Design

1 Decisions

1.1 Language Selection

Python was chosen because of its succinctness, and its ability to hash values of varying data type.

1.2 Request Handling

Heavy use of regular expressions were used, despite the number of problems that can arise from using them. It allowed the flexibility of parsing all HTTP GET requests into hash maps, which were then easily accessible whenever needed.

1.3 Threading

I chose to use ThreadedTCPServer objects over implementing threading myself because there is a lot of inherent control built in that prevents read/write access violations, and enforces locks behind the scenes.

1.4 Caching

Caching is done relative to the directory of pyproxy.py, and the extra credit functionality is implemented. It scans for a max-age parameter, and then acts according to that.

Note: For testing and evaluation, the cache is cleared before every run of pyproxy.py, although feasibly it is persistent across runs.