

# Computer Network[Socket]

유명성

# 0. Python File API

---

# 0. Python File API

## 0.1 What is file

### File

- ❖ 0과 1로 구성된 연속적인 데이터
- ❖ Text, Image 등 모든 파일은 Binary로 기록되며, 읽고/쓰는 방식의 차이가 있다.
- ❖ 읽고 쓰는 방식을 '인코딩' 이라 한다.

```
00000000 0000 0001 0001 1010 0010 0001 0004 0128
00000010 0000 0016 0000 0028 0000 0010 0000 0020
00000020 0000 0001 0004 0000 0000 0000 0000 0000
00000030 0000 0000 0000 0010 0000 0000 0000 0204
00000040 0004 8384 0084 c7c8 00c8 4748 0048 e8e9
00000050 00e9 6a69 0069 a8a9 00a9 2828 0028 fdfe
00000060 00fc 1819 0019 9898 0098 d9d8 00d8 5857
00000070 0057 7b7a 007a bab9 00b9 3a3c 003c 8888
00000080 8888 8888 8888 8888 288e be88 8888 8888
00000090 3b83 5788 8888 8888 7667 778e 8828 8888
000000a0 d61f 7abd 8818 8888 467c 585f 8814 8188
000000b0 8b06 e8f7 88aa 8388 8b3b 88f3 88bd e988
000000c0 8a18 880c e841 c988 b328 6871 688e 958b
000000d0 a948 5862 5884 7e81 3788 1ab4 5a84 3eec
000000e0 3d86 dcb8 5cbb 8888 8888 8888 8888 8888
000000f0 8888 8888 8888 8888 8888 8888 8888 0000
0000100 0000 0000 0000 0000 0000 0000 0000 0000
*
0000130 0000 0000 0000 0000 0000 0000 0000
000013e
```

# 0. Python File API

## 0.2 File API

### File 처리과정

1. 읽고 쓰는 방법인 인코딩 방식을 정한다.
2. 모든 데이터는 Python에서 bytes 및 bytearray 타입으로 취급된다.
3. 문자 등을 파일에 저장하기 위해 인코딩(byte 타입으로 변환)한다.
4. 데이터를 읽을 땐 디코딩(byte 타입에서 다른 타입으로) 한다.
5. Text 파일은 3, 4과정이 자동으로 수행된다.

### 인코딩

- ❖ Python은 내부적으로 유니코드를 사용한다.
- ❖ 기본적으로 인코딩은 UTF-8을 사용한다.
  - ❖ UTF-8은 1~4Byte로 문자를 표현하며 ASCII와 호환된다.

character	encoding	bits
A	UTF-8	01000001
A	UTF-16	00000000 01000001
A	UTF-32	00000000 00000000 00000000 01000001
あ	UTF-8	11100011 10000001 10000010
あ	UTF-16	00110000 01000010
あ	UTF-32	00000000 00000000 00110000 01000010

# 0. Python File API

## 0.2 File API



### open

- ❖ 파일을 사용하려면 먼저 열어야 한다.
- ❖ 연다는 것은 커널에 사용할 파일에 관련된 구조체를 생성하고 받는 것이다.

그 구조체에 접근할 수 있는 식별자(FD)를

```
open(파일경로와이름, 모드, 인코딩) -> 파일 디스크립터
```

- ❖ FD(File descriptor)를 통해 파일에 접근 가능하다.
- ❖ write(data) : byte 혹은 bytearray data를 파일에 쓴다.
- ❖ read(data) : 파일에서 byte 혹은 bytearray data를 읽는다.
- ❖ close() : FD를 제거하고 파일을 닫는다.

# 0. Python File API

## 0.2 File API

플래그	의미	비고
r	읽기모드	
w	쓰기모드	파일을 쓰기 모드로 연다. 파일에 데이터를 쓰면 기존 파일의 내용은 모두 날아간다. 주어진 파일이 존재하지 않으면 새로운 파일을 만든다.
x	쓰기 전용	새 파일 쓰기 모드로 연다. 주어진 이름의 파일이 존재하면 에러가 발생한다.
a	추가모드	파일을 추가 모드로 연다. 기존 파일의 내용의 끝에 새 내용을 추가하여 기록한다.
+	갱신모드	파일을 읽기와 쓰기가 모두 가능한 모드로 연다.
rb	이진 읽기 모드	이진 파일을 읽기 모드로 연다. 읽어들이는 데이터는 디코딩되지 않은 바이트배열이 된다.
wb	이진 쓰기 모드	이진 파일을 쓰기 모드로 연다. 쓰는 데이터는 raw 데이터 그대로 쓰인다.

# 0. Python File API

## 0.2 File API

### ■ Text 파일 읽기

1. utf-8 인코딩으로 test.txt를 읽기 모드로 연다.
2. test.txt의 데이터를 읽는다.(디코딩은 자동으로 수행됨)
3. 파일을 닫는다.

```
1 filename = "test.txt"
2 file = open(filename, 'r', encoding="utf8") ## 1
3 text_str = file.read() ## 2
4 print(text_str)
5 file.close() ## 3
```

# 0. Python File API

## 0.2 File API

### ■ Text 파일 쓰기

1. 99dan.txt를 쓰기모드로 연다.(파일이 디스크에 없으면 생성)
2. 입력할 문자열을 만든다. -> f' '은 포맷스트링
3. 데이터를 파일에 쓴다. (인코딩은 자동으로 수행된다.)
4. 파일을 닫는다.

```
1 file = open('99dan.txt', 'w', encoding='utf8') ## 1
2 for a in range(1, 9):
3     line = f'7 * {a} = {a * 7}\n' ## 2
4     file.write(line) ## 3
5 file.close() ## 4
```



