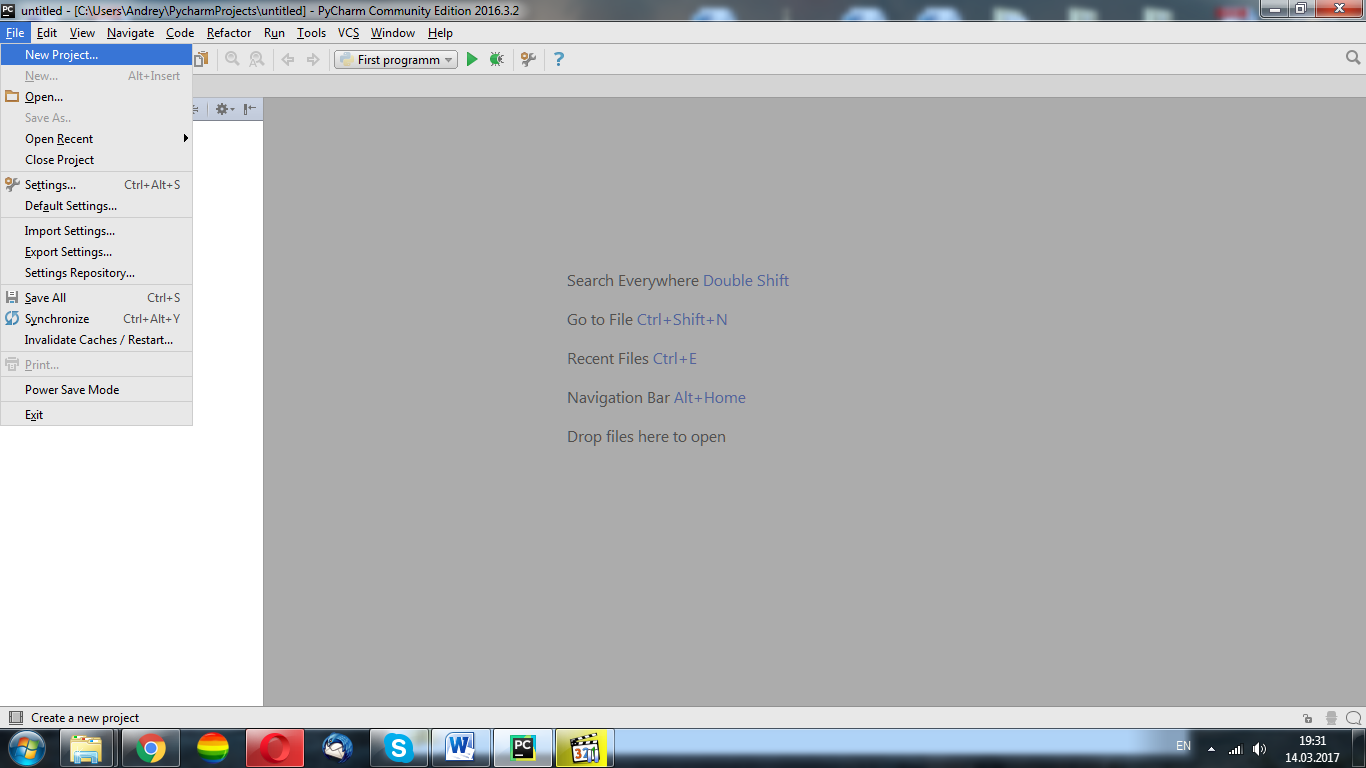
Урок № 4!

Изменение цветового пространства и работа с пороговыми значениями.

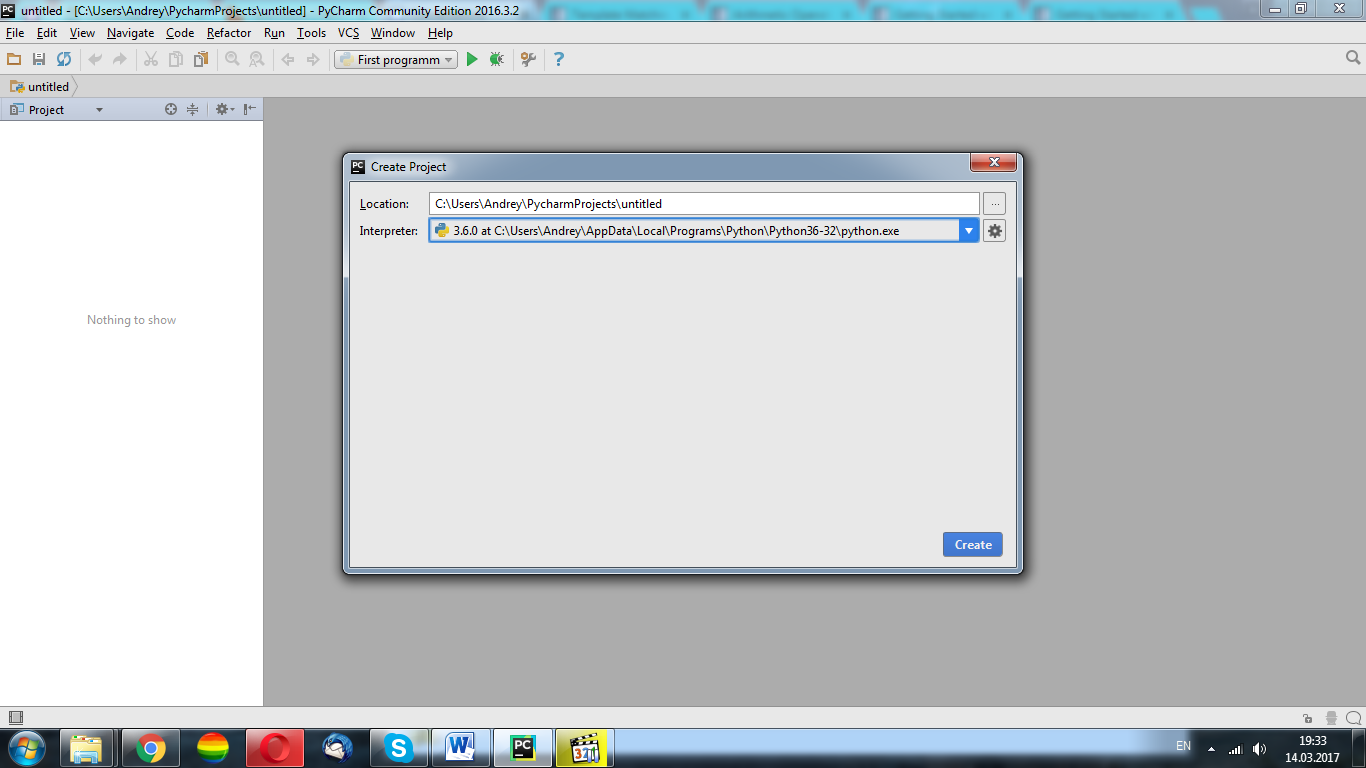
Запустите PyCharm Community Edition

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.

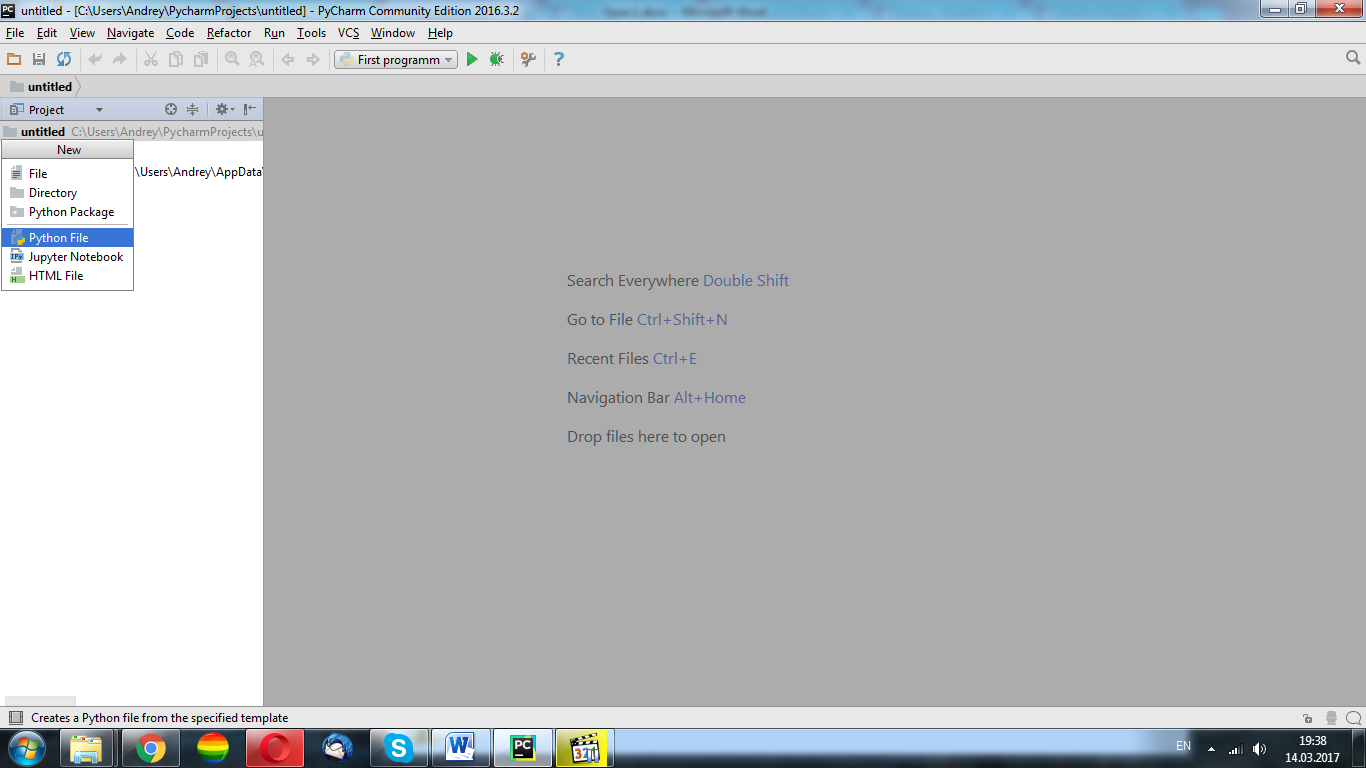
Во второй строке версия python. Выберите последнюю версию 3.6. Нажмите Create



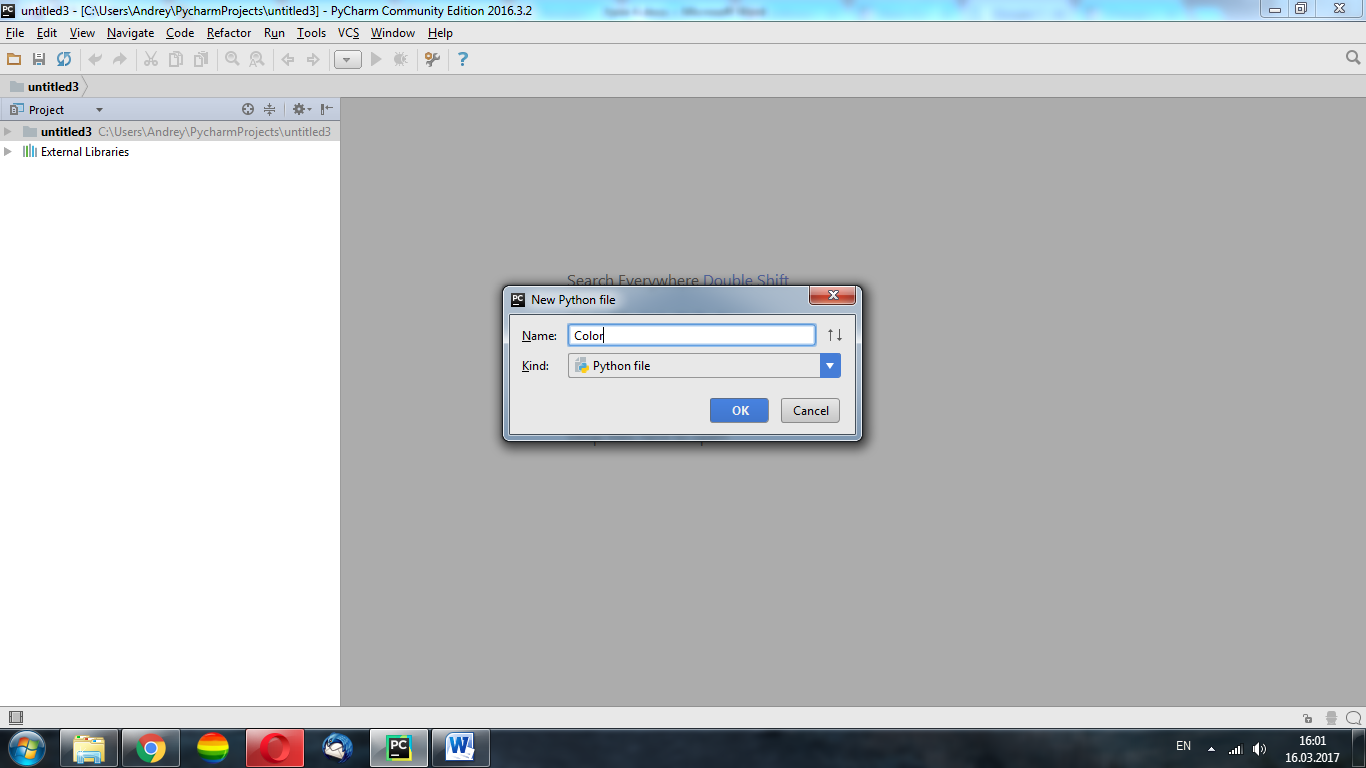
Далее создадим файл, в котором будем работать. Во вкладке File нажмите New…



Python file



Назовем этот файл Color



В OpenCV доступно более 150 методов преобразования цветового пространства. Но мы рассмотрим только два, которые наиболее широко используются, BGR Gray и BGR HSV.

