클래스 시작하기

학습 목표

- 클래스에 대해서 직접 만들어보면서 기본을 이해해본다.
- 함수와 클래스를 비교해 보면서 왜 클래스를 사용하는지 이해해본다.

목차

01 함수와 클래스

- 함수는 def 예약어로 시작, 클래스는 class의 예약어로 시작.

목차로

In [1]:

```
def Cal():
    pass
```

In [2]:

```
# 클래스 기본 선언 형태
class Cal:
    pass
```

In [3]:

```
### 인스턴스
### 클래스를 선언 후, 다음과 같이 2대의 계산기를 만들 수 있다.
a = Cal()
b = Cal()
```

02 함수를 이용한 계산기 구현하기

- global을 이용하여 전역변수의 형태로 변수를 사용할 수 있다.

목차로

In [4]:

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

print( plus1(3) )  # 기본값 0 + 3
print( plus1(7) )  # 기본값 0 + 3 + 7
```

```
3
10
```

In [5]:

```
# 두번째 계산기
result1 = 0
def plus2(num):
    global result1
    result1 += num
    return result1

print( plus2(2) )  # 기본값 0 + 2
print( plus2(10) ) # 기본값 0 + 10
```

```
2
12
```

- [확인] 첫번째 계산기는 result로 결과값을 변경. 두번째 계산기는 result1을 이용

03 여러대의 계산기 만들어보기

- 여러대의 계산기를 만들어보자.

목차로

In [6]:

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

print( plus1(3) )
print( plus1(7) )

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

print( plus2(2) )
print( plus2(10) )

# 세번째 계산기
result3 = 0
def plus3(num):
    global result3
    result3 += num
    return result3

print( plus3(2) )
print( plus3(10) )
```

```
3
10
2
12
2
12
```

In [7]:

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

print( plus1(3) )
print( plus1(7) )
```

```
3
10
```

In [8]:

```
### 계산기1, 계산기2, 계산기3 의 현재 결과값
print( "계산기1 현재 결과 :", result1 )
print( "계산기2 현재 결과 :", result2 )
print( "계산기3 현재 결과 :", result3 )
```

```
계산기1 현재 결과 : 10
계산기2 현재 결과 : 12
계산기3 현재 결과 : 12
```

In [9]:

```
# 계산기 1번의 3을 더하기
print( plus1(3) )

# 계산기 1번의 결과에 1을 더하기
print( plus1(1) )
```

```
13
14
```

In [10]:

```
# 계산기 3번의 3을 더하기
print( plus3(3) )

# 계산기 3번의 5를 더하기
print( plus3(1) )
```

```
15
16
```

[생각해보기] 계산기를 10대를 만들고 싶다. 어떻게 해야 할까?

In [11]:

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

# 두번째 계산기
result1 = 0
def plus2(num):
    global result1
    result1 += num
    return result1
```

In [12]:

```
plus1(2)
```

Out[12]:

2

In [13]:

```
print( plus1(5) )
print( plus2(1) )
```

7
1

04 클래스을 활용한 계산기 만들기

- 클래스를 이용하여 계산기를 만들어보자.

목차로

In [14]:

```
class Cal:
    result = 0

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

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

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

다섯대의 계산기 만들기

In [15]:

```
# 인스턴스 (객체를 생성)
c1 = Cal()
c2 = Cal()
c3 = Cal()
c4 = Cal()
c5 = Cal()
```

In [16]:

```
# 계산기1에 3을 두번 더한다.
print( c1.plus(3) )
print( c1.plus(3) )

# 계산기2에 4을 두번 더한다.
print( c2.plus(4) )
print( c2.plus(4) )

# 계산기3에 5, 5을 연속으로 더한다.
print( c3.plus(5) )
print( c3.plus(5) )

# 계산기4에 6, 6을 연속으로 더한다.
print( c4.plus(6) )
print( c4.plus(6) )
```

```
3
6
4
8
5
10
6
12
```

5대의 현재 결과를 확인해 보기

In [17]:

```
print( "계산기1 현재 결과 : " , c1.result )  # 0 + 3 + 3
print( "계산기2 현재 결과 : " , c2.result )  # 0 + 4 + 4
print( "계산기3 현재 결과 : " , c3.result )  # 0 + 5 + 5
print( "계산기4 현재 결과 : " , c4.result )  # 0 + 6 + 6
print( "계산기5 현재 결과 : " , c5.result )  # 0
```

```
계산기1 현재 결과 :  6
계산기2 현재 결과 :  8
계산기3 현재 결과 :  10
계산기4 현재 결과 :  12
계산기5 현재 결과 :  0
```

(실습 4-1) 내 계산기 클래스에 값을 빼주는 -와 /를 하는 연산을 추가해 보자.

개별 계산기 연산

In [18]:

```
# 5번 계산기에 4를 더하기
print( "계산기5 현재 결과 : " , c5.result )
print( c5.plus(4) )
print( "계산기5 현재 결과 : " , c5.result )
```

```
계산기5 현재 결과 :  0
4
계산기5 현재 결과 :  4
```

In [19]:

```
# 2번 계산기에 3를 더하기
print( "계산기2 현재 결과 : " , c2.result )
c2.plus(3)
print( "계산기2 현재 결과 : " , c2.result )
```

```
계산기2 현재 결과 :  8
계산기2 현재 결과 :  11
```

In [20]:

```
c2.divide(3)
```

Out[20]:

```
3.6666666666666665
```

In [21]:

```
c2.minus(2)
```

Out[21]:

```
1.6666666666666665
```

In [22]:

```
c3.plus(3)
```

Out[22]:

```
13
```

예외 상황 발생

In [23]:

```
c3.divide(0)
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
~₩AppData₩Local₩Temp₩ipykernel_30236₩1490046659.py in <module>
----> 1 c3.divide(0)

~₩AppData₩Local₩Temp₩ipykernel_30236₩1537597794.py in divide(self, num)
     11
     12     def divide(self, num):
---> 13         self.result /= num
     14         return self.result

ZeroDivisionError: division by zero
```

- [Error 내용] 0으로 나눌 수 없어 에러 발생

05 에러를 개선한 계산기

- 에러를 개선한 계산기를 만들어보자

목차로

In [24]:

```
class Cal_change():
    pass
```

[생각해 보기] 기존의 계산기를 다 다시 구현해야 할까? 아니면 다른 방법이 없을까?

[기본] 기존의 클래스의 기능 상속이 가능하다.

- 기존의 클래스의 모든 기능에 대해 사용이 가능하다.
- 기본 문법

```
class 클래스명(상속받을 클래스명):
추가할 실행문1
추가할 실행문2
```

In [25]:

```
# 아무 것도 없다.
# 어떤 클래스를 상속을 받아서,
# 해당 클래스가 가진 변수 및 메소드를 사용 가능하다.
class Cal_change(Cal):
    pass
```

In [26]:

```
c_ch1 = Cal_change()

# 3을 더하고, 5를 빼기
print( "기능 개선 계산기 현재 결과 : " , c_ch1.result )
c_ch1.plus(3)
print( "기능 개선 계산기 현재 결과 : " , c_ch1.result )
c_ch1.minus(5)
print( "기능 개선 계산기 현재 결과 : " , c_ch1.result )
```

```
기능 개선 계산기 현재 결과 :  0
기능 개선 계산기 현재 결과 :  3
기능 개선 계산기 현재 결과 :  -2
```

06 오버라이딩(overridding)을 이해하기

- 상속의 개념을 이해해 보자.
- 기존의 메소드(함수)와 동일한 이름으로 정의할 경우, 상속받은 기능보다 우선적으로 기능이 수행된다.
- 기존에 0이 들어올 경우, 에러 발생하여, 이를 보완한 클래스 생성

목차로

In [27]:

```
# 오버라이딩
# 상속받아서 사용하는데, 기존에 상속받은 존재하는 메소드를 변경하는 것.
class Cal_change(Cal):
    def divide(self, num):
        if num==0:
            return "0으로 나눌 수 없습니다."
        else:
            self.result /= num
        return self.result
```

In [28]:

```
c_ch2 = Cal_change()

# 3을 더하고, 5를 빼기
print( "기능 개선 계산기 현재 결과 : " , c_ch2.result )
print( c_ch2.plus(3) )
print( "기능 개선 계산기 현재 결과 : " , c_ch2.result )
print( c_ch2.minus(5) )
print( "기능 개선 계산기 현재 결과 : " , c_ch2.result )
print( c_ch2.divide(0) )
```

```
기능 개선 계산기 현재 결과 :  0
3
기능 개선 계산기 현재 결과 :  3
-2
기능 개선 계산기 현재 결과 :  -2
0으로 나눌 수 없습니다.
```

(실습 4-2) Cal 클래스를 상속받아서,

- 해당 클래스에 하나의 메소드(곱하기)를 추가해 봅시다.
- 그리고 계산기에 C(0으로 세팅하는) 기능을 추가해 봅시다.
- 생성한 계산기로 아래 연산을 수행해 보자.
  - 초기값 + 10 / 0 * 5
  - C를 눌러 초기화
  - 결과값 + 5

In [29]:

```
# 오버라이딩
# 상속받아서 사용하는데, 기존에 상속받은 존재하는 메소드를 변경하는 것.
class Cal_change(Cal):
    def Czero(self):
        self.result = 0
        return self.result

    def mul(self, num):
        self.result *= num
        return self.result

    def divide(self, num):
        if num==0:
            return "0으로 나눌 수 없습니다."
        else:
            self.result /= num
        return self.result
```

In [30]:

```
c1 = Cal_change()
print( c1.plus(10) )
print( c1.divide(0) )
print( c1.mul(5) )
print( c1.Czero() )
print( c1.plus(5))
```

```
10
0으로 나눌 수 없습니다.
50
0
5
```

[실습] tv class를 생성해 보자.

- tv class 이름 : Tv_Basic
- tv 채널 기본 : channel = 0
- tv 채널 변경 : change_channel(self, num) : num 채널로 변경

[도전] 나만의 자동차 class를 만들어보자.

- 자동차 색
- 자동차 속도 변경
- 자동차 전진, 후진, 오른쪽 회전, 왼쪽 회전