채팅 클라이언트 프로그램 명세서 (오픈채팅방)

목차

---

1. 프로그램 설명

1.1 프로그램 개요

이 프로그램은 카카오톡 오픈채팅방을 모티브로 만들어졌으며, 최대 10명까지 채팅할 수 있는 단체 채팅방입니다. 서버-클라이언트 구조를 기반으로 하며, 서버는 클라이언트들의 연결과 메시지 전달을 관리하고, 클라이언트는 서버에 연결하여 메시지를 보내고 받을 수 있습니다.

1.2 프로그램 흐름

1. **서버 시작:** 서버 프로그램을 실행하여 클라이언트의 연결을 대기합니다.
2. **클라이언트 연결:** 사용자는 클라이언트 프로그램을 실행하고 서버에 연결합니다.
3. **닉네임 설정:** 클라이언트는 채팅에서 사용할 닉네임을 입력합니다.

4. **채팅 참여:** 사용자는 메시지를 보내고 다른 사용자로부터 메시지를 수신합니다.
5. **명령어 사용:** 특정 기능(예: 닉네임 변경, 사용자 목록 조회 등)을 위해 명령어를 사용할 수 있습니다.
6. **채팅 종료:** 사용자가 프로그램을 종료하거나 서버에서 강퇴될 수 있습니다.

---

2. 서버, 클라이언트, 시나리오

- 서버(server.py) 역할

1. 클라이언트 관리

- 새로운 클라이언트 연결 처리.
- 닉네임 중복 확인 및 클라이언트 추가.
- 클라이언트 연결 해제 시 그룹에서 제거.

2. 메시지 처리

- 일반 메시지 브로드캐스트.
- 특수 명령어 처리

3. 로그 기록

- 메시지 및 명령 실행 내용을 로그 파일에 기록.

-클라이언트(client.py) 역할

- 서버 연결 후 닉네임 설정.
- 메시지 입력 및 전송.
- 특수 명령어 실행:
  - `/rename [new_nickname]`: 닉네임 변경.
  - `/whisper [nickname] [message]`: 특정 사용자에게 귓속말 전송.
  - `/schedule [HH:MM] [message]`: 특정 시간에 메시지 예약.

- 수신된 메시지를 콘솔에 출력

-시나리오

- **일반 채팅**:
  - 클라이언트 Alice, Bob, Charlie가 서버에 연결.
  - Alice가 `Hello everyone!` 메시지를 입력.
  - 서버는 Bob과 Charlie에게 메시지를 전달:

    ```
    Alice >> Hello everyone!
    ```

- **닉네임 변경**
  - Alice가 `/rename Alice123` 입력.
  - 서버는 닉네임을 변경하고 그룹에 알림:

    ```
    [Server]: User "Alice" has changed their nickname to
    "Alice123".
    ```

- **귓속말**:
  - Bob이 `/whisper Charlie Hi, Charlie!` 입력.
  - 서버는 Charlie에게만 메시지를 전달:

    ```
    [Whisper from Bob] Hi, Charlie!
    ```

- **예약 메시지**:
  - Charlie가 `/schedule 15:30 Meeting starts soon` 입력.
  - 서버는 15:30이 되면 모든 클라이언트에게 메시지를 전송:

    ```
    [Scheduled by Charlie] Meeting starts soon
    ```

- **강퇴**:
  - 관리자(클라이언트1)가 `/kick Bob` 입력.
  - 서버는 Bob을 그룹에서 제거하고 알림:

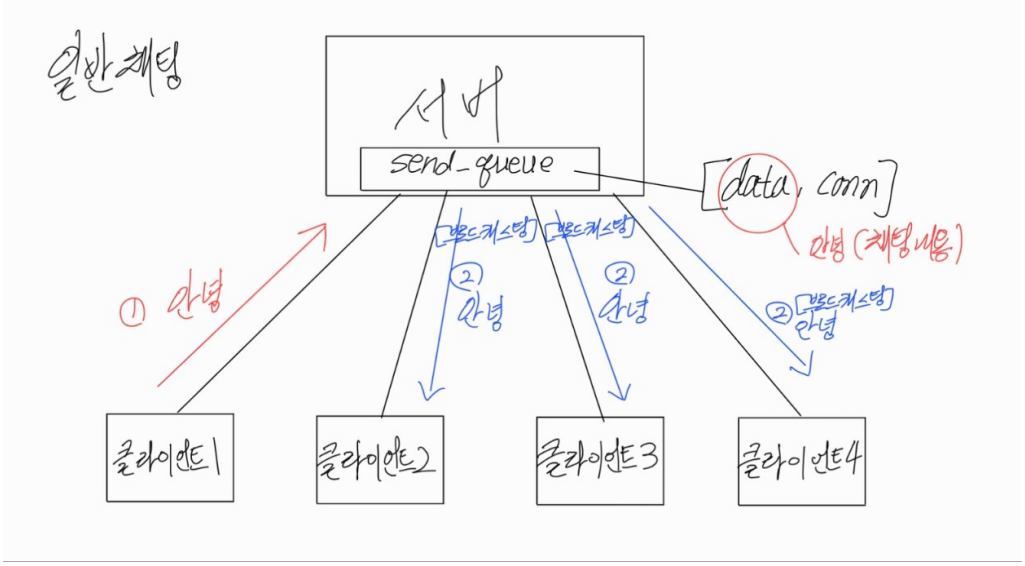
```
[Server]: Bob has been kicked from the chat.
```

---

3. 주요 기능

3.1 send_queue사용(일반 채팅, 귓속말, 강퇴 기능)

- **다중 클라이언트 채팅**: 여러 사용자가 동일한 채팅방에서 대화 가능.

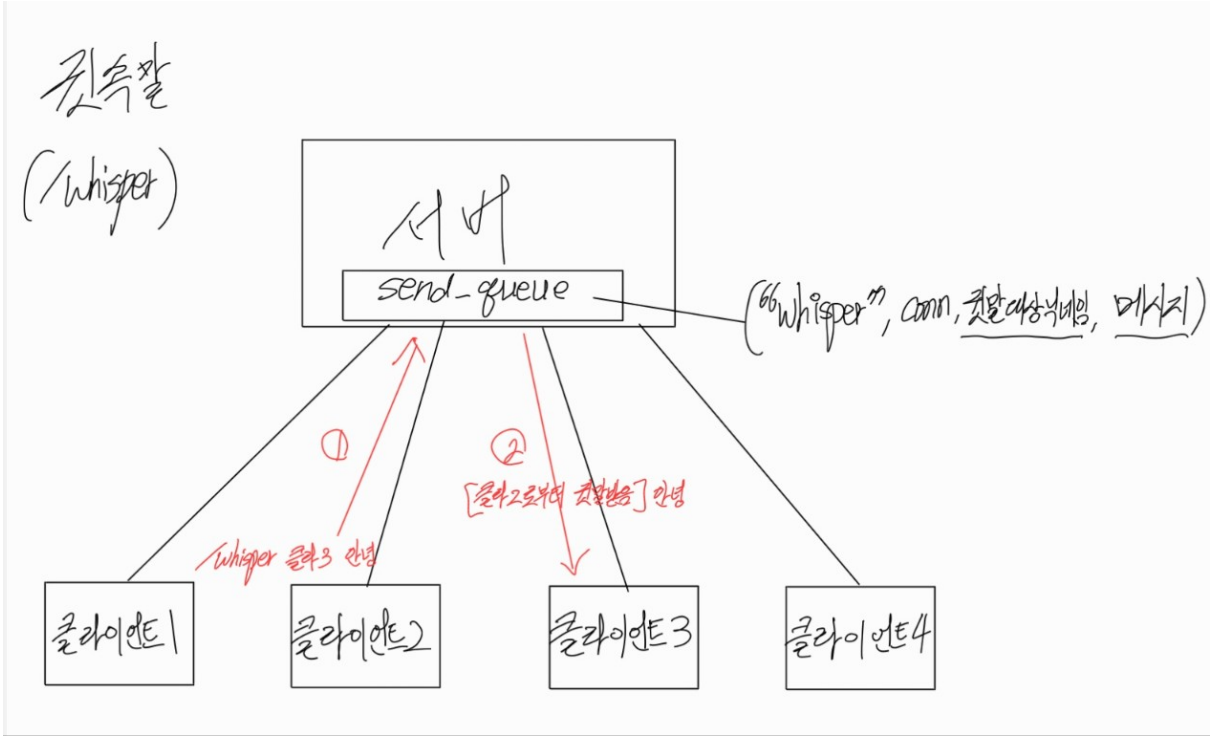


- **설명**) 일반 채팅의 경우는 send_queue에 채팅(data)과 연결(conn)을 저장하고, 송신자를 제외한 모든 클라이언트에게 브로드캐스팅으로 채팅(data)을 뿌려준다.

