Python Lists

List :

Lists are one of 4 built-in data types in python used to store collections of data, the other 3 are tuple, set, dictionary, all with different qualities and usage.

Lists are created using the square brackets :
Example :

```
mylist = [21, "Faheem", 180, "Male"]
print(mylist)
```

```
[21, 'Faheem', 180, 'Male']
```

List Items

List items are ordered, changeable, and allow duplicate values.

Ordered

When we say that lists are ordered, it means that the items have a defined orderd, and that order will not change.

If you add new items to a list, the new items wil be placed at the end of the list.

Changeable

The list is changeable, meaning that we can change, add, and rmove items in a list after it has been created.

Allow Duplicates

since lists are indexed, lists can have items with the same value.

List items - Data Types

List items can be of any data types:

```
#Programme to show the different data types in the list:
#list using the int:
Integers = [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
print(Integers)
print(type(Integers))

#list using the float:
float = [2.3, 5.6, 6.7, 8.9, 8.8, 9.0, 6.5, 4.5, 33.56, 55.678]
print(float)
print(type(float))
```

```
#list using the complex:
complex = [2+3j, 5+6j, 6+7j, 8+9j, 8+8j, 9+0j, 6+5j]
print(complex)
print(type(complex))

#list using the string:
string = ["Faheem", "Ahmed", "Shaik"]
print(string)
print(type(string))

#list using the boolean:
bool = [True, 6<2, False, 4>2, ]
print(bool)
print(type(bool))

#list using the all data types:
mylist = [21, "Faheem", 180.1, 2+3j, 3>5]
print(mylist)


#type of the given input:
print(type(mylist))
```

```
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
<class 'list'>
[2.3, 5.6, 6.7, 8.9, 8.8, 9.0, 6.5, 4.5, 33.56, 55.678]
<class 'list'>
[(2+3j), (5+6j), (6+7j), (8+9j), (8+8j), (9+0j), (6+5j)]
<class 'list'>
['Faheem', 'Ahmed', 'Shaik']
<class 'list'>
[True, False, False, True]
<class 'list'>
[21, 'Faheem', 180.1, (2+3j), False]
<class 'list'>
```

Operations in list:

INDEXING :

Python list index method is used to find position of element in list python. It returns the position of the first occurence of that element in the list. if the item is not found in the list, index function rasises a "value error".

List index method searches for agiven element from the start of the list and the returns the position of the first occurrence.

There are two types of the index;

positive index :

This index starts from the 0 for the first element and 1 for the second element.

The positive index starts from left to right.

Negative index :

This index starts from the "-1" from the last element of the list and "-2" for the second element.

The negative index strarts from the right to left.

```
#accessing the elements using the positive index :
marks = [50, 60, 70, 45, 90, 35, 55]
names = ["suresh", "Ravi", "Ganesh", "Vijay", "Faheem", "kalam",
"Arshi"]
#            0        1         2        3         4         5       6
print(marks[4])
print(names[4])
print(marks[0])
print(names[0])
print(marks[1])
print(names[1])
print(marks[2])
print(names[2])
print(marks[3])
print(names[3])
print(marks[5])
print(names[5])
print(marks[6])
print(names[6])
print(marks)
print(names)
```

```
90
Faheem
50
suresh
60
Ravi
70
Ganesh
45
Vijay
35
kalam
55
Arshi
[50, 60, 70, 45, 90, 35, 55]
['suresh', 'Ravi', 'Ganesh', 'Vijay', 'Faheem', 'kalam', 'Arshi']
```

```
#accessing the elements using the Negative  index :
marks = [50, 60, 70, 45, 90, 35, 55]
names = ["suresh", "Ravi", "Ganesh", "Vijay", "Faheem", "kalam",
"Arshi"]
#           -7        -6       -5       -4        -3       -2       -1
print(marks[-3])
print(names[-3])
print(marks[-4])
print(names[-4])
print(marks[-5])
print(names[-5])
print(marks[-6])
print(names[-6])
print(marks[-7])
print(names[-7])
print(marks[-1])
print(names[-1])
print(marks[-2])
print(names[-2])
print(marks)
print(names)
```

```
90
Faheem
45
Vijay
70
Ganesh
60
Ravi
50
suresh
55
Arshi
35
kalam
[50, 60, 70, 45, 90, 35, 55]
['suresh', 'Ravi', 'Ganesh', 'Vijay', 'Faheem', 'kalam', 'Arshi']
```

SLICING :

In python, list slicing is a common practice and it is the most use technique for programmers to solve efficient problems. consider a python list, in order to access a range of elements in a list, you need to slice a list.

one way to do this is to use the simple slicing operator i.e. colon(:). with this operator, one can specifiy where to start the slicing, where to end, and specify the step. List slicing returns a new list from the existing list.

