Understanding the code

Let's break down the actual code.

You first pass in the image and cascade names as command-line arguments.

Now we create the cascade and initialize it with our face cascade. This loads the face cascade into memory so it's ready for use.

Then we read the image and convert it to grayscale. Many operations in OpenCV are done in grayscale.

The function detects the actual face – and is the key part of the code.

1. The `detectMultiScale` function is a general function that detects objects. Since we are calling it on the face cascade, that's what it detects. The first option is the grayscale image.
2. The second is the `scaleFactor`. Since some faces may be closer to the camera, they would appear bigger than those faces in the back. The scale factor compensates for this.
3. The detection algorithm uses a moving window to detect objects. `minNeighbors` defines how many objects are detected near the current one before it declares the face found. `minSize`, meanwhile, gives the size of each window.

"I took commonly used values for these fields. In real life, you would experiment with different values for the window size, scale factor, etc., until you find one that best works for you."

The function returns a list of rectangles where it believes it found a face. Next, we loop over where it thinks it found something.

This function returns 4 values: the x and y location of the rectangle, and the rectangle's width and height (*w,h*).

We use these values to draw a rectangle using the built-in `rectangle()` function.

In the end, we display the image, and wait for the user to press a key.

To check the result.

```
$ python face_detect.py image.jpeg face.xml
```