**Title: List**

**COLLECTION TYPES:**

There are four collection data types in the python programming language.

1. List : it is ordered and changeable allows duplicate members.
2. Tuple : it is ordered and un changeable. Allows duplicate members.
3. Set : it is unordered and un indexed. No duplicate members.
4. Dictionary : unordered, changeable and indexed. No duplicate members.

We can perform several operations on all sequences. they include indexing, slicing, adding, multiplying and checking for membership.

Python has built-in functions for finding the length of a sequence, largest and smallest elements.

1. **List:**

* List can be written as list of comma-separated values(items) between square brackets.
* Items in the list need not be of the same type.
* List indices start at ‘0’ in forward direction and with ‘-1’ in backward direction.
* Eg: list1=[1,2,3]

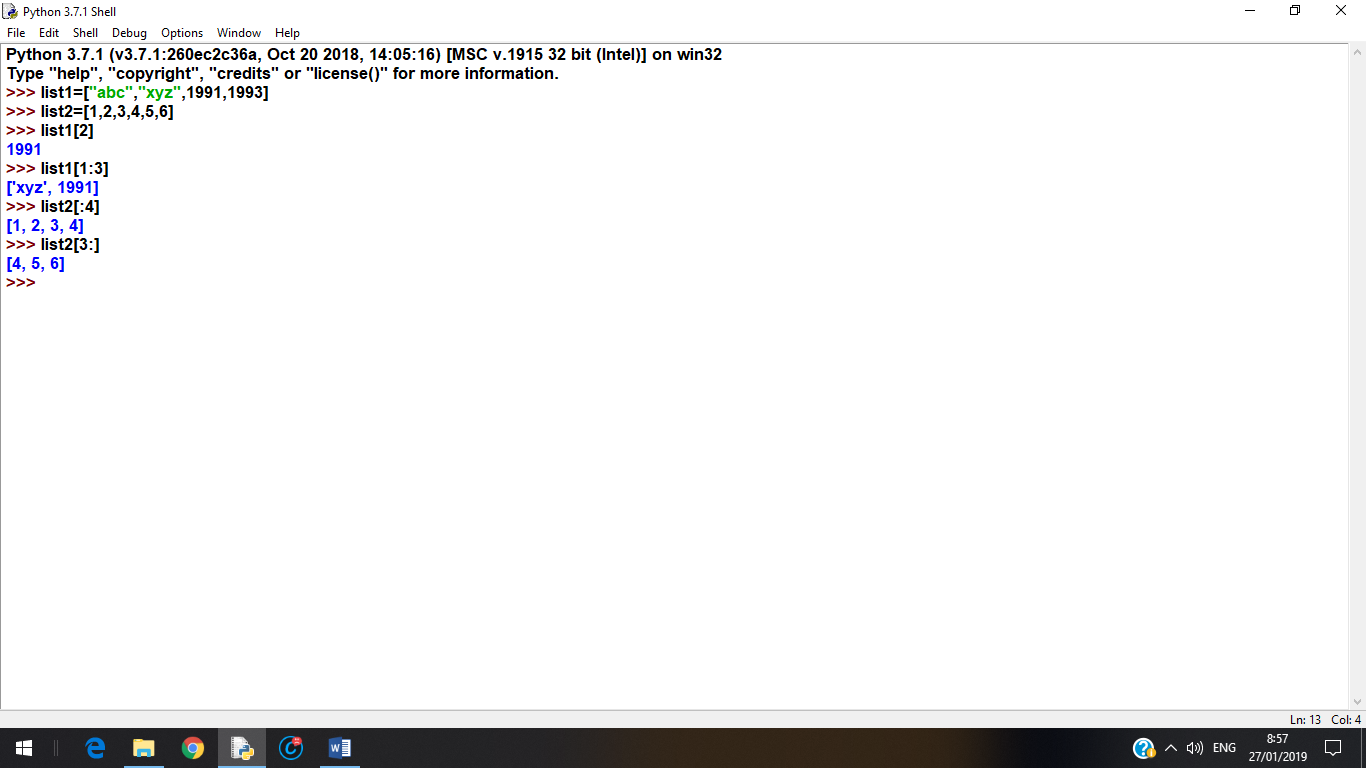
List2=[1,’abc’,12.3]

List3=[‘a’,’b’,’c’]

**Accessing values in lists:**

To access values in lists, use the square brackets along with the index to obtain value available at the index.

