집합 자료형

집합 자료형은 어떻게 만들까?

집합(set)은 파이썬 2.3부터 지원하기 시작한 자료형으로, 집합에 관련된 것을 쉽게 처리하기 위해 만든 자료형이다.

집합 자료형은 다음과 같이 set 키워드를 사용해 만들 수 있다.

```
>>> s1 = set([1,2,3])
>>> s1
{1, 2, 3}
```

위와 같이 set()의 괄호 안에 리스트를 입력하여 만들거나 다음과 같이 문자열을 입력하여 만들 수도 있다.

```
>>> s2 = set("Hello")
>>> s2
{'e', 'H', 'l', 'o'}
```

> ※ 비어 있는 집합 자료형은 s = set()로 만들수 있다.

집합 자료형의 특징

자, 그런데 위에서 살펴본 set("Hello")의 결과가 좀 이상하지 않은가? 분명 "Hello" 문자열로 set 자료형을 만들었는데 생성된 자료형에는 l 문자가 하나 빠져 있고 순서도 뒤죽박죽이다. 그 이유는 set에 다음과 같은 2가지 큰 특징이 있기 때문이다.

- 중복을 허용하지 않는다.
- 순서가 없다(Unordered).

리스트나 튜플은 순서가 있기(ordered) 때문에 인덱싱을 통해 자료형의 값을 얻을 수 있지만 set 자료형은 순서가 없기(unordered) 때문에 인덱싱으로 값을 얻을 수 없다. 이는 마치 02-5에서 살펴본 딕셔너리와 비슷하다. 딕셔너리 역시 순서가 없는 자료형이라 인덱싱을 지원하지 않는다.

만약 set 자료형에 저장된 값을 인덱싱으로 접근하려면 다음과 같이 리스트나 튜플로 변환한후 해야 한다.

```
>>> s1 = set([1,2,3])
>>> l1 = list(s1)
>>> l1
[1, 2, 3]
>>> l1[0]
1
>>> t1 = tuple(s1)
>>> t1
(1, 2, 3)
>>> t1[0]
1
```

> ※ 중복을 허용하지 않는 set의 특징은 자료형의 중복을 제거하기 위한 필터 역할로 종종 사용하기도 한다.

교집합, 합집합, 차집합 구하기

set 자료형을 정말 유용하게 사용하는 경우는 교집합, 합집합, 차집합을 구할 때이다.

우선 다음과 같이 2개의 set 자료형을 만든 후 따라 해 보자. s1은 1부터 6까지의 값을 가지게 되었고, s2는 4부터 9까지의 값을 가지게 되었다.

```
>>> s1 = set([1, 2, 3, 4, 5, 6])
>>> s2 = set([4, 5, 6, 7, 8, 9])
```

1. 교집합

s1과 s2의 교집합을 구해 보자.

```
>>> s1 & s2
{4, 5, 6}
```

"&" 기호를 이용하면 교집합을 간단히 구할 수 있다.

또는 다음과 같이 intersection 함수를 사용해도 동일한 결과를 돌려준다.

```
>>> s1.intersection(s2)
{4, 5, 6}
```

s2.intersection(s1)을 사용해도 결과는 같다.

2. 합집합

합집합은 다음과 같이 구할 수 있다. 이때 4, 5, 6처럼 중복해서 포함된 값은 한 개씩만 표현된다.

```
>>> s1 | s2
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

"|" 기호를 사용한 방법이다.

```
>>> s1.union(s2)
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

또는 union 함수를 사용하면 된다. 교집합에서 사용한 intersection 함수와 마찬가지로 s2.union(s1)을 사용해도 동일한 결과를 돌려준다.

3. 차집합

차집합은 다음과 같이 구할 수 있다.

```
>>> s1 - s2
{1, 2, 3}
>>> s2 - s1
{8, 9, 7}
```

빼기(-) 기호를 사용한 방법이다.

```
>>> s1.difference(s2)
{1, 2, 3}
>>> s2.difference(s1)
{8, 9, 7}
```

difference 함수를 사용해도 차집합을 구할 수 있다.

집합 자료형 관련 함수들

값 1개 추가하기(add)

이미 만들어진 set 자료형에 값을 추가할 수 있다. 1개의 값만 추가(add)할 경우에는 다음과 같이 한다.

```
>>> s1 = set([1, 2, 3])
>>> s1.add(4)
>>> s1
{1, 2, 3, 4}
```

값 여러 개 추가하기(update)

여러 개의 값을 한꺼번에 추가(update)할 때는 다음과 같이 하면 된다.

```
>>> s1 = set([1, 2, 3])
>>> s1.update([4, 5, 6])
>>> s1
{1, 2, 3, 4, 5, 6}
```

특정 값 제거하기(remove)

특정 값을 제거하고 싶을 때는 다음과 같이 하면 된다.

```
>>> s1 = set([1, 2, 3])
>>> s1.remove(2)
>>> s1
{1, 3}
```