

[Python] 파이썬 기초 10 - 클래스에 대한 정의와 사용법 — 나무늘보의 개발 블로그

노트북: 첫 번째 노트북

만든 날짜: 2020-10-23 오후 10:33

URL: <https://continuous-development.tistory.com/70?category=736681>

Python

[Python] 파이썬 기초 10 - 클래스에 대한 정의와 사용법

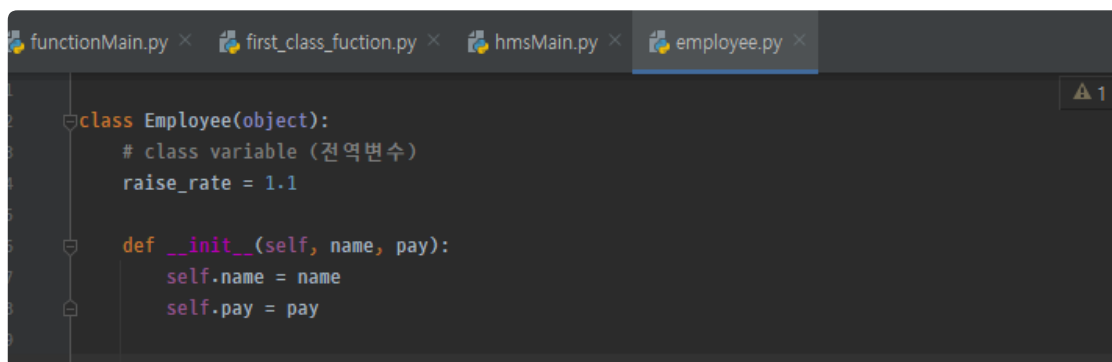
2020. 8. 18. 00:55 수정 삭제 공개

<클래스>

클래스란 하나의 틀로서 객체(인스턴스)를 생성하기 위한 틀이다. 틀을 통해 객체를 만드는 데 있어서 클래스의 사용은 같은 경우에는 불필요한 반복 작업을 줄여준다.

클래스의 구조는 객체의 구조(변수)와 행동(함수)을 정의한다. 현실 사물에 있는 객체(인스턴스)를 프로그램상으로 옮기는 작업이라고 생각하면 된다.

또한 클래스는 생성자를 통해 초기화가 필요하다.



```
functionMain.py × first_class_fuction.py × hmsMain.py × employee.py ×
class Employee(object):
    # class variable (전역변수)
    raise_rate = 1.1

    def __init__(self, name, pay):
        self.name = name
        self.pay = pay
```

class에서는 def __init__ 함수가 사용된다. 이 부분은 명시적으로 쓰지 않으면 묵시적으로 생성자가 존재한다. 그때는

`__init__(self)`: 로만 정의된 생성자 초기화 함수이다. 클래스에서는 기본적으로 생성자 호출을 통해 객체를 생성한다.

이 생성자를 거쳐야만 객체라는 것이 생긴다. 여기서 `self`는 자기 자신을 나타낸다. 고로 `self.name`은 인스턴스의 `name`을 매개변수 `name`이라는 것을 받아서 넣는다는 것을 의미한다.

```
package_function.py × oop_first_class.py × student.py × emp_caller.py ×
1 from service.oop.hms.employee import *
2
3 emp01 = Employee('임경희', '1000') # 생성자
4 emp02 = Employee('섭섭해', '2000') # 생성자
5
6 print(dir(emp01)) # 내장 되어 있는 함수를 보여준다. ['__class__', '__delattr__', 'ef__', 'raise_rate'] 멤버변수까지 보여준
7 print(emp01.raise_rate)
8
9 print(id(emp01)) # 2065438248800
10 print(id(emp02)) # 2065438248856
11 |
```

위에 정의해놓은 클래스를 생성자 호출(`__init__`)을 통해 객체를 생성하고 있다. 그리고 만들어진 객체에 대해서 서로 다른 주소 값을 가진다. 서로 다른 객체이기 때문이다.

