Open-Source Report

[Flask]

General Information & Licensing

Code Repository	https://github.com/pallets/flask
License Type	BSD-3-Clause license
License Description	• Do whatever you want with it as long as the copyright in Flask sticks around, the conditions are not modified, and the disclaimer is present. • Furthermore, you must not use the names of the authors to promote derivatives of the software without written consent. • https://flask.palletsprojects.com/en/0.12.x/license/
License Restrictions	● You may not use the names of the original company or its members ● Describes the warranty and if the software/license owner can be charged for damages. ● https://tldrlegal.com/license/bsd-3-clause-license-(revised)

Magic ⋆★˚｡⋆୨୧˚ ˚☽˚｡⋆ ⁀➷｡˚★ 彡⋆ ༄

After the TCP socket is created, the server is ready to listen for any TCP connection in the assigned port. Once the client sends the request to the server, the three-way handshake will be used to establish TCP connection.

In the first step, the client sends a packet with random sequence number X and SYN. The client enters SYN_SENT status waiting for confirmation from the server.

In the second step, the server receives the packet from the client (X and syn). The server is going to respond with the packet that includes SYN, ACK, a random sequence number Y, and ACK number X + 1(X is the sequence number received from the client). The server enters SYN_RECV status waiting for the response from the client.

In the third step, the client sends the packet with ACK and Y+1(Y is the sequence number received from the server)

The TCP connection is established after the three-way handshake. Now the client can send get, post, or any other requests to the server.

Flask is using the Werkzeug library to handle TCP such as creating a TCP socket and listening on a certain port.

First, we need to call **app.run** with Flask to listen and create the socket. In our implementation, we use flask_socketio which requires starting the app with socketio.run, a wrapper function that ends up calling app.run.

After app.run gets called, it will enter the app.py file first.
For instance, this line will be called after app.run (line 1132)

This line is where app.py is importing Werkzeug.serving (line 1188)

After all the configuration is done, the Werkzeug serving.py is going to handle the socket and listening tasks.

Because the socket is created by .socket directly, therefore, it will use the socket class first.

Socket.py -> socket class (lines 215 – 662)
Class used to create the socket

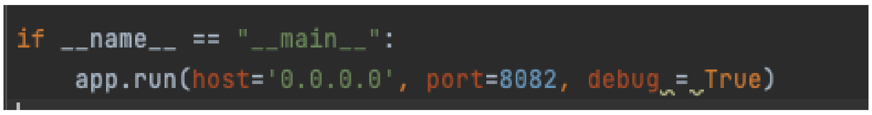
ForkingWSGIServer class (lines 799 – 1096)

We need to create a socket before listening on any port.
This line of code is used to create a socket. (line 895)

After socket is created, the server is ready to listen on a certain port.
This line indicates that we are listening for the request. (line 930)

Once the client sends a request, the server will perform the three-way handshake to establish the TCP connection. The three-way handshake is handled by the operating system. Once the three-way handshake is completed, the client can send requests.

In our project, we are calling the function app.run to make it works.



```
if __name__ == "__main__":
    app.run(host='0.0.0.0', port=8082, debug = True)
```