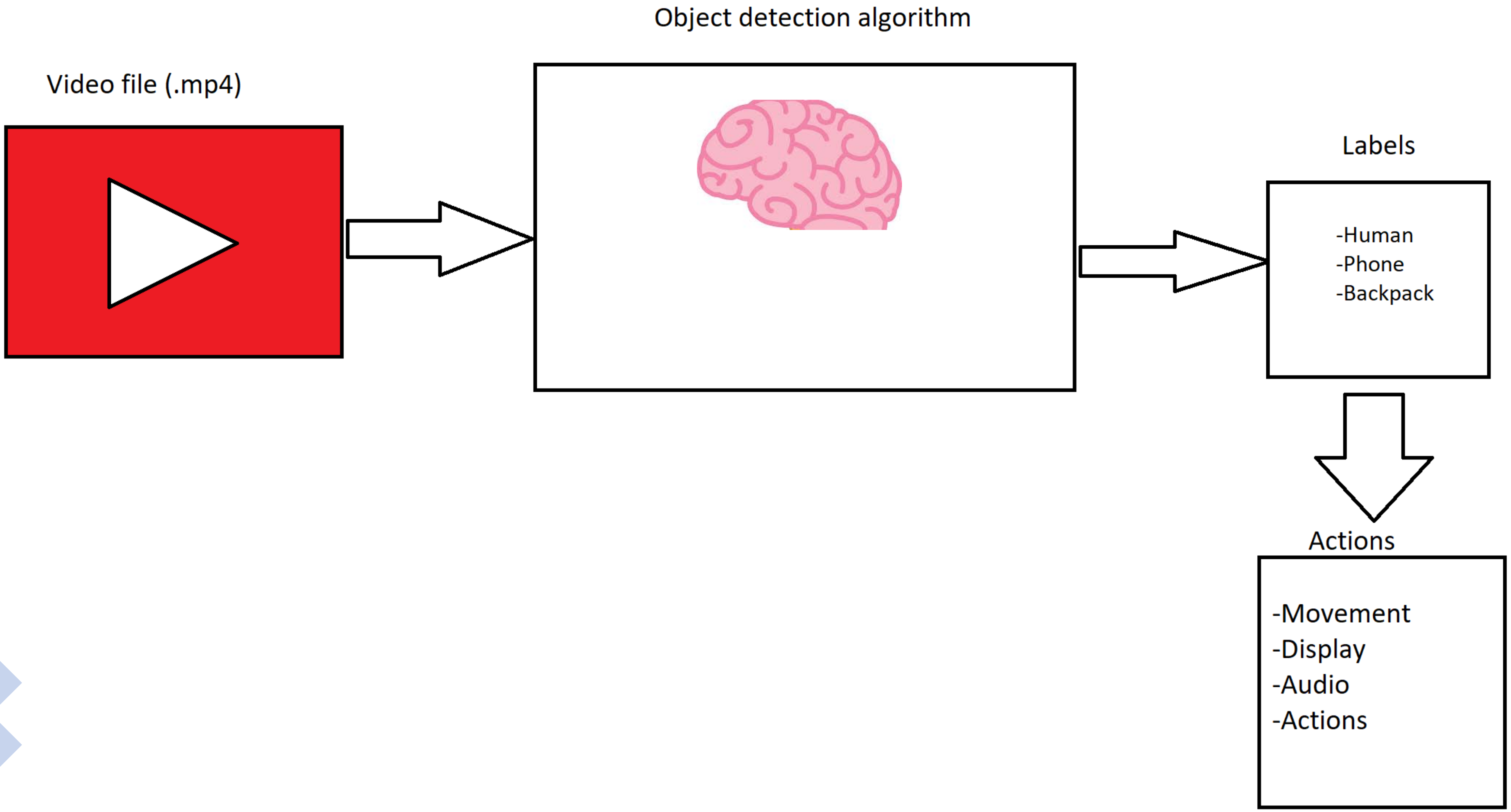
Implementing object detection using TensorFlow Lite on a Rpi

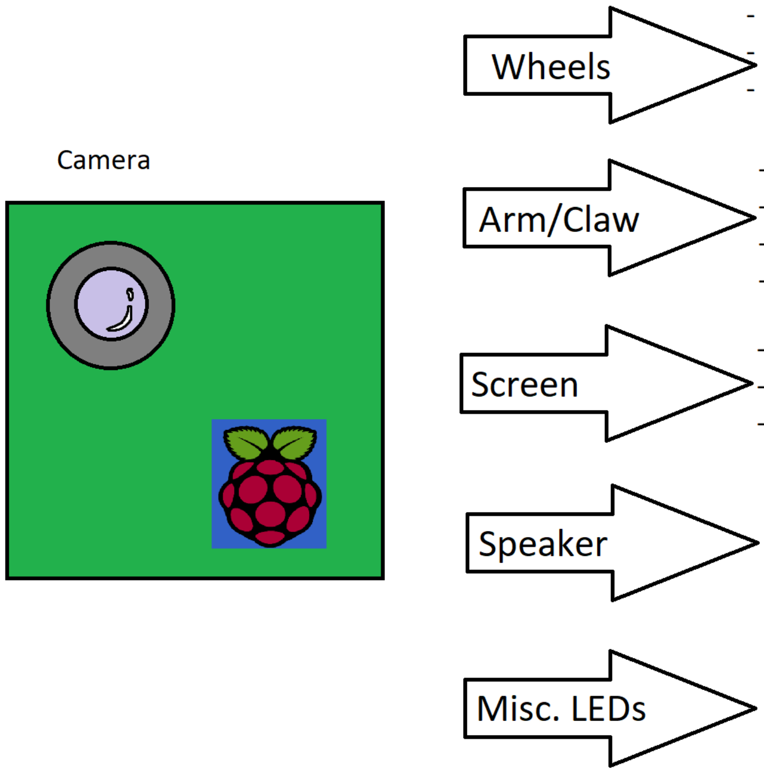
Top-level overview



Actions

Detected:

-Person
-Backpack



Camera

Output to:

Wheels

- Move any direction
- Move a specific sequence of directions
- Move a direction for x amount of time

Arm/Claw

-Raise Arms
-Clench Claw
-Wave
-Expand/Retract arm(pass out flyer?)

Screen

-Display pre-recorded/pre-made IEEE video
-Display image slideshow
-Display predefined images/videos

Speaker

-Hello greeting
-Play music
-Play pre-recorded audio
-Play video audio

Misc. LEDs

-Flash in a sequence
-Turn off/on

What it has no control on:
-Manual remote control trigger
-Shutdown

Current Labels

/home/pi/tflite1/Sample_TFLite_model/labelmap.txt

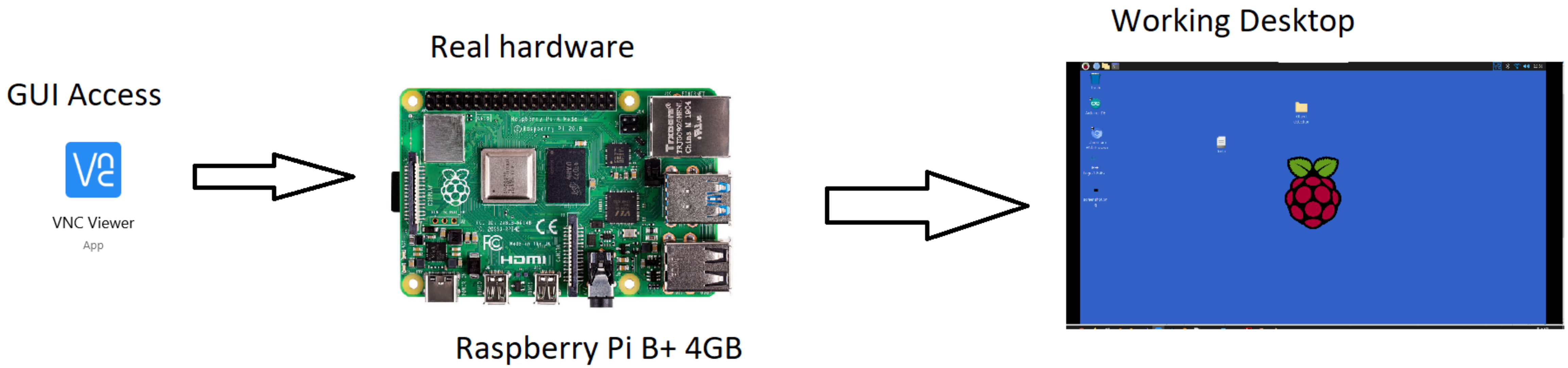
Person	Bicycle	Car	Motorcycle	Airplane	Bus	Train	Truck	Boat	Traffic light
Fire hydrant	Stop sign	Parking meter	Bench	Bird	Cat	Dog	Horse	Sheep	Cow
elephant	Bear	Zebra	Giraffe	Backpack	Umbrella	Handbag	Tie	Suitcase	frisbee
Skis	Snowboard	Sports ball	Kite	Baseball bat	Baseball glove	Skateboard	Surfboard	Tennis racket	Bottle
Wine glass	Cup	Fork	Knife	Spoon	Bowl	Banana	Apple	Sandwich	Orange
Broccoli	Carrot	Hot dog	Pizza	Donut	Cake	Chair	Couch	Potted plant	Bed
Dining table	Toilet	Tv	Laptop	Mouse	Remote	Keyboard	Cell phone	Microwave	Oven
Toaster	Sink	Refrigerator	Book	Clock	Vase	Scissors	Teddy bear	Hair drier	toothbrush

Things to consider

- Multiple different objects detected simultaneously
- Have a priority of objects detected
- ex: Person and phone both detected choose which to focus on
- Pause detection after a person is identified and action is started to avoid processing/battery power wasted.

