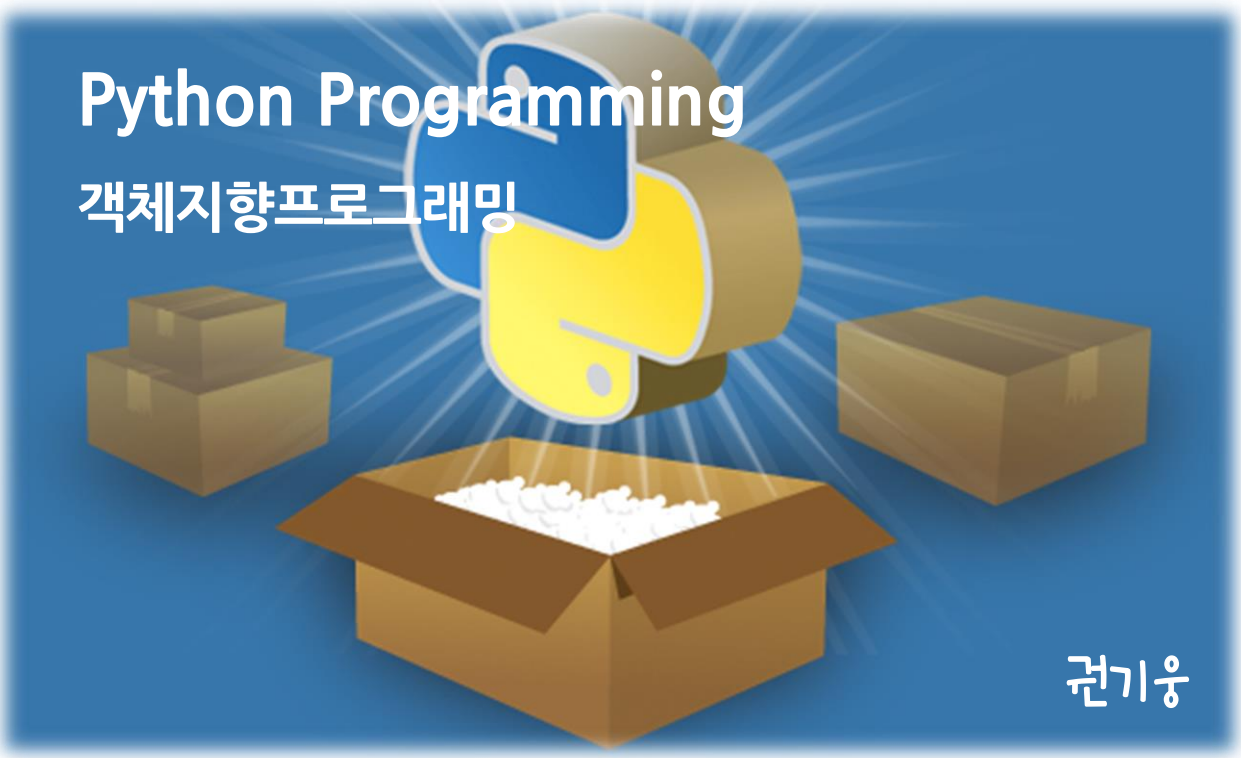
Python Programming
객체지향프로그래밍
권기웅

1. 객체지향프로그래밍이란

1.1 객체지향프로그래밍이란

Object Oriented Programming

객체 단위로 실생활을 표현.

실생활이 표현 되므로 파이썬에 대한 지식이 없어도 소스 보기에 쉬워짐

객체단위로 코드가 짜여져 있어 유지보수에도 용이해짐

2. OOP 기본

2.1 OOP 기본

2.1.1 OOP 기본

```
Python 3.6.3 (v
Type "copyright
>>>
==========
멍
멍
기본이름
기본값
기본이름
기본값
기본이름
기본값
기본이름
기본값
---
멍
뽀삐
2
뽀삐
2
뽀삐
2
>>>
```

```
# 클래스 : 객체를 찍어내는 틀 같은 존재, 클래스를 만들어야 객체를 만들 수 있음
# 클래스 정의
class Dog:
    name = "기본이름"   #멤버변수 : 그 객체가 가지는 속성을 표현
    age = "기본값"

    def bark(self):     #메소드 : 그 객체가 하는 행동을 표현
        print("멍")

    def printInfo(self, c): # 메소드 : 프로그램상 필요한 기능
        for i in range(c):
            print(self.name)
            print(self.age)

# 객체 만들어서 사용
d1 = Dog()  # d1이라는 개 생성
d1.bark()   # 개 d1이 짖게
Dog.bark(d1)# 개 d1이 짖게
d1.printInfo(2) # 개 d1의 정보 출력
Dog.printInfo(d1, 2)# 개 d1의 정보 출력

print("---")

d2 = Dog()
d2.name = "뽀삐"  # 개 d2의 이름 변경
d2.age = 2        # 개 d2의 나이 변경
d2.bark()
d2.printInfo(3)
```

2.2 멤버변수

2.2.1 멤버변수 추가

Python 3.6.3 Shell

```
Python 3.6.3 (v
Type "copyright
>>>
================
파이썬
10000
>>>
```

OOP.py - D:/workspacePython/OOP.py (3.6.3)

```
# 클래스 정의
class Book:
    title = ""

    def printInfo(self):
        print(self.title)

# 객체 만들어서 사용
b1 = Book()
b1.title = "파이썬"
b1.price = 10000 # 없는 멤버변수 추가해서
b1.printInfo()
print(b1.price) # 사용 가능
```

