Implementation of an HTTP client and server running a pared down version of HTTP/1.0. This project is coded in python.

PROGRAM DESIGN:

• The HTTP client and server are implemented using python programming language.

• The python version used is 3.7.0 (on windows 10 pro).

• The python’s socket and request modules are used to build the server.

HTTP Server:

• Server is initialized in the beginning.

• A method for server to launch on a specified port is defined.

• The number of concurrent requests the server can support is set to 3 in the get\_request method.

• The server is set around an infinite loop and needs to be interrupted with a terminal signal before it stops.

• The type of HTTP request method is identified using the socket request. Then, it will parse the input and serve the request accordingly.

• If server receives a GET request, then it sends out HTTP version:1.0, status code: 200, response phrase: OK to the client. It displays the content of that file.

• If server receives a PUT request, it saves the file as test.txt and on successful creation of this file, server responds to the client with “200 OK File Created” message.

HTTP Client:

• The client can invoke either the local server or any of the intranet servers for either sending files to those servers or receiving files from them.

• The client can send/receive files to/from local server, but for the real web servers, files can only be received/ retrieved from them.

• The client cannot upload a file on the real web servers.

• The requests sent here are http requests and have a specific format. For any other request than the valid http request, the server does not respond.

To run the HTTP server and Client:

Step1: Run the server

• Server: server.py --port=9999

Step2: Execute the client using GET method to retrieve file Step3: Execute the client using PUT method to upload file

• Client: Test cases client.py --host=127.0.0.1 --port=8080 --GET --file=index.html

client.py --host=www.cnn.com --port=80 --GET --file=index.html

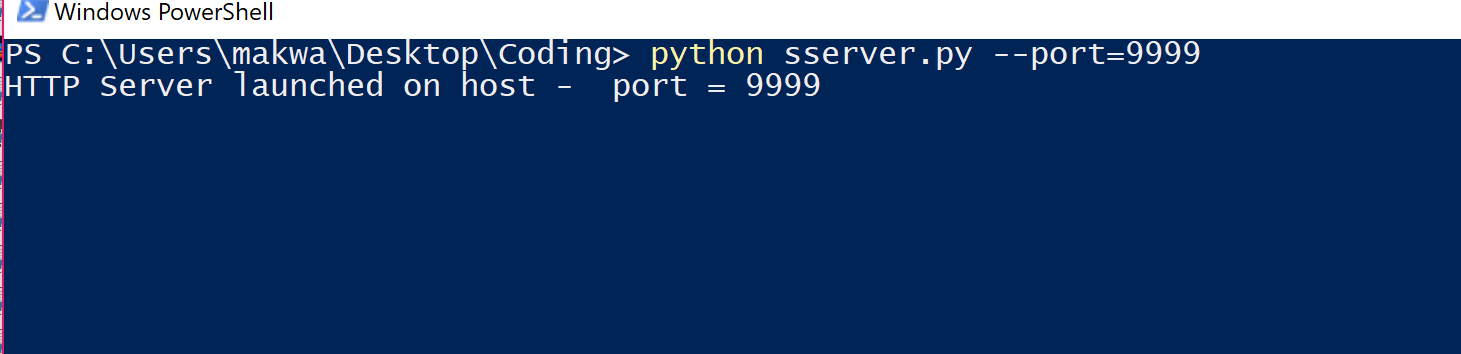
HTTP Request methods:

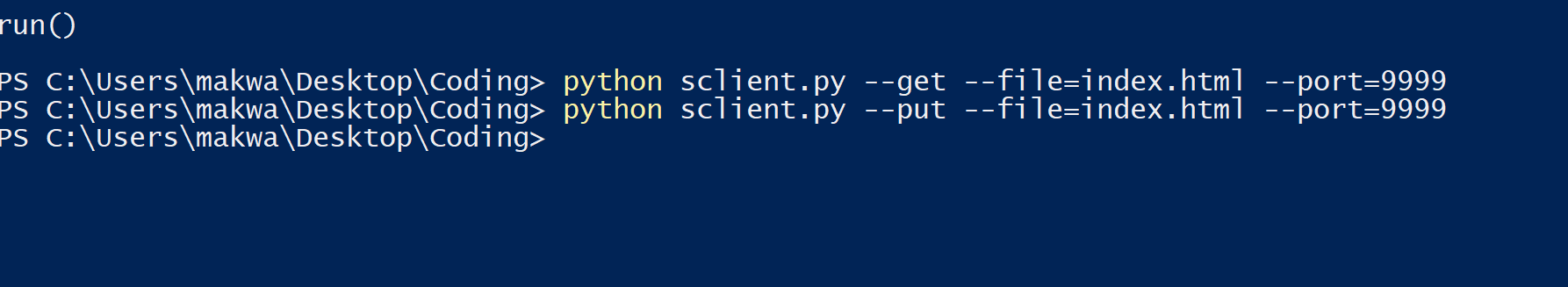
The HTTP request methods implemented in this program are as following.

