PYTHON REVISION TOUR – II(Cont…)

LISTS

List in Python

- List is a standard data type of Python that can store a sequence of values belonging to any type.
- List is mutable (modifiable) sequence i.e. element can be changed in place.

Example :

–List=[1,2,3,4,5]

–List1=['p','r','o','b','l','e','m']

–List2=['pan‘ , ‘ ran‘ , 'blade', ‘lemon', 'egg', 'mango']

Creating a List

List can be created by assigning a variable with the values enclosed in square bracket separated by comma.

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

- **Creating Empty List:** List with no item is a empty list. E.g List=[ ].
- It can be created with the function list1=list().it generates the empty list with the name list1. This list is equivalent to 0 and has truth value false.

- **Creating List from Existing Sequence:** List1=list(sequence)

Example: List1=list('Computer')

>>>List1

Output: ['C','o','m','p','u','t','e','r']

- **Creating List from keyboard Input:**

list1=list(input('Enter the list item:'))

or

list1=eval(input('Enter the list item:'))

List vs String

- **Similarities:**

–Length

–Indexing Slicing

–Membership operators

–Concatenation and Replication operators

–Accessing Individual elements

- **Differences**

–Storage : are similar to string , but it stores reference at each index instead of single characters.

–Mutability: Strings are not mutable, while list are.

Traversing a List

Syntax: for <item> in <List>

Example:

```
List1=['C','o','m','p','u','t','e','r']
for a in List1
        print(a)
```

C
o
m
p
u
t
e
r

List Operations

- **Joining Lists:** Two Lists can be joined through addition.

```
>>>l1=[1,2,3]
>>>l2=[4,5,6]
>>>l3=l1+l2
>>>l3
[1,2,3,4,5,6]
```

- **Repeating or Replicating Lists:** Multiply(*) operator replicates the List specified number of times

```
>>>l1=[1,2,3]
>>>l1*3
[1,2,3,1,2,3,1,2,3]
```

SLICING THE LIST

- List slices are the subpart of a list extracted out. List slices can be created through the use of indexes.

Syntax: Seq=List[start:stop] creates list slice out of List1 with element falling in between indexes start and stop not including stop.

Example:

```
>>>List1=[1,2,3,4,5,6,7,8]
>>>seq=List1[2,-3]
>>>seq
```

Output: [3,4,5]

- List also supports slice steps. Example, Seq=List[start:stop:step] creates list slice out of List with element falling in between indexes start and stop not including stop, skipping step-1 element in between.

Example:

```
>>>List1=[1,2,3,4,5,6,7,8]
>>>seq=List1[2,7,2]
>>>seq
```

Output:

[3,5,7]

Using Slices for List Modification

Example 1:

>>> List=['add','sub','mul']

>>>List[0:2]=['div,'mod']

>>>List

Output: ['div,'mod','mul']

Example 2:

>>> List=['add','sub','mul']

>>>List[0:2]="a"

>>>List

Output: ["a","mul"]

Example 3:

>>> List=[1,2,3]

>>>List[2:]="604"

>>>List

Output: [1,2,3,'6','0','4']

Example 4:

>>> List=[1,2,3]

>>>List[10:20]="abcd"

>>>List

Output: [1,2,3,'a','b','c','d']

List Manipulation

Appending Elements to a list

append() method adds a single item to the end of the list.

Syntax: List.append(item)

Example:

```
>>> List=[1,2,3]
>>>List.append(6)
>>>List
```

Output: [1,2,3,6]

Updating Element to a list

Assign new value to the element's index in list.

Syntax: List[index]=<new value>

Example :

>>> List=[1,2,3]

>>>List[1]=4

>>>List

Output: [1,4,3]

Deleting Element from a list

Del statement can be used to remove an individual item, or to remove all items identified by a slice.

Example 1:

```
>>> List=[1,2,3,4,5,6,7,8,9,10]
 >>>del List[5]
>>>List
 [1,2,3,4,5,7,8,9,10]
```

Example 2:

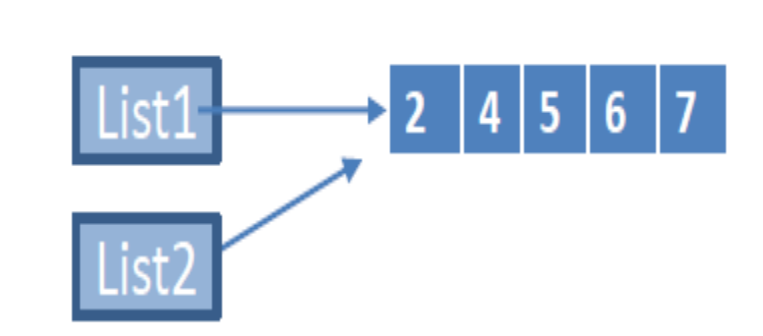
```
>>> List=[1,2,3,4,5,6,7,8,9,10]
>>>del List[5:8]
>>>List
[1,2,3,4,5,10]
```

Example 3:

```
>>> List=[1,2,3,4,5,6,7,8,9,10]
>>>del List
>>>List
[ ]
```

Making True copy of a List

- Assignment does not make a copy of a list.
- Assignment makes two variable to point to same list in memory, called shallow copying.



- The changes made in one list will also be reflected to the other list.
- For creating a true copy we need to make

  List2=list(List1)

now List1 and List2 are separate list, called Deep Copy

List Functions and Methods

The index() Method:

This function returns the index of first matched item from the list.

Syntax: List.index(<item>)

Example:

```
>>>list=[12,14,15,17,14,18]
>>>list.index(14)
```

Output:

1

Note: If item is not in the list it raises exception value error.

The append() Method

- This function adds an item to the end of the list.

Syntax:

List.append(<item>)

Example:

```
>>>list=[12,14,15,17]
>>>list.append(18)
>>>list
```

Output:

[12,14,15,17,18]

The extend() Method

This function adds multiple item to the end of the list.

Syntax:

List.extend(<item>)

Example:

```
>>>list1=[12,14,15,17]
 >>>list2=[18,19,20]
 >>>list1.extend(list2)
>>>list1
```

Output:

[12,14,15,17,18,19,20]

The insert() Method

This function inserts item at the given position in the list.

Syntax:

List.insert(<pos>,<item>)

Example:

```
>>>list1=[12,14,15,17]
>>>list1.insert(2,200)
>>>list1
```

Output:

[12,14,200,15,17]

The pop() Method

This removes the item at the given position in the list.

List1.pop() removes last item in the list

List1.pop(3) removes item at index 3 in the list.

The remove() Method:

This function removes first occurrence of given item from the list.

Syntax:

List.remove(<item>)

Example:

```
>>>list1=[12,14,15,17]
>>>list1.remove(15)
>>>list1
```

Output: [12,14,17]

The clear() Method:

This function removes all the items from the list.

Syntax:

List.clear()

```
>>>list1=[12,14,15,17]
>>>list1.clear()
>>>list1
```

Output: [ ]

The count() Method

This function returns the count of the item passed as argument. If given item is not in the list it returns zero.

```
>>>list1=[12,14,15,14,17]
>>>list1.count(14)
```

Output:

2

```
>>>list1=[12,14,15,14,17]
 >>>list1.count(18)
```

Output:

0

The reverse() Method

This function reverses the item of the list. This is done “in place”,i.e it does not create new list.

Syntax:

List.reverse()

Example:

```
>>>list1=[12,14,15,14,17]
 >>>list1.reverse()
```

Output:

[17,14,15,14,12]

The sort() Method

This function sorts the items of the list, by default in increasing order. This is done “in place”,i.e it does not create new list.

Syntax: List.sort()

>>>list1=[12,14,15,14,17]

>>>list1.sort()

Output: [12,14,14,15,17]

It sorts the string in lexicographic manner. If we want to sort the list in decreasing order, we need to
>>>list1.sort(reverse=True)

Output: [17,15,14,14,12]