**FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY (FISAT)TM**

**HORMIS NAGAR, MOOKKANNOOR, ANGAMALY-683577**



**FOCUS ON EXCELLENCE**

**20MCA131 PROGRAMMING LAB**

**LABORATORY RECORD**

**Name: AJEEBA SHARAF M S**

**Branch: MASTER OF COMPUTER APPLICATIONS**

**Semester: 1 Batch: A Roll No: 6**

**MARCH 2022**

**FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY (FISAT)TM**

**HORMIS NAGAR, MOOKKANNOOR, ANGAMALY-683577**



**FOCUS ON EXCELLENCE**

**CERTIFICATE**

*This is to certify that this is a* ***Bonafide*** *record of the Practical work done by* ***AJEEBA SHARAF M S*** *in the* ***20MCA131 PROGRAMMING LAB*** *Laboratory**towards the partial fulfilment for the award of the Master Of Computer Applications during the academic year 2021-2022.*

Signature of Staff in Charge Signature of H O D

Name: Name:

**Date of University practical examination ………………………**

Signature of Signature o

Internal Examiner External Examiner

**CONTENT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SI No:** | **Date :** | **Name of Experiment:** | **Page No:** | **Signature of**  **Staff –In –Charge:** |
| **1** |  | Display future leap years from current year to a final year entered by user. | **7** |  |
| **2** |  | List comprehensions: (a) Generate positive list of numbers from a given list of integers (b) Square of N numbers (c) Form a list of vowels selected from a given word (d) List ordinal value of each element of a word (Hint: use ord() to get ordinal values) | **8** |  |
| **3** |  | Count the occurrences of each word in a line of text. | **9** |  |
| **4** |  | Prompt the user for a list of integers. For all values greater than 100, store ‘over’ instead | **12** |  |
| **5** |  | Store a list of first names. Count the occurrences of ‘a’ within the list | **13** |  |
| **6** |  | Enter 2 lists of integers. Check (a) Whether list are of same length (b) whether list sums to same value (c) whether any value occur in both. | **14** |  |
| **7** |  | Get a string from an input string where all occurrences of first character replaced with ‘$’, except first character | **16** |  |
| **8** |  | Create a string from given string where first and last characters exchanged. [eg: python - > nythop] | **17** |  |
| **9** |  | Accept the radius from user and find area of circle. | **18** |  |
| **10** |  | Find biggest of 3 numbers entered. | **19** |  |
| **11** |  | Accept a file name from user and print extension of that. | **20** |  |
| **12** |  | Create a list of colors from comma-separated color names entered by user. Display first and last colors. | **21** |  |
| **13** |  | Accept an integer n and compute n+nn+nnn. | **22** |  |
| **14** |  | Print out all colors from color-list1 not contained in color-list2. | **23** |  |
| **15** |  | Create a single string separated with space from two strings by swapping the character at position 1. | **24** |  |
| **16** |  | Sort dictionary in ascending and descending order. | **25** |  |
| **17** |  | Merge two dictionaries. | **26** |  |
| **18** |  | Find gcd of 2 numbers. | **28** |  |
| **19** |  | From a list of integers, create a list removing even numbers. | **29** |  |
| **20** |  | Program to find the factorial of a number | **30** |  |
| **21** |  | Generate Fibonacci series of N terms | **31** |  |
| **22** |  | Find the sum of all items in a list | **32** |  |
| **23** |  | Generate a list of four digit numbers in a given range with all their digits even and the number is a perfect square. | **33** |  |
| **24** |  | Display the given pyramid with step number accepted from user. | **34** |  |
| **25** |  | Count the number of characters (character frequency) in a string. | **35** |  |
| **26** |  | Add ‘ing’ at the end of a given string. If it already ends with ‘ing’, then add ‘ly’. | **36** |  |
| **27** |  | Accept a list of words and return length of longest word. | **37** |  |
| **28** |  | Construct following pattern using nested loop.  \*  \* \*  \* \* \*  \* \* \* \*  \* \* \* \* \*  \* \* \* \*  \* \* \*  \* \*  \* | **38** |  |
| **29** |  | Generate all factors of a number | **39** |  |
| **30** |  | Create a package graphics with modules rectangle, circle and sub-package 3D-graphics with modules cuboid and sphere. Include methods to find area and perimeter of respective figures in each module. Write programs that finds area and perimeter of figures by different importing statements. (Include selective import of modules and import \* statements) | **39** |  |
| **31** |  | Create Rectangle class with attributes length and breadth and methods to find area and perimeter. Compare two Rectangle objects by their area | **42** |  |
| **32** |  | Create a Bank account with members account number, name, type of account and balance. Write constructor and methods to deposit at the bank and withdraw an amount from the bank | **44** |  |
| **33** |  | Create a class Rectangle with private attributes length and width. Overload ‘<’ operator to compare the area of 2 rectangles | **46** |  |
| **34** |  | Create a class Time with private attributes hour, minute and second. Overload ‘+’ operator to find sum of 2 time. | **48** |  |
| **35** |  | Create a class Publisher (name). Derive class Book from Publisher with attributes title and author. Derive class Python from Book with attributes price and no\_of\_pages. Write a program that displays information about a Python book. Use base class constructor invocation and method overriding. | **49** |  |
| **36** |  | Write a Python program to read a file line by line and store it into a list. | **51** |  |
| **37** |  | Write a Python program to read each row from a given csv file and print a list of string | **52** |  |