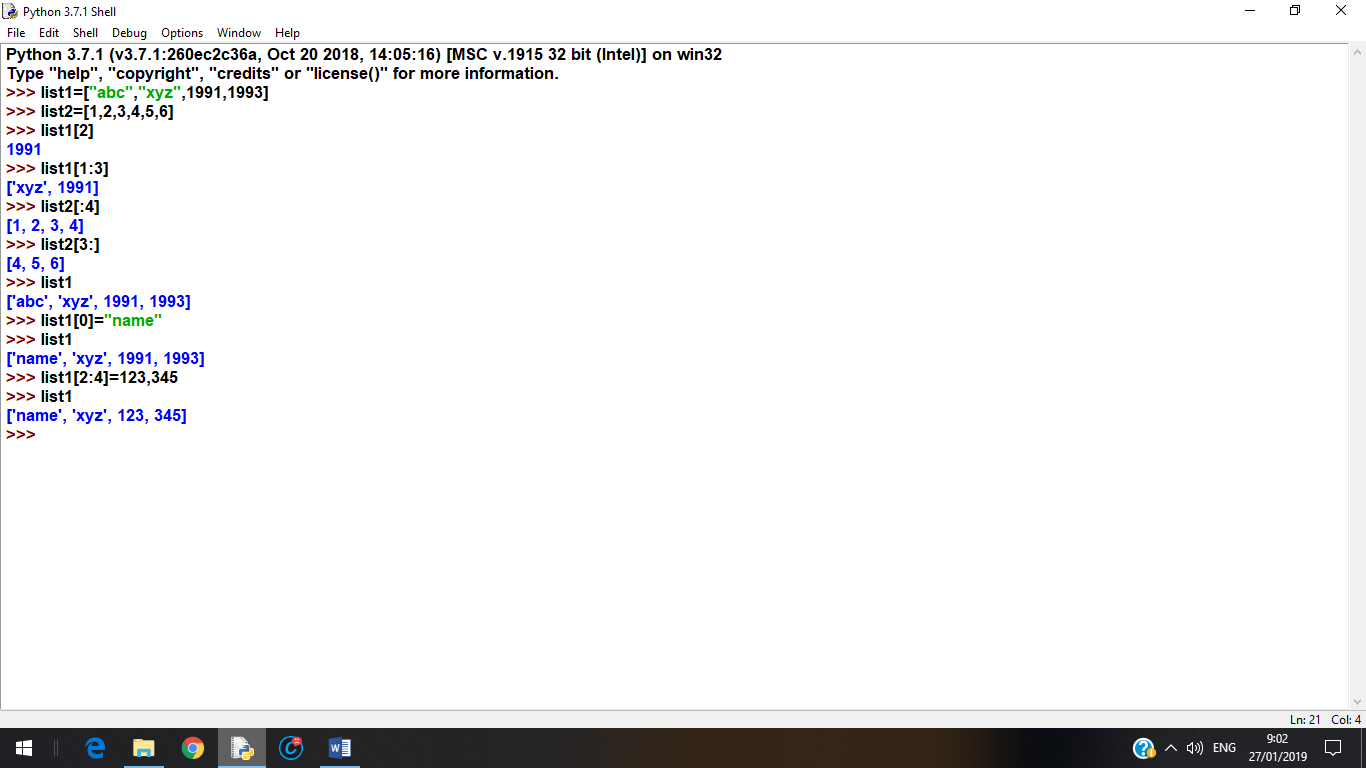
**Example:**



**Updating lists:**

We can update a single list or multiple elements of lists by giving the slice on the left-hand side of the assignment operator.

