문화 A0019

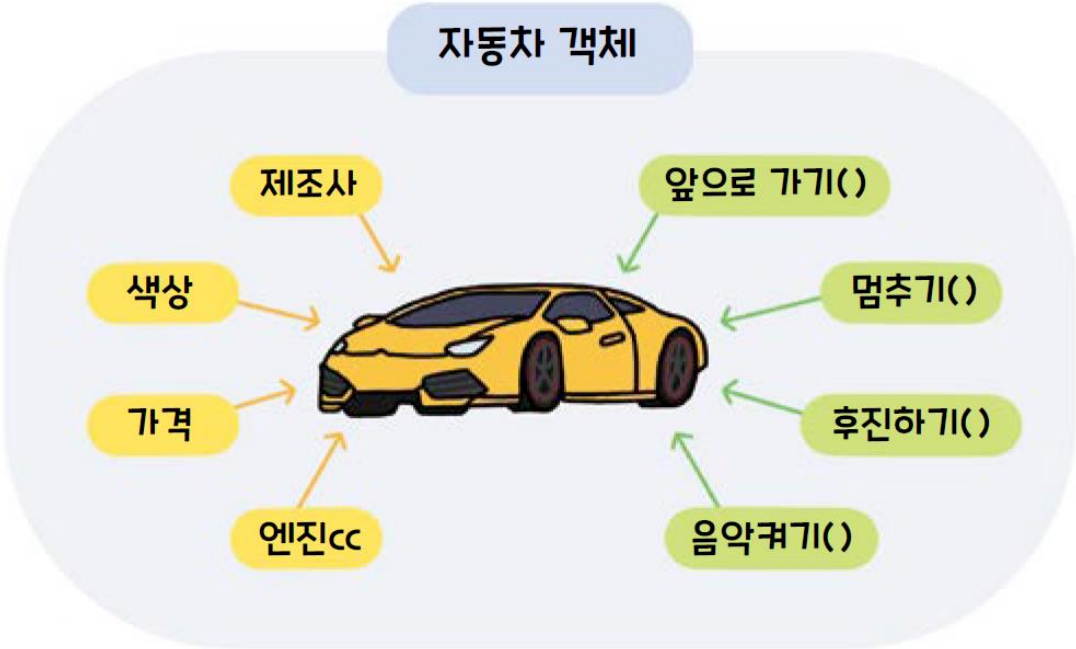
파이썬프로그래밍

김 태 완

kimtwan21@dongduk.ac.kr

객체

- 객체
  - 수많은 사물을 프로그래밍 관점에서 객체(Object 또는 Instance)라고 부름
    - 자동차, 건물, 고양이, 물고기 등
  - 객체의 정의
    - 어떤 속성과 행동을 가지고 있는 데이터
- 객체들로부터 공통적 특징을 뽑아 Class를 만드는 과정을 **추상화(Abstraction)**라고 함



객체와 클래스

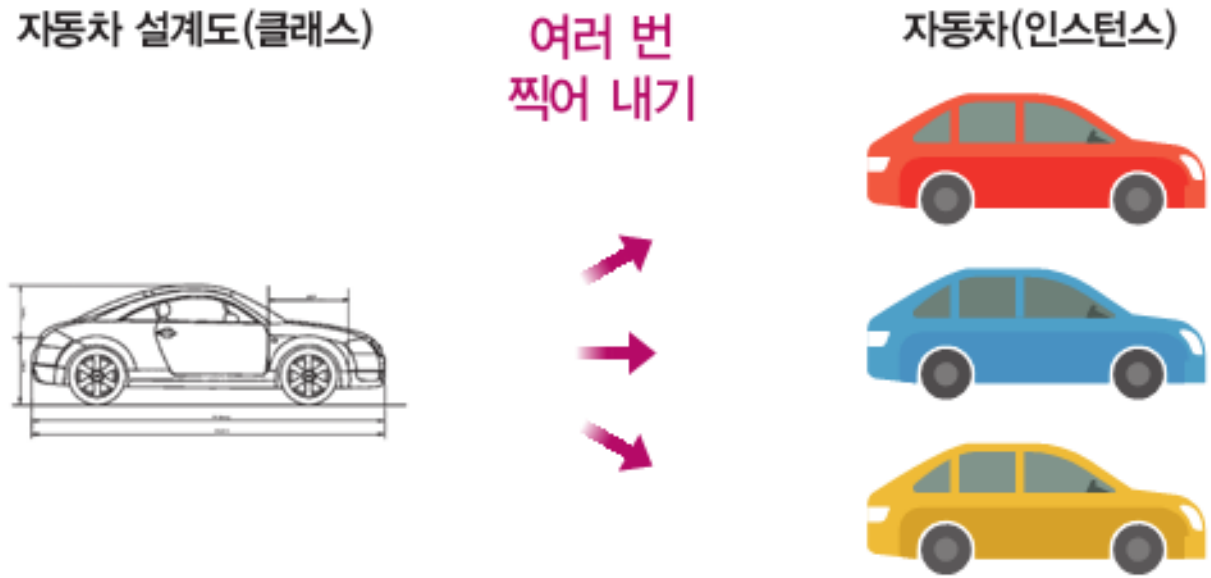
- 클래스 (class)
  - Instance(인스턴스)는 Class라는 틀을 이용해 Object(객체)를 만든 것
  - Class를 이용해 객체를 만드는 과정을 Instantiation(인스턴스화)라고 함
  - 하나의 Class로 여러 개의 Instance를 찍어낼 수 있음
  - 주로 Instance 와 Object는 같은 의미로 쓰임

