❖ Python List:

- List is an ordered sequence of items.
- All the items in a list do not need to be of the same type.
- List is a collection of mixed data types of items.
- List should be represented by using square bracket [ ].
- For List data type,predefined class **list** is used.
- Positive and Negative index associated with the list items.
- List positive index should begin with 0 and negative index should begin with -1 from the last element.
- Lists are mutable. It means once it is created, you can change it later.
- Subscript [] and index number is used to access any individual element from the list.
- We can use slice [:] operator to access the range of items from the list.
- Example:

```
a = [5,10,15,20,25,30,35,40]
print(a)
print(a[2])
print(a[-3])
print(a[0:3])
print(a[5:])
```

When you run the above program, the output will be:

```
[5, 10, 15, 20, 25, 30, 35, 40]
15
30
[5, 10, 15]
[30, 35, 40]
```

Methods of List:

1) **len()** :- This method is used to find out the how many items present in list.

- Example:

```
a = [10,20,30,40,50]
print("Length of list=",len(a))
```

- Output:

```
Length of list= 5
```

2) **append()** :- To add an item to the end of the list, we use the append() method

- Example:

```
a = [10,20,30,40,50]
a.append(60)
print("All Items of list=", a)
```

- Output:

```
All Items of list= [10, 20, 30, 40, 50, 60]
```

3) **extend()** :- To add no of items to the end of the list, we use the extend() method.

- Example:

```
a = [10,20,30,40,50]
a.extend([60,70,80]);
print("All Items of list=", a)
```

- Output:

```
All Items of list= [10, 20, 30, 40, 50, 60, 70, 80]
```

4) **insert()** :- To add an item at the specified index, we use the insert() method

- Example:

```
a = [10,20,30,40,50]
a.insert(1,15)
print("All Items of list=", a)
```

- Output:

```
All Items of list= [10, 15, 20, 30, 40, 50, 60]
```

5) **del** :- The del keyword is used to delete the specified index value or it can also delete the complete list

- Example:

```
a = [10,20,30,40,50]
del a[1]
print("list after deleting index 1 value=", a)
#if you want to delete complete list
del a
```

- Output:

```
list after deleting index 1 value= [10, 30, 40, 50]
```

6) **remove()** :- This method is used to remove the specified item from the list.

- Example:

```
a = [10,20,30,40,50]
a.remove(40)
print("list after removing 40 value=", a)
```

- Output:

```
list after removing 40 value = [10, 20, 30, 50]
```

6) **pop()** :- This method is used to remove the specified index value from the list or it will remove the last value if index is not specified.

- Example:

```
a = [10,20,30,40,50]
a.pop(1)
print("list after removing index 1 value=", a)
a.pop()
print("list after removing last value=", a)
```

- Output:

```
list after removing index 1 value= [10, 30, 40, 50]
list after removing last value= [10, 30, 40]
```

7) **clear()** :- This method is used to make an empty list.

- Example:

```
a = [10,20,30,40,50]
a.clear()
print("List of Values=",a)
```

- Output:

```
List of items= []
```

8) **index()** :- This method finds the given element in a list and returns its index number. If the same element is present more than once, the method returns the index of the first occurrence of the element.

- Example:

```
a = [10,20,30,40,50]
print("Index of 30 element=", a.index(30))
```

- Output:

```
Index of 30 element= 2
```

9) **count()** :- This method counts how many times an element present in a list and returns it.

- Example:

```
a = [10,20,30,40,50,30,30]
print("Count of 30 =", a.index(30))
```

- Output:

```
Count of 30 = 3
```

10) **sort()** :- This method sorts the elements of a given list in a specific order - Ascending or Descending.

- Example:

```
a = [10,20,30,40,50,30,30]
a.sort();
print("Sort the element in Ascending order =", a)
b = [10,20,30,40,50,30,30]
b.sort(reverse=True)
print("Sort the element in Descending order =", b)
```

- Output:

```
Sort the element in Ascending order = [10, 20, 30, 30, 30, 40, 50]
Sort the element in Descending order = [50, 40, 30, 30, 30, 20, 10]
```

11) **reverse()** :- This method is used to reverses the elements of given list.

- Example:

```
a = [10,20,30,40,50]
a.reverse();
print("Reverse the list elements =", a)
```

- Output:

```
Reverse the list elements = [50, 40, 30, 20, 10]
```

12) **copy()** :- This method is used copy all elements of one list to another list.

- Example:

```
a = [10,20,30,40,50]
b=a.copy();
print("Copied list b elements =", b)
```

- Output:

```
Copied list b elements = [10, 20, 30, 40, 50]
```

13) **min()** :- This method is used to find out the minimum element in the list.

- Example:

```
a = (10,20,30,40,50,30,30)
print("Minimum element =", min(a))
```

- Output:

```
Minimum element = 10
```

14) **max()** :- This method is used to find out the maximum element in the list.

- Example:

```
a = (10,20,30,40,50,30,30)
```

```
print("Maximum element =", max(a))
```

- Output:

```
Maximum element = 50
```

❖ Python Tuple:

- Tuple is an ordered sequence of items.
- All the items in a Tuple do not need to be of the same type.
- Tuple is a collection of mixed data types of items.
- Tuple should be represented by using parentheses ().
- For Tuple data type,predefined class **tuple** is used.
- Positive and Negative index associated with the Tuple items.
- Positive index should begin with 0 and negative index should begin with -1 from the last element.
- Tuple are immutable. It means once it is created, you can not change it later.
- Subscript [] and index number is used to access any particular element from the Tuple.
- We can use slice [:] operator to access the range of items from the Tuple
- Tuples are used to read only data and it is faster than list as it cannot change dynamically.
- Example:

```
a = (5,10,15,20,25,30,35,40)
print(a)
print(a[2])
print(a[-3])
print(a[0:3])
print(a[5:])
```

When you run the above program, the output will be:

```
(5, 10, 15, 20, 25, 30, 35, 40)
15
30
(5, 10, 15)
(30, 35, 40)
```

Methods of Tuples:

1) **len()** :- This method is used to find out the how many items present in tuple.

- Example:

```
a = (10,20,30,40,50)
print("Length of Tuple=",len(a))
```

- Output:

```
Length of Tuple= 5
```

2) **del** :- The del keyword is used to delete the tuple completely.

- Example:

```
a = (10,20,30,40,50)
del a
print(a);
```

- Output:

```
NameError: name 'a' is not defined
```

3) **index()** :- This method finds the given element in a tuple and returns its index number. If the same element is present more than once, the method returns the index of the first occurrence of the element.

- Example:

```
a = (10,20,30,40,50)
print("Index of 30 element=", a.index(30))
```

- Output:

```
Index of 30 element= 2
```

4) **count()** :- This method counts how many times an element present in a tuple and returns it.

- Example:

```
a = (10,20,30,40,50,30,30)
print("Count of 30 =", a.index(30))
```

- Output:

```
Count of 30 = 3
```

5) **min()** :- This method is used to find out the minimum element in the Tuple.

- Example:

```
a = (10,20,30,40,50,30,30)
print("Minimum element =", min(a))
```

- Output:

```
Minimum element = 10
```

6) **max()** :- This method is used to find out the maximum element in the Tuple.

- Example:

```
a = (10,20,30,40,50,30,30)
print("Maximum element =", max(a))
```

- Output:

```
Maximum element = 50
```

❖ Python Set:

- Set is an unordered collection of unique items.
- Set is defined by values separated by comma inside curly bracket { }.
- Items in a set are not ordered.
- We can perform set operations like union, intersection on two sets.
- Set have unique values. They eliminate duplicates.
- Set are unordered collection of items and index numbers are not associated with it. Hence the slicing operator [:] does not work.
- Example:

```
a = {5,10,15,20,25,30,35,40}
print(a)
```

When you run the above program, the output will be:

```
{35, 5, 40, 10, 15, 20, 25, 30}
```

Methods of Set:

1) **len()** :- This method is used to find out the how many items present in list.

- Example:

```
a = {10,20,30,40,50}
print("Length of set=",len(a))
```

- Output:

```
Length of set= 5
```

2) **add()** :- To add new item to the set, we use the add() method

- Example:

```
a = {10,20,30,40,50}
a.add(60)
print("All elements of set=", a)
```

- Output:

```
All elements of set= {40, 10, 50, 20, 60, 30}
```

3) **update()** :- To add multiple items to the set, we use the update() method.

- Example:

```
a = {10,20,30,40,50}
a.update([60,70,80]);
print("All elements of set=", a)
```

- Output:

```
All elements of set= {70, 40, 10, 80, 50, 20, 60, 30}
```

4) **del** :- The del keyword is used to delete the set completely.

- Example:

```
a = {10,20,30,40,50}
del a
print(a)
```

- Output:

```
NameError: name 'a' is not defined
```

5) **remove()** :- This method is used to remove the specified item from the set.

- Example:

```
a = {10,20,30,40,50}
a.remove(40)
print("Set after removing 40 value=", a)
```

- Output:

```
Set after removing 40 value= {10, 50, 20, 30}
```

6) **pop()** :- This method is used to remove remove the last item. Remember that sets are unordered, so you will not know what item that gets removed. The return value of the pop() method is the removed item.

- Example:

```
a = {10,20,30,40,50}
print("Removed Element=", a.pop())
```

- Output:

```
Removed Element= 40
```

7) **clear()** :- This method is used to make an empty list.

- Example:

```
a = {10,20,30,40,50}
a.clear()
print("Set elements=",a)
```

- Output:

```
Set elements= set()
```

8) **union()** :- This method return a set that contains all items from both sets, duplicates are removed.

- Example:

```
a = {10,20,30,40,50}
b={20,60,70}
c=a.union(b)
print("Union set=",c)
```

- Output:

```
Union set= {50, 20, 70, 40, 10, 60, 30}
```

9) **difference()** :- This method return set that contains the items that only exist in set a, and not in set b.

