Python List insert()
It inserts the element to the list at the given index.

Syntax:
```
list.insert(index, element)
```

Parameters:
index - Position where element needs to be inserted
element - this is the element to be inserted in the list

Example:
```
MyData = ['Big', 'Data', 'Hadoop', 'Spark']
MyData.insert(2, 'TERADATA')
print('Updated List: ', MyData)
```

remove(): Remove an item from the list

Syntax:
```
list.remove(item)
```

Example:
```
py_list=["Big Data", "Hadoop", "Spark", "IoT"]
print(py_list)
py_list.remove("IoT")
print(py_list)
```

Example:
```
# Removing an item from a specific position
Fruit_List = ["Apple", "Banana", "Cherry", "Jackfruit", "Grape"]
del Fruit_List[0]
print(Fruit_List)
```

clear(): Removes all items from the list

Syntax:
```
list.clear()
```

Example:
```
py_list=["Big Data", "Hadoop", "Spark", "IoT"]
print(py_list)
py_list.clear()
print(py_list)  #Empty List Displayed
```

index():
It returns the index in the list of the first item whose value is x.

Syntax:
```
list.index(item[,start][,end])
```

Example:
```
#list.index(item[,start][,end])
PyList=[1,2,3,3,2,1,5,4]
print(PyList)
print(PyList.index(1))
print(PyList.index(3))
print(PyList.index(4))
print(PyList.index(3,3))
print(PyList.index(1,3))
print(PyList.index(2,4))
print(PyList.index(4,4,10))
```

Example:
```
PyList=[1,2,3,1,4,5,2,5,6,1,4]
print(PyList)
print(PyList.index(2))
print(PyList.index(2,2))
print(PyList.index(1,2,7))
print(PyList.index(4,3,9))
```

Example:
```
PyList=["Data","ML","DL","Data","ML"]
print(PyList)#['Data','ML','DL','Data','ML']
print(PyList.index("ML"))
print(PyList.index("Data"))
print(PyList.index("Data",1))
print(PyList.index("ML",2,6))
```

Example:
```
listy = list("HELLO WORLD")
print(listy)
index = listy.index("L")
print(index)
index = listy.index("L", 4)
print(index)
index = listy.index("O", 3, 5)
print(index)
```

count():
It returns the count of number of items passed as an argument

Syntax:
```
list.count(item)
```

Example:
```
PyList=["Big Data", "Hadoop", "Spark", "IoT", "Hadoop"]
print(PyList.count("Hadoop"))
```

Example:
```
PyList=[1,2,3,1,2,3,3,3,1,2,33]
print(PyList.count(2))
print(PyList.count(3))
print(PyList.count(1))
```

sort():Sort the items of the list in place.

Syntax:
```
list.sort(reverse=True|False, key=myFunc)
```

Parameters:

reverse	Optional. reverse=True will sort the list descending. Default is reverse=False
key	Optional. A function to specify the sorting criteria(s)

```
NOTE:sort() doesn't supports mixed data lists..!!

Example:
PyList=["Big Data", "Hadoop", "Spark", "IoT","Big Data"]
print(PyList) #['Big Data','Hadoop','Spark','IoT','Big Data']
PyList.sort()
print(PyList)#['Big Data','Big Data','Hadoop','IoT','Spark']
PyList.sort(reverse=True)
print(PyList)#['Spark','IoT','Hadoop','Big Data','Big Data']

NOTE: reverse - If true, the sorted list is reversed

Example:
PyList=['cc','b','aaa','eeee','ddddddd']
print(PyList)
PyList.sort(key=len)
print(PyList)

Example: Nested List:Sorting on First Value default
PyList=[[1,2],[2,4],[4,5],[3,1]]
print(PyList)
PyList.sort()
print(PyList)

Example:Sorting based on second value..!!
PyList=[[1,2],[2,4],[4,5],[3,1]]
print(PyList)
def SortBySecondValue(item):
    return item[1]
PyList.sort(key=SortBySecondValue)
print(PyList)

reverse(): Reverse the order of items in the list
Syntax:
list.reverse()

Example:
PyList=["Big Data", "Hadoop", "Spark", "IoT"]
print(PyList)
PyList.reverse()
print(PyList)

copy(): Returns a shallow copy of the list

Syntax:
list.copy()

Example:
PyList=[1,2,3,1,2,3,3,3,1,2,33]
print(PyList)
PyList.append(333)
print(PyList)
PyList1=PyList.copy()
print(PyList1)
PyList1.append(444)
print(PyList1)
```

Example:

```
# Copying the list to a different name
Flowers = ["Lotus", "Rose", "Lily", "Sunflower"]
New_List = Flowers.copy()
print(New_List)
```

Using Lists as Stacks
A stack is a container of objects that are inserted and removed according to the last-in first-out (LIFO) principle

pop():
It is used to remove elements from right to left

Syntax:

```
list.pop()
```

Example:

```
py_list=["Big Data", "Hadoop", "Spark", "IoT"]
print(py_list)
py_list.append("DataScinece")
py_list.append("PYTHON")
print(py_list)
py_list.pop()
py_list.pop()
print(py_list)
```

Differences between remove() and pop()
remove()
1) We can use to remove specific element
2) It never return any value.
3) If element not existed returns ' VALUE ERROR'

pop()
1) Remove last element from the List.
2) It will return removed element.
3) If List has no items then we get Error.

Using Lists as Queues: It is a FIFO(First in First Out) structure.
A deque, also known as a double-ended queue, is an ordered collection of items similar to the queue. It is important to note that even though the deque can assume many of the characteristics of stacks and queues, it does not require the LIFO and FIFO orderings that are enforced by those data structures.

Example:

```
from collections import deque
py_list = deque(["Big Data", "Hadoop", "Spark", "IoT"])
py_list.append("DataScinece")
print(py_list)
py_list.append("PYTHON")
print(py_list)
py_list.popleft()
print(py_list)
py_list.popleft()
print(py_list)
```

Convert a list to a tuple in Python

```
MyNumList=[1,2,3,4,5]
print(type(MyNumList))
print(MyNumList)
tup=tuple(MyNumList)
print(tup)
print(type(tup))
```

Compare two lists in Python

```
PyList1=["Data","BigData","Hadoop"]
print(PyList1)
PyList2=["BigData","Hadoop","Data"]
print(PyList2)
print(PyList1==PyList2)
PyList1.sort()
PyList2.sort()
print(PyList1==PyList2)
```

Example:

```
PyList=[]
for i in range(10):
    if i%2==1:
        PyList.append(i)
        print(PyList)
```