# Python을 이용한 Socket 통신 이해 및 응용

@서울시립대학교

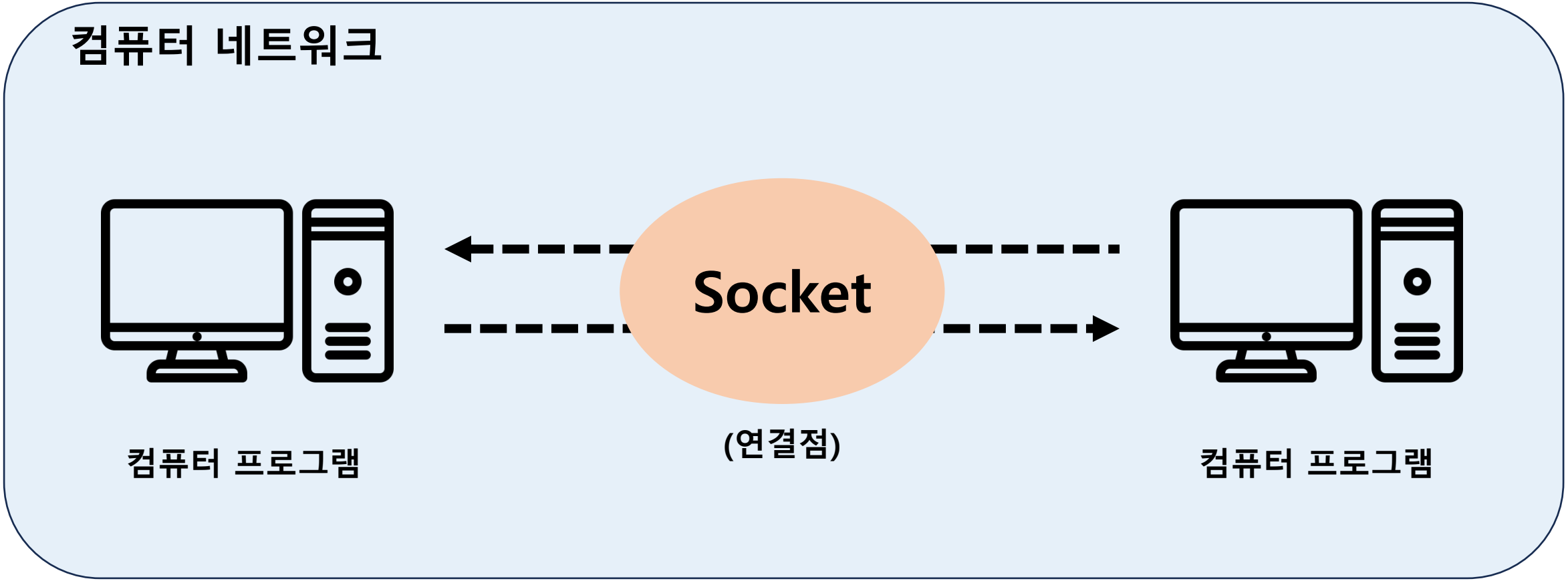
이 민 호

2024. 01. 22

# Contents

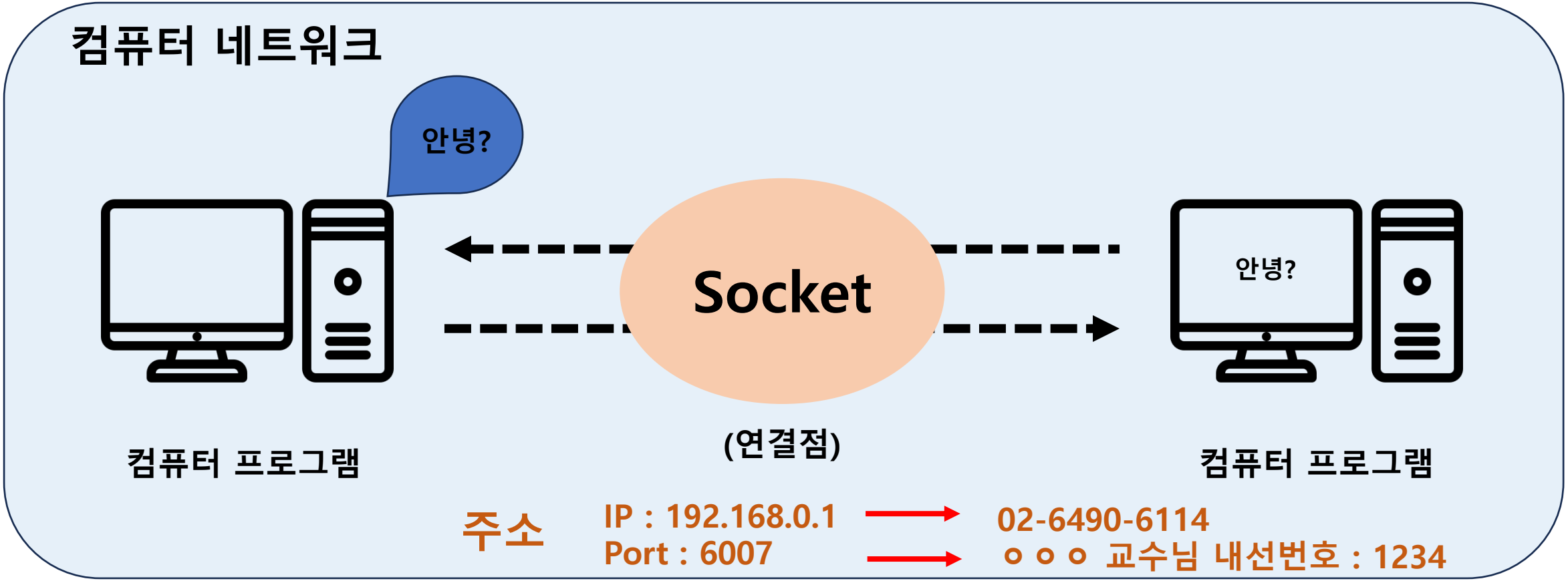
# Part 1 : Introduction to Socket Programming

## Overview of Socket Programming – 소켓이란?



