# Computational Thinking and Programming - 2

## Lists

XI

## Definition

A Python list is a standard data type of Python that can store a sequence of elements belonging to any type. Python lists are mutable, i.e. you can change the elements of a list in place.

Each element of a list can be individually accessed using its index. These elements can be accessed using either forward indexing or backward indexing.

You can access any element as <listname>[index]. The index can be a forward index or a backward index.

- 0, 1, 2,.... In the forward direction
- -1, -2, -3….. In the backward direction

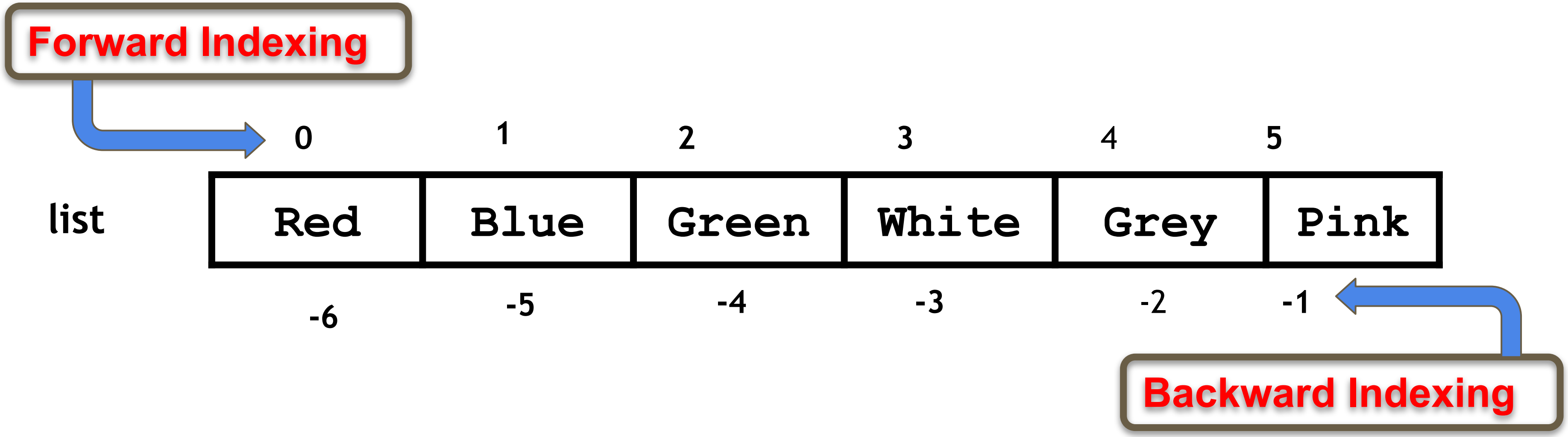
## Examples of Lists

Examples of Lists are:

```
L = [ ]                                 # Empty list
Number = [10,20,30,40]                  # List of integer values
Values = [1.2,5.6,8,9,123.8]            # List of float values
Names = ['Arun', 'Sheetal',’Mohit']     # List of strings

# List of different types of data
Student = [10,'Akshay Rawat',99.5]      # List containing int, string and float values
```

# Indexing


Forward Indexing
0
1
2
3
4
5
list
Red
Blue
Green
White
Grey
Pink
-6
-5
-4
-3
-2
-1
Backward Indexing

## Indexing

The elements in a list are ordered (numbered), starting from 0.

It means that the first element of the list is numbered 0, the second element is numbered 1, third is numbered 2, and so on.

These numbers are called **indices** of list elements.

Python supports negative indices also.

In case of negative indices, -1 is the index for the last element, -2 for the second last element, and so on.

## Creating Lists

To create a list, put a number of values in square brackets.

In general we can create a list in the form given below:

```
L = [ ]                        # creating an empty list
L = [value1, value2,....]      #creating list with values
```

Example :

```
L1=[1,2,3,4,5]            # list containing elements of the same type
L2=[1,"RAHUL",80.0]       # list containing elements of mixed data types
```

# Creating Lists (Contd.)

A list can be created from a sequence as per the syntax:

```
L = list(<sequence>)
```

where sequence can be any kind of sequence object including strings, tuples and lists.

