Flask

General Information & Licensing

Code Repository	https://github.com/pallets/flask
License Type	BSD-3
License Description	Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: ● Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. ● Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. ● Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.
License Restrictions	● Prohibits others from using the name of the copyright holder or its contributors to promote derived products without written consent.

- app.run(host="127.0.0.1", port=8080) #We have debug=true but it's not pertinent to the actual project
  - This run method is found in the Flask 'app.py' file and is located at https://github.com/pallets/flask/blob/cc66213e579d6b35d9951c21b685d0078f373c44/src/flask/app.py#L1067
  - This gets passed the host - in this case "127.0.0.1" and port 8080 not unlike the code in our homework: [socketserver.TCPServer(("0.0.0.0",8000),MyTCPHandler) as server]
  - First, it checks to see if the method is ran from the command line. If it is, it ignores the call so that it doesn't start another server. https://github.com/pallets/flask/blob/cc66213e579d6b35d9951c21b685d0078f373c44/src/flask/app.py#L1132
  - Next (dotenv). If we haven't it loads the necessary files. (Not relevant to our homework - we have no dotenv files), it checks whether the user has disabled a certain loading file type https://github.com/pallets/flask/blob/cc66213e579d6b35d9951c21b685d0078f373c44/src/flask/app.py#L1144
  - If the flash environment is instantiated in the operating system environment, it determines whether debug mode should be enabled for the flask application. Moreover, it checks if a deprecated "FLASK_ENV" is used, and if so throws an error, changes it to the new "FLASK_DEBUG" and checks whether the debug flag is on or off. https://github.com/pallets/flask/blob/cc66213e579d6b35d9951c21b685d0078f373c44/src/flask/app.py#L1148
  - Further, it tests to see whether or not debug was instantiated. If it was, it typecasts it to a boolean type. https://github.com/pallets/flask/blob/cc66213e579d6b35d9951c21b685d0078f373c44/src/flask/app.py#L1160
  - Next, it pulls the name of the server from the config file, and initializes the host and port to be none. https://github.com/pallets/flask/blob/cc66213e579d6b35d9951c21b685d0078f373c44/src/flask/app.py#L1163
  - If the server name was instantiated in the config file, it sets the host and port to the server name's components (separated by the colon). https://github.com/pallets/flask/blob/cc66213e579d6b35d9951c21b685d0078f373c44/src/flask/app.py#L1166
  - Proceeding, it sets the host to the default ("127.0.0.1") if none is given (otherwise it uses the given host). https://github.com/pallets/flask/blob/cc66213e579d6b35d9951c21b685d0078f373c44/src/flask/app.py#L1169
  - If the port was given, it typecasts the port, if the port was not given but is included in the server name, it typecasts the port (to int) in the server name. If no port is given in either, it sets the port to the default (5000). https://github.com/pallets/flask/blob/cc66213e579d6b35d9951c21b685d0078f373c44/src/flask/app.py#L1175
  - Consequently, flask makes changes to the default configurations of the server, setting "use_reloader" and "use_debugger" to whether or not debug mode is enabled, and sets "threaded" to true *so that the server can handle multiple clients.*

https://github.com/pallets/flask/blob/cc66213e579d6b35d9951c21b685d0078f373c44/src/flask/app.py#L1184

- If debug is enabled, it tells the client code to show extra startup messages the first time the server is run, unless the server is run using the reloader (which makes sense because that would be unnecessary using the reloader because it was already done). https://github.com/pallets/flask/blob/cc66213e579d6b35d9951c21b685d0078f373c44/src/flask/app.py#L1186
- The library then imports “run_simple” method from ‘werkzeug” https://github.com/pallets/flask/blob/cc66213e579d6b35d9951c21b685d0078f373c44/src/flask/app.py#L1188
    - This code is what starts the TCP server. It takes in the hostname (localhost for most of our purposes), the port number, defines the application as a WSGI, defines whether or not the reloader is being used (it isn’t, it’s used for an initial start up), makes the debugger interactive through a python terminal, notes the files to be watched & updated for changes, and how it handles multithreading. https://github.com/pallets/werkzeug/blob/3115aa6a6276939f5fd6efa46282e0256ff21f1a/src/werkzeug/serving.py#L907
    - It checks the validity of the port and throws a tantrum (exception) if the port doesn’t check out.
    - This also imports SharedDataMiddleware if serving static files (which we aren’t so I won’t go into this part) - we might be when actually serving the HTML but that isn’t exactly part of *hosting* the TCP server. But it is relevant for *creating* the actual packets sent to the user - which I don’t *THINK* we have to discuss? **IF WE DO PLEASE LET ME KNOW, TAs OR JESSE** https://github.com/pallets/werkzeug/blob/3115aa6a6276939f5fd6efa46282e0256ff21f1a/src/werkzeug/middleware/shared_data.py#L38
    - Next, it imports DebuggedApplication which launches an application that enables debugging support. It allows evaluating expressions in any frame of a traceback, meaning it uses a python terminal here if needed for any frame. A PIN is required unless it is a publicly visible server. https://github.com/pallets/werkzeug/blob/3115aa6a6276939f5fd6efa46282e0256ff21f1a/src/werkzeug/debug/__init__.py#L223
    - If the server is not going to run from the reloader, it prepares the socket dedicated for the hostname and port for use by the server and reloader. The socket is marked as inheritable so it can be preserved after reloads instead of simply breaking off the TCP connection. Next, it sets the os environment to the file number of the prepared socket.
    - If the server runs from the reloader, it sets the OS environment to the inherited socket that was preserved for use after reloads. https://github.com/pallets/werkzeug/blob/3115aa6a6276939f5fd6efa46282e0256ff21f1a/src/werkzeug/serving.py#L1035
    - It then uses the make_server method (which seems to involve the bulk of this TCP server/connections)

