

# IGRUS Winter Bootcamp: Socket Project

Due on 2026.01.12

IGRUS

## 도입

파이썬의 `socket` 라이브러리를 사용해서 간단한 서버와 클라이언트 프로그램을 만들어 봅시다. 소켓 프로그래밍을 직접 해보면 소켓이 무엇인지, 소켓을 통해 어떻게 통신이 이루어지는지 이해할 수 있습니다.

## 목표

- **소켓과 TCP 통신의 기본 흐름 이해:** 서버 소켓의 `bind-listen-accept` 과정과 클라이언트 소켓의 `connect` 과정을 직접 구현하며, TCP 연결이 어떻게 생성되고 유지되는지 이해합니다.
- **클라이언트-서버 모델의 요청/응답 패턴 학습:** 클라이언트가 요청 메시지를 보내고 서버가 처리 결과를 응답하는 구조를 구현하며, 애플리케이션 레벨에서의 프로토콜(요청 형식, 응답 형식)을 설계하는 감각을 익힙니다.
- **애플리케이션 프로토콜 파싱 및 예외 처리 능력 향상:** 콤마(,) 구분자 기반 텍스트 명령(PUT/GET/DELETE/LIST)을 파싱하고, 잘못된 입력(필드 부족, 알 수 없는 명령 등)에 대해서도 서버가 죽지 않도록 방어적으로 처리하는 방법을 학습합니다.
- **상태를 가지는 서버(Stateful Server) 설계 경험:** 서버 프로세스가 실행되는 동안 key-value 데이터를 메모리에 유지하고, PUT/GET/DELETE/LIST 동작을 통해 CRUD 개념을 소켓 통신 위에서 구현해 봅니다.
- **지속 실행 가능한 서버 프로그램 작성:** 서버가 한 번의 요청 처리 후 종료되지 않고 반복적으로 클라이언트 요청을 처리하도록 구현하며, 반복 처리 루프와 자원 정리(소켓 종료 등)의 기본을 익힙니다.

## 요구사항

- 서버는 실행 이후 항상 죽지 않고 클라이언트의 요청을 수용할 수 있어야 합니다.
- 클라이언트가 보내는 콤마(,) 구분자 기반 텍스트 데이터에 대하여 아래와 같은 처리가 가능해야 합니다.
  - PUT, textA, textB 를 전송 시 서버는 textA 라는 key 에 textB 라는 value 가 매핑되었다는 정보를 저장하고, 저장이 성공했음을 클라이언트에게 알려주어야 합니다.
    - \* 예) PUT, myname, yyjun → Success!
    - \* 동일한 key에 대해서 PUT 요청이 오면 덮어쓰기 하도록 구현합니다.
  - GET, textA 를 전송 시 서버는 textA 라는 key 를 갖는 value 를 찾아 클라이언트에게 알려주어야 합니다.
    - \* 예) GET, myname → yyjun (myname 이라는 key 가 존재할 경우)
    - \* 예) GET, myname → Not exist! (myname 이라는 key 가 존재하지 않을 경우)
  - DELETE, textA 를 전송 시 서버는 textA 라는 key 를 갖는 key-value 쌍을 찾아 삭제한 뒤 성공했음을 클라이언트에게 알려주어야 합니다.
    - \* 예) DELETE, myname → Success! (myname 이라는 key 가 존재할 경우)
    - \* 예) DELETE, myname → Not exist! (myname 이라는 key 가 존재하지 않을 경우)
  - LIST 를 전송 시 서버는 자신이 가지고 있는 모든 key 에 대하여 key-value 의 리스트를 클라이언트에게 알려주어야 합니다.
    - \* 예) LIST →  
name, yyjun  
age, 20  
role, backend (3개의 key 가 존재할 경우 / key의 순서는 상관 없음)

\* 예) LIST  $\rightarrow$  Not exist! (어떠한 key 도 존재하지 않을 경우)

- 서버가 처음 실행된 순간엔 어떠한 key-value 도 가지고 있지 않아야 합니다.
- 서버가 key-value 데이터를 유지하기 위해서 map 을 사용해도 되고, 본인만의 구조체를 만들어 사용해도 됩니다.

## 보고서

- 보고서에는 전체 코드에 대한 간단한 설명과 실행 화면 캡처 사진을 포함해 주세요.