# Observations aside from the comments in opencv-intro.py

-anu, username: auppulur, id: 025152445

## How to run the code

* Folder containing the code should also contain the frames folder
* Press any key to see the next output and to eventually end the program

## Part 1

1. Reference for getting at the parent dir path with import os: [Get parent of current directory using python](https://www.geeksforgeeks.org/get-parent-of-current-directory-using-python/)
2. [How to use os.join](https://www.geeksforgeeks.org/python-os-path-join-method/)
3. pip freeze to check all the packages installed
4. [cv.imread()](https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_gui/py_image_display/py_image_display.html)
5. While showing the image refer [the mysteries of cv.waitkey()](https://stackoverflow.com/questions/35372700/whats-0xff-for-in-cv2-waitkey1)
6. im1.shape() throws an exception: TypeError 'tuple' object is not callable
7. removing the parenthesis fixed this as shape is an attribute not a method

### Part 1 inline question 1

1. The sizes of the dimensions mean that the image read in is a color image of the BGR (blue-green-red) format used by OpenCV.
2. Here it is (240, 320, 3) forming a 3-D matrix where the first dimension states that the image is 240 pixels high, the second dimension states that the width of the image is 320 pixels and the third dimension is for the 3 color channels.
3. Alternatively, it can be thought of as a collection of three 2D matrices – one for each color channel – in each of which, each and every value represents the respective color intensity of the corresponding pixel in the image in the interval [0,225].

### Part 1 inline question 2

1. Printing the image itself yields
2. [[[217 191 167]
3. [219 191 167]
4. [221 191 166]
5. ...
6. [227 193 169]
7. [226 190 166]
8. [226 190 166]]
9. ...
10. [[ 64 78 77]
11. [ 59 73 72]
12. [ 55 69 68]
13. ...
14. [ 18 0 0]
15. [ 21 0 0]
16. [ 23 2 1]]]
17. where each entry in each populated line (lines 10, 11, 12, 14, etc.) represents the pixel color intensity of each of the BGR color channels, for e.g., in line 10, [217 191 167] means the color intensities for the first pixel in the image are Blue = 217, Green = 191, Red = 167.
18. Since the height of the image is determined by the number of rows of pixels in each channel, the first “…” on line 13 would expand into 240 rows.
19. And since the width of the image is determined by the number of columns of pixels in each channel, the second “…” on line 18 would expand into 320 blocks.

## Part 2

1. What was tricky in this part was to figure out the axis along which the mean of all the frames was to be computed; I figured that this being a 4-D array, since there is more than one image, the first dimension would have to refer to the number of images. And hence I averaged along the first axis, axis=0. Reference: [numpy.mean](https://docs.scipy.org/doc/numpy/reference/generated/numpy.mean.html#numpy.mean)
2. [np.mean solution](https://stackoverflow.com/questions/35668074/how-i-can-take-the-average-of-100-image-using-opencv) along with astype(‘uint8’).

### Part 2 inline question

1. Why have the cars disappeared from the “background” image? At first glance, that is because the cars were constantly moving from frame to frame. The chance of there being a moving car pixel in any given frame is much less than the constant background of the freeway and the trees. And since the mean would essentially average out the infrequent pixel values from the stack of pixel value distributions, that is the effect we see.

## Part 3

1. OpenCV function to compute the absolute difference: [cv.absdiff()](https://docs.opencv.org/3.4.1/d2/de8/group__core__array.html#ga6fef31bc8c4071cbc114a758a2b79c14)
2. The right way to use OpenCV flags is, for example, cv.IMREAD\_GRAYSCALE
3. In picking the threshold value used to classify the pixel values in simple binary thresholding, the median value 127 would miss out on most of the cars, 63 would miss out on a few cars, and 32 would include some unnecessary spots. Hence, I picked 40.

### Part 3 inline question

1. The difference between the Otsu’s adaptive thresholding method and the simple thresholding one is that Otsu’s automatically picks a more precise thresholding value based on the average of the peaks of the histogram of the binary image data. This leads to less loss of the target data in the final image and obviates the need for us to do considerable trial-and-error in determining such a threshold value.
2. As also illustrated in the tutorial, noise filtering improves the result of Otsu’s method.

## Part Bonus

1. Figured out that we should use ideas like the absolute difference from the previous parts to get the desired output.
2. Also figured out that we shouldn’t draw the contours but the bounding boxes and that the color of the bounding boxes should be specified in
3. cv.rectangle(thresh,(x,y),(x+w,y+h),(255,255,255),2)