Fastcampus Web Programming SCHOOL

Decorator, Socket Programming

Decorator

Decorator는

- function의 앞, 뒤로 해야 할 일이나 로깅, 벤치마킹 등 다양한 용도로 쓰일 수 있습니다.
- 데이터 전처리 과정 또한 미리 정의해둔 뒤, 붙여 사용할 수 있습니다.

How does work?

- First-class function: 언어가 함수를 일급시민(first class citizen)으로 취급
- 함수를 다른 함수의 argument로 전달하거나, 결과값으로 return 가능
- 함수를 변수에 할당하거나 데이터 영역에 저장가능

```
def make_difference(operator):
    if operator=='+':
        return lambda x,y:x+y
    elif operator=='-':
        return lambda x,y:x-y
    else:
        print('you can only set plus or minus')

plus = make_difference('+')
plus(1,2)
minus = make_difference('-')
minus(1,2)
```

Prove it!

- Call by object reference

```
def print_hello(msg):
    print(msg)

f = print_hello
print(f, print_hello)
```

Closure

- First-class function을 지원하는 언어의 네임 바인딩
- 함수와 함께 함수 자신의 주변 환경을 저장한 레코드
- 함수가 가진 free variable을 당시의 값과 레퍼런스에 매핑
    - free variable: 스코프내에 정의되지 않았지만 사용된 변수

```
def outer():
    text_a = 'John'
    def inner():
        b = 'Doe'
        print('outer text: {}, inner text: {}'.format(text_a, b))
    return inner()
```

Take a look at cell_contents

```
def outer():
    text_a = 'John'
    def inner():
        b = 'Doe'
        print('outer text: {}, inner text: {}'.format(text_a, b))
    return inner

func = outer()
```

```
func.__closure__[0].cell_contents
```

Syntax of decorator

```
def decorator1(func):
    def wrapper():
        # statements
    return wrapper

@decorator1
def actual_work():
    # statements
```

Fibonacci with memoize

```
def memoize(func):
    memo = {}
    def wrapper(seq):
        if seq not in memo:
            memo[seq] = func(seq)
        return memo[seq]
    return wrapper

@memoize
def fib_memo(num):
    if num<2:
        return num
    else:
        return fib(num-1) + fib(num-2)
```

Back to the Fibonacci..

```
start_time = time.time()
fib_rec(10)
end_time = time.time()
print(end_time-start_time)
```

```
start_time = time.time()
fib_binet(10)
end_time = time.time()
print(end_time-start_time)
```

Let's wrap with decorator

```
import time


def time_checker(func):
    def wrapper():
        start_time = time.time()
        result = func()
        end_time = time.time()
        print("execution time: {time} sec".format(
            time=end_time-start_time))
        return result
    return wrapper
```

and just add `@{decorator}`

```
@time_checker
def fib_binet(num):
        ...

@time_checker
def fib_rec(num):
    ...
```

to avoid referencing problem, ..

```
time_checker(fib_rec)(10)
```

What if argument exist in wrapped function?

```
def print_args(func):
    def wrapper():
        func()
    return wrapper

@print_args
def get_user_info(name, mail):
    print('your name is {}, and your mail address is {}.'.format(name, mail))
    return name, mail

get_user_info('John Doe', 'johndoe@gmail.com')
```

solution

```
def print_args(func):
    def wrapper(*args):
        print(args)
        func(*args)
    return wrapper

@print_args
def get_user_info(name, mail):
    print('your name is {}, and your mail address is {}.'.format(name, mail))
    return name, mail

get_user_info('John Doe', 'johndoe@gmail.com')
```

For kwargs

```
def print_args(func):
    def wrapper(**kwargs):
        func(**kwargs)
    return wrapper

@print_args
def get_user_info(name='', mail=''):
    print('your name is {}, and your mail address is {}.'.format(name, mail))
    return name, mail

get_user_info(name='John Doe', mail='johndoe@gmail.com')
```

Do it yourself!

"Hi, {name}. You might be loved with {lang}" 이라는 문자열이 존재할 때,
이 문자열의 앞 뒤로 `<h1>, <em>` 태그가 붙도록 하는 데코레이터를 생성하세요

ex output)

```
<h1><em> {{text}} </em></h1>
```

`Advanced problem: Decorator 하나로 html 태그 이름을 지정할 수 있도록 수정`

Network

우리는 어떻게 다른 컴퓨터와 통신하고, 웹서핑을 할 수 있을까?