- Example:

```
a = {10,20,30,40,50}
b={20,60,70}
c=a.difference(b)
print("Difference set=",c)
```

- Output:

```
Difference set= {40, 10, 50, 30}
```

10) **intersection()** :- This method return a set that contains the items that exist in both set a and b.

- Example:

```
a = {10,20,30,40,50}
b={20,60,70}
c=a.intersection(b)
print("Intersection set=",c)
```

- Output:

```
Intersection set= {20}
```

❖ Python Dictionary:

- Dictionary is an ordered collection of items and it should be represented by using (key:value) pairs format.
- It is generally used when we have a huge amount of data.
- Dictionaries are optimized for retrieving data. We must know the key to retrieve the value.
- In Python, dictionaries are defined by using curly bracket {} with each item being a pair in the form of **key:value**.
- Key and value can be of any data type.
- Example:

```
d = {1:'Pune', 2:'Solapur', 3:'Tuljapur', 4:'Thane'}
print("First city is ",d[1])
print("Fourth city is ",d[4])
print (d)
```

When you run the above program, the output will be:

```
First city is  Pune
Fourth city is  Thane
{1: 'Pune', 2: 'Solapur', 3: 'Tuljapur', 4: 'Thane'}
```

Methods of Dictionary:

1) **len()** :- This method is used to find out the how many items (key:value pairs) present in dictionary.

- Example:

```
a = {1:'Pune', 2:'Solapur', 3:'Tuljapur', 4:'Thane'}
print("Length of Dictionary=",len(a))
```

- Output:

```
Length of Dictionary= 3
```

2) **del** :- The del keyword is used to delete the specified key name or it can also delete the complete dictionary

- Example:

```
a = {1:'Pune', 2:'Solapur', 3:'Tuljapur', 4:'Thane'}
del a[1]
print("list after deleting key 1 =", a)
#if you want to delete complete dictionary
del a
```

- Output:

```
list after deleting key 1 = {2: 'Solapur', 3: 'Tuljapur', 4: 'Thane'}
```

3) **pop()** & **popitem()** :- The pop() method is used to remove the specified key value from the dictionary. The popitem() method is used to remove the last inserted element from the dictionary.

- Example:

```
a = {1:'Pune', 2:'Solapur', 3:'Tuljapur', 4:'Thane'}
a.pop(1)
print("Dictionary after removing key 1=", a)
a.popitem()
print("Dictionary after removing last element=", a)
```

- Output:

```
Dictionary after removing key 1= {2: 'Solapur', 3: 'Tuljapur', 4: 'Thane'}
Dictionary after removing last element= {2: 'Solapur', 3: 'Tuljapur'}
```

4) **clear()** :- This method is used to make an empty list.

- Example:

```
a = {1:'Pune', 2:'Solapur', 3:'Tuljapur', 4:'Thane'}
a.clear()
print("Elements of Dictionary=",a)
```

- Output:

5) **copy()** :- This method is used copy all elements of one dictionary to another dictionary.

- Example:

```
a = {1:'Pune', 2:'Solapur', 3:'Tuljapur', 4:'Thane'}
b=a.copy();
print("Copied dict b elements =", b)
```

- Output:

```
Copied dict b elements = {1: 'Pune', 2: 'Solapur', 3: 'Tuljapur', 4: 'Thane'}
```

6) **min()** :- This method is used to find out the minimum key element in the dictionary.

- Example:

```
a = {1:'Pune', 2:'Solapur', 3:'Tuljapur', 4:'Thane'}
print("Minimum key element =", min(a))
```

- Output:

```
Minimum key element = 1
```

7) **max()** :- This method is used to find out the maximum key element in the dictionary.

- Example:

```
a = {1:'Pune', 2:'Solapur', 3:'Tuljapur', 4:'Thane'}
print("Maximum key element =", max(a))
```

- Output:

```
Maximum key element = 4
```

8) **get()** :- This method is used to returns the value for the specified key if key is in dictionary.

- Example:

```
a = {1:'Pune', 2:'Solapur', 3:'Tuljapur', 4:'Thane'}
print("Value =", a.get(3))
```

- Output:

```
Value = Tuljapur
```

9) **update()** :- This method is used to updates the dictionary. If key value is already present then its corresponding value will get changed. If key value not present then it will add new entry in the dictionary.

- Example:

```
d = {1: "one", 2: "three"}
d1 = {2: "two"}

# updates the value of key 2
d.update(d1)
print(d)

d1 = {3: "three"}

# adds element with key 3
d.update(d1)
print(d)
```

- Output:

```
{1: 'one', 2: 'two'}
{1: 'one', 2: 'two', 3: 'three'}
```

10) **keys()** & **values()** :- The keys() method display list of all keys in the dictionary. The values() method display list of all values in the dictionary.

- Example:

```
a = {1:'Pune', 2:'Solapur', 3:'Tuljapur', 4:'Thane'}
print("All Keys=",a.keys())
print("All Values=",a.values())
```

- Output:

```
All Keys= dict_keys([1, 2, 3, 4])
All Values= dict_values(['Pune', 'Solapur', 'Tuljapur', 'Thane'])
```