



# Python Programming: Web Server

## Part 1: What to Hand In

- Your code must be saved in a program file named **webServer.py**
- Submit the code to GradeScope **using your GitHub repository specifically created for this assignment.**
- Assignment submissions with **multiple files in the same GitHub repository will not be accepted.** Only one file per assignment is to be submitted.
- Multiple submission attempts are allowed up until the deadline or due date.

### IMPORTANT

! Your submission will be evaluated by a Client program testing the functionality of valid and invalid requests.

! **GradeScope will test your code using port 13331.**

## Part 2: Assignment Expectations and Instructions

**Objective:** In this lab, you will learn some of the basics of the HTTP header format and the basics of socket programming for TCP connections in Python, which include:

1. How to create a socket
2. Bind socket to a specific address and port
3. Send and receive an HTTP packet

### Required Textbook Reference Reading

- Chapter 2: 2.7 Socket Programming: Creating Network Applications


### Recommended Reference Reading

- Good tutorial on Python sockets, <https://realpython.com/python-sockets/>
- Socket library docs, <https://docs.python.org/3/library/socket.html>
- MDN Basics of HTTP, [https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics\\_of\\_HTTP](https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP)

### Assignment Instructions:

- You are expected to complete the skeleton code for a web server by providing the missing code in the sections marked with comments **#Fill in start** and **#Fill in end**.




 **Hint:** Each section may require one or more lines of code.


**It is not recommended to deviate or remove any skeleton code.**

You are an engineer, so you are welcome to add code outside the fill in start and fill in end areas. However, if you delete skeleton code, the **TAs are limited on the support we can provide if you drift too far off the intended path.**

You **never** need to modify code outside the start and end fill areas, **do so at your own peril.**

 **The skeleton code can be found at this link:**  
<https://github.com/NYUCyberFellows-CSGY6843/assignment2-webserver>

- Using the skeleton code provided, you will develop a web server that handles **one** HTTP request at a time.
- Your web server should:
  - Accept and parse the HTTP request
  - Get the requested file from the server's file system
  - Create an HTTP response message consisting of the requested file preceded by header lines
  - Send the response directly to the client.
- For the header 'Server' feel free to use one of your own choice/making.
- Ensure you include all relevant header lines. If you forget any required headers, GradeScope will tell you.
- If the requested file is not present on the server, the server should send an HTTP "404 Not Found" error message back to the client.
  - i.e. a valid request should serve an HTML file to the client, and an invalid request should serve a "404 Not Found" error message.

 **Hint:** There are clients (browsers) that will not present HTML content unless encoded HTTP headers are submitted (such as Content-Type, Server, Connection) with the message from the web server.

**IMPORTANT**

- ! HTTP status codes “[200 OK](#)” and “[404 Not Found](#)” are **required** to be part of the Web Server in order to receive full credit on this assignment. Follow the official format of those responses per the [REC](#).
- ! Do not send line by line. Prepare your entire transmission and send as one event. That means, **send should be called only once** for transmission.

### Part 3: Running your server locally

- ✓ You should always test locally to verify that your server handles valid and invalid requests as expected.

#### Steps for running the server locally:

1. Put an HTML file (e.g., helloworld.html) in the same directory that the server is in.
2. Run the server program (e.g. python webServer.py).
3. Determine the IP address of the host that is running the server (e.g., 127.0.0.1).
4. Open a browser and provide the corresponding URL.
  - a. For example: <http://127.0.0.1:13331/helloworld.html>

💡 **Hint:** ‘helloworld.html’ is the name of the file you placed in the **server directory**.

- ✓ Note that in the above example, we have made use of the port number after the colon and have chosen to use port number 13331 in the URL.
- ✓ If you omit “:13331”, the browser will assume port 80, and you will get the web page from the server only if your server is listening at port 80.

5. Next, try to get a file that is not present at the server.
  - a. You should get a “404 Not Found” message.



---

## Part 4: FAQ

**Q:** I am getting the following error in gradescope:

*`cp: cannot stat '/autograder/submission/webServer.py': No such file or directory`*

**A:** If you submit a Python solution, this Python assignment submission must have the filename titled “webServer.py” (minus the quotation marks). Make sure your file meets this naming requirement.

## Part 5: Most Common Issues

1. Print statements are not commented out. If uncommented, they may cause errors in your GradeScope submission. Print statements should only be used for local troubleshooting.
2. Improper encoding
  - a. See <https://pythontic.com/modules/socket/send>
  - b. You can send data as bytes in a variety of formats. You can even just read data from a file as bytes to avoid having to configure encoding later.
3. Inappropriate socket management (i.e. not opening or closing the socket appropriately)
4. Syntactical errors