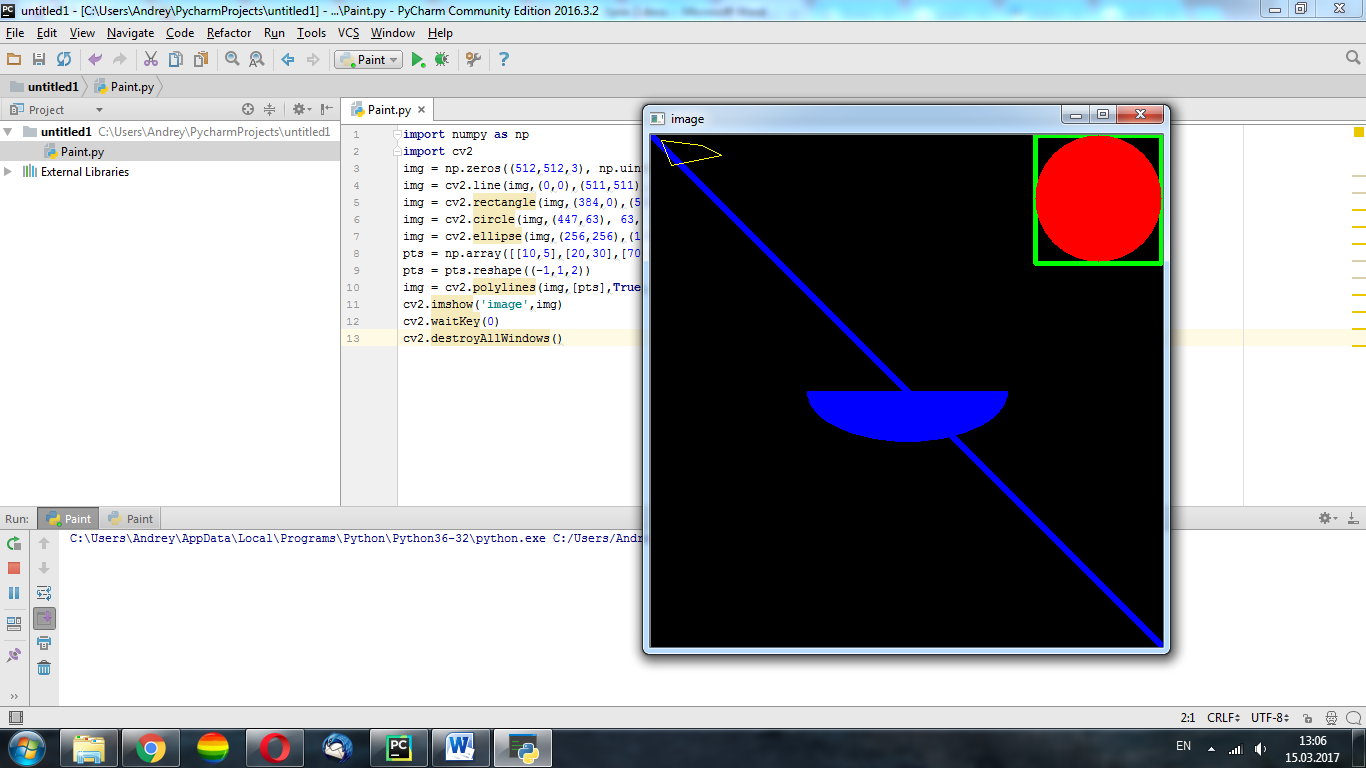
Чтобы нарисовать многоугольник, сначала вам нужны координаты вершин. Нам необходимо будет указать координаты всех вершин, и оно должно быть типа int32. Здесь мы рисуем маленький многоугольник с четырьмя вершинами желтого цвета.

pts **=** np**.**array([[10,5],[20,30],[70,20],[50,10]], np**.**int32)

pts **=** pts**.**reshape((**-**1,1,2))

img **=** cv2**.**polylines(img,[pts],True,(0,255,255))





Добавление текста в изображения:

Чтобы поместить текст в изображения, нам необходимо указать следующие вещи:

Текстовые данные, которые мы хотите записать

Установить координаты места, где вы хотите поместить его (то есть нижний левый угол, где начинаются данные).