…

Workflow



GUI Access

VNC Viewer

App

Real hardware

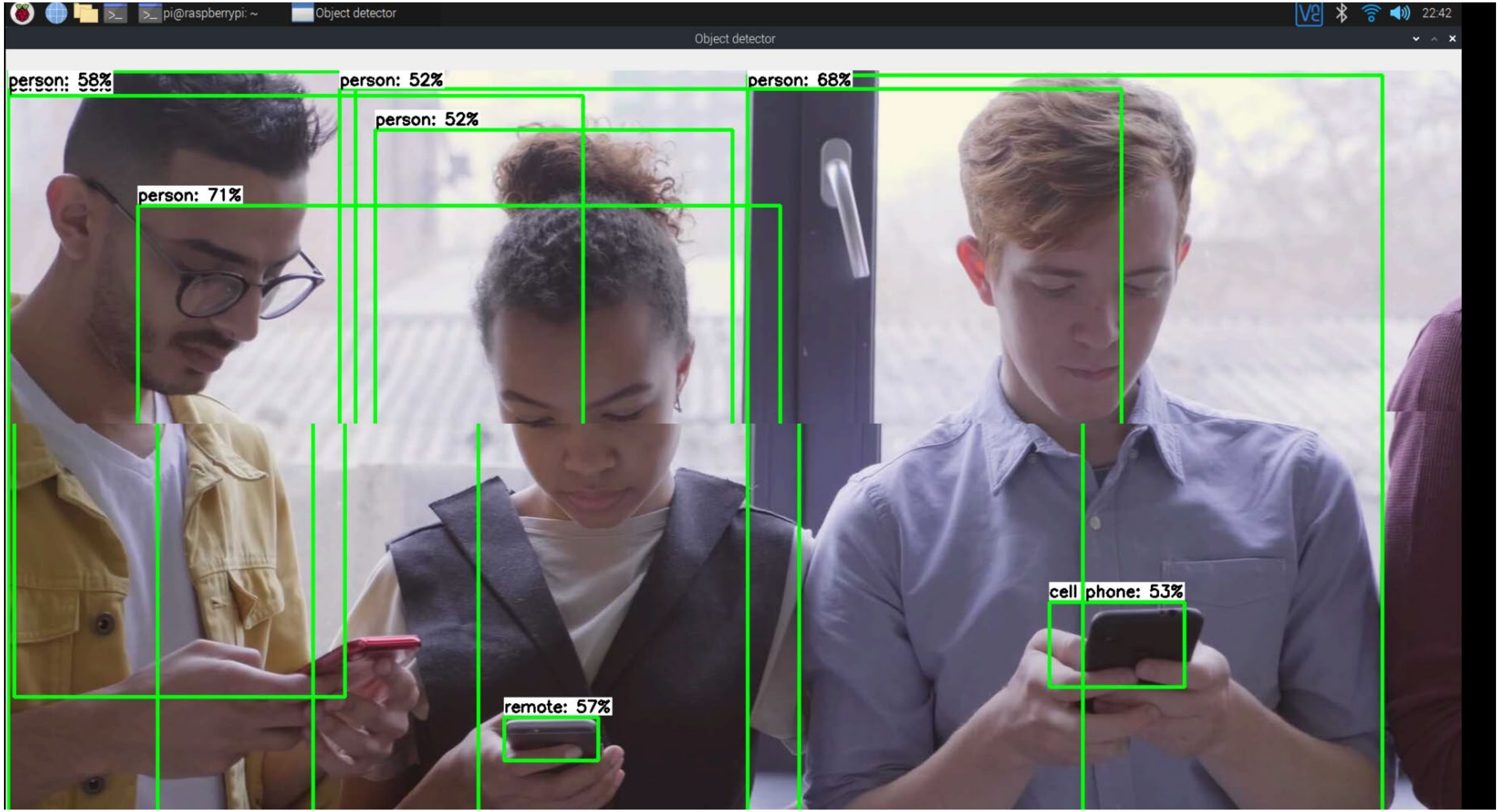
Raspberry Pi B+ 4GB

Working Desktop

Why VNC viewer and not SSH?

- GUI is needed to observe live feedback with label mapping.
- Can’t be done through terminal unless manual console printing of labels detected with video.
- More convenient.
- SSH is still possible if you’re more comfortable with it.

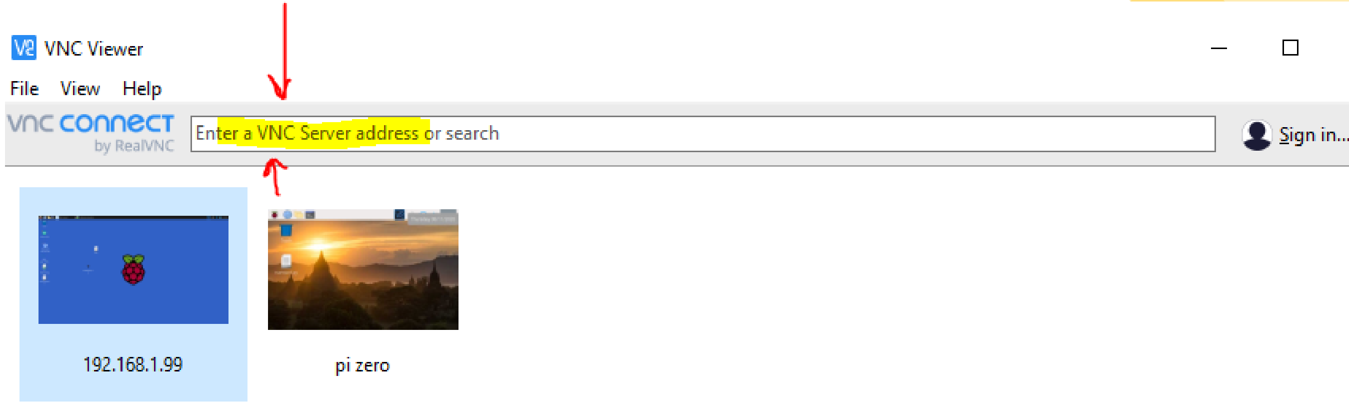
Why VNC viewer and not SSH?



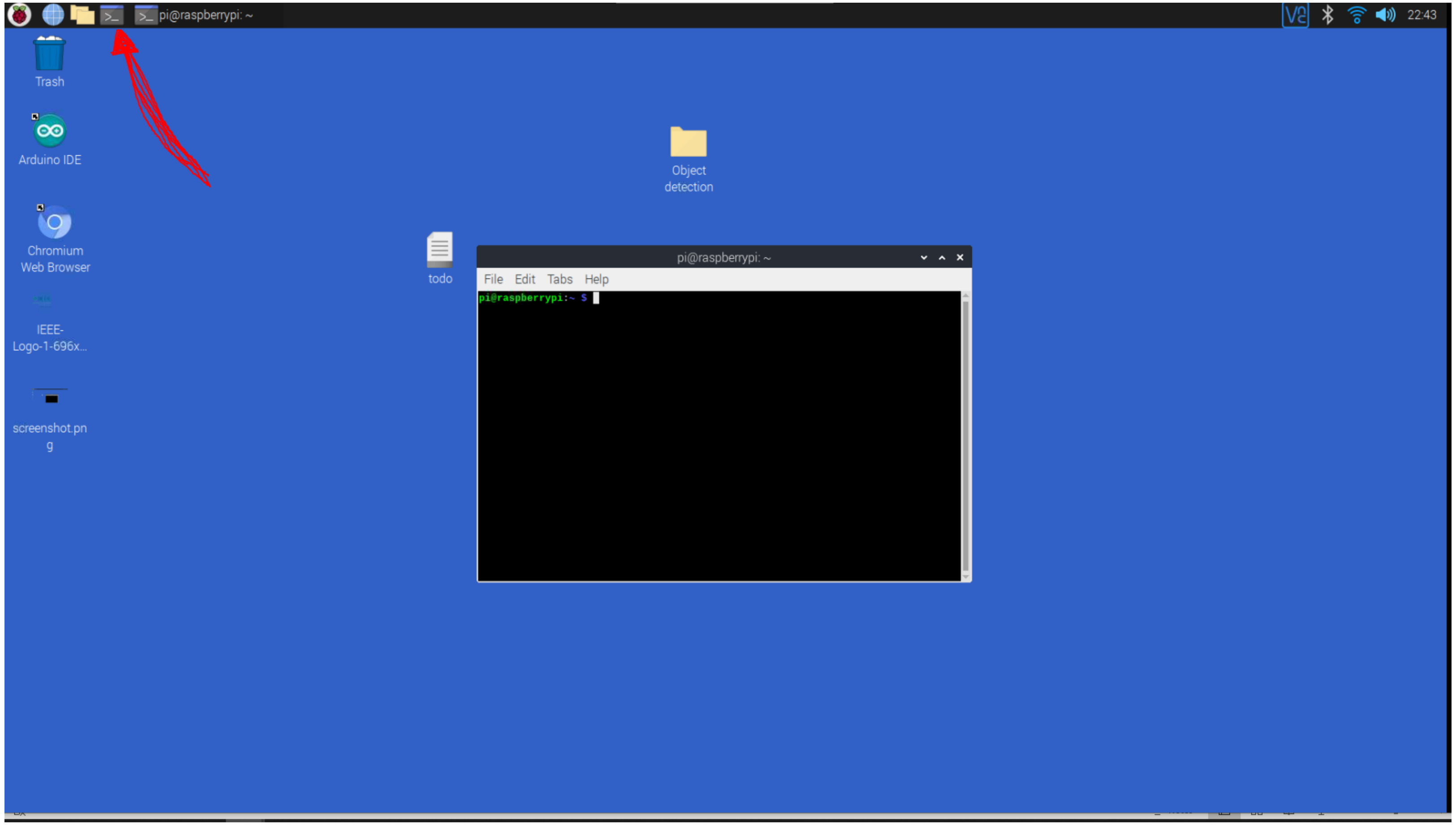
Connecting

- First make sure you have **VNC Viewer** installed
- Start VNC viewer
- Type the following IP:

  173.173.156.125
- Username: **pi**
- Password: **IEEE**
- Click yes or accept on any prompts.



Using the terminal



```
pi@raspberrypi:~ $
```

Basic commands(lowercase)

- ls -> list files in current directory

Can also open file browser

- dir -> show directory and folders
- cd -> change directory to

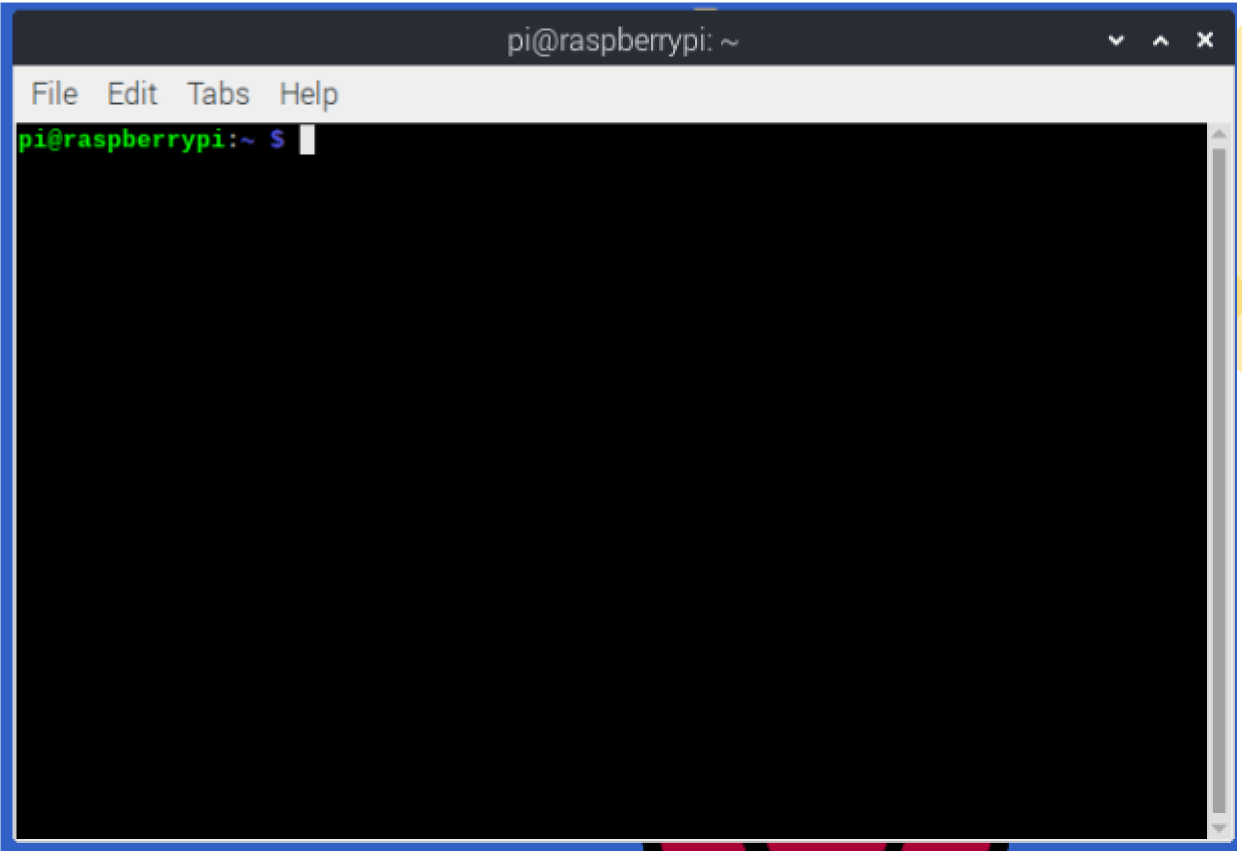
a folder or up a directory

Example uses:

cd tflite1 -> changes to tflite1

cd .. -> Goes up one folder in the directory

cd -> goes to home directory



```
pi@raspberrypi:~ $
```

Basic commands(lowercase)

- pwd -> prints working(current) directory

Useful to know where you are in the command prompt

- nano filename -> opens file in nano text editor for editing

Executing shell scripts

- All shell scripts have a .h extension
- To execute simply call it with a ./ right before the name
- Ex: to call rpivideotest.h
- ./rpivideotest.h
- Can also edit shell script with nano
- Nano rpivideotest.h

Useful to open the environment already and start testing

How to start up the TF Lite Python environment manually

- Manually:

1. Make sure you’re in directory: /home/pi/tflite1

2. Know the file name of your testing script here testing script: TF_video.py

3. call it on commandline as follows:

```
cd tflite1/
source tflite1-env/bin/activate
python3 TF_video.py --modeldir=Sample_TFLite_model
```

Dissecting the commands

```
cd tflite1/ //Changes directory to tflite1 folder
source tflite1-env/bin/activate  //Activates tensorflow environment
python3 TF_video.py --modeldir=Sample_TFLite_model
//runs script called TF_video.py using python3 with a model directory in the folder called Sample_TFLite_model
```

How to start up the TF Lite Python environment with a script

- First know that you can change the filename of the video to run on the script itself or you can simply rename the videofile.
- Open up a text editor either by GUI or by calling nano on the commandline with the filename of your choice, make sure its extension is .h
- Simply type in these 3 lines into it and save as a .h file:

```
cd tflite1/
source tflite1-env/bin/activate
python3 TFLite_detection_video.py --modeldir=Sample_TFLite_model
```

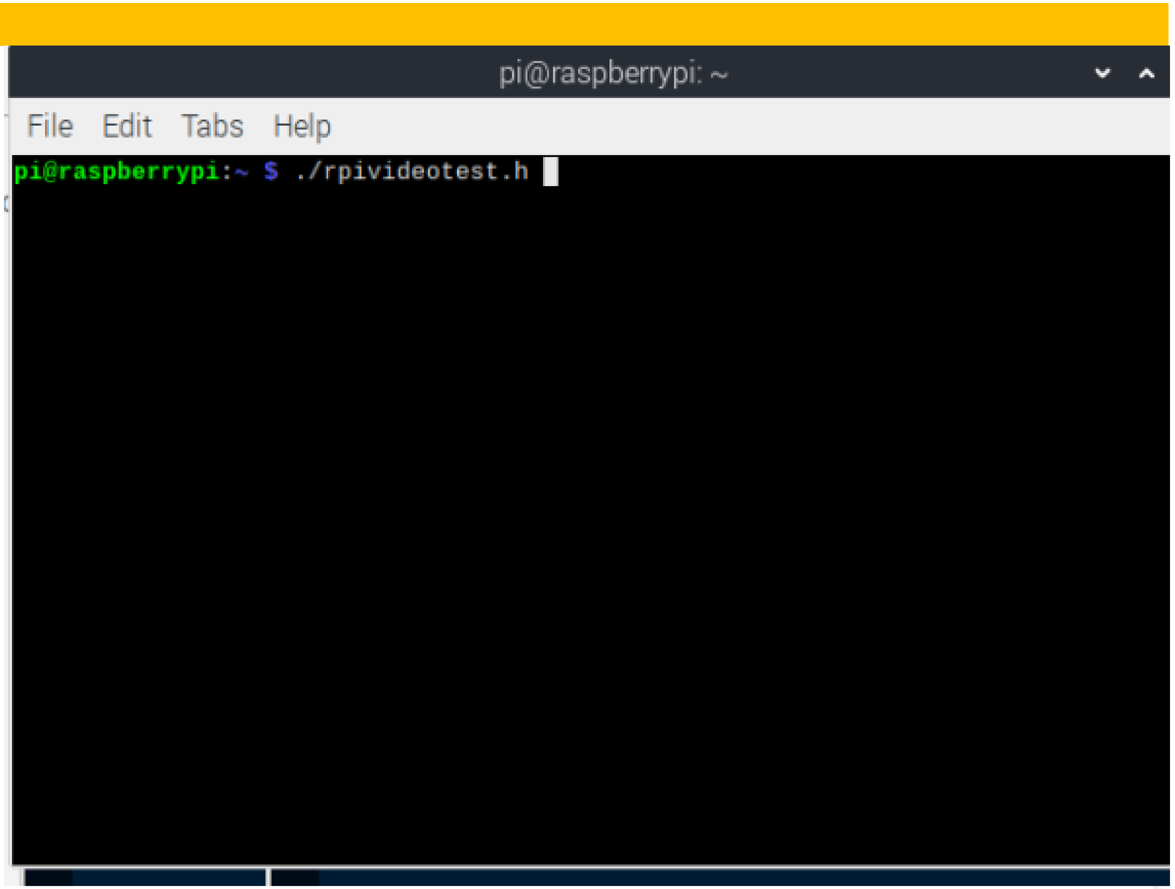
Remember this is your script and you can re-name it to anything

Calling shell script

- Change directory to wherever you saved it.
- Home directory would be ideal.
- Call it by ./ before the name

Ex: calling rpivideotest.h located in home directory

Success!



```
pi@raspberrypi:~ $ ./rpivideotest.h
```

Why video script and not live?

Unforeseen circumstances resulted in inability to safely meet in person in groups.

I rather have 0% risk than any small amount.

Should work as well as the script doesn’t “know” that it’s a video or live feed.

More controlled. We can replay the same video with the object of our choosing to see how well it detects it and adjust it on that specific scenario.

It will work.

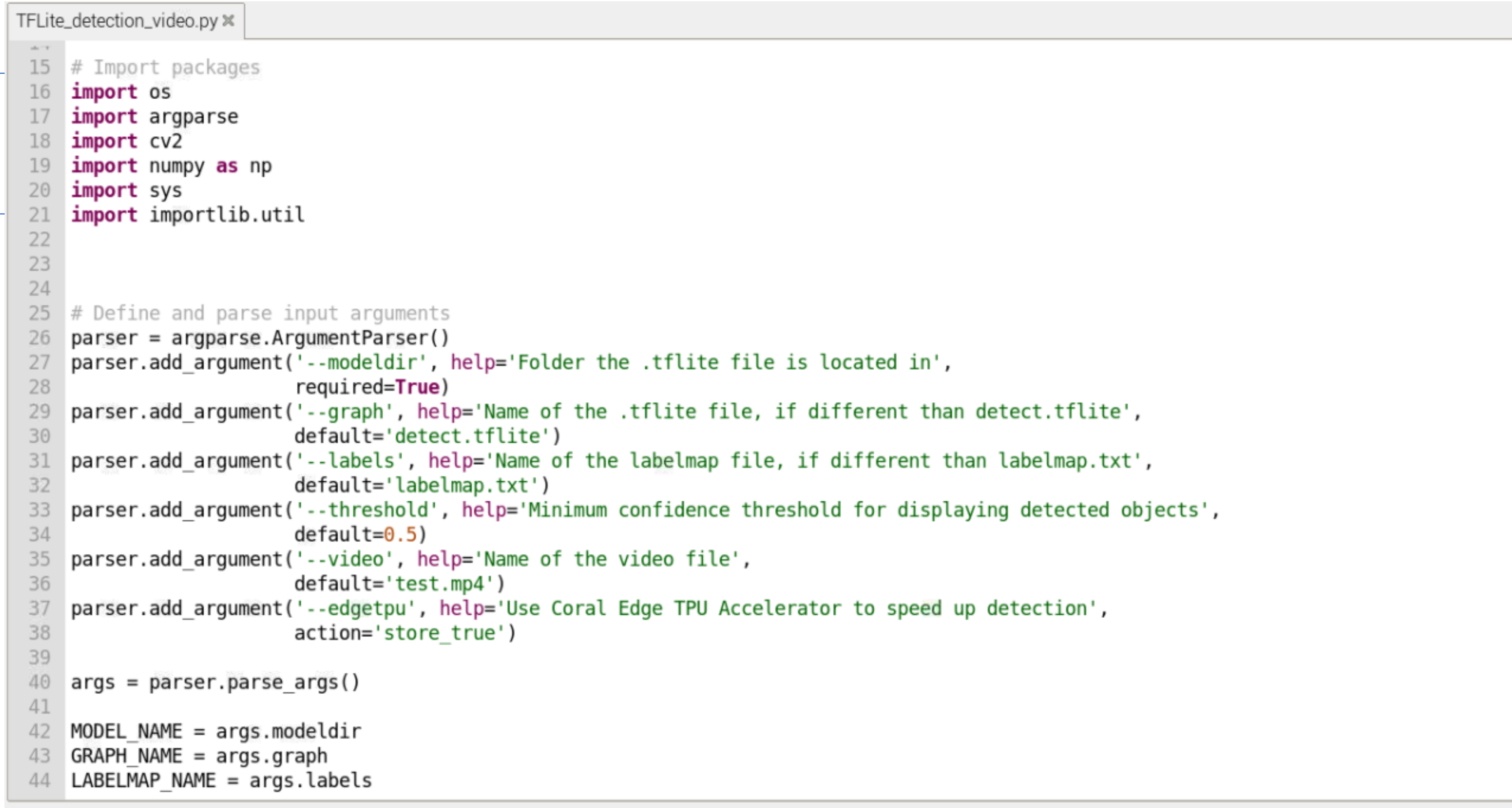
Video script basics

- We will be using the script located in ***/home/pi/tflite1/*** called ***TFLite_detection_video.py***
- ***DO NOT** modify this script.* Simply create a copy with a different name within the same directory or any of your choosing.
- If you are already python savvy, then you can simply modify the script to your choosing.
- This will be a team collaboration of code which will get difficult to work together if comments are omitted.
- Suggestion: Each of you have a version of your own with your own modifications and separate joined code to see if it works properly.

Code explanation

Packages needed, not recommended to remove any

Recommended to only modify the default name of the video if testing multiple videos each run

