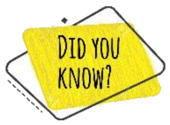
Getting Started with Python Programming Languages

Group Members:

1. Lim Zhou Yi (Project Leader)
2. Chua Jiaming
3. Chan Ting Chang
4. Ivan Lee Kang Zheng
5. Khong Fan Yi
6. Tan Yi Fan

First of all, do you guys know that Python programming languages does not need to declare the variable type?



First Section – Introduction to Python’s List Operations

Lists => Store an indexed list of items (like arrays in C++)

*Please copy the code below to your Python compiler

Example 1:

```
nums = [7, 7, 3, 10]
nums[2] = 5
print(nums)                    #Output: [7, 7, 5, 10]
```

Explanation: The number in the list number 2 (position 3) which is 3 will be replace with 5.

Example 2:

```
name = [“Emily”, “Andrew”, “Lily”]
print(“Andrew” in name)        #Output: True
print(“John” in name)          #Output: False
```

Explanation: This will check whether the items is in the list and if the item is in the list, it will return True else return False.

Example 3:

```
nums = [1, 2, 3]

nums.append(4)

print(nums)           #Output: [1, 2, 3, 4]
```

Explanation: This append keyword in Python is used to add the number to the end of the list.

Example 4:

```
words = ["Python", "fun"]

words.insert(1, "is")

print(words)          #Output: ['Python', 'is', 'fun']
```

Explanation: With this insert keyword in Python, it is quite similar to append but the only difference is it can state the position that we want to add in the list.

Second Section – Dictionaries Function in Python

Dictionaries => Map keys to value

Example 1:

```
ages = {"Emily": 24, "Andrew": 20}

print(ages["Emily"])      #Output: 24
```

Explanation: This will print the age of the person that we want.

*Note that: Please do not make your key this way => [1, 2, 3], it will not work

Example 2:

```
square = {1: 1, 3: "error", 4: 16}

square[8] = 64

square[3] = 9

print(square)        #Output: {8: 64, 1: 1, 3: 9, 4: 16}
```

Explanation: First, we will create key number 8 with the value of 64, then we will replace key number 3 which is "error" with the value of 9.

*Note that: the number in the list [ ], is the key number, not the position of the list.

Third Section – If-else statement and Looping Structures in Python

*Note that: The if-else and while loop statement are similar to C++, the only differences is we do not need to use bracket for the parameter and also the open & close curly brackets but then we need to replace the open curly bracket with colon => :

Example 1: If-else statement

```
if 10 > 5:

    print("10 is greater than 5")

print("Program ended")
```

Explanation: For the first print command, it belongs to the if statement as it is indented, but for the second print command, it is outside the if statement.

*Note that: We can use elif as a shortcut keyword to replace else if statement only in Python.

Example 2: While loop statement

```
count = 0
while count < 4:
    print("Hello World")
    count = count + 1
```

Explanation: This example is exactly same with C++, the while-loop statement will check the counter each time and when the counter is less than 4, it will stop the loop.

Example 2: For loop statement

```
for count in range (2, 5)      # =>for (int i = 2; i < 5; i ++) in C++
    print(count)               #Output: 2   3   4
```

Explanation: In this for-loop statement, we set the range from 2 to less than 5, so that we will get 2 3 4 as an output.

Example 3: For loop statement

```
for count in range[5]          # => for (int i = 0; i <= 5; i ++) in C++
    print["Hello World!"]
```

Explanation: We set the range from 0 to less than or equal to 5, so that the "Hello World!" will be printed out repeatedly for 5 times.

Example 4: For loop statement

```
for name in ["Jack", "Andrew", "Emily"]
    print ("The student is:", name)
```

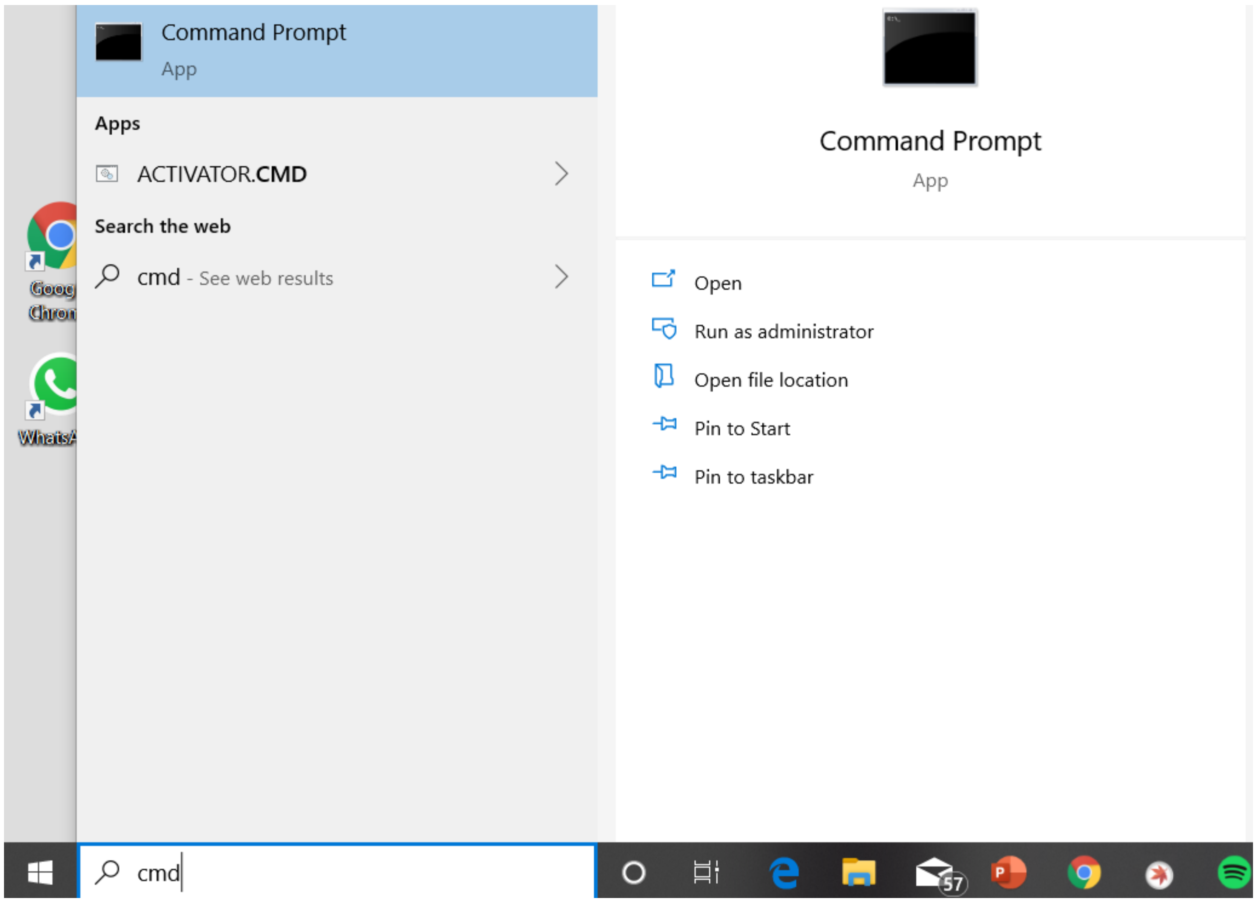
Explanation: This is a special for-loop statement in Python, it will print each of the item in the list depends the number of items you have in it. This for-loop statement will stop looping until it finishes print out all the items in the list.

