# A Simple Web Server in Python

by:

Eddy Zhang

Tufts University

EE129 - Computer Networks

March 25 2024

# Introduction

This project explores the development of a basic web server in Python, showcasing the fundamentals of socket programming and web technologies. The objective was to create a server-client communication system using the HTTP protocol, illustrating the flow of data between a web server and a browser.

In this project, we can interact with the webpage by sending different requests through url.
When requested by a web browser, the web server checks the URL sent by browser and
act accordingly:

- If the URL is just “/”, it displays a page with a form for the user to select from 2 different colors (red, green), then submit it to the web server with the GET method.

- If the URL has the keyword “red”, it sends back a web page with a sentence:

  Your color is red!

  with the red color.

- If the URL has the keyword “green”, it should send back a web page with a sentence:

  Your color is green!

  with the green color.

- Otherwise, the requested page does not exist. Return the status code 404 to the web browser.

# Experimental Setup

What did you do?

This section should be detailed enough so that someone could replicate your experiment, while keeping it clear and concise.

- Download python.
  https://www.python.org/downloads/

- Create a project folder, and in the project folder create a file called main.py.

- Import all the libraries. In this project only the socket library is used


- Set up the TCP server socket.

```
# Define the IP address and port number
HOST = '127.0.0.1'
PORT = 12345

# Create a TCP server socket
server_socket = socket(AF_INET, SOCK_STREAM)

# Bind the socket to the address and port
server_socket.bind((HOST, PORT))

# Listen for incoming connections
server_socket.listen(1)
```

- Write the html code for the webpage to show. I wrapped it in a function called create_from.

```
def create_form(message=''):
    return f"""HTTP/1.1 200 OK
Content-Type: text/html

<html>
<head>
    <title>Color Selection</title>
    <style>
        html {{
            height: 100%;
        }}

        body{{
```

- Constantly check, accept, and process connections from a client and send back the corresponding results.

```
while True:
    # Accept a connection from a client
    connection_socket, addr = server_socket.accept()

    # Receive the request from the client
    request = connection_socket.recv(1024).decode()

    # Parse the request to get the requested URL
    url = request.split()[1]

    color = url.split('=')[1]

    if url == '/':
        # Display the color selection form
        response = create_form()
    elif url.startswith('/submit') and '=' in url and (color == "red" or color == "green"):
        # Process the form submission
        response = create_form(f'Your color is <span style="color:{color};">{color}</span>!')
    else:
        # Respond with a 404 status code for unknown URLs
        response = """HTTP/1.1 404 Not Found
                    Content-Type: text/html

                    <html>
                    <head><title>Not Found</title></head>
                    <body>
                    404 Not Found
                    </body>
                    </html>
                    """

    # Send the response back to the client
    connection_socket.sendall(response.encode())

    # Close the connection
    connection_socket.close()
```