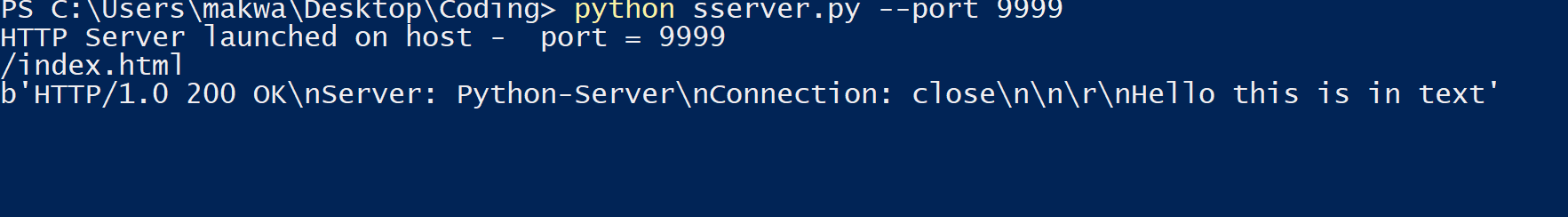
• GET: GET request indicates that the client wants to retrieve a file from the server specified.

• PUT: PUT request indicates that the client wants to upload a file to the server specified.

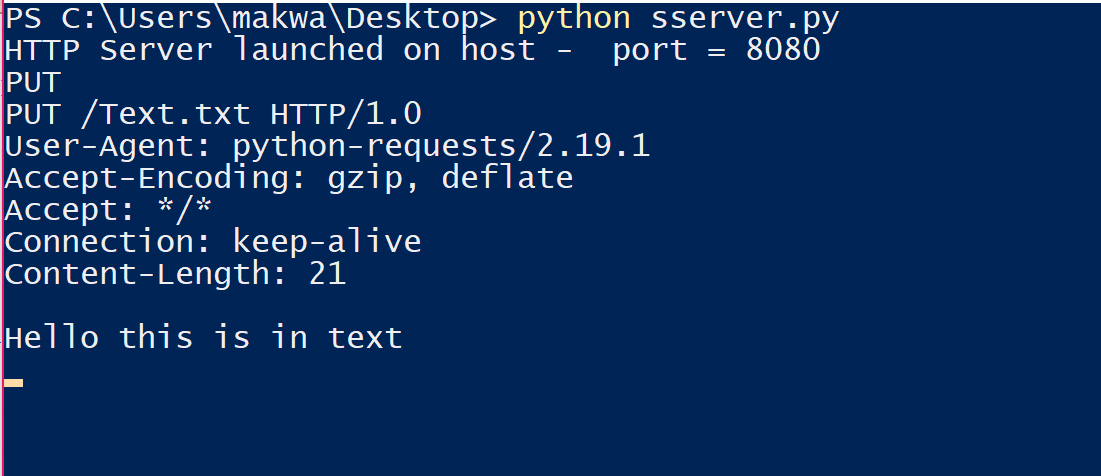
**Result:**











In your Python interpreter, type the following:

pip install requests

**Importing the Requests:** Module To work with the Requests library in Python, you must import the appropriate module.

You can do this simply by adding the following code at the beginning of your script:

import requests

**Making a Request:** When you ping a website or portal for information this is called making a request. That is exactly what the Requests library has been designed to do.

To get a webpage you would do something like the following: r = request.get(‘https://github.com/timeline.json’)

**Get the Content:** After a web server returns a response, you can collect the content you need. This is also done using the get requests function.

import requests r = requests.get('https://github.com/timeline.json') print r.text

# The Requests library also comes with a built-in JSON decoder, # just in case you have to deal with JSON data

import requests r = requests.get('https://github.com/timeline.json')

print r.json

**Make an HTTP Post Request:** You can also handle post requests using the Requests library.

r = requests.post(http://httpbin.org/post)

But you can also rely on other HTTP requests too, like PUT, DELETE, HEAD, and OPTIONS.

r = requests.put("http://httpbin.org/put") r = requests.delete("http://httpbin.org/delete") r = requests.head("http://httpbin.org/get")

r = requests.options("http://httpbin.org/get")