**Computer Network Project 1**



학과 전자공학과

학번 12181457

이름 김범수

제출일 23.04.29

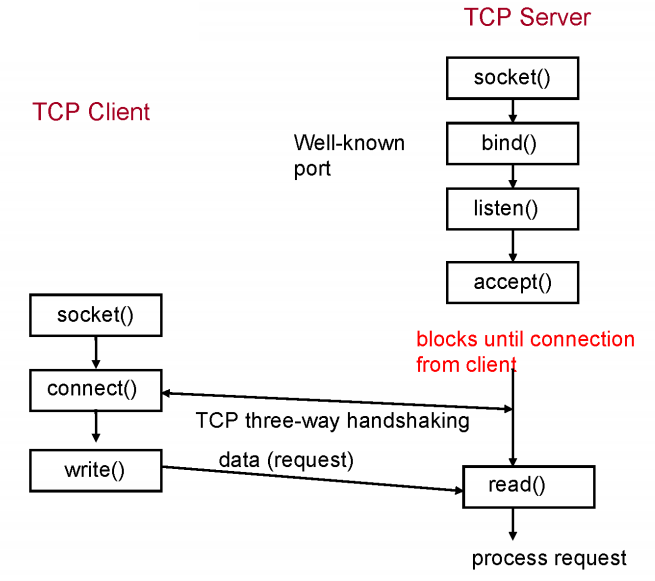
**구현 철학**

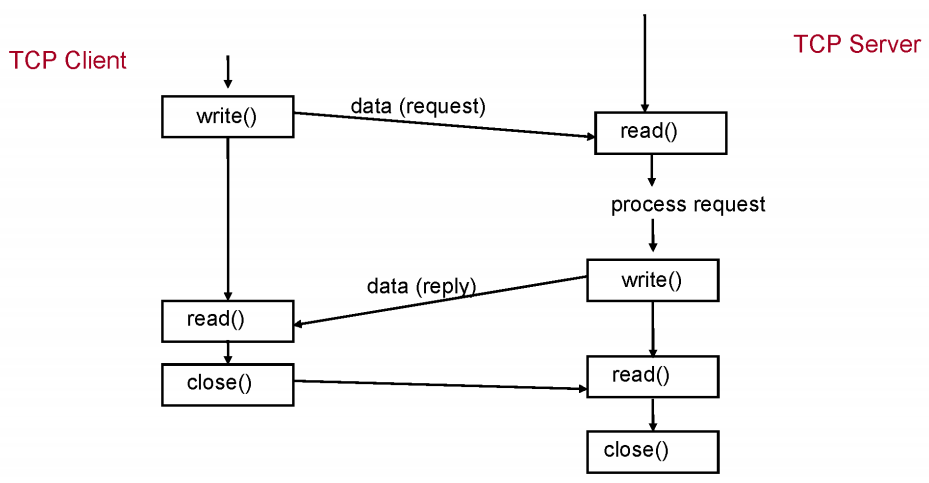
Python 프로그래밍 언어를 이용하여 TCP 소켓 기반 Server/Client를 구현하는 것을 목표로 하였다.

Client의 요청에 따라 서버가 key-value 기반의 데이터 저장소 기능을 제공할 수 있도록 구현하였으며 Server는 항상 OFF 되지 않고 Client의 요청을 수용할 수 있도록 코딩하였다.

코드를 구현 함에 앞서 주어진 동작을 수행할 수 있는 가장 간단한 코드를 구현하는 것을 목표로 하였다.

사용자가 사용하기 쉽도록 하였고 사용자의 실수로 인하여 발생할 수 있는 대부분의 상황에서 오류가 발생하지 않도록 예외 Case 설정하여 오류를 방지하였다.





코드의 구현은 위와 같은 구조를 기반으로 하였다.

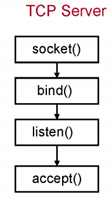
Server는 socket, bind, listen, accept를 통해 Client를 기다리도록 설정하였다.

