UNIT 35

클래스 속성과 정적, 클래스 메서드 사용하기

35.1 클래스 속성과 인스턴스 속성 알아보기

클래스 속성 사용하기

- 클래스 속성은 다음과 같이 클래스에 바로 속성을 만듦

```
def 클래스이름:
    속성 = 값
```

35.1 클래스 속성과 인스턴스 속성 알아보기

» 클래스 속성 사용하기

- 다음과 같이 Person 클래스에 바로 bag 속성을 넣고, put_bag 메서드를 만듦
- 인스턴스 두 개를 만든 뒤 각각 put_bag 메서드를 사용함

class_class_attribute.py

```
class Person:
    bag = []

    def put_bag(self, stuff):
        self.bag.append(stuff)

james = Person()
james.put_bag('책')

maria = Person()
maria.put_bag('열쇠')

print(james.bag)
print(maria.bag)
```

실행 결과

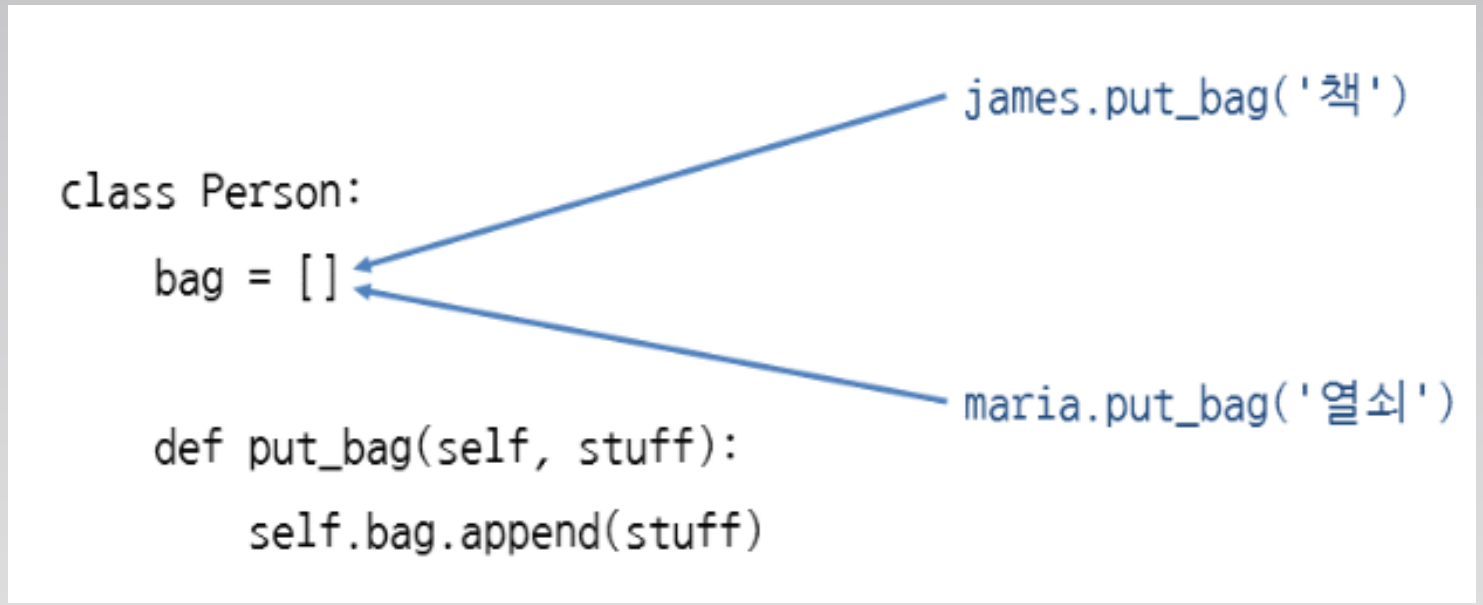
```
['책', '열쇠']
['책', '열쇠']
```

35.1 클래스 속성과 인스턴스 속성 알아보기

» 클래스 속성 사용하기

- 클래스 속성은 클래스에 속해 있으며 모든 인스턴스에서 공유함

▼ 그림 클래스 속성



35.1 클래스 속성과 인스턴스 속성 알아보기

» 클래스 속성 사용하기

- put_bag 메서드에서 클래스 속성 bag에 접근할 때 self를 사용함
- 사실 self는 현재 인스턴스를 뜻하므로 클래스 속성을 지칭하기에는 조금 모호함

```
class Person:
    bag = []

    def put_bag(self, stuff):
        self.bag.append(stuff)
```

- 클래스 속성에 접근할 때는 다음과 같이 클래스 이름으로 접근하면 좀 더 코드가 명확해짐

• 클래스.속성

```
class Person:
    bag = []

    def put_bag(self, stuff):
        Person.bag.append(stuff)    # 클래스 이름으로 클래스 속성에 접근
```

35.1 클래스 속성과 인스턴스 속성 알아보기

클래스 속성 사용하기

- Person.bag이라고 하니 클래스 Person에 속한 bag 속성이라는 것을 바로 알 수 있음
- 클래스 바깥에서도 다음과 같이 클래스 이름으로 클래스 속성에 접근하면 됨

```
print(Person.bag)
```

35.1 클래스 속성과 인스턴스 속성 알아보기

인스턴스 속성 사용하기

- 가방을 여러 사람이 공유하지 않으려면 bag을 인스턴스 속성으로 만들면 됨

class_instance_attribute.py

```
class Person:
    def __init__(self):
        self.bag = []

    def put_bag(self, stuff):
        self.bag.append(stuff)

james = Person()
james.put_bag('책')

maria = Person()
maria.put_bag('열쇠')

print(james.bag)
print(maria.bag)
```

