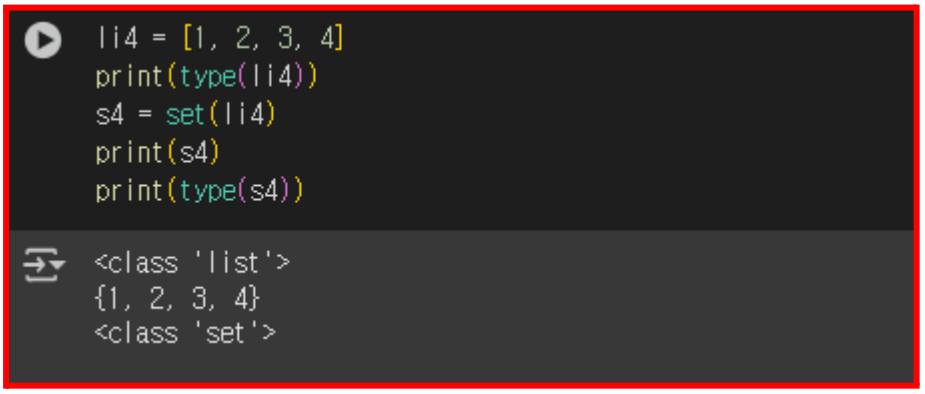
Day 5 - Python Collection Type (Set)

Set에 아무것도 없으면 Dict 타입으로 인식한다:

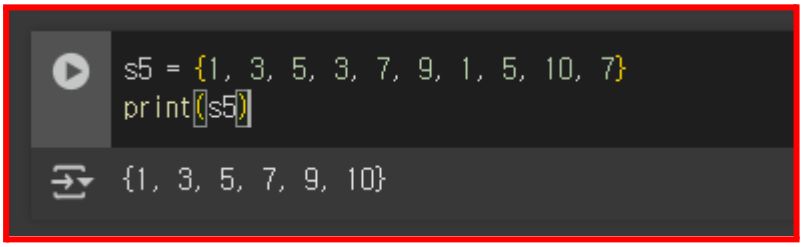
- 안에 있는 자료는 변경불가하다

리스트 → Set 형변환:



- Set() 함수를 이용한다.

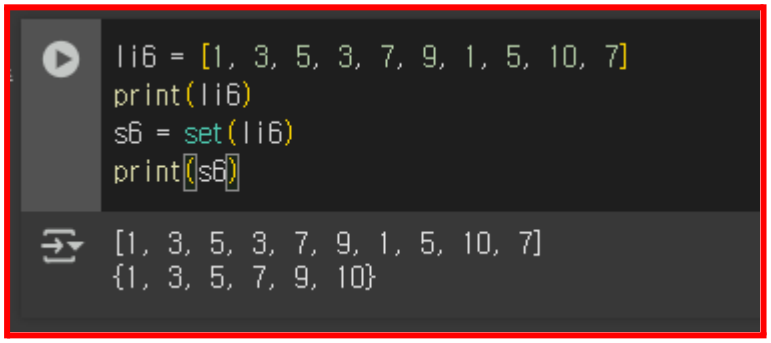
Set 형태는 중복된 값을 없애버린다:



```
s5 = {1, 3, 5, 3, 7, 9, 1, 5, 10, 7}
print(s5)
```

```
{1, 3, 5, 7, 9, 10}
```

리스트에 중복된 값이 있을경우:

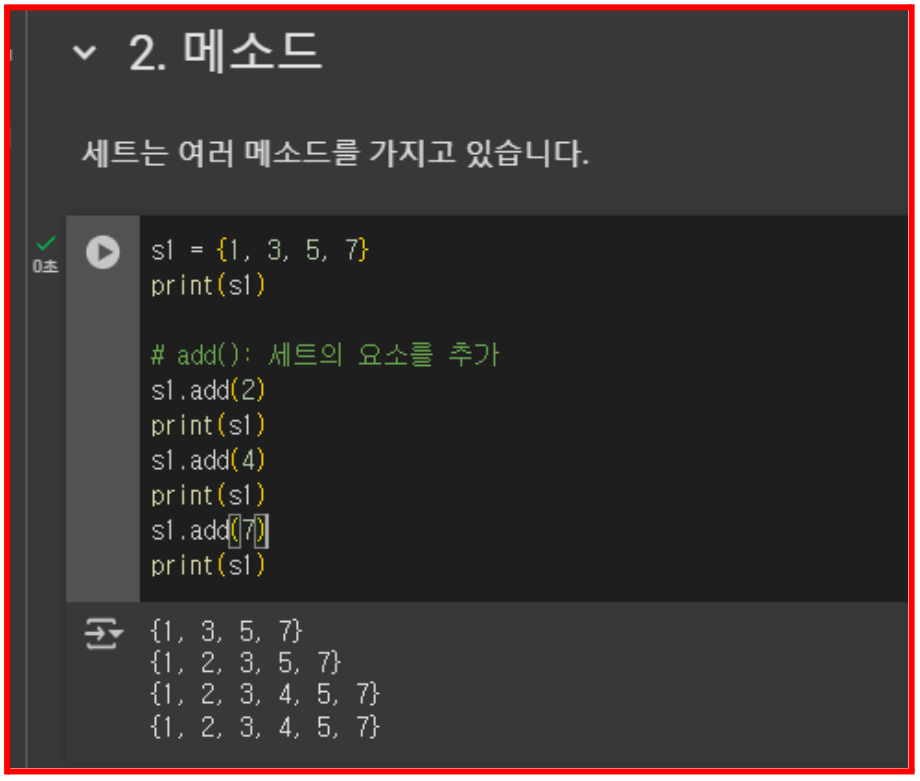


```
li6 = [1, 3, 5, 3, 7, 9, 1, 5, 10, 7]
print(li6)
s6 = set(li6)
print(s6)
```

```
[1, 3, 5, 3, 7, 9, 1, 5, 10, 7]
{1, 3, 5, 7, 9, 10}
```

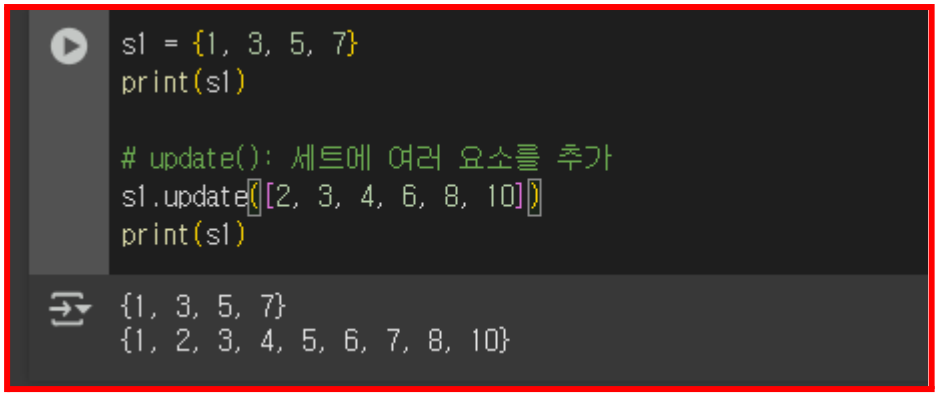
- set 형태로 변환시켜서 중복된 값을 없애버릴 수 있다.

Set 메서드의 특징 add()



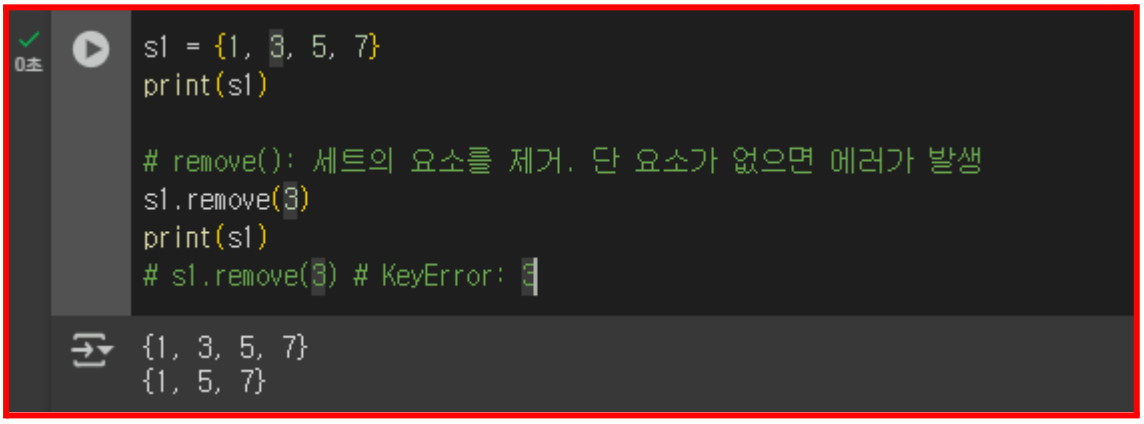
- 인덱싱 불가
- 데이터를 넣을때 무작위로 들어감
- 순서가 없음.
- 정렬된 순서로 나오는것같지만 아니다.

update() 메서드



- 여러 요소를 추가할 수 있다.

remove() 메서드



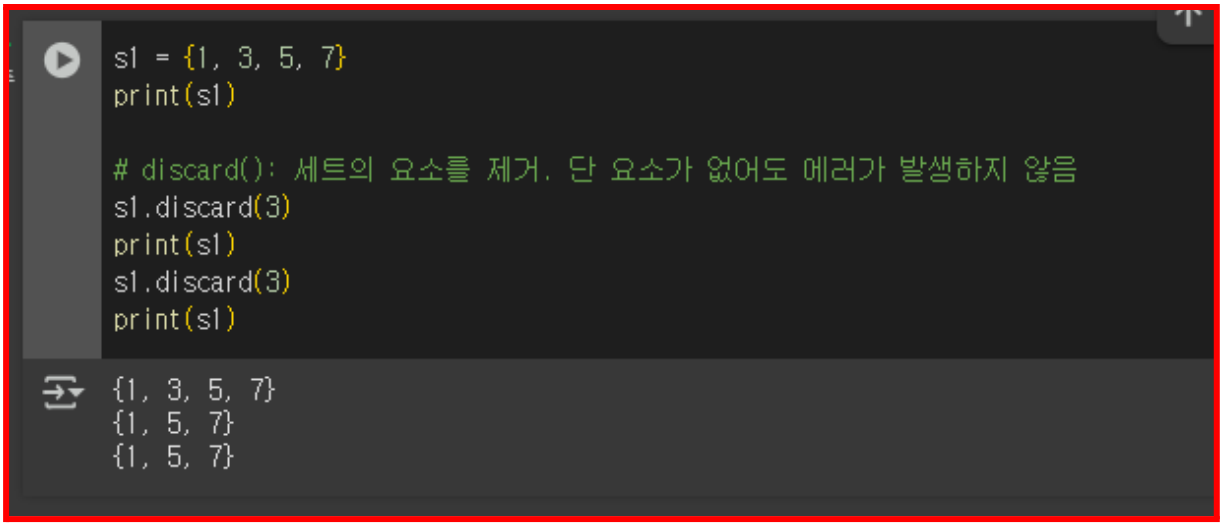
```
s1 = {1, 3, 5, 7}
print(s1)

# remove(): 세트의 요소를 제거. 단 요소가 없으면 에러가 발생
s1.remove(3)
print(s1)
# s1.remove(3) # KeyError: 3
```

```
{1, 3, 5, 7}
{1, 5, 7}
```

- remove(값) 으로 그 값을 set 안에서 지울 수 있다.
- 요소가 없으면 에러가 발생한다.

discard() 메서드



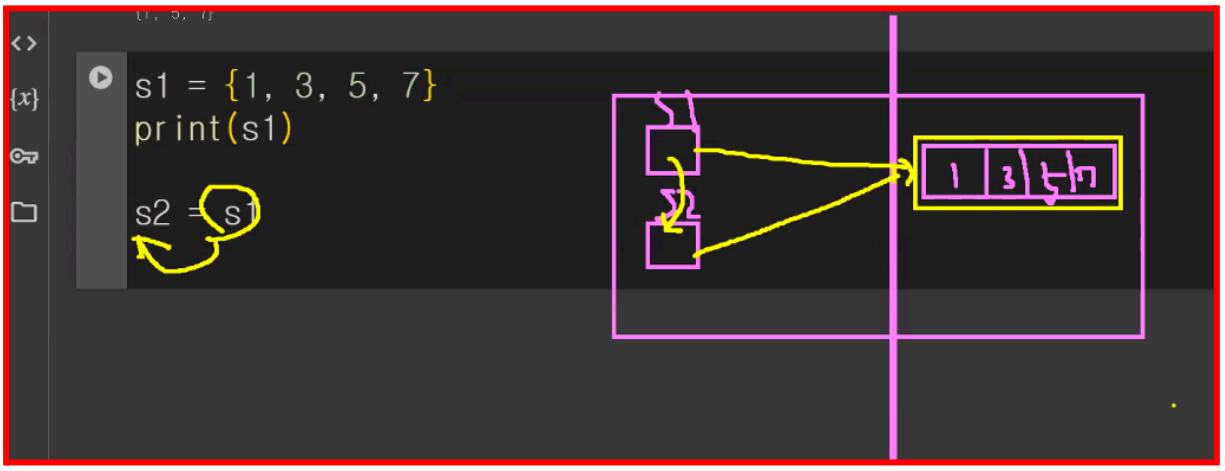
```
s1 = {1, 3, 5, 7}
print(s1)

# discard(): 세트의 요소를 제거. 단 요소가 없어도 에러가 발생하지 않음
s1.discard(3)
print(s1)
s1.discard(3)
print(s1)
```

```
{1, 3, 5, 7}
{1, 5, 7}
{1, 5, 7}
```

- discard(값)으로 set 안에서 지울 수 있다.
- 요소가 없어도 에러가 발생하지 않는다.

set의 메모리 주소, 값 복사



- 메모리주소를 복사하면 같은 주소값을 가진다.

```
s1 = {1, 3, 5, 7}
print(s1)

s2 = s1 # 메모리 주소를 복사
print(id(s1))
print(id(s2))

# copy(): 세트를 복사
s2 = s1.copy() # 값을 복사
print(s1)
print(s2)
print(id(s1))
print(id(s2))
```

```
{1, 3, 5, 7}
139422361974400
139422361974400
{1, 3, 5, 7}
{1, 3, 5, 7}
139422361974400
139421424467552
```

- 같은 값이지만 메모리 주소 두개가 다르다.

Set 합집합: union()

```
s3 = {10, 20, 30, 40, 50}
s4 = {30, 40, 50, 60, 70}

# union(): 합집합을 계산하여 반환
result1 = s3.union(s4)
print(result1)
```

```
{70, 40, 10, 50, 20, 60, 30}
```

- union() 메서드로 합집합을 구할 수 있다.

연산자로 합집합 구하는법

```
result2 = s3 | s4
print(result2)
```

```
{70, 40, 10, 50, 20, 60, 30}
{70, 40, 10, 50, 20, 60, 30}
```

Set 교집합: intersection()

```
s3 = {10, 20, 30, 40, 50}
s4 = {30, 40, 50, 60, 70}

# intersection() : 교집합을 계산하여 반환
result1 = s3.intersection(s4)
print(result1)

result2 = s3 & s4
print(result2)
```

```
{40, 50, 30}
{40, 50, 30}
```

- 연산자로도 구할 수 있다.

Set 차집합 계산: difference()

```
s3 = {10, 20, 30, 40, 50}
s4 = {30, 40, 50, 60, 70}

# difference(): 차집합을 계산하여 반환

result1 = s3.difference(s4)
print(result1)

result2 = s3 - s4
print(result2)
```

```
{10, 20}
{10, 20}
```

- 이것 또한 연산자로 구할 수 있다.

Set 대칭 차집합 계산: symmetric_difference()

```
s3 = {10, 20, 30, 40, 50}
s4 = {30, 40, 50, 60, 70}

# symmetric_difference(): 대칭 차집합을 계산하여 반환
result1 = s3.symmetric_difference(s4)
print(result1)

result2 = s3 ^ s4
print(result2)
```

```
{20, 70, 10, 60}
{20, 70, 10, 60}
```

- 연산자 계산 가능