# Part 1 : Introduction to Socket Programming

## Overview of Socket Programming – 소켓이란?



# Part 1 : Introduction to Socket Programming

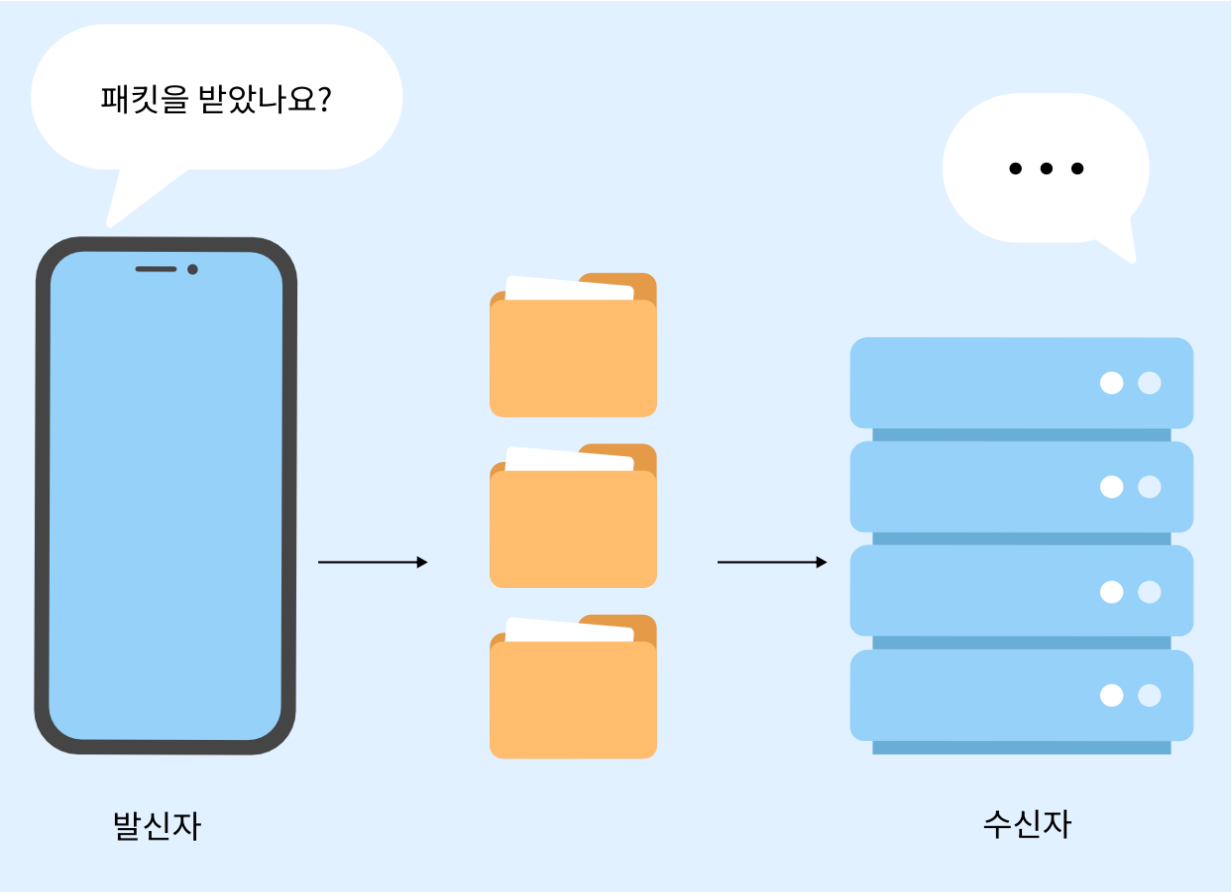
## Overview of Socket Programming – 소켓의 종류와 기능

