Today I'll Cover :

1. List Data Structure
2. Creation of List
3. Accessing of List
4. Traversing in List
5. Methods of List
6. Operator that can be applied on list.
7. Nested List & List Comprehension

Mohammad Altaf Hussain

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

In the name of ALLAH,
the Most Beneficent, the Most Merciful

Mohammad Altaf Hussain

List []

If we want to represent a group of individual objects as a single entity where :-

- ✓ insertion order preserved.
- ✓ duplicate objects are allowed
- ✓ heterogeneous objects are allowed
- ✓ Modification are allowed, once object is created.

then we should go for List. The elements are placed within square brackets and with comma separator.

Creation of List

❑ Empty List:-

1. []
2. list()

❑ List with element:-

```
[ element1, element2, elementN  ]
```

▪ list() function used to cast other data type into list type.

Accessing of List

- ❑ By using indexing we can access single element :-
- +(ve) index means left to right (forward).
- -(ve) index means right to left (backward).
- Indexing start from 0 in forward direction -1 in backward direction.
- If we are trying to access characters of a string with out of range index then we will get IndexError.

- **By using List slicing, we can access one or more elements.**

<u>**Syntax**</u> **:** **any_list [bIndex:eIndex:step]**

Where,

bIndex: slice has to start.

eIndex: slice has to end.

step: increment value.

Note :-

- Default value of begin index is 0.
- Default value of ending index is max allowed index of list ie length of the list.
- Default value for step is 1.

Mohammad Altaf Hussain

Note :-

- if step value is +ve then it should be traverse in forward direction(left to right) and we have to consider begin to end -1.
- if step value is -ve then it should be traverse in backward direction(right to left) and we have to consider begin to end +1

Traversing in List

We can traverse in list using:-

- ❑ For loop
- ❑ While loop

Methods of List

index() :- It return the index of the specified element. If element is not available then we will get ValueError. So first We should check the availability of element in the list.

Syntax: any_list.index(element, begin, end)

Where, **begin** and **end** indicates beginning index position and ending index position respectively. It is optional.

Mohammad Altaf Hussain

❖ Counting element in given List

- **count() :-** It is used to find the number occurrence of specified element.

Syntax: any_list.count(element)

❖ Manipulation of List.

- **append()** :- It is used to add element at the end of list.

Syntax: any_list.append(element)

- **insert()** :- It is used to insert item at specified index position.

Syntax: any_list.insert(index,element)

- **extend() :-** It is used to add all items of one list to another list.

Syntax: another_list.extend(one_list)

- **remove() :-** It is used to remove specified item from the list. If the item present multiple times then only first occurrence will be removed. If the specified item not present in list then we will get ValueError.

Syntax: any_list.remove(element)

Mohammad Altaf Hussain

- **pop()** :- It is used to removes and returns the element based on the index.
- Index is optional, if we do not provide index then it remove the last element.
- If the list is empty then pop() function raises IndexError.

Syntax: any_list.pop(index)

<u>Note:-</u> generally append() and pop() functions use to implement stack data structure by using list, which follows LIFO(Last In First Out) order.

- **reverse() :-** It is used to use to reverse the order of elements of list.

Syntax: any_list.reverse()

- **sort() :-** It is used to sort the element of list based on ascending order. To use sort() function, compulsory list should contain only homogeneous elements otherwise we will get TypeError. To sort in descending order use reverse=True argument.

Syntax: any_list.sort()

Mohammad Altaf Hussain

- **Clear()** :- It is used to remove all elements of List.

Syntax : any_list.clear()

Mohammad Altaf Hussain

❖ Aliasing & Cloning of List.

- The process of giving another reference variable to the existing list is called **aliasing.**
- By using one reference variable if we are changing content, then those changes will be reflected to the other reference variable.
- We can create alias with the help of = operator.

- The process of creating exactly duplicate independent object is called **cloning.**

❖ **Cloning list using slice operator**

syntax: new_list = older_list[:]

❖ **Cloning list using copy() method**

syntax: new_list = older_list.copy()

Operator that can be applied on List

+			Concatenation		
*			Repetition		
<	<=	==	!=	>=	>
in	not in		Membership		

1. To use + operator for List, compulsory both arguments should be list type
2. To use * operator for List, compulsory one argument should be list and other argument should be int
3. We can compare list, comparison is based on how the letters are arranged in dictionary that we use in real life.

Mohammad Altaf Hussain

Nested Lists

- List within list is known as nested list.
- We can access nested list elements by using index just like accessing multi dimensional array elements.

List Comprehension

- It is very easy and compact way of creating list objects from any iterable objects(like list,tuple,dictionary,range etc) based on some condition.

Syntax:-

```
list=[expression for item in list if condition]
```