**Single Threaded Web Server Test Procedure**

The single threaded web server is running on port 8000. Postman is used to send different requests for 200, 304, 400, 404 response code tests. A Python client script is used for the 408 test.



Figure : Terminal of the running server on port 8000

* 200 OK

Graphical user interface, text, application, email

Description automatically generated

Figure : Postman 200 OK

* 304 Not Modified
* If-Modified-Since is added to its header
* Last modified date of test.html is Nov 22 and requested date is Nov 25. Therefore, it returns 304 Not Modified

A picture containing calendar

Description automatically generated

Figure : Terminal showing the last modified date of test.html

Graphical user interface, text, application, email

Description automatically generated

Figure : Postman 304 Not Modified

* 400 Bad Request
* The server checks if the file exists and if true, it opens and reads the file. For the purpose of 400 Bad Request, I renamed the file name to `return400error.html` after it passed the 404 check.
* Since test.html does not exist, opening and reading the file will raise an error.
* `reply\_to\_client()` will catch the error and return a 400 error to the client.

Text

Description automatically generated

Figure : Python code changing file name after the 404 check

Graphical user interface, text, application, email

Description automatically generated

Figure : Postman 400 Bad Request

* 404 File Not Found
* Sent a request with non-existent file name.

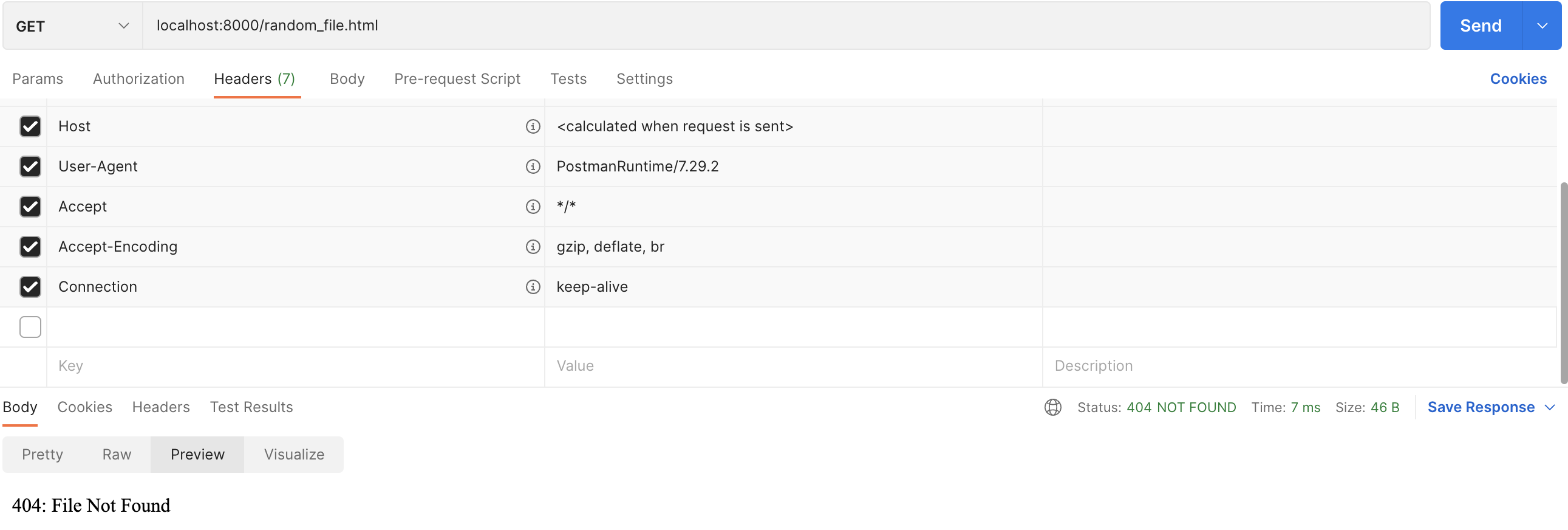


Figure : Postman 404 File Not Found

* 408 Request Timed Out
* A Python client script connects to the server and waits for 6 seconds to send data. The cut-off for the 408 error is 5 seconds; thus, the server returns the 408 error.

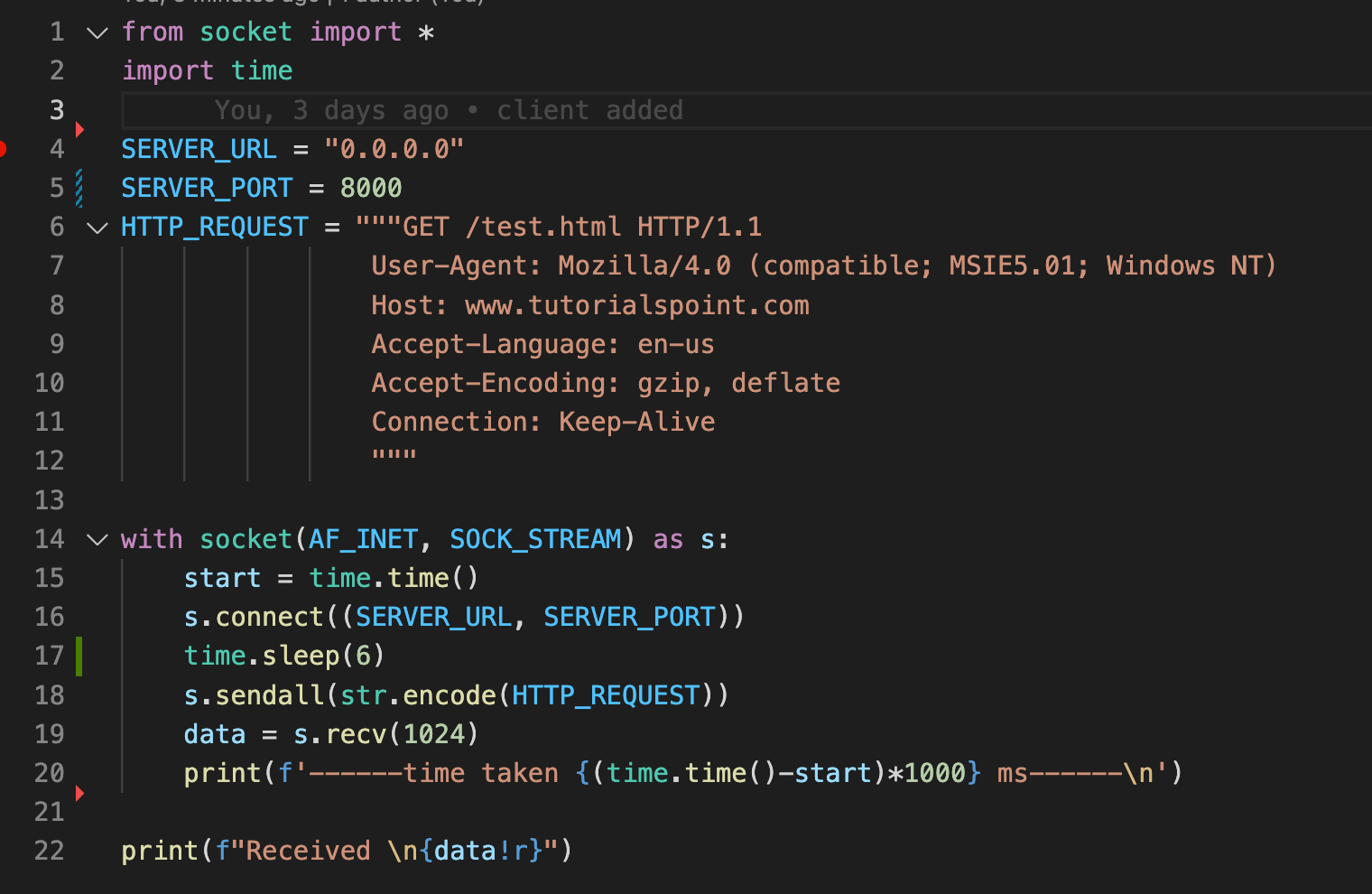


Figure : Python Client script with a delay

Text

Description automatically generated

Figure : Server return for 408 Request Timed Out

**Proxy Server Specifications**

* What is a proxy server?

A web server refers to software and hardware which uses the Hypertext Transfer Protocol (HTTP) to serve files to clients as a response to their requests. On the other hand, a proxy server sits between a web server and a client. When a user sends an HTTP request, it goes to the proxy server first. If the requested data is cached in the proxy server, it returns the cached data to the client. Otherwise, the request is forwarded to the server, which then returns the data to the client.