# 1. Python Tip

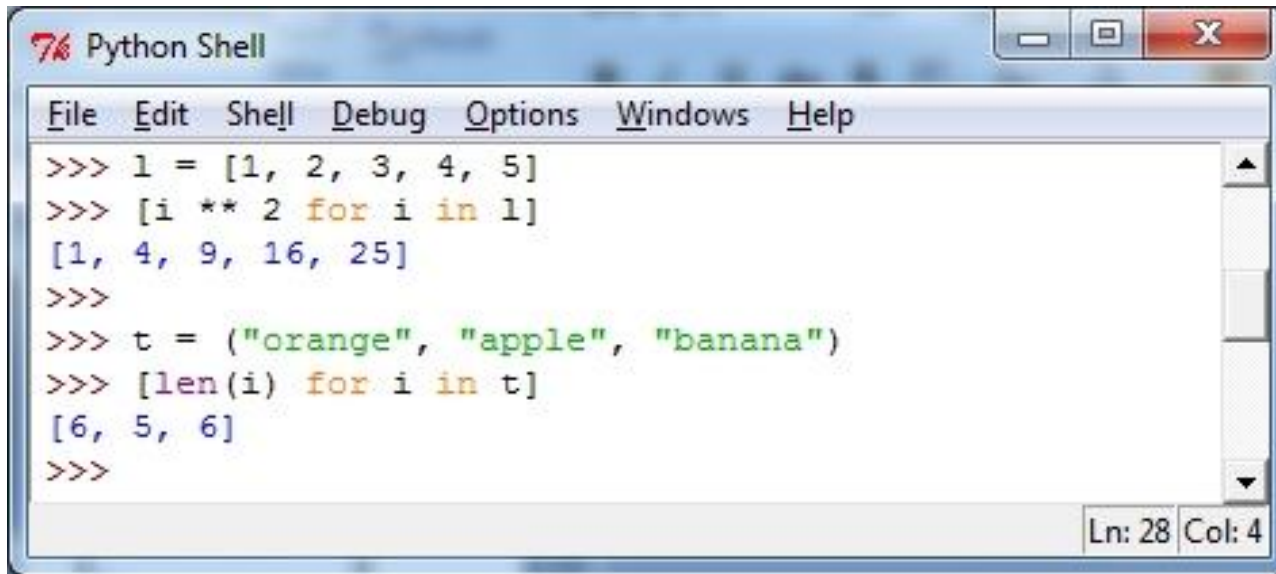
---

# 1. Python Tip

## 1.1 List Comprehension

### List Comprehension

- ❖ [<표현식> for <아이템> in <시퀀스 객체> (if <조건식>)]
- ❖ 기존 시퀀스 객체를 이용하여 추가적인 연산을 통하여 새로운 리스트 객체를 생성



```
Python Shell
File Edit Shell Debug Options Windows Help
>>> l = [1, 2, 3, 4, 5]
>>> [i ** 2 for i in l]
[1, 4, 9, 16, 25]
>>>
>>> t = ("orange", "apple", "banana")
>>> [len(i) for i in t]
[6, 5, 6]
>>>
Ln: 28 Col: 4
```

# 1. Python Tip

## 1.1 List Comprehension

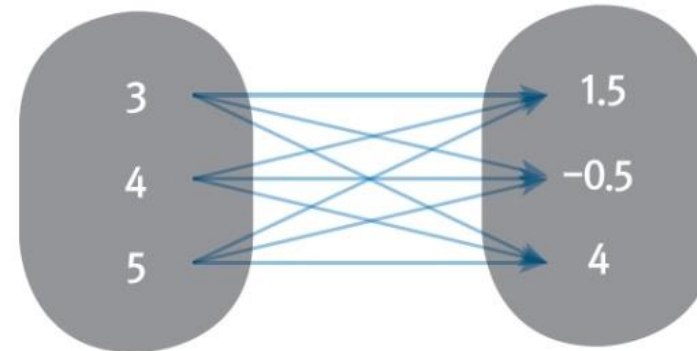
### List Comprehension

- ❖ [ $\langle$ 표현식 $\rangle$  for  $\langle$ 아이템 $\rangle$  in  $\langle$ 시퀀스 객체 $\rangle$  (if  $\langle$ 조건식 $\rangle$ )]
- ❖ 조건식을 이용하여 원본 객체에서 조건을 만족하는 아이템만 선별

```
Python Shell
File Edit Shell Debug Options Windows Help
>>> t = ("orange", "apple", "banana", "kiwi")
>>> [i for i in t if len(i)>5]
['orange', 'banana']
>>>
>>>
>>>
```

- ❖ 원본 리스트가 2개인 경우

```
Python Shell
File Edit Shell Debug Options Windows Help
>>> L_1 = [3, 4, 5]
>>> L_2 = [1.5, -0.5, 4]
>>> [x*y for x in L_1 for y in L_2]
[4.5, -1.5, 12, 6.0, -2.0, 16, 7.5, -2.5, 20]
>>>
```

