

```
In [1]: from keras.preprocessing.image import ImageDataGenerator

Arguments for ImageDataGenerator class

In [2]: train_datagen=ImageDataGenerator(rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True)
text_datagen=ImageDataGenerator(rescale=1./255)

Applying ImageDataGenerator functionality to trainset and testset

In [3]: from tensorflow.keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rescale= 1./255, horizontal_flip = True, vertical_flip = True, zoom_range = 0.2)
text_datagen = ImageDataGenerator(rescale= 1./255)

In [5]: x_train = train_datagen.flow_from_directory("/content/drive/MyDrive/1bm project/RAIN_SET", target_size = (64,64),
class_mode = "categorical", batch_size = 24)

Found 4118 images belonging to 5 classes.

In [6]: x_test = test_datagen.flow_from_directory("/content/drive/MyDrive/1bm project/TEST_SET", target_size = (64,64),
class_mode = "categorical", batch_size = 24)

Found 929 images belonging to 3 classes.

In [7]: # !pip install opencv.python

In [8]: import cv2

In [9]: # imread is used to read the image

In [10]: img = cv2.imread("/content/drive/MyDrive/1bm project/TEST_SET/APPLES/n07740461_1191.jpg")

In [11]: img

Out[11]: array([[174, 188, 207],
[173, 187, 206],
[171, 185, 204],
...,
[181, 192, 206],
[180, 192, 204],
[179, 191, 203]],

[[175, 189, 208],
[174, 188, 207],
[174, 188, 207],
...,
[182, 193, 207],
[182, 193, 207],
[181, 193, 205]],

[[178, 192, 211],
[177, 191, 210],
[177, 191, 210],
...,
[184, 195, 209],
[184, 195, 209],
[184, 195, 209]],

...,

[[161, 185, 209],
[164, 188, 212],
[163, 191, 215],
...,
[184, 198, 216],
[186, 200, 218],
[187, 201, 220]],

[[157, 185, 209],
[158, 186, 210],
[156, 187, 210],
...,
[185, 199, 217],
[187, 201, 219],
[187, 201, 220]],

[[154, 186, 209],
[153, 185, 208],
[150, 182, 205],
...,
[187, 199, 217],
[188, 202, 221],
[189, 203, 222]]], dtype=uint8)

