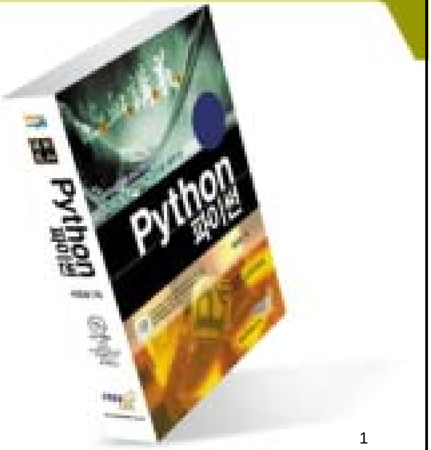
熱血講義

프리렉의 열혈강의 시리즈

Python
파이썬



파이썬 (Python)

열혈강의 Python
파이썬

❖제 17 강

주제 : 소켓 프로그래밍

이강성(gslee@mail.gwu.ac.kr)

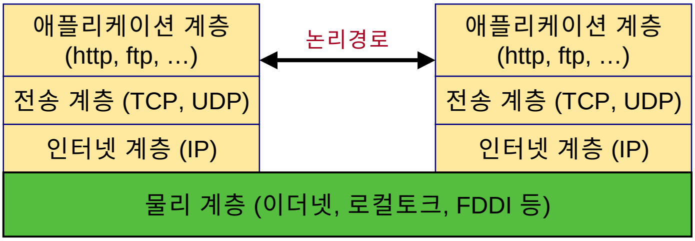
❖강의 목차

1. 소켓 개요
2. 소켓 객체의 기본 메쏘드
3. 소켓 프로그래밍 절차
4. 간단한 소켓 프로그램
5. 소켓 프로그래밍 예제
   1. Finger
   2. ftp
   3. telnet
6. 소켓 서버 프로그래밍
7. HTTP 서버 프로그래밍

소켓 개요

- Socket은 언제 생겼나
  - 1982년 BSD(Berkeley Software Distribution) UNIX 4.1에서 처음 소개
  - 현재 널리 사용되는 것은 1986년의 BSD UNIX 4.3에서 개정된 것
- Socket 이란?
  - TCP/IP 프로토콜의 프로그래머 인터페이스
  - 프로세스간 양방향(bidirection) 통신
  - 프로세스는 동일/다른 컴퓨터에 있어도 됨

네트워크의 계층 구조



애플리케이션 계층 (http, ftp, …)

논리경로

애플리케이션 계층 (http, ftp, …)

전송 계층 (TCP, UDP)

전송 계층 (TCP, UDP)

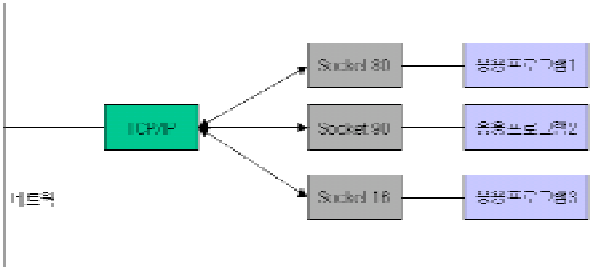
인터넷 계층 (IP)

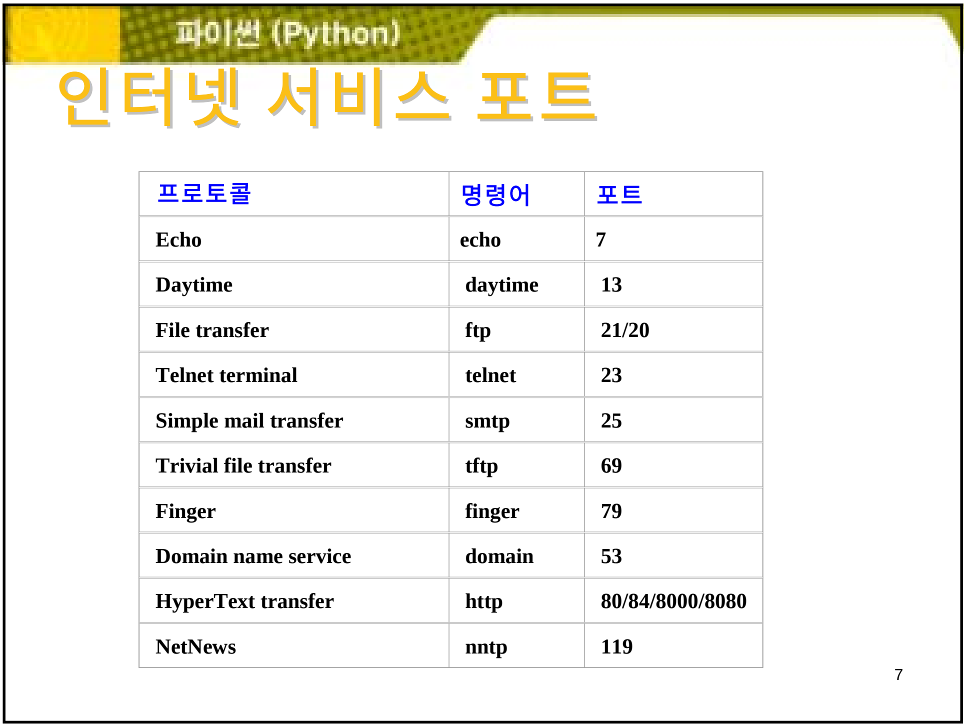
인터넷 계층 (IP)

물리 계층 (이더넷, 로컬토크, FDDI 등)