데이터그램 소켓　　스트림 소켓　　로우 소켓

# Part 1 : Introduction to Socket Programming

## Overview of Socket Programming – 소켓의 종류와 기능

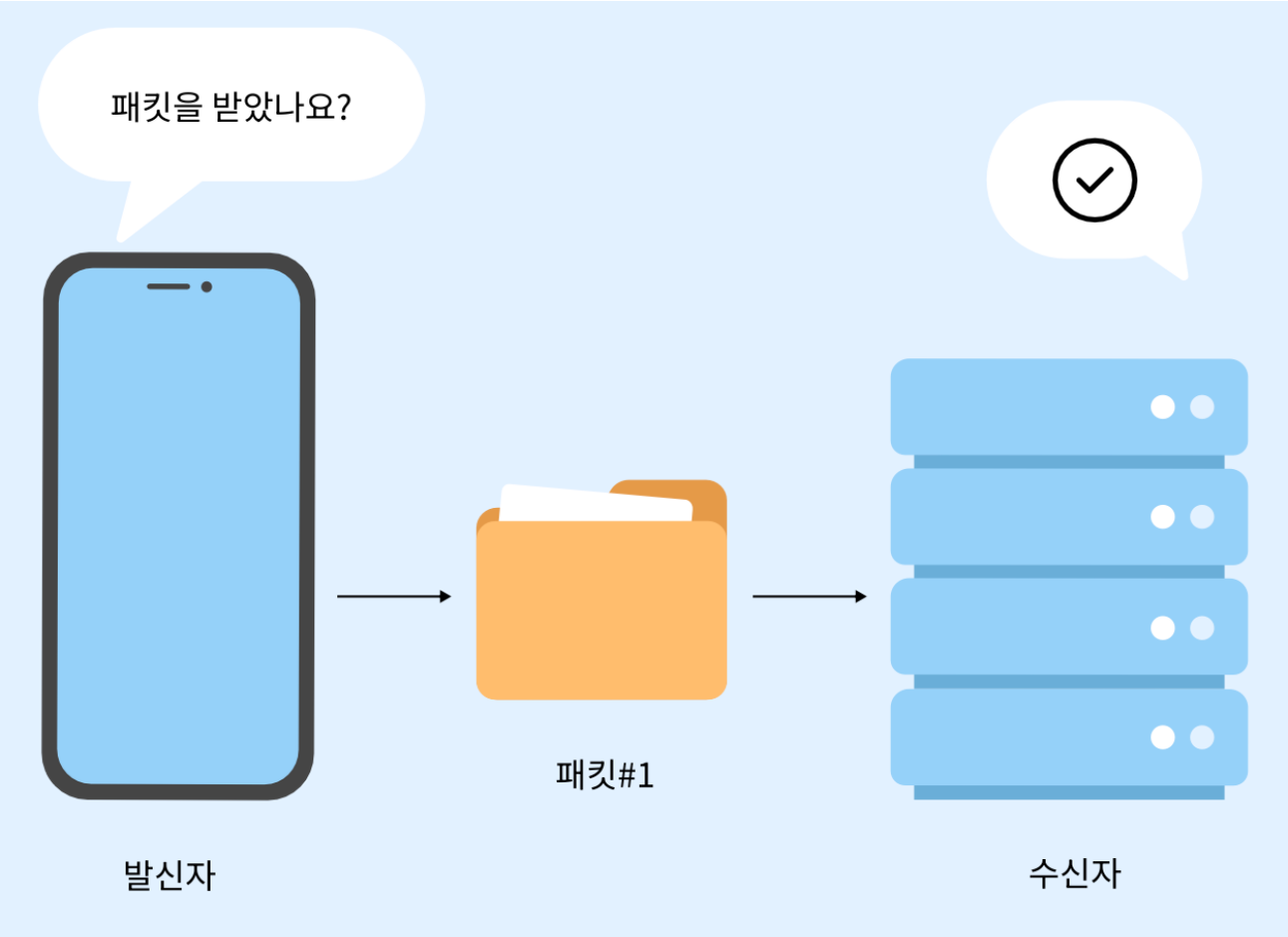
데이터그램 소켓 스트림 소켓 로우 소켓



# Part 1 : Introduction to Socket Programming

## Overview of Socket Programming – 소켓의 종류와 기능

데이터그램 소켓 스트림 소켓 로우 소켓



# Part 1 : Introduction to Socket Programming

## Overview of Socket Programming – 소켓의 종류와 기능

데이터그램 소켓　　스트림 소켓　　로우 소켓

### 로우 소켓

### IP 패킷을 직접 보내고 받을 수 있는 형태

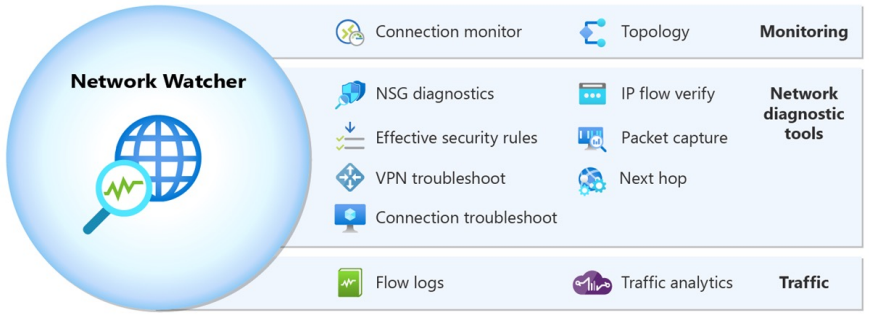
# Part 1 : Introduction to Socket Programming

## Overview of Socket Programming – 소켓 프로그래밍 사용 예시


채팅 애플리케이션


파일전송 프로그램


스트리밍 서비스




네트워크 모니터링

# Part 1 : Introduction to Socket Programming

## Overview of Socket Programming – 소켓 프로그래밍 사용 예시


데이터그램 소켓
(UDP)

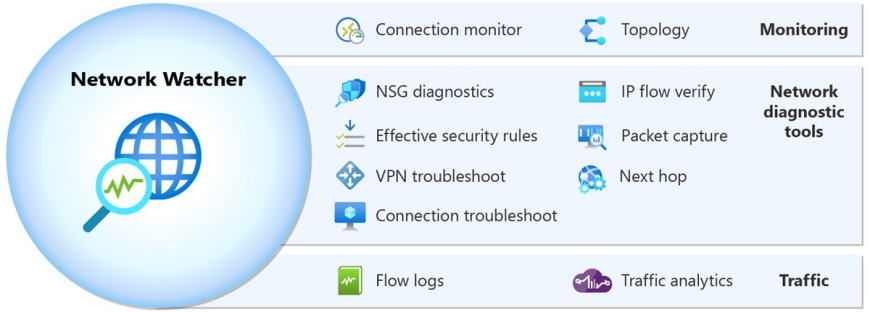
채팅 애플리케이션


스트림 소켓
(TCP)

파일전송 프로그램


데이터그램 소켓
(UDP)

스트리밍 서비스




로우 소켓
(IP패킷 직접전송)

