Programming using Python Lab (20AD53): Syllabus

Module 0: Introduction: Language basics and example problems (Two weeks)
Implement Python Script for checking if the given year is leap year or not.
Implement Python Script for finding the biggest number among 3 numbers.
Implement Python Script for displaying reversal of a number.
Implement Python Script to check if the given number is Armstrong or not.
Implement Python Script to print sum of N natural numbers.
Implement Python Script to check if a given number is palindrome or not.
Implement Python script to print the factorial of a number.
Implement Python Script to print all prime numbers within the given range.
Implement Python Script to calculate the series: S=1+x+x2+x3+.......xn
Implement Python Script to print the following pattern:

```
*
* *
* * *
```

Module 1: Exercise Programs on Lists.
Write a Python script to display elements of a list in reverse order.
Write a Python script to find the minimum and maximum elements without using built-in operations in the lists.
Write a Python script to remove duplicates from a list.
Write a Python script to append a list to the second list.
Write a Python script to count the number of strings in a list where the string length is 2 or More.

Module 2: Exercise Programs on Tuples.
Write a Python script to create a tuple with different data types.
Write a Python script to find the repeated items of a tuple.
Write a Python script to replace the last value of tuples in a list.
Sample list: [(10, 20, 40), (40, 50, 60), (70, 80, 90)]
Expected Output: [(10, 20, 100), (40, 50, 100), (70, 80, 100)]
Write python script to sort a tuple in different ways
Write a Python script to sort a tuple by its float element.
Sample data: [('item1', '12.20'), ('item2', '15.10'), ('item3', '24.5')]
Expected Output: [('item3', '24.5'), ('item2', '15.10'), ('item1', '12.20')]

Module 3: Exercise Programs on Sets.
Write a Python script to add member(s) in a set.
Write a Python script to perform Union, Intersection, difference and symmetric difference of given two sets.
Write a Python script to test whether every element in S is in T and every element in T is in S.

Module 4: Exercise Programs on Dictionaries
Write a Python script to sort (ascending and descending) a dictionary by value.
Write a Python script to check whether a given key already exists or not in a dictionary.

Write a Python script to concatenate the following dictionaries to create a new one.
Sample Dictionary : dic1={1:10, 2:20} dic2={3:30, 4:40} dic3={5:50,6:60}
Expected Result : {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
Write a Python script to print a dictionary where the keys are numbers between 1 and 15 (both included) and the values are squares of keys.
Write a Python program to map two lists into a dictionary.

Module 5: Exercise Programs on functions and recursion.
a) Define a function max_of_three() that takes three numbers as arguments and returns the largest of them.
b) Write a program which makes use of function to display all such numbers which are divisible
by 7 but are not a multiple of 5, between the given range X and Y.
c) Define functions to find mean, median, mode for the given numbers in a list.
d) Define a function which generates Fibonacci series up to n numbers.
e) Implement a python script for factorial of number by using recursion.
f) Implement a python script to find GCD of given two numbers using recursion.

Module 6: Exercise programs on Strings
a) Implement Python Script to perform various operations on string using string libraries.
b) Implement Python Script to check if the given string is palindrome or not.
c) Implement a python script to accept lines of text and find the number of characters, number of vowels and number of blank spaces in it.
d)Implement a python script that takes a list of words and returns the length of the longest one.

Module 7: Exercise programs on Regular Expressions
a) Write a Python script to check that a string contains only a certain set of characters (in this case a-z, A-Z and 0-9).
b) Write a Python script to check whether the password is valid or not.
Conditions for a valid password are:

- Should have at least one number.
- Should have at least one uppercase and one lowercase character.
- Should have at least one special symbol.
- Should be between 6 to 20 characters long.

Module 8 : Exercise programs on Matplotlib Library
a) Write a Python program to draw a line with suitable label in the x axis, y axis and a title.
b) Write a Python program to plot two or more lines with legends, different widths and colours.
c) Write a Python program to create multiple plots.
d) Write a Python programming to display a bar chart using different colour for each bar.
e) Write Python programming to create a pie chart with a title.
f) Write a Python program to draw a scatter plot with empty circles taking a random distribution in X and Y and plotted against each other.

****

Programming using Python Lab (20AD53): Algorithms & Programs

Algorithms: Module-0-Loops:

Algorithm for finding biggest number among 3 numbers:

1. Read a,b,c values from input function
2. If a>b and a>c, then print “a is bigger number”
3. Else If b>c and b>c, then print “b is bigger number”
4. Else , print “c is bigger number”

Algorithm to check the year is a leap year or not:

A year may be a leap year if it is evenly divisible by 4. Years that are divisible by 100 (century years such as 1900 or 2000) cannot be leap years unless they are also divisible by 400. (For this reason, the years 1700, 1800, and 1900 were not leap years, but the years 1600 and 2000 were.**)**

1. Read the year using input function
2. If the year%4==0 and year%100!=0, then print “the year is leap year”
3. Else If the year%400==0 and year%100=0, then print “the year is leap year”
4. If step-2 and step-3 fails, print “the year is not a leap year”

Algorithm to print sum of N natural numbers:

1. Read N
2. Initialise sum=0
3. Use loop to iterate the sum operation step-4
4. sum=sum+i
5. Print the sum value

Algorithm to print factorial of a number:

1. Read the number N
2. Initialise variable, fact=1
3. If N=0, print “factorial of 0 is 1”
4. Use loop to iterate the operation in step-5 for i from 1 to N
5. fact=fact*i
6. Print factorial value

Algorithm to print the number series 1+x+x2+x3+x4+....xn:

1. Read the x, n values of from input function
2. Initialise sum=1
3. Use for loop to iterate the operation in step-4 in the range(1,n+1)
4. sum=sum+x**i
5. Print sum value

Algorithm to print all the prime numbers with in a given range:

1. Read start, end values of a range from input function
2. Use for loop to check all the numbers within a range(start,end+1)
3. If number>1, then go to step-4
4. Use another for loop to perform the operation in step-5 from 2 to number

5. If number is divisible by any number between 2 to that number, then break the loop
6. Else, print the numbers

Algorithm to print the reversal of a number:

1. Initialise a variable to store reverse number to 0
2. Read the number from input function
3. Find the number of digits of the number using len(str(n))
4. Use for loop to perform the following operations from step-5 to 7 for all digits
5. Find the remainder of the number divided by 10
6. Multiply reverse number with 10 and add the reminder
7. Divide the number with 10 using floor division operator
8. Print the reverse number

