3주차 클래스와 객체

01 클래스란?

[1] 객체지향 프로그램

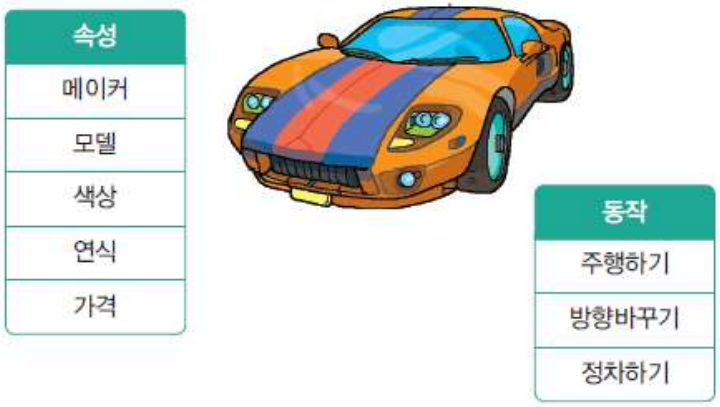
- 객체지향 프로그램(Objected Oriented Program)
  - 모든 개념을 **객체**(object, 사물)로 보고 객체를 중심으로 프로그램을 진행하는 기법.
  - 여기서 클래스는 객체의 '표준 틀' 또는 '설계도'라고 볼 수 있다.

- 파이썬은 거의 모든 것들이 객체지향
  - 리스트도 list라는 클래스의 객체
    - [1,2,3].append(5)

[1] 객체

- 객체(object)는 속성과 동작을 가진다.

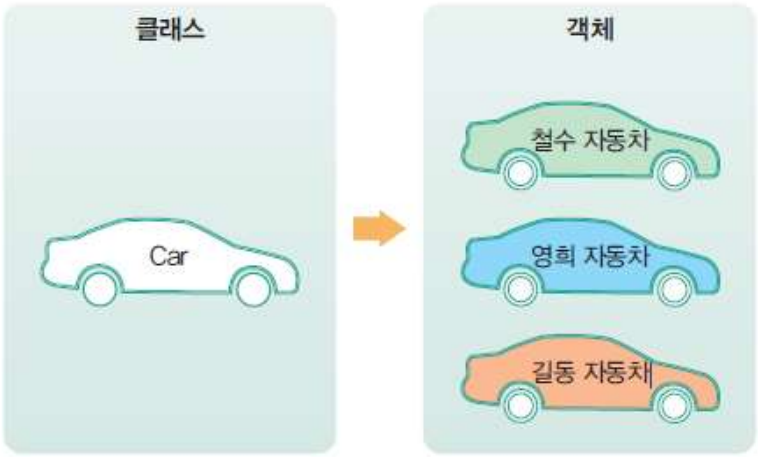
- 자동차는 메이커나 모델, 색상, 연식, 가격 같은 속성(attribute)을 가지고 있다. 또 자동차는 주행할수 있고, 방향을 전환하거나 정지할 수 있다. 이러한 것을 객체의 동작(action)이라고 한다.



[2] 클래스

- 객체에 대한 설계도를 클래스(class)라고 한다. 클래스란 특정한 종류의 객체들을 찍어내는 형틀(template) 또는 청사진(blueprint)이라고도 할 수 있다.

- 클래스로부터 만들어지는 객체를 그 클래스의 인스턴스(instance)라고 한다.



클래스 작성하기

Syntax

클래스 정의

형식

```
class 클래스이름 :
    def __init__(self, ...) :
        ...
    def 메소드1(self, ...) :
        ...
    def 메소드2(self, ...) :
        ...
```

예

```
class Counter:
    def __init__(self):
        self.count = 0
    def increment(self):
        self.count += 1
```

생성자를 정의한다.

메소드를 정의한다.

02 클래스 만들기

[1] 클래스의 구성

- 클래스 구성
  - 속성(attribute)
    - 객체가 가지는 여러 가지 데이터를 보관한다.
    - 멤버 변수라고도 부름
  - 메서드(method)
    - 객체와 관련된 동작을 하는 내부의 함수이다.
    - 멤버 함수라고도 부름

[2] 클래스의 정의

정의 방법

- 클래스 헤더
  - class와 클래스_이름 :
- 클래스 몸체
  - 들여 쓰기로 구별
  - 속성, 생성자, 메서드 등을 정의

생성자란?

- 특별한 메서드로 클래스로 객체를 생성할 때 한번 실행됨.
- 초기화가 필요 없으면 생략 가능.

[3] 객체 생성 방법

객체 생성 방법

생성자가 없는 경우

- 객체 = 클래스이름()

```
객체이름 = 클래스이름()
```

생성자가 있는 경우

- 객체 = 클래스_이름(a1,a2, ...)
- a1, a2,... : 생성자 함수의 인수들

```
객체이름 = 클래스이름(a1, a2, ...)
```

[4] 속성의 정의

- 클래스 속성의 정의
  - 일반 변수처럼 사용
  - 사용 예
    - `speed=100`
- 객체 속성 정의
  - **`self.`**로 시작하고 속성이름으로 정의
  - 사용 예

```
self.속성이름
```

```
self.speed = 200
self.area = self.width*self.height
```

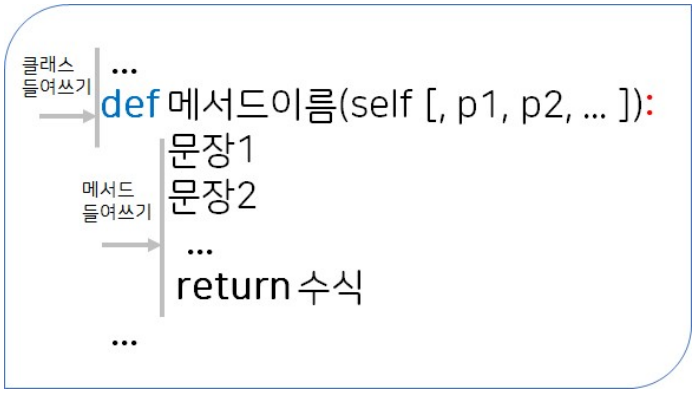
  - self의 의미
    - "객체의" 또는 "객체에 속해 있는" 으로 이해

[5] 메서드의 정의

메서드 정의 방법

해당 클래스 내부에서

- 함수처럼 **def**로 시작하고
- 매개변수, return 생략 가능
- 헤더의 마지막은 『:』으로 끝냄
- 몸체는 들여쓰기(indent)로 구별



주의

- 메서드의 첫번째 매개변수로 self가 있어야 한다.
- 객체 자신에 대한 정보를 받는 것으로 이해

[6] 클래스 만들기

간단한 Dog 클래스 만들기

- 클래스 속성: name
- 메서드: 없음

```
01 class Dog:           # 클래스 정의
02     name='Rocky'     # 클래스 속성
03 dog1=Dog()           # 객체 생성
04 print(Dog.name)      # 클래스 속성 출력
05 print(dog1.name)     # 객체 속성 출력
06 Dog.name='Cody'      # 클래스 속성 변경
07 print(dog1.name)     # 객체 속성 출력
08 dog1.name='Pepe'     # 객체 속성 변경
09 print(Dog.name)      # 클래스 속성 출력
10 print(dog1.name)     # 객체 속성 출력
```

```
Rocky
Rocky
Cody
Cody
Pepe
```

[6] 클래스 만들기

Dog 클래스에 메서드 추가

- 클래스 속성: name
- 메서드: show

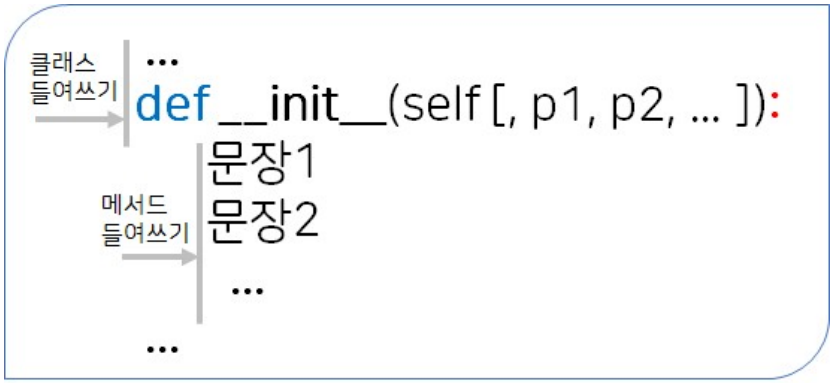
```
01 class Dog:                    # 클래스 정의
02     name='Rocky'              # 클래스 속성
03     def show(self):           # 메서드
04         print(self.name)
05 puppy=Dog()                   # 객체 생성
06 puppy.name='Toby'             # 속성 변경
07 print(puppy.name)             # 속성 출력
08 puppy.show()                  # 메서드 실행
```

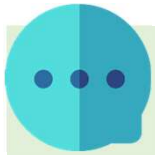
```
Toby
Toby
```

[7] 생성자 메서드

생성자 메서드

- 객체를 생성하는 시점에 한번 실행되는 메서드
  - 메서드 이름: `__init__`
  - 밑줄 문자가 앞뒤 모두 두 개씩
  - 주로 매개변수로 속성의 초기값 지정
  - 생성과 더불어 초기화 작업을 처리
  - 일반 메서드와 형식이 같음.
- 소멸자 함수
  - 객체가 소멸될 때 한번 실행됨
  - `__del__()`



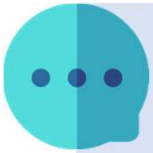


4개의 서로 다른 변수

- 이름이 같지만 모두 다른 이름 공간을 갖는 변수들
  - 클래스 속성(변수), 매개 변수, 객체 속성(변수), 전역 변수

```
01 class Dog:
02     name='happy'                    # 클래스 속성 name
03     def __init__(self, name):       # 매개변수  name
04         self.name=name              # 객체 속성 self.name
05     def show(self):
06         print(self.name)            # 객체 속성 출력
07 name='badugi'                       # 전역 변수 name
08 a=Dog('angry')                      # 객체 생성
09 s=Dog('sad')
10 a.show()
11 s.show()
12 print(Dog.name)                     # 클래스 속성 출력
13 print(name)                         # 전역 변수 출력
```

```
angry
sad
happy
badugi
```



데이터의 은닉

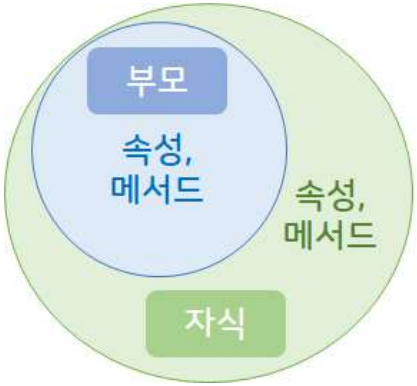
- 클래스를 사용하는 장점 중 하나는 데이터의 은닉이다.
  - 외부의 접근으로부터 보호하고자 하는 속성과 메서드는 밑줄 2개로 시작

```
01 class People:
02     def __init__(self, name):
03         self.__weight=88
04         self.__height=1.50
05         self.name=name
06     def __calc(self):
07         return self.__weight/self.__height**2
08     def BMI(self):
09         return self.__calc()
10 me=People('Mario')
11 print(me.BMI())      # Ok!
12 print(me.__weight)   # Error, it's private attribute
13 print(me.__calc())   # Error, it's private method
```

03 클래스의 상속

[1] 클래스의 상속(inheritance)

- 상속 필요성:
  - 객체지향 프로그램의 강력한 기능으로서 **생산성**을 높일 수 있다.
  - 이미 만들어진 클래스가 있다면 상속을 통하여 속성과 메서드를 자식클래스에서 **재활용**할 수 있다.
  - 상속하는 클래스는 **부모(parent)** 클래스, 받는 클래스는 **자식(child)** 클래스라 한다.
  - 부모 클래스에 있는 자산은 자식에서 다시 정의할 필요가 없고 추가의 속성, 메서드만 지정한다.