네트워크 모니터링

# Part 1 : Introduction to Socket Programming

## Overview of Socket Programming – TCP와 UDP의 차이

| TCP | | UDP |
|---|---|---|
| 높음 | 신뢰성 | 낮음 |
| 낮음 | 속도 | 높음 |
| 패킷이 순서대로 | 전송 방법 | 패킷이 스트레이트로 |
| 있음 | 혼잡도 제어 | 없음 |

# Part 1 : Introduction to Socket Programming

## Overview of Socket Programming – Python을 사용하는 이유

아주 쉬움

빠르게 작성 가능

다양한 기능

# Part 1 : Introduction to Socket Programming

## Basics of Python's Socket Library – Python 소켓 라이브러리 소개

### Socket 라이브러리

import socket

TCP/IP 와 UDP 등 다양한 네트워크 프로토콜 인터페이스 제공

# Part 1 : Introduction to Socket Programming

## Basics of Python's Socket Library – 자주 사용되는 함수

| | | | |
|---|---|---|---|
| socket() | bind() | listen() | accept() |
| connect() | send() | recv() | close() |

# Part 1 : Introduction to Socket Programming

## Basics of Python's Socket Library – 자주 사용되는 함수

**socket()**

소켓 객체 생성 함수

**bind()**

특정 네트워크와 연결 (바인딩)

**listen()**

서버가 " 듣기" 모드

**accept()**

클라이언트가 연결 수락

**connect()**

클라이언트가 서버에 연결 시도

**send()**

데이터 전송

**recv()**

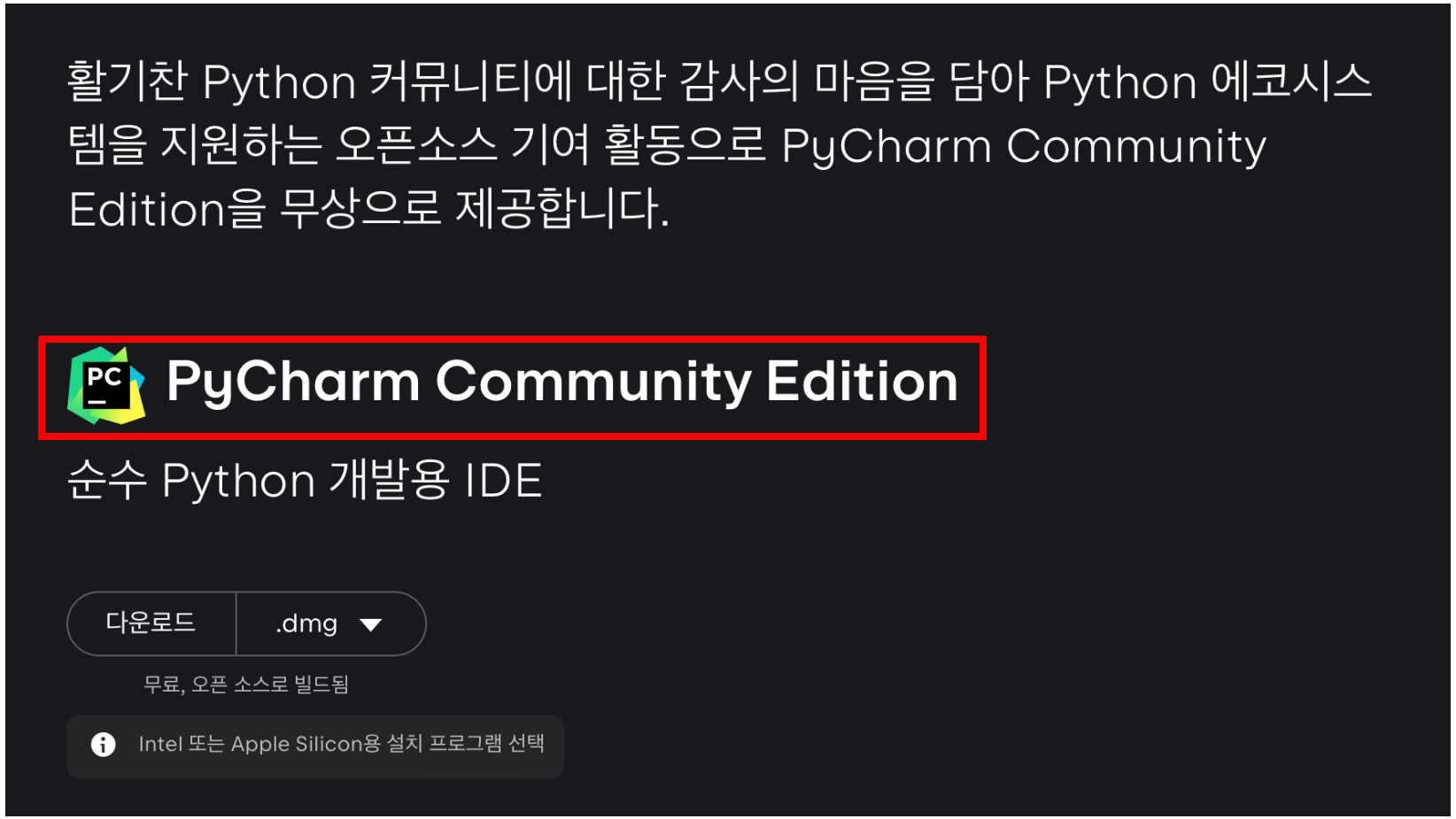
데이터 수신

**close()**

소켓 연결 종료

# Part 1 : Introduction to Socket Programming

## 실습 준비 – PyCharm 을 이용한 실습



# Part 1 : Introduction to Socket Programming

## 실습 준비 – Github 실습 코드 다운로드

https://github.com/minoTrey/20240122_socket.git

# Part 2 : Setting Up a Basic Socket Server and Client

## Creating a Simple Echo Server – 에코 서버?


서버가 클라이언트로부터 받은 메시지를

그대로 다시 클라이언트에 전송

# Part 2 : Setting Up a Basic Socket Server and Client

## Creating a Simple Echo Server – 에코 서버 코드

### tcp_server.py

```python
import socket


1 usage
def run_tcp_server(port=65432):
    # 소켓 객체 생성 (AF_INET: IPv4, SOCK_STREAM: TCP)
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        # 주소와 포트 바인딩
        s.bind(('127.0.0.1', port))
        # 클라이언트의 연결 요청 대기
        s.listen()
        # 연결 요청 수락
        conn, addr = s.accept()
        with conn:
            print('Connected by', addr)
            while True:
                # 클라이언트로부터 데이터 수신
                data = conn.recv(1024)
                if not data:
                    break
                # 수신한 데이터 출력
                print(f"Received (TCP): {data.decode()}")
                # 수신한 데이터를 그대로 클라이언트에게 전송
                conn.sendall(data)


if __name__ == "__main__":
    run_tcp_server()
```

# Part 2 : Setting Up a Basic Socket Server and Client

## Building a Simple Client – 에코 클라이언트 코드

### tcp_client.py

```python
