III B.Tech.

Computer Science & Engineering

CSE304: PYTHON PROGRAMMING WITH WEB FRAMEWORKS

UNIT – I: Lists

By
Mrs. S. KAMAKSHI, AP-III / CSE
School of Computing

LIST

- To store a collection of heterogeneous items
- Ordered collection
- Mutable – that is in-place change is allowed
- Can be nested to any depth
- To create list:
- Syntax:

```
list_name = [item1, item2, …]
```

- Eg.

```
[]                                        # Empty List
[1, 2, 3]                                 # List of int
[‘a’, ‘b’, ‘c’, ‘d’, ‘e’, ‘f’]            # List of strings
[“Ramu”, “Somu”, 25, 32.0, [1, 2, 3]]     # Heterogeneous List
```

Get and Set List Items

- Index starts at 0
- For referring an item: list_name[index]
  - If index >= 0 position from the beginning
  - If index < 0 position from last
- Eg.

```
L = [10, 20, 30, 40, 50, 60]
print (L[2])                # prints 30
print (L[-2])               # prints 50
L[3] = 25                   # modifies 40 to 25
L[7] = 10                   # IndexError Exception
L = [10]*5                  # Same as L = [10, 10, 10, 10, 10]
```

Copy, Slice & Concatenate Lists

- Slicing
  - List_name[start:end:step]
- Eg.

```
L = [10, 20, 30, 40, 50, 60]
L[1:4]                          # [20, 30, 40]
L[1:4:2] = [5, 15]                      # [10, 5, 30, 15, 50, 60]
L[-1:-3:-1]                             # [60, 50]
```

- Copy
  - Reference copy (Shallow copy):
    - L1 = L
  - Deepcopy
    - L1 = L.copy()
    - L1 = L.deepcopy()
- Concatenation
  - L1 = L1 + [4, 5, 6, 7]   # returns a new list
  - L1 += [8]   # in-place concatenation

List Functions

Method	Description
len(list)	Returns the number of items in the list
append(item)	Appends the specified item to the end of the list. Increases the length of the list by one
insert(index, item)	Inserts the specified item at the specified index, Increases the length of the list by one and shifts all the items after the specified index by one position ahead
remove(item)	If item is present in the list then, it removes the first occurrence of the item in the list and decreases the length of the item by one If not found raises ValueError exception
index(item)	If found, returns the index of the specified item in the list If not found, raises ValueError
pop([index])	If index is not specified, removes and returns the last item from the list . If index is specified and is within length of the list, removes and returns the elements at the specified location, if index >= length raises IndexError exception. Decreases the length of the list by one.

List Functions

Method	Description
count(item)	Returns the number of occurrences of an item in the list. If not found, returns 0
reverse(list)	Reverses the order of the items in the list
sort([key = func.])	Sorts the items in-place. The optional key argument specifies the function to be called on each item before sorting

Built-in Functions for working with lists

Method	Description
sorted(list, [key = func])	Returns a new list consisting of sorted items of the original list. The optional key argument specifies a function to be called on each item before sorting
min(list)	Returns the minimum value in the list
max(list)	Returns the maximum value in the list

Random Module Functions for working with lists

Method	Description
choice(list)	Returns a randomly selected item from the list
shuffle(list)	Shuffles the items in the list on a random basis

Using lists in loops

- **In for loop:**

```
scores = [80, 73, 92, 64]
total = 0
for s in scores:
        total += s
print (total)
```

- **In while loop:**

```
scores = [80, 73, 92, 64]
total = 0
i = 0
while i < len(scores):
        total += scores[i]
        i += 1
print (total)
```

Passing List to Functions

- Passed by reference
- If any in-place change is made to list items in the function that is reflected in the calling function also
- Eg.

```
def  modify(L):
        for i in range(len(L))
                L[i] += 5
        print (L)
mylist = [10, 20, 30, 40]
modify(mylist)
print (mylist)
```

Output:

```
[15, 25, 35, 45]
[15, 25, 35, 45]
```

```
def no_modify(L):
     L = L * 2
     print (L)
mylist = [10, 20, 30, 40]
modify(mylist)
print (mylist)
```

Output:

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

Passing List to Functions

- Assignment operator *= modifies the list elements in-place.
- Eg.

```
def modify(L):
       L *= 2
        print (L)
mylist = [10, 20, 30, 40]
modify(mylist)
print (mylist)
```

- **Output:**

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

List of Lists

- A list in which its elements are lists
- Similar to a 2D list
  - Accessed by using two subscripts
- May be extended to any depth – nD list
  - Accessed by using n subscripts
- Eg.

  LL = [[1, 2, 3], [‘a’, ‘b’, ‘c’, ‘d’], [[12,14], ‘abcd’, 145], 62]

  LL[0] is [1, 2, 3]

  LL[0][0] is 1

  LL[1] is [‘a’, ‘b’, ‘c’, ‘d’]

  LL[1][3] is ‘d’

  LL[2] = [[12,14], ‘abcd’, 145]

  LL[2][0] is [12, 14]

  LL[2][0][0] is 12