Set Data Structure:

- Rules and guidelines of Set:
  - Set is an unordered collection of item, due to this set is not indexable.
  - Every element in the set is unique. (Set does not allow duplicate item)
  - Set is very powerful for mathematical operations
  - Set can be represented by {} and set function
  - Set is mutable, element can add or remove
  - Set only allowed immutable data type.

In [2]:
```
1 print("All Functions from Set: ",[i for i in dir(set) if "__" not in i])
```

```
All Functions from Set:  ['add', 'clear', 'copy', 'difference', 'difference_
update', 'discard', 'intersection', 'intersection_update', 'isdisjoint', 'is
subset', 'issuperset', 'pop', 'remove', 'symmetric_difference', 'symmetric_d
ifference_update', 'union', 'update']
```

In [3]:
```
1 st = {} # now it is dictionary
```

In [4]:
```
1 type(st)
```

Out[4]:
```
dict
```

In [5]:
```
1 st = {1,3,4,5,11,3,55,23,45,"A","g","A","12",23.40}
```

In [6]:
```
1 st
```

Out[6]:
```
{1, 11, '12', 23, 23.4, 3, 4, 45, 5, 55, 'A', 'g'}
```

In [7]:
```
1 setA = {1,3,5,7,8,9}
```

Adding a element

- Add : Using add function we can add only single element at a time
- update: using update function we can add multiple element at a time if needed

In [8]:
```
1 setA.add("10")
```

In [9]:
```
1 setA
```

Out[9]:
```
{1, '10', 3, 5, 7, 8, 9}
```

Adding multiple element

In [11]:
```
1 setA.update([10,20,30,"A","X","a",10,20,30])
```

In [12]:
```
1 setA
```

Out[12]:
```
{1, 10, '10', 20, 3, 30, 5, 7, 8, 9, 'A', 'X', 'a'}
```

Note:

Set does not allowed a list, dict, and set itself.

In [13]:
```
1 st = {1,2,3,3,3,3,3,3,3,3,[12,34]}
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[13], line 1
----> 1 st = {1,2,3,3,3,3,3,3,3,3,[12,34]}

TypeError: unhashable type: 'list'
```

In [14]:
```
1 st = {1,2,3,3,3,3,3,3,3,3,{12,34}}
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[14], line 1
----> 1 st = {1,2,3,3,3,3,3,3,3,3,{12,34}}

TypeError: unhashable type: 'set'
```

In [15]:
```
1 st = {1,2,3,{"A":10,"b":10}}
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[15], line 1
----> 1 st = {1,2,3,{"A":10,"b":10}}

TypeError: unhashable type: 'dict'
```

In [16]:
```
1 st = {1,2,3,4,(1,2,3,4,5)}
```

In [17]:
```
1 st
```

Out[17]:
```
{(1, 2, 3, 4, 5), 1, 2, 3, 4}
```

How to remove, discard, pop elements from set

In [18]:
```
1 st = {1,3,4,5,11,3,55,23,45,"A","g","A","12",23.40}
```

Remove

We can remove the element if element is part of set otherwise set function raise an exception.

Discard

Discard function does not raise an error if element is not part of set.

Pop

Using pop we can remove random element.

In [19]:
```
1 st
```

Out[19]:
```
{1, 11, '12', 23, 23.4, 3, 4, 45, 5, 55, 'A', 'g'}
```

In [20]:
```
1 st.remove("M")
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[20], line 1
----> 1 st.remove("M")

KeyError: 'M'
```

In [21]:
```
1 st.remove(23)
```

In [22]:
```
1 st
```

Out[22]:
```
{1, 11, '12', 23.4, 3, 4, 45, 5, 55, 'A', 'g'}
```

In [23]:
```
1 st.discard("M")
```

In [24]:
```
st.discard(45)
```

In [25]:
```
st
```

Out[25]: {1, 11, '12', 23.4, 3, 4, 5, 55, 'A', 'g'}

In [26]:
```
st.pop()
```

Out[26]: 1

In [27]:
```
st.pop()
```

Out[27]: 3

In [29]:
```
st
```

Out[29]: {11, '12', 23.4, 4, 5, 55, 'A', 'g'}

In [30]:
```
setA
```

Out[30]: {1, 10, '10', 20, 3, 30, 5, 7, 8, 9, 'A', 'X', 'a'}

In [31]:
```
setA.clear()
```

In [32]:
```
setA
```

Out[32]: set()

In [33]:
```
newst = st.copy()
```

In [34]:
```
newst
```

Out[34]: {11, '12', 23.4, 4, 5, 55, 'A', 'g'}

Set Mathematical Operations

In [35]:
```
setA = {1,3,5,7,8,"a","z","o",11,45}
setB = {7,11,"a","c","m","1",8,12,0}
```

In [36]:
```
setA
```

Out[36]: {1, 11, 3, 45, 5, 7, 8, 'a', 'o', 'z'}

In [37]:
```
1 setB
```

Out[37]:
```
{0, '1', 11, 12, 7, 8, 'a', 'c', 'm'}
```

Set Union (pipe |)

Set union will not allowed the duplicate element, It will combined more than 1 set and return unique element from the sets.

In [38]:
```
1 setA.union(setB)
```

Out[38]:
```
{0, 1, '1', 11, 12, 3, 45, 5, 7, 8, 'a', 'c', 'm', 'o', 'z'}
```

In [39]:
```
1 setB.union(setA)
```

Out[39]:
```
{0, 1, '1', 11, 12, 3, 45, 5, 7, 8, 'a', 'c', 'm', 'o', 'z'}
```

In [40]:
```
1 setA | setB
```

Out[40]:
```
{0, 1, '1', 11, 12, 3, 45, 5, 7, 8, 'a', 'c', 'm', 'o', 'z'}
```

In [41]:
```
1 setB | setA
```

Out[41]:
```
{0, 1, '1', 11, 12, 3, 45, 5, 7, 8, 'a', 'c', 'm', 'o', 'z'}
```

Set Intersection (&)

common values from the sets

In [42]:
```
1 setA.intersection(setB)
```

Out[42]:
```
{11, 7, 8, 'a'}
```

In [43]:
```
1 setB & setA
```

Out[43]:
```
{11, 7, 8, 'a'}
```

In [44]:
```
1 setA
```

Out[44]:
```
{1, 11, 3, 45, 5, 7, 8, 'a', 'o', 'z'}
```

In [45]:
```
1 setB
```

Out[45]:
```
{0, '1', 11, 12, 7, 8, 'a', 'c', 'm'}
```

Set Difference (-)

Elements which are not the part of another set.

In [46]:
```
1 setA.difference(setB)
```

Out[46]:
```
{1, 3, 45, 5, 'o', 'z'}
```

In [47]:
```
1 setB.difference(setA)
```

Out[47]:
```
{0, '1', 12, 'c', 'm'}
```

In [48]:
```
1 setA - setB
```

Out[48]:
```
{1, 3, 45, 5, 'o', 'z'}
```

Set Symmetric_difference (^)

Except common in both.

Return the symmetric difference of two sets as a new set.

(i.e. all elements that are in exactly one of the sets.)

In [50]:
```
1 setA
```

Out[50]:
```
{1, 11, 3, 45, 5, 7, 8, 'a', 'o', 'z'}
```

In [51]:
```
1 setB
```

Out[51]:
```
{0, '1', 11, 12, 7, 8, 'a', 'c', 'm'}
```

In [52]:
```
1 setA.symmetric_difference(setB)
```

Out[52]:
```
{0, 1, '1', 12, 3, 45, 5, 'c', 'm', 'o', 'z'}
```

In [53]:
```
1 setB ^ setA
```

Out[53]:
```
{0, 1, '1', 12, 3, 45, 5, 'c', 'm', 'o', 'z'}
```

In [54]:
```
1 setA & setB
```

Out[54]:
```
{11, 7, 8, 'a'}
```

issubset, issuperset, isdisjoint

In [55]:
```
setX = {4,6,8,9}
setY = {6,7,9,0,11,4,67,6,8,9,4,6,"a"}
```

issubset or issuperset

if all the element on one set are found in another 2nd set then first set is issubset of another set, or you can say that 2nd set is superset of first set.

In [56]:
```
setX.issubset(setY)
```

Out[56]:
```
True
```

In [57]:
```
setY.issuperset(setX)
```

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

isdisjoint

Return True if two sets have a null intersection.

In [58]:
```
setX
```

Out[58]:
```
{4, 6, 8, 9}
```

In [59]:
```
setY
```

Out[59]:
```
{0, 11, 4, 6, 67, 7, 8, 9, 'a'}
```

In [60]:
```
setX.isdisjoint(setY)
```

Out[60]:
```
False
```

In [62]:
```
setX.add("Komal")
```

In [63]:
```
setX
```

Out[63]:
```
{4, 6, 8, 9, 'Komal'}
```

In [64]:
```
1 setX.isdisjoint(setY)
```

Out[64]: False

In [65]:
```
1 setX.remove(4)
```

In [66]:
```
1 setX.remove(8)
```

In [67]:
```
1 setX.remove(6)
```

In [68]:
```
1 setX.remove(9)
```

In [69]:
```
1 setX
```

Out[69]: {'Komal'}

In [70]:
```
1 setX.update([100,"AB","BD","MD"])
```

In [71]:
```
1 setX
```

Out[71]: {100, 'AB', 'BD', 'Komal', 'MD'}

In [72]:
```
1 setY
```

Out[72]: {0, 11, 4, 6, 67, 7, 8, 9, 'a'}

In [73]:
```
1 setX.isdisjoint(setY)
```

Out[73]: True

- difference_update
- intersection_update
- symmetric_difference_update

In [74]:
```
1 setA
```

Out[74]: {1, 11, 3, 45, 5, 7, 8, 'a', 'o', 'z'}

In [75]:
```
1 setB
```

Out[75]: {0, '1', 11, 12, 7, 8, 'a', 'c', 'm'}

In [76]:
```
setA - setB
```

Out[76]:
```
{1, 3, 45, 5, 'o', 'z'}
```

Whatever the difference I am getting from above statement that result or difference I want to override on SetA and SetB

In [77]:
```
setA.difference_update(setB)
```

In [78]:
```
setA
```

Out[78]:
```
{1, 3, 45, 5, 'o', 'z'}
```

In [80]:
```
setB
```

Out[80]:
```
{0, '1', 11, 12, 7, 8, 'a', 'c', 'm'}
```

In [81]:
```
setY
```

Out[81]:
```
{0, 11, 4, 6, 67, 7, 8, 9, 'a'}
```

In [82]:
```
setY & setB
```

Out[82]:
```
{0, 11, 7, 8, 'a'}
```

In [83]:
```
setY.intersection_update(setB)
```

In [84]:
```
setY
```

Out[84]:
```
{0, 11, 7, 8, 'a'}
```

In [85]:
```
setB
```

Out[85]:
```
{0, '1', 11, 12, 7, 8, 'a', 'c', 'm'}
```

In [86]:
```
setB.symmetric_difference_update(setY)
```

In [87]:
```
setB
```

Out[87]:
```
{'1', 12, 'c', 'm'}
```

In [ ]:

```
1
```