import socket


1 usage
def run_tcp_client(server_host='127.0.0.1', server_port=65432):
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        # 서버에 연결
        s.connect((server_host, server_port))
        # 서버에 메시지 전송
        message = "Hello, server"
        s.sendall(message.encode())
        # 서버로부터 데이터 수신
        data = s.recv(1024)
        # 수신한 데이터 출력
        print('Received:', data.decode())


if __name__ == "__main__":
    run_tcp_client()
```

# Part 2 : Setting Up a Basic Socket Server and Client

## Building a Simple Client – 결과

### tcp_server.py

```
/usr/bin/python3 /Users/trey_macbook/Pyc
Connected by ('127.0.0.1', 49809)
Received (TCP): Hello, server

Process finished with exit code 0
```

### tcp_client.py

```
/usr/bin/python3 /Users/trey_macbook/Py
Received: Hello, server

Process finished with exit code 0
```

# Part 2 : Setting Up a Basic Socket Server and Client

## Hands-On Coding Exercise – 연습해보세요

### tcp_client.py

```python
import socket


1 usage
def run_tcp_client(server_host='127.0.0.1', server_port=65432):
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        # 서버에 연결
        s.connect((server_host, server_port))
        # 서버에 메시지 전송
        message = "Hello, server"
        s.sendall(message.encode())
        # 서버로부터 데이터 수신
        data = s.recv(1024)
        # 수신한 데이터 출력
        print('Received:', data.decode())


if __name__ == "__main__":
    run_tcp_client()
```

원하는 문장을 넣어보세요!

