In OpenCV, face detection in video only reads each frame of image from the camera, and then uses the static image detection method for detection. Face detection requires the classifier:
@ Face detector(default): haarcascade_frontalface_default.xml
@ Face detector(fast Harr): haarcascade_frontalface_alt2.xml
@ Face detector(Side view): haarcascade_profileface.xml
@ Eye detector(left eye): haarcascade_lefteye_2splits.xml
@ Eye detector(right eye): haarcascade_righteye_2splits.xml
@ Mouth detector: haarcascade_mcs_mouth.xml
@ Nose detector: haarcascade_mcs_nose.xml
@ Body detector: haarcascade_fullbody.xml
@ Face detector(fast LBP): lbpcascade_frontalface.xml
@ Only open eyes can be detected: haarcascade_eye.xml
@ Only person with glasses can be detected: haarcascade_eye_tree_eyeglasses.xml
@ https://github.com/opencv/opencv/tree/master/data Download classifier file link
haarcascade_profileface.xml is the cascading data of Haar. This xml can be obtained from this link
https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_profileface.xml
Next, we can start face detection by face_cascade.detectMultiScale (). We need to convert the image into a grayscale, then, transfer each frame of the image obtained by the camera into .detectMultiScale ().
(Note: we need to ensure enter the correct location of haarcascade_profileface.xml correctly.)

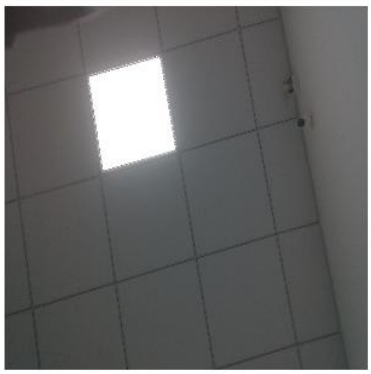
1. Program analysis

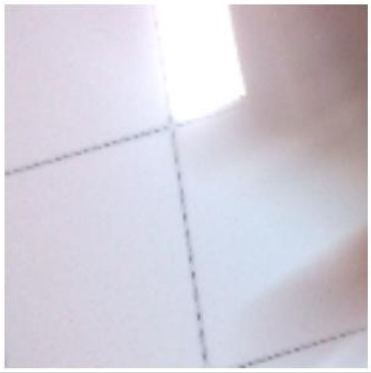
1.1 Import the dependent packages related to the camera and set the resolution.

```
from camera import Camera
from image import bgr8_to_jpeg
camera = Camera.instance(width=720, height=720)
```

1.2 Create two display spaces. The first is used for taking video. The second is used to display the recognized face parts.

[2]:
```
import traitlets
import ipywidgets.widgets as widgets
from IPython.display import display
face_image = widgets.Image(format='jpeg', width=300, height=300)
face = widgets.Image(format='jpeg', width=300, height=300)
display(face_image)
display(face)
```