**AIM**

1. Display future leap years from current year to a final year entered by User**.**

**CODE**

print("Enter the two years:")

print("Enter the start year:")

startyear=int(input("startyear"))

print("Enter the last year:")

lastyear=int(input("lastyear"))

for year in range(startyear,lastyear):

if (year%4==0) and (year%100!=0) or (year%400==0): print (year)

**OUTPUT**



**AIM**

2.List comprehensions:

1. Generate positive list of numbers from a given list of  integers.
2. Squares of N numbers
3. Form a list of vowels selected from a given word
4. List ordinal value of each element of a word

**CODE**

a)Generate positive list of numbers from a given list of integers.

list1=[12,-1,-2,0,45,67]

for num in list1:

if (num>=0):

print(num)

**OUTPUT**



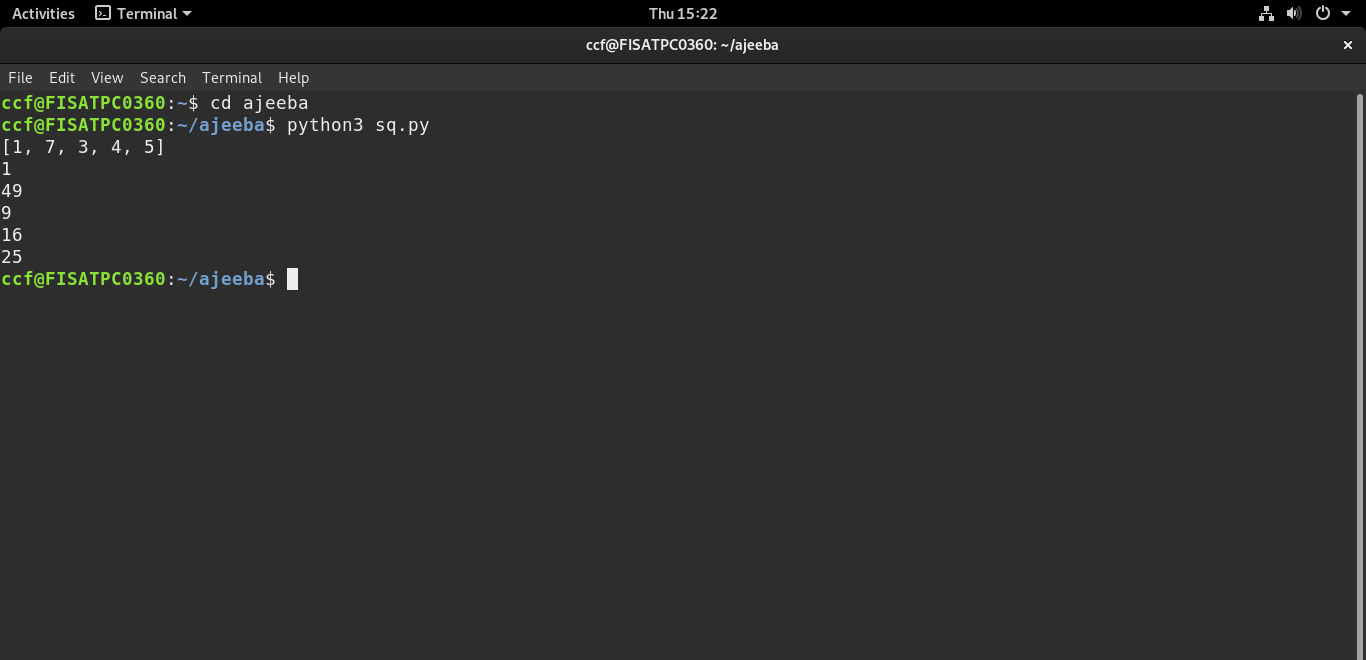
**b)**Square of N numbers

n=int(input("enter the range"))

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

num=num\*num

print(num)

**OUTPUT**

c)Form a list of vowels selected from a given word.

s=input("enter any statement:")

L=[]

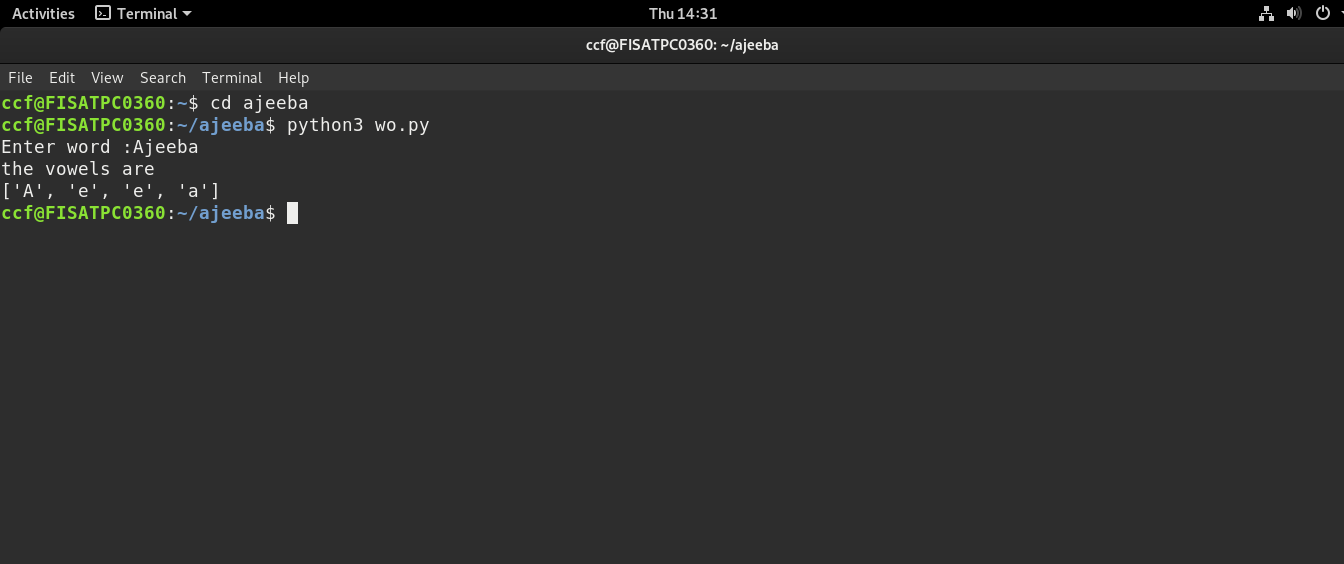
for i in s:

if i in "aeiouAEIOU":

L.append(i)

print(L)

**OUTPUT**



d)List ordinal values of each element of a word.

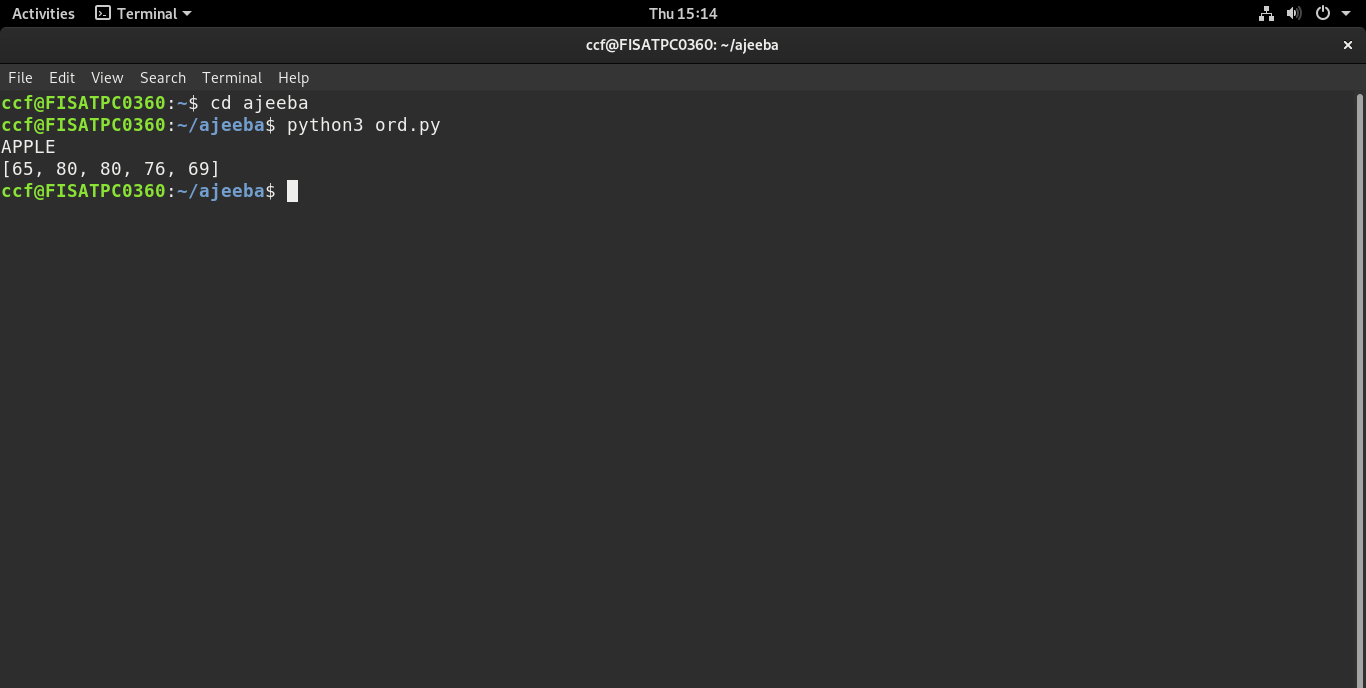
list=['f','i','s','a','t']

for i in range(0,5):

value=ord(list[i])

print(value)

**OUTPUT**



**AIM**

3.Count the occurrences of each word in a line of text.

**CODE**

list1=[]

list2=[]

x=input("Enter a string:")

for i in x.split(" "):

 list1.append(i)

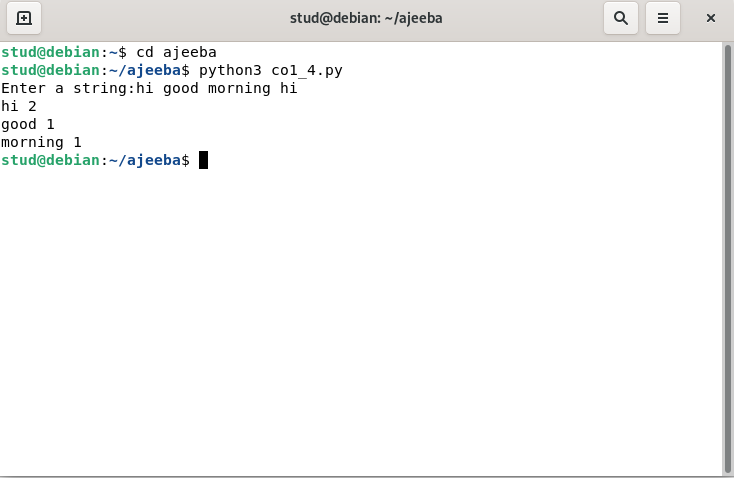
 if i not in list2:

 list2.append(i)

for i in list2:

 print(i,"\t",list1.count(i))

**OUTPUT**



**AIM**

4. Prompt the user for a list of integers. For all values greater than 100, store ‘over’ instead.

**CODE**

list=[]

n=int(input("enter the limit:"))

print("enter integer numbers")

for i in range(0,n):

j=int(input())

if(j> 100):

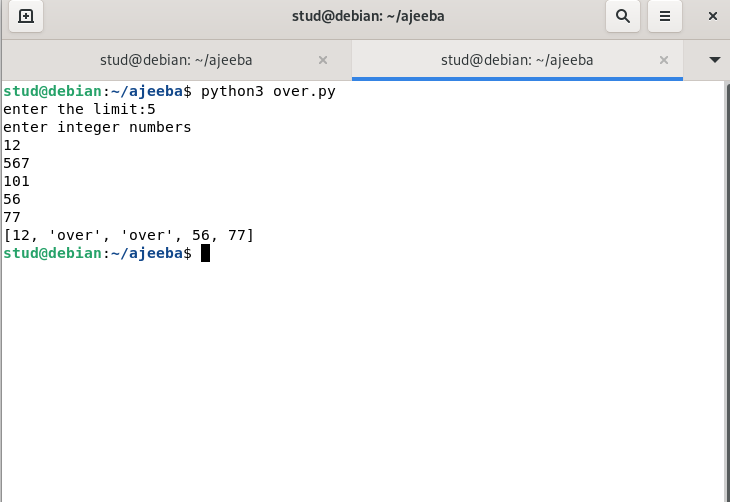
list.append("over")

else:

list.append(j)

print(list)

**OUTPUT**



**AIM**

5.Store a list of first names. Count the occurrences of ‘a’ within the list.

**CODE**

list=["anna","deepa","lali"]

count=0

for word in list:

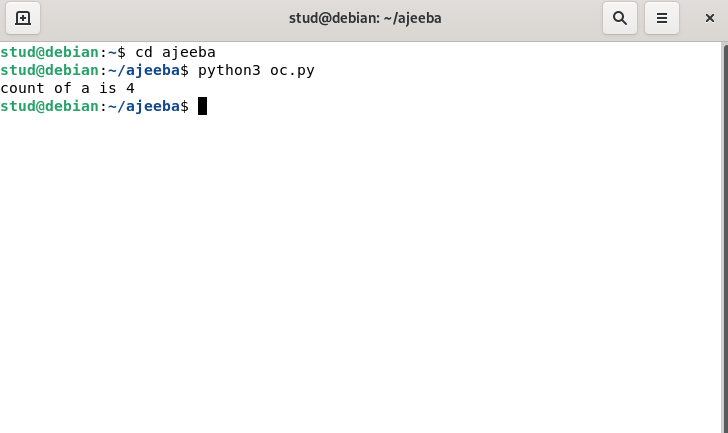
for i in word:

if (i=='a'):

count+=1

print("count of a is",count)

