데이터 처리를 위한 Python 프로그래밍 입문

11강. 객체지향 프로그래밍

ERICA 2018-2

강의 내용

- 개요
- 객체지향 프로그래밍

개요

- 객체 지향 프로그래밍 언어
  - (Object Oriented Programming Language)
  - 객체를 우선적으로 생각하여 프로그래밍
- 클래스 기반 객체 지향 프로그래밍 언어
  - (Class based ~)
  - 클래스 기반으로 객체 만듦
- 파이썬 이외 다른 프로그래밍 언어들
  - 자바스크립트, PHP, C++, 스칼라 등

객체지향 프로그래밍(1/9)

- 클래스 (Class)
  - 타입, 실체가 없는 개념
  - 클래스의 모양과 생성

```
class 클래스명:
    # 이 부분에 관련 코드 구현
```

  - 현실 세계의 사물을 컴퓨터 안에서 구현하려고 고안된 개념

객체지향 프로그래밍(2/9)

- 메소드 (Method)
  - 클래스 안에서 구현된 함수
  - 메소드의 모양과 생성

```
class 클래스명:

    def upSpeed(self, value):
        self.speed = value
```

  - self는 클래스 자신을 말한다.
  - upSpeed() 메소드의 매개변수는 value 하나뿐이다.

객체지향 프로그래밍(3/9)

- 인스턴스 (Instance)
  - 실제 객체가 생성
  - 자동차 세 대의 인스턴스 생성 코드

```
myCar1 = Car()
myCar2 = Car()
myCar3 = Car()
```

  - 인스턴스를 객체라고도 한다.

객체지향 프로그래밍(4/9)

- 클래스의 작동 구현

```
## 클래스 선언 부분 ##
class  Car :
    color = ""
    speed = 0

    def  upSpeed(self, value) :
        self.speed += value

## 메인 코드 부분 ##
myCar1 = Car()
myCar1.color = "빨강"
myCar1.speed = 0

myCar2 = Car()
myCar2.color = "파랑"
myCar2.speed = 0

myCar1.upSpeed(30)
print("자동차1의 색상은 %s이며, 현재 속도는 %dkm입니다."
      % (myCar1.color, myCar1.speed))

myCar2.upSpeed(60)
print("자동차2의 색상은 %s이며, 현재 속도는 %dkm입니다."
      % (myCar2.color, myCar2.speed))
```

- 실행결과

```
자동차1의 색상은 빨강이며, 현재 속도는 30km입니다.
자동차2의 색상은 파랑이며, 현재 속도는 60km입니다.
>>>
```

객체지향 프로그래밍(5/9)

- 생성자 (Constructor)
  - 인스턴스를 생성하면서 필드값을 초기화시키는 함수
  - 생성자의 기본 형태

```
class 클래스명:

    def __init__(self):
        # 이 부분에 초기화할 코드 입력
```

  - 언더바가 2개 붙은 것은 파이썬에서 예약해 놓은 것이다.

객체지향 프로그래밍(6/9)

- 생성자 작동 구현

```
## 클래스 정의 부분 ##
class  Car :
    name = ""
    speed = 0

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

    def  getName(self) :
        return self.name

    def getSpeed(self) :
        return self.speed

## 메인 코드 부분 ##
car1 = Car("빨강", 0)
car2 = Car("파랑", 30)

print("자동차1의 색상은 %s이며, 현재 속도는  %d입니다."
      % (car1.getName(), car1.getSpeed()))
print("자동차2의 색상은 %s이며, 현재 속도는  %d입니다."
      % (car2.getName(), car2.getSpeed()))
```

- 실행결과

```
자동차1의 색상은 빨강이며, 현재 속도는  0입니다.
자동차2의 색상은 파랑이며, 현재 속도는  30입니다.
```

객체지향 프로그래밍(7/9)

- 상속(Inheritance)의 개념
  - 기존 클래스에 있는 필드와 메소드를 그대로 물려받은 새로운 클래스를 만드는 것
  - 상속을 구현하는 문법

```
class 서브_클래스(슈퍼_클래스):
    # 이 부분에 서브 클래스의 내용 코딩
```

  - 상위 클래스를 슈퍼 클래스 또는 부모 클래스라 한다.
  - 하위 클래스를 서브 클래스 또는 자식 클래스라 한다.

객체지향 프로그래밍(8/9)

- 매소드 오버라이딩(Overriding)
  - 사위 클래스의 매소드를 서브 클래스에서 재정의
  - 메소드 오버라이딩 구현 코드

```
class Car:

    def upSpeed(self, value):
        self.speed += value

class Truck(Car):

    def upSpeed(self, value):    # 서브 클래스(Truck)의 upSpeed()메소드를 다시 정의
        self.speed += value

        if self.speed > 110:
            self.spee = 110
```

객체지향 프로그래밍(9/9)

- 상속과 메소드 오버라이딩 작동 구현

```
# 클래스 정의 부분
class Car:
    speed = 0

    def upSpeed(self, value):
        self.speed += value

        print("현재 속도(슈퍼 클래스) : %d" %self.speed)

class Sedan(Car):
    pass

class Truck(Car):
    def upSpeed(self, value):
        self. speed += value

        if self.speed > 110:
            self.speed = 110

        print("현재 속도(서브 클래스): %d" %self.speed)
```

```
# 메인 코드 부분
sedan1 = Sedan()
truck1 = Truck()

print("==승용차==")
sedan1.upSpeed(200)

print("==트  럭==")
truck1.upSpeed(200)
```

- 실행결과

```
==승용차==
현재 속도(슈퍼 클래스) : 200
==트  럭==
현재 속도(서브 클래스): 110
>>>
```

Thank you