Lists

Introduction: List is the basic data type in Python .List contains set of different data types arranged in a sequence. Each element of a sequence is assigned a number called index. The first element of the list index is zero, the second element index is one, and so forth. List literals are written within square brackets [ ]. Lists can also be used for implementing stacks and queues. Lists are mutable, i.e., they can be altered once declared.

E.g.:a_list=[1,2,”cat”,”dog”,4]

Accessing List: To access values from the list, index numbers are used . We can access lists in indexing, slicing

Indexing list: Every element in a list is assigned with index number starting with zero.By indexing the value we can access the elements present in that particular index. Negative indexing can also be done in python. In negative indexing the last element index starts with -1 .

E.g.:
```
a_list = [1,2,”cat”,”dog”,4]
print (a_list[1])    ## 2
print (a_list[0])    ## 1
print (a_list[-1])    ## 4
```

Here we are accessing values of a_list with index number.

Slicing list: Slices work on lists just as with strings, these can be used to change sub-parts of the list. Slicing of a can be done even in reverse order, here in reverse order index starts with -1 at the last element of the list ,-2 for second last element of the list , and so on.

E.g.::
```
list = ['a', 'b', 'c', 'd']
print (list[1:-1])   ## ['b', 'c']
list[0:2] = 'z'    ## replace ['a', 'b'] with ['z']
print (list)        ## ['z', 'c', 'd']
```

Operations: List responds to concatenation (+) and repetition (*) operators.

Concatenation: Two lists can be concatenated using +operator.

E.g.:
```
a_list=[1,2,3,’a’,’b’,’c’]
b_list=[4,5,6,’d’,’e’,’e’]
print(a_list+ b_list)   #[1,2,3,’a’,’b’,’c’, 4,5,6,’d’,’e’,’e’]
```

Repetition: List repetetion can be done using * operator. Here elements in the list can be repeated by * operators.

E.g:
```
print[‘Hi’] *4  ##[‘Hi’,‘Hi’,‘Hi’,‘Hi’]
```

List Comprehension: List Comprehensions is a tool, which creates a new list in a single and readable line. Suppose If we want to create list of integers from 0 to 10,in a single line we can use list comprehension.

E.g.:
```
S = [x**2 for x in range(10)]
print (S)
```
Here we are assigning values to an empty list name 'S'. The values in the list ranges from 0 to 10 and append in the list name'S'.

<u>Functions and Methods:</u> Python contains functions and methods to perform operations on lists.

Cmp: Cmp compared elements of both the lists.

Syntax: cmp(list1, list2)

E.g.:
```
list_a=[123, "hello"]
list_b=[321,"hai"]
cmp(list_a,list_b)
```

Len: Len finds the length of the list and returns the length of the list.

Sytax: len(list_name)

E.g:
```
len(list_a)  #2
a_list=[1,2,3,4,"ad","sub',"hello",5,3,3]
len(a_list) #10
```

max: finds the maximun element in the list and returns the maximum element in the list

Sytax: max(list_name)

E.g:
```
list_max=[1,22,3,5,44,333,765,4215]
max(list_max) #4215
```

min: finds the minimum element in the list and returns the minimum number in the list.

Sytax: min(list_name)

E.g:
```
list_min=[1,22,3,5,44,333,765,4215]
min(list_max) #1
```

append: Append adds the given element at the end of the list.

Syntax: list_name.append(index)

E.g:
```
 list_ac
list_a.append("hello world")
print(list_a)  #[123, "hello", 1,22,3,5,44,333,765,"hai", "hello world"]
```

pop: By using Pop method we can removes the last most element of the list or by defining a particular element using index value.

Syntax: list_name.pop(index)

E.g:
```
list = ['larry', 'curly', 'moe']
list.pop(1)  #larry
```

del:del Using del method it deletes the element in the list by using index value

Syntax: del list_name[index]

E.g:
```
list1 = ['physics', 'chemistry', 1997, 2000];
print list1
del list1[2]
```

insert: inserts the element at the given index, shifting the elements to the right.

Syntax: insert(index, element)

E.g:
```
list1 = ['physics', 'chemistry', 1997, 2000];
insert(1,"maths")
print(list1) # ['physics',"maths", 'chemistry', 1997, 2000]
```

extend: extend adds the elements in second list to the end of the list. Using + or += on a list is similar to using extend().

Syntax:list1.extend(list2)

E.g:
```
list1=['physics','chemistry',1997,2000];
list2=[123, "hello", 1,22,3,5,44,333,765,"hai", "hello world"]
list1.extend(list2)
print(list1)
#['physics','chemistry',1997,2000, 123, "hello", 1,22,3,5,44,333,765,"hai", "hello world"]
```

sort: Sort method sorts the value in an order.

Syntax:list1.sort()

E.g:
```
list1=[1,22,3,5,44,333,765]
list1.sort()
```

reverse: Reverse method reverses the entire elements present in the list.

Syntax: list.reverse()

E.g:
```
list1=[1,22,3,5,44,333,765]
list1.revers()
```