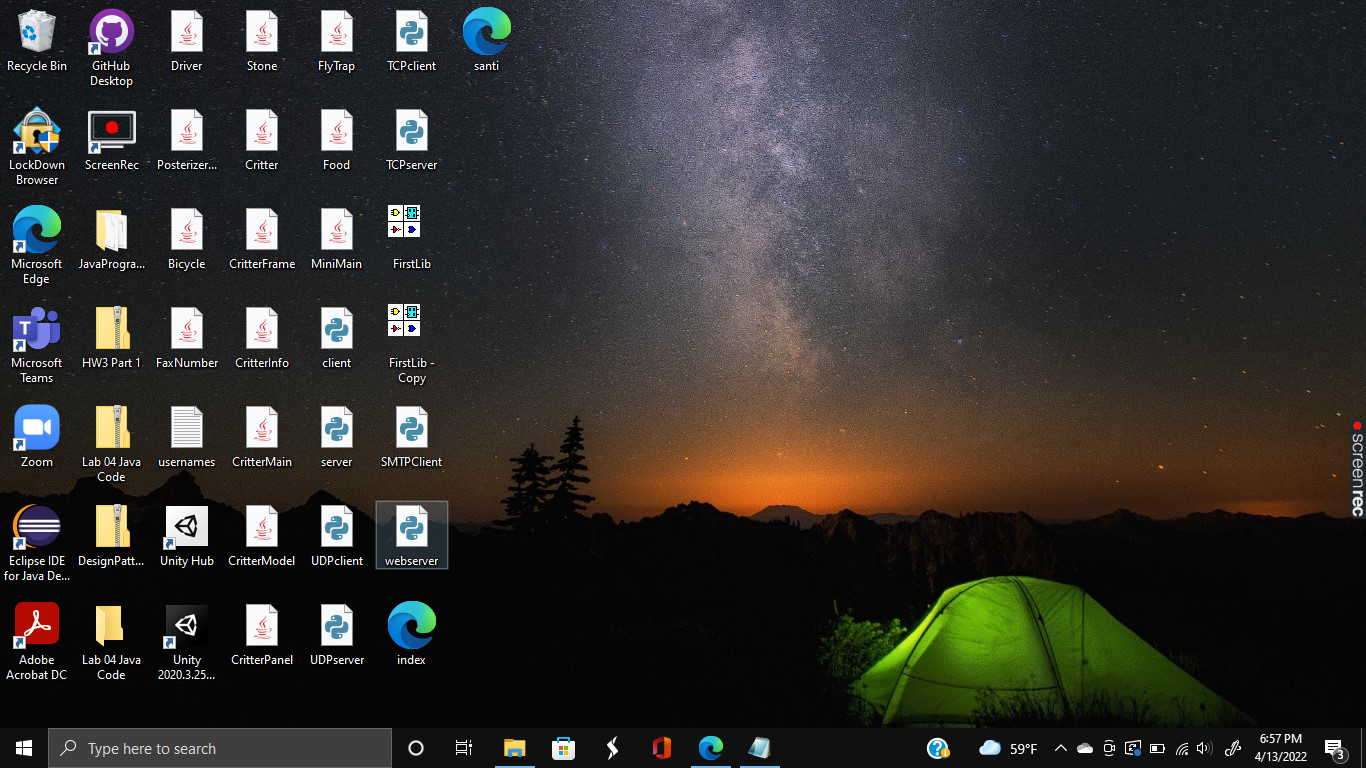
Author: Santiago Bermudez

CSC 138

Section 4