In [12]: img.ndim

Out[12]: 3

In [13]: type(img)

Out[13]: numpy.ndarray

In [14]: img.shape

Out[14]: (256, 256, 3)

In [15]: #flag 1 means color image

In [16]: img_flag = cv2.imread("/content/drive/MyDrive/1bm project/TEST_SET/APPLES/n07740461_1191.jpg")

In [17]: img_flag

Out[17]: array([[174, 188, 207],
[173, 187, 206],
[171, 185, 204],
...,
[181, 192, 206],
[180, 192, 204],
[179, 191, 203]],

[[175, 189, 208],
[174, 188, 207],
[174, 188, 207],
...,
[182, 193, 207],
[182, 193, 207],
[181, 193, 205]],

[[178, 192, 211],
[177, 191, 210],
[177, 191, 210],
...,
[184, 195, 209],
[184, 195, 209],
[184, 195, 209]],

...,

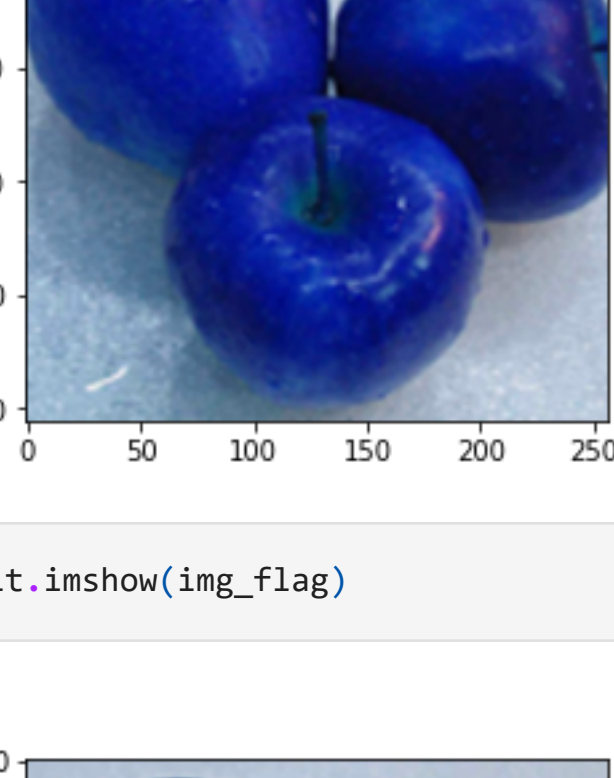
[[161, 185, 209],
[164, 188, 212],
[163, 191, 215],
...,
[184, 198, 216],
[186, 200, 218],
[187, 201, 220]],

[[157, 185, 209],
[158, 186, 210],
[156, 187, 210],
...,
[185, 199, 217],
[187, 201, 219],
[187, 201, 220]],


[[154, 186, 209],
[153, 185, 208],
[150, 182, 205],
...,
[187, 199, 217],
[188, 202, 221],
[189, 203, 222]]], dtype=uint8)

In [18]: import matplotlib.pyplot as plt

In [19]: plt.imshow(img)

Out[19]: 

In [20]: plt.imshow(img_flag)

Out[20]: 


In [21]: #resize the image

In [22]: resized_img = cv2.resize(img, (100,100))

In [23]: resized_img.shape

Out[23]: (100, 100, 3)

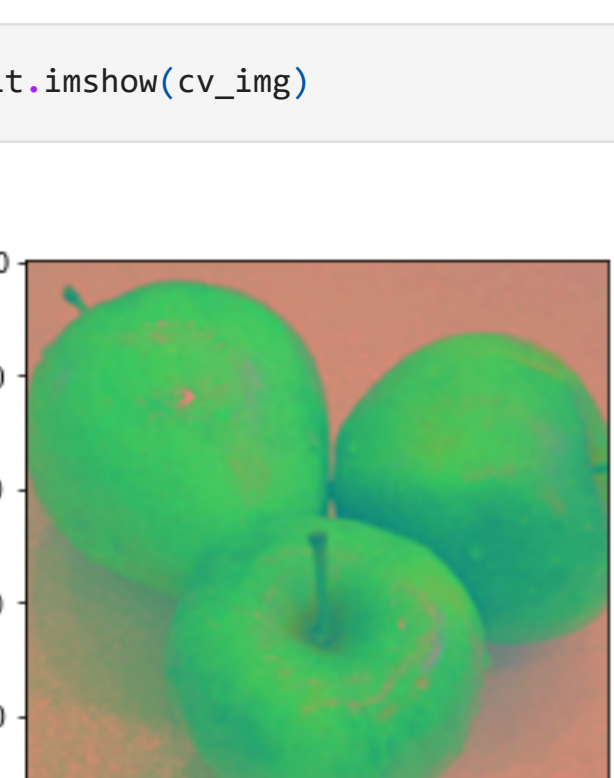
In [24]: plt.imshow(resized_img)

Out[24]: 

Covert color

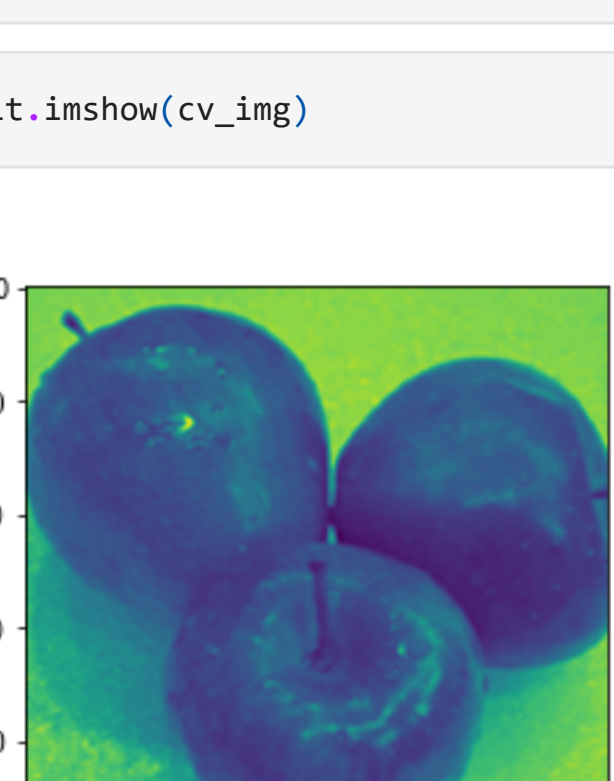
In [25]: cv_img = cv2.cvtColor(img, cv2.COLOR_BGR2YCR_CB)

In [27]: plt.imshow(cv_img)

Out[27]: 

In [28]: cv_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

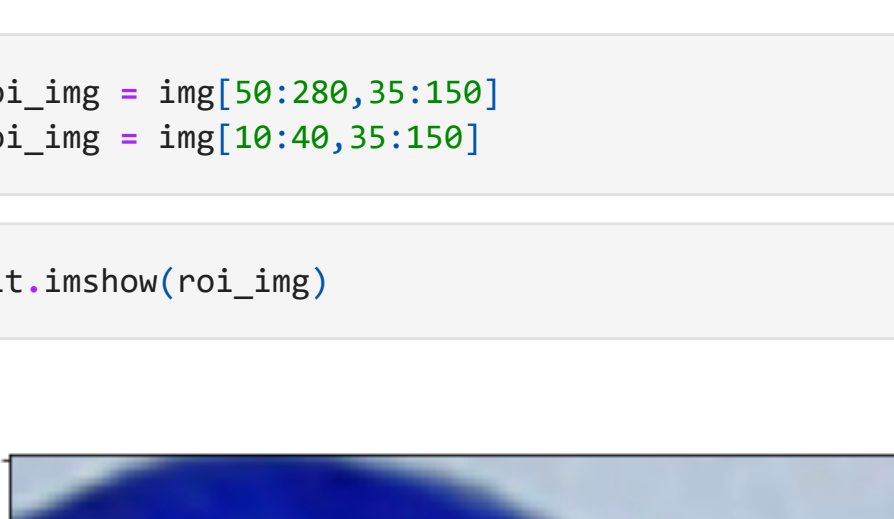
In [29]: plt.imshow(cv_img)

Out[29]: 

Roi or crop of image

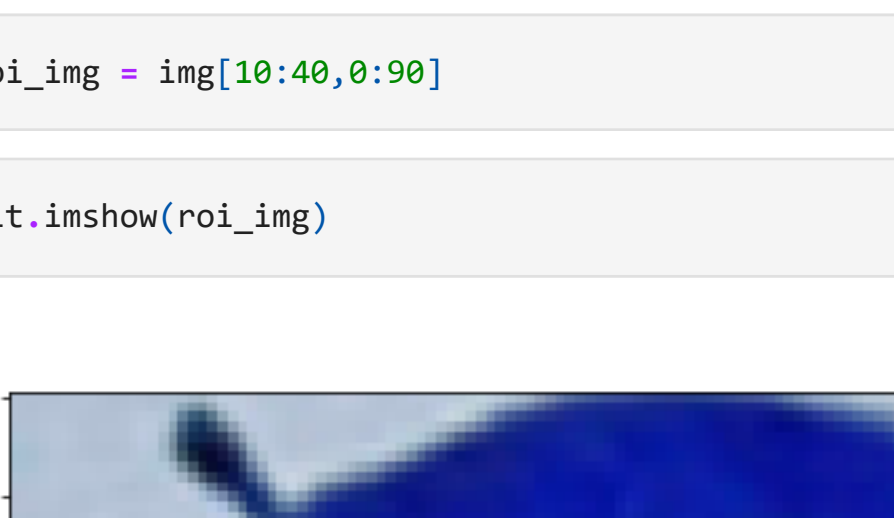
In [30]: roi_img = img[50:200, 35:150]
roi_img = img[10:40, 35:150]

In [32]: plt.imshow(roi_img)

Out[32]: 


In [33]: roi_img = img[10:40, 0:90]

In [34]: plt.imshow(roi_img)

Out[34]: 

In [35]: roi_img = img[0:90, 10:40]

In [36]: plt.imshow(roi_img)

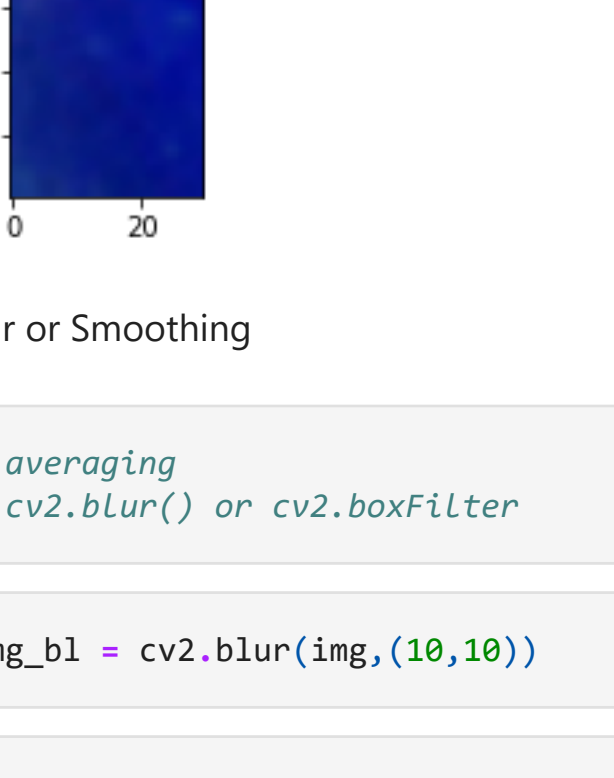
Out[36]: 

Blur or Smoothing

In [37]: # averaging
# cv2.blur() or cv2.boxFilter

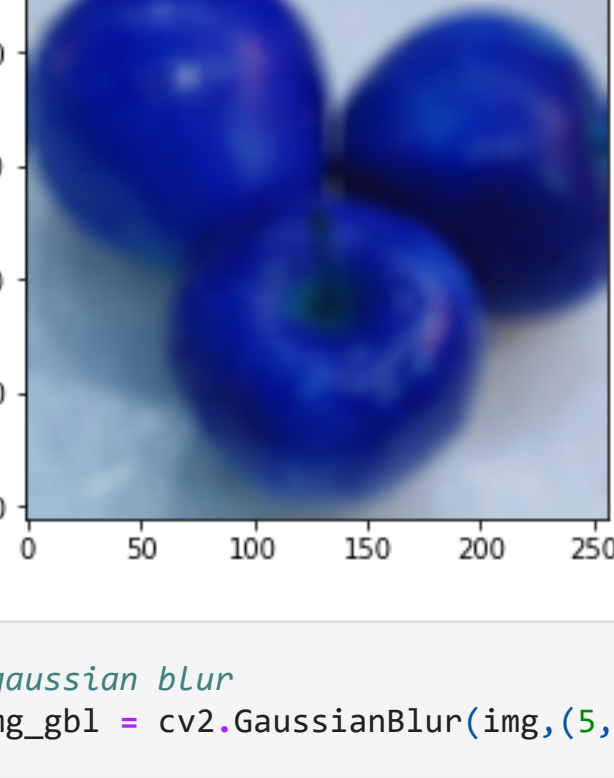
In [38]: img_bl = cv2.blur(img, (10,10))

In [39]: plt.imshow(img_bl)

Out[39]: 

In [40]: #gaussian blur
img_gbl = cv2.GaussianBlur(img, (5,5), 0)

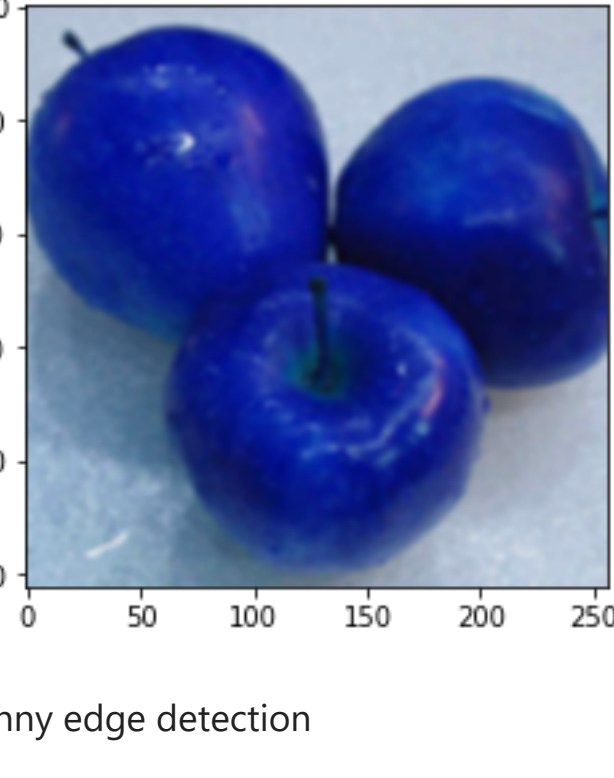
In [41]: plt.imshow(img_gbl)

Out[41]: 

Canny edge detection

In [43]: img_edge = cv2.Canny(img, 230, 350)

In [44]: plt.imshow(img_edge)

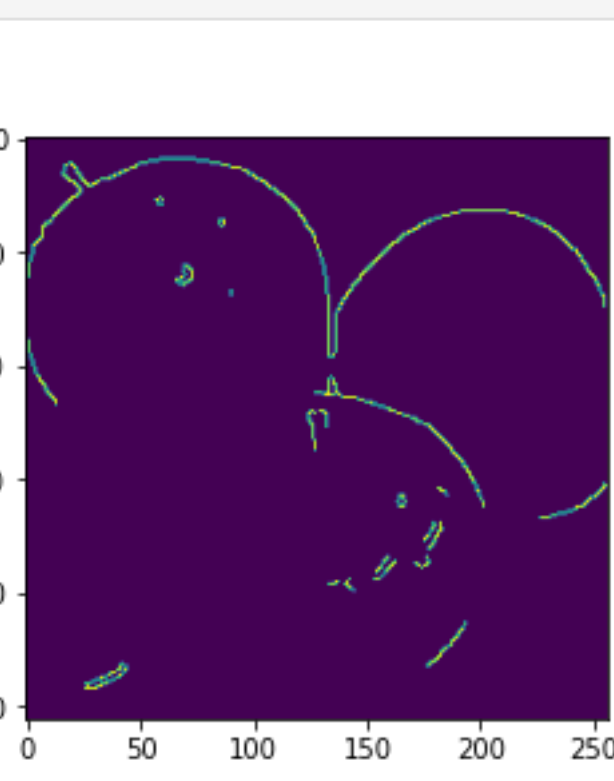
Out[44]: 

Thershold

In [45]: #binary

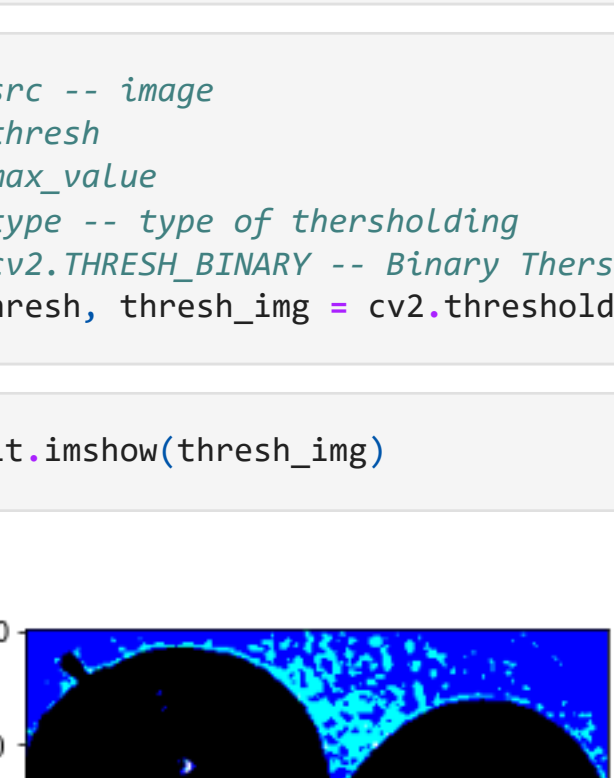
In [46]: #src -- image
#thresh
#max value
#type -- type of thresholding
cv2.THRESH_BINARY -- Binary Thresholding
thresh, thresh_img = cv2.threshold(img, 200, 255, cv2.THRESH_BINARY) #img

In [47]: plt.imshow(thresh_img)

Out[47]: 

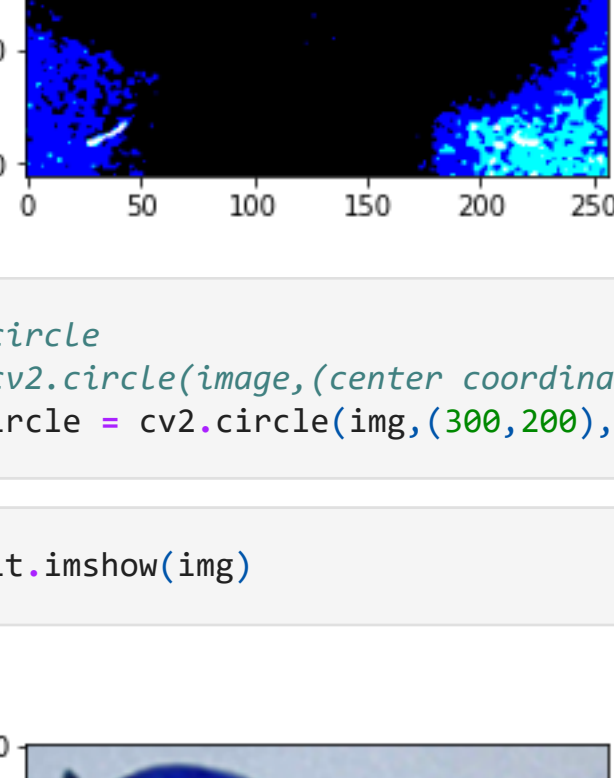
In [48]: #circle
#cv2.circle(image, (center coordinates), radius, (color), thickness)
circle = cv2.circle(img, (300, 200), 60, (255, 0, 0), 5)

In [49]: plt.imshow(img)

Out[49]: 

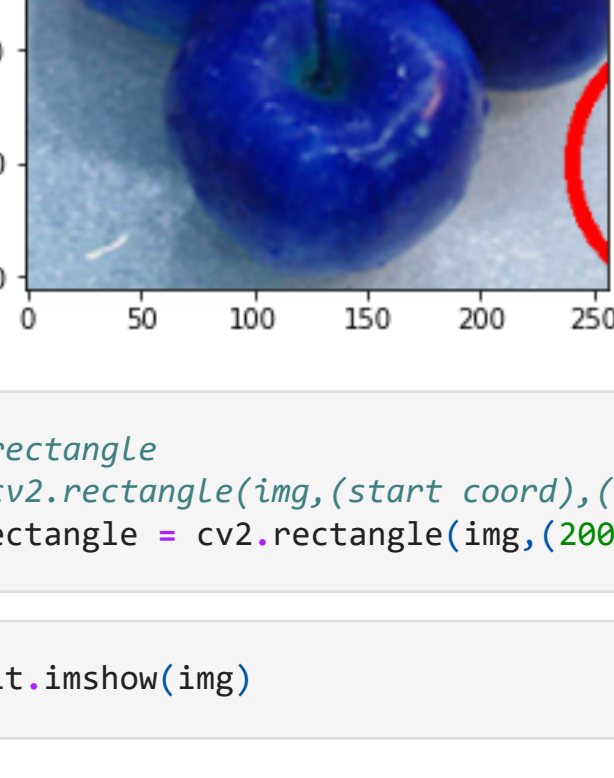
In [50]: #rectangle
#cv2.rectangle(image, (start coord), (end coord), color, thickness)
rectangle = cv2.rectangle(img, (200, 100), (400, 300), (0, 0, 255), 10)

In [51]: plt.imshow(img)

Out[51]: 

In [52]: #line
#cv2.line(image, (start coord), (end coord), color, thickness)
line = cv2.line(img, (200, 100), (400, 300), (0, 255, 0), 3)

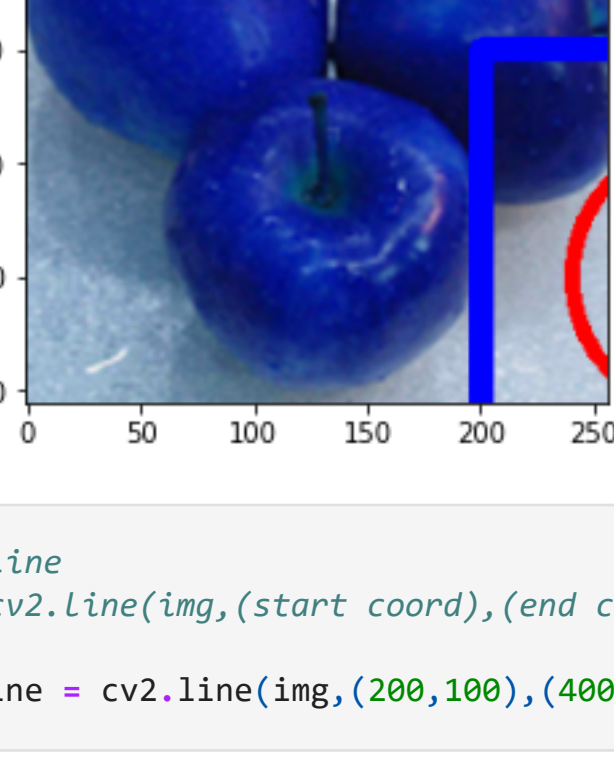
In [53]: plt.imshow(img)

Out[53]: 

In [54]: #creating or writing text an image

In [55]: #cv2.putText(image, text, (coord), fontstyle, fontsize, color, thickness)
text = cv2.putText(img, "opencv", (200, 50), cv2.FONT_HERSHEY_SIMPLEX, 2, (255, 255, 255), 5)

In [56]: plt.imshow(img)

Out[56]: 
```