# Tutorial works as of 7/21/2023

# Necessary files should be included in .zip

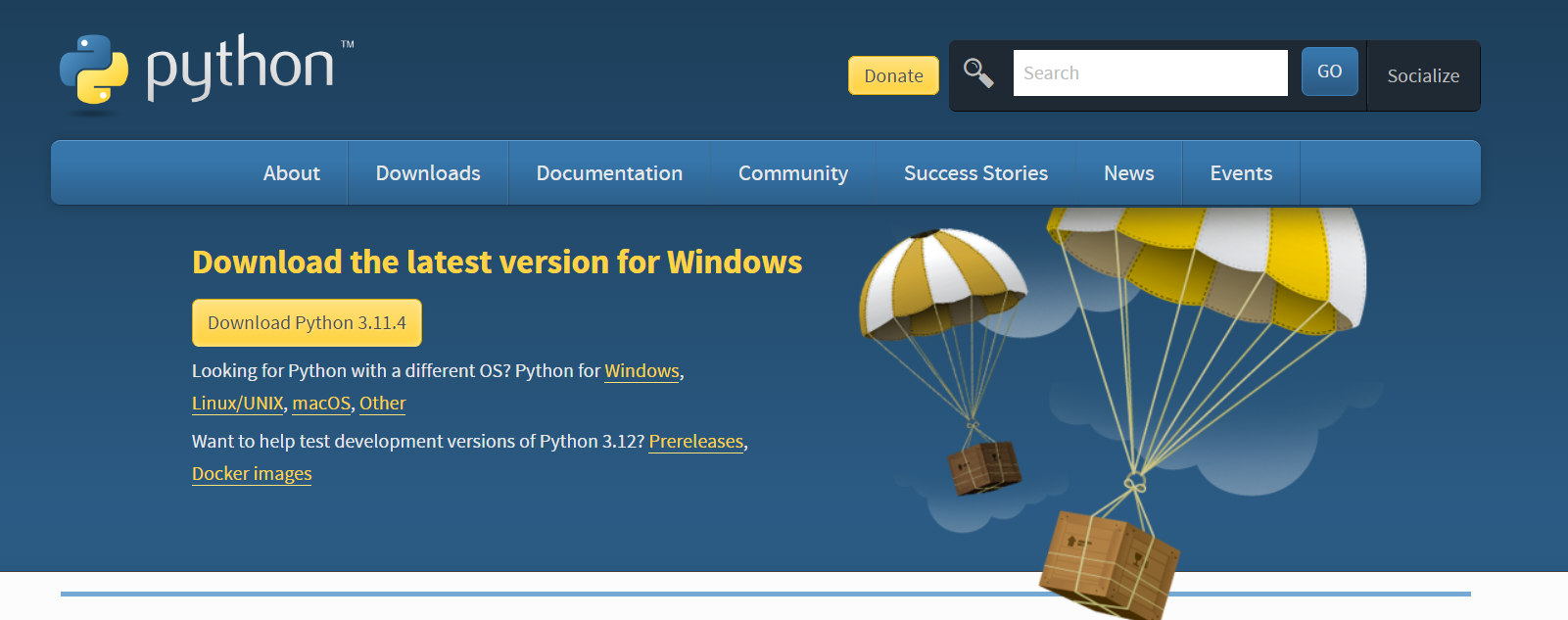
# Objective:

* Install OpenCV and Python on Windows
* Create simple python script that incorporates YOLO through OpenCV