2.2.2 global 변수와 구분

Python 3.6.3 Shell

```
Python 3
Type "co
>>>
========
red
black
>>>
```

OOP.py - D:/workspacePython/OOP.py (3.6.3)

```
color = "black"

# 클래스 정의
class Pen:
    color = ""

    def printColor(self):
        print(self.color)
        print(color) # self.를 붙이지 않으면 global변수로 인식


# 객체 만들어서 사용
p = Pen()
p.color = "red"
p.printColor()
```

2.3 method

2.3.1 static method

```
Python 3.6.3 Shell
File Edit Shell Deb
Python 3
Type "co
>>>
========
국산
39900
29900
>>>
```

OOP.py - D:/workspacePython/OOP.py (3.6.3)

File Edit Format Run Options Window Help

```
# static 메소드 :  객체들의 속성을 사용하지 않는
#                  객체가 실제로 존재하지 않더라도 사용할 수 있는 메소드

# 김치 쇼핑몰 프로그램에서
class Kimchi:
    price = 0

    def printPrice(self): # 가격은 특정 김치의 속성, 김치마다 다름
        print(self.price)

    def printOrigin(): # 원산지는 특정 김치의 속성이 아닌, 무조건 국산(static 메소드)
        print("국산")

Kimchi.printOrigin()

k1 = Kimchi()
k1.price = 39900
k1.printPrice()

k2 = Kimchi()
k2.price = 29900
k2.printPrice()
```

2.3.2 생성자(constructor), 소멸자(destructor)

```
Python 3.6.3 Shell
File Edit Shell Debug Options
Python 3.6.3 (
Type "copyrigh
>>>
==============
사람 객체 생성
홍길동
홍길딩
사람 객체 소멸
>>>
```

OOP.py - D:/workspacePython/OOP.py (3.6.3)

File Edit Format Run Options Window Help

```
# 생성자 : 객체가 만들어질때 호출되는 메소드
# 소멸자 : 객체가 사라질 때 호출되는 메소드
class Human:
    def __init__(self, name): # 생성자
        self.name = name      # 객체가 만들어질 때 속성을 부여하는 용도로 사용 가능
        print("사람 객체 생성")

    def __del__(self):        # 소멸자
        print("사람 객체 소멸") # 객체가 없어질 때 정리하는 용도로 사용

    def printName(self):
        print(self.name)

hong = Human("홍길동")
hong.printName()
myBrother = hong
myBrother.name = "홍길딩"
hong.printName()

hong = None
myBrother = None
```

python

3. 상속

3.1 상속 기본

Python 3.6.3 Shell

```
Python 3.6.3 (
Type "copyrigh
>>>
==========
?
0
msy노트북
1500000
>>>
```

OOP.py - D:/workspacePython/OOP.py (3.6.3)

```
class Item:
    def __init__(self, name, price):
        self.name = name
        self.price = price

    def printInfo(self):
        print(self.name)
        print(self.price)

# class 클래스명(상속받을 클래스명)
class Computer(Item):
    pass

i = Item("?", 0)
i.printInfo()

# Item클래스(상위클래스)에 정의된 모든것을 물려받아 사용 가능
c = Computer("msy노트북", 1500000)
c.printInfo()
```

3.2 기능 확장, overriding

Python 3.6.3 Shell

```
Python 3.6.3 (v
Type "copyright
>>>
==========
?
0
컴퓨터 정보 출력
msy노트북
1500000
msy노트북
1500000
>>>
```

OOP.py - D:/workspacePython/OOP.py (3.6.3)

```
class Item:
    def __init__(self, name, price):
        self.name = name
        self.price = price

    def printInfo(self):
        print(self.name)
        print(self.price)

class Computer(Item):
    def printASInfo(self): # 메소드 추가(기능 확장)
        print("구입후 1년간 AS무료")

    def printInfo(self): #메소드 재정의(overriding)
        print("컴퓨터 정보 출력")
        super().printInfo() #상위클래스에 정의되어있는 printInfo호출
        Item.printInfo(self) #Item클래스에 정의되어있는 printInfo호출

i = Item("?", 0)
i.printInfo()

c = Computer("msy노트북", 1500000)
c.printInfo()
```

3.3 생성자

Python 3.6.3 Shell

```
Python 3.6.3 (v
Type "copyright
>>>
================
삼디수
1000
2
>>>
```

OOP.py - D:/workspacePython/OOP.py (3.6.3)

```
class Item:
    def __init__(self, name, price):
        self.name = name
        self.price = price

    def printInfo(self):
        print(self.name)
        print(self.price)

class Water(Item):
    def __init__(self, name, price, l):
        super().__init__(name, price)
        self.l = l

    def printInfo(self):
        super().printInfo()
        print(self.l)

w = Water("삼디수", 1000, 2)
w.printInfo()
```

3.4 다중상속

Python 3.6.3 Shell

```
Python 3.6.3 (v
Type "copyright
>>>
================
밥먹기
공격
사람
>>>
```

OOP.py - D:/workspacePython/OOP.py (3.6.3)

```
class Human:
    def eat(self):
        print("밥먹기")

    def say(self):
        print("사람")

class Avengers:
    def attack(self):
        print("공격")

    def say(self):
        print("어벤져스")


class Ironman(Human, Avengers): # ,를 사용해 여러 클래스로부터 상속받아올 수 있음
    pass

i = Ironman()
i.eat()
i.attack()
i.say() # 같은 이름의 메소드를 동시에 받아오면 먼저 상속받아온 클래스 우선
```