Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет инфокоммуникационных технологий

**Лабораторная работа № 1**

**по дисциплине**

**«Веб-программирование»**

**Выполнил**:

группы D3310

Ван Цзиян

Санкт-Петербург

2024

**Задание 1:**

Реализовать клиентскую и серверную часть приложения. Клиент отправляет серверу сообщение «Hello, server», и оно должно отобразиться на стороне сервера. В ответ сервер отправляет клиенту сообщение «Hello, client», которое должно отобразиться у клиента.

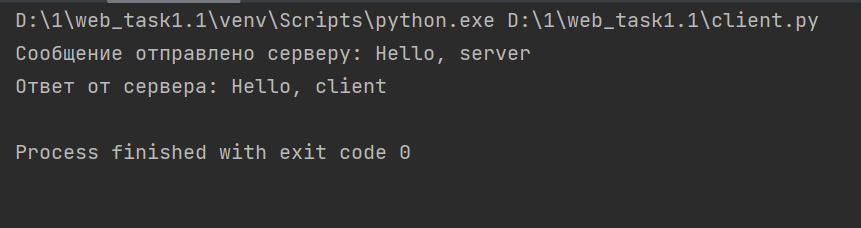
**Требования:**

* Обязательно использовать библиотеку socket.
* Реализовать с помощью протокола UDP.

Code:

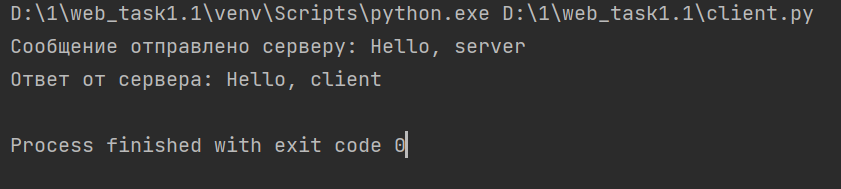
server.py:

import socket  
  
def udp\_server():  
 # Создаем UDP сокет  
 server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)  
  
 # Привязываем сокет к адресу и порту  
 server\_address = ('localhost', 12345)  
 server\_socket.bind(server\_address)  
  
 print("Сервер запущен и ожидает сообщение...")  
  
 while True:  
 # Ждем сообщения от клиента  
 message, client\_address = server\_socket.recvfrom(1024)  
 print(f"Получено сообщение от клиента: {message.decode()}")  
  
 # Отправляем ответ клиенту  
 response = "Hello, client"  
 server\_socket.sendto(response.encode(), client\_address)  
 print("Ответ отправлен клиенту.")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 udp\_server()



client.py:

import socket  
  
def udp\_client():  
 # Создаем UDP сокет  
 client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)  
  
 # Адрес сервера  
 server\_address = ('localhost', 12345)  
  
 # Отправляем сообщение серверу  
 message = "Hello, server"  
 client\_socket.sendto(message.encode(), server\_address)  
 print(f"Сообщение отправлено серверу: {message}")  
  
 # Получаем ответ от сервера  
 response, \_ = client\_socket.recvfrom(1024)  
 print(f"Ответ от сервера: {response.decode()}")  
  
 # Закрываем сокет  
 client\_socket.close()  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 udp\_client()



**Задание 2:**

Реализовать клиентскую и серверную часть приложения. Клиент запрашивает выполнение математической операции, параметры которой вводятся с клавиатуры. Сервер обрабатывает данные и возвращает результат клиенту.

**Варианты операций:**

1. Теорема Пифагора.
2. Решение квадратного уравнения.
3. Поиск площади трапеции.
4. Поиск площади параллелограмма.

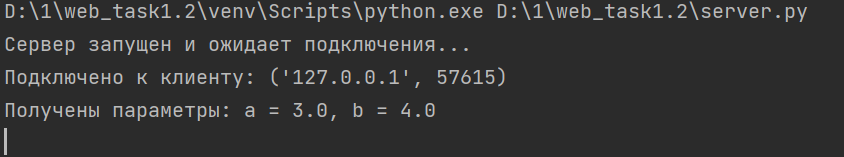
**Порядок выбора варианта:** Выбирается по порядковому номеру в журнале (пятый студент получает вариант 1 и т.д.).

**Вариант 1:**

Code:

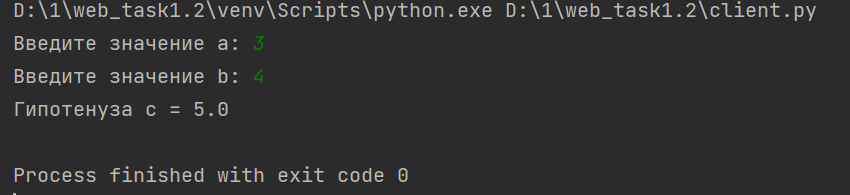
server.py:

import socket  
import math  
  
def tcp\_server():  
 # Создаем TCP сокет  
 server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
  
 # Привязываем сокет к адресу и порту  
 server\_address = ('localhost', 12345)  
 server\_socket.bind(server\_address)  
  
 # Начинаем прослушивание входящих соединений  
 server\_socket.listen(1)  
 print("Сервер запущен и ожидает подключения...")  
  
 while True:  
 # Принимаем подключение от клиента  
 connection, client\_address = server\_socket.accept()  
 try:  
 print(f"Подключено к клиенту: {client\_address}")  
  
 # Получаем данные от клиента  
 data = connection.recv(1024).decode()  
 a, b = map(float, data.split(","))  
 print(f"Получены параметры: a = {a}, b = {b}")  
  
 # Вычисляем гипотенузу  
 c = math.sqrt(a\*\*2 + b\*\*2)  
  
 # Отправляем результат обратно клиенту  
 connection.sendall(str(c).encode())  
 finally:  
 connection.close()  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 tcp\_server()



client.py:

import socket  
  
def tcp\_client():  
 # Создаем TCP сокет  
 client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
  
 # Указываем адрес сервера  
 server\_address = ('localhost', 12345)  
 client\_socket.connect(server\_address)  
  
 try:  
 # Запрашиваем ввод параметров у пользователя  
 a = input("Введите значение a: ")  
 b = input("Введите значение b: ")  
  
 # Отправляем параметры на сервер  
 message = f"{a},{b}"  
 client\_socket.sendall(message.encode())  
  
 # Получаем результат от сервера  
 result = client\_socket.recv(1024).decode()  
 print(f"Гипотенуза c = {result}")  
 finally:  
 client\_socket.close()  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 tcp\_client()



**Задание 3:**

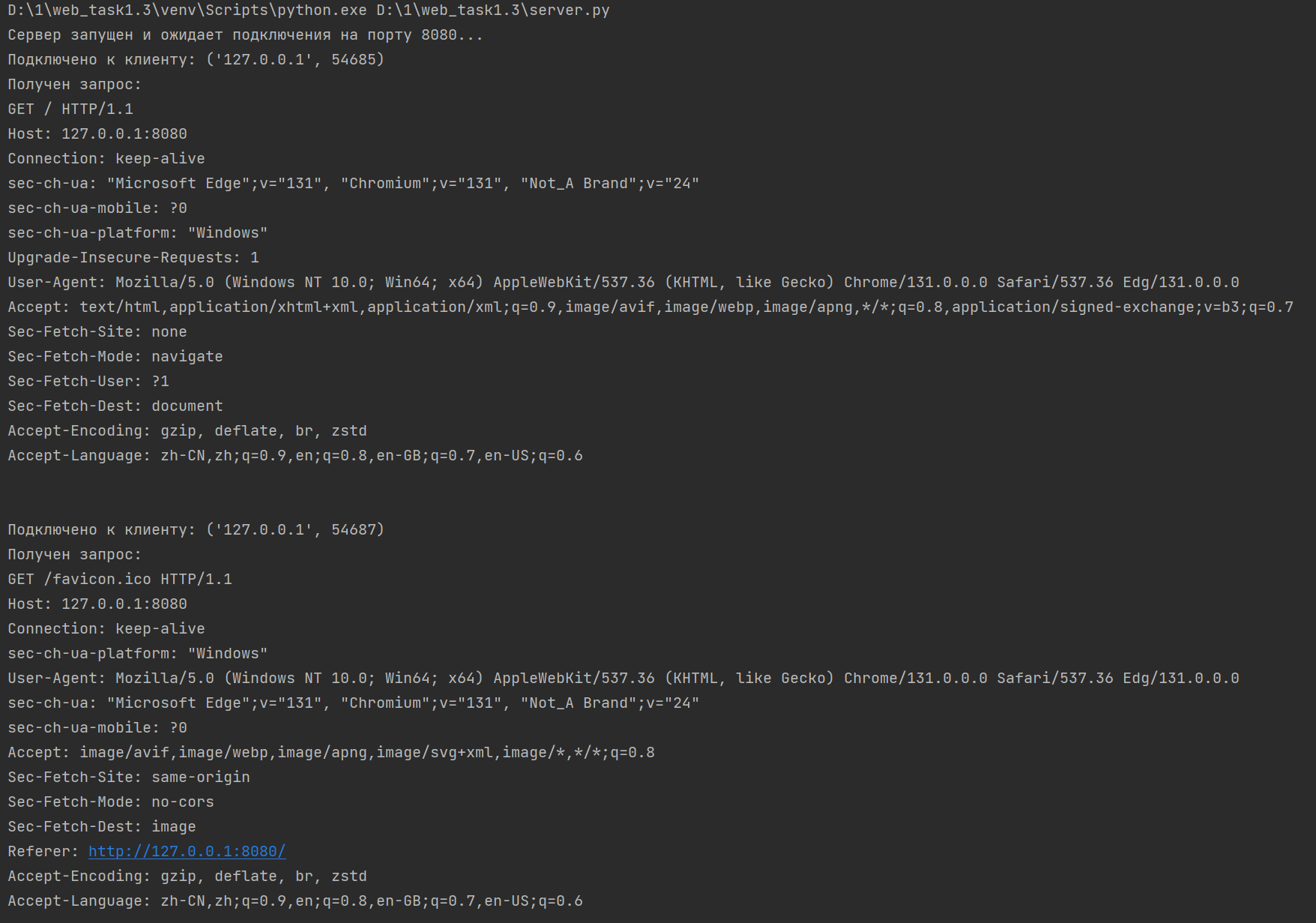
Реализовать серверную часть приложения. Клиент подключается к серверу, и в ответ получает HTTP-сообщение, содержащее HTML-страницу, которая сервер подгружает из файла index.html.

Code:  
index.html:

<!DOCTYPE html>  
<html lang="ru">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Привет, мир!</title>  
</head>  
<body>  
 <h1>Добро пожаловать на наш сервер!</h1>  
 <p>Это пример HTML-страницы, обслуживаемой через сокеты.</p>  
</body>  
</html>

server.py:

import socket  
  
def http\_server():  
 # Создаем TCP сокет  
 server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
  
 # Привязываем сокет к адресу и порту  
 server\_address = ('localhost', 8080)  
 server\_socket.bind(server\_address)  
  
 # Начинаем прослушивание входящих соединений  
 server\_socket.listen(1)  
 print("Сервер запущен и ожидает подключения на порту 8080...")  
  
 while True:  
 # Принимаем подключение от клиента  
 connection, client\_address = server\_socket.accept()  
 try:  
 print(f"Подключено к клиенту: {client\_address}")  
  
 # Получаем запрос от клиента  
 request = connection.recv(1024).decode()  
 print(f"Получен запрос:\n{request}")  
  
 # Загружаем содержимое HTML-файла  
 with open('index.html', 'r', encoding='utf-8') as file:  
 html\_content = file.read()  
  
 # Формирование HTTP-ответа  
 http\_response = f"HTTP/1.1 200 OK\nContent-Type: text/html; charset=utf-8\n\n{html\_content}"  
  
 # Отправляем ответ клиенту  
 connection.sendall(http\_response.encode())  
 except FileNotFoundError:  
 # Обработка ошибки, если файл не найден  
 http\_response = "HTTP/1.1 404 Not Found\n\n<h1>404 Not Found</h1>"  
 connection.sendall(http\_response.encode())  
 finally:  
 connection.close()  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 http\_server()





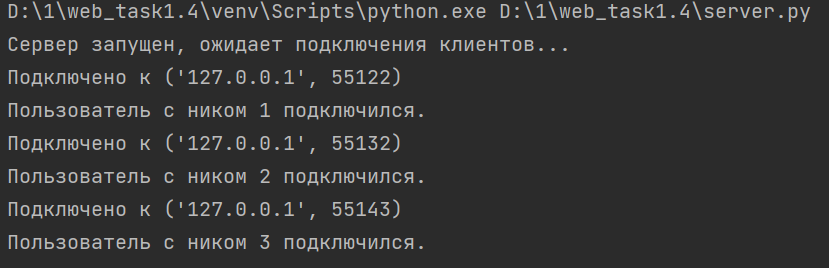
**Задание 4:**

Реализовать двухпользовательский или многопользовательский чат. Для максимального количества баллов реализуйте многопользовательский чат.

Code:

server.py:

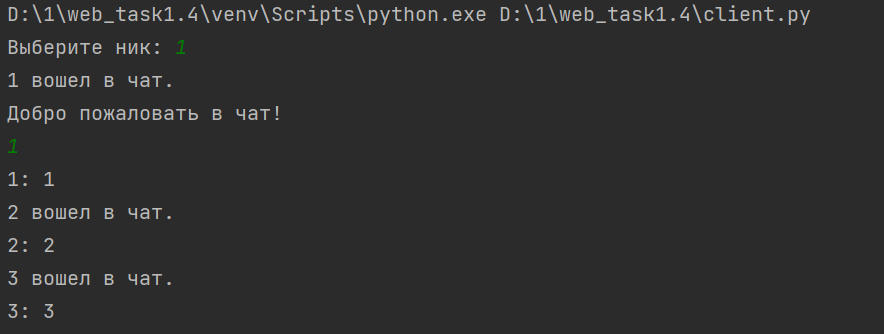
import socket  
import threading  
  
# Хранит подключенных клиентов  
clients = []  
nicknames = []  
  
def broadcast(message):  
 for client in clients:  
 client.send(message)  
  
def handle\_client(client):  
 while True:  
 try:  
 # Прием сообщения  
 message = client.recv(1024)  
 broadcast(message)  
 except:  
 index = clients.index(client)  
 clients.remove(client)  
 client.close()  
 nickname = nicknames[index]  
 broadcast(f"{nickname} покинул чат.".encode('utf-8'))  
 nicknames.remove(nickname)  
 break  
  
def receive():  
 while True:  
 client, address = server.accept()  
 print(f"Подключено к {str(address)}")  
  
 client.send("Пожалуйста, введите ваш ник: ".encode('utf-8'))  
 nickname = client.recv(1024).decode('utf-8')  
 nicknames.append(nickname)  
 clients.append(client)  
  
 print(f"Пользователь с ником {nickname} подключился.")  
 broadcast(f"{nickname} вошел в чат.".encode('utf-8'))  
 client.send("Добро пожаловать в чат!".encode('utf-8'))  
  
 # Создать новый поток для каждого клиента  
 thread = threading.Thread(target=handle\_client, args=(client,))  
 thread.start()  
  
server = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
  
# Привязка IP-адреса сервера и порта  
server.bind(('127.0.0.1', 55555))  
server.listen()  
  
print("Сервер запущен, ожидает подключения клиентов...")  
receive()

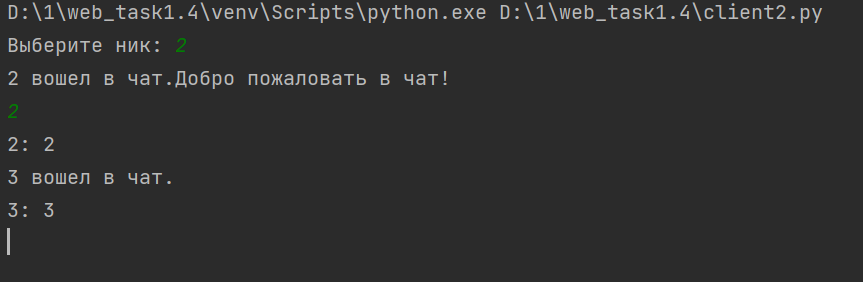


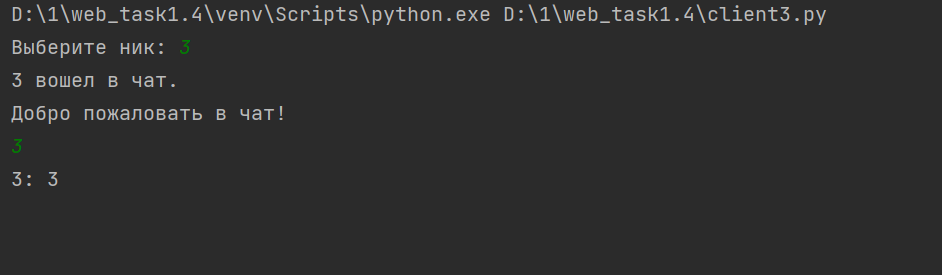
client.py:

import socket  
import threading  
  
def receive():  
 while True:  
 try:  
 message = client.recv(1024).decode('utf-8')  
 if message == 'Пожалуйста, введите ваш ник: ':  
 client.send(nickname.encode('utf-8'))  
 else:  
 print(message)  
 except:  
 print("Произошла ошибка! Невозможно подключиться к серверу.")  
 client.close()  
 break  
  
def write():  
 while True:  
 message = f"{nickname}: {input('')}"  
 client.send(message.encode('utf-8'))  
  
client = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
  
# Подключение к серверу  
client.connect(('127.0.0.1', 55555))  
  
nickname = input("Выберите ник: ")  
  
# Запуск потоков для приема и отправки сообщений  
receive\_thread = threading.Thread(target=receive)  
receive\_thread.start()  
  
write\_thread = threading.Thread(target=write)  
write\_thread.start()

Три клиентских кода одинаковы







**Задание 5:**

Написать простой веб-сервер для обработки GET и POST HTTP-запросов с помощью библиотеки socket в Python.

**Задание:**

* Сервер должен:
  1. Принять и записать информацию о дисциплине и оценке по дисциплине.
  2. Отдать информацию обо всех оценках по дисциплинам в виде HTML-страницы.

Code:

import socket  
from urllib.parse import parse\_qs  
  
# Store subjects and grades  
grades = {}  
  
def handle\_request(request):  
 method, path, \_ = request.split(' ', 2)  
  
 if method == 'POST' and path == '/submit':  
 # Handle POST request  
 body = request.split('\r\n\r\n')[1]  
 data = parse\_qs(body)  
  
 # Extract subject and grade  
 subject = data.get('subject', [''])[0]  
 grade = data.get('grade', [''])[0]  
  
 # Save information  
 if subject and grade:  
 grades[subject] = grade  
  
 # Return response  
 response = 'HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n'  
 response += '<html><body><h1>Data has been successfully saved!</h1>'  
 response += '<a href="/grades">View all grades</a><br>'  
 response += '<a href="/">Go back</a></body></html>'  
 return response  
  
 elif method == 'GET' and path == '/grades':  
 # Handle GET request  
 response = 'HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n'  
 response += '<html><body><h1>Subjects and Grades</h1>'  
 response += '<table border="1">'  
 response += '<tr><th>Subject</th><th>Grade</th></tr>'  
  
 for subject, grade in grades.items():  
 response += f'<tr><td>{subject}</td><td>{grade}</td></tr>'  
  
 response += '</table><br>'  
 response += '<a href="/">Go back</a>'  
 response += '</body></html>'  
 return response  
  
 elif method == 'GET' and path == '/':  
 # Return form page  
 response = 'HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n'  
 response += '<html><body>'  
 response += '<h1>Submit Grade Form</h1>'  
 response += '<form method="POST" action="/submit">'  
 response += 'Subject: <input type="text" name="subject" required><br>'  
 response += 'Grade: <input type="text" name="grade" required><br>'  
 response += '<input type="submit" value="Submit">'  
 response += '</form></body></html>'  
 return response  
  
 else:  
 # Handle other requests  
 response = 'HTTP/1.1 404 NOT FOUND\r\nContent-Type: text/html\r\n\r\n'  
 response += '<html><body><h1>Page Not Found</h1></body></html>'  
 return response  
  
def start\_server():  
 server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
 server\_socket.bind(('127.0.0.1', 8080))  
 server\_socket.listen(5)  
  
 print("Server is running, access it at http://127.0.0.1:8080/ ")  
  
 while True:  
 client\_socket, address = server\_socket.accept()  
 print(f'Connection from {address}')  
  
 request = client\_socket.recv(1024).decode('utf-8')  
 print(f'Received request:\n{request}')  
  
 response = handle\_request(request)  
 client\_socket.sendall(response.encode('utf-8'))  
  
 client\_socket.close()  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 start\_server()