Python list slicing syntax

The format for list slicing is of python list slicing is as follows:

lst[Initial : End : IndexJump]

```
Intials = ['s', 'k', 'p', 'v', 't', 'h', 'f', 'h', 'j', 'l', 'c', 'd',
'g', 'w', 'm', 'a', 'x', 'z', 'v', 'b']
#           0    1    2    3    4    5    6    7    8    9   10   11
12   13   14   15   16   17   18   19
names = ["shaik", "kolluri", "pasupulati", "vangaveeti", "tippu",
"hesinberg",  "jacob", "linda", "catherine", "dutch", "grefinder",
"wattson", "malaika", "annii", "xavior", "zlatan", "verdict",
"branch", "faheem", "Hutsson"]
#           0        1            2            3            4
5           6       7         8           9         10           11
12          13       14      15        16          17          18
19
#positive slicing
print(Intials[3:19])
print(names[:])
print(Intials[4:])
print(names[:18])
print(Intials[10:18])
print(names[10:])
print(Intials[:19])
print(names[6:9])
print(Intials[10:15])
print(names[10:13])
print(Intials[13:19])
print(names[13:19])
print(Intials[16:19])
print(names[16:19])
print(Intials[19:20])
print(names[19:20])
print(Intials[18:20])
print(names[18:20])
```

```
['v', 't', 'h', 'f', 'h', 'j', 'l', 'c', 'd', 'g', 'w', 'm', 'a', 'x',
'z', 'v']
['shaik', 'kolluri', 'pasupulati', 'vangaveeti', 'tippu', 'hesinberg',
'jacob', 'linda', 'catherine', 'dutch', 'grefinder', 'wattson',
'malaika', 'annii', 'xavior', 'zlatan', 'verdict', 'branch', 'faheem',
'Hutsson']
['t', 'h', 'f', 'h', 'j', 'l', 'c', 'd', 'g', 'w', 'm', 'a', 'x', 'z',
'v', 'b']
['shaik', 'kolluri', 'pasupulati', 'vangaveeti', 'tippu', 'hesinberg',
'jacob', 'linda', 'catherine', 'dutch', 'grefinder', 'wattson',
'malaika', 'annii', 'xavior', 'zlatan', 'verdict', 'branch']
['c', 'd', 'g', 'w', 'm', 'a', 'x', 'z']
['grefinder', 'wattson', 'malaika', 'annii', 'xavior', 'zlatan',
'verdict', 'branch', 'faheem', 'Hutsson']
```

```
['s', 'k', 'p', 'v', 't', 'h', 'f', 'h', 'j', 'l', 'c', 'd', 'g', 'w',
'm', 'a', 'x', 'z', 'v']
['jacob', 'linda', 'catherine']
['c', 'd', 'g', 'w', 'm']
['grefinder', 'wattson', 'malaika']
['w', 'm', 'a', 'x', 'z', 'v']
['annii', 'xavior', 'zlatan', 'verdict', 'branch', 'faheem']
['x', 'z', 'v']
['verdict', 'branch', 'faheem']
['b']
['Hutsson']
['v', 'b']
['faheem', 'Hutsson']

#Negative slicing
#         -18    -17  -16  -15  -14  -13  -12  -11  -10  -9  -8    -7
-6  -5    -4   -3   -2  -1
Intials = ['s', 'k', 'p', 'v', 't', 'h', 'f', 'h', 'j', 'l', 'c', 'd',
'g', 'w', 'm', 'a', 'x', 'z']

#         -20      -19        -18             -17             -16
-15         -14      -13       -12           -11       -10        -9
-8        -7        -6        -5        -4          -3       -2
-1
names = ["shaik", "kolluri", "pasupulati", "vangaveeti", "tippu",
"hesinberg",  "jacob", "linda", "catherine", "dutch", "grefinder",
"wattson", "malaika", "annii", "xavior", "zlatan", "verdict",
"branch", "faheem", "Hutsson"]

print(Intials[-18:-1])
print(names[-18:-1])
print(Intials[-17:-1])
print(names[-17:-1])
print(Intials[-16:-1])
print(names[-20:-6])
print(names[-16:-6])
print(Intials[-15:-4])
print(names[-15:-9])
print(Intials[-14:-5])
print(names[-11:-1])
print(Intials[:-1])
print(Intials[-8:])
print(names[:])
print(Intials[:])

['s', 'k', 'p', 'v', 't', 'h', 'f', 'h', 'j', 'l', 'c', 'd', 'g', 'w',
'm', 'a', 'x']
['pasupulati', 'vangaveeti', 'tippu', 'hesinberg', 'jacob', 'linda',
'catherine', 'dutch', 'grefinder', 'wattson', 'malaika', 'annii',
'xavior', 'zlatan', 'verdict', 'branch', 'faheem']
```

```
['k', 'p', 'v', 't', 'h', 'f', 'h', 'j', 'l', 'c', 'd', 'g', 'w', 'm',
'a', 'x']
['vangaveeti', 'tippu', 'hesinberg', 'jacob', 'linda', 'catherine',
'dutch', 'grefinder', 'wattson', 'malaika', 'annii', 'xavior',
'zlatan', 'verdict', 'branch', 'faheem']
['p', 'v', 't', 'h', 'f', 'h', 'j', 'l', 'c', 'd', 'g', 'w', 'm', 'a',
'x']
['shaik', 'kolluri', 'pasupulati', 'vangaveeti', 'tippu', 'hesinberg',
'jacob', 'linda', 'catherine', 'dutch', 'grefinder', 'wattson',
'malaika', 'annii']
['tippu', 'hesinberg', 'jacob', 'linda', 'catherine', 'dutch',
'grefinder', 'wattson', 'malaika', 'annii']
['v', 't', 'h', 'f', 'h', 'j', 'l', 'c', 'd', 'g', 'w']
['hesinberg', 'jacob', 'linda', 'catherine', 'dutch', 'grefinder']
['t', 'h', 'f', 'h', 'j', 'l', 'c', 'd', 'g']
['dutch', 'grefinder', 'wattson', 'malaika', 'annii', 'xavior',
'zlatan', 'verdict', 'branch', 'faheem']
['s', 'k', 'p', 'v', 't', 'h', 'f', 'h', 'j', 'l', 'c', 'd', 'g', 'w',
'm', 'a', 'x']
['c', 'd', 'g', 'w', 'm', 'a', 'x', 'z']
['shaik', 'kolluri', 'pasupulati', 'vangaveeti', 'tippu', 'hesinberg',
'jacob', 'linda', 'catherine', 'dutch', 'grefinder', 'wattson',
'malaika', 'annii', 'xavior', 'zlatan', 'verdict', 'branch', 'faheem',
'Hutsson']
['s', 'k', 'p', 'v', 't', 'h', 'f', 'h', 'j', 'l', 'c', 'd', 'g', 'w',
'm', 'a', 'x', 'z']
```

Concatenation:

This operation is useful when we have a number of lists of elements that need to be processed in a similar manner.

The most conventional method to perform the list concatintion, the use of "+" operator can easily add the whole of one list behind the other list and hence perform the concatenation.

```
#concatination of two lists into one:
a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
b = [11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
c = a + b
print(c)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
20]
```

```
a = ["Faheem is"]
b = ["21 years old"]
c = a + b
print(c)
```

```
['Faheem is', '21 years old']
```

```
cinema = ["Mirchi", "Devara", "Dhruva", "july",]
Hero = ["prabhas", "Junior N.T.R", "Ramcharan", "Alluarjun"]
cinema_Hero_names = cinema + Hero
print(cinema_Hero_names)
```

```
['Mirchi', 'Devara', 'Dhruva', 'july', 'prabhas', 'Junior N.T.R',
'Ramcharan', 'Alluarjun']
```

```
complex1 = [2+3j, 5+7j, 6+9j, 8+10j]
complex2 = [2+3j, 5+7j, 6+9j, 8+10j]
complex3 = complex1 + complex2
print(complex3)
```

```
[(2+3j), (5+7j), (6+9j), (8+10j), (2+3j), (5+7j), (6+9j), (8+10j)]
```

```
list1 = [1, "Faheem", 6+3j, 2.45, True]
list2 = ["int", "string", "complex", "float", "boolean"]
list3 = list1 + list2
print(list3)
```

```
[1, 'Faheem', (6+3j), 2.45, True, 'int', 'string', 'complex', 'float',
'boolean']
```

Repetition:

You can use the repeat method to acheive the desired result of adding a duplicate element to the list. here is how you can do it:

"*" this is the symbole used for the repetition

```
#Repetition
a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Repetition = a * 2
print(Repetition)

bikes = ["Yamaha", "Honda", "Suzuki", "TVS", "Hero"]
Repetition = bikes * 3
print(Repetition)

a = ["Faheem is a good boy"]
Repetition = a * 5
print(Repetition)

mylist = [23, "yes", 3.45, 2+3j, True]
Repetition = mylist * 4
print(Repetition)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
['Yamaha', 'Honda', 'Suzuki', 'TVS', 'Hero', 'Yamaha', 'Honda',
'Suzuki', 'TVS', 'Hero', 'Yamaha', 'Honda', 'Suzuki', 'TVS', 'Hero']
['Faheem is a good boy', 'Faheem is a good boy', 'Faheem is a good
boy', 'Faheem is a good boy', 'Faheem is a good boy']
[23, 'yes', 3.45, (2+3j), True, 23, 'yes', 3.45, (2+3j), True, 23,
'yes', 3.45, (2+3j), True, 23, 'yes', 3.45, (2+3j), True]
```

Change list items:

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

For Example:

```
#strings

countrys = ["INDIA", "U.S.A", "FAHEEM", "FRANCE", "KUWAIT"]
countrys[2] = "JAPAN"
print(countrys)

#int

int = [1, 2, 3, 4, 11, 6, 7, 8, 9, 10]
print(int)
int[4] = 5
print(int)

#float

float = [2.34, 3.45, 4.56, 5.67, 6.78, 1.01, 9.01, 10.1]
print(float)
float[5] = 7.89
print(float)

#complex

complex = [2+3j, 5+7j, 6+9j, 8+10j]
print(complex)
complex[1] = 3+4j
print(complex)

#boolean

bool = [True, False, True, 345, True, False]
print(bool)
bool[3] = False
print(bool)

['INDIA', 'U.S.A', 'JAPAN', 'FRANCE', 'KUWAIT']
[1, 2, 3, 4, 11, 6, 7, 8, 9, 10]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
[2.34, 3.45, 4.56, 5.67, 6.78, 1.01, 9.01, 10.1]
[2.34, 3.45, 4.56, 5.67, 6.78, 7.89, 9.01, 10.1]
[(2+3j), (5+7j), (6+9j), (8+10j)]
[(2+3j), (3+4j), (6+9j), (8+10j)]
[True, False, True, 345, True, False]
[True, False, True, False, True, False]
```

```
#To change more than one element in the list we use slicing method
#For Example

#strings

countrys = ["INDIA", "U.S.A", "FAHEEM", "Arshi", "Kalam", "Chaitu"
"FRANCE", "KUWAIT"]
print(countrys)
countrys[2:5] = ["JAPAN", "RUSSIA", "DUBAI", "CHINA"]
print(countrys)

#int

int = [1, 2, 3, 4, 11, 12, 13, 14, 9, 10]
print(int)
int[4:7] = [5, 6, 7, 8]
print(int)

#float

float = [2.34, 3.45, 4.56, 5.67, 6.78, 1.01, 92.01, 103.1]
print(float)
float[5:7] = [7.89, 8.90, 9.01]
print(float)

#complex

complex = [2+3j, 5+7j, 6+9j, 8+10j]
print(complex)
complex[1:3] = [3+4j, 4+8j, 5+9j]
print(complex)

#boolean

bool = [True, False, True, 345, 456, 558, 890, 345, 334, 234, True,
False]
print(bool)
bool[3:9] = [False, True, False, True, False, True, False, True]
print(bool)
```

```
['INDIA', 'U.S.A', 'FAHEEM', 'Arshi', 'Kalam', 'ChaituFRANCE',
'KUWAIT']
['INDIA', 'U.S.A', 'JAPAN', 'RUSSIA', 'DUBAI', 'CHINA',
'ChaituFRANCE', 'KUWAIT']
[1, 2, 3, 4, 11, 12, 13, 14, 9, 10]
[1, 2, 3, 4, 5, 6, 7, 8, 14, 9, 10]
[2.34, 3.45, 4.56, 5.67, 6.78, 1.01, 92.01, 103.1]
[2.34, 3.45, 4.56, 5.67, 6.78, 7.89, 8.9, 9.01, 103.1]
[(2+3j), (5+7j), (6+9j), (8+10j)]
[(2+3j), (3+4j), (4+8j), (5+9j), (8+10j)]
[True, False, True, 345, 456, 558, 890, 345, 334, 234, True, False]
[True, False, True, False, True, False, True, False, True, False,
True, 234, True, False]
```

List Methods

Python has a set of built - in methods that you can use on lists/arrays.

Append() :

Adds an element at the end of the list.

```
#append()

#string
string = ["Faheem", "Ahmed", "Shaik"]
string.append("Male")
print(string)

#int
int = [1,2,3,4,5,6,7,8,9]
int.append(10)
print(int)

#float
float = [1.23, 2.34, 3.45, 4.56, 5.67, 6.78]
float.append(7.89)
print(float)

#complex
complex = [1+2j, 2+3j, 3+4j, 4+5j]
complex.append(5+6j)
print(complex)

#boolean
bool = [True, False, True, False, True, False]
bool.append(True)
print(bool)
```

```
['Faheem', 'Ahmed', 'Shaik', 'Male']
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1.23, 2.34, 3.45, 4.56, 5.67, 6.78, 7.89]
[(1+2j), (2+3j), (3+4j), (4+5j), (5+6j)]
[True, False, True, False, True, False, True]
```

Clear() :

Removes all the elements from the list.

```
#Clear()

#string
string = ["Faheem", "Ahmed", "Shaik"]
string.clear()
print(string)

#int
int = [1,2,3,4,5,6,7,8,9,10]
int.clear()
print(int)

#float
float = [1.23, 2.34, 3.445, 4.567, 5.667, 6.778, 7.89]
float.clear()
print(float)

#complex
complex = [1+2j, 2+3j, 3+4j, 4+5j]
complex.clear()
print(complex)

#boolean
bool = [True, False, True, False, True, False]
bool.clear()
print(bool)
```

```
[]
[]
[]
[]
[]
```

Copy() :

Returns a copy of the list

```
#copy()

#strings
```

```
strings = ["Faheem", "Ahmed", "Shaik"]
string1 = strings.copy()
print(strings)
print(string1)

#int
int = [1,2,3,4,5,6,7,8,9,10]
int1 = int.copy()
print(int)
print(int1)

#float
float = [1.23, 2.34, 3.45, 4.56, 5.67, 6.78]
float1 = float.copy()
print(float)
print(float1)

#complex
complex = [1+2j, 2+3j, 3+4j, 4+5j]
complex1 = complex.copy()
print(complex)
print(complex1)

#boolean
bool = [True, False, True, False, True, False]
bool1 = bool.copy()
print(bool)
print(bool1)
```

```
['Faheem', 'Ahmed', 'Shaik']
['Faheem', 'Ahmed', 'Shaik']
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1.23, 2.34, 3.45, 4.56, 5.67, 6.78]
[1.23, 2.34, 3.45, 4.56, 5.67, 6.78]
[(1+2j), (2+3j), (3+4j), (4+5j)]
[(1+2j), (2+3j), (3+4j), (4+5j)]
[True, False, True, False, True, False]
[True, False, True, False, True, False]
```

Count() :

Returns the number of elements with the specified value.

```
#Count()

#strings
strings = ["Shaik","Faheem", "Ahmed", "Shaik"]
count = strings.count("Shaik")
print(count)
```

```
#int
int = [1,2,3,4,5,3,7,8,6,10,1,2,3,6,5,6,7,8,9,10]
count = int.count(6)
print(count)

#float
float = [1.23, 2.34, 3.45, 4.56, 5.67, 6.78, 6.78, 6.78,6.78]
count = float.count(6.78)
print(count)

#complex
complex = [1+2j, 2+3j, 3+4j, 4+5j, 2+3j, 3+4j, 2+3j, 5+4j, 2+3j, 6+6j,
2+3j]
count = complex.count(2+3j)
print(count)

#boolean
bool = [True, False, True, False, True, False, True, False, True,
False, True, False]
count = bool.count(True)
print(count)
```

```
2
3
4
5
6
```

Extend() :

