(3.5) Set and Frozenset

unique collection of immutable objects

If we want to represent a group of unique values as a single entity then we should go for set.

set objects are mutable

i.e once we creates set object we can perform any changes in that object based on our requirement.

Duplicates are not allowed

unordered

Insertion order is not preserved.But we can sort the elements.

Indexing and slicing notallowed

Heterogeneous elements are allowed

We can represent set elements within curly braces and with comma seperation

We can apply mathematical operations like union,intersection,difference etc on set objects

Growable in nature

In [5]:
```
s1 = {'sun','mon','tue','wed','thu','fri','sat'}
print(s1,type(s1))
```

```
{'fri', 'sat', 'tue', 'wed', 'mon', 'thu', 'sun'} <class 'set'>
```

In [6]:
```
s = {1,1,23,4,2,3,4314, "ok"}
print(s,type(s))
```

```
{1, 2, 3, 4, 23, 4314, 'ok'} <class 'set'>
```

In [8]:
```
s = { 1, 2, 3,1, 1,2, 1, 1, 2, 2, 1,  1 ,2, 1, 2, 1}
print(s, type(s))
```

```
{1, 2, 3} <class 'set'>
```

mutable are not allowed

In [16]:
```
s1 = {'sun','mon','tue','wed','thu','fri','sat',['jan','feb','mar']}
print(s1,type(s1))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
C:\Users\PANKAJ~1\AppData\Local\Temp/ipykernel_10732/1757285045.py in <module>
```

```
----> 1 s1 = {'sun','mon','tue','wed','thu','fri','sat',['jan','feb','mar']}
      2 print(s1,type(s1))

TypeError: unhashable type: 'list'
```

1. Creation of Set objects

In [29]:
```
s = {10,20,30,40}
print(s,type(s))
```

```
{40, 10, 20, 30} <class 'set'>
```

using set() ffunction

s = set(any sequence)

In [30]:
```
l = [10,20,30,40,10,20,10]
s=set(l)
print(s) # {40, 10, 20, 30}
```

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

In [31]:
```
s=set(range(5))
print(s) #{0, 1, 2, 3, 4}
```

```
{0, 1, 2, 3, 4}
```

for empty set

s={} It is treated as dictionary but not empty set.

In [32]:
```
s={}
```

```
{} <class 'dict'>
```

In [33]:
```
s=set()
print(s, type(s))
```

```
set() <class 'set'>
```

In [ ]:
```

```

In [54]:
```
print(set())

print(int())

print(float())

print(complex())

print(repr(str()))

print(list())

print(tuple())

print(dict())
```

```
set()
0
0.0
0j
''
[]
()
{}
```

2. Important functions of set

A. add(x)

add individual item x to the set

In [38]:
```
s = set()

s.add(5)
s.add(10)
s.add(5)
s.add(9)
s.add(20)

print(s)
```

```
{9, 10, 20, 5}
```

In [40]:
```
s.add((1, 2, 3, 4))
print(s)
```

```
{5, 9, 10, (1, 2, 3, 4), 20}
```

In [39]:
```
s.add([1, 2, 3, 4])
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
C:\Users\PANKAJ~1\AppData\Local\Temp/ipykernel_10732/2854902495.py in <module>
----> 1 s.add([1, 2, 3, 4])

TypeError: unhashable type: 'list'
```

B. update(x,y,z)

To add multiple items to the set.

Arguments are not individual elements and these are Iterable objects like List,range etc.

All elements present in the given Iterable objects will be added to the set.

In [36]:
```
s={10,20,30}
l=[40,50,60,10]
s.update(l,range(5))
print(s)
```

```
{0, 1, 2, 3, 4, 40, 10, 50, 20, 60, 30}
```

Q. What is the difference between add() and update() functions in set?

We can use add() to add individual item to the Set

where as we can use update() function to add multiple items to Set.

add() function can take only one argument where as update() function can take any number of arguments but all arguments should be iterable objects

Q. Which of the following are valid for set s?

s.add(10) #valid

s.add(10,20,30) TypeError: add() takes exactly one argument (3 given)

s.update(10) TypeError: 'int' object is not iterable

s.update(range(1,10,2),range(0,10,2)) #valid

C. copy()

Returns copy of the set.
It is cloned object.

In [42]:
```
s={10,20,30}
s1=s.copy()
print(s,id(s))
print(s1,id(s1))
```

```
{10, 20, 30} 2640565704736
{10, 20, 30} 2640565533376
```

D. pop():

It removes and returns some random element from the set.

In [43]:
```
s={40,10,30,20}
print(s)

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

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

E. remove(x):

It removes specified element from the set.
If the specified element not present in the Set then we will get KeyError

In [44]:
```
s={40,10,30,20}
s.remove(30)
print(s)
```

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

In [45]:
```
s.remove(50)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
C:\Users\PANKAJ~1\AppData\Local\Temp/ipykernel_10732/2383628386.py in <module>
----> 1 s.remove(50)

KeyError: 50
```

F. discard(x):

It removes the specified element from the set.
If the specified element not present in the set then we won't get any error.

In [46]:
```
s={10,20,30}
s.discard(10)
print(s)

s.discard(50)
print(s)
```

```
{20, 30}
{20, 30}
```

G. clear()

To remove all elements from the Set.

In [47]:
```
s={10,20,30}
print(s)
s.clear()
print(s)
```

```
{10, 20, 30}
set()
```

3. Mathematical operations on the Set

A. union()

x.union(y) ==>We can use this function to return all elements present in both sets

x.union(y) or x|y

In [48]:
```
x={10,20,30,40}
y={30,40,50,60}
print(x.union(y)) #{10, 20, 30, 40, 50, 60}
print(x|y)
```

```
{40, 10, 50, 20, 60, 30}
{40, 10, 50, 20, 60, 30}
```

B. intersection():

x.intersection(y) or x&y
Returns common elements present in both x and y

In [49]:
```
x={10,20,30,40}
y={30,40,50,60}
print(x.intersection(y)) #{40, 30}
print(x&y) #{40, 30}
```

```
{40, 30}
{40, 30}
```

C. difference():

x.difference(y) or x-y
returns the elements present in x but not in y

In [50]:
```
x={10,20,30,40}
y={30,40,50,60}
print(x.difference(y)) #{10, 20}
print(x-y) #{10, 20}
print(y-x) #{50, 60}
```

```
{10, 20}
{10, 20}
{50, 60}
```

D. symmetric_difference():

x.symmetric_difference(y) or x^y
Returns elements present in either x or y but not in both

In [51]:
```
x={10,20,30,40}
y={30,40,50,60}
print(x.symmetric_difference(y)) #{10, 50, 20, 60}
print(x^y) #{10, 50, 20, 60}
```

```
{10, 50, 20, 60}
{10, 50, 20, 60}
```

E. isdisjoint()

In [52]:
```
s1 = { 1, 2, 3, 4, }
s2 = { 4, 5, 6, 7, 8}

s1.isdisjoint(s2)
```

Out[52]:
```
False
```

F. isupperset()

In [53]:
```
s1 = { 1, 2, 3, 4, 5}
s2 = { 2, 4, 5}

print(s1.issuperset(s2))
```

```
True
```

4. Membership operators: (in , not in)

In [55]:
```
s = set("Pankaj")
print(s)

print('a' in s)
print('z' in s)
```

```
{'n', 'a', 'k', 'P', 'j'}
True
False
```

5. Set Comprehension

Set comprehension is possible

In [59]:
```
s = {x*x for x in range(5)}
print(s)
```

```
{0, 1, 4, 9, 16}
```

In [60]:
```
s = {2**x for x in range(2,10,2)}
print(s)
```

```
{16, 256, 64, 4}
```

6. set objects won't support indexing and slicing

In [61]:
```
s={10,20,30,40}
print(s[0])
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
C:\Users\PANKAJ~1\AppData\Local\Temp/ipykernel_10732/2652238546.py in <module>
      1 s={10,20,30,40}
----> 2 print(s[0])

TypeError: 'set' object is not subscriptable
```

In [62]:
```
print(s[1:3])
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
C:\Users\PANKAJ~1\AppData\Local\Temp/ipykernel_10732/2090310509.py in <module>
----> 1 print(s[1:3])

TypeError: 'set' object is not subscriptable
```

7. Problems

Q.Write a program to eliminate duplicates present in the list?

Approach-1:

In [63]:
```
```

```
l=eval(input("Enter List of values: "))
s=set(l)
print(s)
```

```
Enter List of values: [10,20,30,10,20,40]
{40, 10, 20, 30}
```

Approach-2: order preserved

In [65]:
```
l=eval(input("Enter List of values: "))
l1=[]
for x in l:
    if x not in l1:
        l1.append(x)
print(l1)
```

```
Enter List of values: [10,20,30,10,20,40]
[10, 20, 30, 40]
```

Q. Write a program to print different vowels present in the given word?

In [67]:
```
w=input("Enter word to search for vowels: ")
s=set(w)
v={'a','e','i','o','u'}
d=s.intersection(v)
print("The different vowel present in",w,"are",d)
```

```
Enter word to search for vowels: pankaj kumar
The different vowel present in pankaj kumar are {'a', 'u'}
```

(3.5.1)Frozenset

immutable set Hence we cannot use add or remove functions

In [68]:
```
s = frozenset([1, 1, 2, 1, 1 ,1, 2, 3, 4, 1])
print(s)
```

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

In [69]:
```
s = {10,20,30,40}
fs = frozenset(s)

print(fs,type(fs))
```

```
frozenset({40, 10, 20, 30}) <class 'frozenset'>
```

In [70]:
```
fs.add(70)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
C:\Users\PANKAJ~1\AppData\Local\Temp/ipykernel_10732/1993397630.py in <module>
----> 1 fs.add(70)

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

In [71]:
```
fs.remove(30)
```

```
---------------------------------------------------------------------------
```

```
AttributeError                            Traceback (most recent call last)
C:\Users\PANKAJ~1\AppData\Local\Temp/ipykernel_10732/836129544.py in <module>
----> 1 fs.remove(30)

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

In [ ]: