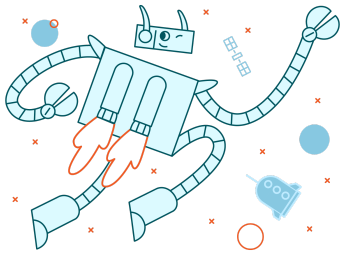
Hello, HTTP Server



Goal

In this exercise, we'll create a simple HTTP server using Python that responds to GET requests with a basic "Hello, World!" message.

Background

HTTP (Hypertext Transfer Protocol) is the foundation of communication on the World Wide Web. Servers listen for HTTP requests from clients and send back responses. In this exercise, we'll build a basic HTTP server that listens for incoming GET requests and sends a simple response.

Tasks

1. Create a new Python file called `simple_http_server.py`.
2. Import the necessary module:

```
import socket
```

3. Define a function called `handle_request` that takes a client socket as a parameter. This function will handle incoming client requests.

```
def handle_request(client_socket):
    # Your code here
```

4. Inside the `handle_request` function, receive the client request data using `client_socket.recv(1024).decode()`.
5. Parse the request to get the requested path (e.g., "/", "/hello", etc.).

6. If the requested path is "/", send a response with a "200 OK" status and a simple HTML message:

```
<html>
<body>
    <h1>Welcome to my simple HTTP server!</h1>
</body>
</html>
```

7. If the requested path is anything else, send a response with a "404 Not Found" status and a simple error message.

8. Close the client socket after sending the response.

9. Define a `start_server` function that creates a server socket, binds it to a host and port, and starts listening for incoming connections.

```
def start_server():
    # Your code here
```

10. Inside the `start_server` function, create a loop that accepts client connections, calls the `handle_request` function to handle each request, and then closes the client socket.

11. Call the `start_server` function to start your HTTP server.

12. **Note: you are not allowed to use the builtin http.server module in this exercise.** Well, you can try, but that's cheating. Don't do that.

Example

Here's an example of how your `simple_http_server.py` file might look:

```
import socket

def handle_request(client_socket):
    request_data = client_socket.recv(1024).decode()
    # Get the requested path from the request
    # ...

    if requested_path == "/":
        response = "HTTP/1.1 200 OK\r\n\r\n"
        response += "<html><body><h1>Welcome to my
simple HTTP server!</h1></body></html>"
    else:
        response = "HTTP/1.1 404 Not Found\r\n\r\n"
        response += "<html><body><h1>404 Not
Found</h1></body></html>"

    client_socket.send(response.encode())
    client_socket.close()

def start_server():
    server_socket = socket.socket()
    server_socket.bind(("localhost", 8000))
    server_socket.listen()
    print("Server is running on
http://localhost:8000")

    while True:
        client_socket, address =
server_socket.accept()
        print(f"New connection from {address}")
        handle_request(client_socket)

start_server()
```

Running the server

1. Save your `simple_http_server.py` file.
2. Open a terminal or command prompt and navigate to the directory where you saved the file.
3. Run the server using the command: `python simple_http_server.py`.
4. Open a web browser and visit `http://localhost:8000` to see the welcome message.
5. Try visiting other paths like `http://localhost:8000/hello` to see the 404 Not Found message.

Congratulations! You've created a simple HTTP server that responds to GET requests with basic messages and immediately closes the connection after sending the response.

To submit

- Your `simple_http_server.py` file containing the complete code for the HTTP server.