포트번호

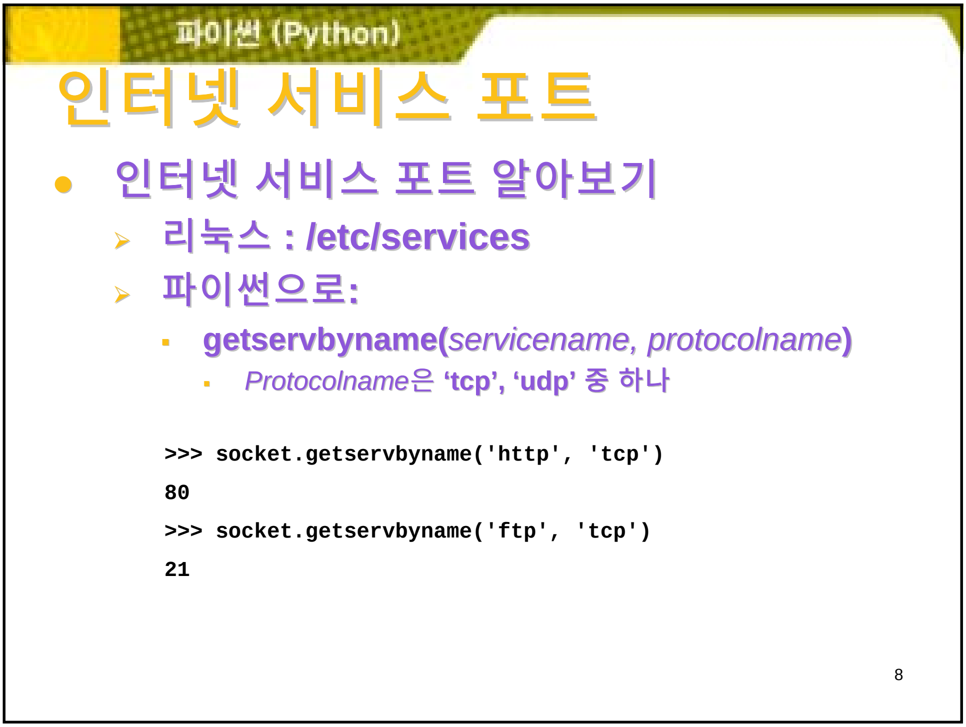
- 하나의 전송선에 여러 개의 응용 프로그램을 실행하기 위해 포트 개념 도입
- 포트 번호 : 0 ~ 65535
  - 잘 알려진 서비스 : 1~255
  - 그 밖의 예약 : 256~1023
  - 임시 클라이언트 : 1024~4999





인터넷 서비스 포트

프로토콜	명령어	포트
Echo	echo	7
Daytime	daytime	13
File transfer	ftp	21/20
Telnet terminal	telnet	23
Simple mail transfer	smtp	25
Trivial file transfer	tftp	69
Finger	finger	79
Domain name service	domain	53
HyperText transfer	http	80/84/8000/8080
NetNews	nntp	119



인터넷 서비스 포트

- 인터넷 서비스 포트 알아보기
  - 리눅스 : /etc/services
  - 파이썬으로:
    - **getservbyname(***servicename, protocolname***)**
      - *Protocolname*은 **'tcp', 'udp'** 중 하나

```
>>> socket.getservbyname('http', 'tcp')
80
>>> socket.getservbyname('ftp', 'tcp')
21
```

소켓의 종류

- Domain
  - AF_UNIX : c/s는 동일 기계에 존재
    - 유닉스 도메인 소켓
  - AF_INET : c/s는 다른 기계에 존재할 수 있다(IP 소켓)
- Type
  - SOCK_STREAM : TCP 소켓
  - SOCK_DGRAM : UDP 소켓

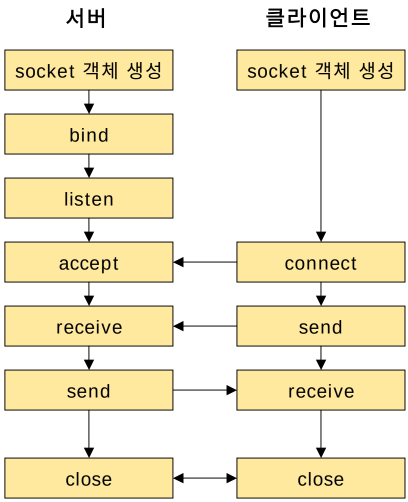
소켓 객체 기본 메쏘드

- 서버측
- 클래스 socket()
  - 소켓 객체를 생성한다.
- 소켓 객체에 대한 메써드들
  - bind((host, port))
  - listen(n)
  - accept()
  - recv(bufsize)
  - send(string)
  - close()

소켓 객체 기본 메쏘드

- 클라이언트 측
- 클래스 socket()
  - 소켓 객체를 생성한다.
- 소켓 객체에 대한 메써드들
  - connect((host, port))
  - recv(bufsize)
  - send(string)
  - close()

소켓 프로그래밍 절차



서버

클라이언트

socket 객체 생성

bind

listen

accept

receive

send

close

socket 객체 생성

connect

send

receive

close

간단한 소켓 프로그램 - 1

- 대화적으로 소켓 서버와 클라이언트를 생성하고 연결하는 예제

소켓의 동작 모드

- 블록킹 모드
  - accept, recv, send 등을 호출 했을 때, 연결하고자 하는 대상이 없거나 데이터를 즉시 쓸 수 없을 때 처리될 때 까지 대기
- 비블록킹 모드
  - 반대 상황. error 예외 발생
- 설정
  - setblocking(*flag*)
  - 0 – nonblocking, 1 – blocking

간단한 소켓 프로그램 - 2

- Daytime 예제
  - telnet으로 ...
  - 소켓 프로그램으로...

- Echo 예제
  - telnet으로 ...

