Sets

-> A set is an unordered collection of items. Every element is unique (no duplicates).

-> The set itself is mutable. We can add or remove items from it.

-> Sets can be used to perform mathematical set operations like union, intersection, symmetric difference etc.

Set Creation

In [1]:

```
#set of integers
s = {1, 2, 3}
print(s)

#print type of s
print(type(s))
```

```
set([1, 2, 3])
<type 'set'>
```

In [1]:

```
#set doesn't allow duplicates. They store only one instance.
s = {1, 2, 3, 1, 4}
print(s)
```

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

In [2]:

```
#we can make set from a list
s = set([1, 2, 3, 1])
print(s)
```

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

In [3]:

```
#initialize a set  with set() method
s = set()

print(type(s))
```

```
<class 'set'>
```

Add element to a Set

In [4]:

```
#we can add single element using add() method and
#add multiple elements using update() method
s = {1, 3}

#set object doesn't support indexing
print(s[1]) #will get TypeError
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-4-c52fc339e293> in <module>()
      4
      5 #set object doesn't support indexing
----> 6 print(s[1]) #will get TypeError

TypeError: 'set' object does not support indexing
```

In [5]:

```
#add element
s.add(2)
print(s)
```

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

In [6]:

```
#add multiple elements
s.update([5, 6, 1])
print(s)
```

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

In [11]:

```
#add list and set
s.update([8, 9], {10, 2, 3})
print(s)
```

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

Remove elements from a Set

In [12]:

```
#A particular item can be removed from set using methods,
#discard() and remove().

s = {1, 2, 3, 5, 4}
print(s)

s.discard(4)    #4 is removed from set s

print(s)
```

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

In [13]:

```
#remove an element
s.remove(2)

print(s)
```

```
{1, 3, 5}
```

In [14]:

```
#remove an element not present in a set s
s.remove(7) # will get KeyError
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-14-f37cc9806699> in <module>()
      1 #remove an element not present in a set s
----> 2 s.remove(7) # will get KeyError

KeyError: 7
```

In [15]:

```
#discard an element not present in a set s
s.discard(7)
print(s)
```

```
{1, 3, 5}
```

In [16]:

```
#we can remove item using pop() method

s = {1, 2, 3, 5, 4}

s.pop() #remove random element

print(s)
```

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

In [17]:

```
s.pop()
print(s)
```

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

In [18]:

```
s = {1, 5, 2, 3, 6}

s.clear()   #remove all items in set using clear() method

print(s)
```

```
set()
```

Python Set Operations

In [19]:

```
set1 = {1, 2, 3, 4, 5}
set2 = {3, 4, 5, 6, 7}

#union of 2 sets using | operator

print(set1 | set2)
```

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

In [20]:

```
#another way of getting union of 2 sets
print(set1.union(set2))
```

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

In [21]:

```
#intersection of 2 sets using & operator
print(set1 & set2)
```

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

In [22]:

```
#use intersection function
print(set1.intersection(set2))
```

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

In [23]:

```
#set Difference: set of elements that are only in set1 but not in set2

print(set1 - set2)
```

```
{1, 2}
```

In [24]:

```
#use differnce function
print(set1.difference(set2))
```

```
{1, 2}
```

In [25]:

```
"""symmetric difference: set of elements in both set1 and set2
#except those that are common in both."""

#use ^ operator

print(set1^set2)
```

```
{1, 2, 6, 7}
```

In [26]:

```
#use symmetric_difference function
print(set1.symmetric_difference(set2))
```

```
{1, 2, 6, 7}
```

In [27]:

```
#find issubset()
x = {"a","b","c","d","e"}
y = {"c","d"}

print("set 'x' is subset of 'y' ?", x.issubset(y)) #check x is subset of y

#check y is subset of x
print("set 'y' is subset of 'x' ?", y.issubset(x))
```

```
set 'x' is subset of 'y' ? False
set 'y' is subset of 'x' ? True
```

Frozen Sets

Frozen sets has the characteristics of sets, but we can't be changed once it's assigned. While tuple are immutable lists, frozen sets are immutable sets

Frozensets can be created using the function frozenset()

Sets being mutable are unhashable, so they can't be used as dictionary keys. On the other hand, frozensets are hashable and can be used as keys to a dictionary.

This datatype supports methods like copy(), difference(), intersection(), isdisjoint(), issubset(), issuperset(), symmetric_difference() and union(). Being immutable it does not have method that add or remove elements.

In [28]:

```
set1 = frozenset([1, 2, 3, 4])
set2 = frozenset([3, 4, 5, 6])

#try to add element into set1 gives an error
set1.add(5)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-28-8f5ea3d0c7e1> in <module>()
      3
      4 #try to add element into set1 gives an error
----> 5 set1.add(5)

AttributeError: 'frozenset' object has no attribute 'add'
```

In [27]:

```
print(set1[1]) # frozen set doesn't support indexing
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-27-8fc108f08ec8> in <module>()
----> 1 print(set1[1]) # frozen set doesn't support indexing

TypeError: 'frozenset' object does not support indexing
```

In [28]:

```
print(set1 | set2) #union of 2 sets
```

```
frozenset({1, 2, 3, 4, 5, 6})
```

In [29]:

```
#intersection of two sets
print(set1 & set2)

#or
print(set1.intersection(set2))
```

```
frozenset({3, 4})
frozenset({3, 4})
```

In [30]:

```
#symmetric difference
print(set1 ^ set2)

#or
print(set1.symmetric_difference(set2))
```

```
frozenset({1, 2, 5, 6})
frozenset({1, 2, 5, 6})
```

In [ ]:

```

```

In [ ]:

```

```