**Example:**



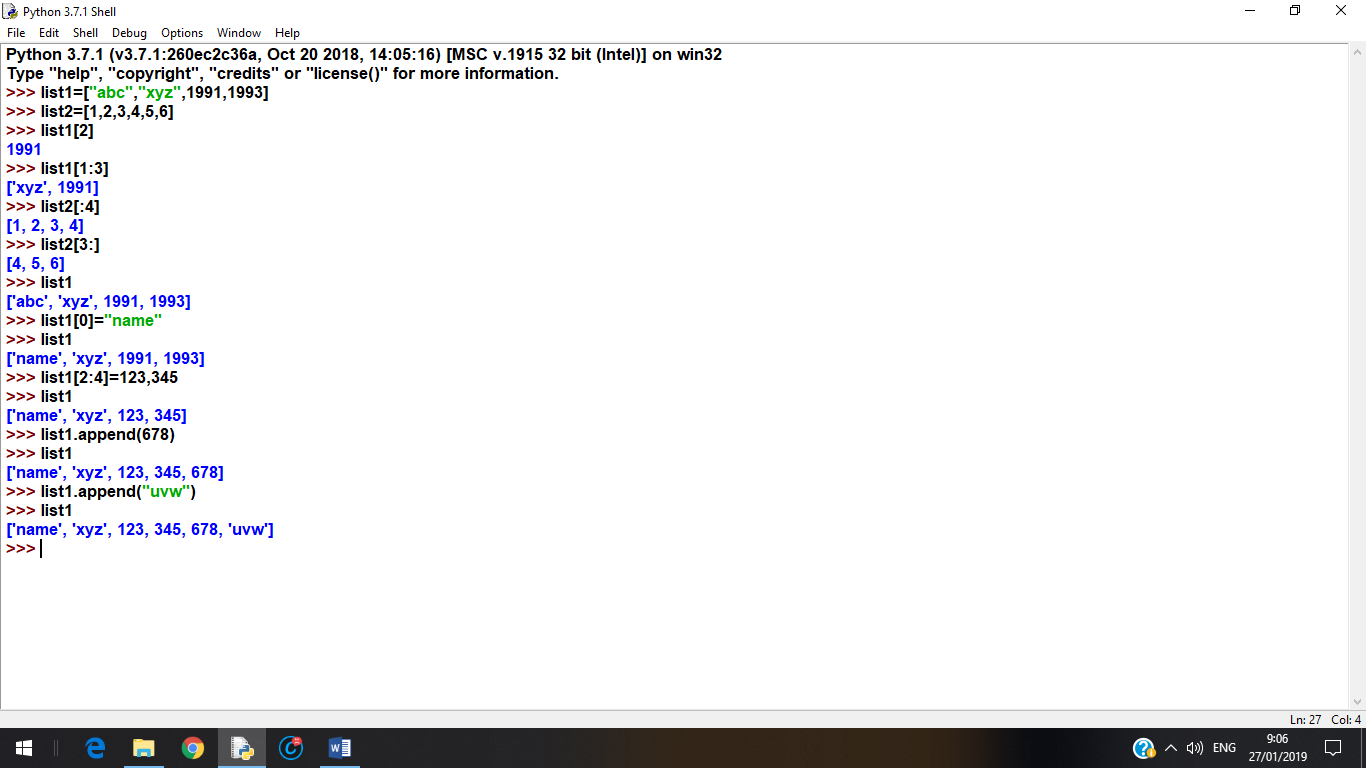
**Using append() method:**

You can add elements into a list with append() method.

Syntax: list.append(obj)

This method appends passed element or object into the existing list. It does not return any value but updates existing list.

