ED5340 - Data Science: Theory and Practise

L5 - List container

Ramanathan Muthuganapathy (https://ed.iitm.ac.in/~raman)
Course web page: https://ed.iitm.ac.in/~raman/datascience.html
Moodle page: Available at https://courses.iitm.ac.in/

Container data type

- They can hold multiple data types.
- Lists, Tuples, Sets, Dictionaries
- Also called as collections / compound data types

List representation

- List is a collection of dissimilar data types (mostly used for similar datatype)
  - lst1 = [10, 20, 300, 400, 50]; lst2 = [‘rat’, ‘cat’, ‘bat’, ‘lion’, ‘tiger’, ‘crocodile’]
  - lst3 = [10.5, 22, ‘Antelope’, ‘rabbit’, 456789, 1, 1, 2, 2, 4, 89.9]
  - lst4 = [100] * 5
  - lst5 - [ ] #Empty list, basically empty ‘square’ brackets.

List basics and accessing

- Elements in the list can be repeated.
- It is an ordered collection - can be indexed and sliced
- Entire list or each element can be printed.
- List is an `iterable' i.e. you can iterate over its elements.

Demo using L5_list_ex_access.py

CW: Define a list of strings and find the number of characters in each of them.

List operations

- Lists are mutable (unlike strings).
- Lists can be concatenated
- searching (containment) and sorting
- deletion - using index or range of indices
- conversion / other functions - str to list, len, max, min, sum
- shallow copy, deep copy, difference
- Lists comparison

Demo using L5_list_operations.py

CW: Define a list of marks for a student and find the max, min and average of them. Arranged the in sorted order. Print all of them clearly. Delete the entire list and check for emptiness.

CW: 1) Define two lists of strings. Use relational operators to compare them. What can you say?
2) Instead of both lists as strings, change one of them to integers and then use relational operators. What can you say?

List methods - Member functions in the list

- Given a list, you can apply the following member functions using the object.
  - append - at the end
  - remove - the element
  - pop - removes last item (also removes a particular item, if given)
  - insert - after a certain given position
  - reverse
  - sort - both ascending and descending are available
  - count
  - index - of a particular item

List methods - Member functions in the list

Example

```
lst = [10, 20, 30, 30, 50, 60]

    lst.append(25)

    lst.remove(30)

    lst.pop( )
```

Demo using L5_list_functions.py

CW: Start from empty list. Add few elements. Sort the list in the reverse order. Delete the elements one by one till the list is empty.

HW: Count the number of occurrences of each element in the list.

Demo using L5_list_copy_comp.py

List Varieties

- List of lists
- List embedding
- List unpacking (using * operator)

Demo using L5_list_varieties.py

List Comprehension

shorter form for list creation

The syntax goes like this:

lst = [expression for var in sequence [optional for and/or if]]

The above is a replacement for the following

lst = [ ] #empty list

for var in sequence:

    lst.append(expression)

Demo using L5_list_comprehension.py

Demo using L5_list_of_lists_comprehension.py

Some nuances on list and list of lists

- Lists are like 1D array (though not contiguous in mem. allocation)
- List of lists are like 2D array (matrix)

Some nuances on list and list of lists

Lists are like 1D array (row vector)

- lst = [10, 20, 30, 40]
- for num in lst:
- num iterates over each element in lst (which is an iterable) using ‘for’

Some nuances on list and list of lists

Lists are like 2D array (matrix)

```
arr = [ [1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]

for ele in arr:

    for num in ele:

        print num
```

Some nuances on list and list of lists

Lists are like 2D array (matrix)

arr = [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]

for ele in arr:

    print(ele)

ele iterates over each element in arr (which is an iterable) using ‘for’

    What is each element of arr? - i.e. Each row - arr[0], arr[1] etc

arr[0] = [1, 2, 3, 4]

arr[1] = [5, 6, 7, 8]

So, what will the print statement give?

Some nuances on list and list of lists

Lists are like 2D array (matrix)

Remember, arr[0] = [1, 2, 3, 4], arr[1] = [5, 6, 7, 8] and so on

for num in ele:

    print(num)

num will iterate over each ele - i.e. iterate over arr[0], arr[1] etc.

iterate over arr[0] —> each element in arr[0]

So, What will be the print(num) give?