SETS

- 1.Unorderd :- The itenms in the set are unorderd,i.e., it will not maintain the order in witch the items are inserted.
- 2.Unindexed :- Set items are unindexed.
- 3.Unique :- There cannot be two items with the same value in the set.
- 4.Set items must be immutable :- We cannot change the set items.
- 5.Sets are mutable.

```
# SET METHOD

set = {1,"string",2}

# .add(item)----------This is same as .append() in lists.
set.add(45)
print (set)
```

```
{1, 2, 45, 'string'}
```

```
# .update (iterable)-----------This is asme as .extend()in lists.(str,
tuple ,dis, set)
set.update("str")
print(set)
```

```
{1, 2, 't', 45, 'string', 'r', 's'}
```

```
# .remove ()
set.remove(45)
print(set)
```

```
{1, 2, 't', 'string', 'r', 's'}
```

```
v = {1,2,3}
print(id(v))
v.add(4)
print(id(v))
print(v)
v.update("r")
print(v)
print(id(v))
v.remove("r")
print(v)
print(id(v))
```

```
2458001735968
2458001735968
{1, 2, 3, 4}
{1, 2, 3, 4, 'r'}
2458001735968
```

```
{1, 2, 3, 4}
2458001735968

# set methods

 # union
a = {1,2,3}
b = {1,2,3,4}
print("union method has been used",a.union(b))
a.update(b)

#intersection
a = {1,2,3}
b = {1,2,3,4}

a.intersection(b)   # This will not apply the changes to a
a.intersection_update(b)   #This will apply the changes to a
print(a)


# Difference
x = {"str","hello",True,1}
y = {1,2,3,"Hello","str"}
x.difference(y)
x.difference_update(y)
print(x)




# symmetric difference
x = {"str","hello",True,1}
y = {1,2,3,"Hello","str"}


x.symmetric_difference(y)
x.symmetric_difference_update(y)
print(x)

union method has been used {1, 2, 3, 4}
{1, 2, 3}
{'hello'}
{2, 3, 'hello', 'Hello'}

x = {"str","hello",True,1}
y = {1,2,3,"Hello","str"}

x.symmetric_difference(y)
x.symmetric_difference_update(y)
```

```
print(id(x))
print(id(x.symmetric_difference_update(y)))
```

```
2109158300736
140728909393600
```

```
x = {"str","hello",True,1}
y = {1,2,3,"Hello","str"}
z = x.difference(y)
k = x.difference_update(y)
print(id(x))
print(z)
print(k)
```

```
1666841653120
{'hello'}
None
```

Subset and Superset

```
# supset
a ={1,2,3,4,5,6}
b ={1,2,3}
print(a.issubset(b))
print(a.issubset(a))
print(b.issubset(a))
```

```
False
True
True
```

```
# superset
a ={1,2,3,4,5,6}
b ={1,2,3}
print(a.issuperset(b))
print(a.issuperset(a))
print(b.issuperset(a))
```

```
True
True
False
```

Disjoint

```
a ={1,2,3,4,5,6}
b ={7,8,9}
print(a.isdisjoint(b))
```

```
print(a.isdisjoint(a))
print(b.isdisjoint(a))
```

```
True
False
True
```

```
x = [1,1,2,3,3]
print(x)
```

```
[1, 1, 2, 3, 3]
```

```
x = {1,1,2,3,3}
print(x)
```

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

Function for set.

```
# finding length of the set
a = {1,2,3}
len(a)
```

```
3
```

```
# Minimum and maximum values
a = {1,2,3,4,5,6,78}
print(min(a))
print(max(a))
```

```
1
78
```

```
#adding all the valuse inside a set
x = {300,64,65}
print(sum(x))
```

```
429
```

```
min({"a","b"})
```

```
'a'
```

```
max({"A","a"})
```

```
'a'
```

```
x = {}
x
print(type(x))
```

```
<class 'dict'>
```

