**Intro To List**

-- **Python Lists** are just like dynamically sized arrays.

-- A list is a collection of things, enclosed in [ ] and separated by commas.

-- The list is a sequence data type which is used to store the collection of data.

Var = ["Chandlar", "Joey", "Ross"]

-- Lists are the simplest containers that are an integral part of the Python language.

-- Lists need not be homogeneous always which makes it the most powerful tool in Python.

-- A single list may contain DataTypes like Integers, Strings, as well as Objects.

-- Lists are mutable, and hence, they can be altered even after their creation.

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

# *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 = ["Monica", "Rachel", "Pheebe"]

print("\nList Items: ")

print(List[0])

print(List[2])

-- A list may contain duplicate values with their distinct positions and hence, multiple distinct or duplicate values can be passed as a sequence at the time of list creation.

# *Creating a List with the use of Numbers (Having duplicate values)*

List = [1, 2, 4, 4, 3, 3, 3, 6, 5]

print("\nList with the use of Numbers: ")

print(List)

# *Creating a List with mixed type of values (Having numbers and strings)*

List = [1, 2, 'The 100', 4, 'For', 6, 'The Office']

print("\nList with the use of Mixed Values: ")

print(List)

**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.

# *Creating a List with the use of multiple values*

List = ["Lagaan", "Vastav", "Tumbaad"]

# *accessing a element from the*

# *list using index number*

print("Accessing a element from the list")

print(List[0])

print(List[2])

# *Creating a Multi-Dimensional List (By Nesting a list inside a List)*

List = [['The Last Czar', 'Sense and Sensibility'], ['The Crown']]

# *accessing an element from the Multi-Dimensional List using index number*

print("Accessing a element from a Multi-Dimensional list")

print(List[0][1])

print(List[1][0])

**Negative Indexing**

-- In Python, negative sequence indexes represent positions from the end of the array. Instead of having to compute the offset as in List[len(List)-3], it is enough to just write List[-3].

-- Negative indexing means beginning from the end, -1 refers to the last item, -2 refers to the second-last item, etc.

**Getting Size of the Python List**

-- Python len() is used to get the length of the list.

# *Creating a List*

List1 = []

print(len(List1))

# *Creating a List of numbers*

List2 = [10, 20, 14]

print(len(List2))

**Taking Input of a Python List**

-- We can take the input of a list of elements as string, integer, float, etc. But the default one is a string.

# *To take space separated input as a string split and store it to a list and print the string list*

# *input the list as string*

string = input("Enter elements (Space-Separated): ")

# *split the strings and store it to a list*

lst = string.split()

print('The list is:', lst)   # *printing the list*

# *input size of the list*

n = int(input("Enter the size of list : "))

# *store integers in a list using map, split and strip functions*

lst = list(map(int, input("Enter the integer elements:").strip().split()))[:n]

# *printing the list*

print('The list is:', lst)

**Adding Elements to a Python List**

Method 1: 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.

-- Tuples can also be added to the list with the use of the append method because tuples are immutable.

-- Unlike Sets, Lists can also be added to the existing list with the use of the append() method.

# *Python program to demonstrate Addition of elements in a List*

# *Creating a List*

List = []

print("Initial blank List: ")

print(List)

# *Addition of Elements in the List*

List.append(1)

List.append(2)

List.append(4)

print("\nList after Addition of Three elements: ")

print(List)

# *Adding elements to the List using Iterator*

*for* i in range(1, 4):

    List.append(i)

print("\nList after Addition of elements from 1-3: ")

print(List)

# *Adding Tuples to the List*

List.append((5, 6))

print("\nList after Addition of a Tuple: ")

print(List)

# *Addition of List to a List*

List2 = ['Bellamy', 'Blake']

List.append(List2)

print("\nList after Addition of a List: ")

print(List)

Method 2 : 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.

-- Unlike append() which takes only one argument, the insert() method requires two arguments(position, value).

# *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, 'Tokyo')

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

print(List)

Method 3 : Using extend() method

-- Other than append() and insert() methods, there’s one more method for the Addition of elements, extend(), this method is used to add multiple elements at the same time at the end of the list.

-- append() and extend() methods can only add elements at the end.

# *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 multiple elements to the List at the end (using Extend Method)*

List.extend([8, 'Ratnakar', 'Matkari'])

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

print(List)

**Reversing a List**

Method 1 : Using reverse() method

### -- A list can be reversed by using the reverse() method in Python.

# *Reversing a list*

mylist = [1, 2, 3, 4, 5, 'Narayan', 'Dharap']

mylist.reverse()

print(mylist)

Method 2 : Using reversed() method

-- The reversed() function returns a reverse iterator, which can be converted to a list using the list() function.

my\_list = [1, 2, 3, 4, 5]

reversed\_list = list(reversed(my\_list))

print(reversed\_list)

What is the difference between reverse() and reversed() method in list in Python ?

**Removing Elements From the List**

Method 1 : Using remove() method

-- Elements can be removed from the List by using the built-in **remove()** function but an Error arises if the element doesn’t exist in the list.

-- Remove() method only removes one element at a time, to remove a range of elements, the iterator is used.

-- The remove() method removes the specified item.

-- Remove method in List will only remove the first occurrence of the searched element.

# *Creating a List*

List = [1, 2, 3, 4, 5, 6,

        7, 8, 9, 10, 11, 12]

print("Initial List: ")

print(List)

# *Removing elements from List using Remove() method*

List.remove(5)

List.remove(6)

print("\nList after Removal of two elements: ")

print(List)

# *Creating a List*

List = [1, 2, 3, 4, 5, 6,

        7, 8, 9, 10, 11, 12]

# *Removing elements from List using iterator method*

*for* i in range(1, 5):

    List.remove(i)

print("\nList after Removing a range of elements: ")

print(List)

Method 2 : Using pop() method

-- pop() function can also be used to remove and return an element from the list, but by default it removes only the last element of the list.

-- To remove an element from a specific position of the List, the index of the element is passed as an argument to the pop() method.

List = [1, 2, 3, 4, 5]

# *Removing element from the Set using the pop() method*

List.pop()

print("\nList after popping an element: ")

print(List)

# *Removing element at a specific location from the Set using the pop() method*

List.pop(2)

print("\nList after popping a specific element: ")

print(List)

**Slicing Of a List**

-- We can get substrings and sub lists 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]

# *Creating a List*

List = ['C', 'H', 'A', 'K', 'S', 'F',

        'O', 'Z', 'G', 'O', 'O', 'J', 'S']

print("Initial List: ")

print(List)

# *Print elements of a range using Slice operation*

Sliced\_List = List[3:8]

print("\nSlicing elements in a range 3-8: ")

print(Sliced\_List)

# *Print elements from a pre-defined point to end*

Sliced\_List = List[5:]

print("\nElements sliced from 5th "

      "element till the end: ")

print(Sliced\_List)

# *Printing elements from beginning till end*

Sliced\_List = List[:]

print("\nPrinting all elements using slice operation: ")

print(Sliced\_List)