In [1]:
```python
import cv2 as cv
import numpy as np

# https://www.youtube.com/watch?v=R0hipZXJjlI&list=PLZBN9cDu0MSk4IFFnTOIDihvhnHWhAa8W
```

In [2]:
```python
#Write down conf, nms thresholds,inp width/height

# https://www.learnopencv.com/deep-learning-based-object-detection-using-yolov3-with-opencv-python-c/

confThreshold = 0.25  #Confidence threshold
nmsThreshold = 0.40  #Non-maximum suppression threshold
inpWidth = 416     #Width of network's input image
inpHeight = 416     #Height of network's input image


#Load names of classes and turn that into a list
classesFile = "coco.names"
classes = None
```

In [3]:
```python
with open(classesFile,'rt') as f:
    classes = f.read().rstrip('\n').split('\n')

#Model configuration
modelConf = 'yolov3.cfg'
modelWeights = 'yolov3.weights'

```

In [4]:

```
def drawPred(classId, conf, left, top, right, bottom):
    # Draw a bounding box.
    cv.rectangle(frame, (left, top), (right, bottom), (255, 178, 50), 3)

    label = '%.2f' % conf

    # Get the label for the class name and its confidence
    if classes:
        assert (classId < len(classes))
        label = '%s:%s' % (classes[classId], label)

    cv.putText(frame, label, (left,top), cv.FONT_HERSHEY_SIMPLEX, 2, (255, 0, 0), 3)
```

In [5]:
```python
def postprocess(frame, outs):
    frameHeight = frame.shape[0]
    frameWidth = frame.shape[1]

    classIDs = []
    confidences = []
    boxes = []

    for out in outs:
        for detection in out:

            scores = detection [5:]
            classID = np.argmax(scores)
            confidence = scores[classID]

            if confidence > confThreshold:
                centerX = int(detection[0] * frameWidth)
                centerY = int(detection[1] * frameHeight)

                width = int(detection[2]* frameWidth)
                height = int(detection[3]*frameHeight )

                left = int(centerX - width/2)
                top = int(centerY - height/2)

                classIDs.append(classID)
                confidences.append(float(confidence))
                boxes.append([left, top, width, height])


    # Perform non maximum suppression to eliminate redundant overlapping boxes with
    # lower confidences.

    indices = cv.dnn.NMSBoxes (boxes,confidences, confThreshold, nmsThreshold )

    indices = cv.dnn.NMSBoxes(boxes, confidences, confThreshold, nmsThreshold)
    for i in indices:
        i = i[0]
        box = boxes[i]
        left = box[0]
        top = box[1]
        width = box[2]
        height = box[3]

        drawPred(classIDs[i], confidences[i], left, top, left + width, top + height)

```

In [6]:

```
def getOutputsNames(net):
    # Get the names of all the layers in the network
    layersNames = net.getLayerNames()

    # Get the names of the output layers, i.e. the layers with unconnected outputs
    return [layersNames[i[0] - 1] for i in net.getUnconnectedOutLayers()]


```

In [7]:

```
#Set up the net

net = cv.dnn.readNetFromDarknet(modelConf, modelWeights)
net.setPreferableBackend(cv.dnn.DNN_BACKEND_OPENCV)
net.setPreferableTarget(cv.dnn.DNN_TARGET_CPU)


#Process inputs
winName = 'DL OD with OpenCV'
cv.namedWindow(winName, cv.WINDOW_NORMAL)
cv.resizeWindow(winName, 1000,1000)





cap = cv.VideoCapture(0)

while cv.waitKey(1) < 0:

    #get frame from video
    hasFrame, frame = cap.read()

    #Create a 4D blob from a frame

    blob = cv.dnn.blobFromImage(frame, 1/255, (inpWidth, inpHeight), [0,0,0], 1, crop = False)

    #Set the input the the net
    net.setInput(blob)
    outs = net.forward (getOutputsNames(net))


    postprocess (frame, outs)

    #show the image
    cv.imshow(winName, frame)

```

```

```