소켓 프로그래밍 예제

- Echo 서버

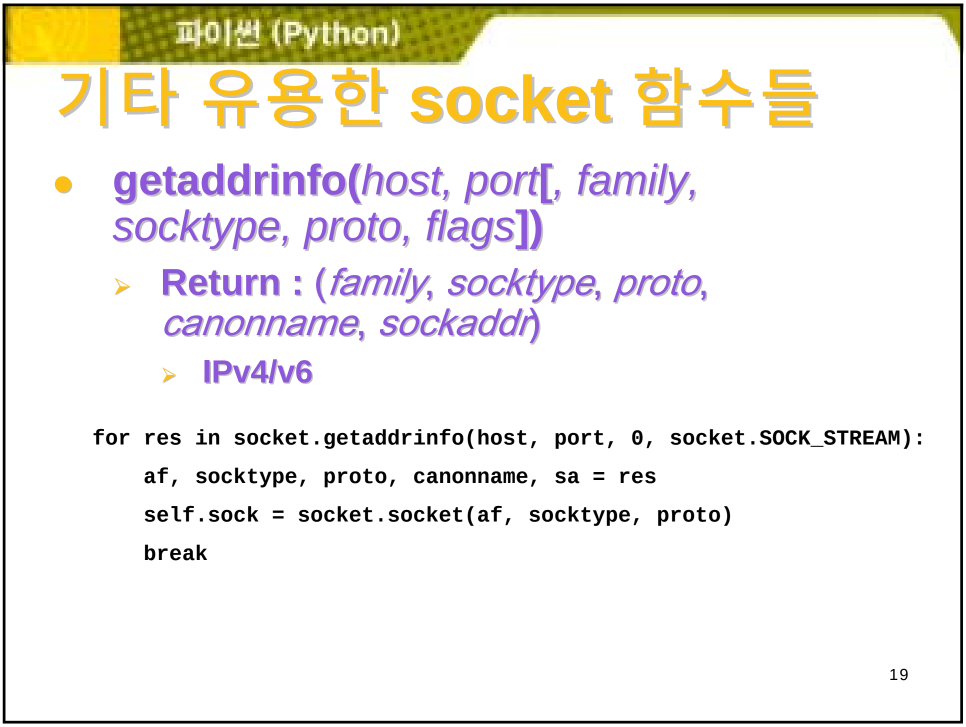
```
# echoserver.py 에코 - 서버 프로그램
from socket import *
HOST = ''           # localhost를 의미하는 심볼릭 이름
PORT = 50007        # 임의의 사용가능 한 안쓰는 포트
s = socket(AF_INET, SOCK_STREAM)
s.bind((HOST, PORT))
s.listen(1)
conn, addr = s.accept()
print 'Connected by', addr
while 1:
    data = conn.recv(1024)
    if not data: break
    conn.send(data)
conn.close()
```

- **Echo 클라이언트**

```
# echoclient.py 에코 - 클라이언트 프로그램
from socket import *
HOST = 'localhost'    # host name 현재는 한 컴퓨터 내에서 사용
PORT = 50007          # 서버와 같은 포트번호를 써야 함
s = socket(AF_INET, SOCK_STREAM)
s.connect((HOST, PORT))
s.send('Hello, world')
data = s.recv(1024)
s.close()
print 'Received', `data`
```

소켓 프로그래밍 예제

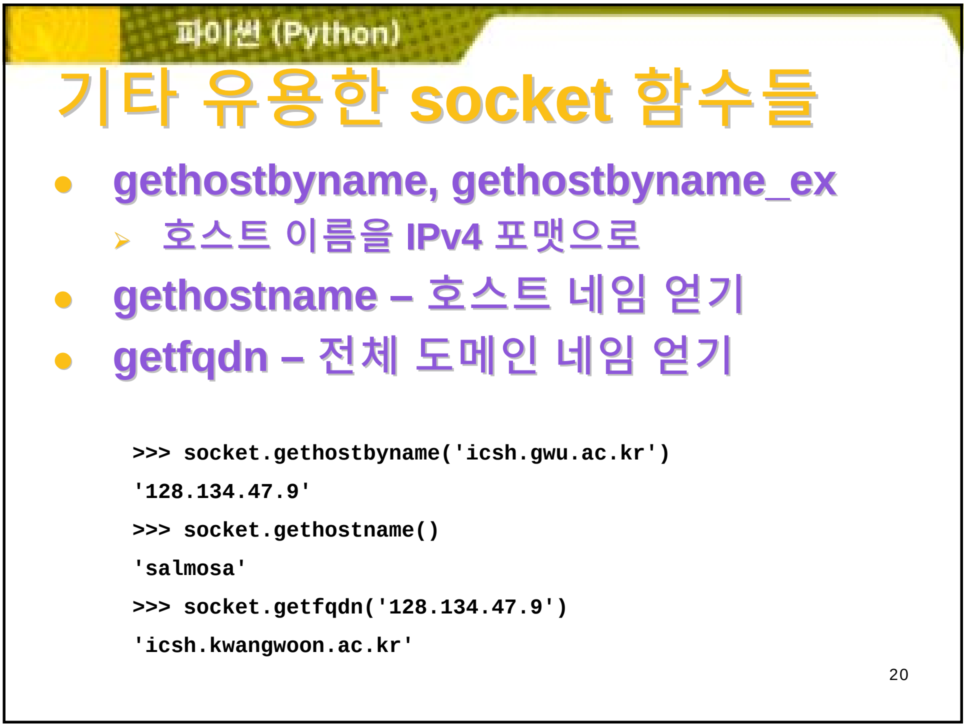
- **finger 서버 / 클라이언트**



기타 유용한 socket 함수들

- **getaddrinfo(***host, port***[***, family, socktype, proto, flags***])**
  - **Return : (***family, socktype, proto, canonname, sockaddr***)**
    - **IPv4/v6**

```
for res in socket.getaddrinfo(host, port, 0, socket.SOCK_STREAM):
    af, socktype, proto, canonname, sa = res
    self.sock = socket.socket(af, socktype, proto)
    break
```



기타 유용한 socket 함수들

- **gethostbyname, gethostbyname_ex**
  - 호스트 이름을 **IPv4** 포맷으로
- **gethostname – 호스트 네임 얻기**
- **getfqdn – 전체 도메인 네임 얻기**

```
>>> socket.gethostbyname('icsh.gwu.ac.kr')
'128.134.47.9'
>>> socket.gethostname()
'salmosa'
>>> socket.getfqdn('128.134.47.9')
'icsh.kwangwoon.ac.kr'
```