*Note that: In Python programming languages, it does not support do… while loop statement unlike C++ programming languages

Last Section – Human Tracking System

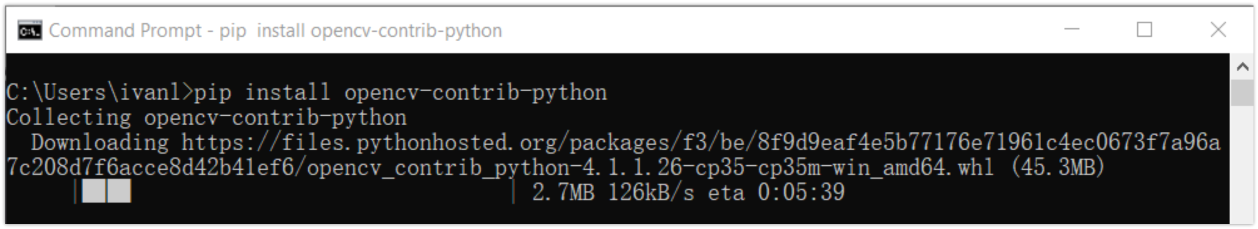
Steps To Run Human Tracking Program

1) Run Command Prompt



*You can open cmd through many other ways but this is the simplest. [ALTERNATIVE: Press **win + R**, then type **in cmd**, then press **Enter**]

2) Install opencv-contrib-python in cmd **(THIS CAN BE SKIPPED AS WE DONE FOR YOU)**



```
C:\Users\ivanl>pip install opencv-contrib-python
Collecting opencv-contrib-python
  Downloading https://files.pythonhosted.org/packages/f3/be/8f9d9eaf4e5b77176e71961c4ec0673f7a96a
7c208d7f6acce8d42b41ef6/opencv_contrib_python-4.1.1.26-cp35-cp35m-win_amd64.whl (45.3MB)
     |██                                | 2.7MB 126kB/s eta 0:05:39
```

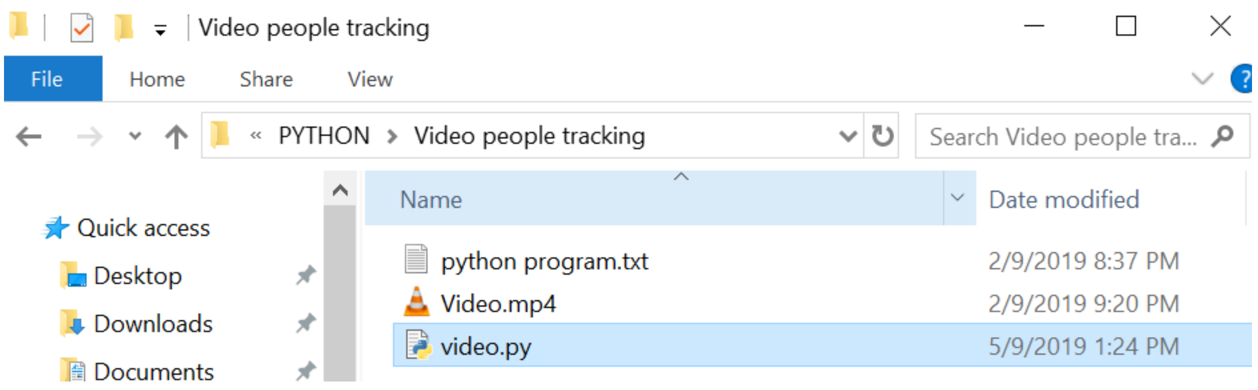
*opencv-contrib-python is a wrapper package for OpenCV python bindings. OpenCV-Python is a library of Python bindings designed to solve computer vision problems.

*EXTRA KNOWLEDGE: Actually Python is slower than languages like C and C++, but it is able to extended with C and C++, we can write computationally intensive code in C and C++ and create Python wrappers that can be used as Python modules, this made the code run as fast as C++ code (since it is the actual C and C++ code running in the background) ~

3) Open the 'PYTHON' folder and then open the 'Video people tracking' folder on the desktop.

4) There are 2 ways to run the program.

a) **Double click** and run the **‘video.py’**.

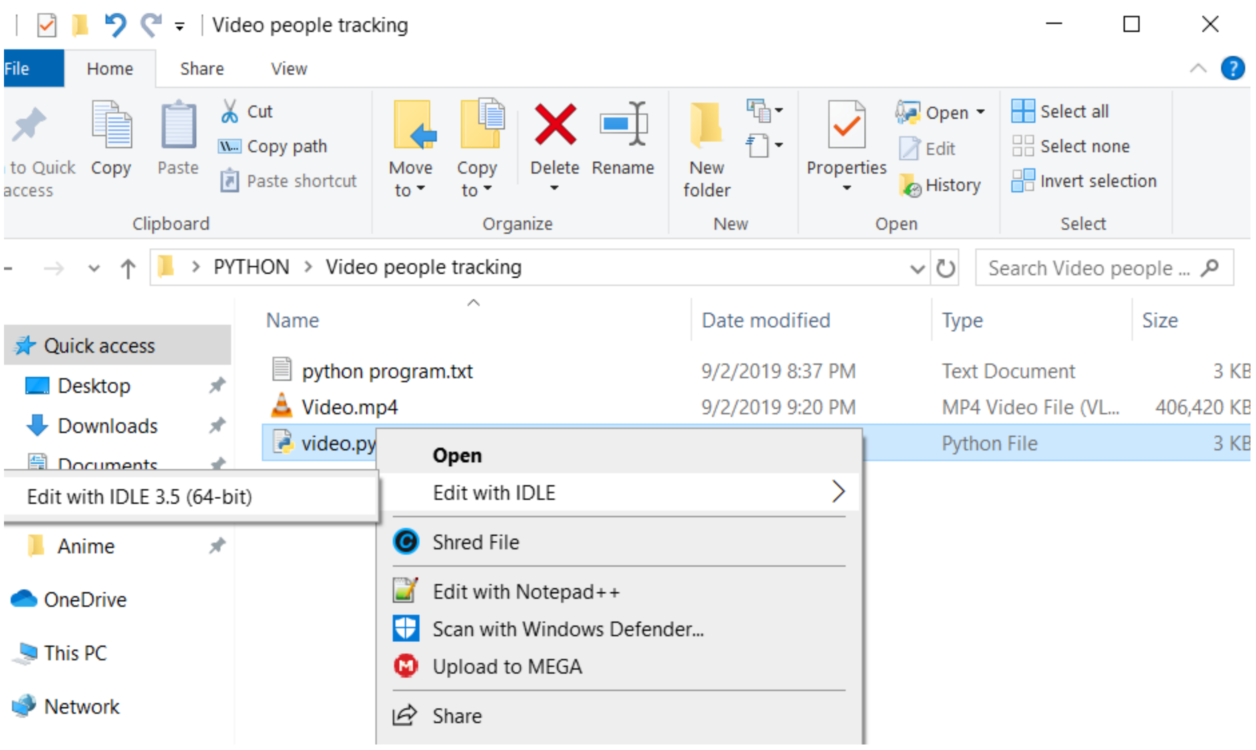


Name	Date modified
python program.txt	2/9/2019 8:37 PM
Video.mp4	2/9/2019 9:20 PM
video.py	5/9/2019 1:24 PM

b) **Right click** the **‘video.py’**.