실행 결과

```
['책']
['열쇠']
```

35.1 클래스 속성과 인스턴스 속성 알아보기

인스턴스 속성 사용하기

- 인스턴스 속성은 인스턴스별로 독립되어 있으며 서로 영향을 주지 않음
- 클래스 속성과 인스턴스 속성의 차이점을 정리해보자

- 클래스 속성: 모든 인스턴스가 공유. 인스턴스 전체가 사용해야 하는 값을 저장할 때 사용
- 인스턴스 속성: 인스턴스별로 독립되어 있음. 각 인스턴스가 값을 따로 저장해야 할 때 사용

35.1 클래스 속성과 인스턴스 속성 알아보기

비공개 클래스 속성 사용하기

- 클래스 안에서만 접근할 수 있고, 클래스 바깥에서는 접근할 수 없음

```
def 클래스이름:
    __속성 = 값    # 비공개 클래스 속성
```

35.1 클래스 속성과 인스턴스 속성 알아보기

비공개 클래스 속성 사용하기

- 클래스에서 공개하고 싶지 않은 속성이 있다면 비공개 클래스를 사용해야 함

class_private_class_attribute_error.py

```
class Knight:
    __item_limit = 10    # 비공개 클래스 속성

    def print_item_limit(self):
        print(Knight.__item_limit)    # 클래스 안에서만 접근할 수 있음


x = Knight()
x.print_item_limit()    # 10

print(Knight.__item_limit)    # 클래스 바깥에서는 접근할 수 없음
```

실행 결과

```
10
Traceback (most recent call last):
  File "C:\project\class_private_class_attribute_error.py ", line 11, in <module>
    print(Knight.__item_limit)    # 클래스 바깥에서는 접근할 수 없음
AttributeError: type object 'Knight' has no attribute '__item_limit'
```

35.2 정적 메서드 사용하기

정적 메서드 사용하기

- 정적 메서드는 다음과 같이 메서드 위에 @staticmethod를 붙임
- 정적 메서드는 매개변수에 self를 지정하지 않음

```
class 클래스이름:
    @staticmethod
    def 메서드(매개변수1, 매개변수2):
        코드
```

- @staticmethod처럼 앞에 @이 붙은 것을 데코레이터라고 하며 메서드(함수)에 추가 기능을 구현할 때 사용함

35.2 정적 메서드 사용하기

정적 메서드 사용하기

- 그럼 간단하게 덧셈과 곱셈을 하는 클래스를 만들어보자

class_static_method.py

```
class Calc:
    @staticmethod
    def add(a, b):
        print(a + b)

    @staticmethod
    def mul(a, b):
        print(a * b)

Calc.add(10, 20)    # 클래스에서 바로 메서드 호출
Calc.mul(10, 20)    # 클래스에서 바로 메서드 호출
```

실행 결과

```
30
200
```

35.2 정적 메서드 사용하기

정적 메서드 사용하기

- 클래스.메서드()

```
Calc.add(10, 20)    # 클래스에서 바로 메서드 호출
Calc.mul(10, 20)    # 클래스에서 바로 메서드 호출
```

- 정적 메서드는 self를 받지 않으므로 인스턴스 속성에는 접근할 수 없음
- 정적 메서드는 인스턴스 속성, 인스턴스 메서드가 필요 없을 때 사용함
- 정적 메서드는 메서드의 실행이 외부 상태에 영향을 끼치지 않는 순수 함수(pure function)를 만들 때 사용함
- 순수 함수는 부수 효과(side effect)가 없고 입력 값이 같으면 언제나 같은 출력 값을 반환함
- 정적 메서드는 인스턴스의 상태를 변화시키지 않는 메서드를 만들 때 사용함

35.3 클래스 메서드 사용하기

클래스 메서드 사용하기

- 클래스 메서드는 다음과 같이 메서드 위에 @classmethod를 붙임
- 클래스 메서드는 첫 번째 매개변수에 cls를 지정해야 함(cls는 class에서 따옴)

```
class 클래스이름:
    @classmethod
    def 메서드(cls, 매개변수1, 매개변수2):
        코드
```

35.3 클래스 메서드 사용하기

클래스 메서드 사용하기

- 그럼 사람 클래스 Person을 만들고 인스턴스가 몇 개 만들어졌는지 출력하는 메서드를 만들어보자

class_class_method.py

```
class Person:
    count = 0    # 클래스 속성

    def __init__(self):
        Person.count += 1    # 인스턴스가 만들어질 때
                             # 클래스 속성 count에 1을 더함

    @classmethod
    def print_count(cls):
        print('{0}명 생성되었습니다.'.format(cls.count))    # cls로 클래스 속성에 접근

james = Person()
maria = Person()

Person.print_count()    # 2명 생성되었습니다.
```

```
2명 생성되었습니다.
```

35.3 클래스 메서드 사용하기

클래스 메서드 사용하기

- 클래스 메서드는 정적 메서드처럼 인스턴스 없이 호출할 수 있다는 점은 같지만
  클래스 메서드는 메서드 안에서 클래스 속성, 클래스 메서드에 접근해야 할 때 사용

```
class Person:
    count = 0    # 클래스 속성

    def __init__(self):
        Person.count += 1    # 인스턴스가 만들어질 때
                             # 클래스 속성 count에 1을 더함
```

```
    @classmethod
    def print_count(cls):
        print('{0}명 생성되었습니다.'.format(cls.count))    # cls로 클래스 속성에 접근
```

```
james = Person()
maria = Person()

Person.print_count()    # 2명 생성되었습니다.
```