Python implementation of OOP I (Class, Instance, Method)

ESM2014-41

객체지향프로그래밍 및 실습

SKKU 시스템경영공학과

조영일

Python : Pure Object-oriented Language

- Python을 구성하는 것은 모두 객체이다.
- Python의 모든 자료형(type)은 Class로 표현됨
- Python의 모든 변수는 Object로 표현됨

```
>>> type(10)
<class 'int'>
>>> type("A string")
<class 'str'>
>>> type([1, 2, 3])
<class 'list'>
>>> type({1, 2, 3})
<class 'set'>
```

Class 정의(Using class attribute)

```
class Movie:
    # Class Attribute
    name = ""
    genre = ""
    director = ""
    rating = 0 # Initial Value

    # Class Method
    def show_name(self):
        print("Movie name is " + self.name)

    def show_rating(self):
        print("Rating : " + str(self.rating))
```

Object from Class

```
new_movie = Movie()
new_movie.name = "Lion King"
new_movie.genre = "Animation"
new_movie.director = "Disney Pictures"

new_movie.show_name()
>>> "Lion King"
new_movie.show_rating()
>>> 0
```

def __init__ : Class Initializer(Using Instance Attribute)

- def __init__ : Magic Method – Class Initializer(초기화자), 객체의 초기화를 담당하는 method. 객체가 최초에 생성될 때 호출되어 객체 내의 Attribute(Instance Attribute) 값을 지정하고, 객체 생성 시점에 실행되어야 하는 로직들을 포함 시킬 수 있음. (주의 생성자(__new__)와는 다름)

```
class Movie:
    def __init__(self, name, genre, director, rating=0):
        self.name = name
        self.genre = genre
        self.director = director
        self.rating = rating

    # Class Method
    def show_name(self):
        print("Movie name is " + self.name)

    def show_rating(self):
        print("Rating : " + str(self.rating))


new_movie = Movie(name="Lion King", genre="animation", director="Disney Pictures")
new_movie.show_name()
```

Class Attribute vs Instance Attribute

- 앞서 정의한 두가지 방식은 같은 동작을 하는 것 같이 보이지만 실제로는 다르다.

- Class Attribute : 같은 클래스에서 생성된 객체들간 공통적으로 공유되는 값들을 저장 할 때 사용

- Instance Attribute : 개별 객체마다 다른 값을 저장할 때 사용, __init__ method를 사용하여 초기화 할 경우 모두 Instance Attribute.

Class Attribute vs Instance Attribute

```
class Movie:
    # Class Attribute
    name = "Ad Astra"

new_movie_1 = Movie()
new_movie_2 = Movie()

print(Movie.name)
>>> "Ad Astra"

print(new_movie_1.name)
>>> "Ad Astra"

print(new_movie_2.name)
>>> "Ad Astra"

# Modify Class Attribute
new_movie_1.name = "Lion King"

# 공유되지 않는 것처럼 보임
print(new_movie_2.name)
>>> "Ad Astra"
```

Class Attribute vs Instance Attribute

```
class Movie:
    # Class Attribute
    name = "Ad Astra"
    actor_list = []

    def add_actor(self, name):
        self.actor_list.append(name)

new_movie_1 = Movie()
new_movie_2 = Movie()

print(new_movie_1.actor_list)
>>> []

print(new_movie_2.actor_list)
>>> []

new_movie_1.add_actor("Brad Pitt")

# 공유되는 것으로 보이는 결과
print(new_movie_2.actor_list)
>>> ["Brad Piit"]
```

Class Attribute vs Instance Attribute

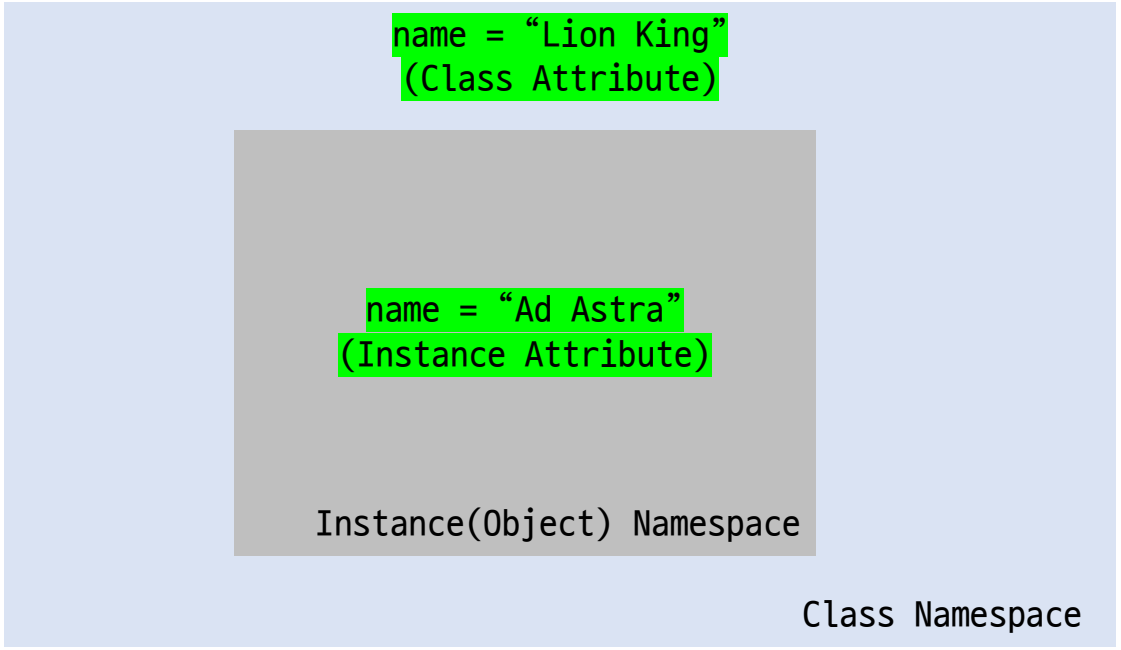
- Why does this difference occur?

- Immutable(String, Int..) 과 Mutable(List..) 사이의 차이

- Class Attribute의 경우 같은 클래스로 부터 생성된 객체들 간에는 공유하도록 설계되어 있지만, Immutable의 경우 그 자료형의 특성 상 객체가 생성된 경우 값을 변경 할 수 없기 때문에 값을 변경하는 순간 Instance Attribute로 변경됨

Attribute Namespace

- Instance Attribute 와 Class Attribute 의 이름이 같을 경우에는?
- Attribute Namespace : 클래스와 그로부터 생성된 객체 사이에서 Attribute(변수)의 이름이 유효하게 참조 되는 공간
- Instance(Object)와 Class는 각각의 Namespace를 가진다.
- 항상 객체 내의 Instance Attribute 가 Class Attribute에 비해 우선한다.



Attribute Namespace

- __init__ 과 self. 를 사용하여 객체를 초기화 할 경우 : 항상 Instance Attribute로 선언되므로 객체 간 값이 공유되는 현상을 피할 수 있다.

- General Rule : 하나의 Class로 부터 생성된 모든 객체가 공통적으로 값을 공유 해야 하는 경우가 아니라면 __init__ method를 구현하고 Instance Attribute를 사용 할 것

self keyword & Class Method

- self : class로 부터 생성된 객체 자기 자신을 의미하며, class method 의 경우 반드시 method 정의 시점에 첫번째 인수로 self를 명시적으로 기입하여야 함

- class method는 class 코드 블록 내에 정의 되며, 해당 class로 부터 생성되는 모든 객체는 해당 method를 가지게 된다.

```
class Circle:
    def __init__(self, radius):
        self.radius = radius

    def change_radius(self, radius):
        self.radius = radius

    def get_area(self):
        return self.radius * self.radius * 3.141592

c = Circle(radius=5)
c.change_radius(6)
c.get_area()
>>> 113.097312
```

Magic Method in Class

- Magic Method : 일반적인 Method 들과는 달리 특정한 기능을 하기 위해 Python Syntax 상 지정되어 있는 Method 들(__xx__)

- __new__ method : 생성자. 객체 생성단계 최초에 호출되며, 클래스가 새로운 객체를 만드는 방법을 포함하는 메서드(Python에서 기본으로 제공하므로 사용할 일이 거의 없음)

- __init__ method : 초기화자, 객체 생성 시 __new__ 다음으로 호출되어 객체의 내용을 초기화(initialize) 하는데 사용됨

- __del__ method : 소멸자, 객체가 소멸될 때의 동작을 정의

- __repr__ method : 객체를 출력(print()) 했을 때 표시될 값을 정의

Practice

- 다음과 같은 Attribute / Method 를 가지는 Class를 작성하고, Class 로 부터 Object를 생성해보세요.(__init__ 포함)
- Student Class
  - Attribute
    - 이름 <string>
    - 학번 <string>
    - 전공 <string>
    - 수강교과목 <list>
    - 수강교과목 중간고사 점수 <dict>
    - 중간고사 평균 점수 <float>
  - Method
    - __init__ : 초기화 시 이름, 학번, 전공, 수강교과목, 수강교과목 중간고사 점수를 arguments로 받아 객체의 attribute에 할당하고, 중간고사 평균 점수는 해당 자료를 토대로 계산하여 넣는다.
    - get_lecture_count : 수강교과목의 개수를 반환하는 함수
    - get_summary : 해당 학생의 attribute 를 요약하여 print()하는 함수
    - change_major : 해당 학생의 전공 attribute 를 변경하는 함수

Practice

https://gist.github.com/youngilcho/07fe905b6c0ec0f64d2922f8140734f8

Summary

- Class Definition & Object
- Class Initializer
- Name Space
- Class Method
- Class Magic Method