기타 유용한 socket 함수들

- makefile([*mode*[, *bufsize*]])
  - 소켓에 관련된 파일 객체 생성
  - *mode*는 파일 오픈할 때의 모드와 동일
  - 사용이 끝났으면 close() 한다.
- ssl(Secure Sockets Layer) 클래스 지원
- 기타 자세한 내용은 Library reference 7.2절 socket 모듈 참조

ftp 클라이언트

- ftp 프로토콜
  - 2중 채널 프로토콜
    - Control connection
    - Data connection
  - 기본 포트 21 / 20
  - 규약 : RFC 959
- ftplib 모듈
  - 고급 메쏘드 제공
  - 기본 모드
    - Passive

ftplib 모듈 주요 메써드

- 접속관련
  - **class FTP([***host***[,** *user***[,** *passwd***[,** *acct***]]]])**
  - **connect(***host***[,** *port***])**
  - **login([***user***[,** *passwd***[,** *acct***]]])**
  - **close()**
- 목록 얻기
  - **dir(***argument***[, ...])**
  - **nlst(***argument***[, ...])**
- 다운로드 / 저장
  - **retrlines(***command***[,** *callback***])**
  - **retrbinary(***command, callback***[,** *maxblocksize***[,** *rest***]])**
  - **storlines(***command, file***)**
  - **storbinary(***command, file***[,** *blocksize***])**
  - **abort()**

ftp 클라이언트 주요 메써드

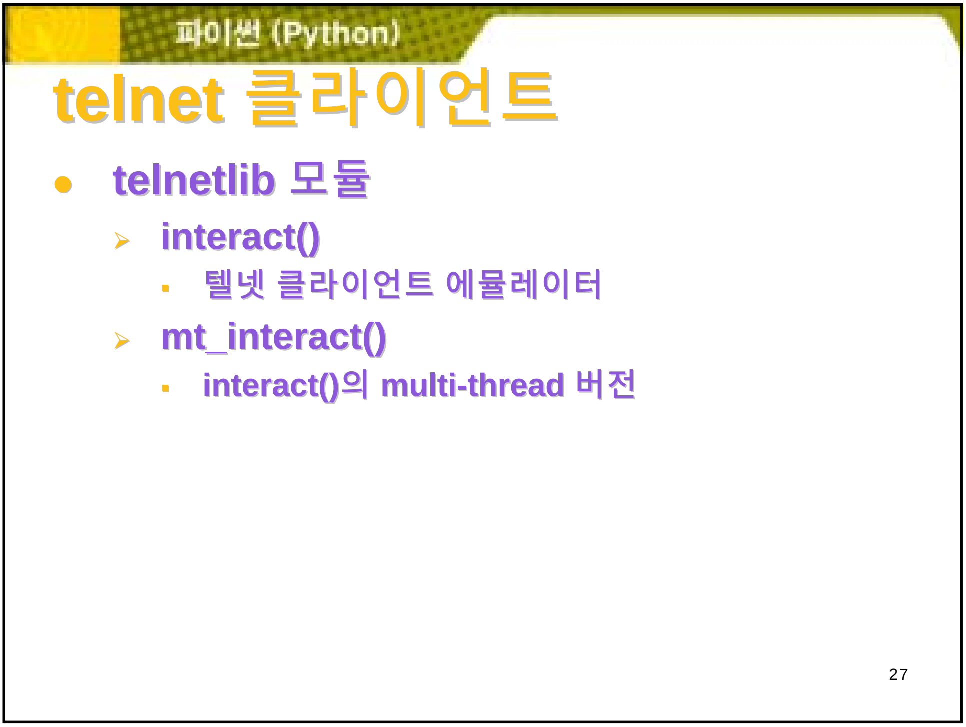
- 파일 관련
  - **size(***filename***)**
  - **rename(***fromname, toname***)**
  - **delete(***filename***)**
- 디렉토리 관련
  - **pwd()**
  - **cwd(***pathname***)**
  - **mkd(***pathname***)**
  - **rmd(***dirname***)**
- 기타
  - **set_pasv(***boolean***)**
  - **sendcmd(***command***)**
  - **voidcmd(***command***)**
  - **…**