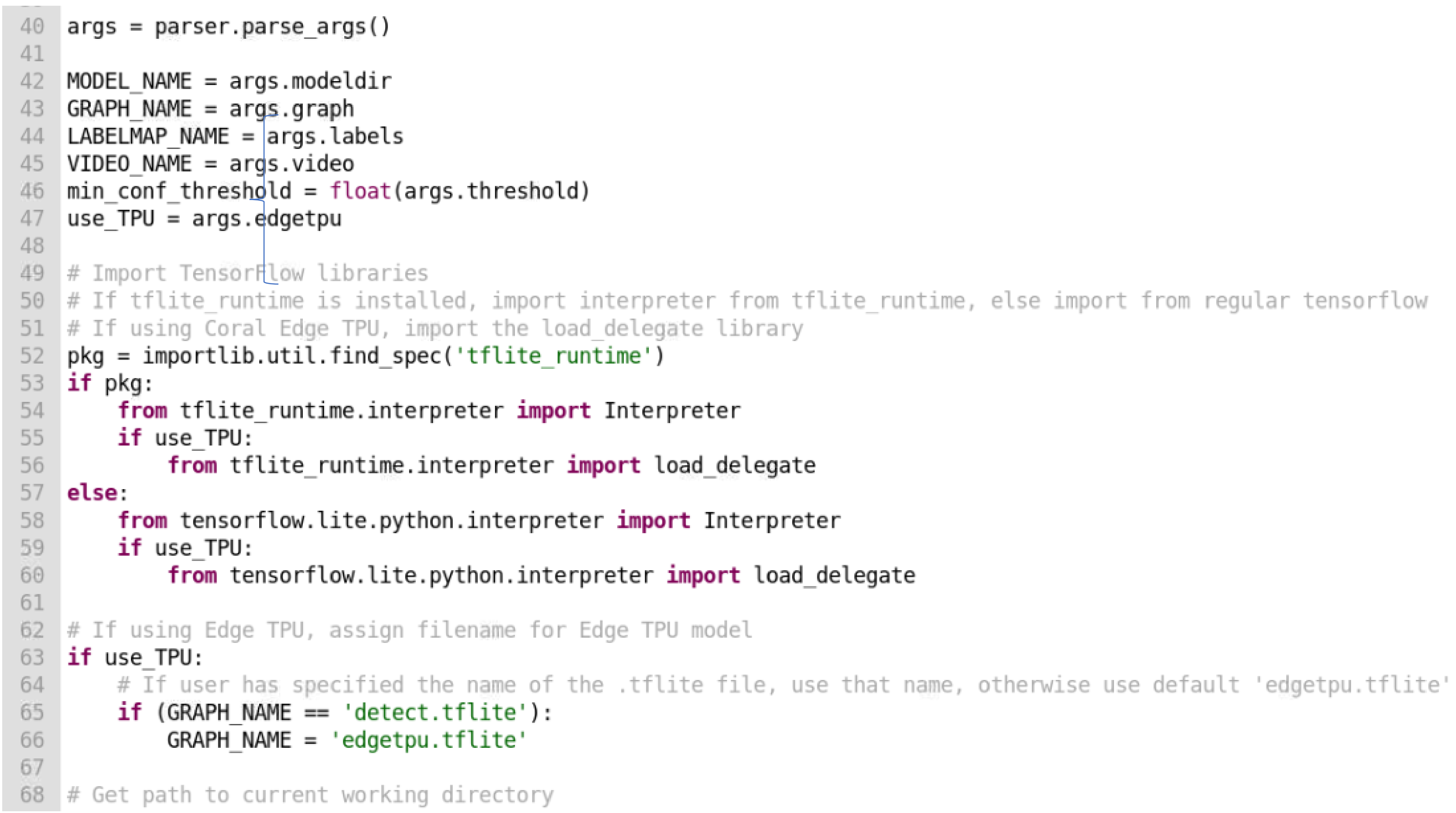


```python
# Import packages
import os
import argparse
import cv2
import numpy as np
import sys
import importlib.util



# Define and parse input arguments
parser = argparse.ArgumentParser()
parser.add_argument('--modeldir', help='Folder the .tflite file is located in',
                    required=True)
parser.add_argument('--graph', help='Name of the .tflite file, if different than detect.tflite',
                    default='detect.tflite')
parser.add_argument('--labels', help='Name of the labelmap file, if different than labelmap.txt',
                    default='labelmap.txt')
parser.add_argument('--threshold', help='Minimum confidence threshold for displaying detected objects',
                    default=0.5)
parser.add_argument('--video', help='Name of the video file',
                    default='test.mp4')
parser.add_argument('--edgetpu', help='Use Coral Edge TPU Accelerator to speed up detection',
                    action='store_true')

args = parser.parse_args()

MODEL_NAME = args.modeldir
GRAPH_NAME = args.graph
LABELMAP_NAME = args.labels
```

Simply stating the needed parameters for the model to run that were already defined, not really needed to change

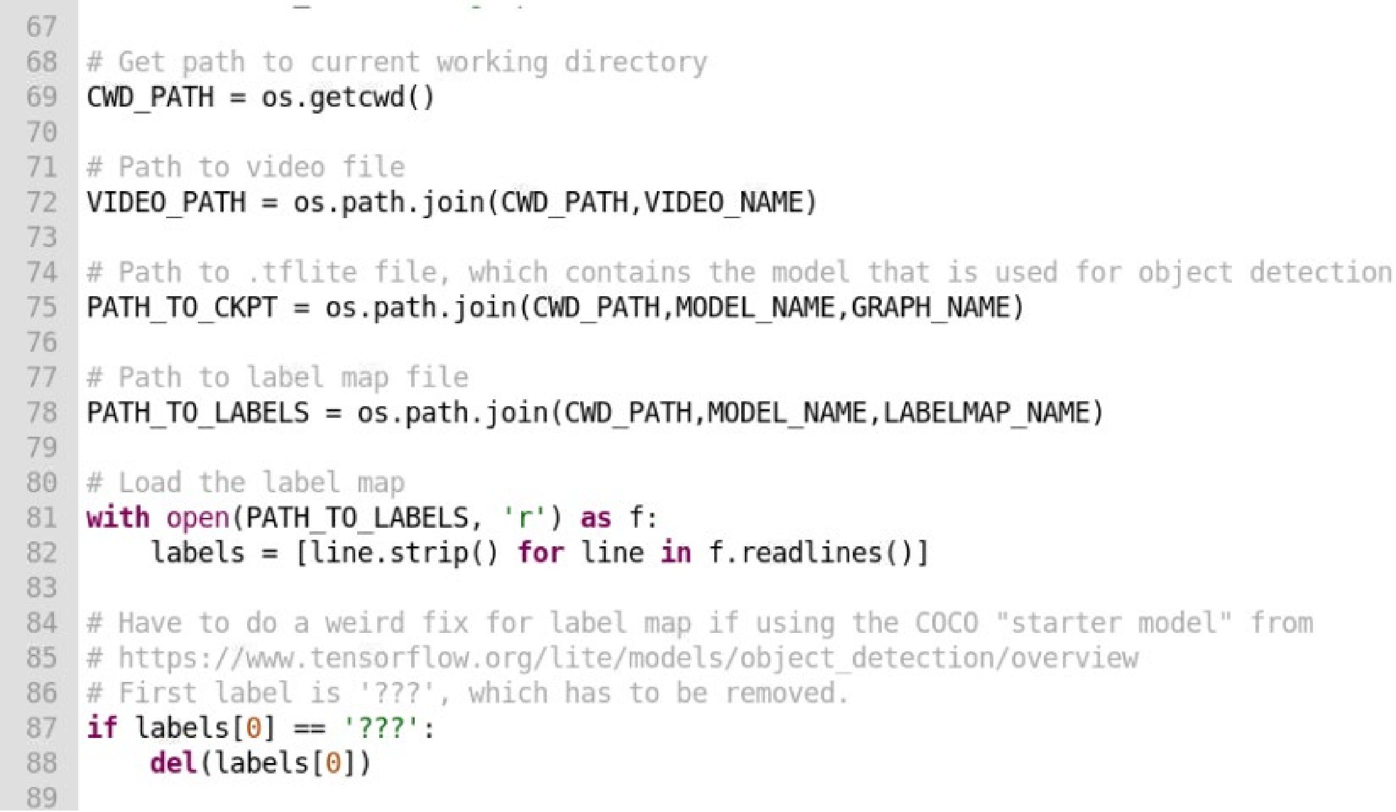
Needed imports, part of the code to not be modified

Not using TPU, this part is not needed at all



```python
args = parser.parse_args()

MODEL_NAME = args.modeldir
GRAPH_NAME = args.graph
LABELMAP_NAME = args.labels
VIDEO_NAME = args.video
min_conf_threshold = float(args.threshold)
use_TPU = args.edgetpu

# Import TensorFlow libraries
# If tflite_runtime is installed, import interpreter from tflite_runtime, else import from regular tensorflow
# If using Coral Edge TPU, import the load_delegate library
pkg = importlib.util.find_spec('tflite_runtime')
if pkg:
    from tflite_runtime.interpreter import Interpreter
    if use_TPU:
        from tflite_runtime.interpreter import load_delegate
else:
    from tensorflow.lite.python.interpreter import Interpreter
    if use_TPU:
        from tensorflow.lite.python.interpreter import load_delegate

# If using Edge TPU, assign filename for Edge TPU model
if use_TPU:
    # If user has specified the name of the .tflite file, use that name, otherwise use default 'edgetpu.tflite'
    if (GRAPH_NAME == 'detect.tflite'):
        GRAPH_NAME = 'edgetpu.tflite'

# Get path to current working directory
```

Simply a method to get the directories of the files that are needed, not recommended to modify unless you really want to



```
# Get path to current working directory
CWD_PATH = os.getcwd()

# Path to video file
VIDEO_PATH = os.path.join(CWD_PATH,VIDEO_NAME)

# Path to .tflite file, which contains the model that is used for object detection
PATH_TO_CKPT = os.path.join(CWD_PATH,MODEL_NAME,GRAPH_NAME)

# Path to label map file
PATH_TO_LABELS = os.path.join(CWD_PATH,MODEL_NAME,LABELMAP_NAME)

# Load the label map
with open(PATH_TO_LABELS, 'r') as f:
    labels = [line.strip() for line in f.readlines()]

# Have to do a weird fix for label map if using the COCO "starter model" from
# https://www.tensorflow.org/lite/models/object_detection/overview
# First label is '???', which has to be removed.
if labels[0] == '???':
    del(labels[0])

```

Since we’re using the google model, the first label is just question marks. The programmer did a fix for this, not recommended to remove

Not using edge TPU and probably won’t, so only else statement is needed as a normal statement

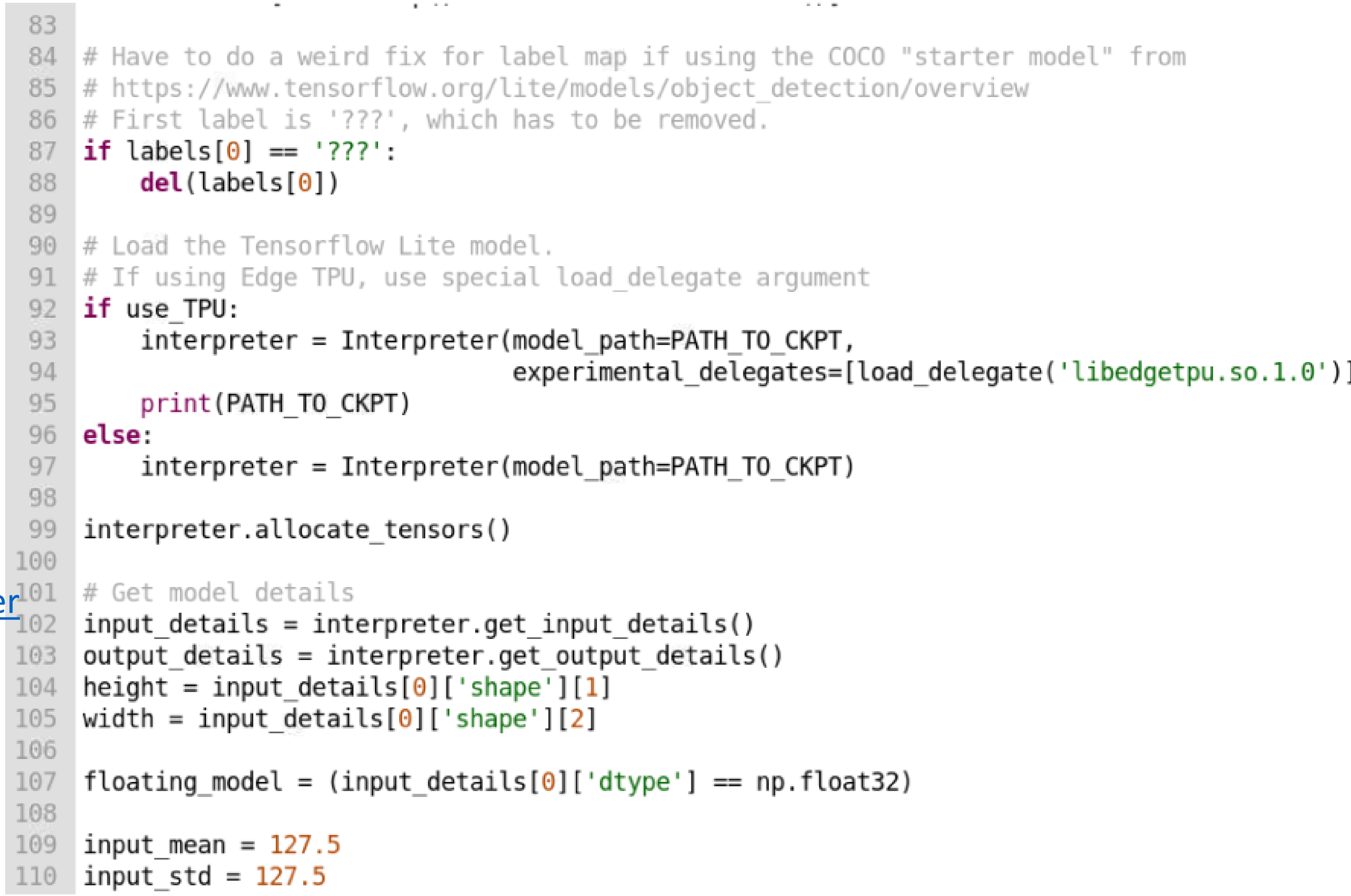
See https://www.tensorflow.org/lite/guide/inference

On the

Load and run a model in Python

Platform: Linux

Section but doesn’t really need to be modified unless you think there’s a better way for our application

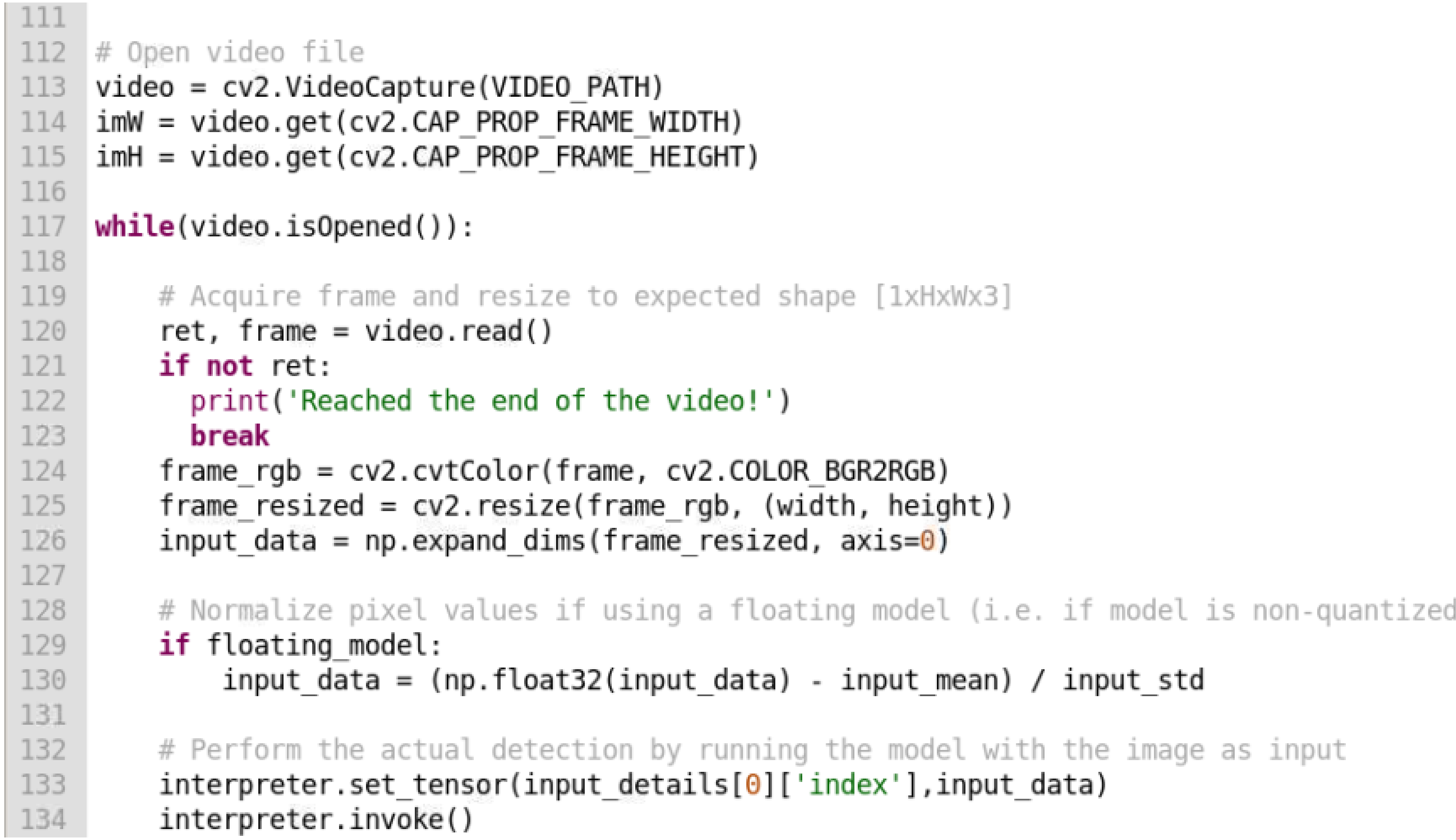


```

# Have to do a weird fix for label map if using the COCO "starter model" from
# https://www.tensorflow.org/lite/models/object_detection/overview
# First label is '???', which has to be removed.
if labels[0] == '???':
    del(labels[0])

# Load the Tensorflow Lite model.
# If using Edge TPU, use special load_delegate argument
if use_TPU:
    interpreter = Interpreter(model_path=PATH_TO_CKPT,
                              experimental_delegates=[load_delegate('libedgetpu.so.1.0')]
    print(PATH_TO_CKPT)
else:
    interpreter = Interpreter(model_path=PATH_TO_CKPT)

interpreter.allocate_tensors()

# Get model details
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()
height = input_details[0]['shape'][1]
width = input_details[0]['shape'][2]

floating_model = (input_details[0]['dtype'] == np.float32)

input_mean = 127.5
input_std = 127.5
```

As it says, opens video

What to do when the video is opened”

