**BÀI TẬP 4**



**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**







**NHẬP MÔN MẠNG MÁY TÍNH**

**LỚP: IT005.O118**

**Nhóm: CYBER SQUAD**

**Học ít hiểu nhiều**



**MỤC LỤC**

[Bảng đánh giá thành viên. 2](#_Toc10183)

[Câu 1. Vẽ bảng chú giải (STT, Query/Answer) của truy vấn đệ quy DNS trong slide 72. 3](#_Toc15014)

[Câu 2: Xây dựng chương trình : Chat console theo mô hình client - server dùng python, dùng TCP và UDP. 4](#_Toc7894)

[a. Mô hình client – server dùng TCP: 4](#_Toc12107)

[\* File server.py: 4](#_Toc19596)

[\* File client.py: 5](#_Toc14301)

[\* Kết quả chạy được: 7](#_Toc24545)

[- Sau khi chạy file server.py thì server đã được khởi động. 7](#_Toc20576)

[b. Mô hình client – server dùng UDP: 7](#_Toc4591)

[\* File server.py: 7](#_Toc28169)

[\* File client.py: 9](#_Toc10648)

[\* Kết quả chạy được: 11](#_Toc12489)

[Kinh nghiệm rút ra từ bài báo cáo 11](#_Toc12621)

[Nguồn tham khảo: 12](#_Toc25190)

**Bảng đánh giá thành viên.**

|  |  |  |  |
| --- | --- | --- | --- |
| **MSSV** | **Tên thành viên** | **Phân chia công việc** | **Đánh giá** |
| 22521060 | Lê Minh Nhựt(C) | Câu 2: UDP, chỉnh sửa | 100% |
| 22520195 | Trần Đình Khánh Đăng | Câu 2: TCP, chỉnh sửa | 100% |
| 22521189 | Thái Ngọc Quân | Câu 1 | 100% |
| 22521078 | Bùi Nhật Phi | Câu 2: UDP | 100% |
| 22520127 | Võ Ngọc Bảo | Câu 2: TCP | 100% |

**Câu 1. Vẽ bảng chú giải (STT, Query/Answer) của truy vấn đệ quy DNS trong slide 72.**

Máy tại engineering.nyu.edu muốn có địa chỉ IP cho gaia.cs.umass.edu.

A diagram of a computer network

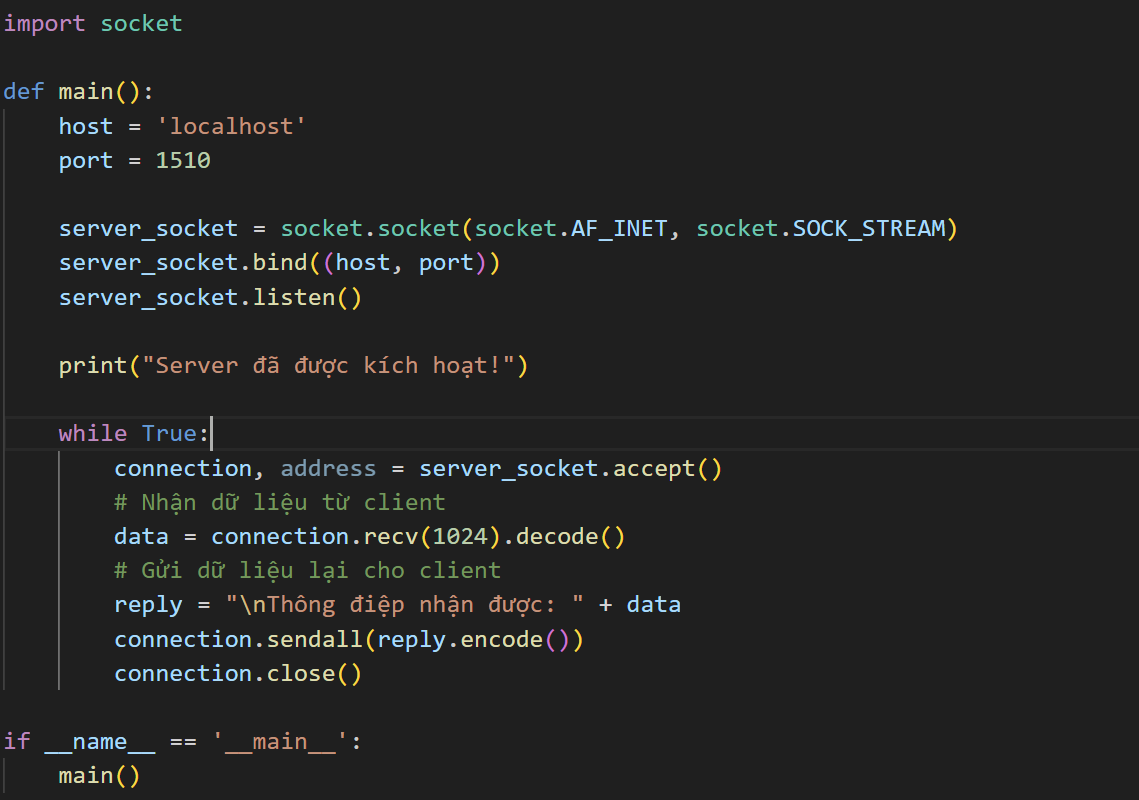
Description automatically generated

|  |  |
| --- | --- |
| **STT** | **Query/Answer** |
| 1 | **Q**: who is gaia.cs.umass.edu? |
| 2 | **Q**: who is gaia.cs.umass.edu? |
| 3 | **Q**: who is gaia.cs.umass.edu? |
| 4 | **Q**: who is gaia.cs.umass.edu? |
| 5 | **A**: 128.119.245.12 |
| 6 | **A**: 128.119.245.12 |
| 7 | **A**: 128.119.245.12 |
| 8 | **A**: 128.119.245.12 |

**Câu 2: Xây dựng chương trình : Chat console theo mô hình client - server dùng python, dùng TCP và UDP.**

1. **Mô hình client – server dùng TCP:**

**\* File server.py:**

 - **import socket**: Dòng code này thêm thư viện socket vào chương trình. Thư viện socket cung cấp các hàm và lớp cần thiết để tạo và sử dụng các kết nối mạng.

- **host**: Biết host lưu trữ địa chỉ IP của máy chủ.

- **port**: Biến này lưu trữ số cổng của máy chủ.

- **server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)**: Sử dụng hàm **socket()** để tạo một socket mới. Cú pháp của hàm **socket()** như sau:

*socket(family, type, proto)*

Trong trường hợp này, chúng ta sử dụng các tham số sau:

* **family:** **socket.AF\_INET** để chỉ định rằng socket sử dụng địa chỉ **IPv4**.
* **type:** **socket.SOCK\_DGRAM** để chỉ định rằng socket là loại **TCP/IP**.
* **proto:** Không sử dụng.

- **server\_socket.bind((host, port))**: sử dụng hàm **bind()** để liên kết socket với địa chỉ server. Cú pháp của hàm **bind()** như sau:

*bind(address)*

- **server\_socket**.**listen()**: Dòng này đặt socket vào trạng thái lắng nghe.

- **while True**: tạo một vòng lặp vô hạn, trong đó client sẽ liên tục nhập và gửi tin nhắn đến máy chủ cho đến khi được tắt.

- **connection, address** = **server\_socket**.**accept()**: Dòng này chấp nhận kết nối từ một khách hàng, Hàm **accept()** trả về một đối tượng socket đại diện cho kết nối với khách hàng và địa chỉ IP của khách hàng.

- **data = connection.recv(1024).decode():** được sử dụng để nhận dữ liệu từ một kết nối mạng và giải mã nó từ một chuỗi byte thành một chuỗi.

-> Đoạn mã này liên tục nhận dữ liệu từ khách hàng. Nếu không có dữ liệu nào được nhận, vòng lặp sẽ thoát.

- **reply = “Thông điệp nhận được:’’ + data**

**connection.sendall(reply.encode())**

**connection.close()**

-> Dòng này tạo một phản hồi cho khách hàng. Phản hồi có nội dung **“Thông điệp nhận được: ’’** theo sau là dữ liệu mà khách hàng đã gửi.

- **if \_\_name\_\_ == "\_\_main\_\_":**

**main()**

Chỉ định rằng hàm **main()** là hàm chính của chương trình. Khi chương trình được chạy, hàm **main()** sẽ được thực thi đầu tiên.

**\* File client.py:**



- **import socket**: Dòng code này thêm thư viện socket vào chương trình. Thư viện socket cung cấp các hàm và lớp cần thiết để tạo và sử dụng các kết nối mạng.

**client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)**: dùng hàm **socket()** để tạo một socket **TCP** mới và địa chỉ là **IPv4**.

- **client\_socket.connect((host, port))**: Dòng này kết nối socket với máy chủ TCP. Đối số của hàm **connect()** là địa chỉ IP và số cổng của máy chủ.

- **message = input(‘Nhập thông điệp của bạn: ’)**

**client\_socket.send(message.encode())**

**data = client\_socket.recv(1024).decode()**

**print(‘Phản hồi từ server: ’, data)**

**client\_socket.close()**

-> Đoạn mã này gửi và nhận dữ liệu từ máy chủ. Người dùng sẽ được cập nhật một thông điệp và chương trình sẽ gửi thông điệp đó đến máy chủ. Chương trình sau đó sẽ nhận phản hồi từ máy chủ và in nó ra màn hình.

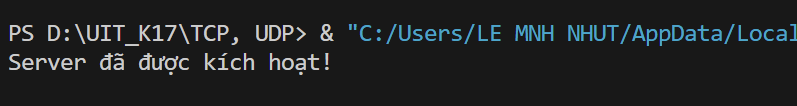
- **if \_\_name\_\_ == "\_\_main\_\_"**:

**main()**

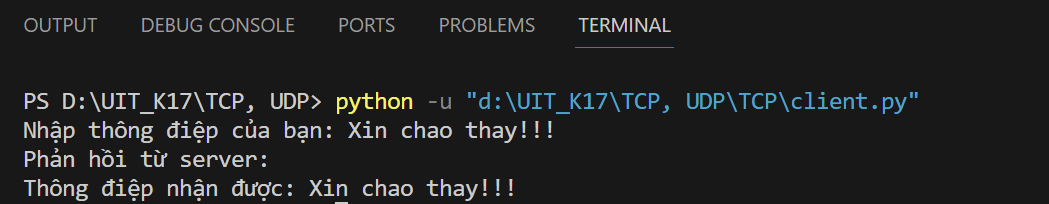
Chỉ định rằng hàm **main()** là hàm chính của chương trình. Khi chương trình được chạy, hàm **main()** sẽ được thực thi đầu tiên.

**\* Kết quả chạy được:**

- Sau khi chạy file server.py thì server đã được khởi động.

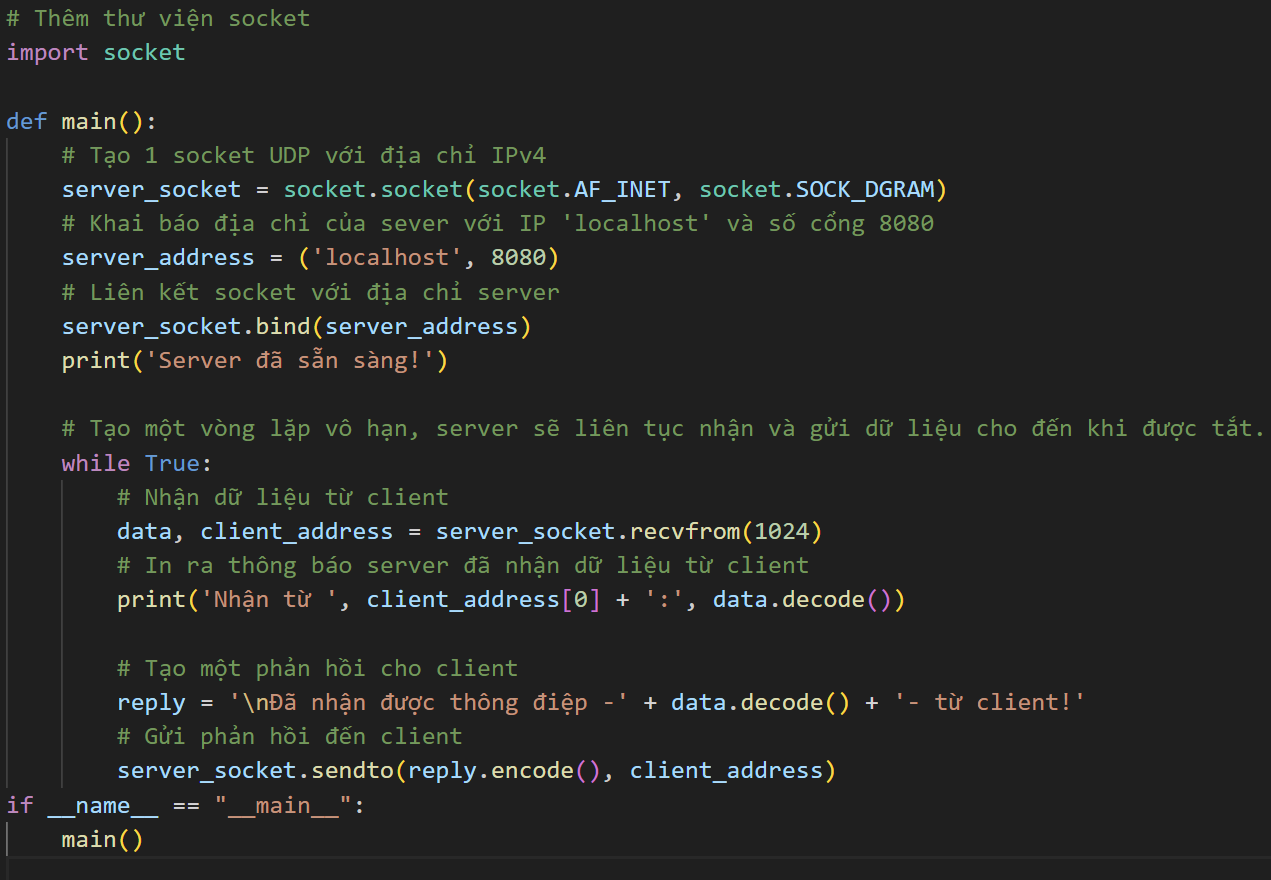


- Client nhập thông điệp và nhận được phản hồi từ server.



1. **Mô hình client – server dùng UDP:**

**\* File server.py:**

****

- **import socket**: Dòng code này thêm thư viện socket vào chương trình. Thư viện socket cung cấp các hàm và lớp cần thiết để tạo và sử dụng các kết nối mạng.

- **server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)**: Sử dụng hàm **socket()** để tạo một socket mới. Cú pháp của hàm **socket()** như sau:

*socket(family, type, proto)*

Trong trường hợp này, chúng ta sử dụng các tham số sau:

* **family:** socket.AF\_INET để chỉ định rằng socket sử dụng địa chỉ **IPv4**.
* **type:** socket.SOCK\_DGRAM để chỉ định rằng socket là loại **UDP**.
* **proto:** Không sử dụng.

- **server\_address = ('localhost', 8080)**: khai báo địa chỉ của server với IP localhost và số cổng 8080.

- **server\_socket.bind(server\_address)**: sử dụng hàm **bind()** để liên kết socket với địa chỉ server. Cú pháp của hàm **bind()** như sau:

*bind(address)*

Trong trường hợp này, chúng ta sử dụng tham số address là một tuple chứa địa chỉ IP và số cổng của server.

- **while True**: Dòng code này tạo một vòng lặp vô hạn, trong đó server sẽ liên tục nhận và gửi dữ liệu cho đến khi được tắt.

- **data, client\_address = server\_socket.recvfrom(1024)**: sử dụng tham số bufsize là 1024 để chỉ định kích thước của bộ đệm nhận dữ liệu.

- **print('Nhận từ ', client\_address[0] + ':', data.decode())**: lưu trữ địa chỉ của client trong biến client\_address

- **reply = '\nĐã nhận được thông điệp -' + data.decode() + '- từ client!'**: in ra thông báo cho biết server đã nhận dữ liệu từ client

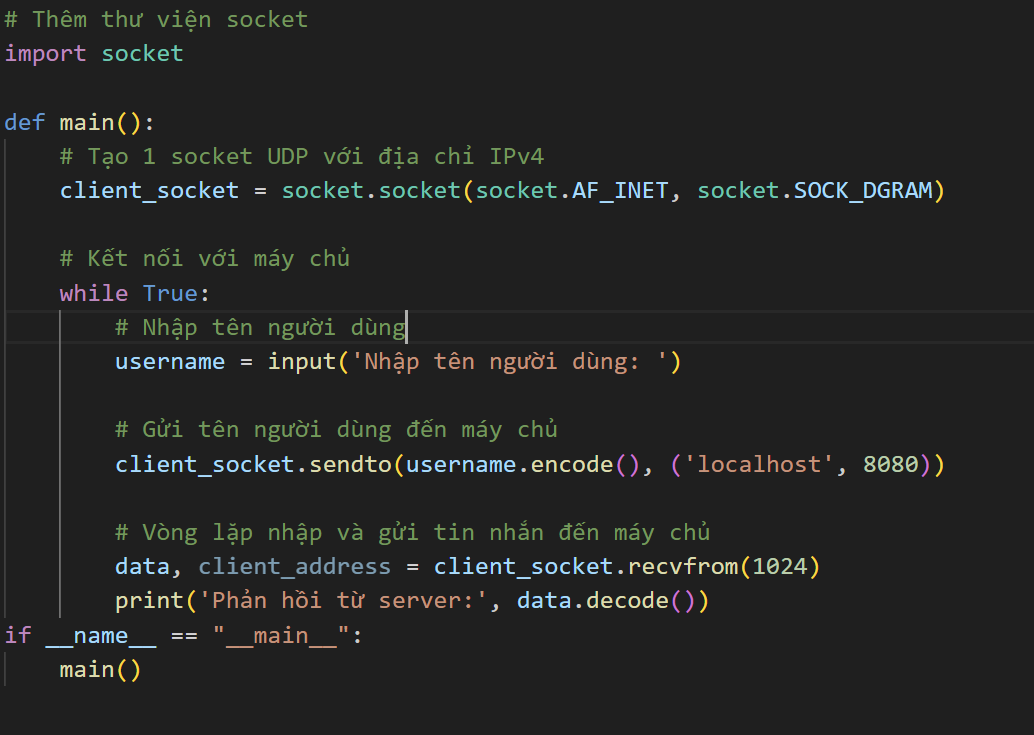
- **server\_socket.sendto(reply.encode(), client\_address)**: chúng ta sử dụng hàm **sendto()** với tham số data là phản hồi đã tạo và tham số address là địa chỉ của client.

- **if \_\_name\_\_ == "\_\_main\_\_":**

**main()**

Chỉ định rằng hàm **main()** là hàm chính của chương trình. Khi chương trình được chạy, hàm **main()** sẽ được thực thi đầu tiên.

**\* File client.py:**



- **import socket**: Dòng code này thêm thư viện socket vào chương trình. Thư viện socket cung cấp các hàm và lớp cần thiết để tạo và sử dụng các kết nối mạng.

**client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)**: dùng hàm **socket()** để tạo một socket **UDP** mới và địa chỉ là **IPv4**.

- **while True**: tạo một vòng lặp vô hạn, trong đó client sẽ liên tục nhập và gửi tin nhắn đến máy chủ cho đến khi được tắt.

- **username = input('Nhập tên người dùng: ')**: nhập tên người dùng từ bàn phím.

- **client\_socket.sendto(username.encode(), ('localhost', 8080))**: sử dụng hàm **sendto()** với tham số data là tên người dùng đã được mã hóa thành dạng nhị phân và tham số address là một tuple chứa địa chỉ IP và số cổng của máy chủ.

- **data, client\_address = client\_socket.recvfrom(1024)**: sử dụng tham số bufsize là 1024 để chỉ định kích thước của bộ đệm nhận dữ liệu.

- **print('Phản hồi từ server:', data.decode())**: in ra phản hồi từ máy chủ.