ftp 클라이언트

- 파일 가져오기 예제
  - ftpsample.py

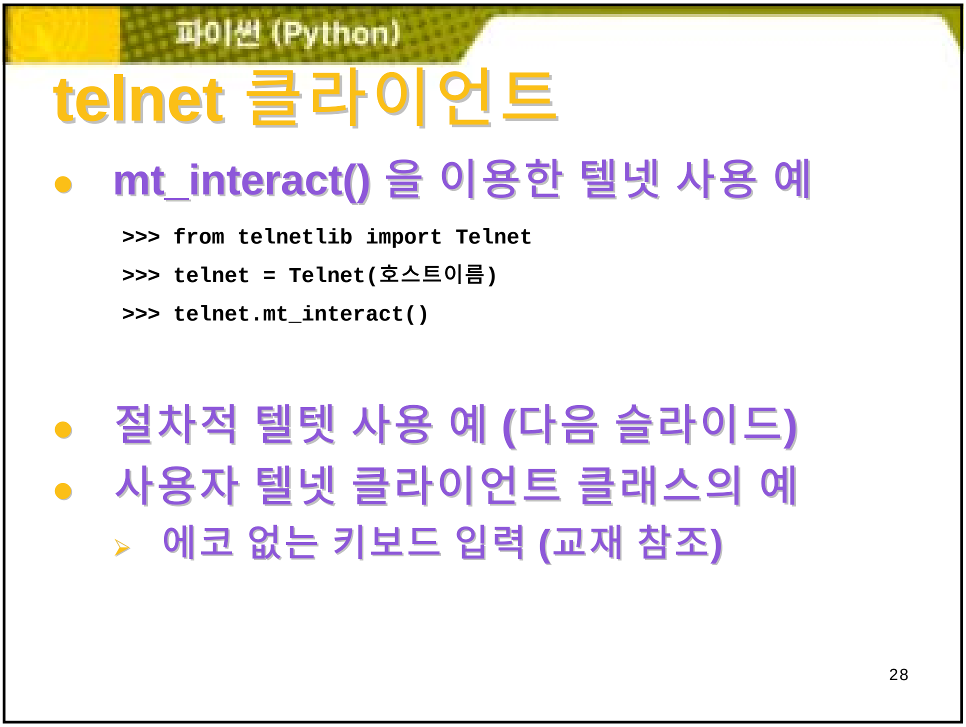
telnet 클라이언트

- telnetlib 모듈
  - **class Telnet([*host*[, *port*]])**
  - **open(*host*[, *port*]) / close()**
  - **Read 관련 메써드들**
    - **read_until(*expected*[, *timeout*])**
      - 지정 문자열을 읽을 때 까지 읽는다
    - **read_all()**
      - 연결이 끊어질 때 까지 전체를 읽는다(읽기만 함)
    - **read_eager() 등**
      - 당장 읽을 수 있는 문자열을 읽는다
    - **expect(*list*[, *timeout*])**
      - 정규식 중 하나가 매치될 때 까지 읽는다
  - **write(*buffer*)**



telnet 클라이언트

- telnetlib 모듈
  - interact()
    - 텔넷 클라이언트 에뮬레이터
  - mt_interact()
    - interact()의 multi-thread 버전



telnet 클라이언트

- mt_interact() 을 이용한 텔넷 사용 예

```
>>> from telnetlib import Telnet
>>> telnet = Telnet(호스트이름)
>>> telnet.mt_interact()
```

- 절차적 텔텟 사용 예 (다음 슬라이드)
- 사용자 텔넷 클라이언트 클래스의 예
  - 에코 없는 키보드 입력 (교재 참조)

```
# telnetclient.py
from telnetlib import Telnet
import getpass

def telnetClient(host):
    telnet = Telnet(host)
    telnet.read_until('login:')
    user = raw_input("Enter your remote account: ")
    telnet.write(user+'\n')   # 사용자 이름 보낸다
    telnet.read_until('Password:')
    password = getpass.getpass()     # 패스워드를 받고
    telnet.write(password+'\n')             # 보낸다
    cmd = raw_input("Enter your command: ")
    telnet.write(cmd+'\n')    # 명령을 보낸다.
    telnet.write('exit\n')    # 종료한다.
    return telnet.read_all()  # 모든 메시지를 출력한다.

if __name__ == '__main__':
    r = telnetClient('daisy')
    print r
```

소켓 서버 프로그래밍

- **SocketServer 모듈**
- 소켓 서버 클래스
  - 소켓의 유형과 서비스 방법에 따른 객체 생성
- 리퀘스트 핸들러 클래스
  - 클라이언트로부터의 요구를 서비스 해주는 객체 생성

소켓 서버 프로그래밍

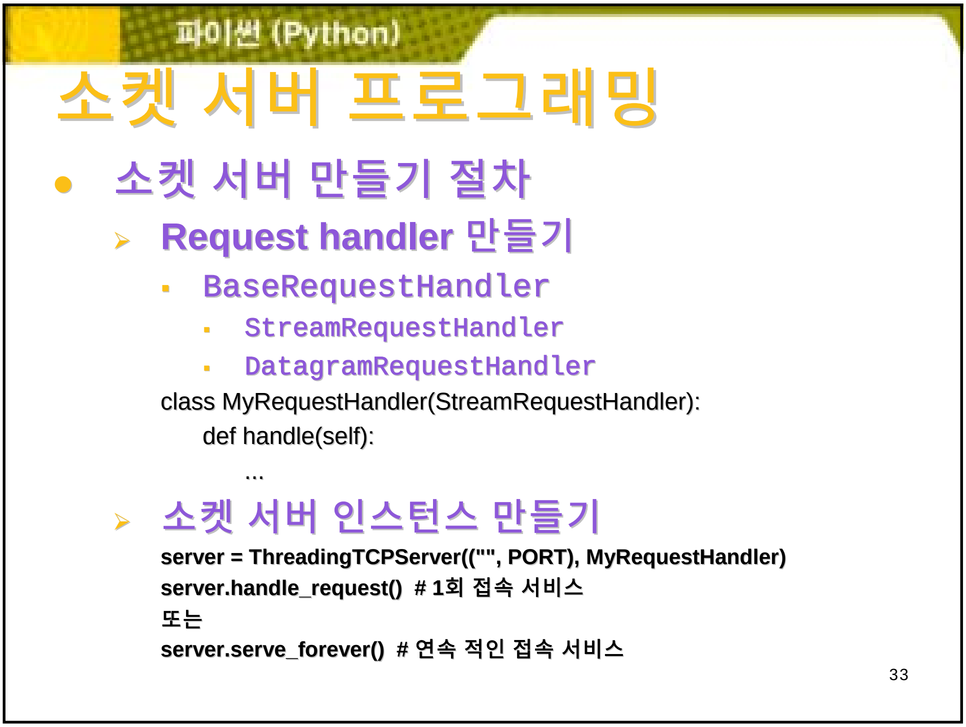
- 소켓 서버의 유형과 서비스 방법

소켓유형	도메인	AF_INET (IP 소켓)	AF_UNIX (유닉스 소켓)
SOCK_STREAM (TCP)	Synchronous	TCPServer	UnixStreamServer
	Forking	ForkingTCPServer	
	Threading	ThreadingTCPServer	ThreadingUnixStreamServer
SOCK_DGRAM (UDP)	Synchronous	UDPServer	UnixDatagramServer
	Forking	ForkingUDPServer	
	Threading	ThreadingUDPServer	ThreadingUnixDatagramServer

