HTTP Client and Server

This repository is a relatively simple implementation of an HTTP client and server program using the Python builtin `socket` module.

Contents

myclient/

- main.py - driver for the HTTP client
- static/ - files ready to upload to an HTTP server via a PUT request

myserver/

- myserver/main.py - driver for the HTTP server
- myserver/static/ - "database" of available files on the server

tests/

- test_client.py - tests for the http client
- test_server.py - tests for the http server

writeups/

- design.* - submission discussion design, "how it works," tradeoffs, and future improvements
- design.* - submission describing test cases

Getting started

Once you have cloned the repository, install the required modules: `pip install -r requirements.txt`. Then, install the server and client modules so that they can be run in the command line `pip install -e .`.

Start the HTTP server

To run the HTTP server, enter `myserver <port>` in the command line, where `<port>` is the port number on which you desire to run the server. For example `myserver 5678`. Once the server has started, you will see a log message stating that the server is running. The message will also indicate the hostname and port number of the server.

The server is designed to handle GET and PUT requests. When receiving a GET request, the server will search `myserver/static` for the requested file. If found, the server will respond with a 200 code and the content of the file. Otherwise, the server will respond with 404 Not Found. When receiving a PUT request, the server will create the file in `myserver/static/` if it doesn't already exist or update the contents of the file.

Send requests with the HTTP client

In a similar way, the HTTP client is exposed as a command line tool. To make a request with the client, enter `myclient <host> <port> <method> <filename>`. For example `myclient localhost 5678 index.html` or `myclient www.cnn.com 80 GET index.html`.

If using the client in tandem with the HTTP server in this repository files contained within the `myserver/static` directory can be requested with GET requests.

When executing a POST request, the client will search the `myclient/static` directory for the filename given. If that file is found, the client will send the POST request and file data out to the server. Otherwise, the client will display a friendly error message asking you to check your spelling.

Run the tests

First, start the server on port 5678 if you haven't already: `myserver 5678`.

Run the client tests: `python3 tests/test_client.py`

Run the server tests: `python3 tests/test_server.py`