Basically just open video with a frame on a video player

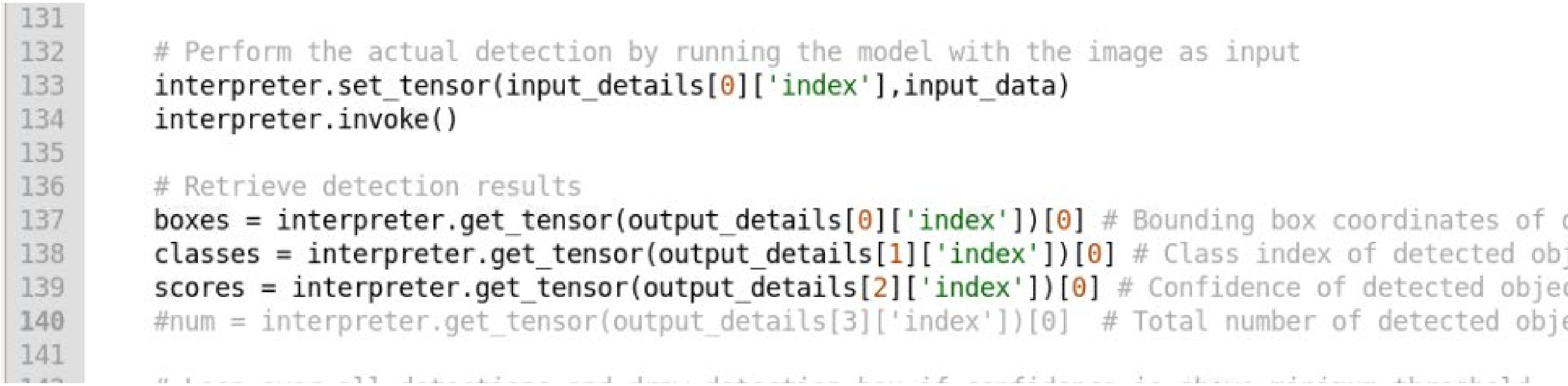


```
# Open video file
video = cv2.VideoCapture(VIDEO_PATH)
imW = video.get(cv2.CAP_PROP_FRAME_WIDTH)
imH = video.get(cv2.CAP_PROP_FRAME_HEIGHT)

while(video.isOpened()):

    # Acquire frame and resize to expected shape [1xHxWx3]
    ret, frame = video.read()
    if not ret:
      print('Reached the end of the video!')
      break
    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    frame_resized = cv2.resize(frame_rgb, (width, height))
    input_data = np.expand_dims(frame_resized, axis=0)

    # Normalize pixel values if using a floating model (i.e. if model is non-quantized
    if floating_model:
        input_data = (np.float32(input_data) - input_mean) / input_std

    # Perform the actual detection by running the model with the image as input
    interpreter.set_tensor(input_details[0]['index'],input_data)
    interpreter.invoke()
```

As it says, run the model on each frame

Gets the data and stores them in variable that corresponds to the data stored in output_details

As you can see, with each detection a coordinate of the detected object, a class aka label, and a confidence score is returned

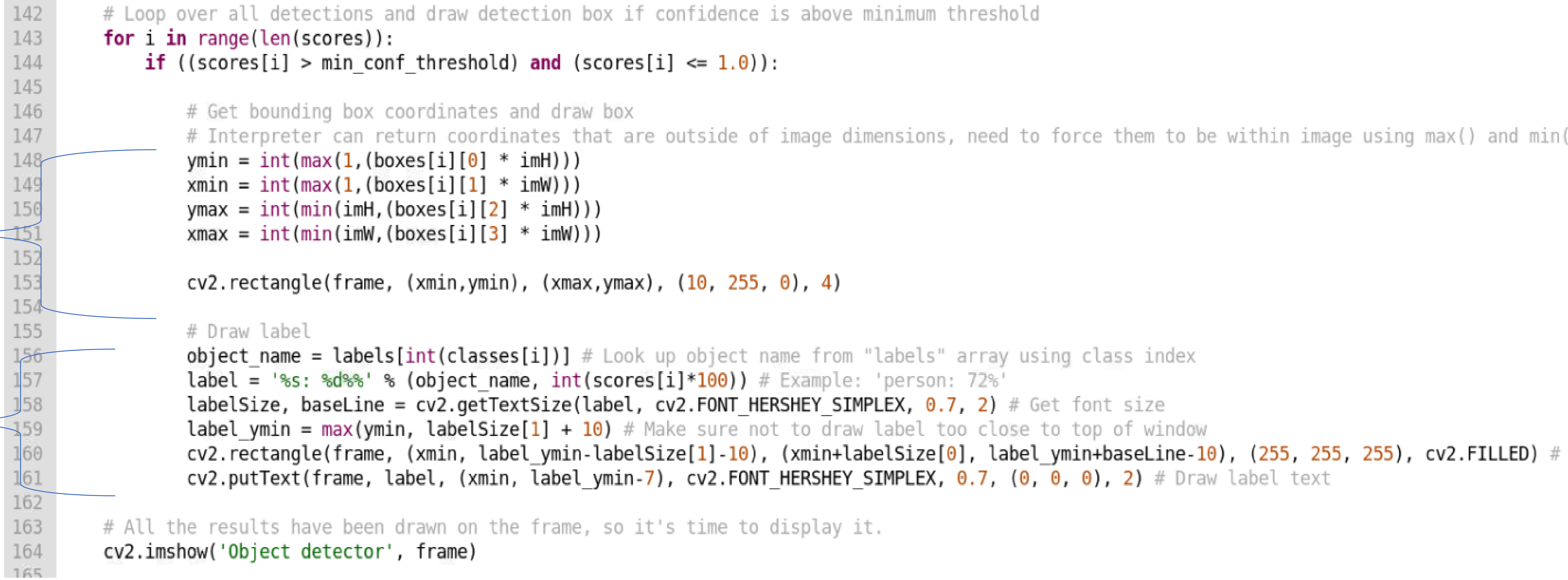


```

        # Perform the actual detection by running the model with the image as input
        interpreter.set_tensor(input_details[0]['index'],input_data)
        interpreter.invoke()

        # Retrieve detection results
        boxes = interpreter.get_tensor(output_details[0]['index'])[0] # Bounding box coordinates of
        classes = interpreter.get_tensor(output_details[1]['index'])[0] # Class index of detected obj
        scores = interpreter.get_tensor(output_details[2]['index'])[0] # Confidence of detected objec
        #num = interpreter.get_tensor(output_details[3]['index'])[0]  # Total number of detected obje

```

This is only for visual feedback while running the algorithm on the video. It draws the boxes around the detected object.
Here we can modify the threshold.

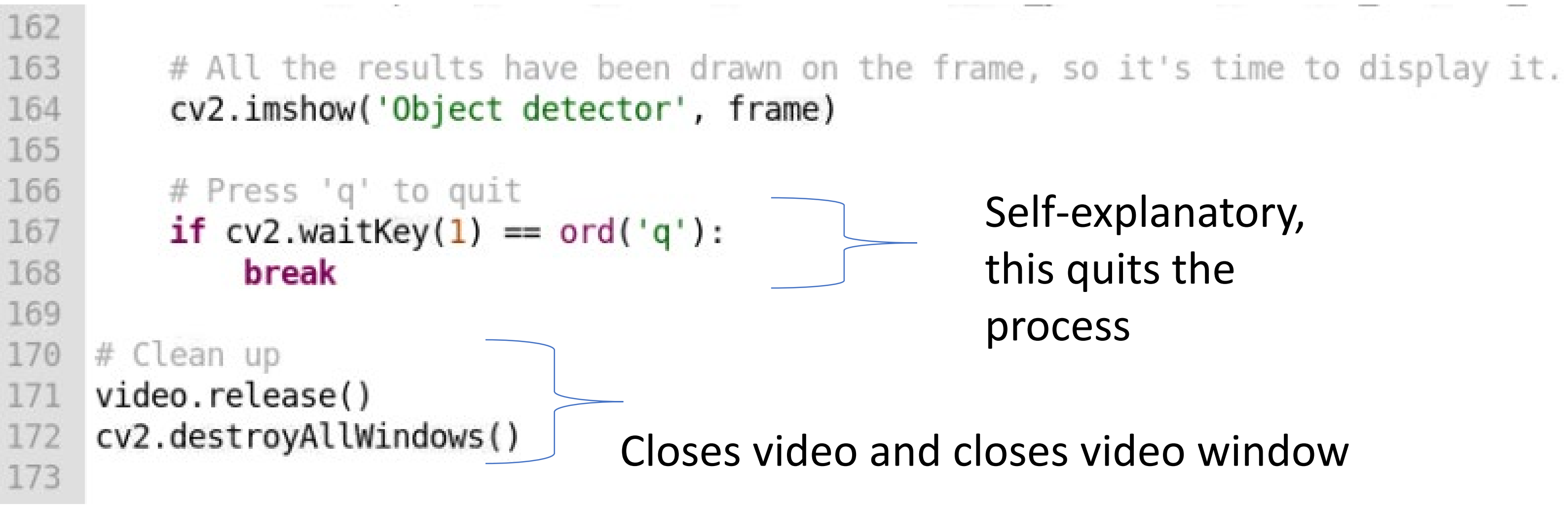
Just drawing boxes on the object using opencv

