Data Structure: Set

a set unordered collection of items. Every element is unique(no duplications).

The set it self mutable. We can add or remove items from it.

sets can be used to perform mathematical set operations like union, intersection, symmetric difference.

Set creation

```
a = {} # empty set
print(a)

s = {1,2,3,4,5,6} # set of integers
print(s)
print(type(s))

d = {1,2,3,'a','b','c','d','e',4,5,6, 'A', 'B'}
print(d)

# set does't allow duplication. They store only one instance.
s = { 0,0,1,1,1,2,2,2,3,3,3,3,4,4}
print(s)

# we can make set from a list
a = [1,2,3,4,5,6]
print(type(a))

a = set(a)
print(a)
print(type(a))


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

Add element to a set

```
# we can add single element using add() method and
# add multiple elements using update() method
s = {1,2,3}

# set does't support indexing
print(s[1])
```

```
# add element
s = {1,2,3}
s.add(11)
print(s)
s.add(10)
print(s)

# add multiple elements
s.update([6,60,100,75,1000])
print(s)
```

Remove element form a set

```
# discard() and remove() method can be remove particular items form
set.
s = {1, 2, 3, 4, 5, 6, 6, 7}
print(s)
s.discard(6)
print(s)

s = {1, 2, 3, 4, 5, 6, 7, 1}
s.remove(1)
print(s)

# remove element that not present in set
s.remove(11)
print(s)

# discard an element that not present in set.
s.discard(11)
print(s)

# we can remove items using pop() method
s.pop()
print(s)

s.pop()
print(s)

s.clear()
print(s)
s.add(1)
print(s)
```

Set operations

```
# union operation
set1 = {1,2,3,4,5,6,7}
set2 = {5,6,7,8,9,10}
print( set1 | set2)
print( set2 | set1)

print(set1.union(set2))
print("")
print(set1)
print(set2)

# intersection
print(set1 & set2)

print(set1.intersection(set2))
print("")
print(set1)

# difference
print(set1 - set2)

print(set1.difference(set2))
print("")
print(set1)

""" symmetric difference: set of elements in both set1 and set2 expect
those that are comman in both"""
# use ^ operator
print (set2 ^ set1)

set1 = {1,2,3,4,5,6,7,0}
set2 = {5,6,7,8,9,10,11,12}
print(set1.symmetric_difference(set2))

# 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 a subset of 'x' ?", y.issubset(x))

x = {'a', 'b', 'c', 'd', 'e'}
y = {'a', 'b', 'c', 'd', 'e'}
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 a subset of 'x' ?", y.issubset(x))
```

Frozen sets

Frozen sets has the characteristic of sets, but we can't change once it's assigned. While Tuple are immutable lists, Frozen sets are immutable sets.

Frozensets can be created using function frozenset().

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

This datatype supports methods like copy(), difference(), intersection(),isdisjoint(),issubset(),issuperset(), symmetric_difference() and union().

being immutable it doesn't have method that add and remove elements.

```
set1 = frozenset([1,2,3,4,5,6,7])
set2 = frozenset([4,5,6,7,8,9,10])

# try to add element into set1 given error
set1.add(8)
```

```
print(set1[2]) # frozen set doesn't support indexing.
```

```
#set1 = frozenset([1,2,3,4,5,6,7])
#set2 = frozenset([4,5,6,7,8,9,10])
print(set1 | set2)
print(set1.union(set2))
```

```
print(set1 & set2)
```

```
print( set1 ^ set2)
```

```
set3 = frozenset([])
print(set3)
```

```
set3 = frozenset([])
set3.add(1)
print(set3)
```