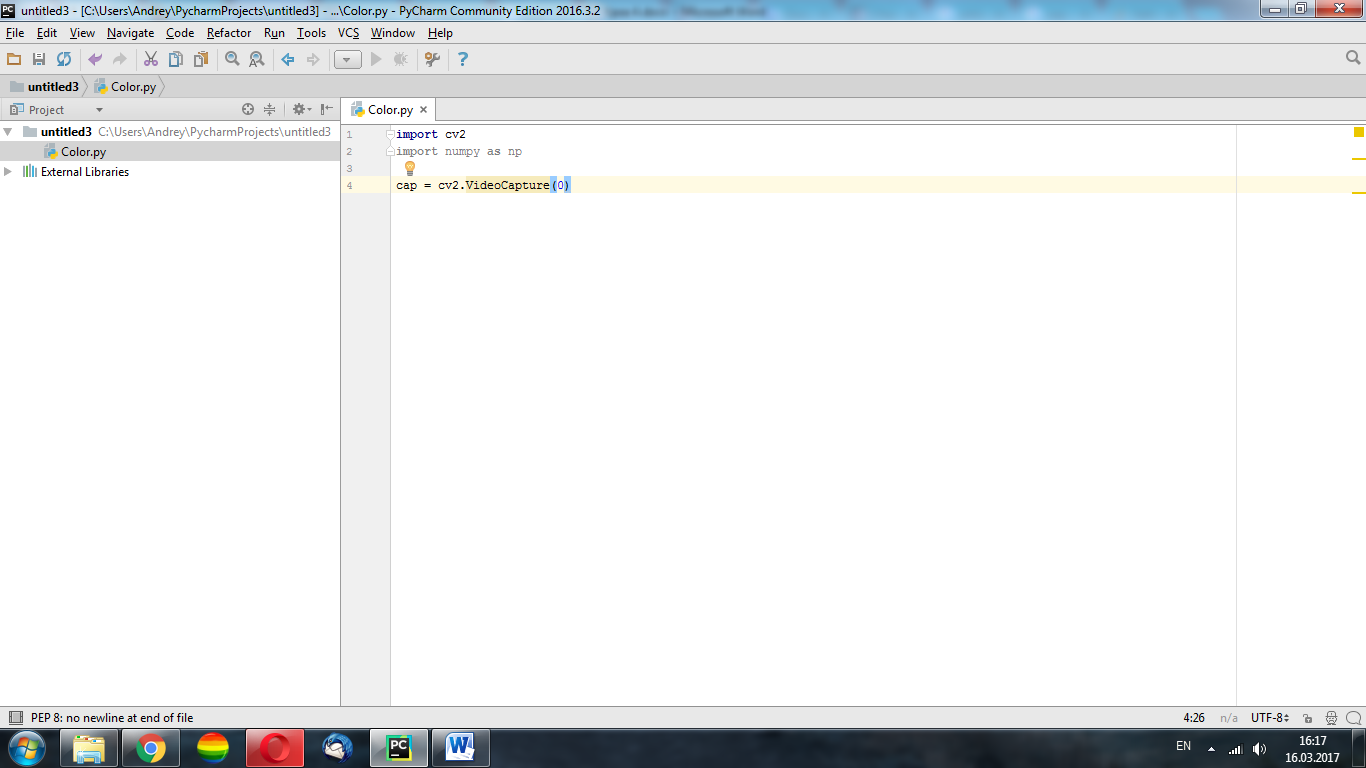
Для преобразования цветов мы будем использовать функцию cv2.cvtColor (input\_image, flag), где флаг определяет тип преобразования.

Для преобразования BGR Gray мы используем флаги cv2.COLOR\_BGR2GRAY. Аналогично для HSV мы используем флаг cv2.COLOR\_BGR2HSV.

Теперь мы знаем, как преобразовать BGR-изображение в HSV, мы можем использовать его для извлечения определенного цвета объекта. В HSV легче представить цвет, чем цветовое пространство RGB. В нашем приложении мы попытаемся извлечь синий цвет. Использовать будем веб камеру.

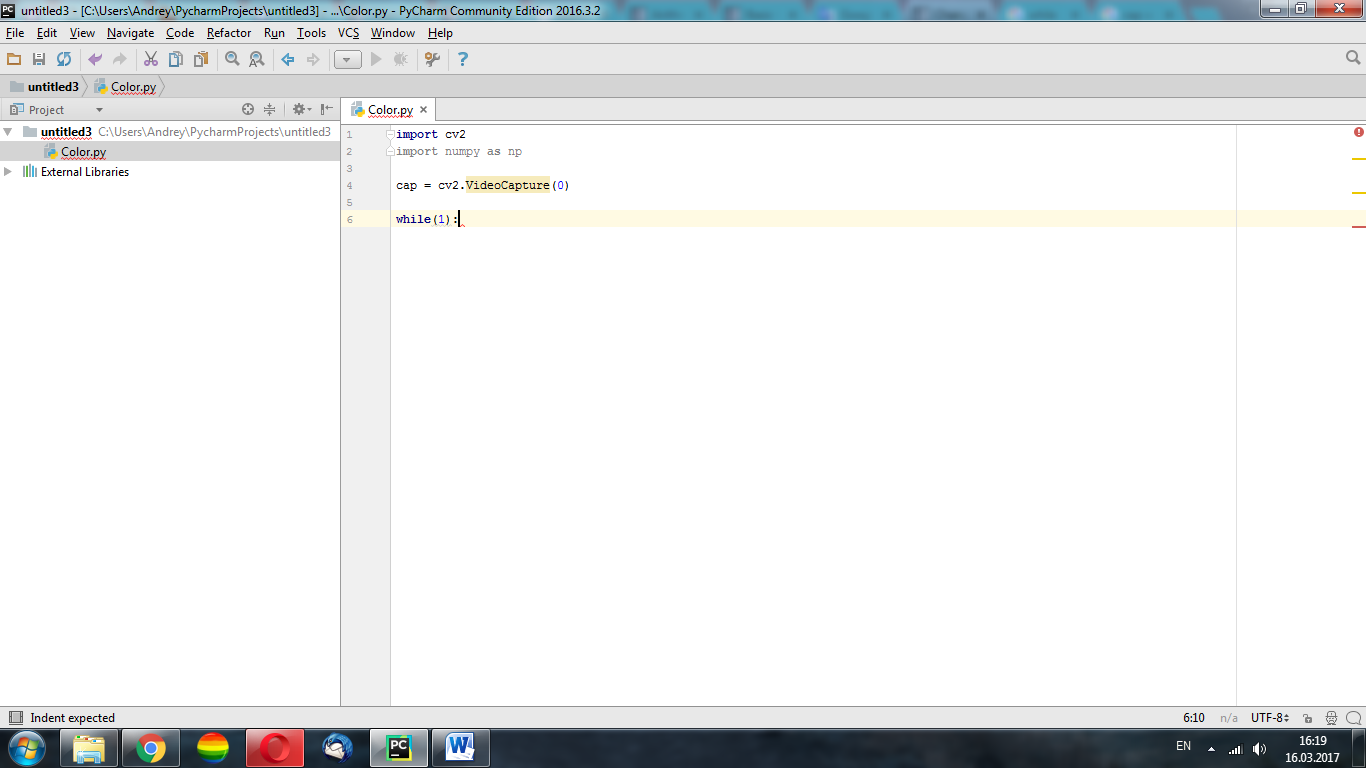
Подключим библиотеки и захват с веб камеры (0 это камера по умолчанию, если у вас подключенная камера, то поставьте 1)

**import** cv2  
**import** numpy **as** np  
  
cap = cv2.VideoCapture(0)



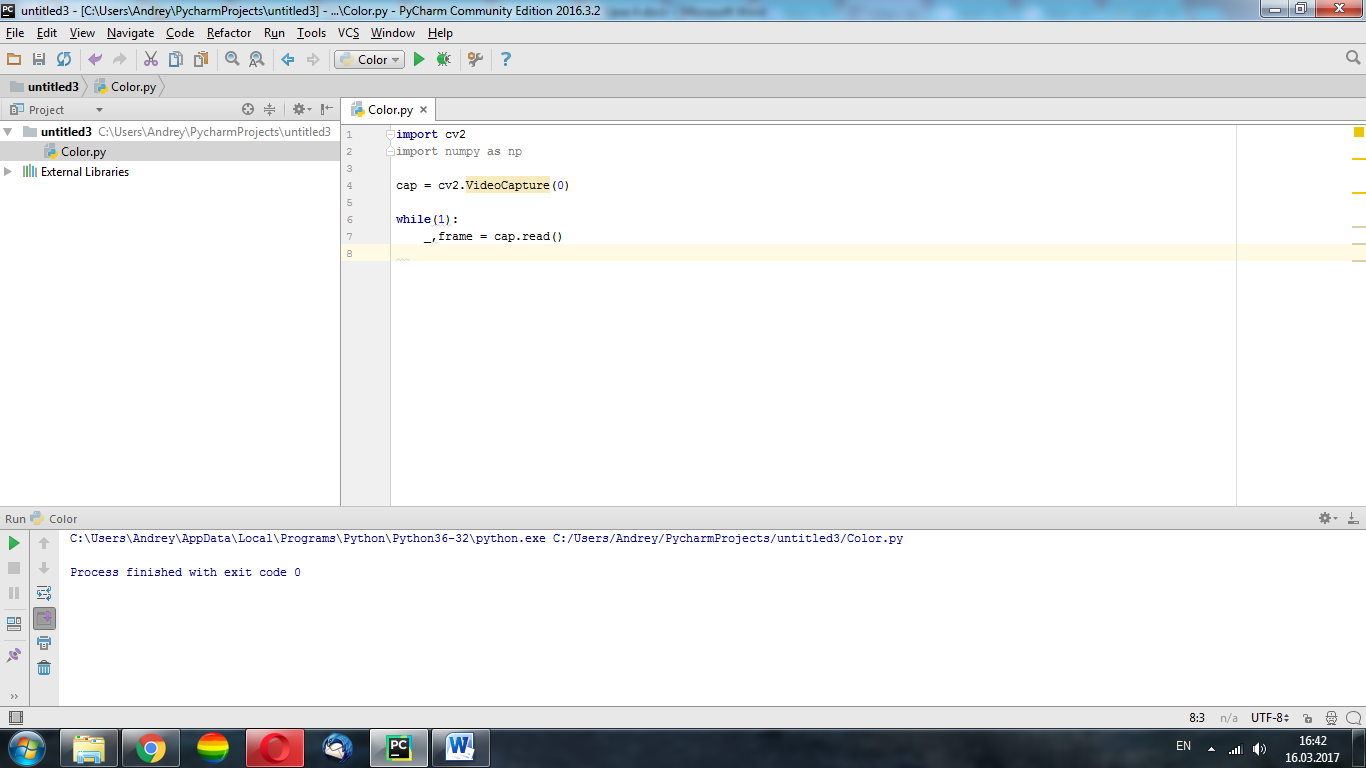
Открываем цикл

**while**(1):



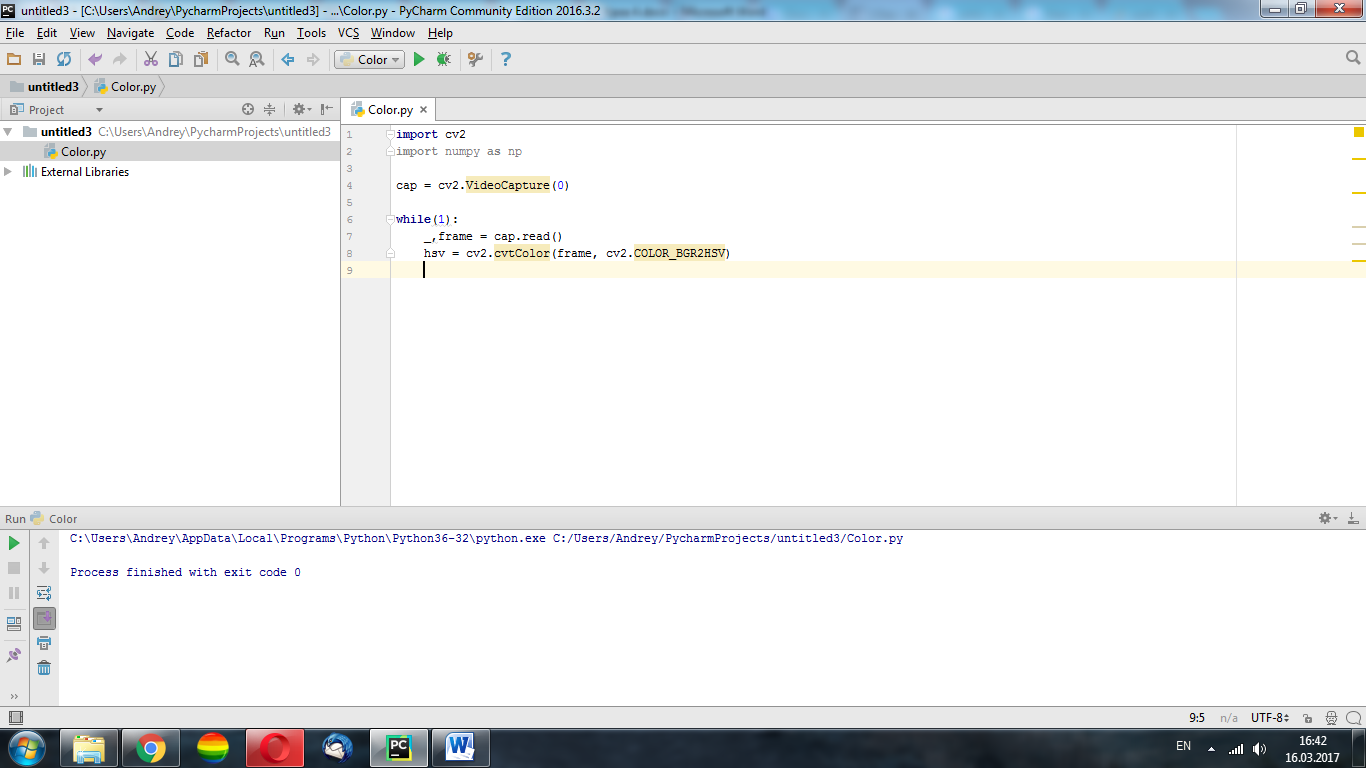
\_,frame **=** cap**.**read()

