SET

In [6]:
```
s={}
s
```

Out[6]: {}

In [8]:
```
type(s)
```

Out[8]: dict

In [10]:
```
s1=set()
s1
```

Out[10]: set()

In [12]:
```
type(s1)
```

Out[12]: set

In [14]:
```
l10=list()
type(l10)
```

Out[14]: list

In [17]:
```
s1
```

Out[17]: set()

In [19]:
```
s2={200,3,1,20,10,10} #similar data types
s2
```

Out[19]: {1, 3, 10, 20, 200}

In [21]:
```
s3={'m','z','r'} # string
s3
```

Out[21]: {'m', 'r', 'z'}

In [23]:
```
s4={1,'m',3.3,1+2j}
s4
```

Out[23]: {(1+2j), 1, 3.3, 'm'}

Set is ordered element for same datatype ,but in mix data type it is inordered.

In [25]:
```
s4.add(1000)
s4
```

Out[25]: {(1+2j), 1, 1000, 3.3, 'm'}

In [27]:
```
s5=s3.copy()
s5
```

Out[27]:
```
{'m', 'r', 'z'}
```

In [29]:
```
s4
```

Out[29]:
```
{(1+2j), 1, 1000, 3.3, 'm'}
```

In [31]:
```
len(s4)
```

Out[31]:
```
5
```

In [33]:
```
s4[0]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[33], line 1
----> 1 s4[0]

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

In [35]:
```
s4[:]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[35], line 1
----> 1 s4[:]

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

slicing and indexing is not allowed in set

In [45]:
```
s3
```

Out[45]:
```
{'0', 'm', 'o', 'r', 'z'}
```

In [47]:
```
s3[0]='n' # immutable
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[47], line 1
----> 1 s3[0]='n'

TypeError: 'set' object does not support item assignment
```

In [49]:
```
s3.add('o')
s3 #mutable
set sometimes work as both
```

Out[49]:
```
{'0', 'm', 'o', 'r', 'z'}
```

In [51]:
```
s3
```

Out[51]:
```
{'0', 'm', 'o', 'r', 'z'}
```

In [53]:
```
s3.pop()
```

Out[53]:
```
'r'
```

In [55]:
```
s2
```

Out[55]:
```
{1, 3, 10, 20, 200}
```

In [57]:
```
s2.pop()
```

Out[57]:
```
1
```

In [59]:
```
s2
```

Out[59]:
```
{3, 10, 20, 200}
```

In [61]:
```
s2.pop(1) # index is not allowed
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[61], line 1
----> 1 s2.pop(1)

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

In [63]:
```
s2
```

Out[63]:
```
{3, 10, 20, 200}
```

In [65]:
```
s2.remove()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[65], line 1
----> 1 s2.remove()

TypeError: set.remove() takes exactly one argument (0 given)
```

In [67]:
```
s2.remove(200)
s2
```

Out[67]:
```
{3, 10, 20}
```

In [69]:
```
s2
```

Out[69]:
```
{3, 10, 20}
```

In [71]:
```
s2.remove(5000) # 5000 is not part of it will get error
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[71], line 1
----> 1 s2.remove(5000)

KeyError: 5000
```

In [73]:
```
s2.discard(5000)
s2 #it will not give error
```

Out[73]:
```
{3, 10, 20}
```

In [75]:
```
s2.discard(10)
s2
```

Out[75]:
```
{3, 20}
```

In [87]:
```
s5={2,3,'sri',1+2j,True,45.8}
s5
```

Out[87]:
```
{(1+2j), 2, 3, 45.8, True, 'sri'}
```

In [79]:
```
type(s5)
```

Out[79]:
```
set
```

In [81]:
```
2 in s5
```

Out[81]:
```
True
```

In [83]:
```
200 in s5
```

Out[83]:
```
False
```

In [85]:
```
s5
```

Out[85]:
```
{(1+2j), 2, 3, 45.8, True, 'sri'}
```

In [89]:
```
for i in s5:
    print(i)
```

```
True
2
3
(1+2j)
sri
45.8
```

In [91]:
```
for i in enumerate(s5):
    print(i)
```

```
(0, True)
(1, 2)
(2, 3)
(3, (1+2j))
(4, 'sri')
(5, 45.8)
```

In [93]:
```
s5
```

Out[93]:
```
{(1+2j), 2, 3, 45.8, True, 'sri'}
```

In [95]:
```
s5.update ([1,2,3])
s5
```

Out[95]:
```
{(1+2j), 2, 3, 45.8, True, 'sri'}
```

In [97]:
```
s5.update([100,200,300])
s5
```

Out[97]:
```
{(1+2j), 100, 2, 200, 3, 300, 45.8, True, 'sri'}
```

In [99]:
```
s6=s5.copy()
s6
```

Out[99]:
```
{(1+2j), 100, 2, 200, 3, 300, 45.8, True, 'sri'}
```

SET OPERATIONS

In [107…
```
A={1,2,3,4,5}
B={4,5,6,7,8}
C={8,9,10}
```

In [109…
```
A.union(B)
```

Out[109…
```
{1, 2, 3, 4, 5, 6, 7, 8}
```

In [111…
```
A.union(B,C)
```

Out[111…
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

In [113…
```
A.union(B,C,56)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[113], line 1
----> 1 A.union(B,C,56)

