OPEN ELECTIVE
OCS1903-PROGRAMMING USING PYTHON
WEEK 6

CONTENTS

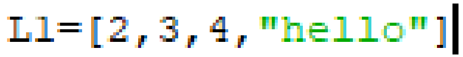
- Python Lists
  - List Definition
  - List Creation
  - Accessing List
  - Mutability
  - List Operators
  - List Functions
  - MCQ
  - Problem Solving

List – []

1. List is a data structure(collection) in python which is used to store the sequence of element of various types.

2. **List** is a collection which is ordered and changeable. Allows duplicate members

3. [] is used to represent the list data structure.



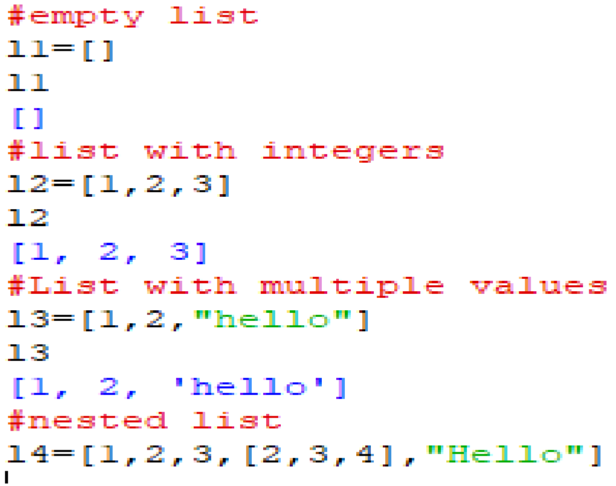
```
L1=[2,3,4,"hello"]
```

4. List are mutable-means the elements in the list can be modified.

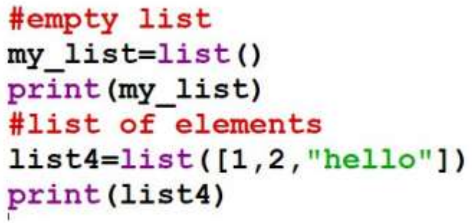
5. The items in the list are separated with the comma (,).Elements of the list can be accessed by their index.

List creation

List is created by placing all the elements within square brackets , separated by commas or using list() command



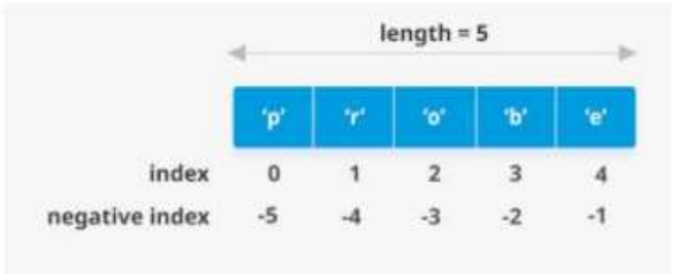
```
#empty list
l1=[]
l1
[]
#list with integers
l2=[1,2,3]
l2
[1, 2, 3]
#List with multiple values
l3=[1,2,"hello"]
l3
[1, 2, 'hello']
#nested list
l4=[1,2,3,[2,3,4],"Hello"]
```

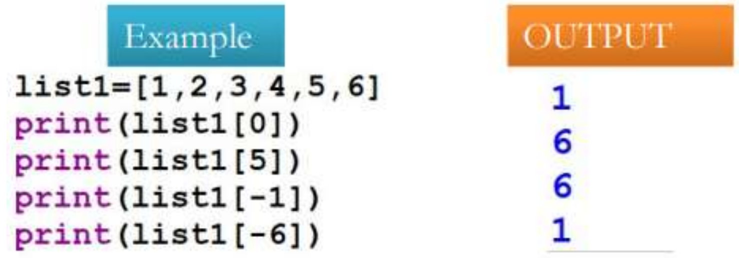


```
#empty list
my_list=list()
print(my_list)
#list of elements
list4=list([1,2,"hello"])
print(list4)
```

Accessing List – List Indexing

- List elements are indexed starting from 0.
- List supports negative indexing starting from the list element indexed as -1.
- List elements can be accessed through their index.





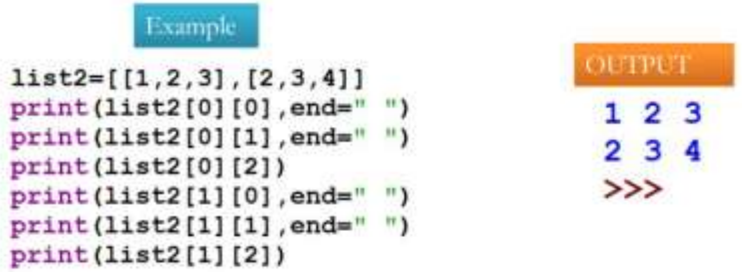
Example

```
list1=[1,2,3,4,5,6]
print(list1[0])
print(list1[5])
print(list1[-1])
print(list1[-6])
```

OUTPUT

```
1
6
6
1
```

Accessing Nested List



Example

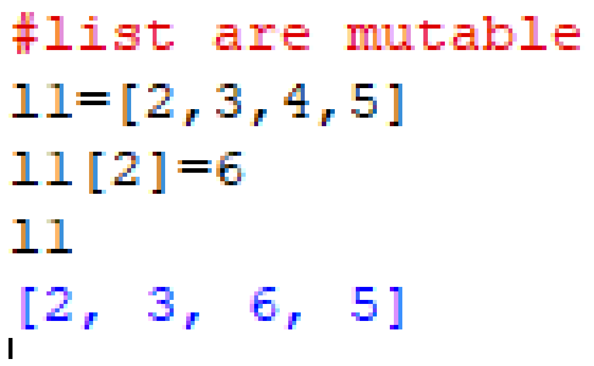
```
list2=[[1,2,3],[2,3,4]]
print(list2[0][0],end=" ")
print(list2[0][1],end=" ")
print(list2[0][2])
print(list2[1][0],end=" ")
print(list2[1][1],end=" ")
print(list2[1][2])
```

OUTPUT

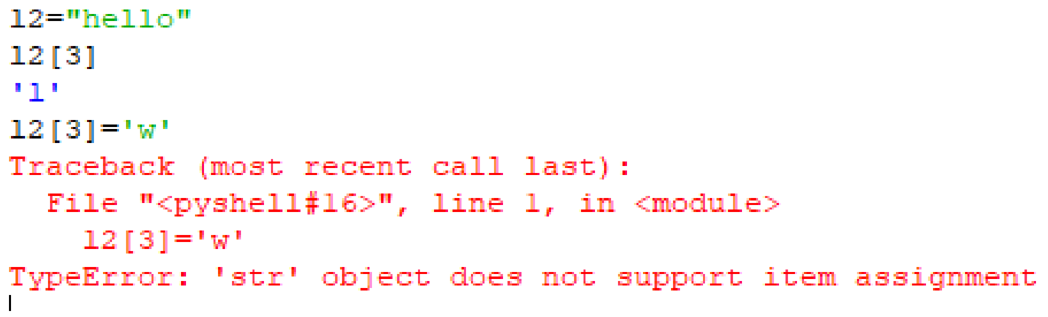
```
1 2 3
2 3 4
>>>
```

List Mutability

Unlike Strings List are mutable , so the ability for certain types of data to be changed without entirely recreating it. Elements can be modified.



```
#list are mutable
l1=[2,3,4,5]
l1[2]=6
l1
[2, 3, 6, 5]
```



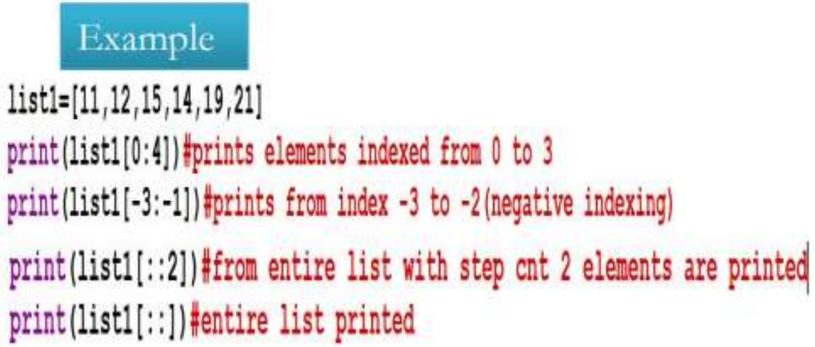
```
l2="hello"
l2[3]
'l'
l2[3]='w'
Traceback (most recent call last):
  File "<pyshell#16>", line 1, in <module>
    l2[3]='w'
TypeError: 'str' object does not support item assignment
```

List Operators

- Slicing
- Concatenation
- Repetition
- Membership
- Identity

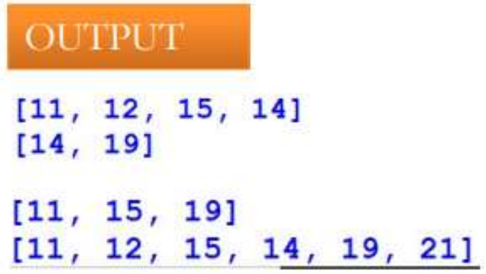
List Slicing

- To access a range of elements from the list , we can go for list slicing.
- Slicing operator[::] , that is list[start : end : step]
- End always represents end-1. example list[1:3] will print 1 and 2 elements from list.



Example

```
list1=[11,12,15,14,19,21]
print(list1[0:4])#prints elements indexed from 0 to 3
print(list1[-3:-1])#prints from index -3 to -2(negative indexing)
print(list1[::2])#from entire list with step cnt 2 elements are printed
print(list1[::])#entire list printed
```



OUTPUT

```
[11, 12, 15, 14]
[14, 19]

[11, 15, 19]
[11, 12, 15, 14, 19, 21]
```

List concatenation and repetition

- To concatenate two lists we can use + operator
- * enables the list elements to be repeated multiple times

Example

```
list1=[1,2,3]
print(list1)
list2=[5,6,7]
print(list2)
list1=list1+list2
print(list1)
```

OUTPUT

```
[1, 2, 3]
[5, 6, 7]
[1, 2, 3, 5, 6, 7]
```

Example

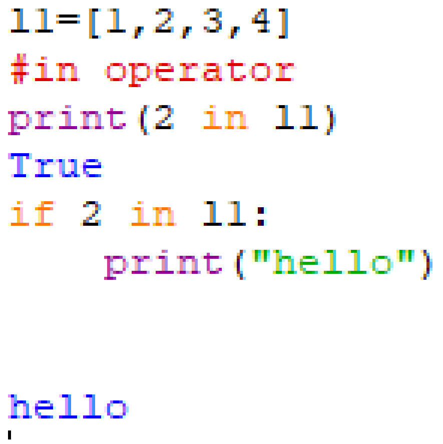
```
list1=[1,2,3]
print(list1)
list1=list1*2
print(list1)
```

OUTPUT

```
[1, 2, 3]
[1, 2, 3, 1, 2, 3]
```

List –in and is operators

- In operator returns only two possible values True when the element is in the list, and False when it is not.
- Is checks whether given object is List or not.

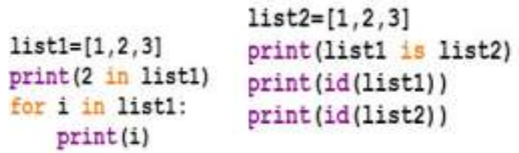


```
l1=[1,2,3,4]
#in operator
print(2 in l1)
True
if 2 in l1:
    print("hello")


hello
```

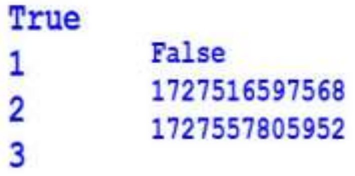
Example

OUTPUT



```
list1=[1,2,3]
print(2 in list1)
for i in list1:
    print(i)
```

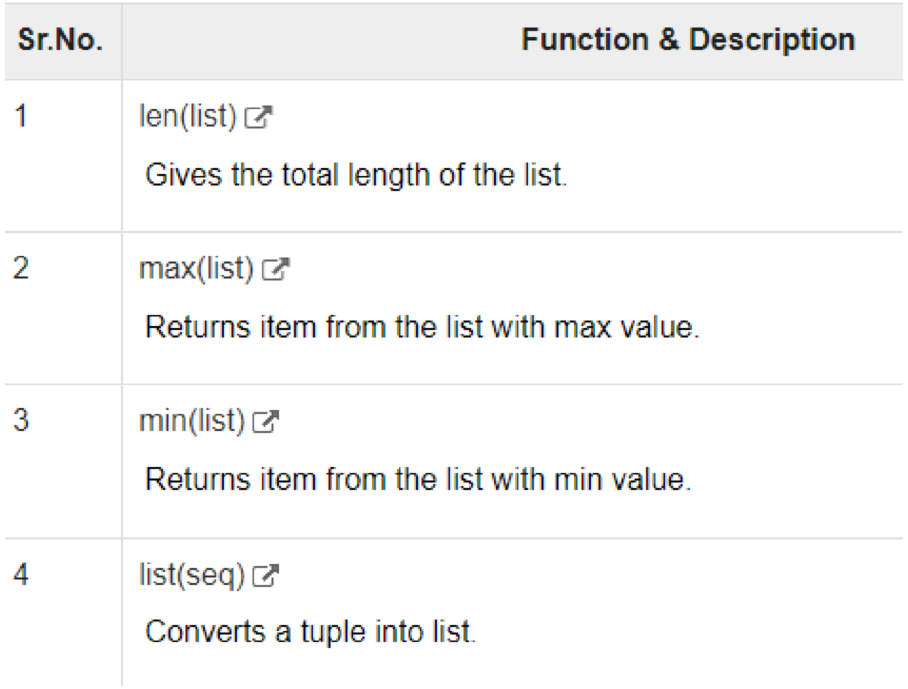
```
list2=[1,2,3]
print(list1 is list2)
print(id(list1))
print(id(list2))
```



```
True
1
2
3
```

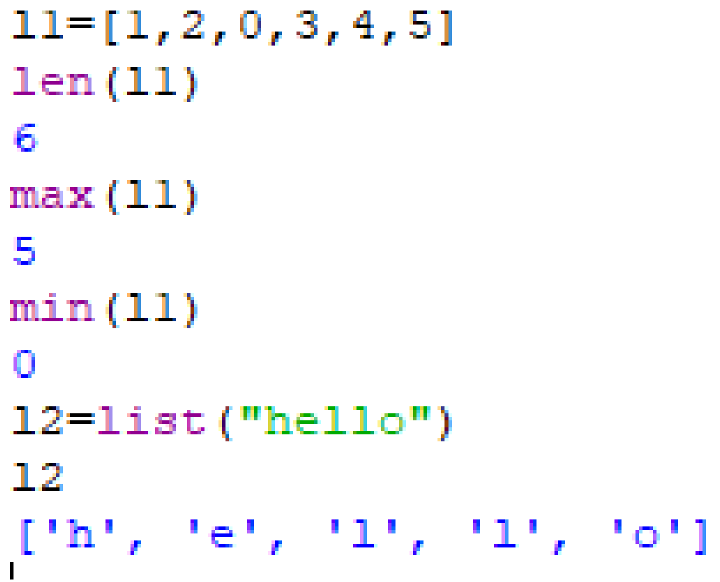
```
False
1727516597568
1727557805952
```

List Built in Functions



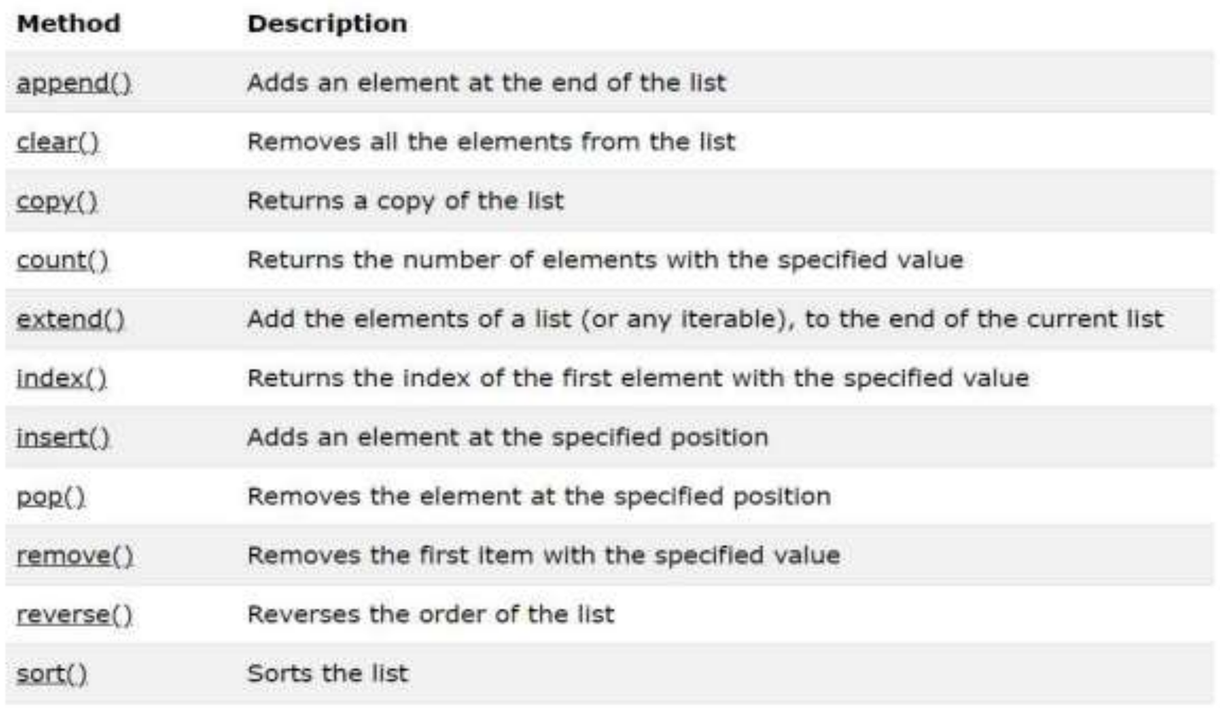
Sr.No.	Function & Description
1	len(list) Gives the total length of the list.
2	max(list) Returns item from the list with max value.
3	min(list) Returns item from the list with min value.
4	list(seq) Converts a tuple into list.

List Built in Functions Example



```
l1=[1,2,0,3,4,5]
len(l1)
6
max(l1)
5
min(l1)
0
l2=list("hello")
l2
['h', 'e', 'l', 'l', 'o']
```

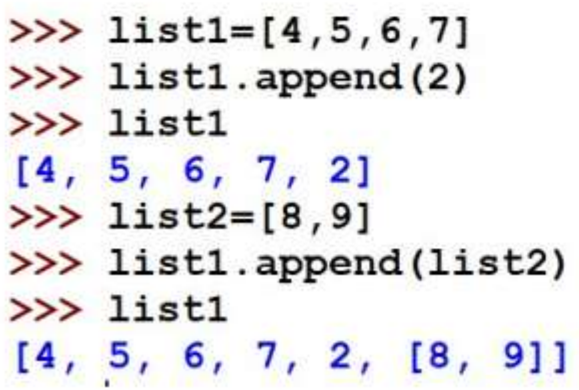
List Built in Methods



Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the first item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

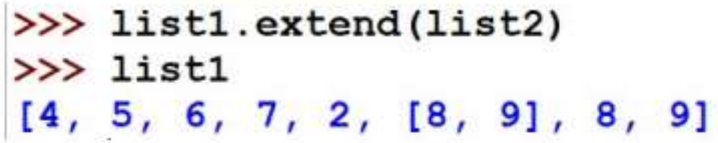
Append Extend and Insert Methods

Append()



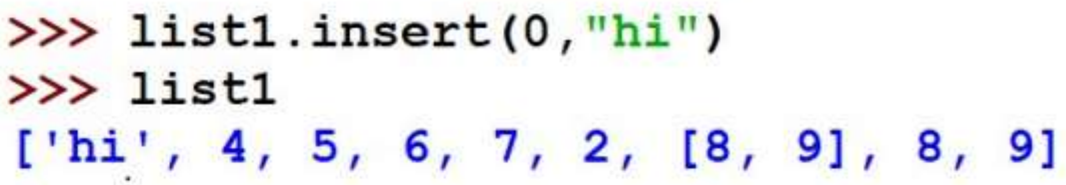
```
>>> list1=[4,5,6,7]
>>> list1.append(2)
>>> list1
[4, 5, 6, 7, 2]
>>> list2=[8,9]
>>> list1.append(list2)
>>> list1
[4, 5, 6, 7, 2, [8, 9]]
```

Extend()



```
>>> list1.extend(list2)
>>> list1
[4, 5, 6, 7, 2, [8, 9], 8, 9]
```

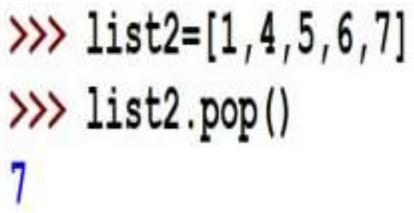
insert()



```
>>> list1.insert(0,"hi")
>>> list1
['hi', 4, 5, 6, 7, 2, [8, 9], 8, 9]
```

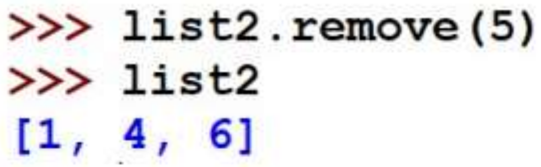
pop remove and reverse Methods

pop()



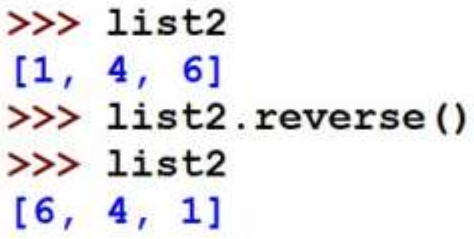
```
>>> list2=[1,4,5,6,7]
>>> list2.pop()
7
```

remove()



```
>>> list2.remove(5)
>>> list2
[1, 4, 6]
```

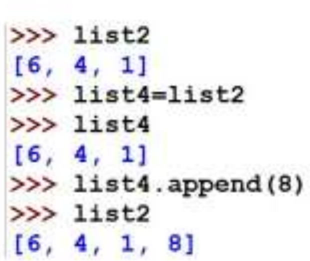
reverse()



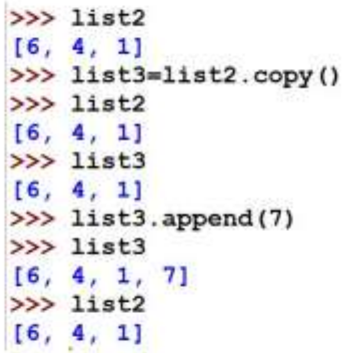
```
>>> list2
[1, 4, 6]
>>> list2.reverse()
>>> list2
[6, 4, 1]
```

copy count and index Methods

copy()

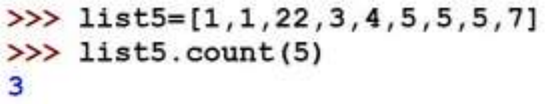


```
>>> list2
[6, 4, 1]
>>> list4=list2
>>> list4
[6, 4, 1]
>>> list4.append(8)
>>> list2
[6, 4, 1, 8]
```



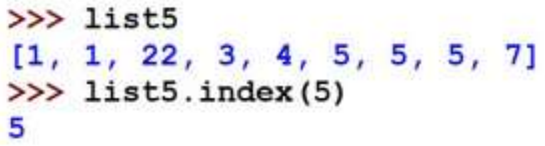
```
>>> list2
[6, 4, 1]
>>> list3=list2.copy()
>>> list2
[6, 4, 1]
>>> list3
[6, 4, 1]
>>> list3.append(7)
>>> list3
[6, 4, 1, 7]
>>> list2
[6, 4, 1]
```

count()



```
>>> list5=[1,1,22,3,4,5,5,5,7]
>>> list5.count(5)
3
```

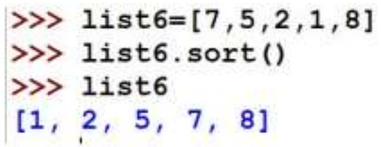
index()



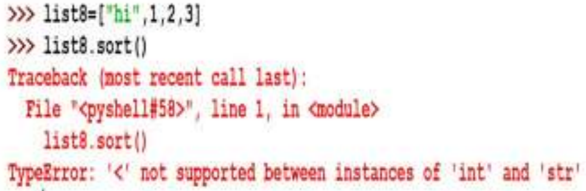
```
>>> list5
[1, 1, 22, 3, 4, 5, 5, 5, 7]
>>> list5.index(5)
5
```

Clear and sort method

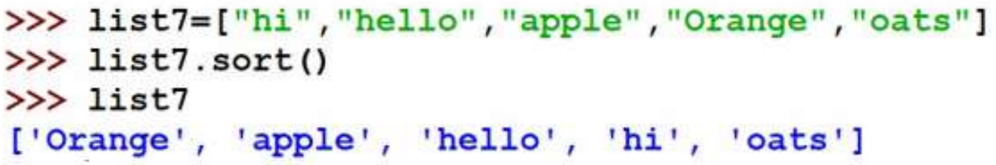
sort()



```
>>> list6=[7,5,2,1,8]
>>> list6.sort()
>>> list6
[1, 2, 5, 7, 8]
```

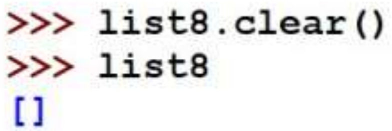


```
>>> list8=["hi",1,2,3]
>>> list8.sort()
Traceback (most recent call last):
  File "<pyshell#58>", line 1, in <module>
    list8.sort()
TypeError: '<' not supported between instances of 'int' and 'str'
```



```
>>> list7=["hi","hello","apple","Orange","oats"]
>>> list7.sort()
>>> list7
['Orange', 'apple', 'hello', 'hi', 'oats']
```

clear()



```
>>> list8.clear()
>>> list8
[]
```

Nested List

- Nested list can be initialized/values can be moved using two ways
  - Using Nested for loop
  - Using list comprehension

```
#initializing nested list
l=[]
m=int(input())
n=int(input())
for i in range(0,m):
    r=[]
    for j in range(0,n):
        r.append(0)
    l.append(r)
print(l)
```

Output

```
2
2
[[0, 0], [0, 0]]
```

```
#list comprehension
l = [[j for j in range(5)] for i in range(3)]
print(l)
```

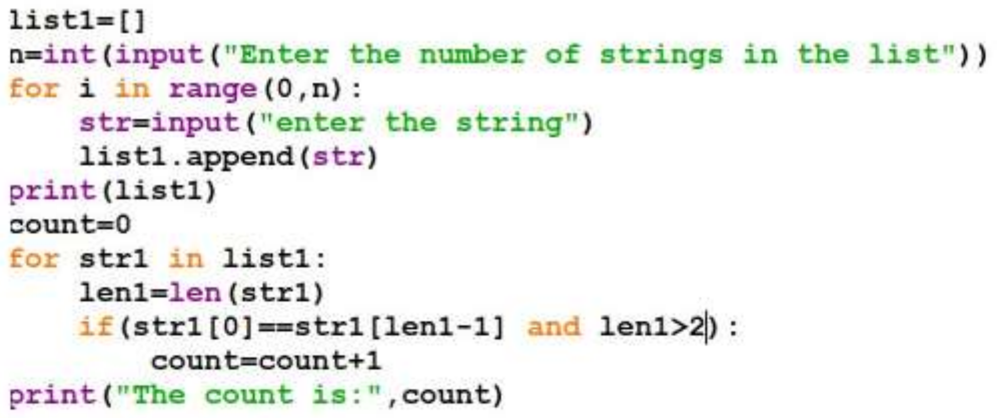
Output

```
[[0, 1, 2, 3, 4], [0, 1, 2, 3, 4], [0, 1, 2, 3, 4]]
```

Python code 1

Write a Python program to count the number of strings where the string length is 2 or more and the first and last character are same from a given list of strings. Sample List : ['abc', 'xyz', 'aba', '1221'] Expected Result : 2.

