객체지향프로그래밍 이해하기 1

13주차_01

한동대학교
김경미 교수

객체지향프로그래밍
(Object Oriented Programming)

- **사물을 객체(object)로 정의**
  - 객체는 속성(property)과 기능(method)으로 구성
  - 각 개체 간의 상호작용은 메시지로 송수신

- **프로그램에서 사용되는 모든 것들을 객체로 정의**

- **객체를 만들기 위해 '클래스(class)'라는 도구 제공**

객체지향프로그래밍 용어

- **Class**
  - 변수, 속성, method를 포함한 데이터 형태를 정의한다
- **Instance**
  - 실행 시 만들어진 class의 한 객체이다
  - 객체의 class들 안에서 정의된 상태와 행동으로 구성되어 있다
- **Method**
  - 한 class 안에서 정의되고, 그 class의 class data와 instance에서 동작한다
- **http://www.python-course.eu/python3_object_oriented_programming.php**

Class 1

- **Python class는 프로그래밍을 목적으로 객체에 기준이 되는 특징(Standard feature)을 제공한다**
  - 상속(inheritance) 체계를 가진다
  - Base class나 class내 method들을 override한다
  - 한 method는 같은 이름의 base class내 method를 호출할 수 있다
- **객체들은 data 크기와 종류를 포함할 수 있다**

Class 2

- **지금까지 활용한 모듈들은 누군가가 이미 만들어 놓은 사전과 같다**
- **Class는 자신이 스스로 만든 작은 사전과 같다**
  - 여러 기능이나 data응답을 그룹화하여 하나의 상자에 넣는다
  - '.'오퍼레이터를 이용해서 그것들에 접근할 수 있다.
  - 우리는 이미 몇몇 Class를 살펴보았다(str, turtle, tkinter..)

Class 3

- **Class정의의 가장 간단한 형태**

```
class Person:                                  # Class 정의, 이름
      name = "Default Name"                    # member 변수
      def Print(self):                         # method 정의
            print("Hello, my name is", self.name)


p1 = Person( )                                 # instance 객체 생성
p1.Print()                                     # member 변수 값 출력

>>> "Hello, my name is Default Name"
```

Instance and method

- **앞의 예제에서**
  - Method를 정의할 때는 인수 'self' 적어야 한다
    - Method name은 Print()이다
  - p1은 instance 이름이다
    - 함수를 호출하는 형태로 class을 assign 하면
    - 해당 class 객체와 동일한 데이터와 method를 사용 가능하다
  - 대소문자를 구분한다
    - Class name, method name, instance name

Class 생성자, 소멸자 1

- **Instance 객체를 생성 할 때 초기화 기능 제공**
  - 기본 method __init__() 를 정의한다
  - Instance 객체가 생성될 때 자동으로 호출되어 실행된다
- **소멸자 method는 instance 객체의 레퍼런스 카운트가 '0' 될 때 호출**
  - 기본 method __del__() 를 정의한다

Class 생성자, 소멸자 2

```
class MyClass:
    def __init__(self, value):
        self.value = value
        print('Class is created!, Value = ', self.value)

    def __del__(self):
        print('Class is deleted!')

d = MyClass(10)                    # Object instantiation syntax

d1=MyClass(20)
```

```
>>> d = MyClass(10)
Class is created!, Value =  10
Class is deleted!
>>> f()
Class is created!, Value =  20
Class is deleted!
>>>
```

Class, instance 활용

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

    def Hello(self):
        print('Hello, my name is', self.name)

    def __del__(self):
        print(self.name, 'says bye!')

p = Person("Kyungmi Kim")          # Object instantiation syntax
print(p.name)                      # Attributes invoke
p.Hello()                          # methods invoke
```

```
>>>
Kyungmi Kim
Hello, my name is Kyungmi Kim
>>>
```

연습문제 1

- **이전 슬라이드에서 정의한**
  - class MyClass, Person 을 사용하여
  - Instance 2개를 정의해 보세요

연습문제 1 코드

```
class MyClass:
    def __init__(self, value):
        self.value = value
        print('Class is created!, Value = ', self.value)

    def __del__(self):
        print('Class is deleted!')

class Person:
    def __init__(self, name):
        self.name = name

    def Hello(self):
        print('Hello, my name is', self.name)

    def __del__(self):
        print(self.name, 'says bye!')
```

```
Instance01 = MyClass(100)
Instance02 = Person("Kyungmi kim")

print(Instance01.value)
Instance02.Hello()
```

```
>>>
====== RESTART: D:/1_Works/2017Work/KMooc
Class is created!, Value =  100
100
Hello, my name is Kyungmi kim
>>>
```

Class, method

- **Class를 정의하면서, 사용 가능한 함수를 정의한 것을 method라고 한다**

```
class BankAccount:
    def __init__(self):
        self.balance = 0

    def withdraw(self, amount):
        self.balance -= amount
        return self.balance

    def deposit(self, amount):
        self.balance += amount
        return self.balance

a = BankAccount()
b = BankAccount()

print(a.deposit(200))
print(b.deposit(100))

print(a.withdraw(20))
print(b.withdraw(20))
```

```
200
100
180
80
>>>
```

Instance and method

```
class Employee:
   empCount = 0

   def __init__(self, name, salary):
      self.name = name
      self.salary = salary
      Employee.empCount += 1

   def displayCount(self):
     print("Total Employee  = ", Employee.empCount)

   def displayEmployee(self):
      print("Name : ", self.name,  ", Salary: ", self.salary)

emp1 = Employee("Kim", 2000)
emp2 = Employee("Choi", 5000)
emp3 = Employee("Park", 4500)

emp1.displayEmployee()
emp2.displayEmployee()
emp3.displayEmployee()

print("Total Employee: ", Employee.empCount)
```

```
>>>
Name :  Kim , Salary:  2000
Name :  Choi , Salary:  5000
Name :  Park , Salary:  4500
Total Employee: 3
>>>
```

연습문제 2

- 이전 슬라이드의 class에서 salary를 모두 저장 한 후, 평균을 계산하는 과정을 추가하시오

연습문제 2 코드(1)

```
class Employee:
   empCount = 0
   TotalSalary = 0
   MeanSalary = 0

   def __init__(self, name, salary):
      self.name = name
      self.salary = salary
      Employee.empCount += 1
      Employee.TotalSalary += salary
      Employee.MeanSalary = Employee.TotalSalary / Employee.empCount

   def displayCount(self):
      print("Total Employee : ", Employee.empCount)

   def displayEmployee(self):
      print("Name : ", self.name,   ", Salary: ", self.salary)

   def displayMeanSalary(self):
      print("Mean Salary : ", Employee.MeanSalary)

    # continue to...
```

연습문제 2 코드(2)

```
emp1 = Employee("Kim", 2000)
emp2 = Employee("Choi", 5000)
emp3 = Employee("Park", 4500)

emp1.displayEmployee()
emp2.displayEmployee()
emp3.displayEmployee()

print("Total Employee : ", Employee.empCount)

emp3.displayMeanSalary()
```

```
>>>
Name :  Kim , Salary:  2000
Name :  Choi , Salary:  5000
Name :  Park , Salary:  4500
Total Employee :  3
Mean Salary :  3833.3333333333335
>>>
```

숙제

- 연습문제 1, 2를 입력한 코드와
- 실행 결과를 캡쳐하여 게시판에 올리시오

요약

- 객체지향 프로그래밍에서 사용하는 용어를 이해한다
- Class 사용하여 object, method, instance를 작성한다

감사합니다

13주차_01 객체지향프로그래밍 이해하기 1