클래스

Update – 2018.07

Contents

- 클래스 개요
- 클래스 생성
- 객체 개요
- 객체 생성
- 속성(Attribute)
- 메소드(Method)
- 상속(Inheritance)

클래스 개요

- 클래스(Class)란?

  똑같은 무엇인가를 계속해서 만들어낼 수 있는 설계 도면

- 클래스(Class)의 장점

  1. 코드의 재사용
  2. 구조화
  3. 업데이트 용이
  4. 코딩의 단순화

클래스 개요

- 클래스(Class)란?
  - 똑같은 무엇인가를 계속해서 만들어낼 수 있는 설계 도면
  - 다수의 변수와 다수의 함수 집합
  - 고유한 특성안에서 변경은 가능

- 클래스(Class)의 장점
  1. 코드의 재사용
  2. 구조화
  3. 업데이트 용이
  4. 코딩의 단순화

클래스 개요

클래스는 비효율의 극복하기 위한 것!

클래스 사례#1 붕어빵 설명

"붕어빵 장사를 시작했다. 한 땀 한 땀 장인정신으로 붕어빵을 만들고 있다."

"옆에서 붕어빵틀로 마구마구 붕어빵을 만들어 낸다. 나도 붕어빵 틀이 필요해ㅠ_ㅠ"

완벽하게 같지는 않지만 붕어빵을 찍어낼 수 있는 붕어빵 틀은 클래스이고

붕어빵은 붕어빵 틀을 구체화(instance)한 객체(object)이다.

붕어빵(객체)의 추상화가 붕어빵 틀(클래스)

- 고유한 특성 안에서는 변경은 가능

속 재료를 달리 넣을 순 있어도 붕어빵 틀에서 사과, 금이 나오진 않는다.

클래스 개요

과자틀 → 클래스 (class)

과자틀에 의해서 만들어진 과자들 → 객체 (object)

클래스 개요

객체(Object) = 속성(Attribute) + 기능(Method)

- 속성은 사물의 특징

  예) 자동차의 속성 : 바디의 색, 바퀴의 크기, 엔진의 배기량
- 기능은 어떤 것의 특징적인 동작

  예) 자동차의 기능 : 전진, 후진, 좌회전, 우회전

- 속성과 기능을 들어 자동차를 묘사하면?

  "18인치의 바퀴를 가진 2,000cc의 빨간 차는 전진, 후진, 좌회전, 우회전의 기능이 있다."

클래스 생성

- 클래스 정의

class 클래스이름:
    코드블록

```
class Square:
    pass

>>> print(type(Square))
<class 'type'>
```

객체 개요

- 인스턴스(Instance)란?

  "구체적인 사례로 만들다"

- 객체(Object)란?
  - 클래스를 인스턴스화한 것
  - 클래스에 의해서 만들어진 피조물
  - 인간 클래스의 인스턴스화한 것(=객체)은 홍길동이다

- 객체 지향 프로그래밍
  - OOP: objected-oriented programming
  - 객체를 기본으로 프로그래밍 한다
  - 자바, 파이썬

객체 생성

- 객체 생성

객체 = 클래스()

```
class Square:
    pass
Square = Square()

>>> print(type(square))
<class '__main__.Square'>
```

속성(Attribute)

- 속성(attribute)란?
  - 클래스에서 정의된 변수
  - 클래스안에 생성자 함수인 __init__() 함수안에 정의
  - self 예약어 사용

```
class 클래스이름:
        def __init__(self, 인자1):
                self.인자1 = 인자2
```

생성자 함수(인자 입력때 self는 무시), 사용할 인자만 가능

생성자 함수(__init__)의 인자 입력해서 객체 생성

```
객체이름 = 클래스이름(인자값)
객체이름.인자
```

인스턴스화

속성(Attribute)

```
class Square:
    def __init__(self,height,width):
        self.height = height
        self.width = width
```

```
square1 = Square(50,100)
print('square1의 높이는 {} 입니다'.format(square1.height))
print('square1의 가로 크기는 {} 입니다'.format(square1.width))
print('square1의 넓이는 {} 입니다'.format(square1.width*square1.height))
```

```
>>>
square1의 높이는 50 입니다
square1의 가로 크기는 100 입니다
square1의 넓이는 5000 입니다
```

메소드(Method)

- 메소드(Method)?
  - 클래스에서 정의된 코드블록의 묶음.
  - 함수와 비슷하다.

```
class 클래스이름:
        def 메소드이름(self):
                코드블록
                return value
```

인스턴스화

```
객체이름 = 클래스이름(인자값)
객체이름.메소드이름()
```

메소드(Method)

```
class Square:
    def __init__(self,height,width):
        self.height = height
        self.width = width
    def area(self):
        print('높이 : {} , 가로: {} '.format(self.height,self.width))
        area = self.height * self.width
        return area
```

```
sq = Square(30,50)
print('넓이: ', sq.area())
```

```
>>>
높이 : 30 , 가로: 50
넓이:  1500
```

상속(Inheritance)

- 상속(Inheritance)?
  - 물려받다
  - 어떤 클래스를 만들 때 다른 클래스의 기능을 물려받을 수 있게 만드는 것

```
class 클래스명(상속할 클래스명):
        메소드 또는 속성
```

상속(Inheritance)

```
class Square:
    def __init__(self,height,width):
        self.height = height
        self.width = width
    def area(self):
        print('높이 : {} , 가로: {} '.format(self.height,self.width))
        area = self.height * self.width
        return area
```

```
class SquareAdd(Square):
    def __init__(self,height,width,color):
        self.color = color
        self.height = height
        self.width = width
    def printColor(self):
        print('사각형의 색상 : ',self.color)
```

상속(Inheritance)

```
sq = SquareAdd(50,50,'red')
sq.area()
sq.printColor()
```

```
>>>
높이 : 50 , 가로: 50
2500
사각형의 색상 :  red
```