

**St. Aloysius Sr. Sec. School, Cantt., Jabalpur**

**Class 12 (IP)**

**Unit 1 : (week 4)**

**More of Python**

**Dated – 27/04/2020**

**Welcome children –**

In class XI, you had studied the concepts of Python and many more concepts in it. We have already covered our topics in the second week, till loops and their types. At the end of the learning there will be questions for you to examine yourself.

**Introduction –**

As you already know that strings are immutable and thus we cannot make changes in the stored values although we can extract the string and its values i.e. cell values using indexing and slicing methods.

There are other data types that can be applied in programming. One of the most commonly used is the list data type. The list data type is quite easy to learn and also to apply.

In a list the values are stored in a sequence. A sequence is an object that contains multiple items of data. The items are stored in a sequence one after the other such as a sequence of items, movements, events or sequence of records. The number of elements is called the length of the sequence. There are other data types also which helps to store values in a sequence.

The values stored in a list can be of similar types or dissimilar types. It depends on the nature and need of the programmer. Each value in a list is stored in a single cell and can be accessed in a sequence.

## 6.2 LISTS

A list is a collection of values or an ordered sequence of values/items. The items in a list can be of any type such as string, integers, float, objects, or even a list. Elements of a list are enclosed in **square brackets** [ ], separated by commas. Values in the list can be modified, *i.e.*, it is mutable. The values that make up a list are called its elements. Elements in a list need not be of the same type. In other words, we say that lists are heterogeneous (of multiple types) in nature. Like a string (sequence of characters), list also is a sequenced data type.

**CTM:** A list is a collection of comma-separated values (items) within square brackets. Items in a list need not be of the same type.

### 6.2.1 Declaring/Creating List

Lists can be created by putting comma-separated values/items within square brackets.

The syntax for creating a list is:

```
<list_name> = []
```

Square brackets in the above syntax contain a set of values/elements constituting the list with name given as <list\_name>. This is termed as **initialization** of the list with a blank or no values or we can say an **empty list**. A list can be populated with any type of values/collection of items by enclosing inside square brackets [ ] as shown in the example given below:

```
Fruits = ['Mango', 'Apple', 'Grapes']
```

In the above example, we have a list of fruits having three items such as Mango, Apple, and Grapes, separated by comma and enclosed inside square [ ] brackets. Other examples of lists are:

#### > List Types and Examples:

```
list1=[10, 20, 30, 40, 50]
list2=["Deep", 450, 30.4, "Python", -2200]
list3=['A', 'E', 'I', 'O', 'U']
list4=[]
list5= ["Delhi", "Mumbai", "Chennai"]
list6= [3, 4, [5,6,7], 8]
```

**Learning Tip:** Make sure that strings are enclosed in single/double quotes. Python can't differentiate between single or double quotes. So, you can use whatever you want.

#list of Integers

#list with elements of different data types

#list of characters

#empty list, *i.e.*, list with no element

#list of 3 strings

#list containing another list as an element, *i.e.*, Nested lists

```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32
bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> list1= [10,20,30,40,50] #list of integers
>>> list1
[10, 20, 30, 40, 50]
>>> list2 = ["Deep",450,30.4,"python",-2200] #list with elements of dif
ferent data types
>>> list2
['Deep', 450, 30.4, 'Python', -2200]
>>> list3 = ['A','E','I','O','U'] #list of characters
>>> list3
['A', 'E', 'I', 'O', 'U']
>>> list4 = [] #empty list, i.e., list with no element
>>> list4
[]
>>> list5= ["Delhi","Mumbai","Chennai"] #list of 3 strings
>>> list5
['Delhi', 'Mumbai', 'Chennai']
>>> list6= [3,4,[5,6,7],8] #list containing another list as an element
, i.e., Nested lists
>>> list6
[3, 4, [5, 6, 7], 8]
>>>
```

Fig. 6.3: Creating Different Types of Lists

### ➤ Creating a List from an existing sequence

The list() method in Python takes sequence types and converts a given record/tuple or string into list. We can create a list from an existing sequence using this built-in function list() in the following manner:

1. Creating a list from a sequence (string):

```
<new_list_name> = list(sequence/string)
```

For example,

```
list1=list('Computer')
```

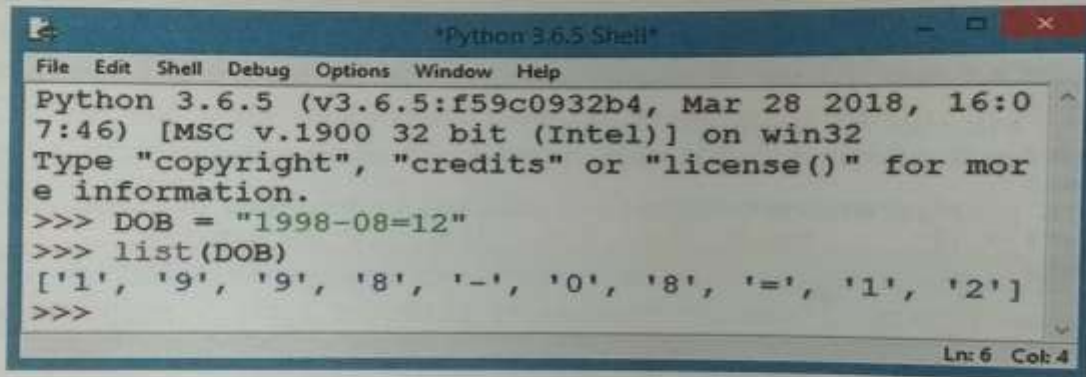
```
>>>list1
```

```
>>['C','o','m','p','u','t','e','r']
```

```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
Type "copyright", "credits" or "license()" for more inform
ation.
>>> list1= list('Computer')
>>> list1
['C', 'o', 'm', 'p', 'u', 't', 'e', 'r']
>>> list2= list() #Creating an empty list
>>> list2
[]
>>> type(list2)
<class 'list'>
>>>
```

In the given example, a new list, list1 is created from a sequence 'Computer' and displayed by typing the name of the list, *i.e.*, >>>list1 as shown in Fig. 6.4(a).

Another example is for accepting date of birth as input and converting it into a list using list() method as shown in Fig. 6.4(b).



```
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> DOB = "1998-08=12"
>>> list(DOB)
['1', '9', '9', '8', '-', '0', '8', '=', '1', '2']
>>>
```

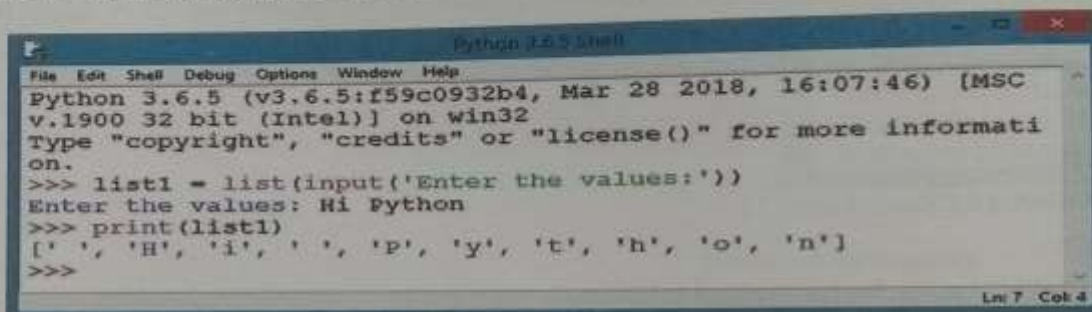
**Fig. 6.4(b):** Converting Date of Birth into a List using list() Function

2. An empty list with function list(): Fig. 6.4(a).

```
>>>list2= list()
>>>list2
[]
```

3. Creating a list through user input using list() method:

You can create a new list by extracting the elements from the user through command prompt and using built-in method list() as shown below:



```
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> list1 = list(input('Enter the values:'))
Enter the values: Hi Python
>>> print(list1)
[' ', 'H', 'i', ' ', 'P', 'y', 't', 'h', 'o', 'n']
>>>
```

**POINTS TO REMEMBER**

- print() function can be used to display an entire list.
- list() function can convert certain types of objects into lists.

➤ **Creating a List from an existing list**

You can create a new list from an already existing list as follows:

Consider list1 created in the previous example, list1= [10, 20, 30, 40, 50]. Now, we shall be creating new lists by extracting values from this list.

Python statement	Description
<code>list5 = list1[:]</code>	list5 is a copy of list1. list5 = [10, 20, 30, 40, 50]
<code>list6 = list1[1:4]</code>	list6 shall contain elements 1, 2, 3 from list1. list6 = [20, 30, 40]
<code>list7 = list1</code>	list7 shall have all the elements present in list1. list7 = [10, 20, 30, 40, 50]

## 6.2.2 Accessing List Elements

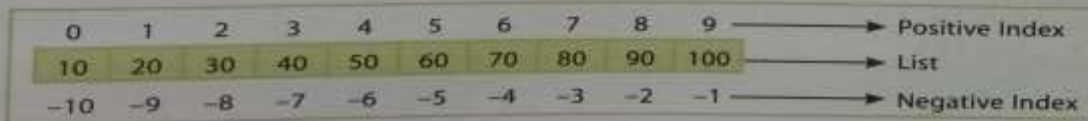
A list consists of any collection of items which are stored according to its index. Like strings, any item in a list can be accessed by using its index value.

List indices start with 0 (zero). The list index can be a positive or a negative integer value or an expression which evaluates to an integer value. Hence, indexing in lists can be positive or negative, *i.e.*, you can access the elements from a list either moving from left to right (positive index) or from right to left (negative index). The index of -1 refers to the last item, -2 to the second last item and so on.

For example, consider list1 containing 10 elements.

```
list1 = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
```

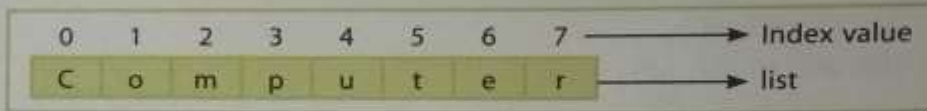
This list in computer memory shall be represented as shown in Fig. 6.5 below:



Each element of the above list can be accessed through their indexes enclosed inside square brackets.

```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v
.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more informatio
n.
>>> list1 = [10,20,30,40,50,60,70,80,90,100]
>>> list1
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
>>> list1[0]
10
>>> list1[6]
70
>>> list1[9]
100
>>> list1[-10]
10
>>> list1[-5]
60
>>>
```

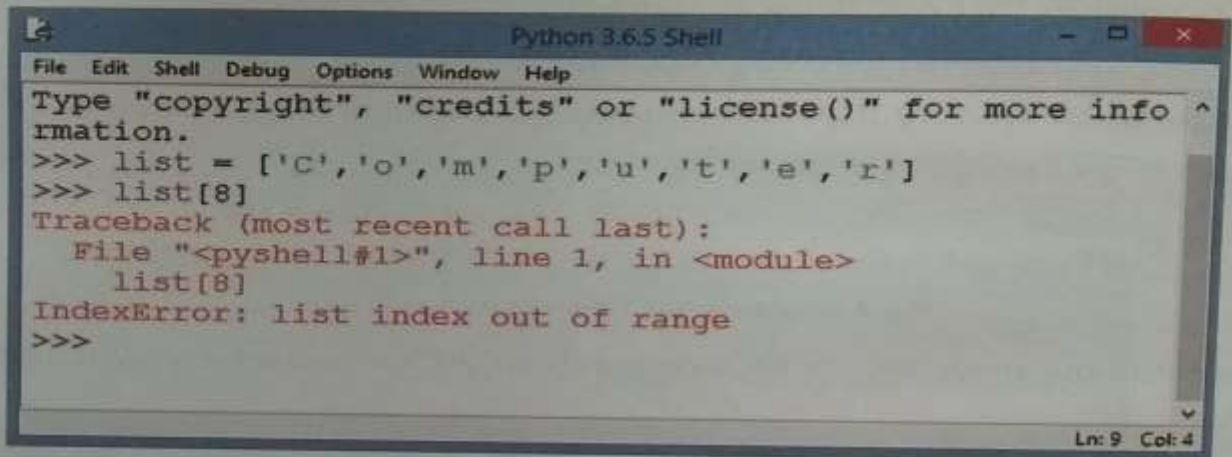
Consider another list with elements, 'C','o','m','p','u','t','e','r' stored inside memory as shown:



```
>>> list=['C','o','m','p','u','t','e','r']
>>>list[3]
>>>'p'
```

```
===== RESTART: Shell =====
>>> list= ['C','o','m','p','u','t','e','r']
>>> list[3]
'p'
>>>
```

> An Index Error appears if you try and access an element that does not exist in the list. It will display the error message as: IndexError: list index out of range



```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
Type "copyright", "credits" or "license()" for more info ^
>>> list = ['C','o','m','p','u','t','e','r']
>>> list[8]
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    list[8]
IndexError: list index out of range
>>>
```

Now, in the above code, we are trying to access eighth element (>>>list[8]) from the list which does not exist as we are having a total of 8 elements for which the last index is 7. Since we are trying to access the element at index 8, Python will give an "IndexError".

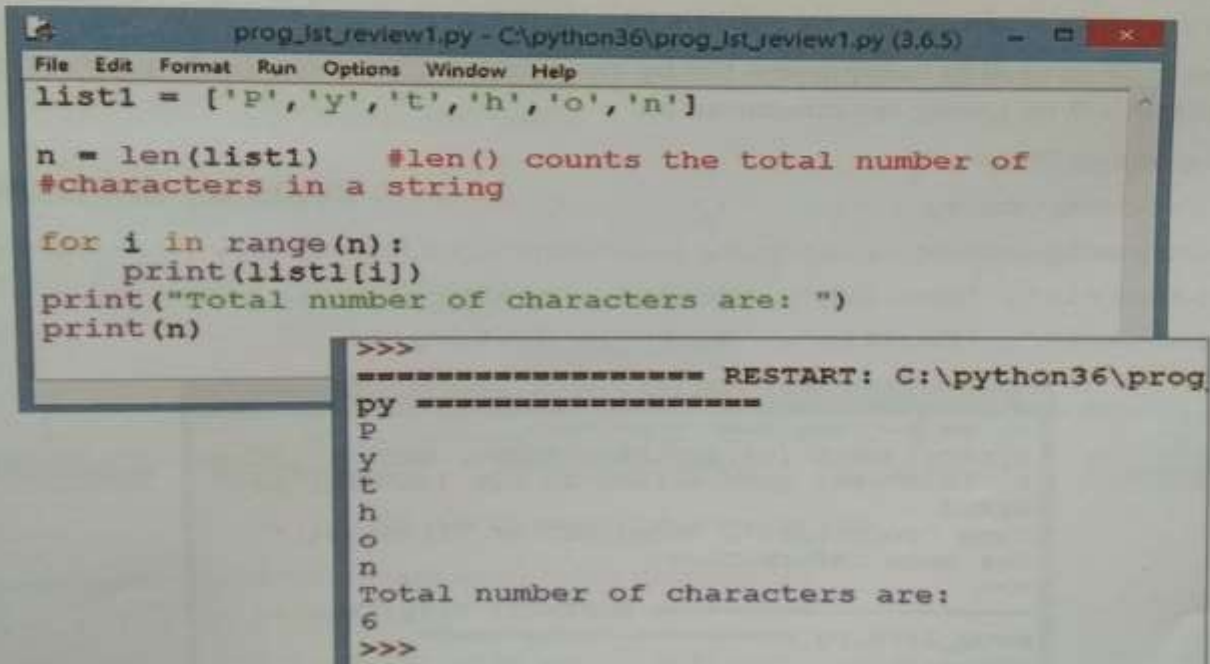
**Example:**    my\_list = ['p','r','o','b','e']  
              print(my\_list[-1]) Output: e  
              print(my\_list[-5]) Output: p

### 6.2.3 Traversing a List

Traversing a list means accessing each element of a list. This can be done by using either for or while looping statement. Traversing can be done in the following two ways:

## 2. Using range() function

We have already discussed this function in Chapter-4. It checks for the given set of values mentioned as the argument to range() function. This function can be used for accessing the individual elements of a list.



```
prog_lst_review1.py - C:\python36\prog_lst_review1.py (3.6.5)
File Edit Format Run Options Window Help
list1 = ['P','y','t','h','o','n']

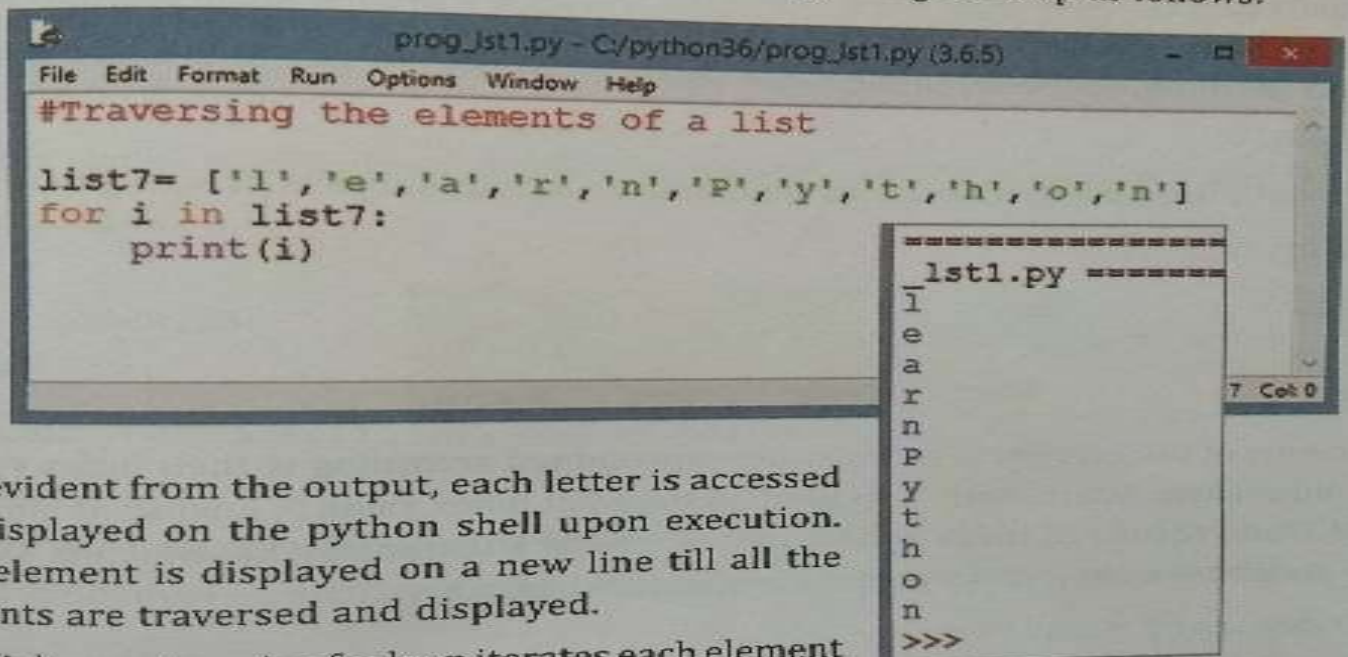
n = len(list1)    #len() counts the total number of
#characters in a string

for i in range(n):
    print(list1[i])
print("Total number of characters are: ")
print(n)

>>>
===== RESTART: C:\python36\prog
py =====
P
Y
t
h
o
n
Total number of characters are:
6
>>>
```

## 1. Using 'in' operator inside for loop

The elements of a list can be accessed individually using for loop as follows:



```
prog_lst1.py - C:\python36\prog_lst1.py (3.6.5)
File Edit Format Run Options Window Help
#Traversing the elements of a list

list7= ['l','e','a','r','n','P','y','t','h','o','n']
for i in list7:
    print(i)

=====
lst1.py =====
l
e
a
r
n
P
y
t
h
o
n
>>>
```

As is evident from the output, each letter is accessed and displayed on the python shell upon execution. Each element is displayed on a new line till all the elements are traversed and displayed.

Here, 'in' operator using for loop iterates each element of a list in sequence.

## **Built in functions :**

Python offers several built-in functions to perform various operations on lists which are –

1. append()
2. extend()
3. insert()
4. index()
5. remove()
6. sort()
7. reverse()

In order to remove a complete list you can also try the del command which can remove the complete list or part of it (sliced).

## **TASK for you**

Study the examples of the given function above from your class XI text book.

\*\*\*\*\*End\*\*\*\*\*