Unit 26.

Status	완료
담당자	
마감일	
완료일	@2022년 11월 28일

Unit 26. 세트 사용하기

- 집합을 표현하는 세트 자료형
- 세트는 합집합, 교집합, 차집합 등의 연산이 가능

26.1 세트 만들기

- 세트는 { } 중괄호 안에 값을 저장하며 가 값은 ,(콤마)로 구분

```
>>> fruits = {'strawberry', 'grape', 'orange', 'pineapple', 'cherry'}
>>> fruits
{'pineapple', 'orange', 'grape', 'strawberry', 'cherry'}
```

- 세트는 요소의 순서가 정해져 있지 않다
- 세트에 들어가는 요소는 중복될 수 없다

```
>>> fruits = {'orange', 'orange', 'cherry'}
>>> fruits
{'cherry', 'orange'}
```

26.1.2 set을 사용하여 세트 만들기

• set(반복가능한객체)

- 중복된 문자는 표현되지 않는다

```
>>> a = set('apple')    # 유일한 문자만 세트로 만듦
>>> a
{'e', 'l', 'a', 'p'}
```

- 빈 세트 만들기

```
>>> c = set()
>>> c
set()
```

!!! 빈 세트를 만드려고 c = { } 와 같이 만들면 딕셔너리가 만들어지므로 주의

26.2 집합 연산 사용하기

1. | 연산자는 합집합(union)을 구하며 OR 연산자 |를 사용하고 set.union 메서드와 동작이 같다.

- **세트1 | 세트2**
- **set.union(세트1, 세트2)**

```
>>> a = {1, 2, 3, 4}
>>> b = {3, 4, 5, 6}
>>> a | b
{1, 2, 3, 4, 5, 6}
>>> set.union(a, b)
{1, 2, 3, 4, 5, 6}
```

2. & 연산자는 교집합(intersection)을 구하며 AND 연산자 &를 사용한다

- **세트1 & 세트2**
- **set.intersection(세트1, 세트2)**

```
>>> a & b
{3, 4}
>>> set.intersection(a, b)
{3, 4}
```

3. - 연산자는 차집합(difference)을 구하며 뺄셈 연산자 -를 사용한다

- **세트1 - 세트2**
- **set.difference(세트1, 세트2)**

```
>>> a - b
{1, 2}
>>> set.difference(a, b)
{1, 2}
```

4. ^ 연산자는 대칭차집합(symmetric difference)을 구하며 XOR 연산자 ^를 사용하고 set.symmetric_difference 메서드와 동작이 같다.

- **세트1 ^ 세트2**
- **set.symmetric_difference(세트1, 세트2)**

```
>>> a ^ b
{1, 2, 5, 6}
>>> set.symmetric_difference(a, b)
{1, 2, 5, 6}
```

26.2.1 집합 연산 후 할당 연산자 사용하기

- 세트 자료형에 |, &, -, ^ 연산자와 할당 연산자 =을 함께 사용하면 집합 연산의 결과를 변수에 다시 저장(할당)한다.
- **세트1 |= 세트2**
- **세트1.update(세트2)**

```
>>> a = {1, 2, 3, 4}
>>> a |= {5}
>>> a
{1, 2, 3, 4, 5}
>>> a = {1, 2, 3, 4}
>>> a.update({5})
>>> a
{1, 2, 3, 4, 5}
```

- **세트1 &= 세트2**
- **세트1.intersection_update(세트2)**

```
>>> a = {1, 2, 3, 4}
>>> a &= {0, 1, 2, 3, 4}
>>> a
{1, 2, 3, 4}
>>> a = {1, 2, 3, 4}
>>> a.intersection_update({0, 1, 2, 3, 4})
>>> a
{1, 2, 3, 4}
```

- **세트1 -= 세트2**
- **세트1.difference_update(세트2)**

```
>>> a = {1, 2, 3, 4}
>>> a -= {3}
>>> a
```

```
{1, 2, 4}
>>> a = {1, 2, 3, 4}
>>> a.difference_update({3})
>>> a
{1, 2, 4}
```

- **세트1 ^= 세트2**
- **세트1.symmetric_difference_update(세트2)**

```
>>> a = {1, 2, 3, 4}
>>> a ^= {3, 4, 5, 6}
>>> a
{1, 2, 5, 6}
>>> a = {1, 2, 3, 4}
>>> a.symmetric_difference_update({3, 4, 5, 6})
>>> a
{1, 2, 5, 6}
```

26.2.2 부분 집합과 상위 집합 확인하기

- 세트는 부분집합, 상위집합, 진상위집합과 같이 속하는 관계를 표현할 수 있다.
- <=은 현재 세트가 다른 세트의 부분집합(subset)인지 확인하며 issubset 메서드와 같다.
- **현재세트 <= 다른세트**
- **현재세트.issubset(다른세트)**

```
>>> a = {1, 2, 3, 4}
>>> a <= {1, 2, 3, 4}
True
>>> a.issubset({1, 2, 3, 4, 5})
True
```

- <은 현재 세트가 다른 세트의 진부분집합(proper subset)인지 확인하며 메서드는 없다
- 진부분집합은 부분집합이지만 같지는 않을 때 참
- **현재세트 < 다른세트**

```
>>> a = {1, 2, 3, 4}
>>> a < {1, 2, 3, 4, 5}
True
```

- >=은 현재 세트가 다른 세트의 상위집합(superset)인지 확인하며 issuperset 메서드와 같다
- **현재세트 >= 다른세트**
- **현재세트.issuperset(다른세트)**

```
>>> a = {1, 2, 3, 4}
>>> a >= {1, 2, 3, 4}
True
>>> a.issuperset({1, 2, 3, 4})
True
```

- >은 현재 세트가 다른 세트의 진상위집합(proper superset)인지 확인하며 메서드는 없다

• 현재세트 > 다른세트

```
>>> a = {1, 2, 3, 4}
>>> a > {1, 2, 3}
True
```

26.2.3 세트가 같은지 확인하기

- 세트는 = 연산자를 사용해서 같은지 확인 가능
- ! = 연산자는 세트가 다른지 확인

- 세트는 요소 순서가 정해져 있지 않으므로 요소만 같으면 된다

26.2.4 세트가 겹치지 않는지 확인하기

- disjoint는 현재 세트가 다른 세트와 겹치지 않는지 확인한다.
- **현재세트.isdisjoint(다른세트)**

26.3 세트 조작하기

- add는 세트에 요소를 추가한다
- remove는 세트에서 특정 요소를 제거하고 요소가 없으면 에러
- discard는 세트에서 특정 요소를 제거하고 요소가 없으면 그냥 넘어간다
- pop( )은 세트에서 임의의 요소를 삭제하고 해당 요소를 반환 요소가 없다면 에러
- cleat( )는 세트의 모든 요소를 삭제
- len(세트)는 세트의 요소 개수(길이)를 구한다