Sets

1) unordered & Unindexed collection of items. 2) Set elements are unique. Duplicate elements are not allowed. 3) Set elements are immutable. 4)Set itself is mutable. We can add or remove items from it.

Set Creation

In [1]:
```
myset={1,2,3,4,5} #set of numbers
myset
```

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

In [2]:
```
len(myset)
```

Out[2]:
```
5
```

In [4]:
```
my_set={1,1,2,2,3,4,4,5}   #duplicate lements are not allowed
my_set
```

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

In [5]:
```
myset1={1.79,2.08,3.99,4.56,5.45}  #set of float elements
myset1
```

Out[5]:
```
{1.79, 2.08, 3.99, 4.56, 5.45}
```

In [6]:
```
myset2={'Venky','Surya','Tyson'}  #set of float elements
myset2
```

Out[6]:
```
{'Surya', 'Tyson', 'Venky'}
```

In [11]:
```
myset3={10,2.57,'Hello',True,2+5j}  #set of mixed data types
myset3
```

Out[11]:
```
{(2+5j), 10, 2.57, 'Hello', True}
```

In [12]:
```
myset4={10,20,'Hello',(5,10,15),4+6j}
myset4
```

Out[12]:
```
{(4+6j), (5, 10, 15), 10, 20, 'Hello'}
```

In [13]:
```
myset5={10,20,'Hello',True,2+6j,[2,4,6]}
myset5
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[13], line 1
----> 1 myset5={10,20,'Hello',True,2+6j,[2,4,6]}
      2 myset5

TypeError: unhashable type: 'list'
```

In [14]:
```
s=set()
s
```

Out[14]:
```
set()
```

In [15]:
```
type(s)
```

Out[15]:
```
set
```

In [16]:
```
s=set(('one','two','three','four','five'))
s
```

Out[16]:
```
{'five', 'four', 'one', 'three', 'two'}
```

Loop through a set

In [17]:
```
for i in s:
    print(i)
```

```
one
four
five
two
three
```

In [18]:
```
for i in enumerate(s):
    print(i)
```

```
(0, 'one')
(1, 'four')
(2, 'five')
(3, 'two')
(4, 'three')
```

Set memebership

In [19]:
```
s1={'six','one','eight','nine','two','four','three'}
s1
```

Out[19]:
```
{'eight', 'four', 'nine', 'one', 'six', 'three', 'two'}
```

In [20]:
```
'one' in s1
```

Out[20]:
```
True
```

In [21]:
```
'seven' in s1
```

Out[21]:
```
False
```

In [22]:
```
'seven' not in s1
```

Out[22]:
```
True
```

In [23]:
```
if  'five' in s1:
    print("Five is present in the set")
else:
    print("Five is not present in the set")
```

```
Five is not present in the set
```

In [24]:
```
if 'four' in s1:
    print("Four is present in the set")
else:
    print("Four is not present in the set")
```

```
Four is present in the set
```

Add & Remove items

In [25]:
```
s1
```

Out[25]:
```
{'eight', 'four', 'nine', 'one', 'six', 'three', 'two'}
```

In [27]:
```
s1.add('five')
s1
```

Out[27]:
```
{'eight', 'five', 'four', 'nine', 'one', 'six', 'three', 'two'}
```

In [29]:
```
s1.update(['TEN','TWELVE','ELEVEN'])
s1
```

Out[29]:
```
{'ELEVEN',
 'TEN',
 'TWELVE',
 'eight',
 'five',
 'four',
 'nine',
 'one',
 'six',
 'three',
 'two'}
```

In [30]:
```
s1.remove('nine')
s1
```

Out[30]:
```
{'ELEVEN',
 'TEN',
 'TWELVE',
 'eight',
 'five',
 'four',
 'one',
 'six',
 'three',
 'two'}
```

In [31]:
```
len(s1)
```

Out[31]: 10

In [32]:
```
s1.discard('TEN')
s1
```

Out[32]: {'ELEVEN', 'TWELVE', 'eight', 'five', 'four', 'one', 'six', 'three', 'two'}

In [33]:
```
len(s1)
```

Out[33]: 9

In [34]:
```
s1.clear()
```

In [35]:
```
s1
```

Out[35]: set()

In [36]:
```
del s1
```

In [37]:
```
s1
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[37], line 1
----> 1 s1

NameError: name 's1' is not defined
```

Copy set

In [39]:
```
s1={'one','two','three','four','five','six','seven','eight'}
s1
```

Out[39]: {'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}

In [41]:
```
s2=s1
s1
```

Out[41]: {'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}

In [42]:
```
s1==s2
```

Out[42]: True

In [43]:
```
id(s1)
```

Out[43]: 2446369676576

In [44]:
```
id(s2)
```

Out[44]: 2446369676576

```
In [45]: len(s1)
Out[45]: 8

In [46]: len(s2)
Out[46]: 8

In [50]: s3=s1.copy()

In [51]: s3
Out[51]: {'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}

In [52]: s1==s3
Out[52]: True

In [53]: s3
Out[53]: {'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}

In [54]: s1
Out[54]: {'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}

In [55]: s2
Out[55]: {'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}

In [56]: id(s3)
Out[56]: 2446369674560

In [57]: s3.add('nine')

In [58]: s3
Out[58]: {'eight', 'five', 'four', 'nine', 'one', 'seven', 'six', 'three', 'two'}

In [59]: len(s3)
Out[59]: 9

In [60]: s1
Out[60]: {'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}

In [61]: s2
Out[61]: {'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

In [62]:
```
s3
```

Out[62]:
```
{'eight', 'five', 'four', 'nine', 'one', 'seven', 'six', 'three', 'two'}
```

Set operation

Union

In [64]:
```
A={1,2,3,4,5}
B={4,5,6,7,8}
C={8,9,10}
```

In [65]:
```
A|B
```

Out[65]:
```
{1, 2, 3, 4, 5, 6, 7, 8}
```

In [85]:
```
A.union(B)
```

Out[85]:
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

In [84]:
```
A.union(C)
```

Out[84]:
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

In [83]:
```
A.union(B,C)
```

Out[83]:
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

In [94]:
```
'''Updates the set calling the update() method with union of A,B&C.
For below example set A will be updated with union of A,B&C).'''
A.update(B,C)
A
```

Out[94]:
```
{4, 5, 6, 7, 8, 9, 10}
```

Intersection

In [86]:
```
A
```

Out[86]:
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

In [87]:
```
B
```

Out[87]:
```
{4, 5, 6, 7, 8}
```

In [88]:
```
A&B
```

Out[88]:
```
{4, 5, 6, 7, 8}
```

In [89]:
```
A.intersection(B)
```

Out[89]: {4, 5, 6, 7, 8}

In [90]:
```
A.intersection(B) Intersection of A and B
```

```
  Cell In[90], line 1
    A.intersection(B) Intersection of A and B
                      ^
SyntaxError: invalid syntax
```

In [93]:
```
'''Updates the set calling the interection_update() method with the intersection of
For below example set A will be updated with intersection of A&B.'''
A.intersection_update(B)
A
```

Out[93]: {4, 5, 6, 7, 8}

In [96]:
```
A.intersection_update(C)
A
```

Out[96]: {8, 9, 10}

In [97]:
```
A
```

Out[97]: {8, 9, 10}

In [98]:
```
B
```

Out[98]: {4, 5, 6, 7, 8}

In [99]:
```
C
```

Out[99]: {8, 9, 10}

Difference

In [100…
```
A={1,2,3,4,5}
B={4,5,6,7,8}
```

In [101…
```
A-B  #set of elements that are only in A not in B
```

Out[101… {1, 2, 3}

In [102…
```
A.difference(B)  #difference of sets
```

Out[102… {1, 2, 3}

In [103…
```
B-A
```

Out[103… {6, 7, 8}

In [104…
```
B.difference(A)
```

Out[104… {6, 7, 8}

In [105…
```
B.difference_update(A)
B
```

Out[105… {6, 7, 8}

In [106…
```
B
```

Out[106… {6, 7, 8}

In [107…
```
A
```

Out[107… {1, 2, 3, 4, 5}

Symmetric Difference

In [108…
```
A1={2,4,6,8,10}
B1={8,10,12,14,16}
C1={20,22,24}
```

In [110…
```
A1^B1
```

Out[110… {2, 4, 6, 12, 14, 16}

In [111…
```
A1.symmetric_difference(B1)
```

Out[111… {2, 4, 6, 12, 14, 16}

In [112…
```
B1.symmetric_difference(A1)
```

Out[112… {2, 4, 6, 12, 14, 16}

In [113…
```
A1^B1^C1
```

Out[113… {2, 4, 6, 12, 14, 16, 20, 22, 24}

In [117…
```
B1.symmetric_difference(C1)
```

Out[117… {8, 10, 12, 14, 16, 20, 22, 24}

In [121…
```
'''Updates the set calling the symmetric_difference_update() method  with the symme
For below example set A will be updated with the symmetric difference of A&B'''
A1.symmetric_difference_update(B1)
A1
```

Out[121… {2, 4, 6, 12, 14, 16}

In [122…
```
print(A1)
print(B1)
print(C1)
```

```
{16, 2, 4, 6, 12, 14}
{16, 8, 10, 12, 14}
{24, 20, 22}
```

Subset, Superset and Disjoint

In [126…
```
A2={1,3,5,7,9,11,13}
B2={7,9,11}
C2={2,4,6,8,10}
```

In [127…
```
B2.issubset(A2)
```

Out[127…
```
True
```

In [128…
```
A2.issubset(B2)
```

Out[128…
```
False
```

In [129…
```
C2.issubset(B2)
```

Out[129…
```
False
```

In [130…
```
A2.issuperset(B2)
```

Out[130…
```
True
```

In [134…
```
B2.issuperset(A2)
```

Out[134…
```
False
```

In [131…
```
C2.isdisjoint(B2)
```

Out[131…
```
True
```

In [132…
```
C2.isdisjoint(A2)
```

Out[132…
```
True
```

In [133…
```
B2.isdisjoint(A2)
```

Out[133…
```
False
```

Other Builtin Functions

In [135…
```
A
```

Out[135…
```
{1, 2, 3, 4, 5}
```

In [136…
```
B
```

Out[136…
```
{6, 7, 8}
```

In [137…
```
C
```

Out[137…
```
{8, 9, 10}
```

In [138…
```
A1
```

Out[138…
```
{2, 4, 6, 12, 14, 16}
```

In [139…
```
sum(A1)
```

Out[139…
```
54
```

In [140…
```
max(A1)
```

Out[140…
```
16
```

In [141…
```
min(A1)
```

Out[141…
```
2
```

In [142…
```
list(enumerate(A1))
```

Out[142…
```
[(0, 16), (1, 2), (2, 4), (3, 6), (4, 12), (5, 14)]
```

In [144…
```
D=sorted(A,reverse=True)
D
```

Out[144…
```
[5, 4, 3, 2, 1]
```

In [145…
```
sorted(D)
```

Out[145…
```
[1, 2, 3, 4, 5]
```

In [ ]:
```

```