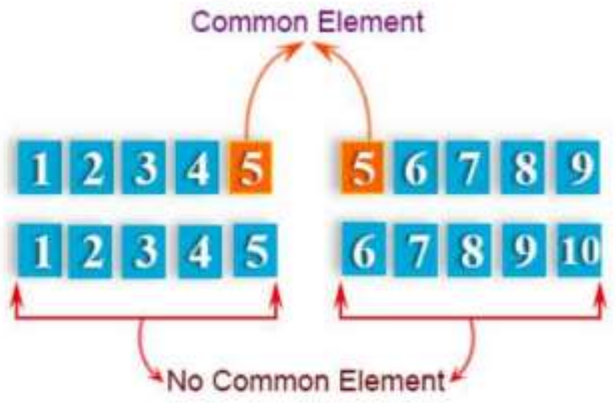
Input:
List of strings
Eg: ['abc', 'xyz', 'aba', '1221']
Output:
Count of the strings
1.Length more than 2
2.First and last character Should be same
aba and 1221 count=2

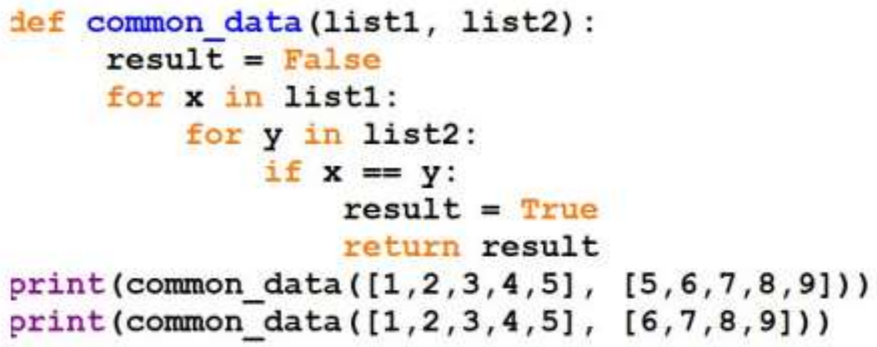


```
list1=[]
n=int(input("Enter the number of strings in the list"))
for i in range(0,n):
    str=input("enter the string")
    list1.append(str)
print(list1)
count=0
for str1 in list1:
    len1=len(str1)
    if(str1[0]==str1[len1-1] and len1>2):
        count=count+1
print("The count is:",count)
```

Python code 2

Write a Python function that takes two lists and returns True if they have at least one common member.





```
def common_data(list1, list2):
        result = False
        for x in list1:
                for y in list2:
                        if x == y:
                                result = True
                                return result
print(common_data([1,2,3,4,5], [5,6,7,8,9]))
print(common_data([1,2,3,4,5], [6,7,8,9]))
```

Python code 3

Write a Python program to add two matrix

```
l1=[]
r1=[]
l2=[]
r2=[]
#getting input to Listl
m=int(input())
n=int(input())
for i in range(0,m):
    r1=[]
    for j in range(0,n):
        r1.append(int(input()))
    l1.append(r1)
#getting input to Listl
for i in range(0,m):
    r2=[]
    for j in range(0,n):
        r2.append(int(input()))
    l2.append(r2)
#addind two matrix
l3=[[0 for i in range(0,m)] for j in range(0,n)]
for i in range(0,m):
    for j in range(0,n):
        l3[i][j]=l1[i][j]+l2[i][j]
print(l3)
```

```
2
2
1
2
3
4
1
2
3
4
[[2, 4], [6, 8]]
```

MCQ

1. What is the output when we execute list("hello")?
   a) ['h', 'e', 'l', 'l', 'o']
   b) ['hello']
   c) ['llo']
   d) ['olleh']

Output:
A

1. What will be the output of below Python code?

```
list1=[8,0,9,5]
print(list1[::-1])
```

A. [5,9,0,8]
B. [8,0,9]
C. [8,0,9,5]
D. [0,9,5]

Output:
A

THANK
YOU