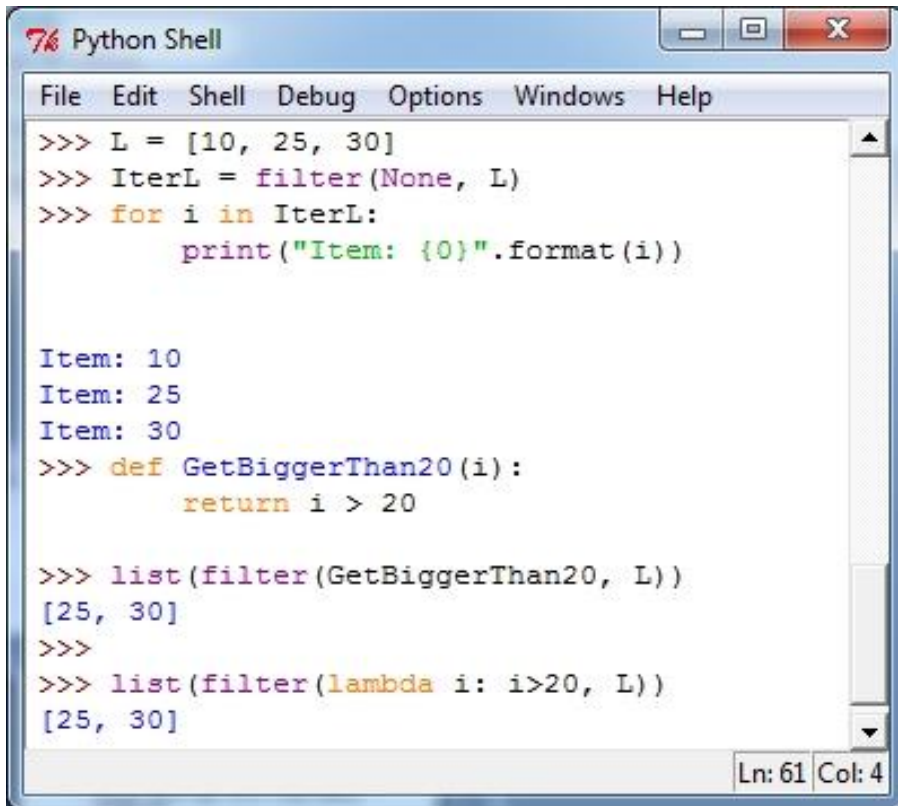


# 1. Python Tip

## 1.2 Filter

### filter

- ❖ filter(<function>|None, 시퀀스 객체)
- ❖ 함수의 결과 값이 참인 시퀀스 객체의 이터레이터를 반환
- ❖ None이 오는 경우 필터링하지 않음



```
Python Shell
File Edit Shell Debug Options Windows Help
>>> L = [10, 25, 30]
>>> IterL = filter(None, L)
>>> for i in IterL:
    print("Item: {}".format(i))

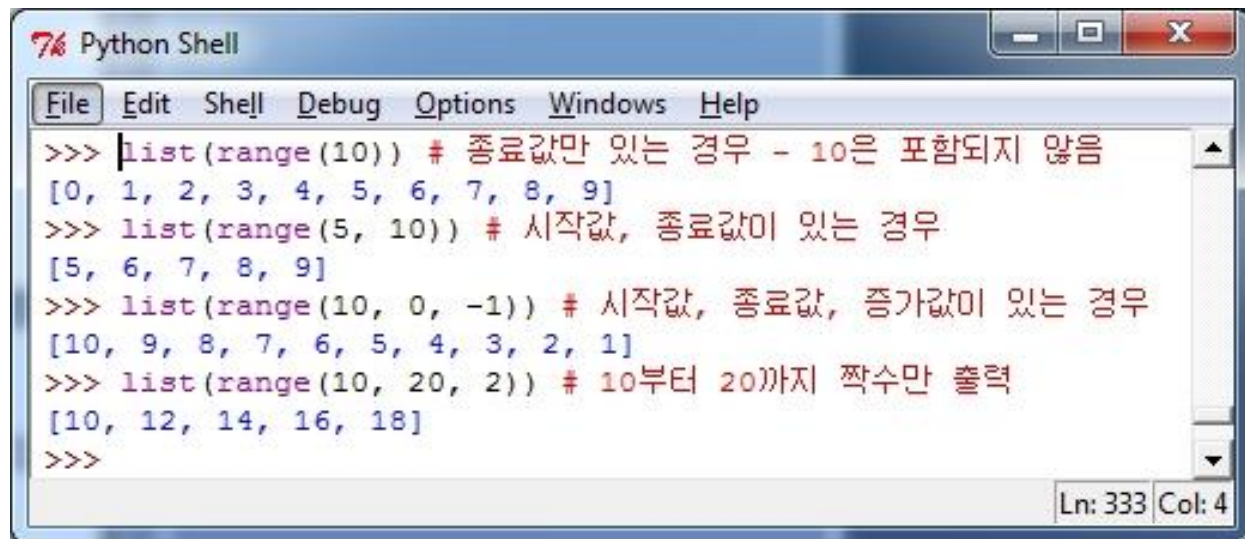
Item: 10
Item: 25
Item: 30
>>> def GetBiggerThan20(i):
    return i > 20
>>> list(filter(GetBiggerThan20, L))
[25, 30]
>>>
>>> list(filter(lambda i: i>20, L))
[25, 30]
Ln: 61 Col: 4
```

# 1. Python Tip

## 1.3 range

### range

- ❖ `range(['시작값', '종료값', '증가값'])`
- ❖ 수열을 순회하는 이터레이터 객체를 반환
- ❖ 시작값과 증가값은 생략 가능하며, 이때는 각 0, 1이 할당



```
Python Shell
File Edit Shell Debug Options Windows Help
>>> list(range(10)) # 종료값만 있는 경우 - 10은 포함되지 않음
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(5, 10)) # 시작값, 종료값이 있는 경우
[5, 6, 7, 8, 9]
>>> list(range(10, 0, -1)) # 시작값, 종료값, 증가값이 있는 경우
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
>>> list(range(10, 20, 2)) # 10부터 20까지 짝수만 출력
[10, 12, 14, 16, 18]
>>>
```

Ln: 333 Col: 4

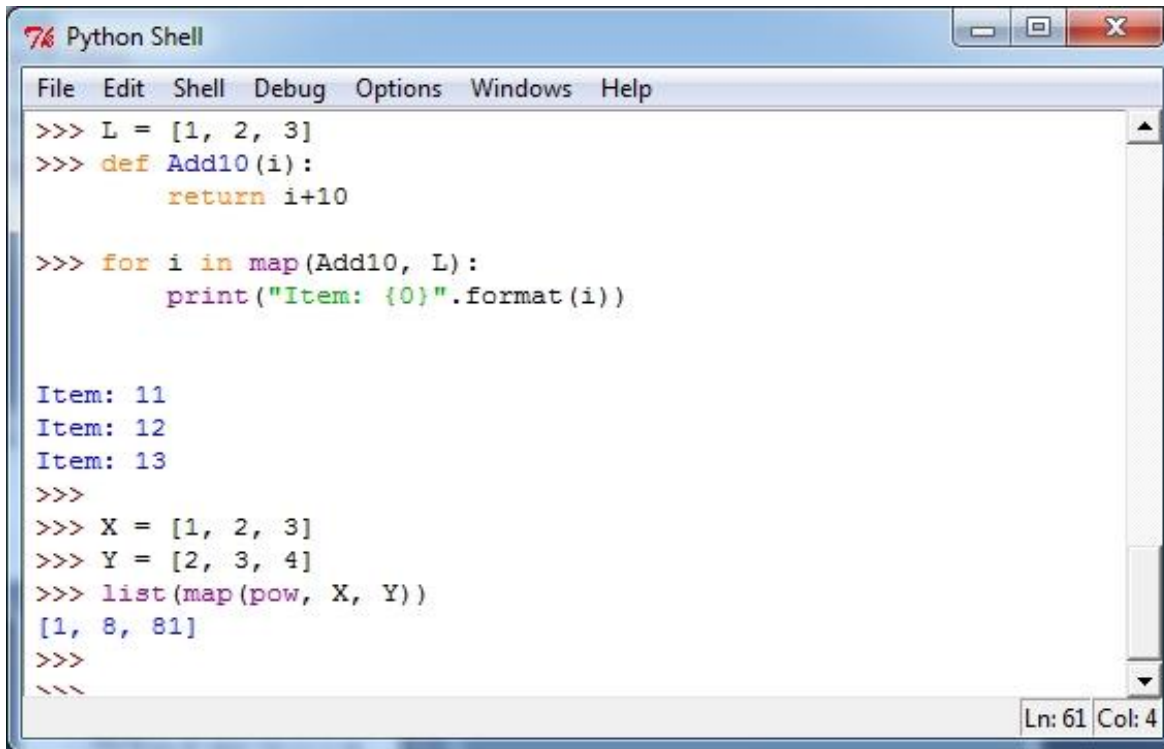
# 1. Python Tip

## 1.4 map



### map

- ❖ map(<function>, 시퀀스 객체, ...)
- ❖ 시퀀스 객체를 순회하며 function의 연산을 수행
- ❖ 함수의 인자수만큼 시퀀스 객체를 전달



```
Python Shell
File Edit Shell Debug Options Windows Help
>>> L = [1, 2, 3]
>>> def Add10(i):
>>>     return i+10
>>> for i in map(Add10, L):
>>>     print("Item: {0}".format(i))

Item: 11
Item: 12
Item: 13
>>>
>>> X = [1, 2, 3]
>>> Y = [2, 3, 4]
>>> list(map(pow, X, Y))
[1, 8, 81]
>>>
>>>
```

