SET & SET-Operation

1).add() 1).union()

2).copy() 2).intersection()

3).clear() 3).difference()

4).remove() 4).symmetric_difference()

5).discard()

6).pop()

7).update()

In [2]:
```
s={}#empty set#set start with curly bracket
s
```

Out[2]: {}

In [3]:
```
type(s)
```

Out[3]: dict

In [4]:
```
s1=set()#set Type
type(s1)
```

Out[4]: set

In [5]:
```
s1
```

Out[5]: set()

In [6]:
```
s2={20,30,4,7,59}#set always give output in ordered,if the elements are in same dat
s2
```

Out[6]: {4, 7, 20, 30, 59}

In [7]:
```
s3={'a','s','d','f','g'}
s3
```

Out[7]: {'a', 'd', 'f', 'g', 's'}

In [8]:
```
s4={5+9j,4+3j,8+6j,7+8j}#it judge the complex data type from its real number.
s4
```

Out[8]:
```
{(4+3j), (5+9j), (7+8j), (8+6j)}
```

In [9]:
```
s5={2.2,2.2,3.4,5.6,4.9}#repeated elements are not allowed.
s5
```

Out[9]:
```
{2.2, 3.4, 4.9, 5.6}
```

In [10]:
```
s6={True,False}
s6
```

Out[10]:
```
{False, True}
```

In [11]:
```
s7={2,2.3,"sk",1+2j,True}#in mix Data Type we can't consider,
                         #the elements are ordered or not
s7
```

Out[11]:
```
{(1+2j), 2, 2.3, True, 'sk'}
```

In [12]:
```
s8 = {2, 3.4, 'nit', 1+2j, False}
s8
```

Out[12]:
```
{(1+2j), 2, 3.4, False, 'nit'}
```

In [18]:
```
s9={2, 3.4, 'nit', 1+2j, False,[2+3j],(23,45)}
              #since list is mutable,and its not allowed in set.
s9
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[18], line 1
----> 1 s9={2, 3.4, 'nit', 1+2j, False,[2+3j],(23,45)}
      2               #since list is mutable,and its not allowed in set.
      3 s9

TypeError: unhashable type: 'list'
```

In [20]:
```
s0={[2,3,4],[4,5]}
s0
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[20], line 1
----> 1 s0={[2,3,4],[4,5]}
      2 s0

TypeError: unhashable type: 'list'
```

In [22]:
```
s9={2, 3.4, 'nit', 1+2j, False,(2,3),(23,45)}#tuble is immutable and it's allowed i
s9
```

Out[22]:
```
{(1+2j), (2, 3), (23, 45), 2, 3.4, False, 'nit'}
```

In [24]:
```
print(s)
print(s1)
```

```
print(s2)
print(s3)
print(s4)
print(s5)
print(s6)
print(s7)
print(s8)
print(s9)
print(s0)#not defiend cause list present as a element in s0 set.
```

```
{}
set()
{4, 20, 7, 59, 30}
{'f', 'a', 's', 'd', 'g'}
{(5+9j), (8+6j), (4+3j), (7+8j)}
{2.2, 3.4, 4.9, 5.6}
{False, True}
{True, 2, 2.3, (1+2j), 'sk'}
{False, 2, 3.4, (1+2j), 'nit'}
{False, (2, 3), 2, 3.4, (1+2j), (23, 45), 'nit'}
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[24], line 11
      9 print(s8)
     10 print(s9)
---> 11 print(s0)

NameError: name 's0' is not defined
```

In [26]:
```
s8
```

Out[26]:
```
{(1+2j), 2, 3.4, False, 'nit'}
```

In [28]:
```
s8.add(30)
s8
```

Out[28]:
```
{(1+2j), 2, 3.4, 30, False, 'nit'}
```

In [30]:
```
s9.add(45)#it add the elments randomly
s9
```

Out[30]:
```
{(1+2j), (2, 3), (23, 45), 2, 3.4, 45, False, 'nit'}
```

In [32]:
```
s9[0]#indexing is not allowed
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[32], line 1
----> 1 s9[0]

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

In [34]:
```
s9[:]#slicing is also not allowed
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[34], line 1
----> 1 s9[:]