Dictionaries

Dictionaries are orderd collections of unique items stored in (key-value)pairs. (pairs of key-value is an items)

1. ordered :- Dictionsries are ordered, which means that the items have a defined order, and that order will not change.
2. Unique :- The keys in Dictionaries should be unique. if we store any value with a key that allready exists, then the most recent value will replace the old value.
3. Keys must be of immutable data type.
4. Dictionaries is mutable.

```
x = {"apple":12,"mango":25}
x
```

```
{'apple': 12, 'mango': 25}
```

```
x = {"apple":12,34:True,"tuple":(1,2,3,4)}
print(x)
print(id(x))
```

```
{'apple': 12, 34: True, 'tuple': (1, 2, 3, 4)}
1666863615424
```

```
y = {"apple":1, (1,2,3,4):"hello"}
y
```

```
{'apple': 1, (1, 2, 3, 4): 'hello'}
```

```
y = {"apple":1, (1,2,3,4):"hell0"}
y
```

```
{'apple': 1, (1, 2, 3, 4): 'hell0'}
```

```
a = {123:"rohit",3456:"rohit",123:"mohit"}
a
```

```
{123: 'mohit', 3456: 'rohit'}
```

Accesing items in a Dictionary

```
a = {"rohit":1234,"mohit":14}
a["rohit"]
```

```
1234
```

```
a["mohit"]
```

```
14
```

```
# keys
print(a.keys())
print(type(a.keys()))
```

```
dict_keys(['rohit', 'mohit'])
<class 'dict_keys'>
```

```
# values
print(a.values())
print(type(a.values()))
```

```
dict_values([1234, 14])
<class 'dict_values'>
```

```
# itens
print(a.items())
print(type(a.items()))
```

```
dict_items([('rohit', 1234), ('mohit', 14)])
<class 'dict_items'>
```

```
# keys-value assignment

a["rahul"] = 4567
print(a)
```

```
{'rohit': 1234, 'mohit': 14, 'rahul': 4567}
```

```
#  .update
a.update({"rishi":9877,})
print(a)
```

```
{'rohit': 1234, 'mohit': 14, 'rahul': 4567, 'rishi': 9877, 'boby':
56789}
```

```
a.update({"rishi":9877,"boby":56789})
print(a)
```

```
{'rohit': 1234, 'mohit': 14, 'rahul': 4567, 'rishi': 9877, 'boby':
56789}
```

Removing items from a dicitionary

```
# .popitems()    - Returns and removes the last imserted item from the
dictionary.
a = {"india":"new delhi","us":"washington"}
```

```
x = a.popitem()
print(a)
print(x)
```

```
{'india': 'new delhi'}
('us', 'washington')
```

```
# pop()   - Removes the item with the key and return its value.


a = {"india":"new delhi","us":"washington"}
a.pop("india")
print(a)
```

```
{'us': 'washington'}
```

```
# del


a = {"india":"new delhi","us":"washington"}
del(a["us"])
print(a)
```

```
{'india': 'new delhi'}
```

```
a = {"india":["new delhi"],"pakistan":"islamabad","us":"washington"}
a
print(type(a))
```

```
<class 'dict'>
```

```
x = "rohit"
x.endswith('')
```

```
True
```

```
X = " RISHI "
X.find("")
```

```
0
```

Range object

Range object(or range data type) is an immutable sequence of integers starting from the given start integer to the stop integer.

```
# range (start , stop , step)
```

```
x = range(1,11,2)
x[4]
```

```
9
```

```
# Let us check the data type of range object
```

```
type(range(1,10))
```

```
range
```

```
# range indexing
```

```
range_x = range(1,10)
range_x[6]
```

```
7
```

```
# What happend when i pass one argument in range()
```

```
list(range(10))
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
tuple(range(10))
```

```
(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
```

```
set(range(10))
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
str(range(10))
```

```
('r', 'a', 'n', 'g', 'e', '(', '0', ',', ' ', '1', '0', ')')
```