Algorithm to check the number is Palindrome or not:

1. Initialise a variable to store reverse number to 0
2. Read the number from input function
3. Store the number in another temporary variable
4. Find the number of digits of the number using len(str(n))
5. Use for loop to perform the following operations from step-5 to 7 for all digits
6. Find the remainder of the number divided by 10
7. Multiply reverse number with 10 and add the reminder
8. Divide the number with 10 using floor division operator
9. If reverse_number==number, then print “ the number is palindrome”
10. Else, print “the number is not a palindrome”

Algorithm to check the number is Amstrong or not:

1. Initialise a variable to store sum to 0
2. Read the number from input function
3. Store the number in another temporary variable
4. Find the number of digits of the number using len(str(n))
5. Use for loop to perform the following operations from step-5 to 7 for all digits
6. Find the remainder of the number divided by 10
7. sum=sum+remainder**3
8. Divide the number with 10 using floor division operator
9. If sum==number, then print “ the number is Amstrong number”
10. Else, print “the number is not Amstrong”

Algorithm to print the following pattern:

```
"""
*
* *
* * *
* * * *
"""
```

1. Read the number of rows required to print the given * pattern
2. Use for loop to iterate all the rows
3. Use another for loop to iterate upto the row number value at present iteration
4. Print “*” without new line
5. Print empty print statement for new line after each row

List of python programs under module-0: Loops

# finding biggest number among 3 numbers
```
a=int(input("enter a value: "))
b=int(input("enter b value: "))
c=int(input("enter c value: "))
if a>b and a>c:
        print("a is bigger number")
elif b>c:
        print("b is bigger number")
else:
        print("c is bigger number")
```

# print sum of N natural numbers
```
n=int(input("enter n value: "))
i=1
sum=0
while(i<=10):
        sum=sum+i
        i=i+1
print(sum)
```

# to check the year is a leap year or not
```
y=int(input("enter the year"))
if y%4==0 and y%100!=0:
        print("the year",y," is a leap year")
elif y%100==0 and y%400==0:
        print("the year",y," is a leap year")
else:
        print("the year",y," is not a leap year")
```

# print factorial of a number
```
n=int(input("enter n value"))
temp=n
fact=n
if n==0:
        print("factorial of given number 0 is: ",1)
i=1
while (n>1):
        fact=fact*(n-1)
        n=n-1
print("factorial of a number",temp, "is: ",fact)
```

```
# print the number series 1+x+x2+x3+x4+....xn
n=int(input("enter n"))
x=int(input("enter x"))
sum=1
for i in range(1,n+1):
        sum=sum+x**i
print("sum", sum)
--------------------------------------------------

# To find the reversal of a number
rev=0
n=int(input("enter n"))
temp=n
dig=len(str(n))
for i in range(dig):
        r=n%10
        rev=r+rev*10
        n=n//10
print("Reverse number is", rev)
--------------------------------------------

# To check the  number is Palindrome or not
rev=0
n=int(input("enter n"))
temp=n
dig=len(str(n))
for i in range(dig):
        r=n%10
        rev=r+rev*10
        n=n//10
if temp==rev:
        print("Palindrome")
else:
        print("not Palindrome")
--------------------------------------------
# To check the number is Amstrong or not
rev=0
n=int(input("enter n"))
temp=n
dig=len(str(n))
for i in range(dig):
        r=n%10
        rev=rev+r**3
        n=n//10
if temp==rev:
        print("Amstrong")
else:
        print("not Amstrong")
```

#print the following pattern:

```
for i in range(1,5):
        for j in range(1,i):
        print("*", end=" ")
        print("")
```

#print the following pattern:

```
"""
*
* *
* * *
* * * *
* * * * *
"""
for i in range(5):
        for j in range(i+1):
        print("*", end=" ")
        print("")
```

# print the following pattern:

```
"""
* * * * *
* * * *
* * *
* *
*
"""
for m in range(5,0,-1):
        for j in range(0,m):
        print("*", end=" ")
        print("")
```

# print the following pattern:

```
"""
0 1 2 3 4
0 1 2 3
0 1 2
0 1
0
"""
for m in range(5,0,-1):
        for j in range(0,m):
        print(j, end=" ")
        print("")
```

Algorithms: Module-1-Lists:

Algorithm to display elements of list in reverse order

1. Initialise a list contains different data types
2. Use reverse() function to reverse the order
3. Print the list

Algorithm to find the minimum and maximum elements without using built-in operations in the lists

1. Initialise the list of integers
2. Initialise maximum and minimum values with first value of start
3. Use for loop to iterate all the list_values in the list
4. If list_value>maximum, then maximum=list_value
5. Use for loop to iterate all the list_values in the list
6. If list_value<minimum, then minimum=list_value
7. Print the maximum and minimum values

Algorithm to remove duplicates from a list

1. Initialise the list of values using input function and loop
2. Declare a new list to store the output list
3. Use for loop to iterate all values in the list
4. If value is not in new list, append that value to new list
5. Print the new list

Algorithm to append a list to the second list

1. Initialise the first list of values using input function and loop
2. Initialise the second list of values using input function and loop
3. Add the two lists using extend() method **or** using + operator
4. Print the final list after adding

Algorithm to count the number of strings in a list where the string length is 2 or More.

1. Initialise the first list of strings using input function and loop
2. Declare variable to store the count
3. Use a for loop to iterate all the values in the list
4. If length of each string>=2, the increment the count
5. Print the final count

<u>List of python programs under module-1: Lists</u>

# To display elements of list in reverse order

```
x=[]
n=int(input("Enter no.of elements required for list:"))
for i in range(0,n):
        a=int(input("Enter a element to the list:"))
        x.append(a)
print("Original input list is:",x)
# print("List in reverse order is:",x[::-1]) (OR)
x.reverse()
print("List in reverse order is:",x)
--------------------------------------------------
```

# To find the minimum and maximum elements without using built-in operations in the lists

```
x=[]
n=int(input("Enter no.of elements required for list:"))
for i in range(0,n):
        a=int(input("Enter a element to the list:"))
        x.append(a)
print("Given list is:",x)
max=x[0]
min=x[0]
for i in x:
        if i>max:
        max=i
        if i<min:
        min=i
print("the maximum element of list is:",max)
print("the minimum element of list is:",min)
------------------------------------------------
```

# To remove duplicates from a list and print the list

```
'''
x=[]
n=int(input("Enter no.of elements required for list:"))
for i in range(0,n):
        a=int(input("Enter a element to the list:"))
        x.append(a)
print("Given list is:",x)
'''
x=[1,2,3,4,5,4,4,6,7,8,8,9]
print("Given list is:",x)
y=[]
```

```
for i in x:
        if i not in y:
        y.append(i)
print("the List after removing duplicates is:",y)
----------------------------------------------
```

# To append a list to the second list

```
x=[]
n=int(input("Enter no.of elements required for First list:"))
for i in range(0,n):
        a=int(input("Enter a element to First list:"))
        x.append(a)
print("First list is:",x)
y=[]
n=int(input("Enter no.of elements required for Second list:"))
for i in range(0,n):
        a=int(input("Enter a element to Second list:"))
        y.append(a)
print("Second list is:",y)
# print("The output list after appending the two lists is:",x+y)
x.extend(y)
print("The output list after appending the two lists is:",x)
-------------------------------------------------
```

# To count the number of strings in a list where the string length is 2 or More.

```
x=[]
n=int(input("Enter no.of strings required for list:"))
for i in range(0,n):
        a=input("Enter a string to the list:")
        x.append(a)
print("Given list is:",x)
count=0
for i in x:
        if len(i)>=2:
        count=count+1
print("Number of strings with length 2 or more in the list is:",count)
```

Algorithms: Module-2-Tuples:

Algorithm to create a tuple with different data types

1. Create a tuple with and without using parentheses () using different data types, lists, tuples, nested tuples and tuple with single value..etc.
2. Print the tuple in each case

Algorithm to find the repeated items of a tuple.

Method 1: (To print the count of repeated item)

1. Create a user defined list with integers
2. Print the original list
3. Use count() function to find how many times a value is repeated
4. Print the count

Method 2: (To identify the repeated items in a tuple)

1. Create a list of items with two or more repeated items
2. Print the input tuple as x
3. Create two empty tuples: y, z
4. Using for loop, for every element in x: if x.count(i)>1 then add that element to y
5. Using for loop, for every element in y: if i not in z, then add that element to z
6. Print the repeated items from the input tuple as z

Algorithm to replace the last value of tuples in a list.

1. Create a list which contains 3 or more tuples with values
2. Print the input list as list1: list1=[(10, 20, 40), (40, 50, 60), (70, 80, 90)]
3. Create an empty list for output as: list2
4. For every element in list1: perform tuple addition as a=i[:-1]+(100,)
5. Append this ‘a’ value to output list
6. Print the output list after the loop as : list2

Algorithm to sort a tuple in different ways:

Method-1: (sorting a tuple in ascending order)

1. initialise a user defined tuple using input function and loop
2. print the tuple
3. use the function sorted( ) to get result
4. use the function tuple( ) to convert the list into a tuple
5. print the final result as tuple

Method-2 (sorting a tuple in descending order)

1. initialise a user defined tuple using input function and loop
2. print the tuple
3. use the function sorted(given tuple name,reverse=True)
4. use the function tuple( ) to convert the output list into a tuple
5. print the final result as tuple

Method-3 (sorting a tuple of strings according to their length)

1. initialise a user defined tuple of strings using input function and loop
2. print the tuple
3. use the function sorted(given tuple name,key=len)
4. use the function tuple( ) to convert the output list into a tuple
5. print the final result as tuple

Algorithm to sort a list of tuples by their float element:

1. create a list which contains 3 tuples as shown below
   price=[('item1','12.20'),('item2','15.10'),('item3','21.5')]
2. Print the input list
3. print the result of sorted list using lambda function i.e, print(sorted (price, key=lambdax:float(x[1]),reverse=True))

List of python programs under module-2: Tuples

#create and print a tuple with different data types in different ways

```
tup1=(1, 2, 3.5, 5.8, "LBRCE","ECE",False)
#  (OR)
#  tup1=1, 2, 3.5, 5.8, "LBRCE","ECE",False
print("The tuple with different data types is:",tup1)
tup2=(2, 5.8, "LBRCE", [1,2,3], (4,5,6))
print("The tuple with different data types, lists & tuples is:",tup2)
tup3=(100,)
print("The tuple with single element is:",tup3)
```

# To find find the repeated items of a tuple

```
'''# Use defined Tuple
x=( )
n=int(input("Enter no.of values required for tuple:"))
for i in range(0,n):
        a=int(input("Enter a element to the tuple:"))
        x=x+(a,)
print("The given input tuple is",x)
'''

# Method-1 to display the count of repetition
x=(1,2,4,3,5,4,6,8,4)
print("The given input tuple with repeated items is",x)
c=x.count(4)
print("Number of times the value 4 is repeated is:",c)

# Method-2 to identify the repeated items in a tuple
x=(1,2,4,3,5,4,6,8,4,9,5,6,5,1,1,2,3,4,5,4,5)
y=( )
z=( )
print("The given input tuple with repeated items is",x)
for i in x:
        if x.count(i)>1:
        y=y+(i,)
#print("tuple with only repetitions is:",y)
for i in y:
        if i not in z:
        z=z+(i,)
print("The items which are repeated in the tuple are:",z)
```

—---

# To replace the last value of tuple in a list of tuple

```
list1=[(10, 20, 40), (40, 50, 60), (70, 80, 90)]
print("The input list of tuples is:",list1)
list2=[ ]
for i in list1:
        a=i[:-1]+(100,)
        list2.append(a)
print("The output list after replacing last value of each tuple:",list2)
```
—---

# To sort a tuple in different ways

```
n=int(input("Enter number of elements required for tuple:"))
for i in range(n):
        a=int(input("Enter a element to the tuple:"))
        tup=tup+(a,)
print("The input tuple is:",tup)
# METHOD-1
print("The given tuple sorted in ascending order is:")
print(sorted(tup))
# METHOD-2
print("The given tuple sorted in descending order is:")
print(sorted(tup,reverse=True))
# METHOD-3
tup2=()
n=int(input("Enter number of strings required for tuple:"))
for i in range(n):
        a=input("Enter a string to the tuple:")
        tup2=tup2+(a,)
print("The input tuple of strings is:",tup2)
print("The given tuple of strings sorted according to their length is:")
print(sorted(tup2, key=len))
```
—---

# Sorting a list of tuples according to their float element

```
price=[('item1','12.20'),('item2','15.10'),('item3','21.5')]
print("the input list of tuples is:", price)
print("the output list of tuples sorted according to floating element is:")
print(sorted (price, key=lambda x:float(x[1]),reverse=True))
```
—---

Algorithms: Module-3-Sets:

Algorithm to add member(s) in a set:

1. Create a set using curly braces { } or create a user defined set
2. Print the given input set
3. Add an element to the given set using add( ) method
4. Print the set after adding
5. Add some list of elements to set using update( ) method
6. Print the set after updating

Algorithm to perform Union, Intersection, difference and symmetric difference of given two sets:

1. Create two sets using curly braces { } with some common elements as set1, set2
2. Print the Union of two sets using the operation: set1|set2
3. Print the Intersection of two sets using the operation: set1&set2
4. Print Set Difference of two sets using the operation: set1-set2
5. Print the Symmetric Difference of two sets using the operation: set1^set2)

Algorithm to test whether every element in S is in T and every element in T is in S:

1. Create a user defined input set as S
2. Print the input set S
3. Create a user defined input set as T
4. Print the input set T
5. Check whether all elements of S in T (True or False?) using the operation: S<=T
6. Check whether all elements of T in S (True or False?) using the operation: T<=S
7. Check whether S is equivalent to T (True or False ?) using the operation: S==T

List of python programs under module-3: Sets

#To add member(s) in a set.

```
'''# Use defined Set
set1=set( )
n=int(input("enter number of elements required for set:"))
for i in range(0,n):
        a=int(input("enter a element into set:"))
        set1.add(a)
print("the user defined set is:",set1)
'''
set1={1,2,3,4,5,6}
print("Input set is:",set1)
set1.add(7)
#set1.add("Hi")
print("set after adding one element:",set1)
set1.update([8,9,10])
print("set after adding list of 3 elements:",set1)
```

#To perform Union, Intersection, difference and symmetric difference of given two sets.

```
set1={1,2,3,4,5,6,"hi"}
set2={5,6,7,8,9,"hello"}
print("Union of two sets:",set1|set2)
print("Intersection of two sets:",set1&set2)
print("Set Difference of two sets:",set1-set2)
print("Symmetric Difference of two sets:",set1^set2)
```

#To test whether every element in S is in T and every element in T is in S.

```
S=set( )
n=int(input("enter number of elements required for set S:"))
for i in range(0,n):
        a=int(input("enter a element into set S:"))
        S.add(a)
print("the input set S is:",S)
T=set( )
n=int(input("enter number of elements required for set T:"))
for i in range(0,n):
        a=int(input("enter a element into set T:"))
        S.add(a)
print("the input set T is:",T)
print("Is all elements of S in T ? (True/False)?:",S<=T)  # subset operation
print("Is all elements of T in S ? (True/False)?:",T<=S)
print("Is T equivalent to S ? (True/False)?:",T==S)
```

—--

Algorithms:Module-4: Dictionaries

Algorithm to sort (ascending and descending) a dictionary by value

1. Create a user defined dictionary and print the given input
2. Print the keys, items and values in a sorted order
3. Print the result in ascending order using lambda function (sorted(variable name.items(),key=lambda:x[1]))
4. Print the result in descending order using lambda function(sorted(variable name.items(),key=lambda:x[1], reverse=True))

Algorithm to check whether a given key already exists or not in a dictionary.

1. Create a user defined dictionary and print the given input
2. Initialise a variable to search a key value in dictionary
3. Use if else condition i.e; if it is in that variable print it is present in dictionary otherwise,doesn't present in that dictionary

Algorithm to concatenate the following dictionaries to create a new one.

method-1:

1. Initialise 3 dictionaries as follows dic1{1:10,2:20}dic2{3:30,4:40}dic3{5:50,6:60}
2. Update dic2 and dic3 into dic1 and print dic1

method-2:

1. Initialise 3 dictionaries as follows dic1{1:10,2:20}dic2{3:30,4:40}dic3{5:50,6:60} and initialise an empty dictionary as dic4
2. Use for loop in the range of dic1,2,3
3. Update this range of i in dic4 and print dic4

Algorithm to print a dictionary where the keys are numbers between 1 and 15 (both included) and the values are squares of keys.

1. Initialise an empty dictionary
2. Use for loop in range(1,16)
3. Use the function variable name[i]=squares of i values
4. Print the result

Algorithm to map two lists into a dictionary

1. Create a user defined list-1&2 and print the given input
2. Initialise an empty dictionary and initialise another variable and store the length of list1
3. Use for loop and take a variable and store the list values as{keys:values}
4. Update that variable into the empty dictionary and print that dictionary

List of python programs under module-4: Dictionaries

# To sort (ascending and descending) a dictionary by value.

```
employee={'salary':2500,'id number':405,'team no':2,'experience':3}
print("the given input dictionary is:")
print(employee)
'''
print("\nthe list of keys of dictionary in sorted order :")
print(sorted(employee.keys()))
print("\nthe list of values of dictionary in sorted order:")
print(sorted(employee.values()))
print("\nthe copy of sorted dictionary printed as a list of tuples:")
print(sorted(employee.items()))
'''
print("\nthe dictionary after sorting by values in ascending order:")
asc=sorted(employee.items(),key=lambda x:x[1])
d={ }
for i,j in asc:
    d[i]=j
print(d)

print("\nthe dictionary after sorting by values in descending order:")
desc=sorted(employee.items(),key=lambda x:x[1],reverse=True)
d={}
for i,j in desc:
    d[i]=j
print(d)
```

# To check whether a given key already exists or not in a dictionary.

```
'''
# User defined dictionary
dic={}
n=int(input("enter no of items required for dictionary"))
for i in range(n):
        a=input("enter a key to dict")
        b=int(input("enter a value to the key"))
        dic[a]=b
        ''' (OR)
        d={a:b}
        dic.update(d)
        '''
print(dic)
'''
```

```
week={'mon':1,'tue':2,'wed':3,'thu':4,'fri':5,'sat':1,'sun':1}
print("the given input dictionary is:")
print(week)
k=input("enter a key value to search in dictionary")
if k in week:
        print(k,"is present in the dictionary")
else:
        print(k,"is not present in the dictionary")
```

# To concatenate the following 3 dictionaries to create a new one.

# Method-1: without using for loop

```
dic1={1:10, 2:20}
dic2={3:30, 4:40}
dic3={5:50,6:60}
dic1.update(dic2)
dic1.update(dic3)
print(dic1)
```

# Method 2: Using for loop (in case of more no.of dictionaries)

```
dic1={1:10, 2:20}
dic2={3:30, 4:40}
dic3={5:50,6:60}
dic4={}
for i in [dic1,dic2,dic3]:
        dic4.update(i)
print(dic4)
```

# To print a dictionary where the keys are numbers between 1 and 15 (both included) and the values are squares of keys.

```
dic={}
for i in range(1,16):
        dic[i]=i*i
print("the dictionary with square values between 1 and 15 is:\n",dic)
```

# To map two lists into a dictionary

```
list1=[1,2,3,4,5]
list2=[3,6,9,12,15]
dic={}
n=len(list1)
for i in range(n):
        d={list1[i]:list2[i]}
        dic.update(d)
print(dic)
```

Algorithms: Module-5-Functions and Recursion

**Algorithm for a function max_of_three() that takes three numbers as arguments and returns the
largest of them.**

1.Initialise 3 numbers using input function
2.Use the function def max_of_three(a,b,c):
3.Compare that numbers using if,elif,and else statements and return biggest
4.Print the biggest number among three numbers using function call max_of_three(x,y,z)

Algorithm for a function to display all such numbers which are divisible by 7 but are not a multiple of 5, between the given range X and Y.

1.Read the starting & ending ranges using input function
2.Initialise the numbers using def numbers(a,b) and use a for loop in range of (a,b+1)
3.use if statement, the number which are divisible by 7 and not divisible by 5 store that values in a variable
4.Print the final output

Algorithm for the functions to find mean, median, mode for the given numbers in a list.

1.initialise a variable to take number of elements as input
2.Initialise another variable to enter elements and use for loop in the range of (0,n+1)
3.Use the function def mean
 Mean=(sum of given variables)/(total number of variables)
4.Use the function def median and consider the values in sorted order
5.If the number of elements are divisible by 2 then consider the mean of that two middle numbers else consider the median as middle number
6.Use another function def mode
Mode= the value that appears most often in a given data
7.print the mean,median and mode values

Algorithm for a function which generates Fibonacci series up to n numbers.

1.Read a number using input function
2.Use the function def fib(n)
3.If n==1 print 0 else print a and b values
4.Use for loop in the range (2,n) use the formula for calculating Fibonacci series
f=fn-1+fn-2 and print the final result

Algorithm for factorial of number by using recursion.

1.Read a number using input function
2.Use the recursion def recur_factorial(n)
3.If n==1 return to n value else use the formula for calculating the factorial
n!=nx(n-1)!
4.If n<0 print ("factorial doesn't possible for negative numbers")
5.elif n==0 print ("the factorial of 0 is 1")
6.else print the obtained output

Algorithm to find GCD of given two numbers using recursion.

1.Read two numbers using input function
2.use the recursion def gcd_fun(x,y)
3.Use it statement, if (y==0) return to x else
GCD(a,b)= (axb)/LCM (a,b)
4.Print the final output

List of python programs under module-5:

Functions and recursions

a) Define a function max_of_three() that takes three numbers as arguments and returns the largest of them.

```
 n= [int(input("Enter Number %d : " %(i+1))) for i in range(3)]
def max_of_three(a,b,c):
    if  (a>b and a>c):
        return a
    elif  (b>c):
        return b
    else:
        return c
 print("Maximum Number is: ", max_of_three(n[0], n[1], n[2]))
```

b) Write a program which makes use of function to display all such numbers which are divisible by 7 but are not a multiple of 5, between given range X and Y.

```
def numbers(a,b):
    X= [i for i in range(a,b+1) if(i%7 == 0 and i%5 != 0)]
    print("Numbers divisible by 7, but not by 5 between %d and %d are:" %(a,b))
    print(X)
start_range = int(input("Enter the Starting Range: "))
end_range = int(input("Enter the Ending Range: "))
numbers(start_range, end_range)
```

c) Define functions to find mean, median, mode for the given numbers in a list.

```
def mean(n, a):
    return sum(a)/n
def median(n,a):
    a.sort()
    if(n%2 == 0):
        return (a[(n-1)//2] + a[n//2])/2
    else:
        return a[n//2]
def mode(a):
     c= [a.count(i) for i in a]
     b= [i for i in range(len(c)) if(c[i]==max(c))]
     return b[0]
n= int(input(”How many elements you want to enter into List? "))
a= [int(input("Enter Element %d: " %(i+1))) for i in range(n)]
print("Mean =%.2f" %(mean(n,a)))
print("Median = %.2f" %(median(n,a)))
print("Mode = %d" %(a[(mode(a))]))
```

d) Define a function which generates Fibonacci series up to n numbers.

```
def fib(n):
    a=0
    b=1
    if n==1:
        print(a)
    else:
        print(a)
        print(b)
        for i in range(2,n):
            c=a+b
            a=b
            b=c
            print(c)
num = int(input("Number that you want to Print Fibonacci Sequence:"))
fib(num)
```

e) Implement a python script for factorial of number by using recursion.

```
def recur_factorial(n):
   if (n == 1):
       return n
   else:
       return n*recur_factorial(n-1)
num = int(input("Enter a number:"))
if num <0:
   print("Sorry, factorial does not exist for negative numbers")
elif num == 0:
   print("The factorial of 0 is 1")
else:
   print("The factorial of",num,"is",recur_factorial(num))
```

f) Implement a python script to find GCD of given two numbers using recursion.

```
def gcd_fun(x, y):
   if (y == 0):
       return x
   else:
       return gcd_fun(y, x % y)
x=int(input("Enter the first number: "))
y=int(input("Enter the second number: "))
num = gcd_fun(x, y)
```

```
print("GCD of two numbers is", num)
```

Algorithms:Module-6-Strings

Algorithm to implement Python Script to perform various operations on string using string methods.

1. initialise two strings using input function
2. perform various operations on strings using string   operations
3. print the output in each case

Algorithm to implement Python Script to check if a given string is palindrome or not.

1. initialise a string using input function
2. Initialise another variable to store the reverse value
3. Reverse the given string using reverse () or [::-1] operation
4. Use if statement, if the reverse string==given string print the given string as palindrome else print that given string is not a palindrome

Algorithm to implement a python script to accept lines of text and find the number of characters, number of vowels and number of blank spaces in it.

1. Initialise a string using input function
2. initialise 3 variables as vowels, consonants, blanks and equal them to 0 and print the length of given string
3. use for loop in string, if the letters in string equal to a,e,i,o,u or A,E,I,O,U increment the letter by 1 at last print the count of vowels else print the count of consonants
4. if(i==' ') increment the value by one and print the number of blanks

Algorithm to implement a python script that takes a list of words and returns the length of the longest one.

1. create a user defined list using input function and loop and print the given list
2. initialise a variable and store the max length of given strings in that
3. print the longest word and the length of that longest word

List of python programs under module-6:
Strings

a) Implement Python Script to perform various operations on string using string methods.

```
name1 = "Ram Kumar"

name2 = "Ram_Kumar”

print(name1.split())

print(name2.split('_'))

print(name1.isalnum())

print(name1.isdigit())

print(name2.lower())

print(name2.upper())

print(name1.capitalize())

print(len(name1))

print(name1.title())
```

b) Implement Python Script to check if a given string is palindrome or not.

```
my_str = input("Enter any string: ")
my_str = my_str.casefold()
rev_str = reversed(my_str)
#rev_str=my_str[::-1]
if list(my_str) == list(rev_str):
   print("The string is a palindrome.”)
else:
   print("The string is not a palindrome.")
```

c) Implement a python script to accept lines of text and find the number of characters, number of vowels and number of blank spaces in it.

```
str1 = input("Please Enter Your Own String : ")
vowels = 0
consonants = 0
blanks=0
print("No.of characters in the string is=",len(str1))
for i in str1:
        if(i == 'a' or i == 'e' or i == 'i' or i == 'o' or i == 'u' or i == 'A' or i == 'E' or i == 'I' or i ==
'O' or i == 'U'):
        vowels = vowels + 1
        else:
        consonants = consonants + 1
        if(i==' '):
        blanks=blanks+1
print("Total Number of Vowels in this String = ", vowels)
print("Total Number of Consonants in this String = ", consonants)
print("Total Number of Blank spaces in this String = ", blanks)
```

d) Implement a python script that takes a list of words and returns the length of the longest one.

def find_longest(word_list):	**word_list = [ ]**	def find_longest_word(words_list):
	n=int(input("enter number of strings	word_len = [ ]
m=max(word_list,key=len)	**required:"))**	for n in words_list:
for i in word_list:	**for i in range(n):**	word_len.append((len(n), n))
if len(i)==len(m):	**a=input("enter a string:")**	word_len.sort( )
print("longest word	**word_list.append(a)**	return word_len[-1][0],
in the list is=",i)	**print("given input list of**	word_len[-1][1]
print("length of the	**words:",word_list)**	result = find_longest_word(
longest word =",len(i))	**m=max(word_list,key=len)**	["PHP", "Exercises", "Backend"])
find_longest(['lbrce','college	**print("longest word in the list is=",m)**	print("\nLongest word: ",result[1])
','ece','excellent'])	**print("length of the longest word**	print("Length of the longest word:
	=",len(m))	",result[0])

Algorithms:Module-7-Regular Expressions

Algorithm to implement a Python script to check that a string contains only a certain set of characters (in this case a-z, A-Z and 0-9).

1. Initialise a string using input function
2. import required libraries based on our requirement
3. Def is_allowed_specific_char(string) and compose all the values between (a-z, A-Z and 0-9) and store that elements in a variable search in that variable whether it has only a-z,A-Z or 0-9 numbers return to given string
4. print the output using function call

Algorithm to implement a Write a Python script to check whether the password is valid or not.

Conditions for a valid password are:

1.Should have at least one number.

2.Should have at least one uppercase and one lowercase character.

3. Should have at least one special symbol.

4. Should be between 6 to 20 characters long.

1. enter a password using input function
2. import required libraries based on our requirement
3. assume x=true use while loop in x
4. If the length is between 6 to 12 break the loop a-z are not in password break the loop,0-9 are not in that password break the loop,A-Z are not in password break the loop,if there are not special characters break the loop and if there are any spaces then break the loop again
5. Else print your password is valid assume x=False in that case print the password is not valid

List of python programs under module-7:

Regular Expressions

a) Write a Python script to check that a string contains only a certain set of characters (in this case a-z, A-Z and 0-9).

```
import re
def is_allowed_specific_char(string):
    charRe = re.compile(r'[^a-zA-Z0-9.]')
    string = charRe.search(string)
    return not bool(string)
string = input("Enter a String to verify: ")
print(is_allowed_specific_char(string))
```

b) Write a Python script to check whether the password is valid or not.
Conditions for a valid password are:

1. **Should have at least one number.**
2. **Should have at least one uppercase and one lowercase character.**
3. **Should have at least one special symbol.**
4. **Should be between 6 to 20 characters long.**

```
import re
password = input("Enter a Password: ”)
x = True
while x:
    if (len(password)<6 or len(password) >12):
        break
    elif not re.search("[a-z]", password):
        break
    elif not re.search("[0-9]", password):
        break
    elif not re.search("[A-Z]", password):
        break
    elif not re.search("[$#@]", password):
        break
    elif re.search("\s", password):
        break
    else:
        print("Your Password is Valid")
        x=False
        break
if x:
    print("Your Password is INVALID")
```

Algorithms:Module-8-Matplotlib library

Algorithm to implement a Python program to draw a line with a suitable label in the x axis, y axis and a title.

1. Import required libraries for program execution
2. Range of(1,50) are Stored in X
3. Thrice of X values are stored in Y
4. Print X,Y and plot X,Y
5. mention x-label as x-axis and y-label as y-axis and give title for that plotting
6. display the image using the statement plt.show( )

Algorithm to implement a Python program to plot two or more lines with legends, different widths and colours.

1. Import required libraries for program execution
2. Initialise x,y points for line-1 and line-2
3. mention x-label as x-axis and y-label as y-axis and give title for that plotting
4. plot x1,y1 points and consider colour, line width and label for line-1
5. plot x2,y2 points and consider colour,line width and label for line-2
6. Show the legends using statement plt.legend( )
7. display the figure using the statement plt.show()

Algorithm to implement a Python program to create multiple plots.

1. Import required libraries for program execution
2. arrange the numbers in the range of 0 to 10 and it's step size is 0.01
3. Plot the sine wave using function plt.plot(x, np.sin(x))
4. plot the cosine wave using function plt.plot(x, np.cos(x))
5. plot the another wave using function plt.plot(x, x)
6. display all the images on the same window

Algorithm to implement a Python program to display a bar chart using different color for each bar.

1. Import required libraries for program execution
2. Initialise two variables and store values and strings respectively
3. keep them on x and y axis give title for the graph
4. Plot the xlabel and ylabel on bar graph using different colours and display the graph

Algorithm to implement a Python program to create a pie chart with a title.

1. Import required libraries for program execution
2. initialise two variables and store values and strings respectively
3. Keep them on x and y axis give title for the graph
4. Plot the xlabel and ylabel on pie chart using percentages and display the graph

Algorithm to implement a Python program to draw a scatter plot with empty circles taking a random distribution in X and Y and plotted against each other.

1. Import required libraries for program execution
2. initialise two variables for in the range (50) as x and y
3. plot them using scatter plot as plt.scatter(x, y, s=70, facecolors='none', edgecolors='g')
4. Mention x and y label names and give title and display the graph

List of python programs under module-8:

Matplotlib Library

Online Editor Python with libraries: https://trinket.io/embed/python3/a5bd54189b

a) **Write a Python program to draw a line with a suitable label in the x axis, y axis and a title.**

```
import matplotlib.pyplot as plt
X = range(1, 50)
Y = [value * 3 for value in X]
print("Values of X:")
print(*range(1,50))
print("Values of Y (thrice of X):")
print(Y)
# Plot lines and/or markers to the Axes.
plt.plot(X, Y)
# Set the x axis label of the current axis.
plt.xlabel('x - axis')
# Set the y axis label of the current axis.
plt.ylabel('y - axis')
# Set a title
plt.title('Draw a line.')
# Display the figure.
plt.show( )
```

b) **Write a Python program to plot two or more lines with legends, different widths and colours.**

```
import matplotlib.pyplot as plt
```

```
# line 1 points
x1 = [10,20,30]
y1 = [20,40,10]
# line 2 points
x2 = [10,20,30]
y2 = [40,10,30]
# Set the x axis label of the current axis.
plt.xlabel('x - axis')
# Set the y axis label of the current axis.
plt.ylabel('y - axis')
# Set a title
plt.title('Two or more lines with different widths and colors with suitable legends ')
# Display the figure.
plt.plot(x1,y1, color='blue', linewidth = 3,  label = 'line1-width-3')
plt.plot(x2,y2, color='red', linewidth = 5,  label = 'line2-width-5')
# show a legend on the plot
plt.legend( )
plt.show( )
```

c) Write a Python program to create multiple plots.

```
import matplotlib.pyplot as plt
import numpy as np
x = np.arange(0, 10, 0.01)
# notice that nrows = number of images
plt.subplot(3, 1, 1)
plt.plot(x, np.sin(x))
# ncols stays as 1
plt.subplot(3, 1, 2)
plt.plot(x, np.cos(x))
# each image has a unique index
plt.subplot(3, 1, 3)
plt.plot(x, x)
plt.show( )
```

d) Write a Python program to display a bar chart using different color for each bar.

```
import matplotlib.pyplot as plt

marks = [79, 82, 75, 96]

subjects = ['C', 'JAVA', 'Python', 'WT']

plt.xlabel("Subjects")

plt.ylabel("Obtained Marks")

plt.title("Student Marks")

plt.bar(subjects, marks, color=['r', 'g', 'b', 'm'])

plt.show( )
```

e) Write a Python program to create a pie chart with a title.

```
import matplotlib.pyplot as plt
marks = [79, 82, 75, 96]
subjects = ['C', 'JAVA', 'Python', 'WT']
plt.pie(marks, labels=subjects, autopct = '%1.1f%%')
plt.title("Student Marks")
plt.legend( )
plt.show( )
```

f) Write a Python program to draw a scatter plot with empty circles taking a random distribution in X and Y and plotted against each other.

```
import matplotlib.pyplot as plt
import numpy as np
x = np.random.randn(50)
y = np.random.randn(50)
plt.scatter(x, y, s=70, facecolors='none', edgecolors='g')
plt.xlabel("X")
plt.ylabel("Y")
plt.show()
```

Addition python program beyond the syllabus:

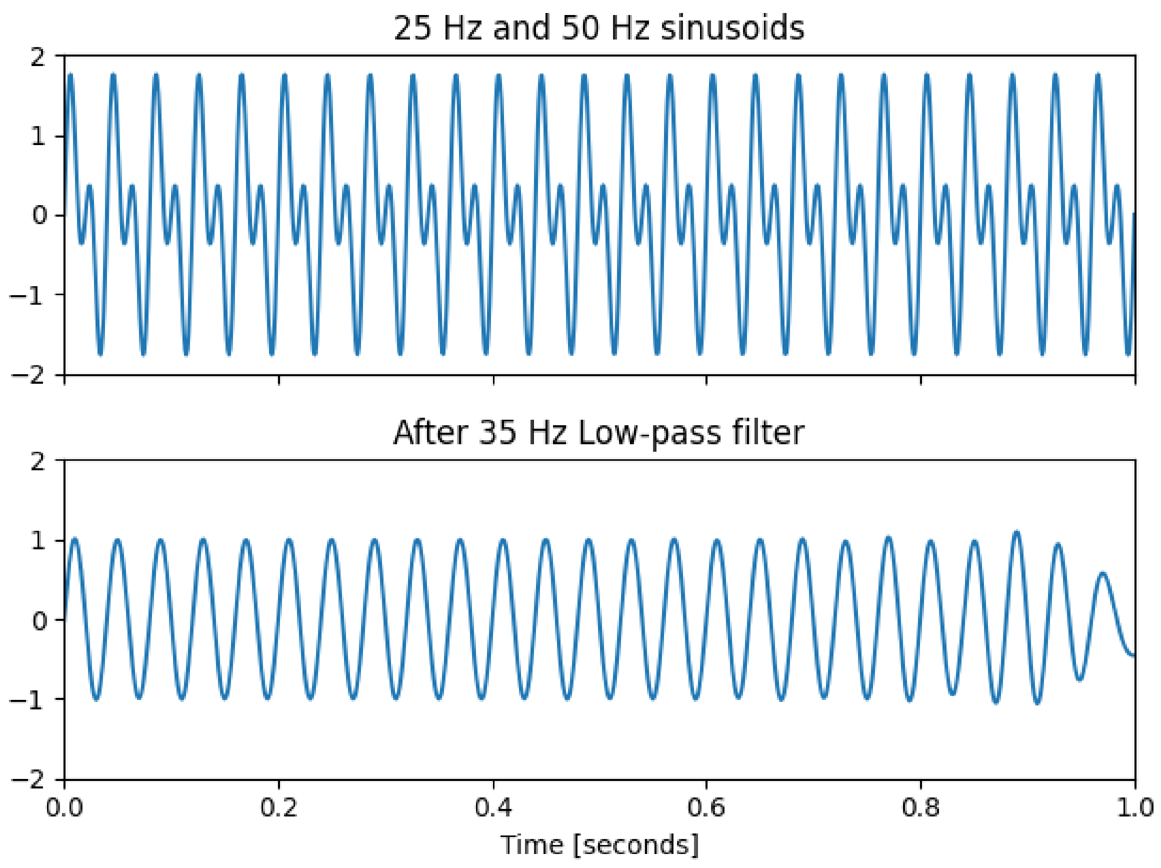
1. Implement python script to remove noise from a signal

Program(method-1):

```
# Removing noise using simple butterworth low pass filter

import numpy as np
import scipy.signal as signal
import matplotlib.pyplot as plt
# Specifications of the filter
f1 = 25  # Frequency of 1st signal
f2 = 50  # Frequency of 2nd signal
N = 10  # Order of the filter


# Generate the time vector of 1 sec duration
t = np.linspace(0, 1, 1000)  # Generate 1000 samples in 1 sec

# Generate the signal containing f1 and f2
sig = np.sin(2*np.pi*f1*t) + np.sin(2*np.pi*f2*t)
# Display the signal
fig, (ax1, ax2) = plt.subplots(2, 1, sharex=True)
ax1.plot(t, sig)
ax1.set_title('25 Hz and 50 Hz sinusoids')
ax1.axis([0, 1, -2, 2])

# Design the Butterworth filter using
# signal.butter and output='sos'
sos = signal.butter(50, 35, 'lp', fs=1000, output='sos')
# Filter the signal by the filter using signal.sosfilt
# Use signal.sosfiltfilt to get output inphase with input
filtered = signal.sosfiltfilt(sos, sig)
```

```
# Display the output signal
ax2.plot(t, filtered)
ax2.set_title('After 35 Hz Low-pass filter')
ax2.axis([0, 1, -2, 2])
ax2.set_xlabel('Time [seconds]')
plt.tight_layout()
plt.show()
```

Output:

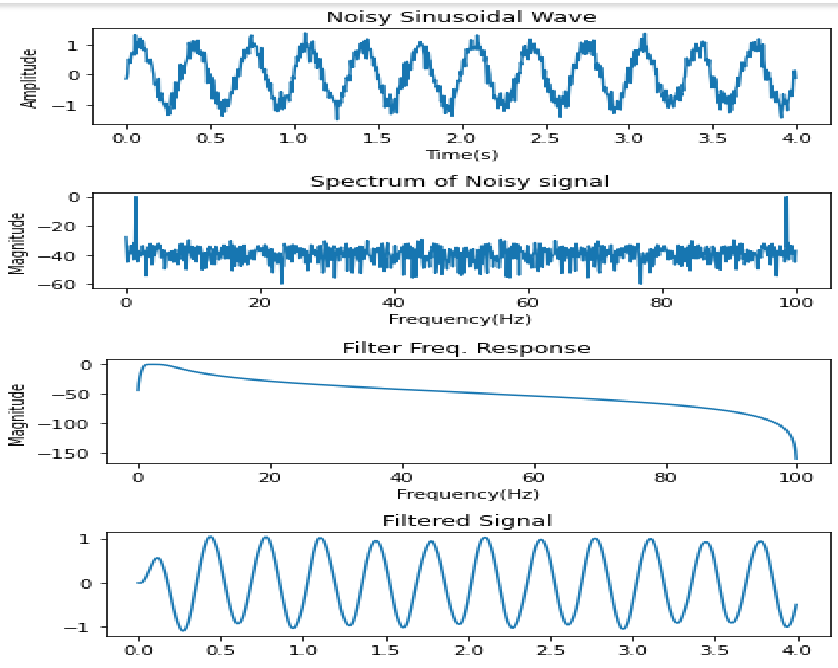


Program (method-2):

```
import matplotlib.pyplot as plt
import numpy as np
from math import pi
import scipy.fftpack as sf
import scipy.signal as sig

plt.close('all')

# Generate a signal
Fs = 200;
t = 4;
n = np.arange(0,t,1/Fs)
f = 3;
x = np.sin(2*pi*f*n)
# Generate a noise
y = np.random.normal(0, 0.2, np.size(x)); # AWGN
x = x + y; # noisy signal

plt.figure(1)
plt.subplot(2,1,1)
plt.plot(n,x); plt.title('Noisy Sinusoidal Wave')
plt.xlabel('Time(s)'); plt.ylabel('Amplitude')

# Take spectral analysis
X_f = abs(sf.fft(x))
l = np.size(x)
fr = (Fs/2)*np.linspace(0,1,l)
xl_m = (2/l)*abs(X_f[0:np.size(fr)]);

plt.subplot(2,1,2)
plt.plot(fr,20*np.log10(xl_m)); plt.title('Spectrum of Noisy signal')
plt.xlabel('Frequency(Hz)'); plt.ylabel('Magnitude')
plt.tight_layout()

# Create a BPF
o = 2;
fc = np.array([1,5])
wc = 2*fc/Fs;
[b,a] = sig.butter(o, wc, btype = 'bandpass')
```

```
# filter response
[W,h] = sig.freqz(b,a, worN = 1024)

W = Fs* W/(2*pi)

plt.figure(2)
plt.subplot(2,1,1)
plt.plot(W, 20*np.log10(h)); plt.title('Filter Freq. Response')
plt.xlabel('Frequency(Hz)'); plt.ylabel('Magnitude')

# Filter signal
x_filt = sig.lfilter(b,a, x)

plt.subplot(2,1,2)
plt.plot(n,x_filt); plt.title('Filtered Signal')
plt.tight_layout();
```

Output:



Additional programs for practice:

1. Given a list L write a program to make a new list to fix the indexes of numbers to in list L to its same value in the new list. Put 0 at remaining indexes. Also print the elements of the new list in the single line. (See explanation for more clarity.)

Input:
[1,5,6]

Output:
```
0 1 0 0 0 5 6
```

Explanation:

List L contains 1,5,9 so at 1,5,9, index of new list the respective values are put and rest are initialized as zero.

Program:

```
l=[1,5,6,9]
m=[]
k=0
for i in range(len(l)):
    for j in range(k,10):
        if l[i]!=j:
            m.append(0)
        else:
            m.append(l[i])
            k=l[i]+1
            break
#print(m)
for i in m:
    print(i,end=" ")
```