Network

A computer network or data network is a telecommunications network which allows nodes to share resources.

--> 컴퓨터간 리소스를 공유 가능하게 만드는 통신망

Charcteristics of Network

- 컴퓨터사이의 리소스를 공유
- 네트워크로 연결된 다른 컴퓨터에 접근하여 파일을 생성, 수정, 삭제할 수 있음
- 프린터와 스캐너, 팩스 등의 출력장치에 네트워크를 연결하여 여러 컴퓨터가 동시 접근 가능

Requirements of Network

- Network Cable
- Distributor(Switch Hub)
- Router
- Network card

커버 범위에 따른 네트워크 구분

LAN

- Local Area Network(근거리 통신망)
- 학교, 회사 등 가까운 지역의 좁은 범위

WAN

- Wide Area Network(광역 통신망)
- 국가, 대륙 등 넓은 지역을 커버

MAN

- Metropolitan Area Network(도시권 통신망)
- LAN < `MAN` < WAN

WLAN

- Wireless Local Area Network(무선 근거리 통신망)
- IEEE 802.11 표준을 기반
- http://standards.ieee.org/about/get/802/802.11.html

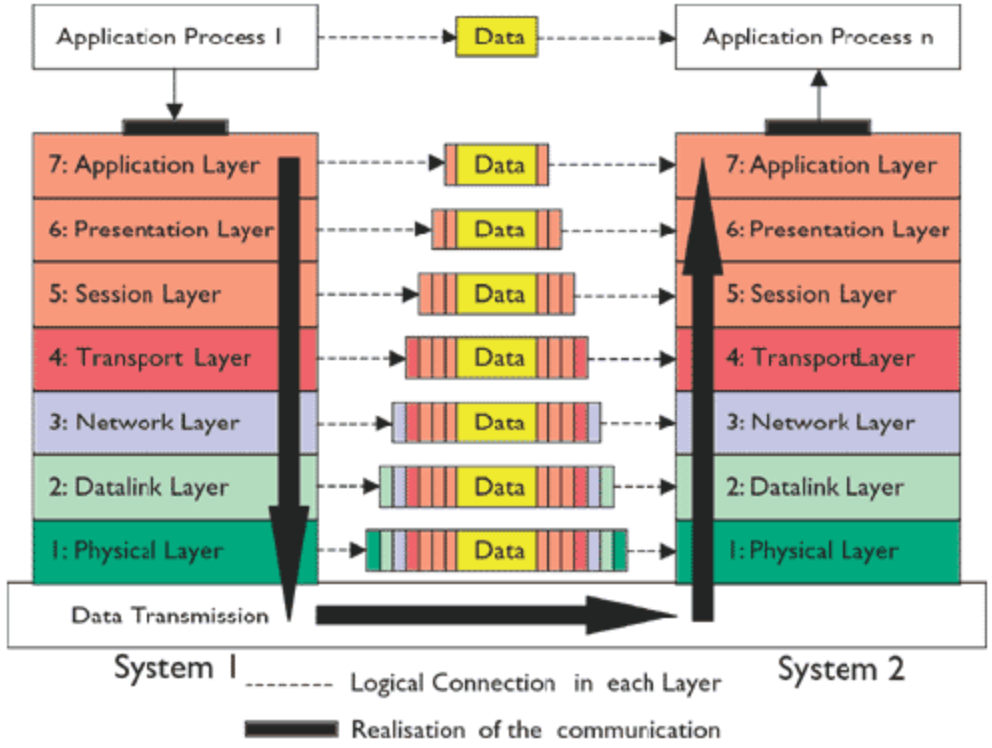
Network OSI 7 layer

- `O` pen `S` ystems `I` nterconnection Reference Model
- 국제 표준화기구에서 개발한 컴퓨터 네트워크 프로토콜 디자인과 통신을 계층으로 나누어 설명한 것

Packet

- 데이터를 한번에 전송할 단위로 자른 데이터의 묶음 혹은 그 크기
- 1492 ~ 1500 bytes(프로토콜에 따라 다름)
- 네트워크에서는 바이트(byte)라는 표현 대신 옥텟(octet)으로 표현

Network OSI 7 layer



TCP/IP

Transmission Control Protocol / Internet Protocol

- 전송제어 프로토콜 + 송수신 호스트의 패킷교환을 위한 프로토콜

TCP

- 전송제어프로토콜 / Transmission(Transfer) Control Protocol
- 근거리 통신망이나 인트라넷, 인터넷에 연결된 컴퓨터에서 실행되는 프로그램 간에 일련의 옥텟(==byte)을 안정적으로, 순서대로, 에러없이 교환할 수 있게 함

STREAM

- 문자형식의 데이터가 열의 형태로 연속성을 띄게 됨

DATAGRAM

- 하나의 패킷이 발신지와 수신지 정보를 모두 담고 있는 독립적인 패킷

STREAM socket

- 연결형 스트림 소켓은 `두개의 시스템` 이 `연결된 후` 데이터를 `교환`
- 패킷 순서 신경쓰지 않아도 되어 안정적인 데이터 전송 가능

DATAGRAM socket

- 비연결형 데이터그램 소켓은 명시적으로 `연결되지 않은 상태` 로 데이터를 주고 받음
- 연결과 해제 과정이 없어 빠른 데이터 교환이 가능

IP

IPv4, IPv6

- Internet Protocol version 4
  - 32bit로 구성
  - 0.0.0.0 ~ 255.255.255.255
  - 0000 0000.0000 0000. 0000 0000. 0000 0000
  - `2^32` = `42.9억`

IP	용도
0.0.0.0/8	자체 네트워크
10.0.0.0/8	사설 네트워크
127.0.0.0/8	루프백(loopback) 즉, 자기자신
169.254.0.0/16	링크 로컬(link local)
172.16.0.0/12	사설 네트워크
192.0.2.0/24	예제 등 문서에서 사용
192.88.99.0/24	6to4 릴레이 애니캐스트
192.168.0.0/16	사설 네트워크
198.18.0.0/15	네트워크 장비 벤치마킹 테스트

127.0.0.1 vs 192.168.0.x

127.0.0.1

- Loopback: 컴퓨터가 가지고 있는 무조건 반대신호를 반환하는 대역
- Localhost

192.168.0.x

- LAN에서 라우터가 할당한 내컴퓨터의 IP address

Global IPv4 depletion

IPv4, IPv6

- Internet Protocol version 6
  - 128bit로 구성
  - 0000:0000:0000:0000:0000:0000:0000:0000 ~ FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF
  - `2^128` = `16*16*16*16*16*16^8` = `340,282,366,920,938,463,463,374,607,431,768,211,456` = `3.4*10^38`

Public, Private

Public IP address

- Globally Unique

Private IP address

- Private network 내에서 유효

DNS

- Domain Name System
- 외우기 힘들며, 더 힘들어질 ip address를 사람이 판별하기 쉬운 url을 매핑하는 시스템

UDP

User(Universal) Datagram Protocol

- 데이터그램을 전송하기 위한 프로토콜
- 메시지 수신확인x, 도착순서 예측x
- 빠른 속도, 적은 오버헤드

TCP vs UDP segment

TCP Segment Header Format

Bit #	0 – 7	8 – 15	16 – 23	24 – 31
0	Source Port		Destination Port	
32	Sequence Number			
64	Acknowledgment Number			
96	Data Offset	Res	Flags	Window Size
128	Header and Data Checksum		Urgent Pointer	
160...	Options			

UDP Datagram Header Format

Bit #	0 – 7	8 – 15	16 – 23	24 – 31
0	Source Port		Destination Port	
32	Length		Header and Data Checksum	

intranet vs Internet vs internet

- intranet: internet의 www기술을 활용하여 특정 단체의 내부 정보시스템을 구축하는 것 혹은 그 네트워크
- Internet(`International Network`): TCP/IP를 활용하여 정보를 주고 받는 통신 네트워크(www)
- internet(`internetwork`): 패킷을 교환하는 방식으로 기기간의 정보를 주고 받는 방식