이후 Client가 socket, connect를 통해 Server에 연결하고 주어진 명령어인 PUT, GET, DELETE, LIST를 수행할 수 있는 간단한 코드를 구현하였으며 다른 명령어가 들어오면 다시 입력하도록 유도하였다.

각각의 상황에 맞는 간단한 코드를 구현하였으며 입력을 받았을 때 해당 상황을 출력하여 정상적으로 동작하는 것을 확인할 수 있도록 하였다.

**코드 설명**

1. **Server**

텍스트이(가) 표시된 사진

자동 생성된 설명

먼저 Server의 코드는 socket을 import하는 것으로 시작하였다.

IP는 Local Host의 IP 주소인 127.0.0.1로 설정하였고 PORT는 1234로 설정하였다.

127.0.0.1은 Local Host IP로 동일 기기간의 통신을 구성하고 Packet을 보내고 수신할 수 있도록 할 수 있는 IP이다.

이후 socket를 생성하였다.

인자는 IPv4, TCP를 사용하도록 설정하였다.

Bind된 Socket을 생성하기 위한 Socket이다.

이후 server\_sckt.setsockopt(socket.SOL\_SOCKET, socket.SO\_REUSEADDR, 1)을 통해 오류가 발생하지 않도록 설정하였다.

소켓을 이용한 서버 프로그램을 운용하다가 비정상 종료되는 경우 bind error가 발생할 수 있다.

위와 같은 문제는 기존 프로그램이 종료되어도 커널이 bind 정보를 유지하고 있음으로 발생하는 문제이다.

위 코드를 사용하면 위와 같은 문제가 발생하지 않도록 할 수 있다.

이후 bind를 진행한다.

Server의 IP와 PORT를 넣어 bind를 진행한다.

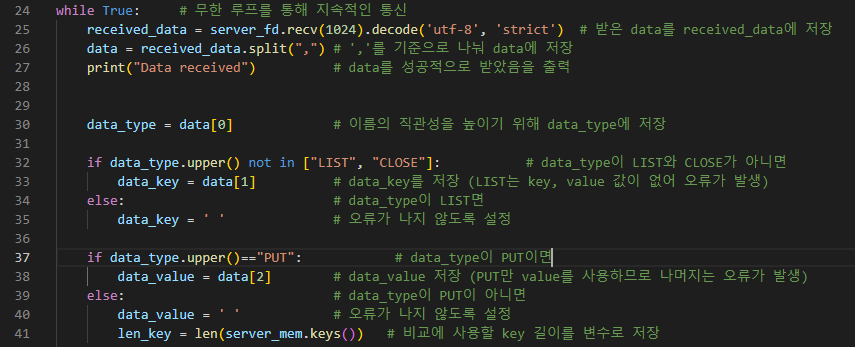
이는 Client가 Server에 연결할 때 사용될 Socket의 주소를 설정하는 것이다,

5의 대기열을 설정하여 listen을 하고 accept를 통해 Client의 connect를 받아들이며 새로운 소켓을 생성해 server\_fd 변수에 할당되고 client\_addr에 Client의 주소가 할당 된다.

이후 Client의 주소를 print한다.



Data를 저장할 memory를 설정하였다.



이후 while문을 통해 항상 Server가 동작할 수 있도록 설정하고 Client에서 받은 data를 저장한다.

이때 1024의 buffer 크기로 설정하였다.

받은 Data를 received\_data에 저장하고 split을 이용하여 ‘,’를 기준으로 data를 나눠 data에 저장한다.

Data를 성공적으로 받았음을 출력하고 이름의 직관성을 높이기 위해 data를 새로운 변수로 저장한다.

새로운 data는 type, key, value로 구분하여 저장하였다.

LIST와 CLOSE는 type만을 가지고 나머지 key와 value 값은 입력 받지 않는다.

