SRI VASAVI INSTITUTE OF ENGINEERING & TECHNOLOGY

2020-21 1st Semester

LABORATORY MASTER MANUAL

of

PYTHON PROGRAMMING LAB

for

II B.Tech CSE

Prepared by

A.Pavan Kumar

Associate Professor

DEPARTMENT OF

COMPUTER SCIENCE & ENGINEERING

II Year – I SEMESTER	L	T	P	C
	0	0	3	1.5

Python Programming Lab

1. Write a program that asks the user for a weight in kilograms and converts it to pounds. There are 2.2 pounds in a kilogram.

2. Write a program that asks the user to enter three numbers (use three separate input statements). Create variables called total and average that hold the sum and average of the three numbers and print out the values of total and average.
3. Write a program that uses a *for* loop to print the numbers 8, 11, 14, 17, 20, . . . , 83, 86, 89.
4. Write a program that asks the user for their name and how many times to print it. The program should print out the user's name the specified number of times.
5. Use a *for* loop to print a triangle like the one below. Allow the user to specify how high the triangle should be.

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

6. Generate a random number between 1 and 10. Ask the user to guess the number and print a message based on whether they get it right or not.
7. Write a program that asks the user for two numbers and prints *Close* if the numbers are within .001 of each other and Not close otherwise.
8. Write a program that asks the user to enter a word and prints out whether that word contains any vowels.
9. Write a program that asks the user to enter two strings of the same length. The program should then check to see if the strings are of the same length. If they are not, the program should print an appropriate message and exit. If they are of the same length, the program should alternate the characters of the two strings. For example, if the user enters *abcde* and *ABCDE* the program should print out *AaBbCcDdEe*.
10. Write a program that asks the user for a large integer and inserts commas into it according to the standard American convention for commas in large numbers. For instance, if the user enters 1000000, the output should be 1,000,000.
11. In algebraic expressions, the symbol for multiplication is often left out, as in 3x+4y or 3(x+5). Computers prefer those expressions to include the multiplication symbol, like 3*x+4*y or 3*(x+5). Write a program that asks the user for an algebraic expression and then inserts multiplication symbols where appropriate.

12. Write a program that generates a list of 20 random numbers between 1 and 100.
    a) Print the list.
    b) Print the average of the elements in the list.
    c) Print the largest and smallest values in the list.
    d) Print the second largest and second smallest entries in the list
    e) Print how many even numbers are in the list.
13. Write a program that asks the user for an integer and creates a list that consists of the factors of that integer.
14. Write a program that generates 100 random integers that are either 0 or 1. Then find the longest run of zeros, the largest number of zeros in a row. For instance, the longest run of zeros in [1,0,1,1,0,0,0,0,1,0,0] is 4.
15. Write a program that removes any repeated items from a list so that each item appears at most once. For instance, the list [1,1,2,3,4,3,0,0] would become [1,2,3,4,0].
16. Write a program that asks the user to enter a length in feet. The program should then give the user the option to convert from feet into inches, yards, miles, millimeters, centimeters, meters, or kilometers. Say if the user enters a 1, then the program converts to inches, if they enter a 2, then the program converts to yards, etc. While this can be done with if statements,it is much shorter with lists and it is also easier to add new conversions if you use lists.
17. Write a function called *sum_digits* that is given an integer num and returns the sum of the digits of num.
18. Write a function called *first_diff* that is given two strings and returns the first location in which the strings differ. If the strings are identical, it should return -1.
19. Write a function called *number_of_factors* that takes an integer and returns how many factors the number has.
20. Write a function called *is_sorted* that is given a list and returns True if the list is sorted and False otherwise.
21. Write a function called root that is given a number x and an integer n and returns $x^{1/n}$. In the function definition, set the default value of n to 2.
22. Write a function called primes that is given a number n and returns a list of the first n primes. Let the default value of n be 100.
23. Write a function called merge that takes two already sorted lists of possibly different lengths, and merges them into a single sorted list.
    a. Do this using the sort method. (b) Do this without using the sort method.

24. Write a program that asks the user for a word and finds all the smaller words that can be made from the letters of that word. The number of occurrences of a letter in a smaller word can't exceed the number of occurrences of the letter in the user's word.
25. Write a program that reads a file consisting of email addresses, each on its own line. Your program should print out a string consisting of those email addresses separated by semicolons.
26. Write a program that reads a list of temperatures from a file called *temps.txt*, converts those temperatures to Fahrenheit, and writes the results to a file called ftemps.txt.
27. Write a class called Product. The class should have fields called name, amount, and price, holding the product's name, the number of items of that product in stock, and the regular price of the product. There should be a method *get_price* that receives the number of items to be bought and returns a the cost of buying that many items, where the regular price is charged for orders of less than 10 items, a 10% discount is applied for orders of between 10 and 99 items, and a 20% discount is applied for orders of 100 or more items. There should also be a method called *make_purchase* that receives the number of items to be bought and decreases amount by that much.

28. Write a class called Time whose only field is a time in seconds. It should have a method called *convert_to_minutes* that returns a string of minutes and seconds formatted as in the following example: if seconds is 230, the method should return '5:50'. It should also have a method called *convert_to_hours* that returns a string of hours, minutes, and seconds formatted analogously to the previous method.

29. Write a class called Converter. The user will pass a length and a unit when declaring an object from the class—for example, c = Converter(9,'inches'). The possible units are inches, feet, yards, miles, kilometers, meters, centimeters, and millimeters. For each of these units there should be a method that returns the length converted into those units. For example, using the Converter object created above, the user could call c.feet() and should get 0.75 as the result.
30. Write a Python class to implement pow(x, n).
31. Write a Python class to reverse a string word by word.
32. Write a program that opens a file dialog that allows you to select a text file. The program then displays the contents of the file in a textbox.
33. Write a program to demonstrate Try/except/else.
34. Write a program to demonstrate try/finally and with/as.

Exercise 1 -

AIM:

Write a program that asks the user for a weight in kilograms and converts it to pounds. There are 2.2 pounds in a kilogram.

Algorithm:

Step1: Start
Step2: Read the weight in kilograms using input() function and convert it to number using eval() or int() or float() function and store it into a variable(say wtkg)
Step3: convert the weight in kilograms to pounds by multiplying it with 2.2 and store it into a variable say (wtpd)
Step4: print the weight in pounds
Step5: Stop

Program:

```
wtkg=eval(input("enter weight in kilograms:"))
wtpd= 2.2*wtkg
print(wtkg,"kgs= ",wtpd,"pounds")
```

Output:

```
enter weight in kilograms:35
35 kgs=  77.0 pounds
```

Exercise 2 -

AIM:

Write a program that asks the user to enter three numbers (use three separate input statements). Create variables called total and average that hold the sum and average of the three numbers and print out the values of total and average.

Algorithm:

Step1: Start
Step2: Read the first number using input() function and convert it to a number and store it into a variable (say a)
Step3: Read the second number using input() function and convert it to a number and store it into a variable (say b)
Step4: Read the Third number using input() function and convert it to a number and store it into a variable (say c)
Step5: perform a+b+c and store it in a variable(say total)
Step6: perform total/3 and store it in a variable (say average)
Step7: Print total and average
Step8: Stop

Program:

```
a=eval(input("enter first number:"))
b=eval(input("enter second number:"))
c=eval(input("enter third number:"))
total=(a+b+c)
average=total/3
print("sum of three numbers is:",total)
print("average of three numbers is:",average)
```

Output:

```
enter first number:34
enter second number:24
enter third number:14
sum of three numbers is: 72
average of three numbers is: 24.0
```

Exercise 3 -

AIM:

Write a program that uses a *for* loop to print the numbers 8, 11, 14, 17, 20, . . . , 83, 86, 89.

Algorithm:

Step1: Start
Step2: iterate using the for loop and range() function starting from 8 to 89 with step value 3 and print the number one by one (or initialize the variable with 8 and print the variable value and increment the value by 3 in each iteration using while loop)

Program:

```
for i in range(8,92,3):
   print(i,end=" ")
```

Output:

```
8 11 14 17 20 23 26 29 32 35 38 41 44 47 50 53 56 59 62 65 68 71 74 77 80 83 86 89
```

Exercise 4 -

AIM:

Write a program that asks the user for their name and how many times to print it. The program should print out the user's name the specified number of times.

Algorithm:

Step1: Start
Step2: read the user name using input() function and store it into a variable (say name)
Step3: read the integer that indicate the number of times to print their name using input() and int() functions and store it into a variable(say count)
Step4: iterate using the for loop (count number of times) using range() function and print the name in each iteration
Step5: Stop

Program:

```
name=input("enter your name:")
count=int(input("enter how many times you want to print your name:"))
for i in range(count):
    print(name)
```

Output:

```
enter your name:sviet
enter how many times you want to print your name:5
sviet
sviet
sviet
sviet
sviet
```

Exercise 5 -

AIM:

Use a *for* loop to print a triangle like the one below. Allow the user to specify how high the triangle should be.

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

Algorithm:

Step1: Start
Step2: Iterate using the for loop 4 times(since there are 4 rows) using range() function and print * using repetition operator (i+1) times
Step3: Stop

Program:

```
for i in range(4):
   print('*'*(i+1))
```

Output:

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

Exercise 6 -

AIM:

Generate a random number between 1 and 10. Ask the user to guess the number and print a message based on whether they get it right or not.

Algorithm:

Step1: Start
Step2: import the random module
Step3: generate a random number between 1 and 10 using randint() method of random module and store it into a variable (say num)
Step4: read the number that indicate the user guess using input() and int() functions and store it into a variable(say guess)
Step5: if num value is equal to guess value then print the message "you guessed right" else print "sorry you guessed wrong"
Step6: Stop

Program:

```
import random
num = random.randint(1,10)
guess = eval(input('Enter your guess: '))
if guess==num:
   print('You guessed right!')
else:
   print('Sorry you guessed wrong.  The random number is ', num)
```

Output1:

```
Enter your guess: 5
Sorry you guessed wrong.  The random number is  3
```

Output2:

```
Enter your guess: 2
You guessed right!
```

Exercise 7 -

AIM:

Write a program that asks the user for two numbers and prints *Close* if the numbers are within .001 of each other and Not close otherwise

Algorithm:

Step1: Start
Step2: read the first number using input() function and float() functions and store it into a variable (say num1)
Step3: read the second number using input() and float() functions and store it into a variable (say num2)
Step4: check whether the numbers are within .001 of each other by adding .001 to the number to get other number
Step5: If the numbers are within .001 of each other then print the message "close" otherwise print the message "not close"
Step6: Stop

Program:

```
num1 = float(input("Enter number 1 : "))
num2 = float(input("Enter number 2 : "))
if num1 + 0.001 == num2:
    print("close")
elif num2 + 0.001 == num1:
    print("Close")
else:
    print("Not  Close")
```

Output1:

```
Enter number 1 : 3
Enter number 2 : 3.001
Close
```

Output2:

```
Enter number 1 : 5
Enter number 2 : 7
Not  Close
```

Exercise 8 -

AIM:

Write a program that asks the user to enter a word and prints out whether that word contains any vowels.

Algorithm:

Step1: Start
Step2: read the word using input() function and store it into a variable (say word)
Step3: set the flag variable to zero.
Step4: Iterate over word using for loop and check whether any character is vowel or not. If any character is vowel then set the flag variable to one.
Step5: if flag variable is one then print the message "the word contain vowels" else print "the word doesn't contain vowels.
Step6: Stop

Program:

```
word=input("enter a word:")
flag=0
for i in word:
   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'):
      flag=1
if(flag==1):
   print("the given word",word,"contains vowels")
else:
   print("the given word",word,"doesn't contain vowels")
```

Output1:

```
enter a word:sviet
the given word sviet contains vowels
```

Output2:

```
enter a word:lynx
the given word lynx doesn't contain vowels
```

Exercise 9 -

AIM:

Write a program that asks the user to enter two strings of the same length. The program should then check to see if the strings are of the same length. If they are not, the program should print an appropriate message and exit. If they are of the same length, the program should alternate the characters of the two strings. For example, if the user enters *abcde* and *ABCDE* the program should print out *AaBbCcDdEe*.

Algorithm:

Step1: Start
Step2: read the first string using input() function and store it into a variable (say str1)
Step3: read the second string using input() function and store it into a variable (say str2)
Step4: find the lengths of two strings using len() function. If lengths of two strings doesn't equal Then print the message "lengths of two strings doesn't match" else goto Step 5
Step5: Take the empty string new. Now iterate using the for loop using range() function from 0 to len(str1) and in each iteration do the following
    i) Extract one character from str1 and concatenate to the string new
    ii) Extract one character from str2 and concatenate to the string new
Step6: After entire iterations are completed, print the string new
Step7: Stop

Program:

```
str1=input("enter the first string: ")

str2=input("enter the second string: ")

if(len(str1)!=len(str2)):

   print("lengths of two strings doesn't match")

else:
   new=""
   for i in range(0,len(str1)):

      new=new+str1[i]

      new=new+str2[i]

   print(new)
```

Output1:

```
enter the first string: abcde
enter the second string: ABCDE
aAbBcCdDeE
```

Output2:

```
enter the first string: sviet
enter the second string: Nandamuru
lengths of two strings doesn't match
```

Exercise 10 -

AIM:

Write a program that asks the user for a large integer and inserts commas into it according to the standard American convention for commas in large numbers. For instance, if the user enters 1000000, the output should be 1,000,000.

Algorithm:

Step1: Start
Step2: Read the large integer using input() function and convert it to number using eval() or int() function and store it into a variable(say number)
Step3: Apply the formatting to the number using format() function by comma separator and using the format specifier d and print the number
Step4: Stop

Program:

```
number = eval(input('enter a number:'))
print("the required format of the number",number,"is",format(number,',d'))
```

Output1:

```
enter a number:1000000
the required format of the number 1000000 is 1,000,000
```

Output2:

```
enter a number:1234567890
the required format of the number 1234567890 is 1,234,567,890
```

Exercise 11 -

AIM:

In algebraic expressions, the symbol for multiplication is often left out, as in 3x+4y or 3(x+5). Computers prefer those expressions to include the multiplication symbol, like 3*x+4*y or 3*(x+5). Write a program that asks the user for an algebraic expression and then inserts multiplication symbols where appropriate.

Algorithm:

Step1: Start
Step2: Read the valid algebraic expression using input() function and store it into a variable (say s)
Step3: Convert the string to the list (say l) using list() function
Step4: Now iterate using the entire list from first character of the string to the last before character of the list using range() function and in each iteration do the following
i) If the character at index i in list l is either digit or alphabet then if the character at index i+1 is either alphabet or left paranthesis then insert * at the i+1 index using insert() method of the list
Step5: now join the characters of the list using join method and store the string into a variable (say s1) and print the string s1
Step6: Stop

Program:

```
s=input("enter valid algebraic expression:")
l=list(s)
for i in range(0,len(l)):
    if(l[i].isdigit()or l[i].isalpha()):
        if(l[i+1].isalpha()or l[i+1]=='('):
            l.insert(i+1,'*')
s1=''.join(l)
print("the required expression is:",s1)
```

Output1:

```
enter valid algebraic expression:3x+4y
the required expression is: 3*x+4*y
```

Output2:

```
enter valid algebraic expression:3(x+5)
the required expression is: 3*(x+5)
```

Exercise 12 -

AIM:

Write a program that generates a list of 20 random numbers between 1 and 100.
(a) Print the list.
(b) Print the average of the elements in the list.
(c) Print the largest and smallest values in the list.
(d) Print the second largest and second smallest entries in the list
(e) Print how many even numbers are in the list.

Algorithm:

Step1: Start
Step2: import the random module and initialize the empty list to the variable (say list1)
Step3: now iterate using the for loop using range() function 20 times and in each iteration append the random integer between 1 and 100 generated by randint() method of random module using append() method of the list and print the list
Step4: find the sum of the elements of the list using sum() method and total no. of elements of the list using len() method and perform division over one another to get the average of the elements of the list. print the average of the elements of the list
Step5: to find the largest, smallest, second largest and second smallest elements of the list do the following

i) Sort the elements of the list using sort() method
ii) Now print the largest element which is the last element of the list
iii) Print the smallest element which is the first element of the list
iv) Print the second largest element which is the second element from the end of the list
v) Print the second smallest element which is the second element from the beginning of the list

Step6: To find the total number of even numbers of the list does the following

i) Initialize the count variable to zero
ii) Iterate over the list using for loop and in each iteration if the element of the list is even number then increment the count variable by one
iii) After the last iteration ,print the count value

Step7: Stop

Program:

```
import random
list1=[]
for i in range(20):
    list1.append(random.randint(1,100))
print("the list is:",list1)

#finding the average of the elements of the list
average=sum(list1)/len(list1)
print("the average of the elements in the list is:",average)

#finding two largest and two smallest elements of the list
list1.sort()
print("the largest element of the list is:",list1[-1])
print("the smallest element of the list is:",list1[0])
print("the second largest element of the list is:",list1[-2])
print("the second smallest element of the list is:",list1[1])

#finding the total number of even numbers of the list
count=0
for num in list1:
    if num%2==0:
        count=count+1
print("the total number of even numbers of the list are:",count)
```

Output1:

the list is: [59, 5, 26, 74, 5, 85, 69, 93, 17, 57, 35, 81, 62, 96, 16, 78, 86, 73, 68, 38]
the average of the elements in the list is: 56.15
the largest element of the list is: 96
the smallest element of the list is: 5
the second largest element of the list is: 93
the second smallest element of the list is: 5
the total number of even numbers of the list are: 9

Output2:

the list is: [30, 1, 54, 15, 21, 91, 91, 47, 89, 14, 23, 25, 13, 45, 74, 23, 44, 19, 11, 60]
the average of the elements in the list is: 39.5
the largest element of the list is: 91
the smallest element of the list is: 1
the second largest element of the list is: 91
the second smallest element of the list is: 11
the total number of even numbers of the list are: 6

Exercise 13 -

AIM:

Write a program that asks the user for an integer and creates a list that consists of the factors of that integer.

Algorithm:

Step1: Start
Step2: Read the integer using input() function and convert it to number using eval() or int() function and store it into a variable(say num)
Step3: Initialize the empty list to the variable (say list1)
Step4: Now iterate using the for loop using the iterating variable 'i' in the range(1,num+1) and in each iteration do the following
    i) If i divides num then i is a factor of num. So append i to the list1 using the append() method
Step5: print the list1 that contains the factors of the given integer
Step6: Stop

Program:

```
num=int(input("enter an integer:"))

list1=[]

for i in range(1,num+1):

   if(num%i)==0:

      list1.append(i)

print("the list consisting of factors of the integer are:",list1)
```

Output1:

```
enter an integer:20
the list consisting of factors of the integer are: [1, 2, 4, 5, 10, 20]
```

Exercise 14 -

AIM:

Write a program that generates 100 random integers that are either 0 or 1. Then find the longest run of zeros, the largest number of zeros in a row. For instance, the longest run of zeros in [1,0,1,1,0,0,0,0,1,0,0] is 4.

Algorithm:

Step1: Start
Step2: import the random module and initialize the empty list to the variable (say l)
Step3: Call the function my_list() that creates the list and returns the list
Step4: Call the function largest_row_of_zeros() that takes the list as a parameter and returns the largest run of zeros, largest number of zeros in a row.
Step5: Print the value returned by the function largest_row_of_zeros()
Step6: Stop

Algorithm for writing my_list() function:

Step1: Iterate using the for loop using range() function 100 times and in each iteration append the random integer between 0 and 1 generated by randint() method of random module using append() method of the list to the list l
Step2: print the list l
Step3: Return the list l

Algorithm for writing largest_row_of_zeros() function:

This function takes the list l as a parameter.

Step1: Initialize the variables c and max_count to zero
Step2: Iterate the over the list l using the iterating variable j and in each iteration do the following
    i) If j==0 then add 1 to c and store it in c else set c to 0
    ii) If c> max_count then set max_count = c
Step3: return the max_count value

Program:

```
import random

l = []

def my_list():
    for j in range(0,100):
        x = random.randint(0,1)
        l.append(x)
    print (l)
    return l

def largest_row_of_zeros(l):

    c = 0
    max_count = 0

    for j in l:
        if j==0:
            c+=1
        else:
            c = 0
        if c > max_count:
            max_count = c

    return max_count

l = my_list()
print("the longest run of zeros in the list is",largest_row_of_zeros(l))
```

Output:

```
[0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0,
1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1,
1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0]
the longest run of zeros in the list is 8
```

Exercise 15 -

AIM:

Write a program that removes any repeated items from a list so that each item appears at most once. For instance, the list [1,1,2,3,4,3,0,0] would become [1,2,3,4,0].

Algorithm:

Step1: Start
Step2: Initialize the empty list to the variable (say list1)
Step3: Read the total number of elements of the list using input() function and convert it to integer using eval() or int() function and store it in a variable (say n)
Step4: Iterate using for loop using range(n) and in each iteration do the following
- i) Read the number using input() function and convert it to number using eval() or int() or float() and store it in a variable(say num)
- ii) Append the num element to the list list1 using append() method

Step5: print the list list1
Step6: Initialize the empty list to the variable (say list1)
Step7: Iterate using for loop over list1 using iterating variable i and in each iteration do the following
- i) If i is not an element of list2 then append the element 'i' to the list2 using append() method

Step8: print the list2
Step9: Stop

Program:

```
list1=[]
n=eval(input("enter the total number of elements of the list:"))
for i in range(n):
   num=eval(input("enter the number:"))
   list1.append(num)
print("the list is:",list1)

# creating another list with unique elements
# declare another list
list2 = []

# appending elements
for i in list1:
        if i not in list2:
                list2.append(i)
```

```
print("the required list is:",list2)
```

Output:

```
enter the total number of elements of the list:8
enter the number:1
enter the number:1
enter the number:2
enter the number:3
enter the number:4
enter the number:3
enter the number:0
enter the number:0
the list is: [1, 1, 2, 3, 4, 3, 0, 0]
the required list is: [1, 2, 3, 4, 0]
```

Exercise 16 -

AIM:

Write a program that asks the user to enter a length in feet. The program should then give the user the option to convert from feet into inches, yards, miles, millimeters, centimeters, meters, or kilometers. Say if the user enters a 1, then the program converts to inches, if they enter a 2, then the program converts to yards, etc. While this can be done with if statements,it is much shorter with lists and it is also easier to add new conversions if you use lists.

Algorithm:

Step1: Start
Step2: Read the length in feet using input() function and convert it to number using eval() or int() function and store it in a variable (say feet)
Step3: display the following menu string using print() function
"""" choose 1 to convert into inches,
choose 2 to convert into yards,
choose 3 to convert into miles,
choose 4 to convert into millimeters,
choose 5 to convert into centimeters,
choose 6 to convert into meters,
choose 7 to convert into kilometers"""
Step4: Read the option (an integer between 1 and 7) using input() function and convert it to integer using eval() or int() function and store it in a variable (say option)
Step5: convert feet to inches by multiplying feet with 12 and store it in a variable(say inch)
Step6: convert feet to yards by multiplying feet with 0.33333 and store it in a variable(say yard)
Step7: convert feet to miles by multiplying feet with 0.000189393939 and store it in a variable(say mile)
Step8: convert feet to millimetres by multiplying feet with 304.8 and store it in a variable(say millimeter)
Step9: convert feet to centimetres by multiplying feet with 30.48 and store it in a variable(say centimeter)
Step10: convert feet to meters by multiplying feet with 0.3048 and store it in a variable(say meter)
Step11: convert feet to inches by multiplying feet with 0.0003048 and store it in a variable(say kilometer)
Step12: Initialize the list (say convert) with variables feet, inch, yard, mile, millimeter, centimeter, meter, kilometre
Step13: print convert[option] to get the required answer

Program:

```
feet = int(input("Enter a length in feet: "))
print(""" choose 1 to convert into inches,
        choose 2 to convert into yards,
        choose 3 to convert into miles,
        choose 4 to convert into millimeters,
        choose 5 to convert into centimeters,
        choose 6 to convert into meters,
        choose 7 to convert into kilometers""")
option = int(input("enter an option between 1 and 7: "))
inch = feet * 12
yard = feet * 0.33333
mile = feet * 0.000189393939
millimeter = feet * 304.8
centimeter = feet * 30.48
meter = feet * 0.3048
kilometer = feet * 0.0003048
required=["feet","inches","yards","miles","millimeters","centimeters","meters","kilometers"]
convert = [
            feet,
            inch,
            yard,
            mile,
            millimeter,
            centimeter,
            meter,
            kilometer
            ]

print("the length in",required[option],"is",convert[option])
```

Output:

```
Enter a length in feet: 5
        choose 1 to convert into inches,
        choose 2 to convert into yards,
        choose 3 to convert into miles,
        choose 4 to convert into millimeters,
        choose 5 to convert into centimeters,
        choose 6 to convert into meters,
        choose 7 to convert into kilometers
enter an option between 1 and 7: 4
the length in millimeters is 1524.0
```

Exercise 17 -

AIM:

Write a function called *sum_digits* that is given an integer num and returns the sum of the digits of num.

Algorithm:

Step1: Start
Step2: Read the integer using input() function and convert it to integer using int() or eval() function and store it into a variable (say number)
Step3: Call the function sum_digits with number as a parameter
Step4: print the value returned by the function sum_digits()
Step5: Stop

Algorithm for writing sum_digits () function:

This function takes the integer num as a parameter and returns the sum of the digits of the integer num

Step1: Initialize sum to zero
Step2: repeatedly perform the following steps(i)-(iii) until num>0 using while loop
i) reminder = num % 10
ii) sum = sum + reminder
iii) num = num //10
Step3: Return the sum

Program:

```
def sum_digits(num):
    sum = 0

    while(num > 0):
        reminder = num % 10
        sum = sum + reminder
        num = num //10

    return sum

number = int(input("Please Enter any Number: "))
sum=sum_digits(number)
print("sum of the digits of the number",number,"is",sum)
```

Output:

Please Enter any Number: 23453
sum of the digits of the number 23453 is 17

Exercise 18 -

AIM:

Write a function called *first_diff* that is given two strings and returns the first location in which the strings differ. If the strings are identical, it should return -1.

Algorithm:

Step1: Start
Step2: Read the first string using input() function and store it into a variable (say s1)
Step3: Read the second string using input() function and store it into a variable (say s2)
Step4: Call the function first_diff() with s1 and s2 as parameters
Step5: store the return value of the function into the variable (say loc)
Step6: if value of the loc is -1 then print " two strings are identical" and print loc else print the loc value which indicates the first location at which the strings differ.
Step7: Stop

Algorithm for writing first_diff() function:

This function takes the strings str1 and str2 as parameters and returns the first location in which the strings differ if the two strings are not identical and returns -1if strings are identical

Step1: Initialize i=len(str1)
Step2: Initialize j=len(str2)
Step3: Initialize k=0
Step4: repeatedly perform the following using while loop until characters of the two strings at the index k is same and k<i and k<j
    i) k=k+1
Step5: if k==i and k==j(all the characters of the two strings are examined) then return -1 else return k+1(the first location where two strings differ)

Program:

```
def first_diff(str1,str2):
    i=len(str1)
    j=len(str2)
    k=0
    while((k<i) and (k<j)and(str1[k]==str2[k])):
        k=k+1
    if(k==i) and (k==j):
        return -1
    else:
        return k+1
```

```
s1=input("enter first string:")
s2=input("enter second string:")
loc=first_diff(s1,s2)
if(loc==-1):
   print("strings are identical and return value of the function is",loc)
else:
   print("The first location at which strings differ is",loc)
```

Output1:

```
enter first string:SVIET
enter second string:SVIET
strings are identical and return value of the function is -1
```

Output2:

```
enter first string:SVIET
enter second string:SViet
The first location at which strings differ is 3
```

Exercise 19 -

AIM:

Write a function called *number_of_factors* that takes an integer and returns how many factors the number has.

