# Fastcampus Web Programming SCHOOL

## Class, OOP

# Class, instance, static Method

- Class Method: @classmethod 데코레이터 이용
  - cls 인자를 받음
  - class variable 관련 일을 할때 사용
- Instance Method: self 인자 이용
  - 객체의 고유 속성 값 사용
- Static Method: 아무 인자 없이 사용
  - 일반 함수와 같은 역할
  - class와 관련있는 함수를 class 내부의 정의코자 할 때 사용

# Method Example

```
class Wallet:
    def __init__(self, account):
        self.account = account

    balance = 0
    name = 'Your wallet'

    @classmethod
    def set_default(cls, amount):
        while amount < 1:
            print("Err: You should set default value over 1. Try again!")
            amount = int(input("Set default value: "))
        cls.balance = amount
        print('Set default balance to {}'.format(amount))
    #instance method
    def add_to_account(self, amount):
        self.account += amount
        print('Your total balance is {}'.format(self.account + Wallet.balance))

    @staticmethod
    def see_static_name():
        print(Wallet.name)

    def get_class_name(cls):
        print(cls.name)

class MyWallet(Wallet):
    name = 'My wallet'
```

# Polymorphism

- Polymorphism: 상속관계의 다양한 Class object에서 같은 이름의 method에 대해 각 object가 서로 다르게 구현된 method를 호출하여 같은이름 - 다른 행동, 기능, 결과를 가져오는 것.

```
class SmartPhone:
    def get_ap(self):
        print('Application Proccessor')

class IPhone(SmartPhone):
    def get_ap(self):
        print('A series')

class Galaxy(SmartPhone):
    def get_ap(self):
        print('Exynos')
```

# Abstract Base Class

- 공통 특성을 가진 부모 Class를 만들고 해당 Class가 Object instance를 생성할 수 없게 하는 것
- 특성을 유지할 수 없도록 하여 추상적인 Class가 개념으로 존재하게 만듦

```
from abc import *



class SmartPhone(metaclass = ABCMeta):
    @abstractmethod
    def get_ap(self):
        pass

    ...
```

# Practice(1)

추상화 할 수 있는 개념과 대상에 대해 자유롭게 이를 코드로 구현하세요.
(ex - animals, monsters, heroes, ..)

## Module

- 본체에 대한 하위 단위라는 필연적인 개념
- python 코드로 이루어진 파일

## Package

- 관련된 여러 모듈을 하나의 폴더로 묶은 것

# Set Package

```
# fibos/fibo.py
def fibo(n):
    if n < 2:
        return n
    else:
        return fibo(n-1) + fibo(n-2)

# fibos/binet.py
from math import sqrt


def binet(n):
    return int(((1+sqrt(5))**n - (1-sqrt(5))**n) / (2**n*sqrt(5)))
```

```
# print_something.py
print('something')
```

# Module, Package

```
#from fibos import fibo
#from fibos import binet as bn
#from fibos import fibo, binet
#from fibos import *

#import print_something
#from . import print_something
```

# Library & Framework

## Library(jQuery, http)

- 함수와 기능의 모음
- 파이썬 표준 라이브러리: 내장함수(print) + C 내장모듈(os), Python 모듈(math)
- You are in control, you call the library.

```
math.pi
```

## Framework(React, django)

- 개념들의 추상화를 제공하는 클래스와 컴포넌트로 구성
- Framework is in control, it calls you.

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def get_index():
    return 'this is home'
```

## Poetry

- Python Packaging and dependency manager

## for pyenv

```
# pyenv의 PATH
export PYENV_PATH=$HOME/.pyenv
if which pyenv > /dev/null; then eval "$(pyenv init -)"; fi
if which pyenv-virtualenv-init > /dev/null; then eval "$(pyenv virtualenv-init -)"; fi

# poetry실행파일의 PATH가 pyenv의 PATH보다 우선되도록 설정
export PATH=$HOME/.poetry/bin:$PATH
```

# Commands

```
# initialize
$ poetry init

# add package
$ poetry add <package name>

# remove package
$ poetry remove <package name>

# show installed package
$ poetry show --no-dev --tree

# export package list to requirements.txt
$ poetry export -f requirements.txt > requirements.txt
```