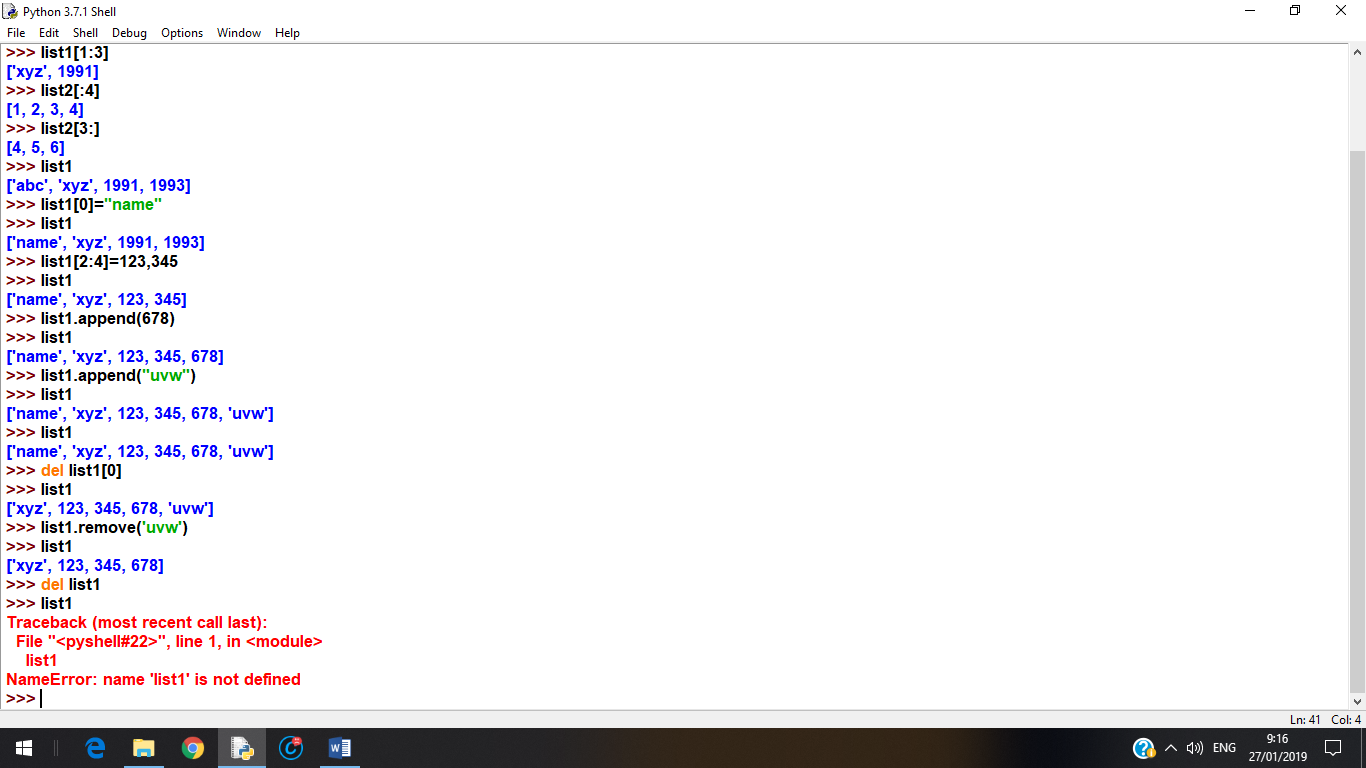
**Example:**



**Delete list elements:**

To remove a list elements, you can use either the ‘del’ statement if you know the indexes of the element(s) exactly which you are deleting or if you know the elements which you are deleting then go for remove() method.

**Example:**



When you delete a complete list using ‘del’ statement then that list will be deleted permanently. When you are trying to access the same list you will get an error indicating object not defined.

