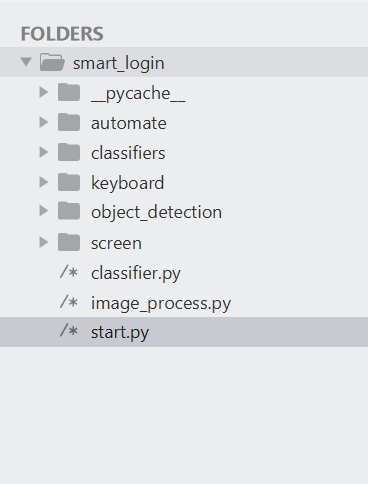
**README**

**FOLDER STRUCTURE: -**

Automation folder

****

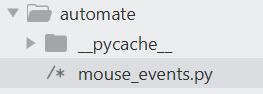
Screen to read and Process

Pretrained Object detectors

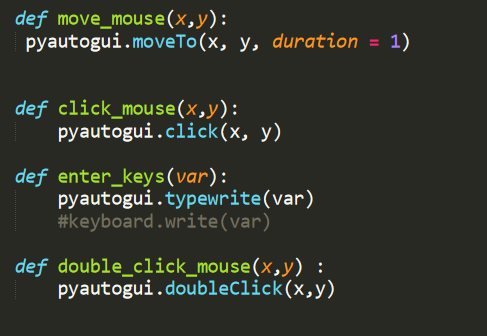
Keyboard library

Pretrained classifiers

**AUTOMATION FOLDER: -**

****

Consists of the automation script for mouse and keyboard events.

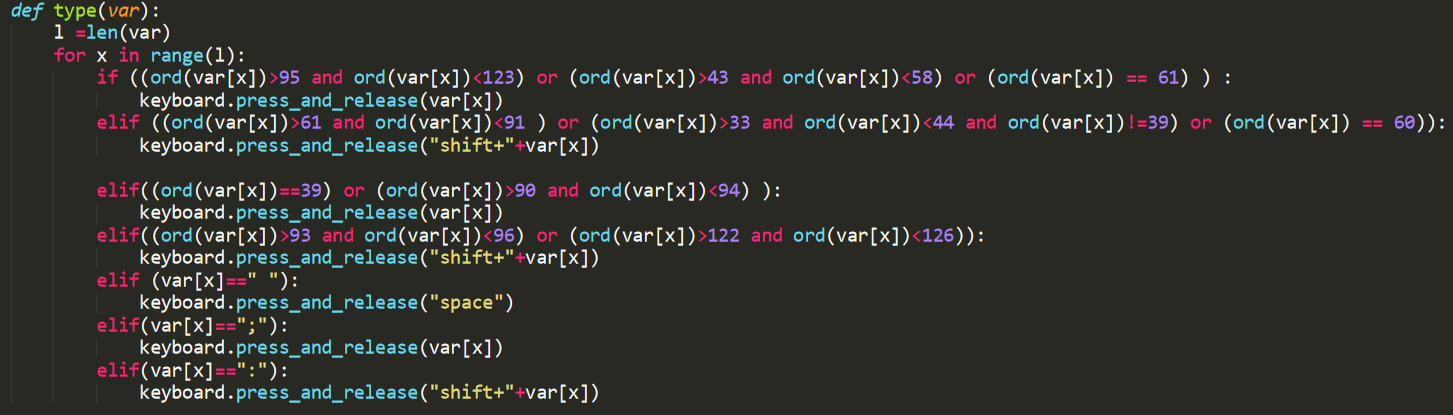


click Mouse

Enter keys of keyboard

Double click Mouse

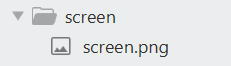
Moving Mouse

****

Typing keyboard keys classified based on ASCII values

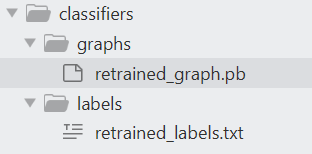
**SCREEN: -**

Contains the image for processing which is being written and read on runtime.

****

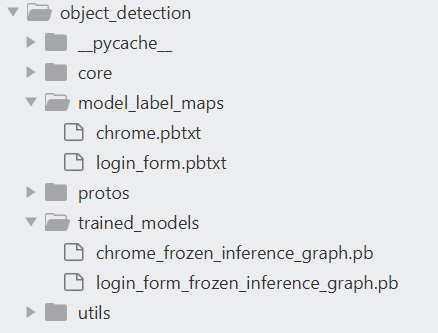
**PRETRAINED CLASSIFIERS: -**

Contains two sub-folders. Graphs and labels of the trained models to classify.

****

**PRETRAINED OBJECT DETECTORS: -**

Requirements for Object detection API

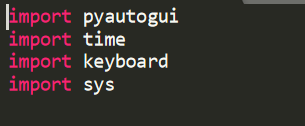
****

Trained models’ labels

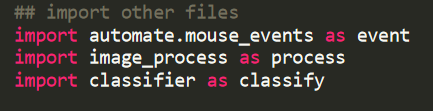
Trained models inference graphs

**START.PY: -**

1. **Importing libraries**

****

1. **Importing functions of other layers**

****

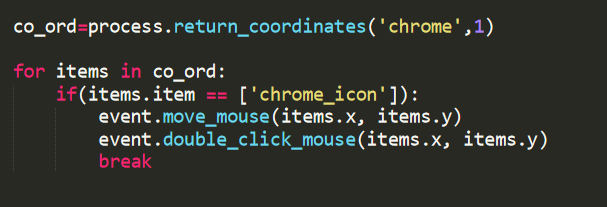
1. **Moving to desktop**

****

1. **Capturing screen and writing it**

****

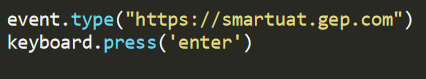
1. **Obtaining chrome coordinates and clicking it**

****

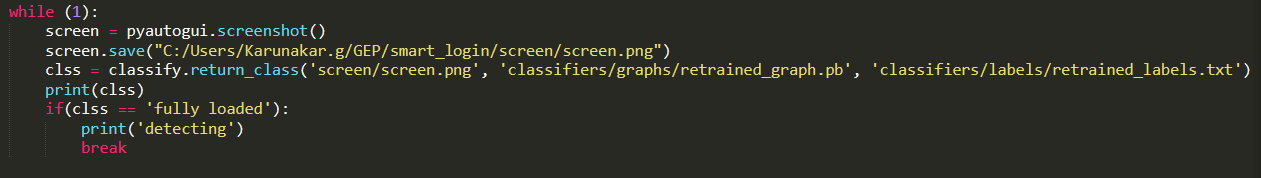
1. **Maximize the screen of chrome and open a new tab**

****

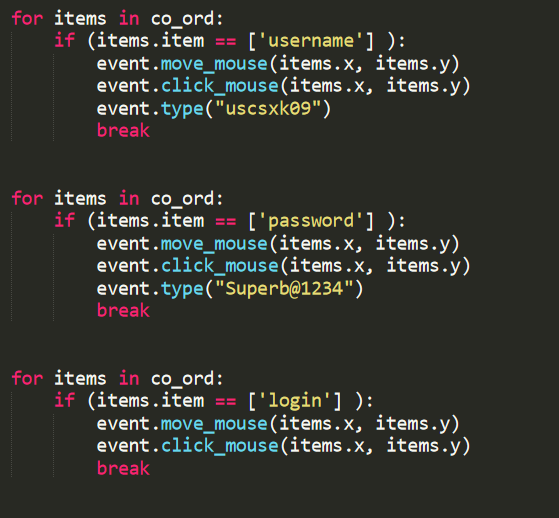
1. **Enter the URL and proceed**

****

1. **Waiting for the chrome to load**

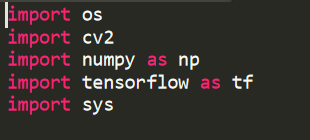
****

1. **Filling out the fields according to obtained coordinates.**

****

**IMAGE\_PROCESS.PY: -**

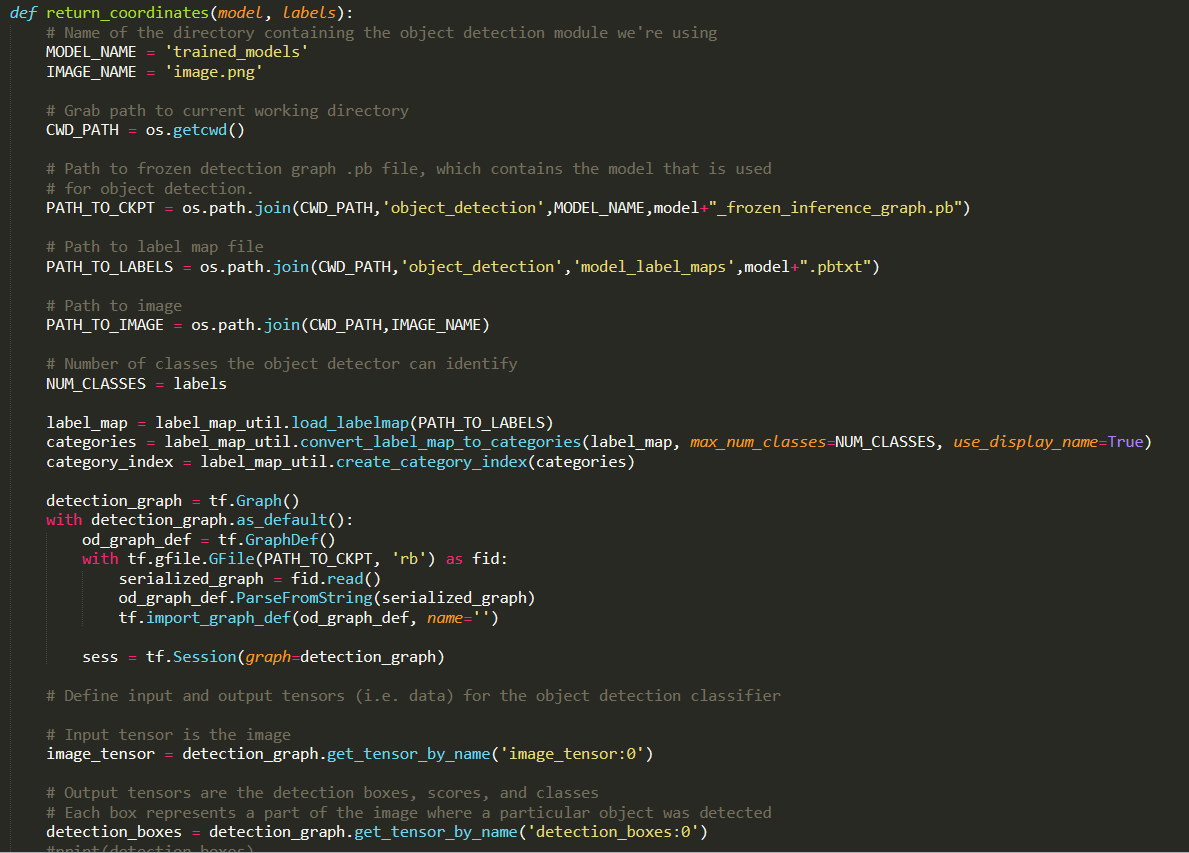
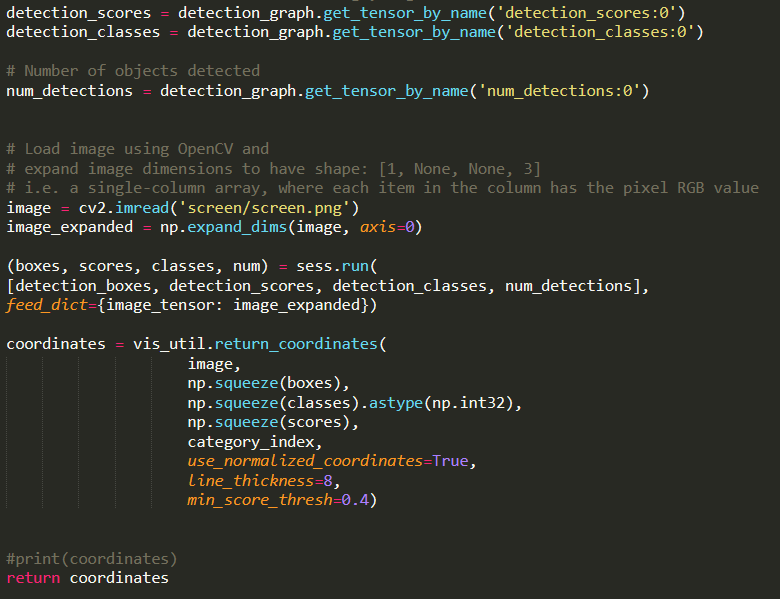
1. **Import Libraries**

****

1. **Utilities**

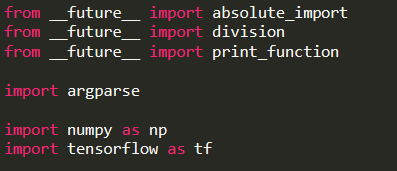
****

1. **Function to return co-ordinates**

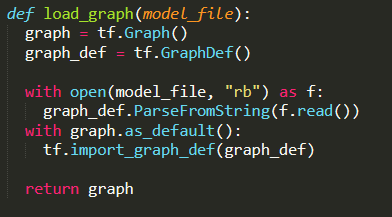
** **

**CLASSIFIER.PY: -**

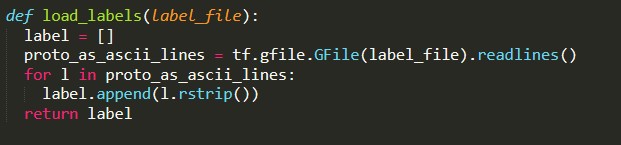
1. **Load libraries**

****

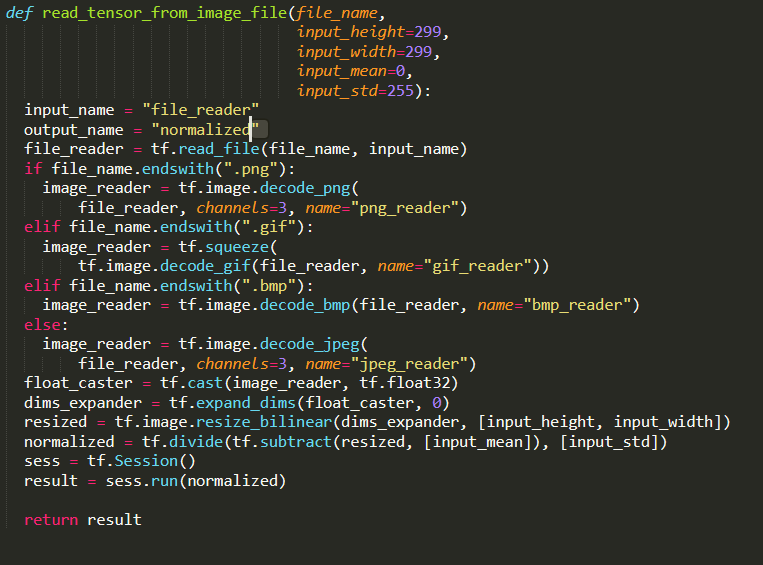
1. **Loading graph**

****

1. **Loading labels**

****

1. **Reading tensors**

****

1. **Return the class**

****

**TRAINING NEW MODEL: -**

There is Folder for training a new model named new training.

**STEPS: -**

1. Copy and rename the folder
2. Prepare your training data and place it in the Object\_detection/images/train (80%) and Object\_detection/images/test (20%)
3. Label the images with the classes to detect using labelimg
4. Open command prompt and navigate to your new model directory
5. Create csv files from labeled images … using command :- python xml\_to\_csv.py
6. open the generate\_tfrecord.py file in a text editor. Replace the label map starting at line 31 to your own classes.

