CHAPTER -7
PYTHON LIST MANIPULATION

Bimlendu Kumar
PGT Computer Sc.
Kendriya Vidyalaya Garhara

INTRODUCTION

- Python Lists are containers that are used to store a list of values of any type.
- Python List are mutable i.e. we can change the elements of a list in place.
- Python do not create another list when a change is made in list in place.

We are going to discuss

1. Creating a list and accessing a list
2. Various operations on list
3. List manipulation with some built-in function.

CREATING AND ACCESSING LISTS

Points to Remember

1. A list is a standard data type of Python that can store a sequence of values belonging to any type.
2. The lists are depicted through square bracket.
3. Lists are mutable that is value of list can be changed in place.
4. List can contain values of mixed data types.
5. Lists are formed by placing a comma seperated list of expressions in square brackets.

```
blanklist = [ ]                                  #Empty List
intlist=[1,2,3,4,5]                              #list of integers
realist=[1.2, 2.5, 7.3, 8.9, 6.6]                #list of real numbers
characterlist= ['a','b','c']                     #list of characters
fruitlist=["mango", "apple", "grapes"]  #list of strings
recordlist=[1, 59.90, ,'m', "Sandeep", "cs"]     #List of mixed data types
```

THE EMPTY LIST

The Empty List: A list that does not have any element.

The empty list is [ ].

It is the list equivalent to '0' or '' and its also have truth value false.

```
>>>Emptylist1=[]
```

It can also be created as

```
>>Emptylist1=list()
>>>Emptylist1
[]
>>>Emptylist5
[]
```

LONG LIST AND NESTED LIST

Long List: A list containing many values.

```
>>>Emptylist1=[0, 1, 4, 9, 16, 25, 36, 49, 64, 81,
   100, 121, 144, 169, 196, 225, 256, 289, 324,
   361, 400, 441, 484, 529, 576, 625]
```

Nested Lists: A list containing some list as its value

```
>>>nestedlist=[2,4,6,[1,3,5],8,10]
```

LIST FROM TUPLE

creation of list from tuple

```
>>>tuple1=('W','E','L','C','O','M','E‘)
>>> List2=list(tuple1)
>>> List2
```

output

```
['W', 'E', 'L', 'C', 'O', 'M', 'E']
```

LIST CREATING FROM INPUT TAKEN BY USER

creation of list from keyboard input

```
>>> List3=list(input("Enter List Elements:- "))

Enter List Elements:- 123456789
>>> List3

['1', '2', '3', '4', '5', '6', '7', '8', '9']
```

LIST CREATING FROM INPUT TAKEN BY USER

creation of list from keyboard input

```
>>> List3=list(input("Enter List Elements:- "))
Enter List Elements:- 123456789
>>> List3
```

Output

```
['1', '2', '3', '4', '5', '6', '7', '8', '9']
```

Though we have typed digits but it is taken as string

```
>>> List4=eval(input("Enter 5 integers as elements of list :- "))
Enter 5 integers as elements of list :- 10,20,30,40,50
>>> List4
```

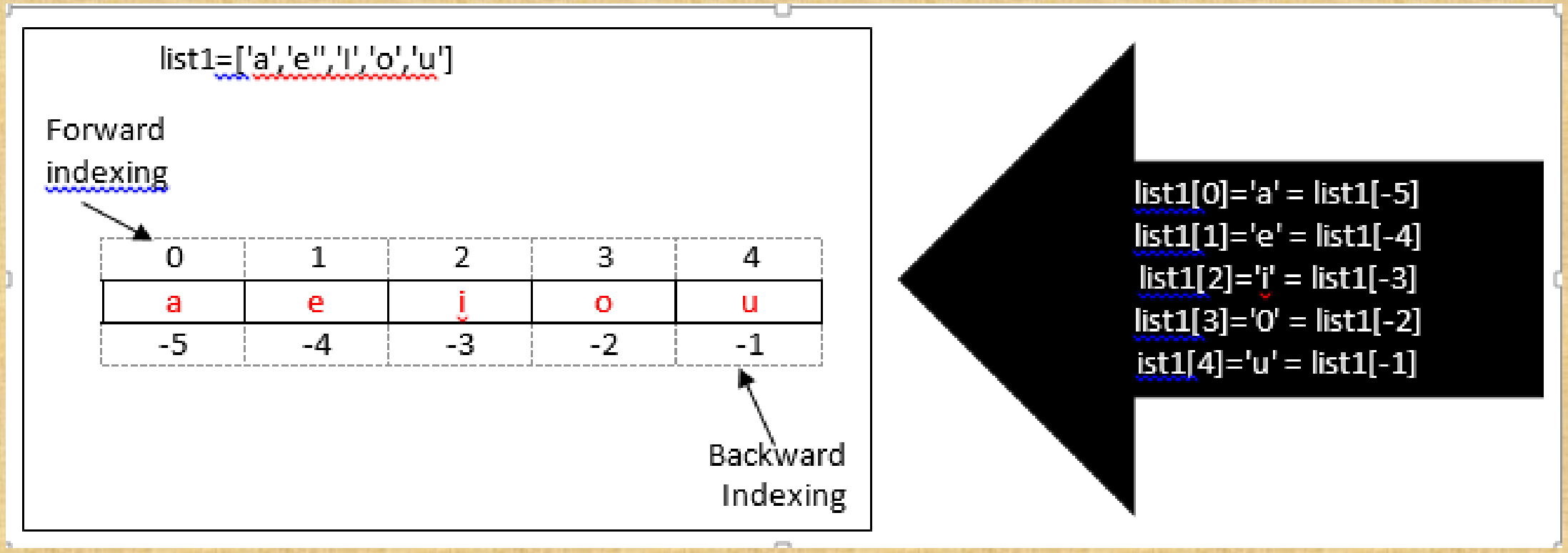
Output

```
(10, 20, 30, 40, 50)
```

ACCESSING LISTS

Lists are mutable sequences having a progression of elements. So, there must be a way to access its individual element. But before moving to this, let us discuss its smililarities with string.

1. **Lists are sequences just like strings. They also index their individual elemens just like strings. (Figure showing 2 way indexing)**



SMILILARITIES WITH STRING.

2. Length: Function len() returns the number of items in the list and it is same as that of string

```
>>> List=['C','o','m','p','u','t','e','r']
>>> print(len(List))
8

>>> str="Computer"
>>> print(len(str))
```

output:

```
8
```

SMILILARITIES WITH STRING

Smililarities with string.

3. Indexing:

List[i] will return the value at index i of the List. The first item has index 0.

Example

```
>>> print(List[1])
```

o #output → it is 2nd element of list whose index is 1

```
>>> print(str[1])
```

o #output →it is 2nd element of string whose index is 1

```
>>> print(List[-1])
```

r #output → it is last element of list whose index is 1

```
>>> print(str[-1])
```

r #output →it is last element of string whose index is 1

SMILILARITIES WITH STRING

Smililarities with string.

4. Slicing:

List[i:j] will return a new list, containing objects at indexes between i and j (including i but excluding j index)

Example

```
>>> print(List[0:5])
compu   #output → it is 2nd element of list whose index is 1
>>> print(str[0:5])
compu    #output →it is 2nd element of string whose index is 1
>>> print(List[-1:])
r   #output → it is last element of list whose index is 1
>>> print(str[-1])
r    #output →it is last element of string whose index is 1
```

SMILILARITIES WITH STRING.

Smililarities with string.

5. Concatenation and Replication Operators + and *:-

The + operator adds one list to the end of second list.

```
>>>List1=[10,20,30]
>>>List2=[40,50,60]
>>>List3=List1+List2
>>>print(List1)
[10, 20, 30]
>>>Print(List2)
[40, 50, 60]
>>>Print(List3)
[10, 20, 30, 40, 50, 60]
```

Replication of List using * operator

```
>>> vowels=['a','e','i','o','u']

>>> vowels*2

['a', 'e', 'i', 'o', 'u', 'a', 'e', 'i', 'o', 'u']
```

SMILILARITIES WITH STRING

Smililarities with string.

5. Membership Operators (in and not in):

```
>>>Str=”Computer”
>>> List1=['C','o','m','p','u','t','e','r']
>>> 'r' in str
Output
True
>>> 's' in str
Output
False
>>> 'r' in List
Output
True
>>> 's' in List
Output
False
```

```
>>> 'r' not in str
Output
False
>>> 's‘ not in str
Output
true
>>> 'r'  not in List
Output
False
>>> 's' not in List
Output
True
```

DIFFERENCE FROM STRING

Difference in list and String

Lists are mutable but string is immutable.

```
>>> vowels=['a','e','i','o','u']
>>> str="aeiou“
>>> vowels[4]='y‘       #changing list element in place
>>> vowels
['a', 'e', 'i', 'o', 'y']      #look changed list value.

>>> str[4]='h'            #trying to change element of string but not allowed
Traceback (most recent call last):
  File "<pyshell#88>", line 1, in <module>
    str[4]='h'
TypeError: 'str' object does not support item assignment
```

ACCESSING LISTS

Traversing a List

```
>>> vowels=['a','e','i','o','u']
```

Accessing List elements one by one using for loop

```
>>> for i in vowels:
        print(i)
```

output

```
a
e
i
o
u
```

ACCESSING LISTS

How loop works on List

```
>>> L=['Q','W','E','R','T','Y']
>>> length=Len(L)
>>> for a in range(length):
        print("at index ",a," and ",(a-length)," element is ",L[a])
```

```
at index  0  and  -6  element is  Q
at index  1  and  -5  element is  W
at index  2  and  -4  element is  E
at index  3  and  -3  element is  R
at index  4  and  -2  element is  T
at index  5  and  -1  element is  Y
```

ACCESSING LISTS

Comparing List

Two elements of a list can be compared using relational operators

```
>>> L1, L2=[10,20,30],[10,20,30]
>>> L1==L2
True
```

Comparison Result of two lists with explanation

Comparison	Result	Explanation
[1, 2, 8, 9]< [9, 1]	True	1 is less than 9
[1, 2, 8, 9] < [1, 2, 9, 8]	True	8 at 3rd place in list1 and 9 in list2
[1, 2, 8, 9] < [1, 2, 7, 8]	False	8 at 3rd place in list1 and 7 in list2

LIST OPERATIONS

1. Joining Lists: Two or more lists can be concatenated using + operator in between the list operands.

```
>>>List1=[1, 2, 3]
>>>List2=[4, 5, 6]
>>>list3=[7, 8, 9]
>>>List1+List2
[1, 2, 3, 4, 5, 6]
>>>List1+List2+List3
[1, 2, 3, 4, 5, 6 , 7, 8, 9]
```

Both the operands must be of list type. Following list operations not allwed

List+Number

List+Complex Number

List+String

LIST OPERATIONS

2. Repeating or Replicating Lists: A list can be replicated or repeated an integer number of times.

```
>>> list=[10,20]
>>> list*2
[10, 20, 10, 20]

>>> list*2.5
Traceback (most recent call last):
  File "<pyshell#118>", line 1, in <module>
    list*2.5
TypeError: can't multiply sequence by non-int of type 'float'
```

LIST OPERATIONS

3. Slicing the Lists: List slices are like string slices and are the subpart of a list extracted out. We can use indexes of list elements to create list slices as per following format

seq=L[start:stop]

```
>>> Lst=[10, 12, 14, 20, 22, 24, 30, 32, 34]
>>> seq=Lst[3:-3]     #slicing index 3 not included
>>> seq
[20, 22, 24]
>>> seq[1]=28
>>> seq
[20, 28, 24]
>>> Lst[3:30]  #since there are not 30 indexes hence wil extract elemenst starting from index 3 to the end of the list
[20, 22, 24, 30, 32, 34]
```

LIST OPERATIONS

3. Slicing the Lists:

```
>>> Lst[-17:7]  #Starting index is very low but Python will start form -15 and will extract element onward < 7
[10, 12, 14, 20, 22, 24, 30]
>>>Lst[10:20]
[]  #since no element falls in between given indexes
```

Note: L[Start:Stop] creates a list slice with elements falling between Start and Stop indexes (Stop index not included) skipping step 1 elements in between

```
>>>Lst[Start:Stop:Skip]
```

Example

```
>>>Lst[0:10:2]
[10, 14 , 22, 30, 34]   #Look 2 alternate elements are extracted as skip is 2
>>>Lst[::3]  #Start and Stop not given ony skip is given hence it will pick every 3rd element from the list
[10, 20, 30]
```

LIST OPERATIONS

3. Slicing the Lists:

```
Seq1=Lst[::2]  # Seq1 will have every second item of the list
Seq2=Lst[5::2] #Seq2 will have every second element starting from
                # index 5 i.e. sixth element
>>>Lst[::-1]      # Will reverse the list
```

WORKING WITH LISTS

1. Appending Element to a List:

append() function is used to append item to the list. Its general syntax is

List.append(item)

```
>>>List1=[10,20]
>>>List1.append(30)
>>>List1
[10, 20, 30]
```

WORKING WITH LISTS

2. Updating Element to a List:

To update or change an element of List in place we just have to assign new value to the element's index in the list as per syntax given below

List[index]=<new value>

```
>>>List1=[10, 20, 30, 40, 45]
>>>List1
[10, 20, 30, 40, 45]
>>>List1[4]=50
>>>List1
[10, 20, 30, 40, 50]
```

WORKING WITH LISTS

3. Deleting Element from a List:

To remove item from list **del** statement can be used. It can

(a) Remove single element

(b) Remove multiple items identified by list slicingSyntax

Syntax

(a) del List[Index]

Example:

```
>>>List1=[1, 2, 3, 4, 5, 6, 7 , 8, 9, 10]
>>>List1
[1, 2, 3, 4, 5, 6, 7 , 8, 9, 10]
>>>del List1[5]
>>>List1
[1, 2, 3, 4, 5, 7 , 8, 9, 10]    #6th element is deleted from list
```

WORKING WITH LISTS

3. Deleting Element from a List:

To remove item from list **del** statement can be used. It can

(b) Remove multiple items identified by list slicingSyntax

Syntax

(a) del List[start:stop]

Example:

```
>>>List1=[1, 2, 3, 4, 5, 6, 7 , 8, 9, 10]
>>>List1
[1, 2, 3, 4, 5, 6, 7 , 8, 9, 10]
>>>del List1[3:5]
>>>List1
[1, 2, 3, 6, 7 , 8, 9, 10]    #6th element is deleted from list
```

WORKING WITH LISTS

3. Deleting Entire List:

Use del <listname> command to delete entire list.

```
>>>list1=[1,2,3,4,5]
>>>list1
[1,2,3,4,5]
>>>del list1  #all elements of list as well as list object deleted
>>>list1
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    list1
NameError: name 'list1' is not defined
```

WORKING WITH LISTS

3. Deleting an element of List using pop() method:

pop() method can also be used to remove / delete one element from specified index position of the list like del command but it also returns the deleted value that can be stored in some variable and can be used later on.

Syntax List.pop(index)

if Index is skipped last element is deleted from list.

```
>>> List1=[1,2,3,4,5,6,7,8,9,10]
>>> List1
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> List1.pop()
10
>>> List1
[1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> List1.pop(0)
1
>>> List1
[2, 3, 4, 5, 6, 7, 8, 9]
```

LIST FUNCTIONS AND METHODS

Python offers many built-in-functions and methods for list manipulation. Some of them are listed below

Function	Syntax	Example

index()
It returns the index position of the element in the list if found oyherwise error
	List.index(item in list)	>>>
		List1=[10,20,30,40,50,60,70,80,90,100]
		>>>List1
		[10,20,30,40,50,60,70,80,90,100]
		>>> List1.index(40)
		3
		>>> List1.index(120)
		Traceback (most recent call last):
		File "<pyshell#18>", line 1, in <module>
		List1.index(120)
		ValueError: 120 is not in list

LIST FUNCTIONS AND METHODS

Function	Syntax	Example

append()
It appends the given item to the end of the list the new list
note: append() does not return the new list
	List.append()	>>> List1=[10,20,30,40,50,60,70,80,90,100]
		>>>List1
		>>>List1.append(110)
		>>>List1
		[10,20,30,40,50,60,70,80,90,100,110]
		Note:
		>>>List3=List1.append(120)
		>>>List3
		[ ] //Blank because append do not return anything
		>>> Lis1.append(120,130) #error
		Traceback (most recent call last):
		File "<pyshell#22>", line 1, in <module>
		Lis1.append(120,130)
		NameError: name 'Lis1' is not defined

LIST FUNCTIONS AND METHODS

Function	Syntax	Example
extend() extend() method is used to append a list to the existing list but it also does not return any value.	List1.extend(List)	>>>List1=[10,20,30] >>>List2=[40,50,60] >>>List1 [10,20,30] >>>List2 [40,50,60] >>>List1.extend(List2) >>>List1 [10,20,30,40,50,60] >>List2 [40,50,60] >>>List3=List1.extend(List2) >>>List3 [] # empty

LIST FUNCTIONS AND METHODS

Function	Syntax	Example
extend() extend() method is used to append a list to the existing list but it also does not return any value.	List1.extend(List)	>>>List1=[10,20,30] >>>List2=[40,50,60] >>>List1 [10,20,30] >>>List2 [40,50,60] >>>List1.extend(List2) >>>List1 [10,20,30,40,50,60] >>List2 [40,50,60] >>>List3=List1.extend(List2) >>>List3 [] # empty >>>List1.extend(130) #Error it can not add one element rather it requires list. Either Provide a list or list object >>>List1.extend([120,130]) #OK it will work

LIST FUNCTIONS AND METHODS

Function	Syntax	Example

insert()
insert() method is used to insert in between or any position of your choice.
	List1.insert(index, item)	>>>List1=[10,20,30]
		>>>List1.insert(2,25)
		>>>List1
		[10,20,25,30]
		>>>List1.insert(0,5)
		>>>List1
		[5,10,20,25,30]
		>>> List1.insert(len(List1),40)
		>>> List1
		[5, 10, 20, 25, 30, 40]

LIST FUNCTIONS AND METHODS

Function	Syntax	Example

pop()
pop() method removes data from the specified index position of the list. It also returns the data popped.
	List.pop(index)
	List.pop()
	#if without index used pop() function will remove last element from the list
		>>> List1=[10,20,30,40,50]
		>>> List1
		[10,20,30,40,50]
		>>>List1.pop(0)
		10
		>>>List1
		[20,30,40,50]
		>>>List.pop()
		50
		It can not pop data from empty list
		>>>List2=[]
		>>>List2.pop() #Error

LIST FUNCTIONS AND METHODS

Function	Syntax	Example

remove() remove() method removes the first occurrence of the instance from the specified list. It does not return anything.	List.remove(<value>)	>>> List1=[10,20,30,40,50, 30, 90]
		>>> List1
		[10,20,30,40,50, 30, 90]
		>>>List1.remove(30)
		>>>List1
		[10,20,40,50,30,90]
		>>>List.remove(150)
		Traceback (most recent call last):
		File "<pyshell#11>", line 1, in <module>
		List1.remove(150)
		ValueError: list.remove(x): x not in list

LIST FUNCTIONS AND METHODS

Function	Syntax	Example
clear() clear() method removes all the items from the list. Unlike del clear removes only the items of the list and not the list itself.	List.clear()	>>>List1=[10,20,30] >>>List1 [10,20,30] >>>List1.clear() >>>List1 [ ] #empty list

LIST FUNCTIONS AND METHODS

Function	Syntax	Example
count() count() method return the number of occurrence of the items in the list which has been provided as argument to the function	List.count(item)	>>>List1=[10,20,30,10] >>>List1 [10,20,30,10] >>>List1.count(10) 2 >>>List1.count(100) 0

LIST FUNCTIONS AND METHODS

Function	Syntax	Example
reverse() reverse() method reverse the list in place.	List.reverse()	>>>List1=[10,20,30] >>>List1 [10,20,30,10] >>>List1.reverse() >>>List1 [30,20,10]

sort()
sort() method sorts the list in ascending order by default. it can also be used for sorting in descending order.
	List.sort()
	# Sort in ascending order
	List.sort(reverse=True)
		>>>List1=[15,5,25,20,40,60,50]
		>>>List1
		[15,5,25,20,40,60,50]
		>>>List1.sort()
		>>>List1
		[5,15,20,25,40,50,60] #sorted in ascending order
		>>>List1.sort(reverse=True)
		>>>List1
		[60,50,40,25,20,15,5]

LIST PROGRAMS

```
"""“  Program to minimum and maximum element in a list  """
lst=eval(input("Enter List:- "))
length=len(lst)
min=lst[0]
minindex=0
max=lst[0]
maxindex=0
for i in range(1, length-1):
    if lst[i]<min:
        min=lst[i];
        minindex=i;
    if lst[i]>max:
        max=lst[i]
        maxindex=i;
print("Given list is ",lst)
print("smallest value in list = ",min," and its index is ",minindex)
print("Largest value in list - ",max," ans its index is ",maxindex)
```

Output

```
Enter List:- [2,3,4,-2,6,-7,8,11,-9,11]
Given list is  [2, 3, 4, -2, 6, -7, 8, 11, -9, 11]
smallest value in list =  -9  and its index is  8
Largest value in list -  11  ans its index is  7
```

LIST PROGRAMS

```
"""
Program to find mean of the list
"""
lst=eval(input("Enter List:- "))
length=len(lst)
mean=sum=0
for i in range(0, length-1):
    sum+=lst[i]
mean=sum/length
print("Given list is ",lst)
print("Mean value is ",mean)
```

==========

```
Enter List:- [10,20,30,40,50]
Given list is  [10, 20, 30, 40, 50]
Mean value is  20.0
```

Thanks for Watching This Presentation