https://github.com/pallets/werkzeug/blob/3115aa6a6276939f5fd6efa46282e0256ff21f1a/src/werkzeug/serving.py#L1037
Make_server also deals with threading/processes/forking etc, but not pertinent to the TCP connections, technically.

- This make_server method uses BaseWSGIServer as well as WSGIRequestHandler (THIS will handle the TCP Connections to the server, yay!)
  - This is where each connection to the TCP server (WSGI) is handled.
  - https://github.com/pallets/werkzeug/blob/3115aa6a6276939f5fd6efa46282e0256ff21f1a/src/werkzeug/serving.py#L148
  - First, it checks the server version https://github.com/pallets/werkzeug/blob/3115aa6a6276939f5fd6efa46282e0256ff21f1a/src/werkzeug/serving.py#L154
  - Then it makes the environment. https://github.com/pallets/werkzeug/blob/3115aa6a6276939f5fd6efa46282e0256ff21f1a/src/werkzeug/serving.py#L159
    - This includes the request_url (the path the user is going to), The url scheme (http/https), Client's address (their IP), Path info (the request_url if same domain - I think, otherwise url and path. https://github.com/pallets/werkzeug/blob/3115aa6a6276939f5fd6efa46282e0256ff21f1a/src/werkzeug/serving.py#L160
    - The remainder of the environment variables are set by the fields of the WSGIRequestHandler https://github.com/pallets/werkzeug/blob/3115aa6a6276939f5fd6efa46282e0256ff21f1a/src/werkzeug/serving.py#L178
    - THEN, it iterates through each of the headers (dictionary) which parses the headers, replaces the CRLF with empty strings in the header data, and replaces the "-" in the actual headers with "_" (e.g. Content-Length would become Content_Length). https://github.com/pallets/werkzeug/blob/3115aa6a6276939f5fd6efa46282e0256ff21f1a/src/werkzeug/serving.py#L203
      - These headers also become environment variables in the WSGI environment.
      - If the data is encoded as chunked, it dechunks it (kind of irrelevant it seems).
      - It also checks certifications, unsure what that's for as they aren't referenced.

        - Lastly, it calls “execute”, that is considering there’s no errors https://github.com/pallets/werkzeug/blob/3115aa6a6276939f5fd6efa46282e0256ff21f1a/src/werkzeug/serving.py#L319
            - “Execute” immediately passes “start_response” to the app https://github.com/pallets/werkzeug/blob/3115aa6a6276939f5fd6efa46282e0256ff21f1a/src/werkzeug/serving.py#L305 Which goes through setting headers.

    - Next, if the server is not running from the reloader, it launches a command prompt to show information about the address when starting the server, (warning the developer that it is a development server not for use in production, for instance). https://github.com/pallets/werkzeug/blob/3115aa6a6276939f5fd6efa46282e0256ff21f1a/src/werkzeug/serving.py#L1051
    - If the server is being launched with the reloader, it imports “run_with_reloader”, which sets certain configurations of the server that will run forever (excluding certain facets because it isn’t on its initial startup). https://github.com/pallets/werkzeug/blob/3115aa6a6276939f5fd6efa46282e0256ff21f1a/src/werkzeug/serving.py#L1055
    - Finally, it establishes a connection with a developer that runs forever until it is shut down by the dev via KeyboardInterrupt exception. https://github.com/pallets/werkzeug/blob/3115aa6a6276939f5fd6efa46282e0256ff21f1a/src/werkzeug/serving.py#L1058

- Finally, it attempts to run the server “simply” (and indefinitely) through another library that runs servers used during development only, or in a Web Server Gateway Interface. It is meant to be convenient but not stable, secure or efficient in a “simple” server run. Upon the development server resetting normally, it resets the first request information if the development server in order to restart the server without the reloader. https://github.com/pallets/werkzeug/blob/3115aa6a6276939f5fd6efa46282e0256ff21f1a/src/werkzeug/serving.py#L1068