SET

If we want to represent unique values, we use python sets.
In this data type,

1. Duplicates are not allowed.
2. Insertion order is not preserved so use of slicing and indexing is not allowed.
3. It is mutable
4. It is heterogeneous

- **Creating a set:**

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

Output:

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

- **Important functions of set:**

1. add()

```
s={10,20,30,40,50}
s.add(60)
print(s)
```

Output:

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

2. update()

```
s={10,20,30,40,50}
l=[100,200,300,400]
s.update(l)
print(s)
```

Output:

```
{200, 10, 400, 20, 30, 100, 40, 300, 50}
```

Note:

```
s={10,20,30,40}
s.add(50)
```

```
s.add(50,60)
```

It is invalid to add multiple values. In this case, we have to use the update function.

3. cloning/copy()

```
s={10,20,30,40}
s1=s.copy()
print(s1)
```

Output:

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

4. pop()

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

Output:

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

Note:

- As the insertion order is not preserved, pop will pop out or delete a random element from the set.
- To remove a specific item we use remove() in the same manner.
- clear() is used to empty the whole set.

- **Mathematical Operations On Set**

1. Union

```
x={10,20,30,40}
y={50,60,70,80}
print(x.union(y))
```

Output:

```
{70, 40, 10, 80, 50, 20, 60, 30}
```

2. Intersection

```
x={10,20,30,40}
y={10,20,70,80}
print(x.intersection(y)
```

Output:

```
{10, 20}
```

3. Difference

```
x={10,20,30,40}
y={10,20,70,80}
print(x.difference(y))
```

Output:

```
{40, 30}
```

4. Membership

```
x={10,20,30,40}
y={10,20,70,80}
print(10 in x)
print(200 in y)
```

Output:

```
True
False
```

- **Set Comprehension:**

  The process of creating set from some sequence.

```
s={x for x in range(40) }
print(s)
```

Output:

```
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,
36, 37, 38, 39}
```