Python!

By- Mugdha Panhale,

Finitely Infinite Systems Pvt. Ltd.

Sets

What is Set?

1. Unordered collection of items
2. Every item is unique
3. Every element must be immutable
4. A set is mutable
5. Can be used to perform mathematical set operations

Creating a Set

```
my_set = {1, 2, 3, 4}
type(my_set)
o/p : Set
```

Sets

```
my_set_1 {9.2, "hello", (1, 2, 3,4)}
my_set_2 = {1, 1, 2, 2, 3, 4, 5, 5}
my_set_2
o/p : {1, 2, 3, 4, 5}
```

Creating Set using set() function
Providing list to set function in the below example

```
my_set_3 set([1, 1, 2, 2, 3, 3])
my_set_3
o/p : {1, 2, 3}
```

Sets

```
my_set_4 = {1, 2, [3, 4]}
o/p : TypeError: unhashable type: 'list'

my_set_5 set()
type (my_set_5)
o/p : set
```

Sets

Modifying a Set

```
my_set = {1, 2, 3}
my_set[0]
o/p : TypeError: 'set' object is not subscriptable
```

Using add method

```
my_set.add(4)
my_set
o/p : {1, 2, 3, 4}
```

Using update method

```
my_set.update([5, 6, 7])
my_set
o/p : {1, 2, 3, 4, 5, 6, 7}
```

Sets

Modifying a Set

```
my_set.update([8, 9], {1, 4, 10})
my_set
o/p : {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

Sets

Removing elements from Sets

Using discard method

```
my_set = {1, 2, 3, 4, 5}
my_set.discard (5)
my_set
o/p : {1, 2, 3, 4}
```

Using remove method

```
my_set.remove(4)
my_set
o/p : {1, 2, 3}
my_set.discard (6)
my_set
o/p : {1, 2, 3}
```

Sets

Removing elements from Sets

my_set.remove(6)

o/p : KeyError: 6

pop method: poping random element.

my_set = set("Python")

my_set

o/p : {'P', 'h', 'n', 'o', 't', 'y'}

my_set.pop()

o/p : 't’

my_set.pop()

o/p : ‘h’

my_set.pop()

o/p : ‘y’

my_set.pop()

o/p : ‘o'

Sets

Removing elements from Sets

clear method for deleting all the items.

```
my_set.clear()
my_set
```

o/p : set()

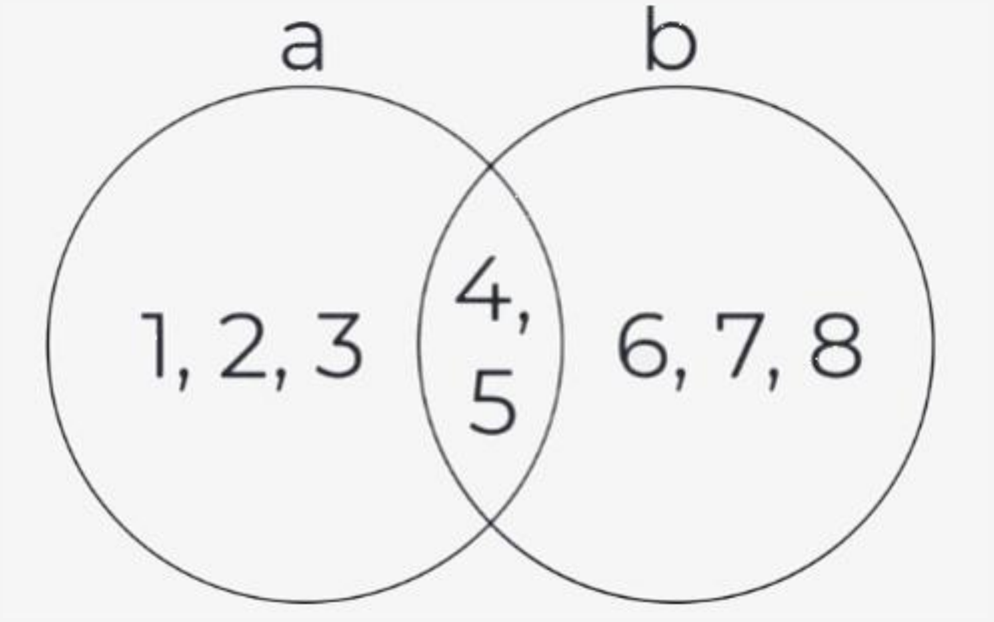
del keyword

