N3C
파이콘 튜토리얼 진행후기
네트워크 프로그래밍 개념 맛보기

하재승 (@ipkn, ipknhama@gmail.com)

Who is ipkn?

개발 덕후, C++, Python 위주지만 자바빼곤 다 함

http://github.com/ipkn/crow

왜 튜토리얼 발표를 했나?

직접 해보는게 강연보다 더 많이 배울 수 있다고 생각함.

발표를 진행하는 기준: **단짠단짠**

NDC ZERO
- 심하게 짠 맛
- 발표자 일부는 뭘 들었는지 모를지도

파이콘 기조
- “Back to the Basic”

대상

파이썬 기본 문법은 이해하지만

네트워크는 다뤄본 적 없는 프로그래머

혼자 네트워크 프로그래밍을 하려고 하면
정말 혼돈과 파괴

실습 1-1 (쉘): 클라이언트 코드

```
import socket
s = socket.socket()
s.connect((‘ipkn.me’, 10000))
s.send(b‘Hello socket’)
s.send(‘안녕 세계!’.encode())
```

예

실습 2: 서버 뼈대 코드

```
import socket
s = socket.socket()
s.bind((‘0.0.0.0’, 10001))
s.listen(5)
while True:
    cs, ca = s.accept()
    f = cs.makefile(‘rw’)
    A=1;B=5;C=A+B
    print(f‘{A}+{B}’, file=f)
    f.flush()
    if int(f.readline().strip()) == C:
        print(‘yes’, file=f, flush=True)
```

예

실습 4: 채팅 서버 만들기

```
# 접속된 유저들을 저장
users = []

def handler(cs, ca):
    # 메시지가 올때마다 모든 유저들에게 전달
    f = cs.makefile('rw')
    users.append(f) # 유저 리스트에 추가
    try:
        while True:
            line = f.readline()
            # print(line, file=f)
            for u in users: # 모든 유저들에게
                print(line, file=u, flush=True)
    finally:
        users.remove(f) # 유저 리스트에서 삭제
```

예

결과 & 반응

4x명 참가

목표 했던 기본 파트는 대부분 참여자들이 소화함

- 기본 파트: 소켓 사용, 산수 문제 풀기, 채팅 서버
- 고급 파트는 시간적 한계(6시간)가..

"처음으로 제대로 된 소켓 가이드를 들었다"

예상했던 어려움 - 외적인 요소

강연은 여러번 해봤으나 **튜토리얼은 처음**

듣는 사람들 수준의 분포를 알 수 없음

- 네트워크 를 실습하기 위한 기능을 작성하지 못할 수도 있다.
- 너무 잘하는 사람을 어쩌지?

YouTube 영상들 참고

- 사람들이 생각보다 잘 따라하지 못함
- 네트워크 장애가 발생하기도

대비 (1)