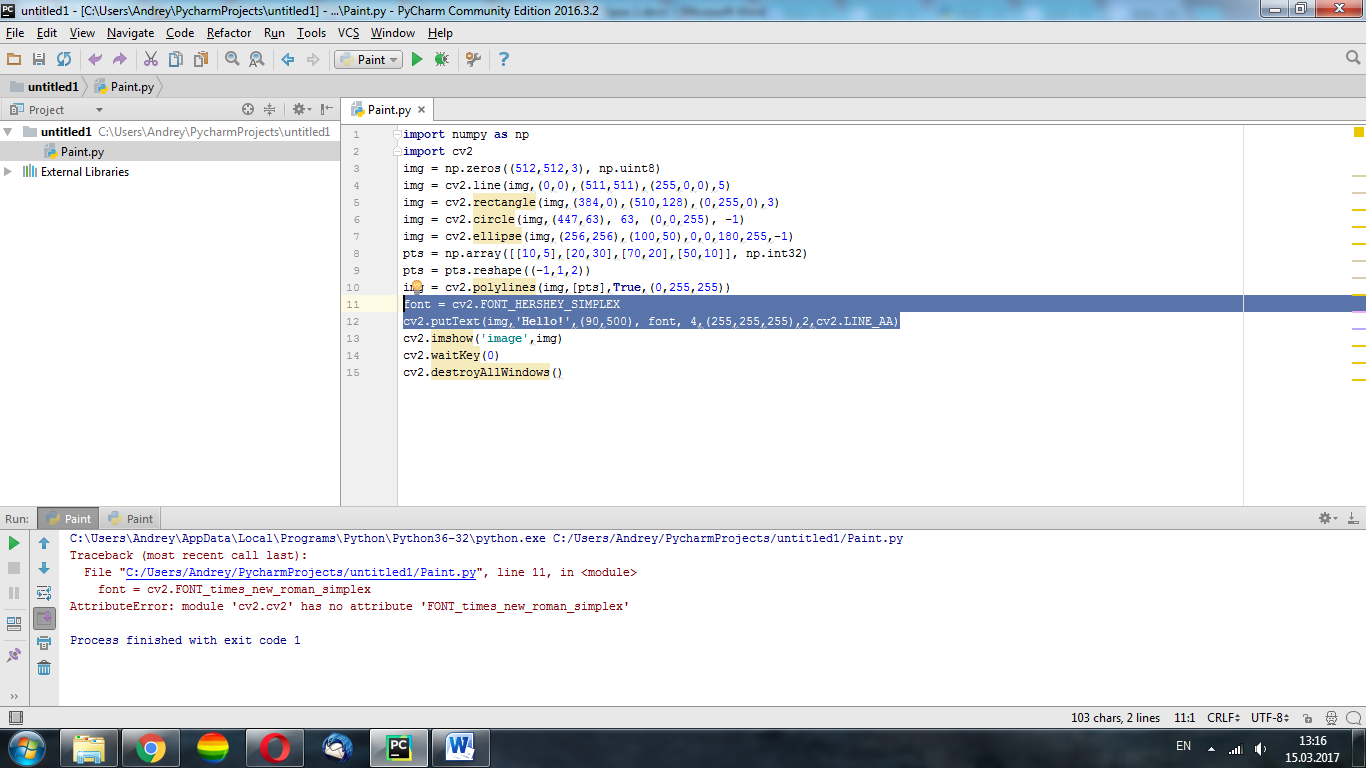
Тип шрифта

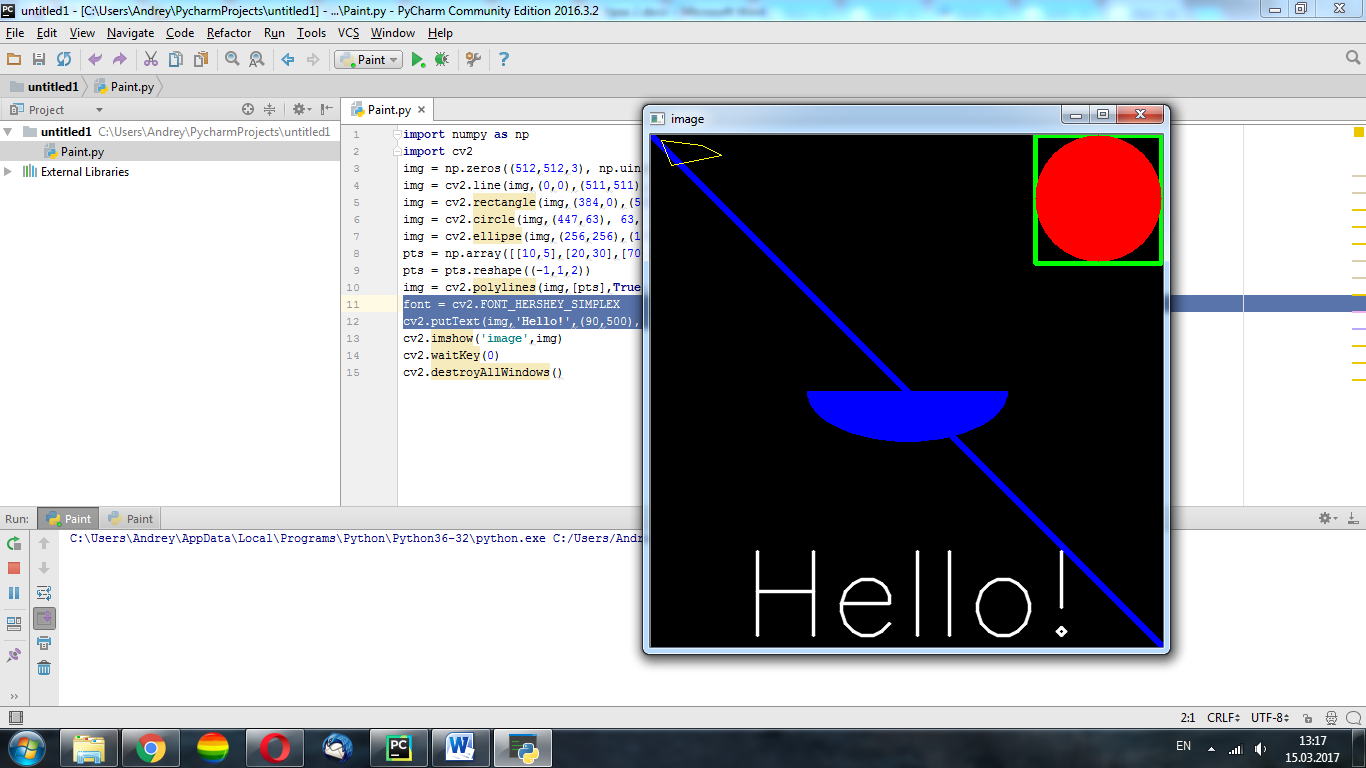
Масштаб шрифта (определяет размер шрифта)

И цвет, толщина, тип линии и т.д. будем использовать lineType = cv2.LINE\_AA.

Напишем слово «Hello!» белым цветом и постам его внизу посередине изображения. Используем шрифт hershey simplex

font = cv2.FONT\_HERSHEY\_SIMPLEX  
cv2.putText(img,**'Hello!'**,(90,500), font, 4,(255,255,255),2,cv2.LINE\_AA)





Задание: Используя полученные знания, попробуйте нарисовать домик или снеговика. В общем – развлекайтесь!

Перейдем к рисованию мышкой.

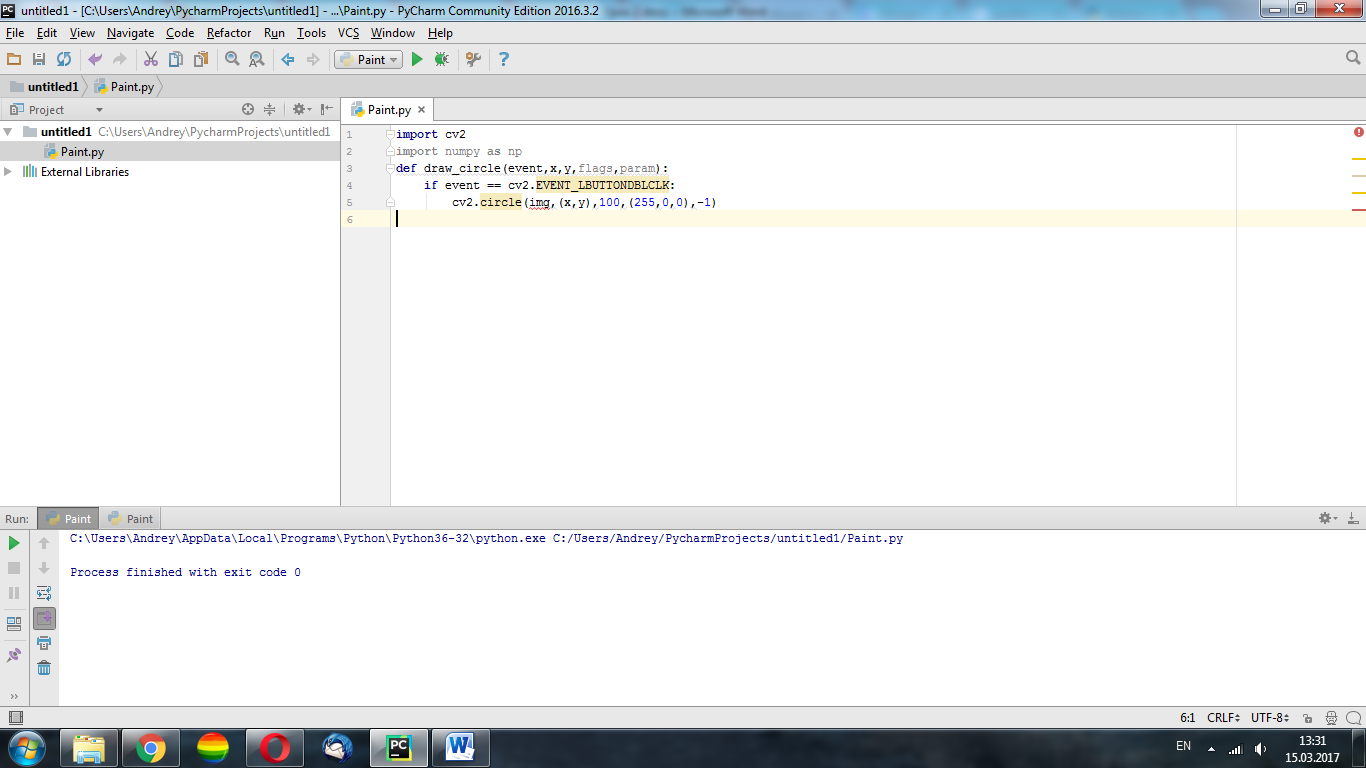
Можете создать новый проект или просто сохранить и удалить код.

Создадим простое приложение, которое рисует круг на изображении по двойному клику левой кнопки мыши.

Сначала мы создаем функцию обратного вызова с мышью, которая выполняется, когда происходит двойной клик. Событие мыши может быть любым, связанным с мышью, как левая кнопка, левая кнопка вверх, левая кнопка двойным щелчком и тд. Она дает нам координаты (x, y) для каждого события мыши. Благодаря этому событию и местоположению мы можем делать все, что захотим.

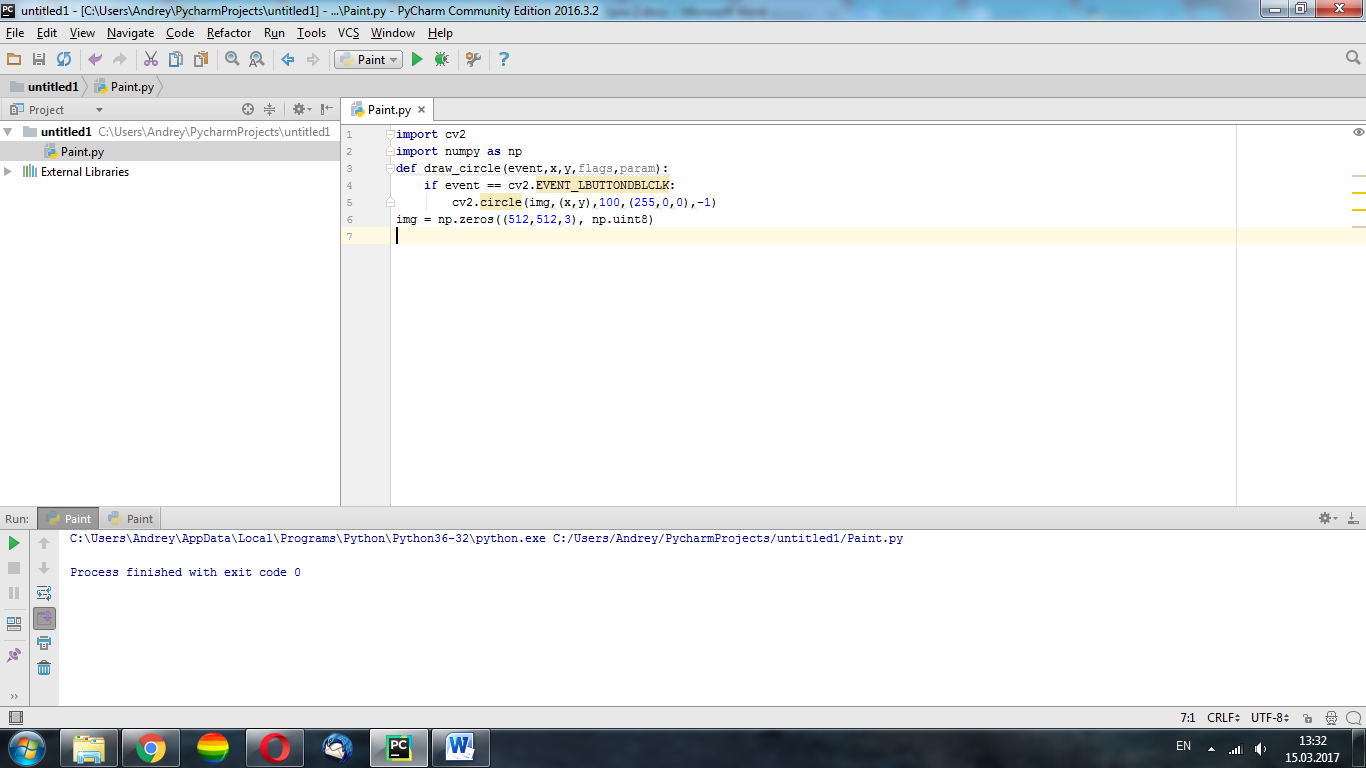
Создаем функцию обратного вызова по двойному клику, запоминаем координаты клика и рисуем в этой точки синий круг в 100пикселей