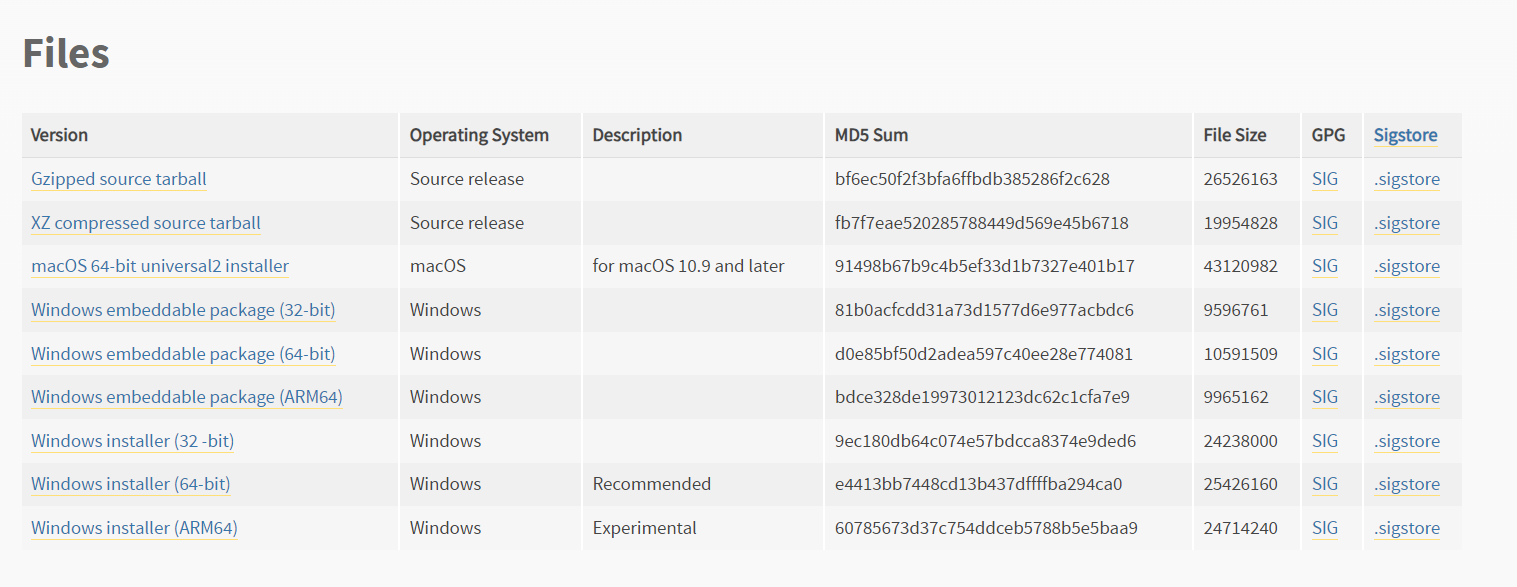
# Python and OpenCV Install (Windows)

1. Go to <https://www.python.org/downloads/> and download Python for windows. The latest version as of this tutorial is 3.11.4

Click download

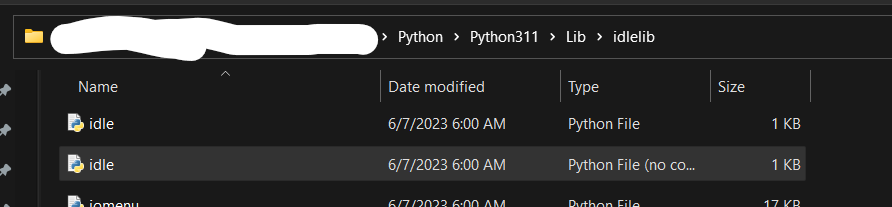


And select windows installer 64 bit

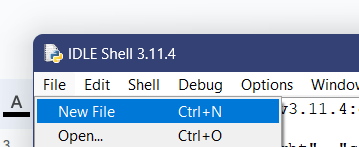


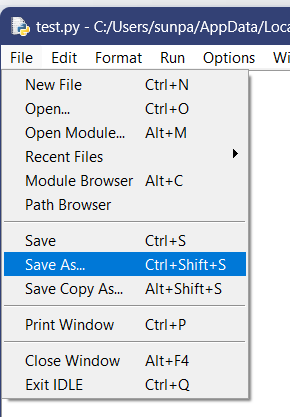
1. Once the installer is downloaded, run exe and follow steps, take note of the file path of the installation. I also recommend pinning the Python311 folder to quick access in file explorer. The Python311 folder will be the working directory for this tutorial as it is the working directory for Python IDLE. To run IDLE (and pin it to the task bar) in this installation, find it using the path from the python folder which should be: “Python\Python311\Lib\idlelib\idle.pyw"

Run IDLE



1. Within IDLE, click File -> New File to make a new file



From the new file window click File -> Save As to save your file, it should save to the Python311 directory by default: the working directory of IDLE

1. Now you need to install the libraries we will be using: OpenCV, Numpy, and Matplotlib

Open the command line and these commands

pip install opencv-contrib-python

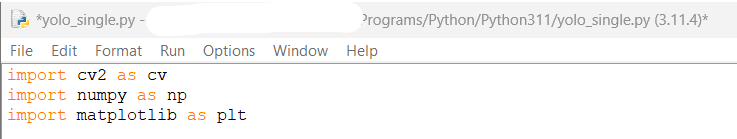
pip install numpy

pip install matplotlib

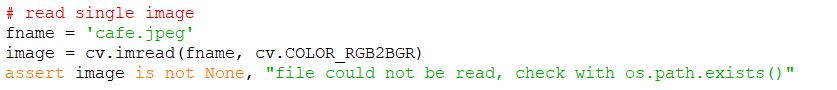
Now that you are familiar with Python in Windows and IDLE you are ready for the second tutorial.

Creating a Basic YOLOv3 Object Detection Script with Python and OpenCV

1. Create a new file in from IDLE and save it as “yolo\_single.py”
2. Begin the script by importing necessary libraries, cv2, numpy and matplotlib

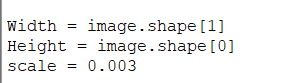


1. To begin our Computer Vision program, we need to start by reading an image into our instantiation of a cv2 image object with a file path. Remember that the Python311 folder is the working directory of Python, you may also use absolute file paths to another directory if you wish. (Note that when using absolute file paths in python in a windows environment you may use forward slashes or backslashes in your string literal file path, however, back slashes must also be escaped so they will be entered in as double back slashes.)



Use the assertion for error handling, incase you type in a file name incorrectly or it doesn’t exist

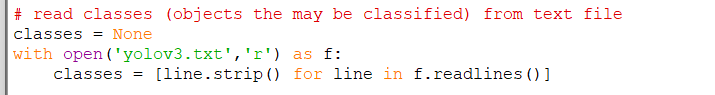
1. We will need to make some variables from our image dimensions to use as parameters in later function calls:



The .shape property is an array of rows,columns and color channels of the read image. Also take note of the “cv.COLOR\_RGB2BGR” flag passed into the cv2 image object creation in the last step. Python OpenCV works with images in BGR instead of rgb format so conversions must made when an image is instantiated and checked at output as well.

1. The next files we need should be included in the .zip file this tutorial came with otherwise you can get the weights file at: <https://pjreddie.com/media/files/yolov3.weights> ; and the other yolov3 files at this github repository: <https://github.com/arunponnusamy/object-detection-opencv> . Place the “yolov3.cfg”,”yolov3.txt” and “yolov3.weights” files in the Python311 working directory.

In your script type

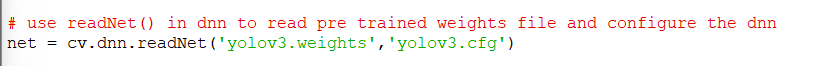


The yolov3.txt files contains the names of trained everyday objects in order on each line, so we need to read them into an array.

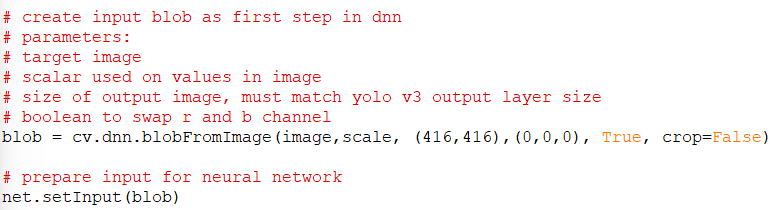
1. It is common practice in OpenCV YOLO projects to randomly assign colors to use when creating boxes that will be placed around detected objects in the image. Let’s set up the random color variable now



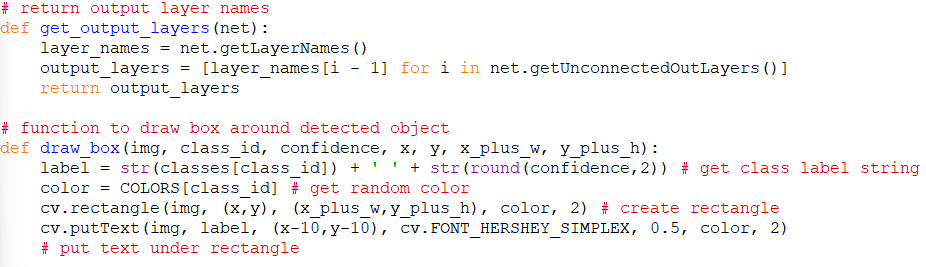
1. Now we need to create a dnn (deep neural network) object and prepare it using the weights and configuration files from YOLOv3(Which is the real heavy work done by the developers of YOLO). OpenCV contains the DNN module and its the data from YOLOv3 in the weights and config files that set up the layers and weights of the DNN for us.