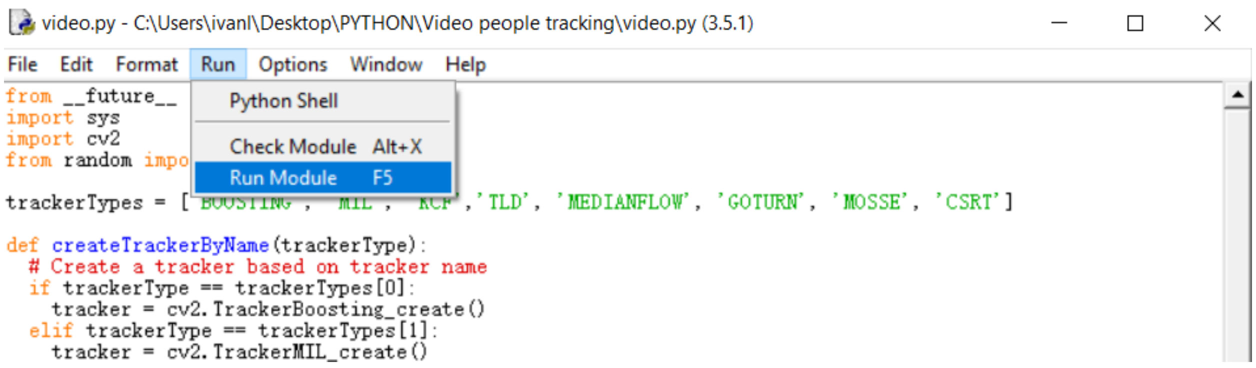
Click **‘Edit with IDLE’**, then click **‘Edit with IDLE 3.5’**.

*Please do make sure that you choose Edit with **IDLE 3.5.**



Name	Date modified	Type	Size
python program.txt	9/2/2019 8:37 PM	Text Document	3 KB
Video.mp4	9/2/2019 9:20 PM	MP4 Video File (VL...	406,420 KB
video.py		Python File	3 KB

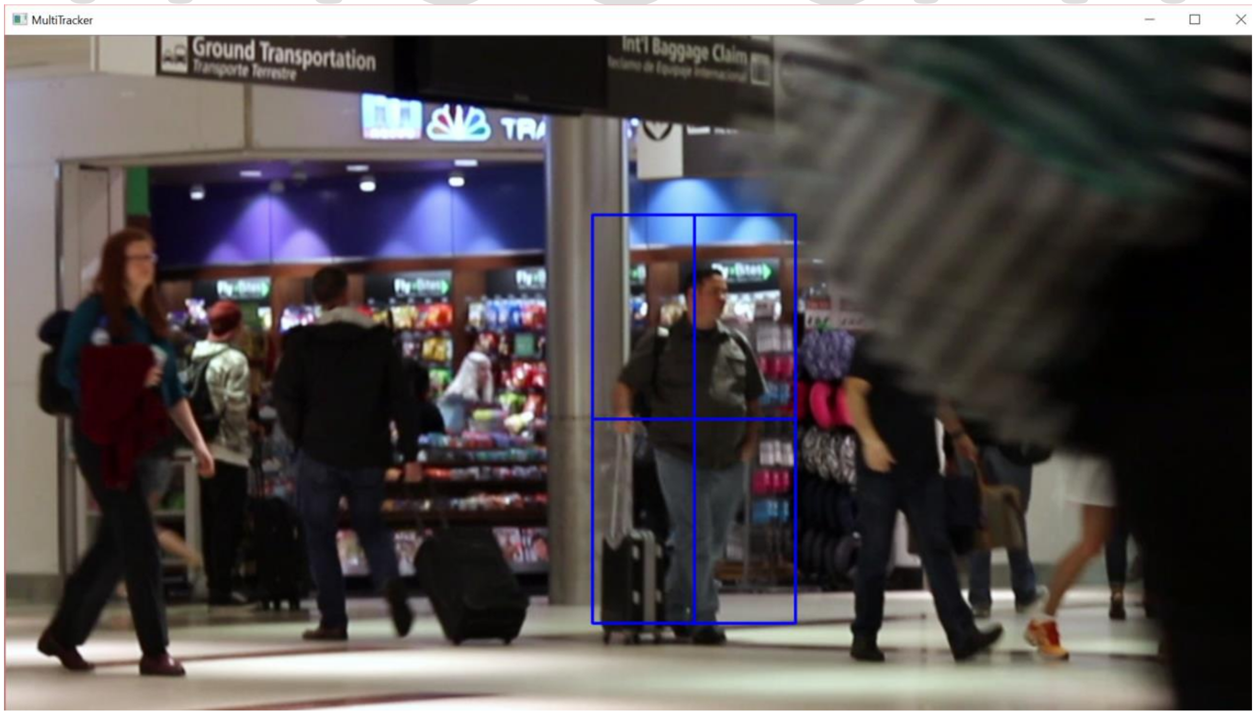
Click **'Run'** tab and click **'Run Module'** on the top of the window or press **'F5'** to run the program.


