

Computer Vision Final Project Program Steps for Execution:

There are basically two python files which has our code namely yolo.py and yoloDetection.py

Yolo.py code mainly contains taking image as input and how it processes the input and where the output will be stored. Along with that how video is taken as input and how frames are calculated and objects in the frame's detection and output video and image where it is stored after execution will be given in yolo.py python file

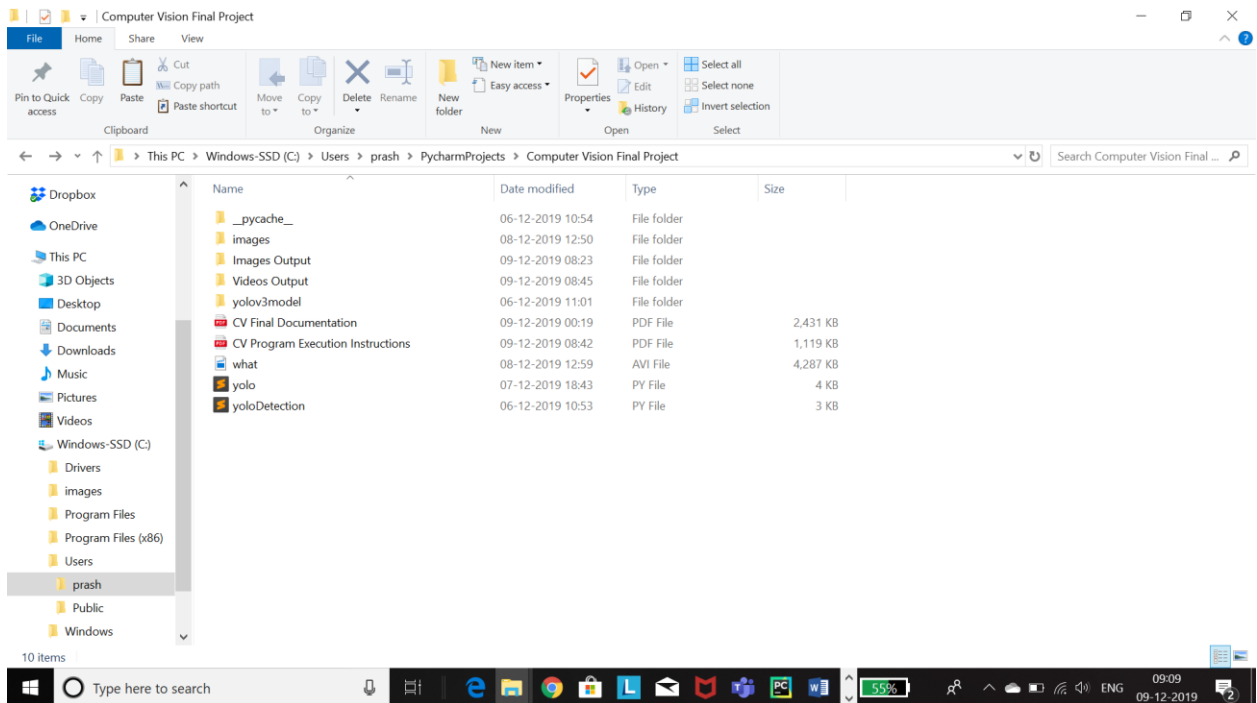
Yolodetection.py python file contains how bounding boxes are created and labels generation and bounding boxes names are stored in list, and finally how image is represented with bounding box all this information is executed through this file.

Yolo.py code needs to be executed to get the output. The output gets stored in the file itself as **what.avi** because we gave that name in the code. You can check any new inputs as you wish by

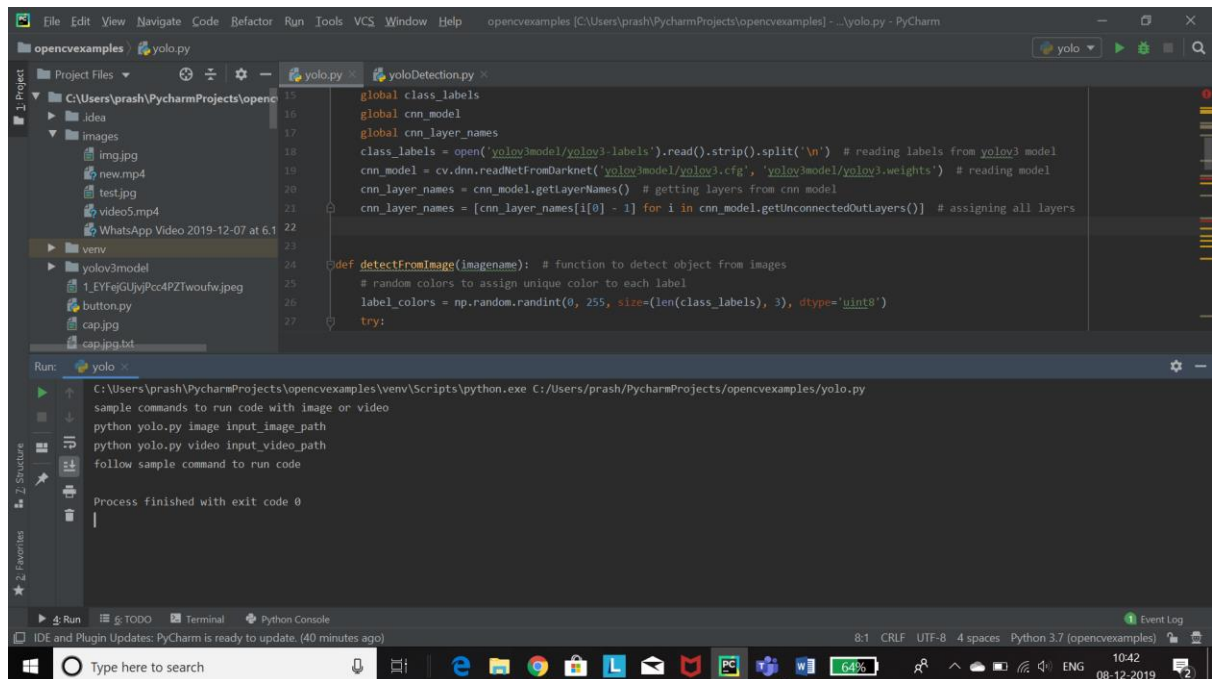
Download an image or video (less than 3 mb) and place these images and video under the images folder.

Steps to execute the program and to get the output:

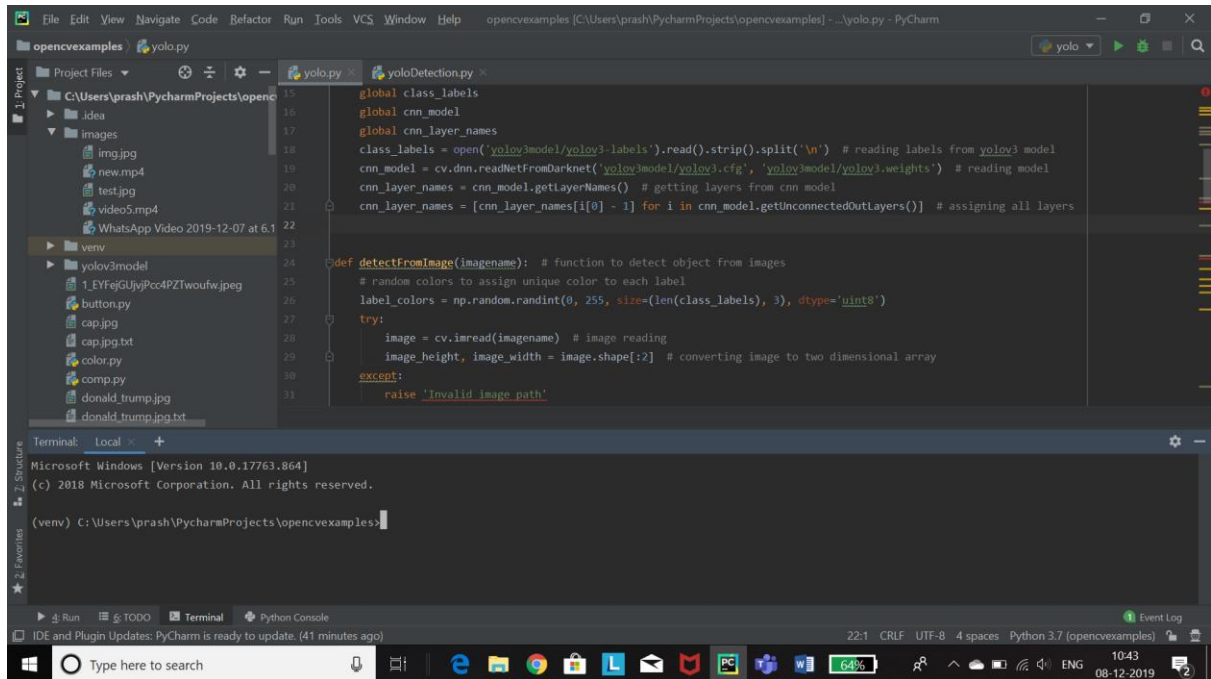
- 1.) Inorder to run this program, make sure we have all the required individual folders like _pycache, images, yolov3model, yolo.py and yoloDetection.py files in one single folder as shown in the below screenshot.



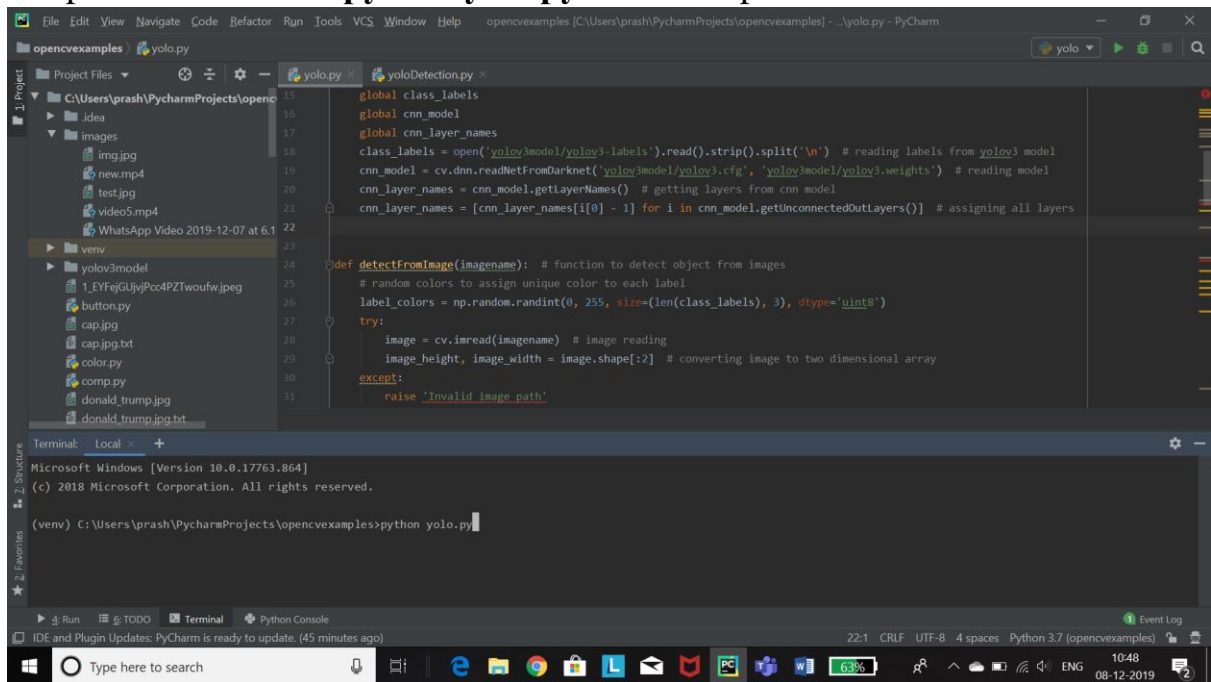
2.) In Step 2 We need to run the yolo.py file and make sure we get **“Process finished with exit code 0”** as shown in the below screenshot.



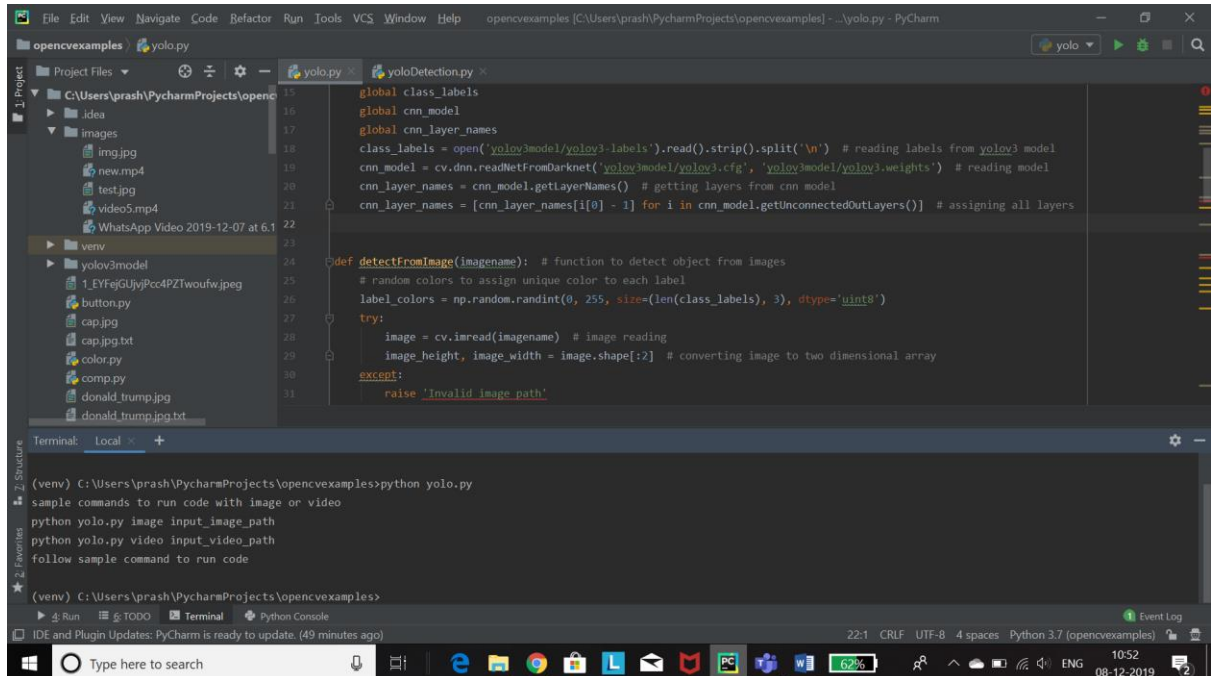
3.) In Step 3 need to go to Terminal or Command prompt as shown in the below screenshot:



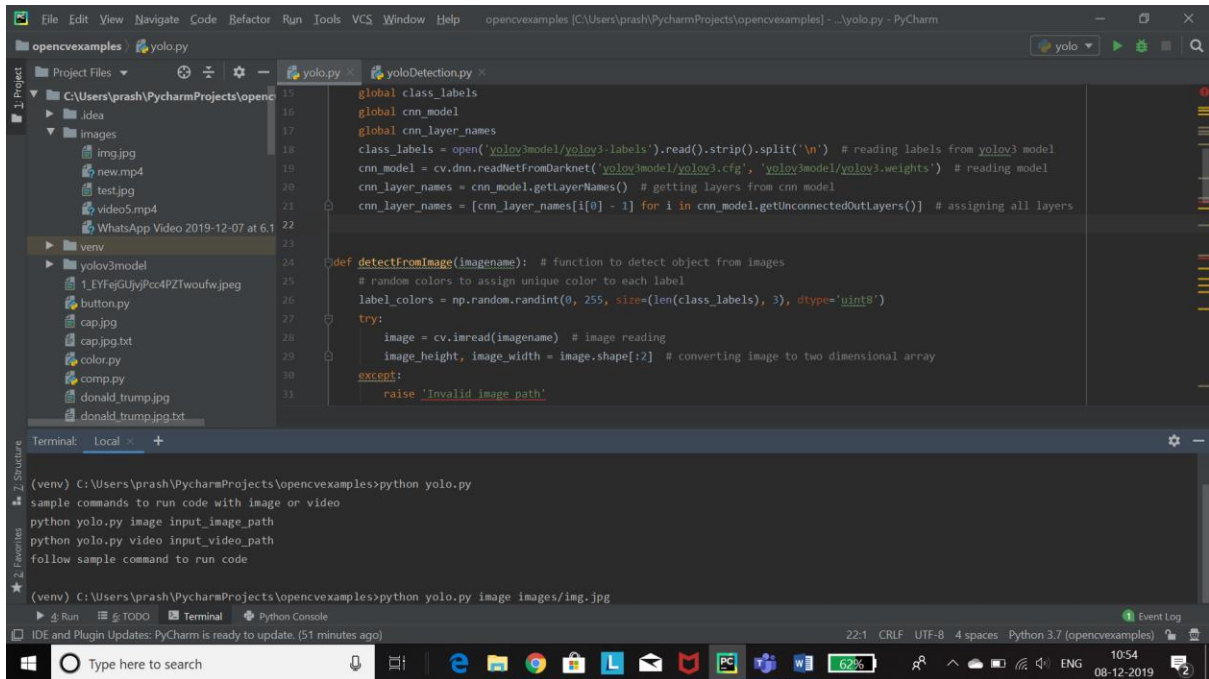
4.) In Step 4 need to enter “python yolo.py” after the path



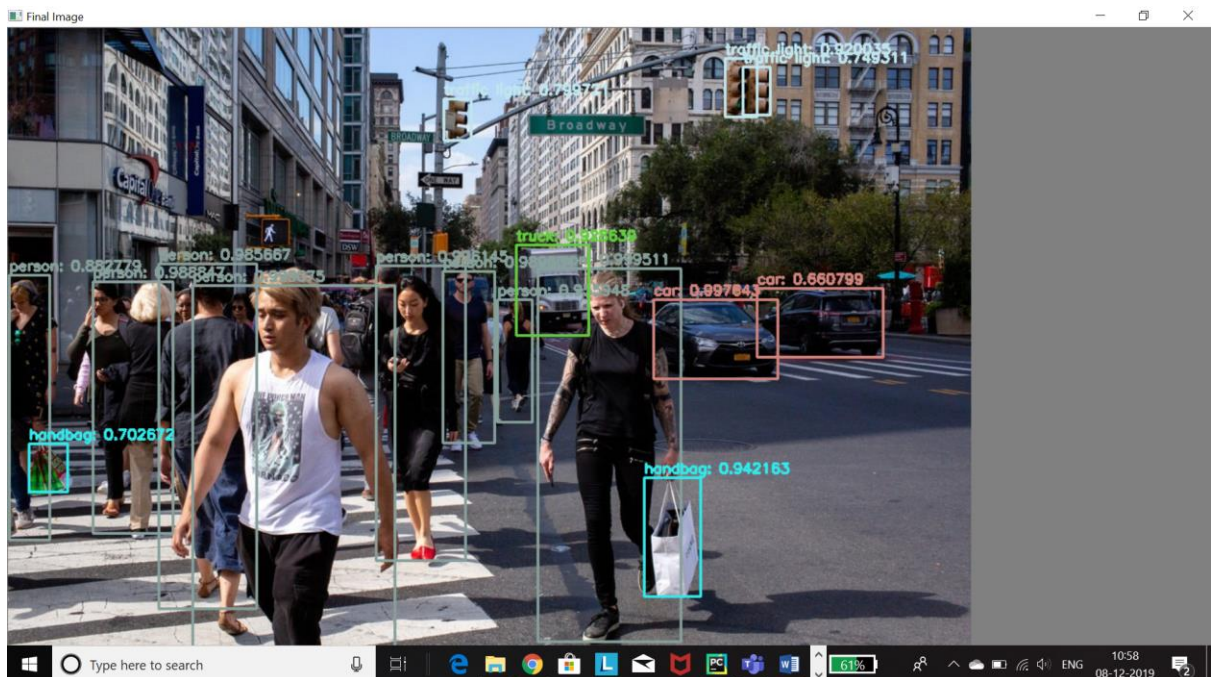
5.) In Step 5 after running the command, we will get two options in the terminal. One option is for identifying the objects in the image and another option for identifying the objects in the video as shown in the below screenshot



