15장. 클래스

파이썬 정복

Contents

❖ 목차

- 1. 클래스
- 2. 여러 가지 메서드
- 3. 유틸리트 클래스

1. 클래스

❖ 클래스

- 객체지향의 가장 기본적 개념
- 관련된 속성과 동작을 하나의 범주로 묶어 실세계의 사물을 흉내냄
- 모델링
  - 사물 분석하여 필요한 속성과 동작 추출
- 캡슐화
  - 모델링 결과를 클래스로 포장

capsule

```
balance = 8000

def deposit(money):
    global balance
    balance += money

def inquire():
    print("잔액은 %d원입니다." % balance)

deposit(1000)
inquire()
```

실행결과

```
잔액은 9000원입니다.
```

1. 클래스

- 사물의 속성은 변수로, 동작은 함수로 표현

- 멤버
  - 클래스 구성하는 변수와 함수
- 메서드
  - 클래스에 소속된 함수

1. 클래스

❖ 생성자

- 클래스 선언 형식

```
class 이름:
    def __init__(self, 초기값):
        멤버 초기화
    메서드 정의
```

- __init__ 생성자
  - 통상 객체 초기화

human

```
class Human:
    def __init__(self, age, name):
        self.age = age
        self.name = name
    def intro(self):
        print(str(self.age) + "살 " + self.name + "입니다.")
kim = Human(29, "김상형")
kim.intro()
lee = Human(45, "이승우")
lee.intro()
```

실행결과

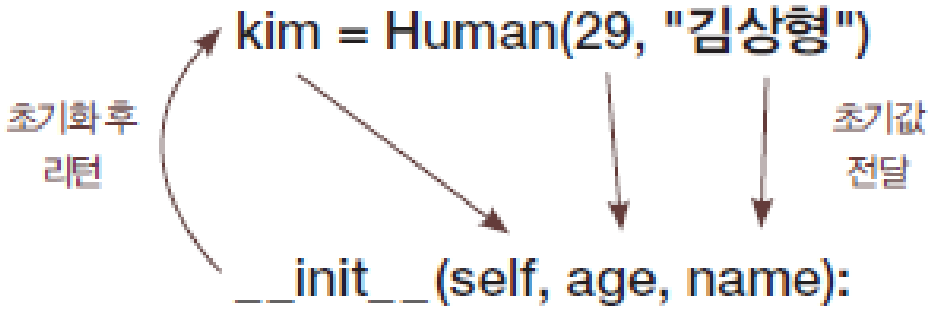
```
29살 김상형입니다.
45살 이승우입니다.
```

1. 클래스

- 객체 생성 구문

```
객체 = 클래스명(인수)
```

  - 객체를 __init__의 첫 번째 인수 self로 전달
  - 생성문에서 전달한 인수를 두 번째 이후의 인수로 전달
  - 새로 생성되는 객체 멤버에 대입



- 메서드는 필요한 만큼 선언할 수 있음
  - 객체.메서드()

1. 클래스

❖ 상속

- 기존 클래스 확장하여 멤버 추가하거나 동작 변경
  - 클래스 이름 다음의 괄호 안에 부모 클래스 이름 지정

```
class 이름(부모):
    ....
```

student

```
class Human:
    def __init__(self, age, name):
        self.age = age
        self.name = name

    def intro(self):
        print(str(self.age) + "살 " + self.name + "입니다")


class Student(Human):
    def __init__(self, age, name, stunum):
        super().__init__(age, name)
```

1. 클래스

```
        self.stunum = stunum

    def intro(self):
        super().intro()
        print("학번 : " + str(self.stunum))

    def study(self):
        print("하늘천 따지 검을현 누를황")


kim = Human(29, "김상형")
kim.intro()
lee = Student(34, "이승우", 930011)
lee.intro()
lee.study()
```

실행결과

```
29살 김상형입니다
34살 이승우입니다
학번 : 930011
하늘천 따지 검을현 누를황
```

- **super() 메서드**
  - **자식 클래스에서 부모의 메서드 호출할 때 사용**

1. 클래스

❖ 액세서

- 파이썬 클래스의 멤버는 모두 공개되어 누구나 외부에서 액세스 가능
- 일정한 규칙 마련하고 안전한 액세스 보장
  - 게터(Getter) 메서드
    - 멤버 값 대신 읽음
  - 세터(Setter) 메서드
    - 멤버 값 변경

getset

```
class Date:
    def __init__(self, month):
        self.month = month
    def getmonth(self):
        return self.month
    def setmonth(self, month):
        if 1 <= month <= 12:
            self.month = month

today = Date(8)
today.setmonth(15)
print(today.getmonth())
```

실행결과

```
8
```

2. 여러 가지 메서드

❖ 클래스 메서드

- 특정 객체에 대한 작업 처리하는 것이 아니라 클래스 전체에 공유
  - @classmethod 데커레이터
  - 첫 번째 인수로 클래스에 해당하는 cls 인수

classmethod

```
class Car:
    count = 0
    def __init__(self, name):
        self.name = name
        Car.count += 1
    @classmethod
    def outcount(cls):
        print(cls.count)

pride = Car("프라이드")
korando = Car("코란도")
Car.outcount()
```

실행결과

```
2
```

2. 여러 가지 메서드

❖ 정적 메서드

- 클래스에 포함되는 단순 유틸리티 메서드
- 특정 객체에 소속되거나 클래스 관련 동작 하지 않음
- @staticmethod 데커레이터

staticmethod

```
class Car:
    @staticmethod
    def hello():
        print("오늘도 안전 운행 합시다.")
    count = 0
    def __init__(self, name):
        self.name = name
        Car.count += 1
    @classmethod
    def outcount(cls):
        print(cls.count)

Car.hello()
```

실행결과

```
오늘도 안전 운행 합시다.
```

2. 여러 가지 메서드

❖ 연산자 메서드

- 연산자 사용하여 객체끼리 연산
- 연산자 오버로딩
  - 클래스별로 연산자 동작을 고유하게 정의

연산자	메서드	우변일 때의 메서드
==	__eq__	
!=	__ne__	
<	__lt__	
>	__gt__	
<=	__le__	
>=	__ge__	
+	__add__	__radd__
-	__sub__	__rsub__
*	__mul__	__rmul__
/	__div__	__rdiv__
/(division 임포트)	__truediv__	__rtruediv__

2. 여러 가지 메서드

//	__floordiv__	__rfloordiv__
%	__mod__	__rmod__
**	__pow__	__rpow__
<<	__lshift__	__rlshift__
>>	__rshift__	__lshift__

eqop

```
class Human:
    def __init__(self, age, name):
        self.age = age
        self.name = name
    def __eq__(self, other):
        return self.age == other.age and self.name == other.name

kim = Human(29, "김상형")
sang = Human(29, "김상형")
moon = Human(44, "문종민")
print(kim == sang)
print(kim == moon)
```

실행결과

```
True
False
```

2. 여러 가지 메서드

❖ 특수 메서드

- 특정한 구문에 객체 사용될 경우 미리 약속된 작업 수행

메서드	설명
__str__	str(객체) 형식으로 객체를 문자열화한다.
__repr__	repr(객체) 형식으로 객체의 표현식을 만든다.
__len__	len(객체) 형식으로 객체의 길이를 조사한다.

clsstr

```
class Human:
    def __init__(self, age, name):
        self.age = age
        self.name = name
    def __str__(self):
        return "이름 %s, 나이 %d" % (self.name, self.age)

kim = Human(29, "김상형")
print(kim)
```

실행결과

```
이름 김상형, 나이 29
```

3. 유틸리티 클래스

❖ Decimal

- 정수 혹은 문자열 실수로 초기화
- 오차 없이 정확하게 10진 실수를 표현
  - 컴퓨터에서 이진 실수로 십진 실수를 정확하게 표현하기 어려움

floaterror

```
f = 0.1
sum = 0
for i in range(100):
    sum += f
print(sum)
```

실행결과

```
9.99999999999998
```

decimal

```
from decimal import Decimal

f = Decimal('0.1')
sum = 0
for i in range(100):
    sum += f
print(sum)
```

실행결과

```
10.0
```

3. 유틸리티 클래스

- 실수형과는 연산할 수 없음

- Context 객체
  - 연산 수행 방법을 지정
  - getcontext / setcontext 함수로 컨텍스트 변경
  - 같은 연산이라도 컨텍스트 따라 결과 다를 수 있음

컨텍스트	설명
BasicContext	유효 자리수 9, ROUND_HALF_UP 반올림
ExtendedContext	유효 자리수 9, ROUND_HALF_EVEN 반올림 처리
DefaultContext	유효 자리수 28, ROUND_HALF_EVEN 반올림 처리

3. 유틸리티 클래스

❖ Fraction

- 유리수를 표현
  - 분모와 분자를 따로 전달하여 분수 형태 숫자 표현함

```
Fraction([부호] 분자, 분모)
```

fraction

```
from fractions import *

a = Fraction(1,3)
print(a)
b = Fraction(8, 14)
print(b)
```

실행결과

```
1/3
4/7
```

3. 유틸리티 클래스

❖ array 모듈

```
array(타입코드, [초기값])
```

- 동일 타입 집합인 배열을 지원
- 대량 자료를 메모리 낭비 없이 저장 및 고속 액세스 가능

타입	C 타입	설명
b, B	char	1바이트의 정수
u		2바이트의 유니코드 문자(3.3 이후 지원 안 함)
h, H, i, I	shirt, int	2바이트의 정수
l, L	long	4바이트의 정수
q, Q	long long, __int64	8바이트의 정수(3.3이상에서만 지원)
f	float	4바이트의 실수
d	double	8바이트의 실수

Thank You !

파이썬 정복