작성자 : 김기희 (cupid072@naver.com)

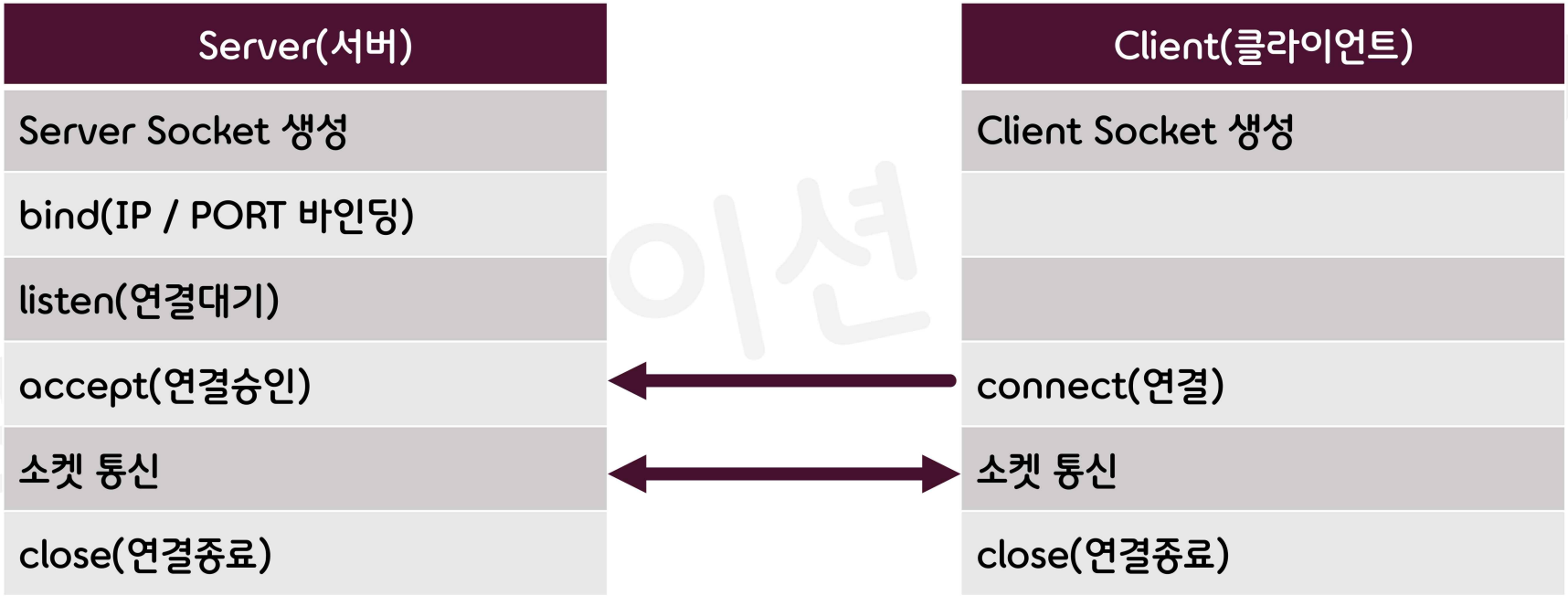
파이썬 표준 라이브러리

- socket 모듈

작성자 : 김기희 (cupid072@naver.com)

Socket 이란?

- 소켓(Socket)이란? 네트워크 상에서 동작하는 프로그램 간 통신의 종착점(Endpoint) 이다.
- 소켓(Socket)은 프로그램이 네트워크를 통해서 다른 호스트와 통신을 할 수 있도록 해주는 연결부이다.



서버 측 프로그램

- 서버에서는 서버 소켓을 생성하고 클라이언트의 접속에 대기하여야 하며 절차는 다음과 같다.
  - 서버 소켓 생성
    socket.socket(socket.AF_INET, socket.SOCK_STREAM)
  - 생성된 소켓에 옵션을 지정한다.
    server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
  - 소켓을 IP 주소와 PORT에 바인딩한다.
    server_socket.bind(('', PORT))
  - 서버 소켓을 대기 상태로 전환한다.
    server_socket.listen()
  - 클라이언트 접속 시 연결을 승인한다. accept() 함수는 클라이언트의 접속을 허용한 후 클라이언트 소켓과 클라이언트 주소 정보를 튜플 형식으로 반환한다.
    (client_socket, client_addr) = server_socket.accept() # 소켓과 정보를 따로 저장
  - 서버 소켓을 종료한다.
    server_socket.close()

클라이언트 측 프로그램

- 클라이언트에서는 소켓을 생성하여 서버에 접속하며 절차는 다음과 같다.

  ➢ 소켓 생성

```
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

  ➢ 서버 연결

```
client_socket.connect((HOST, PORT))
```

  ➢ 연결 종료

```
client_socket.close()
```

여러명이 동시에 접속이 가능하도록 코드를 고쳐보세요.

```
import socket
HOST = '127.0.0.1'
PORT = 9999

print(">>> Server Start")
try:
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    server_socket.bind(('', PORT))
    server_socket.listen()
    print(f">>> 서버가 포트번호 {PORT}번으로 대기중입니다.")
    while True:
        (client_socket, addr) = server_socket.accept()
        print(f"{addr[0] 클라이언트가 접속하였습니다.")
        while True:
            data = client_socket.recv(1024)
            print(f"수신 <<< {data.decode()}")
            client_socket.send(data)
except Exception as e:
    print(e)
finally:
    server_socket.close()

print("프로그램을 종료합니다.")
```

추가기능

- ❖ 여러 클라이언트가 접속이 가능해야 합니다.
- ❖ 클라이언트 접속 시 현재 접속중인 모든 클라이언트에게 클라이언트 입장 메시지를 전송해야 합니다.
- ❖ 클라이언트에서 수신된 메시지는 해당 클라이언트를 제외한 모든 클라이언트에게 중계가 되어야 합니다.
- ❖ 클라이언트와의 연결 종료 시 클라이언트의 퇴장을 알리는 메시지를 전체 클라이언트에게 전송해야 합니다.
- ❖ 서버에서 전체 클라이언트에게 메시지를 전송할 수 있어야 합니다.

클라이언트 코드 #1

```
import socket
from threading import Thread

HOST = 'localhost'
PORT = 9999

def recv_data(client_socket):
    while True:
        try:
            data = client_socket.recv(1024)          # 수신 대기중 소켓이 닫히면 Error 발생
            print(f"수신 >>> {data.decode()}")
        except Exception as e:                       # 수신 대기중 에러 발생시 스레드 종료
            print("서버와의 연결이 종료되어 수신대기를 종료합니다.")
            break
```

클라이언트 코드 #2

```
try:
        client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        client_socket.connect((HOST, PORT))
        print(">>> 서버에 연결하였습니다.")
        Thread(target=recv_data, args=(client_socket,)).start()     # 메시지 수신자 스레드 시작
        while True:
            data = input("> ")
            if not data: continue
            if data.upper() == 'QUIT': break
            client_socket.send(data.encode())
    except Exception as e:
        print(e)
    finally:
        print("서버와의 연결을 종료합니다.")
        client_socket.close()

print("프로그램을 종료합니다.")
```