# Part 3 : Advanced Socket Programming Concepts

## Data Transmission Nuances – 데이터 전송의 신뢰성

| TCP | UDP |
| --- | --- |
| 패킷 손실시 재전송 | 패킷 그대로 손실 |

# Part 3 : Advanced Socket Programming Concepts

## Data Transmission Nuances – 연결 및 연결 없음

| TCP | UDP |
| --- | --- |
| 핸드쉐이크 과정으로 연결 수립후 데이터 전송 | 연결이 수립되지 않아도 데이터 전송 |

# Part 3 : Advanced Socket Programming Concepts

## Data Transmission Nuances – 데이터 순서와 흐름 제어

| TCP | UDP |
| --- | --- |
| 데이터가 순서대로 도착 | 데이터 순서에 대해 보장하지 않음 |

# Part 3 : Advanced Socket Programming Concepts

## TCP vs. UDP Sockets – UDP 예제 코드

### udp_server.py

```python
import socket


1 usage
def run_udp_server(port=65432):
    # 소켓 객체 생성 (AF_INET: IPv4, SOCK_DGRAM: UDP)
    with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as s:
        # 주소와 포트 바인딩
        s.bind(('127.0.0.1', port))
        while True:
            # 클라이언트로부터 데이터 수신
            data, addr = s.recvfrom(1024)
            # 수신한 데이터와 클라이언트 주소 출력
            print(f"Received (UDP): {data.decode()} from {addr}")


if __name__ == "__main__":
    run_udp_server()
```

# Part 3 : Advanced Socket Programming Concepts

## TCP vs. UDP Sockets – UDP 예제 코드

### udp_client.py

```python
import socket


1 usage  Minho Lee *
def run_udp_client(server_host='127.0.0.1', server_port=65432):
    with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as s:
        message = "Hello, UDP server"
        s.sendto(message.encode(), (server_host, server_port))
        # 서버로부터의 응답을 기다리지 않고 종료


if __name__ == "__main__":
    run_udp_client()
```

# Part 3 : Advanced Socket Programming Concepts

## TCP vs. UDP Sockets – UDP 예제 결과

### udp_server.py

```
/usr/bin/python3 /Users/trey_macbook/PycharmProjects/20240122_sc
Received (UDP): Hello, UDP server from ('127.0.0.1', 59546)
```

### udp_client.py

```
/usr/bin/python3 /Users/trey_macbook/PycharmP

Process finished with exit code 0
```

# Part 4 : Practical Application

## Chatting Application – 채팅 서버 코드

### chat_server.py

```python
import socket
import threading


1 usage
def handle_client(conn, addr):
    # 클라이언트 처리 함수
    print(f"[NEW CONNECTION] {addr} connected.")

    connected = True
    while connected:
        # 클라이언트로부터 메시지 수신
        msg = conn.recv(1024).decode("utf-8")
        if msg:
            # 수신된 메시지 출력
            print(f"[{addr}] {msg}")

    # 연결 종료
    conn.close()
```

```python
def start_server():
    # 서버 소켓 설정
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind(('127.0.0.1', 50000))
    server.listen()

    print("[STARTING] Server is starting...")

    while True:
        # 클라이언트 연결 대기
        conn, addr = server.accept()
        # 새 클라이언트 연결에 대한 스레드 시작
        thread = threading.Thread(target=handle_client, args=(conn, addr))
        thread.start()
        # 활성 연결 수 출력
        print(f"[ACTIVE CONNECTIONS] {threading.activeCount() - 1}")


if __name__ == "__main__":
    start_server()
```

# Part 4 : Practical Application

## Chatting Application – 채팅 클라이언트 코드

### chat_client.py

```python
import socket

1 usage
def start_client():
    # 클라이언트 소켓 설정
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect(('127.0.0.1', 50000))

    while True:
        # 사용자로부터 메시지 입력 받음
        msg = input("Message: ")
        # 입력 받은 메시지를 서버에 전송
        client.send(msg.encode("utf-8"))

if __name__ == "__main__":
    start_client()
```

# Part 4 : Practical Application

## Chatting Application – 채팅 앱 결과

### chat_server.py

```
/usr/bin/python3 /Users/trey_macbook/PycharmProje
[STARTING] Server is starting...
[NEW CONNECTION] ('127.0.0.1', 50384) connected.
[ACTIVE CONNECTIONS] 1
[('127.0.0.1', 50384)] 안녕하세요?
```

### chat_client.py

```
/usr/bin/python3 /Users/trey_macbook/I
Message: 안녕하세요?
Message:
```