**import** cv2  
**import** numpy **as** np  
**def** draw\_circle(event,x,y,flags,param):  
 **if** event == cv2.EVENT\_LBUTTONDBLCLK:  
 cv2.circle(img,(x,y),100,(255,0,0),-1)



И конечно же нам надо создать самое окно в котором мы будем рисовать. Черный квадрат размером 512 пикселей

img = np.zeros((512,512,3), np.uint8)

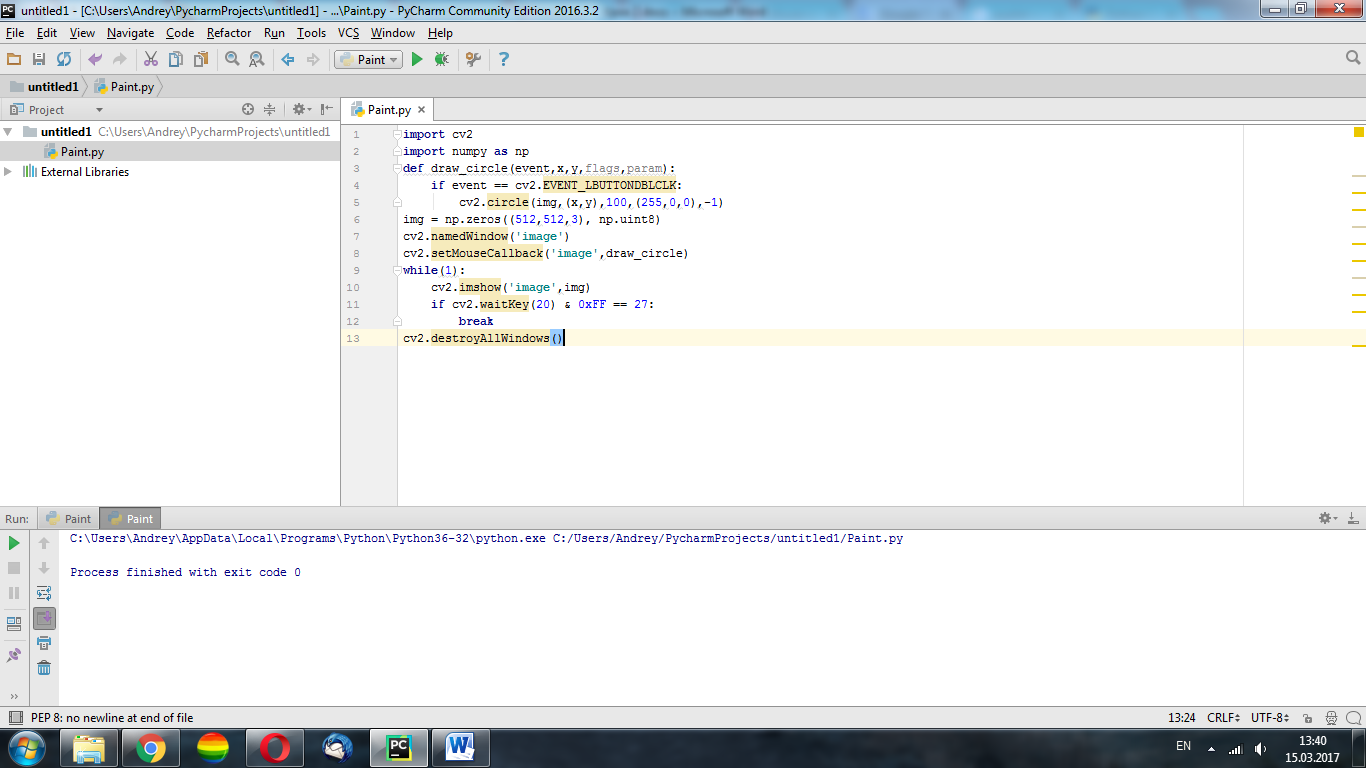


Поставим имя для окна программы, вызовем функцию setMouseCallback для этого окна.

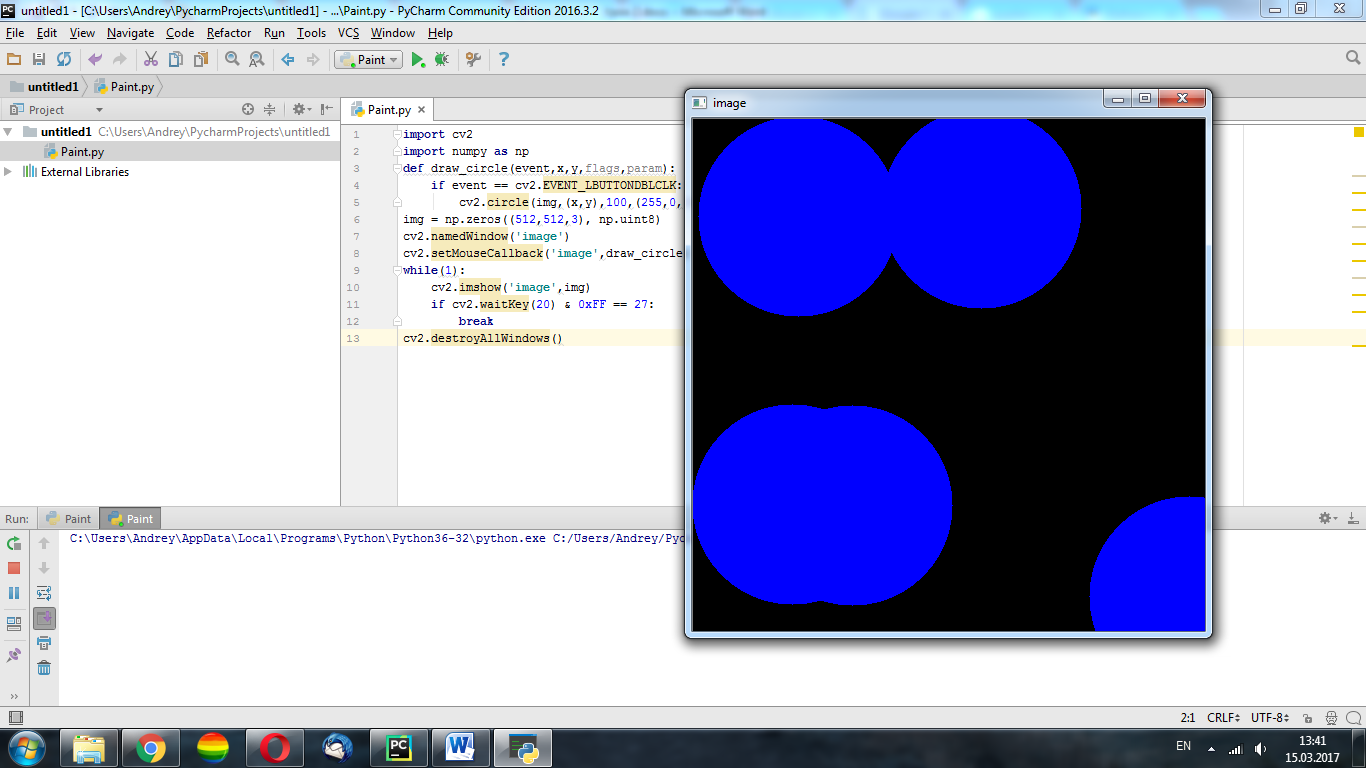
cv2.namedWindow(**'image'**)  
cv2.setMouseCallback(**'image'**,draw\_circle)

Выведем картинку в это окно и сделаем ее удаление и закрытие окна при нажатии на клавишу esc (код 27)

**while**(1):  
 cv2.imshow(**'image'**,img)  
 **if** cv2.waitKey(20) & 0xFF == 27:  
 **break**cv2.destroyAllWindows()



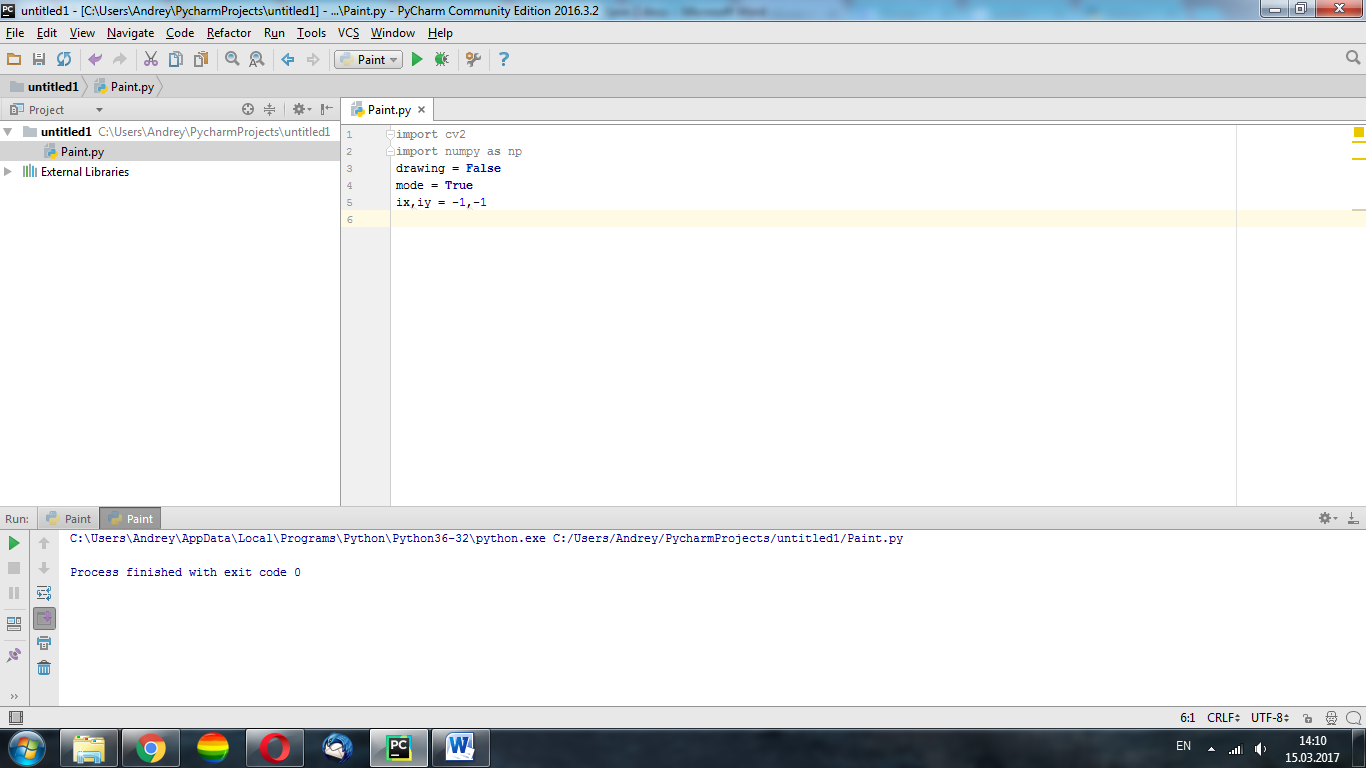
Теперь по двойному клику у нас будет рисоваться синий круг



Теперь мы перейдем к гораздо более эффективному приложению. Мы будем рисовать прямоугольники или круги (в зависимости от выбранного режима), передвигая мышь, как мы это делаем в приложении Paint. Таким образом, наша функция обратного вызова с мышью состоит из двух частей, одна для рисования прямоугольника, а другая для рисования кругов. Этот конкретный пример будет очень полезен при создании и понимании некоторых интерактивных приложений, таких как отслеживание объектов, сегментация изображений и т.д.

**import** cv2  
**import** numpy **as** np  
drawing = **False**mode = **True**ix,iy = -1,-1

Для начала функция рисования будет отключена, она перейдет в true при нажатии на клавишу мыши. mode = **True** по умолчанию мы будем рисовать прямоугольники, а при нажатии на клавишу «m» (это мы опишем позже) будем переходит на рисование кругов. ix,iy = -1,-1 позже мы их будем приравнивать координатам мышки

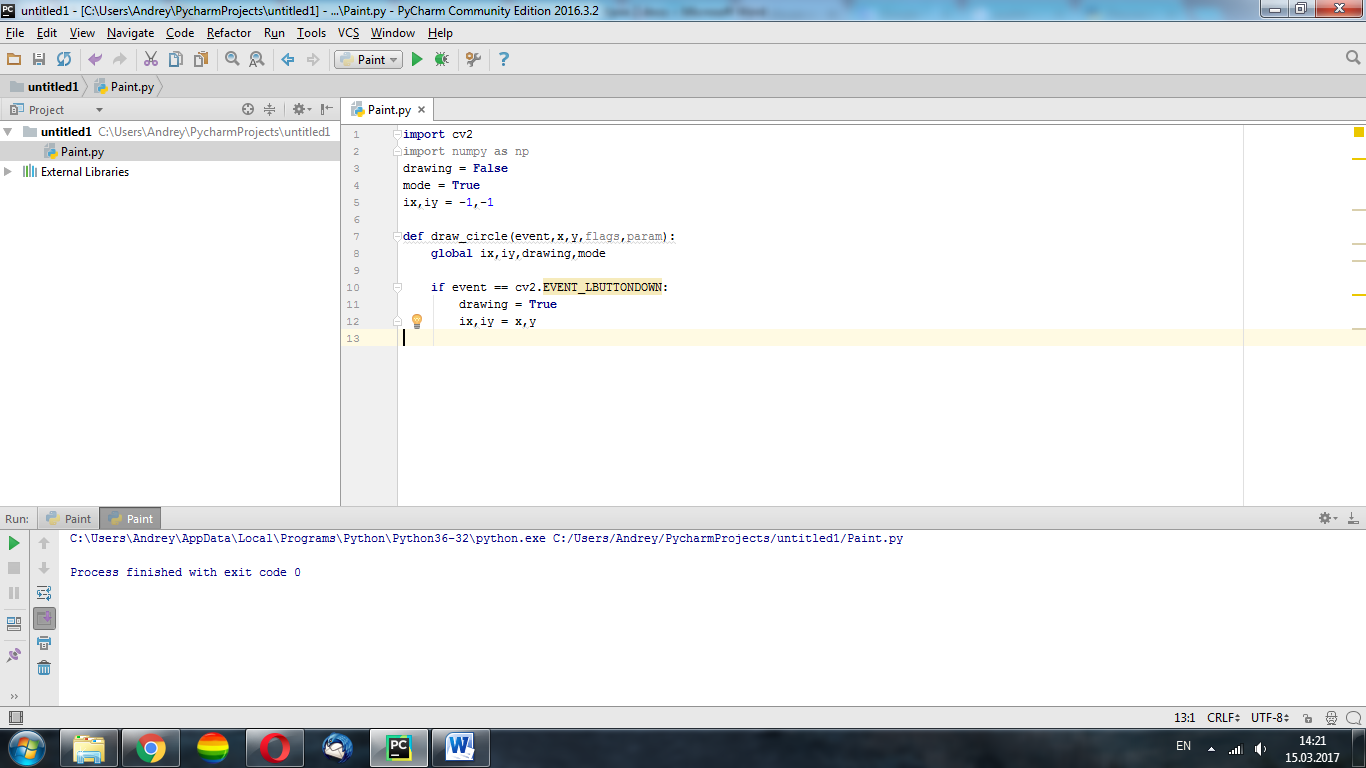


Перейдем к функции обратного вызова мышки.

**def** draw\_circle(event,x,y,flags,param):  
 **global** ix,iy,drawing,mode

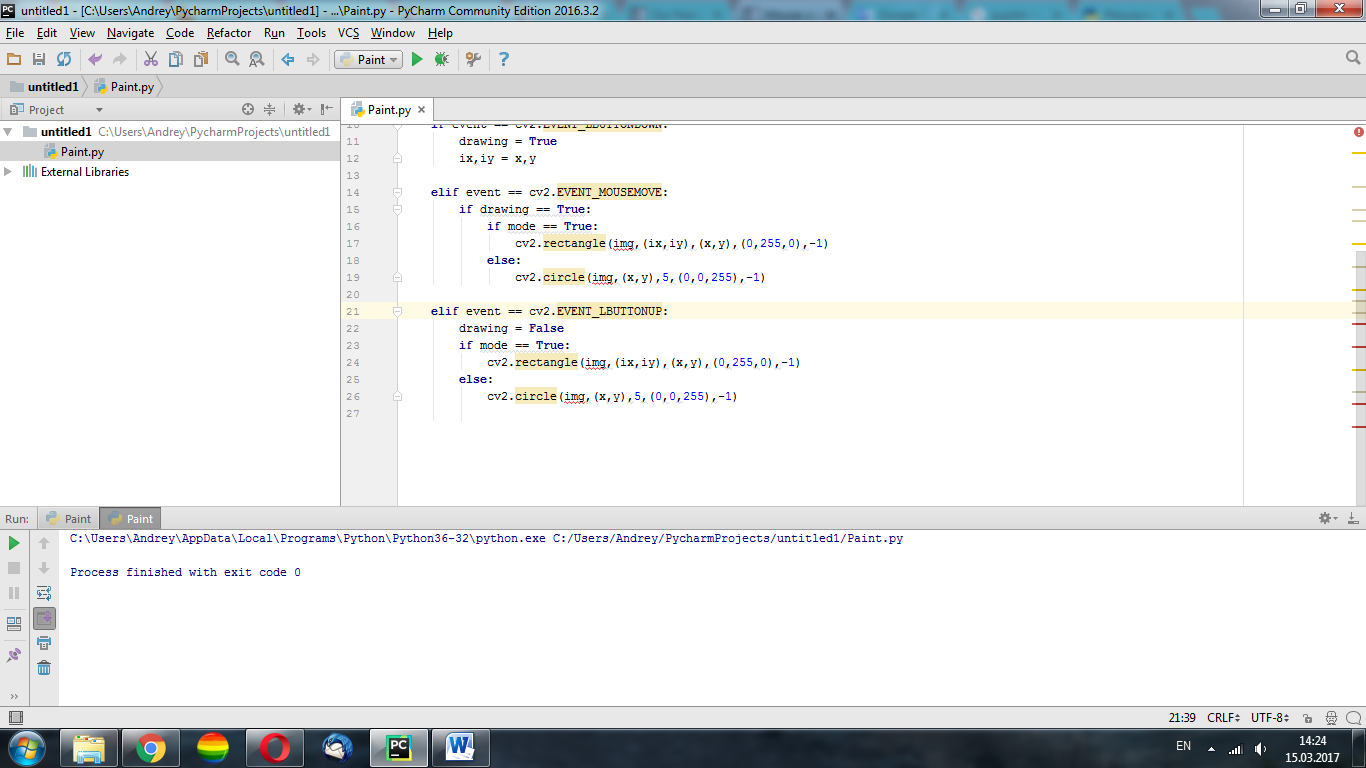
И если нажата левая клавиша мышки то мы подключаем функцию рисования для текущий координат.

