**SESSION -**

**LISTS - PROPERTIES**

* They are positionally ordered collections of arbitrarily typed objects,
* They have no fixed size.
* They are also mutable—unlike strings, lists can be modified in place by assignment to

offsets as well as a variety of list method calls.

Because they are sequences, lists support all the sequence operations like strings; the only difference is that the results are usually lists instead of strings.

**EXAMPLES**

given a three-item list:

**LEN**

>>> L = [123, 'spam', 1.23] # A list of three different-type objects

>>> len(L) # Number of items in the list

3

we can index, slice, and so on, just as for strings:

**INDEXING**

>>> L[0] # Indexing by position

123

>>> L[:-1] # Slicing a list returns a new list

[123, 'spam']

**CONCATENATION**

>>> L + [4, 5, 6] # Concat/repeat make new lists too

[123,'spam', 1.23, 4, 5, 6]

**REPETITION**

>>> L \* 2

[123, 'spam', 1.23, 123, 'spam', 1.23]

>>> L

[123, 'spam', 1.23] # We're not changing the original list



The list we just looked at,

* Contains three objects of completely different types (an integer, a string, and a floating-point number).
* Lists have no fixed size, which means they can grow and shrink on demand, in response to list-specific operations.

**TYPE SPECIFIC OPERATIONS**

**APPEND**

[root@centos72-bgururaj ~]# python

Python 2.7.5 (default, Apr 11 2018, 07:36:10)

[GCC 4.8.5 20150623 (Red Hat 4.8.5-28)] on linux2

Type "help", "copyright", "credits" or "license" for more information.

>>> L = [123, 'spam', 1.23]

>>> L

[123, 'spam', 1.23]

>>> L.append('NI')

>>> L

[123, 'spam', 1.23, 'NI']

>>> L.append(1345)

>>> L

[123, 'spam', 1.23, 'NI', 1345]

**POP**

>>> L.pop(1)

'spam'

>>> L

[123, 1.23, 'NI', 1345]

**COUNT**

>>> L.append('spam')

>>> L

[123, 1.23, 'NI', 1345, 'spam']

>>> L.count('spam')

1

>>> L.append('spam') # append again same string element.

>>> L

['spam', 'NI', 1345, 123, 1.23, 'spam']

>>> L.count('spam')

2

**SORT**

>>> L.sort()

>>> L

[1.23, 123, 1345, 'NI', 'spam']

**REVERSE**

>>> L.reverse()

>>> L

['spam', 'NI', 1345, 123, 1.23]

**BOUNDS CHECKING**

Indexing off the end of a list is always a mistake, but so is assigning off the end.

>>> L

['spam', 'NI', 1345, 123, 1.23]

>>> L[99]

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

IndexError: list index out of range

>>> L[99] = 1

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

IndexError: list assignment index out of range

**NESTING**

* Nice feature of Python’s core data types is that they support arbitrary nesting—we can nest them in any combination, and as deeply as we like.
* we can have a list that contains a dictionary, which contains another list.

Example: Matrixes:

>>> M = [[1,2,3],

... [4,5,6],

... [7,8,9]]

>>> M

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

>>> M[1]

[4, 5, 6]

>>> M[1][2]

6