Consider the following examples:

```
>>>L1 = list("PYTHON")
>>>print(L1)
['P','Y','T','H','O','N']
```

We can also create a list from single characters or single digits entered via the keyboard.

```
>>>L1 = list(input("Enter the values of the list:"))

Enter the values of the list:123456

>>>L1

['1','2','3','4','5','6']

>>> s1=eval(input("enter the values of a list"))

[10,20,30,40]

>>> s1

[10, 20, 30, 40]
```

## Types of Lists

- **Empty list** :The empty list is a list containing no values.

  You can create an empty list as:

  L = list()   or   L = [ ]

- **Long lists** : If a list contains many elements, then to enter such long lists, you can split them across several lines such as:

  List1 = [1,2,3,4,5,6,7,8,9,10,11,12,13]

- **Nested lists** : A list can have an element in it, which itself is a list.

  Such a list is called a nested list, such as:

  List2 = [1,2,[3,4],5]

## Accessing elements of List:

- By using the list name and index numbers.

  For example:

  ```
  color = ['Red', 'Green', 'Pink', 'Black', 'Orange']
  print (color[2])              displays 'Pink'.
  print (color[-2])             displays 'Black'.
  print(color[6])               error
  ```

**Note : If you give an index outside the legal indices ( 0 to length-1 or -length, -length+1), Python will raise IndexError**

## Accessing Elements of List:

- By using for loop where the range is specified by the list name.

  For example:

```
color = ['Red', 'Green', 'Pink', 'Black', 'Orange']
for i in color:
   print(i, end=' ')
```

OR

```
color = ['Red', 'Green', 'Pink', 'Black', 'Orange']
for i in range ( 0, len(color)):
  print(color[i], end=' ')
```

## Strings vs Lists

| Similarity | Difference |
|---|---|
| Lists are sequences just like strings.<br><br>They also index their individual elements just like strings do.<br><br>We can have forward indexing as 0,1,2,3,4………… and backward indexing as -1,-2,-3,-4…….. | Strings are not mutable while lists are mutable.<br><br>Individual elements of the string cannot be changed while the elements of the lists can be changed. |

# Operations on Lists

O
P
E
R
A
T
I
O
N
S

1. Concatenation
2. Replication
3. Membership
4. Slicing

# Concatenation

The + operator adds one list to the end of the other list. This operator requires both the operands to be of list type. You cannot add a number/ complex number / string.

**Syntax : listname1+listname2**

**Example**

```
l1 = [10,20,30]
l2 = [40,50,60]
print("List1 :")
print(l1)
print("List2 :")
print(l2)
l3 = l1 + l2
print(l3, len(l3))
```

OUTPUT

```
List1 :
[10,20,30]
List2 :
[40,50,60]
[10,20,30,40,50,60] 6
```

# Replication

The * operator as a replication operator needs two operand i.e a list and an integer. Python replicates the list elements the given number of times.

**Syntax : listname * integer data item**

**Example**

```
l1 = [10,20,30]
print(l1*2)
```

OUTPUT

```
[10,20,30,10,20,30]
```

# Comparison

All relational and comparison operators (**==, >, <, <=, >=, !=**) can be used to compare two lists .

Python internally compares individual elements of the lists in lexicographical order. This means that to check if the two lists are equal, each corresponding element is compared and the two sequence must be of the same type.

**Examples**

| | |
|---|---|
| [1,2,8,9] < [9,1] | Returns True when comparing corresponding first element of two lists : **1< 9 is True** |
| [1,2,8,9] < [1,2,9,1] | Returns True when comparing corresponding third element of two lists : **8 < 9 is True** |

# Membership

The operators in and not in are used to check whether element exists in the given list or not.

The in operator  returns True if an element exists in the given list; False otherwise.

The not in operator returns True if an element does not exist in the given list;  False otherwise.

**Syntax : element in <listname>**
**element not in <listname>**

**Example**

```
l1 = [10,20,30]
print(10 in l1)
print(33 in l1)
print(23 not in l1)
```

OUTPUT

```
True
False
True
```

# List Slicing

List slicing refers to retrieve few elements from the list which falls between two indexes.To extract the slice  two indexes are given in square bracket separated by a colon (:) .

**Syntax : listname [start : end : step]**

**where start, end and step are integers**
**start represents the starting index of list**
**end denotes the end index of list which is not inclusive**
**step denotes the distance between the elements to be sliced**

# Slicing of lists

```
list1 = [1, 2, 3, 5, 6, 7, 10, 11,12]
list2=list1[2:6]
print("List after slicing :")
print(list2)
```

OUTPUT

```
List after slicing :
[3,5,6,7]
```

**Lists also supports slice steps.**

```
list1 = [1, 2, 3, 5, 6, 7, 10, 11, 12]
list2=list1[ 1 : 8 : 2 ]
print("List after slicing with step :")
print(list2)
```

OUTPUT

```
List after slicing with
step:
[2,5,7,11]
```

# Slicing for modification

Examples

```
list1=["one", "two", "THREE"]
list1[0:2]=[0,1]
print("List after modification:")
print(list1)
```

OUTPUT

```
List after
modification :
[0, 1, 'THREE']
```

If the range of the list slice is much outside the length of the list, it will simply add the values at the end.

```
list1=[ 10, 20, 30]
list1[ 10 : 20 ] = "abcd"
print("List after modification:")
print(list1)
```

OUTPUT

```
List after modification:
[10, 20, 30, 'a', 'b', 'c', 'd']
```

# Exercise :

Find the output of the following statements :

```
A=[ 3, 6, 7,  8, 10, 12, 23, 33, 16, 6, 2, 1 ]
```

(i) print(len(A))

(ii) print(A[3])

(iii) print(A[-3])

(iv) print(A[3:])

(v) print(A[-3:])

(vi) print(A[::3])

(vii) print(A[3::])

(viii) print(A[3::-2])

(ix) print(A[-3::-2])

(x) print(A[::-3])

(xi) print(A[3:-3:3])

(xii) print(A[3:-3:-3])

(xiii) print(len(A[3:]))

(xiv) print(len(A[-3:])

(xv) print(len(A[::-3]))

## Solutions :

(i) 12

(ii) 8

(iii) 6

(iv) [8, 10, 12, 23, 33, 16, 6, 2, 1]

(v) [6, 2, 1]

(vi)[3, 8, 23, 6]

vii)[8, 10, 12, 23, 33, 16, 6, 2, 1]

(viii) [8, 6]

(ix) [6, 33, 12, 8, 6]

(x) [1, 16, 12, 7]

(xi) [8, 23]

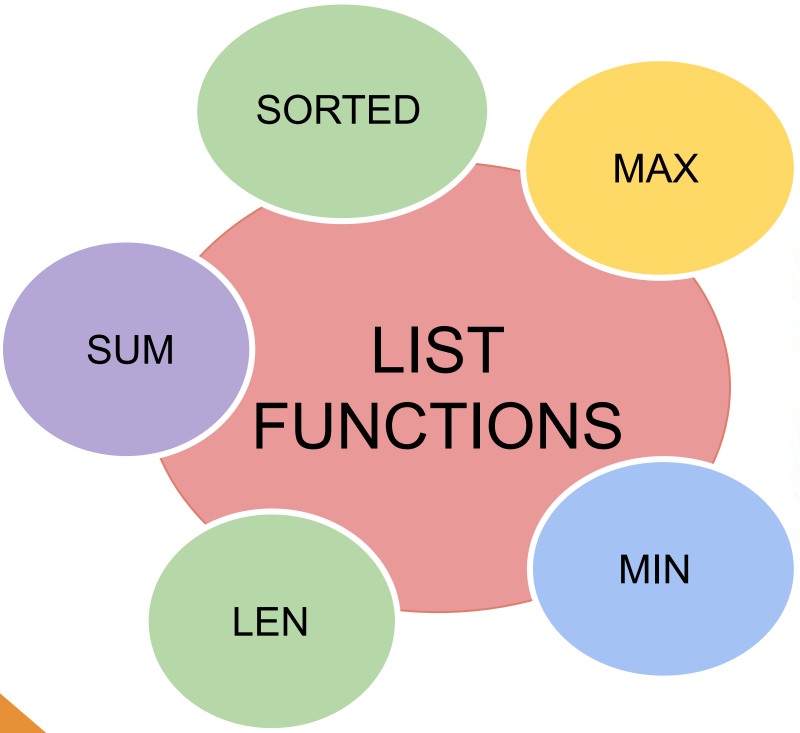
(xii) [ ]

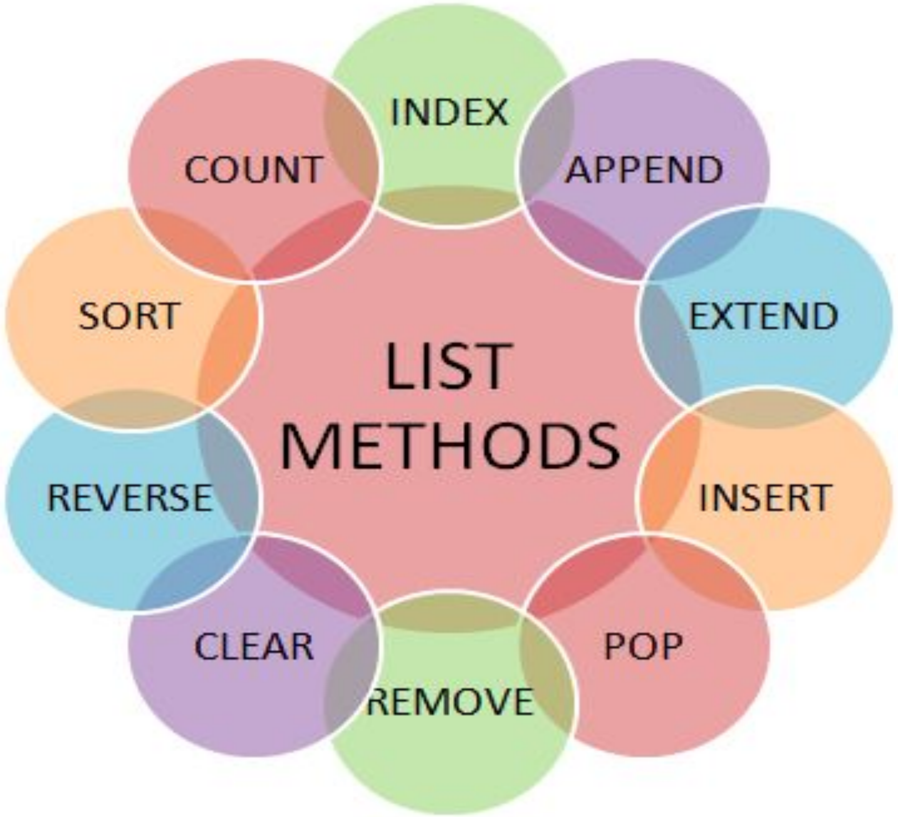
(xiii) 9

(xiv) 3

(xv) 4

# List Functions and Methods





## List Functions and Methods

**1. len()**

Returns the length of the list or the number of elements in the list

**Syntax : len(listname)**

**Example**

```
list1=[10,20,30,40,50]
print("Length:",len(list1))
```

OUTPUT

```
Length : 5
```

# List Functions and Methods

**2. max()**

Returns the largest element of the list. All the elements must be of the same data type.

**Syntax : max(listname)**

**Example**

```
list1=[1,45,34,11,78]
print("Maximum:",max(list1))
```

OUTPUT

```
Maximum: 78
```

## List Functions and Methods

**3. min()**

Returns the smallest element of the list. All the elements must be of the same data type.

**Syntax : min(listname)**

**Example**

```
list1=[100,45,34,11,78]
print("Minimum:",min(list1))
```

OUTPUT

```
Minimum: 11
```

## List Functions and Methods

4. **sorted()**

This function returns the sorted values in ascending order (by default) or descending order in the list. All the values in the list must be of the same data type.

**Syntax : sorted(<list>,<reverse = True/False (default)>)**

**Example**

```
list1=[1,45,34,11,78]
print("Sorted List:")
print(sorted(list1))
```

OUTPUT

```
Sorted List:
[1,11,34,45,78]
```

# List Functions and Methods

## 5. sum()

This function returns the sum of all elements in the list.

**Syntax : sum(<listname>)**

**Example**

```
list1=[10,20,30,40,50]
print("Sum:")
print(sum(list1))
```

OUTPUT

```
Sum:
150
```

# List Functions and Methods

**6. append()**

The append() method adds a single item to the end of the list.

**Syntax : listname.append(element)**

**Example**

```
list1=[1,100,78]
print("List :",list1)
list1.append(88)
print("After appending :")
    print(list1)
```

OUTPUT

```
List :[1,100,78]
After appending :
[1,100,78,88]
```

# List Functions and Methods

**7. extend()**

It is used for adding multiple elements to the list. It takes a list as an argument and appends all the elements of the argument list to the list object on which extend() is applied.

**Syntax : listname.extend(<listname>)**

**Example**

```
list1=[1,100,78]
list2=[10,11,12]
print("List :",list1)
list1.extend(list2)
print("After extending :")
    print(list1)
```

OUTPUT

```
List :[1,100,78]
After extending :
[1,100,78,10,11,12]
```

## List Functions and Methods

**8. insert()**

It is used to insert an element somewhere in between/ any specified position in the list.

**Syntax : listname.insert(<pos>,<element>)**

**Example**

```
list1=[1,100,78]
print("List :",list1)
list1.insert(2,90)
print("After inserting :")
    print(list1)
```

OUTPUT

```
List :[1,100,78]
After inserting :
[1,100,90,78]
```

# insert() method

Note : If the pos index is greater than the len(list) (i.e. the index is out of bound), the object is simply appended to the list.

```
l1=['a','e','o','u']
print('List :',l1)
l1.insert(8,'z')
print('After inserting')
print(l1)
```

OUTPUT

```
List :['a','e','o','u']
After inserting :
['a','e','o','u','z']
```

# insert() method

Note : If the pos index is less than zero or not equal to any of the valid negative indexes of the list, the object is prepended to the list (added at the beginning of the list)

```
l1=['a','e','o','u']
l1.insert(-9, 'xyz' )
print('After inserting')
print(l1)
```

OUTPUT

```
After inserting :
['xyz', 'a', 'e', 'o', 'u']
```

# List Functions and Methods

7. **pop()**

It is used to remove an item from the specified position in the list. If no index is specified then pop() removes the last element from the list. It gives an error in case there is no element at the index specified.

**Syntax :** listname.pop(<index>)

**Example**

```
list1=["x","y","z"]
print("List:",list1)
x=list1.pop(1)
print("After popping:")
print(list1)
```

OUTPUT

```
List: ["x","y","z"]
After popping :
["x","z"]
```

# List Functions and Methods

## 8. remove()

It removes the first occurrence of the given item from the list.
If the item to be deleted is not present in the list, Python generates an exception.

**Syntax :** **listname.remove(<value>)**

**Example**

```
l1=[1,2,3,4,3,2,1]
print("List:",l1)
l1.remove(3)
print("After removing:")
print(l1)
```

OUTPUT

```
List: [1,2,3,4,3,2,1]
After removing :
[1,2,4,3,2,1]
```

# List Functions and Methods

## 9. clear()

It is used to remove all the items from the list and the list becomes empty after the function. The list object still exists as an empty list.

**Syntax : listname.clear()**

**Example**

```
l1=[1,2,3,4,3,2,1]
l1.clear()
print("After clearing:")
print(l1)
```

OUTPUT

```
List:[1,2,3,4,3,2,1]
After clearing :
[ ]
```

## List Functions and Methods

**10. count()**

It returns the count of the item that has been passed as an argument. If the item is not present in the list, it returns zero.

**Syntax : listname.count(value)**

**Example**

```
l1=[11,22,33,44,33]
print("List:",l1)
x=l1.count(33)
print("After counting:")
print(x)
```

OUTPUT

```
List:[11,22,33,44,33]
After counting :
2
```

# List Functions and Methods

## 11. reverse()

It reverses the items of the list without creating any new list.

**Syntax :** listname.reverse()

**Example**

```
l1=[4,3,3,2,2,1,1]
print("List1:",l1)
l1.reverse()
print("After reversing:")
print(l1)
```

OUTPUT

```
List1: [4,3,3,2,2,1,1]
After reversing :
[1,1,2,2,3,3,4]
```

# del statement

It also be used to remove individual items/ all items identified by the slice from the list. If the del statement is used as del List, it deletes all the items as well as the list object.

