클래스

클래스 : 설계 도면, 객체 : 설계도로 만든 것

클래스로 객체를 생성할 수 있고, 생성된 각 객체는 독립적이다.

클래스 정의

클래스는 class 예약어를 사용해서 만든다.

```
class 클래스이름:
    클래스 내용
```

ex)
```
class simple:
    pass
```

객체 생성

↳ 정의된 클래스로 객체 (인스턴스)를 생성 가능하다.

ex) a = simple()

↳ simple class로 a객체 생성

메소드 생성

메소드 : 클래스 내에서 정의된 함수
메소드는 함수와 똑같이 정의하면 된다.

ex)
```
class fourcal:
    def setdata(self, first, second):
        self.first = first
        self.second = second
```

self 매개변수

↳ 인스턴스.메소드를 실행할 때, 항상 매개변수 self가 인수로 넘어간다.

ex)
```
a = fourcal()
a.setdata(4,2)      # 사실은 a.setdata(a,4,2) 이다.
```

생성자

생성자는 객체를 생성할 때, 자동으로 생성되는 method이다.
↳ 이름은 __init__이다.

ex) class fourcal:
    def __init__(self, first, second):
        self.first = first
        self.second = second

a = fourcal(4,2)
↳ 생성자의 매개변수 때문에, 이제 객체 생성시 매개변수 필요

클래스의 상속

어떤 클래스를 생성할 때, 다른 클래스의 메서드를 고대로 받아온다.

언제 사용하나? — 부모 메서드 + 메서드를 추가할 때
구조:
    class 자식 클래스 (부모클래스):
        추가할 메서드

ex) class more (fourcal):
        def pow (self):
            return self.first ** self.second

    a = more(4,2)
    a.pow() ≠ 16 리턴

상속은 메서드의 추가
그럼 메서드의 변경은 뭘로해?
⇒ 메서드 오버라이딩

메서드 오버라이딩
↳ 기본 구조는 상속과 같다.
But! 메서드를 변경할 때는 **똑같은 이름으로** 메서드를 재정의

ex)
```
class safe (fourcal):
    def div (self):          #원래 div가 fourcal에 있었다 가정하에
        if (self.second == 0):
            return 0
        else:
            return self.first / self.second
```

```
a = safe(4,0)
a.div()                      # 0 리턴
```

클래스 변수

클래스 내에서 성성된 변수

⚠ 객체 변수는 서로 독립적이다.

ex)
```
class family:
    lastname = "김"
```

```
a = family()
b = family()
b.lastname = "박"
# a.lastname은 여전히 김 , b.lastname만 박으로 변경
```