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

In [36]:
```
s9[1:5]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[36], line 1
----> 1 s9[1:5]

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

In [38]:
```
s9
```

Out[38]:
```
{(1+2j), (2, 3), (23, 45), 2, 3.4, 45, False, 'nit'}
```

In [44]:
```
s3
```

Out[44]:
```
{'a', 'd', 'f', 'g', 's'}
```

In [46]:
```
s4
```

Out[46]:
```
{(4+3j), (5+9j), (7+8j), (8+6j)}
```

In [48]:
```
s3=s4.copy()#s3 copied the s4 elments and s3's before elments removed
s3          #cause s3 now points to a copy of s4,s3 become a new set
```

Out[48]:
```
{(4+3j), (5+9j), (7+8j), (8+6j)}
```

In [50]:
```
s3
```

Out[50]:
```
{(4+3j), (5+9j), (7+8j), (8+6j)}
```

In [54]:
```
a1=s3.copy()
a1
```

Out[54]:
```
{(4+3j), (5+9j), (7+8j), (8+6j)}
```

In [56]:
```
a1
```

Out[56]:
```
{(4+3j), (5+9j), (7+8j), (8+6j)}
```

In [9]:
```
e1={23,3.4,1+2j+True}
e1
```

Out[9]:
```
{(2+2j), 23, 3.4}
```

In [11]:
```
e1.clear()
```

In [13]:
```
e1
```

Out[13]: set()

In [17]:
```
e2={33,3.3,3+4j,False,"SK"}
e2
```

Out[17]: {(3+4j), 3.3, 33, False, 'SK'}

In [66]:
```
e3={43,4.3,3+4j,False,"SK"}
e3
```

Out[66]: {(3+4j), 4.3, 43, False, 'SK'}

In [68]:
```
e3.remove(43)
e3
```

Out[68]: {(3+4j), 4.3, False, 'SK'}

In [70]:
```
e3.remove(4.3)
e3
```

Out[70]: {(3+4j), False, 'SK'}

In [72]:
```
e3.remove(False)
e3
```

Out[72]: {(3+4j), 'SK'}

In [81]:
```
e3.remove('SK')
```

In [83]:
```
e3
```

Out[83]: {(3+4j)}

In [87]:
```
d1={'a','b','c','d','e','f'}
```

In [91]:
```
d1.discard('a')
```

In [94]:
```
d1
```

Out[94]: {'b', 'c', 'd', 'e', 'f'}

In [96]:
```
d1.discard('z')#discard never give error,however the element is not present inn set
d1
```

Out[96]: {'b', 'c', 'd', 'e', 'f'}

In [98]:
```
d1.remove('z')#but remove give error,if the elemnt is not present in the set
d1
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[98], line 1
----> 1 d1.remove('z')
      2 d1

KeyError: 'z'
```

In [100…
```
d1.pop()#.pop() randomly remove any elemen, if you not give the element
```

Out[100…
```
'c'
```

In [102…
```
d1
```

Out[102…
```
{'b', 'd', 'e', 'f'}
```

In [108…
```
d1.pop(3)#since indexing is not allowed in set
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[108], line 1
----> 1 d1.pop('e')

TypeError: set.pop() takes no arguments (1 given)
```

In [112…
```
for i in d1:
    print(i)
```

