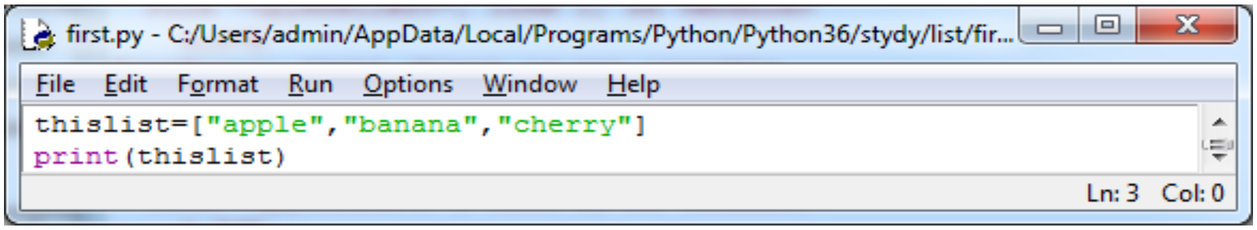
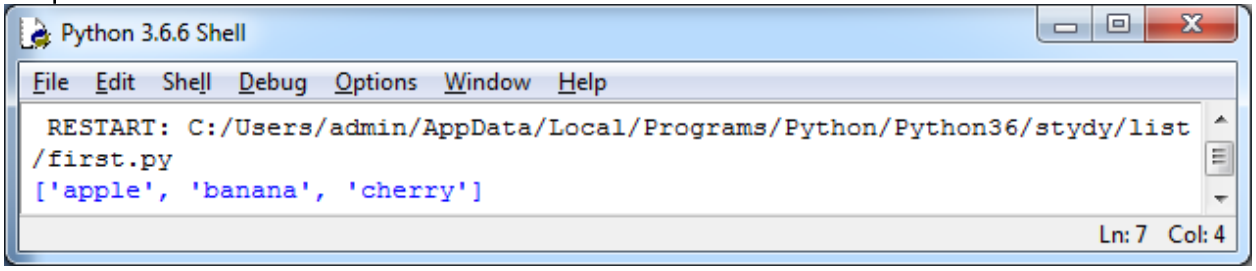
LIST

Like a String, list also is sequence data type. It is an ordered set of values enclosed in square brackets []. Values in the list can be modified, i.e. it is mutable. As it is set of values, we can use index in square brackets [] to identify a value belonging to it. The values that make up a list are called its elements, and they can be of any type.
Example:


first.py - C:/Users/admin/AppData/Local/Programs/Python/Python36/stydy/list/fir...
File Edit Format Run Options Window Help
Ln: 3 Col: 0

```
thislist=["apple","banana","cherry"]
print(thislist)
```

Output:



```
 RESTART: C:/Users/admin/AppData/Local/Programs/Python/Python36/stydy/list
/first.py
['apple', 'banana', 'cherry']
```

For accessing an element of the list, indexing is used.
Its syntax is:
Variable name [index] (variable name is name of the list).
It will provide the value at „index+1‟ in the list. Index here, has to be an integer value which can be positive or negative. Positive value of index means counting forward from beginning of the list and negative value means counting backward from end of the list.

Let‟s look at some example of simple list:
i) L1 = [1, 2, 3, 4] # list of 4 integer elements.
ii)L2 = ["Delhi", "Chennai", "Mumbai"] #list of 3 string elements.
iiiL3 = [ ] # empty list i.e. list with no element
iv)L4 = ["abc", 10, 20] # list with different types of elements
v) L5 = [1, 2, [6, 7, 8], 3] # A list containing another list known as nested list
Example

```
Print(L1)                  # let‟s get the values of list before change
[1, 2, 3, 4]
L1 [2] = 5
Print(L1)                   # modified list
[1, 2, 5, 4]
```

Here, 3rd element of the list (accessed using index value **2**) is given a new value, so instead of **3** it will be **5.**

List can be created in many ways:
i) By enclosing elements in [ ], as we have done in above examples.
ii) Using other Lists
Example

```
L5=L1 [:]
```
Here L5 is created as a copy of L1.
```
>>>print( L5)
L6 = L1 [0:2]
>>>print (L6)
```
will create L6 having first two elements of L1.

- List comprehension

Example
```
>>>n = 5
>>>l= range(n)
>>>print (l)
[0, 1, 2, 3, 4]
```
Example
```
>>> S= [x**2 for x in range (10)]
>>> print(S)
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Example
```
>>> A = [1, 2, 3]
>>> B = A
>>> print(A, B)
[1, 2, 3] [1, 2, 3]
```

List Slices
Slice operator works on list also. We know that a slice of a list is its sub-list. For creating a list slice, we use
[n:m] operator.
```
L5 = [1, 2, [6, 7, 8], 3]
>>>print(L5 [0])
1
>>>print (L5 [2])
[6, 7, 8]


>>>print (L5 [2] [0])
6
>>> L1 = [1, 2, 3, 4]
>>> L1 [1:2]
```

will give
[2]

Slices are treated as boundaries, and the result will contain all the elements between boundaries.   Its Syntax is:
seq = L [start: stop: step]
Where start, stop & step- all three are optional. If you omit first index, slice starts from „0" and omitting of stop will take it to end. Default value of step is 1.
Example
For list L2 containing ["Delhi", "Chennai", "Mumbai"]

```
>>>L2 [0:2]
["Delhi", "Chennai"]
```

Example

```
>>>list = [10, 20, 30, 40, 50, 60]
>>> list [::2]                  # produce a list with every alternate element
[10, 30, 50]
>>>list [4:]                    # will produce a list containing all the elements from 5th position
                                    till end
[50, 60]
```

Example

```
>>>list [:3]
[10, 20, 30]
>>>list [:]
[10, 20, 30, 40, 50, 60]
```

Example

```
>>> list [-1]                              # „-1" refers to last elements of list
60
```

Lists also support operations like **concatenation:**

```
>>>squares[:]
[1, 4, 9, 16, 25]
>>> squares + [36, 49, 64, 81, 100]
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

Appending in the list
Appending a list is adding more element(s) at the end of the list. To add new elements at the end of the list, Python provides a method append ( ).   Its Syntax is:
List. append (item)

```
>>>L1=[1, 2, 5, 4]
>>>L1. append (70)
>>> print(L1)
 [1, 2, 5, 4, 70]
```

For adding more than one element, extend ( ) method can be used, this can also be used to add elements of another list to the existing one.
Example

```
>>>A = [100, 90, 80, 50]
>>> L1. extend (A)
>>> print L1
[1, 2, 5, 4, 70, 100, 90, 80, 50]
>>>print A
[100, 90, 80, 50]
```

Deleting Elements

It is possible to delete/remove element(s) from the list. There are many ways of doing so:

(i) If index is known, we can use pop ( ) or del
(ii) If the element is known, not the index, remove ( ) can be used.
(iii) To remove more than one element, del ( ) with list slice can be used.
(iv) Using assignment operator

POP( )

It removes the element from the specified index, and also return the element which was removed. Its syntax is:

List.pop ([index])

Example

```
>>> L1 = [1, 2, 5, 4, 70, 10, 90, 80, 50]
>>> a= L1.pop (1)                    # here the element deleted will be returned to 'a'
>>> print(L1)
[1, 5, 4, 70, 10, 90, 80, 50]
>>> print(a)
2
```

If no index value is provided in pop ( ), then last element is deleted.

```
>>>L1.pop ( )
50
```

del removes the specified element from the list, but does not return the deleted value.

```
>>> del L1 [4]
>>> print(L1)
[1, 5, 4, 70, 90, 80]
```

remove ( )

In case, we know the element to be deleted not the index, of the element, then remove () can be used.

```
>>> L1. remove (90)
```

will remove the value 90 from the list
206

```
>>> print(L1)
[1, 5, 4, 70, 80]
```

del () with slicing

Consider the following example:

Examples

```
>>> del L1 [2:4]
>>>print(L1)
[1, 5, 80]
```

will remove 2nd and 3rd element from the list. As we know that slice selects all the elements up to 2nd index but not the 2nd index element. So **4th element** will remain in the list.