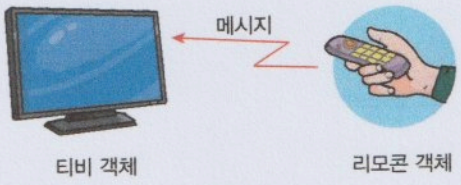
02 객체 지향 프로그래밍

객체 지향 프로그래밍(OOP: object-oriented programming)은 우리가 살고 있는 실제 세계가 객체(object)들로 구성되어 있는 것과 비슷하게, 소프트웨어도 객체로 구성하는 방법이다. 실제 세계에는 사람, 텔레비전, 세탁기, 냉장고 등의 많은 객체가 존재한다. 객체들은 객체 나름대로의 고유한 기능을 수행하면서 다른 객체들과 상호 작용한다.

예를 들면, 사람이 리모컨을 이용하여서 텔레비전을 조작하는 상황을 생각해보자. 텔레비전 리모콘은 모두 특정한 기능을 수행하는 객체라고 생각할 수 있고 텔레비전과 리모콘은 메시지를 통하여 서로 상호 작용하고 있다.

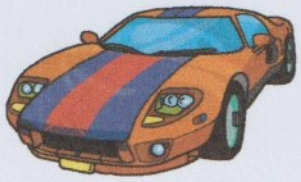


소프트웨어 개발도 이와 같이 하는 방식을 객체 지향이라고 한다. 다양한 기능을 하는 소프트웨어 객체들이 존재하고 이러한 객체들을 조합하여 자기가 원하는 기능을 구현하는 기법이다. 위의 그림에서 텔레비전과 리모콘은 소프트웨어 객체로 표현되며 이들 소프트웨어 객체들이 메시지를 전달하여 서로 상호 작용하면서 원하는 작업을 수행하게 된다. 프로그램에서는 현실 세계에서 볼 수 있는 물리적인 객체도 사용하지만 소프트웨어 세계에서만 존재하는 객체도 사용한다. 예를 들면 화면의 원도우나 버튼도 하나의 객체로 취급된다.

03 객체란?

객체(object)란 무엇인가? 객체는 하나의 물건이라고 생각하면 된다. 우리가 일상생활에서 사용하는 물건이 객체가 될 수 있다. 예를 들어서 자동차를 생각해보자. 자동차는 메이커나 모델, 색상, 연식, 가격 같은 속성(attribute)을 가지고 있다. 또 자동차는 주행할 수 있고, 방향을 전환하거나 주차할 수 있다. 이러한 것을 객체의 동작(action)이라고 한다.

속성
메이커
모델
색상
연식
가격



동작
주행하기
방향바꾸기
주차하기

파이썬에서 객체의 속성은 객체 안의 변수에 저장된다. 객체의 이름 다음에 점을 찍고 변수의 이름을 적는다.

```
object.speed
object.color
object.make
```

파이썬에서는 객체의 동작을 메소드라고 부른다. 메소드는 객체 안에 정의된 함수이다. 객체의 메소드는 다음과 같이 객체 이름 다음에 점을 찍어서 호출할 수 있다.

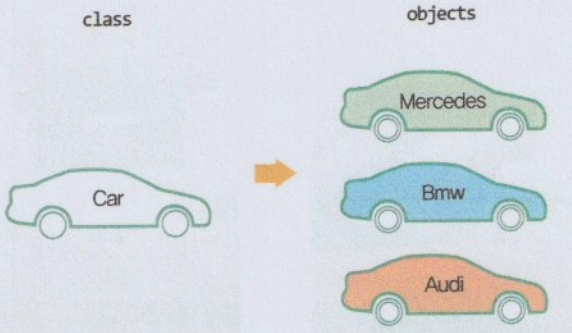
```
object.drive()
object.park()
```

객체는 다음과 같은 수식으로 정의된다.

