This Project is Created By Jyotirmay Chowdhury.

https://jyotirmaychowdhury.pages.dev/

This Python code uses the OpenCV library (cv2) to perform face detection in an image. Here's a simple explanation of what each part of the code does:

1. `import cv2`: This line imports the OpenCV library, which is used for computer vision tasks like image processing and face detection.

2. `face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')`: This line loads a pre-trained face detection classifier from a file named 'haarcascade_frontalface_default.xml'. This classifier is trained to recognize frontal faces in images.

3. `img = cv2.imread('test.png')`: This line reads an image named 'test.png' and stores it in the variable `img`.

4. `gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)`: This line converts the color image (`img`) to grayscale. Grayscale images are often used for face detection because they simplify the image and make it easier to detect features like faces.

5. `faces = face_cascade.detectMultiScale(gray, 1.1, 4)`: This line detects faces in the grayscale image using the pre-trained classifier (`face_cascade`). It stores the detected faces as rectangles in the `faces` variable. The `1.1` and `4` are scaling factors and minimum neighbors used for the detection process.

6. `for (x, y, w, h) in faces:`: This line iterates through the list of detected faces, where `(x, y)` represents the top-left corner of the face rectangle, and `(w, h)` represents its width and height.

7. `cv2.rectangle(img, (x, y), (x + w, y + h), (225, 0, 0), 2)`: For each detected face, this line draws a blue rectangle around it on the original color image (`img`). It uses the coordinates `(x, y)` for the top-left corner and `(x + w, y + h)` for the bottom-right corner of the rectangle. The `(225, 0, 0)` specifies the color (blue in BGR format), and `2` is the thickness of the rectangle's border.

8. `cv2.imshow('img', img)`: This line displays the original image with the detected faces in a window with the title 'img'.

9. `cv2.waitKey()`: This line waits indefinitely for a key press. It keeps the window open until you press any key.

10. `cv2.imwrite("face_detected.jpg")`: This line attempts to save the modified image with detected faces as 'face_detected.jpg'. However, there is a missing argument for the image data to be saved, so it won't work as intended. To fix it, you should provide the image data as the first argument, like this: `cv2.imwrite("face_detected.jpg", img)`.

In summary, this code uses OpenCV to detect faces in an image, draw rectangles around the detected faces, and display the modified image in a window. It also attempts to save the modified image, but the saving part needs to be fixed.