소켓 서버 프로그래밍

- 소켓 유형에 따른 리퀘스트 핸들러

소켓 유형	해당 핸들러
SOCK_STREAM (TCP)	StreamRequestHandler
SOCK_DGRAM (UDP)	DatagramRequestHandler



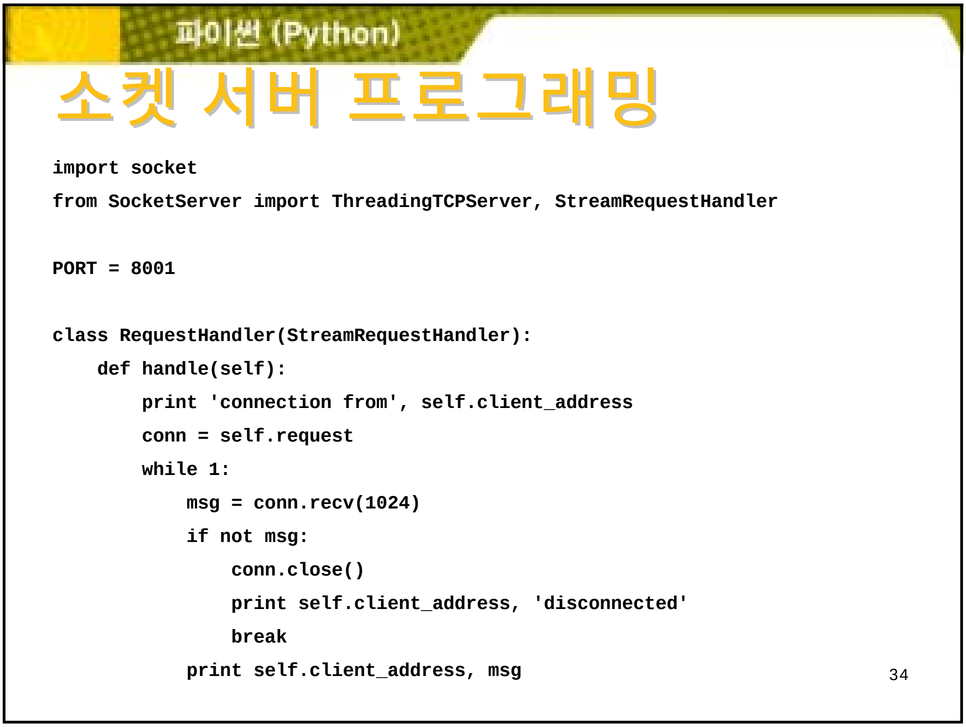
소켓 서버 프로그래밍

- 소켓 서버 만들기 절차
  - Request handler 만들기
    - BaseRequestHandler
      - StreamRequestHandler
      - DatagramRequestHandler

```
class MyRequestHandler(StreamRequestHandler):
    def handle(self):
        ...
```

  - 소켓 서버 인스턴스 만들기

```
server = ThreadingTCPServer(("", PORT), MyRequestHandler)
server.handle_request()  # 1회 접속 서비스
또는
server.serve_forever()  # 연속 적인 접속 서비스
```



소켓 서버 프로그래밍

```
import socket
from SocketServer import ThreadingTCPServer, StreamRequestHandler

PORT = 8001

class RequestHandler(StreamRequestHandler):
    def handle(self):
        print 'connection from', self.client_address
        conn = self.request
        while 1:
            msg = conn.recv(1024)
            if not msg:
                conn.close()
                print self.client_address, 'disconnected'
                break
            print self.client_address, msg
```

소켓 서버 프로그래밍

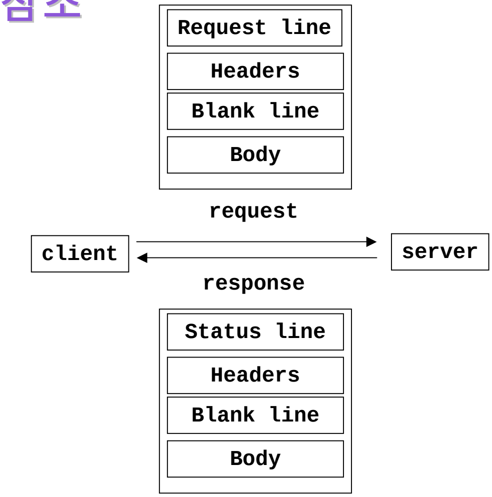
```
if __name__ == '__main__':
    server = ThreadingTCPServer(('', PORT), RequestHandler)
    print 'listening on port', PORT
    server.serve_forever()
```