Putting label next to the box with opencv



```
# Loop over all detections and draw detection box if confidence is above minimum threshold
for i in range(len(scores)):
    if ((scores[i] > min_conf_threshold) and (scores[i] <= 1.0)):

        # Get bounding box coordinates and draw box
        # Interpreter can return coordinates that are outside of image dimensions, need to force them to be within image using max() and min(
        ymin = int(max(1,(boxes[i][0] * imH)))
        xmin = int(max(1,(boxes[i][1] * imW)))
        ymax = int(min(imH,(boxes[i][2] * imH)))
        xmax = int(min(imW,(boxes[i][3] * imW)))

        cv2.rectangle(frame, (xmin,ymin), (xmax,ymax), (10, 255, 0), 4)

        # Draw label
        object_name = labels[int(classes[i])] # Look up object name from "labels" array using class index
        label = '%s: %d%%' % (object_name, int(scores[i]*100)) # Example: 'person: 72%'
        labelSize, baseLine = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX, 0.7, 2) # Get font size
        label_ymin = max(ymin, labelSize[1] + 10) # Make sure not to draw label too close to top of window
        cv2.rectangle(frame, (xmin, label_ymin-labelSize[1]-10), (xmin+labelSize[0], label_ymin+baseLine-10), (255, 255, 255), cv2.FILLED) #
        cv2.putText(frame, label, (xmin, label_ymin-7), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 0), 2) # Draw label text

# All the results have been drawn on the frame, so it's time to display it.
cv2.imshow('Object detector', frame)
```

Opencv first draws then displays what it drew, this simply calls it to display the boxes. Similar to how LCD screens work in displaying images.



```
    # All the results have been drawn on the frame, so it's time to display it.
    cv2.imshow('Object detector', frame)

    # Press 'q' to quit
    if cv2.waitKey(1) == ord('q'):
        break

# Clean up
video.release()
cv2.destroyAllWindows()
```

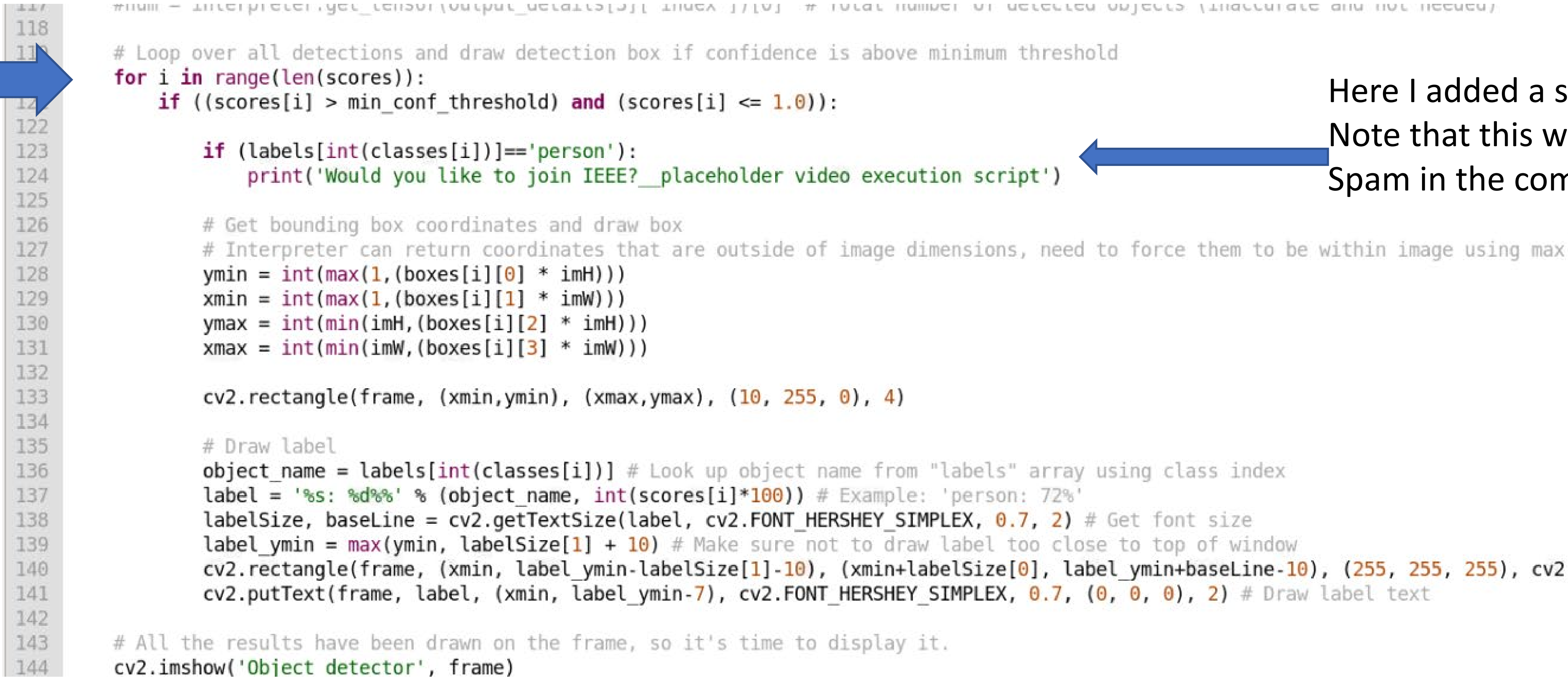
Self-explanatory, this quits the process

Closes video and closes video window

That’s it, now to modify.

Where to modify

Within this loop since this is where the objects are being put through the recognition algorithm



```
#num = interpreter.get_tensor(output_details[3]['index'])[0]  # Total number of detected objects (inaccurate and not needed)

# Loop over all detections and draw detection box if confidence is above minimum threshold
for i in range(len(scores)):
    if ((scores[i] > min_conf_threshold) and (scores[i] <= 1.0)):

        if (labels[int(classes[i])]=='person'):
            print('Would you like to join IEEE?__placeholder video execution script')

        # Get bounding box coordinates and draw box
        # Interpreter can return coordinates that are outside of image dimensions, need to force them to be within image using max
        ymin = int(max(1,(boxes[i][0] * imH)))
        xmin = int(max(1,(boxes[i][1] * imW)))
        ymax = int(min(imH,(boxes[i][2] * imH)))
        xmax = int(min(imW,(boxes[i][3] * imW)))

        cv2.rectangle(frame, (xmin,ymin), (xmax,ymax), (10, 255, 0), 4)

        # Draw label
        object_name = labels[int(classes[i])] # Look up object name from "labels" array using class index
        label = '%s: %d%%' % (object_name, int(scores[i]*100)) # Example: 'person: 72%'
        labelSize, baseLine = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX, 0.7, 2) # Get font size
        label_ymin = max(ymin, labelSize[1] + 10) # Make sure not to draw label too close to top of window
        cv2.rectangle(frame, (xmin, label_ymin-labelSize[1]-10), (xmin+labelSize[0], label_ymin+baseLine-10), (255, 255, 255), cv2
        cv2.putText(frame, label, (xmin, label_ymin-7), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 0), 2) # Draw label text

# All the results have been drawn on the frame, so it's time to display it.
cv2.imshow('Object detector', frame)
```

Here I added a s
Note that this w
Spam in the com

Even more considerations

- First go to slide 5 to re-read the considerations.
- The algorithm is frame-dependent meaning that it will detect something each frame(ideally, but some frames may be lost due to the Rpi performance limitations)
- Consider adding thresholds or triggers to make actions.
- Most of it is working on the logic on detecting multiple things and to avoid conflict. Also precedence in objects. Use placeholder text for any actions. Example: if a person is detected and we want it to turn on LEDs or display something on the screen, but we still don’t have the screen programming working we can simply put something like print(“Show IEEE video and move arms in wave motion”) for now. We will later modify this.
- I would not suggest pure if and else statements, maybe case or any other variations as for the else and if statements Python has to check every single condition which reduces performance and response time.

Suggestions on coding

- While I am not a master programmer, nor a very adept one. I do realize that the practice of using pure if and else statements is a beginner thing to do and not very efficient when running software that monitors each frame.
- I would suggest doing some research if not yet familiar on alternate methods such as case statements or dictonaries, etc.
- Either way test the performance of each implementation with the script running on a test video file.
- You can also edit the code on your own computer and then paste it on the raspberry pi since I know sometimes the input delay can be a bit tedious.

Videos to use

- You are free to use your own videos. You can download them on the raspberry pi if you have the link, you can simply paste it on the browser or you can send it through VNCviewer file transfer.
- I downloaded stock footage which I know is not the best thing to test it on, so I suggest recording your own footage if possible and get friends to help by appearing in front of it or at least part of them.
- Make sure to include a filename on the script or just name it as test.mp4