List in Python

List Creation

List in Python

- In python, a list can be defined as a collection of values or items.
- The items in the list are separated with the comma (,) and enclosed with the square brackets [ ].

Syntax:

list-var = [value1, value2, value3,….]

Example: "listdemo.py"

```
L1 = []
L2 = [123,"python", 3.7]
L3 = [1, 2, 3, 4, 5, 6]
L4 = ["C","Java","Python"]
print(L1)
print(L2)
print(L3)
print(L4)
```

Output:

python listdemo.py

```
[]
[123, 'python', 3.7]
[1, 2, 3, 4, 5, 6]
['C','Java','Python']
```

List Indexing

List Indexing in Python

- Like string sequence, the indexing of the python lists starts from 0, i.e. the first element of the list is stored at the 0th index, the second element of the list is stored at the 1st index, and so on.
- The elements of the list can be accessed by using the slice operator [].

Example:

mylist=['banana','apple','mango','tomato','berry']

banana	apple	mango	tomato	berry
0	1	2	3	4

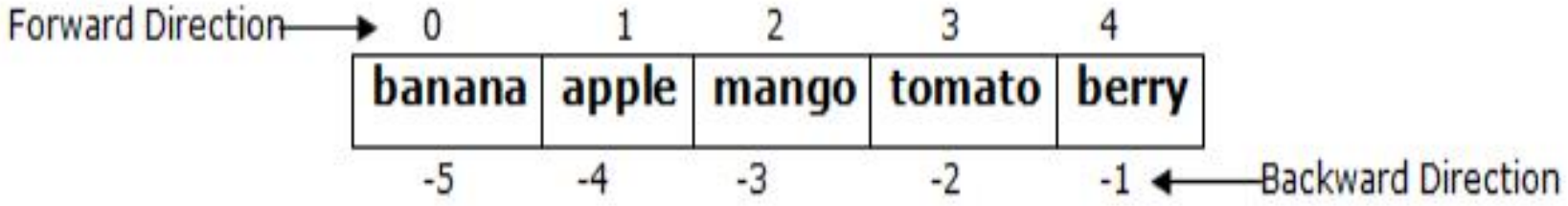
- mylist[0]="banana"
- mylist[2]="mango"

mylist[1:3]=["apple","mango"]

List Indexing in Python cont…

- Unlike other languages, python provides us the flexibility to use the negative indexing also. The negative indices are counted from the right.
- The last element (right most) of the list has the index -1, its adjacent left element is present at the index -2 and so on until the left most element is encountered.

Example: **mylist=['banana','apple','mango','tomato','berry']**



- mylist[-1]="berry"
- mylist[-3]="mango"

mylist[-4:-2]=["apple",mango"]

List Operators

List Operators in Python

+	It is known as concatenation operator used to concatenate two lists.
*****	It is known as repetition operator. It concatenates the multiple copies of the same list.
[]	It is known as slice operator. It is used to access the list item from list.
[:]	It is known as range slice operator. It is used to access the range of list items from list.
in	It is known as membership operator. It returns if a particular item is present in the specified list.
not in	It is also a membership operator and It returns true if a particular list item is not present in the list.

List Operators in Python

cont…

Example: "listopdemo.py"

```
num=[1,2,3,4,5]
lang=['python','c','java','php']
print(num + lang) #concatenates two lists
print(num * 2) #concatenates same list 2 times
print(lang[2]) # prints 2nd index value
print(lang[1:4]) #prints values from 1st to 3rd index.
print('cpp' in lang) # prints False
print(6 not in num) # prints True
```

Output:

```
python listopdemo.py
[1, 2, 3, 4, 5, 'python', 'c', 'java', 'php']
[1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
java
['c', 'java', 'php']
False
True
```

How to update or change elements to a list?

- Python allows us to modify the list items by using the **slice** and **assignment** operator.
- We can use assignment operator (**=**) to change an item or a range of items.

Example: "listopdemo1.py"

```
num=[1,2,3,4,5]
print(num)
num[2]=30
print(num)
num[1:3]=[25,36]
print(num)
num[4]="Python"
print(num)
```

Output:

```
python listopdemo1.py
[1, 2, 3, 4, 5]
[1, 2, 30, 4, 5]
[1, 25, 36, 4, 5]
[1, 25, 36, 4, 'Python']
```

How to delete or remove elements from a list?

- Python allows us to delete one or more items in a list by using the **del** keyword.

Example: "listopdemo2.py"

```
num = [1,2,3,4,5]
print(num)
del num[1]
print(num)
del num[1:3]
print(num)
```

Output:

```
python listopdemo2.py
[1, 2, 3, 4, 5]
[1, 3, 4, 5]
[1, 5]
```

Iterating a List

- A list can be iterated by using a **for - in** loop. A simple list containing four strings can be iterated as follows..

Example: "listopdemo3.py"

```
lang=['python','c','java','php‘]
print("The list items are \n")
for i in lang:
      print(i)
```

Output:

```
python listopdemo3.py
The list items are
python
c
java
php
```

List Functions & Methods

List Functions & Methods in Python

- Python provides various in-built functions and methods which can be used with list. Those are
  - len()
  - max()
  - min()
  - sum()
  - list()
  - sorted()
  - append()
  - remove()
  - sort()
  - reverse()
  - count()
  - index()
  - insert()
  - pop()
  - clear()

☞ len():

- In Python, **len()** function is used to find the length of list,i.e it returns the number of items in the list..

Syntax: `len(list)`

Example: **listlendemo.py**

```
num=[1,2,3,4,5,6]
print("length of list :",len(num))
```

Output:

```
python listlendemo.py
length of list : 6
```

List Functions & Methods in Python

Cont..

☞ max ():

- In Python, max() function is used to find maximum value in the list.

Syntax: `max(list)`

Example: **listmaxdemo.py**

```
list1=[1,2,3,4,5,6]
list2=['java','c','python','cpp']
print("Max of list1 :",max(list1))
print("Max of list2 :",max(list2))
```

Output:

python listmaxdemo.py
Max of list1 : 6
Max of list2 : python

List Functions & Methods in Python

Cont..

☞ min ():

- In Python, min() function is used to find minimum value in the list.

Syntax: `min(list)`

Example: **listmindemo.py**

```
list1=[1,2,3,4,5,6]
list2=['java','c','python','cpp']
print("Min of list1 :",min(list1))
print("Min of list2 :",min(list2))
```

Output:

python listmindemo.py
Min of list1 : 1
Min of list2 : c

☞ sum ():

- In python, sum() function returns sum of all values in the list. List values must in number type.

Syntax: `sum(list)`

Example: **listsumdemo.py**

```
list1=[1,2,3,4,5,6]
print("Sum of list items :",sum(list1))
```

Output:

python listsumdemo.py

Sum of list items : 21

List Functions & Methods in Python

Cont..

☞ list ():

- In python, list() is used to convert given sequence (string or tuple) into list.

Syntax: `list(sequence)`

Example: **listdemo.py**

```
str="python"
list1=list(str)
print(list1)
```

Output:

python listdemo.py

['p', 'y', 't', 'h', 'o', 'n']

List Functions & Methods in Python

Cont..

☞ sorted ():

- In python, sorted() function is used to sort all items of list in an ascending order.

Syntax: `sorted(list)`

Example: **sorteddemo.py**

```
num=[1,3,2,4,6,5]
lang=['java','c','python','cpp']
print(sorted(num))
print(sorted(lang))
```

Output:

python sorteddemo.py
[1, 2, 3, 4, 5, 6]
['c', 'cpp', 'java', 'python']

List Functions & Methods in Python

Cont..

☞ append ():

- In python, append() method adds an item to the end of the list.

Syntax: `list.append(item)`

where item may be number, string, list and etc.

Example: **appenddemo.py**

```
num=[1,2,3,4,5]
lang=['python','java']
num.append(6)
print(num)
lang.append("cpp")
print(lang)
```

Output:

```
python appenddemo.py
[1, 2, 3, 4, 5, 6]
['python', 'java', 'cpp']
```

List Functions & Methods in Python

Cont..

☞ remove ():

- In python, remove() method removes item from the list. It removes first occurrence of item if list contains duplicate items. It throws an error if the item is not present in the list.

Syntax: `list.remove(item)`

Example: **removedemo.py**

```
list1=[1,2,3,4,5]
list2=['A','B','C','B','D']
list1.remove(2)
print(list1)
list2.remove("B")
print(list2)
list2.remove("E")
print(list2)
```

Output:

python removedemo.py

[1, 3, 4, 5]

['A', 'C', 'B', 'D']

ValueError: x not in list

List Functions & Methods in Python

Cont..

☞ sort ():

- In python, sort() method sorts the list elements into descending or ascending order. By default, list sorts the elements into ascending order. It takes an optional parameter 'reverse' which sorts the list into descending order.

Syntax: `list.sort([reverse=true])`

Example: **sortdemo.py**

```
list1=[6,8,2,4,10]
list1.sort()
print("\n After Sorting:\n")
print(list1)
print("Descending Order:\n")
list1.sort(reverse=True)
print(list1)
```

Output:

python sortdemo.py

After Sorting:

[**2**, **4**, **6**, **8** , **10**]

Descending Order:

[**10**, **8**, **6**, **4** , **2**]

List Functions & Methods in Python

Cont..

☞ reverse ():

- In python, reverse() method reverses elements of the list. i.e. the last index value of the list will be present at 0th index.

Syntax: `list.reverse()`

Example: **reversedemo.py**

```
list1=[6,8,2,4,10]
list1.reverse()
print("\n After reverse:\n")
print(list1)
```

Output:

python reversedemo.py

After reverse:

[10, 4, 2, 8 , 6]

List Functions & Methods in Python

Cont..

☞ count():

- In python, count() method returns the number of times an element appears in the list. If the element is not present in the list, it returns 0.

Syntax: `list.count(item)`

Example: **countdemo.py**

```
num=[1,2,3,4,3,2,2,1,4,5,8]
cnt=num.count(2)
print("Count of 2 is:",cnt)
cnt=num.count(10)
print("Count of 10 is:",cnt)
```

Output:

python countdemo.py

Count of 2 is: **3**

Count of 10 is: **0**

List Functions & Methods in Python

Cont..

☞ index():

- In python, index () method returns index of the passed element. If the element is not present, it raises a ValueError.
- If list contains duplicate elements, it returns index of first occurred element.
- This method takes two more optional parameters start and end which are used to search index within a limit.

Syntax: `list. index(item [, start[, end]])`

Example: **indexdemo.py**

```
list1=['p','y','t','o','n','p']
print(list1.index('t'))
Print(list1.index('p'))
Print(list1.index('p',3,10))
Print(list1.index('z')) )
```

Output:

python indexdemo.py
2
0
5
Value Error

List Functions & Methods in Python

Cont..

☞ insert():

- In python, insert() method inserts the element at the specified index in the list. The first argument is the index of the element before which to insert the element.

Syntax: `list.insert(index,item)`

Example: **insertdemo.py**

```
num=[10,20,30,40,50]
num.insert(4,60)
print(num)
num.insert(7,70)
print(num)
```

Output:

python insertdemo.py

[10, 20, 30, 40, 60, 50]

[10, 20, 30, 40, 60, 50, 70]

List Functions & Methods in Python

Cont..

☞ pop():

- In python, pop() element removes an element present at specified index from the list.

Syntax: `list.pop(index)`

Example: **popdemo.py**

```
num=[10,20,30,40,50]
num.pop()
print(num)
num.pop(2)
print(num)
num.pop(7)
print(num)
```

Output:

python popdemo.py

[10, 20, 30, 40]

[10, 20, 40]

IndexError: Out of range

List Functions & Methods in Python

Cont..

☞ clear():

- In python, clear() method removes all the elements from the list. It clears the list completely and returns nothing.

Syntax: `list.clear()`

Example: **cleardemo.py**

```
num=[10,20,30,40,50]
num.clear()
print(num)
```

Output:

python cleardemo.py

```
[]
```