Rules of set

- set does not allow duplicates
- set have no indexing/slicing
- set does not allow mutable data types
- set itself is a mutable data type

Set : is created using {} curly braces

```
In [ ]:
```

In python the default behaviour of curly braces {}, is set to dict. rather than set

- therefore here the dic is created instead of set
- So to create an empty set we need to use type casting of set

```
In [2]: s1 = {}
        s1

Out[2]: {}
```

```
In [3]: type(s1)

Out[3]: dict
```

```
In [ ]:
```

Creating an empty set in python

```
In [9]: s2 = set()
        s2

Out[9]: set()
```

```
In [10]: type(s2)

Out[10]: set
```

```
In [ ]:
```

Creating a set

```
In [12]: # homogeneous set
         s1 = {1,2,3,4}
         s1

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

```
In [13]: type(s1)

Out[13]: set
```

```
In [14]: # heterogeneus set
         s2 = {'hey','android',12,'and',18}
         s2

Out[14]: {12, 18, 'and', 'android', 'hey'}
```

```
In [15]: type(s2)

Out[15]: set
```

```
In [ ]:
```

```
In [16]: # set does not allow duplicates
         s3 = {11,23,45,11}
         s3

Out[16]: {11, 23, 45}
```

```
In [ ]:
```

```
In [17]: # set does not allow mutable data types
         s4 = {(1,2,3),'hey'}
         s4

Out[17]: {(1, 2, 3), 'hey'}
```

```
In [18]: s4 = {[1,2,3],'hey'}
         s4

         ---------------------------------------------------------------------------
         ----
         TypeError                                 Traceback (most recent call l
         ast)
         Input In [18], in <module>
         ----> 1 s4 = {[1,2,3],'hey'}
               2 s4

         TypeError: unhashable type: 'list'
```

```
In [ ]:
```

Note:

- Sets have no indexing
- the randomness of the set is due to the fact it follows hashing

```
In [19]: s5 = {1,2,3,'hey',55,533,33,1,2,'gio'}
         s5

Out[19]: {1, 2, 3, 33, 533, 55, 'gio', 'hey'}
```

```
In [ ]:
```

We also cannot create 2D 3D or 4D sets

```
In [20]: s6 = {{4,5,6},{1,2,3}}
         s6
```

```
---------------------------------------------------------------------------
----
TypeError                                 Traceback (most recent call l
ast)
Input In [20], in <module>
----> 1 s6 = {{4,5,6},{1,2,3}}
      2 s6

TypeError: unhashable type: 'set'
```

```
In [ ]:
```

Note :

- we cannot edit the items inside the set
- but we cann add items into the set
    - This Proves set is a mutable data type

```
In [29]: s1 = {1,2,2,3,4}
         print(s1,id(s1))

         {1, 2, 3, 4} 139631726131456
```

```
In [23]: s1.add(4)
         s1

Out[23]: {1, 2, 3, 4}
```

```
In [26]: s1.add(30)
         s1

Out[26]: {1, 2, 3, 4, 30}
```

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

Out[27]: {1, 2, 3, 30, 4, 'bye'}
```

```
In [30]: print(s1,id(s1))

         {1, 2, 3, 4} 139631726131456
```

```
In [ ]:
```

Since set is mutable, we can use the folln fucntions on it

- remove
- pop
- del

```
In [34]: s1 = {1, 2, 3, 4, 'ok',66}
         s1

Out[34]: {1, 2, 3, 4, 66, 'ok'}
```

```
In [35]: # pop
         s1.pop()

Out[35]: 1
```

```
In [36]: s1

Out[36]: {2, 3, 4, 66, 'ok'}
```

```
In [ ]:
```

```
In [37]: s1.pop()

Out[37]: 2
```

```
In [38]: s1

Out[38]: {3, 4, 66, 'ok'}
```

```
In [39]: # remove

         s1.remove('ok')
         s1

Out[39]: {3, 4, 66}
```

```
In [ ]:
```

```
In [40]: # deleting set
         s1

Out[40]: {3, 4, 66}
```

```
In [41]: del s1
```

```
In [42]: s1

         ---------------------------------------------------------------------------
         ----
         NameError                                 Traceback (most recent call l
         ast)
         Input In [42], in <module>
         ----> 1 s1

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

```
In [ ]:
```

we cannot only use the folln operators on set

- concatinate
- multiplication

We can only iterate over the set and use membership operator on it

- looping
- membership operator are available

In [52]:
```
s1 = {3, 4, 66}
s2 = {33, 24, 66, 'oki'}
```

In [ ]:
```

```

In [53]:
```
s1
```

Out[53]:
```
{3, 4, 66}
```

In [54]:
```
s2
```

Out[54]:
```
{24, 33, 66, 'oki'}
```

In [55]:
```
4 in s1
```

Out[55]:
```
True
```

In [56]:
```
'oki' in s1
```

Out[56]:
```
False
```

In [57]:
```
'oki' in s2
```

Out[57]:
```
True
```

In [58]:
```
s2
```

Out[58]:
```
{24, 33, 66, 'oki'}
```

In [59]:
```
for i in s2:
    print(i,end=' ')
```

```
24 33 66 oki
```

In [ ]:
```

```

Fucctions applicable on set

- len
- min/max/sum : if the set consist only int values

In [60]:
```
s1 = {3, 4, 66}
s2 = {33, 24, 66, 'oki'}
```

In [63]:
```
s1
```

Out[63]:
```
{3, 4, 66}
```

```
In [61]: max(s1)

Out[61]: 66
```

```
In [62]: min(s1)

Out[62]: 3
```

```
In [64]: len(s1)

Out[64]: 3
```

Sorted fuctn

- sorted fuctn returns list
- sorted fuctn only works when the datatype only consist of integer values

```
In [73]: sorted(s1)

Out[73]: [3, 4, 66]
```

```
In [67]: sorted(s1,reverse=True)

Out[67]: [66, 4, 3]
```

```
In [ ]:
```

Fucntn that are specific to set are :

- union
- intersection
- difference

```
In [81]: s1 = {1,2,3}
         s2 = {4,5,6}
         s3 = {3,4,5}
```

```
In [84]: print('s1 ',s1)
         print('s2 ',s2)
         s1.union(s2)

         s1  {1, 2, 3}
         s2  {4, 5, 6}

Out[84]: {1, 2, 3, 4, 5, 6}
```

```
In [ ]:
```

```
In [85]: print('s1 ',s1)
         print('s2 ',s2)
         s1.intersection(s2)

         s1  {1, 2, 3}
         s2  {4, 5, 6}

Out[85]: set()
```

In [83]:

```
print('s1 ',s1)
print('s3 ',s3)
s1.intersection(s3)
```

```
s1  {1, 2, 3}
s3  {3, 4, 5}
```

Out[83]:

```
{3}
```

In [ ]:

```

```

difference
- s1.difference(s2)
- means present in s1 and not in s2

In [1]:

```
s1 = {1,2,3}
s2 = {4,5,6}
s3 = {3,4,5}
```

In [2]:

```
s1.difference(s2)
```

Out[2]:

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

In [3]:

```
s1.difference(s3)
```

Out[3]:

```
{1, 2}
```

In [ ]:

```

```

symmetric difference
- all the items which are not present in each other

In [4]:

```
s1 = {1,2,3}
s2 = {4,5,6}
s3 = {3,4,5}
```

In [5]:

```
s1.symmetric_difference(s2)
```

Out[5]:

```
{1, 2, 3, 4, 5, 6}
```

In [6]:

```
s1.symmetric_difference(s3)
```

Out[6]:

```
{1, 2, 4, 5}
```

In [ ]:

```

```

disjoint sets
- when no item is common between the 2 sets

In [7]:

```
s1 = {1,2,3}
s2 = {4,5,6}
s3 = {3,4,5}
```

In [8]:

```
s1.isdisjoint(s2)
```

Out[8]:

```
True
```

```
In [9]: s1.isdisjoint(s3)

Out[9]: False
```

```
In [ ]:
```

subset
- set A is a subset of another set B if all elements of the set A are elements of the set B.

```
In [12]: s1 = {1,2,3}
         s2 = {4,5,6}
         s3 = {3,4,5}
         s4 = {1,2,3}
         s5 = {1,2,3,4,5}
```

```
In [13]: s1.issubset(s2)

Out[13]: False
```

```
In [14]: s1.issubset(s3)

Out[14]: False
```

```
In [15]: s1.issubset(s4)

Out[15]: True
```

```
In [16]: s1.issubset(s5)

Out[16]: True
```

```
In [ ]:
```

Superset
- set A is considered as the superset of B, if all the elements of set B are the elements of set A

```
In [17]: s1 = {1,2,3}
         s3 = {3,4,5}
         s4 = {1,2,3}
         s5 = {1,2,3,4,5}
```

```
In [19]: s1.issuperset(s3)

Out[19]: False
```

```
In [20]: s1.issuperset(s4)

Out[20]: True
```

```
In [21]: s1.issuperset(s5)

Out[21]: False
```

```
In [ ]:
```

```
clear()- to clear/empty the set
```

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

```
Out[22]: {1, 2, 3, 4, 5}
```

```
In [23]: s1.clear()
```

```
In [24]: s1
```

```
Out[24]: set()
```

```
In [ ]:
```