1.3 Load the "Haar"" cascade classifier file "123.xml" for face detection.

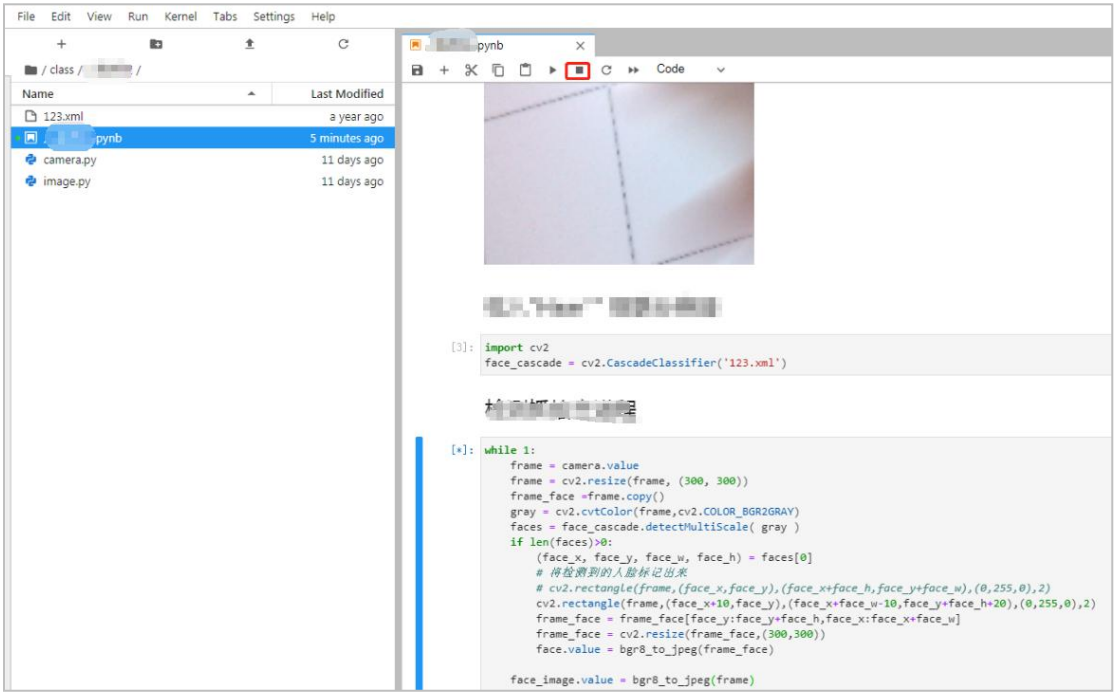
```
import cv2
face_cascade = cv2.CascadeClassifier('123.xml')
```

1.4 Obtain the video image, and detect the image to obtain the x, y coordinate value, width and height of the face. The face part is cut out, transmitted and transmitted to the space of the face part.
Finally refresh the picture to the camera screen (The second is used to display the recognized face parts.)

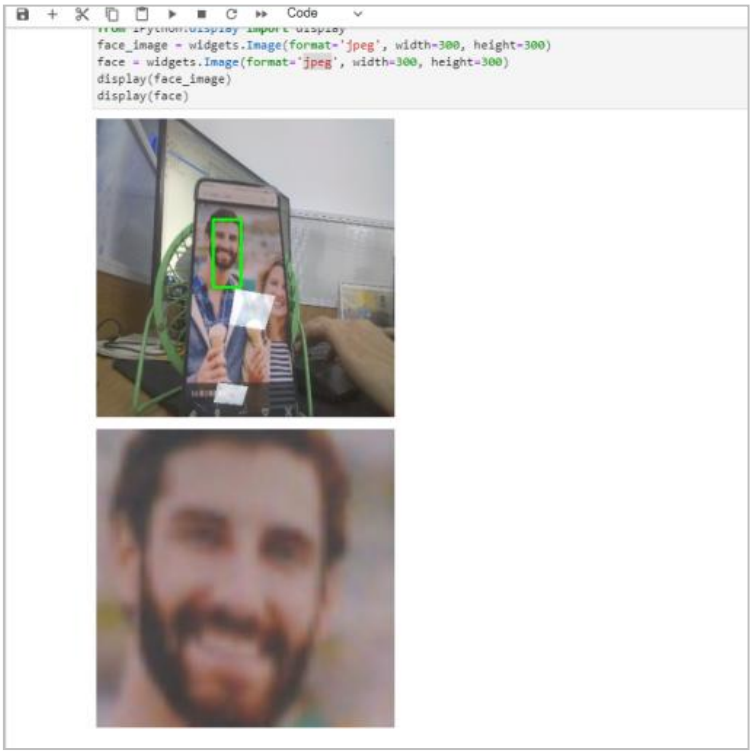
```
while 1:
    frame = camera.value
    frame = cv2.resize(frame, (300, 300))
    frame_face =frame.copy()
    gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale( gray )
    if len(faces)>0:
        (face_x, face_y, face_w, face_h) = faces[0]
        # 将检测到的人脸标记出来
        # cv2.rectangle(frame,(face_x,face_y),(face_x+face_h,face_y+face_w),(0,255,0),2)
        cv2.rectangle(frame,(face_x+10,face_y),(face_x+face_w-10,face_y+face_h+20),(0,255,0),2)
        frame_face = frame_face[face_y:face_y+face_h,face_x:face_x+face_w]
        frame_face = cv2.resize(frame_face,(300,300))
        face.value = bgr8_to_jpeg(frame_face)

    face_image.value = bgr8_to_jpeg(frame)
```