## 2. TCP/UDP Socket

---

## 2. TCP/UDP Socket

---

### 2.1 Socket



A socket is a **logical endpoint** of communication

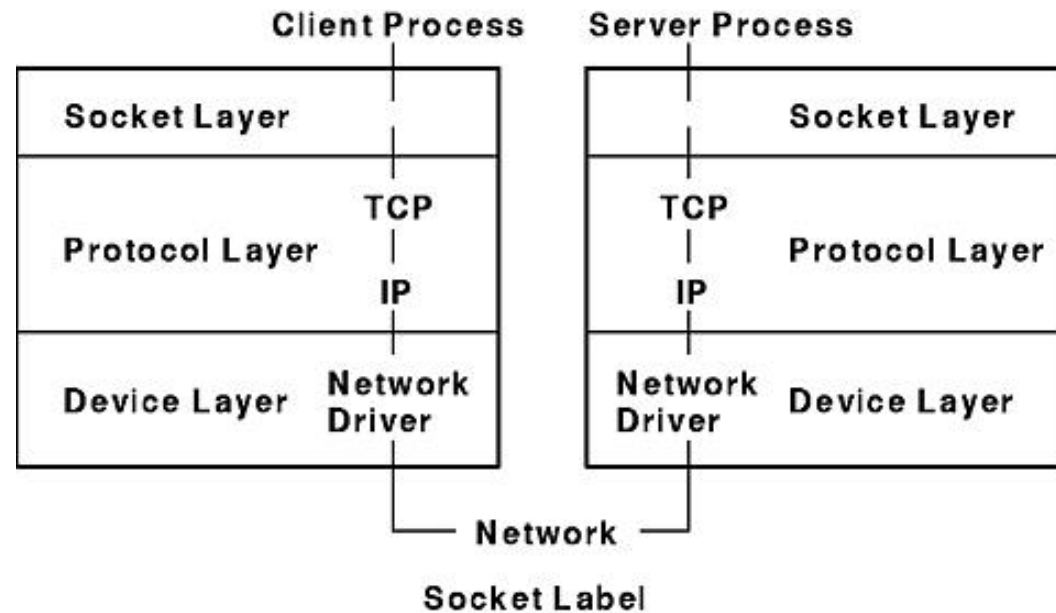


## 2. TCP/UDP Socket

### 2.1 Socket

#### Socket

BSD UNIX의 file을 통한 IPC(Inter Process Communication)을 확장해 원격지 호스트의 프로세스와 통신할 수 있도록 만든 인터페이스



## 2. TCP/UDP Socket

### 2.1 Socket

#### Socket의 특징

- Socket 인터페이스 자체가 TCP/IP 표준은 아니다.
- 특정 운영체제 및 언어에 종속적으로 동작
- Server-Client 모델
- 양방향(two-way) 통신 모델

