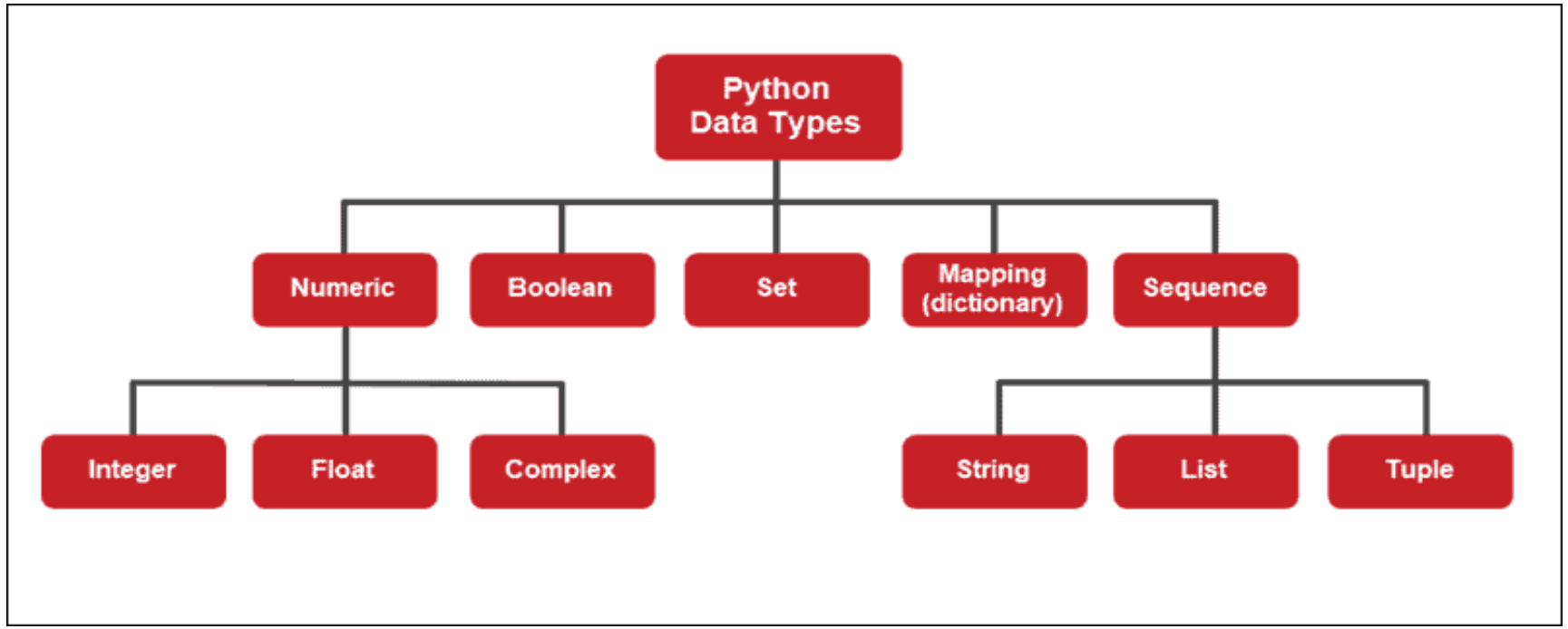
LISTS in Python

Introduction

- Previously, we have talked about sequence data types, which is an orderly collection of items and each item is indexed by an integer



Lists

- The data type list is an ordered sequence which is mutable and made up of one or more elements. Elements of a list are enclosed in square brackets and are separated by comma.

- Unlike a string which consists of only characters, a list can have elements of different data types, such as integer, float, string, tuple or even another list.

  for e.g.)

  list1 = [2, 3,"Huia",5.8, 7]

  list2 = [1,2,[1,12,13],5,6]

Lists

- A list is very useful to group together elements of mixed data types.

  for e.g.)

  list1 = [100, 23.5, 'Hello']

  list2 = [‘a’, 2+4j, 7.3, 5, “hi”]

Accessing elements in a List

- The elements of a list are accessed in the same way as characters are accessed in a string.

- Like string indices, list indices also start from 0 and the last character is n-1 where n is the length of the list.

  for e.g.) list1= [0, 2, 5, 10, 24]

Table: Indexing of elements in list

Positive Indices	0	1	2	3	4
List	0	2	5	10	24
Negative indices	-5	-4	-3	-2	-1

Accessing elements in a List

- The index specifies the character to be accessed in the list and is written in square brackets ([ ]).

e.g.)

```
In [1]: list1 = ["Ram", 45, 2.3, 9+2j]

In [2]: list1[2]
Out[2]: 2.3
```

- The index can also be an expression including variables and operators but the expression must evaluate to an integer.

