S.No: 1	Exp. Name: ***Write the code***	Date:

ID: 0201DCS281

Aim:

Write a python program to demonstrate the working of 'id' functions using in lists. And compare id's of first and second elements in the list

1. Take two list inputs from the user
2. Compare the two lists
3. Then compare the ids of two lists and observe the difference in output

Source Code:

id.py

```
list1 = input("Enter input list 1: ")
list2 = input("Enter input list 2: ")


if(list1==list2):
      print("Result of comparing two lists:  True")


else:
   print("Result of comparing two lists:  False")


if(id(list1[0])==id(list1[1])):
   print("Result of comparision of two ids:  True")

else:
   print("Result of comparision of two ids:  False")
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter input list 1: 1 2 3
Enter input list 2: 1 2 3
Result of comparing two lists: True
Result of comparision of two ids: False

Test Case - 2
User Output
Enter input list 1: 1 2 3
Enter input list 2: 4 5 6
Result of comparing two lists: False
Result of comparision of two ids: False

Noida Institute of Engineering and Technology

S.No: 2	Exp. Name: ***Write the code***	Date:

Aim:

Write a python program that takes in command line arguments as input and print the number of arguments.

Source Code:

cmdarg.py

```
import sys
print('Number of arguments:',len(sys.argv),'arguments.')
print('Argument List:',str(sys.argv))
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Number of arguments: 2 arguments.
Argument List: ['cmdarg.py', 'Codetantra']

Test Case - 2
User Output
Number of arguments: 3 arguments.
Argument List: ['cmdarg.py', '3', '6']

Page No:
ID: 0201DCS281
Noida Institute of Engineering and Technology

S.No: 3	Exp. Name: ***Write a python program to perform Matrix Multiplication.***	Date:

Aim:

Write a python program to perform Matrix Multiplication.

Source Code:

matrixmul.py

```
print("Enter values for matrix - A")
m = int(input("Number of rows, m = "))
n = int(input("Number of columns, n = "))
A=[]
B=[]
for i in range(1,m+1):
   x=[]
   for j in range(1,n+1):
      rc = int(input("Entry in row: {0} column: {1}\n".format(i,j)))
      x.append(rc)
   A.append(x)





print("Enter values for matrix - B")
p = int(input("Number of rows, m = "))
q = int(input("Number of columns, n = "))
for i in range(1,p+1):
   x=[]
   for j in range(1,q+1):
      qc = int(input("Entry in row: {0} column: {1}\n".format(i,j)))
      x.append(qc)
   B.append(x)

print("Matrix - A =",A)
print("Matrix - B =",B)
res=[]
for i in range(1,m+1):
   x=[]
   for j in range(1,q+1):
      x.append(0)
   res.append(x)


for i in range(len(A)):
   for j in range(len(B[0])):
      for k in range(len(B)):
         res[i][j]+=A[i][k]*B[k][j]

print("Matrix - A * Matrix- B =", res)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter values for matrix - A 3
Number of rows, m = 3
Number of columns, n = 3
Entry in row: 1 column: 1 12

Test Case - 1

```
Entry in row: 1 column: 2 7
Entry in row: 1 column: 3 3
Entry in row: 2 column: 1 4
Entry in row: 2 column: 2 5
Entry in row: 2 column: 3 6
Entry in row: 3 column: 1 7
Entry in row: 3 column: 2 8
Entry in row: 3 column: 3 9
Enter values for matrix - B 3
Number of rows, m =  3
Number of columns, n =  4
Entry in row: 1 column: 1 5
Entry in row: 1 column: 2 8
Entry in row: 1 column: 3 1
Entry in row: 1 column: 4 2
Entry in row: 2 column: 1 6
Entry in row: 2 column: 2 7
Entry in row: 2 column: 3 3
Entry in row: 2 column: 4 0
Entry in row: 3 column: 1 4
Entry in row: 3 column: 2 5
Entry in row: 3 column: 3 9
Entry in row: 3 column: 4 1
Matrix - A = [[12, 7, 3], [4, 5, 6], [7, 8, 9]]
Matrix - B = [[5, 8, 1, 2], [6, 7, 3, 0], [4, 5, 9, 1]]
Matrix - A * Matrix- B = [[114, 160, 60, 27], [74, 97, 73, 14], [119, 157, 112, 23]]
```

Page No:
ID: 0201DCS281
Noida Institute of Engineering and Technology

S.No: 4	Exp. Name: ***Write the code***	Date:

Aim:

Write a python program to find all prime numbers within a given range.

Source Code:

prime.py

```
r=int(input("Enter upper limit: "))
for a in range(2,r+1):
   k=0
   for i in range(2,a//2+1):
      if(a%i==0):
         k=k+1
   if(k<=0):
         print(a)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter upper limit: 10
2
3
5
7

Test Case - 2
User Output
Enter upper limit: 20
2
3
5
7
11
13
17
19

Test Case - 3
User Output
Enter upper limit: 100
2
3
5
7
11
13
17
19
23
29
31
37

Test Case - 3
41
43
47
53
59
61
67
71
73
79
83
89
97

ID: 0201DCS281 Page No:

Noida Institute of Engineering and Technology

S.No: 5	Exp. Name: ***Write the code***	Date:

Aim:

Write a Python program to print all primes smaller than or equal to, n using Sieve of Eratosthenes

Source Code:

primes.py

```
def SieveOfEratosthenes(n):

   prime = [True for i in range(n + 1)]
   p = 2
   while (p * p <= n):

      if (prime[p] == True):

         for i in range(p * 2, n + 1, p):
            prime[i] = False
      p += 1
   prime[0]= False
   prime[1]= False

   for p in range(n + 1):
      if prime[p]:
         print (p)

n = int(input("Enter the value: "))
print ("Following are the prime numbers smaller")

print ("than or equal to", n)

SieveOfEratosthenes(n)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the value: 10
Following are the prime numbers smaller
than or equal to 10
2
3
5
7

Test Case - 2
User Output
Enter the value: 15
Following are the prime numbers smaller
than or equal to 15
2
3
5
7
11
13

S.No: 6	Exp. Name: ***Write the code***	Date:

Aim:

Write a Python program to print double sided stair-case pattern

Source Code:

stairpattern.py

```
def pattern(n):
   for i in range(1,n+1):
      k =i + 1 if(i % 2 != 0) else i
      for g in range(k,n):
         if g>=k:
            print(end="  ")
      for j in range(0,k):
         if j == k - 1:
            print(" * ")
         else:
            print(" * ", end = " ")
n =int(input("Enter the number of steps to be printed: "))
pattern(n)
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter the number of steps to be printed:  10
                 *   *
                 *   *
             *   *   *   *
             *   *   *   *
         *   *   *   *   *   *
         *   *   *   *   *   *
     *   *   *   *   *   *   *   *
     *   *   *   *   *   *   *   *
 *   *   *   *   *   *   *   *   *   *
 *   *   *   *   *   *   *   *   *   *
```

Test Case - 2

User Output

```
Enter the number of steps to be printed:  5
       *   *
       *   *
   *   *   *   *
   *   *   *   *
 *   *   *   *   *   *
```

S.No: 7	Exp. Name: ***Write a program to find gcd of two given numbers.***	Date:

Aim:

Write a program to find gcd(highest common factor) of two given numbers. Print the result as shown in the example.

Sample Input and Output:

```
Enter x value: 50
Enter y value: 20
10
```

Source Code:

Gcd.py

```
def computeGCD(x, y):
   gcd = 1
   if x % y == 0:
      return y
   for k in range(int(y / 2), 0, -1):
      if x % k == 0 and y % k == 0:
         gcd = k
         break
   return gcd
x=int(input("Enter x value: "))
y=int(input("Enter y value: "))

print(computeGCD(x, y))
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter x value: 50
Enter y value: 20
10

Test Case - 2
User Output
Enter x value: 500
Enter y value: 36
4

Test Case - 3
User Output
Enter x value: 360
Enter y value: 230
10

Test Case - 4
User Output
Enter x value: 1200
Enter y value: 360
120

S.No: 8	Exp. Name: ***A simple program using recursion***	Date:

Aim:

Write a program to find the `factorial` of a given number using recursion process. Print the result as shown in the examples.

In the below program it writes a **recurfact()** function. In the main program it takes the input and sends it to the **recurfact()** function.

The **recurfact()** function contains base and recursive criterion, where base condition executes only when the value equals **zero (0).**

Sample Input and Output 1:

```
Enter a number: 7
The factorial of the given number is  5040
```

Sample Input and Output 2:

```
Sample Input and Output 1:
```

Source Code:

Func_Ex22.py

```
#Program to illustrate recursion
def recurfact(n):

   if n == 0:
      return 1
   elif n>0:
      return n*recurfact(n-1)
n = int(input("Enter a number: "))
if n < 0:
   print("The factorial does not exist for a negative number")
else:
   print("The factorial of the given number is ",recurfact(n))
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter a number: 1
The factorial of the given number is 1

Test Case - 2
User Output
Enter a number: 10
The factorial of the given number is 3628800

Test Case - 3
User Output
Enter a number: 7
The factorial of the given number is 5040

Test Case - 4

Test Case - 4
User Output
Enter a number: -10
The factorial does not exist for a negative number

Test Case - 5
User Output
Enter a number: 20
The factorial of the given number is 2432902008176640000

ID: 0201DCS281 Page No:

Noida Institute of Engineering and Technology

S.No: 9	Exp. Name: ***Write a program that calculates recursive sum***	Date:

Aim:

Write a program to find the `sum` of a recursive number list like L1 = [1, 10, 9, [3, 5, 7], [5, [6, 7], 97]], print the result as shown in the exmple.

We cannot use **sum(L1)** because it will produce the following error:

```
Traceback (most recent call last):
  File "<interactive input>", line 1, in <module>
TypeError: unsupported operand type(s) for +: 'int' and 'list'
```

We need to go through the list recursively as follows:

```
tot = 0
for element in nested_num_list:
    if type(element) == type([]):
        tot += r_sum(element)
    else:
        tot += element
```

Sample Input and Output:

```
150
```

Source Code:

Func_Ex23.py

```
def r_sum(nested_num_list):

# write your code here
   total = 0
   for element in nested_num_list:
      if type(element) == type([]):
         total = total + r_sum(element)
      else:
         total = total + element
   return total



L1 = [1, 10, 9, [3, 5, 7], [5, [6, 7], 97]]
print(r_sum(L1))
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
150

S.No: 10	Exp. Name: ***Write a program to add two numbers using recursion***	Date:

ID: 0201DCS281

Aim:

Take two integers `a` and `b` from the user. Write a program to add two inputs using recursion. Print the result as shown in the examples.

Sample Input and Output:

```
Please enter an integer: 499
Please enter another integer: 0
499
```

Source Code:

RecurAdd.py

```
#Program to add two numbers using recursion
def add(x, y):
   if(y == 0):
      return x;
   else:
      return(1+add(x,y-1));
a=int(input("Please enter an integer: "))
b=int(input("Please enter another integer: "))

print(add(a, b))
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Please enter an integer: 10
Please enter another integer: 5
15

Test Case - 2
User Output
Please enter an integer: -5
Please enter another integer: 20
15

Test Case - 3
User Output
Please enter an integer: 360
Please enter another integer: 32
392

Test Case - 4
User Output
Please enter an integer: 499
Please enter another integer: 0
499

Noida Institute of Engineering and Technology

S.No: 11	Exp. Name: ***Write the code***	Date:

Aim:

Write a python program to print 'n' terms of Fibonacci series using iteration(Recursion).

Source Code:

recursion.py

```
def recur_fibo(n):
   if n<= 1:
      return n
   else:
      return(recur_fibo(n-1) + recur_fibo(n-2))
nterms =int(input("Enter the fibonacci term to be printed: "))
if nterms <= 0:
   print("Invalid input")
else:
   print("Fibonacci Series is: ")
   for i in range(nterms):
      print(recur_fibo(i),end = " ")
   print()
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the fibonacci term to be printed: 5
Fibonacci Series is:
0 1 1 2 3

Test Case - 2
User Output
Enter the fibonacci term to be printed: 8
Fibonacci Series is:
0 1 1 2 3 5 8 13

Test Case - 3
User Output
Enter the fibonacci term to be printed: 0
Invalid input

S.No: 12	Exp. Name: ***Write the code***	Date:

Aim:

Write a python program to print 'n' terms of Fibonacci series using iteration(Use Dynamic Programming).

Source Code:

recursion.py

```
def recursion(n):
   a=1
   b=1
   if n==1:
      print('0')
   elif n==2:
      print('0','1')
   else:
      print("Fibonacci sequence:")
      print('0')
      print(a)
      print(b)
      for i in range(n-3):
         total = a + b
         b=a
         a= total
         print(total)
      print()
      return b
nterms = int(input("How many terms to print?: "))
recursion(nterms)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
How many terms to print?: 5
Fibonacci sequence:
0
1
1
2
3

Test Case - 2
User Output
How many terms to print?: 3
Fibonacci sequence:
0
1
1

Page No: ID: 0201DCS281
Noida Institute of Engineering and Technology

S.No: 13	Exp. Name: ***Exercise Problem 3 Python Functions***	Date:

Aim:

Write a function **frequency(l)**, which outputs the items with lowest frequency and items with highest frequency.

Consider an example: [1, 2, 1, 2, 3, 4, 6, 4, 6, 6, 7, 8] should return ([3, 7, 8], [6], 1, 3) which means that 3, 7, 8 have appeared only once in the list which is the minimum number of times and 6 has appeared 3 times which is the maximum number of times.

[1, 2, 3, 4] should return ('Null', [1, 2, 3, 4], 0, 1) which means there is no minimum count.

Sample Input and Output:

```
Please enter integers separated by spaces: 1 2 3 1 2 5
([3, 5], [1, 2], 1, 2)
```

Source Code:

Frequency.py

```
def frequency (seq):

   d = {}
   for i in seq:
      d[i] = d.get(i, 0) + 1
   min_num = min(d.values())
   max_num = max(d.values())
   min_list = [k for k,v in d.items() if v == min_num]
   max_list = [k for k,v in d.items() if v == max_num]
   return(sorted(min_list), sorted(max_list), min_num, max_num)





l1 = [int(x) for x in input("Please enter integers separated by spaces: ").split()]
print (frequency(l1))
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Please enter integers separated by spaces: 1 2 3 1 2 5
([3, 5], [1, 2], 1, 2)

Test Case - 2
User Output
Please enter integers separated by spaces: 5 9 6 5 3 2 1 6
([1, 2, 3, 9], [5, 6], 1, 2)

Test Case - 3
User Output
Please enter integers separated by spaces: 8 9 4 66 85 8 4 22 85 3 85
([3, 9, 22, 66], [85], 1, 3)

S.No: 14	Exp. Name: ***Slicing Operator***	Date:

Aim:

Python provides many ways to extract a part of a string using a concept called **Slicing**.

Slicing makes it possible to access parts (segments) of the string by providing a **startIndex** and an **endIndex**.

The syntax for using the slice operation on a string is:

```
[ startIndex : endIndex : step ] #where  is the string variable#startIndex, endIndex
and step are all optional
```

For example, in the below code: slice[start : stop : step] operator extracts sub string from the string. A segment of a string is called a slice. The indexes start and stop should be valid.

step is the increment or decrement . A `positive` step will travel left to right and increments the index by step.

A `negative` step will travel from right to left and decrements the index by step.

If we are not providing starting of index position in Slice `[ ]` operator then interpreter takes starting index zero(**0**) as default.

If end index is not specified for slice `[ ]` operator then Interpreter takes the end of String as default stop index.

Click on the Live Demo button to know more about **Slicing in Strings**.

Consider Example like

```
lang = "Python"
print(lang[0:]) # means it prints 0th index position to ending index of String.
Python
print(lang[:6]) # means it prints 0th index to (n-1) i.e. 6-1 = 5th position of
String.
Python
```

Let us consider an Example:

```
a = "HELLO"
Print(a[0:4]) # index starts from 0 and ends before 4 i.e.3.
HELL
Print(a[ :3]) # prints 0th index to 2nd index(3 - 1)
HEL
Print(a[0: ]) # Starts at 0 prints upto last index.
HELLO
```

In the following example start and stop are not given, so they will be defaulted to beginning and end of the string.
The step is positive indicating left to right traversal and increment is **1**.

```
a = "Python"
print (a[::1])
```

The result is

```
Python
```

In the following example start and stop are not given, so they will be defaulted to end and beginning of the string.
The step is negative indicating right to left traversal and decrement is **1**.

```
a = "Python"
print(a[::-1])
```

The result is

```
nohtyP
```

Let us take another example:

```
a = "Python"
print(a[-1::-3])
```

The result is

```
'nt'
```

Here we are indicating start as **-1** and stop not specified means traverse from right to left till start of the string and step is **-3** means decrement is done by **3** from right to left of a string.

```
a = "Python"
print(a[4:1:-1])
```

The result is

```
'oht'
```

Here we are indicating start as **4** and stop as **1** and index as **-1.** So it will start index **4(o)** and will traverse right to left till index **2(t).**

```
a = "Python"
print(a[2:5:-1])
```

The result is

```
'' (Null string)
```

Here we are indicating start as **2** and stop as **5** (which implies that the direction as left to right, but the step is **-1** which means the direction is right to left.
Because this is not possible it returns a null string

Write the missing code in the below program and print the result as shown in the example. Follow the instructions given in comment lines in the below program.

Sample Input and Output:

```
String is How are you?
are
w a
you
uoy
you?
```

Source Code:

```
String2.py
```

ID: 0201DCS281 Page No:
Noida Institute of Engineering and Technology

```
str = "How are you?"
print("String is", str)
# print 'are' in String using Slicing with Positive Index
print(str[4:7])
# print 'w a' in String using Slicing with Positive Index
print(str[2:5:1])
# print 'you' in String using Slicing with Negative Index
print(str[8:11])
# print 'uoy' in the string using slicing and Negative indexes and negative step
print(str[-2:-5:-1])
# print 'you?' in String using Slicing with Negative Index
print(str[-4:])
```


Execution Results - All test cases have succeeded!

Test Case - 1
User Output
String is How are you?
are
w a
you
uoy
you?

S.No: 15	Exp. Name: ***Write program to print two strings in given below formats***	Date:

Aim:

Take two strings of different lengths as input from the user using `input()` function. Write a program to enclose the longer string with-in the shorter string, print the result as shown in the examples.

Sample Input and Output 1:

```
Enter first string: Big Data
Enter second string: Hadoop
HadoopBig DataHadoop
```

Here **len(Big Data)** is **8** and **len(Hadoop)** is **6** so **Big Data** is enclosed with **Hadoop**.

Sample Input and Output 2:

```
Enter first string: Django
Enter second string: Django
Both strings are of same length - Enclosing Not done
```

Source Code:

StringTest13.py

```
#use len() to find length of String
str1 = input("Enter first string: ")
str2 = input("Enter second string: ")

len1 = len(str1)
len2 = len(str2)

if len1==len2:
   print("Both strings are of same length - Enclosing Not done")
elif len1>len2:
   print(str2+str1+str2)
else:
   print(str1+str2+str1)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter first string: Python
Enter second string: Java
JavaPythonJava

Test Case - 2
User Output
Enter first string: Big Data
Enter second string: Hadoop
HadoopBig DataHadoop

Test Case - 3
User Output
Enter first string: Perl
Enter second string: Swift
PerlSwiftPerl

Test Case - 4
User Output
Enter first string: Django
Enter second string: Django
Both strings are of same length - Enclosing Not done

ID: 0201DCS281 Page No:

Noida Institute of Engineering and Technology

S.No: 16	Exp. Name: ***Write a program to find whether a given string starts with 'Python' or not.***	Date:

Aim:

Take string as input from the user using `input()` function. Write a program to check whether the given input begins with **Python** or not. If yes, print the input string as output, otherwise add **Python** to input string and, print the result as shown in the examples .

Sample Input and Output 1:

```
Enter a string: Python
The given string is: Python
```

Sample Input and Output 2:

```
Enter a string: World
The string after adding 'Python' is: Python World
```

Source Code:

StringTest1.py

```
# Note: Python and python are different.
string1=input("Enter a string: ")
string2="Python"
if  string1=="Python":
   print("The given string is:",string1)
else:
   print("The string after adding 'Python' is:",string2+' '+string1)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter a string: String
The string after adding 'Python' is: Python String

Test Case - 2
User Output
Enter a string: World
The string after adding 'Python' is: Python World

Test Case - 3
User Output
Enter a string: Python
The given string is: Python

Test Case - 4
User Output
Enter a string: python
The string after adding 'Python' is: Python python

Test Case - 5
User Output
Enter a string: PYthon
The string after adding 'Python' is: Python PYthon

S.No: 17	Exp. Name: ***Write the code***	Date:

Aim:

Write a python program to add 'ing' at the end of a given string (length should be at least 3). If the given string already ends with 'ing' then add 'ly' instead. If the string length of the given string is less than 3, leave it unchanged.

Sample String : 'abc' Expected Result : 'abcing' Sample String : 'string' Expected Result : 'stringly'

Source Code:

strlength.py

```
string=input("Enter string: ")
if len(string)>=3:
   if string[-3::1] !="ing":
      print(string+"ing")
   else:
      print(string+"ly")
else:
   len(string)<3
   print(string)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter string: abc
abcing

Test Case - 2
User Output
Enter string: string
stringly

Test Case - 3
User Output
Enter string: is
is

S.No: 18	Exp. Name: ***Write the code***	Date:

ID: 0201DCS281

Aim:

To get a string from a given string where all occurrences of its first char have been changed to '$', except the first char itself.

Source Code:

strReplace.py

```
def change_char(string):
   char=string[0]
   string=string.replace(char,'$')
   string=char+string[1:]
   return string
str1=input("Enter input: ")
print(change_char(str1))
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter input: restart
resta$t

Test Case - 2
User Output
Enter input: python
python

Noida Institute of Engineering and Technology

S.No: 19	Exp. Name: ***Write a program to find how many times each character is repeated in a given string.***	Date:

ID: 0201DCS281

Aim:

Take string as input from the console using `input()` function. Write a program to find how many times each character is repeated in a given string. Print each character in the string in sorted order with number of times it is repeated as shown in the example.

Sample Input and Output:

```
Please enter sentence: Hello Python!
' ' 1
'!' 1
'H' 1
'P' 1
'e' 1
'h' 1
'l' 2
'n' 1
'o' 2
't' 1
'y' 1
[' ', '!', 'H', 'P', 'e', 'h', 'l', 'n', 'o', 't', 'y']
```

Source Code:

CharCount.py

```
from collections import Counter
def char_freq(string):
   s1=[]
   d=Counter(string)
   for i in d:
      print("'"+i+"'"+"\t"+str(d[i]))
      s1.append(i)
   print(s1)
string=input("Please enter sentence: ")
s1=sorted(string);
char_freq(s1)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Please enter sentence: Hello Python!
' ' 1
'!' 1
'H' 1
'P' 1
'e' 1
'h' 1
'l' 2
'n' 1
'o' 2
't' 1
'y' 1
[' ', '!', 'H', 'P', 'e', 'h', 'l', 'n', 'o', 't', 'y']

Noida Institute of Engineering and Technology

Test Case - 2

User Output

```
Please enter sentence:  There are more spaces than words in no sentence.
' '     8
'.'     1
'T'     1
'a'     3
'c'     2
'd'     1
'e'     8
'h'     2
'i'     1
'm'     1
'n'     5
'o'     3
'p'     1
'r'     4
's'     4
't'     2
'w'     1
[' ', '.', 'T', 'a', 'c', 'd', 'e', 'h', 'i', 'm', 'n', 'o', 'p', 'r', 's', 't', 'w']
```

Test Case - 3

User Output

```
Please enter sentence:  NO_SPACES_HERE
'A'     1
'C'     1
'E'     3
'H'     1
'N'     1
'O'     1
'P'     1
'R'     1
'S'     2
'_'     2
['A', 'C', 'E', 'H', 'N', 'O', 'P', 'R', 'S', '_']
```

ID: 0201DCS281 Page No:

Noida Institute of Engineering and Technology

S.No: 20	Exp. Name: ***Write a program to print every character of a string twice.***	Date:

ID: 0201DCS281

Aim:

Take string as input from the user using `input()` function. Write a program to print every character of the given input string twice, print the result to the console as shown in the example.

Sample Input and Output:

```
Enter a string: Lists
The result is:  LLiissttss
```

Source Code:

StringTest12.py

```
s=input("Enter a string: ")
print("The result is: ",end=" ")
for i in s:
   print(i+i,end="")
print()
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter a string: Hello Python
The result is: HHeellllloo PPyytthhoonn

Test Case - 2
User Output
Enter a string: Cool
The result is: CCoooolll

Test Case - 3
User Output
Enter a string: Lists
The result is: LLiissttss

Test Case - 4
User Output
Enter a string: Dictionary
The result is: DDiiccttiioonnaarryy

Test Case - 5
User Output
Enter a string: Sets
The result is: SSeettss

Noida Institute of Engineering and Technology

S.No: 21	Exp. Name: ***Write a program to print half of a given string.***	Date:

ID: 0201DCS281

Aim:

Take string as input from the console using `input()` function. Calculate the length of the string, If length is **even** print the first half of the string, if the length is **odd** remove the middle character and print the second half of the string as shown in examples.

Sample Input and Output 1:

```
Enter a string: Python
First half string of given even length string is: Pyt
```

Sample Input and Output 2:

```
Enter a string: Hello World
Second half string of given odd length string is: World
```

Source Code:

StringTest19.py

```
string=input("Enter a string: ")
length=len(string)
if length%2==0:
   print("First half string of given even length string is:",string[ : length//2])
else:
   print("Second half string of given odd length string is:",string[(length+1)//2 : ])
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter a string: Python
First half string of given even length string is: Pyt

Test Case - 2
User Output
Enter a string: Hello World
Second half string of given odd length string is: World

Test Case - 3
User Output
Enter a string: PythonWorld
Second half string of given odd length string is: World

Test Case - 4
User Output
Enter a string: Strings
Second half string of given odd length string is: ngs

Noida Institute of Engineering and Technology

S.No: 22	Exp. Name: ***Write the code***	Date:

Aim:

Write a program that accepts a comma-separated sequence of words as input and prints the words in a comma-separated sequence after sorting them alphabetically.

Source Code:

sortInput.py

```
words=input("Input comma separated sequence of words: ")
items=[word for word in words.split(",")]
print(",".join(sorted(list(set(items)))))
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Input comma separated sequence of words: red,black,pink,green,yellow
black,green,pink,red,yellow

Test Case - 2
User Output
Input comma separated sequence of words: ram,ravi,raj
raj,ram,ravi

S.No: 23	Exp. Name: ***Write the code***	Date:

ID: 0201DCS281

Aim:

Write a program that accepts a sequence of whitespace separated words as input and prints the words after removing all duplicate words and sorting them alphanumerically.

Source Code:

removeSpaces.py

```
input_seq=input("Enter input: ")
input_splitted=input_seq.split(" ")
word_list=[]
for i in input_splitted:
   if i not in word_list:
      word_list.append(i)
   else:
      continue
print((" ").join(word_list))
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter input: Python is great and Java is also great
Python is great and Java also

Test Case - 2
User Output
Enter input: python is easy
python is easy

Noida Institute of Engineering and Technology

S.No: 24	Exp. Name: ***Write a Program to create a List***	Date:

ID: 0201DCS281

Aim:

Take multiple string values from the console using **input()** function. Write a program to create a list with the given input string values, print the result as shown in the example.

Sample Input and Output:

```
Enter list elements separated by ,(comma): Krishna,Godavari,Kaveri
List of elements: ['Krishna', 'Godavari', 'Kaveri']
Type of list:
```

Source Code:

List2.py

```
elements=input("Enter list elements separated by ,(comma): ").split(",")
list=elements
print("List of elements:",list)
print("Type of list:",type(list))
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter list elements separated by ,(comma): Krishna,Godavari,Kaveri
List of elements: ['Krishna', 'Godavari', 'Kaveri']
Type of list:

Test Case - 2
User Output
Enter list elements separated by ,(comma): Hello,World,welcome,to,Python,World
List of elements: ['Hello', 'World', 'welcome', 'to', 'Python', 'World']
Type of list:

Test Case - 3
User Output
Enter list elements separated by ,(comma): Red,Green,Blue
List of elements: ['Red', 'Green', 'Blue']
Type of list:

Test Case - 4
User Output
Enter list elements separated by ,(comma): Python,Java
List of elements: ['Python', 'Java']
Type of list:

Noida Institute of Engineering and Technology

S.No: 25	Exp. Name: ***Write a Program to print the first and last elements of a List.***	Date:

Aim:

Create an input list with the user given elements. Write a program to find the first and last elements in the list, print the result as shown in the example.

Sample Input and Output 1:

```
Enter list elements separated by ,(Comma): Godavari,Krishna,Kaveri
The first and last elements of a list are: Godavari Kaveri
```

Sample Input and Output 2:

```
Enter list elements separated by ,(Comma): 10,20,30,45,56
The first and last elements of a list are: 10 56
```

Source Code:

List4.py

```
elements=list(input("Enter list elements separated by ,(Comma): ").split(","))
print("The first and last elements of a list are: "+elements[0]+" "+elements[-1])
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter list elements separated by ,(Comma): Godavari,Krishna,Kaveri
The first and last elements of a list are: Godavari Kaveri

Test Case - 2
User Output
Enter list elements separated by ,(Comma): Ramesh,Suresh,Mahesh,Kamesh,Satish
The first and last elements of a list are: Ramesh Satish

Test Case - 3
User Output
Enter list elements separated by ,(Comma): 10,20,30,45,56
The first and last elements of a list are: 10 56

Test Case - 4
User Output
Enter list elements separated by ,(Comma): One,Two,Three
The first and last elements of a list are: One Three

S.No: 26	Exp. Name: ***Write a Program to find the largest of the first and last elements of a List***	Date:

ID: 0201DCS281

Aim:

Create an integer list with the user given inputs. Compare the first and last elements in the list, and print the largest one as shown in the example.

Sample Input and Output:

```
Enter integer values separated by ,(Comma): 48,96,55,63,255,96,800
[48, 96, 55, 63, 255, 96, 800]
Largest element among first and last elements: 800
```

Source Code:

List8.py

```
int_list=list(input("Enter integer values separated by ,(Comma): ").split(","))
for i in range(len(int_list)):
   int_list[i]=int(int_list[i])
print(int_list)
if int_list[0]>int_list[-1]:
   print("Largest element among first and last elements:",int_list[0])
else:
   print("Largest element among first and last elements:",int_list[-1])
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter integer values separated by ,(Comma): 48,96,55,63,255,96,800
[48, 96, 55, 63, 255, 96, 800]
Largest element among first and last elements: 800

Test Case - 2
User Output
Enter integer values separated by ,(Comma): 78,96,54,12,30
[78, 96, 54, 12, 30]
Largest element among first and last elements: 78

Test Case - 3
User Output
Enter integer values separated by ,(Comma): 77,99
[77, 99]
Largest element among first and last elements: 99

Noida Institute of Engineering and Technology

S.No: 27	Exp. Name: ***Write a program to remove all the duplicates from a List***	Date:

ID: 0201DCS281

Aim:

Create a list with the user given input elements. Write a program to remove all the duplicate elements from the given input list.

Sample Input and Output:

```
Enter list of elements separated by ,(comma): 10,20,10,30
['10', '20', '10', '30']
List after removing all duplicates: ['10', '20', '30']
```

Source Code:

List11.py

```
element=input("Enter list of elements separated by ,(comma): ").split(",")
element_list=list(element)
print(element_list)
res=[]
for i in element_list:
   if i not in res:
      res.append(i)
print("List after removing all duplicates:",str(res))
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter list of elements separated by ,(comma): 10,20,10,30
['10', '20', '10', '30']
List after removing all duplicates: ['10', '20', '30']

Test Case - 2
User Output
Enter list of elements separated by ,(comma): Blue,Green,Yellow,Blue,Orange
['Blue', 'Green', 'Yellow', 'Blue', 'Orange']
List after removing all duplicates: ['Blue', 'Green', 'Yellow', 'Orange']

Test Case - 3
User Output
Enter list of elements separated by ,(comma): KKD,HYD,RJY,VIZ,HYD,AMP,AMP
['KKD', 'HYD', 'RJY', 'VIZ', 'HYD', 'AMP', 'AMP']
List after removing all duplicates: ['KKD', 'HYD', 'RJY', 'VIZ', 'AMP']

Test Case - 4
User Output
Enter list of elements separated by ,(comma): 11,22,33,44,55
['11', '22', '33', '44', '55']
List after removing all duplicates: ['11', '22', '33', '44', '55']

Noida Institute of Engineering and Technology

S.No: 28	Exp. Name: ***Write the code***	Date:

Aim:

Write a program to implement stack using the list

Source Code:

stackImplement.py

```
stack=[]
while(True):
   s=input("Enter element, 'XXX' to end: ")
   if s=="XXX":
      break;
   stack.append(s)
print("Initial stack")
print(stack)
print("Elements poped from stack:")
for i in range(3):
   l=stack.pop()
   print(l)
print("Stack after elements are poped:")
print(stack)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter element, 'XXX' to end: a
Enter element, 'XXX' to end: b
Enter element, 'XXX' to end: c
Enter element, 'XXX' to end: XXX
Initial stack
['a', 'b', 'c']
Elements poped from stack:
c
b
a
Stack after elements are poped:
[]

Test Case - 2
User Output
Enter element, 'XXX' to end: a
Enter element, 'XXX' to end: b
Enter element, 'XXX' to end: c
Enter element, 'XXX' to end: d
Enter element, 'XXX' to end: XXX
Initial stack
['a', 'b', 'c', 'd']
Elements poped from stack:
d
c
b
Stack after elements are poped:
['a']

S.No: 29	Exp. Name: ***Write the code***	Date:

Aim:

Write a program to implement queue using the list

Source Code:

stackImplement.py

```
stack=[]
while(True):
   s=input("Enter element, 'XXX' to end: ")
   if s=="XXX":
      break;
   stack.append(s)
print("Initial queue")
print(stack)
print("Elements dequeued from queue:")
for i in range(3):
   l=stack.pop(0)
   print(l)
print("Queue after removing elements:")
print(stack)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter element, 'XXX' to end: a
Enter element, 'XXX' to end: b
Enter element, 'XXX' to end: c
Enter element, 'XXX' to end: XXX
Initial queue
['a', 'b', 'c']
Elements dequeued from queue:
a
b
c
Queue after removing elements:
[]

Test Case - 2
User Output
Enter element, 'XXX' to end: 1
Enter element, 'XXX' to end: 2
Enter element, 'XXX' to end: 3
Enter element, 'XXX' to end: 5
Enter element, 'XXX' to end: 6
Enter element, 'XXX' to end: 2
Enter element, 'XXX' to end: 3
Enter element, 'XXX' to end: 2
Enter element, 'XXX' to end: XXX
Initial queue
['1', '2', '3', '5', '6', '2', '3', '2']
Elements dequeued from queue:
1
2
3
Queue after removing elements:
['5', '6', '2', '3', '2']

Page No:
ID: 0201DCS281
Noida Institute of Engineering and Technology

S.No: 29	Exp. Name: ***Write the code***	Date:

Aim:

Write a program to implement queue using the list

Source Code:

stackImplement.py

```
stack=[]
while(True):
   s=input("Enter element, 'XXX' to end: ")
   if s=="XXX":
      break;
   stack.append(s)
print("Initial queue")
print(stack)
print("Elements dequeued from queue:")
for i in range(3):
   l=stack.pop(0)
   print(l)
print("Queue after removing elements:")
print(stack)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter element, 'XXX' to end: a
Enter element, 'XXX' to end: b
Enter element, 'XXX' to end: c
Enter element, 'XXX' to end: XXX
Initial queue
['a', 'b', 'c']
Elements dequeued from queue:
a
b
c
Queue after removing elements:
[]

Test Case - 2
User Output
Enter element, 'XXX' to end: 1
Enter element, 'XXX' to end: 2
Enter element, 'XXX' to end: 3
Enter element, 'XXX' to end: 5
Enter element, 'XXX' to end: 6
Enter element, 'XXX' to end: 2
Enter element, 'XXX' to end: 3
Enter element, 'XXX' to end: 2
Enter element, 'XXX' to end: XXX
Initial queue
['1', '2', '3', '5', '6', '2', '3', '2']
Elements dequeued from queue:
1
2
3
Queue after removing elements:
['5', '6', '2', '3', '2']

Page No:
ID: 0201DCS281
Noida Institute of Engineering and Technology

S.No: 31	Exp. Name: ***Membership Test in a Dictionary***	Date:

Aim:

Write a program to check whether the given `key` exists in the dictionary or not. If the **key** exists in the dictionary then print `True`, otherwise print `False` as shown in the example.

Sample Input and Output 1:

```
Enter elements separated by ,(comma) for keys: 11,22,33,44,55
Enter elements separated by ,(comma) for values: 111,222,333,444,555,666
Enter key to check whether the element exist in dictionary or not: 33
True
```

Sample Input and Output 2:

```
Enter elements separated by ,(comma) for keys: R,G,B
Enter elements separated by ,(comma) for values: Red,Green,Blue
Enter key to check whether the element exist in dictionary or not: O
False
```

Source Code:

Dictmemb.py

```
key=input("Enter elements separated by ,(comma) for keys: ").split(",")
element=input("Enter elements separated by ,(comma) for values: ").split(",")
k=input("Enter key to check whether the element exist in dictionary or not: ")
if (k in key):
   print("True")
else:
   print("False")
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter elements separated by ,(comma) for keys:  11,22,33,44,55
Enter elements separated by ,(comma) for values:  111,222,333,444,555,666
Enter key to check whether the element exist in dictionary or not:  33
True
```

Test Case - 2

User Output

```
Enter elements separated by ,(comma) for keys:  P,F,D,H,S
Enter elements separated by ,(comma) for values:  Python,Flask,Django,Hibernate,Spring
Enter key to check whether the element exist in dictionary or not:  D
True
```

Test Case - 3

User Output

```
Enter elements separated by ,(comma) for keys:  R,G,B
Enter elements separated by ,(comma) for values:  Red,Green,Blue
Enter key to check whether the element exist in dictionary or not:  O
False
```

Test Case - 4

User Output

```
Enter elements separated by ,(comma) for keys:  101,102,103,104
Enter elements separated by ,(comma) for values:  James,Oliver,John,Charlie
Enter key to check whether the element exist in dictionary or not:  104
True
```

S.No: 32	Exp. Name: ***Membership test in a Tuple***	Date:

Aim:

Create a tuple with the user given inputs. Write a program using membership operators to check whether the given element is present in the tuple or not. Print the result as shown in the examples.

Sample Input and Output 1:

```
Enter tuple elements separated by ,(Comma): 10,20,30,40,50,60
Elements of tuple: ('10', '20', '30', '40', '50', '60')
Enter an element to check whether it exist in tuple or not: 50
True
```

Sample Input and Output 2:

```
Enter tuple elements separated by ,(Comma): Python,Perl,Php,Java,Swift
Elements of tuple: ('Python', 'Perl', 'Php', 'Java', 'Swift')
Enter an element to check whether it exist in tuple or not: R
False
```

Source Code:

Tuple03.py

```
tup_ele= input("Enter tuple elements separated by ,(Comma): ").split(",")
tup=tuple(tup_ele)
print('Elements of tuple: '+str(tup))
element=input("Enter an element to check whether it exist in tuple or not: ")
if (element in tup):
   print("True")
else:
   print("False")
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter tuple elements separated by ,(Comma):  10,20,30,40,50,60
Elements of tuple: ('10', '20', '30', '40', '50', '60') 50
Enter an element to check whether it exist in tuple or not:  50
True
```

Test Case - 2

User Output

```
Enter tuple elements separated by ,(Comma):  Python,Perl,Php,Java,Swift
Elements of tuple: ('Python', 'Perl', 'Php', 'Java', 'Swift') R
Enter an element to check whether it exist in tuple or not:  R
False
```

Test Case - 3

User Output

```
Enter tuple elements separated by ,(Comma):  Red,Green,Blue,Orange,Yellow,Black,White
Elements of tuple: ('Red', 'Green', 'Blue', 'Orange', 'Yellow', 'Black', 'White') Orange
Enter an element to check whether it exist in tuple or not:  Orange
True
```

Test Case - 4

User Output

```
Enter tuple elements separated by ,(Comma):  1.0,2.0,3.0,4.2,5.8,9.3
Elements of tuple: ('1.0', '2.0', '3.0', '4.2', '5.8', '9.3') 2.0
Enter an element to check whether it exist in tuple or not:  2.0
True
```

S.No: 33	Exp. Name: ***Write a Program to Find Sum of Tuple Elements***	Date:

ID: 0201DCS281

Aim:

Create a tuple with integer elements. Write a program to find **sum** of all tuple elements, print the result as shown in the example.

Sample Input and Output:

```
Enter integer elements separated by ,(Comma): 10,20,30,40,50,60,70,80,90,100
Tuple elements: (10, 20, 30, 40, 50, 60, 70, 80, 90, 100)
Sum of tuple elements: 550
```

Source Code:

Tuple13.py

```
tup_ele=tuple(input("Enter integer elements separated by ,(Comma): ").split(","))
res=tuple(map(int,tup_ele))
print("Tuple elements:",res)
sum=0
for i in tup_ele:
   sum+=int(i)
print("Sum of tuple elements:" ,sum)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter integer elements separated by ,(Comma): 10,20,30,40,50,60,70,80,90,100
Tuple elements: (10, 20, 30, 40, 50, 60, 70, 80, 90, 100)
Sum of tuple elements: 550

Test Case - 2
User Output
Enter integer elements separated by ,(Comma): 100,200,300,400
Tuple elements: (100, 200, 300, 400)
Sum of tuple elements: 1000

Test Case - 3
User Output
Enter integer elements separated by ,(Comma): 150,250,360,789
Tuple elements: (150, 250, 360, 789)
Sum of tuple elements: 1549

Test Case - 4
User Output
Enter integer elements separated by ,(Comma): 78,88,59,63,25
Tuple elements: (78, 88, 59, 63, 25)
Sum of tuple elements: 313

Test Case - 5
User Output
Enter integer elements separated by ,(Comma): 77,44,11,22,55,33,66,88
Tuple elements: (77, 44, 11, 22, 55, 33, 66, 88)
Sum of tuple elements: 396

Noida Institute of Engineering and Technology

S.No: 34	Exp. Name: ***Write the code***	Date:

Aim:

To write a python program find the square root of a number using Newton's method

Source Code:

square.py

```
def squareroot(n,l):
   x=n;
   count=0
   while(1):
      count+=1
      ans=0.5*(x+(n/x))
      if(abs(ans-x)<l):
         break
      x=ans
      g=float("{0:.4f}".format(ans))
   return g

n=int(input("Enter n: "))
l=float(input("Enter l: "))
print(squareroot(n,l))
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter n: 327
Enter l: 0.00001
18.0831

Test Case - 2
User Output
Enter n: 16
Enter l: 0.0001
4.0

S.No: 35	Exp. Name: ***Write the code***	Date:

Aim:

To write a python program exponentiation (power of a number).

Note: Take number and exponent from the user.

Source Code:

exponent.py

```
n=int(input("Enter n value: "))
e=int(input("Enter exponent value: "))
result=pow(n,e)
print("The value of %d to the power %d is: %d"%(n,e,result))
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter n value: 9
Enter exponent value: 5
The value of 9 to the power 5 is: 59049

Test Case - 2
User Output
Enter n value: 2
Enter exponent value: 10
The value of 2 to the power 10 is: 1024

Test Case - 3
User Output
Enter n value: 3
Enter exponent value: 2
The value of 3 to the power 2 is: 9

S.No: 36	Exp. Name: ***Write the code***	Date:

Aim:

To write a python program linear search.

Source Code:

linearSearch.py

```
def linearsearch(list,x):
   for i in range(len(list)):
      if list[i]==x:
         return i
   return -1
list=[int(a) for a in input("Enter the list of numbers: ").split(" ")]
x=int(input("The number to search for: "))
res=linearsearch(list,x)
if res==-1:
   print("{} was not found.".format(x))
else:
   print("{} was found at index {}.".format(x,res))
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the list of numbers: 12 23 45 3 2 1
The number to search for: 3
3 was found at index 3.

Test Case - 2
User Output
Enter the list of numbers: 12 58 -9 89 78 93
The number to search for: 100
100 was not found.

S.No: 37	Exp. Name: ***Write the code***	Date:

Aim:

To write a python program Binary search.

Source Code:

binarySearch.py

```
def binary_search(list,low,high,x):
   if high>=low:
      mid=(high+low)//2
      if list[mid]==x:
         return mid
      elif list[mid]>x:
         return binary_search(list,low,mid-1,x)
      else:
         return binary_search(list,mid+1,high,x)
   else:
      return -1
lst=[]
size=int(input("Enter size of list: "))
for i in range(0,size):
   ele=int(input("Enter your number: "))
   lst.append(ele)
list=sorted(lst)
print("After sorting list is: ",list)
x=int(input("The number to search for: "))
res=binary_search(list,0,len(list)-1,x)
if res != -1:
   print("{} was found at index {}.".format(x,res))
else:
   print("{} was not found.".format(x))
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter size of list: 5
Enter your number: 12
Enter your number: 3
Enter your number: 45
Enter your number: 68
Enter your number: 95
After sorting list is: [3, 12, 45, 68, 95] 3
The number to search for: 3
3 was found at index 0.

Test Case - 2
User Output
Enter size of list: 4
Enter your number: 89
Enter your number: 87
Enter your number: 14
Enter your number: 25
After sorting list is: [14, 25, 87, 89] 14
The number to search for: 14
14 was found at index 0.

S.No: 38	Exp. Name: ***Write the code***	Date:

Aim:

To write a python program selection sort.

Source Code:

selectionSort.py

```
def selectionsort(item):
   n=len(item)
   for i in range(n):
      min_val=i
      for j in range(i+1,n):
         if item[min_val]>item[j]:
            min_val=j
      if min_val !=i:
         temp=item[i]
         item[i]=item[min_val]
         item[min_val]=temp
   return item
lst=input("Enter the list of numbers: ").split(" ")
print(selectionsort(lst))
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the list of numbers: 44 55 2 3
['2', '3', '44', '55']

Test Case - 2
User Output
Enter the list of numbers: 89 63 25 12
['12', '25', '63', '89']

S.No: 39	Exp. Name: ***Write the code***	Date:

Aim:

To write a python program Insertion sort.

Source Code:

insertionSort.py

```
def insertionsort(item):
   for i in range(1,len(item)):
      key=item[i]
      j=i-1
      while j>=0 and key<item[j]:
         item[j+1]=item[j]
         j-=1
      item [j+1]=key
lst=list(input("Enter the list of numbers: ").split(" "))
print(lst, sep=", ")
insertionsort(lst)
print(lst)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the list of numbers: 25 98 63 78 99 54
['25', '98', '63', '78', '99', '54']
['25', '54', '63', '78', '98', '99']

S.No: 40	Exp. Name: ***To write a python program merge sort.***	Date:

Aim:

To write a python program merge sort.

Source Code:

mergeSort.py

```
def mergesort(lis):
   print("Splitting ",lis)
   if len(lis)>1:
      mid = len(lis)//2
      left = lis[:mid]
      right = lis[mid:]
      mergesort(left)
      mergesort(right)
      i=j=k=0
      while i < len(left) and j < len(right):
         if left[i]<=right[j]:
            lis[k]=left[i]
            i+=1
         else:
            lis[k]=right[j]
            j+=1
         k+=1
      while i<len(left):
         lis[k]=left[i]
         i+=1
         k+=1
      while j<len(right):
         lis[k]=right[j]
         j+=1
         k+=1
   print("Merging ",lis)
num=[a for a in input("Enter the list of numbers: ").split(" ")]
mergesort(num)
print(num)
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter the list of numbers:  14 56 98 35 26 89 32
Splitting  ['14', '56', '98', '35', '26', '89', '32']
Splitting  ['14', '56', '98']
Splitting  ['14']
Merging  ['14']
Splitting  ['56', '98']
Splitting  ['56']
Merging  ['56']
Splitting  ['98']
Merging  ['98']
Merging  ['56', '98']
Merging  ['14', '56', '98']
Splitting  ['35', '26', '89', '32']
Splitting  ['35', '26']
Splitting  ['35']
Merging  ['35']
Splitting  ['26']
Merging  ['26']
Merging  ['26', '35']
Splitting  ['89', '32']
Splitting  ['89']
Merging  ['89']
Splitting  ['32']
Merging  ['32']
Merging  ['32', '89']
Merging  ['26', '32', '35', '89']
Merging  ['14', '26', '32', '35', '56', '89', '98']
['14', '26', '32', '35', '56', '89', '98']
```

Page No:
ID: 0201DCS281
Noida Institute of Engineering and Technology

S.No: 41	Exp. Name: ***Python Program for Tower of Hanoi***	Date:

Aim:

Python Program for Tower of Hanoi

Source Code:

towerOfHanoi.py

```
def tower_of_hanoi(disks,source,auxiliary,destination):
   if(disks==1):
      print("Move disk 1 from tower {} to tower {}.".format(source,destination))
      return
   tower_of_hanoi(disks-1,source,destination,auxiliary)
   print("Move disk {} from tower {} to tower {}.".format(disks,source,destination))
   tower_of_hanoi(disks-1,auxiliary,source,destination)
disks=int(input("Enter number of disks: "))
tower_of_hanoi(disks,'A','B','C')
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter number of disks: 4
Move disk 1 from tower A to tower B.
Move disk 2 from tower A to tower C.
Move disk 1 from tower B to tower C.
Move disk 3 from tower A to tower B.
Move disk 1 from tower C to tower A.
Move disk 2 from tower C to tower B.
Move disk 1 from tower A to tower B.
Move disk 4 from tower A to tower C.
Move disk 1 from tower B to tower C.
Move disk 2 from tower B to tower A.
Move disk 1 from tower C to tower A.
Move disk 3 from tower B to tower C.
Move disk 1 from tower A to tower B.
Move disk 2 from tower A to tower C.
Move disk 1 from tower B to tower C.

S.No: 42	Exp. Name: ***Python Program for Bogo Sort or Permutation Sort***	Date:

Aim:

Python Program for Bogo Sort or Permutation Sort

Source Code:

permutationSort.py

```
import random
def bogosort(a):
   n=len(a)
   while(is_sorted(a)==False):
      shuffle(a)
def is_sorted(a):
   n=len(a)
   for i in range(0,n-1):
      if(a[i]>a[i+1]):
         return False
   return True
def shuffle(a):
   n=len(a)
   for i in range(0,n):
      r=random.randint(0,n-1)
      a[i],a[r],=a[r],a[i]
a=list(input("Enter input array: ").split(" "))
bogosort(a)
print("Sorted array :")
for i in range(len(a)):
   print("{}".format(a[i]))
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter input array: 3 2 4 1 0 5
Sorted array :
0
1
2
3
4
5

Test Case - 2
User Output
Enter input array: -23 23 0 12
Sorted array :
-23
0
12
23

S.No: 43	Exp. Name: ***Python program to demonstrate Basic Euclidean Algorithm***	Date:

Aim:

Python program to demonstrate Basic Euclidean Algorithm

Source Code:

euclideanAlgorithm.py

```
def gcd(a,b):
   if a==0:
      return b
   return gcd(b%a,a)
a=int(input("Enter num1: "))
b=int(input("Enter num2: "))
print("gcd(",a,",",b,") = ",gcd(a,b))
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter num1: 10
Enter num2: 15
gcd(10 , 15) = 5

Test Case - 2
User Output
Enter num1: 35
Enter num2: 10
gcd(35 , 10) = 5

S.No: 44	Exp. Name: ***Write the code***	Date:

Aim:

Write Python Program to read and write from a file

1. Open a file **MyFile.txt** in reading mode and print the content of the file to the terminal and close the file
2. Again open the file **MyFile.txt** in append mode and write the word "World" to the file
3. Read the content in the file and observe the ouput

Source Code:

readAndWrite.py

```
z=input("Enter file name: ")
f = open(z,"r")
print(f.read())
f.close()
f=open(z,"a")
f.write("World")
f.close()
f=open(z,"r")
print(f.read())
```

MyFile.txt

```
Hello
```

MyFile2.txt

```
Hi how are you
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter file name: MyFile.txt
Hello
Hello World

S.No: 45	Exp. Name: ***Problem Solving***	Date:

Aim:

Write a program to print the number of **lines**, **words**, and **characters** in a file.

Note: Let us consider the input file as '**InputData2.txt**'.

Source Code:

FileCounts.py

```
fin = open('InputData2.txt' , 'r')
charCount = wordCount = lineCount = 0        #Initialize Counters
for line in fin:                        #Read each Line

   # write your logic here
   line=line.strip(" ")
   #Increment Line count
   word=line.split()
   #split() gives the words in a list
   lineCount+=1
   wordCount+=len(word)
   charCount+=len(line)
   #Increment Character Count

print("Line count = ", lineCount)  #Print the Counts
print("Word count = ", wordCount)
print("Char count = ", charCount)
```

InputData2.txt

```
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
Now is better than never.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Line count = 7
Word count = 51
Char count = 306

S.No: 46	Exp. Name: ***Problem Solving***	Date:

Aim:

Write a Python program to **copy** one file to another file.

Note: Let us consider the input file as '**InputData3.txt**'.

Source Code:

CopyFile1.py

```
fin = open('InputData3.txt', 'r')         #Open the text Files for input and output
fout = open('OutputData3.txt', 'w')       #if the file is big we will read and write li
ne by line.
                        #for each line
                       #Write the line
fout.write(fin.read())
fin.close()             #Close the input file
fout.close()                #Close the output file
                         #Close the input file
                           #Close the output file
fin = open('OutputData3.txt' , 'r')       #Open the new file as input
print(fin.read())                      #for each line
                     #Print line
fin.close()                #Close the File
```

InputData3.txt

```
This is this.
That is this.
That is that.
This is That.
Now tell me what is that?
And what is this?
Confused?
Haha!!
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
This is this.
That is this.
That is that.
This is That.
Now tell me what is that?
And what is this?
Confused?
Haha!!

S.No: 47	Exp. Name: ***Write the code***	Date:

Aim:

Wite a Python program to eliminating repeated lines from a file using Python
Let us see how to delete several repeated lines from a file using Python's File Handling power. If the file is small with a few lines, then the task of deleting/eliminating repeated lines from it could be done manually, but when it comes to large files, this is where Python comes to your rescue.
Approach :
1.Open the input file using using the open() function and pass in the flag -r to open in reading mode.
2.Open an output file, using the -w flag, where we would store the contents of file after deleting all repeated lines from it.
3.Using the set() method keep track of all the lines seen so far, so that we can compare it with the current reading line.
4.Now, iterate over each line of input file and compare it with the lines seen so far.
5.If the current line is also present in lines seen so far, then skip that line else write that line to the output file, and don't forget to add the current line to the lines seen so far.
6.Close the files.

Source Code:

eliminatingLines.py

```
z=input("Enter file name: ")
fin=open(z,"r")
fout=open("output.txt","w")
seen=set()
for line in fin:
   if line not in seen:
      fout.write(line)
      seen.add(line)
fin.close()
fout.close()
print("The new file after removing duplicates is: ")
f=open("output.txt","r")
print(f.read())
```

input.txt

```
A cow is an animal.
A cow is an animal.
A buffalo too is an animal.
Lion is the king of jungle.
```

input1.txt

```
Hi how are you
Hi how are you
Hello fine
Hello fine
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter file name: input.txt
The new file after removing duplicates is:
A cow is an animal.
A buffalo too is an animal.
Lion is the king of jungle.

S.No: 48	Exp. Name: ***To write a python program to find the most frequent words in a text file.***	Date:

Aim:

To write a python program to find the most frequent words in a text file.

Source Code:

freqCount.py

```
z=input("Enter file name: ")
d=[]
frequency=0
frequent=""
fin=open(z,"r")
for line in fin:
   lin_word=line.lower().split(" ")
   for word in lin_word:
      d.append(word)
fin.close()
for i in range(0,len(d)):
   count=0
   for j in range(i,len(d)):
      if(d[i]==d[j]):
         count+=1
   if(count>frequency):
      frequency=count
      frequent=d[i]
print("Most repeated word:",frequent)
print("Frequency:",frequency)
```

demofile.txt

```
"It is better to kill them all in one go. Get explosives in 15 bags and kill them at on
e go. Why should people suffer all this?" the bench comprising justices Arun Mishra and
Deepak Gupta asked Solicitor General Tushar Mehta, who appeared on behalf of the Centr
e.

The SC bench took cognizance of continued stubble burning in the two states and said th
at the life span of millions of citizens has shortened and people are "suffocating" due
to pollution in Delhi-NCR.

"Can you treat people like this and permit them to die due to pollution?" asked the ben
ch.

Pulling up the Punjab chief secretary first, the top court said, "You should explain wh
y we should not impose fine on you for not being able to control your state. Despite ou
r orders, it happened again."

While the chief secretary tried explaining that they are trying to control it, Justice
Mishra asked whether stubble burning is still happening in the state. To this, Punjab c
hief secretary said some instances of stubble burning are still occurring in the state.

"Look at the number of fires. We have called you for that reason. Why is it going on?"
the bench asked.

"Everyone is aware of what is happening in Delhi...Every year its taking place...it wil
l take place next year also if no steps are taken," the bench added.

The Supreme Court also slammed Haryana chief secretary for failing to curb stubble burn
ing activities in the state.

Coming down hard on both the chief secretaries, the Supreme Court bench said, "Life is
n't so cheap in India anymore. What was given [compensation] in Bhopal gas tragedy is n
```

Noida Institute of Engineering and Technology

```
othing compared to what is given to victims in similar cases worldwide. The chief secre
taries have no right to be in chair if they can't serve people well."

The top court also instructed the Centre and Delhi government to keep their differences
aside and prepare a comprehensive plan within 10 days to install air-purifying towers o
n a permanent basis across Delhi to deal with pollution.
```

demofile1.txt

```
hi good morning
how are you
how is your health
Welcome to Hyderabad
Happy coding
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter file name: demofile.txt
Most repeated word: the
Frequency: 21

Page No:
ID: 0201DCS281
Noida Institute of Engineering and Technology

S.No: 49	Exp. Name: ***Writing a class and creating an instance***	Date:

Aim:

Write a program that defines a class `Employee` and add a `pass` **keyword**

- Create an instance **Emp_1** of class `Employee`
- Take string input from user and store it in a variable **name**
- Set Employee **name** to the **user input**
- Set Employee **age** to **25**
- Set Employee **salary** to **25000**
- Print **name** # *print('Emp_1.name: ' , Emp_1.name)*
- Print **salary** # *print('Emp_1.salary:' , Emp_1.salary)*

Sample Input and Output:

```
Please enter an employee name: Devika
Emp_1.name: Devika
Emp_1.salary: 25000
```

Source Code:

ClassesExample2.py

```
class Employee:
   pass
   def __init__(self,name):
      self.name=name
      self.age=25
      self.salary=25000
   def display(self):
      print("Emp_1.name:",self.name)
      print("Emp_1.salary:",self.salary)
a=input("Please enter an employee name: ")
Emp_1=Employee(a)
Emp_1.display()
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Please enter an employee name: Devika
Emp_1.name: Devika
Emp_1.salary: 25000

Test Case - 2
User Output
Please enter an employee name: Kanuka
Emp_1.name: Kanuka
Emp_1.salary: 25000

Test Case - 3
User Output
Please enter an employee name: Renuka
Emp_1.name: Renuka
Emp_1.salary: 25000

Test Case - 4
User Output
Please enter an employee name: Premika
Emp_1.name: Premika
Emp_1.salary: 25000

S.No: 50	Exp. Name: ***Writing a class in Python***	Date:

Aim:

Create a class `Student`. which contains the details of the student like **name**, **age**, **graduation**, take these inputs from the user and print the result as shown in the example.

Sample Input and Output:

```
Enter name of the student:Prasad
Enter age of the student:26
Enter degree of the student:Engineer
Stud_1.name: Prasad
Stud_1.age: 26
Stud_1.graduate: Engineer
```

Source Code:

ClassesExample1.py

```
class Student:
   pass
name=input("Enter name of the student:")
age=input("Enter age of the student:")
deg=input("Enter degree of the student:")
stud_1=Student()
print("Stud_1.name:",name)
print("Stud_1.age:",age)
print("Stud_1.graduate:",deg)
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter name of the student: Srujana
Enter age of the student: 22
Enter degree of the student: MBA
Stud_1.name: Srujana
Stud_1.age: 22
Stud_1.graduate: MBA

Test Case - 2
User Output
Enter name of the student: Krishna
Enter age of the student: 22
Enter degree of the student: Doctor
Stud_1.name: Krishna
Stud_1.age: 22
Stud_1.graduate: Doctor

Test Case - 3
User Output
Enter name of the student: Prasad
Enter age of the student: 26
Enter degree of the student: Engineer
Stud_1.name: Prasad
Stud_1.age: 26
Stud_1.graduate: Engineer

S.No: 51	Exp. Name: ***Writing a class and creating an instance***	Date:

Aim:

Create a class `Employee`, which contains the details of an employee like **name** and **salary.** Take **name** and **salary** as inputs from the console, print the result as shown in the example.

Sample Input and Output:

```
Please enter an employee name: Radha
Enter salary of an employee: 32000
Emp_1.name: Radha
Emp_1.salary: 32000
```

Source Code:

ClassesExample2.py

```
class Employee:
   pass
   def __init__(self,nam,sal):
      self.nam=nam
      self.sal=sal
   def display(self):
      print("Emp_1.name:",self.nam)
      print("Emp_1.salary:",self.sal)
name=input("Please enter an employee name: ")
sal=input("Enter salary of an employee: ")
emp=Employee(name,sal)
emp.display()
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Please enter an employee name: Radha
Enter salary of an employee: 32000
Emp_1.name: Radha
Emp_1.salary: 32000

Test Case - 2
User Output
Please enter an employee name: Sravani
Enter salary of an employee: 25000
Emp_1.name: Sravani
Emp_1.salary: 25000

S.No: 52	Exp. Name: ***Writing a program with methods***	Date:

Aim:

Write a program by following the below steps:

- Define a `class` **Person.**
- Add a `method` **setName()** which takes **self** and **name** as parameters.
- Inside the method, set **self.name = name.**
- Add a `method` **getName()** which takes **self** as parameter.
- Inside the method return **self.name.**
- Create two `instances` **p1** and **p2** of the `class` **Person.**
- Call the `method` **setName()** on **p1** and pass **name** as parameter .
- Call the `method` **setName()** on **p2** and pass **name** as parameter.
- Call the `method` **getName()** on **p1** and print the result.
- Call the `method` **getName()** on **p2** and print the result.

Source Code:

Method_Example1.py

```
class Person:
   i=1
   def setname(self,name):
      self.name=name
   def getname(self):
      print("Person P"+str(Person.i)+" name:",self.name)
      Person.i+=1
p10=input("Enter name of the person1: ")
p20=input("Enter name of the person2: ")
p1=Person()
p2=Person()
p1.setname(p10)
p2.setname(p20)
p1.getname()
p2.getname()
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter name of the person1: Ram
Enter name of the person2: Sita
Person P1 name: Ram
Person P2 name: Sita

Test Case - 2
User Output
Enter name of the person1: Lakshmi
Enter name of the person2: Anusha
Person P1 name: Lakshmi
Person P2 name: Anusha

S.No: 53	Exp. Name: ***Using the `init` method in the class***	Date:

Aim:

Write a program to print the details of the student as shown in the sample test cases.

1. Take the input from the user for **name**, **age** and **email**.
2. Inside the **__init__** method body, **initialise** the arguments with **self.** (Hint: **self.name = name**)
3. Write a statement to print the student details inside the method **studentDetails.**
4. Create an **instance** from the class **Student.**
5. Call the method **studentDetails** to print the output.

Source Code:

ClassesExample4.py

```
class Student:
   def __init__(self, name, age, email):
      self.name=name
      self.age=age
      self.email=email
      # Write your code here

   def studentDetails(self):
      print("Name: "+self.name+" , age: "+self.age+ " , email: "+self.email)
name=input("Enter name of the student: ")
age=input("Enter age of the student: ")
email=input("Enter emial of the student: ")
stud=Student(name,age,email)
stud.studentDetails()      # Write your code here
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter name of the student: Ram
Enter age of the student: 24
Enter emial of the student: abc@gmail.com
Name: Ram , age: 24 , email: abc@gmail.com

Test Case - 2
User Output
Enter name of the student: Ravi
Enter age of the student: 23
Enter emial of the student: abc@gmail.in
Name: Ravi , age: 23 , email: abc@gmail.in

S.No: 54	Exp. Name: ***Write a program to print name and salary of an Employee and print total salary of all Employees.***	Date:

Aim:

Write a program to take name and salary as input from the user and print the details of the employee as shown in the sample test cases.

Source Code:

ClassEmployee.py

```
class Employee:
   def __init__(self, name, salary):
      self.name=name
      self.salary=salary
      # Initialize name and salary of the employee


   def displayEmployee(self):
      print("Name: "+self.name+" , Salary: "+self.salary)
      # Write a function to display employee details
name=input("Enter name of the employee: ")
salary=input("Enter salary of the employee: ")
emp=Employee(name,salary)
emp.displayEmployee()

# Print the details of the employee
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter name of the employee: Prasad
Enter salary of the employee: 35000
Name: Prasad , Salary: 35000

Test Case - 2
User Output
Enter name of the employee: Ram
Enter salary of the employee: 50000
Name: Ram , Salary: 50000

S.No: 55	Exp. Name: ***Write a program for Multiple Inheritance.***	Date:

Aim:

Write a program to understand **Multiple Inheritance.**

Fill the **missing code** in the below program by following the below instructions:

- Create a `base class` **vehicle** with `__init__` method, and pass **self**, **name**, **price** and **regno** as arguments
- Derive a `class` **car** from `base class` **vehicle**
- Define a method `__init__` in the `class` **car** which contains the `parameters` **self**, **name**, **price**, **regno** and **gear**
- Derive a `class` **boat** from `class` **vehicle**
- Derive a `class` **hover** from `class` **car** and **boat**

Sample Input and Output:

```
car      toyota      1500000     car2121     auto
boat     maruti      1000000     boat0121
hover    toyota      1500000     hover1212   manual
class car inherits from class vehicle
```

Source Code:

MultipleInher.py

```
class vehicle:
   '''General Vehicle class'''
   def __init__(self,name,price,regno):
      self.name=name
      self.price=price
      self.regno=regno
# write your code here


class car(vehicle):
   ''' Class car inherits from Vehicle'''
   def __init__(self,name,price,regno,gear):
      self.name=name
      self.price=price
      self.regno=regno
      self.gear=gear

# write your code here

class boat(vehicle):
   ''' Class boat inherits from vehicle'''
   pass

# write your code here

class hover(car, boat):
   '''Class hovercraft inherits from both car and boat'''
   pass
c1 = car('toyota', 1500000, 'car2121', 'auto')
b1 = boat('maruti', 1000000, 'boat0121')
h1 = hover('toyota', 1500000, 'hover1212', 'manual')
print(type(c1).__name__, "\t", c1.name, "\t", c1.price, "\t", c1.regno, "\t", c1.gear)
print(type(b1).__name__, "\t", b1.name, "\t", b1.price, "\t", b1.regno, "\t")
print(type(h1).__name__, "\t", h1.name, "\t", h1.price, "\t", h1.regno, "\t", h1.gear)
print(c1.__doc__)
```

Execution Results - All test cases have succeeded!

Test Case - 1				
User Output				
car	toyota	1500000	car2121	auto
boat	maruti	1000000	boat0121	
hover	toyota	1500000	hover1212	manual
Class car inherits from Vehicle				

S.No: 56	Exp. Name: ***Understanding Inheritance, Multiple Inheritance and Method Overriding***	Date:

Aim:

Write a program that implements `method overriding` following the below steps:

1. Define `class` **Animal**
2. Use `constructor` to set the **name** with a default value = **"This Animal"**
3. Define a `method` **eat** with a parameter **food** with a default value = **"Grass"**
4. Inside the method **print (self.name, " eats", food)**
5. Define a `class` **Mammal**, inherit from **Animal**
6. Inside the class, override **eat** method to **print(self.name, " does not eat. It only drinks")**
7. Define `class` **WingedAnimal**, inherit from **Animal**
8. Override **eat** method to **print(self.name," eats anything and everything")**
9. Define a `class` called **Bat**, inherit from **WingedAnimal**, **Mammal**
10. Define `method` **smell**, which prints **"This Animal Stinks"**
11. Define a `class` called **FruitBat**, inherit from **Mammal**, **WingedAnimal** (Notice the Order)
12. rabbit1 = Animal("Rabbit")
13. print("Rabbit1 is an instance of Animal")
14. rabbit1.eat() # *Animal's eat method without food parameter*
15. rabbit1.eat("Peanuts") # *Animal's eat method with food parameter*
16. print("Cow1 is an instance of Mammal")
17. cow1 = Mammal("Cow")
18. cow1.eat() # *Mammal's eat method*
19. print("Vulture1 is an instance of WingedAnimal")
20. vulture1 = WingedAnimal("Vulture")
21. vulture1.eat() # *WingedAnimal's eat method*
22. print("Bat1 is an instance of Bat")
23. bat1 = Bat("Bat")
24. bat1.eat() # *WingedAnimal's eat method*
25. print("fbat is an instance of FruitBat")
26. fbat = FruitBat("Fruitbat")
27. fbat.eat() # *Mammal's eat method*

Sample Input and Output:

```
Rabbit1 is an instance of Animal
Rabbit eats Grass
Rabbit eats Peanuts
Cow1 is an instance of Mammal
Cow1 does not eat. It only drinks
Vulture1 is an instance of WingedAnimal
Vulture eats anything and everything
Bat1 is an instance of Bat
Bat eats anything and everything
fbat is an instance of FruitBat
Fruitbat does not eat. It only drinks
```

Source Code:

Overiding_Example2.py

```
class Animal():
   def __init__(self,name):
      self.name=name
   def eat(self,food="Grass"):
      self.food=food
      print(self.name,"eats",self.food)
class Mammal(Animal):
   pass
   def eat(self):
      print(self.name,"does not eat. It only drinks")
class WingedAnimal(Animal):
   pass
   def eat(self):
      print(self.name,"eats anything and everything")
class Bat(WingedAnimal,Mammal):
   pass
```

```
def smell(self):
   print("Animal Stinks")
class FruitBat(Mammal,WingedAnimal):
   pass
print("Rabbit1 is an instance of Animal")
rabbit1=Animal("Rabbit")
rabbit1.eat()
rabbit1.eat("Peanuts")
print("Cow1 is an instance of Mammal")
Cow1=Mammal("Cow")
Cow1.eat()
print("Vulture1 is an instance of WingedAnimal")
Vulture1=WingedAnimal("Vulture")
Vulture1.eat()
print("Bat1 is an instance of Bat")
Bat1=Bat("Bat")
Bat1.eat()
print("fbat is an instance of FruitBat")
fbat=FruitBat("Fruitbat")
fbat.eat()
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Rabbit1 is an instance of Animal
Rabbit eats Grass
Rabbit eats Peanuts
Cow1 is an instance of Mammal
Cow does not eat. It only drinks
Vulture1 is an instance of WingedAnimal
Vulture eats anything and everything
Bat1 is an instance of Bat
Bat eats anything and everything
fbat is an instance of FruitBat
Fruitbat does not eat. It only drinks

ID: 0201DCS281 Page No:

Noida Institute of Engineering and Technology