Class

Instance

Instance

객체와 클래스



자동차 설계도(클래스)

```
class 자동차 :
  # 자동차의 속성
    색상
    속도
  # 자동차의 기능
    속도 올리기()
    속도 내리기()
```

자동차(인스턴스)

```
자동차1 = 자동차()
자동차2 = 자동차()
자동차3 = 자동차()
```

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

객체와 클래스

- 인스턴스 (객체)의 필드에 값 대입

색상 ← 빨강
속도 ← 0km

색상 ← 파랑
속도 ← 0km

색상 ← 노랑
속도 ← 0km

```
myCar1 = Car()
myCar1.color = "빨강"
myCar1.speed = 0

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

myCar3 = Car()
myCar3.color = " 노랑"
myCar3.speed = 0
```

객체와 클래스

```
class Car:
    color = ""
    speed = 0

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

    def downSpeed(self, value):
        self.speed -= value

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 %dkm/h 입니다."%(myCar1.color, myCar1.speed))

myCar2.upSpeed(50)
print("자동차2의 색상은 %s이며, 현재 속도는 %dkm/h 입니다."%(myCar2.color, myCar2.speed))
```

##변수 초기값 설정 (선언과 동시에 할당)

객체와 클래스

```
class Car:
    color = ""
    speed = 0

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

    def downSpeed(self, value):
        self.speed -= value

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 %dkm/h 입니다."%(myCar1.color, myCar1.speed))

myCar2.upSpeed(50)
print("자동차2의 색상은 %s이며, 현재 속도는 %dkm/h 입니다."%(myCar2.color, myCar2.speed))
```

##클래스 내부 함수의 첫 번째 인자는
반드시 self(클래스 자기자신)

##지역변수나 함수의 인자의 경우
self를 사용하지 않음

객체와 클래스

```
class Car:
    color = ""
    speed = 0

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

    def downSpeed(self, value):
        self.speed -= value

myCar1 = Car()
myCar1.color = "빨강"
myCar1.speed = 0

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

myCar1.upSpeed(30)
print("자동차1의 색상

myCar2.upSpeed(50)
print("자동차2의 색상은 %s이며, 현재 속도는 %dkm/h 입니다."%(myCar2.color, myCar2.speed))
```

##인스턴스.Method() 로 호출할 경우 클래스 멤버 함수의
첫 번째 인자(self)에 자동으로 인스턴스가 들어간다

객체와 클래스

- Step 1 : 클래스 선언
- Step 2 : 인스턴스 생성
- Step 3 : 필드나 메서드 사용

```
class Car:
    color = ""
    speed = 0

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

    def downSpeed(self, value):
        self.speed -= value

myCar1 = Car()
myCar1.color = "빨강"
myCar1.speed = 0

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

myCar1.upSpeed(30)
myCar2.upSpeed(50)
```

