Python List

(8) List:

- Lists are used to store multiple items in a single variable.
- Lits are used to store collection of data.
- List are just like dynamic size Arrays.
- List can contain heterogenous values such as integers, float, strings, tuples list, and dictionaries, but they are commonly used to store collection of homogeneous objects.
- list are mutable sequences. (can be change after creation).
- list should be always used when we want to store item in some kind of order.
- Element in a list are indexed according to definite sequence.
- Indexing of a list done with [0] being first indexed.
- Each element in a list are definite place in a list, which allows duplicating of elements in a list.
- In list each element having its own distict place and credibility.

(i) List Items:

- list items are changeble, ordered, and allow duplicate values.
- list item are indexed, first item has indexed [0], second item index [1] etc.

(ii) ordered:

- list items are define order, and that order will not change.
- If new item added in list, new item placed at the end of list.

(iii) changable:

- list are changeble, that means we can change, add, and remove item in a list after has been created.

(iv) Allow Duplicates:

- list are indexed, list can have items with the same value.

(x) list length:

- In list, how many items to list has, use len-function.

Example :

```
thislist = [ "Apple", "Banana", "Mango", "Papaya" ]
print (len(thislist)).
```

(ii). List Item - Data types.

- List item can be any data type. (string, int, float)

Example :

```
list_1 = [ "Apple", "Banana", "Mango" ]
list_2 = [ 1, 5, 7, 9, 3 ]
list_3 = [ True, false, false ]
```

(iii) type()

- It is used for checking what datatypes of a list?

```
print(type(thislist)).
```

(iii) list() constructor :

- It is also possible to use the list () constructor, when creating a new list.

Example :

```
thislist = list(("apple", Banana")
print(thislist).
```

(1) Access Items :

- List item are indexed and you can Access them by referring to the indexed Numbers.

```
thislist = ["Apple", "Banana", "cherry", "Mango"]
print (thislist[1]).
```

(i) positive Indexing. : print (thislist[2])

(ii) Negative Indexing. : print (thislist[-1]).

(2) Range of Indexes :

- you can specify a range of indexes by specifying where to start and where to End the range.

- When a specifying a range, the return value will be a new list with specified items.

```
thislist = ["Apple", "Banana", "Mango", "cherry", "Kiwi"]
print(thislist[2:5]).
```

(ii) By leaving out start value, then range will start at first item.

```
thislist = ["Apple", "Banana", "cherry", "orange"]
print(thislist[:4])
```

(iii) By leaving out end value, range will go on to the end of list.

```
print(thislist[2:])
```

(i) positive Indexing : `print(thislist[1:4])`
(ii) Negative Indexing : `print(thislist[-4:-1])`.

③ Change Item value:

- To change a item value of specific item, refer to the index number.

```
thislist[1] = "Blackcurrent"
print(thislist).
```

(i) change Range of Item value:

- To change the value within specific Range, define list with new values, and refer to the range of Index numbers

```
thislist[1:3] = ["Black currant", "Watermallon"]
print(thislist).
```

- If you insert more item than you replace, the new item will be inserted where you specified, and remaining item move accordingly.

```
thislist = ["Apple", "Banana", "Mango"]
thislist[1:2] = ["Blackcurrent", "Watermalon"]
print(thislist)
```

- the length of the list will change when number of items inserted does not match number of item replaced.

- If you insert less item than you replace, the new item will be inserted where you specified, and remaining items will move accordingly.

```
thislist[1:3] = ["Watermallon"]
print(thislist)
```

(ii) Insert Items:

- To insert new item list, without replacing ~~without~~ any of the existing values, we can use insert method.
- insert() method inserts an item at the specified index.

```
thislist.insert(2, "Watermallon")
print(thislist).
```

(4) ~~Change~~ Add list items: (i) append()

- To add an item to the end of list, use the append() method.
- Using append() method to append an item.

```
thislist.append("orange")
print(thislist).
```

(ii) Insert Items:

- To insert a list item at a specified index, use the insert() method.
- The insert() methods inserts an item at the specified index.

```
thislist.Index(1, "orange")
print(thislist).
```

(iii) Extend list

- append element from another list to the current list
- use Extend() method.

```
thislist = ["~~apple~~ A", "B", "C"]
list2 = ["D", "E", "F"]
thislist.extend(list2)
Print(thislist).
```

- the element will added to the end of the list.

(iv) Add Any Iterable :

- the extend() method does not have to append list, you can add any iterable objects.
- add elements to tuple to a list

```
thislist = ["A", "B", "C"]
list2 = ("D", "E")
thislist.extend(list2)
print(thislist).
```

(5) Remove list Items.

(i) Remove.

- remove() method removes specified items.

```
thislist.remove("Banana")
print(thislist).
```

(ii) Remove specified Index :

- pop() method removes the specified index.

```
thislist.pop(1)
print(thislist).
```

- if you do not specify the index, pop() method removes last items.

(iii) del keywords also removes the specified index.

```
del thislist[1]
print(thislist).
```

- del keywords can also delete complete list

```
del thislist
```

(6) Loop List :

(i) Loop through list :

- you can loop through the list item by using a for loop.

```
thislist = ["A", "B", "C"]
for x in thislist:
    print(x).
```

(ii) Loop Through the Index Numbers.

- You can also loop through the list items by referring to their index numbers.
- Use range() and len() function to create a creole suitable iterable.

```
thislist = ["A", "B", "C"]
For i in range(len(thislist)):
    print(thislist[i]).
```

(iii) Using a While loop:

- You can loop through the list item by using while loop
- Use len() function to determine length of a list item.
- Remember to increase the index by 1 after each iteration.

```
thislist = ["Apple", "Banana", "cherry"]
i=0
while i < len(thislist):
    print(thislist[i])
    i= i+1.
```

(6) (iv) List comprehension:

- A short hand for loop that will print all items in a list.

```
thislist = ["Apple", "Banana", "cherry"]
[print(x) for x in thislist]
```

Example

```
Fruits =["Apple", "Banana", "cherry"]
newlist =[]
For x in Fruits:
    if "a" in x:
        newlist.append(x).
print(newlist).
```

Syntax:

```
newlist = [expression for item in iterable if condition == True].
```

⑦ sort list :

- The sort() method sort the elements of a given list in a specific ascending or descending order.
- also can use built-in sorted() function for the same purpose.

(i) sort list Alphanumerically :

- list object have a sort method that will sort alphanumerically, ascending by default

```
thislist = ["Apple", "Banana", "Mango", "cherry"].
thislist.sort()
print(thislist).
```

(ii) sort Descending :

- To sort descending use keywords argument reverse = TRUE.

```
thislist.sort(reverse = True)
print(this.list).
```

(ii) customize sort function

- you can customize your own function by using keyword arguments key = function.
- This function will return a number that will used to sort the list

```
def myfunc(n):
    return abs(n-50).
thislist =
thislist.sort(key = myfunc)
print(thislist).
```

(iv) case Insensitive sort :

- By default sort() method is case sensitive.
- all capital letters being sorted before lower letters.

```
thislist.sort()
print(thislist).
```

- so if you want a case-sensitive sort function, use str.lower as key function.

(v) Reverse order :

- The reverse() method reverses the current sorting elements.

```
thislist.reverse()
print(thislist)
```

(8) copy list

- There are different ways to copy() list.
- one way is to built-in list method [copy()].

Example :

```
thislist = ["A", "B", "C"]
mylist = thislist.copy()
print(mylist)
```

- another way to make a copy is use to built-in method list().
- using list() method.

```
mylist = list(thislist).
print(mylist).
```

(9) Join ~~Two~~ List :

(i) Join Two list :

- There are different ways to join, or concatenate, two or more list in Python.
- one of the easiest way are by using + operator.

```
list1 = [ ]
list2 = [ ]
list3 = list1 + list2
print(list3).
```

- another way to join two list are by appending all items from list2 into list1, one by one.

```
For x in list2:
    list1.append(x)
print(list1).
```

- another way is used to extend() method,
- which is use for add one list element into another list elements.

```
list1.extend(list2).
print(list1).
```

(10) list methods :

Sr.No.	Method	Description.
(i)	append()	Add an element at the end of list.
(ii)	clear()	Remove all the elements from list.
(iii)	copy()	Returns the copy of a list.
(iv)	count()	Returns number of elements with specified value.
(v)	extend()	Add the element of a list, to the end of current list.
(vi)	index()	Return the first element with specified value (index of first).
(vii)	insert()	Add an element to a specified location.
(viii)	pop()	Removes element from specified position.
(ix)	remove()	Remove the item with specified value.
(x)	reverse()	Reverse the order of list.
(xi)	sort().	sort list. (ascending or descending).