CONTENTS

(Learning Outcomes) ☺

LIST in Python

XI CS 2020-21

→ **Lists Introduction:** Definition, Creation of a list, Traversal of a list.
→ **Operations on a list –** concatenation/joining, repetition/replication, membership, Comparison of Lists;
→ **Functions/methods–**len(), list(),append(), extend(), insert(), count(), index(), remove(), pop(), reverse(), sort(),min(), max(), sum()
→ **Lists Slicing;**
→ **Nested lists;**
→ **finding the maximum, minimum, mean of numeric values stored in a list**
→ **linear search on list of numbers**
→ **counting the frequency of elements in a list**

XI IP 2020-21

Lists: list operations - creating, initializing, traversing and manipulating

LIST

INTRODUCTION

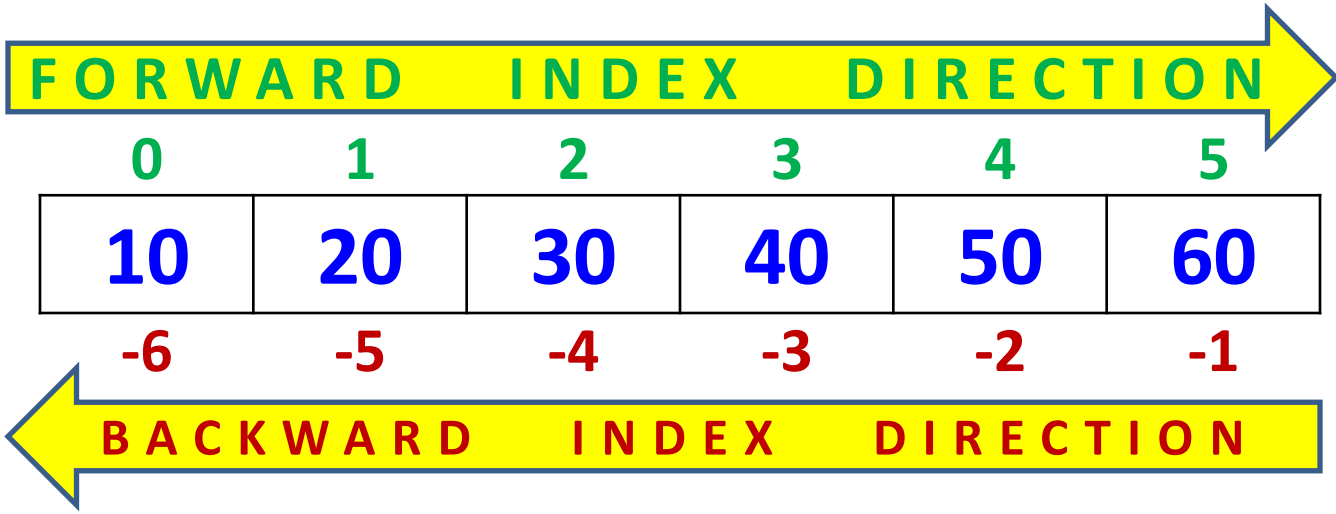
What is a LIST ?

CONTENTS

- **A list is a collections of items which are formed by placing a comma-separated-list of expressions in square brackets.**
- **Each item/element has its own index number.**
- **Index of first item is 0 and the last item is n-1. Here n is number of items in a list.**
- **Lists are mutable sequences i.e. we can change elements of a list in place.**
- **Lists can contain values of mixed data types.**

INDEXING in a LIST

CONTENTS



NOTE:-

The index (also called subscript) is the numbered position of a letter in the LIST. In python, indices begin from

0,1,2,…. upto (length-1) in the ***forward direction*** ***and***

-1,-2,-3,….. (-length) in the ***backward direction.***

where length is the length of the List.

CREATION of LIST

CONTENTS

- **→Lists Intro:**
  - List Definition
  - Indexing in a List
  - **Creation of a list**
  - Traversal of a list
- →Operations on a list
- →Functions/methods
- → Lists Slicing;
- → Nested lists;
- →finding the maximum, minimum, mean
- → linear search on list
- → counting the frequency

```
>>> [1,2,3,4,5]
[1, 2, 3, 4, 5]
>>> ['a','g','n','i']
['a', 'g', 'n', 'i']
>>> [10,20,12.5,'P',64]
[10, 20, 12.5, 'P', 64]
>>> L=[10,20,30,40,50,60]
>>> L
[10, 20, 30, 40, 50, 60]
```

List of numbers created without name

List of characters created without name

List of mixed data type created without name

List of numbers with name **L**

```
>>> []
[]
>>> L=[]
>>> print(L)
[]
>>> l=list()
>>> l
[]
```

Empty List created without name

Empty List created with name **L**

Empty List created with name **l** using funcion **list()**

Creation of LIST contd.. (Long List and Nested List)

CONTENTS

Creation of a long list with name *MultiTens*

```
>>> MultiTens=[10,20,30,40,50,60,70,80,90,100,110,120,130,140,150,160,170,180,19
0,200,210,220,230,240,250,260,270,280,290,300]
>>> MultiTens
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180
, 190, 200, 210, 220, 230, 240, 250, 260, 270, 280, 290, 300]
```

Creation of a Nested List with name L that is List inside a List

```
>>> L=[10,[50,'World'],40,['India']]
>>> L
[10, [50, 'World'], 40, ['India']]
```

Creation of LIST from existing sequences

CONTENTS

Creating list from string

```
>>> L1=list('hello')
>>> L1
['h', 'e', 'l', 'l', 'o']
```

Creating list from tuple

```
>>> t=('I','n','d','i','a')
>>> L2=list(t)
>>> L2
['I', 'n', 'd', 'i', 'a']
```

Creating list by taking input from the keyboard(user)

```
>>> L3=list(input('Enter list elements'))
Enter list elementsGlobe
>>> L3
['G', 'l', 'o', 'b', 'e']
```

**Note: All the elements are considered as string

Creating list by taking input from the keyboard [using eval()]]method]

```
>>> L4=eval(input('Enter list you want to add'))
Enter list you want to add[11,22,30,50]
>>> L4
[11, 22, 30, 50]
```

**Note: eval( ) identifies the type by looking at the given expression

TRAVERSAL of a LIST

CONTENTS


```
>>> L=[10,20,25,34,50,'Hi',30]
>>> for i in L:
        print(i)


10
20
25
34
50
Hi
30
>>> L[0]
10
>>> L[4]
50
>>> L[6]
30
>>> L[-1]
30
>>> L[-6]
20
```

```
>>> C=['I','n','d','i','a']
>>> for i in C:
        print(i)


I
n
d
i
a
```

TRAVERSAL of NESTED LIST

CONTENTS

Enclosing List

```
>>> M=[10,[15,30],[80,'Mask',60]]
```

Nested List 1

Nested List 2

```
>>> M[1][1]
30
>>> M[2][1]
'Mask'
>>> M[0]
10
>>> M[1][0]
15
```

Note: Denoting Nested elements using Index nos.

M[0] → 10

M[1] → [15,30]

M[1][0]→15

M[1][1]→30

M[2]→ [80,'Mask',60]

M[2][0]→80

M[2][1]→'Mask'

M[2][2]→60

Assign/Change values in a LIST

CONTENTS

Forward Index→ 0 1 2 3

-4 -3 -2 -1 → Backward index

```
>>> MyList=[20,'Apples',30,'Oranges']
>>> MyList
[20, 'Apples', 30, 'Oranges']
>>> MyList[1]='Mangoes'
>>> MyList[-2]=50
>>> MyList
[20, 'Mangoes', 50, 'Oranges']
>>> MyList[2]
50
>>> MyList[-2]
50
>>> MyList[-1]
'Oranges'
>>> MyList[-4]=100
>>> MyList
[100, 'Mangoes', 50, 'Oranges']
>>> MyList[-4]
100
```

Assigning / Changing values of elements of a List

Assigning / Changing values of elements of a List

OPERATIONS ON LIST

OPERATIONS on LIST Concatenation

CONTENTS

Concatenation refers to Joining of two objects

```
>>> [10,20,30]+[40,50]
[10, 20, 30, 40, 50]
```

List CAN be concatenated or joined with another List

```
>>> [10,20,30]+40
Traceback (most recent call last):
  File "<pyshell#12>", line 1, in <module>
    [10,20,30]+40
TypeError: can only concatenate list (not "int") to list
```

List CANNOT be concatenated with a number

```
>>> [10,20,30]+'Hi'
Traceback (most recent call last):
  File "<pyshell#13>", line 1, in <module>
    [10,20,30]+'Hi'
TypeError: can only concatenate list (not "str") to list
```

List CANNOT be concatenated with a string

```
>>> [10,20,30]+5j+6
Traceback (most recent call last):
  File "<pyshell#14>", line 1, in <module>
    [10,20,30]+5j+6
TypeError: can only concatenate list (not "complex") to list
```

List CANNOT be joined with a complex nos.

```
>>> [10,20,30]+[40,'Hi']
[10, 20, 30, 40, 'Hi']
```

List CAN be concatenated or joined with another List

OPERATIONS on LIST Replication

CONTENTS

Replication refers to Repetition of objects as it is. Here, in LIST we can see NON-VECTORISED operation with a list where the entire list as a whole is replicated.

NOTE: Vectorised operation, refers to the replication of individual elements of the object.

```
>>> [10,20,30,'Mask']*2
[10, 20, 30, 'Mask', 10, 20, 30, 'Mask']
>>> RL=[3,'Hi',20]
>>> RL*4
[3, 'Hi', 20, 3, 'Hi', 20, 3, 'Hi', 20, 3, 'Hi', 20]
```

Replicating the unnamed List two times

Replicating the named List **RL** four times

OPERATIONS on LIST Membership

CONTENTS

Membership operator tests for Membership in a sequence. Returns Boolean **True** or **False**

OPERATOR	DESCRIPTION
in	Results to **True** value **if it finds** a variable at the LHS of **in** operator in the sequence mentioned in the RHS of the **in** operator, else it returns False
not in	Results to **True** value **if it does not find** a variable at the LHS of **in** operator in the sequence mentioned in the RHS of the **in** operator, else it returns False

```
L=[10,20,45,60]
n=int(input('Enter a no.'))
if n in L:
    print(n,' is a member of ',L)
else:
    print(n,' is NOT a member of ',L)
```

```
Enter a no.10
10  is a member of  [10, 20, 45, 60]
```

OPERATIONS on Comparison of LIST

CONTENTS

Comparison of list returns Boolean **True** or **False.**

```
>>> L1=[11,22,33]
>>> L2=[11,22,33]
>>> L3=[11,20,33]
>>> L1==L2
True
>>> L1<L2
False
>>> L1>L2
False
>>> L1==L3
False
>>> L1==L2
True
```

```
>>> [10,20,30]>[15,20,30]
False
>>> [10,20,30]<[10,25,30]
True
```

Relational operators ==,!=,<,<=,>,>= are used in comparing two objects. Here, for example two List objects are compared.

Relational operators ==,!= can be used in comparing two objects eg. List with number/string/tuple. But, Relational operators <,<=,>,>= always returns ERROR when used in comparing two objects eg. List with number/string/tuple.

```
>>> [10,20,30]<=60
Traceback (most recent call last):
  File "<pyshell#12>", line 1, in <module>
    [10,20,30]<=60
TypeError: '<=' not supported between instances of 'list' and 'int'
```

Eg. List CANNOT be compared with a number

Built-in FUNCTIONS/ METHODS ON LIST

Functions / Methods on LIST-len()

CONTENTS

len() function counts the no. of elements in the List

```
>>> L=[10,30,'GO','HAPPY']
>>> len(L)
4
>>> L1=[]
>>> len(L1)
0
>>> L2=[10,30,[50,'hand','wash'],60,[80,100]]
>>> len(L2)
5
```

TOTAL 5 elements of the List object

L2=[10,30,[50,'hand','wash'],60,[80,100]]

Functions / Methods on LIST-list()

CONTENTS

The list() function creates a list object.
A list object is a collection which is ordered and changeable.

```
>>> A=list((-2,-1,1,2))
>>> A
[-2, -1, 1, 2]
>>> print(A)
[-2, -1, 1, 2]
```

list() function takes elements/values inside the parantheses i.e. **()**

```
>>> Z=list(("Stay","Happy","and","Healthy"))
>>> z
Traceback (most recent call last):
  File "<pyshell#43>", line 1, in <module>
    z
NameError: name 'z' is not defined
>>> Z
['Stay', 'Happy', 'and', 'Healthy']
```

NOTE: Python language is case-sensitive.
Here, **Z** is the List object , so, **z** in small letters gives an error.

Functions / Methods on LIST-append()

CONTENTS

The **append()** function takes one single parameter and adds exactly one element at the end of the List. It takes exactly one element and returns no value.

```
>>> A
[-2, -1, 1, 2]
>>> A.append(30)
>>> A
[-2, -1, 1, 2, 30]
>>> A.append(40,50)
Traceback (most recent call last):
  File "<pyshell#51>", line 1, in <module>
    A.append(40,50)
TypeError: append() takes exactly one argument (2 given)
```

Here, **append()** function takes one single parameter . In this example, 30.

Trying to append two values gives an ERROR!!

Functions / Methods on LIST-extend()

CONTENTS

The extend() function adds iterable (Eg. Another List/ Tuple/String) at the end of the List.

```
>>> A=[-2,-1,1,2]
>>> A.extend([30])
>>> A
[-2, -1, 1, 2, 30]
>>> A.extend([4,50,77])
>>> A
[-2, -1, 1, 2, 30, 4, 50, 77]
>>> B=['know','python','bytes']
>>> A.extend(B)
>>> A
[-2, -1, 1, 2, 30, 4, 50, 77, 'know', 'python', 'bytes']
```

Here, trying to extend list A , with another unnamed List containing one element [30].

Here, trying to extend list A , with another unnamed List containing three elements [4,50,77].

Here, trying to extend list A , with another named List B containing three elements ['know','python','bytes']

```
>>> A.extend((10))
Traceback (most recent call last):
  File "<pyshell#8>", line 1, in <module>
    A.extend((10))
TypeError: 'int' object is not iterable
>>> A.extend((10,20))
>>> A
[-2, -1, 1, 2, 30, 4, 50, 77, 'know', 'python', 'bytes', 10, 20]
>>> A.extend('Global')
>>> A
[-2, -1, 1, 2, 30, 4, 50, 77, 'know', 'python', 'bytes', 10, 20, 'G', 'l', 'o', 'b', 'a', 'l']
```

Here, trying to extend list A , with a number 10. Hence error !!!

NOTE: A no. inside parentheses () is not a tuple, until it ends with a comma ,

Here, trying to extend list A , with an unnamed tuple with two values

Here, trying to extend list A , with a string.
Every character is considered as an element of the list.

Functions / Methods on LIST- insert()

CONTENTS

insert( ) method,

inserts an item / element at a given position.

It takes two arguments and returns no value.

ListObject.insert(<*position*>,<item>)

Position denotes the index no.

Item denotes the item/element/value to be inserted.

Forward Index→ 0 1 2
-3 -2 -1 →Backward index

```
>>> L=[11,-24,'Hi']
>>> L.insert(1,'Wifi')
>>> L
[11, 'Wifi', -24, 'Hi']
>>> L.insert(0,77)
>>> L
[77, 11, 'Wifi', -24, 'Hi']
>>> L.insert(-2,'Net')
>>> L
[77, 11, 'Wifi', 'Net', -24, 'Hi']
>>> L.insert(-1,55)
>>> L
[77, 11, 'Wifi', 'Net', -24, 55, 'Hi']
```

NOTE: Whenever the element is inserted, whether forward or backward index… it is always inserted at the left hand side of the current index item's position.
Prepended→ inserted at the beginning i.e. index 0
Appended → inserted at the last

Functions / Methods on LIST- count()

CONTENTS

count( ) method,

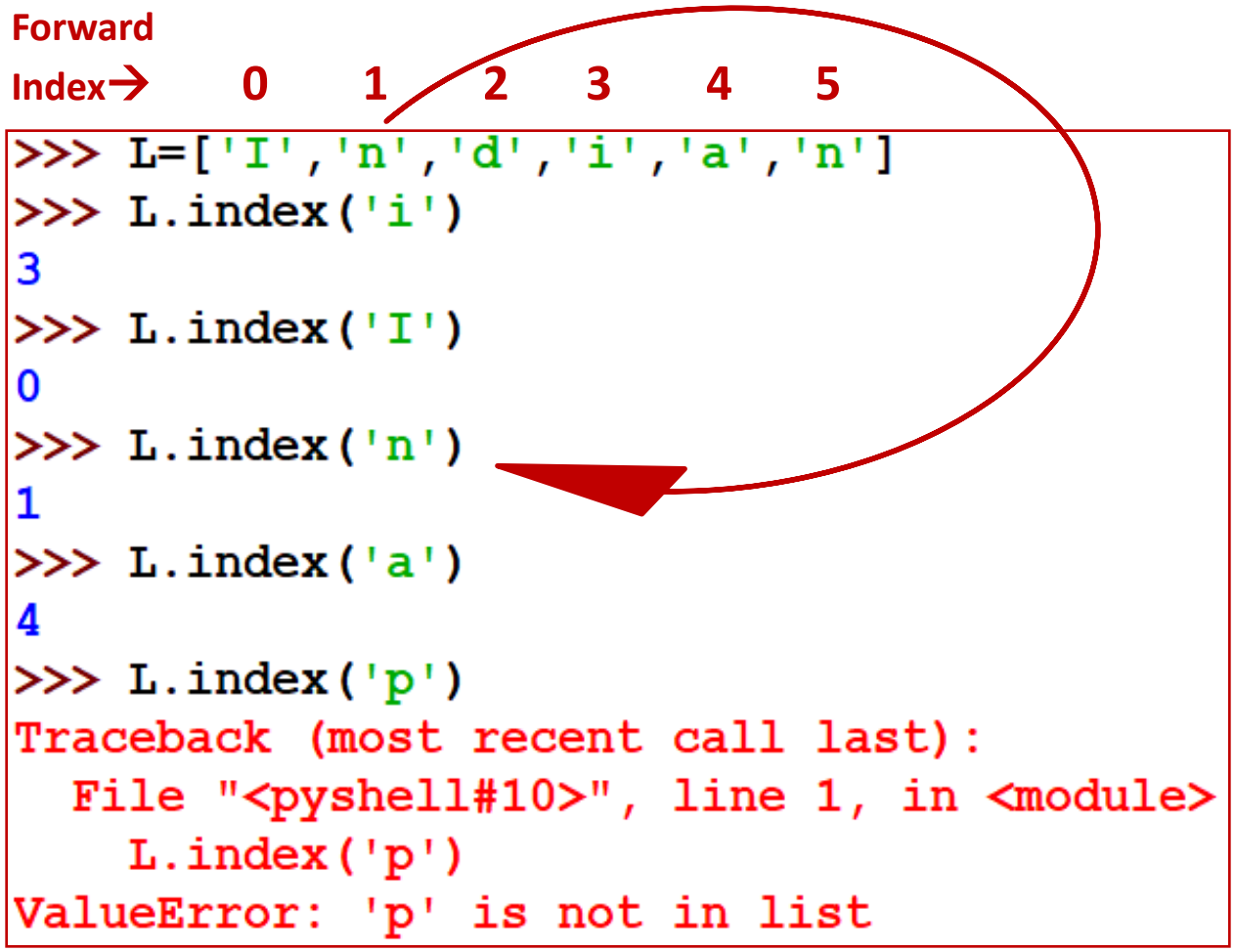
returns the count of the item *(no. of times the item /element/value appears)* that is passed as argument. If the given item is not in the list, it will return 0

```
>>> L=['a','p','p','l','e',20,'p','e','a','r',32]
>>> L.count('p')
3
>>> L.count('a')
2
>>> L.count(20)
1
>>> L.count('b')
0
>>> L.count(10)
0
```