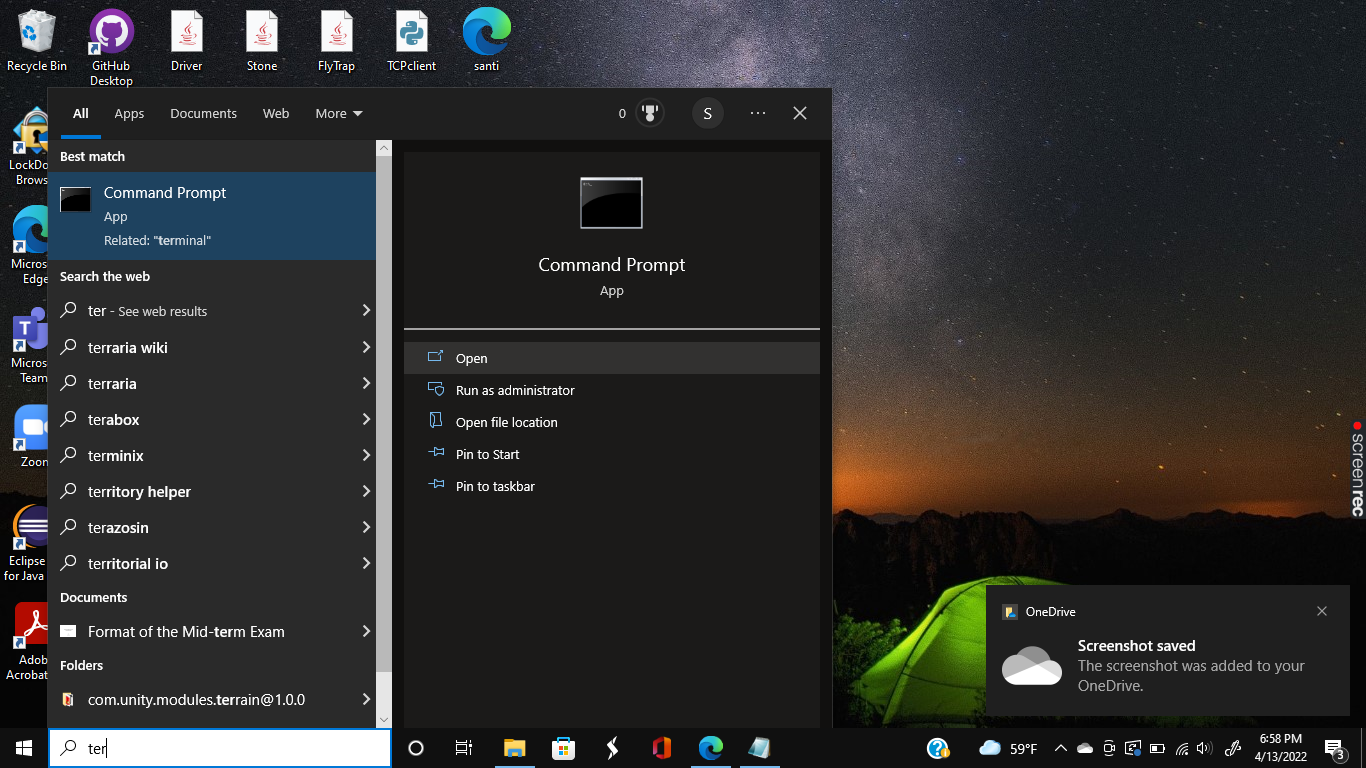
First of all, you will want to have your webserver.py program on your desktop.