생성자

- 객체를 생성하면 무조건 호출되는 메소드를 의미함
- 객체를 생성하면서 변수의 값을 초기화하는 메소드
- 생성자로 초기화를 하면 코드가 간결해짐
- 생성자는 클래스 안에서 __init__()라는 이름으로 지정되어 있음

```
class Car:
    color = ""
    speed = 0

    def __init__(self) :
        self.color = "빨강"
        self.speed = 0
```

__init__()는 앞뒤에 언더바(_)가 2개씩 있습니다. init은 Initialize의 약자로 초기화 한다는 의미를 갖습니다.

생성자

```
class Car:
    color = ""
    speed = 0

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

    def downSpeed(self, value):
        self.speed -= value

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 %dkm/h 입니다."%(myCar1.color, myCar1.speed))

myCar2.upSpeed(50)
print("자동차2의 색상은 %s이며, 현재 속도는 %dkm/h 입니다."%(myCar2.color, myCar2.speed))
```

생성자

```
class Car:
    color = ""
    speed = 0

    def __init__(self) :
        self.color = "빨강"
        self.speed = 0

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

    def downSpeed(self, value):
        self.speed -= value

myCar1 = Car()
myCar2 = Car()

myCar1.upSpeed(30)
print("자동차1의 색상은 %s이며, 현재 속도는 %dkm/h 입니다."%(myCar1.color, myCar1.speed))

myCar2.upSpeed(50)
print("자동차2의 색상은 %s이며, 현재 속도는 %dkm/h 입니다."%(myCar2.color, myCar2.speed))
```

생성자 – 매개변수가 있는 경우

```
class Car:
    color = ""
    speed = 0

    def __init__(self, value1, value2) :
        self.color = value1
        self.speed = value2

myCar1 = Car("빨강", 30)
myCar2 = Car("파랑", 50)

print("자동차1의 색상은 %s이며, 현재 속도는 %dkm/h 입니다."%(myCar1.color, myCar1.speed))
print("자동차2의 색상은 %s이며, 현재 속도는 %dkm/h 입니다."%(myCar2.color, myCar2.speed))
```

생성자 – 매개변수가 있는 경우

```
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

myCar1 = Car("아우디", 30)
myCar2 = Car("벤츠", 50)

print("자동차1의 이름은 %s이며, 현재 속도는 %dkm/h 입니다."%(myCar1.getName(), myCar1.getSpeed()))
print("자동차2의 이름은 %s이며, 현재 속도는 %dkm/h 입니다."%(myCar2.getName(), myCar2.getSpeed()))
```

소멸자

- __init__( ) 메소드가 생성자라면, __del__( ) 메소드는 소멸자라고 부름
- __del( )__는 객체가 제거될 때 자동으로 호출됨
- 객체를 제거할 때는 del(객체)로 지우는데, 이때 호출됨

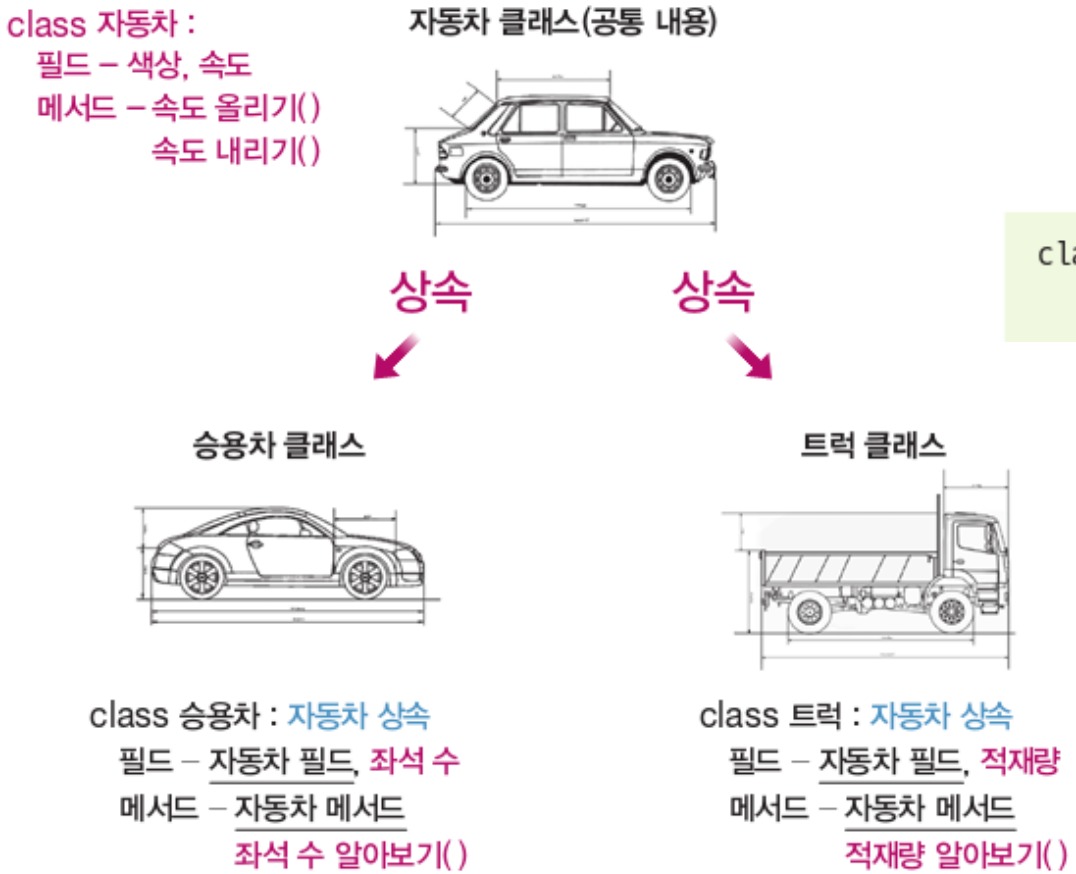
```
class Car:
    color = ""
    speed = 0

    def __del__(self) :
        print("이제", self.name, "차량은 없습니다." )

del(myCar1)
del(myCar2)
```

클래스 변수와 인스턴스 변수

```
class Car:
    color = ""
    speed = 0
    count = 0

    def __init__(self, value1, value2) :
        self.color = value1
        self.speed = value2
        Car.count += 1

myCar1 = Car("빨강", 30)
print("자동차1의 색상은 %s이며, 현재 속도는 %dkm/h, 차량의 총 대수는 %d 입니다."%(myCar1.color,
myCar1.speed, myCar1.count))

myCar2 = Car("파랑", 50)
print("자동차2의 색상은 %s이며, 현재 속도는 %dkm/h, 차량의 총 대수는 %d 입니다."%(myCar2.color,
myCar2.speed, myCar2.count))
```

상속

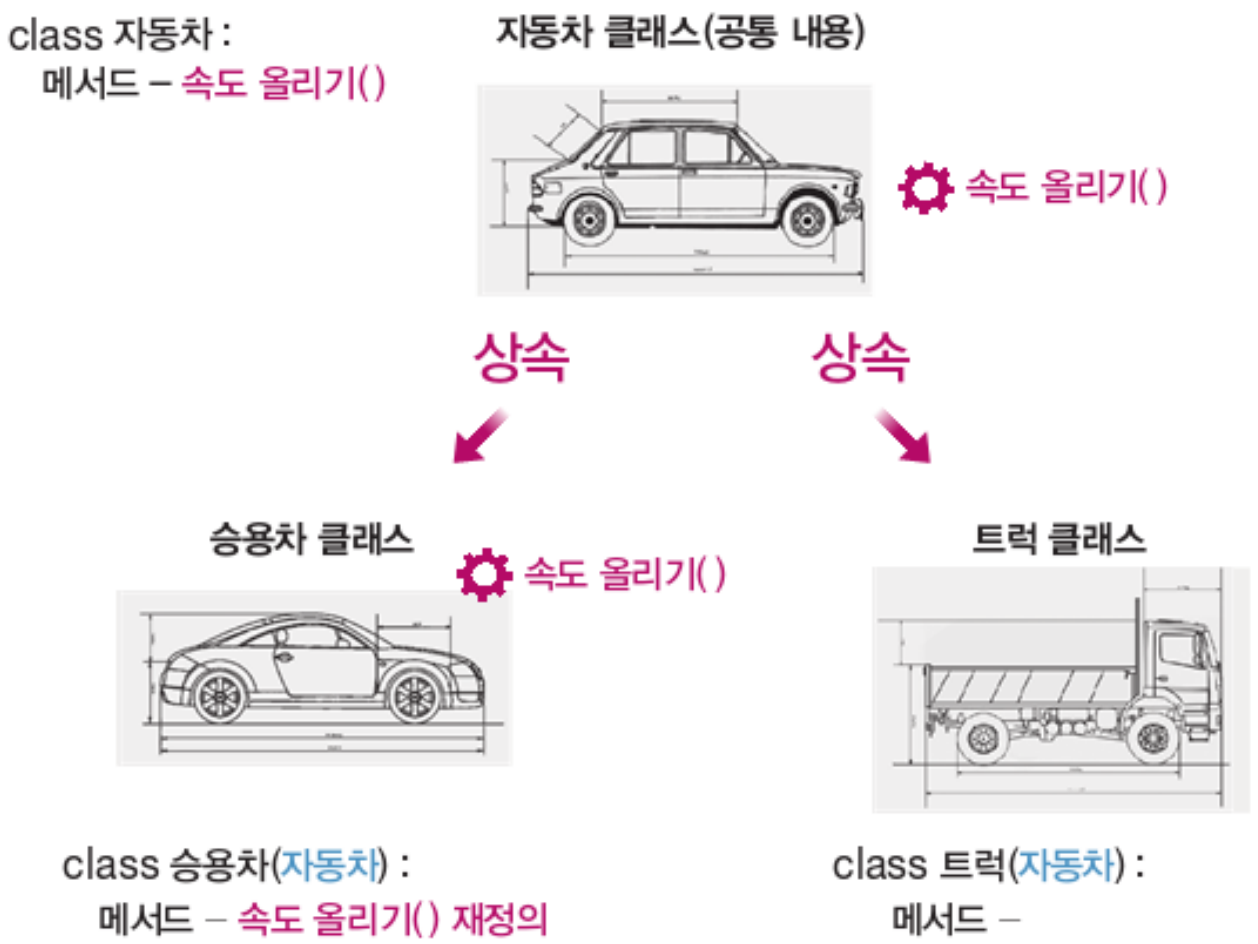
- 클래스의 상속(Inheritance) : 기존 클래스에 있는 필드와 메서드를 그대로 물려받는 새로운 클래스를 만드는 것
- 공통된 내용을 자동차 클래스에 두고 상속을 받음으로써 일관되고 효율적인 프로그래밍 가능
- 상위 클래스인 자동차 클래스를 슈퍼 클래스 또는 부모 클래스
- 하위의 승용차와 트럭 클래스는 서브 클래스 또는 자식 클래스




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

상속

- 메서드 오버라이딩
  - 상위 클래스의 메서드를 서브 클래스에서 재정의



상속

```
class Car:
    speed = 0

    def upSpeed(self, value) :
        self.speed += value
        print("현재 속도(슈퍼 클래스) :%d"%self.speed)

class Sedan(Car):
    def upSpeed(self, value) :
        self.speed += value
        if self.speed > 150 :
            self.speed = 150
        print("현재 속도(서브 클래스) :%d"%self.speed)

sedan1 = Sedan()
sedan1.upSpeed(300)
```

Examples

- **예제 1 :**
  - 사람 (Human) 클래스를 "생성"
  - areum 이름의 변수로 인스턴스를 "생성"
  - 사람 (Human) 클래스에 (이름, 나이, 성별)을 받는 생성자를 추가

```
areum = Human("아름", 25, "여자")

print(areum.name)
print(areum.age)
print(areum.gender)
```

Examples

- **예제 2 :**
  - 사람 (Human) 클래스에 이름, 나이, 성별을 출력하는 who( ) 메소드를 추가
  - 사람 (Human) 클래스에 "나의 죽음을 알리지 말라"를 출력하는 소멸자를 추가

```
areum = Human("아름", 25, "여자")

areum.who()

del areum
```

Examples

- **예제 3 :**
  - 다음 코드가 동작하도록 차 클래스를 "생성"

```
car = 차(2, 1000)

car.바퀴
>> 2
car.가격
>> 1000
```

Examples

- **예제 4 :**
  - 다음 코드가 동작하도록 차 클래스를 상속받은 자동차 클래스를 " 생성“
  - 차 클래스를 상속받은 자동차 클래스를 "생성"

```
car = 자동차(4, 1000)

car.정보()
>> 바퀴 수 4
   가격 1000
```

감사합니다

kimtwan21@dongduk.ac.kr

김 태 완