

클래스 알아보기

학습목표

- 클래스의 개념을 이해해본다.
- 클래스의 오버라이딩 개념을 이해해 본다.
- 클래스의 상속 개념을 이해해 본다.

목차

- 1-4-1 객체와 클래스 알아보기
- 1-4-2 클래스 작성하기
- 1-4-3 계산기 클래스 만들어보기
- 1-4-4 그렇다면 처음 객체가 생성될때, 값을 초기화시킬 수 있을까?
- 1-4-5 메서드를 덮어쓰기 (메서드 오버라이딩)

1-4-1 객체와 클래스 알아보기

- 객체는 상태와 동작을 가지고 있다.
- 객체의 상태(state)는 객체의 속성.
- 텔레비전의 객체의 경우 상태는 채널번호, 볼륨, 전원상태 등이다.
- 객체의 동작(behavior)은 객체가 취할 수 있는 동작(기능)
 - 텔레비전의 경우, 켜기, 끄기, 채널 변경, 볼륨 변경

객체의 변수

- 객체안의 변수를 **인스턴스 변수(instance variable)**이라 함.
- 객체의 동작을 나타내는 부분을 **메소드(method)**라 함.
- 즉, 객체는 인스턴스 변수와 메소드로 이루어진 소프트웨어 묶음

클래스

- 객체에 대한 설계도.
- 특정한 종류를 찍어내는 형틀(template)

인스턴스

- 클래스로부터 만들어부터 만들어지는 각각의 객체

1-4-2 클래스 작성하기

In [1]:



```
class Cal:
    pass
```

인스턴스 만들기

In [2]:



```
a = Cal()
b = Cal()
```

왜 클래스가 필요할까?

함수를 이용하여 계산기 하나 만들기

- 추가해서 값을 더해 준다.

In [3]:



```
result = 0
def plus(num):
    global result
    result += num
    return result

print(plus(3))
print(plus(4))
```

3
7

함수를 이용한 계산기 두 개 만들기

- 추가해서 값을 더해 준다.

In [4]:



```
# 첫번째 계산기
result1 = 0
def plus1(num):
    global result1
    result1 += num
    return result1

print(plus1(3))
print(plus1(4))

# 두번째 계산기
result2 = 0
def plus2(num):
    global result2
    result2 += num
    return result2

print(plus2(3))
print(plus2(4))
```

3
7
3
7

그렇다면 계산기 10대를 만들게 되면..

- [아이디어] 계산기의 형태의 하나의 틀을 만들고, 이를 찍어내자.
 - 다만, 일부 형틀 이외의 일부 기능들은 조금씩 변경이 가능하다.

1-4-3 계산기 클래스 만들어보기

- 클래스의 이름의 첫글자는 일반적으로 대문자로 함.
- 5대의 계산기를 만들기
- 클래스 안의 메소드는 매개 변수가 추가(self)되었다. 이는 전달되는 객체를 의미

In [5]:



```
class Cal:
    result = 0

    def plus(self, num):
        self.result += num
        return self.result
```

In [6]:



```
a = Cal() # 계산기 한대
print(a.plus(10))
print(a.plus(20))
```

10
30

In [7]:



```
# 나머지 계산기 네대
a2 = Cal() # 계산기 한대
a3 = Cal() # 계산기 한대
a4 = Cal() # 계산기 한대
a5 = Cal() # 계산기 한대
```

In [8]:



```
# 네대의 계산기는 각각 동작
print( a5.plus(100) )
print( a5.plus(100) )
```

100
200

(실습해 보기)

- 더하기, 빼기 기능 추가해 보기

In [9]:



```
class CalFnc2:
    result = 0

    def plus(self, num):
        self.result += num
        return self.result

    def sub(self, num):
        self.result -= num
        return self.result
```

In [10]:



```
a = CalFnc2()

# 3을 더하고 5를 빼기
print( a.plus(3) )
print( a.sub(5) )
```

3
-2

1-4-4 그렇다면 처음 객체가 생성될때, 해당 계산기의 값을 초기화시킬 수 없을까?

