Set, 집합

집합에 관련된 데이터들을 쉽게 처리하기 위해 만들어진 자료형입니다.

사용법은 2가지로 볼수 있습니다.

- set()
- {}

집합은 딕셔너리와는 다르게 key값이 없고 값만 들어갑니다.

1. 집합의 특징

1. 중복을 허용하지 않습니다.
2. 순서가 없습니다. -> 인덱싱을 지원하지 않습니다.
3. 수정 가능 객체(mutable)입니다.

집합은 중복을 허용하지 않기에 중복을 제거하기 위해 iterable객체를 집합으로 변환한 후, 재변환을 하여 주로 사용하기도 합니다.

```
# 집합 할당
set_data = {1, 2, 3}
print(type(set_data)) # <class 'set'>
print(set_data) # {1, 2, 3}

# 문자열 집합 변환
set_data = set("hello")
print(set_data) # {'l', 'h', 'e', 'o'}

# in 연산자 사용
print('o' in set_data) # True
print('w' in set_data) # False
```

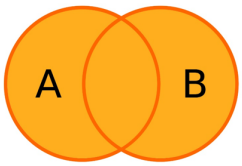
주의]

> **집합 내부 원소는 mutable 할 수 없습니다.**
> 집합 Set 내부 원소는 mutable할 경우 Error가 발생합니다.

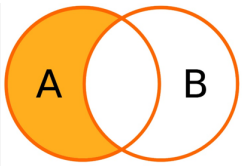
```
# Mutable한 List
set_data = {1, 2, 3, [1,2,3]} # TypeError: unhashable type: 'list'
```

```
# Mutable한 Dictionary
set_data = {1, 2, 3, {"key": "value"}} # TypeError: unhashable type: 'dict'
```

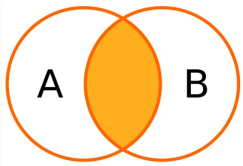
2. 집합 연산



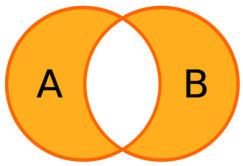
합집합



차집합



교집합



대칭차집합

- 1) 합집합

```
# `|`
print( {1, 2, 3} | {3, 4, 5} ) # {1, 2, 3, 4, 5}

# `union 내장함수`
print( {1, 2, 3}.union({3, 4, 5}) ) # {1, 2, 3, 4, 5}
```

- 2) 차집합

```
# `-`
print( {1, 2, 3} - {3, 4, 5} ) # {1, 2}

# `difference 내장함수`
print( {1, 2, 3}.difference({3, 4, 5}) ) # {1, 2}
```

- 3) 교집합

```
# `&`
print( {1, 2, 3} & {3, 4 ,5} ) # {3}

# `intersection 내장함수`
print( {1, 2, 3}.intersection({3, 4, 5})) # {3}
```

- 4) 대칭차집합

대칭차집합 = 합집합 - 교집합

```
# `^`
print( {1, 2, 3} ^ {3, 4 ,5} ) # {1, 2, 4, 5}

# `symmetric_difference 내장함수`
print( {1, 2, 3}.symmetric_difference({3, 4, 5}) ) # {1, 2, 4, 5}
```

3. 집합 관련 함수들

- 1) add, 추가하기

```
set_data = set() # 빈 집합 할당

set_data.add(1)
set_data.add('3')

print(set_data) # {1, '3'}
```

- 2) update, 여러 개 추가하기

```
set_data = {1, '3'}
add_set_data = set([2, '3', 1, 5])

set_data.update(add_set_data)

print(set_data) # {1, 2, '3', 5}
```

- 3) remove, 특정 값 제거하기

```
set_data = {1, 2, '3', 5}

set_data.remove('3')

print(set_data) # {1, 2, 5}
```

- 4) issubset

부분집합 여부 확인

`집합1.issubset(집합2)`으로 사용합니다.

여기서 집합1이 집합2의 부분집합이라면 `True`, 아니라면 `False`를 반환합니다.

```
large_set = {1, 2, 3, 4, 5}
small_set = {1, 2, 4}

print(large_set.issubset(small_set)) # False
print(small_set.issubset(large_set)) # True
```

- 5) issuperset

issubset과는 반대의 의미를 가집니다.

`집합1.issuperset(집합2)`로 사용합니다.

여기서 집합1이 집합2를 포함하고 있는 관계라면(=집합2가 집합1의 부분집합) `True`, 아니라면 `False`를 반환합니다.

```
large_set = {1, 2, 3, 4 ,5}
small_set = {1, 2, 4}

print(large_set.issuperset(small_set)) # True
print(small_set.issuperset(large_set)) # False
```

- 6) isdisjoint

교집합이 없다면 `True`, 있다면 `False`를 반환합니다.

```
set_a_data = {1, 2, 3, 4 ,5}
set_b_data = {4, 7, 9}
set_c_data = {7, 9}

print(set_a_data.isdisjoint(set_b_data)) # False
print(set_b_data.isdisjoint(set_a_data)) # False

print(set_a_data.isdisjoint(set_c_data)) # True
print(set_c_data.isdisjoint(set_a_data)) # True
```