1. 클라이언트1이 안녕이라는 메시지를 입력한다.
2. 서버는 안녕이라는 메시지를 send_queue의 데이터에 저장한다
3. 메시지(data)를 클라이언트1을 제외한 모든 클라이언트에게 전송한다.

```
for conn in group:
    if conn != message[1]:
        conn.send(f"{sender_nickname} >> {message[0]}".encode
```

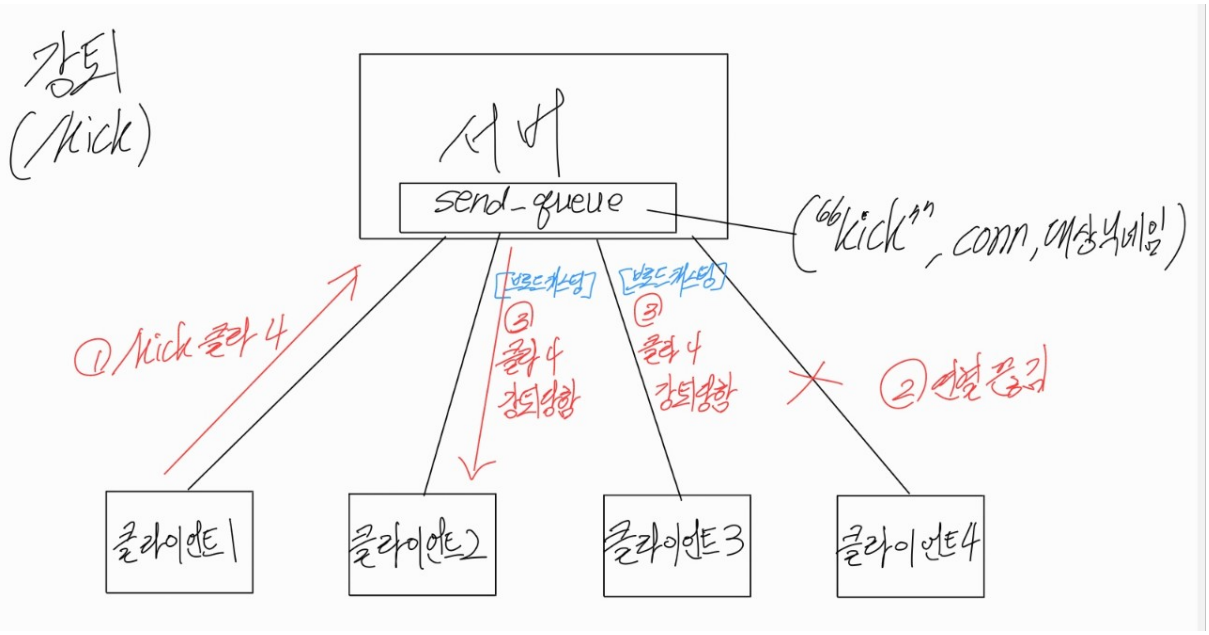
- **귓속말 (/whisper)**: 특정 사용자에게만 비공개 메시지 전송.



- **설명)** 귓속말의 경우 “whisper” 명령어와, 연결(conn), 귓속말 대상 닉네임, 메시지를 send_queue에 저장하고, 대상 클라이언트에게 해당 메시지를 송신한다.

1. 클라이언트2가 클라이언트3에게 안녕이라는 귓속말을 보낸다.
2. 서버는 안녕이라는 메시지와 whisper라는 명령어, 귓속말 대상 닉네임, 연결(conn)을 저장한다.
3. 메시지를 귓속말 대상 클라이언트에게 전송한다.

- **강퇴 기능 (/kick)** : 관리자(클라이언트1)가 특정 사용자를 채팅방에서 강제 퇴장.