Functions / Methods on LIST- index()

CONTENTS

index( ) method,
returns the index of the first matched item.



Functions / Methods on LIST- remove()

CONTENTS

remove( ) method,
removes the *first occurrence* of given item from the list (from left-to-right)
remove( ) will report an error if there is no such value.

```
>>> L=['I','n','d','i','a','n']
>>> L.remove('n')
>>> L
['I', 'd', 'i', 'a', 'n']
>>> L.remove('n')
>>> L
['I', 'd', 'i', 'a']
>>> L.remove('I')
>>> L
['d', 'i', 'a']
>>> L.remove('c')
Traceback (most recent call last):
  File "<pyshell#21>", line 1, in <module>
    L.remove('c')
ValueError: list.remove(x): x not in list
```

Functions / Methods on LIST- pop()

CONTENTS

pop( ) method is used to remove the item from the list. It takes one optional argument . The element being deleted is returned by the method

```
>>> L=['I','n','d','i','a','n']
>>> L.pop()
'n'
>>> L
['I', 'n', 'd', 'i', 'a']
>>> L.pop()
'a'
>>> L
['I', 'n', 'd', 'i']
>>> L.pop(1)
'n'
>>> L
['I', 'd', 'i']
>>> L.pop(7)
Traceback (most recent call last):
  File "<pyshell#29>", line 1, in <module>
    L.pop(7)
IndexError: pop index out of range
>>> L.pop('d')
Traceback (most recent call last):
  File "<pyshell#30>", line 1, in <module>
    L.pop('d')
TypeError: 'str' object cannot be interpreted as an integer
```

DIFFERENCE between pop() and remove()

The pop( ) method removes an individual item (from the right most end → *by default) or the value/element present at the index number mentioned in the parameter* and returns it. Whereas remove( ) searches the first occurrence of the item and removes it from the list

Functions / Methods on LIST- reverse()

CONTENTS

reverse( ) method,

reverses the items of the list.
This doesn't create a new list.
It takes no argument and return no list.
Reverses the list 'in place' and does not return anything.

```
>>> L=[10,'Tech','Talk',-5,80]
>>> L.reverse()
>>> L
[80, -5, 'Talk', 'Tech', 10]
>>> L.reverse()
>>> L
[10, 'Tech', 'Talk', -5, 80]
```

Functions / Methods on LIST- sort()

CONTENTS

sort( ) method,

sort( ) method and **sort(reverse=True)** sorts or orders the items of the list, by default in *increasing order.*
sort(reverse=True) for getting the list in *decreasing order*

```
>>> L=[30,10,40,35,-11,0,55]
>>> L.sort()
>>> L
[-11, 0, 10, 30, 35, 40, 55]
>>> L.sort(reverse=False)
>>> L
[-11, 0, 10, 30, 35, 40, 55]
>>> L.sort(reverse=True)
>>> L
[55, 40, 35, 30, 10, 0, -11]
```

```
>>> L=['c','a','t']
>>> L.sort()
>>> L
['a', 'c', 't']
```

```
>>> L=[10,-2,'Hi']
>>> L.sort()
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    L.sort()
TypeError: '<' not supported between instances of 'str' and 'int'
```

NOTE: The sort( ) method won't be able to sort the values of a list if it contains *complex numbers/tuple / mixed datatype* as its elements.

Functions / Methods on LIST- min(),max(),sum()

CONTENTS

min( ) method,

Returns the minimum out of the *elements/values/items* of the given List

max( ) method,

Returns the maximum out of the *elements/values/items* of the given List

sum( ) method,

Returns the sum of all the values/elements/items of the given List.

```
>>> L=[10,-2,50,-70,0]
>>> min(L)
-70
>>> max(L)
50
>>> sum(L)
-12
```

Functions / Methods on LIST- min(),max(),sum() contd..

```
>>> L=['c','a','t']
>>> min(L)
'a'
>>> max(L)
't'
>>> sum(L)
Traceback (most recent call last):
  File "<pyshell#13>", line 1, in <module>
    sum(L)
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

List of string . Maximum and Minimum is done on the basis of the ASCII values of the characters from the beginning.
Eg. 'A'=65, 'a' = 97, 'b'=98 and so on

A List with mixed data type cannot return values for the *min*, *max* and *sum* functions.

```
>>> N=[10,'c',-2,'a',50,'t']
>>> min(N)
Traceback (most recent call last):
  File "<pyshell#19>", line 1, in <module>
    min(N)
TypeError: '<' not supported between instances of 'str' and 'int'
>>> max(N)
Traceback (most recent call last):
  File "<pyshell#20>", line 1, in <module>
    max(N)
TypeError: '>' not supported between instances of 'str' and 'int'
>>> sum(N)
Traceback (most recent call last):
  File "<pyshell#21>", line 1, in <module>
    sum(N)
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Functions / Methods on LIST- clear()

CONTENTS

clear( ) method,
removes all the items from the list and the list becomes empty after executing this function.

```
>>> Flower=[5,'Roses',10,'Lily']
>>> Flower.clear()
>>> Flower
[]
```

Here, name of the List is Flower

Empty List

LIST

slicing

LIST slicing

L[start:stop] **creates a list slice out of list L with elements falling between indexes start and stop, not including value at the stop index.**

L[start:stop:step] **creates a list slice out of list L with elements falling between indexes start and stop, not including value at the stop index, and skipping step elements in between.**

NOTE: In absence of any start, stop, step values, we have Default start index as the beginning, stop index as the last index and step value as +1.

```
>>> List=['K','N','O','W','P','Y','T','H','O','N','B','Y','T','E','S']
>>> List
['K', 'N', 'O', 'W', 'P', 'Y', 'T', 'H', 'O', 'N', 'B', 'Y', 'T', 'E', 'S']
```

FORWARD INDEX DIRECTION

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
K	N	O	W	P	Y	T	H	O	N	B	Y	T	E	S
-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

BACKWARD INDEX DIRECTION

LIST slicing

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
K	N	O	W	P	Y	T	H	O	N	B	Y	T	E	S
-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> List[3:10]
['W', 'P', 'Y', 'T', 'H', 'O', 'N']
>>> List[5:50]
['Y', 'T', 'H', 'O', 'N', 'B', 'Y', 'T', 'E', 'S']
>>> List[-4:6]
[]
>>> List[-4:25]
['Y', 'T', 'E', 'S']
>>> List[10:30]
['B', 'Y', 'T', 'E', 'S']
>>> List[-10:12]
['Y', 'T', 'H', 'O', 'N', 'B', 'Y']
```

Start index 3, stop index is (10-1=9)

Start index 5, stop index is 50 which is beyond the last index in forward direction. Hence, it will stop at the last index

Start index -4(backward), stop index is (6-1=5 (forward)). Here, it is not getting any value in the range, hence empty.

Start index -4(which is 'Y' as per the backward index,) stop index is 25 which is beyond the last index. Hence, it will stop at the last index.

Start index 10, stop index is 30 which is beyond the last index in forward direction. Hence, it will stop at the last index

Start index -10(which is 'Y' as per the backward index,) stop index is 12-1=11 (forward indexing). Hence, it will stop at the (stop index-1)., 'Y' is at 11 index no.

LIST SLICING contd.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
K	N	O	W	P	Y	T	H	O	N	B	Y	T	E	S
-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> List[-3:-12]
[]
>>> List[-12:-3]
['W', 'P', 'Y', 'T', 'H', 'O', 'N', 'B', 'Y']
>>> List[-3:-12:-1]
['T', 'Y', 'B', 'N', 'O', 'H', 'T', 'Y', 'P']
>>> List[-20:-5]
['K', 'N', 'O', 'W', 'P', 'Y', 'T', 'H', 'O', 'N']
>>> List[1:13:2]
['N', 'W', 'Y', 'H', 'N', 'Y']
>>> List[2:40:5]
['O', 'H', 'T']
>>> List[-1:-12:-3]
['S', 'Y', 'O', 'Y']
>>> List[::3]
['K', 'W', 'T', 'N', 'T']
```

Start index -3(backward), stop index is -12(backward). Here, it is not getting any value in the range as no step value mentioned in reverse. Hence empty.

Start index -12(which is 'W' as per the backward index,) stop index is -3 (which takes values till index -3-1= -4)

Start index -3(backward), stop index is -12(backward). Here, it is getting value in the range as step value is -1 mentioned in reverse.

Start index -20(which is beyond backward index,) stop index is -5 which takes value till (-5-1=-6). Here, -6 is 'N'.

Start index 1, stop index is 13 (till 13-1=12),step value is 2 in forward direction. Hence, it will fetch/take alternate elements with a skip of 1 value in between

?

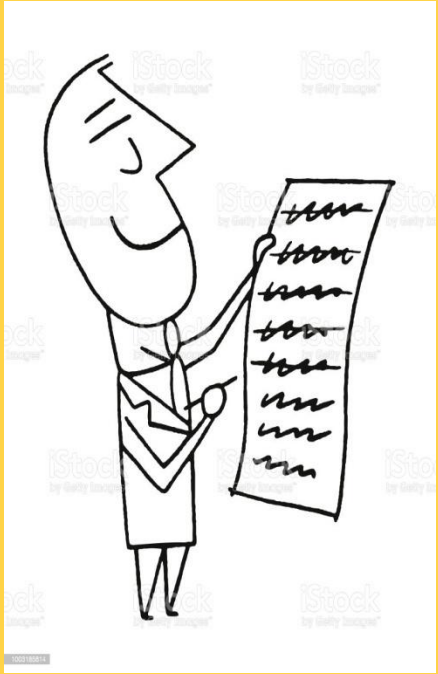
TRY TO UNDERSTAND YOURSELF. A task for you.

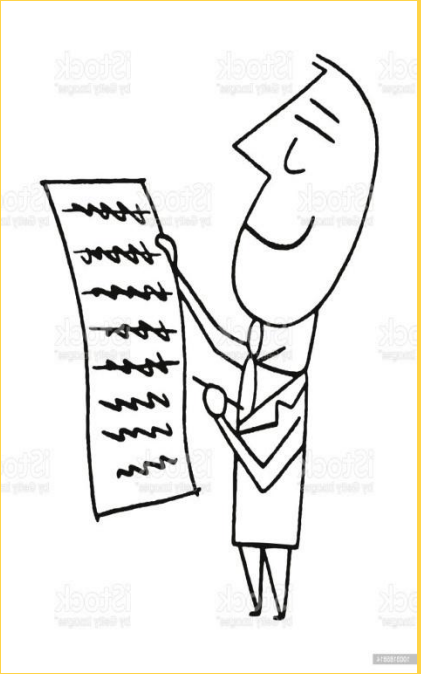
Also write the slice for List to get the output as

'P', 'Y', 'T', 'H', 'O', 'N'

Default start and stop is beginning and end index of the list. Here, step is 3.

NESTED LIST





NESTED LISTS

CONTENTS

- →Lists Intro
- →Operations on a list
- →Functions/methods
- → Lists Slicing;
- → **Nested lists**;
- →finding the maximum, minimum, mean
- → linear search on list
- → counting the frequency

Enclosing List

```
>>> M=[10,[15,30],[80,'Mask',60]]
```

Nested List 1 | Nested List 2

Note: Denoting Nested elements using Index nos.

M[0] → 10

M[1] → [15,30]

M[1][0]→15

M[1][1]→30

M[2]→ [80,'Mask',60]

M[2][0]→80

M[2][1]→'Mask'

M[2][2]→60

```
>>> M=[10,[15,30],[80,'Mask',60]]
>>> M
[10, [15, 30], [80, 'Mask', 60]]
>>> M[0]
10
>>> M[1]
[15, 30]
>>> M[2]
[80, 'Mask', 60]
>>> M[2][1]
'Mask'
>>> M[2][0]
80
>>> M[1][1]
30
>>> M[0]+M[2][2]
70
```

Simple Programs on LIST

Program to find the MAXIMUM element from a list of element along with its index in the list without using max() library function

CONTENTS

- →Lists Intro
- →Operations on a list
- →Functions/ methods
- → Lists Slicing;
- → Nested lists;
- **→finding the maximum**, minimum, mean
- → linear search on list
- → counting the frequency

```
#Program to find maximum among N numbers in a List
L=[ ]
N=eval(input("enter size of list : "))
for i in range(N):
    L.append(eval(input("enter "+ str(i) + " element : ")))
print("Numbers in the list are ")
print(L)
max=L[0]
for i in range(1,N):
    if L[i]>max:
        max=L[i]

print("Maximum value in the list = ", max," at index no.",L.index(max))
```

OUTPUT

```
enter size of list : 5
enter 0 element : 87
enter 1 element : -24
enter 2 element : 35
enter 3 element : 62
enter 4 element : -98
Numbers in the list are
[87, -24, 35, 62, -98]
Maximum value in the list =  87  at index no. 0
```

Program to find the MINIMUM element from a list of element along with its index in the list without using min() library function

CONTENTS

- →Lists Intro
- →Operations on a list
- →Functions/ methods
- → Lists Slicing;
- → Nested lists;
- **→finding the minimum**, maximum , mean
- → linear search on list
- → counting the frequency

```
#Program to find maximum among N numbers in a List
L=[ ]
N=eval(input("enter size of list : "))
for i in range(N):
    L.append(eval(input("enter "+ str(i) + " element : ")))
print("Numbers in the list are ")
print(L)
min=L[0]
for i in range(1,N):
    if L[i]<min:
        min=L[i]

print("Minimum value in the list = ", min," at index no.",L.index(min))
```

OUTPUT

```
enter size of list : 5
enter 0 element : 87
enter 1 element : -22
enter 2 element : 30
enter 3 element : 58
enter 4 element : -11
Numbers in the list are
[87, -22, 30, 58, -11]
Minimum value in the list =  -22  at index no. 1
```

Program to find the MEAN (AVERAGE) of all the elements of the list

CONTENTS


```
#Program to calculate the Average (or mean) of numbers in the list

L=[ ]
sum=0
avg=0
N=eval(input("enter total numbers to be entered"))
for i in range(0,N):
    L.append(eval(input("Enter"+str(i)+"Element")))
    sum=sum+L[i]
avg=sum/N
print("The given List is: " ,L)
print("The average of the numbers= ",avg)
```

OUTPUT

```
enter total numbers to be entered6
Enter0Element11
Enter1Element55
Enter2Element22
Enter3Element67
Enter4Element33
Enter5Element40
The given List is:  [11, 55, 22, 67, 33, 40]
The average of the numbers=  38.0
```

Program to search an element in the List (LINEAR SEARCH)

CONTENTS


```
#Program to search a number in the list (Linear Search)
L=[ ]
N=eval(input("enter total numbers to be entered"))
for i in range(0,N):
    L.append(eval(input("Enter"+str(i)+"Element")))
s=eval(input("enter element to be searched"))
flag=-1
for i in range(0,N):
    if(s==L[i]):
        flag=i
        break

if(flag!=-1):
    print("element found at index",flag)
else:
    print("element not found")
```

OUTPUT

```
enter total numbers to be entered10
Enter0Element100
Enter1Element80
Enter2Element35
Enter3Element-24
Enter4Element68
Enter5Element42
Enter6Element-90
Enter7Element-35
Enter8Element75
Enter9Element20
enter element to be searched-90
element found at index 6
```

Program to count frequency (/no. of times of occurrence) of an element in the List

(FREQUENCY COUNT)

CONTENTS


```
#Program to count frequency of a given element in a list of numbers
L=[]
N=eval(input("enter total numbers to be entered"))
for i in range(0,N):
    L.append(eval(input('enter list element '+str(i)+ '  ')))
s=int(input(" Enter element whose frequency you want to count"))
count=0
for i in range(0,N):
    if s==L[i]:
      count+=1
if count==0:
      print(s," not found in the given list")
else:
      print("In the list ",s," has frequency as ", count)
```

OUTPUT

```
enter total numbers to be entered8
enter list element 0  11
enter list element 1  22
enter list element 2  77
enter list element 3  22
enter list element 4  55
enter list element 5  66
enter list element 6  22
enter list element 7  77
 Enter element whose frequency you want to count22
In the list  22  has frequency as  3
```