[2] 클래스의 상속 방법

- 자식 클래스의 정의
  - 클래스 이름 다음 괄호안에 부모 클래스 이름을 표시한다.

```
class 자식클래스이름(부모클래스이름):
    추가 속성 정의
    추가 메서드 정의
    . . .
```

(들여쓰기)

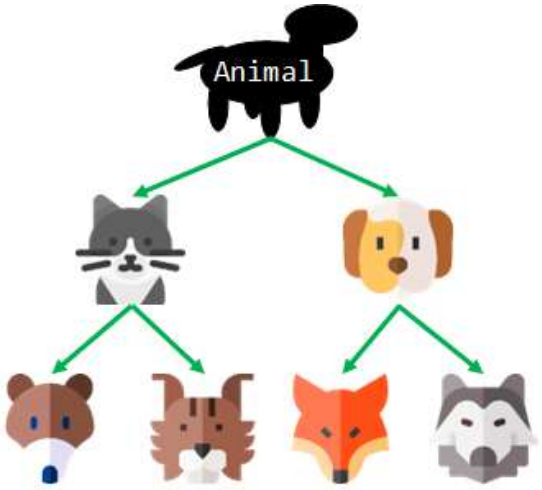
- 자식 클래스 내부에서 부모 클래스 메서드 호출
  - 2가지 방법.

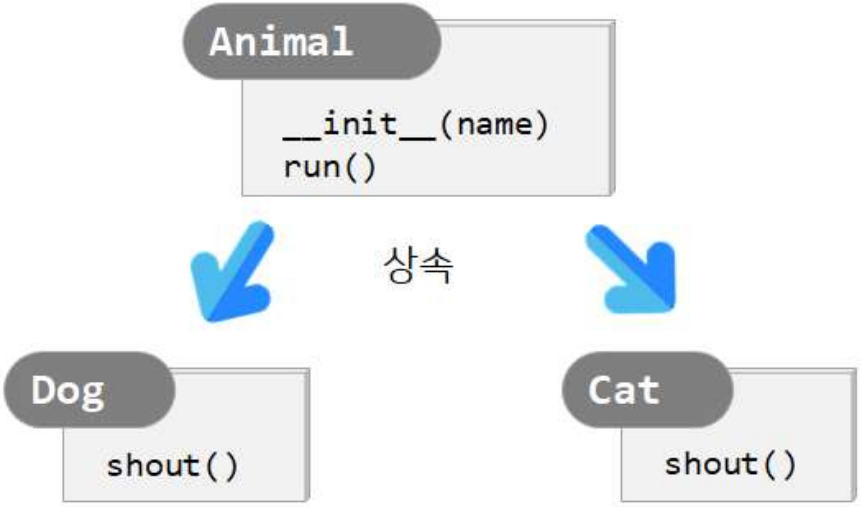
```
부모클래스이름.메서드이름(self,a1,...)
```

```
super(자식클래스이름,self).메서드이름(a1,...)
```