* What is the function of a proxy server?

One of the most critical functions of a proxy server is caching. A proxy server caches previously requested files and returns them to a client directly when they are requested again. With the proxy server, the server and network load will be lessened.

* Test procedure

The screenshot below shows that the initial HTTP request took more than 10 times longer than the second request. This is because the initial request went to the main server to grab the HTML file, where the second request used cached data from the proxy server.

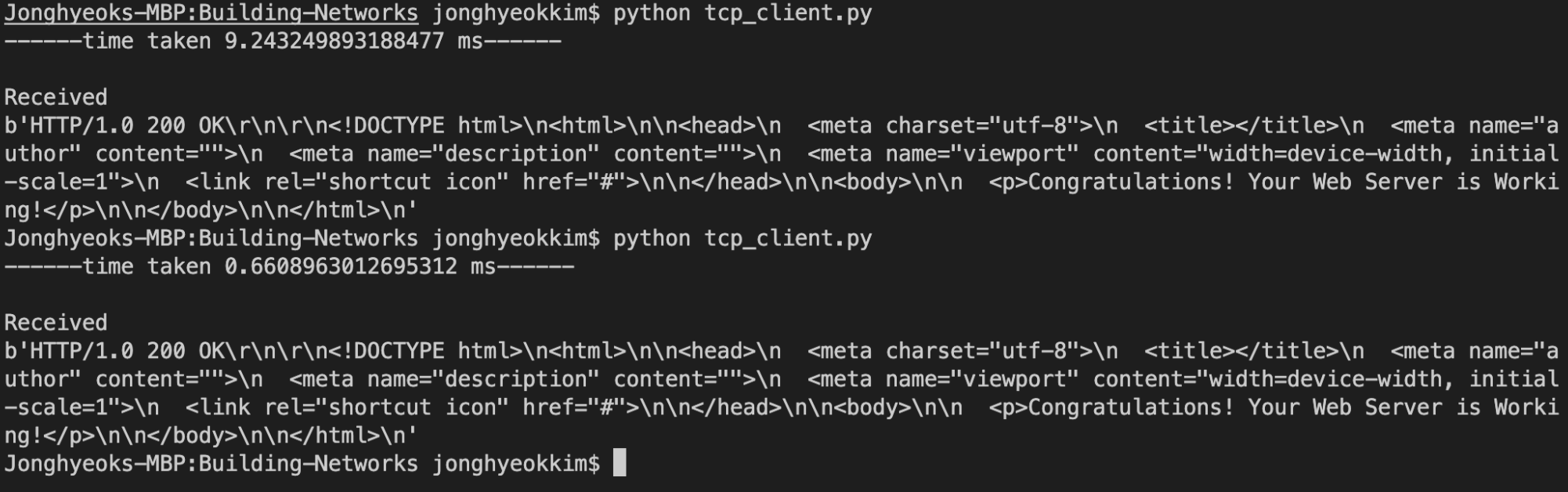


Figure 9: Comparison between the proxy server and main server response time

**Multi-Threaded Web Server Test Procedure**

Diagram

Description automatically generated

Figure 9: Multi-threaded web server overview

Since it is a minimal server, processing a HTTP request is almost instant. As a result, multi-threading almost works synchronously. I added some delay on the multi-threaded server to capture the nature of multi-threading. The code below shows that it prints the thread ID and port number twice, before and after the HTTP request is handled.

Text

Description automatically generated

Figure 10: Multi-threaded web server

If it were a non-multi-threaded server, the server would work synchronously, which means the server starts to handle a request and new requests wait for the existing requests to finish. On the other hand, a multi-threaded server does not wait for existing requests to finish. It starts to handle it as it receives a new request.Chart, waterfall chart

Description automatically generatedGraphical user interface, application

Description automatically generated

Figure 11: Single threaded server Figure 12: Multi-threaded server

The output below shows that our multi-threaded server started multi-thread as it receives a new request with a new port number. It also shows that new requests are not waiting for previous requests to finish. They start as soon as the multi-threaded server receives it.

Text

Description automatically generated with medium confidence

Figure 13: Our multi-threaded server result