Set

- 데이터를 저장하는 추상자료형(ADT)
- 순서를 보장하지 않음
- 데이터 중복을 허용하지 않음
- 데이터 조회(search)가 List보다 빠름

💡 Set은 언제 쓰는 것이 좋을까 ?

- 중복된 데이터를 제거해야 할 때
- 데이터의 존재 여부를 확인해야 할 때

📌 Set 구현체

💡 hash set

- hash table을 사용하여 구현
  - 일반적으로 테이블의 크기에 상관없이 key를 통해 상수 시간에 빠르게 데이터에 접근 가능

✔ Java에서 hash set 구현체 : HashSet

```
// Constructs a new, empty set; the backing HashMap instance has default
initial capacity (16) and load factor (0.75).
public HashSet() {
        map = new HashMap<>();
}

// add 메서드에서는 key 값에 데이터를 넣고, value 에 더미데이터를 넣는 방식
public boolean add(E e) {
        return map.put(e, PRESENT) == null;
}

// Dummy value to associate with an Object in the backing Map
private static final Object PRESENT = new Object();

public boolean contains(Object o) {
    return map.containsKey(o);
}
```

✔ Java에서 HashSet 사용 예제

```
Set<String> unique = new HashSet<>();
unique.add("구독");
unique.add("좋아요");
unique.add("구독");

System.out.println(unique.size()); // 2
System.out.println(unique.contains("구독")); // true
System.out.println(unique.contains("알림설정")); // false
```

✔ CPython에서 hash set 구현체 : Set

- dictionary가 있지만 dictionary를 그대로 사용하지 않음.
- hash table을 사용하는 것은 dictionary 와 동일하지만 최적화된 형태로 Set 만의 자체 구현을 가지고 있음.

```
typedef struct {
        PyObject_HEAD
Py_ssize_t fill;
Py_ssize_t used;
Py_ssize_t mask;
setentry *table;
Py_hash_t hash;
Py_ssize_t finger;

setentry smalltable[PySet_MINSIZE]; // default size 만큼의 hash table을 만듦
PyObject *weakreflist;
} PySetObject;

#define PySet_MINSIZE 8

typedef struct {
        PyObejct *key; // 내부적으로 key 저장. value 는 저장하지 않음.
        Py_hash_t hash;
} setentry;
```

✔ Python 에서 set 사용 예제 1

```
unique = set()
unique.add("쉬운코드")
```

```
unique.add("쉽다")
unique.add("쉬운코드")

print(len(unique)) # 2
print("쉬운코드" in unique) # True
print("고고씽" in unique) # False
```

✔ Python에서 set 사용 예제 2

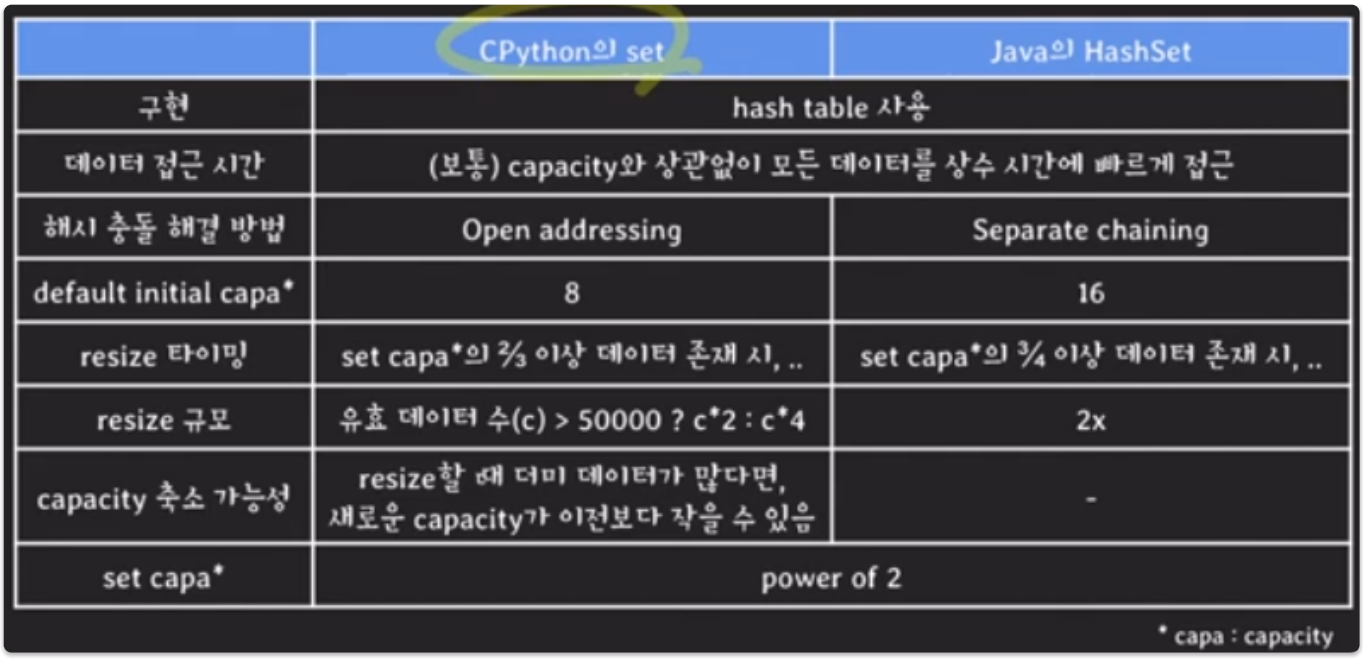
```
a = {1, 2, 3}
b = {3, 6, 9}

print(a - b) # {1, 2}
print(b - a) # {6, 9}
print(a & b) # {3}
print(a | b) # {1, 2, 3, 6, 9}
print(a ^ b) # {1, 2, 6, 9}
```

💡 linked hash set (java)

- 유효한 데이터들끼리 linked list로 연결을 시켜줘서 iteration을 할 때 좀 더 빠르게 순회할 수 있음.
- linked list를 추가적으로 구현해서 사용하기 때문에 java의 hash set 보다 퍼포먼스 적인 손해가 있고 메모리 사용 비용이 더 증가한다는 단점이 있음.

💡 tree set (java)

💡 CPython의 Set 과 Java의 HastSet 비교



	CPython의 set	Java의 HashSet
구현	hash table 사용	
데이터 접근 시간	(보통) capacity와 상관없이 모든 데이터를 상수 시간에 빠르게 접근	
해시 충돌 해결 방법	Open addressing	Separate chaining
default initial capa*	8	16
resize 타이밍	set capa*의 ⅔ 이상 데이터 존재 시, ..	set capa*의 ¾ 이상 데이터 존재 시, ..
resize 규모	유효 데이터 수(c) > 50000 ? c*2 : c*4	2x
capacity 축소 가능성	resize할 때 더미 데이터가 많다면, 새로운 capacity가 이전보다 작을 수 있음	-
set capa*	power of 2	

* capa : capacity

💡 List 와 Set 중 무엇을 쓸까 ?

- Set을 사용하는게 더 적절한 상황이 아니라면, 거의 대부분 List를 사용.
- 데이터들 자체에 이미 중복이 없고, 순서 상관없이 iteration(loop를 돌면서 한 번씩 접근) 목적으로만 저장하려고 한다면 List와 Set 중 아무거나 사용해도 괜찮을까 ?

```
for (String name:names) {
        // name으로 뭔가를 수행
}
```

  - list가 메모리도 적게 쓰고, 구현 특성 상 list가 단순하여 iteration이 더 빠르기 때문에 이 경우에도 list를, 특히 array list를 쓰는 것을 추천.

- set는 비어있는 공간도 확인하면서 iteration을 하기 때문에 추가적인 오버헤드 발생.

	List	Set
구현체	array list, linked list	hash set, linked hash set, tree set
중복 데이터 저장	가능	불가능
데이터 조회 속도	상대적으로 느림	상대적으로 (매우) 빠름
순서(ordering) 존재	순서 무조건 있음	구현체에 따라 다름 (hash set은 불가능)
메모리	덜 씀	더 씀

재생(k)