Set

- Unordered coll. of elements with same / diff. data types.
- Mutable, no duplicate values.
- Indexing & slicing are not performed.
- { }.

Empty set can be def. using set() fun.

```
s = set()
print(s) ⟶ set()
type(s) ⟶ set.
```

Methods

① add():

- Adds an elem. at random position.

ex: s = {1, 2, 3, 4, 5}.

```
s.add(7) ⟶ {1, 2, 7, 3, 4, 5}
s.add(3.4) ⟶ {1, 3.4, 2, 7, 3, 4, 5}.
```

Note: This has no effect if that ele. is already present.

② union() (1).

performs set union. can take multiple parameters.

ex:: S1 = {1, 2, 11, 14, 25}
S2 = {4, 5, 7, 11}
S3 = {55, 79, 8, 10}.

S4 = S1.union(S2).
S5 = S2 | S3.

o/p::
S4 = {1, 2, 4, 5, 7, 11, 14, 25}.
S5 = {4, 5, 7, 8, 10, 11, 55, 79}.

③ update() : Updates a set with union of itself & others. can take multiple sets as arguments. It changes the original set onto which the update oper. has been applied.

ex:: S1 = {1, 2, 11, 14, 25} S2 = {4, 5, 17, 11}
S3 = {55, 79, 8, 10}.

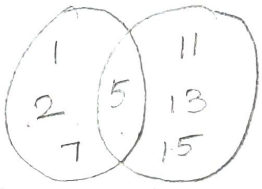
print(S1) // {1, 2, 11, 14, 25}
print(S1.update(S2)).
//{1, 2, 4, 5, 11, 14, 17, 25}.

5) intersection() : (&)

- Returns the intersection of two sets as a new set.
- can take multiple sets as arguments.

ex : S1 = {1, 2, 5, 7}
S2 = {5, 11, 13, 15}
S3 = {1, 6, 13, 12}

S4 = S1.intersection(S2)
S5 = S1 & S3.



S1 ∩ S2 = {5}

o/p :
S4 = {5}.
S5 = {1}.

5) intersection_update() :

- same as intersection.
- But it will change the original set onto which the update op. is performed.

ex : S1 = {1, 2, 5, 7} S2 = {5, 11, 13, 15}
S3 = {1, 6, 13, 12}.

print(S1) // {1, 2, 5, 7}
S1 = S1.intersection_update(S2)
print(S1) {5}.

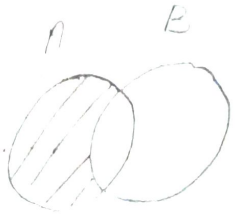
6) Set Difference.

- Returns the difference of 2/ more set as a new set.

ex: a = {1, 2, 3}
b = {3, 7, 2, 6}.

a.difference(b) ⟶ {1}.

print (a - b) ⟶ {1}.



(A B)

A - B = {1}

7) Set difference_update():

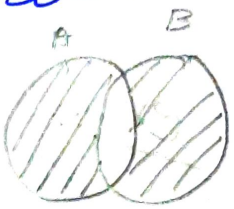
- Removes all the ele. of another set from this set & update the original set.

ex: a = {1, 2, 3}
b = {3, 7, 2, 6}

print(a) ⟶ {1, 2, 3}.
s1 = a.difference_update(b)
print(s1) ⟶ {1, 2, 3}
print(a.difference_update(b))
⟶ {1}.

8. symmetric_difference :(^)

Return set contains a mix of both items that are not present in both sets.



ex :- S1 = {12, 14, 17, 19, 22}
S2 = {14, 15, 16, 17, 18}

S1.symmetric_difference(S2)

S1 ^ S2.

O/P :- {16, 18, 19, 22, 12, 15}
{16, 18, 19, 22, 12, 15}

9. symmetric_difference_update:

same as symmetic diff.
also update the original set.

ex :- S1 = {12, 14, 17, 19, 22}
S2 = {14, 15, 16, 17, 18}

S1.symmetric_difference_update(S2)

op :-

{12, 15, 16, 18, 19, 22}.

10. remove() : Removes an element from a set. If it's not there, then error.

ex : s1 = {1, 2, 3, 4}
s1. remove (1) → {2, 3, 4}
s1. remove (5) → key error.

11. pop() : Randomly removes an item from a set & returns the removed item.

ex : s1 = {1, 2, 3, 4, 5}
s1. pop() → {1, 2, 4, 5}.

12. clear() : Empties the set.

ex : s1 = {1, 2, 3, 4}
s1. clear() → set()

13. discard() : Remove an element from a set. Doesn't gives an error, if the element is not a member.

ex : s1 = {1, 2, 3, 4}
s1. discard(1) → {2, 3, 4}.
s1. discard(5) → #no error.
{2, 3, 4}.

14. isdisjoint()

If two sets don't have any common items → True

else False.

ex: A = {1, 2, 3, }

B = {4, 5, 6}

Print (A.isdisjoint(B))

o/p: True.

15. issubset()

If set(A) (ie., all elements of set A) are present in set(B) → True

Else False.

ex: A = {1, 2, 3}

B = {1, 4, 5, 6}

print(A, issubset(B)) → False.

16. issuperset()

set X is said to be the superset of set Y, if all elements of Y are in X.

ex: A = {1, 2, 3, 4, 5}

B = {1, 2, 3}

C = {1, 2, 3}

A.issuperset(B) → True.
(All ele. of B are in A).

B.issuperset(A) → False.

B.issubset(A) → True.
(All ele. of B are in A).

Built-in Functions in set

1. all(): Returns true if all elements in the gn. iterable are true else False.

ex: a = ['True', 'True']

all(a) → True.

Truth table:

I/P	O/P
All values are True	True.
All values are False	False.
one value is True (other False)	False.
one (False) other (True)	False
Empty iterable	True.

all() in Lists :

L = [1, 3, 4, 5] → True.

L = [0, False] → False.

L = [1, 3, 4, 0] → False

L = [0, False, 5] → False.

L = [ ] → True.

all() in strings :

S = "Hello" → True

S = '000' → True

S = ' ' → True.

all() in dictionaries :

S = { 0 : 'False', 1 : 'False' } → False

S = { 1 : 'False', 1 : False } → True

S = { 1 : 'True', False : 0 } → False

S = { } → True.

S = { '0' : 'True' } → True.

Note :- If all keys are True / Dic is empty → True

else → False.

0 is False, '0' is True.

ii) any() :-

Returns True if any element of an iterable is True. else False.

ex :- a = ['True', 'False']

o/p :- True.

Truth Table :-

I/P	o/p
All values True	True
All values False	False.
One (True) others (False)	True.
one (False) others (True)	True.
empty iterable	False.

any() in lists :-

L = [1, 3, 4, 0] → True

L = [0, False] → False

L = [0, False, 5] → True

L = [ ] → False.

any() in String :-

L = "Hello" → True

L = '00' → True

L = [] → False.

any() in Dictionary :-

d = {0: 'False'} → False

d = {0: 'False', 1: 'True'} → True

d = {} → False

d = {'0': 'False'} → True.

Note :-

If all keys / dic. is empty are false → False

If atleast one key is True → True.

3. Enumerate() :-

- adds a counter to an iterable & returns it.

Ex :-

l = ['python', 'c', 'c++']

e = enumerate(l)

O/P :- [(0, 'python'), (1, 'c'), (2, 'c++')]

ex:

```
l = ['python', 'c', 'c++']
e = enumerate(l, 10)
```

o/p:

```
[(10, 'python'), (11, 'c'), (12, 'c++')]
```

4. len() : Returns the no. of items in the obj.

ex: l = ['python', 'c', 'c++'].

print(len(l)) → 3.

5. max() : Returns the largest item in an iterable.

ex: l = [9, 34, 11, -4, 27]

max(l) → 34.

6. min() : Returns the minimum item in an iterable.

ex: l = [9, 34, 11, -4, 27]

min(l) → -4.

7. sorted() : Sort in Asc/Desc. order.

ex:

```
n = [4, 2, 12, 8]
sorted(n) → [2, 4, 8, 12]
sorted(n, reverse = True) → [12, 8, 4, 2]
```

8. sum()

Adds the items of an iterable & Returns sum.

ex: m = [65, 71, 68, 74, 61]

sum(m) → 339.

Frozen set: (Tuple + set = Frozenset)

- It is same as set but it is immutable, no duplicate values

syntax:

frozenset([iterable])

- If we pass a list/set/tuple to frozenset(), it returns a frozenset created.
- If nothing is passed, then it returns empty set.

a) Frozen set() as list

a = [2, 4, 6, 8]

frozenset(a)

f'Return value: {result}')

↳ ({8, 2, 4, 6})

b.) Frozenset() from set

a = {2, 4, 6, 8}

frozenset (a) ⟶ ({8, 2, 4, 6})

c.) frozenset() from Tuple

a = (2, 4, 6, 8)

frozenset (a) ⟶ ({8, 2, 4, 6})

d.) Empty frozenset()

a = ()

frozenset (a) ⟶ frozenset()

e.) frozenset() from dictionary

a = {"name": 'agi', 'age': 25}

frozenset (a) ⟶ ({'name', 'age'})

Operations in frozenset

1. Len
2. in
3. not in
4. isdisjoint
5. issuperset
6. is subset
7. union
8. Intersection.
9. difference
10. symmetric-diff.