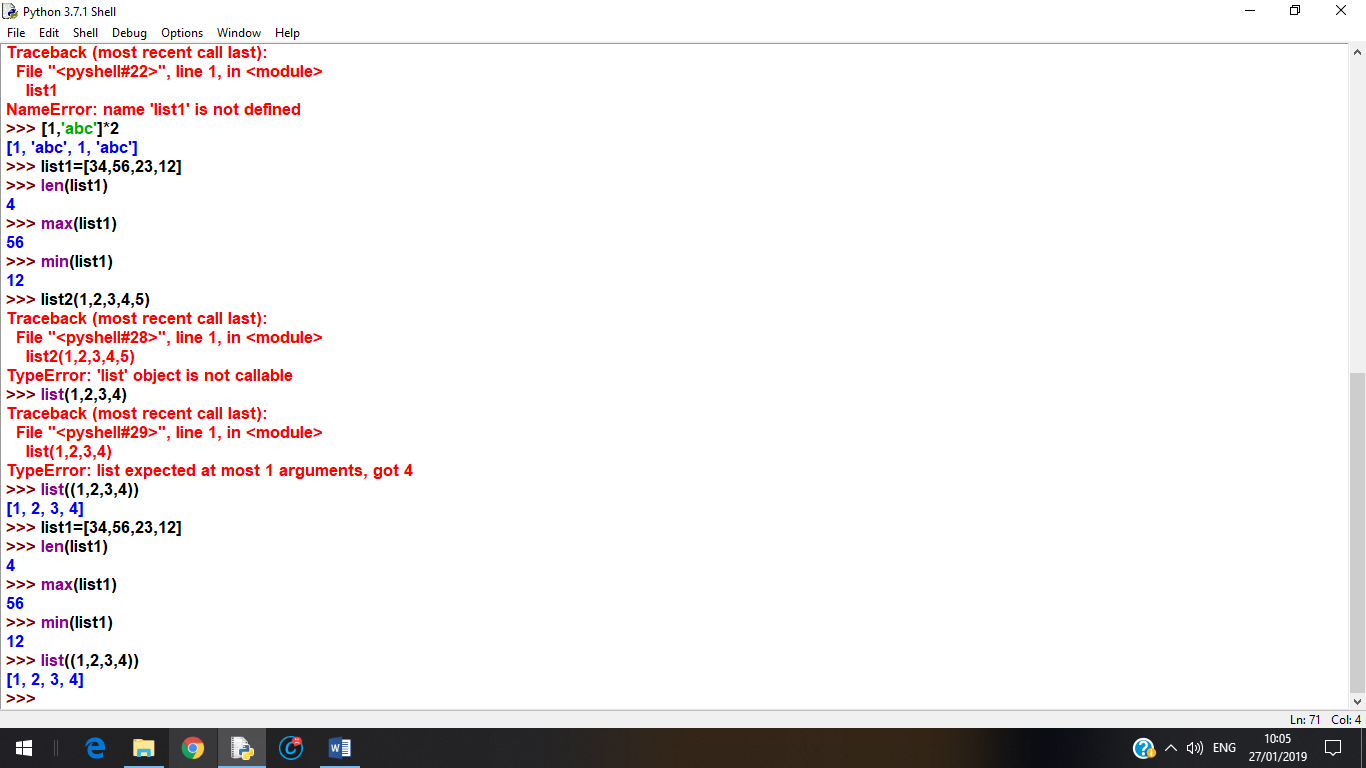
**Basic list operations:**

|  |  |  |
| --- | --- | --- |
| **Python expression** | **Result** | **Description** |
| len([1,2,3]) | 3 | Length |
| [1,2,3]+[4,5,6] | [1,2,3,4,5,6] | Concatenation |
| [1,’abc’]\*2 | [1,’abc’,1,’abc’] | Repetition |
| 3 in [1,2,3] | True | Membership operator |
| for x in [1,2,3’]:  print(x) | 1  2  3 | Iteration |

**Built-in list functions:**

1. len(list) : Gives the total length of the list. This method returns the no. of elements in the list.
2. max(list) : returns item from the list with max value.
3. min(list) : returns item from the list with min value.
4. list(seq) : converts a sequence type to list. This method returns the list.

**Example:**



**Built-in List methods:**

1. append(self, object, /) : Append object to the end of the list.
2. clear(self, /) : Remove all items from list.
3. copy(self, /) : Return a shallow copy of the list.
4. count(self, value, /) : Return number of occurrences of value.
5. extend(self, iterable, /) : Extend list by appending elements from the iterable.
6. index(self, value, start=0, stop=2147483647, /) : Return first index of value.

Raises ValueError if the value is not present.

1. insert(self, index, object, /) : Insert object before index.
2. pop(self, index=-1, /) : Remove and return item at index (default last).

Raises IndexError if list is empty or index is out of range.

1. remove(self, value, /) : Remove first occurrence of value.