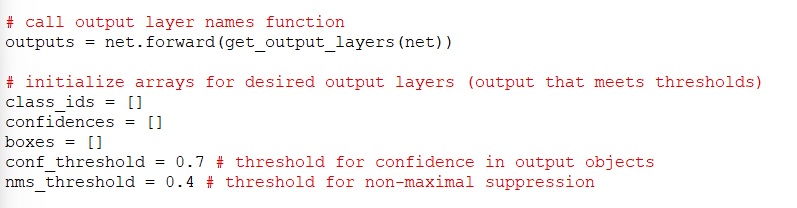
1. Now we need to create an image blob from our target image and feed it as the input to the DNN



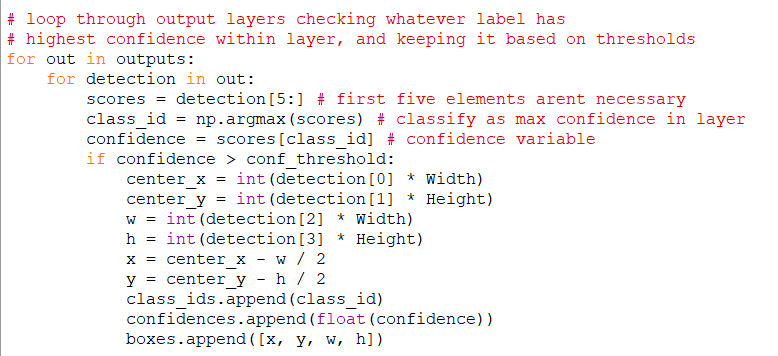
1. Now lets define some helping functions for retrieving our output layers and drawing a box around our found objects



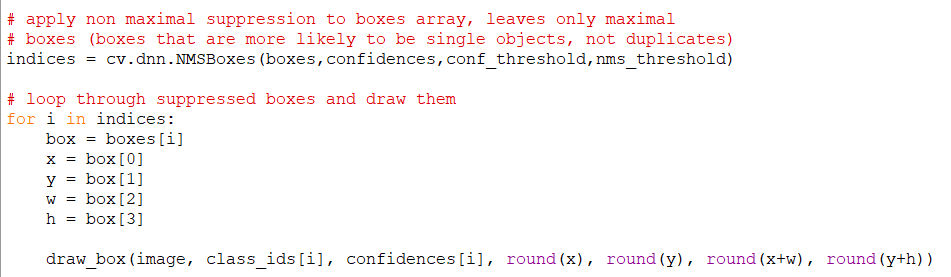
1. Now lets retrieve our output layers after forward propagation through the DNN and set up some arrays to use when checking our output



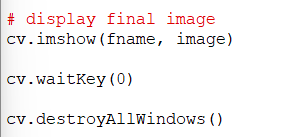
1. Here we iterate through our output layers and for each detection in each layer we keep which ever object detections meet our desired thresholds



1. Now we need to apply non maximal suppression to our results using the NMSupression function with the DNN module, then loop through our suppressed box array and draw them onto our image