소켓 서버 프로그래밍

- 채팅 서버
    - 교재 참조

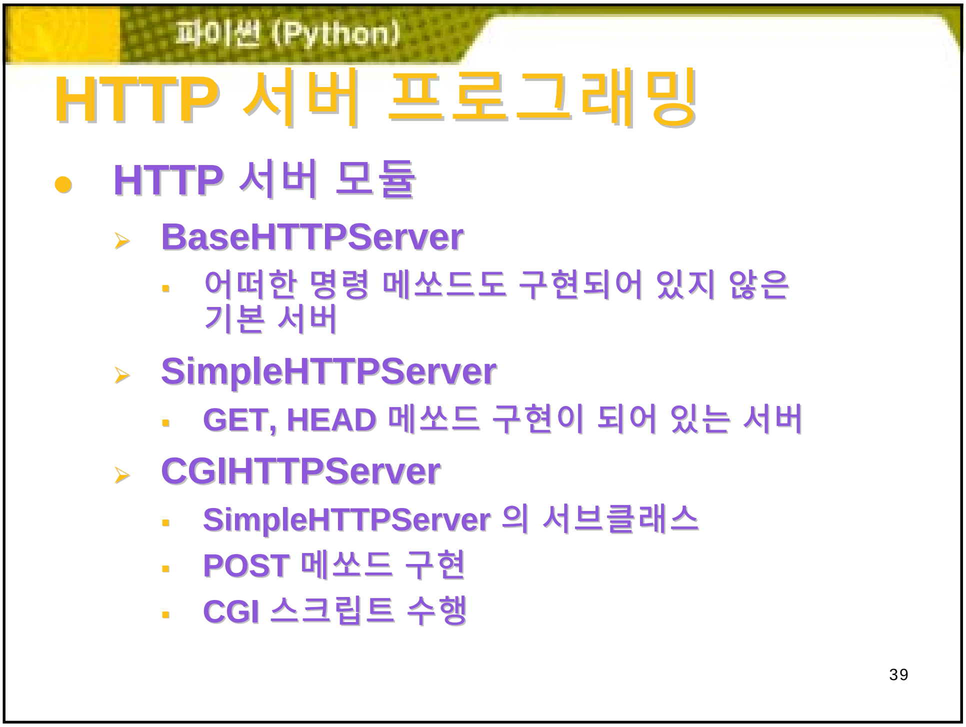
HTTP 서버 프로그래밍

- HTTP에 관한 개괄은 15장 httplib 모듈 강의 참조



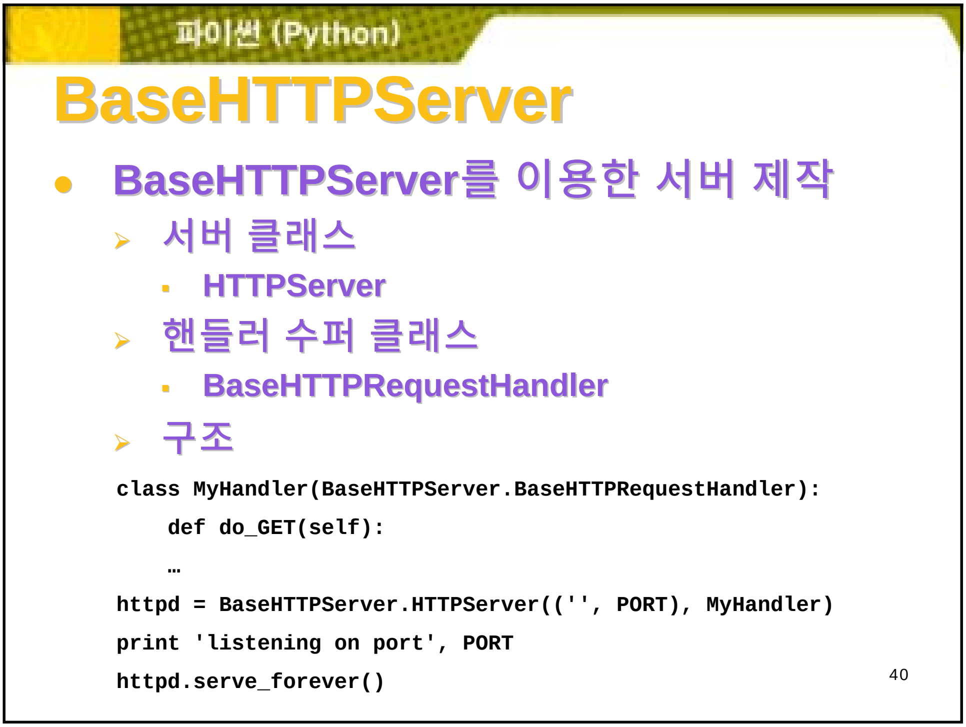
HTTP 서버 프로그래밍

- …HTTPServer가 하는 일?
  - TCP 포트 오픈하고 listen
  - 연결이 들어오면 쓰레드 생성하고
  - 요구(request) 메시지를 읽어들이고 분석
  - 적절한 메소드를 호출해줌
- 메쏘드 형식
  - do_메쏘드명()



HTTP 서버 프로그래밍

- HTTP 서버 모듈
  - BaseHTTPServer
    - 어떠한 명령 메쏘드도 구현되어 있지 않은 기본 서버
  - SimpleHTTPServer
    - GET, HEAD 메쏘드 구현이 되어 있는 서버
  - CGIHTTPServer
    - SimpleHTTPServer 의 서브클래스
    - POST 메쏘드 구현
    - CGI 스크립트 수행



BaseHTTPServer

- BaseHTTPServer를 이용한 서버 제작
  - 서버 클래스
    - HTTPServer
  - 핸들러 수퍼 클래스
    - BaseHTTPRequestHandler
  - 구조

```
class MyHandler(BaseHTTPServer.BaseHTTPRequestHandler):
    def do_GET(self):
    …
httpd = BaseHTTPServer.HTTPServer(('', PORT), MyHandler)
print 'listening on port', PORT
httpd.serve_forever()
```

BaseHTTPServer

- BaseHTTPRequestHandler
  - client_address
  - command – 요구 형태(GET, POST..)
  - path – 요구 경로
  - headers – 헤더 정보
  - rfile – 입력 스트림
  - wfile – 출력 스트림

BaseHTTPServer

- BaseHTTPRequestHandler
  - 요구 형태에 따른 메쏘드 정의
    - def do_요구형태(self):
  - 헤더 정보 읽어내기

```
for key in self.headers.keys():
        print key, self.headers[key]
```

  - 데이터 읽어내기
    - self.rfile.read()

BaseHTTPServer

- BaseHTTPRequestHandler
  - 응답 메시지 보내기
    - send_error(*code*[, *message*])
      - 완전한 에러 메시지 전송
    - send_response(*code*[, *message*])
      - 응답 라인, Server 및 Date 헤더 전송
    - 헤더 전송
      - send_header(*keyword*, *value*)
      - end_headers()
    - Body 전송
      - self.wfile.write(…)