```
e
b
d
f
```

In [116…
```
for i in enumerate(d1):
    print(i)
```

```
(0, 'e')
(1, 'b')
(2, 'd')
(3, 'f')
```

In [118…
```
'k' in d1
```

Out[118…
```
False
```

In [120…
```
'e' in d1
```

Out[120…
```
True
```

In [122…
```
d1
```

Out[122…
```
{'b', 'd', 'e', 'f'}
```

In [128…
```
d1.update(e3)
```

In [132…
```
e3
```

Out[132…
```
{(3+4j)}
```

In [134…
```
d2={20,30,40,False,4+6j}
d2
```

Out[134…
```
{(4+6j), 20, 30, 40, False}
```

In [136…
```
d1
```

Out[136…
```
{(3+4j), 'b', 'd', 'e', 'f'}
```

In [138…
```
d2.update(d1)#d1 elements come first than d2
d2
```

Out[138…
```
{(3+4j), (4+6j), 20, 30, 40, False, 'b', 'd', 'e', 'f'}
```

SET OPERATION

In [147…
```
v1={2,2.3,True,3+4j,"sk"}
v2={3,3.4,False,4+5J,"AK"}
v3={36,7.8,45,78}
```

In [149…
```
v1.union(v2)
```

Out[149…
```
{(3+4j), (4+5j), 2, 2.3, 3, 3.4, 'AK', False, True, 'sk'}
```

In [151…
```
v1.union(v2)
```

Out[151…
```
{(3+4j), (4+5j), 2, 2.3, 3, 3.4, 'AK', False, True, 'sk'}
```

In [153…
```
v3.union(v2)
```

Out[153…
```
{(4+5j), 3, 3.4, 36, 45, 7.8, 78, 'AK', False}
```

In [155…
```
v1|v2#symbol of union
```

Out[155…
```
{(3+4j), (4+5j), 2, 2.3, 3, 3.4, 'AK', False, True, 'sk'}
```

In [157…
```
v1|v2|v3
```

Out[157…
```
{(3+4j), (4+5j), 2, 2.3, 3, 3.4, 36, 45, 7.8, 78, 'AK', False, True, 'sk'}
```

In [159…
```
print(v1)
print(v2)
print(v3)
```

```
{True, 2, 2.3, 'sk', (3+4j)}
{False, 'AK', 3, 3.4, (4+5j)}
{36, 45, 78, 7.8}
```

In [161…
```
v1.intersection(v2)
```

Out[161…
```
set()
```

In [167…
```
r1={3,4,5,6,7}
r1
```

Out[167…
```
{3, 4, 5, 6, 7}
```

In [171…
```
r2={3,4,5,6,1,2}
r2
```

Out[171…
```
{1, 2, 3, 4, 5, 6}
```

In [173…
```
r1.intersection(r2)#take the common elemnts between two sets
```

Out[173…
```
{3, 4, 5, 6}
```

In [175…
```
d1&d2#symbbol of .intersection()
```

Out[175…
```
{(3+4j), 'b', 'd', 'e', 'f'}
```

In [177…
```
v1&v2
```

Out[177…
```
set()
```

In [179…
```
r1&r2
```

Out[179…
```
{3, 4, 5, 6}
```

In [183…
```
r1
```

Out[183…
```
{3, 4, 5, 6, 7}
```

In [185…
```
r2
```

Out[185…
```
{1, 2, 3, 4, 5, 6}
```

In [191…
```
r1.difference(r2)
```

Out[191…
```
{7}
```

In [193…
```
r2.difference(r1)
```

Out[193…
```
{1, 2}
```

In [195…
```
r1-r2#symbol of difference
```

Out[195… {7}

In [197…
```
r2-r1
```

Out[197… {1, 2}

In [199…
```
print(d1)
print(d2)
print(v1)
print(r1)
print(r2)
```

```
{'e', 'b', 'd', 'f', (3+4j)}
{False, 40, (3+4j), 'e', 20, (4+6j), 'b', 'd', 'f', 30}
{True, 2, 2.3, 'sk', (3+4j)}
{False, 'AK', 3, 3.4, (4+5j)}
{3, 4, 5, 6, 7}
{1, 2, 3, 4, 5, 6}
```

In [201…
```
r1.symmetric_difference(r2)#it take uncommon elemnts from both
```

Out[201… {1, 2, 7}

In [203…
```
r10={20,3,5,60,49}
r10
```

Out[203… {3, 5, 20, 49, 60}

In [205…
```
print(r10)
```

```
{49, 3, 20, 5, 60}
```

In [ ]: