LIST

LISTS — CHANGEABLE SEQUENCES OF DATA

The most basic data structure in Python is the sequence. Each element of a sequence is assigned a number - its position or index. The first index is zero, the second index is one, and so forth. There are certain things you can do with all the sequence types. These operations include *indexing, slicing, adding, multiplying, and checking for membership*. In addition, Python has *built-in functions* for finding the length of a sequence and for finding its largest and smallest elements.

Python has six built-in types of sequences, but the most common ones are lists and tuples.

- Python provides a list for collection manipulation.
- We define a list as [].
- A list is a *value that contains multiple values* in an ordered sequence.
- The term list value refers to the list itself (which is a value that can be stored in a variable or passed to a function like any other value), not the values inside the list value.
- Lists are sequences that contain elements referenced starting at zero.
- The list can be changed at any time. THEY ARE MUTABLE OBJECTS

LIST

CREATING

Creating a list is as simple as putting different comma-separated values between square brackets. For example

```
list1= ['physics', 'chemistry', 1997, 2000]
list2 = [1, 2, 3, 4, 5 ]
list3 = ["a", "b", "c", "d"]
```

Similar to string indices, list indices start at 0, and lists can be sliced, concatenated and so on.

ACCESSING VALUES IN LISTS

To access values in lists, use the square brackets for slicing along with the index or indices to obtain value available at that index. For example-

```
list1 = ['physics', 'chemistry', 1997, 2000]
list2 = [1, 2, 3, 4, 5, 6, 7 ]
print ("list1[0]: ", list1[0])
print ("list2[1:5]: ", list2[1:5])
```

When the above code is executed, it produces the following result –

```
list1[0]: physics
list2[1:5]: [2, 3, 4, 5]
```

UPDATING LISTS

You can update single or multiple elements of lists by giving the slice on the left-hand side of the assignment operator, and *you can add to elements in a list with the* append() method. For example-

```
list = ['physics', 'chemistry', 1997, 2000]
print ("Value available at index 2 : ", list[2])
list[2] = 2001
print ("New value available at index 2 : ", list[2])
```

When the above code is executed, it produces the following result –

```
Value available at index 2 :
1997
New value available at index 2 :
2001
```

DELETE LIST ELEMENTS

To remove a list element, you can use either the **del** statement if you know exactly which element(s) you are deleting. You can use the **remove()** method if you do not know exactly which items to delete. For example-

```
list = ['physics', 'chemistry', 1997, 2000]
print (list)
del list[2]
print ("After deleting value at index 2 : ", list)
```

When the above code is executed, it produces the following result-

```
['physics', 'chemistry', 1997, 2000]
After deleting value at index 2 : ['physics', 'chemistry', 2000]
```

LIST

BASIC LIST OPERATIONS

Lists respond to the + and * operators much like strings; they mean concatenation and repetition here too, except that the *result is a new list*, not a string.

Python Expression	Results	Description
len([1, 2, 3])	3	Length
[1, 2, 3] + [4, 5, 6]	[1, 2, 3, 4, 5, 6]	Concatenation
['Hi!'] * 4	['Hi!', 'Hi!', 'Hi!', 'Hi!']	Repetition
3 in [1, 2, 3]	True	Membership
for x in [1,2,3] : print (x,end='')	1 2 3	Iteration

Indexing, Slicing and Matrixes

Since lists are sequences, indexing and slicing work the same way for lists as they do for strings. Assuming the following input-

```
L=['C++'', 'Java', 'Python']
```

Python Expression	Results	Description
L[2]	'Python'	Offsets start at zero
L[-2]	'Java'	Negative: count from the right
L[1:]	['Java', 'Python']	Slicing fetches sections

Indexing Example: use IDLE:

```
# Print out a date, given year, month, and day as numbers

mth = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September',
'October', 'November', 'December']

# A list with one ending for each number from 1 to 31

superscript = ['st', 'nd', 'rd'] + 17 * ['th'] + ['st', 'nd', 'rd'] + 7 * ['th'] + ['st']

year = int(input('Year: '))
month = int(input('Month (1-12): '))
day = int(input('Day (1-31): '))

# Remember to subtract 1 from month and day to get a correct index

month_name = mth[month-1]
ordinal = str(day) + superscript[day-1]
print (month_name + ' ' + ordinal + ', ' + str(year))
```

BUILT-IN LIST FUNCTIONS

Python includes the following list functions-

Function	Description
cmp(list1, list2)	No longer available in Python 3.
len(list)	Gives the total length of the list.
max(list)	Returns item from the list with max value.
min(list)	Returns item from the list with min value.
list(seq)	Converts a tuple into list.

len() method.

```
list1 = ['physics', 'chemistry', 'maths']
print (len(list1))
list2=list(range(5))                #creates list of numbers between 0-4
print (len(list2))
```

When we run above program, it produces following result-

```
3
5
```

max() method.

```
list1, list2 = ['C++','Java', 'Python'], [456, 700, 200]
print ("Max value element : ", max(list1))
print ("Max value element : ", max(list2))
```

When we run above program, it produces following result-

```
Max value element : Python
Max value element : 700
```

min() method.

```
list1, list2 = ['C++','Java', 'Python'], [456, 700, 200]
print ("min value element : ", min(list1))
print ("min value element : ", min(list2))
```

When we run above program, it produces following result

```
Min value element : C++
min value element : 200
```

LIST

list() method.

```
aTuple = (123, 'C++', 'Java', 'Python')
list1 = list(aTuple)
print ("List elements : ", list1)

str="Hello World"
list2=list(str)
print ("List elements : ", list2)
```

When we run above program, it produces following result-

```
List elements : [123, 'C++', 'Java', 'Python']
List elements : ['H', 'e', 'l', 'l', 'o', ' ', 'W', 'o', 'r', 'l', 'd']
```

BUILT-IN LIST METHODS

Python includes the following list methods-

Methods	Description
list.append(obj)	Appends object obj to list…. Returns None
list.count(obj)	Returns count of how many times obj occurs in list
list.extend(seq)	Appends the contents of seq to list…. Returns None
list.index(obj)	Returns the lowest index in list that obj appears
list.insert(index, obj)	Inserts object obj into list at offset index
list.pop(obj=list[-1])	Removes and returns last object or obj from list as per the position
list.remove(obj)	Removes object obj from list .. Returns None
list.reverse()	Reverses objects of list in place … Returns None
list.sort([func])	Sorts objects of list in place, use compare func if given

count() method.

```
aList = [123, 'xyz', 'zara', 'abc', 123];
print ("Count for 123 : ", aList.count(123))
print ("Count for zara : ", aList.count('zara'))
```

When we run the above program, it produces the following result-

```
Count for 123 : 2
Count for zara : 1
```

extend() method.

```
list1 = ['physics', 'chemistry', 'maths']
list2=list(range(5))                    #creates list of numbers between 0-4
list1.extend(list2)
print (list1)
```

When we run the above program, it produces the following result-

```
Extended List : ['physics', 'chemistry', 'maths', 0, 1, 2, 3, 4]
```

index() method.

```
list1 = ['physics', 'chemistry', 'maths']
print ('Index of chemistry', list1.index('chemistry'))
print ('Index of C#', list1.index('C#'))
```

When we run the above program, it produces the following result-

```
Index of chemistry 1
Traceback (most recent call last):
File "test.py", line 3, in print ('Index of C#', list1.index('C#'))
ValueError: 'C#' is not in list
```

insert() method.

```
list1 = ['physics', 'chemistry', 'maths']
list1.insert(1, 'Biology')
print ('Final list : ', list1)
```

When we run the above program, it produces the following result

```
Final list : ['physics', 'Biology', 'chemistry', 'maths']
```

pop() method.

```
list1 = ['physics', 'Biology', 'chemistry', 'maths']
list1.pop()
print ("list now : ", list1)

list1.pop(1)
print ("list now : ", list1)
```

When we run the above program, it produces the following result

```
List now : ['physics', 'Biology', 'chemistry']
list now : ['physics', 'chemistry']
```

remove() method.

```
list1 = ['physics', 'Biology', 'chemistry', 'maths']
list1.remove('Biology')
print ("list now : ", list1)
list1.remove('maths')
print ("list now : ", list1)
```

When we run the above program, it produces the following result

```
List now : ['physics', 'chemistry', 'maths']
list now : ['physics', 'chemistry']
```

reverse() method.

```
list1 = ['physics', 'Biology', 'chemistry', 'maths']
list1.reverse()
print ("list now : ", list1)
```

When we run above program, it produces following result

```
list now : ['maths', 'chemistry', 'Biology', 'physics']
```

sort() method.

```
list1 = ['physics', 'Biology', 'chemistry', 'maths']
list1.sort()
print ("list now : ", list1)
```

When we run the above program, it produces the following result

```
List now : ['Biology', 'chemistry', 'maths', 'physics']
```

Sorted()

Another way of getting a sorted copy of a list is using the sorted function:

```
>>> x = [4, 6, 2, 1, 7, 9]
>>> y = sorted(x)
>>> x
[4, 6, 2, 1, 7, 9]
>>> y
[1, 2, 4, 6, 7, 9]
```

LIST

Operation	Interpretation
L = []	An empty list
L = [123, 'abc', 1.23, {}]	Four items: indexes 0..3
L = ['Bob', 40.0, ['dev', 'mgr']]	Nested sublists

L = list('spam')
L = list(range(-4, 4))
	List of an iterable's items, list of successive integers

L[i] L[i][j] L[i:j] len(L)	Index, index of index, slice, length
L1 + L2 L * 3	Concatenate, repeat
for x in L: print(x) 3 in L	Iteration, membership

L.append(4)
L.extend([5,6,7])
L.insert(i, X)
	Methods: growing

L.index(X)
L.count(X)
	Methods: searching

L.sort()	Methods: sorting, reversing

L.reverse()
L.copy()
L.clear()
	copying (3.3+), clearing (3.3+)

L.pop(i) L.remove(X) del L[i] del L[i:j] L[i:j] = []	Methods, statements: shrinking
L[i] = 3 L[i:j] = [4,5,6]	Index assignment, slice assignment