1. Now we use the cv.imshow function to display our final result

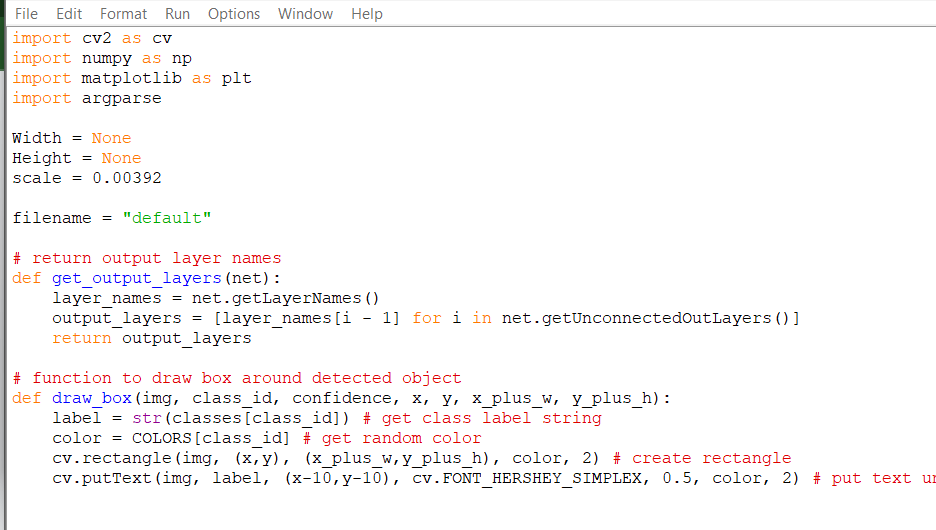




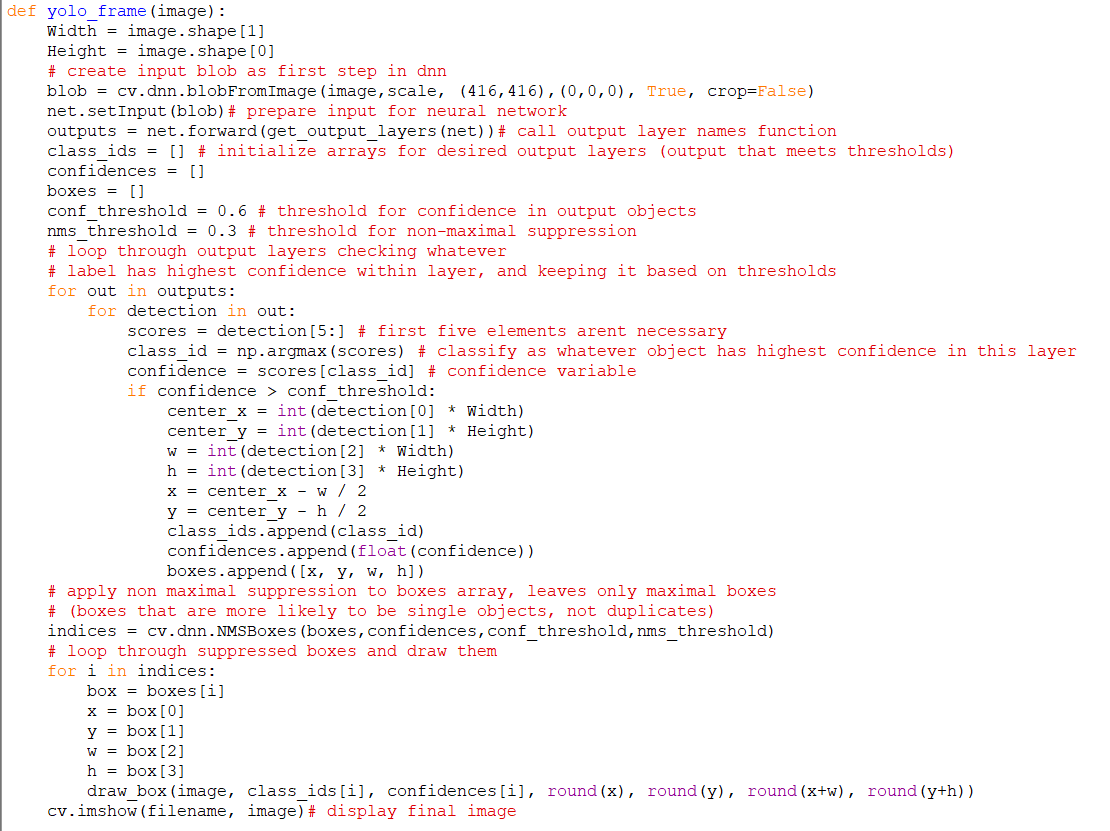
At this point one might wonder how to go about using this in a video

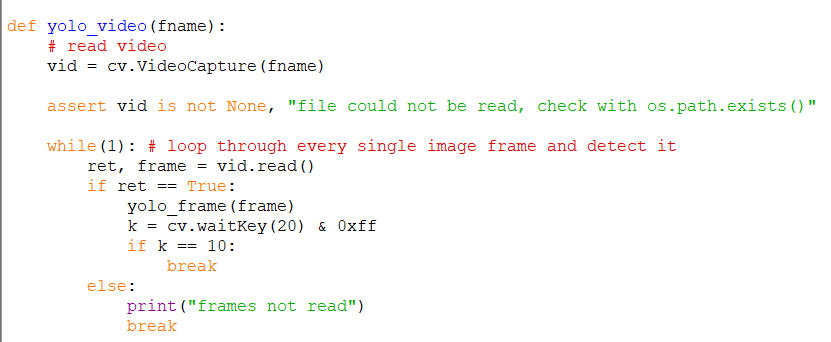
We can do this using OpenCV’s video capture function which allows us to read the video frame by frame as single images. So with the definition of some new helper functions and rearranging our previous code we can create something simple