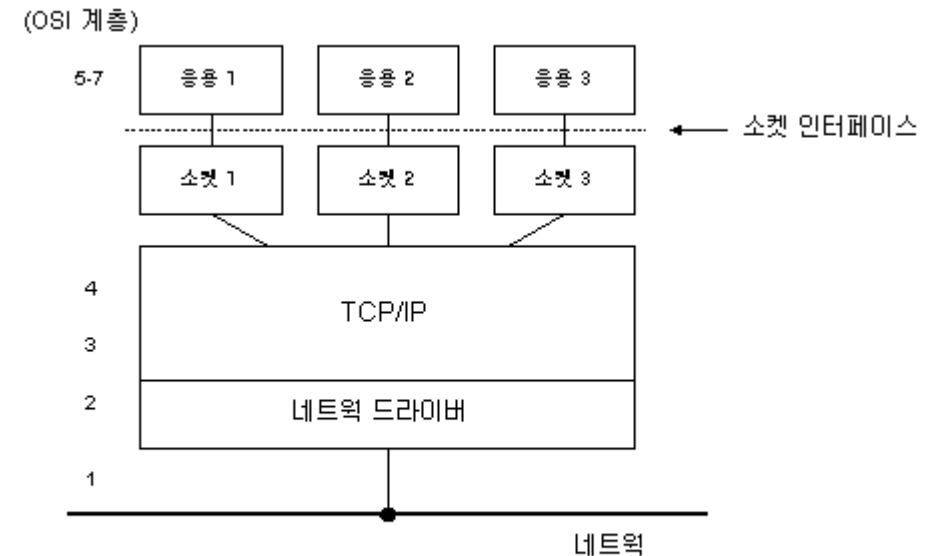


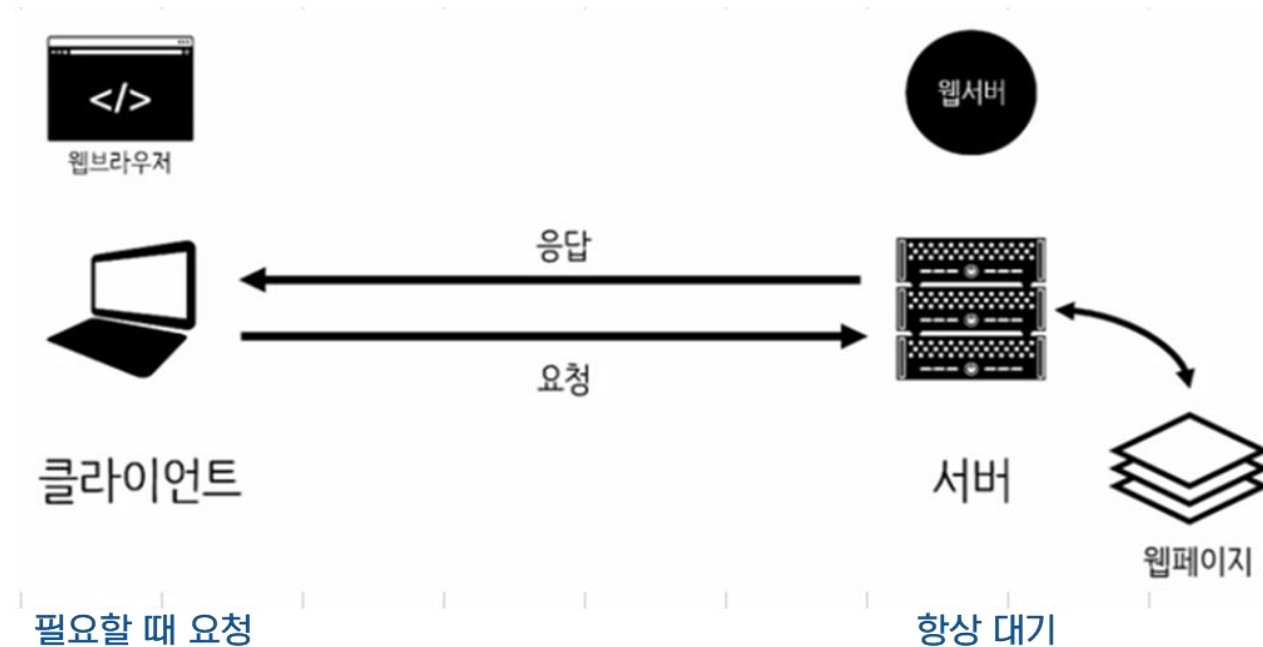
그림 2-1 소켓 인터페이스의 위치

## 2. TCP/UDP Socket

### 2.1 Socket

#### ■ 서버 클라이언트 구조

클라이언트 서버 모델(client-server model)은 서비스 요청자인 클라이언트와 서비스 자원의 제공자인 서버 간에 작업을 분리해주는 분산 애플리케이션 구조이자 네트워크 아키텍처를 나타낸다.

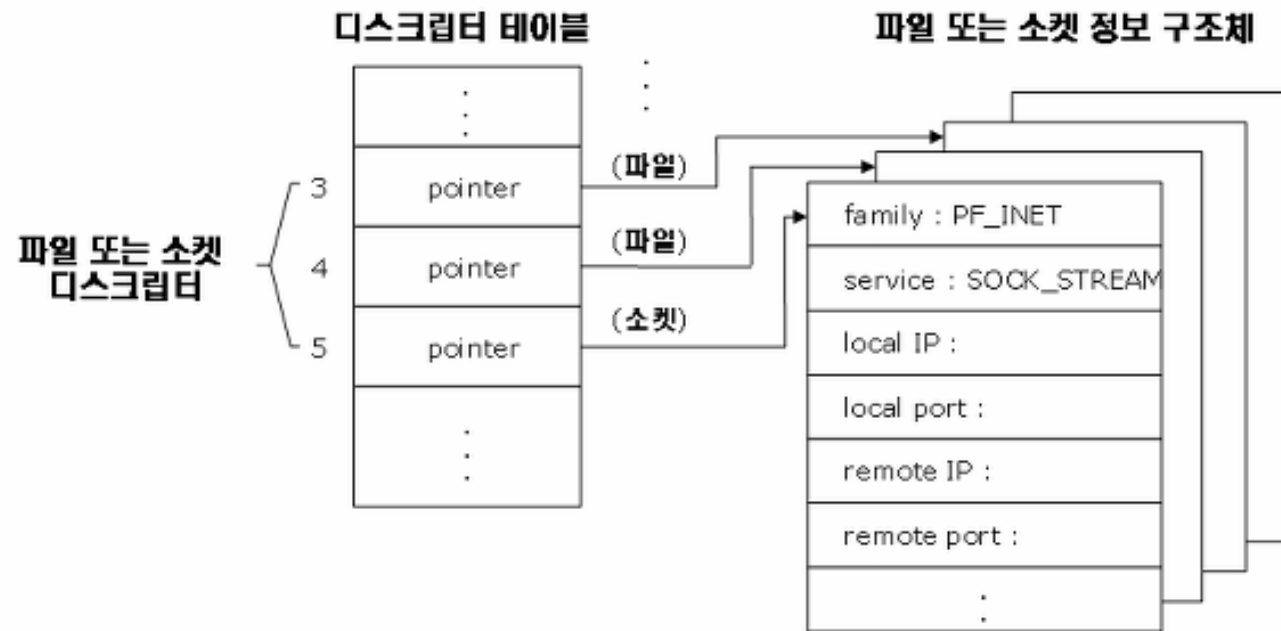


## 2. TCP/UDP Socket

### 2.1 Socket

#### Socket descriptor

File API와 유사하게 socket descriptor를 사용해 접근한다.



## 2. TCP/UDP Socket

### 2.2 TCP/IP

#### TCP/IP Protocol suite

- ❖ 인터넷 표준 프로토콜
- ❖ 계층 구분이 OSI 참조 모델과 정확히 일치하지 않음.
- ❖ TCP, IP 및 UDP, ARP, ICMP, SMTP 등 다양한 프로토콜의 집합체(suite)

TCP/IP 프로토콜 계층

애플리케이션 계층 (Application Layer)
트랜스포트 계층 (Transport Layer)
네트워크 계층 (Network Layer)
네트워크 인터페이스 계층 (Network Interface Layer)

TCP/IP 프로토콜 집합

Telnet	FTP	SMTP	DNS	SNMP
TCP			UDP	
IP, ARP, ICMP, IGMP				
Ethernet	Token Ring	Frame Relay	ATM	

## 2. TCP/UDP Socket

### 2.2 TCP/IP

#### TCP/IP Protocol Layer

- ❖ 네트워크 인터페이스 계층
  - ❖ OSI 7계층의 데이터 링크 계층과 물리 계층에 해당
  - ❖ Ethernet, Token ring, 무선 LAN, X.25, Frame relay...
- ❖ 인터넷 계층
  - ❖ OSI 7계층의 네트워크 계층에 해당
  - ❖ IP, ARP, ICMP, IGMP...
- ❖ 전송 계층
  - ❖ OSI 7계층의 전송 계층에 해당
  - ❖ TCP, UDP
- ❖ 애플리케이션 계층
  - ❖ OSI 7계층의 세션, 표현, 응용 계층에 해당
  - ❖ HTTP, FTP, SMTP, Telnet...

