CS 280
Assignment 1
Due: Thursday, May 6

1. [20 points] In this assignment you will create a minimal web server to serve text, html, and graphics files. It will need to read and send file contents to clients as well as accept incoming data.

   What is this HTTP stuff, anyway?

   HTTP is a client-server request-response protocol. All communication happens over a TCP/IP connection. The server listens on a TCP port, and accepts new connections. The client connects to the server and sends an HTTP request. The server responds with an HTTP response. For this assignment, your webserver should service a single HTTP request per connection (the full HTTP protocol supports multiple requests per connection). HTTP requests and responses are in plain, ASCII text. Here is an example of an HTTP request that a server may receive (line numbers do not appear in the actual transmission):

   ```
   1. GET / HTTP/1.1
   2. User-Agent: curl/7.18.0     (i486-pc-linux)
   3. Host: google.com
   4. Accept: */*
   5.
   ```

   Line 1 of this request contains the HTTP method (GET), the request URI (/), and the HTTP version (HTTP/1.1). For this project, your web-server can ignore all other request fields. Note that the request is terminated by a single blank line (line 5). This is used to indicate to the server that the client has finished sending the HTTP header.

   The server's response to this request might look like the following:

   ```
   1. HTTP/1.1 200 OK
   2. Date: Mon, 03 May 2021 17:17:45 GMT
   3. Server: Apache/2.4.41 (Ubuntu)
   4. Last-Modified: Mon, 28 Dec 2020 00:59:13 GMT
   5. ETag: "2aa6-5b77bc85365ee"
   6. Accept-Ranges: bytes
   7. Content-Length: 10918
   8. Vary: Accept-Encoding
   9. Content-Type: text/html
   10.
   11. … HTTP CONTENT…
   ```

   Line 1 of this response contains the HTTP version (HTTP/1.1), the status code (200), and the reason phrase (OK). This is then followed by some number of response headers. For this

assignment, your web server does not need to generate any other response header fields. Note that like the HTTP request, the response is also terminated with a blank line (line 8). The lines immediately following the blank line (line 11+) contain HTTP content.
Your server can assume that the client sends a single HTTP request, and can close the TCP connection after sending the HTTP response to the client. As you can see on line nine, part of this header information tells the client what the 'type' of resource is that's being sent. Possible values here (initially) include:

- text/html
- text/plain
- image/gif
- image/jpeg
- image/png

You can detect file types according to the location of a file or by its extension (the bit of the file name after the right-most '.'). The `os.path` library in Python has some handy functions for parsing the file names. Look in the documentation for `os.path.split()`, `os.path.splitext()`, `os.path.isfile()` and `os.path.isdir()`.

Project Requirements

Your web server should work with Firefox (or Chrome, Safari, etc.). So, if you are trying to access the file F.html on your webserver, Firefox with a URL of http://localhost:*portnumber*/F.html should display the contents of the file.

Your web server only needs to generate three status codes and respective reasons:

- 200 OK
- 400 Bad Request
- 404 Not Found
- 501 Feature Not Implemented

Your web server should (1) read the request line: request method, request URI, HTTP version, (2) consume header input until a blank line, (3) respond with a Bad Request error message if an invalid/incomplete request is received, (4) respond with the content of the requested file using a valid HTTP response message, (5) return a status code of 200, 400, 404 or 501, (6) and do this *robustly*, meaning the server should not crash! You should be able to deal with any request that is sent. When you send back an error code, you should also send an HTML-formatted error message explaining (at least in general terms) what went wrong.

Your web server should serve files from its current directory. That is, when you run the web server program, the program's current directory will be some directory (e.g., a subdirectory of your home directory). HTTP GET requests for files in only that directory or lower should be satisfied. In other words, if you get a request like "GET /hello.html HTTP/1.1" you should return the file "hello.html" in the current working directory. ***As a security measure, you should strip ".." from your URI!*** Since the web server is running as your username on your local machine, it will have access to *all* the files you have access to!