**Project 1 Report**

**Project: HTTP Client and Server.**

**Team Members:**

**Pratik Parekh - 801076521.**

**Sakshat Surve - 801080533.**

**HTTP 1.1 Protocol:**

The Hypertext Transfer Protocol (HTTP) is a stateless application-level protocol for distributed, collaborative, hypertext information systems. It uses extensible semantics and self-descriptive message payloads for flexible interaction with network-based hypertext information systems.

HTTP is a stateless request/response protocol that operates by exchanging messages across a reliable transport- or session-layer "connection". An HTTP "client" is a program that establishes a connection to a server for the purpose of sending one or more HTTP requests. An HTTP "server" is a program that accepts connections in order to service HTTP requests by sending HTTP responses.

A client sends an HTTP request to a server in the form of a request message, beginning with a request-line that includes a method, URI, and protocol version, followed by header fields containing request modifiers, client information, and representation metadata an empty line to indicate the end of the header section, and finally a message body containing the payload body.

A server responds to a client's request by sending one or more HTTP response messages, each beginning with a status line that includes the protocol version, a success or error code, and textual reason phrase, possibly followed by header fields containing server information, resource metadata, and representation metadata, an empty line to indicate the end of the header section, and finally a message body containing the payload body.

HTTP 1.1 provides faster delivery of Web pages than the original HTTP and reduces Web traffic.

* Instead of opening and closing a connection for each application request, HTTP 1.1 provides a *persistent connection* that allows multiple requests to be batched or *pipelined* to an output [buffer](https://whatis.techtarget.com/definition/buffer) . Because the number of connection and disconnection requests for a sequence of "get a file" requests is reduced, fewer packets need to flow across the Internet. Since requests are pipelined, TCP segments are more efficient. The overall result is less Internet traffic and faster performance for the user.
* When a browser supporting HTTP 1.1 indicates it can decompress HTML files, a server will compress them for transport across the Internet, providing a substantial aggregate savings in the amount of data that has to be transmitted.

**Multithreading:**

The server will be able to handle multiple simultaneous service requests in parallel. This means that the Web server is multi-threaded. A threaded web server is one that handles each request with a new thread, as opposed to handling each request with a new process.Multi-threading is more complex than concurrency with processes, so there's an increased risk of the web server crashing if one of its modules isn't working correctly. Also, if it does crash, it's more likely to crash all of the threads (ie. all of the requests being handled at the time). However, multi-threaded servers consume less memory and are generally faster.

**GET and PUT Methods using HTTP:**

The format for a GET Request is as follows:

GET /images/logo.png HTTP/1.1

This GET request asks for a file that is present on the specified server and then downloads it. It only retrieves data. In our project, we are requesting an HTML and a pdf file from the server.

The format for a PUT Request is as follows:

PUT /images/logo.png HTTP/1.1

This PUT request replaces all current versions of the file that are present in the server. If there are no files present then it creates the specified file.

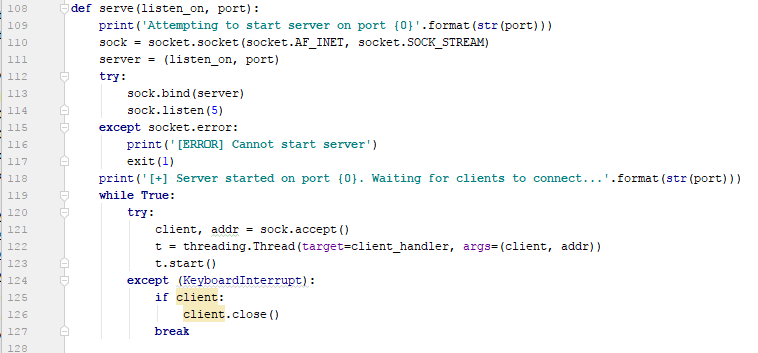
**Explanation of our Project:**

Our project consists of two folders and a ReadMe file which consists of all the instructions for executing the code. There are two executable files, one present in each folder, client.py, and server.py. The server.py consists of four main functions.