이때 data[1]과 data[2]를 사용하려고 하면 오류가 발생하기 때문에 이 문제를 방지하기 위해 if문으로 오류가 나지 않도록 data[1]을 사용하지 않도록 설정하였다.

Data[2]를 사용하지 않도록 설정하는 부분은 바로 아래에서 설정하였다.

또한 PUT을 제외하면 value를 사용하지 않는다.

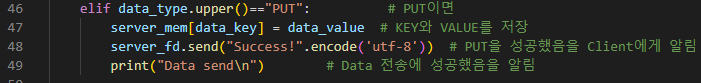
따라서 오류가 나지 않도록 data[2]를 put만 설정하도록 설정하였다.



이후 type값에 따라 원하는 동작을 할 수 있도록 코드를 설정하였다.

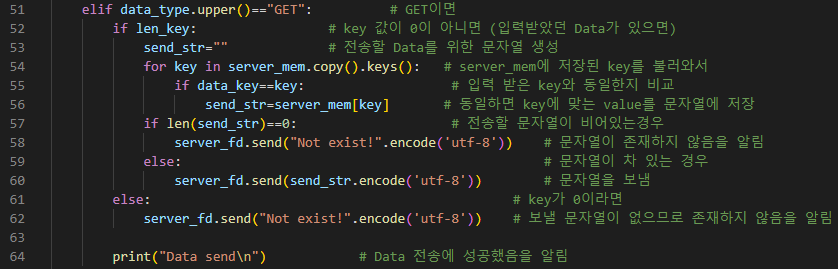
CLOSE가 입력되는 경우 LOOP를 종료한다.

사용자가 소문자를 잘못 입력하는 경우를 방지하기 위해서 upper로 설정하여 비교해 항상 대문자로 비교하도록 설정하였다.



다음으로는 PUT이 입력되면 key값에 value를 저장한다.

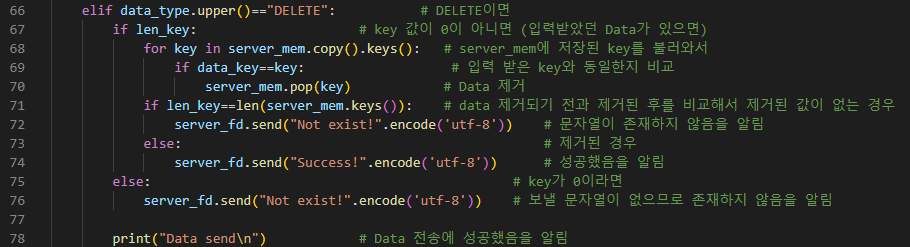
이후 성공적으로 저장했음을 Client에게 알리고 Data 전송에 성공했음을 알린다.



GET이 들어오면 잘못된 key 값을 넣은 것인지 확인하고 key값을 통해 value를 가져오도록 코드를 설정하였다.

전송할 문자열을 생성하고 key값을 비교하여 문자열을 저장해 전송할 수 있도록 설정한다.

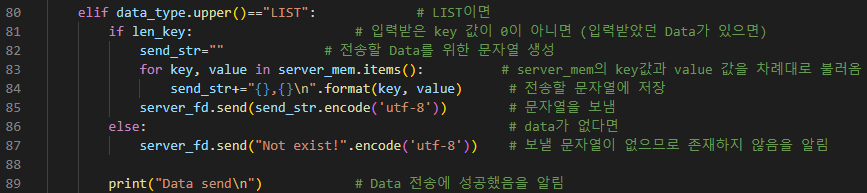
문자열이 존재하지 않는 경우와 key값이 올바르지 않으면 Not exist!을 출력하고 정상적인 경우에는 key 값을 Client에게 보낸다.



DELETE가 들어오면 GET과 유사하게 key 값을 비교한다.

Pop을 통해 Data를 제거하고 성공적으로 제거되었는지 if문을 통해 확인한다.

문자열이 원래 존재하지 않았던 경우와 key값이 잘못된 경우 Not exist!를 출력하고 정상적인 경우에는 key 값을 Client에게 보낸다.



LIST가 들어오면 key값과 value를 불러온다.

잘못된 경우 Not exist!를 출력하고 정상적인 경우 Data를 Client에게 전송한다.



이 코드는 잘못된 type 값을 입력 받은 경우 잘못된 입력임을 출력하고 다시 입력하도록 유도한다.



마지막으로 CLOSE를 입력 받은 경우 SOCKET을 CLOSE 하는 것으로 마무리한다.

1. **Client**



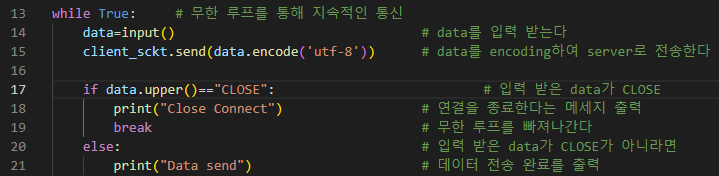
Client의 코드는 Server와 유사하게 설정하였다.

동일하게 Socket을 설정하여 socket을 생성한 후 Server의 IP와 PORT를 인자로 Client와 connect를 진행하였다.

이때 server의 IP와 PORT를 사용하여 server의 주소를 특정하였다.

하지만 Client는 IP와 PORT를 특정하여 제공하지 않았다.

이는 Client가 server에 요청하는 형식이고 Client는 남는 Port 중 아무거나 사용하여 연결하면 되기 때문에 이처럼 코드를 구현하였다.



무한 Loop를 구성하여 close되기 전까지 data를 전송할 수 있도록 하였다.

Input을 받아 data에 저장하고 encoding을 진행하여 server로 전송하였다.

만약 close가 들어온 경우라면 connect를 close하고 무한 loop를 나오고 close가 아니라면 data를 성공적으로 전송했다는 메시지를 출력한다.



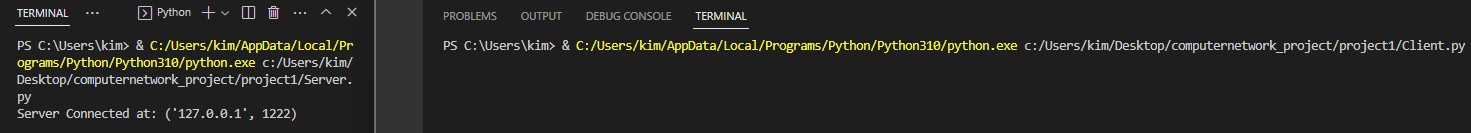
서버에서 받은 data를 저장하고 출력한다.



Close를 통해 무한 loop를 빠져나온 경우 socket을 종료한다.

1. **Start**

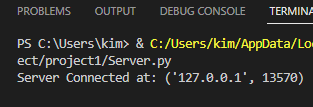
**실행 결과 및 분석**



Client와 Server를 연결한 사진이다.

Client의 Port 번호는 1222로 랜덤한 Port 번호가 할당된 것을 확인할 수 있었다.

이는 Client가 server에 요청하는 것이기 때문에 랜덤한 port가 지정되기 때문이다.



위 그림처럼 프로그램을 실행할 때마다 다른 port가 할당되는 것을 확인할 수 있다.

1. **Data가 없을 때**

텍스트이(가) 표시된 사진

자동 생성된 설명

Server의 server\_mem가 비어 있는 경우 GET, DELETE, LIST를 요구한 경우이다.

아무 data도 저장되어 있지 않기 때문에 Not exist!를 출력한다.

1. **PUT**

|  |  |
| --- | --- |
| KEY | VALUE |
| NAME | KBS |
| AGE | 24 |
| ADDRESS | INCHEON |

텍스트, 스크린샷, 화면이(가) 표시된 사진

자동 생성된 설명

PUT을 통해 data를 Server로 전송하고 저장한다.