Add the elements of a list (or any iterale), to the end of the current list.

```
#Extend()

#strings
strings = ["Shaik","Faheem", "Ahmed"]
string1 = ["Male"]
strings.extend(string1)
print(strings)

#int
a = [1,2,3,4,5]
b = [6.7,8,9,10]
a.extend(b)
print(a)

#float
float = [1.23, 2.34, 3.45, 4.56, 5.67, 6.78]
float1 = [7.89]
```

```
float.extend(float1)
print(float)

#complex
complex = [1+2j, 2+3j, 3+4j, 4+5j]
complex1 = [5+6j]
complex.extend(complex1)
print(complex)

#boolean
bool = [True, False, True, False, True, False]
bool1 = [True]
bool.extend(bool1)
print(bool)
```

```
['Shaik', 'Faheem', 'Ahmed', 'Male']
[1, 2, 3, 4, 5, 6.7, 8, 9, 10]
[1.23, 2.34, 3.45, 4.56, 5.67, 6.78, 7.89]
[(1+2j), (2+3j), (3+4j), (4+5j), (5+6j)]
[True, False, True, False, True, False, True]
```

Index() :

Returns the index of the first element with the specified value.

```
#Index()

#strings
#            0         1        2        3
strings = ["Shaik","Faheem", "Ahmed", "Male"]
index = strings.index("Male")
print(index)

#int
int = [1,2,3,4,5,6,7,8,9,10]
index = int.index(5)
print(index)
```

```
3
4
```

Insert() :

Adds an element at the specified position.

```
#insert()

#strings
#             0        1        2
strings = ["Shaik","Faheem", "Ahmed"]
```

```
strings.insert(3, "Male")
print(strings)

#int
int = [1,2,3,4,5,6,7,8,9,10]
int.insert(5, 11)
print(int)

#float
float = [1.23, 2.34, 3.45, 4.56, 5.67, 6.78]
float.insert(3, 7.89)
print(float)

#complex
complex = [1+2j, 2+3j, 3+4j, 4+5j]
complex.insert(2, 5+6j)
print(complex)

#boolean
bool = [True, False, True, False, True, False]
bool.insert(4, True)
print(bool)
```

```
['Shaik', 'Faheem', 'Ahmed', 'Male']
[1, 2, 3, 4, 5, 11, 6, 7, 8, 9, 10]
[1.23, 2.34, 3.45, 7.89, 4.56, 5.67, 6.78]
[(1+2j), (2+3j), (5+6j), (3+4j), (4+5j)]
[True, False, True, False, True, True, False]
```

pop() :

Removes the element at the specified position

```
#pop()

#strings
#              0        1        2        3
strings = ["Shaik","Faheem", "Ahmed", "Male"]
strings.pop(3)
print(strings)

#int
int = [1,2,3,4,5,6,7,8,9,10]
int.pop(5)
print(int)

#float
float = [1.23, 2.34, 3.45, 4.56, 5.67, 6.78]
float.pop(3)
print(float)
```

```
#complex
complex = [1+2j, 2+3j, 3+4j, 4+5j]
complex.pop(2)
print(complex)

#boolean
bool = [True, False, True, False, True, False]
bool.pop(4)
print(bool)
```

```
['Shaik', 'Faheem', 'Ahmed']
[1, 2, 3, 4, 5, 7, 8, 9, 10]
[1.23, 2.34, 3.45, 5.67, 6.78]
[(1+2j), (2+3j), (4+5j)]
[True, False, True, False, False]
```

remove() :

Removes the first item with the specified value.

```
#remove()

#string
string = ["Shaik","Faheem", "Ahmed", "Male", "Shaik"]
string.remove("Shaik")
print(string)

#int
int = [1,2,3,4,5,6,7,8,9,10,3]
int.remove(3)
print(int)

#float
float = [1.23, 2.34, 3.45, 4.56, 5.67, 6.78, 3.45]
float.remove(3.45)
print(float)

#complex
complex = [1+2j, 2+3j, 3+4j, 4+5j, 2+3j]
complex.remove(2+3j)
print(complex)

#boolean
bool = [True, False, True, False, True, False, True, False]
bool.remove(True)
print(bool)
```

```
['Faheem', 'Ahmed', 'Male', 'Shaik']
[1, 2, 4, 5, 6, 7, 8, 9, 10, 3]
```

```
[1.23, 2.34, 4.56, 5.67, 6.78, 3.45]
[(1+2j), (3+4j), (4+5j), (2+3j)]
[False, True, False, True, False, True, False]
```

reverse() :

