Sets

- Sets can be used to perform mathematical set operations like union, intersection, symmetric difference etc.

empty set

In [1]:

```
set_var= set()
print(set_var)
```

```
set()
```

- A set is an unordered collection of items.
- Set is defined by values separated by comma inside braces { }.

In [2]:

```
s = {10, 30, 20, 40, 5}
print(s)
```

```
{20, 5, 40, 10, 30}
```

- Every item is unique (no duplicates).

In [3]:

```
s = {10, 4, 20, 20, 30, 30, 30,5}
print(s)
```

```
{4, 5, 20, 10, 30}
```

In [4]:

```
len(s)
```

Out[4]:

```
5
```

- Set doesn't support indexing, because it's unorder collection of items

In [5]:

```
print(s[1])
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
Cell In[5], line 1
----> 1 print(s[1])

TypeError: 'set' object is not subscriptable
```

Set Methods

set.add()

- **add** method adds an element to the set. If the elemnt already exists in the set, nothing will happens

In [6]:

```
s={1,2,3}
s.add(4)

print(s)
```

```
{1, 2, 3, 4}
```

set.update()

- **update** method updates the set with another set or any iterable

In [7]:

```
s1={1,2,3}
s2={4,5}

s1.update(["A","Z","B"])
print(s1)
```

```
{1, 2, 3, 'A', 'Z', 'B'}
```

set.discard()

- **discard** method removes the specified element from the set. If the element does not exist in the set, nothing happens

In [8]:

```
s = {1, 2, 3}
s.discard(2)
print(s)
```

```
{1, 3}
```

In [9]:

```
s = {1, 2, 3}
s.discard(7)
print(s)
```

```
{1, 2, 3}
```

set.remove()

- **remove** method removes the specified element from the set. If the element doesn't exist in the set, a key error is raised.

In [10]:

```
s = {1, 2, 3}
s.remove(2)
print(s)
```

```
{1, 3}
```

In [11]:

```
s = {1, 2, 3}
s.remove(7)
print(s)
```

```
---------------------------------------------------------------------------
-
KeyError                                  Traceback (most recent call las
t)
Cell In[11], line 2
      1 s = {1, 2, 3}
----> 2 s.remove(7)
      3 print(s)

KeyError: 7
```

set.clear()

- **clear** method removes all elements from the set

In [12]:

```
s = {1,2,3}

s.clear()
print(s)
```

```
set()
```

In [13]:

```
s = {1,2,3}

del s                       # delete the variable
print(s)
```

```
---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
Cell In[13], line 4
      1 s = {1,2,3}
      3 del s                       # delete the variable
----> 4 print(s)

NameError: name 's' is not defined
```

set.copy()

- **copy** method returns a shallow copy of the set.
- changes to the copied set doesn't affect the original set

In [14]:

```
s1 = {1,2,3}
s2 =s1.copy()
s2.add(4)
print(s1)
print(s2)
```

```
{1, 2, 3}
{1, 2, 3, 4}
```

set.union()

- this method returns a new set with elements from both sets

In [15]:

```
setA = {1,2,3}
setB = {2,3,4}

setA.union(setB)
```

Out[15]:

```
{1, 2, 3, 4}
```

set.intersection()

- **intersection** method returns a new set with elements that are common to all sets

In [16]:

```
setA = {1,2,3}
setB = {2,3,4}

setA.intersection(setB)
```

Out[16]:

```
{2, 3}
```

set.isdisjoint()

- **isdisjoint** method returns True if two sets have no intersection, otherwise it returns False

In [17]:

```
s1 = {1,2,3}
s2 = {4,5,6}
s3 = {7,5,9}

s = s1.isdisjoint(s2)
print(s)

s = s2.isdisjoint(s3)
print(s)
```

```
True
False
```

set.symmetric_difference()

- **symmetric_difference** method returns a new set with elements that are either of the sets,but not in both
- (A union B) - (A intersection B)

In [18]:

```
setA = {1,2,3}
setB = {2,3,4}

setA.symmetric_difference(setB)
```

Out[18]:

```
{1, 4}
```

set.difference()

- **difference** method returns a new set with elements in the set that are not in the other specified sets

In [19]:

```
setA = {1,2,3}
setB = {2,3,4}

print(setA - setB)
print(setB - setA)
```

```
{1}
{4}
```

issubset()

- **issubset** method returns True if all elements of first set are availble in second set, otherwise it returns False

In [20]:

```
A = {3,4,5}
B = {3,4,5,6}

ans = A.issubset(B)
print(ans)

ans = A.issubset(B)
print(ans)
```

```
True
True
```

issuperset()

- **issuperset** method returns True if all elements of second set are availble in first set, otherwise it returns False

In [21]:

```
s1 = {3,4,5}
s2 = {3,4,5,6}

s = s1.issuperset(s2)
print(s)

s = s2.issuperset(s1)
print(s)
```

```
False
True
```