Super HTTP - Multi Request

The HTTP protocol allows multiple requests in one connection, so we will support that too!
This will require some advanced buffering techniques, that are essential to network applications.

Instructions

1. Continue with the same HTTP server `superhttp.py`.
2. Use netcat to send a basic HTTP request (`nc --crlf 127.0.0.1 8005`):

```
GET /abcd HTTP/1.1

HTTP/1.1 404 Not Found
Content-Type: text/html
Content-Length: 22

<h1>404 Not Found</h1>
```

3. Right after the result, in the same netcat window, try to type another request (`GET /abcd HTTP/1.1` with two newlines). There should be no answer, since the server shuts down the socket after completing the request.
4. Change the server code to support multiple requests in the same connection. Once one request is responded, the server will continue waiting for another request on the same connection.

5. To make sure it's working, use netcat to send multiple requests in the same connection. There are two tests you must do: 5.1. Connecting with netcat and writing the requests manually 5.2. Preparing a text file `queries.txt` with two requests, then sending them at once with `type queries.txt | nc 127.0.0.1 8005`. The file `queries.txt` can look like this:

```
GET /abcd HTTP/1.1

GET /abcd HTTP/1.1

```

Hint

In the previous exercise, the data was buffered until a single request is completed, and the entire buffer was considered one request.
In this exercise, the situation is different: The buffer may contain at any moment any number of requests, and the last one can be partial. Your code will need to be able to identify that the buffer contains a completed request, "pop" that one (meaning, remove it from the buffer), handle the "popped" request, and keep the rest of the buffer for later processing.

Since the buffer variable persists across multiple function calls, it is important to make sure that it is cleared every time a new client connection is made.

To submit

Submit file `superhttp.py`.