video.py - C:\Users\ivanl\Desktop\PYTHON\Video people tracking\video.py (3.5.1)
File
Edit
Format
Run
Options
Window
Help
Python Shell
Check Module Alt+X
Run Module F5

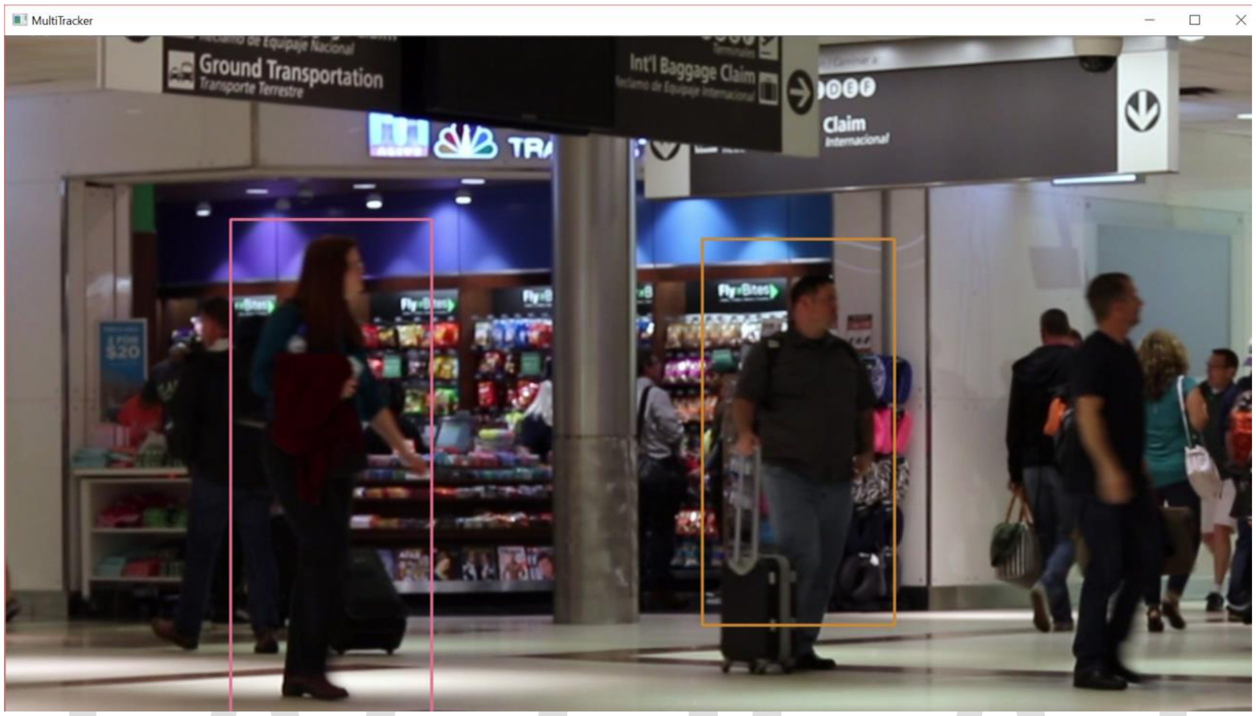
```
from __future__
import sys
import cv2
from random impo

trackerTypes = [ BOOSTING', 'MIL', 'KCF','TLD', 'MEDIANFLOW', 'GOTURN', 'MOSSE', 'CSRT']

def createTrackerByName(trackerType):
  # Create a tracker based on tracker name
  if trackerType == trackerTypes[0]:
    tracker = cv2.TrackerBoosting_create()
  elif trackerType == trackerTypes[1]:
    tracker = cv2.TrackerMIL_create()
```

5) A window which displaying a paused video will pop out, click and draw a rectangle to select the people you want to track.