Дословно: получить каждый кадр с камеры



Конвертируем BGR в HSV

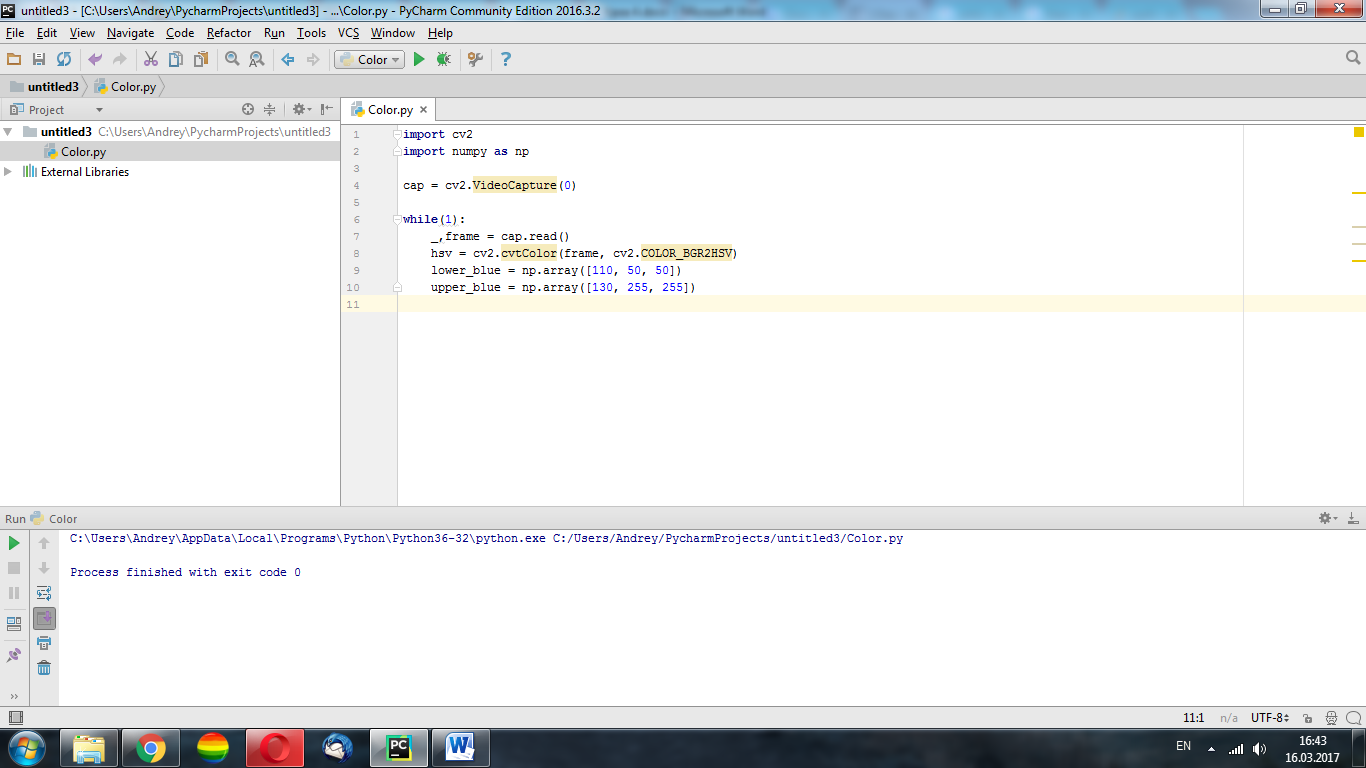
hsv **=** cv2**.**cvtColor(frame, cv2**.**COLOR\_BGR2HSV)



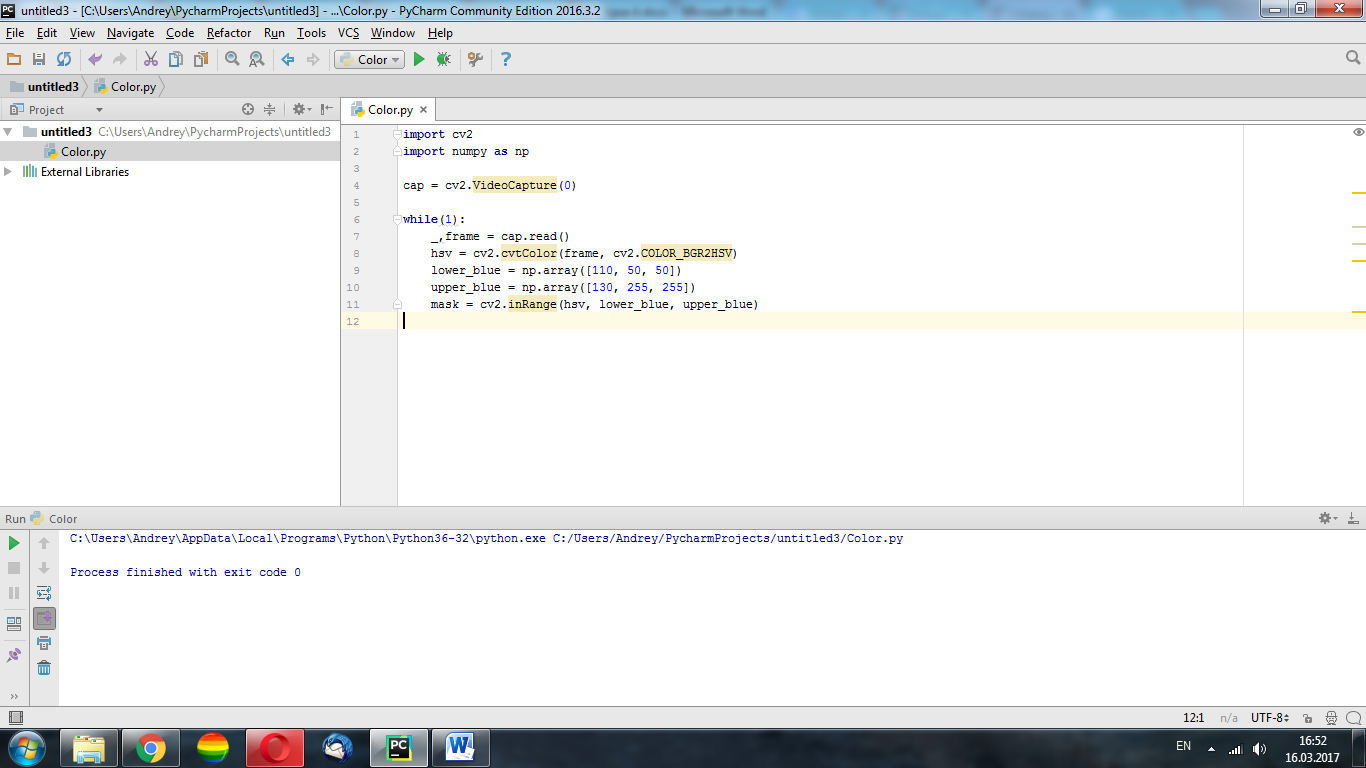
Определим диапазон синего в HSV

lower\_blue **=** np**.**array([110,50,50])

upper\_blue **=** np**.**array([130,255,255])

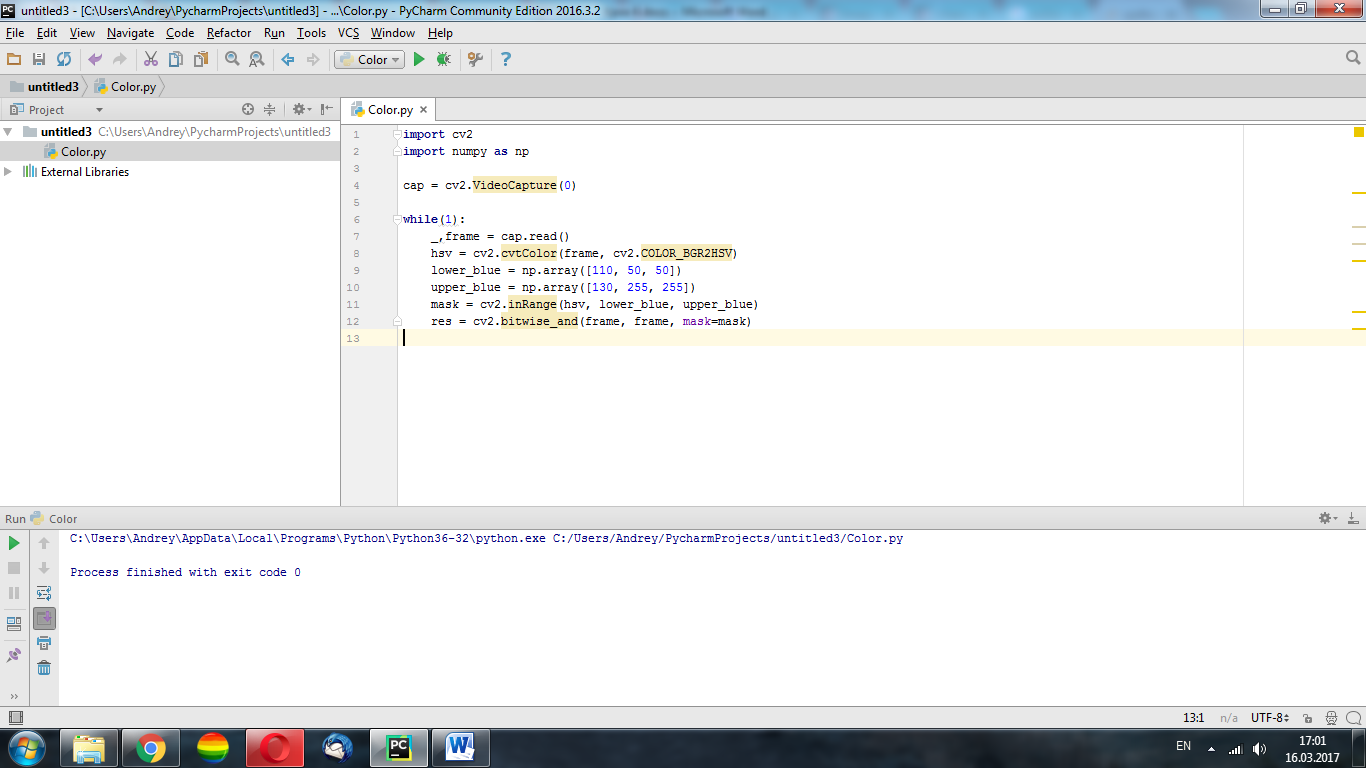


Используя функцию cv2.inRange поставим порог для изображения HSV, чтобы получить только синие цвета



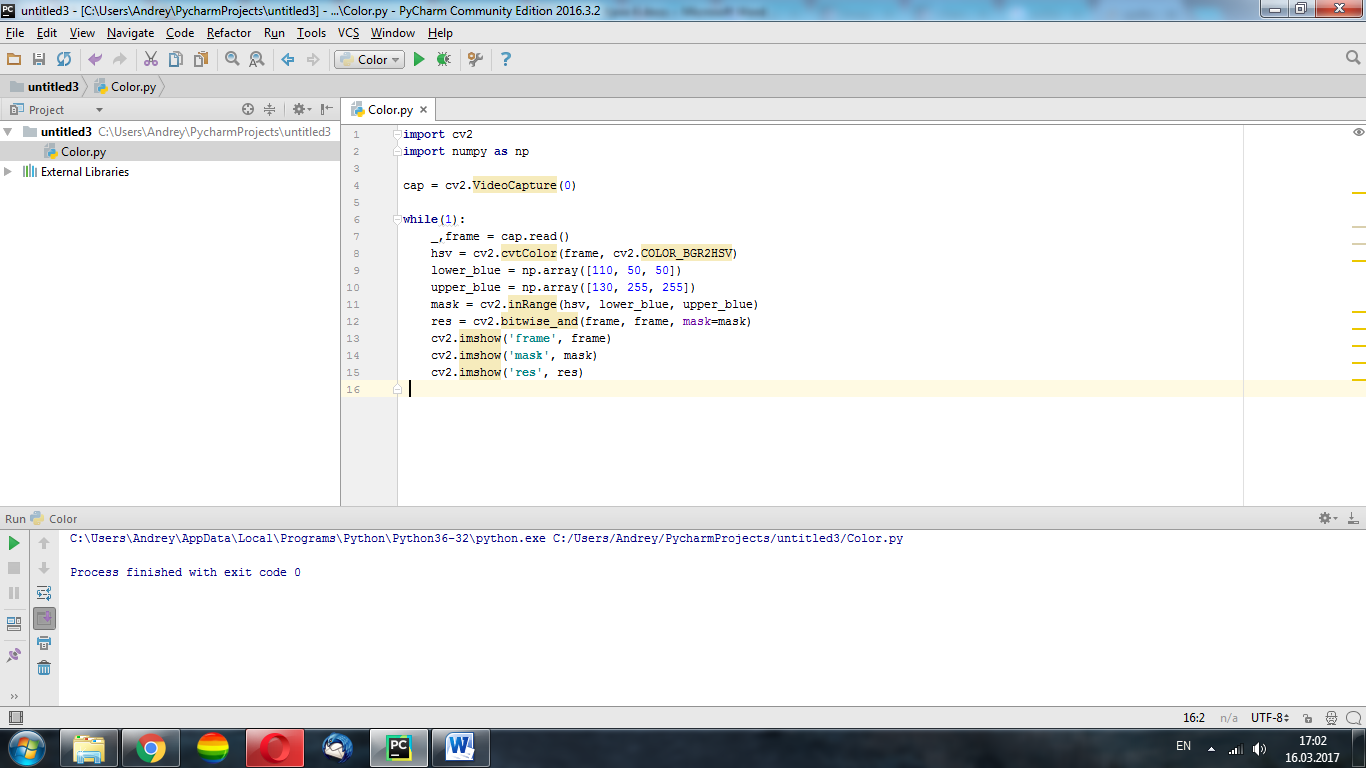
И так как у нас есть маска изображения, мы можем использовать функцию cv2.bitwise\_and

res = cv2.bitwise\_and(frame, frame, mask=mask)



