Lists

- Python **Lists** are just like dynamically sized arrays, declared in other languages.
- The list is a sequence data type which is used to store the collection of data.
- A single list may contain DataTypes like Integers, Strings, as well as Objects.

Example:

```
Var = ["Topper", "World"]

print(Var)
```

Output:

```
["Topper", "World"]
```

❖ Characteristics of Lists

The characteristics of the List are as follows:

- The lists are in order.
- The list element can be accessed via the index.
- The mutable type of List is
- The rundowns are changeable sorts.
- The number of various elements can be stored in a list.

❖ Creating a List in Python

- Lists in Python can be created by just placing the sequence inside the square brackets[].
- A list doesn’t need a built-in function for its creation of a list.

Example:

```
# Python program to demonstrate Creation of List


# Creating a List
List = []
print("Blank List: ")
print(List)


# Creating a List of numbers
List = [10, 20, 14]
print("\nList of numbers: ")
print(List)


# Creating a List of strings and accessing  using index
List = ["Topper", "World"]
print("\nList Items: ")
print(List[0])
print(List[2])
```

Output:

```
Blank List:
[]

List of numbers:
[10, 20, 14]

List Items:
Topper
World
```

❖ Accessing elements from the List

- In order to access the list items refer to the index number. Use the index operator [ ] to access an item in a list.
- The index must be an integer. Nested lists are accessed using nested indexing.

Example:

```
# Python program to demonstrate  accessing of element from list

# Creating a List with the use of multiple values
List = ["Topper", "World"]

# accessing a element from the list using index number
print("Accessing a element from the list")
print(List[0])
```

Output:

```
Accessing a element from the list

Topper
```

❖ Adding Elements to a Python List

Method 1: Using insert() method

Append() method only works for the addition of elements at the end of the List, for the addition of elements at the desired position, insert() method is used.

Example:

```
# Python program to demonstrate

# Addition of elements in a List


# Creating a List

List = [1,2,3,4]

print("Initial List: ")

print(List)


# Addition of Element at specific Position

# (using Insert Method)

List.insert(3, 12)

List.insert(0, 'Topper')

print("\nList after performing Insert Operation: ")

print(List)
```

Output:

```
Initial List:
[1, 2, 3, 4]


List after performing Insert Operation:
['Topper', 1, 2, 3, 12, 4]
```

Method 2: Using append() method

Elements can be added to the List by using the built-in append() function. Only one element at a time can be added to the list by using the append() method, for the addition of multiple elements with the append() method, loops are used.

Example:

```
# Python program to demonstrate Addition of elements in a List
# Creating a List
List = []
print("Initial blank List: ")
print(List)


# Addition of Element  in the List
List.append(1)
List.append(2)
List.append(4)
print("\nList after Addition of Three elements: ")
print(List)
```

Output:

```
Initial blank List:
[]


List after Addition of Three elements:
[1, 2, 4]
```

❖ Slicing of a List

- We can get substrings and sublists using a slice. In Python List, there are multiple ways to print the whole list with all the elements, but to print a specific range of elements from the list, we use the Slice operation.
- Slice operation is performed on Lists with the use of a colon(:).

To print elements from beginning to a range use:

```
[: Index]
```

To print elements from end-use:

```
[:-Index]
```

To print elements from a specific Index till the end use

```
[Index:]
```

To print the whole list in reverse order, use

```
[::-1]
```

❖ List Comprehension

- Python List comprehensions are used for creating new lists from other iterables like tuples, strings, arrays, lists, etc.
- A list comprehens ion consists of brackets containing the expression, which is executed for each element along with the for loop to iterate over each element.

Syntax:

```
newList = [ expression(element) for element in oldList if
condition ]
```

Example:

```
# for understanding, above generation is same as,
odd_square = []


for x in range(1, 11):
    if x % 2 == 1:
        odd_square.append(x**2)


print(odd_square)
```

Output:

```
[1, 9, 25, 49, 81]
```