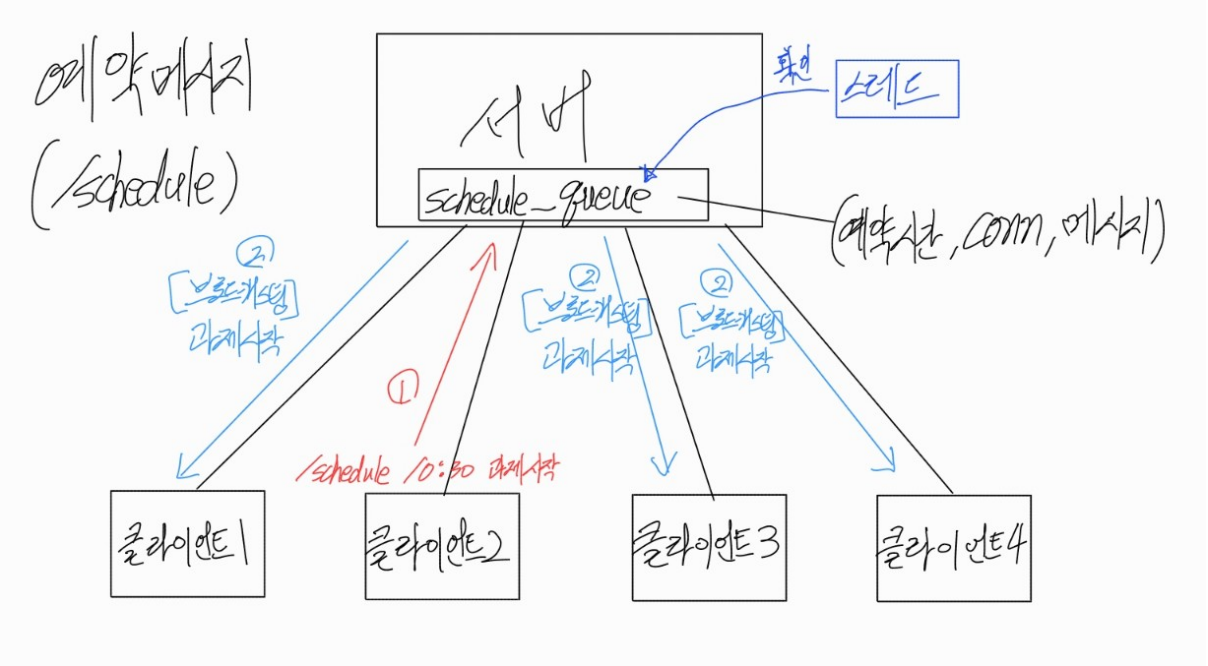
- **설명)** 강퇴의 경우 “kick” 명령어와, 연결(conn), 강퇴 대상 닉네임을 send_queue에 저장한다. 서버가 강퇴 대상 클라이언트의 연결을 끊고, “ ‘닉네임’ has been kicked from the chat.” 메시지를 브로드캐스팅(관리자를 제외한 클라이언트들에게 메시지를 전송)한다.

1. 클라이언트1(관리자)이 클라이언트4를 강퇴하겠다는 명령어(/kick 클라이언트4)를 입력한다.

   서버는 kick명령어와, 연결, 강퇴 대상 닉네임을 send_queue에 저장한다.
2. 서버는 강퇴 대상인 클라이언트4의 연결을 끊는다.
3. 서버가 관리자를 제외한 모든 클라이언트에게 클라이언트4가 추방당했다는 메시지를 전송한다.

3.2 schedule_queue사용(예약 메시지)

- **예약 메시지(/schedule)**: 지정된 시간에 메시지를 자동으로 전송.

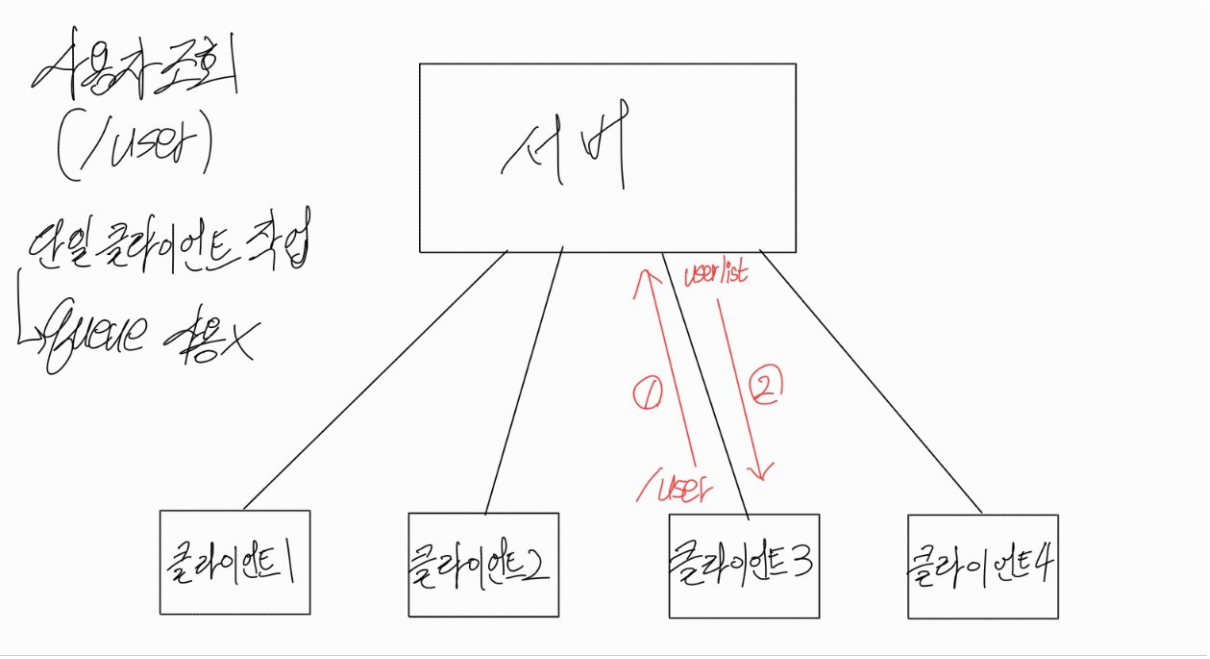


- **설명)** 예약 메시지의 경우 예약 시간, 연결(conn), 메시지를 schedule_queue에 저장한다. handle_scheduled_messages라는 스레드가 1초마다 schedule_queue를 확인하고, 예약된 시간이 되면, 메시지를 일반 채팅처럼 송신자를 제외한 모든 클라이언트에게 전송(브로드캐스팅)한다.

1. 클라이언트2가 '과제 시작'이라는 메시지를 10:30에 전송하도록 예약 메시지를 설정한다. (/schedule 10:30 과제시작)
    a. 서버는 (10:30, conn, 과제 시작) 데이터를 schedule_queue에 저장한다.
    b. handle_scheduled_message 스레드가 schedule_queue를 반복적으로 조회하면서 현재 시간과 일치하는 메시지를 찾아 실행한다.
    c. 현재 시간이 예약된 시간인 10:30분이 되면 handle_scheduled_message 스레드가 예약 메시지를 send_queue에 추가한다.
2. 서버가 예약 메시지를 송신자를 제외한 모든 클라이언트에게 메시지를 전송한다.

3.3 queue 사용 X(사용자 조회, 닉네임 변경)

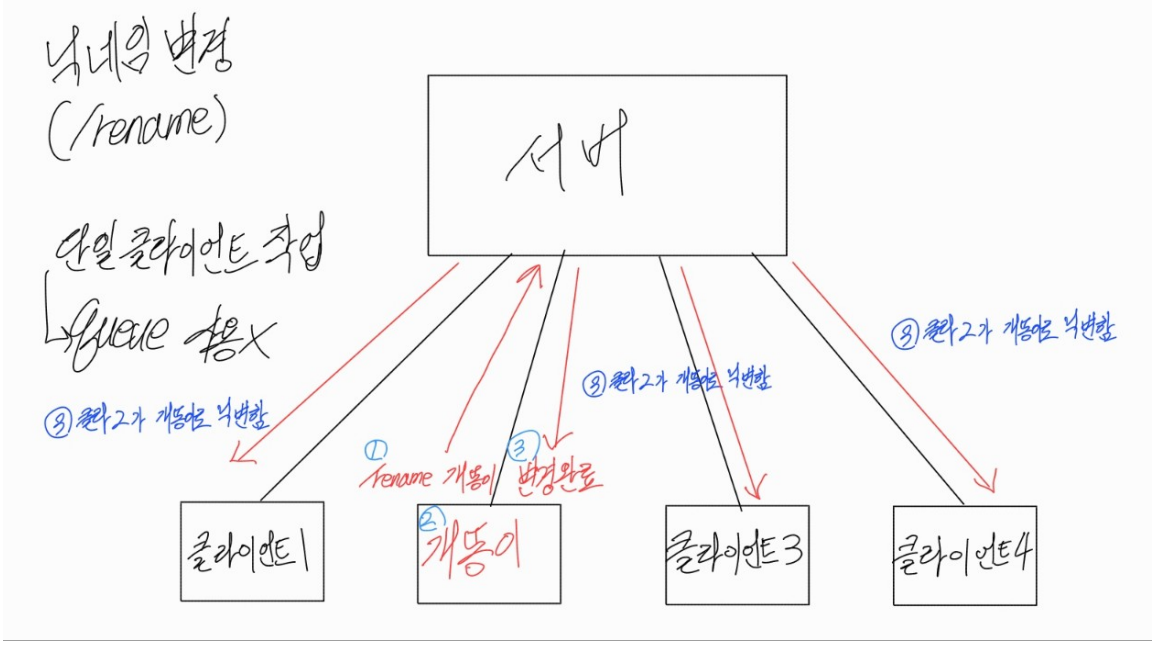
- **사용자 조회 (/user)**: 채팅방에 있는 사용자들 닉네임 조회