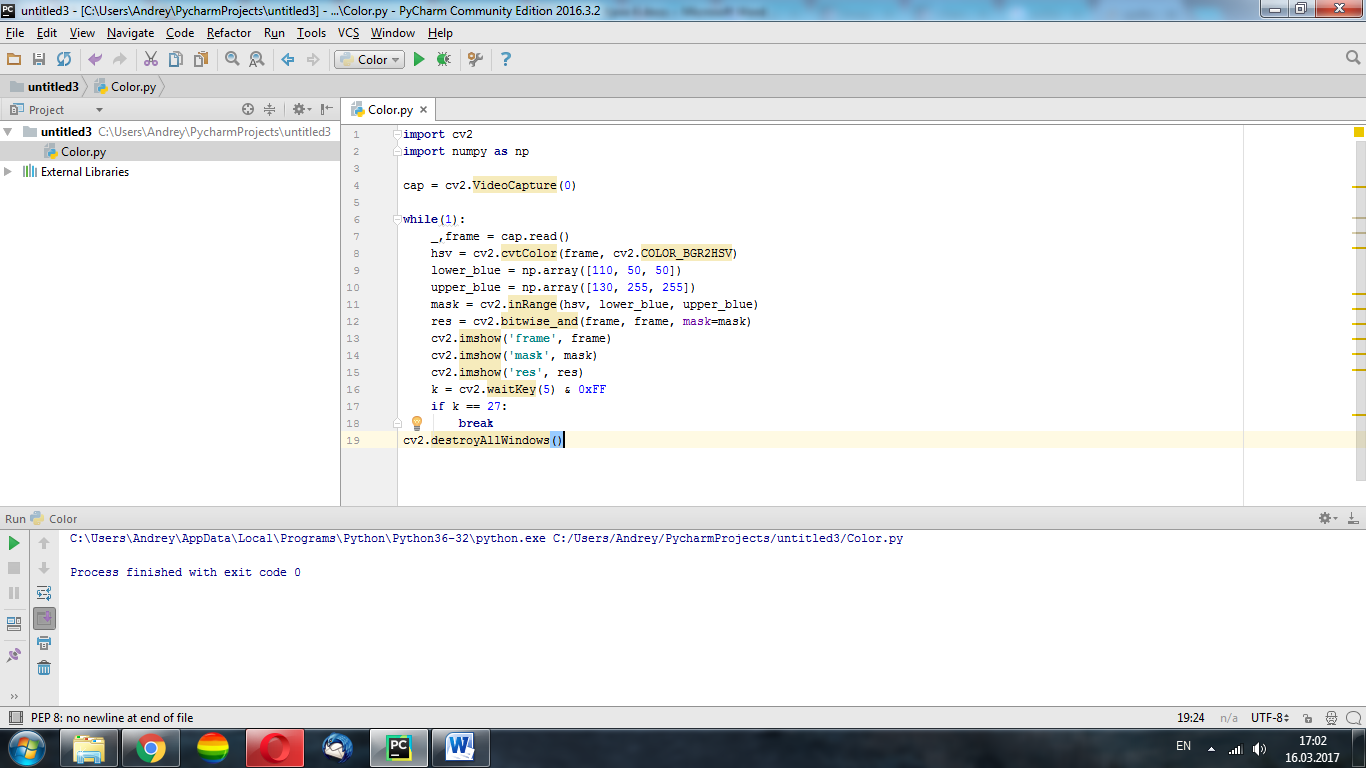
Выведем оригинальное изображение, маску и результат

cv2.imshow(**'frame'**, frame)  
cv2.imshow(**'mask'**, mask)  
cv2.imshow(**'res'**, res)

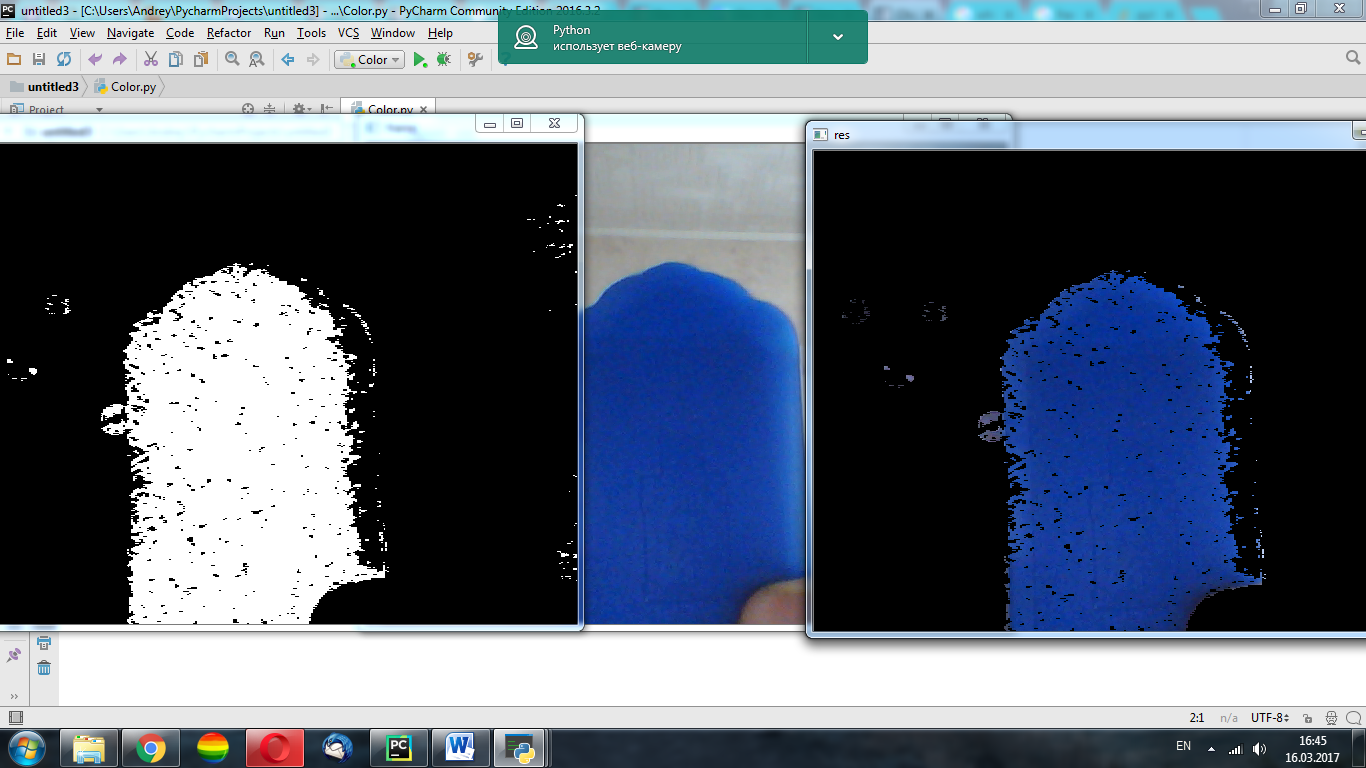


Поставим закрытие программы по нажатию клавиши «esc»

k = cv2.waitKey(5) & 0xFF  
 **if** k == 27:  
 **break**cv2.destroyAllWindows()



Запустите программу и поднесите какой-нибудь обьект синего цвета к веб камере



Задание: Попробуйте найти способ выделения более чем одного цветного объекта, например, одновременного выделения красных, синих и зеленых объектов.

Если значение пикселя больше порогового значения, ему присваивается одно значение (к примеру - белый), иначе ему присваивается другое значение (черный). Используемая функция - cv2.threshold.

Первым аргументом является исходное изображение, которое должно быть полутоновым.

Второй аргумент - это пороговое значение, которое используется для классификации значений пикселей.

Третий аргумент - это maxVal, представляющий значение, которое должно быть задано, если значение пикселя больше, чем пороговое значение (иногда меньше).

OpenCV предоставляет различные стили пороговых значений и определяется четвертым параметром функции. Различными типами являются:

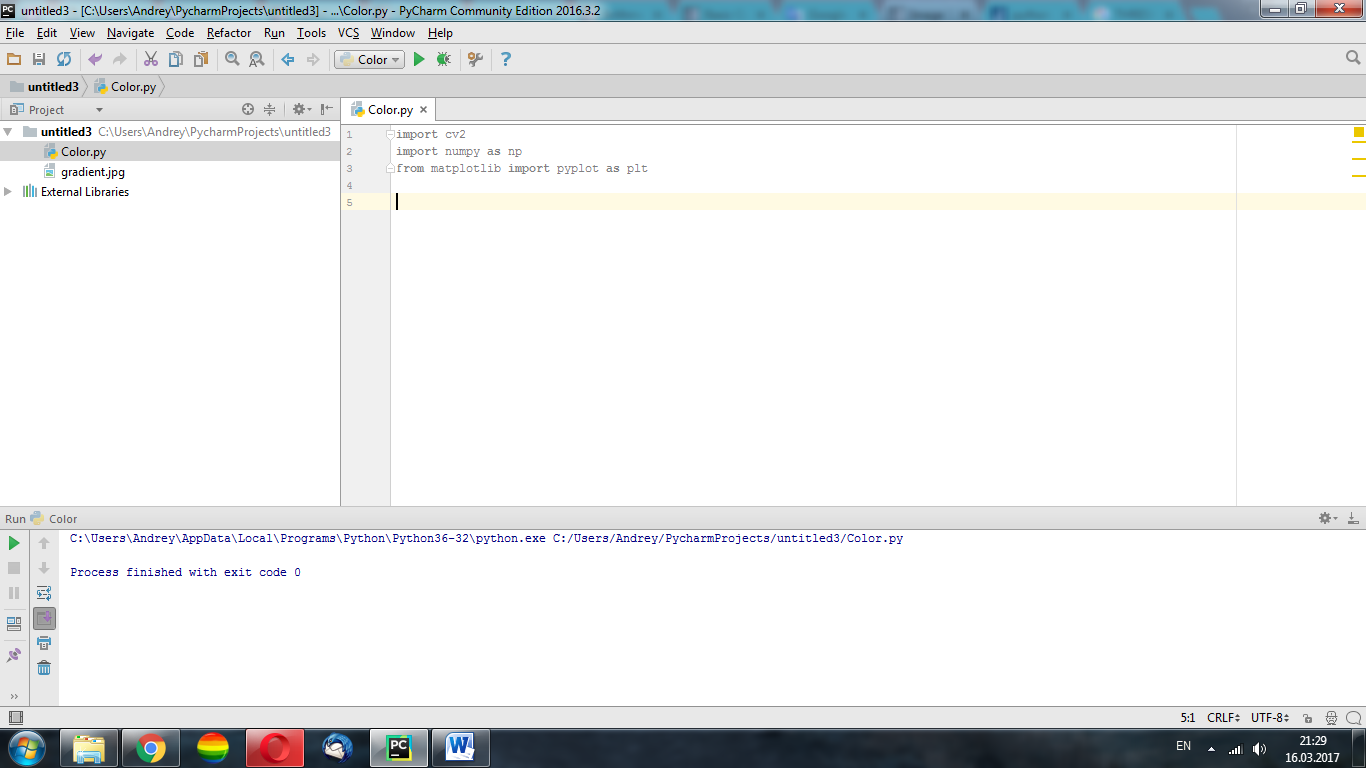
* cv2.THRESH\_BINARY
* cv2.THRESH\_BINARY\_INV
* cv2.THRESH\_TRUNC
* cv2.THRESH\_TOZERO
* cv2.THRESH\_TOZERO\_INV