1. **GET**

|  |  |
| --- | --- |
| KEY | VALUE |
| NAME | KBS |
| AGE | 24 |
| ADDRESS | INCHEON |

텍스트, 스크린샷, 화면이(가) 표시된 사진

자동 생성된 설명

GET을 수행한 결과이다.

PUT을 통해 저장한 Data를 key값을 통해 가져온 경우이다.

정상적으로 Data를 가져오는 것을 확인할 수 있다.

1. **LIST**

|  |  |
| --- | --- |
| KEY | VALUE |
| NAME | KBS |
| AGE | 24 |
| ADDRESS | INCHEON |

텍스트이(가) 표시된 사진

자동 생성된 설명

LIST를 통해 현재 저장된 값을 가져온 경우이다.

정상적으로 Data를 가져오는 것을 확인할 수 있다.

1. **DELETE -> LIST**

|  |  |
| --- | --- |
| KEY | VALUE |
| AGE | 24 |
| ADDRESS | INCHEON |

텍스트이(가) 표시된 사진

자동 생성된 설명

DELETE를 통해 NAME을 key 값으로 가지는 value를 삭제한 후 LIST를 진행하였다.

성공적으로 제거된 것을 확인할 수 있었다.

1. **GET**

|  |  |
| --- | --- |
| KEY | VALUE |
| AGE | 24 |
| ADDRESS | INCHEON |

텍스트, 스크린샷, 화면이(가) 표시된 사진

자동 생성된 설명

DELETE를 진행한 후 GET을 진행한 모습이다.

성공적으로 Not exist!가 출력된 것을 확인할 수 있다.

1. **CLOSE**

텍스트, 스크린샷, 화면이(가) 표시된 사진

자동 생성된 설명

CLOSE를 수행한 결과이다.

Client와 Server가 CLOSE된 것을 확인할 수 있다.

* **개선할 점**

위에서 구현한 코드는 실제 TCP통신과 비교하면 미흡한 점이 존재한다.

현재 구현한 TCP 통신은 Local host 통신으로 Packet error와 loss가 발생할 가능성이 매우 낮다.

따라서 위와 같은 error와 loss에 대한 부분을 구현하지 않았지만 큰 문제가 발생하지 않았다.

실제 TCP 통신에서는 위와 같은 문제가 발생할 수 있기 때문에 이 문제를 해결할 수 있는 방법을 구현해야 한다.

TCP 통신과 비교하면 위 코드에서 부족한 부분은 Check Sum과 ACK이다.

Error가 발생했는지 확인하기 위해 Check Sum Bit가 필요하고 이를 통해 ACK을 보내 Packet을 정상적으로 받았는지 확인해야 한다.

또한 Loss를 확인하기 위해 time out을 위한 코드도 구현해야 한다.

위 코드에서는 ACK/NACK, Check Sum Bit, Timer 모두 구현되어 있지 않다는 아쉬움이 있다.

정리하면 위 코드에 추가하면 좋은 부분은

1. Check Sum Bit를 통한 Error 확인
2. ACK/NACK등을 통한 Error, Loss 확인
3. Sequence Number 도입으로 병렬적인 데이터 송수신 보장
4. 강제 종료시 socket 할당 풀기
5. 예외 처리 보완

으로 정리할 수 있다.

**-참고문헌**

[setsockopt - 소켓옵션 (joinc.co.kr)](https://www.joinc.co.kr/w/Site/Network_Programing/AdvancedComm/SocketOption)

[파이썬 소켓 연결 사용법 · Wireframe (soooprmx.com)](https://soooprmx.com/%ED%8C%8C%EC%9D%B4%EC%8D%AC-%EC%86%8C%EC%BC%93-%EC%97%B0%EA%B2%B0-%EC%82%AC%EC%9A%A9%EB%B2%95/)