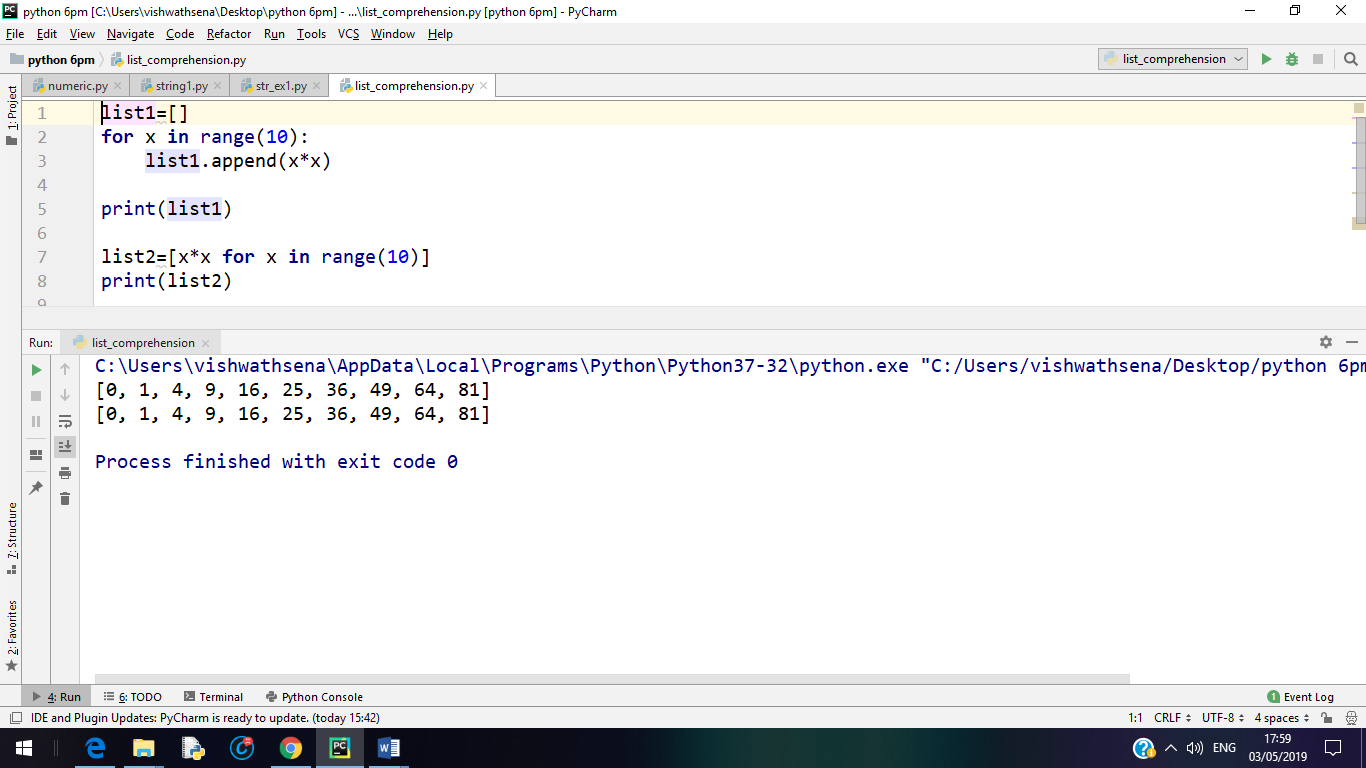
Raises ValueError if the value is not present.

1. reverse(self, /) : Reverse \*IN PLACE\*.
2. sort(self, /, \*, key=None, reverse=False) : Stable sort \*IN PLACE\*.

**List Comprehensions:**

List comprehensions provide a concise way to create lists. Common applications are to make new lists where each element is the result of some operations applied to each member of another sequence or iterable, or to create a subsequence of those elements that satisfy a certain condition.

For example, assume we want to create a list of squares, like:



Note that this creates (or overwrites) a variable named x that still exists after the loop completes. We can calculate the list of squares without any side effects using:

list2=[x\*x for x in range(10)]

(or)

list2= list(map(lambda x: x\*\*2, range(10)))