Reverse the order of the list

```
#reverse()

#string
string = ["Shaik","Faheem", "Ahmed", "Male"]
string.reverse()
print(string)

#int
int = [1,2,3,4,5,6,7,8,9,10]
int.reverse()
print(int)

#float
float = [1.23, 2.34, 3.45, 4.56, 5.67, 6.78]
float.reverse()
print(float)

#complex
complex = [1+2j, 2+3j, 3+4j, 4+5j]
complex.reverse()
print(complex)

#boolean
bool = [True, False, True, False, True, False]
bool.reverse()
```

```
['Male', 'Ahmed', 'Faheem', 'Shaik']
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
[6.78, 5.67, 4.56, 3.45, 2.34, 1.23]
[(4+5j), (3+4j), (2+3j), (1+2j)]
```

sort() :

Sorts the list

```
#sort()

#strings
string = ["Faheem", "Ahmed", "shaik", "Male"]
string.sort()
print(string)
```

```
#int
int = [1,2,7,4,10,6,3,8,9,5]
int.sort()
print(int)

#float
float = [1.23, 5.34, 3.45, 4.56, 2.67, 6.78]
float.sort()
print(float)

#boolean
bool = [True, False, True, False, True, False]
bool.sort()
print(bool)
```

```
['Ahmed', 'Faheem', 'Male', 'shaik']
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1.23, 2.67, 3.45, 4.56, 5.34, 6.78]
[False, False, False, True, True, True]
```

len() : Returns the length of the length

```
#len()

#string
string = ["Shaik","Faheem", "Ahmed", "Male"]
length = len(string)
print(length)

#int
int = [1,2,3,4,5,6,7,8,9,10]
length = len(int)
print(length)

#float
float = [1.23, 2.34, 3.45, 4.56, 5.67, 6.78]
length = len(float)
print(length)

#complex
complex = [1+2j, 2+3j, 3+4j, 4+5j]
length = len(complex)
print(length)

#boolean
bool = [True, False, True, False, True, False]
length = len(bool)
print(length)
```

```
4
10
6
4
6
```

min() :

Returns the minimum elements in a list

```
#min()

#strings
strings = ["Shaik","Faheem", "Ahmed", "Male"]
minimum = min(strings)
print(minimum)

#int
int = [1,2,3,4,5,6,7,8,9,10]
minimum = min(int)
print(minimum)

#boolean
bool = [True, False, True, False, True, False]
minimum = min(bool)
print(minimum)
```

```
Ahmed
1
False
```

max() :

Returns the maximum element in a list

```
#max()

#strings
strings = ["Shaik","Faheem", "Ahmed", "Male"]
maximum = max(strings)
print(maximum)

#int
int = [1,2,3,4,5,6,7,8,9,10]
maximum = max(int)
print(maximum)

#boolean
bool = [True, False, True, False, True, False]
```

```
maximum = max(bool)
print(maximum)
```

```
Shaik
10
True
```

del() :

Delet the entire list

```
#del()

#string
string1 = ["Shaik","Faheem", "Ahmed", "Male"]
del(string1)
print(string1)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call
last)
<ipython-input-64-ac5dea0fca14> in <cell line: 6>()
      4 string1 = ["Shaik","Faheem", "Ahmed", "Male"]
      5 del(string1)
----> 6 print(string1)

NameError: name 'string1' is not defined
```

```
#del() using the index method

#string
string = ["Shaik","Faheem", "Ahmed", "Male"]
del(string[0])
print(string)

#int
int = [1,2,3,4,5,6,7,8,9,10]
del(int[0])
print(int)

#float
float = [1.23, 2.34, 3.45, 4.56, 5.67, 6.78]
del(float[3])
print(float)

#complex
complex = [1+2j, 2+3j, 3+4j, 4+5j]
del(complex[2])
print(complex)
```

```python
#boolean
bool = [True, False, True, False, True, False]
del(bool[4])
print(bool)
```

```
['Faheem', 'Ahmed', 'Male']
[2, 3, 4, 5, 6, 7, 8, 9, 10]
[1.23, 2.34, 3.45, 5.67, 6.78]
[(1+2j), (2+3j), (4+5j)]
[True, False, True, False, False]
```