OSI 7 계층		TCP/IP 프로토콜 계층
애플리케이션계층		애플리케이션계층
프레젠테이션계층		
세션계층		
트랜스포트계층		트랜스포트계층
네트워크계층		인터넷계층
데이터링크계층		네트워크인터페이스계층
물리계층		

## 2. TCP/UDP Socket

### 2.3 TCP/IP Address



#### TCP/IP Address

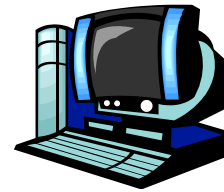
- ❖ 물리 주소
  - ❖ 네트워크 인터페이스 계층에서 사용, 48bit
  - ❖ 네트워크 장비를 구분
- ❖ IP 주소
  - ❖ 인터넷 계층에서 사용, 32bit(IP v4), 64bit(IP v6)
  - ❖ 네트워크 내 호스트를 구분
- ❖ Port 번호
  - ❖ 트랜스포트 계층에서 사용, 16bit
  - ❖ 같은 호스트 내 프로세서를 식별
  - ❖ Well-known port, Dynamic port

## 2. TCP/UDP Socket

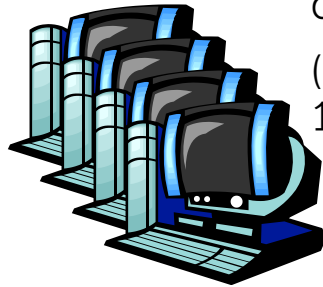
### 2.3 TCP/IP Address



cse.unr.edu  
(134.197.20.22)



newworld.cs.umass.edu  
(128.119.245.93)



cluster.cs.columbia.edu  
(128.59.21.14, 128.59.16.7,  
128.59.16.5, 128.59.16.4)

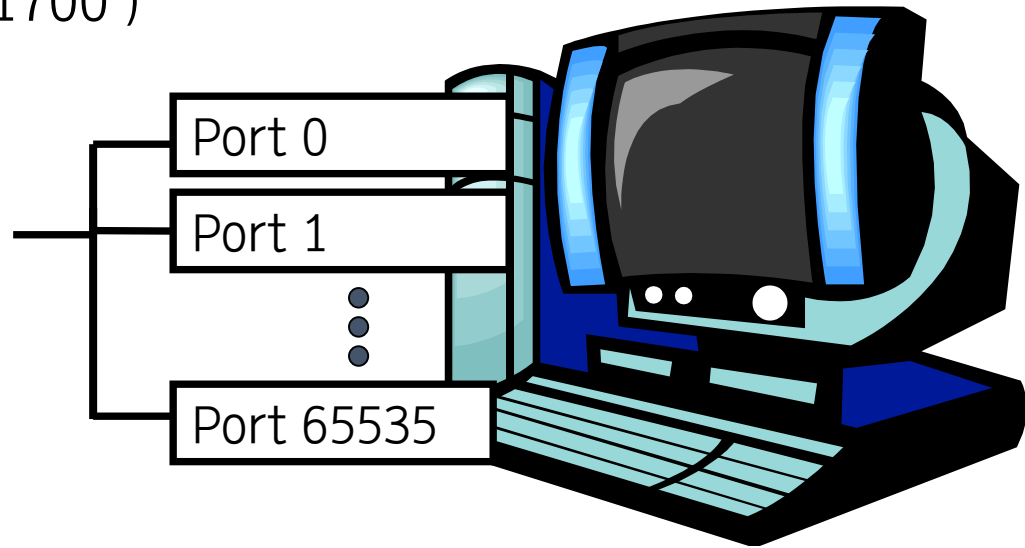
인터넷 내 모든 Host는 IP 주소를 가지고 있다.



## 2. TCP/UDP Socket

### 2.3 TCP/IP Address

- Host별로 65,536개의 포트가 있다.
- 몇몇 포트는 특정한 용도로 예약되어 있다. (RFC 1700 )
  - 22 : SSH
  - 20,21: FTP
  - 23: Telnet
  - 25 : SMTP
  - 53 : DNS
  - 80: HTTP
  - 443 : HTTPS



포트는 호스트가 네트워크로부터 데이터를 읽고/보낼 수 있게 해주는 인터페이스이다.