```
del my_set
my_set
```

o/p: NameError: name 'my_set' is not defined

Sets

Set Operations - Union
a = {1, 2, 3, 4, 5}
b = {4, 5, 6, 7, 8}



Sets

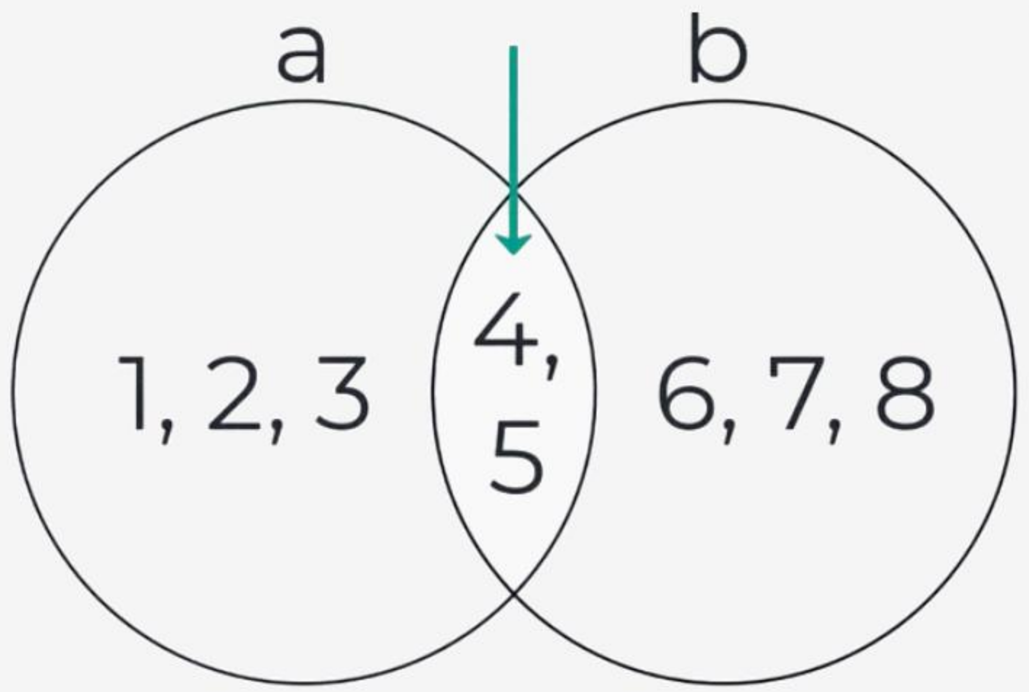
Set Operations - Union

```
a = {1, 2, 3, 4, 5}
b ={4, 5, 6, 7, 8}
a | b
o/p : {1, 2, 3, 4, 5, 6, 7, 8}

a.union (b)
o/p : {1, 2, 3, 4, 5, 6, 7, 8}

b.union (a)
o/p : {1, 2, 3, 4, 5, 6, 7, 8}
```

Sets

Set Operations – Intersection

a = {1, 2, 3, 4, 5}
b = {4, 5, 6, 7, 8}



Sets

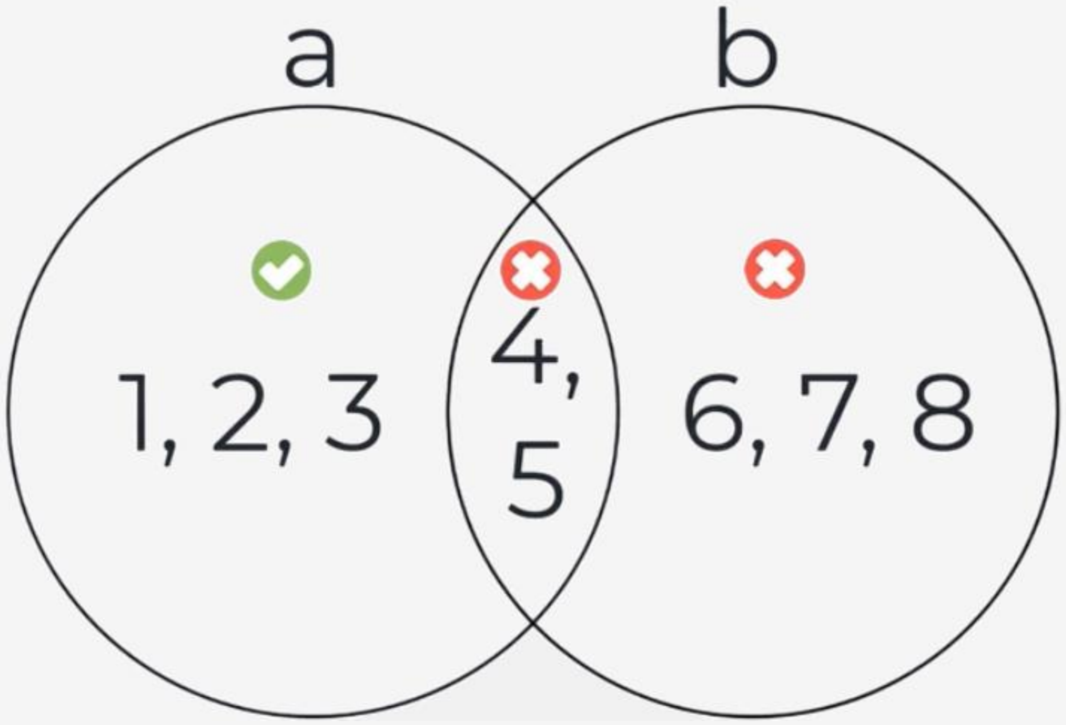
Set Operations – Intersection

```
a = {1, 2, 3, 4, 5}
b = {4, 5, 6, 7, 8}

a & b
o/p : {4, 5}

a.intersection (b)
o/p : {4, 5}

b.intersection (a)
o/p : {4, 5}
```

Sets

Set Operations - Difference

a = {1, 2, 3, 4, 5}
b = {4, 5, 6, 7, 8}



Sets

Set Operations - Difference

a = {1, 2, 3, 4, 5}
b = {4, 5, 6, 7, 8}
a- b
o/p : {1, 2, 3}
b - a
o/p : {6, 7, 8}

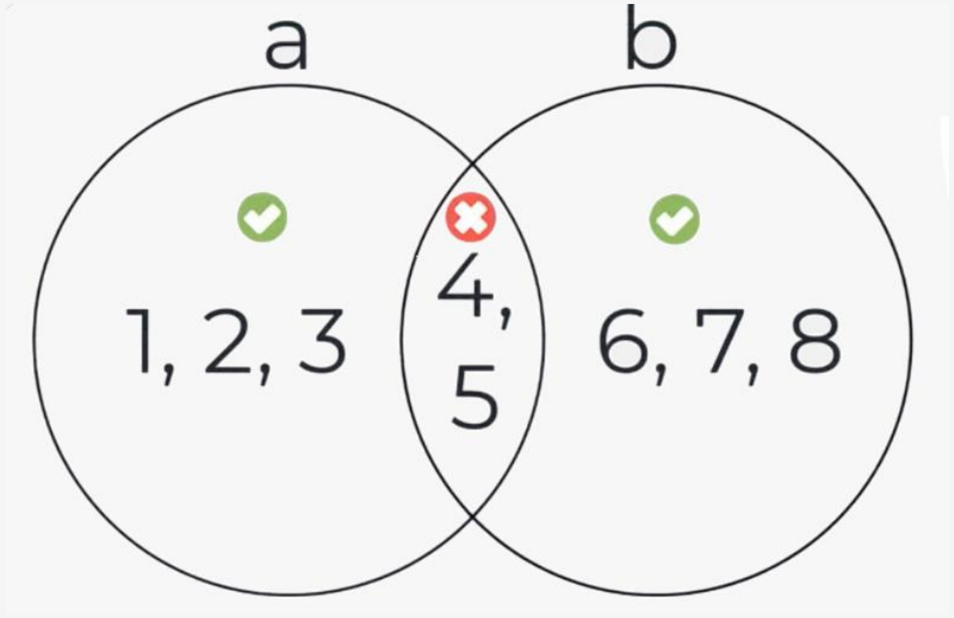
a.difference (b)
o/p : {1, 2, 3}

b.difference(a)
o/p : {6, 7, 8}

Sets

Set Operations - Symmetric Difference

a = {1, 2, 3, 4, 5}
b = {4, 5, 6, 7, 8}



Sets

Set Operations - Symmetric Difference

```
a = {1, 2, 3, 4, 5}
b = {4, 5, 6, 7, 8}

a^ b
o/p : {1, 2, 3, 6, 7, 8}

a.symmetric_difference(b)
o/p : {1, 2, 3, 6, 7, 8}

b.symmetric_difference(a)
o/p : {1, 2, 3, 6, 7, 8}
```

Sets

Useful built-in functions with Sets

```
a = {1, 3, 4, 2, 5, 8, 7, 9, 10, 6}

len(a)
o/p : 10

max(a)
o/p : 10

min(a)
o/p : 1
```

Sets

Useful built-in functions with Sets

a = {1, 3, 4, 2, 5, 8, 7, 9, 10, 6}

sorted(a)
o/p : [1, 2, 3, 4, 5, 6, 7, 8, 9,10]

sum(a)
o/p : 55

1 in a
o/p : True

11 in a
o/p : False

11 not in a
o/p : True

Sets

frozenset

a frozen set has the same characteristics as a set.

For example they will only include unique values and the duplicates will be removed.

But a frozen sets, elements cannot be changed once assigned.

Once they've been assigned, frozen sets are essentially immutable sets.

In other words, you can't change an element of a frozen set.

Now, frozen sets can be created using the frozen set function.

Sets

Frozenset

```
a = set([1, 2, 3, 4, 5])

b = set([4, 5, 6, 7, 8])

c = frozenset([1, 2, 3, 4, 5])

d = frozenset([4, 5, 6, 7, 8])

type(a)
o/p : set

type(c)
o/p : frozenset
```

Sets

Frozenset

For Normal Set

my_dict = {a: "1 to 5", b: "4 to 8"}

o/p : TypeError: unhashable type: 'set’

For Frozen set

my_dict = {c: "1 to 5", d: "4 to 8"}

my_dict[c]

o/p : '1 to 5'

Sets

Frozenset

c | d

o/p : frozenset({1, 2, 3, 4, 5, 6, 7, 8})

c | b

o/p : frozenset({1, 2, 3, 4, 5, 6, 7, 8})

a | d

o/p : {1, 2, 3, 4, 5, 6, 7, 8}

Normal set

type(a | d)

o/p : set

Sets

Frozenset

c | d

o/p : frozenset({1, 2, 3, 4, 5, 6, 7, 8})

c | b

o/p : frozenset({1, 2, 3, 4, 5, 6, 7, 8})

a | d

o/p : {1, 2, 3, 4, 5, 6, 7, 8}

Normal set

type(a | d)

o/p : set

Sets

Frozenset

Summary:

You can use frozen sets as keys in dictionaries because as they are immutable, they are therefore hashable and you will not run into the type error as you do when you use sets as keys.

And finally, frozen sets can be used in the set operations such as union intersection difference and symmetric difference.

And if you're using two frozen sets in the operation, you will return a frozen set object.

if you use a combination of a set and a frozen set in the set operation, you will obtain an object which is identical to the type of object you place first in the set operation.