4강

지능형 자료구조(set)

4-1 set 조작

- set
  - ✓ set은 요소가 중복되지 않고 순서대로 저장되지 않는다.
  - ✓ set은 합집합, 교집합, 차집합 등의 연산 함수를 제공한다.
  - ✓ set 생성은 {} 또는 set()를 이용하여 만들 수 있다.

```
sd1 = {'one','three','two','one','two','one'}
print(sd1)
print(type(sd1))
sd2 = set([50,20,10,30,10,20,30,20])
print(sd2)
```

```
[결과]
{'two', 'three', 'one'}
<class 'set'>
{50, 10, 20, 30}
```

  - ✓ 비어 있는 set을 구성할 시 {}만 사용하면 딕셔너리가 생성되므로 set()를 이용한다.

```
sd3 = {}
print(type(sd3))
sd4 = set()
print(type(sd4))
```

```
[결과]
<class 'dict'>
<class 'set'>
```

4-1 set 조작

- set 집합 연산
  - ✓ 집합 연산은 파이썬의 산술 연산자와 논리 연산자를 사용한다.
  - ✓ | 연산자는 합집합을 구하며 함수는 union()을 사용한다.

```
s1 = {1,2,3,4}
s2 = {3,4,5,6}
print(s1.union(s2))
print(s1 | s2)
```

```
[결과]
{1, 2, 3, 4, 5, 6}
{1, 2, 3, 4, 5, 6}
```

  - ✓ & 연산자는 교집합을 구하며 함수는 intersection()을 사용한다.

```
print(s1.intersection(s2))
print(s1 & s2)
```

```
[결과]
{3, 4}
{3, 4}
```

  - ✓ - 연산자는 차집합을 구하며 함수는 difference()을 사용한다.

```
print(s1.difference(s2))
print(s1 - s2)
```

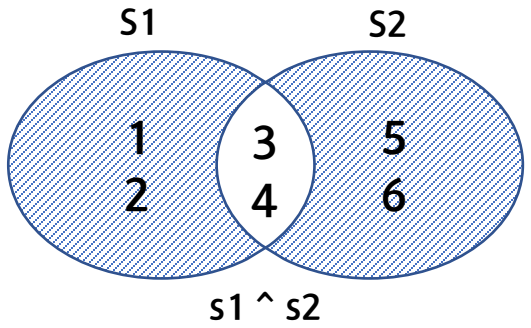
```
[결과]
{1, 2}
{1, 2}
```

4-1 set 조작

- set 집합 연산
  - ✓ ^ 연산자는 대칭차집합을 구하며 함수는 symmetric_difference()을 사용한다.

```
print(set.symmetric_difference(s1, s2))
print(s1 ^ s2)
```

```
[결과]
{1, 2, 5, 6}
{1, 2, 5, 6}
```



set.symmetric_difference(s1, s2)

- set 할당 연산
  - ✓ 세트 자료형에 |, &, -, ^ 연산자와 할당 연산자 =을 함께 사용하면 집합 연산의 결과를 변수에 다시 할당한다.
  - ✓ |=은 현재 세트에 다른 세트를 더하며 update 메서드와 같다.

```
sd1 = {4,3,1,5}
sd1 |= {7}
print(sd1)
sd1.update({8})
print(sd1)
```

```
[결과]
{1, 3, 4, 5, 7}
{1, 3, 4, 5, 7, 8}
```

4-2 set 함수와 컴프리핸션

- set 함수
  - ✓ isdisjoint()는 현재 세트가 다른 세트와 겹치지 않는지 확인해 준다.

```
data1 = {1,2,3,4}
print(data1.isdisjoint({5,6,7,8}))
print(data1.isdisjoint({3,4,5,6}))
```

```
[결과]
True
False
```

  - ✓ add()는 요소를 추가할 때 사용한다.

```
data2 = {7,8,9}
data2.add(5)
print(data2)
```

```
[결과]
{8, 9, 5, 7}
```

  - ✓ remove()는 특정 요소를 삭제할 때 사용한다. 단 요소가 없으면 에러를 발생시킨다.
  - ✓ discard()도 remove와 마찬가지로 특정 요소를 삭제할 때 사용하지만 제거 할 요소가 없어도 에러를 발생시키지 않는다.

```
data3 = {10,30,50}
data3.remove(30)
print(data3)
data3.discard(30)
print(data3)
```

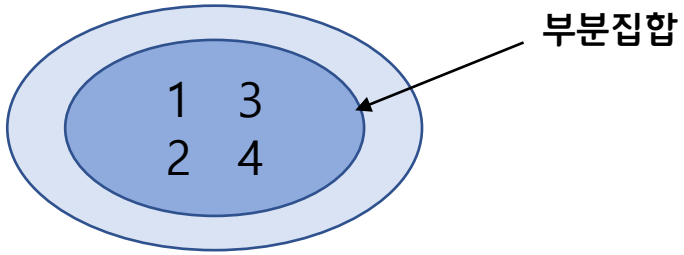
```
[결과]
{10, 50}
{10, 50}
```

4-2 set 함수와 컴프리핸션

- set 함수
  - ✓ set는 부분집합, 상위집합, 같이 집합 관계를 표현할 수도 있다.
  - ✓ <=은 현재 세트가 다른 세트의 부분집합인지 확인하며 함수는 issubset()를 사용할 수 있다.

```
data4 = {1,2,3,4}
print(data4<={1,2,3,4})
print(data4.issubset({1,2,3,4,5}))
```

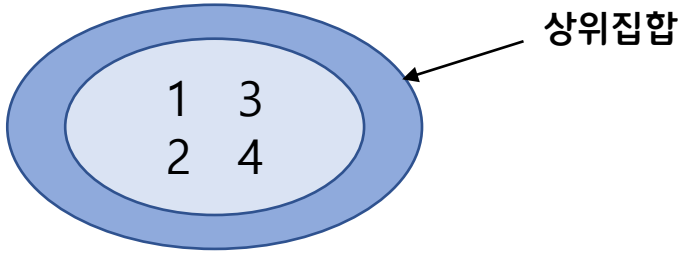
```
[결과]
True
True
```



  - ✓ >=은 현재 세트가 다른 세트의 상위집합(superset)인지 확인하며 함수는 issuperset()를 사용할 수 있다.

```
data5 = {1,2,3,4}
print(data4 >= {1,2,3})
print(data5.issuperset({1,2}))
```

```
[결과]
True
True
```



4-2 set 함수와 컴프리핸션

- set 컴프리핸션
  - ✓ set는 for 반복문과 if 조건문을 사용하여 set를 생성할 수 있다.

```
data6 = {i for i in 'hello'}
print(data6)
```

[결과]
{'o', 'e', 'h', 'l'}

```
data7 = {i for i in 'hello' if i not in 'el'}
print(data7)
```

[결과]
{'h', 'o'}