COMPUTATIONAL THINKING AND PROGRAMMING-1

Lists

Learning Objectives

By the end of this class you will be able to learn:

- *What is a List*
- *List Operations*
- *Nested List*
- *Traversing a List*
- *List Methods and*
- *Built-in Functions*
- *Copying Lists*
- *List Manipulation*

Lists

- The data type list is an ordered sequence which is mutable and made up of one or more elements. Unlike a string which consists of only characters, a list can have elements of different data types, such as integer, float, string, tuple or even another list. A list is very useful to group together elements of mixed data types.
- Elements of a list are enclosed in square brackets and are separated by comma. Like string indices, list indices also start from 0.

  Example:

```
>>> list1 = [2,4,6,8,10,12]          #list1 is the list of six even numbers
>>> print(list1)
[2, 4, 6, 8, 10, 12]
```

Accessing Elements in a List

The elements of a list are accessed in the same way as characters are accessed in string.

```
>>> list1 = [2,4,6,8,10,12]          #initializes a list list1
>>> list1[0]                         #return first element of list1
2
>>> list1[3]                         #return fourth element of list1
8
>>> list1[15]                        #return error as index is out of range
IndexError: list index out of range
>>> list1[1+4]                       #an expression resulting in an integer index
12
```

Lists are Mutable

In Python, lists are mutable. It means that the contents of the list can be changed after it has been created.

```
>>> list1 = ['Red‘,'Green‘,'Blue‘,'Orange']                    #List list1 of colors

>>> list1[3] = 'Black‘                                         #change/override the fourth element of list1

>>> list1 #print the modified list list1

['Red', 'Green', 'Blue', 'Black']
```

Lists Operations

Lists allows manipulation of its contents through various operations as:

1. **Concatenation:**

Python allows us to join two or more lists using concatenation operator depicted by the symbol +.

```
>>> list1 = [1,3,5,7,9]                    #list1 is list of first five odd integers
>>> list2 = [2,4,6,8,10]                   #list2 is list of first five even integers
>>> list1 + list2                          #elements of list1 followed by list2
[1, 3, 5, 7, 9, 2, 4, 6, 8, 10]
>>> list3 = ['Red','Green','Blue']
>>> list4 = ['Cyan', 'Magenta', 'Yellow‘ ,'Black']
>>> list3 + list4
['Red‘,'Green‘,'Blue‘,'Cyan‘,'Magenta', 'Yellow‘,'Black']
```

Lists Operations

Lists allows manipulation of its contents through various operations as:

2. **Repetition :**

   Python allows us to replicate a list using repetition operator depicted by symbol *.

```
>>> list1 = ['Hello']
#elements of list1 repeated 4 times
>>> list1 * 4
['Hello', 'Hello', 'Hello', 'Hello']
```

Lists Operations

Lists allows manipulation of its contents through various operations as:

3. Membership (In and Not In):

The membership operators in checks if the element is present in the list and returns True, else returns False.

```
>>> list1 = ['Red','Green','Blue']

>>> 'Green' in list1        True

>>> 'Cyan' in list1         False
```

The not in operator returns True if the element is not present in the list, else it returns False.

```
>>> list1 = ['Red','Green','Blue']

>>> 'Cyan' not in list1     True

>>> 'Green' not in list1    False
```

Lists Operations

Lists allows manipulation of its contents through various operations as:

4. Slicing:

Like strings, the slicing operation can also be applied to lists.

```
>>> list1 =['Red','Green','Blue','Cyan','Magenta','Yellow','Black']
>>> list1[2:6]
['Blue', 'Cyan', 'Magenta', 'Yellow']          #list1 is truncated to the end of the list
>>> list1[2:20]                                #second index is out of range
['Blue', 'Cyan', 'Magenta', 'Yellow', 'Black']
>>> list1[:5] #first index missing
['Red','Green','Blue','Cyan','Magenta']        #return sublist from index 0 to 4
```

Lists Operations

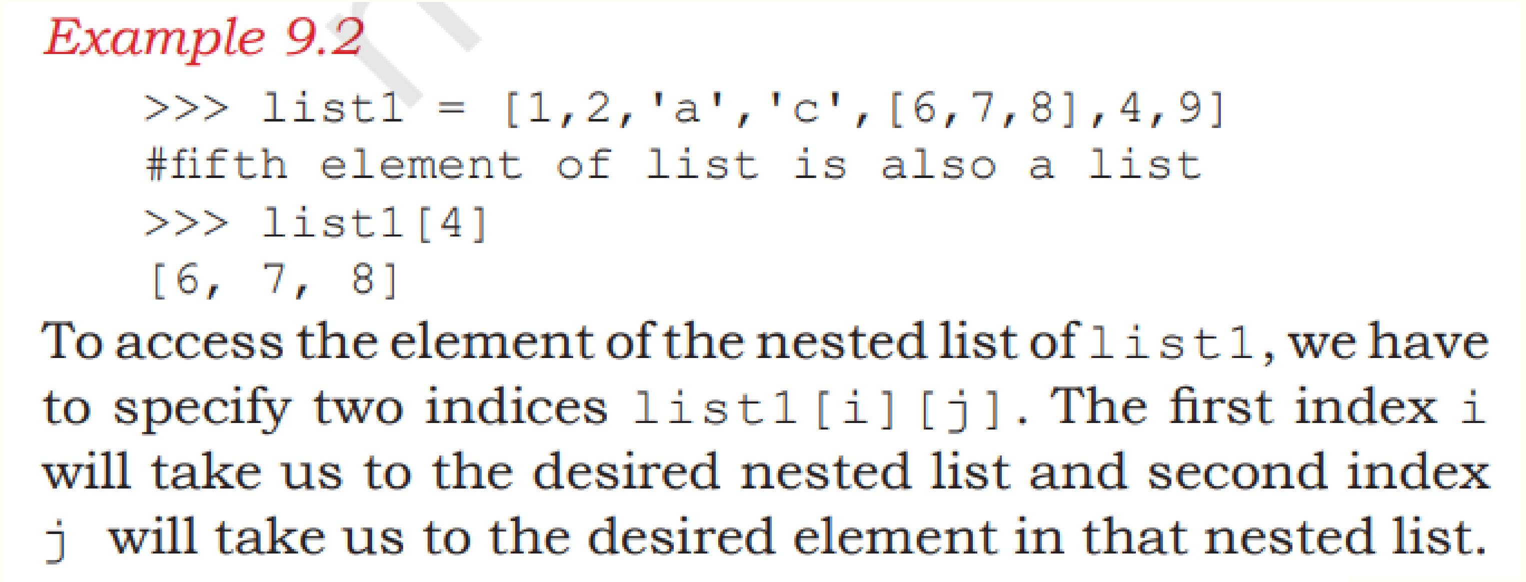
Lists allows manipulation of its contents through various operations as:

4. **Slicing:**

```
#negative indexes
>>> list1 =['Red','Green','Blue','Cyan','Magenta','Yellow','Black']
>>> list1[-6:-2]                              #elements at index -6,-5,-4,-3 are sliced
['Green','Blue','Cyan','Magenta']
>>> list1[::2] #step size 2 on entire list    #both first and last index missing
['Red','Blue','Magenta','Black']
>>> list1[::-1]                               #whole list in the reverse order
['Black','Yellow','Magenta','Cyan','Blue‘,'Green','Red']
```

Nested List

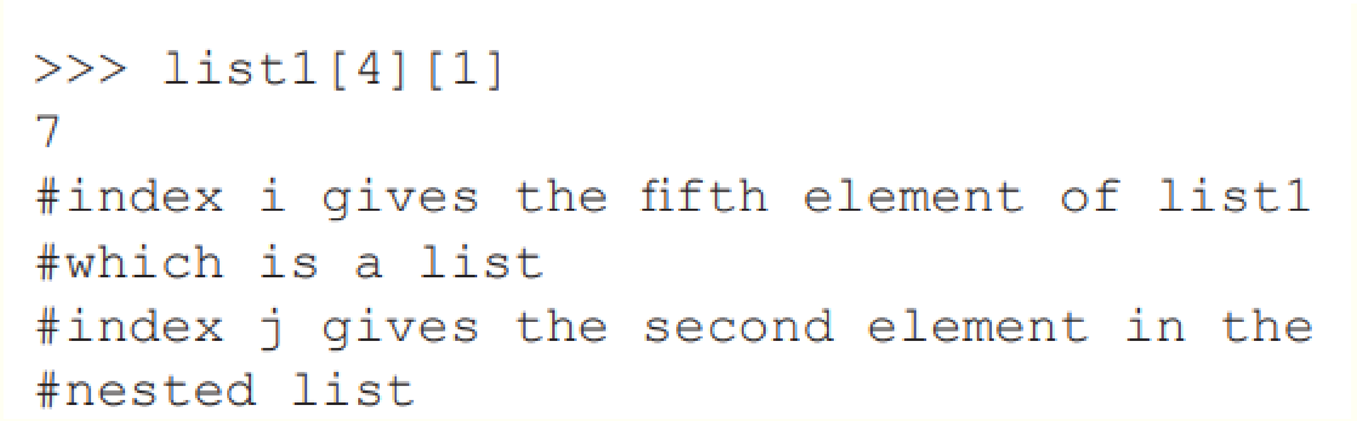
When a list appears as an element of another list, it is called **a nested list**



Example 9.2

```
>>> list1 = [1,2,'a','c',[6,7,8],4,9]
#fifth element of list is also a list
>>> list1[4]
[6, 7, 8]
```

To access the element of the nested list of `list1`, we have to specify two indices `list1[i][j]`. The first index `i` will take us to the desired nested list and second index `j` will take us to the desired element in that nested list.



```
>>> list1[4][1]
7
#index i gives the fifth element of list1
#which is a list
#index j gives the second element in the
#nested list
```

Lists Traversal

We can access each element of the list or traverse a list using a for loop or a while loop.

1. ***List Traversal Using for Loop:***

```
   >>> list1 = ['Red','Green','Blue','Yellow‘,'Black']
 >>> for item in list1:
           print(item)
```

Another way of accessing the elements of the list is

using range() and len() functions:

```
>>> for i in range(len(list1)):
          print(list1[i],endl=“”)
```

Output: Red Green Blue Yellow Black

Lists Traversal

We can access each element of the list or traverse a list using a for loop or a while loop.

2. List Traversal Using while Loop:

```
>>> list1 = ['Red','Green','Blue','Yellow‘,'Black‘]

>>> i = 0

>>> while i < len(list1):

        print(list1[i])

        i += 1
```

Output:

```
Red Green Blue Yellow Black
```

Lists Methods and Built-In Functions

Method	Description	Example

len()	Returns the length of the list passed as the argument	>>> list1 = [10,20,30,40,50]
		>>> len(list1)
		Output: 5

list()	Creates an empty list if no argument is passed
	Creates a list if a sequence is passed as an argument
		>>> list1 = list()
		Output: [ ]
		>>> str1 = 'aeiou'
		>>> list1 = list(str1)
		>>> list1
		Output: ['a', 'e', 'i', 'o', 'u']

append()	Appends a single element passed as an
	argument at the end of the list
		>>> list1 = [10,20,30,40]
		>>> list1.append(50)
		>>> list1
		Output: [10, 20, 30, 40, 50]

Lists Methods and Built-In Functions

Method	Description	Example

extend()	Appends each element of the list passed as argument to the end of the given list	>>> list1 = [10,20,30]
		>>> list2 = [40,50]
		>>> list1.extend(list2)
		>>> list1
		Output: [10, 20, 30, 40, 50]

insert()	Inserts an element at a particular index in the list	>>> list1 = [10,20,30,40,50]
		>>> list1.insert(2,25)
		>>> list1
		Output: [10, 20, 25, 30, 40, 50]

count()	Returns the number of times a given element appears in the list	>>> list1 = [10,20,30,10,40,10]
		>>> list1.count(10)
		Output: 3

Lists Methods and Built-In Functions

Method	Description	Example

remove()	Removes the given element from the list. If the element is present multiple times, only the first occurrence is removed. If the element is not present, then ValueError is generated	>>> list1 = [10,20,30,40,50,30]
		>>> list1.remove(30)
		>>> list1
		Output: [10, 20, 40, 50, 30]
		>>> list1.remove(90)
		Output:
		ValueError:list.remove(x):x not in list

pop()	Returns the element whose index is passed as parameter to this function and also removes it from the list. If no parameter is given, then it returns and removes the last element of the list	>>> list1 = [10,20,30,40,50,60]
		>>> list1.pop(3)
		40
		>>> list1
		Output:
		[10, 20, 30, 50, 60]

Lists Methods and Built-In Functions

Method	Description	Example

sort()	Sorts the elements of the given list in-place	>>> list1 = [34,66,12,89,28,99]
		>>> list1.sort(reverse = True)
		>>> list1
		Output: [99,89,66,34,28,12]

min()	Returns minimum or smallest element of the list	>>> list1 = [34,12,63,39,92,44]
		>>> min(list1)
		12

max()	Returns maximum or largest element of the list	>>> list1 = [34,12,63,39,92,44]
		>>> max(list1)
		92

index()	Returns index of the first occurrence of the element in the list. If the element is not present, ValueError is enerated	>>> list1 = [10,20,30,20,40,10]
		>>> list1.index(20)
		Output: 1
		>>> list1.index(90)
		ValueError: 90 is not in list

Copying Lists

Given a list, the simplest way to make a copy of the list is to assign it to another list.

```
>>> list1 = [1,2,3]

>>> list2 = list1

>>> list1

[1, 2, 3]

>>> list2

[1, 2, 3]
```

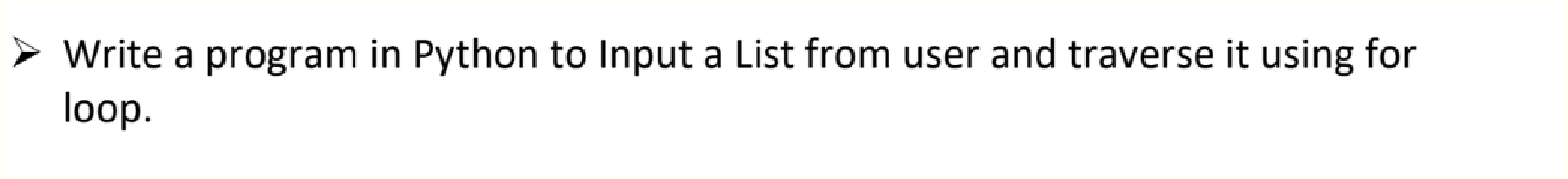
The statement list2 = list1 does not create a new list. Rather, it just makes list1 and list2 refer to the same list object. Here list2 actually becomes an alias of list1. Therefore, any changes made to either of them will be reflected in the other list.

Copying Lists

We can use the built-in function list() as follows:

```
newList = list(oldList)

>>> list1 = [10,20,30,40]

>>> list2 = list(list1)

>>> list2

[10, 20, 30, 40]
```

Assignments



- Write a program in Python to Input a List from user and traverse it using for loop.

Summary

- Lists are mutable sequences in Python, i.e., we can change the elements of the list.
- Elements of a list are put in square brackets separated by comma.
- List indexing is same as that of strings and starts at 0.
- Two way indexing allows traversing the list in the forward as well as in the backward direction.
- Operator + concatenates one list to the end of other list.
- Operator * repeats a list by specified number of times.
- Membership operator in tells if an element is present in the list or not and not in does the opposite.
- Slicing is used to extract a part of the list.
- There are many list manipulation functions including: len(), list(), append(), extend(), insert(),
- count(), find(), remove(), pop(), reverse(), sort(), sorted(), min(), max(), sum().

BIBLIOGRAPHY:

1. Computer Science Textbook for class XI
   by NCERT
2. Computer Science with Python
   by Sumita Arora
3. Computer Science with Python
   by Preeti Arora