UNISTUDY

파이썬

21강. 생성자

1. 생성자

- 생성자(Constructor)는 init () 이란 이름으로 제공되고, 객체가 생성될 때 자동으로 실행된 다. 생성자의 역할은 객체 생성 시 멤버변수에 값을 초기화하는 역할을 한다. 만약 초기화 작업이 필요 없는 경우에는 클래스 설계 시 생성자를 생략할 수 있다. 한편 모든 클래스는 객체를 생성하는 것이 목적이므로 반드시 생성자를 가지고 있어야한다.
- 파이 썬에서 기본 생성자라는 것을 만들어주고, 이를 통해서 객체가 생성되도록 한다.
- 다음은 생성자를 이용하여 객체를 생성하는 형식이다.
- 클래스에서 정의된 생성자의 이름과는 다르게 객체를 생성할 때는 생성자가 포함된 클래스이름 뒤에 괄호()을 붙이고, 매개변수에 전달될 실인수를 괄호에 넣어서 함수를 호출하듯이 객체를 생성한다. 이렇게 생성된 객체의 주소는 참조변수에 할당되 고, 참조변수를 이용해서 객체의 멤버를 호출할 수 있다.

```
참조변수 = 클래스이름 (실인수) ↵
```

1. 생성자

chapter06.lecture.step02_constructor.py

```
# (1) 생성자 이용 멤버변수 초기화
class multiply :
    # 멤버 변수
    x = y = 0

    # 생성자 : 초기화
    def __init__(self, x, y): # 객체만 생성
        self.x = x
        self.y = y

    # 메서드
    def mul(self):
        return self.x * self.y

obj = multiply(10, 20) # 생성자
print('곱셈 =', obj.mul())


# (2) 메서드 이용 멤버변수 초기화
class multiply2 :
    # 멤버 변수
    x = y = 0
```

Python Console

```
곱셈 = 200
```

1. 생성자

```
    # 생성자 없음 : 기본 생성자 제공
    def __init__(self):
        pass
    # 메서드 : 멤버변수 초기화
    def data(self, x, y):
        self.x = x
        self.y = y


    # 메서드 : 곱셈
    def mul(self):
        return self.x * self.y

obj = multiply2() # 기본 생성자
obj.data(10, 20) # 동적 멤버변수 생성
print('곱셈 =', obj.mul())                    곱셈 = 200
```

1. 생성자

- 생성자 예
- # (1) 생성자 이용 멤버변수 초기화
- multiply 클래스는 두 개의 멤버변수(x, y)에 값을 초기화하기 위해서 두 개의 매개변수를 갖는 생성자 init (self, x, y)를 정의하고, 객체가 생성되는 시점에서 실인수 2개가 매개변수로 할당되어 멤버변수 x, y에 초기화된다. 그리고 mul() 메서드는 멤버변수 x와 y를 곱해서 반환하는 함수이다.
- # (2) 메서드 이용 멤버변수 초기화
- multiply2 클래스는 생성자를 이용하여 멤버변수에 값을 초기화하지 않고, data() 메서드에 의해서 초기화되므로 생성자를 쓰지 않았다. 생성자를 생략하면 다음과 같은 기본 생성자(묵시적 생성자)가 자동으로 만들어진다.
- def init (self): pass
- 여기서 pass 명령어는 블록의 내용이 없는 경우 쓸 수 있는 명령어이다. 따라서 기본 생성자는 실행문 이 없는 단지 객체만 생성하는 용도로 사용되다. 코드가 보이지 않아도 암시적으로 클래스에 만들어지 기 때문에 묵시적 생성자라고도 한다.

2. 소멸자(Destructor)

- 생성자의 반대 역할을 하는 소멸자는 del () 이란 이름으로 제공되고, 객체 사용이 완료되면 자동으로 실행되어 객체를 메모리에서 소멸시키는 역할을 한다. 하지만 파이썬 자체에서 메모리 관리를 자동으로 해 주기 때문에 많이 사용되지는 않는다. 다음은 multiply 클래스에서 생성자와 소멸자를 작성한 예문이다.
- class multiply :
- # 생성자 : 객체 생성 + 멤버변수 초기화 def init (self, x, y):
- self.x = x self.y = y

- # 소멸자 : 객체 소멸 def del (self):
- del self.x del self.y

3. self

- 클래스의 생성자와 메서드는 기본적으로 self라는 인수를 갖는다. 만약 self 인수를 생략하면 오류가 발생한다.
- 그렇다면 self 인수는 어떤 역할을 하는가? 클래스를 구성하는 멤버들 즉 멤버변수와 메서 드를 호출하는 역할을 한다.
- 예를 들면 생성자 안에서 멤버변수에 값을 초기화하거나 메서드 안에서 멤 버변수를 참조하거나 또는 다른 메서드를 호출할 경우 이용한다. self 명령문 형식은 다음과 같다.

```
self.멤버변수
self.메서드()
```

3. self

chapter06.lecture.step02_constructor.py

```
class multiply3 :
    # 멤버변수 없음
    # 생성자 없음

    # 동적 멤버변수 생성/초기화
    def data(self, x, y):
        self.x = x


        self.y = y

    # 곱셈 연산
    def mul(self):
        result = self.x * self.y
        self.display(result) # 메서드 호출

    # 결과 출력
    def display(self, result):
        print("곱셈 = %d" %(result) )


obj = multiply3() # 기본 생성자
obj.data(10, 20)
obj.mul()
```

Python Console

```
곱셈 = 200
```

3. self

- self 명령어 예
- multiply3 클래스는 멤버변수와 생성자를 모두 정의하지 않다. 생성자는 기본 생성자가 자동으로 만들어질 것이다. 한편 곱셈 연산을 위해서는 두 개의 자료가 만들어져한다. 하지만 클래스 선언부에 멤버변수가 정의되지 않았기 때문에 data() 메서드가 호출되는 시점에서 self.x, self.y 명령문에 의해서 동적으로 멤버변수가 만들어지고 매개변수에 의해서 값이 초기화된다. 멤버변수와 매개변수는 대입연산자(=)를 기준으로 왼쪽에는 self.멤버변수, 오른쪽은 매개변수로 작성한다.
- self.멤버변수 = 매개변수
- 이렇게 만들어진 멤버변수는 mul()과 display() 메서드에서 'self.매개변수' 형태로 참조한다. 또한 mul() 메서드에서는 곱셈 결과를 출력하기 위해서 display() 메서드를 'self.메서드' 형태로 호출하고 있다.

4. 클래스 멤버

- 객체는 클래스의 생성자를 이용해서 만들어진다. 그리고 객체를 구성하는 객체 멤버는 객체 변수, 객체 메서드가 된다. 이들은 모두 객체(참조 변수)를 통해서 호출이 가능하다.
- 클래스 멤버는 클래스이름으로 호출할 수 있는 클래스 변수와 클래스 메서드를 말한다. 클래스 이름으 로 호출할 수 있기 때문에 클래스 멤버를 호출하기 위해서 객체를 생성할 필요는 없다.
- 객체 멤버와 클래스 멤버를 비교한 표이다.
- 특히 클래스 메서드는 cls라는 기본인수를 사용하고, @classmethod라는 함수 장식자를 이용하여 선언한다.

4. 클래스 멤버

구분	객체 멤버	클래스 멤버
구성요소	객체 변수, 객체 메서드	클래스 변수, 클래스 메서드
객체 생성	객체 생성 필요	객체 생성 필요 없음
멤버 참조	객체의 참조변수 이용	클래스이름 이용
기본 인수	객체 메서드(self)	클래스 메서드(cls)
함수 장식자	필요 없음	필요함(@classmethod)

4. 클래스 멤버

chapter06.lecture.step03_class_member.py

Python Console

```
class DatePro :
    # (1) 멤버 변수
    content = "날짜 처리 클래스"

    # (2) 생성자
    def __init__(self, year, month, day):
        self.year = year
        self.month = month
        self.day = day

    # (3) 객체 메서드(instance method)
    def display(self):
        print("%d-%d-%d"%(self.year, self.month,
self.day))

    # (4) 클래스 메서드(class method)
    @classmethod # 함수 장식자
    def date_string(cls, dateStr): # '19951025'
        year = dateStr[:4]
        month = dateStr[4:6]
        day = dateStr[6:]

        print(f"{year}년 {month}월 {day}일")
```

4. 클래스 멤버

```
# (5) 객체 멤버
date = DatePro(1995, 10, 25) #  생성자
print(date.content) # 날짜 처리 클래스
print(date.year) # 1995
date.display() # 1995-10-25
```

```
날짜 처리 클래스
1995
1995-10-25
```

```
# (6) 클래스 멤버
print(DatePro.content) # 날짜 처리 클래스
#print(DatePro.year) # AttributeError
DatePro.date_string('19951025') # 1995년 10월 25일
```

```
날짜 처리 클래스
1995년 10월 25일
```

4. 클래스 멤버

- 클래스 멤버 예
- # (1) 멤버 변수
- 클래스 선언부에 선언된 content 변수는 기존과 같이 객체의 멤버변수가 된다. 또한 클래스 멤버변수 가 될 수 있다.
- # (2) 생성자
- 생성자에서 'self.멤버변수'로 선언된 변수는 객체 멤버변수만 된다. 클래스 멤버변수는 될 수 없다.
- # (3) 객체 메서드(instance method)
- 객체 메서드는 기본 인수 self를 포함하여 함수가 정의된다. 이러한 함수는 생성자를 이용하여 객체를 생성하고, 객체의 참조변수에 의해서 호출되는 객체 메서드이다.
- # (4) 클래스 메서드(class method)
- 클래스 메서드는 '@classmethod'라는 함수 장식자를 이용하여 먼저 선언한 후 클래스 메서드를 정의한다. 클래스 메서드는 기본 인수 cls를 포함한다. 이러한 함수는 클래스이름으로 호출되는 클래스 메서드이다.

4. 클래스 멤버

- # (5) 객체 멤버
- 생성자를 이용하여 객체를 생성하고, 객체 멤버변수와 객체 메서드를 호출하여 날짜 정보를 출력한다. self로 지정되지 않은 content 멤버변수도 객체 멤버가 된다.
- # (6) 클래스 멤버
- DatePro 클래스이름으로 클래스 멤버를 호출한다. self로 지정되지 않은 content는 클래스 멤버변 수가 될 수 있다. 또한 함수장식자에 의해서 선언된 date_string()은 클래스 메서드 이므로 클래스이 름으로 호출할 수 있다. 만약 self.year로 정의된 멤버변수를 클래스이름으로 호출할 경우에는 오류가 발생한다. year는 객체의 멤버변수이기 때문이다.

THANK YOU