**Syntax : del listname[index]**

**Example**

```
l1=[1,2,3,4,3,2,1]
print("List1:",l1)
del l1[4]
print("After deleting:")
print(l1)
```

OUTPUT

```
List: [1,2,3,4,3,2,1]
After deleting :
[1,2,3,4,2,1]
```

## Program #1

Program to calculate the average of the list of values entered by the user

```
l=eval(input("Enter list:"))
length=len(l)
m=s=0
for  i  in  range(length):
    s+=l[i]
m=s/length
print("Given list is:",l)
print("Sum of elements is : ",s)
print("Average of list is : ",m)
```

**Output :**

```
Enter list:[3,5,1,6,3]
Given list is: [3, 5, 1, 6, 3]
Sum of elements is :  18
Average of list is :  3.6
```

# Program #2

Program to find the maximum element from a list of numbers entered by the user along with the index of the largest element

```
l=eval(input("Enter list:"))
length=len(l)
mx=l[0]
ind=0
for  i in range(1,length):
    if l[i]>mx:
        mx=l[i]
        ind=i
print("Given list is:",l)
print("Maximum element ",mx,"at index ",ind)
```

**Output :**

```
Enter list:[3,5,1,6,3]
Given list is: [3, 5, 1, 6, 3]
Maximum element 6 at index 3
```

## Program #3

Program to enter a list of values and a number x to be searched. Search for x in the list. If present, display the position where found else display the message “element is not present in the list”

```
l=eval(input("Enter elements of the list:"))
length=len(l)
x=int(input("Enter the number"))
for  i  in  range(length):
     if x==l[i]:
         print("Element found at ",i,"position")
         break
else:
     print("Element is not present in the list")
```

**Output :**

```
Enter elements of the list:
[1,4,5,26,33]
Enter element to be searched:
26
Element found at position 4
```

# Program #4

Program to input N numbers from the user and find their sum and average. After that display all those numbers (entered by the user) which are greater than the average.

```
list1=[]
s=0
N=int(input("Enter the limit :"))
for i in range(N):
     num=int(input("Enter the number :"))
     list1=list1+[num]
     s=s+num
avg=s/N
print("Sum :",s)
print("Average :",avg)
print("Numbers which are greater than average:")
for i in list1:
     if i>avg:
          print(i)
```

**Output :**

```
Enter the limit :5
Enter the number :11
Enter the number :12
Enter the number :13
Enter the number :14
Enter the number :15
Sum : 65
Average : 13.0
Numbers which are greater
than average:
14
15
```

## Program #5

Program to count the frequency of a given element in a list of values.

```
l1=eval(input("Enter the list:"))
element=int(input("Enter element to be searched :"))
c=0
for  i  in range(len(l1)):
    if element==l1[i]:
       c+=1
if c==0:
   print(element," not found in given list")
else:
   print(element," has frequency of ",c," in given
   list")
```

**Output :**

```
Enter the list:[1,2,3,4,4,5,4]
Enter element to be searched :4
4  has frequency of  3  in given list
```

## Program #6

Program to input a list, named Pay, from the user and modify each element of Pay, as per the following rules:

| Existing value of Pay | Pay to be changed to |
|---|---|
| If less than 100000 | Add 25% in the existing value |
| If >= 100000 and <200000 | Add 20% in the existing value |
| If >= 200000 | Add 15% in the existing value |

# Program #6

```
pay=eval(input("Enter elements of the list:"))
print("Original List :",pay)
for  i  in range(len(pay)):
     if pay[i]<100000:
        pay[i]=pay[i]+pay[i]*0.25
     elif pay[i]<200000:
        pay[i]=pay[i]+pay[i]*0.20
    else:
        pay[i]=pay[i]+pay[i]*0.15
print("Updated Pay :",pay)
```

**Output :**

```
Enter elements of the list:
[10000, 200000, 350000]
Original List :
[10000, 200000, 350000]
Updated Pay :
[12500.0, 240000.0, 402500.0]
```

## Program #7

Program to input a list from the user and modify its content in such a way that the elements, which are multiples of 10 are swapped with the value present in the very next position in the list.

For example If the content of list P is: 91, 50, 54, 22, 30, 54

Then content of list P should become: 91, 54, 50, 22, 54, 30

## Program #7

```
l1=eval(input("Enter elements of the list:"))
print("Original List :",l1)
i=0
while i<len(l1)-1:
    if l1[i]%10==0:
        l1[i],l1[i+1]=l1[i+1], l1[i]
        i=i+2
    else:
        i=i+1
print("Updated List :",l1)
```

**Output :**

```
Enter elements of the list:[10,22,34,50,67]

Original List : [10, 22, 34, 50, 67]

Updated List : [22, 10, 34, 67, 50]
```

## Program #8

Program to input a list from the user and change all the multiples of 5 in the list to 5 and the rest of the elements to 0.

For example, if the list contains: [55, 43, 20, 16, 39, 90, 83, 40, 48, 25]

Then after executing the program, the list content should be changed as follows:

[5, 0, 5, 0, 0, 5, 0, 5, 0, 5]

## Program #8

```
l1=eval(input("Enter elements of the list:"))
print("Original List :",l1)
for i in range(len(l1)):
    if l1[i]%5==0:
        l1[i]=5
    else:
        l1[i]=0
print("Updated List :",l1)
```

**Output :**

```
Enter elements of the list:[1,2,30,4,5,50]

Original List : [1, 2, 30, 4, 5, 50]

Updated List : [0, 0, 5, 0, 5, 5]
```

## Program #9

Program to input a list of integers and replace each even element of the list with the sum of its digits and each odd element with the product of its digits.

For example, if the list is entered as: [100, 43, 20, 56, 32, 91, 80, 40, 45, 21]

After executing the program, the list content should be changed as follows:

[1, 12, 2, 11, 5, 9, 8, 4, 20, 2]

## Program #9

```
l=eval(input("Enter elements of the list:"))[11,22]
print("Original List :",l)
for i in range(len(l)):
    num=l[i]
    s=0
    p=1
    if l[i]%2==0:
        while num>0:
            r=num%10
            s=s+r
            num=num//10
        l[i]=s
    else:
        while num>0:
            r=num%10
            p=p*r
            num=num//10
        l[i]=p
print("Updated List :",l)
```

**Output :**

```
Enter elements of the list:

[100, 43, 20, 56, 32, 91]

Original List : [100, 43, 20, 56, 32, 91]

Updated List : [1, 12, 2, 11, 5, 9]
```

# Program #10

Write a program to input a list of names of students. Display the names of those students whose names end with ‘A’.

```
l=eval(input("Enter names of the students:"))
print("Names of students which ends with A :")
for i in range(len(l)):
    if l[i][-1] in "Aa":
        print(l[i])
```

```
Enter names of the students:
["Ria"  ","Akshay","Gurshan","Priya"]
Names of students which ends with A :
Ria
Priya
```

## Program #11

Write a program to input a list of numbers and re-positions all the elements of the list by shifting each of them one position ahead and by moving the last element to the first.

| | |
|---|---|
| For example, if the list contains | [25,30,90,11,16] |
| After changing the list content should be displayed as | [16,25,30,90,11] |

# Program #11

```
l=eval(input("Enter elements of the list:"))
print("Original List :",l)
length=len(l)
t=l[-1]
for i in range(length-2,-1,-1):
    l[i+1]=l[i]
l[0]=t
print("Updated List :",l)
```

**Output :**

```
Enter elements of the list:[25,30,90,11,16]

Original List : [25, 30, 90, 11, 16]

Updated List : [16, 25, 30, 90, 11]
```

## Program #12

Write a program to swap the even and odd positions of the values in the list Val.

Note : Assuming that the list has an even number of values in it.

For example :

| | |
|---|---|
| If the list Numbers contain | [25,17,19,13,12,15] |
| After swapping the list content should be displayed as | [17,25,13,19,15,12] |

## Program #12

```
l=eval(input("Enter elements of the list:"))
print("Original List :",l)
for i in range(0,len(l),2):
    l[i],l[i+1]=l[i+1],l[i]
print("Updated List :",l)
```

**Output :**

```
Enter elements of the list:[25,17,19,13,12,15]

Original List : [25, 17, 19, 13, 12, 15]

Updated List : [17, 25, 13, 19, 15, 12]
```

# Output Questions

Consider the lists A and B given below and write the output of the statements that follow:

A=[5, 3, 8]

B=[A, 8, 66, 45, 'A',['A', A], ["nested", "lists"]]

```
(i) print(A)

(ii) print(B)

(iii) print(A[0])

(iv) print(A[0:])

(v) print(A[:0])

(vi) print(B[0])
```

```
(vii) print(B[0][1])

(viii) print(B[0][1:])

(ix) print(B[4])

(x) print(B[5])

(xi) for i in A:
         print(B[i%6])
```

## Solutions

(i) [5, 3, 8]
(ii) [[5, 3, 8], 8, 66, 45, 'A', ['A', [5, 3, 8]], ['nested', 'lists']]
(iii) 5
(iv) [5, 3, 8]
(v) []
(vi) [5, 3, 8]

(vii) 3
(viii) [3, 8]
(ix) A
(x) ['A', [5, 3, 8]]
(xi) ['A', [5, 3, 8]]

```
45
66
```

## Output Questions

Find the output of the following code :

```
mylist=[10,20,30,40]
mylist[0]=0
for i in range(len(mylist)):
    mylist[i]*=2
mylist[3]+=100
print(mylist)
```

## Solutions

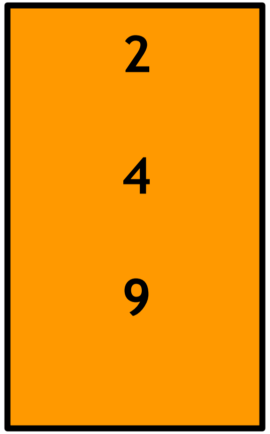
```
[0, 40, 60, 180]
```

## Output Questions

Find the output of the following code :

```
names1 = ['Shreya', 'Mohak', 'Mahesh', 'Dwivedi']
names2 = names1
names3 = names1[:]
names2[0] = 'Vishal'
names3[1] = 'Jay'
sum = 0
for ls in (names1,names2,names3):
    if ls[0] == 'Vishal':
        sum += 2
    if ls[1] == 'Jay':
        sum += 5
    print (sum)
```

## Solutions



## Output Questions

Find the output of the following code :

```
Data=["P",20,"R",10,"S",30]
Times=0
Alpha=""
Add=0
for c in range(1,6,2):
    Times=Times+c
    Alpha=Alpha+Data[c-1]+"$"
    Add=Add+Data[c]
    print(Times,Alpha,Add)
```

## Solutions

```
1 P$ 20
4 P$R$ 30
9 P$R$S$ 60
```

## Output Questions

Find the output of the following code :

```
arr = [1, 2, 3, 4, 5, 6]
for i in range(1, 6):
    arr[i - 1] = arr[i]

for i in range(0, 6):
    print(arr[i], end = " ")
```

## Solutions

```
2 3 4 5 6 6
```

# Output Questions

Find the output of the following code :

```
L=[]
for i in range(4):
    L.append(2*i+1)
print(L[::-1])
```

```
L1=[10,20,30,20,10]
L2=[]
for i in L1:
    if i not in L2:
        L2.append(i)
print(L1,L2,sep="&" )
```

## Solutions

[7,5,3,1]

[10,20,30,20,10] & [10,20,30]

# Practice Questions

Write single line statements for each of the following:

a. To declare a list named **SUBJECTS**, which stores the names of five subjects "English","Maths","Physics","Chemistry", "Computer Science".
b. To arrange the above list **SUBJECTS** in ascending order.
c. To add one more subject, "Physical Education" in the list **SUBJECTS**
d. To remove the subject "Maths" from the list **SUBJECTS**
e. To display the contents of list **SUBJECTS** in reverse order.
f. To remove the last subject from the list.
g. To add three subjects “Accounts”,”English”,”Painting” in one go.
h. To print the every alternate subject from the list.
i.

# Practice Programs

Q1. Write a program to input two different lists and then create a list that all the common elements from the two lists. For example :
If list1=[11, 22, 44, 55, 99, 66] and list2=[44, 22, 55, 12, 56] then

list3 is to be created as list3 = [22, 44, 55]

Q2. Write a program to input a list and then double the even elements of the list and triple the odd elements of the list.

Q3. Write a program to input a list containing names of cities and then display the names of all those cities that start with the alphabet ‘A’.
For example if the list contains [“AHMEDABAD”, CHENNAI”, “NEW DELHI”, “AMRITSAR”,“ AGRA”], then the program should print AHMEDABAD, AMRITSAR and AGRA.

Q4. Write a program to enter a list and then push all the zeros in the list to the end of the list.

For example: If the list contains [0, 2, 3, 4, 6, 7, 0, 1], then the program should re-arrange the elements as [2, 3, 4, 6, 7,1, 0, 0]