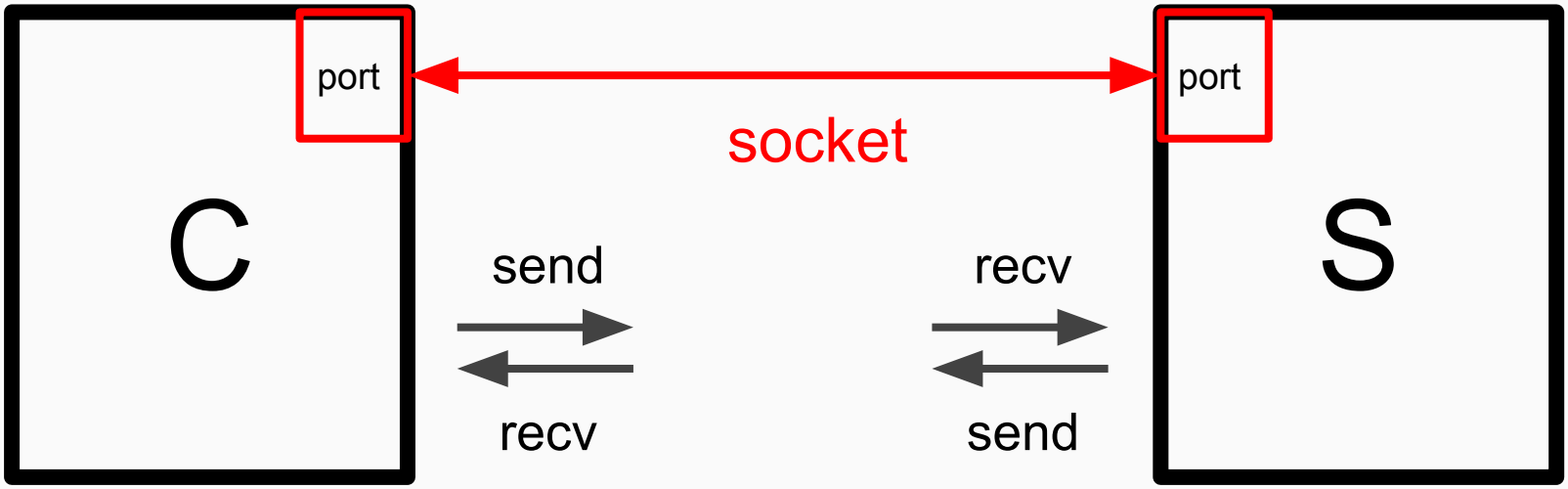
초보 수준을 대상으로 설정

서버만 만들면 사람들이랑 플레이할 수 있는
게임을 미리 만들어 감

네트워크 장애 대비

혼자서 해볼 수 있는 실습 이중으로 준비
다행히 네트워크 문제가 발생하지 않아 사용하지 않음

클라이언트 ↔ 서버 : connect하면 socket이 만들어짐



예

예상했던 어려움 - 내적인 요소

네트워크 자체 특징
    패킷이 잘려 가기도 한다.

기능을 단계별로 쌓아 나가는 방식으로 구성
    앞부분을 완료하지 못하면 뒷부분 진행이 불가능

대비 (2)

"파일" 인터페이스로 다루는 방법 소개
- "소켓"을 다루기 어려워할 수 있으므로.
- 줄바꿈을 기준으로 메시지를 주고 받으면
  - 패킷 잘림에 쉽게 대처 가능

기능 구현
- 최대한 간단하게 표현할 수 있는 방법을 제시하려 노력함
- 네트워크가 없는 케이스를 같이 제공
- 진행 중에 질문도 열심히 받는 걸로...

socket s → file-like f

```
f = s.makefile('rw')

line = f.readline()       # 읽기 (s.recv)

print('yes', file=f)      # 쓰기 (s.send)
f.flush()

print('yes', file=f, flush=True) # 쓰기2 (s.send)
```

예

실습 2: 산수 문제 풀기

```
import random                    # Server
A = random.randrange(10)         # S
B = random.randrange(10)         # S
C = A+B                          # S
print(f‘{A}+{B}’)                # Client
if int(input().strip()) == C:    #Server (판단) & Client (입력)
    print(‘yes’)                 # Client (출력)
else:
    print(‘no’)                  # Client (출력)
```

예

(-) 실제 상황에서 발생한 문제들

실습 1은 socket send/recv 사용
그 이후는 file interface 사용
    참가자들이 소켓 API로 구현하려다가 혼란에 빠짐
    중간에 file을 이용해서 해보는 실습이 필요했다.

Python 3.6, gevent 사용한다고 안내했으나
    확인 안하고 온사람이 10% 정도

실습 목표를 명확히 안내한 슬라이드가 없었음

사람들이 손들고 질문을 안함
    직접 자리를 왔다갔다 하며 바로 옆에서 도움을 줌
    6시간의 반은 서 있었음 (고통)

(+) 좋았던 점

생각보다 옆 사람들이 하는걸 잘 도와주려 한다
    튜토리얼 시작할 때
    너무 어려운 경우
    주위 "프로그래머"틱한 사람에게 도움 요청하라고 안내

강연에 비해 좀더 사람들과 가깝게 진행한 느낌
    (착각일 수도..)

알게된 점 + 더 나아가기

기초적인 내용이더라도 목말라 하는 사람들이 많다.
    참가한 사람 풀도 다양
    NDC 등에서도 튜토리얼 등 다양한 강연을 해보면 좋을듯

튜토리얼 용 자료는 최대한 꼼꼼하게 하자.

사람들 진짜 너무 질문 안한다..
    질문하기 편한 시스템 등을 개발해야 할지도?

끝.

네트워크 프로그래밍 개념 맛보기

하재승 (@ipkn, ipknhama@gmail.com)

0.
시작하기 전에...

목표

소켓을 경험해 보고, 동작하는 서버를 만들어 보는 것.

* 여러 서버 구조 비교나 고성능 서버, asyncio, UDP 등은 다루지 않습니다.

시간 남으면 즉석에서 공부해서 할지도 모릅니다 (?)

Python 2만 써보셨나요?

str과 bytes의 분리

'한글'.encode() == b'\xed\x95\x9c\xea\xb8\x80'

b'\xed\x95\x9c\xea\xb8\x80'.decode() == '한글'

주고 받는 데이터: bytes, 글로 표시되는 것: str

a=3; b=5; f'{a}+{b}' == '3+5'

Python Shell(쉘) / 파일 실행

쉘 = python⏎ 치고 실행하면 뜨는 것

aaa.py 라고 저장 후 python aaa.py⏎ 하면 파일 내용이 실행

설치 형태에 따라 python3, python3.6 이라고 해야 할 수도

1.
socket
사용해보기

socket ?

다른 컴퓨터 상의 두 프로그램이
서로 정보를 주고 받기 위한 기능/도구/API의 이름

서버, 클라이언트, 주소, 포트, 프로토콜

소켓 = (Client Addr, Client Port, Server Addr, Server Port)

오늘은 TCP/IP만 다룹니다.

개념들

클라이언트 - 고객님. 서버가 제공하는 서비스를 받기 위해 서버에 접속한다.

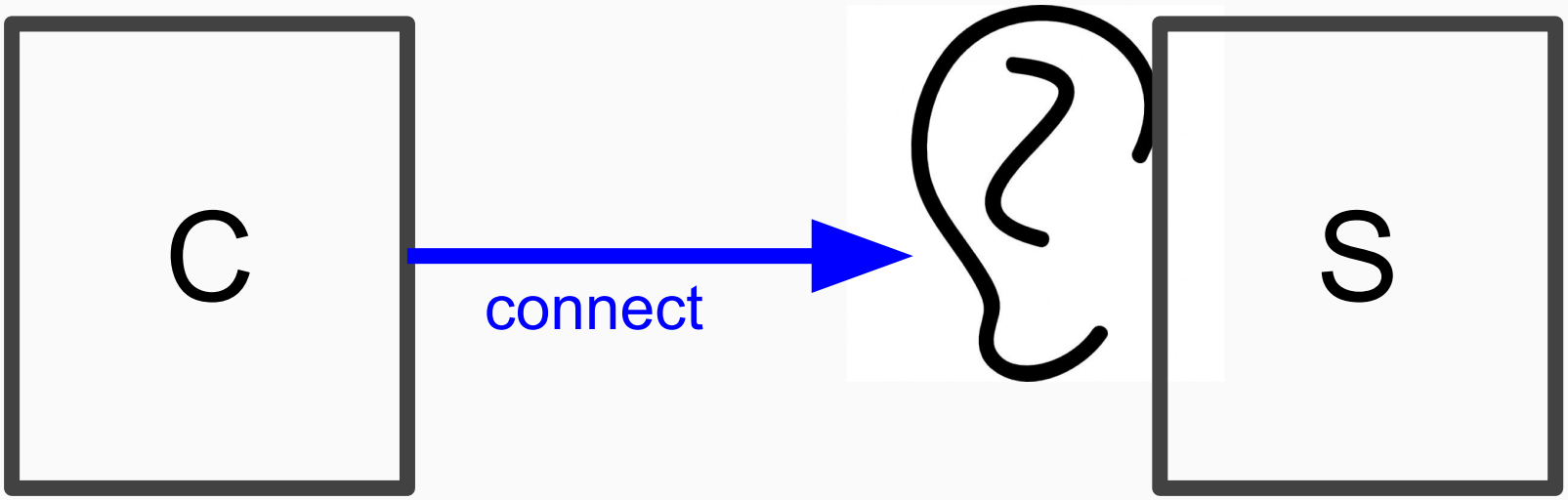
서버 - 클라이언트가 접속할 수 있도록 포트를 열고 기다린다.

주소 (IP주소) - 네트워크에 연결된 컴퓨터를 지칭하는 값.

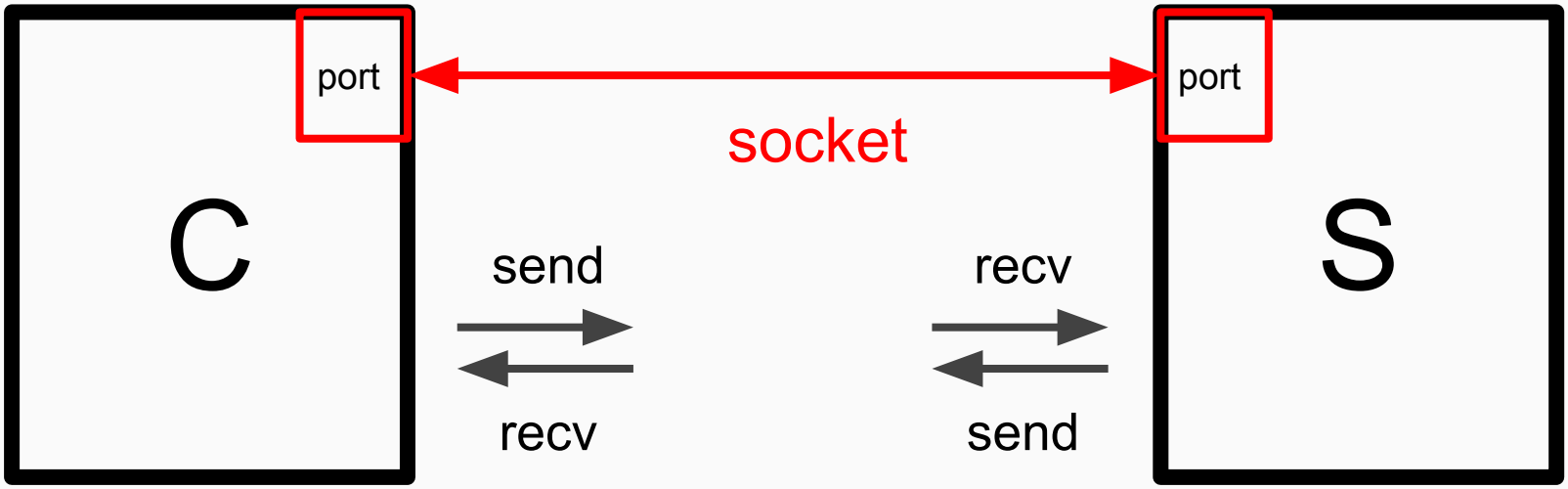
포트 - 한 컴퓨터 안의 여러 소켓을 구분하기 위해, 하나의 주소 아래에 소켓 별로 서로 다른 숫자를 부여함.

프로토콜 - 어떤 식으로 메시지를 주고 받을지에 대한 약속.

클라이언트 → 서버 : connect를 호출하여 접속시도



클라이언트 ↔ 서버 : connect하면 socket이 만들어짐



실습 1-1 (쉘): 클라이언트 코드

```
import socket

s = socket.socket()

s.connect((‘ipkn.me’, 10000))

s.send(b‘Hello socket’)

s.send(‘안녕 세계!’.encode())
```

실습 1: 클라이언트 코드

```
imp
s =
s.connect(('pkh.me', 10000))
s.send(b'Hello socket')
s.send('안녕 세계!'.encode())
```

connect가 호출되면 연결 시도
주소와 포트를 **(주소, 포트)** 형태로 넘겨줌

실습 1: 클라이언트 코드

```
import socket

                                    )
                                    e', 10000))
                                    ket')
s.send('안녕 세계!'.encode())
```

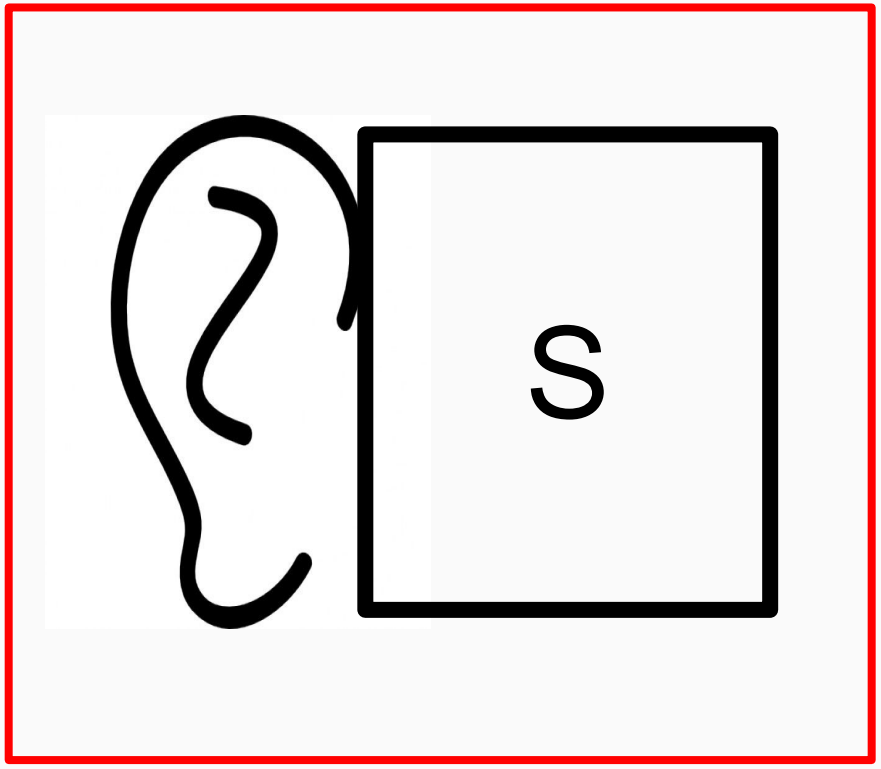
send로 bytes 값을
서버로 보냅니다.

실습 1-2 (쉘): 웹도 소켓으로 동작합니다.

```
import socket

s = socket.socket()

s.connect((‘www.google.com’, 80))

s.send(b’GET /\r\n\r\n’)

s.recv(1024)
```

서버에서는?

C



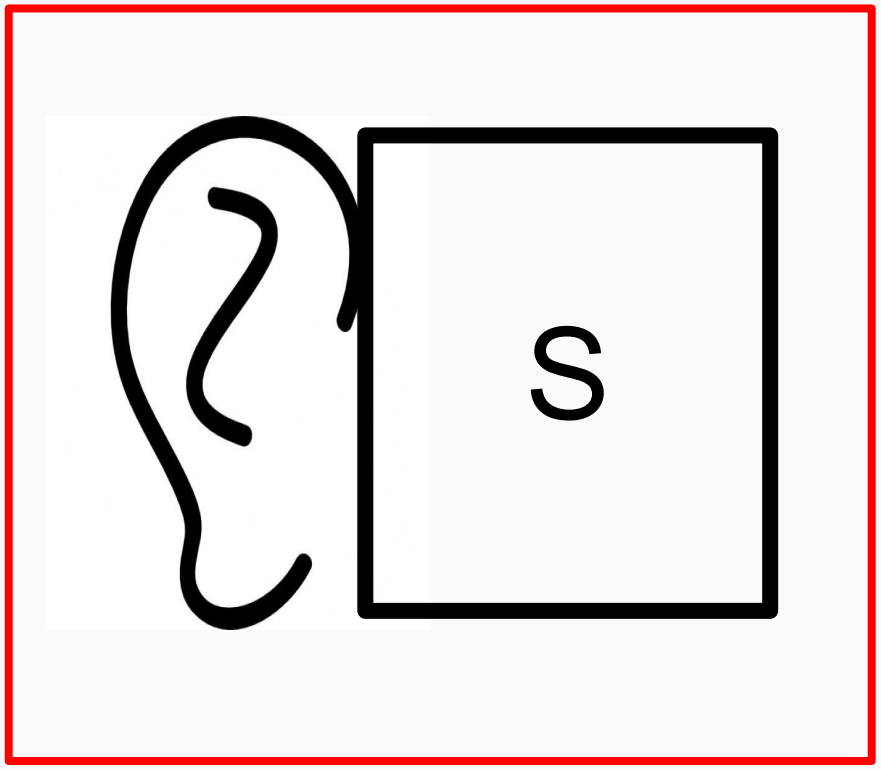
?

서버에서는?

클라이언트가 접속할
포트를 미리 지정해야
→ bind

여럿이 동시에 접속할 때
기다리기 위한 대기열 설정
→ listen

실제 클라이언트 연결 받기
→ accept



실습 1-3 (쉘): 서버/클라이언트 동시에

```
import socket

s = socket.socket()

s.bind(('0.0.0.0', 10000))

s.listen(5)

cs, ca = s.accept()
```

```
import socket

s = socket.socket()

s.connect(('localhost', 10000))

s.send(b'Hello socket')

s.send('안녕 세계!'.encode())
```

쉘을 두 개 띄워서 테스트!

실습 1-3 (쉘): 서버 코드

```
import socket

s = socket.socket()

s.bind((‘0.0.0.0’, 10000))

s.listen(5)

cs, ca = s.accept()

print(ca)
```

실습 1-3 (쉘): 클라이언트 코드

```
import socket
s = socket.socket()
s.connect((‘localhost’, 10000))
s.send(b’Hello socket’)
s.send(‘안녕 세계!’.encode())
```

실습 1-3 (쉘): 클라 연결 후

서버: `cs.recv(100)`

`cs.send(b'haha')`

클라: `s.recv(100)`

실습 1-4 (쉘): 소켓을 파일로

```
import socket

s = socket.socket()

s.connect((‘ipkn.me’, 10000))

f = s.makefile(‘rw’)

print(‘안녕 세계!’file=f, flush=True)

f.readline()
```

2.
socket을 이용한 산수 문제 풀기 서버 만들어보기

간단한 서버를 만듭시다.

초1 교육용 (?) 한 자릿수 덧셈 문제 서버를 만들어 봅시다.

유저가 접속하면 문제를 보내주고, (“3+5\n”)
유저의 답을 받아 (“8\n”)
맞으면 yes 틀리면 no를 보내줍시다. (“yes\n”)

메시지 마다 \n 으로 메시지 끝을 표시합시다.

간단한 서버를 만듭시다.

일반적으로 서버는 여러 명을 동시에 서비스 해야 하지만...

지금은 한 번에 한 명만 서비스 하는 서버를 만들겠습니다.

socket s → file-like f

```
f = s.makefile('rw')

line = f.readline()      # 읽기 (s.recv)

print('yes', file=f)     # 쓰기 (s.send)
f.flush()

print('yes', file=f, flush=True) # 쓰기2 (s.send)
```

실습 2: 산수 문제 풀기

```
import random                    # Server
A = random.randrange(10)         # S
B = random.randrange(10)         # S
C = A+B                          # S
print(f‘{A}+{B}’)                # Client
if int(input().strip()) == C:    #Server (판단) & Client (입력)
    print(‘yes’)                 # Client (출력)
else:
    print(‘no’)                  # Client (출력)
```

실습 2: 클라이언트 코드

```
import socket
s = socket.socket()
s.connect((‘localhost’, 10001))
f = s.makefile(‘rw’)
print(f.readline()) # life is not that easy
print(input(), file=f)
f.flush()
print(f.readline())
```

실습 2: 서버 뼈대 코드

```
import socket
s = socket.socket()
s.bind(('0.0.0.0', 10001))
s.listen(5)
while True:
    cs, ca = s.accept()
    f = cs.makefile('rw')
    A=1;B=5;C=A+B
    print(f'{A}+{B}', file=f)
    f.flush()
    if int(f.readline().strip()) == C:
        print('yes', file=f, flush=True)
```

클라이언트를 여러 개 실행해 보면?

어떻게 될까요?

3. gevent.StreamServer 사용해 보기

gevent ? gevent.StreamServer ?

여러가지 기법을 사용해서 여러 명에게 서빙하는 서버를 쉽게 짤 수 있게 해줍니다.

자세히는 설명 안하고 쓸겁니다.

어떻게 쓰나요?

```
import socket
s = socket.socket()
s.bind(('0.0.0.0', 10001))
s.listen(5)
while True:
    cs, ca = s.accept()
    f = cs.makefile('rw')
    ...
```

서버 공통인 부분

서버 별로 달라지는 부분

클라이언트 소켓, 주소 이용

어떻게 쓰나요?

```
from gevent.server import StreamServer

def handler(cs, ca):
    ...

server = StreamServer(('0.0.0.0', 10001),
                      handler)
server.serve_forever()
```

실습 3: 산수 문제 서버를 StreamServer로

```
from gevent.server import StreamServer

def handler(cs, ca):
    # 산수 문제 서버 구현

server = StreamServer(('0.0.0.0', 10001),
                      handler)
server.serve_forever()
```

4.
채팅 서버 만들기

멀티플레이 서버

산수 문제 서버, 웹 서버

- 접속한 유저 1명에게 서비스를 제공

채팅 서버, 멀티플레이 게임 서버

- 유저들 간의 인터랙션이 존재
- 특정 유저 그룹에게 일괄적으로 패킷을 보내게 됨

실습 4: 채팅 서버 만들기

```
# 접속된 유저들을 저장
users = []

def handler(cs, ca):
    # 메시지가 올때마다 모든 유저들에게 전달
    f = cs.makefile('rw')
    users.append(f) # 유저 리스트에 추가
    try:
        while True:
            line = f.readline()
            # print(line, file=f)
            for u in users: # 모든 유저들에게
                print(line, file=u, flush=True)
    finally:
```

여기서 부턴
고급 단계입니다.

마음껏 구현해 보고 서로에게 자랑해봅시다.

게임 클라이언트 받기:

http://home.ipkn.me:9999/ch5.zip

```
client.py localhost 10005
```

4-2.
채팅 서버 확장하기

어떤 기능을 넣을 수 있을까요?

대화명	(/대화명 입큰; /nick ipkn)
주사위	(/주사위 100; /dice 100) 1~100
개인 메시지	(/메시지 누구 저녁 같이 드실래요?)
투표	(/투표시작 /투표 ...)
채팅방	(/참가)

ex) 대화명

```
msg = f.readline()
for u in user:
   u.전송(msg)
```

```
msg = f.readline()
if msg[0:6] == '/nick ':
   nick = msg[6:]
   continue
for u in user:
   u.전송(nick+': '+msg)
```

실습 4-2: 주사위 기능 구현

화이팅 ^^

5.
제공된 클라이언트와 통신하는 서버 만들기

참고사항

이전 예제들과 마찬가지로 텍스트 기반 프로토콜
각 메시지 끝은 \n으로 표시
각 유저들은 유니크한 숫자로 구분된다

화면 크기: 가로 800px, 세로 600px

s.setsockopt(socket.SOL_TCP, socket.TCP_NODELAY, 1)

핑이 적은 통신을 위해 필요
(작은 패킷을 모아서 보내는 기능인) 네이글 알고리즘 비활성화

단계별 프로토콜

(0)

(1) SPAWN (c→s→c)
(2) POS (c→s→c)
(2) FIRE (c→s→c)
(3) HIT (s→c)
(3) KILL (s→c)
(3) GETITEM (s→c)
(3) PUTITEM (s→c)

* id, bullet_id는 int, 나머지는 모두 float

0 단계: 아무것도 구현 안 함

화면에 내가 보임

WSAD나 화살표로 이동

마우스로 조준 및 총 발사

1 단계: SPAWN

여러 클라이언트가 접속한 것을 서로 확인 가능

새로운 클라 접속 시
    SPAWN 받은 시점에 다른 클라 들에게 알려줌
    새로운 클라에게 기존 유저들의 정보를 알려줌

SPAWN

유저가 나타나거나 다시 살아날 때 보내는 패킷

꼴: `SPAWN [id] [x] [y] [hp]\n`

SPAWN 12345 100 200 100\n

id: 클라이언트 구분용 번호
x, y: 현재 위치
hp: 현재 체력

2 단계: POS, FIRE

서로 이동하고 총을 쏠 수 있다. 그리고 그 모습이 서로 보임.

자세한 체크를 원하는 경우
플레이어의 기본 체력은 100, 크기는 반지름 15이다.

* 동기화 구현이 없어서,
서로 위치가 틀어질 수 있다.

POS

위치, 속도 등을 알려주는 패킷

```
POS [id] [x] [y] [vx] [vy] [aimx] [aimy]\n
```

id:	클라이언트 구분용 번호
x, y:	현재 위치
vx, vy:	현재 속도
aimx, aimy:	현재 조준하는 곳

FIRE

총알이 발사될 때 보내는 패킷

```
FIRE [id] [bullet_id] [x] [y] [vx] [vy] [r] [dmg]\n
```

id: 클라이언트 구분용 번호
bullet_id: 총알 구분용 번호 (서로 다른 클라의 총알은 같은 번호일 수 있다)
x, y: 총알의 초기 위치
vx, vy: 총알의 속도 (초당 이동하는 픽셀)
r: 총알 반지름. 충돌 체크용으로 사용한다.
dmg: 총알 데미지.

3 단계: HIT, KILL

서버에서 총알 관련 충돌, 데미지 계산을 처리한다.

맞은 사람에게 HIT 메시지를 보낸다.

피가 0이하가 되면 KILL 메시지를 보낸다.

- 사망 시 클라이언트가 10초 후 자동으로 SPAWN을 보낸다.

HIT

atk_id의 총알 bullet_id를 def_id가 맞아 최종 체력 def_final_hp가 됨. 총알 맞음 처리를 서버에서 해야한다.

HIT [atk_id] [def_id] [bullet_id] [def_final_hp]

atk_id : 공격자 번호
def_id : 맞은 사람 번호
bullet_id : 공격자 총알 번호 (삭제해야함)
def_final_hp : 총알 맞은 후 최종 체력

KILL

atk_id가 def_id를 살해(막타). 서버가 클라이언트 들에게 알려줘야 한다.

```
KILL [atk_id] [def_id]
```

atk_id : 공격자 번호
def_id : 맞은 사람 번호

구현 힌트

모든 총알 b에 대해
    모든 플레이어 p에 대해
        if (p.x - b.x)**2 + (p.y-b.y)**2 < (20 + b.r)**2
            총알 맞음 처리

추가 목표:
아이템 사용

* 클라이언트 이젠 완성ㅋ

단계별 프로토콜

(1) SPAWN (c->s->c, broadcast)
(2) POS (c->s->c, broadcast)
(2) FIRE (c->s->c, broadcast)
(3) HIT (s->c, broadcast)
(3) KILL (s->c, broadcast)
(4) PUTITEM (s->c, broadcast)
(4) GETITEM (s->c)

GETITEM

특정 유저가 아이템을 획득함. 서버가 판별해서
클라이언트에 알려줌.

GETITEM [id] [item_id] [kind]

id: 아이템 먹은 유저 번호
item_id: 먹은 아이템 번호
kind: 아이템 종류 [GUN, BIGGUN, SHOTGUN, MINIGUN]

PUTITEM

새로운 아이템이 생성됨

```
PUTITEM [item_id] [x] [y] [kind]
```

item_id: 아이템 번호
x, y: 아이템 위치
kind: 아이템 종류

프로토콜

SPAWN (c->s->c, broadcast)
POS (c->s->c, broadcast)
FIRE (c->s->c, broadcast)
PUTITEM (s->c, broadcast)
GETITEM (s->c)
HIT (s->c, broadcast)
KILL (s->c, broadcast)

- Address already in use

setsockopt

SO_REUSEADDR

s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)