For e.g.)

```
In [3]: list1[(1+2)-3]
Out[3]: 'Ram'
```

Accessing elements in a List

- If we give index value out of this range then we get an *IndexError*.

  For e.g.)

```
In [4]: list1[6]
Traceback (most recent call last):

  File "<ipython-input-4-fa2dc7f71df5>", line 1, in <module>
    list1[6]

IndexError: list index out of range
```

List is mutable

- In Python, list is a mutable data type i.e., contents of list can be changed even after it has created

  For e.g.)

```
In [5]: list1 = ["Ram", 34, 4.5, 2+3j, "#"]

In [6]: list1[2] = "Himanshu"

In [7]: list1
Out[7]: ['Ram', 34, 'Himanshu', (2+3j), '#']
```

Operations on list: Concatenation

- To concatenate means to join. Python allows us to join two or more lists using concatenation operator depicted by the symbol +.

For e.g.)

```
In [8]: list1 = ['Ram', 34, 'Himanshu', (2+3j), '#']

In [9]: list2 = [True, False, 1.2, 4]

In [10]: list1 + list2
Out[10]: ['Ram', 34, 'Himanshu', (2+3j), '#', True, False, 1.2, 4]
```

Note:

After the concatenation operation, there will be no change in the values of list1 and list2

Operations on list: Concatenation

- The concatenation operator '+' requires that the operands should be of list type only. If we try to concatenate a list with elements of some other data type, TypeError occurs.

  For e.g.)

```
In [11]: list1 = ["Ram", 3.4, 2+6j, "@"]

In [12]: list1 + "Himanshu"
Traceback (most recent call last):

  File "<ipython-input-12-0de8a5ba0070>", line 1, in <module>
    list1 + "Himanshu"

TypeError: can only concatenate list (not "str") to list
```

Operations on list: Repetition

- Python allows us to repeat the given list using repetition operator which is denoted by symbol *.

For e.g.)

```
In [15]: list1 = [True, False, "@"]

In [16]: list1*3
Out[16]: [True, False, '@', True, False, '@', True, False, '@']
```

Note:

After the repetition operation, there will be no change in the values of list1

Operations on list: Membership

- As we have already studied in Strings, Python has two membership operators:
  - 'in' and
  - 'not in'
- The 'in' operator checks if the element is present in the list, and returns True, else return False if element is not present in the list

For e.g.)

```
In [17]: list1 = [True, False, "@"]

In [18]: True in list1
Out[18]: True

In [19]: "#" in list1
Out[19]: False
```

Operations on list: Membership

- On the other hand, the 'not in' operator returns True if the element is not present in the list and returns False if element is present in the list

For e.g.)

```
In [20]: list1 = [True, False, "@"]

In [21]: "#" not in list1
Out[21]: True

In [22]: False not in list1
Out[22]: False
```

Operations on list: Slicing

- Like string, slicing operation can also be applied to lists.
- Given a list l1, the slice operation l1[n:m] returns the part of the list l1 starting from index n (inclusive) and ending at m (exclusive).

  For e.g.)

```
In [24]: list2 = [True, False, "@", [1,2,3], "mor", 10.2]

In [25]: list2[2:5]
Out[25]: ['@', [1, 2, 3], 'mor']
```

- In other words, we can say that l1[n:m] returns all the elements from l1[n] till l1[m-1]

Operations on list: Slicing

Note:

The numbers of elements in the resulting list after slicing operation will always be equal to difference of two indices m and n, i.e., (m-n).

- If the first index is not mentioned, the slice starts from index 0.

For e.g.)

```
In [26]: list2 = [True, False, "@", [1,2,3], "mor", 10.2]

In [27]: list2[:4]
Out[27]: [True, False, '@', [1, 2, 3]]
```

- If the second index is not mentioned, the slicing is done till the length of the list.

For e.g.)

```
In [28]: list2 = [True, False, "@", [1,2,3], "mor", 10.2]

In [29]: list2[2:]
Out[29]: ['@', [1, 2, 3], 'mor', 10.2]
```

Operations on list: Slicing

- The slice operation can also take a third index that specifies the 'step size'. For example, l1[n:m:k], means every kth element has to be extracted from the list l1 starting from n and ending at m-1.

For e.g.)

```
In [30]: list2 = [True, False, "@", [1,2,3], "mor", 10.2]

In [31]: list2[2:7:2]
Out[31]: ['@', 'mor']
```

- By default, the step size is one and negative indexes can also be used for slicing.

For e.g.)

```
In [34]: list2 = [True, False, "@", [1,2,3], "mor", 10.2]

In [35]: list2[5:1:-1]
Out[35]: [10.2, 'mor', [1, 2, 3], '@']
```

Traversing a list: using for loop

Code:

```
#list traveral using for loop
list1 = [1, "red", [1,2,3], 5.6, "Hi"]
for i in list1:
    print(i)
```

Output:

```
1
red
[1, 2, 3]
5.6
Hi


...Program finished with exit code 0
Press ENTER to exit console.
```

Traversing a list: using for loop

- Another way of accessing the elements of the list is using range() and len() functions:

Code:

```
#list traveral using for loop
list1 = [1, "red", [1,2,3], 5.6, "Hi"]
l = len(list1)
for i in range(l):
    print(list1[i])
```

Output:

```
1
red
[1, 2, 3]
5.6
Hi


...Program finished with exit code 0
Press ENTER to exit console.
```

Traversing a list: using while loop

Code:

```
#list traveral using while loop
list1 = [1, "red", [1,2,3], 5.6, "Hi"]
l = len(list1)
i = 0
while (i<l):
    print(list1[i])
    i += 1
```

Output:

```
1
red
[1, 2, 3]
5.6
Hi


...Program finished with exit code 0
Press ENTER to exit console.
```

List methods and built-in functions

- **built-in functions:**

  len(), list(), append(), extend(), insert(), count(), index(), remove(), pop(), reverse(), sort(), sorted(), min(), max(), sum();

Nested Lists

- When a list appears an element of another list, it is called a nested list

  For e.g.)

  list1 = ["Ram", 1, 2.3, [2,3,5], 2+9j]

- To access the element of the nested list of list1, we have to specify two indices list1[i][j].

  where the first index i will take us to the desired nested list and second index j will take us to the desired element in that nested list.

Copying Lists

- Given a list, the simplest way to make a copy of the list is to assign it to another list. For e.g.)

```
In [37]: list1 = [2,4,"re"]

In [38]: list2 = list1

In [39]: list2
Out[39]: [2, 4, 're']
```

Note:

The statement list2 = list1 does not create a new list. Rather, it just makes list1 and list2 refer to the same list object.

Copying Lists

- Here list2 actually becomes an alias of list1. Therefore, any changes made to either of them will be reflected in the other list.

for e.g.)

```
In [37]: list1 = [2,4,"re"]

In [38]: list2 = list1

In [39]: list2
Out[39]: [2, 4, 're']

In [40]: list1.append("Himansu")

In [41]: list1
Out[41]: [2, 4, 're', 'Himansu']

In [43]: list2
Out[43]: [2, 4, 're', 'Himansu']
```

Copying Lists

- We can also create a copy or clone of the list as a distinct object by three methods:

  1.) using slicing method

  2.) using in-built list() method

  3.) using copy() function of python library copy

Copying Lists: using slicing method

```
In [44]: list1 = ["Re", 3.4, 3, 6]

In [45]: list2 = list1[:]

In [46]: list2
Out[46]: ['Re', 3.4, 3, 6]

In [47]: list1.append(10)

In [48]: list1
Out[48]: ['Re', 3.4, 3, 6, 10]

In [49]: list2
Out[49]: ['Re', 3.4, 3, 6]
```

Copying Lists: using built-in function list()

```
In [58]: list1 = ["Re", 3.4, 3, 6]

In [59]: list2 = list(list1)

In [60]: list2
Out[60]: ['Re', 3.4, 3, 6]

In [61]: list1.append("Rahul")

In [62]: list2
Out[62]: ['Re', 3.4, 3, 6]

In [63]: list1
Out[63]: ['Re', 3.4, 3, 6, 'Rahul']
```

Practice problems on lists

- Sample program: Input a list from user

Code:

```python
"""
Sample Program: WAP to input a list from user
@author: Himanshu Mudgal
"""
n = int(input("How many elements you want to enter: "))
#creating an empty list
list1 = list()
for i in range(n):
    element = input("Enter the element: ")
    list1.append(element)
print(list1)
```

Output:

```
How many elements you want to enter: 4
Enter the element: 15
Enter the element: "Ram"
Enter the element: 2.6
Enter the element: 2+8j
['15', '"Ram"', '2.6', '2+8j']
```

Practice problems on lists

- Sample program: reverse a input list

Code:

```python
"""
Sample Program: WAP to reverse a list
@author: Himanshu Mudgal
"""
list1 = [10, 12.5, "Ram", 2+4j, 78]
print("Input list = ",list1)
#using reverse() function
list1.reverse()
print("Reverse list = ",list1)
```

Output:

```
Input list =  [10, 12.5, 'Ram', (2+4j), 78]
Reverse list =  [78, (2+4j), 'Ram', 12.5, 10]


...Program finished with exit code 0
Press ENTER to exit console.
```

Practice problems on lists

- Program-17: Find the largest/smallest number in a list

Code:

```python
"""
Program-17: Find the largest/smallest number in a list
@author: Himanshu
"""
list1 = [1.5, 2.6, 4, 6, 0.4]
print("Largest number in the list is:",max(list1))
print("Smallest number in the list is:",min(list1))
```

input

Output:

```
Largest number in the list is: 6
Smallest number in the list is: 0.4


...Program finished with exit code 0
Press ENTER to exit console.
```

Code:

```python
"""
Program-17: Find the largest/smallest number in a list
            without using built-in function
@author: Himanshu
"""
list1 = [1.5, 2.6, 4, 6, 0.4]
#defining variables to store max and min of list
max_list1 = list1[0]
min_list1 = list1[0]
for i in list1:
    if i>= max_list1:
        max_list1 = i
    if i<=min_list1:
        min_list1 = i
print("Largest number in the list is:",max_list1)
print("Smallest number in the list is:",min_list1)
```

Output:

```
Largest number in the list is: 6
Smallest number in the list is: 0.4


...Program finished with exit code 0
Press ENTER to exit console.
```

Practice problems on lists

- Program-18: search for a given element in the list and return the index of that element

Code:

```python
'''
Program-18: search for a given number in the list and
            return the index of that element
@author: Himanshu Mudgal
'''
list1 = [12, 4, 5, 3, 7]
element = int(input("Enter the number you want to search: "))
if element in list1:
    print(element,"is present in the given list and its index is",\
    list1.index(element))
else:
    print(element,"is not present in the given list")
```

Output:

```
Enter the number you want to search: 12
12 is present in the given list and its index is 0


...Program finished with exit code 0
Press ENTER to exit console.
```

Code:

```python
'''
Program-18: search for a given number in the list and
            return the index of that element
@author: Himanshu Mudgal
'''
list1 = [12, 4, 5, 3, 7, 5, 34, 5]
element = int(input("Enter the number you want to search: "))
flag = 0
l = len(list1)
for i in range(l):
    if list1[i]==element:
        print(element,"is present at index",i)
        flag = 1
if flag==0:
    print(element, "isn't present in the given list")
```

Output:

```
Enter the number you want to search: 5
5 is present at index 2
5 is present at index 5
5 is present at index 7


...Program finished with exit code 0
Press ENTER to exit console.
```

Practice problems on lists

- Program-19: Input a list of numbers and swap elements at the even location with the elements at the odd location

code:

```python
"""
Program-19: Input a list of numbers and swap elements at the even location
            with the elements at the odd location
            @AUTHOR: Rohit/Vishal/Shashank
"""
list1=list()
x=int(input("how many elements you want in list : "))
for i in range(x):
    l=input("enter element : ")
    list1.append(l)
print(list1)
if x%2==0:
    for i in range (0,x,2):
        a=list1[i]
        list1[i]=list1[i+1]
        list1[i+1]=a
else:
    for j in range (0,x-1,2):
        a=list1[j]
        list1[j]=list1[j+1]
        list1[j+1]=a
print(list1)
```

Output:

input

```
how many elements you want in list : 5
enter element : A
enter element : B
enter element : C
enter element : D
enter element : E
['A', 'B', 'C', 'D', 'E']
['B', 'A', 'D', 'C', 'E']
```

Summary

- Introduction to lists
- Accessing elements in list (indexing)
- Operations on Lists:
  - Concatenation
  - Repetition
  - Membership
  - Slicing
- Traversing a list
- Programming problems on lists

Assignment - 4

- Read and note down the built-in functions in List (table: 9.1) page – 193
- Read the summary (page-204)
- Q1 to Q7 from chapter- 9 “Lists” (NCERT, page-205)

Programming Assignment - 4

- Write a program to reverse a list without using reverse() function and slicing operation
- Write a program to input a list of numbers and find the smallest and largest number from the list.
- Write a program to read a list of n integers and find their median.
- Write a program to read a list of elements. Modify this list so that it does not contain any duplicate elements, i.e., all elements occurring multiple times in the list should appear only once.