1.5 We can stop modifying the program through the stop button above.

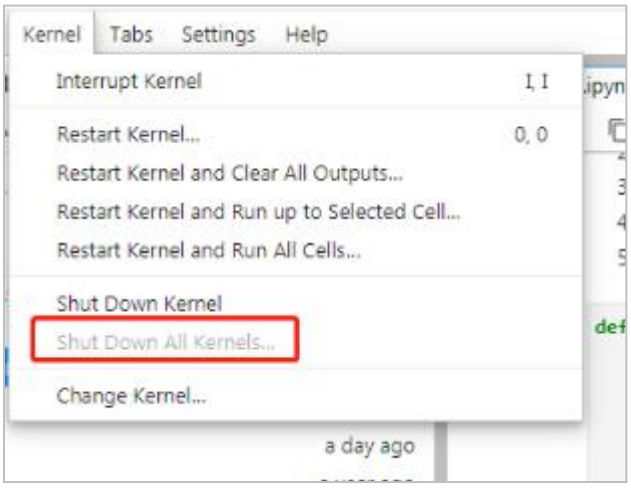


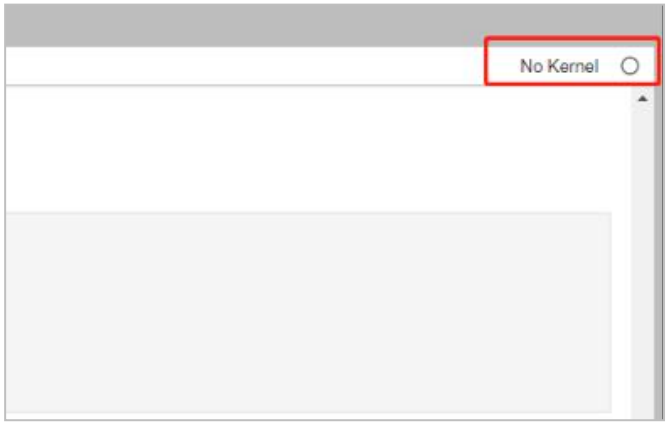
1.6 After the program runs, when the camera detects a human face, it will circle the human face with a green frame, and display part of the human face in the local control. As shown below.



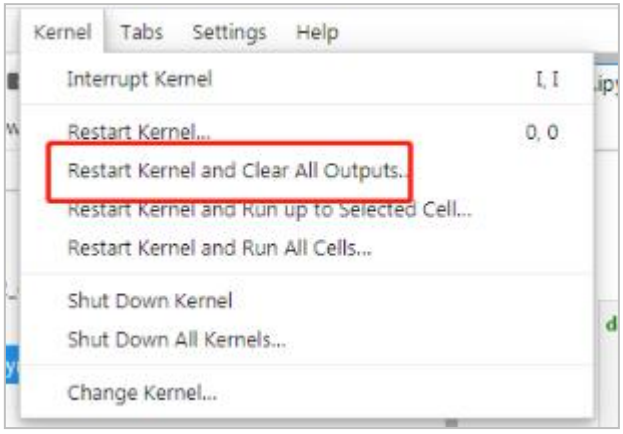
1.7 If you need to shut down this process completely, please do the following operation .

1) Click **[shut down all kernels]** and wait for **[no kernels]** on the upper right corner. After restarting the kernel and clear output, wait for the right side to become python3. If the camera is still occupied, it is recommended to restart





2) Click [restart kernel and clear output], and wait for [Python3] on the upper right corner.




Python 3