그리고 `dir`을 사용하면 내장되어 있는 함수를 볼수있는데 내가 지정해놓은 `def` 외에 다른 `__?__` 것들이 있는 것이 보인다. 이것 같은 경우에는 클래스에는 기본적으로 상속을 받는 `object`라는 것이 있다. 이 `object`는 내장 함수로서 `class`의 기본 구성 요소가 된다. 이것이 들어가서 `dir`을 했을 때 내가 만든 함수 외에도 다른 함수들이 들어가 있다.

```
unctionMain.py × first_class_fuction.py × hmsMain.py × employee.py ×
class Employee(object):
    # class variable (전역변수)
    raise_rate = 1.1

    def __init__(self, name, pay):
        self.name = name
        self.pay = pay

    def appy_raise(self):
        self.pay = int(self.pay * Employee.raise_rate) # 클래스의 변수를 사용하였다. 여기서는 앞에

    def getEmp(self):
        return "{}님의 인상된 급여는 {} 입니다.".format(self.name, self.pay)
        # return "%s님의 인상된 급여는 %d 입니다." % self.name, self.pay
```

이 두가지의 `appy_raise` , `getEmp`의 함수를 만들었다. `appy_raise`는 클래스의 변수인 `raise_rate`를 받아 사용하고 있다.

이때 클래스의 변수를 사용하기 위해서는 해당 변수가 어떤 변수인지 명시해줘야 한다. 여기서는 `Employee`의 변수로서 나타내 진다.

```
package_function.py x oop_first_class.py x student.py x emp_caller.py x
1 from service.oop.hms.employee import *
2
3 emp01 = Employee('임정희', 1000) # 생성자
4 emp02 = Employee('섭섭해', 2000) # 생성자
5
6 print(dir(emp01)) # 내장 되어 있는 함수를 보여준다. ['__class__', '__delattr__', 'ef__', 'raise_rate'] 멤버변수까지
7 print(emp01.raise_rate)
8
9 print(id(emp01)) # 2065438248800
10 print(id(emp02)) # 2065438248856
11
12 emp01.appy_raise() # appy_raise 함수를 호출해서 사용
13 emp02.appy_raise()
14 print(emp01.getEmp()) # 임정희님의 인상된 급여는 1100 입니다.
15 print(emp02.getEmp()) # 섭섭해님의 인상된 급여는 2200 입니다.
```

해당 함수를 객체를 통해 호출한다.

classmethod / staticmethod

정적 메서드는 클래스에서 직접 접근할 수 있는 메서드이다. (원래 클래스에서 함수를 접근할 수 없다. 보통 소유의 주체가 인스턴스 이기 때문이다.)

classmethod

`classmethod`는 `@classmethod`라는 데코레이터를 사용하여 정의할 수 있다.

인스턴스 메서드와 달리 `self`라는 인자 대신 `cls`라는 인자를 가진다

staticmethod

파이썬에서 `staticmethod`는 아래와 같이 `@staticmethod`라는 데코레이터를 사용하여 정의할 수 있다. 인스턴스 메서드와 달리 `self`라는 인자를

가지고 있지 않다.

```
package_function.py x oop_first_class.py x student.py x emp_caller.py x
1 from service.oop.hms.employee import *
2
3 emp01 = Employee('임정희', 1000) # 생성자
4 emp02 = Employee('섭섭해', 2000) # 생성자
5
6 print(dir(emp01)) # 내장 되어 있는 함수를 보여준다. ['__class__', '__delattr__', 'ef__', 'raise_rate'] 멤버
7 print(emp01.raise_rate)
8
9 print(id(emp01)) # 2065438248800
10 print(id(emp02)) # 2065438248856
11
12 emp01.appy_raise() # appy_raise 함수를 호출해서 사용
13 emp02.appy_raise()
14 print(emp01.getEmp()) # 임정희님의 인상된 급여는 1100 입니다.
15 print(emp02.getEmp()) # 섭섭해님의 인상된 급여는 2200 입니다.
16
17
18 emp01.change_raise_rate1(1.5) # 이 두가지 경우 다 가능하다
19 emp02.change_raise_rate2(2.0) #
```

```
functionMain.py x first_class_fuction.py x hmsMain.py x employee.py x
4
5 def __init__(self, name, pay):
6     self.name = name
7     self.pay = pay
8
9 def appy_raise(self):
10     self.pay = int(self.pay * Employee.raise_rate) # 클래스의 변수를 사용하였다. 여기서는 앞에
11
12 def getEmp(self):
13     return "{}님의 인상된 급여는 {} 입니다.".format(self.name, self.pay)
14     # return "%s님의 인상된 급여는 %d 입니다." % self.name, self.pay
15
16 # 정적메소드는 클래스에서 직접 접근할 수 있는 메소드이다.
17 # 파이썬에서 클래스에서 직접 접근할 수 있는 메소드가 @staticmethod 와 @classmethod 두가지가 있다.
18 # classmethod 는 cls 라는게 self 처럼 들어와야 한다.
19 # 두 정적 메소드의 차이는 상속에서 차이가 난다.
20 @classmethod # 클래스 소유
21 def change_raise_rate1(cls, rate): # 매개변수가 self가아니라 함수의 소유가 정의되지 않았다.
22     cls.raise_rate = rate
23     print("인상률 {} 가 적용 되었습니다.".format(cls.raise_rate))
24
25 @staticmethod # 인스턴스 소유
26 def change_raise_rate2(rate): # 매개변수가 self가아니라 함수의 소유가 정의되지 않았다.
27     raise_rate = rate
28     print("인상률 {} 가 적용 되었습니다.".format(raise_rate))
```

staticmethod의 경우 부모 클래스의 클래스 속성 값을 가져오지만 classmethod의 경우 cls인자를 활용하여 현재 클래스의 클래스 속성을 가져온다.

객체를 생성하지 않고 클래스에 접근하는 방법

```
class Car(object):  
    # class variable  
    name = None  
    door = cc = 0
```

```
22 print("Car 객체 생성 후 작업 진행")  
23  
24 Car.name = 'Jeep' # 소유의 주체가 클래스여서 객체를 만들지 않아도 접근이 가능하다.  
25 Car.door = 4  
26 Car.cc = 2000  
27  
28 print("{} , {} , {}".format(Car.name, Car.door, Car.cc))
```

리모컨 예제

```

class TV(object):
    # 1. class variable
    channel = 10
    volume = 5
    power = False # True : on , False : off

    # 전원 관리를 위한 함수
    def changePower(self):
        self.power = not TV.power # 이 때 인스턴스 소유의 power가 생긴다.

    # 채널 변경을 위한 함수
    def channelUp(self):
        # TV.channel = TV.channel + 1
        self.channel += 1

    # 채널 변경을 위한 함수
    def channelDown(self):
        TV.channel -= 1

    def volumnUp(self):
        TV.volume += 1

    def volumnDown(self):
        TV.volume -= 1

    def display(self):
        print("전원상태 : {}, 채널번호 : {}, 볼륨 : {}".format(self.power, self.channel, self.volume))

```

```

# 여기서 생성자 호출을 통해 만든 순간 class의 변수가 인스턴스의 변수가 된다. class 를 통한 생성이니까 이게 들
# 단 클래스 소유의 변수를 가진다.
tv = TV()

# Question
tv.display()
# 1. 전원 on 시킨다.
tv.changePower()
tv.display()

# 🍷. 채널 18번으로 변경
tv.channelUp()
tv.channelUp()
tv.channelUp()
tv.channelUp()
tv.channelUp()
tv.channelUp()
tv.channelUp()
tv.channelUp()
tv.channelUp()
tv.display()

# 3. 볼륨 9로 변경
tv.volumnUp()
tv.volumnUp()
tv.volumnUp()
tv.volumnUp()
tv.display()
# 4. TV 상태를 출력한다.
tv.display()
tv.display()

```

'Python' 카테고리의 다른 글

[Python] 파이썬 기초 12 - 예외처리

[Python] 파이썬 기초 11 - 객체의 4대 특성 (상속화, 캡슐화, 다형성, 추상화)

[Python] 파이썬 기초 10 - 클래스에 대한 정의와 사용법

[Python] 파이썬 기초 9 - 패키지와 모듈에 대한 정의와 다양한 함수 형태

[Python] 파이썬 기초 8 - 반복문(for , while)에 대한 정의와 기본적인 함수 사...

[Python] 파이썬 기초 7 - 조건문(IF, elif ,else)에 대한 정의와 기본적인 함수 사...

python class

python 객체

python 클래스

파이썬 class

파이썬 클래스 정의



나무늘보스

혼자 끄적끄적하는 블로그 입니다.