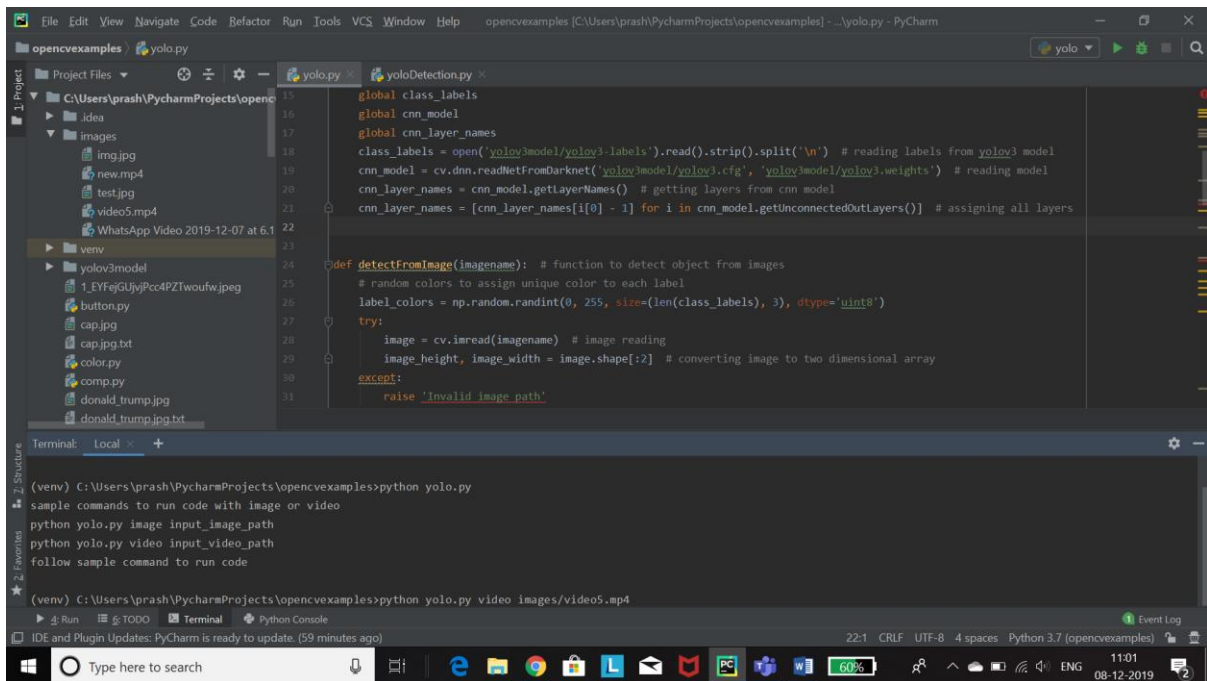
6.) In Step 6 Next we need to identify the objects present in the image, we need to pass the path of the sample images that are presented in the path. So, we give the path “**python yolo.py image images/img.jpg**”. This command python is the software name and `yolo.py` is the program name and ‘image’ means we want application to detect object from image and ‘images/img.jpg’ is the input image. Similarly, for videos instead of image we need to pass video with video path.



7.) In step 7 we need to run the command which we have given in the step5. Then we will get the below output where all the objects in the given image will be displayed as shown in the below screenshot.



8.) In step 8 we need to identify the objects that are present in the video, So We need to pass the path of the sample video that are presented in the path. So, we give the path “**python yolo.py video images/video5.mp4**” as shown in the below screenshot. When we are giving video then it will take time to extract all frames from video for object detection and then create a new video called ‘what.avi’ in the same code folder. If it’s taking long time, then you can press CTRL+C to stop execution and then you can play ‘**what.avi**’ file.



The screenshot shows the PyCharm IDE interface. The left sidebar displays the project structure with folders like 'images' and 'venv'. The main editor window shows the code for 'yolo.py', which includes global variables for class_labels, cnn_model, and cnn_layer_names, and a function 'detectFromImage' for object detection. The terminal at the bottom shows the command 'python yolo.py video images/video5.mp4' being executed in a virtual environment.

```
global class_labels
global cnn_model
global cnn_layer_names
class_labels = open('yolov3model/yolov3-labels').read().strip().split('\n') # reading labels from yolov3 model
cnn_model = cv.dnn.readNetFromDarknet('yolov3model/yolov3.cfg', 'yolov3model/yolov3.weights') # reading model
cnn_layer_names = cnn_model.getLayerNames() # getting layers from cnn model
cnn_layer_names = [cnn_layer_names[i] - 1 for i in cnn_model.getUnconnectedOutLayers()] # assigning all layers

def detectFromImage(imageName): # function to detect object from images
    # random colors to assign unique color to each label
    label_colors = np.random.randint(0, 255, size=(len(class_labels), 3), dtype='uint8')
    try:
        image = cv.imread(imageName) # image reading
        image_height, image_width = image.shape[:2] # converting image to two dimensional array
    except:
        raise 'Invalid image path'
```

Terminal: Local +

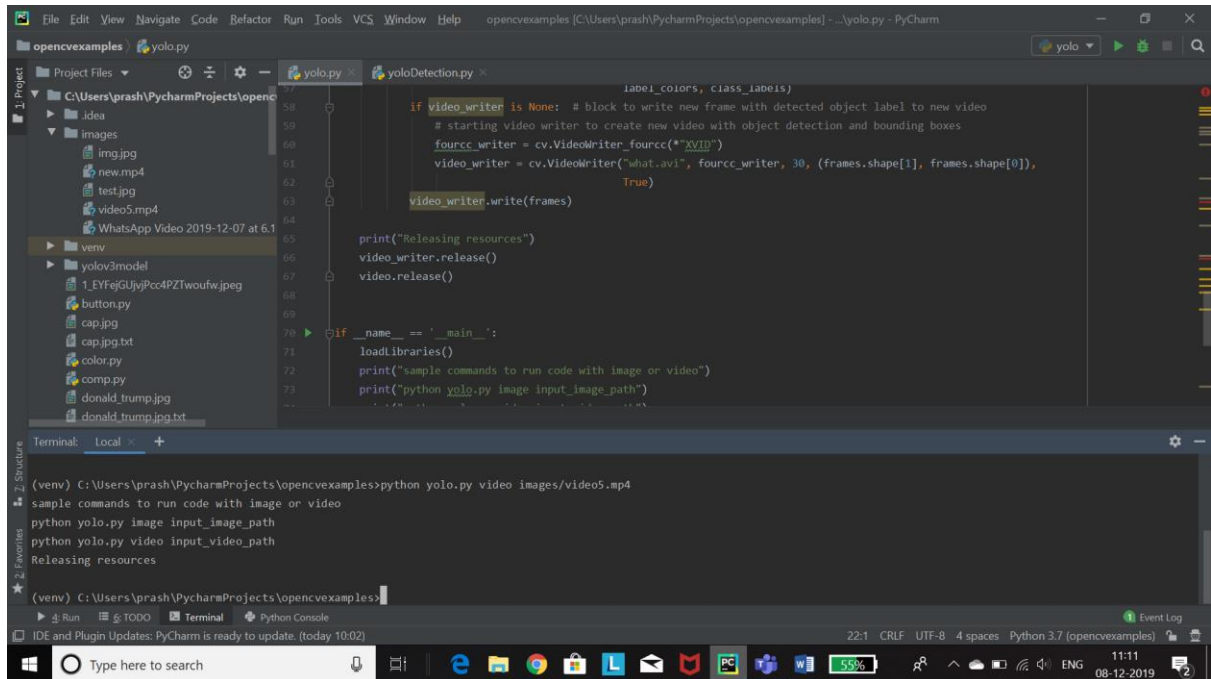
```
(venv) C:\Users\prash\PycharmProjects\opencvexamples>python yolo.py
sample commands to run code with image or video
python yolo.py image input_image_path
python yolo.py video input_video_path
follow sample command to run code

(venv) C:\Users\prash\PycharmProjects\opencvexamples>python yolo.py video images/video5.mp4
```

9.) In step 9 the execution will get starts and the given input will be executed and generates the output and save the output video in the name of “**what.avi**” name in the same folder. Then we need to go the folder and we can see the output where the given video will be played by detecting the objects that are presented in it for each frame.

Note: The input size should be less than the 3 MB. It should not be more than that size.

10.) In step 10 after the video execution gets finished all the resources will get released and the same message will be displayed on the terminal as shown in the below screenshot. Now the output video is ready for our view.



11.) In step 11 we need to go to that folder to see the output video with the name like “what.avi” we have given in the program. We must get to that path and we can see that output video.