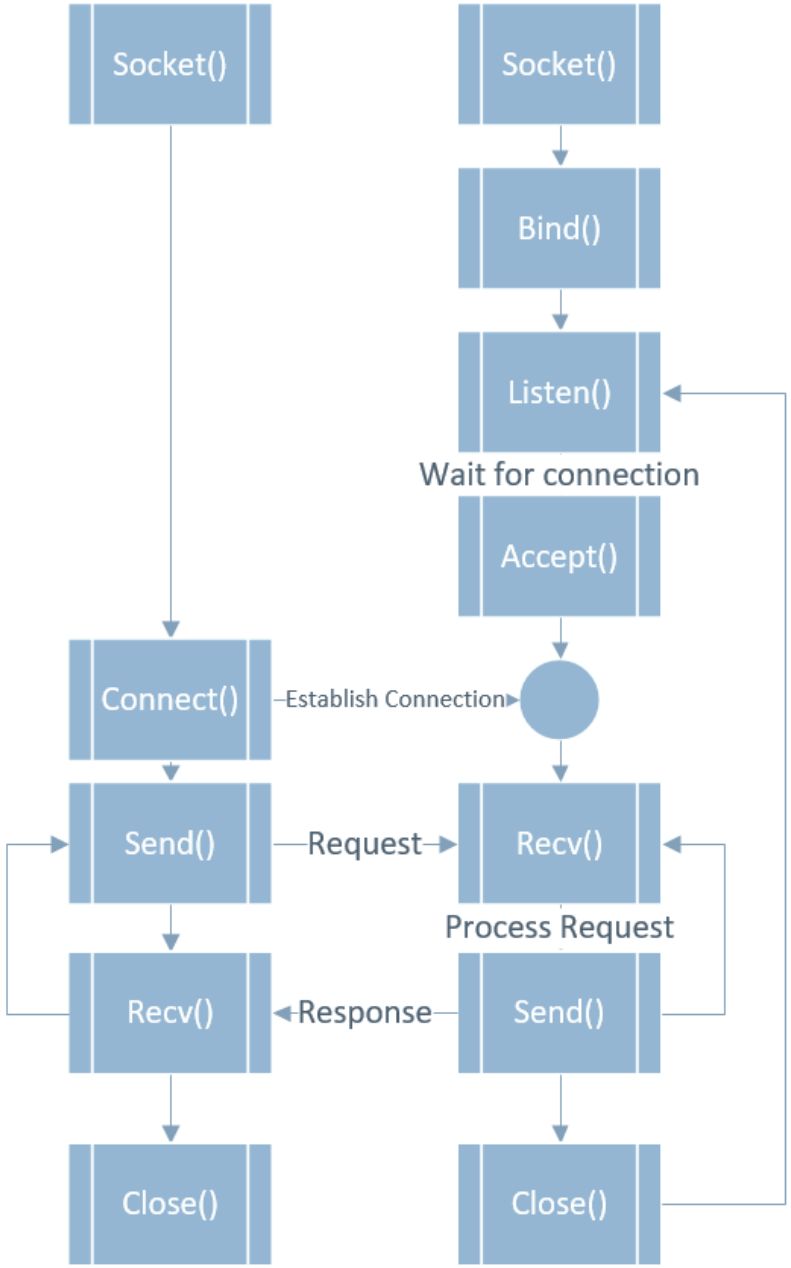
Socket

Socket

- `Virtual End Point` where entities can perform inter-process communication.

So, Socket is ..

> 떨어져 있는 두 컴퓨터를 연결해주는 과정


Socket()
Socket()
Bind()
Listen()
Wait for connection
Accept()
Connect()
Establish Connection
Send()
Request
Recv()
Process Request
Recv()
Response
Send()
Close()
Close()

parameters

socket family(family) - AF_INET, AF_UNIX, AF_BLUETOOTH
socket type(type) - SOCK_STREAM, SOCK_DGRAM

setsockopt()

setsockopt() sets a socket option

SOL_SOCKET the level argument to getsockopt or setsockopt to manipulate the socket-level options described in this section.

SO_REUSEADDR for security(packet hijacking)

socket 통신도 통신이므로.. 통신보안!!

TLS(Transport Layer Security) - 프로토콜에 의한 암호화
SSl(Secure Socket Layer) - 포트에 의한 암호화