## 2. TCP/UDP Socket

---

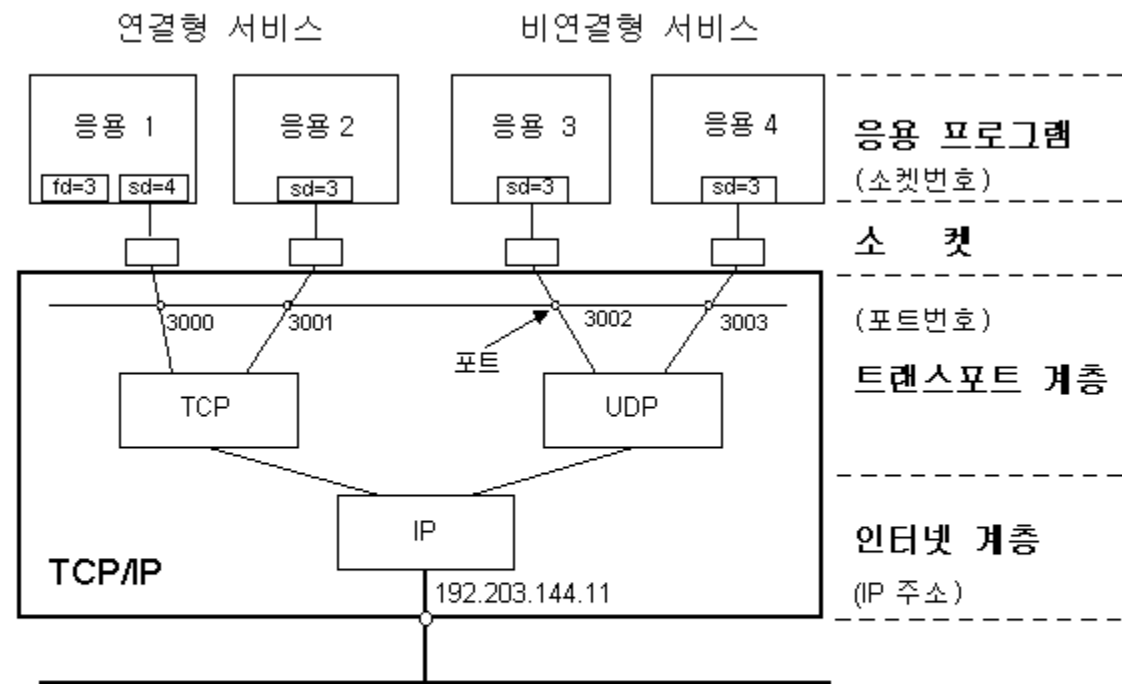
### 2.3 TCP/IP Address

## 통신연관(Association)을 위한 5-Tuple

1. 프로토콜(TCP, UDP)
2. 자신의 IP 주소
3. 자신의 Port 번호
4. 상대방 IP 주소
5. 상대방 Port 번호

## 2. TCP/UDP Socket

### 2.3 TCP/IP Address



IP : 호스트를 식별

Port : 호스트 내 응용을 식별

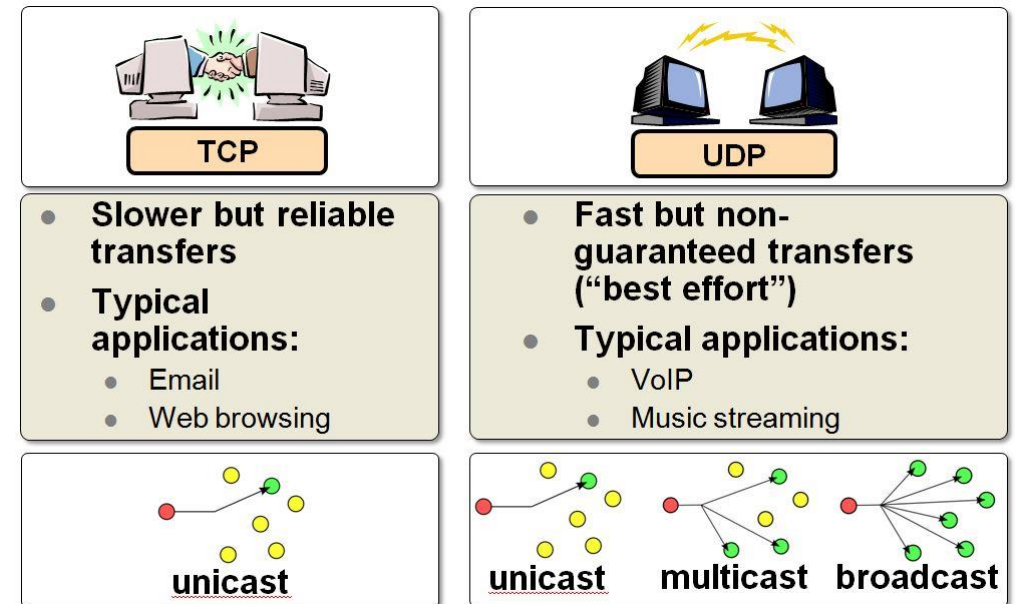
Socket 번호 : 응용 내에서 논리적 연결(5-Tuple) 식별

## 2. TCP/UDP Socket

### 2.4 TCP Socket vs UDP Socket

#### TCP vs UDP

- ❖ TCP(Transmission control protocol)
  - ❖ TCP는 전송계층 프로토콜로 종단 호스트(end-point)간 신뢰성 있는 데이터 전달을 수행
  - ❖ 점대점(point to point) 연결
- ❖ UDP(User datagram protocol)
  - ❖ UDP 역시 전송계층 프로토콜로 종단간 데이터 전달을 수행
  - ❖ TCP와 달리 신뢰성을 보장하지 않는다.

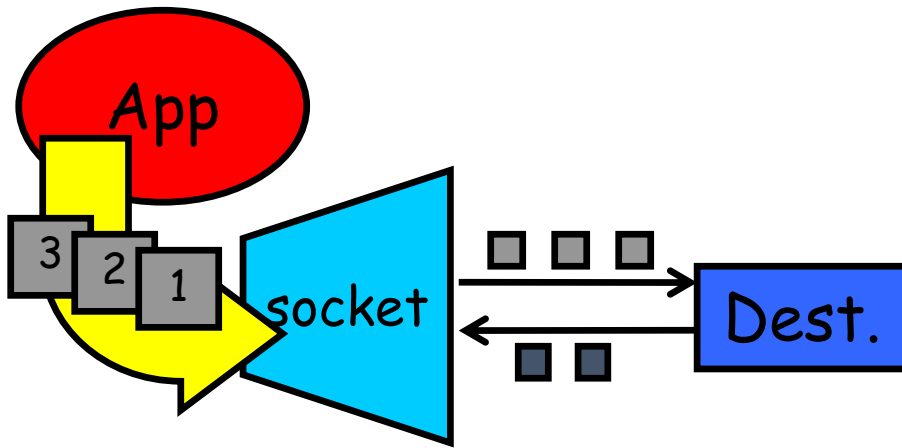


## 2. TCP/UDP Socket

### 2.4 TCP Socket vs UDP Socket

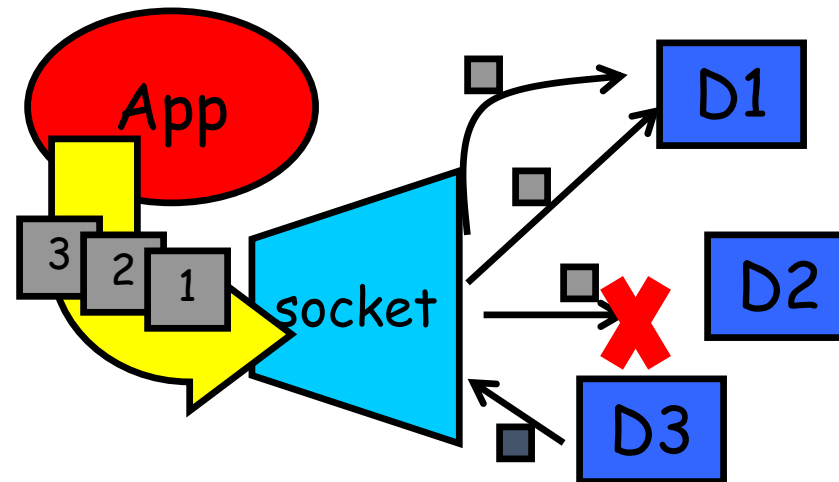
#### SOCK\_STREAM

- ❖ a.k.a. TCP
- ❖ reliable delivery
- ❖ in-order guaranteed
- ❖ connection-oriented
- ❖ bidirectional



#### SOCK\_DGRAM

- ❖ a.k.a. UDP
- ❖ unreliable delivery
- ❖ no order guarantees
- ❖ no notion of “connection” – app indicates dest. for each packet
- ❖ can send or receive

