PA1: Intro to Socket Programming – HTTP Server and Client

Part 1: HTTP Server

For this assignment you will learn the basics of Python TCP socket programming: how to create a socket, how to bind to a specific address and port, and how to send and receive HTTP messages.

You will develop a very simple web server that handles one HTTP GET request. Your web server should accept and parse the HTTP request, retrieve the requested file from the server's file system, and send the file to the client in a correctly formatted HTTP response (i.e., "200 OK"). If the requested file is not present on the server, the server should send the HTTP "404 Not Found" message back to the client along with a simple HTML formatted string so that the browser can display a proper "file not found" message to the user.

Your HTTP web server must be interoperable with a real web browser. However, it does not need to handle any of the header lines beyond the first line in a GET request. Your server also does not need to include any header lines beyond the first in the response back to the browser. In short, your server does not need to gracefully handle any scenarios beyond what is outlined in the preceding paragraph.

Code

The template program for the web server is posted on Canvas in the file PA1Server.py. It will not run as-is because there are comments in place of functional pieces of code. The code you need to fill-in is marked with **`# Fill in start`** and **`# Fill in end`**. Each place may require one or more lines of code.

Running the Server

Create a very simple HTML file called helloworld.html that displays the message "Hello World!" Place this file in the same directory as the server and run the server program. From the same host computer, open a browser and input the appropriate URL to fetch the helloworld.html file:
http://127.0.0.1:6789/HelloWorld.html
The browser should then display the contents of helloworld.html.

In the URL, note the port number argument (the number following the colon after the IP address in the URL). You need to replace this port number with the appropriate port number for your server. If you omit the colon and port number from the URL, the browser will assume port 80 because it is the default. In this case, unless you are running a web server that is listening at port 80, your browser, after trying to reach the server, will eventually display a "connection timed out" message.

Part 2: HTTP Client

Write your own HTTP client to both test your own server and to reach real webservers. Your client will connect to the server using a TCP connection, send an HTTP GET request to the server, and output the raw server response to the console. The client should take command line arguments specifying the server IP address or host name, the port at which the server is listening, and the path of the requested object. For example:

```
C:\> python PA1Client.py 127.0.0.1 6789 helloworld.html

C:\> python PA1Client.py www.example.com 80 index.html
```

What to Hand In:

For this assignment you will hand in the following screenshots to demonstrate that you have completed the assignment. On Windows machines, the built-in Snipping Tool is a convenient way to capture screenshots.

1. A browser with the address bar clearly in view retrieving the helloworld.html file from the server alongside the console output of the server.

2. A browser with the address bar clearly in view showing the "file not found" message alongside the console output of the server.

3. The console output of the client retrieving the helloworld.html file from the server alongside the console output of the server.

4. The console output of the client trying to retrieve a non-existent file from the server alongside the console output of the server (as in screenshot 2).

5. The console output of the client retrieving a webpage from a real webserver of your choice.