Algorithm:

Step1: Start
Step2: Read the integer using input() function and convert it to integer using int() or eval() function and store it into a variable (say num)
Step3: Call the function number_of_factors() with num as a parameter
Step4: print the value returned by the function number_of_factors()
Step5: Stop

Algorithm for writing number_of_factors() function:

This function takes the integer n as a parameter and returns the factors of the integer n

Step1: Initialize list1 to the empty list
Step2: Iterate using the for loop using the iterating variable i in the range (1,n+1) using the range() function and in each iteration do the following
    i) if i divides n , then i is a factor of n. so append the element i to the list1 using append() method
Step3: print the list1that contains the factors of the integer n
Step4: return the len(list1) that contains no. of factors of the integer n

Program:

```
def number_of_factors(n):
    list1=[]
    for i in range(1,n+1):
        if(n%i)==0:
            list1.append(i)
    print("the list consisting of factors of the integer are:",list1)
    return len(list1)

num=int(input("enter an integer:"))
nf=number_of_factors(num)
print("the number of factors that",num,"has are",nf)
```

Output1:

enter an integer:10
the list consisting of factors of the integer are: [1, 2, 5, 10]
the number of factors that 10 has are 4

Output2:

enter an integer:100
the list consisting of factors of the integer are: [1, 2, 4, 5, 10, 20, 25, 50, 100]
the number of factors that 100 has are 9

Exercise 20 -

AIM:

Write a function called *is_sorted* that is given a list and returns True if the list is sorted and False otherwise.

Algorithm:

Step1: Start
Step2: Initialize the empty list to the variable (say l)
Step3: Read the total number of elements of the list using input() function and convert it to integer using eval() or int() function and store it in a variable (say n)
Step4: Iterate using for loop in the range range(n) and in each iteration do the following
iii) Read the number using input() function and store it in a variable(say ele)
iv) Append the ele element to the list l using append() method
Step5: call the function is_sorted() function with l as the parameter
Step6: Print the Boolean value returned by the function is_sorted() that indicates whether the list is sorted or not(if the value is True then the list is sorted otherwise list is not sorted)
Step7: Stop

Algorithm for writing is_sorted() function:

This function takes the list (say a_list) as a parameter and returns the Boolean value

Step1: compare the list a_list with sorted() built-in function. If both are same, then return True(i.e., list is sorted) else return False(i.e., list is not sorted)

Program:

```
def is_sorted(a_list):
    if a_list == sorted(a_list):
        return True
    return False

l=[]
n=int(input("enter the number of elements of the list:"))
for i in range(n):
    ele=input("enter the element:")
    l.append(ele)
print("the list is",l)
print("Is list is sorted:",is_sorted(l))
```

Output1:

```
enter the number of elements of the list:5
enter the element:1
enter the element:2
enter the element:3
enter the element:4
enter the element:5
the list is ['1', '2', '3', '4', '5']
Is list is sorted: True
```

Output2:

```
enter the number of elements of the list:3
enter the element:pavan
enter the element:sviet
enter the element:Nandamuru
the list is ['pavan', 'sviet', 'Nandamuru']
Is list is sorted: False
```

Exercise 21 -

AIM:

Write a function called root that is given a number x and an integer n and returns $x^{1/n}$. In the function definition, set the default value of n to 2.

Algorithm:

Step1: Start
Step2: Read the number x using input() function and convert it to integer using eval() or int() function and store it in a variable (say x)
Step3: Read the integer n using input() function and convert it to integer using eval() or int() function and store it in a variable (say n)
Step4: if n!=2 then call the function root() function with x and n as the parameters else Call the function root() with x as the parameter
Step5: Print the value returned by the function root() that computes $x^{1/n}$
Step6: Stop

Algorithm for writing root() function:

This function takes the number (say x) and integer(say n) as a parameter where the default value of n is 2 and returns the $x^{1/n}$

Step1: compute x**(1/n) and rounded it to two digits and return the result

Program:

```
def root(x,n=2):
    rt=round(x**(1/n),2)
    return rt

x=int(input('Enter a number: ') )
n=int(input('Enter root value: '))

if(n!=2):
   print("the required value is",root(x,n))
else:
   print("the required value is",root(x))
```

Output1:

Enter a number: 8
Enter root value: 3
the required value is 2.0

Output2:

Enter a number: 125
Enter root value: 3
the required value is 5.0

Exercise 22 -

AIM:

Write a function called primes that is given a number n and returns a list of the first n primes. Let the default value of n be 100.

Algorithm:

Step1: Start
Step2: Read the number using input() function and convert it to integer using eval() or int() function and store it in a variable (say num)
Step3: if num!=100 then call the function primes() function with num as the parameters else call the function primes()
Step5: Print the list and no. of elements of the list returned by the function primes() that computes the list of first n primes
Step6: Stop

Algorithm for writing primes() function:

This function takes the number (say n) as a parameter where the default value of n is 100 and returns the list of first n primes

Step1: Initialize the empty list to a variable (say prime)
Step2: Iterate using the for loop with the iterating variable num in the range(1,10000) and in each iteration do the following
    i) If num>1, then do the following
        a) Iterate using the for loop with the iterating variable i in the range(2,num) and in each iteration check whether num has any factors in the range 2 and num-1.
        b) If num has factors between 2 and num-1 then terminate the loop with break statement else append the num to the prime list using append() method
        c) If len(prime)==n(i.e., first n primes are added to the list) then return the list prime

Program:

```
def primes(n=100):
    prime=[]
    for num in range(1,10000):
        if num>1:
            for i in range(2,num):
                if(num%i)==0:
                    break
            else:
                prime.append(num)
            if len(prime)==n:
                return prime



num=int(input("enter an integer:"))
if(num!=100):
    p=primes(num)

else:
    #calling with default value
    p=primes()

print("the required list is:",p)
print("the number of elements of the list is",len(p))
```

Output1:

```
enter an integer:10
the required list is: [2, 3, 5, 7, 11, 13, 17, 19, 23, 29]
the number of elements of the list is 10
```

Output2:

```
enter an integer:100
the required list is: [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79,
83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181,
191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283,
293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409,
```

419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541]
the number of elements of the list is 100

Exercise 23 -

AIM:

Write a function called merge that takes two already sorted lists of possibly different lengths, and merges them into a single sorted list.
(a) Do this using the sort method. (b) Do this without using the sort method.

Algorithm:

Step1: Start
Step2: Initialize the empty list to the variable(say a)
Step3: Initialize another empty list to the variable(say c)
Step4: Read the total number of elements of two lists and read those many elements and append them to the lists using append() method respectively(the elements must be sorted for both the lists)
Step5: Print the single sorted list by merging the elements of two lists by calling merge_using_sort() method with two lists as parameters
Step6: Print the single sorted list by merging the elements of two lists by calling merge_without_sort() method with two lists as parameters
Step7: Stop

Algorithm for writing merge_using_sort() function:

This function takes two lists l1 and l2 as parameters and prints the single sorted list

Step1: concatenate the two lists using concatenation operator and assign the result to the variable (say new)
Step2: apply the sort() method on the list new
Step3: print the sorted list new

Algorithm for writing merge_withou_sort() function:

This function takes two lists l and m as parameters and prints the single sorted list

Step1: Initialize the empty list to the variable(say result)
Step2: Intialize zero to the variables i and j
Step3: find the len(l) and len(m) and add them and store it in a variable(say total)
Step4: Iterate using the while loop until len(result)!=total and in each iteration do the following
i) If the list l is empty then append all the elements of list m to the result list

and terminate the loop else goto step(ii)

ii) If the list m is is empty then append all the elements of list l to the result list and terminate the loop else goto step(iii)

iii) If the element at index i in list l is less than the element at index j in list m, then append the element of list l at index i to the result list and increase the value of i by 1 else goto step(iv)

iv) If the element at index i in list l is greater than the element at index j in list m, then append the element of list m at index j to the result list and increase the value of j by 1

Step5: print the sorted list result

Program:

```
def merge_using_sort(l1,l2):
    new=l1+l2
    new.sort()
    print("Sorted list is:",new)



def merge_without_sort(l, m):
        result = []
        i = j = 0
        total = len(l) + len(m)
        while len(result) != total:
            if len(l) == i:
                result += m[j:]
                break
            elif len(m) == j:
                result += l[i:]
                break
            elif l[i] < m[j]:
                result.append(l[i])
                i += 1
            else:
                result.append(m[j])
                j += 1
        print("Sorted list is:",result)



a=[]
c=[]
n1=int(input("Enter number of elements:"))
```

```
for i in range(1,n1+1):
    b=int(input("Enter element:"))
    a.append(b)
n2=int(input("Enter number of elements:"))
for i in range(1,n2+1):
    d=int(input("Enter element:"))
    c.append(d)


merge_using_sort(a,c)
merge_without_sort(a,c)
```

Output:

```
Enter number of elements:5
Enter element:1
Enter element:3
Enter element:5
Enter element:7
Enter element:9
Enter number of elements:4
Enter element:2
Enter element:4
Enter element:6
Enter element:8
Sorted list is: [1, 2, 3, 4, 5, 6, 7, 8, 9]
Sorted list is: [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Exercise 24 -

AIM:

Write a program that asks the user for a word and finds all the smaller words that can be made from the letters of that word. The number of occurrences of a letter in a smaller word can't exceed the number of occurrences of the letter in the user's word.

Algorithm:

Step1: Start
Step2: Create a text file (say wordlist.txt) that contains words of the English language each on its own line
Step3: open the text file and read line by line of the file after removing leading and trailing characters and append to the list (say words)
Step4: read the word using input() function and store it in a variable(say s)
Step5: Initialize the empty dictionary to a variable(say d)
Step6: use for loop to Iterate over the string using the iterating variable c and in each iteration do the following
  i) If c is not a key of d, then add a key-value (c,1) pair to the dictionary d else update the value of c by adding one to it in the dictionary(create a dictionary of frequencies of letters in user's word s)
Step7: use for loop to Iterate over the words list using the iterating variable word and in each iteration do the following
  i) Create an empty dictionary(say e)
  ii) Update the dictionary e by finding the frequencies of the letters from word (similarly as in 6(i))(create a dictionary e that contain the frequencies of letters in word)
  iii) Initialize the Boolean value True to a variable(say okay)
  iv) Use the for loop to iterate over the dictionary e using the iterating variable (say c) and in each iteration do the following
    a) If c is not in d or e[c] > d[c] then set the variable okay value to False
  v) If okay variable value is True and len(word)<len(s) then print the word
Step8: Stop

Program:

```
words = [line.strip() for line in open('wordlist.txt')]
s = input('Enter a word: ')
print("the smaller words that can be made from the user's word are\n")
d = {} # dictionary of frequencies in user's word
for c in s:
    if c in d:
        d[c] += 1
```

```
    else:
        d[c] = 1
for word in words:
    e = {} # dictionary of frequencies in current word
    for c in word:
        if c in e:
            e[c] += 1
        else:
            e[c] = 1
    okay = True
    for c in e:
        if c not in d or e[c] > d[c]:
            okay = False
    if okay and len(word)<len(s):
        print(word, end=' ')
print()
```

Note:

assume that wordlist.txt contains the following words in separate lines

```
 pan
an
van
pa
pavan
kumar
sviet
nandamuru
car
```

Output:

```
Enter a word: pavan sviet
the smaller words that can be made from the user's word are

pan an van pa pavan sviet
```

Exercise 25 -

AIM:

Write a program that reads a file consisting of email addresses, each on its own line. Your program should print out a string consisting of those email addresses separated by semicolons.

Algorithm:

Step1: Start
Step2: create a text file (say mails.txt) that contains email addresses each on its own line
Step3: open the text file and read line by line of the file after removing leading and trailing characters and add to the list (say email)
Step4: initialize the empty string to the variable(say em)
Step5: use for loop to iterate over email list using iterating variable t and in each iteration do the following
    i) Concatenate t with semicolon(;) and concatenate this result to string em and store it in string em only (perform em=em+t+';')
Step5: remove the last character from the string em
Step6: print the string em
Step7: Stop

Program:

```
email = [line.strip() for line in open('mails.txt')]
print("the string consisting of email addresses separated by semicolons is \n")
em=''
for t in email:
    em=em+t+';'
em=em[:-1]
print(em)
```

Note:

assume that mails.txt contains the following mail addresses each on its own line

kumarap.1980@gmail.com
svietexams@gmail.com
abc@gmail.com
pavank_1980@rediffmail.com

Output:

the string consisting of email addresses separated by semicolons is

kumarap.1980@gmail.com;svietexams@gmail.com;abc@gmail.com;pavank_1980@rediffmail.com

Exercise 26 -

AIM:

Write a program that reads a list of temperatures from a file called *temps.txt*, converts those temperatures to Fahrenheit, and writes the results to a file called ftemps.txt.

Algorithm:

Step1: Start
Step2: create a text file (say temps.txt) that contains temperatures each on its own line
Step3: use open function in write mode 'w' to create a file object and associates it with a ftemps.txt file on the disk(let the variable that reference the file object is file1)
Step4: use open function to read line by line of the temps.txt file after removing leading and trailing characters and add to the list (say temperatures)
Step5: map each value of temperatures list to numeric value using map() function and assign it temperatures only
Step6: use for loop to iterate over temperatures list using the iterating variable t and in each iteration do the following
    i) Compute t*9/5+32 and prints the result to the ftempts.txt file(variable that reference the file object is file1)
Step7: close the file ftempts.txt using the reference variable file1.
Step8: Stop

Program:

```
file1 = open('ftemps.txt', 'w')
temperatures = [line.strip() for line in open('temps.txt')]
temperatures=map(eval,temperatures)
print("list of temperatures contained in temps.txt is:",list(temperatures))
for t in temperatures:
   print(t*9/5+32, file=file1)
print(" the results of the program can be seen in ftemps.txt file")
file1.close()
```

Note:

assume that temps.txt contains the following temperatures each on its own line
30
40
50
60

70
Output:

list of temperatures contained in temps.txt is: [30, 40, 50, 60, 70]
the results of the program can be seen in ftemps.txt file

Exercise 27 -

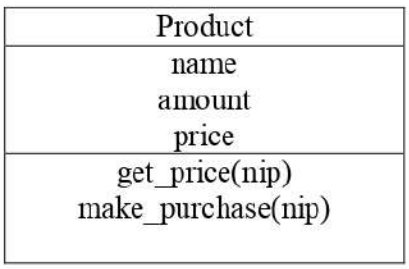
AIM:

Write a class called Product. The class should have fields called name, amount, and price, holding the product's name, the number of items of that product in stock, and the regular price of the product. There should be a method *get_price* that receives the number of items to be bought and returns a the cost of buying that many items, where the regular price is charged for orders of less than 10 items, a 10% discount is applied for orders of between 10 and 99 items, and a 20% discount is applied for orders of 100 or more items. There should also be a method called *make_purchase* that receives the number of items to be bought and decreases amount by that much.

Algorithm:

Step1: Start
Step2: read the product name, number of items of the product in stock, price of the product, number of items to purchase and store it in variables(say n,a,p,nip respectively)
Step3: An instance of the Product class(say c) is created and the data is assigned to its attributes name,amount and price through n,a,p(c=Product(n,a,p))
Step4: print the cost of buying that many items as nip using get_price() method of product class
Step5: print the the number of items of the product left in stock after buying that many items as nip using make_purchase() method
Step6: Stop

class Product is designed as shown in the following class diagram. (Here nip stands for number of items to purchase)



It contains:

Step 1: Constructor to initialize the data member name, amount and price

Step 2: get_price(nip) method

```
If nip<10 then
        return price
else if nip>10 and nip<100 then
        return price – 0.1*nip*price
else
        return price – 0.2*nip*price
```

Step 3: make_purchase(nip) method

```
        amount =amount –nip
        return amount
```

Program:

```
class Product:
    def __init__(self,name,amount,price):
        self.name=name
        self.amount=amount    #amount is number of items in stock
        self.price=price

    def get_price(self,nip):
        if nip<10:
            return "regular price:",nip*self.price
        elif nip>10 and nip<100:
            return "the price after discount of 10% applied is:",nip*self.price-(0.1*nip*self.price)
        else:
            return "the price after discount of 20% applied is:",nip*self.price-(0.2*nip*self.price)

    def make_purchase(self,nip):
        self.amount=self.amount-nip
        return  "the number of items of the product in stock after buying",nip,"items is",self.amount

n=input("enter product name:")
a=int(input("enter the number of items of the product in stock:"))
p=eval(input("enter the price of the product:"))
nip=eval(input("enter no.of items to purchase:"))
c=Product(n,a,p)
print(c.get_price(nip))
print(c.make_purchase(nip))
```

Output:

```
enter product name: book
enter the number of items of the product in stock:100
enter the price of the product:20
enter no.of items to purchase:25
('the price after discount of 10% applied is:', 450.0)
('the number of items of the product in stock after buying', 25, 'items is', 75)
```

Exercise 28 -

AIM:

Write a class called Time whose only field is a time in seconds. It should have a method called *convert_to_minutes* that returns a string of minutes and seconds formatted as in the following example: if seconds is 230, the method should return '3:50'. It should also have a method called *convert_to_hours* that returns a string of hours, minutes, and seconds formatted analogously to the previous method.

Algorithm:

Step1: Start
Step2: import datetime module
Step3: read the time in seconds and store it in variables(say t)
Step4: An instance of the Time class(say a) is created and the data is assigned to its attributes sec through t(a=Time(t))
Step5: print the string of minutes using convert_to_minutes() method of Time class
Step6: print the the string of hours, minutes and seconds formatted in the required way using convert_to_hours() method
Step7: Stop

class Time is designed as shown in the following class diagram. (Here sec stands for time in seconds)

Time
sec
convert_to_minutes() convert_to_hours()

It contains:
Step 1: Constructor to initialize the data member sec
Step 2: convert_to_minutes( ) method

```
return '{}:{}minutes'.format(self.sec//60,self.sec%60)
```

Step 3: convert_to_hours( ) method

```
return str(datetime.timedelta(seconds=self.sec))+'hours'
```

Program:

```
import datetime
class Time:
    def __init__(self,sec):
        self.sec=sec

    def convert_to_minutes(self):
        return '{}:{}minutes'.format(self.sec//60,self.sec%60)

    def convert_to_hour(self):
        return str(datetime.timedelta(seconds=self.sec))+'hours'


t=int(input("enter the time in seconds:"))
a=Time(t)
print(a.convert_to_minutes())
print(a.convert_to_hour())
```

Output1:

```
enter the time in seconds:230
3:50minutes
0:03:50hours
```

Output2:

```
enter the time in seconds:4600
76:40minutes
1:16:40hours
```

Exercise 29 -

AIM:

Write a class called Converter. The user will pass a length and a unit when declaring an object from the class—for example, c = Converter(9,'inches'). The possible units are inches, feet, yards, miles, kilometers, meters, centimeters, and millimeters. For each of these units there should be a method that returns the length converted into those units. For example, using the Converter object created above, the user could call c.feet() and should get 0.75 as the result.

Algorithm:

Step1: Start
Step2: read the length value , unit value and store it in variables(say len,ut respectively)
Step3: An instance of the Converter class(say c) is created and the data is assigned to its attributes length and unit through len and ut (c=Converter(len,ut))
Step4: read the units that length value need to be converted and store it in a variable(say ct)
Step5: if ct=='feet'
call c.feet() method to convert length to feet
elif ct=='inches'
call c.inches() method to convert length to inches
elif ct=='yards':
call c.yards() method to convert length to yards
elif ct=='miles':
call c.miles() method to convert length to miles
elif ct=='kilometers':
call c.kilometers() method to convert length to kilometers
elif ct=='meters':
call c.meters() method to convert length to meters
elif ct=='centimeters':
call c.centimeters() method to convert length to centimeters
elif ct=='millimeters':
call c.millimeters() method to convert length to millimeters
Step6: Stop

class Converter is designed as shown in the following class diagram.

Converter
length unit
feet() inches() yards() miles() kilometers() meters() centimeters() millimeters()

It contains:

Step 1: Constructor to initialize the data members length and unit

Step 2: feet( ) method

```
if unit=='inches'
        return length*0.083333
elif if unit=='yards'
        return length*3
elif if unit=='miles'
        return length*5280
elif if unit=='kilometers'
        return length*3280.84
elif if unit=='meters'
        return length*3.28084
elif if unit=='centimeters'
        return length*0.0328084
elif if unit=='millimeters'
        return length*0.00328084
```

Step 3: similarly define inches() method, yards() method,miles() method, kilometers() method, meters() method,centimeters() method, millimeters() method by choosing appropriate values

Program:

```
class converter:
    def __init__(self,length,unit):
        self.length=length
        self.unit=unit
    #creating feet() function
    def feet(self):
        if self.unit=='inches':
            ft=self.length*0.083333
            return ft
        elif self.unit=='yards':
            ft=self.length*3
            return ft
        elif self.unit=='miles':
            ft=self.length*5280
            return ft
        elif self.unit=='kilometers':
            ft=self.length*3280.84
            return ft
        elif self.unit=='meters':
            ft=self.length*3.28084
            return ft
        elif self.unit=='centimeters':
            ft=self.length*0.032808
            return ft
        elif self.unit=='millimeters':
            ft=self.length*0.003281
            return ft
    #creating inches() function
    def inches(self):
        if self.unit=='feet':
            inch=self.length*12
            return inch
        elif self.unit=='yards':
            inch=self.length*36
            return inch
        elif self.unit=='miles':
            inch=self.length*63360
            return inch
        elif self.unit=='kilometers':
            inch=self.length*39370.08
```

```
        return inch
    elif self.unit=='meters':
        inch=self.length*39.37008
        return inch
    elif self.unit=='centimeters':
        inch=self.length*0.393701
        return inch
    elif self.unit=='millimeters':
        inch=self.length*0.03937
        return inch
#creating yards() function
def yards(self):
    if self.unit=='feet':
        yrds=self.length*0.333333
        return yrds
    elif self.unit=='inches':
        yrds=self.length*0.027778
        return yrds
    elif self.unit=='miles':
        yrds=self.length*1760
        return yrds
    elif self.unit=='kilometers':
        yrds=self.length*1093.613
        return yrds
    elif self.unit=='meters':
        yrds=self.length*1.093613
        return yrds
    elif self.unit=='centimeters':
        yrds=self.length*0.010936
        return yrds
    elif self.unit=='millimeters':
        yrds=self.length*0.001094
        return yrds
#creating miles() function
def miles(self):
    if self.unit=='feet':
        mls=self.length*0.000189
        return mls
    elif self.unit=='inches':
        mls=self.length*0.000016
        return mls
    elif self.unit=='yards':
        mls=self.length*0.000568
        return mls
```

```
    elif self.unit=='kilometers':
        mls=self.length*0.621371
        return mls
    elif self.unit=='meters':
        mls=self.length*0.000621
        return mls
    elif self.unit=='centimeters':
        mls=self.length*0.000006
        return mls
    elif self.unit=='millimeters':
        mls=self.length*0.00000062137
        return mls
#creating millimeters() function
def millimeters(self):
    if self.unit=='feet':
        mm=self.length*304.8
        return mm
    elif self.unit=='inches':
        mm=self.length*25.4
        return mm
    elif self.unit=='yards':
        mm=self.length*914.4
        return mm
    elif self.unit=='kilometers':
        mm=self.length*1000000
        return mm
    elif self.unit=='meters':
        mm=self.length*1000
        return mm
    elif self.unit=='centimeters':
        mm=self.length*10
        return mm
    elif self.unit=='miles':
        mm=self.length*1609344
        return mm
#creating kilometers() function
def kilometers(self):
    if self.unit=='feet':
        km=self.length*0.000305
        return km
    elif self.unit=='inches':
        km=self.length*0.0000025
        return km
    elif self.unit=='yards':
```

```
        km=self.length*0.000914
        return km
    elif self.unit=='millimeters':
        km=self.length*0.000001
        return km
    elif self.unit=='meters':
        km=self.length*0.001
        return km
    elif self.unit=='centimeters':
        km=self.length*0.00001
        return km
    elif self.unit=='miles':
        km=self.length*1.609344
        return km
#creating centimeters() function
def centimeters(self):
    if self.unit=='feet':
        cm=self.length*30.48
        return cm
    elif self.unit=='inches':
        cm=self.length*2.54
        return cm
    elif self.unit=='yards':
        cm=self.length*91.44
        return cm
    elif self.unit=='millimeters':
        cm=self.length*0.1
        return cm
    elif self.unit=='meters':
        cm=self.length*100
        return cm
    elif self.unit=='kilometers':
        cm=self.length*100000
        return cm
    elif self.unit=='miles':
        cm=self.length*160934.4
        return cm
#creating meters() function
def meters(self):
    if self.unit=='feet':
        m=self.length*0.3048
        return m
    elif self.unit=='inches':
        m=self.length*0.0254
```

```
            return m
        elif self.unit=='yards':
            m=self.length*0.9144
            return m
        elif self.unit=='millimeters':
            m=self.length*0.001
            return m
        elif self.unit=='centimeters':
            m=self.length*0.01
            return m
        elif self.unit=='kilometers':
            m=self.length*1000
            return m
        elif self.unit=='miles':
            m=self.length*1609.344
            return m
len=eval(input("enter length value:"))
ut=input("enter unit value:")
c=converter(len,ut)
ct=input("Enter unit value to which it converts:")
if ct=='feet':
    print("after converting",ut,"to feet the value is",round(c.feet(),4))
elif ct=='inches':
    print("after converting",ut,"to inches the value is",round(c.inches(),4))
elif ct=='yards':
    print("after converting",ut,"to yards the value is",round(c.yards(),4))
elif ct=='miles':
    print("after converting",ut,"to miles the value is",round(c.miles(),4))
elif ct=='kilometers':
    print("after converting",ut,"to kilometers the value is",round(c.kilometers(),4))
elif ct=='meters':
    print("after converting",ut,"to meters the value is",round(c.meters(),4))
elif ct=='centimeters':
    print("after converting",ut,"to centimeters the value is",round(c.centimeters(),4))
elif ct=='millimeters':
    print("after converting",ut,"to millimeters the value is",round(c.millimeters(),4))
```

Output1:

```
enter length value:9
enter unit value:inches
Enter unit value to which it converts:feet
after converting inches to feet the value is 0.75
```

Output2:

enter length value:9
enter unit value:inches
Enter unit value to which it converts:yards
after converting inches to yards the value is 0.25

Exercise 30 -

AIM:

Write a Python class to implement pow(x, n).

Algorithm:

Step1: Start
Step2: read the base and exponent values through keyboard and store it into the variables (say base and exp respectively)
Step3: An instance of the Power class(say c) is created (c=Power())
Step4: print the base raised to the exp using pow() method of Power class by passing base and exp values as parameters
Step5: Stop

class Power is designed as shown in the following class diagram. (here x and n are base and exponent values)

Power
pow(x,n) method

It contains:

Step 1: pow(x,n) method which is defined as follows

If x==0 or x==1 or n==1 then return x
If x==-1 then return 1 if n is even else return -1if n is odd
If n==0 then return 1
If n<0 then return 1/self.pow(x,-n) (using recursive function)
In all the remaining cases, do the following
i) Perform val= self.pow(x,n//2)
ii) If n is even return val*val else return val*val*x

Program:

```
class Power:

  def pow(self, x, n):
      if x==0 or x==1 or n==1:
         return x

      if x==-1:
         if n%2 ==0:
            return 1
         else:
            return -1
      if n==0:
         return 1
      if n<0:
         return 1/self.pow(x,-n)
      val = self.pow(x,n//2)
      if n%2 ==0:
         return val*val
      return val*val*x


base=int(input("enter base value:"))
exp=int(input("enter power value:"))
c=Power()
print(c.pow(base,exp))
```

Output1:

```
enter base value:2
enter power value:-3
0.125
```

Output2:

```
enter base value:2
enter power value:8
256
```

Exercise 31 -

AIM:

Write a Python class to reverse a string word by word

Algorithm:

Step1: Start
Step2: read the string through keyboard and store it into the variables (say s1)
Step3: An instance of the Reverse class(say c) is created (c=Reverse())
Step4: reverse of the string word by word using reverse_words() method by passing the string s1 as a parameter and print it.
Step5: Stop

class Reverse is designed as shown in the following class diagram. (here s is the string)

Reverse
reverse_words(s) method

It contains:

Step 1: reverse_words(s) method which is defined as follows

i) Split the string s using split() method to get the list of words of the string
ii) Apply the built_in function reversed() that returns the reverse iterator of the list
iii) Apply the join method over reverse iterator to join the words of the words in reverse order

Program:

```
class Reverse:
   def reverse_words(self, s):
      return ' '.join(reversed(s.split()))

s1=input("enter a string:")
c=Reverse()
print("the string after reversing word by word is\n")
print(c.reverse_words(s1))
```

Output1:

enter a string:This Lab Manual is prepared by pavan kumar
the string after reversing word by word is

kumar pavan by prepared is Manual Lab This

Output2:

enter a string: My college is sviet, Nandamuru
the string after reversing word by word is

Nandamuru sviet, is college My

Exercise 32 -

AIM:

Write a program that opens a file dialog that allows you to select a text file. The program then displays the contents of the file in a textbox

Algorithm:

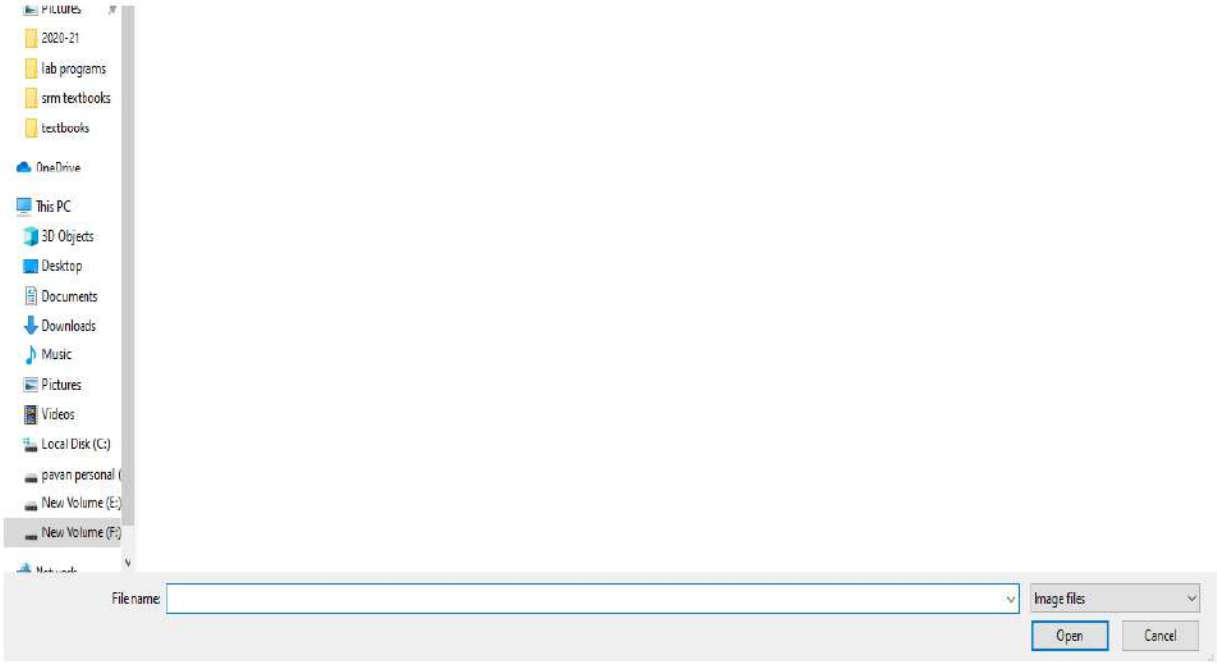
Step1: Start
Step2: import the GUI stuff from the tkinter module and also filedialog module and ScrolledText widget
Step3: create a window on the screen(call it as root)
Step4: create a ScrolledText widget (call it as textbox) and apply the grid() method to the textbox
Step5: Open a file chooser dialog using askopenfilename dialog box and select the file whose contents need to be displayed in a text box.(call it as a filename)
Step6: read the file content after opening the file using open() method and load the entire file into a string (say s)
Step7: insert the content of the string using insert() command into the textbox
Step8: call the tkinter module's mainloop() function.
Step9: Stop

Program:

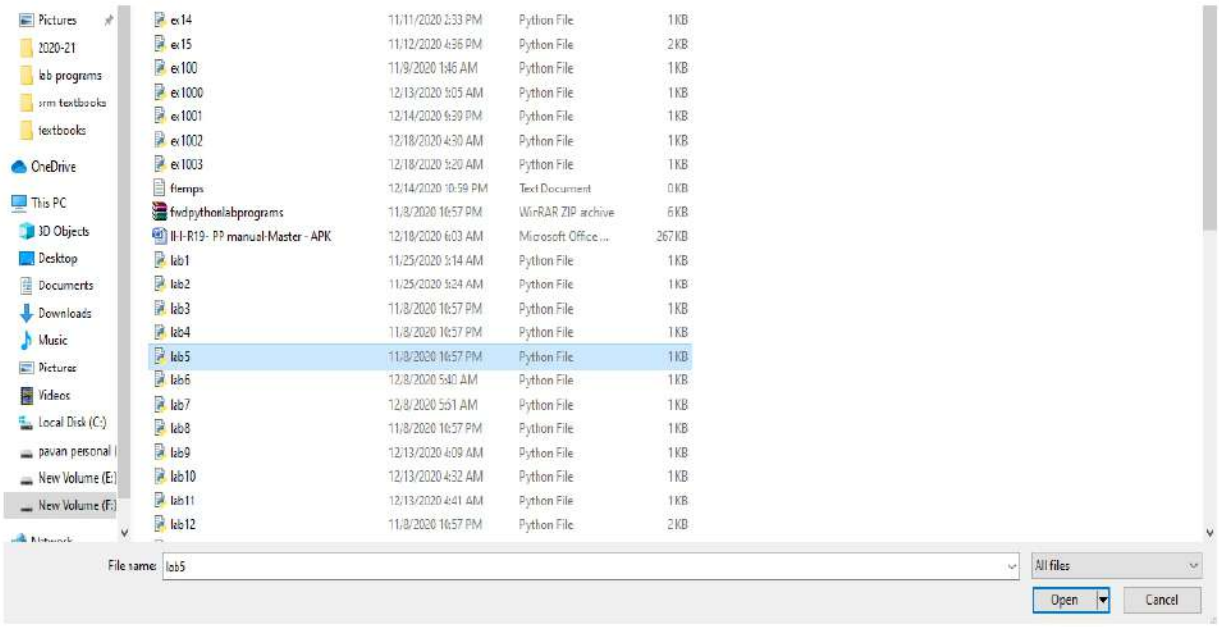
```
from tkinter import *
from tkinter.filedialog import *
from tkinter.scrolledtext import ScrolledText
root = Tk()
textbox = ScrolledText()
textbox.grid()
filename=askopenfilename(initialdir='D:\\python\\python notes\\2020-21\\lab programs',
                         filetypes=[('Image files', '.jpg .png .gif'),
                                    ('All files', '*')])
root.title(filename)
s = open(filename).read()
textbox.insert(1.0, s)
mainloop()
```

Output:

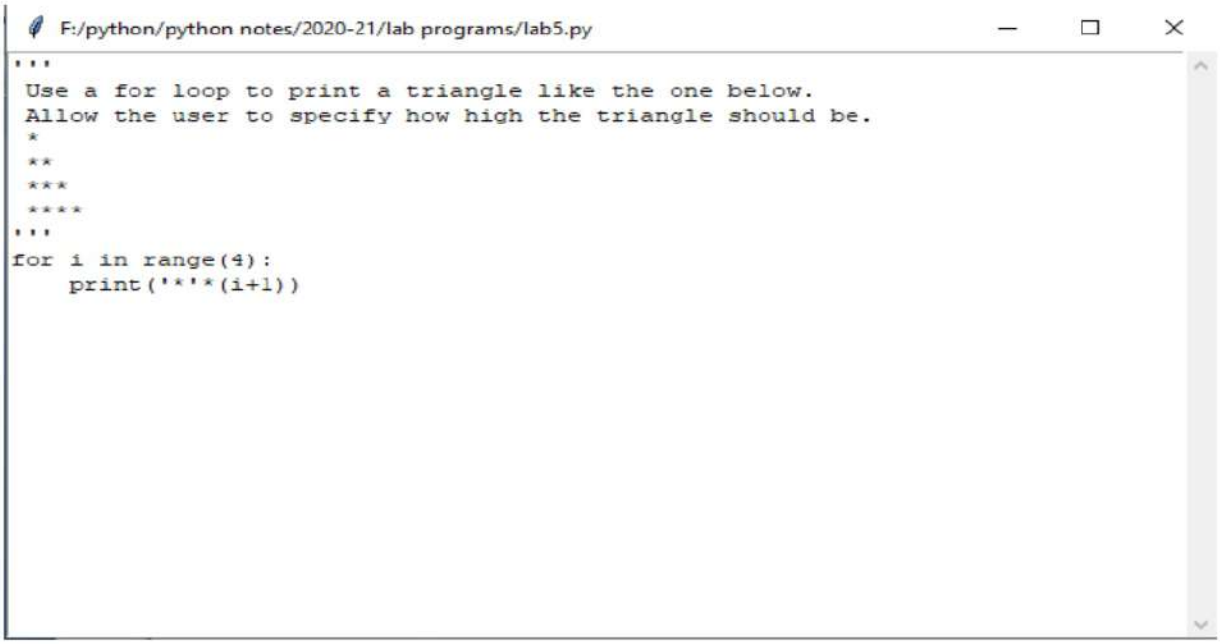
Screen shot of the dialog box to select file



Now select the file whose content need to be displayed in to the text box.



The content of chosen file is displayed in the text box as follows.



Exercise 33 -

AIM:

Write a program to demonstrate Try/except/else.

Algorithm:

Step1: Start
Step2: Try to open the file in read mode in try block.
Step3: If the file doesn't exist, then , an input/output exception called IOError is generated.
Write the code to handle this exception in the except block.
Step4: If the file does exist, then write the code in the else block that want to do with it.

The syntax of try/except/else is

```
try:

    statements

except (Exception1[, Exception2[,...ExceptionN]]]):

    statements  #If there is any exception from the given exception list, then execute this

except :

    statements     #If there is any other exception not listed above, then execute this

else:

    statements     #If there is no exception then execute this block.
```

Step5: Stop

Program:

```
try:
   name=input("enter the filename with extension:")
   file = open(name, 'r')
except IOError:
   print("Could not open file. The file doesn't exist")
else:
   s = file.read()
   print("the content of the file",name,"is\n",s)
```

Output1:

```
enter the filename with extension:mails.txt
the content of the file mails.txt is
 kumarap.1980@gmail.com
svietexams@gmail.com
abc@gmail.com
pavank_1980@rediffmail.com
```

Output2:

```
enter the filename with extension:sviet.txt
Could not open file. The file doesn't exist
```

Exercise 34 -

AIM:

Write a program to demonstrate try/finally and with/as.

Algorithm:

Step1: Start
Step2: Try to open the existing file in read mode in try block.
Step3: write the code to read the data of the file and print it in try block
Step4: close the file in finally block
Step5: Now try to open the same file in read mode and print the data using with/as (no need to close the file. It will automatically close the file)
Step6: Stop

Program:

```
print("using try/finally, the content of the file is")
f = open('data.txt','r')
try:
        data = f.read()
         print(data)
finally:
        f.close()

print("----------------")
print("using with/as, the content of the file is")
with open('data.txt') as f:
        data = f.read()
        print(data)
```

Output:

```
using try/finally, the content of the file is
When you use with statement with open function, you do not need to close the file at the end, because with would automatically close it for you.
----------------
using with/as, the content of the file is
When you use with statement with open function, you do not need to close the file at the end, because with would automatically close it for you.
```

Additional Experiment1:

Exercise 35 -

AIM:

Write a Program to perform matrix multiplication using lists.

Algorithm:

Step1: Start
Step2: Read the no.of rows and columns of first matrix and second matrix and store it in variables (say r1,c1,r2,c2 respectively)
Step3: if c1==r2 then do the following else goto step4
- i) Read th elements of first matrix using list comprehension and store it in matrix1
- ii) Read the elements of second matrix using list comprehension and store it in matrix2
- iii) Initialize the all the elements of resultant matrix as zero using list comprehension. The order of resultant matrix is r1 x c2
- iv) Now use the for loop to iterate using the iterating variable i in the range(r1) and in each iteration do the following
  - a) Use the for loop to iterate using the iterating variable j in the range(c2) and in each iteration do the following
    - I) Use the for loop to iterate using the iterating variable k in the range(c1) and in each iteration do the following

      resmatrix[i][j]=resmatrix[i][j]+matrix1[i][k]*matrix2[k][j]
- v) now print the matrix1, matrix2 and resultant matrix and goto Step5

Step4: print the message " matrix multiplication is not possible"
Step5: Stop

Program:

```
def print_matrix(matrix):

    for i in range(len(matrix)):
        for j in range(len(matrix[0])):
            print("\t",matrix[i][j],end=' ')
        print("\n")

r1=int(input("enter number of rows of first matrix"))
c1=int(input("enter number of columns of first matrix"))
r2=int(input("enter number of rows of first matrix"))
c2=int(input("enter number of columns of second matrix"))

if( c1==r2):
    print("matrix multiplication is possible")
    print("enter elements of first matrix")
    matrix1=[[int(input("enter an element:")) for j in range(c1)] for i in range(r1)]

    print("enter elements of second matrix")
    matrix2=[[int(input("enter an element:")) for j in range(c2) ] for i in range(r2)]

    resmatrix=[[0 for j in range(c2)] for i in range(r1)]
    for i in range(r1):
        for j in range(c2):
            for k in range(c1):
                resmatrix[i][j]=resmatrix[i][j]+matrix1[i][k]*matrix2[k][j]


    print("matrix1 is")
    print_matrix(matrix1)

    print("matrix2 is")
    print_matrix(matrix2)

    print("resultant matrix is")
    print_matrix(resmatrix)

else:
    print("matrix multiplication is not possible")
```

Output:

```
enter number of rows of first matrix2
enter number of columns of first matrix2
enter number of rows of first matrix2
enter number of columns of second matrix2
matrix multiplication is possible
enter elements of first matrix
enter an element:1
enter an element:0
enter an element:0
enter an element:1
enter elements of second matrix
enter an element:1
enter an element:2
enter an element:3
enter an element:4
matrix1 is
        1       0

        0       1

matrix2 is
        1       2

        3       4

resultant matrix is
        1       2

        3       4
```

Additional Experiment2:

Exercise 36 -

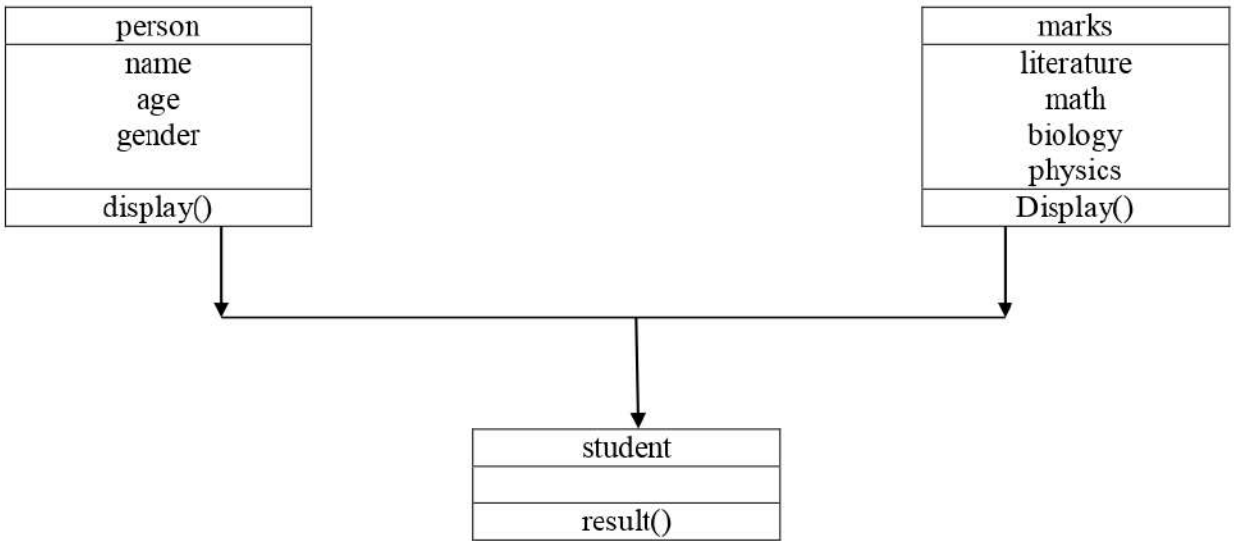
AIM:

Write Python Programs to implement Multiple Inheritance

Algorithm:

Step 1: Start
Step 2: class person designed as shown in the class diagram. It contains:
Step 3: Constructor to initialize the data members.
Step 4: display() method to display information about the person.
Step 5: class marks designed as shown in the class diagram. It contains:
Step 6: Constructor to initialize the data members.
Step 7: display() method to display marks.
Step 8: class student designed as shown in the class diagram and also inherits features from base class's person and makrs. It contains:
Step 9: Constructor to call the base class's constructors from derived class.
Step 10: results() method to display student results.
Step 11: Stop

Class Diagram:



Program:

```
class person:
    # Constructor
    def __init__(self):
        self.name = input("Name: ")
        self.age = input("Age: ")
        self.gender = input("Gender: ")

    # Method
    def display(self):
        print("\n\nName: ",self.name)
        print("Age: ",self.age)
        print("Gender: ",self.gender)


# Define a class as 'marks'
class marks:
    # Constructor
    def __init__(self):
        self.stuClass = input("Class: ")
        print("Enter the marks of the respective subjects")
        self.literature = int(input("Literature: "))
        self.math = int(input("Math: "))
        self.biology = int(input("Biology: "))
        self.physics = int(input("Physics: "))

     # Method
    def display(self):
        print("Study in: ",self.stuClass)
        print("Total Marks: ", self.literature + self.math + self.biology + self.physics)


# Define a class as 'student' and inherit two super classes 'person' and 'marks'
class student(person, marks):
    def __init__(self):
        # Call constructor of super class 'person'
        person.__init__(self)
        # Call constructor of super class 'marks'
        marks.__init__(self)

    def result(self):
        # Call method of class 'person'
        person.display(self)
```

```
        # Call method of class 'marks'
        marks.display(self)

# Objects of class 'student'
stu1 = student()

# Call method of class 'student'
stu1.result()
```

Output:

```
Name: pavan kumar allada
Age: 12
Gender: Male
Class: 8
Enter the marks of the respective subjects
Literature: 67
Math: 89
Biology: 56
Physics: 50


Name:  pavan kumar allada
Age:  12
Gender:  Male
Study in:  8
Total Marks:  262
```