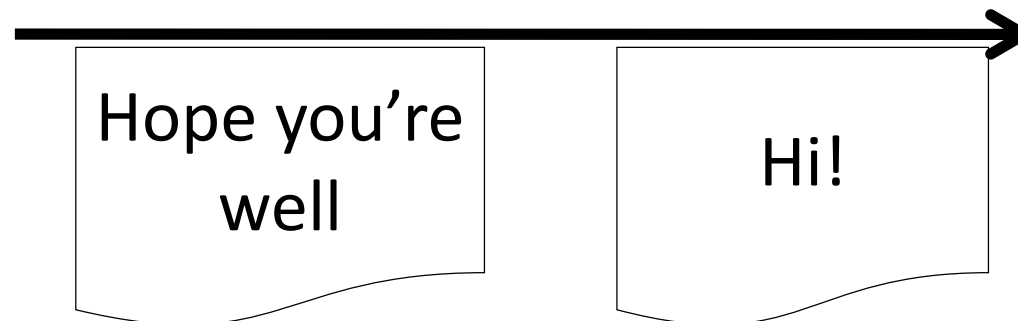
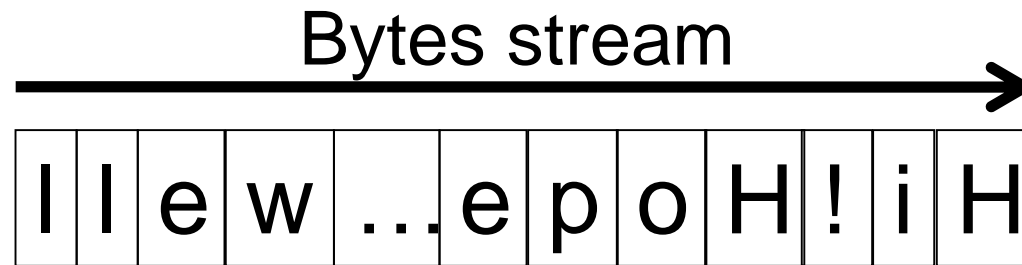


## 2. TCP/UDP Socket

### 2.4 TCP Socket vs UDP Socket

#### 데이터 전송의 차이

- ❖ “Hi!” 와 “Hope you’re well”이라는 문자열을 보낼 경우
- ❖ TCP는 각각을 하나의 스트림으로 취급한다.
- ❖ UDP는 각각을 별개의 메시지로 취급한다.

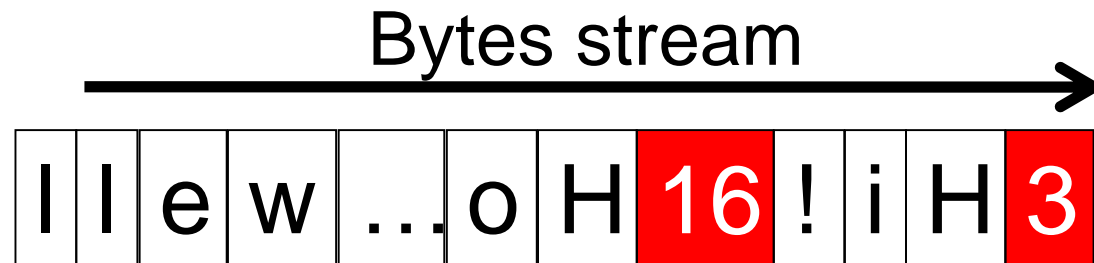


## 2. TCP/UDP Socket

### 2.4 TCP Socket vs UDP Socket

#### 데이터 전송의 차이

- ❖ “Hi!” 와 “Hope you’re well”이라는 문자열을 보낼 경우
- ❖ TCP는 각각을 하나의 스트림으로 취급한다.
- ❖ 때문에 TCP는 상위 계층에서 메시지 길이로 메시지를 구분해야 한다.

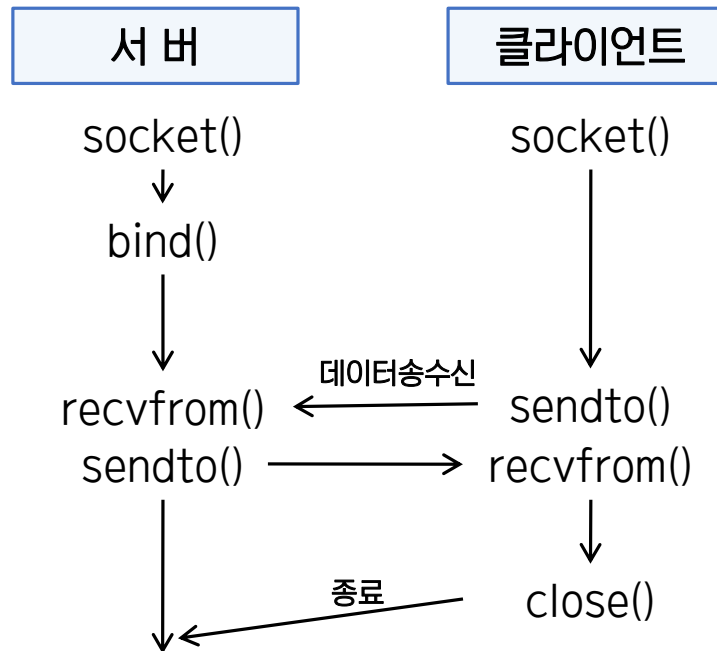


## 2. TCP/UDP Socket

### 2.4 TCP Socket vs UDP Socket

#### ■ UDP Socket API 호출 순서

- ❖ TCP와 달리 1대1 통신에만 사용되지는 않는다
- ❖ socket 생성 후 연결절차 없이 바로 통신가능





## 2. TCP/UDP Socket

### 2.5 UDP Example(Server)

```
from socket import socket, getfqdn, AF_INET, SOCK_DGRAM
s = socket(AF_INET, SOCK_DGRAM)
s.bind(('127.0.0.1', 11111))
print 'server at', getfqdn('')
while True:
    data, addr = s.recvfrom(1024)
    print "Connection from", addr
    s.sendto(data.upper(), addr)
```

Empty -> all interfaces

Note that the *bind()* argument is a two-element tuple of address and port number

## 2. TCP/UDP Socket

---

### 2.5 UDP Example(Client)

```
from socket import socket, AF_INET, SOCK_DGRAM
s = socket(AF_INET, SOCK_DGRAM)
s.bind(('127.0.0.1', 0)) # OS chooses port
server = ('127.0.0.1', 11111)
s.sendto("MixedCaseString", server)
data, addr = s.recvfrom(1024)
print "received", data, "from", addr
s.close()
```

## 2. TCP/UDP Socket

---

### 2.6 Assignment 4

## Assignment #4

- 프로세스와 스레드에 대해서 조사
  - 프로세스와 스레드의 정의 및 차이점
  - 스레드 사용 이유 및 스레드 모델
  - Python에서 프로세스 및 스레드를 사용하는 방법
  - HWP 및 DOC 포맷으로 3장 내외
- 팀 대표가 [barcel@naver.com](mailto:barcel@naver.com)으로 제출 (4.2일까지)
  - Title : [컴퓨터네트워크][학번][이름][과제\_N]
  - Content : github repo url

팀명 : 길동이네

팀원 : 홍길동(학번), 고길동(학번)

# Q & A

---