- 메서드의 이름을 **init**로 하면 생성자로 인식되어 객체 생성 시점에 자동 호출.

```
def __init__(self, 값1, 값2):  
    self.result = result
```

In [11]:



```
class CalFnc2:  
    def __init__(self, result):  
        self.result = result  
  
    def plus(self, num):  
        self.result += num  
        return self.result  
  
    def sub(self, num):  
        self.result -= num  
        return self.result
```

In [12]:



```
a = CalFnc2(0) # 계산기 한대  
  
## 첫 초기값 (result)  
print(a.result)
```

0

In [14]:



```
### 계산기 생성시, 초기값을 100으로 해서 5를 더하기  
b = CalFnc2(100) # 계산기 한대  
  
## 계산기의 초기값  
print("현재 계산기 결과 :", b.result)  
  
print( b.plus(5) )  
  
## 계산기의 결과  
print("현재 계산기 결과 :", b.result)
```

현재 계산기 결과 : 100
105
현재 계산기 결과 : 105

1-4-5 메서드를 덮어쓰기 (메서드 오버라이딩)

- 기존의 클래스의 상속받아, 새롭게 메서드를 정의해 본다.

In [15]:



```
class CalFnc3:
    def __init__(self, result):
        self.result = result

    def plus(self, num):
        self.result += num
        return self.result

    def sub(self, num):
        self.result -= num
        return self.result

    def div(self, num):
        self.result /= num
        return self.result
```

- 초기값을 0으로 정의한 후, 5를 더하고 3으로 나눠보자

In [16]:



```
a = CalFnc3(0) # 계산기 한대 (초기값 0)

## 첫 초기값 (result)
tmp = a.plus(5) # 5더하기
tmp1 = a.div(3)
print(tmp1)
```

1.6666666666666667

위의 내용을 0으로 나눌때,

- 0으로 나누는 것은 안되는데, 현재 클래스를 변경할 수 없다.

클래스 상속

```
class A:
    pass

class B(A):
    # A를 상속받아, B를 만든다.
    pass
```

In [17]:



```
class CalFnc3:
    def __init__(self, result):
        self.result = result

    def plus(self, num):
        self.result += num
        return self.result

    def sub(self, num):
        self.result -= num
        return self.result

    def div(self, num):
        self.result /= num
        return self.result
```

In [18]:



```
class CalFnc3_change(CalFnc3): # CalFnc3를 상속받음
    def __init__(self, result):
        self.result = result

    def plus(self, num):
        self.result += num
        return self.result

    def sub(self, num):
        self.result -= num
        return self.result

    def div(self, num):
        if num == 0:
            return 0
        else:
            self.result /= num
        return self.result
```

In [20]:



```
a = CalFnc3_change(0) # 계산기 한대 (초기값 0)

## 첫 초기값 (result)
tmp = a.plus(5) # 5더하기
tmp1 = a.div(0)
print(tmp1)
```

0

[실습해보기] tv class를 생성해 보자.

- tv 채널 정보 : 기본 0번에서 시작.
- tv 채널 변경 : change_channel() 메서드 필요.
- tv volume 변경 기능.

[도전] 나만의 자동차 class를 만들어보자.

기본에서 초중급으로 업그레이드 파트

클래스 다중 상속

```
class A:
    pass

class B:
    pass

class C(A, B):      # A와 B를 상속받아, C를 만든다.
    pass
```

In [1]:



```
class Person:
    def greeting(self):
        print('안녕하세요.')

class Univ:
    def manage_grade(self):
        print('학점 관리')

class Student(Person, Univ):
    def study(self):
        print('공부하기')

james = Student()
james.greeting()      # 안녕하세요.: 기반 클래스 Person의 메서드 호출
james.manage_grade()  # 학점 관리: 기반 클래스 Univ의 메서드 호출
james.study()         # 공부하기: 파생 클래스 Student에서 추가한 study 메서드
```

안녕하세요.
학점 관리
공부하기

In []:

