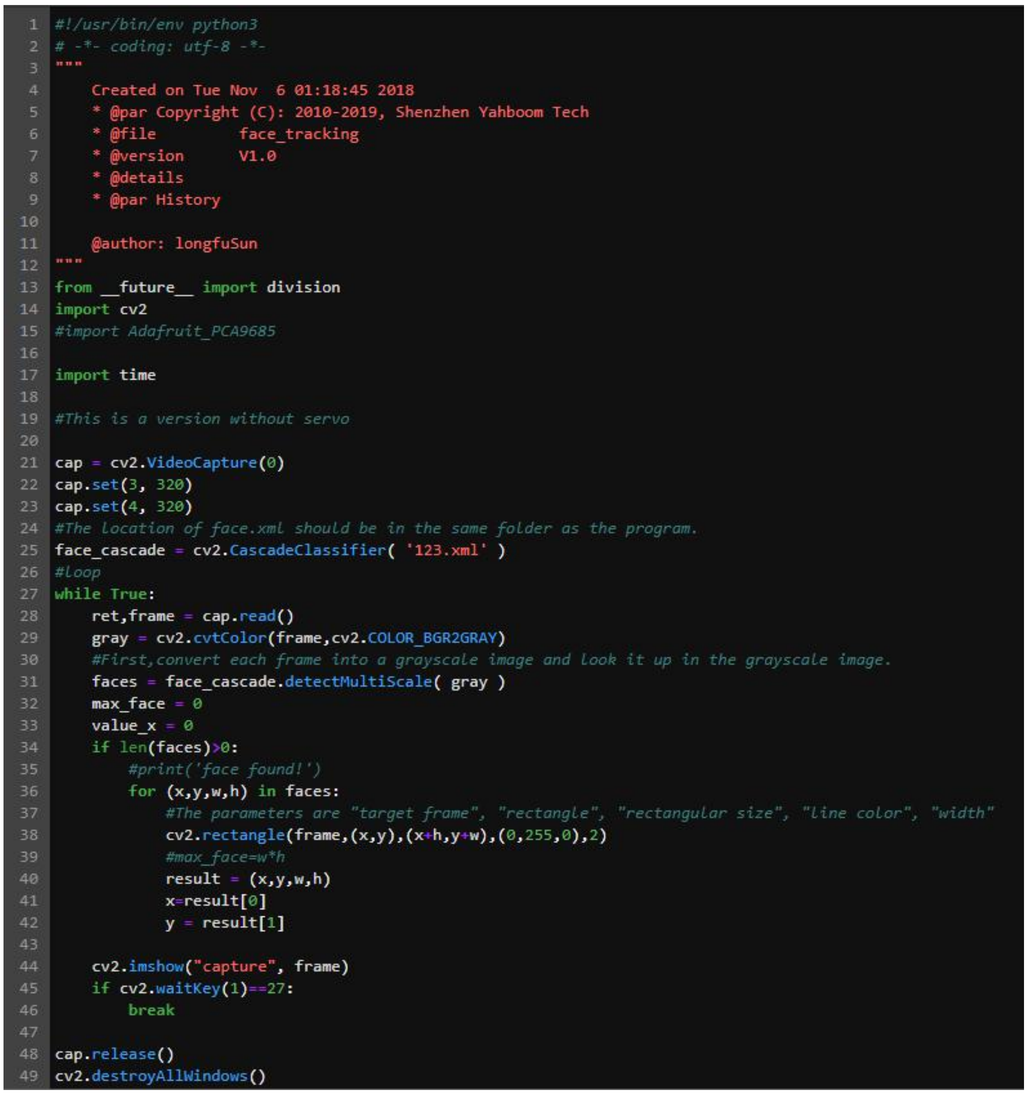
Chapter12:Raspberry Pi face recognition

！ Note: When running the program of this course, there must be a desktop for displaying pictures. It is recommended that you use VNC to log in to the system so that the pictures can be displayed.

