섹션6

파이썬 클래스 및 모듈, 패키지

CONTENTS

01

02

03

클래스

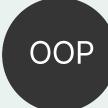
모듈

패키지

01

클래스 CLASS

전체적인 틀



> 객체지향프로그램, 생산성 향상, 재사용 극대화



> 객체 인스턴스화 시 저장된 공간(DICT)



함수. 객체의 동작 정의



객체 > 클래스 기반으로 생성된 구체적인 실체



DEF_INIT_(SELF, NAME, AGE) 변수. 객체의 상태 표시



인스턴스 자신 매서드 내 인스턴스 속성이나 다른 매서드 접근 시 사용

```
class Dog:
    def __init__(self, name, breed):
        self.name = name
        self.breed = breed
    def bark(self):
        return f"{self.name} is barking!"
# 객체 생성
my_dog = Dog(name="Buddy", breed="Golden
Retriever"
print(my_dog.bark()) # Buddy is barking!
```

02

모듈 module

0정의

- 재사용, 공유 가능하도록 만들어 놓은 파일.
- IMPORT하여 다른 파이썬 프로그램에서 사용 가능.
- 〉 .PY 확장자를 가짐

OSTEP1

IMPORT SYS → PRINT(SYS.PATH)

> 경로 찾아서 등록하는 모듈

3 STEP2

SYS.PATH.APPEND('경로') : 모듈 경로 삽입 → PRINT(SYS.PATH)

> IMPORT

4 STEP3

IF __NAME__=="__MAIN__"

> 모듈로 사용 시 기존 코드 불러오지 않는 함수

```
# my_module.py
def greet(name):
    return f"Hello, {name}!"

class Calculator:
    def add(self, x, y):
        return x + y
```

```
import my_module
print(my_module.greet("Alice")) # Hello,
Alice!

calc = my_module.Calculator()
print(calc.add(5, 3)) # 8
```

```
# my_module.py

def greet(name):
    return f"Hello, {name}!"

def main():
    user_name = "Alice"
    print(greet(user_name))

if __name__ == "__main__":
    main()
```

```
import my_module
print(my_module.greet("Bob")) # Hello, Bob!
```

03

배커지 Package

모듈을 모아놓은 파일 __INIT__.PY



SUB패키지 가져올 때 간결하게 사용 가능(어디부터 어디까지)



별명 간결하게 표현하기 위한 수단

코드의 가독성 높이고 내임스페이스의 충돌 방지 역할 수행

```
from datetime import datetime
# datetime 모듈의 datetime 클래스만 임포트
now = datetime.now()
print(now)
```

```
from math import sqrt, pi
# math 모듈 전체를 가져오지 않고 sqrt와 pi만 사용
print(sqrt(16)) # 4.0
print(pi) # 3.141592653589793
```

```
import numpy as np
# numpy 모듈을 np로 사용
array = np.array([1, 2, 3])
print(array) # [1 2 3]
```

from math import factorial as fact

```
# factorial 함수를 fact라는 이름으로 사용
print(fact(5)) # 120
```