Начнем с подключения библиотек cv2, numpy и pyplot

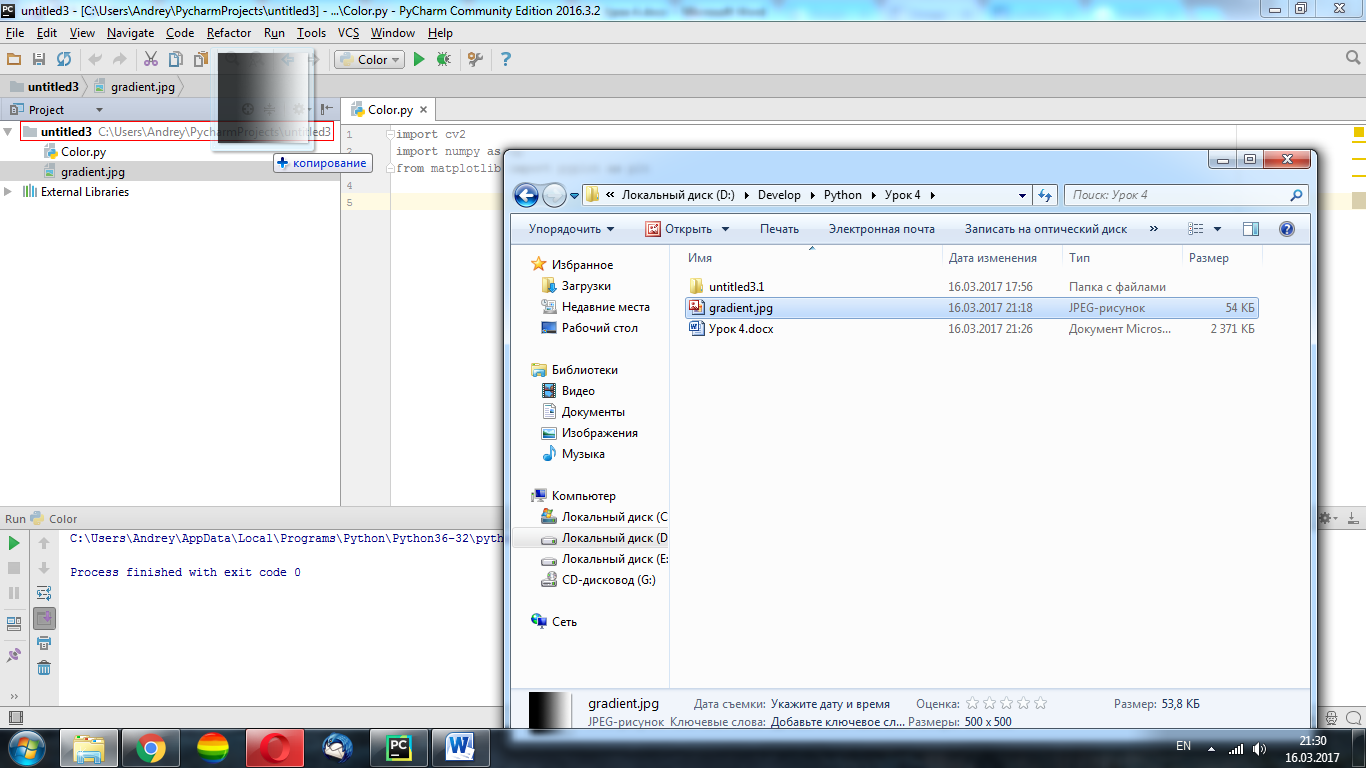
**import** cv2

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

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

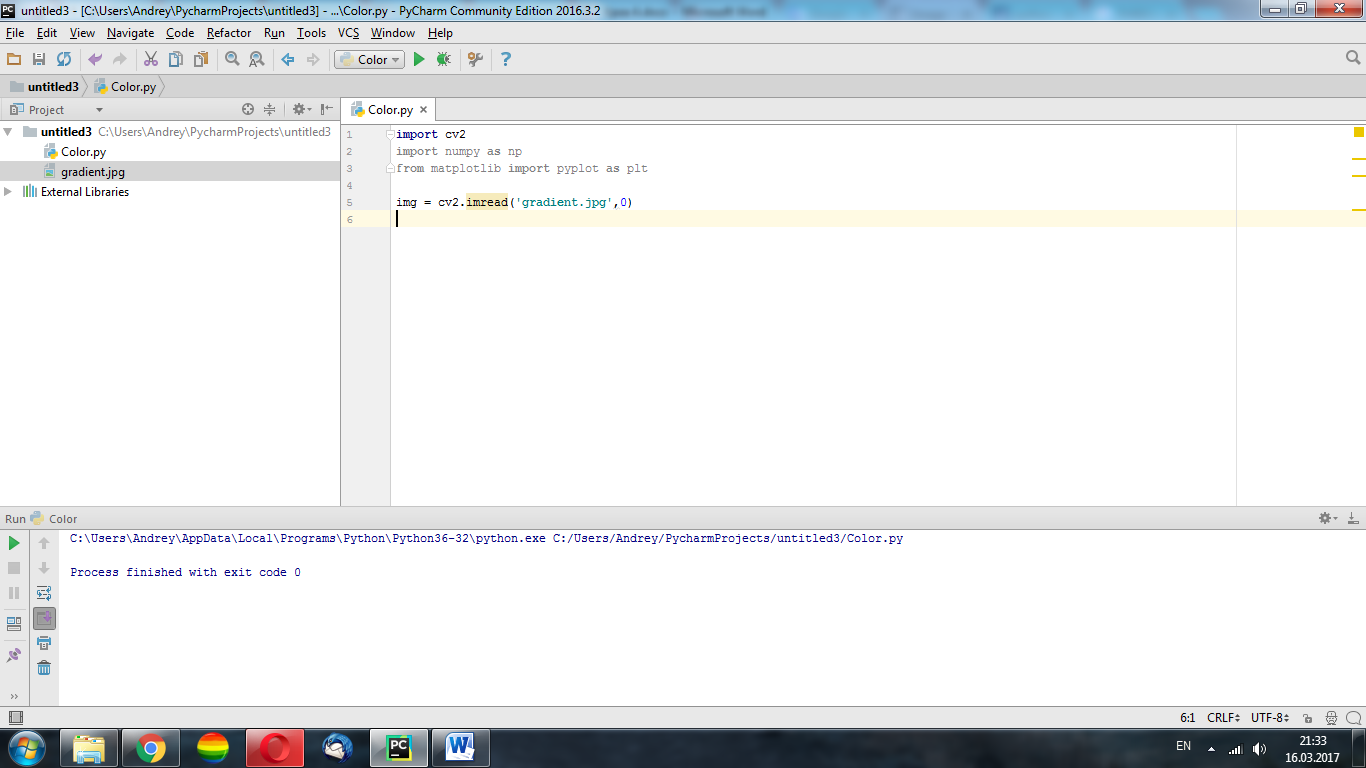


Загрузим картинку из папки с уроком. Выделите ее и зажимая клавишу ctrl перетащите в программу.



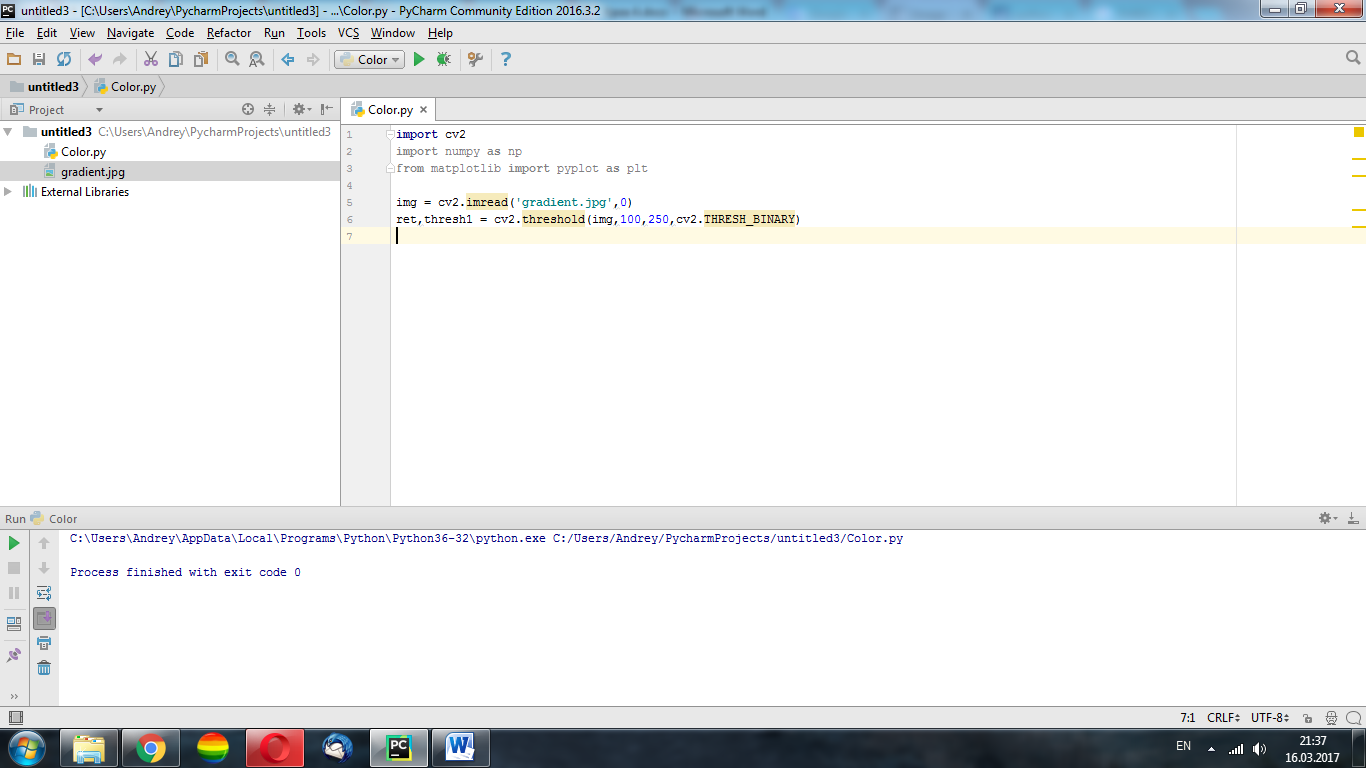
Зададим значение переменной img для чтения нашей картинки в сером цвете.

img = cv2.imread(**'gradient.jpg'**,0)



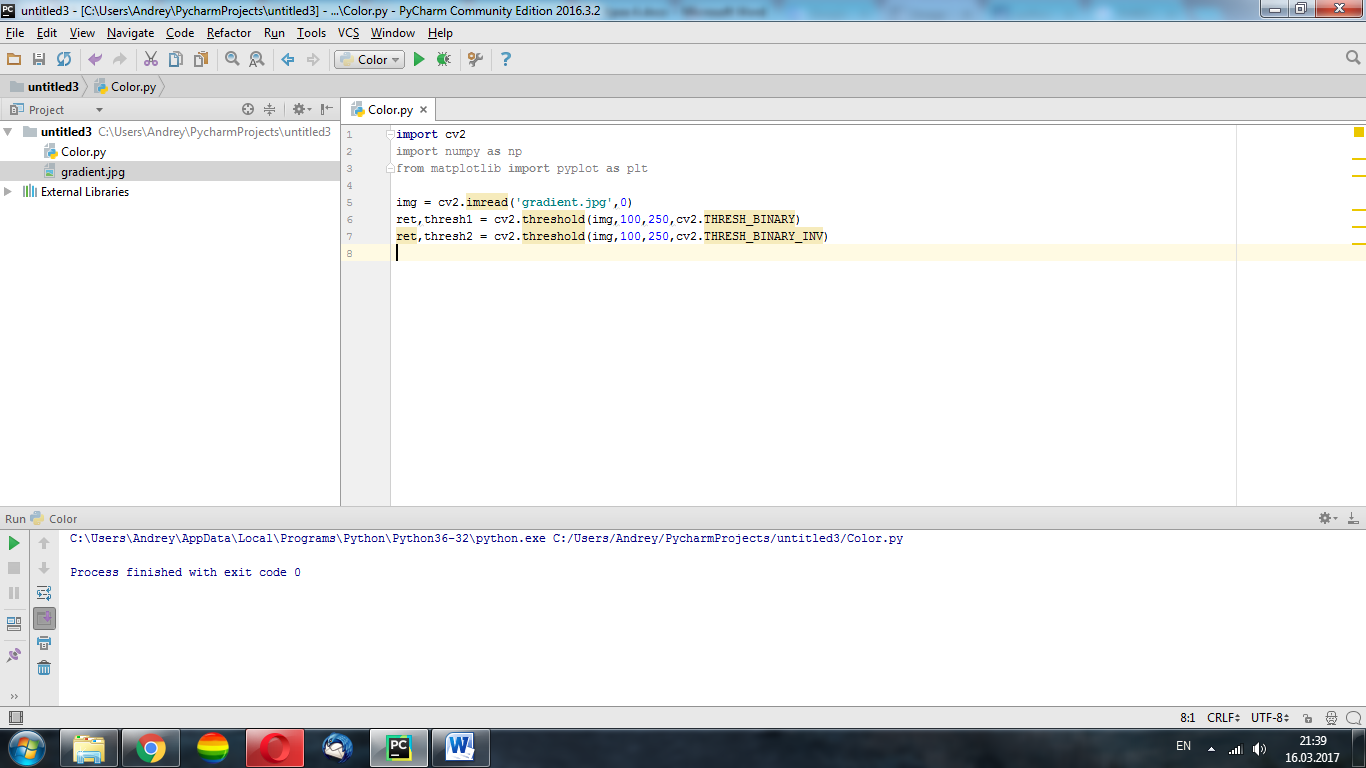
В переменную thresh1 записываем координаты выбранной точки на картинке и все что светлее этой точки будет белым, а все что темнее этой точки станет черным.

ret,thresh1 = cv2.threshold(img,100,250,cv2.THRESH\_BINARY)



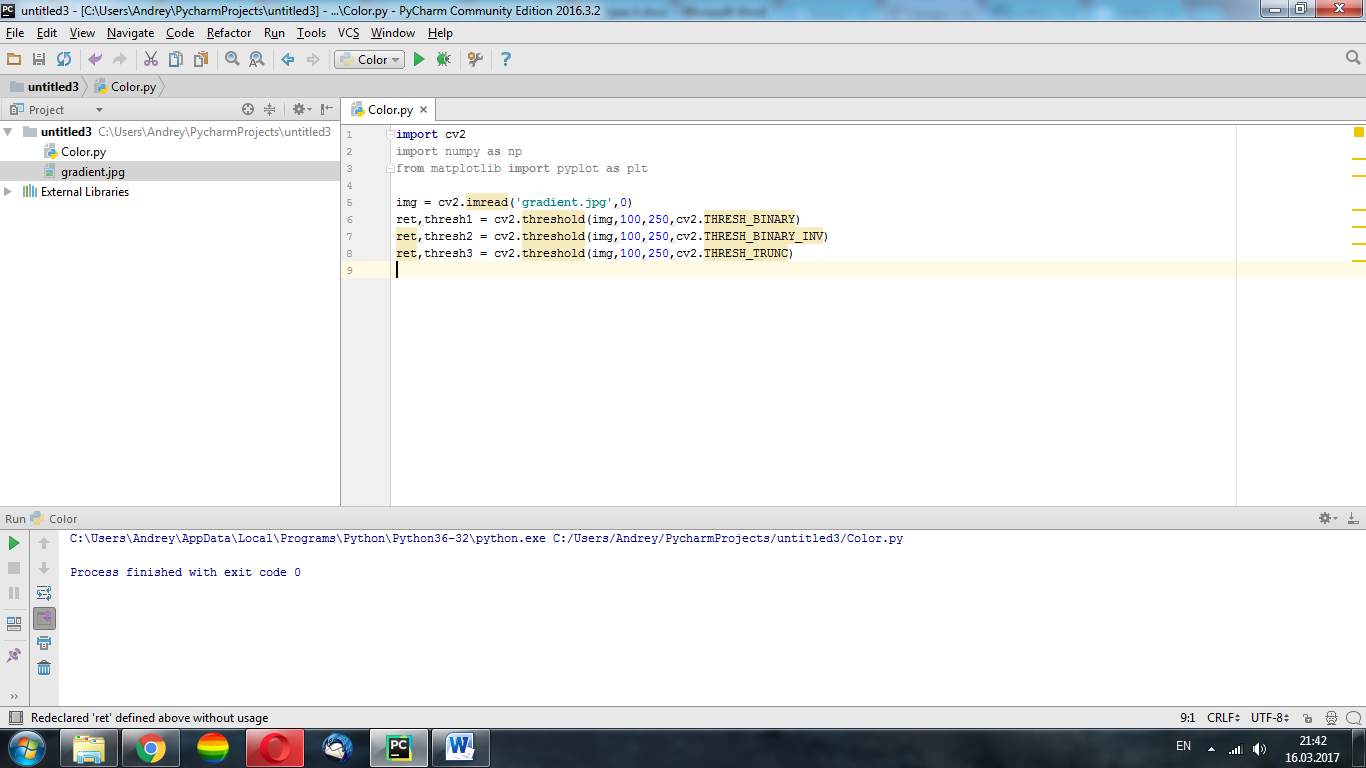
Далее посмотрим функцию инверсии для выбранной точки. Теперь, все что светлее станет черным, а все что темнее станет белым.

ret,thresh2 = cv2.threshold(img,100,250,cv2.THRESH\_BINARY\_INV)



