22.여러가지 메서드

클래스 변수

- 모든 인스턴스화 된 객체에서 공통으로 사용 할 수 있는 변수
  . 클래스 변수 [클래스명.변수] 로 접근 가능 하며 인스턴스화 된 모든 객체에서 값을 공유 한다.
  . 단 인스턴스 객체에서 클래스 변수 에 직접 값을 할당 할 경우 그시점 부터 해당 객체의 인스턴스 변수로 전환된다.

```
156  class Car:
157      totalcount = 0
158      def __init__(self,year) :
159          self.age    = year
160          Car.totalcount += 1
161
162  print("차량 총 생산량 : " , Car.totalcount)
163  pride    = Car(2023)
164  korando = Car(2022)
165  print("차량 총 생산량 : " , Car.totalcount)
```

```
차량 총 생산량 :  0
차량 총 생산량 :  2
```

> 157행 : totalcount 를 클래스 변수로 지정

>160행 : 클래스 인스턴스화 시 클래스 변수에 +1 씩 누적

>165행 : 두개의 객체가 생성 되어 2 가 출력되는것을 확인 할 수 있다.

- 클래스 변수 의 생성 과정
  . car 클래스 인스턴스 화 시작 = 163행 오른쪽 리터럴 'Car(2023)'

메모리

heap	
[메모리주소]100	Car

. Data 영역에 totalcount 변수 가 0 으로 초기화 되어 등록 = 157행

메모리

Data	
Car.totalcount	0

heap	
100	Car

. Totalcount 클래스 변수 에 1 증가 = 160행

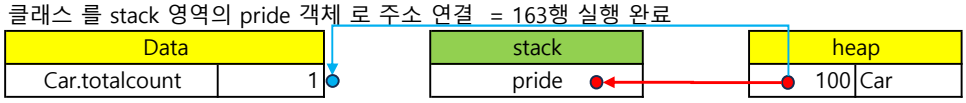
메모리

Data	
Car.totalcount	1

heap	
100	Car

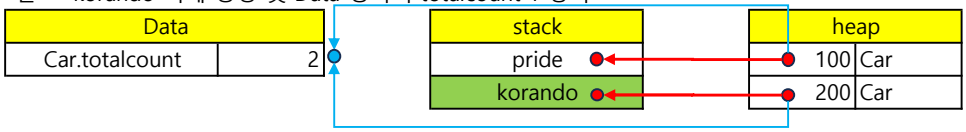
. 인스턴스 화 된 car 클래스 를 stack 영역의 pride 객체 로 주소 연결 = 163행 실행 완료

메모리



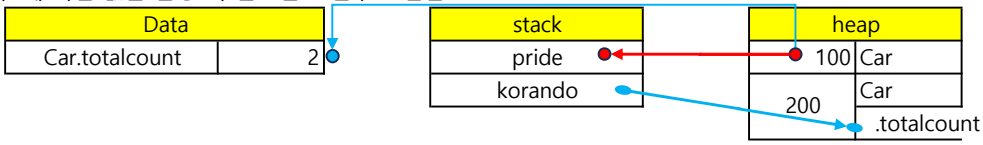
. 새로운 클래스 인스턴스 korando 객체 생성 및 Data 영역의 totalcount 1 증가

메모리



. 객체의 클래스 변수 에 직접 값을 할당 시 인스턴스 변수로 접근

메모리



클래스 메서드

- 클래스 전체가 공유하는 메서드
  . 클래스 메서드 @classmethod 데코레이터 로 표현
  . 클래스 자신과 인스턴스 화 된 클래스 에서 모두 공유 가능한 메서드
  . cls : class 를 줄인 말로서 보편적으로 사용하는 이름이며 클래스 메서드의 첫번째 매개변수로는 반드시 클래스 인자를 지정해야함
  * cls 인자를 통하여 클래스 변수에 접근하거나 수정하는데 주로 사용됨

```
200 class Car:
201     count = 0
202     def __init__(self) :
203         Car.count += 1
204 
205     @classmethod
206     def outcount(cls,cartype): # cls 는 현재 class Car 를 뜻함.
207         cls.count += 1
208         print(cartype , cls.count,sep=" : ")
209 
210 Car.outcount("korando")
211 
212 
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
korando : 1
```

정적 메서드

- 프로그램 내부에서 별도의 객체를 생성 하지 않고 곧바로 접근하여 사용할 수 있는 메서드
  . 프로그램 전체에서 공통으로 수행하는 메서드 를 만들고 싶을때 클래스를 매번 인스턴스화 하지 않고 사용 가능
  . 클래스 인자를 받을 필요가 없이 독립적으로 수행 (@classmethod 와 다른 점)

```
Chap01_intro > 21.Class[여러가지메서드].py > ...
1  class Car:
2      def __init__(self) :
3          pass
4 
5      @staticmethod
6      def GetDbConnection():
7          return "Databasd Cnnection : 111.111.222.222"
8 
9  dbcon = Car.GetDbConnection()
10 print(dbcon)
11 
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
Microsoft Windows [Version 10.0.22621.2283]
(c) Microsoft Corporation. All rights reserved.

D:\Python>C:/Users/MasterD/AppData/Local/Programs/Python/Python311/python.exe d:/Python/Chap
Databasd Cnnection : 111.111.222.222
```

>5행 : outcount 메서드 를 정적 메서드 로 선언

>9행 : Car 객체를 선언하지 않고 곧바로 outcount() 를 호출
>10행 : Car.outcount

연산자 메서드

- 자주 사용하는 연산 기능을 메서드 로 만들어 간편하게 사용할 수 있도록 하는 메서드
- 클래스에 연산자 메서드를 정의 하면 객체에 대해서도 연산자 를 이용 할 수 있다.

연산자	메서드	메서드
==	__eq__	
!=	__ne__	
<	__it__	
>	__gt__	
<=	__le__	
>=	__ge__	
+	__add__	__radd__
-	__sub__	__rsub__
*	__mul__	__rmul__
/	__div__	__rdiv__
/(division 임포트)	__trudiv__	__rturdiv__
//	__floordiv__	__rfloordiv__
%	__mod__	__rmod__
**	__pow__	__rpow__
<<	__lshift__	__rlshift__
>>	__rshift__	__lshift__

. 연산자 메서드 사용의 예

> 단순히 객체 를 비교 할 경우 서로다른 객체이므로 False 를 반환한다.

```
Chap01_intro > 21.Class[여러가지메서드].py > ...
1   class Human:
2       def __init__(self,age,name):
3           self.age = age
4           self.name = name
5
6   kim = Human(38,'김범수')
7   bum = Human(38,'김범수')
8   lee = Human(39,'이수')
9   print(kim == bum)
10  print(kim == lee)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Microsoft Windows [Version 10.0.22621.2283]
(c) Microsoft Corporation. All rights reserved.

D:\Python>C:/Users/MasterD/AppData/Local/Programs/Python/Python311/p
False
False
```

>9행 : kim객 체와 bum 객체 의 비교

>10행 : Kim 객체 와 lee 객체의 비교

> 객체 를 비교하는 __eq__ 메서드를 추가

* self : 비교하는 기준 객체

  other : 비교 할 객체

```
Chap01_intro > 21.Class[여러가지메서드].py > ...
1   class Human:
2       def __init__(self,age,name):
3           self.age = age
4           self.name = name
5
6       def __eq__(self,other):
7           return self.age == other.age and self.name == other.name
```

> __eq__ 메서드를 통해 객체의 연산 처리

* 서로 다른 두 객체의 요소를 비교하여 Ttrue, False 를 반환

```
Chap01_intro > 21.Class[여러가지메서드].py > ...
1   class Human:
2       def __init__(self,age,name):
3           self.age = age
4           self.name = name
5
6       def __eq__(self,other):
7           return self.age == other.age and self.name == other.name
8
9   kim = Human(38,'김범수')
10  bum = Human(38,'김범수')
11  lee = Human(39,'이수')
12  print(kim == bum)
13  print(kim == lee)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Microsoft Windows [Version 10.0.22621.2283]
(c) Microsoft Corporation. All rights reserved.

D:\Python>C:/Users/MasterD/AppData/Local/Programs/Python/Python311/python.exe d:/Python/Chap01_intro/21.Clas
True
False
```

* kim 과 bum 은 서로 다른 객체 이지만 __eq__ 연산자 메서드를 통하여 요소인 name 과 age 를 비교 하여 True 를 반환

. 연산자 메서드 활용 예 2

> 두 클래스 의 차(-) 를 구하면 오류가 발생한다.

```
Chap01_intro > 21.Class[여러가지메서드].py > ...
1  class Human:
2      def __init__(self,age,name):
3          self.age = age
4          self.name = name
5
6      def __eq__(self,other):
7          return self.age == other.age and self.name == other.name
8
9  kim = Human(38,'김범수')
10 lee = Human(39,'이수')
11 print('두 사람의 나이차는 : ' ,lee -  kim  ,' 입니다')
12
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Microsoft Windows [Version 10.0.22621.2283]
(c) Microsoft Corporation. All rights reserved.

D:\Python>C:/Users/MasterD/AppData/Local/Programs/Python/Python311/python.exe d:/Python/Chap01_intro/21.Clas
Traceback (most recent call last):
  File "d:\Python\Chap01_intro\21.Class[여러가지메서드].py", line 11, in <module>
    print('두 사람의 나이차는 : ' ,lee -  kim  ,' 입니다')
TypeError: unsupported operand type(s) for -: 'Human' and 'Human'
```

> 클래스 간 차(-) 를 구하는 연산자를 만났을 경우 수행하는 로직을 구현하면 해당 로직으로 처리 하여 원하는 결과 를 얻을 수 있다.

```
9      def __sub__(self,other):
10         return abs(self.age - other.age)
11
12 kim = Human(38,'김범수')
13 lee = Human(39,'이수')
14 print('두 사람의 나이차는 : ' ,lee -  kim  ,' 입니다')
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Microsoft Windows [Version 10.0.22621.2283]
(c) Microsoft Corporation. All rights reserved.

D:\Python>C:/Users/MasterD/AppData/Local/Programs/Python/Python311/python.exe d:/Python/Ch
두 사람의 나이차는 :  1  입니다
```

* abs : 연산 결과를 절대값 으로 표현하는 함수

. __sub__ 와 __rsub__

> 객체 를 좌변에 두는지 우변에 두는지 에 따라 결정 된다.

* __sub__ : 처리 하려고 하는 객체 가 좌변 에 있을경우

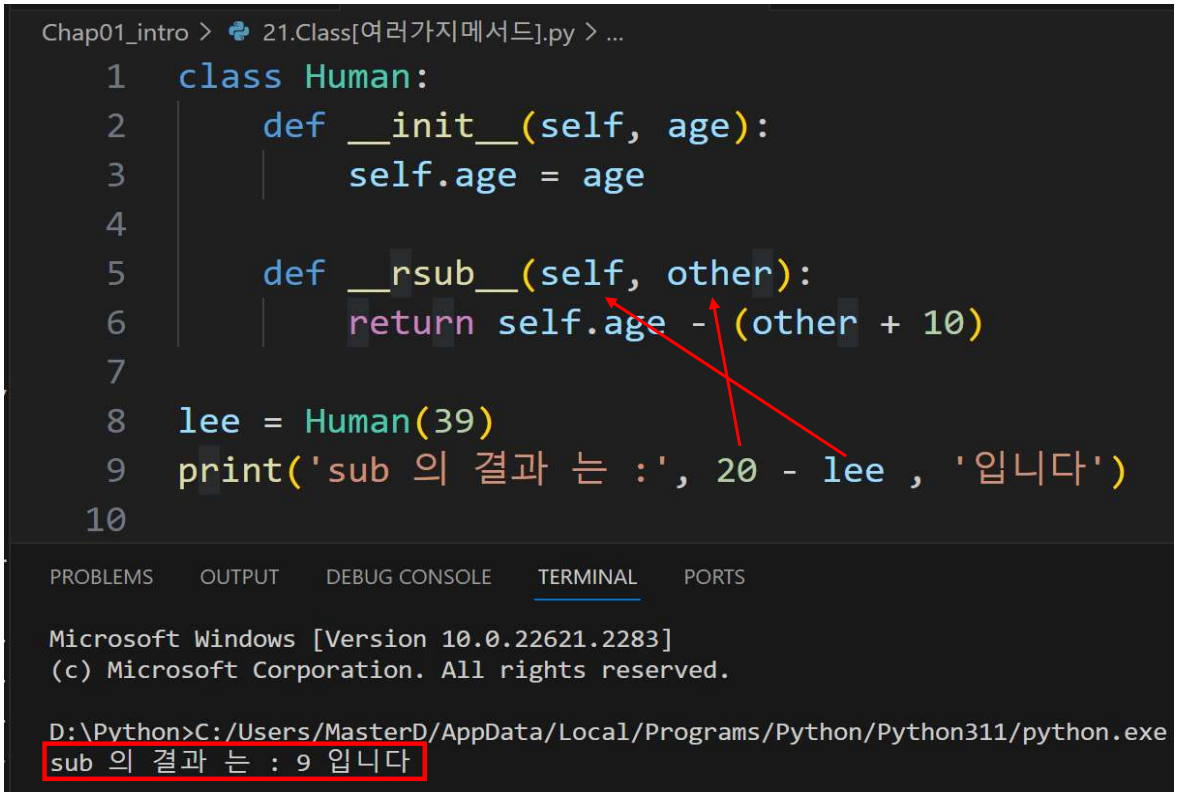
```
Chap01_intro > 21.Class[여러가지메서드].py > ...
1  class Human:
2      def __init__(self, age):
3          self.age = age
4
5      def __sub__(self, other):
6          return abs(self.age - other)
7
8  lee = Human(39)
9  print('sub 의 결과 는 :', lee - 20 , '입니다')
10
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Microsoft Windows [Version 10.0.22621.2283]
(c) Microsoft Corporation. All rights reserved.

D:\Python>C:/Users/MasterD/AppData/Local/Programs/Python/Python311/python.exe d:
sub 의 결과 는 : 19 입니다
```

* __rsub__ : 연산자 메서드 객체 가 우변 에 있을경우

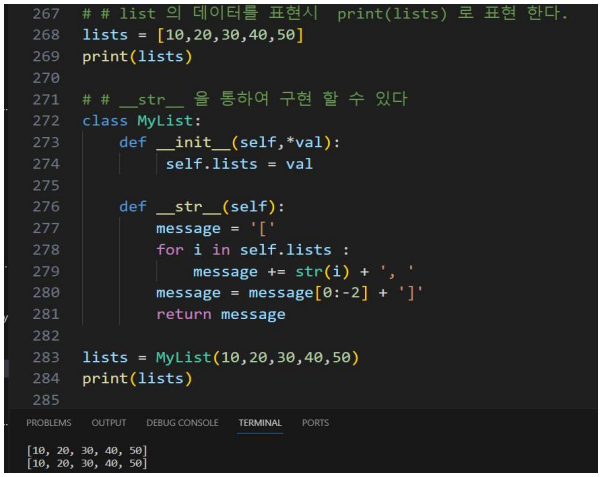


특수 메서드

- 객체 자신이 호출 될때 구현해둔 기능을 수행한다

- __str__

. 객체를 약속된 형식으로 문자열 화 하여 표현한다.



- __len__

. 객체 의 길이 를 조사 할때 값을 반환한다.