**example :-** def class\_text\_to\_int(row\_label):

if row\_label == 'username':

return 1

elif row\_label == 'password':

return 2

elif row\_label == 'login':

return 3

else:

None

1. generate the TFRecord by using ..

python generate\_tfrecord.py --csv\_input=images\train\_labels.csv --image\_dir=images\train --output\_path=train.record

python generate\_tfrecord.py --csv\_input=images\test\_labels.csv --image\_dir=images\test --output\_path=test.record

1. IN object\_detection\training folder. (Make sure the file type is .pbtxt, not .txt !) In the text editor, copy or type in the label map in the format.
2. item {
3. id: 1
4. name: 'username'
5. }
6. item {
7. id: 2
8. name: ‘password'
9. }
10. item {
11. id: 3
12. name: 'login'
13. }

9 . In  \object\_detection\training directory open faster\_rcnn\_inception\_v2\_pets.config file and make the following changes

On Line 9 change the number of classes

On Line 103, 123, 125, 135,137 change the directory path according to your new path.

On Line 130 change num\_examples to the number of images you have in the \images\test directory.

10 . Then run the following

python train.py --logtostderr --train\_dir=training/ pipeline\_config\_path=training/faster\_rcnn\_inception\_v2\_pets.config

stop the training when loss comes under 0.05

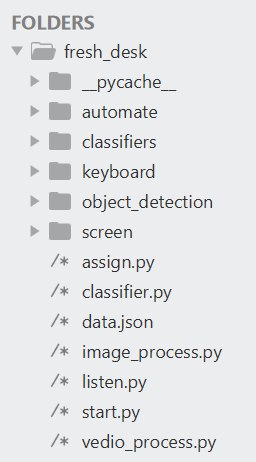
11 . export the inference graph using

python export\_inference\_graph.py --input\_type image\_tensor --pipeline\_config\_path training/faster\_rcnn\_inception\_v2\_pets.config --trained\_checkpoint\_prefix training/model.ckpt-XXXX --output\_directory inference\_graph

replace the above xxxx with the number of last cpkt file in training folder.

You are done .. now move the obtained inference graph file in inference graph folder and labelmap to your required folders.