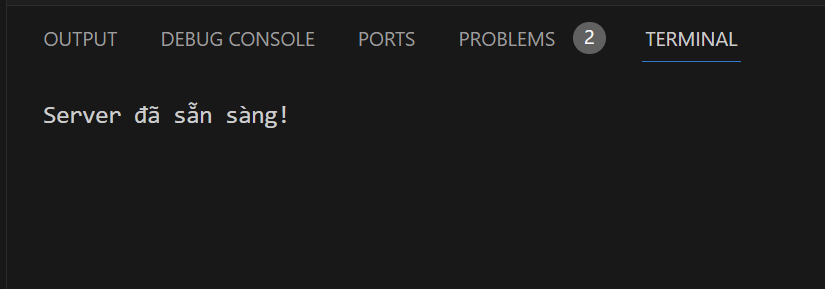
- **if \_\_name\_\_ == "\_\_main\_\_"**:

**main()**

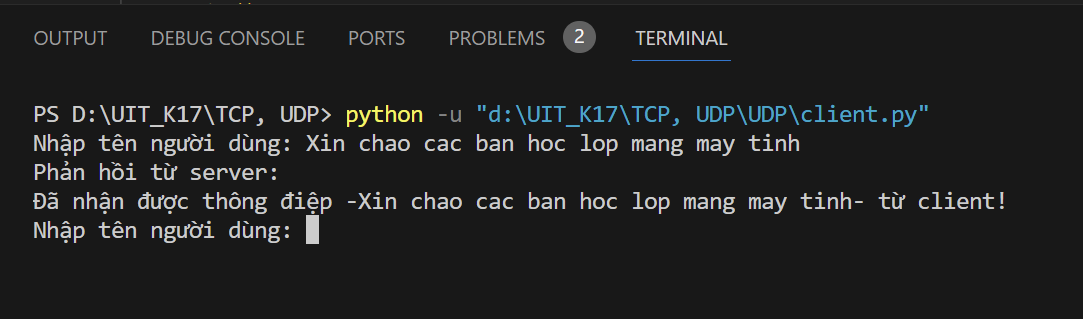
Chỉ định rằng hàm **main()** là hàm chính của chương trình. Khi chương trình được chạy, hàm **main()** sẽ được thực thi đầu tiên.

**\* Kết quả chạy được:**

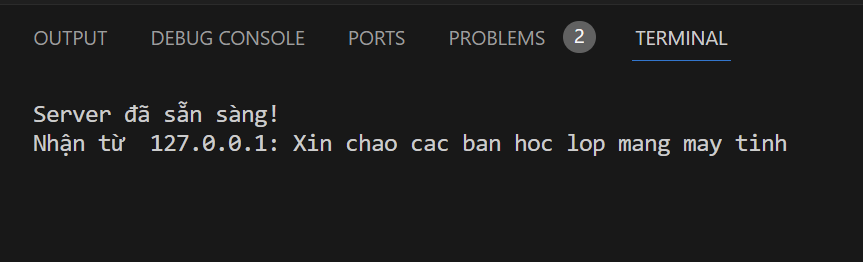
- Chạy file server.py, ta thấy server được kích hoạt:

****

- Người dùng gửi thông điệp và được server phản hồi lại:

****

- Server trả về địa chỉ IP và thông điệp:

****

**Kinh nghiệm rút ra từ bài báo cáo**

- Hiểu được cách hoạt động của truy vấn đệ quy.

- Hiểu rõ hơn cách thức hoạt động của các giao thức TCP, UCP.

- Biết được các thao tác cơ bản xây dựng mô hình client - server theo TCP bằng python.

- Biết được các thao tác cơ bản xây dựng mô hình client - server theo UDP bằng python.

**Nguồn tham khảo:**

* [Tạo ứng dụng trò chuyện bằng giao thức UDP với python (ichi.pro)](https://ichi.pro/vi/tao-ung-dung-tro-chuyen-bang-giao-thuc-udp-voi-python-104606114675664)[Bài 12: Kết nối bền vững và không bền vững trong HTTP (sinhviencntt.net)](https://sinhviencntt.net/ket-noi-ben-vung-va-khong-ben-vung-http-1013)
* [Lập Trình Socket Với TCP/IP Trong Python (codelearn.io)](https://codelearn.io/sharing/lap-trinh-socket-voi-tcpip-trong-python)[So sánh sự khác biệt giữa POP3 và IMAP (funix.edu.vn)](https://funix.edu.vn/chia-se-kien-thuc/so-sanh-su-khac-biet-giua-pop3-va-imap/)
* [How To Build Python Chat Using UDP And TCP Protocol | by Gupta Aditya | Medium](https://gupta-aditya333.medium.com/how-to-build-python-chat-using-udp-protocol-9366722d9af)