- After finishing the code, run the code by using command

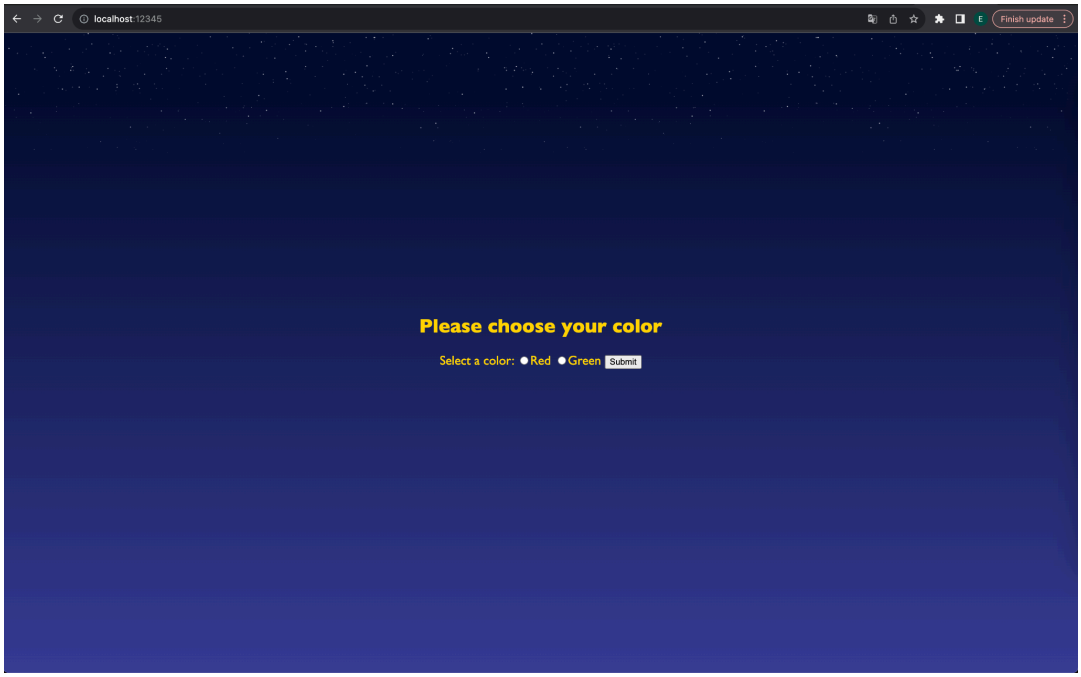
  Python main.py

  Check the webpage at

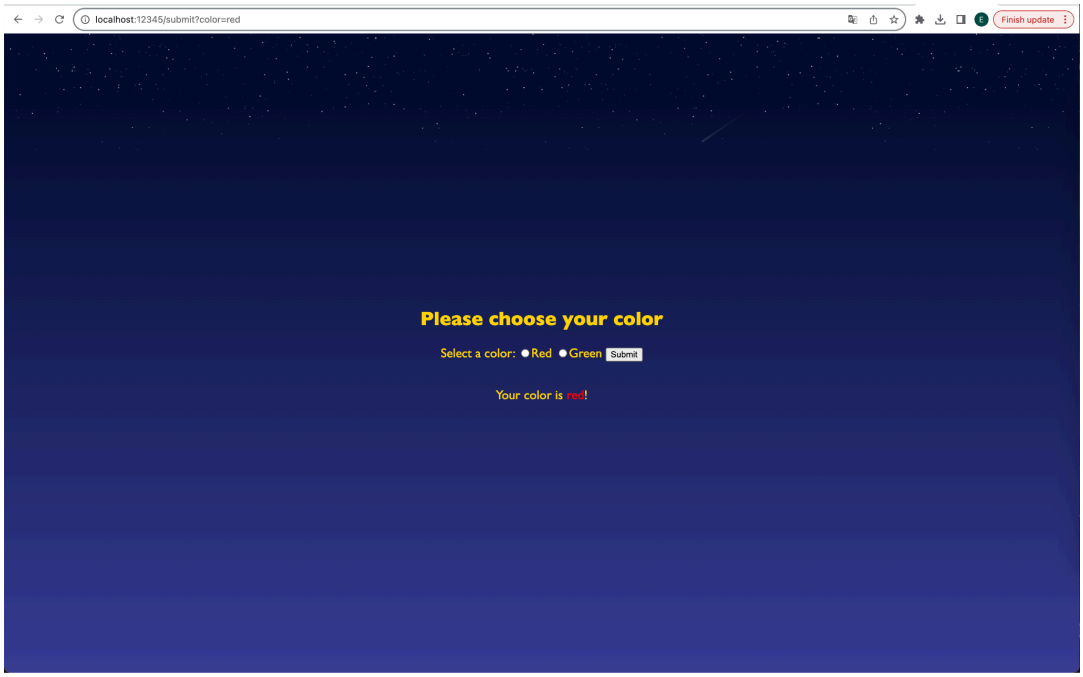
  localhost:your_port_#

# Results

The webpage when nothing is submitted:

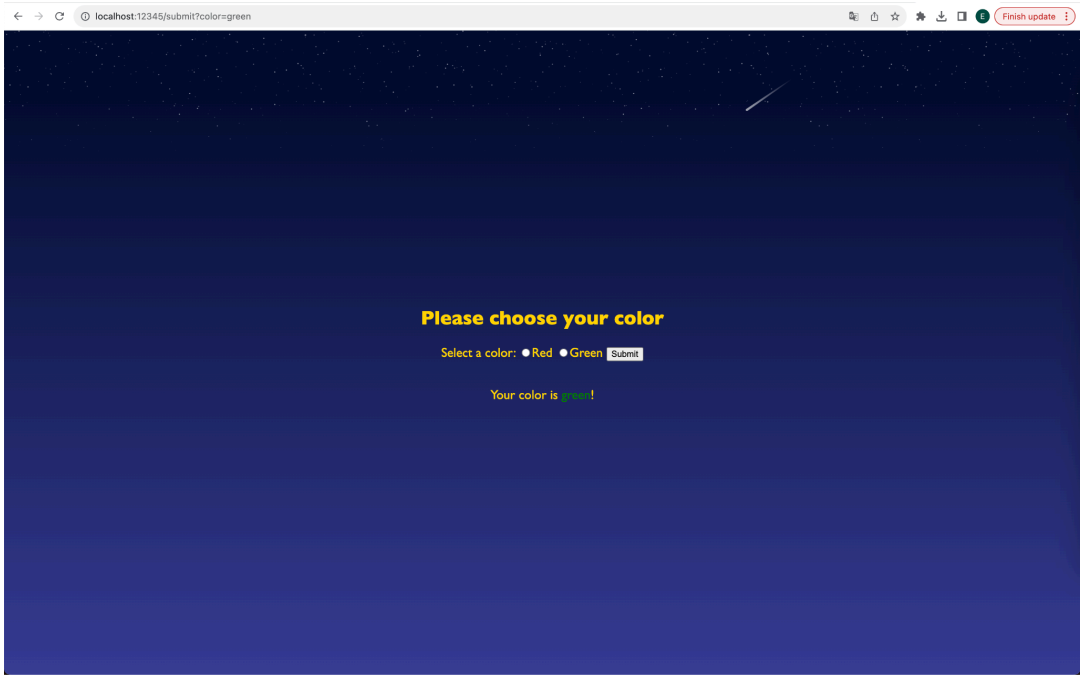



The webpage when the submission is red (i.e. when Red is selected and submitted):



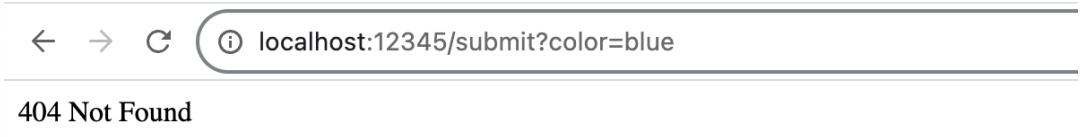

As shown in the picture, now the url contains “/submit?color=red”

The webpage when the submission is green (i.e. when Green is selected and submitted):




As shown in the picture, now the url contains “/submit?color=red”

The webpage when the url contains something else than the cases we mentioned above:




404 Not Found

# Discussion

Ther result indicates that the server received and processed the client's request and updated the webpage correspondingly. It successfully showed the client-server interaction through HTTP protocol.

There are more 200 lines of code written for this project, but more than 70% of them are html and css codes for the additional decoration for the webpage, the main body of the code is quite simple and concise.

The biggest problem I faced in this project happened when the url had undesired formats the server could not parse the url and therefore gave error and turned down the server. The bug was fixed by adding condition checks.

# Conclusion

In this project I used python to build a TCP socket server that could interact with clients through HTTP protocol. The server listened to, receiveed, accepted and processed client's submission in url and sent back the corresponding response updating the webpage.

# Appendix

- Source Code: https://github.com/Eddyzzzzz/Project/blob/main/Project2/main.py