6. 집합 자료형

6.1. 집합 자료형을 만드는 법

- 집합 자료형을 쓰는 이유
  - 집합에 관련된 것을 쉽게 처리 하기 위해

```
S1 = set([1,2,3])
print(S1)
```

- set 키워드를 이용하여 만들 수 있다
- 비어있는 집합은 set() 로 만들 수 있다

```
S1 = set("PYTHON")
print(S1)
```

```
{'T', 'H', 'N', 'Y', 'O', 'P'}
```

- set 안에 문자열을 입력하여 만들 수도 있다

분명 Python 문자열로 set 자료형을 만들었는데 생성된 자료형에는 순서가 뒤죽박죽이다

6.2. 집합 자료형의 특징

- 중복을 허용하지 않는다
- 순서가 없다
- 중복을 허용하지 않는 특징때문에 데이터의 중복을 제거하기 위한 필터로 종종 사용한다
- 인덱싱을 통해 요솟값을 얻을 수 없다

```
S1 = set([1, 2, 3])
l1 = list(S1)
print(l1[0])
```

- set 자료형에 저장된 값을 인덱싱으로 접근하려면 리스트나 튜플로 변환한 후에 가능하다

```
S1 = set([1, 2, 3])
t1 = tuple(S1)
```

```
print(t1[3])
```

6.3. 교집합, 합집합, 차집합 구하기

교집합 구하기

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

- & 을 사용하면 교집합을 간단히 구할 수 있다

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

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

- intersection 함수를 사용해도 결과는 동일하다

합집합 구하기

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

- | 를 이용하여 구할 수 있다

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

print (s1.union(s2))
```

- union 함수를 사용해도 가능하다

차집합 구하기

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

```
print (s1 - s2)
```

- - 를 사용하면 차집합을 구할 수 있다

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

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

- difference 함수를 사용 해도 된다

6.4. 집합 자료형 관련 함수

값 1개 추가하기 (add)

```
s1 = set([1, 2])
s1.add(3)
print(s1)
```

값 여러 개 추가하기 (update)

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

특정 값 제거하기 (remove)

```
s1 = set([1, 2])
s1.remove(1)
print(s1)
```