객체 = 변수 + 메소드

05 객체 생성하기 #1

파이썬에서 우리가 직접 객체를 정의하고 생성하려면 다음과 같은 2가지의 단계를 거쳐야 한다.



(1) 객체의 설계도를 작성하여야 한다. 즉 객체가 가지고 있는 속성을 변수로 표현하고, 객체의 동작은 메소드로 정의한다. 이것은 집을 짓기 전에 설계도를 만드는 것과 유사하다. 이 설계도를 클래스라고 한다. 클래스는 객체가 아니다. 주의하여야 한다. 자동차 클래스를 작성해보자.

```
class Car:
    def drive(self):
        self.speed = 10
```

아주 간단히 만들어 보았다. Car 클래스는 하나의 속성과 하나의 메소드만을 가지고 있다. self.speed는 Car 클래스의 속성이고 self.drive()는 Car 클래스의 메소드이다. 메소드의 첫 번째 매개변수는 항상 self로서 현재 객체를 가리킨다.

(2) 클래스로부터 객체를 생성하여야 한다. 다음과 같은 문장으로 객체가 생성된다.

```
myCar = Car()
```

myCar가 생성된 객체의 이름이 된다. myCar의 속성이 필요하면 얼마든지 추가하여도 된다. 예를 들어서 자동차의 색상, 모델을 나타내는 속성을 다음과 같이 객체에 추가할 수 있다.

```
myCar.color = "blue"
myCar.model = "E-Class"
```

(3) 자동차 객체의 속성과 메소드를 사용할 수 있다.

```
myCar.drive()              # 객체 안의 drive() 메소드가 호출된다.
print(myCar.speed)         # 10이 출력된다.
```

06 객체 생성하기 #2

앞의 코드를 하나로 모으면 다음과 같다.

```
class Car:
    def drive(self):
        self.speed = 60

myCar = Car()

myCar.speed = 0
myCar.model = "E-Class"

myCar.color = "blue"
myCar.year = "2017"

print("자동차 객체를 생성하였습니다.")
print("자동차의 속도는", myCar.speed)
print("자동차의 색상은", myCar.color)

print("자동차의 모델은", myCar.model)
print("자동차를 주행합니다.")
myCar.drive()
print("자동차의 속도는", myCar.speed)
```

클래스를 정의한다.

객체를 생성한다.

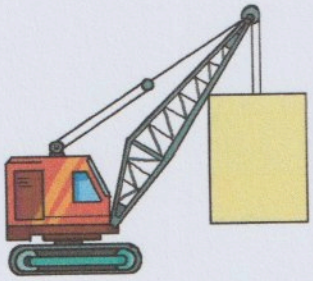
객체의 속성을 설정한다.

객체의 속성을 출력한다.

객체의 메소드를 호출한다.

```
자동차 객체를 생성하였습니다.
자동차의 속도는 0
자동차의 색상은 blue
자동차의 모델은 E-Class
자동차를 주행합니다.
자동차의 속도는 60
```

07 객체를 생성하면서 초기화하기



우리는 앞에서 객체를 생성할 때, speed, color, model에 아무것도 채우지 않았다. 우리는 객체를 생성한 후에 이들 변수를 초기화하였다. 하지만 객체가 생성될 때 객체를 초기화할 수 있는 방법이 있다.

객체를 초기화하려면 클래스 안에 특별한 메소드인 __init__()을 정의해주면 된다. __init__()을 생성자라고 부른다. 메소드 이름에 _가 2번 사용되었다. __init__()는 외부에서 전달되는 초기값들을 받을 수 있다. Car 클래스에 __init__() 메소드를 추가하여 보자.

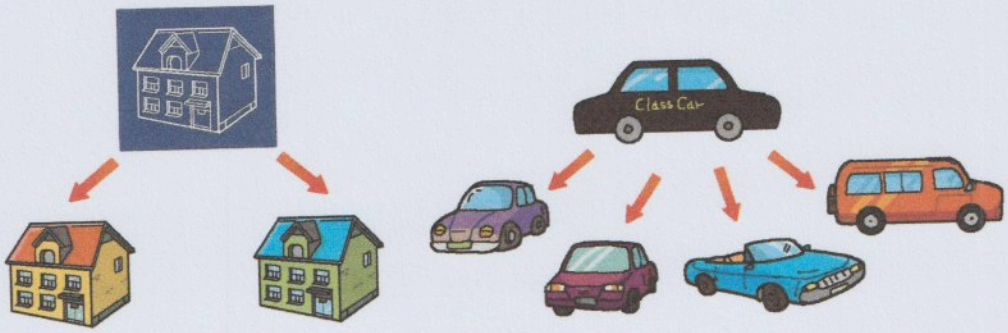
생성자

```
class Car:
    def __init__(self, speed, color, model):
        self.speed = speed
        self.color = color
        self.model = model

    def drive(self):
        self.speed = 60

myCar = Car(0, "blue", "E-class")

print("자동차 객체를 생성하였습니다.")
print("자동차의 속도는", myCar.speed)
print("자동차의 색상은", myCar.color)
print("자동차의 모델은", myCar.model)
print("자동차를 주행합니다.")
myCar.drive()
print("자동차의 속도는", myCar.speed)
```

객체를 초기화하는 특별한 메소드이다. 속도, 색상, 모델을 받아서 객체 안의 변수에 저장한다.

```
자동차 객체를 생성하였습니다.
자동차의 속도는 0
자동차의 색상은 blue
자동차의 모델은 E-class
자동차를 주행합니다.
자동차의 속도는 60
```

08 하나의 클래스로 객체는 많이 만들 수 있다



우리는 하나의 클래스로 여러 개의 객체를 생성할 수 있다. 하나의 설계도만 있으면 동일한 형태의 집을 여러 채 지을 수 있는 것과 마찬가지이다. 하나의 자동차 설계도에서 수많은 자동차를 생산할 수 있는 것과 마찬가지이다.

```
class Car:
    def __init__(self, speed, color, model):
        self.speed = speed
        self.color = color
        self.model = model

    def drive(self):

        self.speed = 60

dadCar = Car(0, "silver", "A6")
momCar = Car(0, "white", "520d")
myCar = Car(0, "blue", "E-class")
```

동일한 클래스로 여러 개의 객체를 만들 수 있다.

여기서 반드시 기억할 사항은 동일한 클래스로 객체를 여러 개 생성하더라도 객체가 가지고 있는 변수의 값은 다를 수 있다는 점이다. 객체가 가진 데이터는 객체마다 다르게 설정할 수 있다. 위의 예에서도 각 자동차 객체마다 색상과 모델은 다르게 초기화되었다.

10 self는 무엇인가?

파이썬의 클래스 정의에서 self는 상당히 많이 등장한다. 도대체 self는 무엇인가?

```
class Car:
    def __init__(self, speed, color, model):
        self.speed = speed
        self.color = color
        self.model = model

    def drive(self):
        self.speed = 60c

myCar = Car(0, "blue", "E-class")
yourCar = Car(0, "white", "S-class")
```

메소드가 호출될 때, 어떤 객체가 메소드를 호출했는지 알아야 한다. 즉 myCar 객체가 drive()를 호출했는지 yourCar 객체가 drive()를 호출하였는지 drive() 메소드가 알아야 한다는 것이다. drive() 메소드의 self 매개 변수는 어떤 객체가 메소드를 했는지를 알려준다.

하지만 우리가 drive()를 호출할 때는 객체를 인수로 넣어서 보내지는 않는다.

```
myCar.drive()
yourCar.drive()
```

우리가 drive()를 호출할 때는 항상 "객체이름.메소드()"와 같은 형식을 사용한다. 객체이름이 바로 self로 전달되는 것이다.