BaseHTTPServer

- 예제
  - httpserver01.py

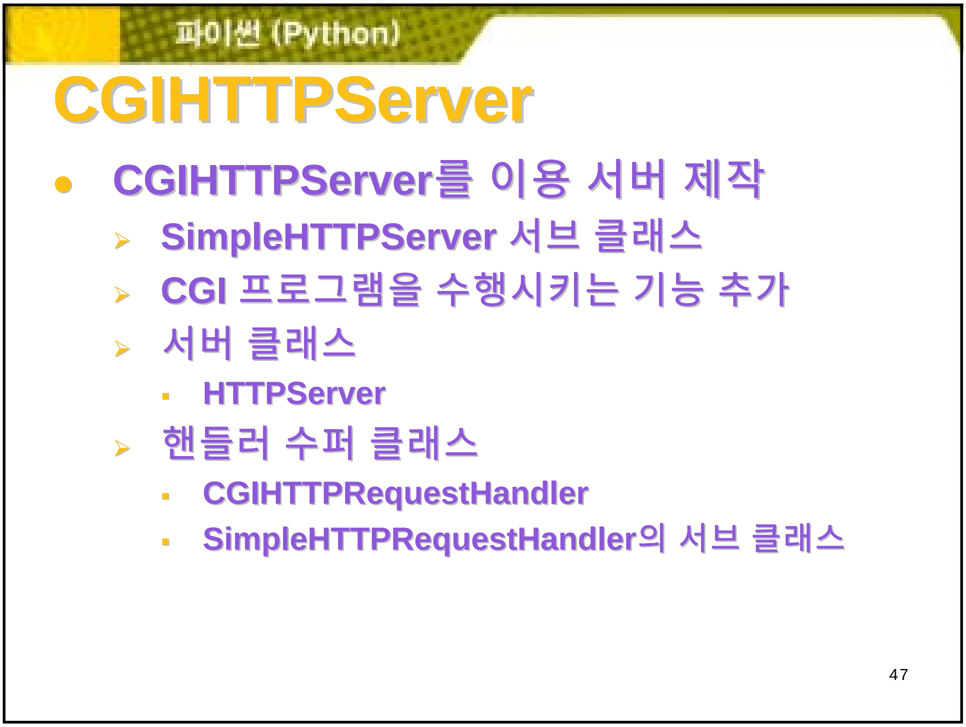
SimpleHTTPServer

- SimpleHTTPServer를 이용 서버 제작
  - GET, HEAD 요구 형태가 구현되어 있음
  - 서버 클래스
    - HTTPServer
  - 핸들러 수퍼 클래스
    - SimpleHTTPRequestHandler
    - BaseHTTPRequestHandler의 서브클래스

SimpleHTTPServer

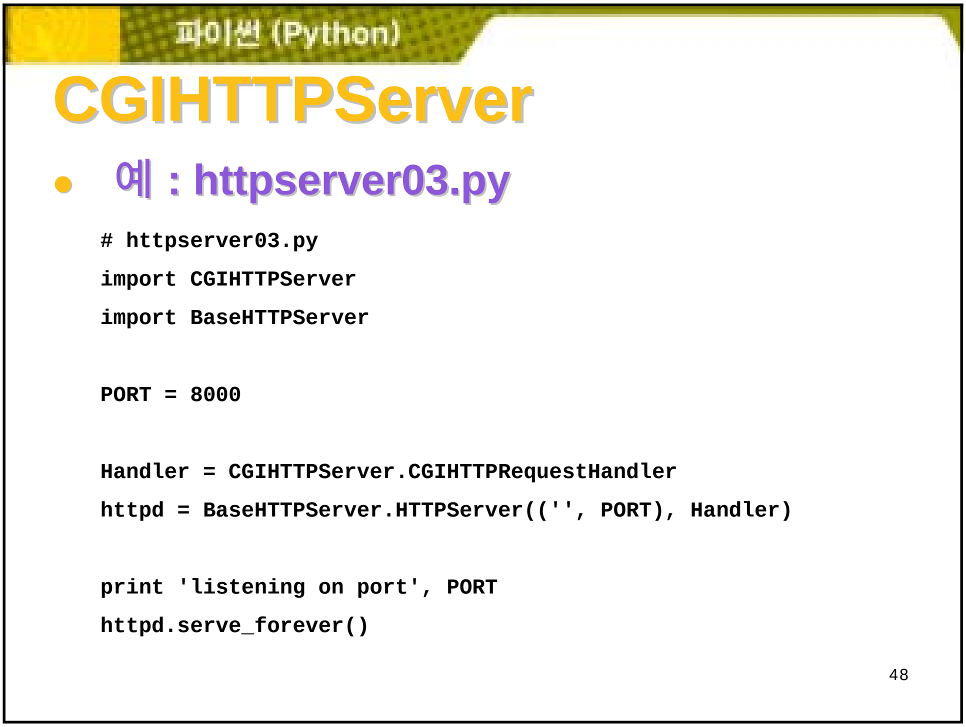
- 예(httpserver02.py)

```
# httpserver02.py
import SimpleHTTPServer
import BaseHTTPServer

PORT = 8000

Handler = SimpleHTTPServer.SimpleHTTPRequestHandler
httpd = BaseHTTPServer.HTTPServer(('', PORT), Handler)

print 'listening on port', PORT
httpd.serve_forever()
```



CGIHTTPServer

- CGIHTTPServer를 이용 서버 제작
  - SimpleHTTPServer 서브 클래스
  - CGI 프로그램을 수행시키는 기능 추가
  - 서버 클래스
    - HTTPServer
  - 핸들러 수퍼 클래스
    - CGIHTTPRequestHandler
    - SimpleHTTPRequestHandler의 서브 클래스



CGIHTTPServer

- 예 : httpserver03.py

```
# httpserver03.py
import CGIHTTPServer
import BaseHTTPServer

PORT = 8000

Handler = CGIHTTPServer.CGIHTTPRequestHandler
httpd = BaseHTTPServer.HTTPServer(('', PORT), Handler)

print 'listening on port', PORT
httpd.serve_forever()
```

CGIHTTPServer

- CGI 프로그램의 수행
  - CGI 스크립트 디렉토리에 있는 파일들
  - CGIHTTPServer.
    CGIHTTPRequestHandler안의 클래스 멤버
    - cgi_directories = ['/cgi-bin', '/htbin']
- 실행 예..