Lab 5: Python – Lists

Sequence Data Structure

Python has a data structure called sequence. Each element in a sequence can be accessed by the index position. The first element has index 0, second element has index 1 and so on.

Python has many built-in types of sequences: **Lists**, **Tuple**, **Sets**, and **Dictionary**.

Chapter I: Basics of Lists

1. Creating a List

Lists are versatile. To create a list, write a list of comma-separated values enclosed by square brackets [].

```
>>> alist = [11, 17, 13, 2, 5]
>>> print(alist)
>>> blist = ['Mumbai', 'Chennai', 'Kharagpur']
>>> clist = ["MScPhd", "IEOR", 2013, 12]
>>> dlist = ["MScPhd", "IEOR", "2013", 12]
>>> emptylist = []
```

Note: In *clist*, the first two elements are strings & last two elements are numbers. In *dlist*, the first three elements are strings, and only the last element is a number.

2. Accessing elements in a list

Similar to a string, we can access the elements of a list using their index number, starting at 0.

`>>> alist[0]` → get the first element of the list

`>>> alist[-1]` → get the first element of the list from the right side

`>>> alist[2:5]` → get the 3rd, 4th and 5th elements of the list

Try the following commands separately, look at the results and understand how lists access the elements

alist[3:], *alist*[:3], *alist*[3:2], *alist*[2:3], *alist*[3:3], *alist*[:]

3. Updating and Deleting elements in a list

`>>> alist[1]` → Displays the current value of 2nd element

`>>> alist[1] = 19` → Updates the 2nd element to 19

`>>> print(alist)` → prints the updated list!

```
>>> alist
```
→ displays the updated list!

4. Basic list operations

```
>>> len(alist)
```
→ **len**() displays the number of elements in the list

```
>>> min(alist)
```
→ **min**() displays the min value in the list

```
>>> max(alist)
```
→ **max**() displays the max value in the list

Try the following commands separately, and understand how lists work:
max(clist), min(clist), max(dlist), min(dlist)

```
>>> alist + blist
```
→ The '+' joins blist at the end of alist

```
>>> 2013 in clist
```
→ returns true since 2013 is in *clist*

```
>>> 2013 in dlist
```
→ returns false since 2013 is not in *dlist*

```
>>> sum(alist)
```
→ returns sum of all elements in *alist*.

5. Traversing a list

An important and often used feature of Python is the ability to iterate over a list. Try the following code:

```
>>> for x in alist:
        print(x)
```

The above code will iterate over *alist*, where in each iteration *x* takes on corresponding value from *alist*. Useful when we are only reading a list.
Suppose we want to update the values of a list, then we need to iterate using *index* of the list.

```
>>> for i in range(len(alist)):
        alist[i] = alist[i]*2
        print 'i=',i,'value=',alist[i]
>>> print(alist)
```

6. Deleting elements in a list

```
>>> clist
```
→ displays the list.

If we know the index location of the element we want to delete, then we can do the following:

```
>>> del clist[1]
```
→ deletes the 2nd element from the list

```
>>> clist
```
→ displays the updated list.

If know the element (but not the index), we can use **remove** function:

```
>>> clist.remove(2013)
```
→ deletes the element *2013* from the list

```
>>> clist
```
→ displays the updated list.

To remove and return the element from a defined index position, we can use **pop:**

```
>>> clist.pop(0)
```
→ deletes & returns the 1st element from list

```
>>> clist
```
→ displays the updated list.

7. Add elements in a list

```
>>> blist
```
→ displays the list.

To add a new element at the end of the list, we can use **append** function or +:

```
>>> blist.append('Delhi')
>>> blist
>>> blist = blist + ['Indore']
>>> blist
```

To insert a new element at a specified index position, we can use **insert** function:

```
>>> blist.insert(2,'Kanpur')
```
→ inserts 'kanpur' at index position 2 in the list.

To combine one list with another, we can use **extend** function:

```
>>> t1 = ['b', 'e']
>>> t2 = ['d', 'a', 'c']
>>> t1.extend(t2)
```
→ *t2* is appended to *t1*. *t2* remains unchanged.

```
>>> t1
>>> t2
```

8. Other List functions

Sorting function is inbuilt in Python:

```
>>> t1.sort()
```
→ *t1* is sorted from low to high, permanently

```
>>> print(t1)
```

We can count the number of occurrences of an element using **count:**

```
>>> t = [1,2,3,2,3,4,5,2,3]
>>> t.count(2)
```
→ Counts the number of times '2' occurs in list t.

Reverse function is inbuilt in Python:

```
>>> t.reverse()
```
→ t is reversed in place, permanently.

```
>>> print(t)
```

9. Copying Lists (Advanced topic)

Consider the following:

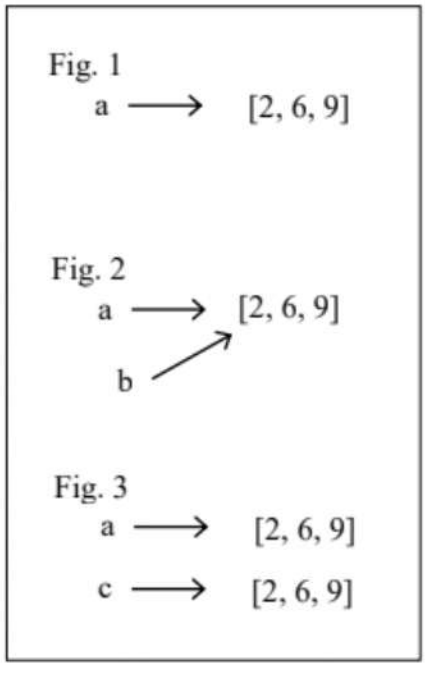
```
>>> a = [2, 6, 9]
```

Variable name a refers to the list object [2, 6, 9] (Fig. 1)

Now, type

```
>>> b = a
>>> print b
>>> b[1] = 38
>>> print b
>>> print a
```

You will see that b refers to the list object [2, 38, 9]. But



interestingly, a also refers to [2, 38, 9] and not [2, 6, 9]!
This is because, when we did b=a, what had happened
is that a and b both refer to the same object (Fig. 2). So if we modify b, a also changes.
To create a separate copy of the list, we should do:

```
>>> a = [2, 6, 9]
>>> c = list(a)
>>> print c
>>> c[1]=38
>>> print c
>>> print a
```

You will see that c refers to the list object [2, 38, 9]. And a refers to [2, 6, 9] only.
This is because, c=list(a) created another copy of the list (Fig. 3). Alternatively, one can also use c = a.copy() which stores another copy of the list a in c.

10. Program using Lists – Statistics of a dataset

In a notebook cell, type the following program to compute the mean, variance, and standard deviation of a given dataset:

```
data=[550, 420, 120, 390, 250, 175, 280, 90, 300, 490]
n = len(data)
print('Total data points=', n)
print('Mean=', sum(data)/n)
ss =0
for x in data:
        ss= ss + x**2
v = ss/(n-1)
print('Var=', v)
print('s.d=', v**0.5)
```

Run to see if the output is shown correctly.

11. Program using Lists – Adding lists

The following program to take 2 lists of the same length as inputs, and adds the elements of the lists together:

```
a =[5, 4, 1, 3]
b =[2, 7, 2, 9]
c =[]
for i in range(len(a)):
    c.append(a[i]+b[i])
print(c)
```

For each of the following, write the code that will give the Output mentioned.

1.
```
>>> alist = [3, 5, 6, 12]
>>> Your Code
6
```

2.
```
>>> alist = [3, 5, 6, 12]
>>> Your Code
[5, 6, 12]
```

3.
```
>>> alist = [3, 5, 6, 12]
>>> Your Code
3
5
6
12
```

4.
```
>>> alist = [3, 5, 6, 12]
>>> Your Code
[12, 6, 5, 3]
```

5.
```
>>> alist = [3, 5, 6, 12]
>>> Your Code
[9, 15, 18, 36]
```

6.
```
>>> alist = [3, 5, 6, 12]
>>> Your Code
[False, False, True, True]
```

12. Tuples

Tuple is a data type similar to list. It varies with lists in the fact that they are immutable. We can not add elements to a tuple because of their immutable property. There's no append() or extend() method for tuples, You can't remove elements from a tuple, also because of their immutability

One can create tuples in the following ways:

```
>>> atuple = ('a', 'b', 'c')
>>> btuple = (10.1, 2, 3.5)
>>> ctuple = ('apples', 2, [1,2,3])
```

Observe that in `ctuple` we have three different data types.

To access each element of the tuple similar list like commands can be used. Consider the following:

```
>>> ctuple = ('apples', 2, [1,2,3])

>>> type(ctuple[2])
>>> ctuple[2].append(4)
>>> print(ctuple[2])
>>> print(ctuple)
```

Notice that `ctuple` is of the data type tuple but `ctuple[2]` is of the data type list. One can perform list operations on `ctuple[2]`. Similarly one can even add a tuple inside a list as given below in the following example:

```
>>> cars = ["Japan",("Mazda", "Nissan", "Mitsubishi")]
>>> print(cars[1])
>>> print(cars[1][1])
```

Try yourself:

1. What is the data type of cars[1]?
2. What is the data type of cars[1][1]?
3. What is the output for len(cars)?
4. What is the output for len(cars[1])?
5. Run the following code:

```
>>> a = (1,2)
>>> b = a
>>> b[1] = 5
```

Observe that `b[1] = 5` will give an error. Why?

Tasks

In a new notebook, do the following exercises. Use comments/ print messages to neatly label the code. For each question write and solve using a single cell, if possible. After you complete the questions, submit your file in Moodle.

1. Given a list of n numbers, write a program that counts the number of negative numbers and the number of positive numbers. Test the program with this list:[3, -4, -6, 9, 0, 7, 2, 10, -2, -1].

2. Given a list of n numbers, write a program that outputs another list that contains the cumulative sum. For e.g. given the input [3, 4, 6, 9, 0, 7, 2, 10, 2, 1] the output must be [3, 7, 13, 22, 22, 29, 31, 41, 43, 44].

3. Update Q1 program to output two lists: one list of negative, and another list of non-negative numbers.

4. Write a program that takes as input a list and returns a list that contains each element of the original list twice. For e.g. for input [2, 3, 1] the output must be [2, 2, 3, 3, 1, 1]

5. Write a program that prompts the user to enter 5 numbers, and stores the numbers as a list. The program then prints the list of numbers, the maximum, the minimum and the sum of the numbers.