Python Lists

```
mylist = ["apple", "banana", "cherry"]
```

List

Lists are used to store multiple items in a single variable.

Lists are one of 4 built-in data types in Python used to store collections of data, the other 3 are Tuple, Set, and Dictionary, all with different qualities and usage.

Lists are created using square brackets:

Create a List:

```
thislist = ["apple", "banana", "cherry"]
print(thislist)
```

List Items

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

List items are indexed, the first item has index [0], the second item has index [1] etc.

Ordered

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

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

Note: There are some list methods that will change the order, but in general: the order of the items will not change.

Changeable

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

Allow Duplicates

Since lists are indexed, lists can have items with the same value:

Lists allow duplicate values:

```
thislist = ["apple", "banana", "cherry", "apple", "cherry"]
print(thislist)
```

List Length

To determine how many items a list has, use the len() function:

Print the number of items in the list:

```
thislist = ["apple", "banana", "cherry"]
print(len(thislist))
```

List Items - Data Types
List items can be of any data type:

String, int and boolean data types:

```
list1 = ["apple", "banana", "cherry"]
list2 = [1, 5, 7, 9, 3]
list3 = [True, False, False]
```

A list can contain different data types:

A list with strings, integers and boolean values:

```
list1 = ["abc", 34, True, 40, "male"]
```

type()
From Python's perspective, lists are defined as objects with the data type 'list':

<class 'list'>
ExampleGet your own Python Server
What is the data type of a list?

```
mylist = ["apple", "banana", "cherry"]
print(type(mylist))
```

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

ExampleGet your own Python Server
Using the list() constructor to make a List:

```
thislist = list(("apple", "banana", "cherry")) # note the double round-brackets
print(thislist)
```

List methods in Python

Python List Methodshas multiple methods to work with Python lists, Below we've explained all the methods you can use with Python lists, for example, append(), copy(), insert() and more.

List Methods

S.no	Method	Description

1	append()	Used for appending and adding elements to the end of the List.
2	copy()	It returns a shallow copy of a list
3	clear()	This method is used for removing all items from the list.
4	count()	This methods count the elements
5	extend()	Adds each element of the iterable to the end of the List
6	index()	Returns the lowest index where the element appears.
7	insert()	Inserts a given element at a given index in a list.
8	pop()	Removes and returns the last value from the List or the given index value.
9	remove()	Removes a given object from the List.
10	reverse()	Reverses objects of the List in place.
11	sort()	Sort a List in ascending, descending, or user-defined order
12	min()	Calculates minimum of all the elements of List
13	max()	Calculates maximum of all the elements of List

Adding Element in List
append()
Used for appending and adding elements to List. It is used to add elements to the last position of the List in Python.

Syntax:

list.append (element)

# Adds List Element as value of List.
List = ['Mathematics', 'chemistry', 1997, 2000]
List.append(20544)
print(List)
Output:

['Mathematics', 'chemistry', 1997, 2000, 20544]
insert()
Inserts an element at the specified position.

Syntax:

list.insert(<position, element)

Note: Position mentioned should be within the range of List, as in this case between 0 and 4, elsewise would throw IndexError.

List = ['Mathematics', 'chemistry', 1997, 2000]
# Insert at index 2 value 10087

```
List.insert(2,10087)
print(List)
```
Output:

```
['Mathematics', 'chemistry', 10087, 1997, 2000, 20544]
```
extend()
Adds contents to List2 to the end of List1.

Syntax

```
List1.extend(List2)
```

```
List1 = [1, 2, 3]
List2 = [2, 3, 4, 5]

# Add List2 to List1
List1.extend(List2)
print(List1)

# Add List1 to List2 now
List2.extend(List1)
print(List2)
```
Output:

```
[1, 2, 3, 2, 3, 4, 5]
[2, 3, 4, 5, 1, 2, 3, 2, 3, 4, 5]
```
Other functions of List
sum()
Calculates the sum of all the elements of the List.

Syntax

```
sum(List)
```

```
List = [1, 2, 3, 4, 5]
print(sum(List))
```
Output:

```
15
```
What happens if a numeric value is not used a parameter?

The sum is calculated only for Numeric values, elsewise throws TypeError.

See example:

```
List = ['gfg', 'abc', 3]
print(sum(List))
```

Output:

```
Traceback (most recent call last):
  File "", line 1, in
    sum(List)
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

count()
Calculates total occurrence of a given element of List.

Syntax:

```
List.count(element)
```

```
List = [1, 2, 3, 1, 2, 1, 2, 3, 2, 1]
print(List.count(1))
```

Output:

```
4
```

length
Calculates the total length of the List.

Syntax:

```
len(list_name)
```

```
List = [1, 2, 3, 1, 2, 1, 2, 3, 2, 1]
print(len(List))
```

Output:

```
10
```

index()
Returns the index of the first occurrence. The start and End index are not necessary parameters.

Syntax:

List.index(element[,start[,end]])

```
List = [1, 2, 3, 1, 2, 1, 2, 3, 2, 1]
print(List.index(2))
```

Output:

1

Another example:

```
List = [1, 2, 3, 1, 2, 1, 2, 3, 2, 1]
print(List.index(2,2))
```

Output:

4

min()

Calculates minimum of all the elements of List.

Syntax:

min(iterable, *iterables[, key])

```
numbers = [5, 2, 8, 1, 9]
print(min(numbers))
```

Output

1

max()

Calculates maximum of all the elements of List.

Syntax:

max(iterable, *iterables[, key])

```
numbers = [5, 2, 8, 1, 9]
print(max(numbers))
```

Output

9

Syntax:

Deletion of List Elements

To Delete one or more elements, i.e. remove an element, many built-in functions can be used, such as pop() & remove() and keywords such as del.

pop()
The index is not a necessary parameter, if not mentioned takes the last index.

Syntax:

list.pop([index])

Note: Index must be in range of the List, elsewise IndexErrors occurs.

List = [2.3, 4.445, 3, 5.33, 1.054, 2.5]
print(List.pop())
Output:

2.5

List = [2.3, 4.445, 3, 5.33, 1.054, 2.5]
print(List.pop(0))
Output:

2.3
del()
Element to be deleted is mentioned using list name and index.

Syntax:

del list.[index]

List = [2.3, 4.445, 3, 5.33, 1.054, 2.5]
del List[0]
print(List)
Output:

[4.445, 3, 5.33, 1.054, 2.5]
remove()
Element to be deleted is mentioned using list name and element.

Syntax

list.remove(element)

```
List = [2.3, 4.445, 3, 5.33, 1.054, 2.5]
List.remove(3)
print(List)
```
Output:

```
[2.3, 4.445, 5.33, 1.054, 2.5]
```