```
# Import library that has the ability to point out other files in the memory
import os
from asyncio import wait_for

# Import the needed libraries
import cv2
import numpy as np
import matplotlib.pyplot as plt
import easyocr

#Import additional py file (like a header file)
import util

# INITIALIZATION
#-------------------------------------------------------------------------------
# Define constants with the available files for the YOLOv3-416 model
model_cfg_path = os.path.join('.', 'model', 'cfg', 'darknet-yolov3.cfg')
model_weights_path = os.path.join('.', 'model', 'weights', 'model.weights')
class_names_path = os.path.join('.', 'model', 'class.names')

# Configure the camera to be used
cap = cv2.VideoCapture(0)

# load class names
with open(class_names_path, 'r') as f:
  class_names = [j[:-1] for j in f.readlines() if len(j) > 2]
  f.close()

# load model
net = cv2.dnn.readNetFromDarknet(model_cfg_path, model_weights_path)
#------------------------------------------------------------------------------

while True:
  # Read the frame of the camera
  ret, frame = cap.read()
  print('Frame captured')

  # Take the height and the width of the frame
  H, W, _ = frame.shape

  # convert image
```

```
blob = cv2.dnn.blobFromImage(frame, 1 / 255, (416, 416), (0, 0, 0), True)

# Set the input for the network
net.setInput(blob)

# Get detections
detections = util.get_outputs(net)

# bboxes, class_ids, confidences
bboxes = []
class_ids = []
scores = []

if detections:
  print('Detection happened')
  for detection in detections:
    # [x1, x2, x3, x4, x5, x6, ..., x85]
    bbox = detection[:4]

    xc, yc, w, h = bbox
    bbox = [int(xc * W), int(yc * H), int(w * W), int(h * H)]

    bbox_confidence = detection[4]

    class_id = np.argmax(detection[5:])
    score = np.amax(detection[5:])

    bboxes.append(bbox)
    class_ids.append(class_id)
    scores.append(score)

  # apply nms
  bboxes, class_ids, scores = util.NMS(bboxes, class_ids, scores)

  # plot
  reader = easyocr.Reader(['en'])



  for bbox_, bbox in enumerate(bboxes):
    xc, yc, w, h = bbox
```

```
        # Draw a rectangle around the licence plate
        frame = cv2.rectangle(frame,
                    (int(xc - (w / 2)), int(yc - (h / 2))),
                    (int(xc + (w / 2)), int(yc + (h / 2))),
                    (0, 255, 0),
                    10)

        # Change the picture to gray to get rid of the unneeded color channels
        licence_plate_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        # Change the picture to a pure black and white regarding the gray value to make
the text reader's job easier
        _, licence_plate_thresh = cv2.threshold(licence_plate_gray, 128, 255,
cv2.THRESH_BINARY_INV)

        # Read the text detected on the picture
        output = reader.readtext(licence_plate_thresh)

        # For every output generated by the text reader, write out its text and how confident
the algorithm is
        for out in output:
          text_bbox, text, text_score = out
          if text_score > 0.7:
            print(text, text_score)

  output = 0

  cv2.imshow('Frame', frame)

  k = cv2.waitKey(1)
  if k ==27:
    break

cap.release()
cv2.destroyAllWindows()
```