Python Lists

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

. Lists are created using square brackets:

Example:

```
letters = ["a", "b", "c"]
print (letters ) # ['a', 'b', 'c']
```

List Items

List items are:

. Ordered: items have defined order, and that order will not change.
(There are some list methods that will change the order)

. Changeable: The list is changeable, we can change, add, and remove items.

. Allow Duplicates: Since lists are indexed, they can have items with same value

. Indexed: The first item has index [0], the second item has index [1] etc.

List Length

To determine how many items has a list, use the 'len()' func

```
Letters = ["a", "b", "c"]
print ( len (Letters)) # Output : 3
```

List items - Data Types

List items can be of any data type.
A List can also contain different data types

Example:

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

Access List Items

We can access items in lists using their index

Example:

```
Letters = ["a", "b", "c"]
print (Letters[1]) # output : b
```

- negative indexing: Start from the end of the list
  -1: last item

Example:

```
Letters = ['a', 'b', 'c']
print (Letters[-2]) # output : b
```

- Range of Indexes:
- We can specify a range of indexes by specifying where to start and where to end the range
- when specifying a range, the return value will be a new list with the specified items.

Example:

```
Letters = ['a', 'b', 'c', 'd', 'e', 'f']
print (Letters[2 : 5] # output: ['c', 'd', 'e']
```

2: included, 5: not included

- If we don't specify the start value, the range will start at the first item

Example:

```
Letters = ['a', 'b', 'c', 'd', 'e', 'f']
print (Letters[:4]) # output: ['a', 'b', 'c', 'd']
```

- Also, if we don't specify the end value, the range will go on to the end of the list

Example:

```
Letters = ['a', 'b', 'c', 'd', 'e', 'f']
print (Letters[2:]) # output: ['c', 'd', 'e', 'f']
```

↳ Check if an item exist

To determine if a specified item is present in a list we use the "in" keyword:

Example:

```
Letters = ['a', 'b', 'c']
if 'a' in Letters:
  print ("Yes, 'a' is in the list")
# output: Yes, 'a' is in the list
```

↳ Change item value

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

```
List = [31, 22, 52, 60]
List[1] = 15
print (List) # output [31, 15, 52, 60]
```

↳ Insert Items

To insert a new item in the list at a specific index, we can use the 'insert()' method

Example:

```
List = ['a', 'b', 'd', 'e']
List.insert(2, 'c')   # output: ['a', 'b', 'c', 'd', 'e']
print(List)
```

(2 → index; 'c' → item)

↳ Append Items

To add an item to the end of the list, use the 'append()' method

Example:

```
List = ['a', 'b', 'c', 'd']
List.append('e')
print(List) # output: ['a', 'b', 'c', 'd', 'e']
```

↳ Extend List

To append elements from another list to the current list, we use the 'extend()' method

Example:

```
List1 = ['a', 'b', 'c', 'd']
List2 = ['e', 'f', 'g']
List1.extend(List2)
print(List1) # output: ['a', 'b', 'c', 'd', 'e', 'f', 'g']
```

↳ Remove Specified Item

The 'remove()' method removes the specified item

Example:

```
List = [31, 52, 63, 4]
List.remove(52)
print(List) # Output: [31, 63, 4]
```

!! If there more than item with the specified value, the 'remove()' method removes the first occurance only.

↳ Remove Specified Index

The 'pop()' method removes the item in a specified index

Example

```
List = [55, 81, 910, 4]
       0   1   2    3
List.pop(1)
print(List) # Output: [55, 910, 4]
```

!! If we do not specify the index, the pop() method removes the last item

↳ Clear the List

The 'clear()' method emptics the list,

The list still remains but without any content.

```
List = [1, 2, 3, 4]
List.clear()
print(List) # Output: []
```

↳ Loop Through a List

We can loop through the list items by using a for loop

Example:

```
List = [1, 2, 3, 4]
for x in List:
    print(x)    # output: 1
                          2
                          3
                          4
```

↳ Sort Lists

List objects have a sort() method that will sort the list alphanumerically, ascending, by default:

Examples:

1)
```
List = ["orange", "mango", "pineapple", "peach"]
List.sort()
print(List) # output: ["mango", "orange", "peach", "pineapple"]
```

2)
```
List = [100, 50, 65, 82, 23]
List.sort()
print(List) # output: [23, 50, 65, 82, 100]
```

. To sort the list in descending order, use reverse = True

```
List = ["orange", "mango", "pineapple", "peach"]
List.sort(reverse = True)
print(List) # output: ["pineapple", "peach", "orange", "mango"]
```

↳ Copy a List

. we cannot copy a list by typing list1 = list2, because list2 will only be a reference to list1 and changes made in list1 will automatically be made in list2

. So if we want to copy a list properly we can use the method 'copy()'

Example:

```
List1 = [1,2,3,4]
List2 = List1.copy()
print(List2) # Output: [1,2,3,4]
```

. or we can use the method list()

```
List2 = list(List1)
print(List2) # Output: [1,2,3,4]
```

↳ Join Lists

. we can join lists in several ways:

1. Using the 'extend()' method that we have seen before
2. Using the + operator

```
list1 = ['a', 'b', 'c']
list2 = [1, 2, 3]
list3 = list1 + list2
print(list3) # Output: ['a', 'b', 'c', 1, 2, 3]
```

→ List Methods

. Python has a set of built-in methods that we can use on lists

. We have seen several methods such as:

append()	pop()
clear()	remove()
copy()	reverse()
extend()	sort()
insert()	

. There are a few more methods like:

count(): Returns the nb of elements with the specified value

index(): Returns the index of the first element with the specified value.

→ Python If... Else

Python supports the usual logical conditions from mathematics:

- Equals: a == b
- Not Equals: a != b
- Less than: a < b
- Less than or equal to: a <= b
- Greater than: a > b
- Greater than or equal to: a >= b