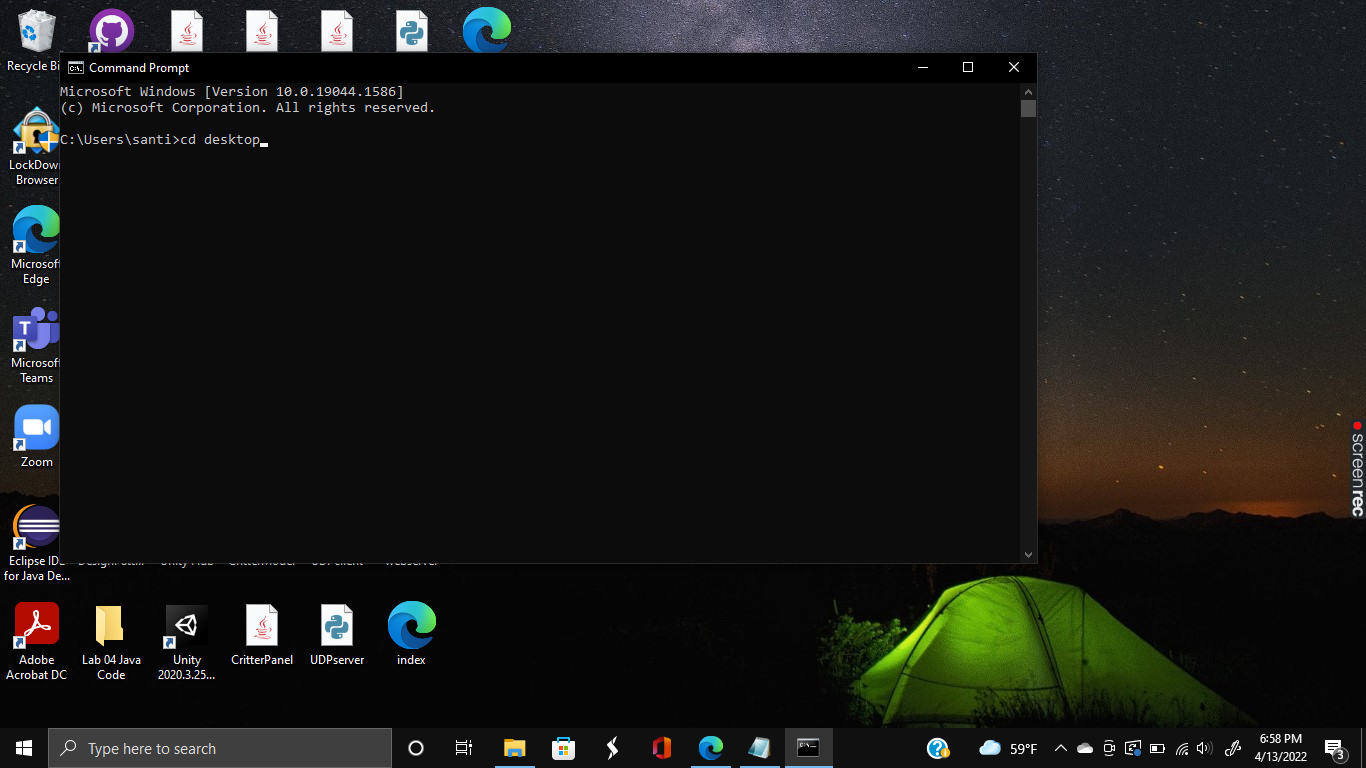


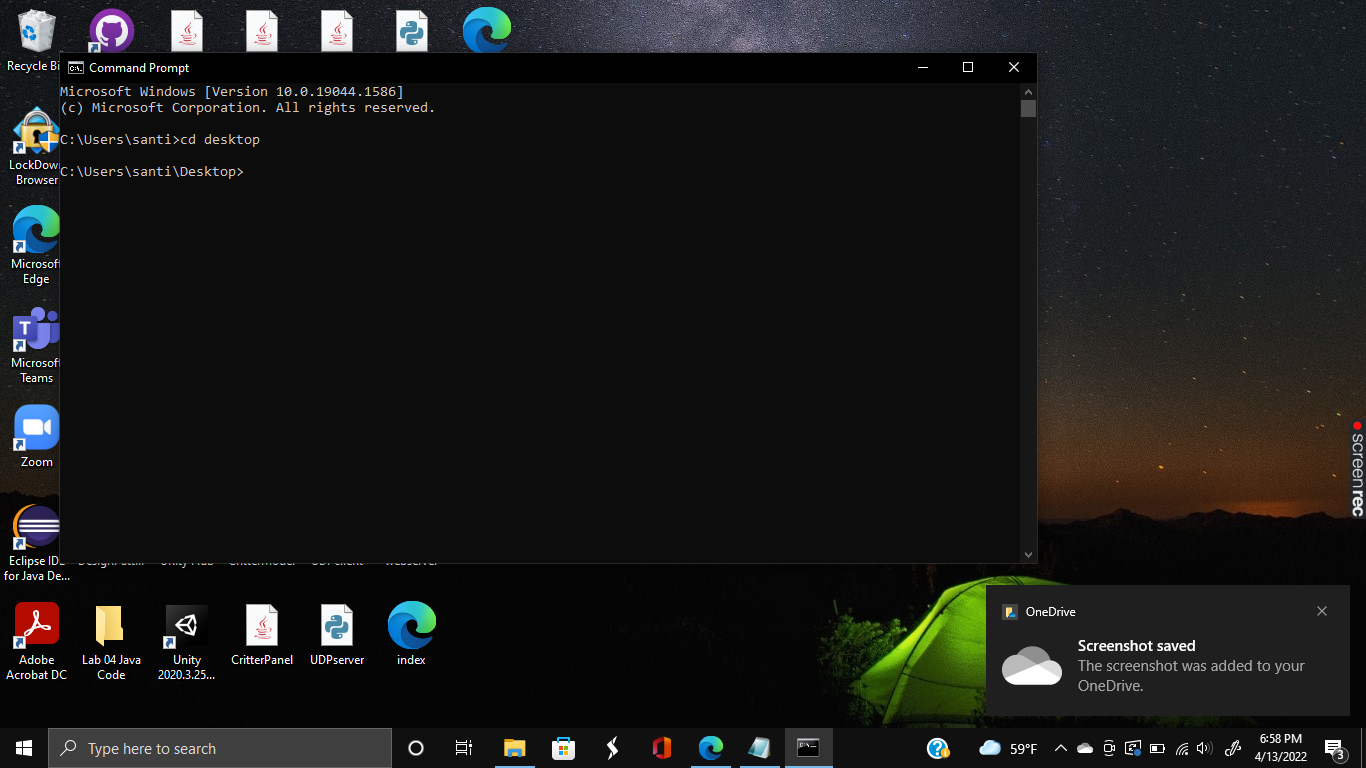
Then, you will want to have your HTML file on the desktop as well. Mine is called “santi.html”, so it looks like the following below:



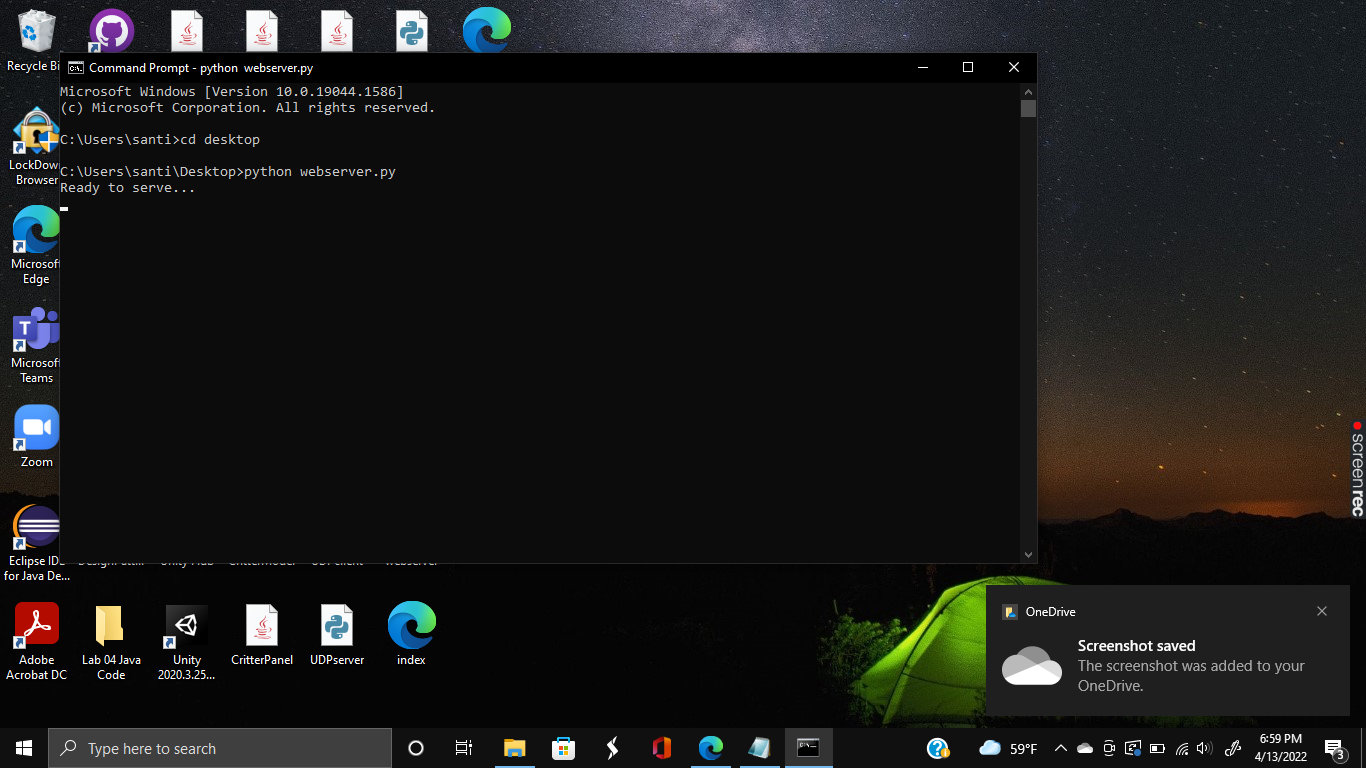
Then, you will want to open the terminal and change your directory to desktop.



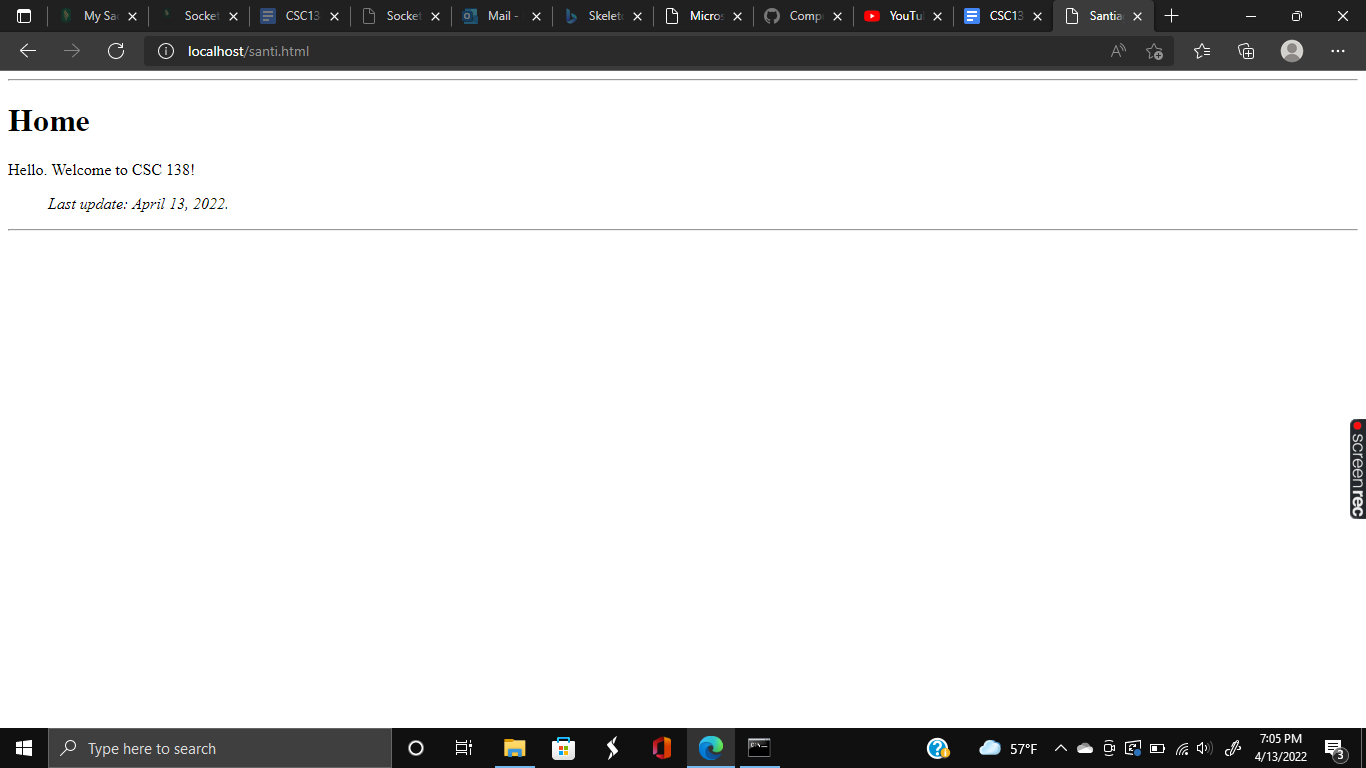




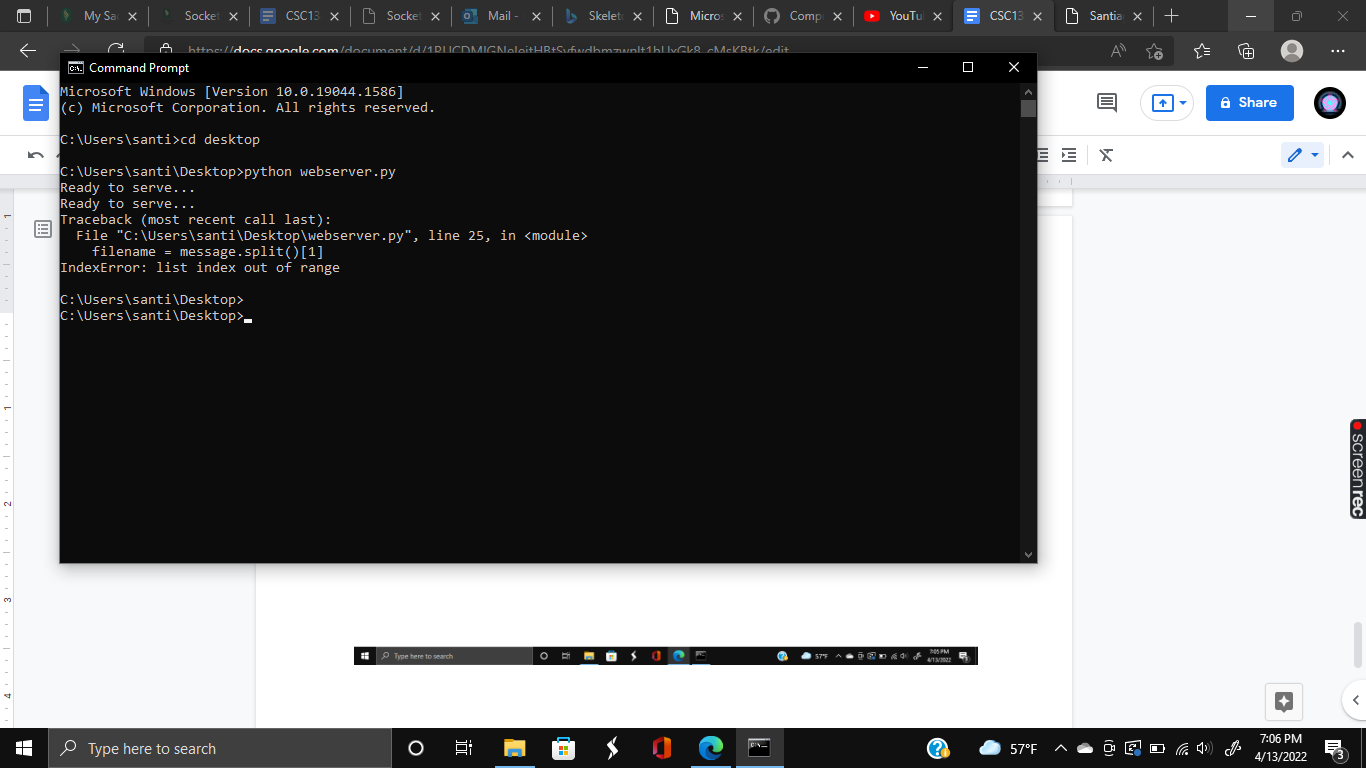
Once you have successfully changed your directory, you will want to run your Python program.



After you run the program and see the “Ready to serve…” message, you will want to open your HTML page on a browser as shown below. I entered “localhost:80/santi.html” in my browser in this case.



In order to know that you have succeeded, you can check your terminal for the response. In this case, you should get another “Ready to serve…” message underneath as feedback.



Things I learned/Mistakes I made/Choices I made and so on:

This assignment was for the most part straight forward and seeing how it built upon everything I had learned previously, I found my experience to be mainly a breeze. One thing I did have issues with however was HTML. Seeing how it was my first time using HTML, I had to do a bit of research and what not. I also found that in order to ensure that everything worked, I had to avoid CSS. For example, before I had something like <...style = ‘text-align:center’> in my code for decorating and I did not realize it was CSS. I had to get rid of such code to keep things simple and found that things worked much better without it while testing.

\*Python code for webserver.py:

#import socket module

from socket import \*

serverSocket = socket(AF\_INET, SOCK\_STREAM)

#Prepare a server socket

#Fill in start

serverPort = 80 #Assign a port number.

serverSocket.bind(('', serverPort)) #Bind the socket to server address and server port.

serverSocket.listen(1) #Listen to 1 connection at a time.

#^Server begins listening for incoming TCP requests.

#Fill in end

while True:

#Establish the connection

print('Ready to serve...')

connectionSocket, addr = serverSocket.accept() #Server waits on accept() for incoming requests.

#^New socket created on return!

try:

message = connectionSocket.recv(1024) #Read bytes from socket (but not address as in UDP).

filename = message.split()[1]

f = open(filename[1:])

outputdata = f.read()#Returns the specified number of bytes from the file.

#Send one HTTP header line into socket

#Fill in start

connectionSocket.send("HTTP/1.1 200 OK\r\n\r\n".encode())#HTTP response header line.

#Fill in end

#Send the content of the requested file to the client

for i in range(0, len(outputdata)):

connectionSocket.send(outputdata[i].encode())

connectionSocket.send("\r\n".encode())

connectionSocket.close()

except IOError:

#Send response message for file not found

#Fill in start

connectionSocket.send('HTTP/1.1 404 Not Found\r\n\r\n'.encode())

connectionSocket.send("<html><head></head><body><h1>404 Not Found</h1></body></html>\r\n")

#Fill in end

#Close client socket

#Fill in start

connectionSocket.close()

#Fill in end

serverSocket.close()

\*HTML code for santi.html:

<!DOCTYPE html>

<HTML lang = "en">

<HEAD>

<meta charset ="UTF-8">

<TITLE>Santiago Bermudez @ CSUS</Title>

</HEAD>

<BODY>

<hr>

<h1>Home</h1>

<span>Hello. Welcome to CSC 138!</span>

<blockquote><i>Last update: April 13, 2022.</i></blockquote>

<hr>

</BODY>

</HTML>