Press **'Enter' TWICE** after done selecting the first people (rectangle drew will disappear, but don't worry, the people selected is still selected), **repeat STEP 5 to select next people**. After done selecting people, press **'Enter' ONCE** and press **'Q' ONCE** to exit the selecting process**.**



6) The video will run and the people selected will be tracked in the video ~



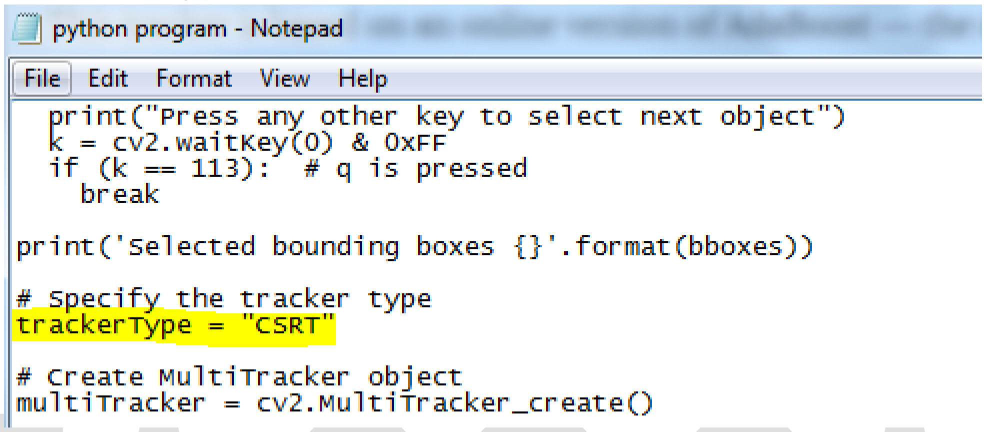
7) To terminate the program, press **'Esc'** and click **'X'** on the right top of the window.

Ps: Sometimes you may close the program straight after pressing **'Esc'**.

*Do remember to press **'Esc' before** you click the **'X'** button, else the program is going to reboot itself again.

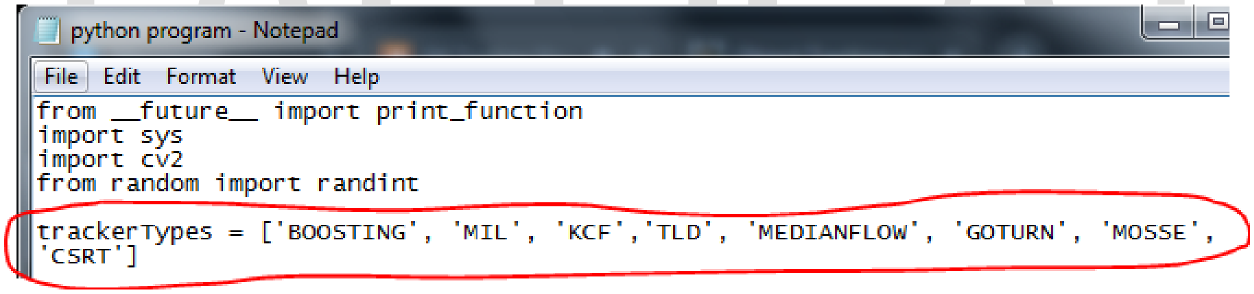
EXTRA 1: How to change the tracker types in the program.

