클래스

❖ 클래스

- 프로그래머가 지정한 이름으로 만든 하나의 독립된 공간(이름 공간)
- 구성요소
  - 클래스 멤버 : 변수
  - 클래스 메소드 : 함수와 동일한 역할

```
class 클래스이름:
    클래스 멤버 정의
    클래스 메소드 정의
```

클래스

❖ 클래스 정의

```
class MyClass:
    var = '안녕하세요'
    def sayHello(self):
        print(self.var)

obj = MyClass()
print(obj.var)
obj.sayHello()
```

- 인스턴스 멤버 접근
  - obj.클래스 멤버
  - obj.클래스 메소드

❖ 클래스 멤버와 인스턴스 멤버

- 클래스 멤버 : 클래스 메소드 바깥에서 선언
- 인스턴스 멤버 : 클래스 메소드 안에서 self와 함께 선언되는 변수

```
class MyClass:
    var = '안녕하세요'
    def sayHello(self):
        param1 = '안녕'
        self.param2 = '하이'
        print(param1)
        print(self.var)

obj = MyClass()
print(obj.var)
obj.sayHello()
# obj.param1  → 지역변수 접근이므로 에러
```

❖ 클래스 메소드

- 클래스 메소드의 첫번째 인자는 반드시 self
  - self : 이 클래스의 인스턴스 객체 참조 (호출문에서 생략)
  - 두 번째 인자가 메소드의 인자

```
class MyClass:

    def sayHello(self):
        print('안녕하세요')

    def sayBye(self, name):
        print('%s! 다음에 보자!'%name)

obj = MyClass()

obj.sayHello()
obj.sayBye('철수')
```

```
안녕하세요
철수! 다음에 보자!
```

❖ 클래스 생성자

ㅇ 클래스의 인스턴스 객체가 생성될 때 자동으로 호출되는 메소드

```
def __init__(sel, *args)
```

```
class MyClass:
    def __init__(self):
        self.var = '안녕하세요!'
        print('MyClass 인스턴스 객체가 생성되었습니다');

obj = MyClass()
print(obj.var)
```

```
MyClass 인스턴스 객체가 생성되었습니다
안녕하세요!
```

❖ 클래스 생성자

- ㅇ 생성자 인자 전달

```
def __init__(sel, *args)
```

```
class MyClass:
    def __init__(self, txt):
        self.var = txt
        print('생성자 인자로 전달받은 값은 <' + self.var + '>입니다')

# obj = MyClass()
obj = MyClass('철수')
print(obj.var)
```

```
생성자 인자로 전달받은 값은 <철수>입니다
철수
```

❖ 클래스 소멸자

- 클래스의 인스턴스 객체가 소멸될 때 자동으로 호출되는 메소드

```
def __del__(sel)
```

- 인스턴스 강제 제거

```
del <인스턴스 객체>
```

```
class MyClass:
    def __del__(self):
        print('MyClass 인스턴스 객체가 메모리에서 제거됩니다.')

obj = MyClass()
del obj
```

```
MyClass 인스턴스 객체가 메모리에서 제거됩니다.
```

❖ 클래스 상속

- 어떤 클래스(부모 클래스, 슈퍼 클래스)가 가지고 있는 모든 멤버나 메소드를 상속받는 클래스(자식 클래스)가 모두 사용할 수 있도록 해주는 것

```
calss 자식 클래스(부모클래스)
```

- 오버라이드
  - 부모 클래스에서 정의한 메소드를 자식 클래스가 동일한 모양으로 재정의하는 것

- 다중 상속 지원

```
calss 자식 클래스(부모클래스1, 부모클래스2, …)
```

❖ 클래스 상속

```
class Add:
    def add(self, n1, n2):
        return n1+n2

class Calculator(Add):
    def sub(self, n1, n2):
        return n1-n2

obj = Calculator()
print(obj.add(1,2))
print(obj.sub(1,2))
```

```
3
-1
```

❖ 클래스 다중 상속

```
class Add:
    def add(self, n1, n2):
        return n1+n2

class Multiply:
    def multiply(self, n1, n2):
        return n1*n2

class Calculator(Add, Multiply):
    def sub(self, n1, n2):
        return n1-n2

obj = Calculator()
print(obj.add(1,2))
print(obj.multiply(3,2))
```

```
3
6
```