**if** event == cv2.EVENT\_LBUTTONDOWN:  
 drawing = **True** ix,iy = x,y

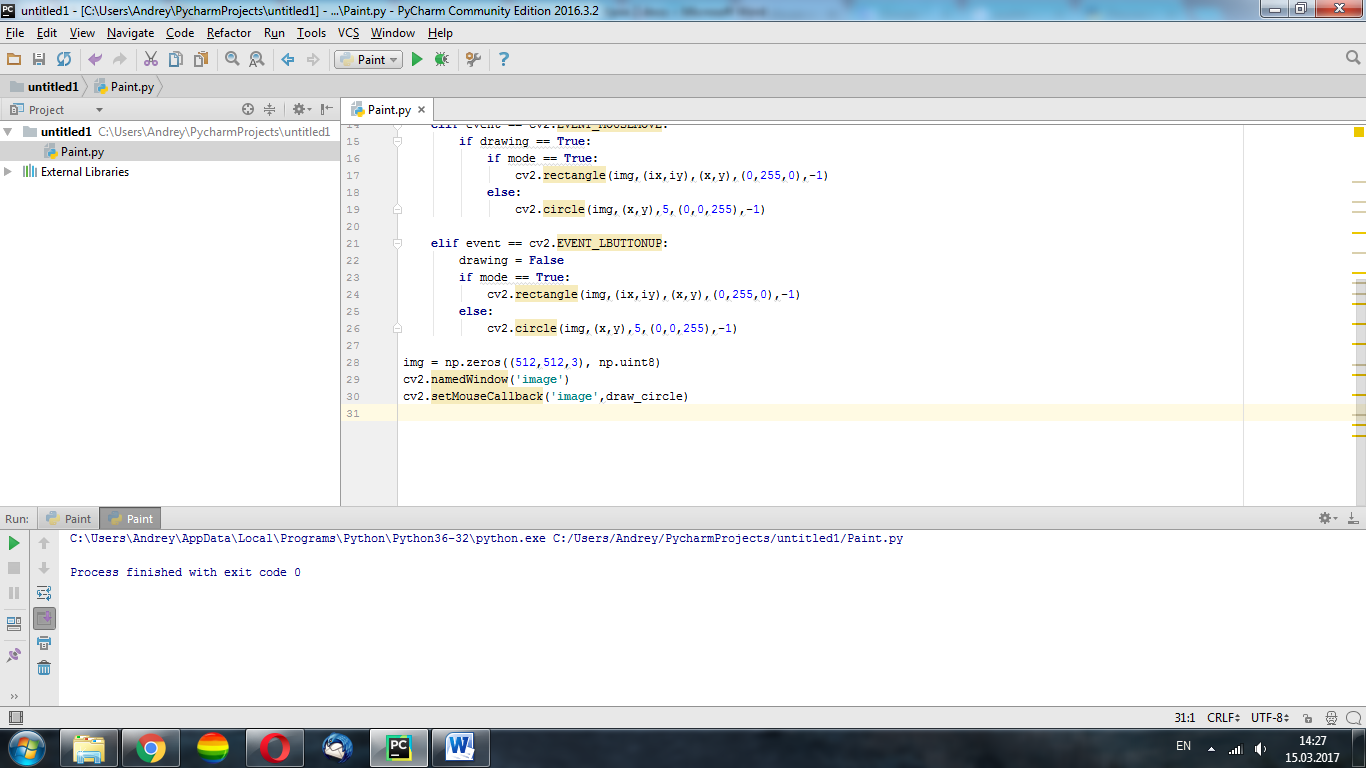


Если мышка двигается то drawing == **True** и рисуется зеленый прямоугольник или (если была нажата клавиша «m») красный круг диаметром 5 пикселей

**elif** event == cv2.EVENT\_MOUSEMOVE:  
 **if** drawing == **True**:  
 **if** mode == **True**:  
 cv2.rectangle(img,(ix,iy),(x,y),(0,255,0),-1)  
 **else**:  
 cv2.circle(img,(x,y),5,(0,0,255),-1)



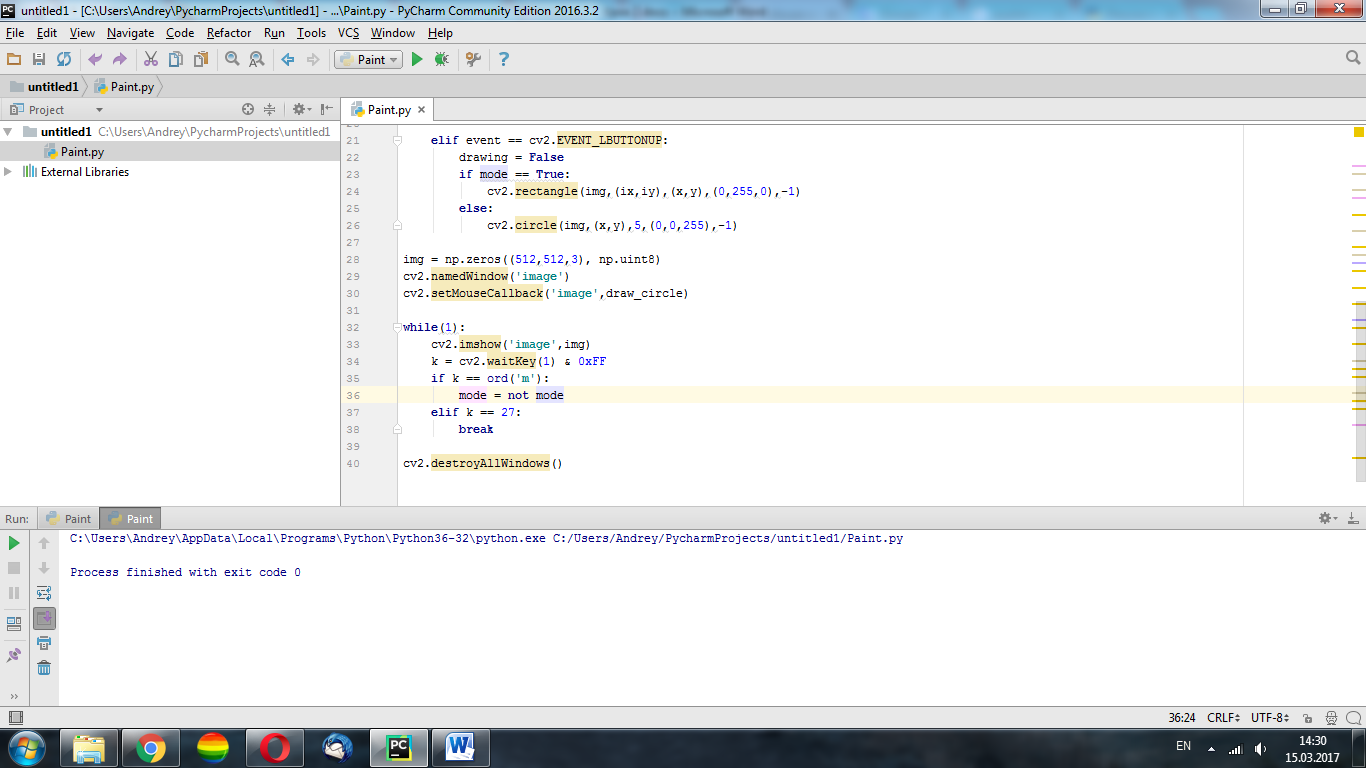
Создадим окно 512 на 512 черного цвета, поставим имя для окна программы, вызовем функцию setMouseCallback для этого окна.



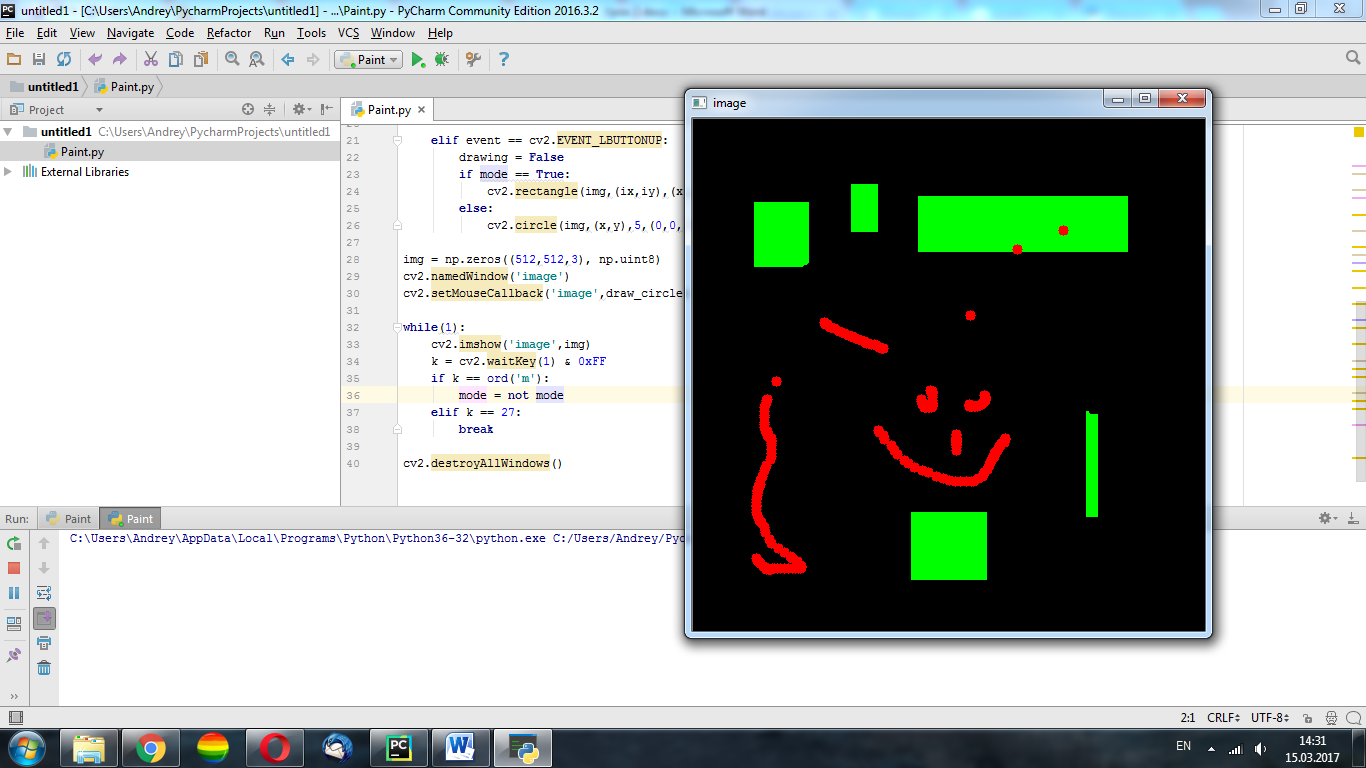
Выведем картинку в это окно и если нажата клавиша «m» то mode принимает обратное значение и вместо прямоугольника мы сможем рисовать круг.

Закрытие окна при нажатии на клавишу esc (код 27)

**while**(1):  
 cv2.imshow(**'image'**,img)  
 k = cv2.waitKey(1) & 0xFF  
 **if** k == ord(**'m'**):  
 mode = **not** mode  
 **elif** k == 27:  
 **break**cv2.destroyAllWindows()



При нажатии левой клавиши будет рисоваться зеленый прямоугольник (или квадрат). Нажмите «m» для смены режима



Задание: измените цвет круга и треугольника и уберите сплошную заливку!