[3] 상속의 예

추상적인 Animal 클래스를 먼저 만들고 이를 상속시켜 구체화된 Dog와 Cat 클래스를 생성





[3] 상속의 예

- 코드

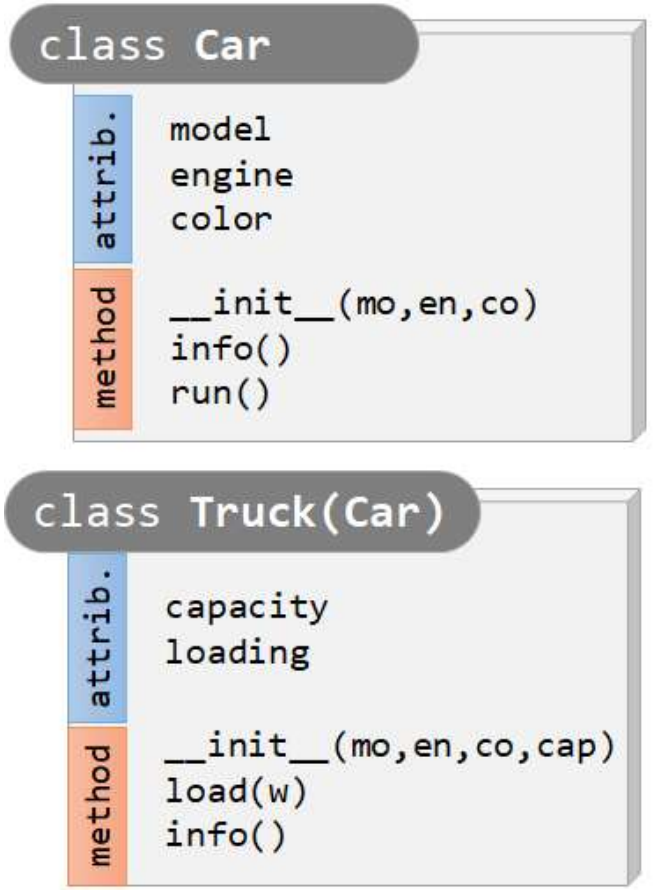
```
01 class Animal:
02     def __init__(self, name):
03         self.name=name
04     def run(self):
05         print(f'{self.name} 달려')
06 class Dog(Animal):
07     def shout(self):
08         print(f'{self.name} 멍멍~')
09 class Cat(Animal):
10     def shout(self):
11         print(f'{self.name} 이야옹~')
12 d=Dog('마음이')
13 c=Cat('냥냥이')
14 d.run()
15 c.shout()
16 d.shout()
```

```
마음이 달려
냥냥이 이야옹~
마음이 멍멍~
```

[3] 상속의 예

Truck 클래스 만들기

- 이전에 정의한 Car에서 상속받아 정의
- Car의 속성과 메서드 사용가능
- 추가 속성
  - capacity
  - loading
- 추가 메서드
  - __init__(mo,en,co,cap)
  - load(w)
  - info() : 같은 이름의 중복 정의 가능



[3] 상속의 예

- Car 클래스 코드

```
01 class Car:
02     def __init__(self,model,engine,color='black'):
03         self.model=model
04         self.engine=engine
05         self.color=color
06     def info(self):
07         print(f'Model: {self.model} / {self.engine}cc')
08         print(f'Color {self.color}')
09     def run(self):
10         print(f'{self.color} {self.model} 붕붕~~~')
```

[3] 상속의 예

- Truck 클래스 코드

```
01 class Truck(Car):
02     def __init__(self,model,engine,color='blue',cap=5000):
03         #super(Truck,self).__init__(model,engine,color)
04         Car.__init__(self,model,engine,color)
05         self.capacity=cap
06         self.loading=0
07     def load(self, w):
08         self.loading+=w
09     def info(self):
10         #super(Truck,self).info()
11         Car.info(self)
12         print(f'Capacity :{self.capacity/1000:.1f} ton')
13         print(f'Current loading: {self.loading} kg')
```

[3] 상속의 예

- 객체 생성 코드

```
14 tr=Truck('Mighty',3900,'white',2500)
15 tr.load(700)
16 tr.info()
17 tr.run()
```

```
Model: Mighty / 3900cc
Color white
Capacity :2.5 ton
Current loading: 700 kg
white Mighty 붕붕~~~
```