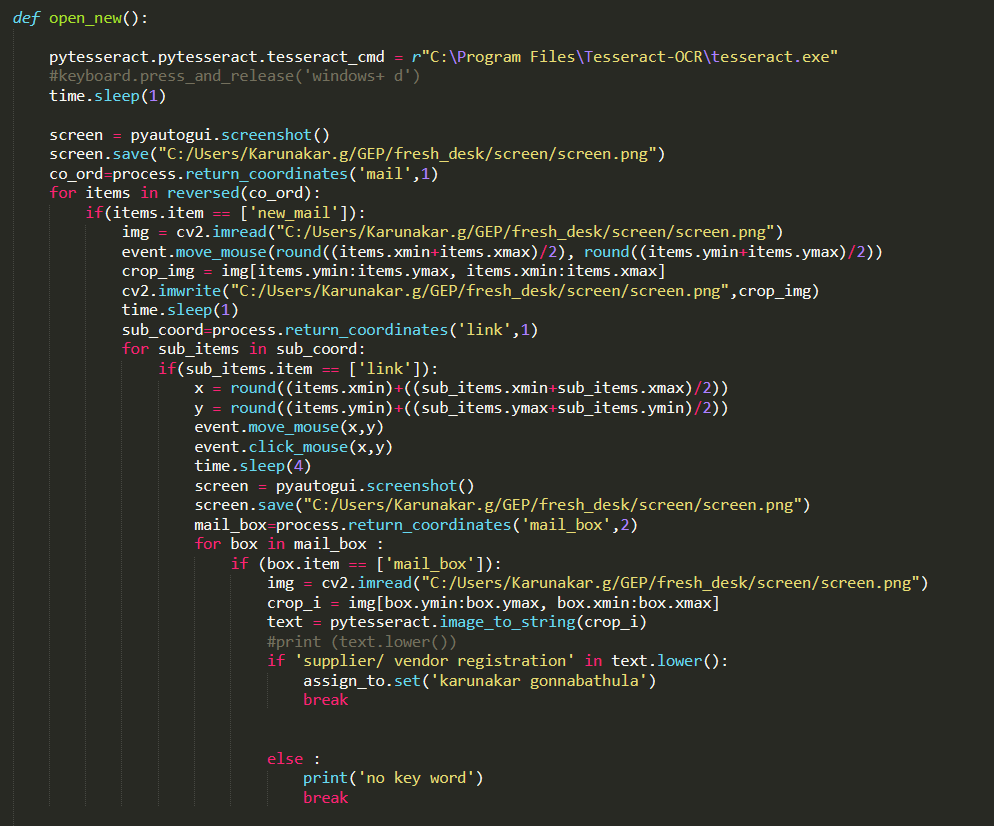
**USE CASE - II: -**

****

**LISTEN.PY** checking for new mails and there is one call for the start.py

****

**START.PY**

****

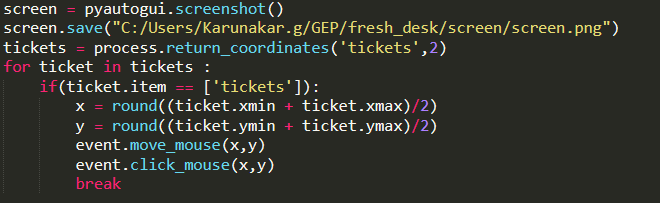
Call assign.py to assign if keyword

Feed mail box to tesseract

Detect mail box

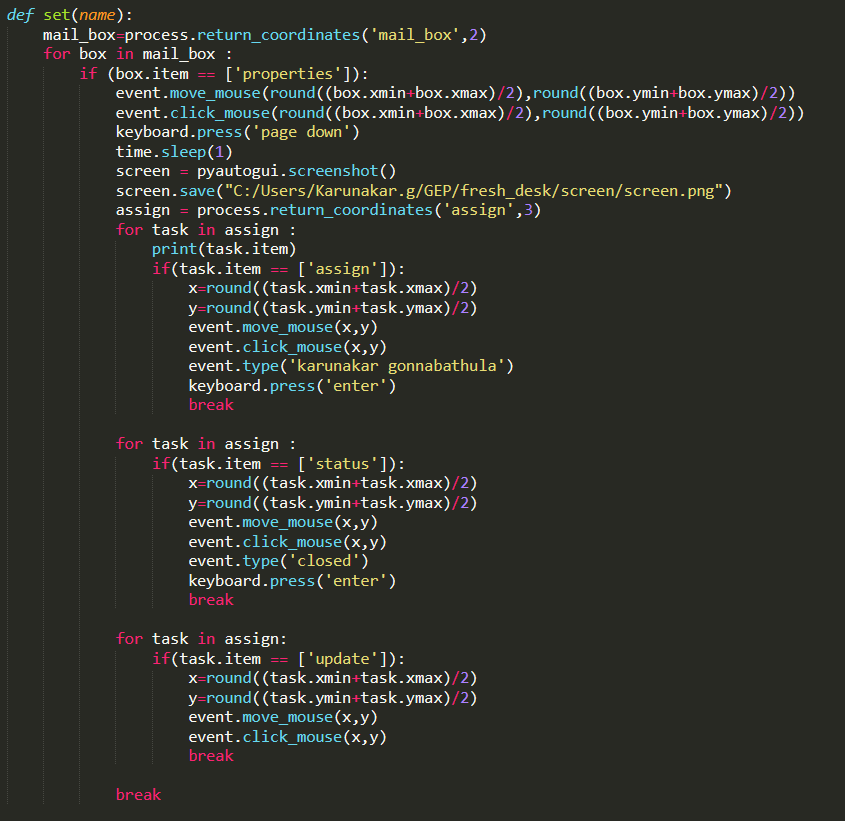
Detect link and open

Detect new mail

****

Return to tickets window

**ASSIGN.PY: -**

****

Find update and update it

Find status and close it

Find assign and allot person

Find Properties and scroll down