- **설명)** 단일 클라이언트 작업이기 때문에 큐를 사용하지 않고, /user라는 명령어를 사용한 클라이언트에게 서버가 userlist를 송신해준다.

1. 클라이언트3이 사용자를 조회하기 위해 /user명령어를 입력한다.
2. 서버는 사용자 목록을 요청한 클라이언트에게 전송한다.

- **닉네임 변경 (/rename)**: 사용자 닉네임 변경



- **설명)** 단일 클라이언트 작업이기 때문에 큐를 사용하지 않고, /rename <변경할 닉네임> 명령어를 사용하면 닉네임이 변경할 닉네임으로 변경이 되고 서버는 닉네임을 변경한 클라이언트에게 변경 완료 문구를 전송한다. 이외 다른 클라이언트들에게는 닉네임이 변경되었다는 메시지가 브로드 캐스팅된다.

1. 클라이언트2가 개똥이로 닉네임을 변경하기 위해 /rename 개똥이 라고 명령어를 입력한다.
2. 개똥이라는 닉네임이 중복되지 않는 경우, 클라이언트2의 닉네임이 개똥이로 바뀌고 변경완료라는 메시지를 서버로부터 전송받는다.
3. 서버는 닉네임을 변경한 클라이언트를 제외한 모든 클라이언트들에게 클라이언트2가 개똥이로 닉네임 변경했다는 메시지를 전송한다.

3.4 부가 기능

- **로그 기록**:
  - 서버: 채팅, 명령 실행 내역을 `server.txt`에 기록.
  - 클라이언트: 개인 채팅 내용을 `[nickname]_chat_log.txt`에 기록.

---

4. 소스코드 컴파일 방법

4.1 외부서버 (GCP 사용)

1. 구글 클라우드에 접속하여 VM instance를 생성한다.
   지역 : us-central1로 설정
   머신 유형 : e2-micro
   부팅 디스크 : Debian
2. 방화벽 규칙을 추가한다
   대상 : 모든 인스턴스 선택
   소스 IP 범위 : 0.0.0.0/0 (모든 IP 주소 허용)
   프로토콜 및 포트 : TCP와 해당 포트를 지정 (port : 9999)
3. 생성된 인스턴스의 SSH를 실행한다.
4. Python과 개발 도구의 패키지들을 설치한다 (Debian 기준)
   sudo apt update
   sudo apt install python3

sudo apt install python3-pip
pip install numpy
pip install numpy scipy
pip install loguru

5. GCP 가상환경을 생성하고 활성화한다.
   python3 -m venv myenv(가상환경 이름)
   source myenv/bin/activate //가상환경 활성화
6. UPLOAD FILE을 클릭하여 server.py를 업로드한다.
   server.py가 업로드된 디렉터리에서 python3 server.py로 server를 실행한다.
7. 로컬에서 powershell 터미널(10개이하)를 열어 python3 client.py로 client를 실행한다. (vscode 터미널에서 실행해도 됨)
8. server에 client가 연결되면 프로그램이 실행된다.
9. 별명을 설정하고 채팅방에 합류한다.

★ client.py에서 host의 IP가 자신이 사용하는 외부 서버 IP인지 확인한다

4.2 로컬에서 실행

1. VScode로 server.py와 client.py를 연다.
2. client.py의 host의 IP가 로컬(127.0.0.1)로 설정되어 있는지 확인한다
3. server.py를 파이썬 디버거 혹은 터미널로 실행한다.
4. client.py를 파이썬 디버거나 터미널로 실행한다. (10개 이하)
5. 별명을 설정하고 채팅방에 합류한다.