Jawaharlal Nehru Technological University Hyderabad

Kukatpally, Hyderabad - 500 085, Telangana, India

Python List

Session 7 , 3 Oct 22

Dr N V Ganapathi Raju
Professor and HOD of IT
GRIET

Python List Features

- Data structure, can hold any type of data
- Mutable, can be modified
- is a container that holds, ordered sequence of items
- Different operation like insertion, updating and deletion can be performed on lists.
- Enclosed between square([]) brackets

Creating List elements

- Lists can be created using **square**([]) **brackets**
- Values in a list are called **elements (items)**
- Each item in a list has an assigned **index** value, first item in the list is at **index 0.**

```
list1 = [100, "python", 'p', 2000.55,True]

print(list1)
```

List contains Heterogenous Elements

Accessing List elements

```
list1 = [100, "python", 'p', 2000.55,True]
print(list1)


print("--Forward indexing--")
print(list1[0])
print(list1[1])
print(list1[4])


print("--Backward Indexing--")
print(list1[-1])
print(list1[-2])
print(list1[-5])
```

Note:

Case 1: print(list1[5])

Case 2: print(list1[-6])

Raises Exception : List index out of range

List Operations

- **Adding Lists**, can be added by using the concatenation operator (+) to join two lists

```
list1=[10,"Python"]
list2=['p',25.50]
list3=list1+list2
print (list3)
```

```
list1=[10,"Python"]
list2= 35.50
list3=list1+list2
print (list3)
```

TypeError: can only concatenate list (not "float") to list

- **Replicating Lists**, can be performed by using '*' operator by a specific number of times.

```
list1=[10,20,30]
print (list1*3)
```

Membership operations on Lists

- Pythons supports membership operators : in , not in operators.
- Both returns Boolean based on expression

```
list1 = [100, "python", 'p', 2000.55,True]
"python" in list1
```

```
list1 = [100, "python", 'p', 2000.55,True]
"python" not in list1
```

List Slicing

- list[start:end] :

  Returns elements from the **start** and up to but not including the **end** element.

Example:

```
x = [1, 2, 3, 4, 5]

x[1:]

x[:1]

x[:-2]

x[-2:]
```

List Mutable

- Lists are **mutable**.
- Because List supports
  - ❑ **appending**
  - ❑ **updating**
  - ❑ **deleting elements**

Appending List elements

- **append**() : is used to **append** i.e., **add** an element at the **end of the existing elements**.

  Syntax:

  <list_name>.append(item)

```
list2=[100,'python',50.8,'a',]
print (list2)

list2.append('True')
print (list2)
```

Updating List elements

- **To update or change the value of an index of a list, assign the value to that index of the List.**

  Syntax: <list_name>[index]=<value>

```
list1=[100,'python',50.8,'a',200,300]
print (list1)

list1[4] = 255.55
print (list1)
```

Deleting List elements

- **del** statement can be used to delete an element from the list.

  Syn: del **<list_name>[index]**

  Index element specified in square brackets will be deleted

  Syn: del **<list_name>[start:end]**

  delete all items from startIndex to endIndex.

```
list3=[100,'python',50.8,'a',200,300]
print (list3)

del (list3[5])
print (list3)
```

```
del (list3[2:4])
print (list3)
```

List Traversal

- The most common way to traverse the elements of a list is with a for loop

```
list1=[100,'python',50.8,'a',200,300]
for lst1 in list1:
    print(lst1)
```

List Traversal

- A for loop over an empty list never runs the body

```
list1=[]
for lst1 in list1:
    print(lst1)
```

Note: The code cannot throw error but print nothing

Nested list

- a list can contain another list, the nested list still counts as a single element.

Note : Nested list

```
list1=[100,'python',50.8,'a',[200,300]]

for lst1 in list1:

    print(lst1)
```

List Traversal

- While loop can also be used to traverse the elements of a list but not frequent

```
list1=[100,'python',50.8,'a',200,300]


index = 0
while index < len(list1):
    print(list1[index])
    index = index + 1
```

List insert elements

- **insert**() : inserts an element to the list at a given index

  syntax: insert(index, element)

- **extend**() extends the list by adding all items of a list to the end.

  syntax: extend(list)

  Note: argument for extend() is another list object

Examples for insert() extend()

```
list1=[100,'python',50.8,'a',200]
print (list1)

list1.insert(6,'True')
print (list1)
```

```
data1=[100,'python',50.8]
data2=['a',200]

data1.extend(data2)
print (data1)
```

List len(), count()

- len() : returns the number of elements in an object

  has a parameter, sequence/collection

  if no items in the list returns **0**

- count() : returns the number of occurrences of an element in a list.

  syntax: count(element)

Examples of len(), count()

```
list1 = [100, "python", 'p', 2000.55,True]
print(len(list1))
```

```
list1 = [100, "python", 'p', 2000.55,True, "python",100]
print ("Number of times python occurred is", list1.count("python"))
print ("Number of times 100 occurred is", list1.count(100))
```

List remove(), pop()

- remove() : removes the first matching element from the list

  - takes a single element as an argument and removes it from the list.

  - If the element does not exist, throws Value Error

- pop() : removes the item at the given index from the list and returns the removed item

  syntax: pop(index)

  - the default index **-1** (index of the last item)

  - if the index is not in a range throws **Index Error**

use remove() method to remove the given item or pop() method to remove an item at the given index.

Examples of remove(), pop()

```
list1 = [100, "python", 'p', 2000.55,True]
print ("Last element is", list1.pop())
print ("3rd position element:", list1.pop(2))
print (list1)
```

```
list1 = [100, "python", 'p', 2000.55,True]
list1.remove('p')
print(list1)
```

List sort() reverse() max() min()

- sort(): sorts the elements of a given list in ascending order

  reverse - If true, the sorted list is reversed in descending order

- reverse() : reverses the elements of a given list

- max()/min() : returns maximum and minimum elements of a given list

Examples of sort() reverse()

```
list1 = [100, 400, 150.50, 600]

list1.sort()
print(list1)

list1.sort(reverse=True)
print(list1)
```

```
list2=[10,20,30,40,50]

list2.reverse()

print (list2)
```

Examples of max() min()

```
print('----------max and min----------')

list3 = [-500, 10, 200, 1000]

print(max(list3))

print(min(list3))
```

List index()

- index(element): returns the index of the first occurrence of the element.

element is to be searched

if not found throws ValueError

```
lst1 = [“Lists","are","mutable"]

index = lst1.index('are')
print('The index of p:', index)
```