Face detection requires a classifier:
face_cascade=cv2.CascadeClassifier(‘123.xml’)

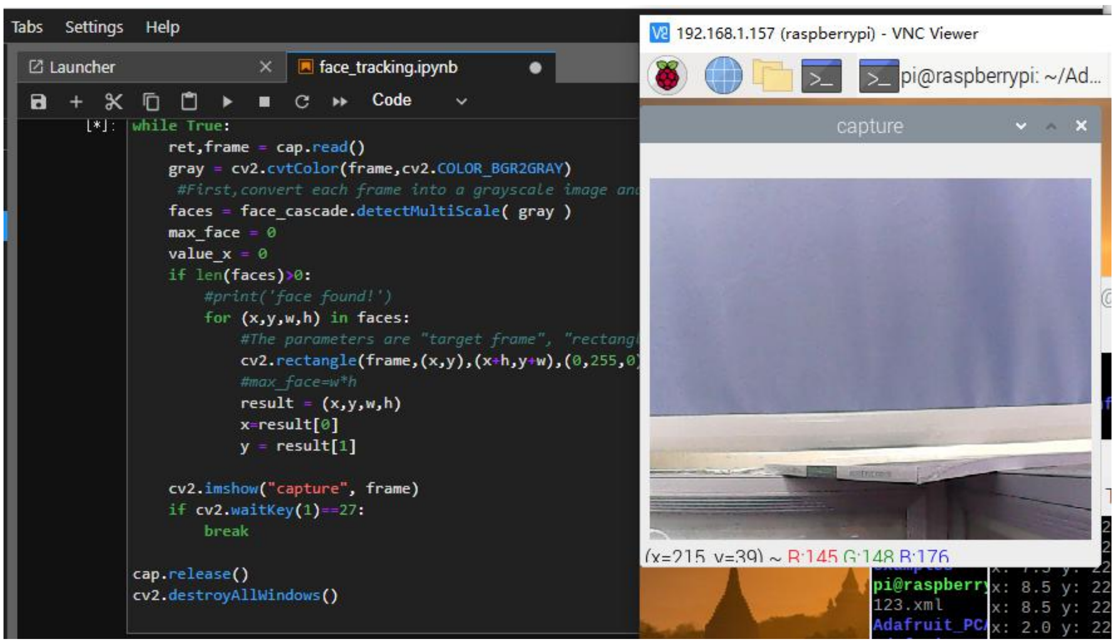
123.xml is Haar cascading data, this xml can be obtained from data/haarcascades in the OpenCV3 source code. The actual face detection is then performed by face_cascade.detectMultiScale(). We can't directly pass each frame of the image captured by the camera into .detectMultiScale(). We should convert the image to a grayscale image, because face detection also requires such a color space.
!!! Note: Be sure to enter the correct location of 123.xml correctly.

The source code of the program is located at:
/home/pi/yahboom/face_tracking/face_tracking.py

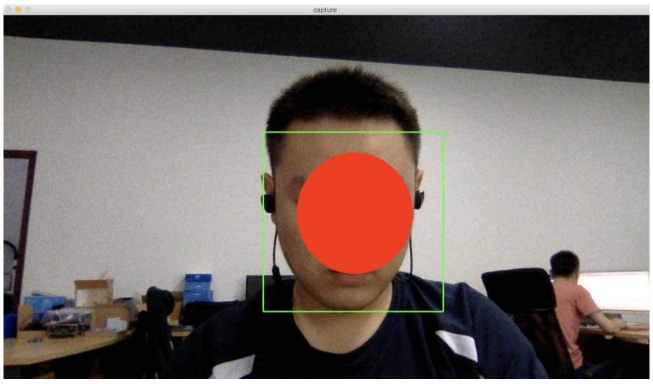


```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
    Created on Tue Nov  6 01:18:45 2018
    * @par Copyright (C): 2010-2019, Shenzhen Yahboom Tech
    * @file         face_tracking
    * @version      V1.0
    * @details
    * @par History

    @author: longfuSun
"""
from __future__ import division
import cv2
#import Adafruit_PCA9685

import time

#This is a version without servo

cap = cv2.VideoCapture(0)
cap.set(3, 320)
cap.set(4, 320)
#The location of face.xml should be in the same folder as the program.
face_cascade = cv2.CascadeClassifier( '123.xml' )
#loop
while True:
    ret,frame = cap.read()
    gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    #First,convert each frame into a grayscale image and look it up in the grayscale image.
    faces = face_cascade.detectMultiScale( gray )
    max_face = 0
    value_x = 0
    if len(faces)>0:
        #print('face found!')
        for (x,y,w,h) in faces:
            #The parameters are "target frame", "rectangle", "rectangular size", "line color", "width"
            cv2.rectangle(frame,(x,y),(x+h,y+w),(0,255,0),2)
            #max_face=w*h
            result = (x,y,w,h)
            x=result[0]
            y = result[1]

    cv2.imshow("capture", frame)
    if cv2.waitKey(1)==27:
        break

cap.release()
cv2.destroyAllWindows()
```

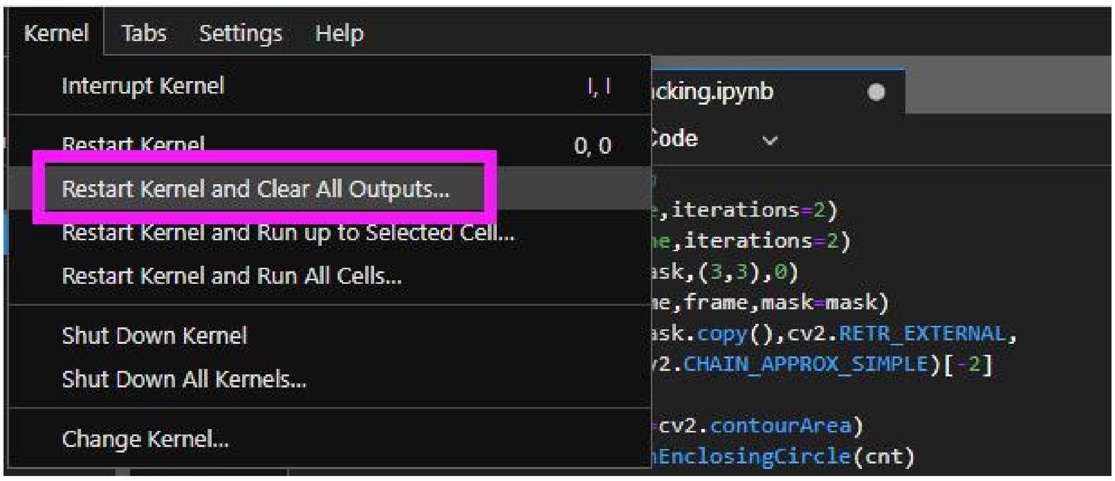
The result is as shown in the figure below. It will generate a rectangular around the face to framed the face.



```
[*]: while True:
         ret,frame = cap.read()
         gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
          #First,convert each frame into a grayscale image and
         faces = face_cascade.detectMultiScale( gray )
         max_face = 0
         value_x = 0
         if len(faces)>0:
             #print('face found!')
             for (x,y,w,h) in faces:
                 #The parameters are "target frame", "rectang
                 cv2.rectangle(frame,(x,y),(x+h,y+w),(0,255,0)
                 #max_face=w*h
                 result = (x,y,w,h)
                 x=result[0]
                 y = result[1]

         cv2.imshow("capture", frame)
         if cv2.waitKey(1)==27:
             break

     cap.release()
     cv2.destroyAllWindows()
```



Then, we can click [Kernel]-[Restart Kernel and Clear All Outputs] to end this process and clear the output results.



Kernel
Tabs
Settings
Help
Interrupt Kernel I, I
Restart Kernel 0, 0
Restart Kernel and Clear All Outputs...
Restart Kernel and Run up to Selected Cell...
Restart Kernel and Run All Cells...
Shut Down Kernel
Shut Down All Kernels...
Change Kernel...