**server.py**

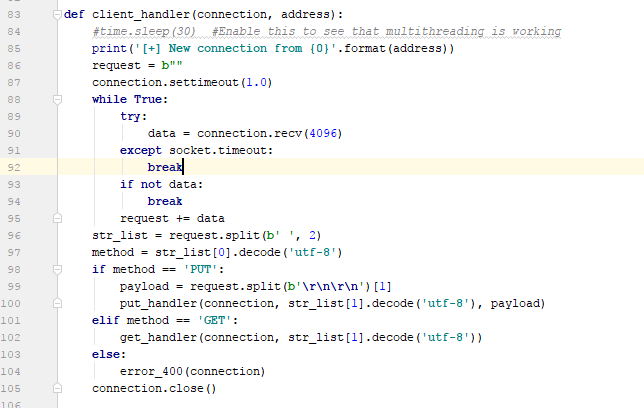
serve:

This is the main function that takes the IP Address and Port Number as the parameters and creates a socket. For every client that tries to connect to this server, it assigns a separate Thread to it. In our project, we are taking port numbers particularly greater than 5000 as these port numbers are not well-known and free to use.



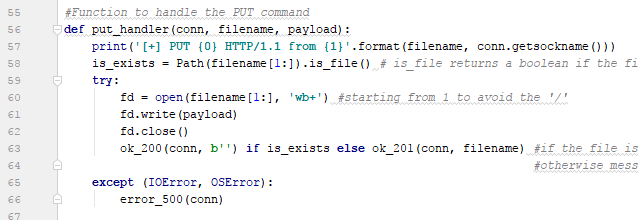
client\_handler:

This function is associated with every client that tries to connect to the server. After performing the split on the commandline arguments we come to know the method that the client wants to use. Accordingly, the function for the respective command is called.



put\_handler:

This functions first checks if the file is present locally or no if it is present then the client is allowed to *put* it on the server if not then a 404 Error is raised.



get\_handler:

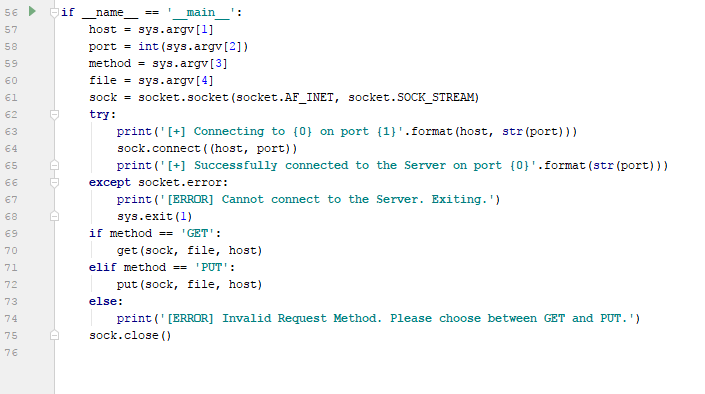
This function reads the data from the server and stores it locally in another file at the client side.



**client.py**

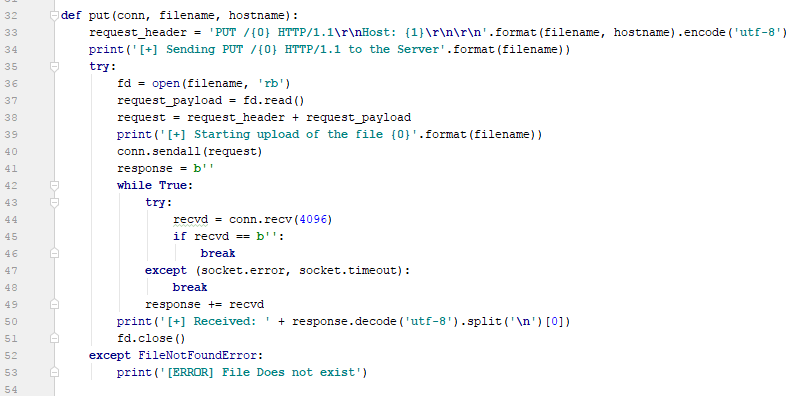
The client includes 2 main functions viz. put and get. The main function accepts 4 arguments from the commandline:- IP address, port number, method, file name.

If the server can’t connect then it throws an error and terminates the connection. If the connection is established then based on the method in the command the respective function is called.



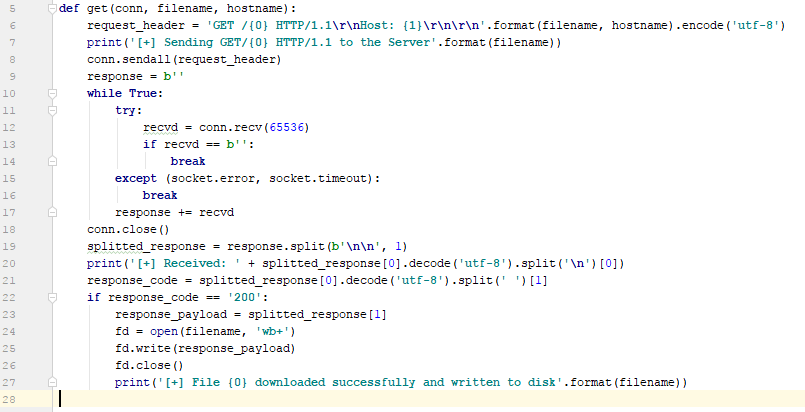
put:

This function uploads the file on the server if it isn’t present, otherwise overwrites the already present file.



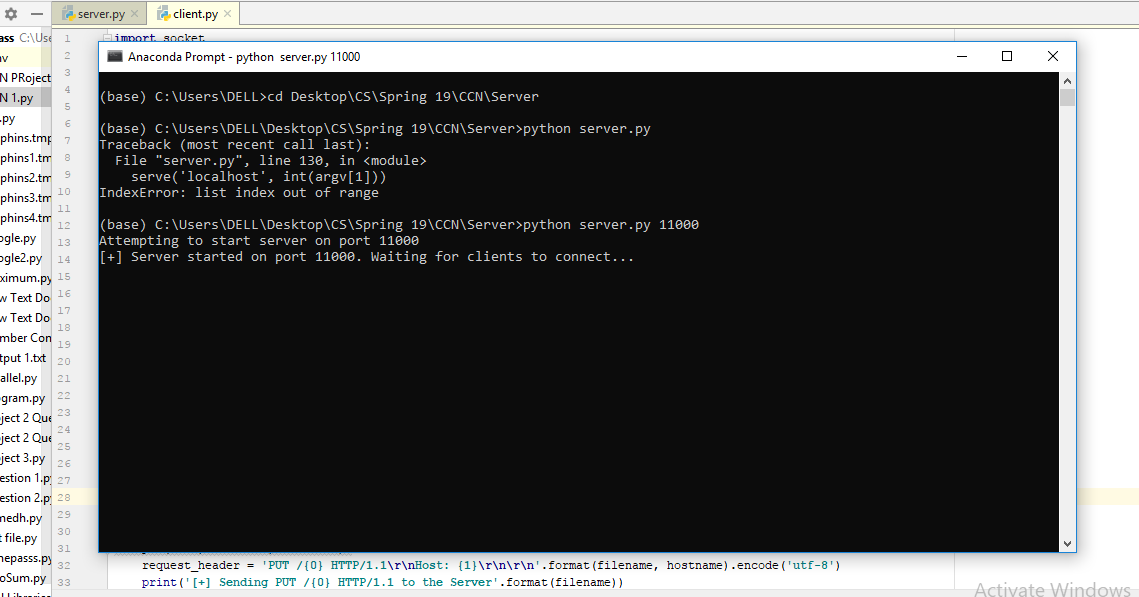
get:

This function downloads the desired file from the server and saves it locally on the client side.

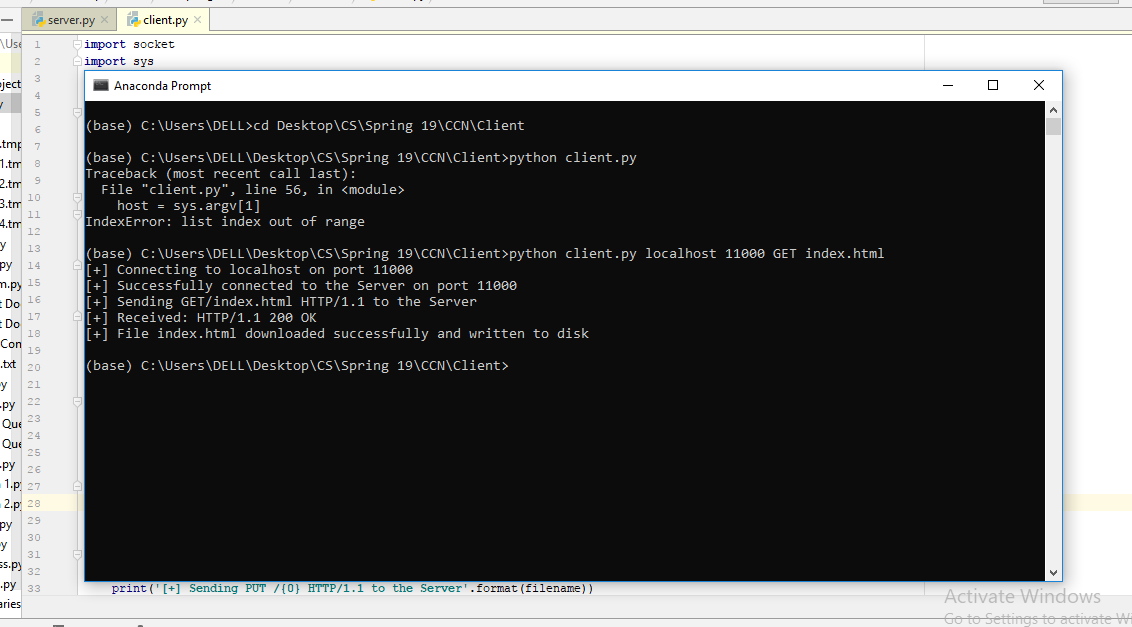


**Implementation:**

Running the server first.

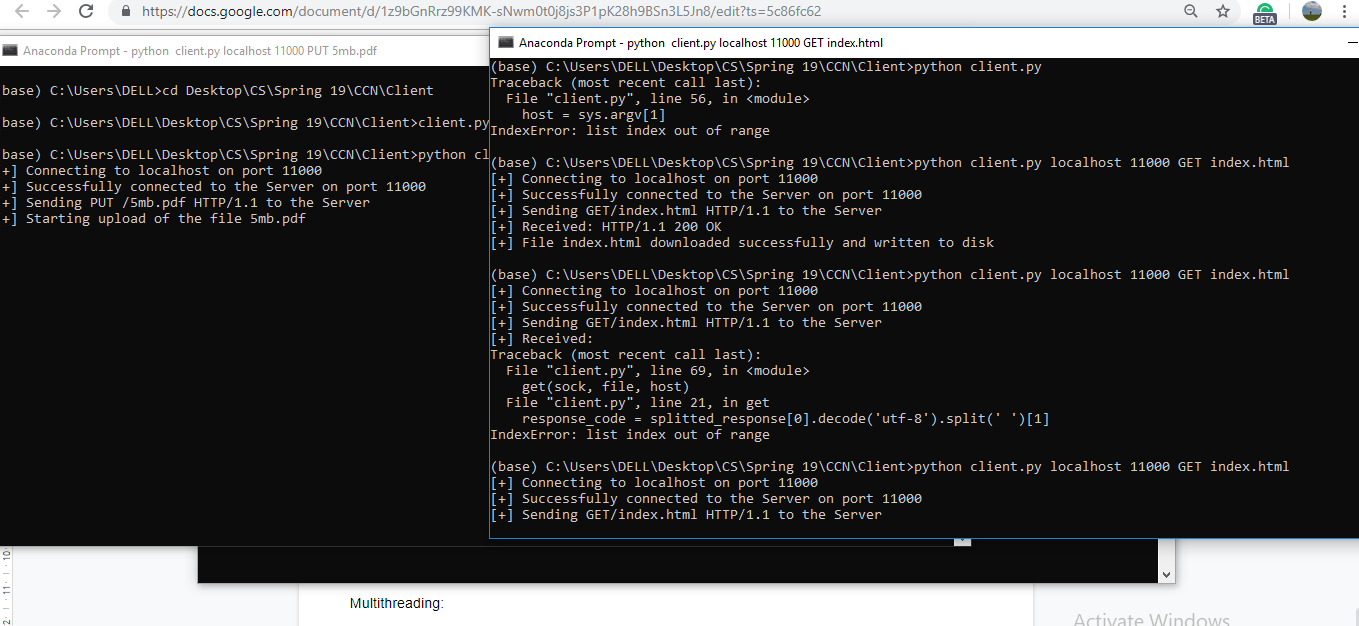


Then requesting from the server:



Multithreading:

As you can see two different clients are trying to access the same server at the same time.



**References :**

1. Details about HTTP 1.1. Retrieved from <https://httpwg.org/specs/rfc7230.html>
2. Details about HTTP 1.1. Retrieved from

<https://searchmicroservices.techtarget.com/definition/HTTP-11>

1. Multithreading. Retrieved from

<https://www.utc.edu/center-academic-excellence-cyber-defense/pdfs/multi.threaded.web.sever.pdf>

4. GET and PUT. Retrieved from <https://stackoverflow.com/>