TypeError: 'int' object is not iterable
```

In [115…
```
print(A)
print(B)
print(C)
```

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

In [117…
```
A | B
```

Out[117…
```
{1, 2, 3, 4, 5, 6, 7, 8}
```

In [119…
```
A | B | C | D
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[119], line 1
----> 1 A | B | C | D

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

In [121…
```
A| B | C
```

Out[121…
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

In [123…
```
A & B #INTERSECTION
```

Out[123…
```
{4, 5}
```

In [125…
```
A.intersection(B)
```

Out[125…
```
{4, 5}
```

In [127…
```
A.intersection(C)
```

Out[127…
```
set()
```

In [129…
```
B.intersection(C)
```

Out[129…
```
{8}
```

In [131…
```
A.intersection(B,C)
```

Out[131…
```
set()
```

In [133…
```
print(A)
print(B)
print(C)
```

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

In [135…
```
A-B # DIFFERENCE
```

Out[135…
```
{1, 2, 3}
```

In [137…
```
A.difference(B)
```

Out[137… {1, 2, 3}

In [139…
```
A.difference(B,C)
```

Out[139… {1, 2, 3}

In [141…
```
A-C
```

Out[141… {1, 2, 3, 4, 5}

In [143…
```
C-A
```

Out[143… {8, 9, 10}

In [145…
```
 print(A)
print(B)
print(C) #SYMMETRIC DIFFERENCE
```

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

In [149…
```
A.symmetric_difference(B) #similar item will be removed
```

Out[149… {1, 2, 3, 6, 7, 8}

In [151…
```
A.symmetric_difference(C)
```

Out[151… {1, 2, 3, 4, 5, 8, 9, 10}

In [153…
```
B.symmetric_difference(C)
```

Out[153… {4, 5, 6, 7, 9, 10}

In [155…
```
B^C
```

Out[155… {4, 5, 6, 7, 9, 10}

In [157…
```
A1={1,2,3,4,5,6,7,8,9}
B1={3,4,5,6,7,8}
C1={10,20,30,40}
```

In [159…
```
B1.issubset(A1)
```

Out[159… True

In [163…
```
A1.issubset(B1)
```

Out[163… False

In [165…
```
A1.issuperset(B1)
```

Out[165… True

In [167…
```
A1={1,2,3,4,5,6,7,8,9}
B1={3,4,5,6,7,8}
C1={10,20,30,40}
```

In [169…
```
C1.issubset(A1)
```

Out[169… False

In [171…
```
B1.issubset(C1)
```

Out[171… False

In [173…
```
C1.isdisjoint(A1)
```

Out[173… True

In [177…
```
C1.isdisjoint(B1)
```

Out[177… True

In [181…
```
A2={1,2,3,4,5,6,7,8,9}
B2={13,14,15,16,17,18}
C2={10,20,30,40}
```

In [183…
```
B2.issubset(A2)
```

Out[183… False

In [185…
```
A2.issuperset(B2)
```

Out[185… False

In [187…
```
B2.isdisjoint(A2)
```

Out[187… True

In [189…
```
for i in enumerate (A):
    print(i)
```

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

In [191…
```
list(enumerate(A))
```

Out[191… [(0, 1), (1, 2), (2, 3), (3, 4), (4, 5)]

DICTIONARY

In [194…
```
d={}
type(d)
```

Out[194…
```
dict
```

In [200…
```
d1={1:'one',2:'two',3:'three'} # dictionary with integer keys
d1
```

Out[200…
```
{1: 'one', 2: 'two', 3: 'three'}
```

In [202…
```
d1.keys()
```

Out[202…
```
dict_keys([1, 2, 3])
```

In [204…
```
d1.values()
```

Out[204…
```
dict_values(['one', 'two', 'three'])
```

In [206…
```
d1.items()
```

Out[206…
```
dict_items([(1, 'one'), (2, 'two'), (3, 'three')])
```

In [208…
```
len(d1.items())
```

Out[208…
```
3
```

In [210…
```
d1
```

Out[210…
```
{1: 'one', 2: 'two', 3: 'three'}
```

In [212…
```
d1[1]
```

Out[212…
```
'one'
```

In [ ]:

In [ ]: