Set in Python

A set in Python is an unordered collection of unique elements. It is similar to a list or tuple, but it has some key differences. Unlike lists and tuples, sets are unordered, which means that their elements are not stored in a specific order. This makes sets faster for membership tests (checking if an element is in the set) and for removing duplicates from a collection of items.
It's important to note that sets only contain unique elements. If you try to add an element to a set that is already in the set, the set won't change.
we can create set using {} braces, these curly braces are used for dictionary but in dictionary we have key:value pair here in set we only have values there is no concept of keys or index in set this is the reason they are unordered.

Set items are unchangable but we can add / remove items from set.

In [4]:
```
myFriendSet = {"Ghulam Jillani" , "Ghulam Murtaza", "Hamza Zaffar", "Chan Sheraz", " Muhammad Awais",
               "Jahantab Siddique", "Jibran Jaffar"
              , }
print(myFriendSet)
```

```
{'Ghulam Jillani', 'Ghulam Murtaza', 'Chan Sheraz', 'Hamza Zaffar', ' Muhammad Awais', 'Jahantab Siddique', 'Jibran Jaffar'}
```

Duplicate not allowed

In [5]:
```
#set ignore duplicate values, it will only add one value and if other values are matching simply will be ignored
mySet = {1,1,1,1,1,1,1,2}
print(mySet)
```

```
{1, 2}
```

as shown above I added 1 for 7 time but set only take one value from it and ignore other this is how set ignore values, ignore your haters in same way and boom :)

In [11]:
```
#
mySet = {True, 1,False,0,2}
print(mySet)
```

```
{False, True, 2}
```

Now why the values 1 and 0 not added even they are not in the set, basically the false value represented by 0 and python consider 0 and false as same value, so this is the reason False is already in set so 0 will be ignored and in same way the True is in set so 1 will be ignored

set constructor

In [18]:
```
#we can create set object using its constructor which is set((values))
mySet =set((1,2,False,4,"IUB"))
print(mySet)
```

```
{False, 1, 2, 4, 'IUB'}
```

In [19]:
```
type(mySet)
```

Out[19]:
```
set
```

Accessing Set Elements

We cannot access items in a set by referring to an index or a key.
But we can loop through the set items using a for loop, or ask if a specified value is present in a set, by using the in keyword.

In [24]:
```
awardedPlayers = {"Babar Azam", "Virat Kohli", "Shaheen Shah Afridi","Naseem Shah"}
#search for specific player in set
if "Virat Kohli" in awardedPlayers:
    print("Virat you're selected")
else:
    print("Sorry, Virat come next year")
```

```
Virat you're selected
```

In [25]:
```
#using for in loop
for player in awardedPlayers:
    print("Congratulation {} You're Selected For Award.".format(player))
```

```
Congratulation Virat Kohli You're Selected For Award.
Congratulation Naseem Shah You're Selected For Award.
Congratulation Shaheen Shah Afridi You're Selected For Award.
Congratulation Babar Azam You're Selected For Award.
```

Add Items in Set

Once a set is created, you cannot change its items, but you can add new items.

In [26]:
```
#add function to add an element in set
print(awardedPlayers)
print("length of set:{}".format(len(awardedPlayers)))
awardedPlayers.add("Saad Abdullah")
print("length of set:{}".format(len(awardedPlayers)))
print(awardedPlayers)
```

```
{'Virat Kohli', 'Naseem Shah', 'Shaheen Shah Afridi', 'Babar Azam'}
length of set:4
length of set:5
{'Virat Kohli', 'Saad Abdullah', 'Naseem Shah', 'Shaheen Shah Afridi', 'Babar Azam'}
```

```
we add a player (element) in our set named Saad Abdullah but after inserting it comes to 2nd place
this is why the set in unordered
```

Update Method

update method can be used to add element in set, we can also add elements of another set as well, it inserts the items of set2 into set1

In [28]:
```
numberSet1 = {1,2,3,4,5}
numberSet2 = {6,7,8,9,1}
numberSet1.update(numberSet2)
print(numberSet1)
```

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

In [30]:
```
#we can also add list elements into a set
numberSet1 = {1,2,3,4,5}
numberSet2 = ["First","Second","Third","Fourth"]
numberSet1.update(numberSet2)
print(numberSet1)
```

```
{1, 2, 3, 4, 5, 'Third', 'Second', 'First', 'Fourth'}
```

In [31]:
```
#we can add dictionary keys in set as well
numberSet1 = {1,2,3,4,5}
numberSet2 = {"key1":"value1","key2":"value2"}
numberSet1.update(numberSet2)
print(numberSet1)
```

```
{1, 2, 3, 4, 5, 'key1', 'key2'}
```

In [44]:
```
#we can add dictionary values in set as well
numberSet1 = {1,2,3,4,5}
numberSet2 = {"key1":"value1","key2":"value2"}
numberSet1.update(numberSet2.values()) #passing values of dictionary
print(numberSet1)
```

```
{1, 2, 3, 4, 5, 'value2', 'value1'}
```

In [35]:
```
#we can also store dictionary entire (key,value) key-value pair as tuple in set,
#we can add dictionary values in set as well
numberSet1 = {1,2,3,4,5}
numberSet2 = {"key1":"value1","key2":"value2"}
numberSet1.update(numberSet2.items()) #passing values of dictionary
print(numberSet1)
```

```
{1, 2, 3, 4, 5, ('key1', 'value1'), ('key2', 'value2')}
```

Remove Item from Set

```
To remove an item in a set, use the remove(), or the discard() method.
```

In [45]:
```
print(numberSet1)
numberSet1.remove(1) #remove element 1
numberSet1.remove('value2') #remove element 'value2'
print(numberSet1)
```

```
{1, 2, 3, 4, 5, 'value2', 'value1'}
{2, 3, 4, 5, 'value1'}
```

Discard Method to remove element

In [46]:
```
#discard method
numberSet1.discard('value1')
numberSet1.discard('i don\'t exist ')
print(numberSet1)
```

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

```
both the discard and remove perform same operation try to remove the given item from set but if the item
is not present in the set then the remove metohd will throw error of item not found and the discard will not throw
any error, it will check if item is present in set remove it otherwise ignore
```

pop() method

pop() method does not take any parameter and remove any random element from set let see an example,

In [59]:
```
numberSet1 = {1,2,3,4,5}
numberSet1.pop() #it will remove any random element
numberSet1.pop() #it will remove any random element from set
print(numberSet1)
```

```
{3, 4, 5}
```

clear() and del

In [47]:
```
#clear method will clear entire set items but it will not delete set object
numberSet1.clear()
print(numberSet1)
```

```
set()
```

In [48]:
```
#del keyword will delete the set object we can't use it later, if we try so an error will be thrown
del numberSet1
print(numberSet1)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[48], line 3
      1 #del keyword will delete the set object we can't use it later, if we try so an error will be thrown
      2 del numberSet1
----> 3 print(numberSet1)

NameError: name 'numberSet1' is not defined
```

Loop with set

```
we can use for loop to iterate through element of set as,
```

In [50]:
```
mySet = {10,99,834,23324,345}
for item in mySet:
    print(item)
```

```
834
99
345
10
23324
```

Join Set

```
There are several ways to join two or more sets in Python. we already seen an update() method which help us to join
two sets of python.
we can also use the union() method that returns a new set containing all items from both sets
```

In [52]:
```
mySet = {10,20,30}
mySet2 = {30,40,50}
mySet3=mySet.union(mySet2)
print(mySet3)
```

```
{50, 20, 40, 10, 30}
```

Math like method with set

we can perform all math like operations which we can perform on set in python like intersection where we only keep the items which are in both sets for example for this we use method intersection_update(). this method does not return any new set instead it modify the first set

In [54]:
```
mySet = {10,20,30}
mySet2 = {30,40,50}
mySet.intersection_update(mySet2)
print(mySet)
```

```
{30}
```

In [57]:
```
#The intersection_update() modify set1 but return nothin, to create a new set which contains intersected object use,
#intersection() method as,
mySet = {10,20,30}
mySet2 = {30,40,50}
mySet3=mySet.intersection(mySet2)#it will not modify any set instead return new set which contain common items
print(mySet3)
```

```
{30}
```

Keep All, But NOT the Duplicates

The symmetric_difference_update() method will keep only the elements that are NOT present in both sets.

In [58]:
```
mySet = {10,20,30}
mySet2 = {30,40,50}
mySet3=mySet.symmetric_difference(mySet2)#it will not modify any set instead return new set
print(mySet3)
```

```
{40, 10, 50, 20}
```

```
There are lot of other method related to set which we can use for difference purpose like to check if set is subset
of another set or not etc.
```

In [ ]:
```

```