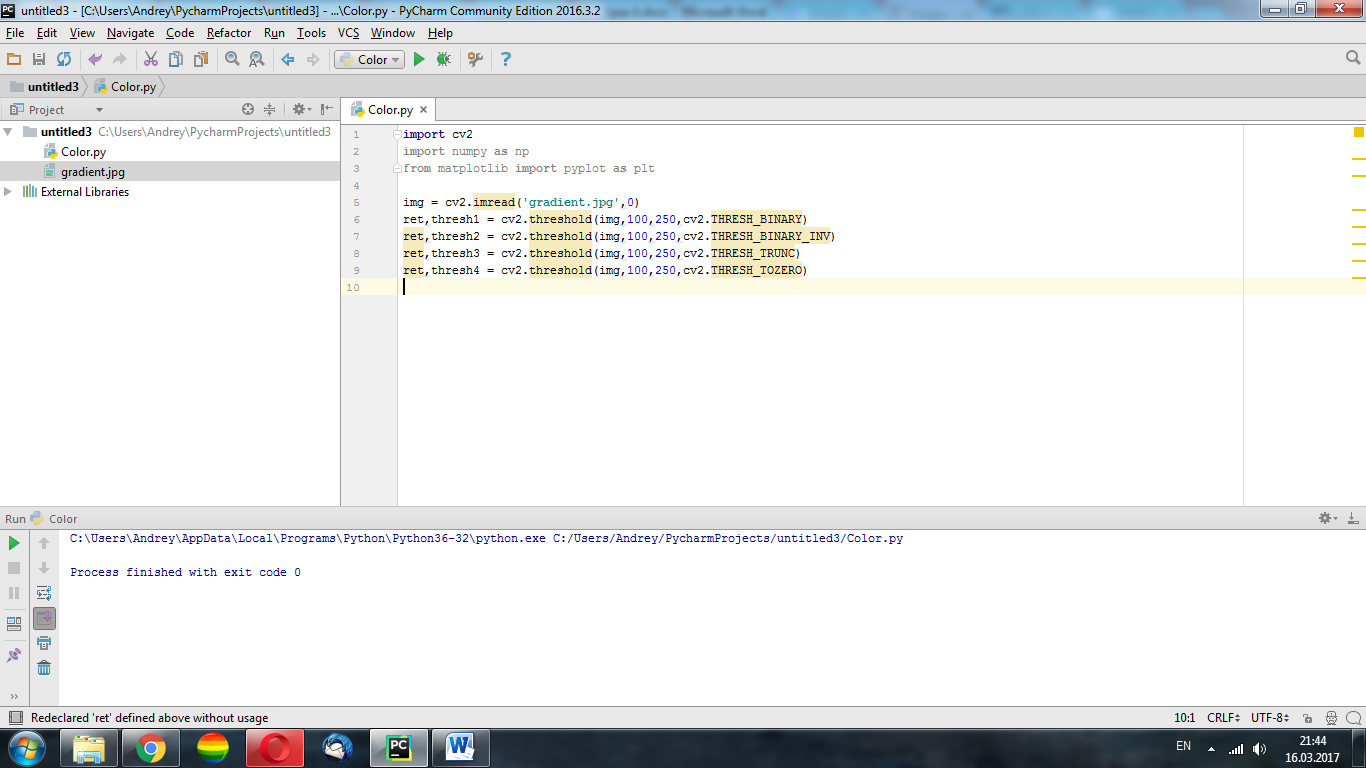
Функция thesh trunk позволяет нам сделать все, что светлее выбранного порогового значения станет белым.

ret,thresh3 = cv2.threshold(img,100,250,cv2.THRESH\_TRUNC)



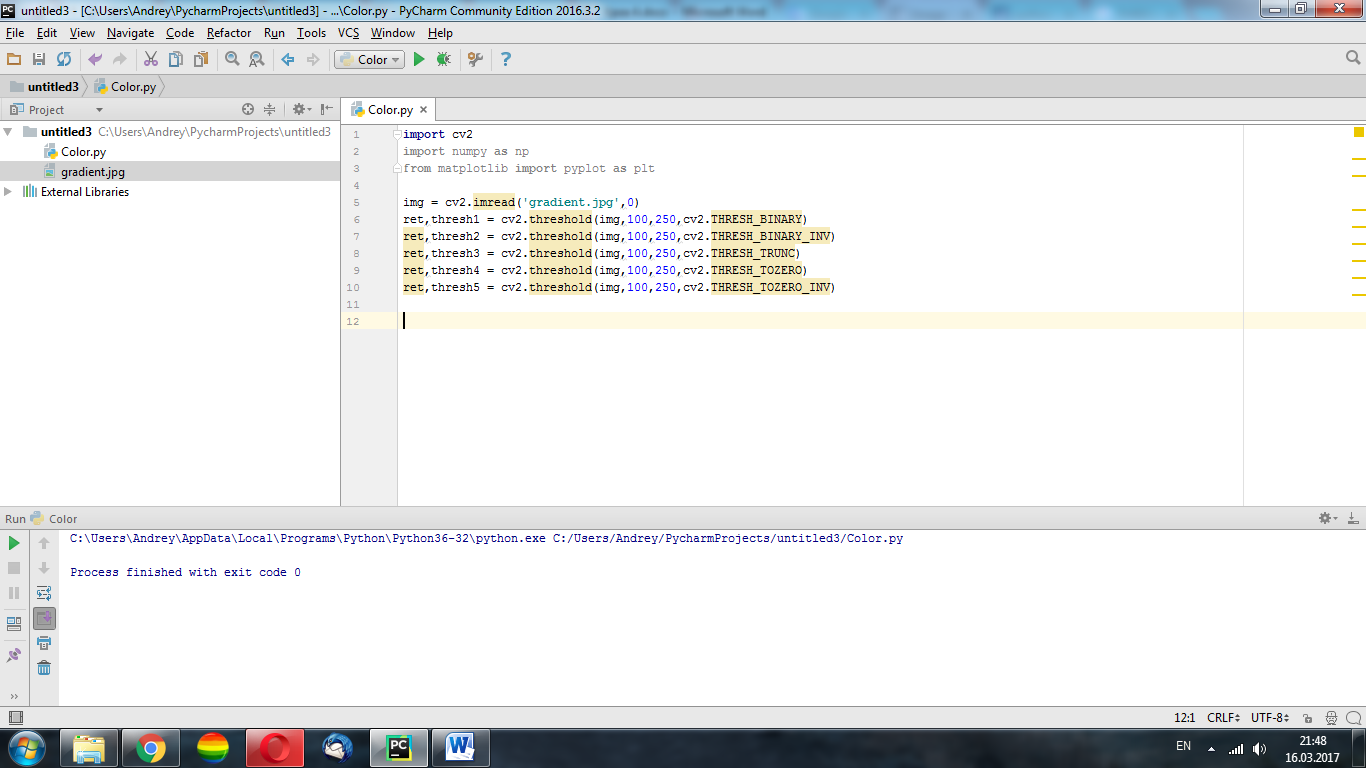
Функция thresh tozero позволяет нам сделать все, что темнее нашего порогового значения станет черным

ret,thresh4 = cv2.threshold(img,100,250,cv2.THRESH\_TOZERO)



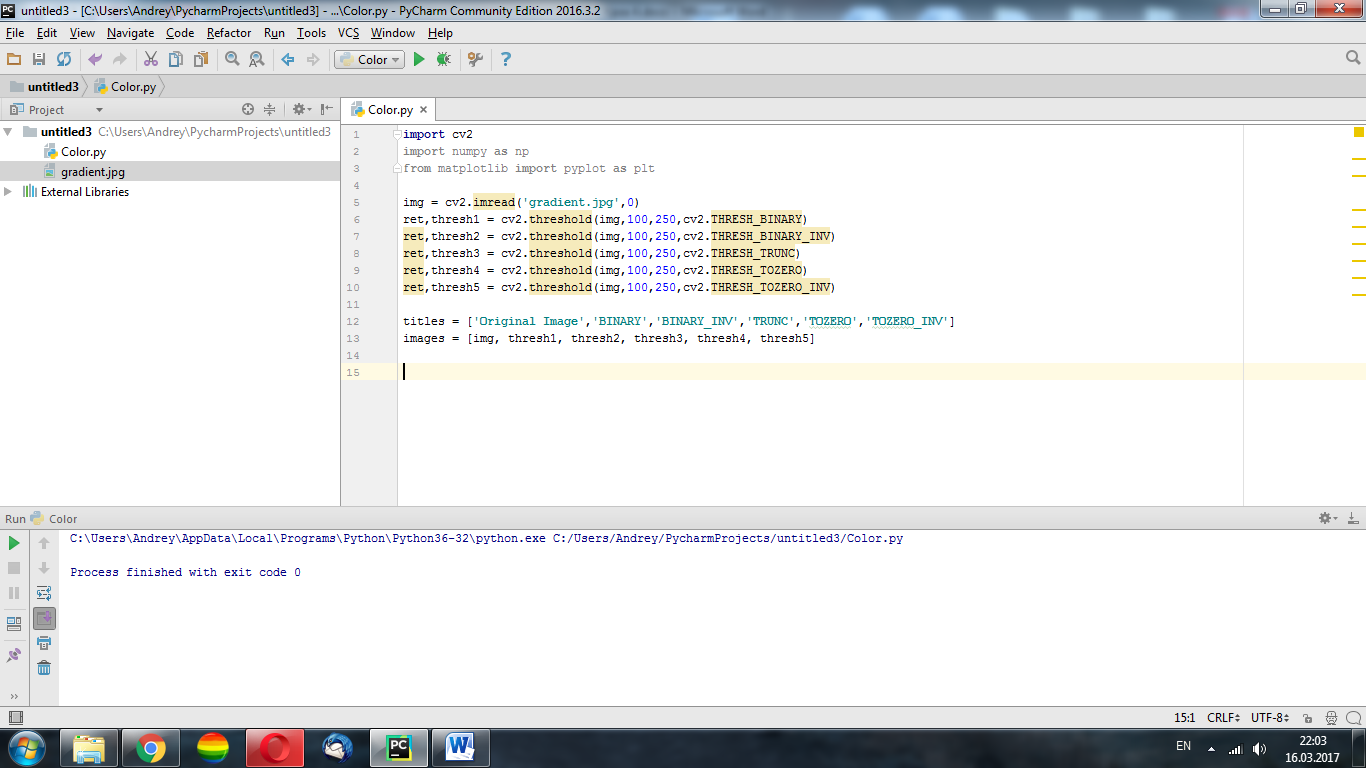
Функция thresh tozero inv инвертирует условие функции thresh tozero .

ret,thresh5 = cv2.threshold(img,100,250,cv2.THRESH\_TOZERO\_INV)



Зададим массивы, которые будут хранить заголовки и сами картинки, которые мы изменяли.

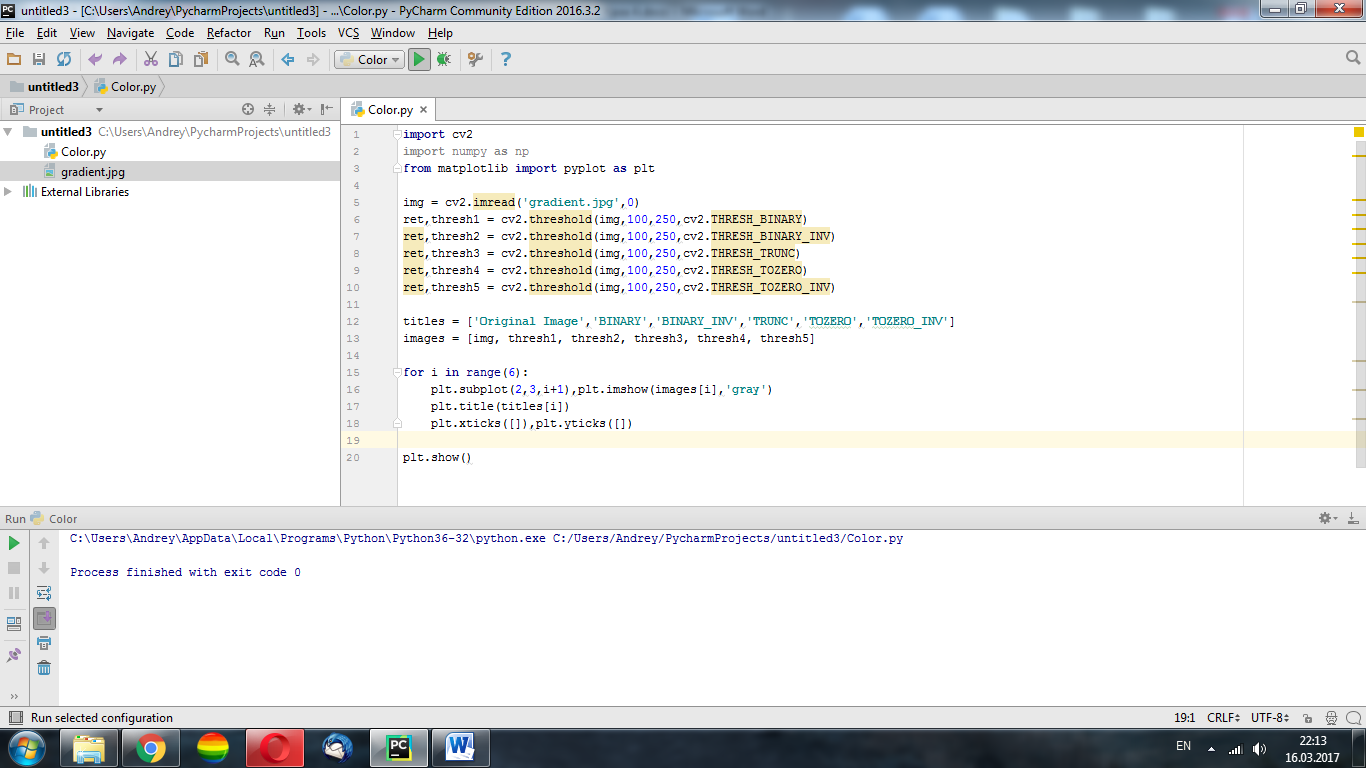
titles = [**'Original Image'**,**'BINARY'**,**'BINARY\_INV'**,**'TRUNC'**,**'TOZERO'**,**'TOZERO\_INV'**]  
images = [img, thresh1, thresh2, thresh3, thresh4, thresh5]



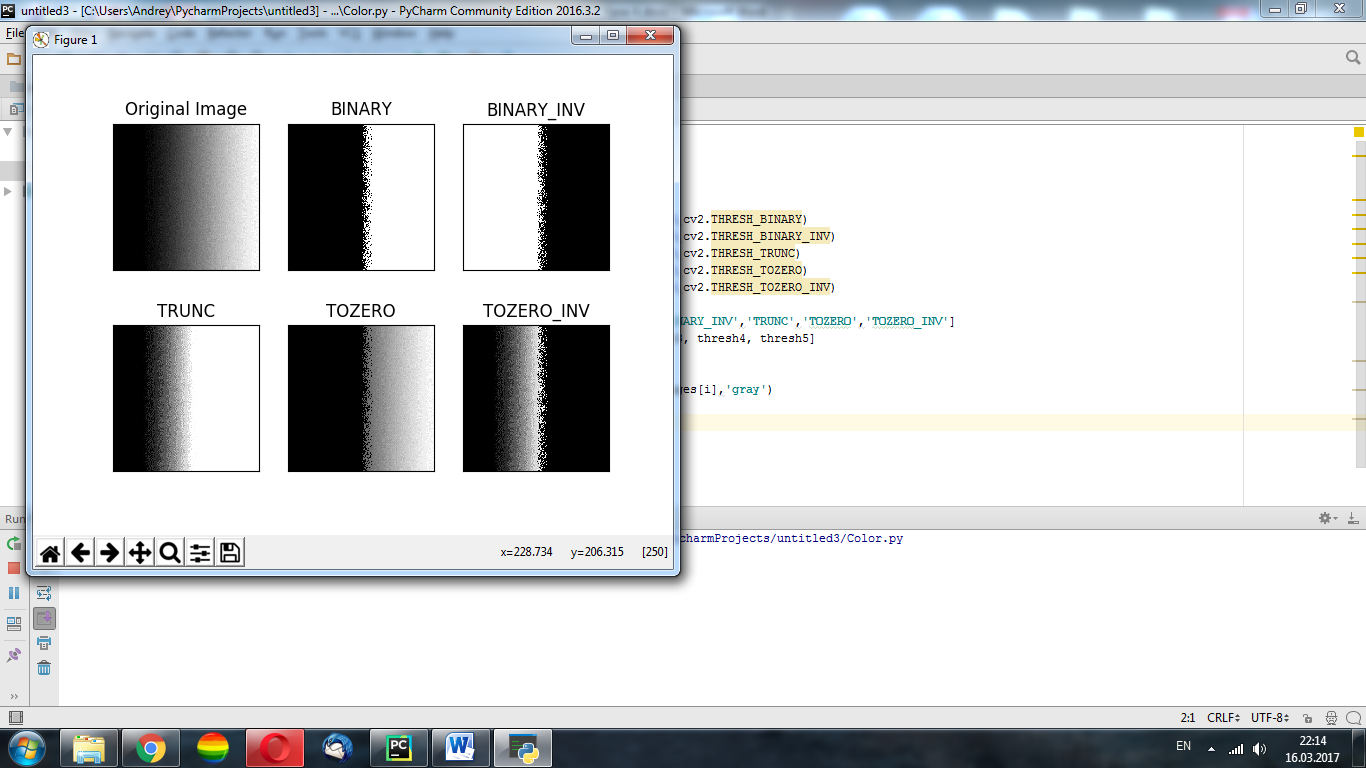
Далее выведем через цикл картинки в две строки и три столбца начиная с первой и повторить 6 раз. И покажем картинку с массива images в сером цвете. Далее выводим заголовок из массива titles. Покажем текущие координаты в нашем окне, а также цвет в том месте где находится курсор.

Выведем окно программы с помощью функции plr.show

**for** i **in** range(6):  
 plt.subplot(2,3,i+1),plt.imshow(images[i],**'gray'**)  
 plt.title(titles[i])  
 plt.xticks([]),plt.yticks([])  
  
plt.show()



Запустите программу



Адаптивное пороговое значение

До этого мы использовали глобальное значение как пороговое значение. Но это не подойдет, если например: изображение имеет разные условия освещения в разных областях. В этом случае мы будем использовать адаптивное пороговое значение. При этом алгоритм вычисляет порог для небольших областей изображения. Таким образом, мы получаем различные пороги для разных областей одного и того же изображения, и это дает лучшие результаты для изображений с различным освещением.

Три «специальных» входных параметра и только один выходной аргумент:

Адаптивный метод. Он определяет, как рассчитывается пороговое значение.

Cv2.ADAPTIVE\_THRESH\_MEAN\_C: пороговое значение - среднее значение для окрестности.

Cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C: пороговое значение представляет собой взвешенную сумму значений окрестности, где результат представляет собой гауссовское окно.

Размер блока - он определяет размер окрестности.

C - это просто константа, которая вычитается из среднего или взвешенного среднего.

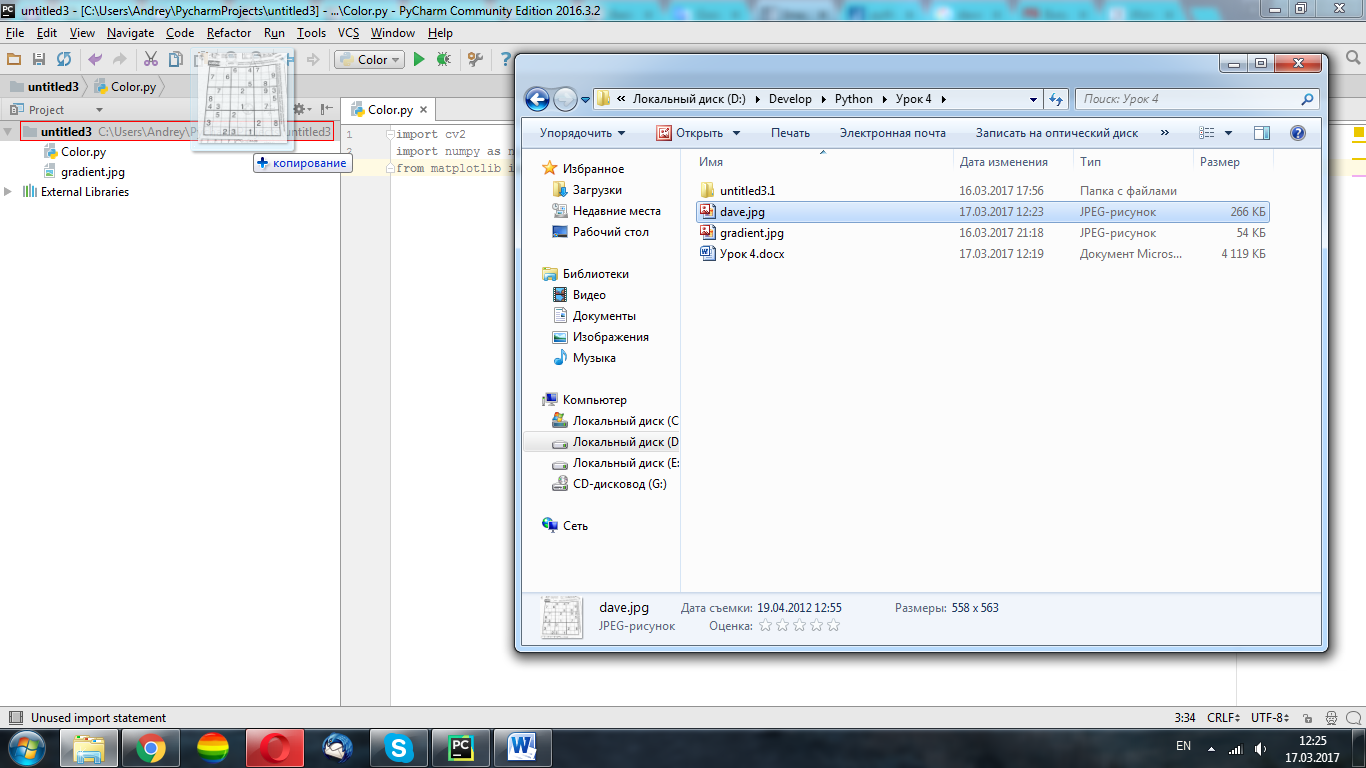
Начнем с подключения библиотек cv2, numpy и pyplot

**import** cv2

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

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

Загрузим картинку dave.jpg

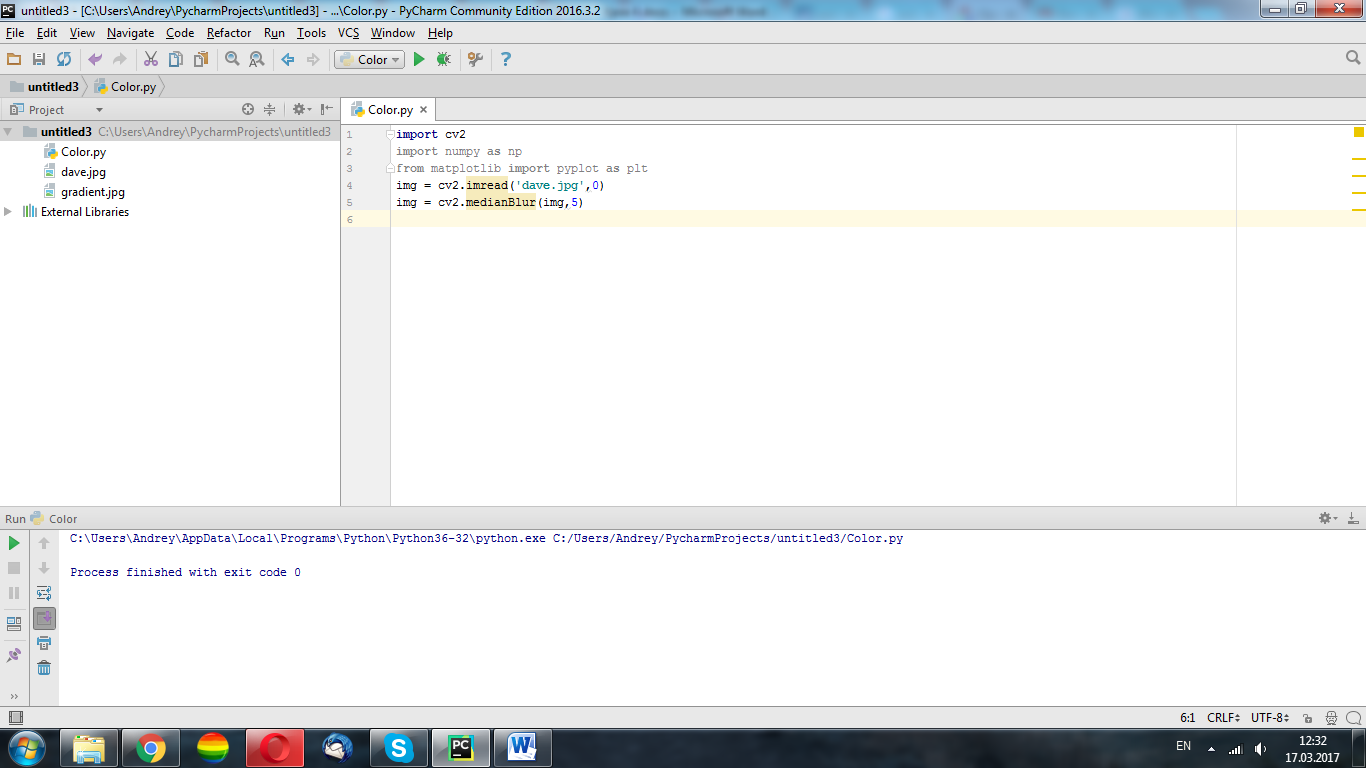


Зададим значение переменной img для чтения нашей картинки в сером цвете.

И с помощью функции cv2.medianBlur уберем лишний шум с картинки

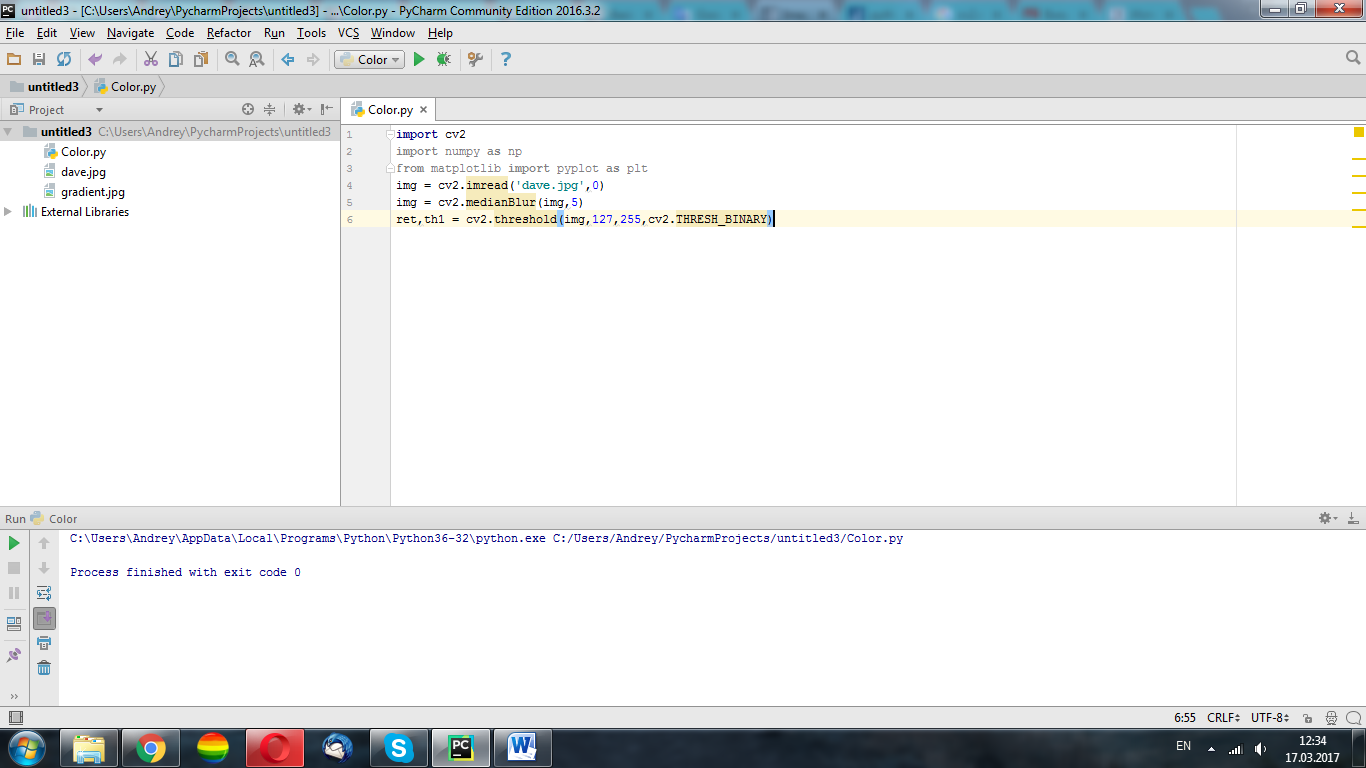
img = cv2.imread(**'gradient.jpg'**,0)

img = cv2.medianBlur(img,5)



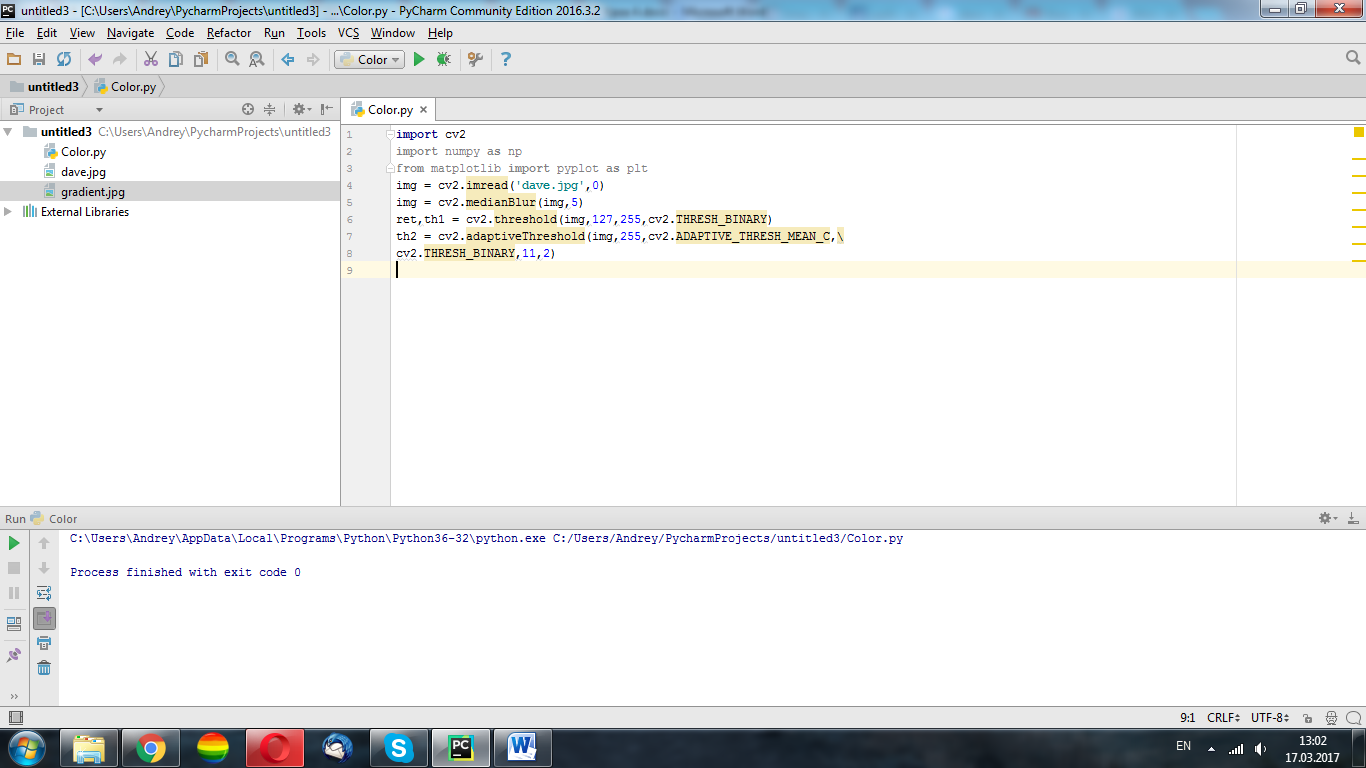
В переменную th1 записываем координаты выбранной точки на картинке и все что светлее этой точки будет белым, а все что темнее этой точки станет черным.

ret,th1 **=** cv2**.**threshold(img,127,255,cv2**.**THRESH\_BINARY)



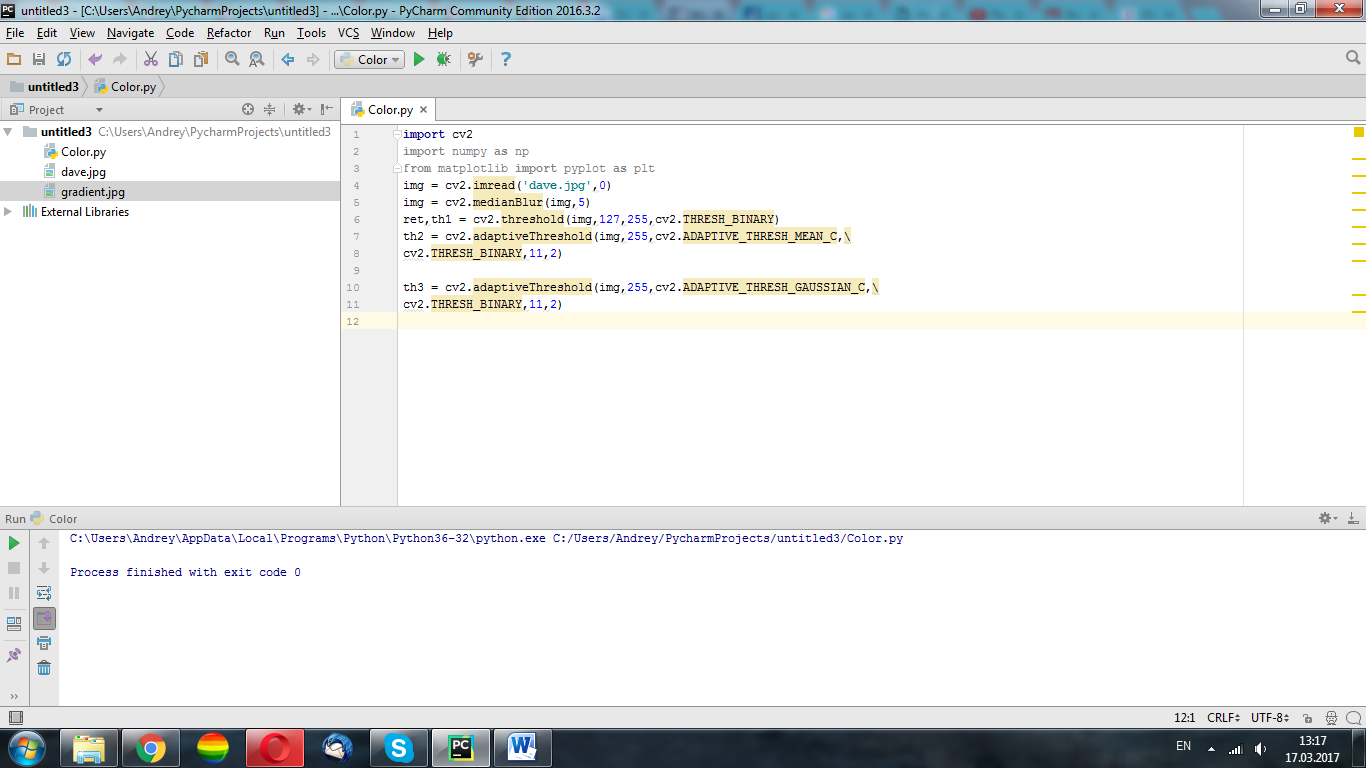
В переменную th2 используя функцию cv2.adaptiveTheshold записываем координаты порогового значения и среднее значение для окружающей области.

th2 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE\_THRESH\_MEAN\_C,\  
cv2.THRESH\_BINARY,11,2)



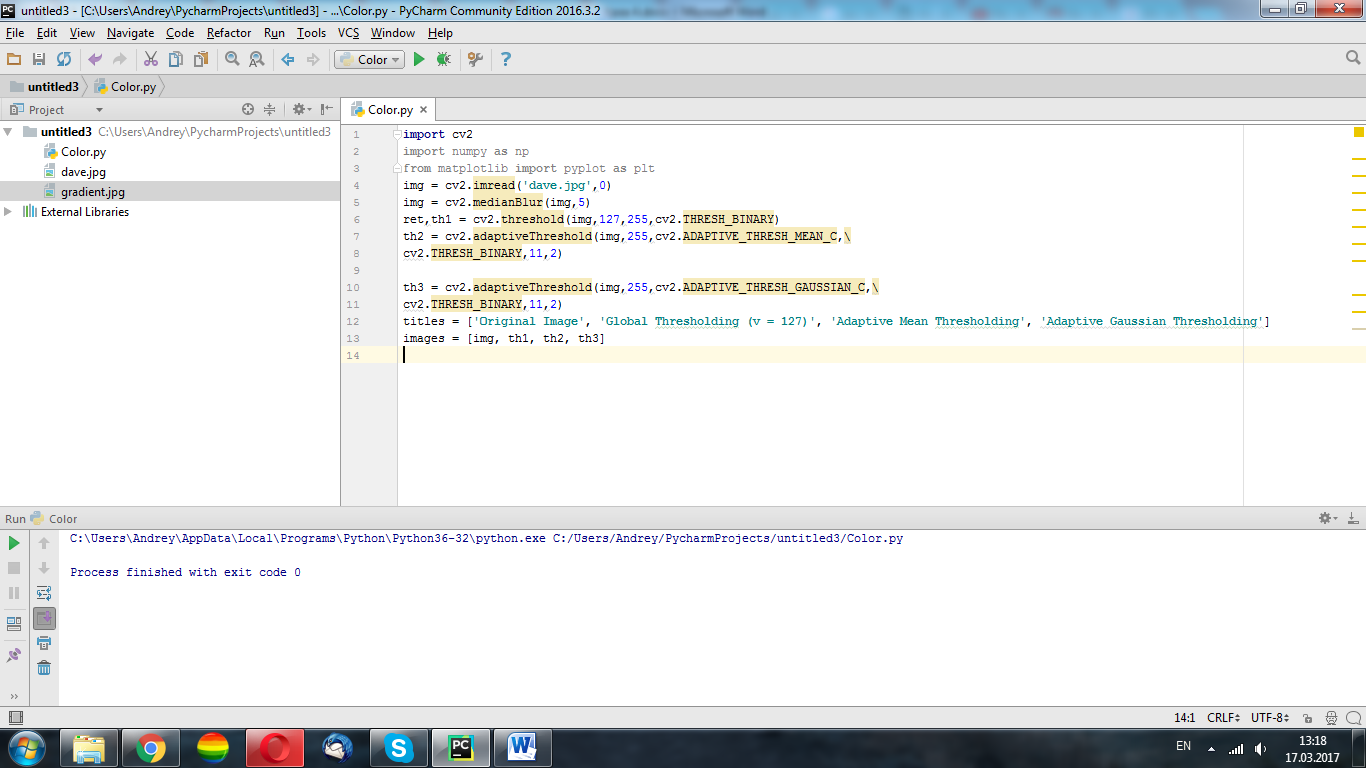
В переменную th3 используя функцию cv2.adaptiveTheshold записываем координаты порогового значения в виде суммы значений окрестности, где эта сумма представляет собой гаусовское окно

th3 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C,\  
cv2.THRESH\_BINARY,11,2)



Зададим массивы, которые будут хранить заголовки и сами картинки, которые мы изменяли.

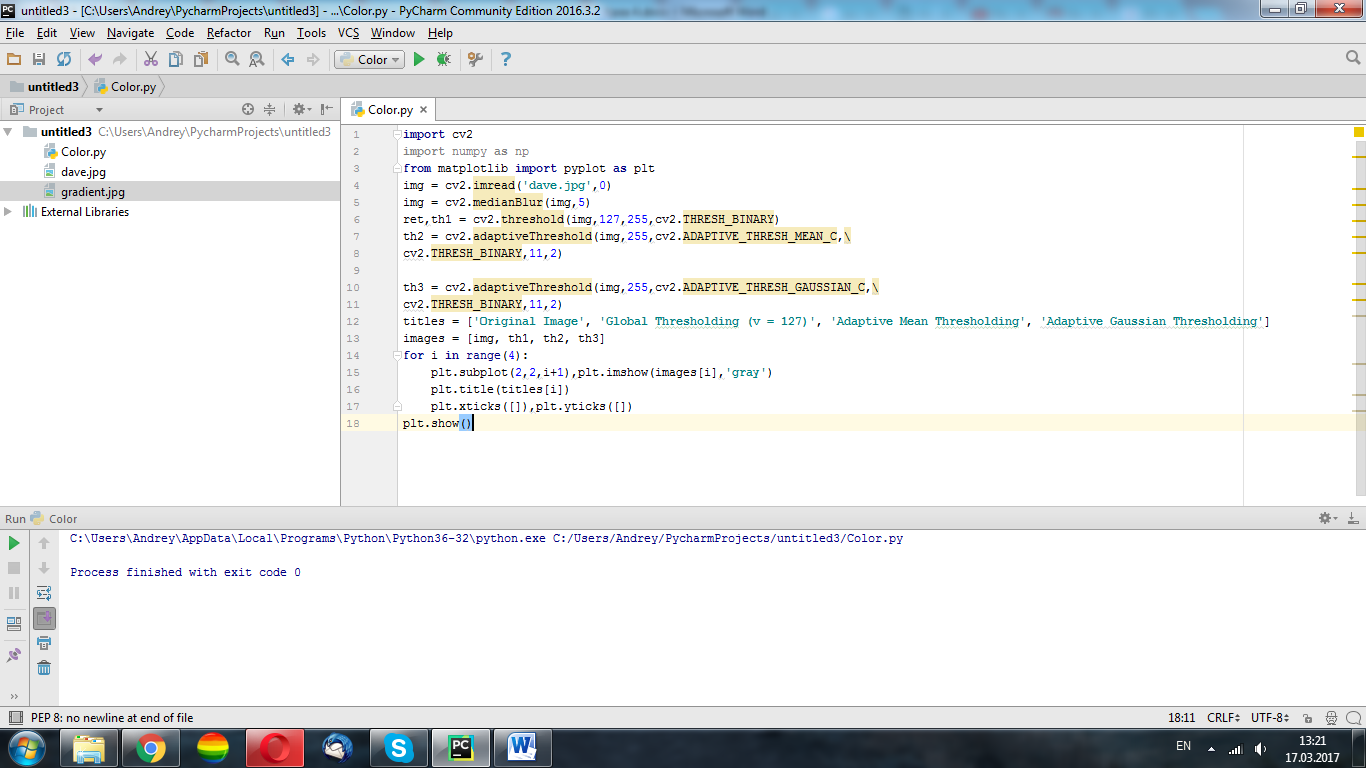
titles = [**'Original Image'**, **'Global Thresholding (v = 127)'**, **'Adaptive Mean Thresholding'**, **'Adaptive Gaussian Thresholding'**]  
images = [img, th1, th2, th3]



И выведем через цикл картинки в две строки и два столбца начиная с первой и повторить 4 раза. Покажем картинку с массива images в сером цвете. Далее выводим заголовок из массива titles. Добавим просмотр текущих координат в нашем окне, а также цвет в том месте где находится курсор.

Выведем окно программы с помощью функции plr.show

**for** i **in** range(4):  
 plt.subplot(2,2,i+1),plt.imshow(images[i],**'gray'**)  
 plt.title(titles[i])  
 plt.xticks([]),plt.yticks([])  
plt.show()



Запустите программу