1. Create a new file called “yolo\_video.py”
2. Copy over our previous imports ( I am also using an arg parser this time, but you can just change the default file name ) and some of our previous helper functions

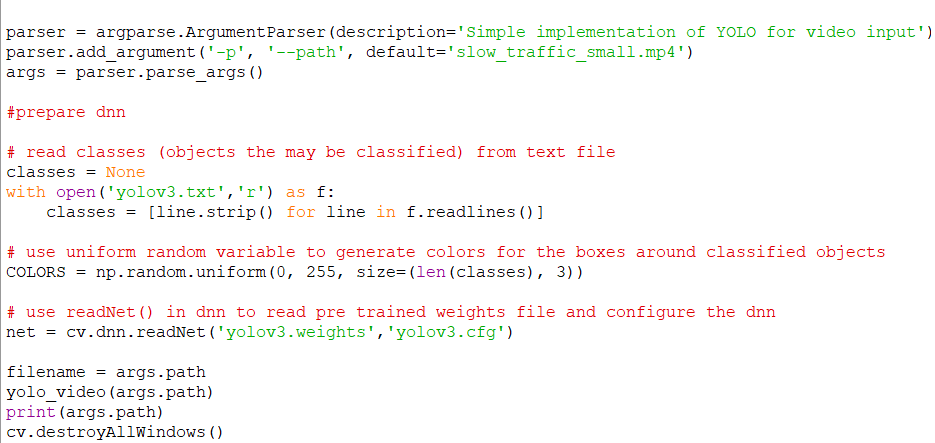


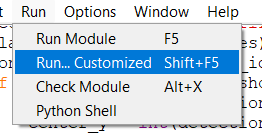
1. Now lets create some new helper functions, one for reading the video with a loop to read all the frames and call our yolo frame function on our current frame, and our yolo frame function to use our yolo configured DNN on the input frame(this one uses a lot of the code from the single image script rearranged):

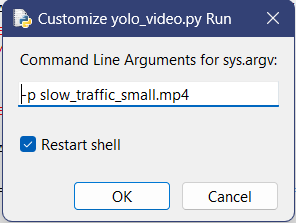


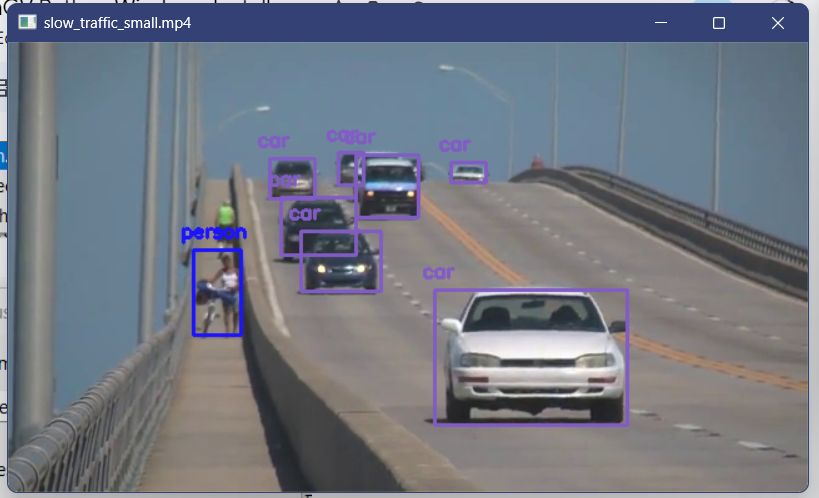


1. Now the rest of the actual function calling is pretty simple and carried over from the previous script. We only need to prepare the DNN once, then we can use it multiple times for the frames in the video. Here I am also using an arg parse to make it easier to test with different filenames using shift f5 to run with special arguments.









Successful, but slow on my laptop. In IDLE click Shell -> Restart Shell to kill the process