List Operations in Python

1. append()

The append() method is used to add elements at the end of the list. This method can only add a single element at a time. To add multiple elements, the append() method can be used inside a loop.

```
L1 = [1, 2, 3, ‘Alisha’, ‘learning Python’]
L1.append(4)
L1.append(5)
L1.append(6)
print(L1)
```

2. extend()

The extend() method is used to add more than one element at the end of the list. Although it can add more than one element, unlike append(), it adds them at the end of the list like append().

```
L1.extend([4, 5, 6])
print(L1)
```

3. insert()

The insert() method can add an element at a given position in the list. Thus, unlike append(), it can add elements at any position, but like append(), it can add only one element at a time. This method takes two arguments. The first argument specifies the position, and the second argument specifies the element to be inserted.

```
L1.insert(3, 4)
L1.insert(4, 5)
L1.insert(5, 6)
print(L1)
```

4. remove()

The remove() method is used to remove an element from the list. In the case of multiple occurrences of the same element, only the first occurrence is removed.

```
L1.remove('learning Python')
print(L1)
```

5. pop()

The method pop() can remove an element from any position in the list. The parameter supplied to this method is the index of the element to be removed.

```
L1.pop(4)
print(L1)
```

6. slice

The slice operation is used to print a section of the list. The slice operation returns a specific range of elements. It does not modify the original list.

```
print(L1[:4]) # prints from beginning to end index
print(L1[2:]) # prints from start index to end of list
print(L1[2:4]) # prints from start index to end index
print(L1[:]) # prints from beginning to end of list
```

7. reverse()

The reverse() operation is used to reverse the elements of the list. This method modifies the original list. To reverse a list without modifying the original one, we use the slice operation with negative indices. Specifying negative indices iterates the list from the rear end to the front end of the list.

```
L1.reverse() # modifies the original list
print(L1)
```

8. len()

The len() method returns the length of the list, i.e. the number of elements in the list.

```
print(len(L1))
```

9. min() & max()

The min() method returns the minimum value in the list. The max() method returns the maximum value in the list. Both the methods accept only homogeneous lists, i.e. lists having elements of similar type.

```
print(min([1, 2, 3]))
print(max([1, 2, 3]))
```

10. count()

The function count() returns the number of occurrences of a given element in the list.

```
print(L1.count(3))
```

11. concatenate

The concatenate operation is used to merge two lists and return a single list. The + sign is used to perform the concatenation. Note that the individual lists are not modified, and a new combined list is returned.

```
L2 = [4, 5, 'Python', 'is great'] print(L1+L2)
```

12. multiply

Python also allows multiplying the list *n* times. The resultant list is the original list iterated *n* times.

```
print(L1*2)
```

13. index()

The index() method returns the position of the first occurrence of the given element. It takes two optional parameters – the beginning index and the end index. These parameters define the start and end position of the search area on the list. When supplied, the element is searched only in the sub-list bound by the begin and end indices. When not supplied, the element is searched in the whole list.

```
print(L1.index('Alisha')) # searches in the whole list
print(L1.index('Alisha', 0, 2)) # searches from 0th to 2nd position
```

14. sort()

The sort method sorts the list in ascending order. This operation can only be performed on homogeneous lists, i.e. lists having elements of similar type.

```
L3 = [4, 2, 6, 5, 0, 1]
L3.sort()
print(L2)
```

15. clear()

This function erases all the elements from the list and empties them.

```
L1.clear()
print(L1)
```