Sets: { }

collection of data.

not ordered → no indexing, slicing. No repetition / duplicate stored.

only similar to list & dict except we use { } & no keys exist only values.

set_a = { value1, value2 ... }

set_a[0] ✗ , set[value1] ✗ , set_a[0:3] ✗

Check membership-

value in set_a → returns True (exist) else False (not-exist)

Looping -

```
for i in set_a:
    print(i)
```

⇒ prints all elements in any order.

length -

```
len(set_a)
```

Adding element -

```
set_a.add(value)
```

set_b = new set with some common elements from set_a & some new elements of (its own)

set_a.update(set_b) → Add set_b element to a which are non-existing in a set_a.

Removing element -

```
set_a.remove('existing_value')
```

↳ if non-existing value → error.

```
set_a.discard(existing_value / non-existing value)
```

↳ removes the element ↳ no error

set_a.pop() → returns any one element & prints it.

## Sets functions:

### (1) Intersection:

set_a.intersection(set_b) → returns common elements

### (2) Union:

set_a.union(set_b) → combines both but no duplicates

### (3) Difference:

set_a.difference(set_b) → elements not in b

set_b.difference(set_a) → " " " a

### (4) Symmetric - difference = union - intersection.

set_a.symmetric_difference(set_b) → ignores common elements & prints rest.

(5) Intersection update

set_a.intersection_update(set_b) → changes set_a to its. b/w set_a & set_b.

(6) ~~set_a~~ set_a.difference_update(set_b)

(7) set_a.symmetric_difference_update(set_b)

(8) set_a.issubset(set_b) → if all elements (set_b) in set_a ⇒ returns true
else ⇒ false

(9) set_a.isdisjoint(set_b) → if no common ⇒ true
else ⇒ false

Empty set
s = { } → it's a dictionary ×
s = set() → empty set ✓

set() → creates an empty set.