SET

DATA STRUCTURE

- If we want to represent a group of unique values as a single entity then we should go for set.
- Duplicates are not allowed.
- Insertion order is not preserved.But we can sort the elements.
- Indexing and slicing not allowed for the set.
- Heterogeneous elements are allowed.
- Set objects are mutable i.e once we creates set object we can perform any changes in that object based on our requirement.
- 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.

Creation of Set Objects:

```
s={10,20,30,40}
print(s)
print(type(s))
```

Output

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

We can create set objects by using set() Function   s = set(any sequence)

Eg 1:

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

Eg 2:

```
s=set(range(5))
print(s) #{0, 1, 2, 3, 4}
```

Note:

- While creating empty set we have to take special care.
- Compulsory we should use set() function.
- s = {} → It is treated as dictionary but not empty set.

```
s={}
print(s)
print(type(s))
```

Output

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

Eg:

```
s=set()
print(s)
print(type(s))
```

Output

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

Important Functions of Set:

1) add(x):

Adds item x to the set.

```
s={10,20,30}
s.add(40);
print(s)  #{40, 10, 20, 30}
```

2) 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.

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

Output: {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?

1) s.add(10)
2) s.add(10,20,30) → TypeError: add() takes exactly one argument (3 given)
3) s.update(10) → TypeError: 'int' object is not iterable
4) s.update(range(1,10,2),range(0,10,2))

3) copy():

- Returns copy of the set.
- It is cloned object.

```
s = {10,20,30}
s1 = s.copy()
print(s1)
```

4) pop():

It removes and returns some random element from the set.

```
s={40,10,30,20}
print(s)
print(s.pop())
print(s)
```

Output

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

5) remove(x):

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

```
s = {40, 10, 30, 20}
s.remove(30)
print {◊(s) 40, 10, 20}
s.remove(50 KeyError: ◊) 50
```

6) discard(x):

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

```
s = {10, 20, 30}
s.discard(10)
```

```
print {◊(s)  20, 30}
s.discard(50)
print {◊(s) 20, 30}
```

Q) What is the difference between remove() and discard() functions in Set?
Q) Explain differences between pop(),remove() and discard() functionsin Set?

7) <u>clear():</u>

To remove all elements from the Set.

```
s={10,20,30}
print(s)
s.clear()
print(s)
```

<u>**Output**</u>

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

<u>Mathematical Operations on the Set:</u>

1) <u>union():</u>

- x.union(y) → We can use this function to return all elements present in both sets
- x.union(y) OR x|y.

```
x = {10, 20, 30, 40}
y = {30, 40, 50, 60}
print (x.union(y))  → {10, 20, 30, 40, 50, 60}
print (x|y) → {10, 20, 30, 40, 50, 60}
```

2) <u>intersection():</u>

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

```
x = {10, 20, 30, 40}
y = {30, 40, 50, 60}
print (x.intersection(y)) → {40, 30}
print(x&y) → {40, 30}
```

3) difference():

- x.difference(y) OR x-y.
- Returns the elements present in x but not in y.

```
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}
```

4) symmetric_difference():

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

```
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}
```

Membership Operators: (in, not in)

```
s=set("basic")
print(s)
print('d' in s)
print('z' in s)
```

Output

```
{'a', 'i', 's', 'b', 'c'}
True
False
```

Set Comprehension:

Set comprehension is possible.

```
s = {x*x  for x in range(5)}
print (s)  → {0, 1, 4, 9, 16}

s = {2**x  for x in range(2,10,2)}
print (s) → {16, 256, 64, 4}
```

Set Objects won't support indexing and slicing:

```
s = {10,20,30,40}
print(s[0]) → TypeError: 'set' object does not support indexing
print(s[1:3]) → TypeError: 'set' object is not subscriptable
```

Q) Write a Program to eliminate Duplicates Present in the List?

Approach - 1

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

```
D:\Python_classes>py test.py
Enter List of values: [10,20,30,10,20,40]
{40, 10, 20, 30}
```

Approach - 2

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

```
D:\Python_classes>py test.py
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?

```
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)
```

```
D:\Python_classes>py test.py
Enter word to search for vowels: basic
The different vowel present in basic are {'a', 'i'}
```