Set

In [1]:
```
s = {} #empty set creation it will be default dict
print(s, type(s))
```

```
{} <class 'dict'>
```

In [2]:
```
si = {1,2,3,4,5,6} #set creation using int
print(si, type(si))
```

```
{1, 2, 3, 4, 5, 6} <class 'set'>
```

In [3]:
```
sf = {1.0,2.0,3.0} #set creation by using float
print(sf,type(sf))
```

```
{1.0, 2.0, 3.0} <class 'set'>
```

In [4]:
```
ss = {'a','b','c','d','a','b'} #set creation by using str #duplicat
print(ss,type(ss))
```

```
{'c', 'a', 'd', 'b'} <class 'set'>
```

In [5]:
```
sb = {True,False,True,False} # set creation using bool #duplicate v
print(sb,type(sb))
```

```
{False, True} <class 'set'>
```

In [6]:
```
sc =  {10+10j,10+10j}    #set creation using complex
print(sc,type(sc))
```

```
{(10+10j)} <class 'set'>
```

In [7]:
```
set = {1,10,10.0,12.9,True,'python',10+10j}
print(set,type(set))
```

```
{1, (10+10j), 10, 12.9, 'python'} <class 'set'>
```

In [8]:
```
set.add(11) #add the elements in set we use set.add(element)
set
```

Out[8]:
```
{(10+10j), 1, 10, 11, 12.9, 'python'}
```

In [9]:
```
set.add(11) #duplicate values it will not add (avoid and dont give
set
```

Out[9]:
```
{(10+10j), 1, 10, 11, 12.9, 'python'}
```

In [10]:
```
sc.clear() # to clear the elements in set we use set.clear()
sc
```

Out[10]:
```
set()
```

In [11]:
```
set1 = set.copy() #to copy the elements will use variable name = va
set1
```

Out[11]: {(10+10j), 1, 10, 11, 12.9, 'python'}

In [12]:
```
si
```

Out[12]: {1, 2, 3, 4, 5, 6}

In [13]:
```
set1.difference(si) #difference of two sets
```

Out[13]: {(10+10j), 10, 11, 12.9, 'python'}

In [22]:
```
s = {1,2,3,4,5,6}
s1 = {4,5,6,7,8}
```

In [23]:
```
s.difference_update(s1)
s
```

Out[23]: {1, 2, 3}

In [24]:
```
s.difference(s1)
```

Out[24]: {1, 2, 3}

In [26]:
```
s.intersection(s1)
```

Out[26]: set()

In [33]:
```
a = {1, 2, 3, 4}  # difference
b = {4, 5, 6}
a.difference(b)
```

Out[33]: {1, 2, 3}

In [34]:
```
a = {1, 2, 3, 4} # difference_update
b = {4, 5, 6}

a.difference_update(b)
a
```

Out[34]: {1, 2, 3}

In [39]:
```
a = {1, 2, 3, 4}   #intersection
b = {4, 5, 6}
a.intersection(b)
a
```

Out[39]: {1, 2, 3, 4}

In [40]:
```
a = {1, 2, 3, 4}   #intersection_update
b = {4, 5, 6}
a.intersection_update(b)
a
```

Out[40]: {4}

In [42]:
```
a = {1, 2, 3, 4}   #isdisjoint
b = {4, 5, 6}
a.isdisjoint(b)
```

Out[42]: False

In [43]:
```
a = {1, 2, 3, 4} #isdisjoint
b = { 5, 6}
a.isdisjoint(b)
```

Out[43]: True

In [44]:
```
a = {1, 2, 3, 4} #subset
b = { 5, 6}
a.issubset(b)
```

Out[44]: False

In [46]:
```
a = {1, 2, 3, 4} #superset
b = {1,2,3,4}
a.issuperset(b)
```

Out[46]: True

In [47]:
```
a = {1, 2, 3, 4} #symmetric_difference
b = { 5, 6}
a.symmetric_difference(b)
```

Out[47]: {1, 2, 3, 4, 5, 6}

In [54]:
```
a = {1, 2, 3, 4} #symmetric_difference_update
b = {1,2,3,4,5,6}
a.symmetric_difference_update(b)
a
```

Out[54]: {5, 6}

In [55]:
```
a = {1, 2, 3, 4} #union
b = {1,2,3,4,5,6}
a.union(b)
```

Out[55]: {1, 2, 3, 4, 5, 6}

In [58]:
```
a = {1, 2, 3, 4}
b = {1, 2, 3, 4, 5, 6}
a.pop()
```

Out[58]: 1

In [63]:
```
b.pop() #removes first element
b
```

Out[63]: {5, 6}

In [65]:
```
a = {1, 2, 3, 4} # remove specific element with remove(element)
b = {1, 2, 3, 4, 5, 6}
a.remove(3)
a
```

Out[65]: {1, 2, 4}

In [68]:
```
a = {1, 2, 3, 4}
b = {1, 2, 3, 4, 5, 6}
b.remove(5)
b
```

Out[68]: {1, 2, 3, 4, 6}

In [70]:
```
b = {1, 2, 3, 4, 5, 6}
b.remove(6)
b
```

Out[70]: {1, 2, 3, 4, 5}

In [72]:
```
a = {1, 2, 3, 4} #by using discard function also we can remove spec
a.discard(10)
a
```

Out[72]: {1, 2, 3, 4}

In [74]:
```
a = {1,2,3,4}     #by using update function it will update all the
b = {4,5,6,7}
a.update(b)
a
```

Out[74]: {1, 2, 3, 4, 5, 6, 7}

In [76]:
```
for i in a: #by using for loop
    print(i)
```

```
1
2
3
4
5
6
7
```

In [77]:
```
for i in enumerate(a): # by using for loop with enumerate
    print(i)
```

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

In [78]:
```
all(a) #All/Any
```

Out[78]: True

In [79]:
```
any(a)
```

Out[79]: True

In [80]:
```
1 in a #membership
```

Out[80]: True

In [81]:
```
0 in a #membership
```

Out[81]: False

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: