ILE3-005 함수, 클래스

함수

특정 동작을 수행하는 코드의 특정 범위

- 복잡한 기능을 여러 부분으로 분할하여 흐름 판단이 쉬움
- 일부분만 수정하기 쉬움
- 반복 사용이 가능

함수의 사용

```
[리턴 값] = [함수이름]([매개변수])
```

```
three = add(1, 2)
six, two = add_sub(4, 2)
```

함수의 선언

```
def [함수이름]([매개변수]):
  ...
  return
```

```
def add(v1, v2):
    return v1+v2
```

```
def add_sub(v1, v2):
    return v1+v2, v1-v2
```

가변 매개변수 함수

매개 변수 입력 개수가 결정되지 않은 경우 사용

```
def [함수이름](*args):
    ....
    for v in args:
        ...
    ...

def [함수이름]([고정 매개변수], *args):
    ....
    for v in args:
        ...
    ...
```

```
def foo(*args):
    print(args)
foo(1, 2, 3, 4, 5)

def bar(x, y, *args):
    print(f'{x} {y}')
```

```
    print(args)
bar(1, 2, 3, 4, 5)
```

```
(1, 2, 3, 4, 5)
1, 2
(3, 4, 5)
```

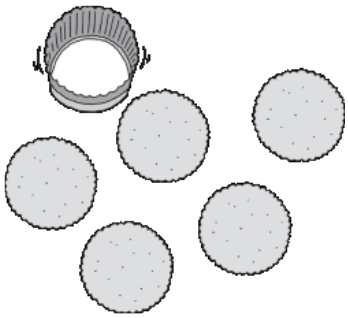
클래스

특정 물체를 추상화 하여 변수 및 함수로 구성된 일종의 Template

- 클래스: 유사한 특징을 지닌 객체들의 속성을 묶어 놓은 집합체
- 인스턴스 or 오브젝트: 클래스를 이용하여 생성된 무엇인가

클래스를 설명한 사진

틀 - 클래스
쿠키 - 인스턴스



Class의 생성

```
class [클래스 이름]:

    def __init__(self, [매개변수]):
        self.[변수명] = [변수값]
        초기화 구문

    def foo(self, [매개변수]):
        ...
        return [리턴값]

    def bar(self, [매개변수]):
        [변수] = self.[변수명]
        self.foo([변수])

[객체명] = [클래스 이름]([생성자 매개변수])
```

```
class Student:

    def __init__(self, name, grade):
        self.name = name
        self.grade = grade

    def getGrade(self):
        return self.grade
```

```
9
10      def getFullGrade(self, school_name):
11          return f'{school_name} {self.grade}'
12
13  tom = Student('톰', '1학년')
14  jerry = Student('제리', '3학년')
15
16  print(tom.getFullGrade('고등학교'))
17  print(jerry.grade)
```

클래스 함수와 멤버 함수

멤버 함수

- Class의 Instance를 이용하여 호출된 함수
- Instance 내의 변수들에 대한 접근 가능
- 함수의 첫번째 매개변수는 항상 자기자신(호출한 Instance)를 받음
- 관습적으로 첫번째 매개변수의 명칭은 `self`를 사용

```
1   class TestClass:
2
3       def __init__(self):
4           self.result = 0
5
6       def member_func_subtract(self, a):
7           self.result = self.result - a
8           return self.result
9
10  print(c.member_func_subtract(1))            # 정상 실행
11  print(TestClass.member_func_subtract(c, 1)) # 특이한 케이스지만 정상 실행
```

클래스 함수

- Class의 명칭을 이용하여 호출된 함수
- Instance가 명시되어 있지 않기 때문에 Instance 내의 변수는 접근 불가
- 종속이 없는 연산 등을 구현할 때 사용

```
1   class TestClass:
2
3       def __init__(self):
4           self.result = 0
5
6       def static_func_add(a, b): # 클래스 함수, self를 안가지고 있음
7           return a+b
8
9       def static_func_using_self(a): # 호출하면 오류 발생
10          self.result = self.result - a
11          return self.result
12
13  c = TestClass()
14  print(TestClass.static_func_add(1, 2))   # 정상 실행
15  print(c.static_func_add(1, 2))          # 오류
```

특별한 함수

__init__ 함수

Class의 Instance가 생성될 때 호출되는 함수
1개만 선언 가능

```
class TestClass:

    def __init__(self, init_val=0):
        self.result = init_val

d = TestClass()
e = TestClass(100)
print(d.result)
print(e.result)
```

```
0
100
```

__str__ 함수

Instance를 문자열로 Typecasting 시 호출되는 함수

```
class TestClass:

    def __init__(self, init_val=0):
        self.result = init_val

    def __str__(self):
        return f'{self.result}점'

d = TestClass()
e = TestClass(100)
print(d)
print(e)
```

```
0점
100점
```

클래스 변수와 멤버 변수

클래스 변수

- Class 범위에서 선언한 변수
- Class에서 공용으로 사용

```
class TestClassA:
    non_self = [] # Class 변수

    def __init__(self):
        pass

    def add_data(self, v):
        self.non_self.append(v)


i1 = TestClassA()
```

```
i1.add_data('a')

i2 = TestClassA()
i2.add_data('b')

print(i1.non_self)
print(i2.non_self)
```

```
['a', 'b']
['a', 'b']
```

멤버 변수

- `self.`을 붙이고 선언한 변수
- Instance에서만 사용

```
class TestClassB:

    def __init__(self):
        self.it_self = [] # 멤버 변수
        pass

    def add_data(self, v):
        self.it_self.append(v)


i1 = TestClassB()
i1.add_data('a')

i2 = TestClassB()
i2.add_data('b')

print(i1.it_self)
print(i2.it_self)
```

```
['a']
['b']
```

계산기를 분석해 보자

계산기가 저장해야 할 것

1. 마지막 계산 결과

> 변수가 됩니다

계산기에 달려있는 버튼

1. 숫자 (이건 뺍시다)
2. 사칙연산 (+-*/)
3. 결과 (=)
4. 초기화 (C/AC)

> 함수가 됩니다.

계산기를 만들어 보자

```
class Calculator:

    def __init__(self):
        self.result = 0.0

    def add(self, v):
        self.result = self.result + v

    def subtract(self, v):
        self.result = self.result - v

    def multiply(self, v):
        self.result = self.result * v

    def division(self, v):
        self.result = self.result / v

    def change_sign(self):
        self.multiply(-1)

calc = Calculator()
calc.add(10)
calc.subtract(2)
calc.multiply(7)
calc.division(2)
calc.change_sign()
print(calc.result)
```

과제

사각형 범위를 표현하는 Region Class를 만들어 주세요

필요한 기능은 아래와 같습니다.

- 생성 시 Min X, Min Y, Max X, Max Y를 입력받음
- 아래의 함수 구현
  - getWidth
  - getHeight
  - getCenterPoint
  - getLeftBottomPoint
  - getRightTopPoint
  - ptInRect

> Point를 반환하는 함수는 Tuple로 출력

Class를 구현하시고 아래의 Test Code를 통해 확인하시면 됩니다.

```
r1 = Region(0, 0, 100, 100)
print(r1.getWidth())
print(r1.getHeight())
print(r1.getCenterPoint())
print(r1.getLeftBottomPoint())
print(r1.getRightTopPoint())
print(r1.ptInRect(1, 1))
```

```
print(r1.ptInRect(200, 1))

r2 = Region(100, 0, 0, 100)
print(r2.getWidth())
print(r2.getHeight())
print(r2.getCenterPoint())
print(r2.getLeftBottomPoint())
print(r2.getRightTopPoint())
print(r2.ptInRect(1, 1))
print(r2.ptInRect(200, 1))
```