1. Edit the program with notepad or others software. Find the line which stated the tracker types.


python program - Notepad

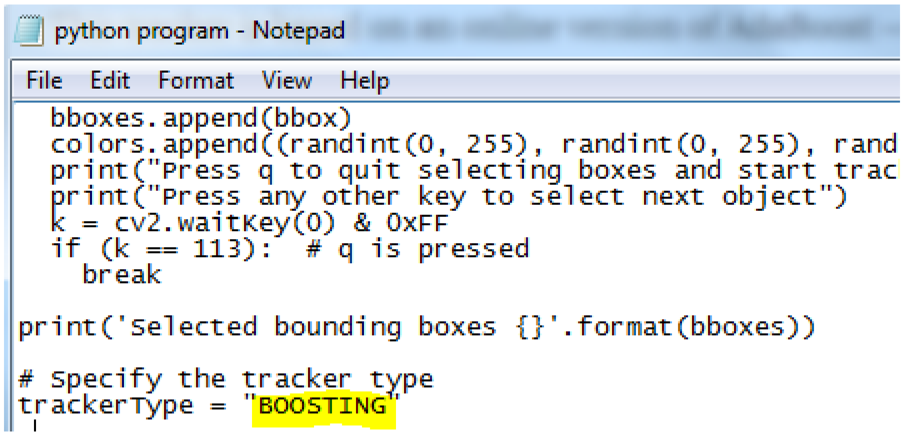
```
    print("Press any other key to select next object")
    k = cv2.waitKey(0) & 0xFF
    if (k == 113):  # q is pressed
      break

print('Selected bounding boxes {}'.format(bboxes))

# Specify the tracker type
trackerType = "CSRT"

# Create MultiTracker object
multiTracker = cv2.MultiTracker_create()
```

2. Choose the tracker types from the types provided on the top of the code.


python program - Notepad

```
from __future__ import print_function
import sys
import cv2
from random import randint

trackerTypes = ['BOOSTING', 'MIL', 'KCF','TLD', 'MEDIANFLOW', 'GOTURN', 'MOSSE',
'CSRT']
```

3. Replace the original tracker type with another you prefer.


python program - Notepad

```
    bboxes.append(bbox)
    colors.append((randint(0, 255), randint(0, 255), rand
    print("Press q to quit selecting boxes and start trac
    print("Press any other key to select next object")
    k = cv2.waitKey(0) & 0xFF
    if (k == 113):  # q is pressed
      break

print('Selected bounding boxes {}'.format(bboxes))

# Specify the tracker type
trackerType = "BOOSTING"
```

4. Save the program file and run it as what we shown above ~ The tracker type is changed successfully.

EXTRA 2: The Pros and Cons of the different tracker types.

1. BOOSTING

   Pros: None. This algorithm is a decade old and works ok, but I could not find a good reason to use it especially when other advanced trackers (MIL, KCF) based on similar principles are available.

   Cons: Tracking performance is mediocre. It does not reliably know when tracking has failed.

2. MIL

   Pros: The performance is pretty good. It does not drift as much as the BOOSTING tracker and it does a reasonable job under partial occlusion. If you are using OpenCV 3.0, this might be the best tracker available to you.

   Cons: Tracking failure is not reported reliably. Does not recover from full occlusion.

3. KCF

   Pros: Accuracy and speed are both better than MIL and it reports tracking failure better than BOOSTING and MIL. If you are using OpenCV 3.1 and above, KCF is recommended for most applications.

   Cons: Does not recover from full occlusion. Not implemented in OpenCV 3.0.

4. TLD

   Pros: Works the best under occlusion over multiple frames. Also, tracks best over scale changes.

   Cons: Lots of false positives making it almost unusable.

5. MEDIANFLOW

   Pros: Excellent tracking failure reporting. Works very well when the motion is predictable and there is no occlusion.

   Cons: Fails under large motion.

6. MOSSE

   Pros: Operates at a higher fps (450 fps and even more).

   Cons: Lags behind the deep learning-based trackers on a performance scale. Loosing accuracy.

7. CSRT

   Pros: Higher accuracy for object tracking.

   Cons: Operates at a comparatively lower fps (25 fps).