**OUTPUT**



**AIM**

**6. Enter 2 lists of integers.Check**

1. **whether list are of same length**
2. **whether list sums of same value**
3. **whether any value occur in both.**

**CODE**

l1=[1,2,3,4]

l2=[1,3,2]

print("List 1",l1)

print("List 2",l2)

x=len(l1)

y=len(l2)

if x==y:

print("List are of same length")

else:

print("Length of lists are different")

s1=0

s2=0

for i in range(x):

s1=s1+l1[i]

print("Sum of elements of List1:",s1)

for j in range(y):

s2=s2+l2[j]

print("Sum of elememts of List2:",s2)

if s1==s2:

print("Sum of list elements is same")

else:

print("Sum of list elements is not same")

print("Common elements are:")

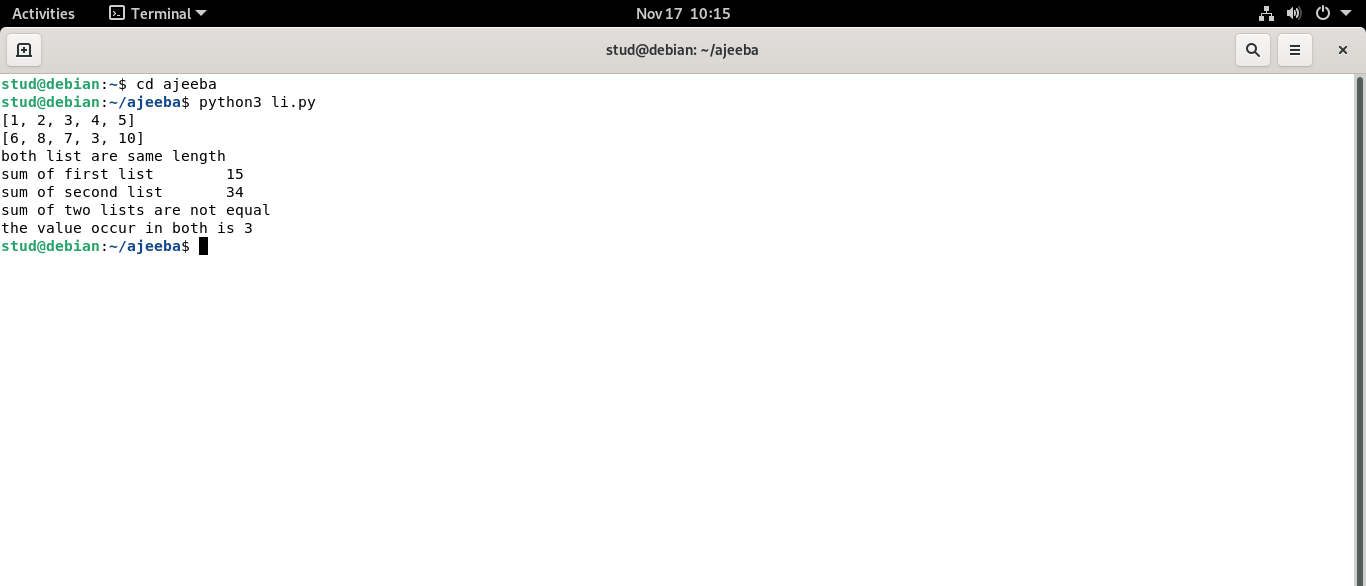
for i in range(x):

for j in range(y):

if l1[i]==l2[j]:

print(l1[i])

**OUTPUT**



**AIM**

**7.Get a string from an input string where all occurrences of first character replaced with ‘$’,except first character.[eg:onion->oni$n]**

**CODE**

str=input("Enter a string: ")

print("Original string is: ",str)

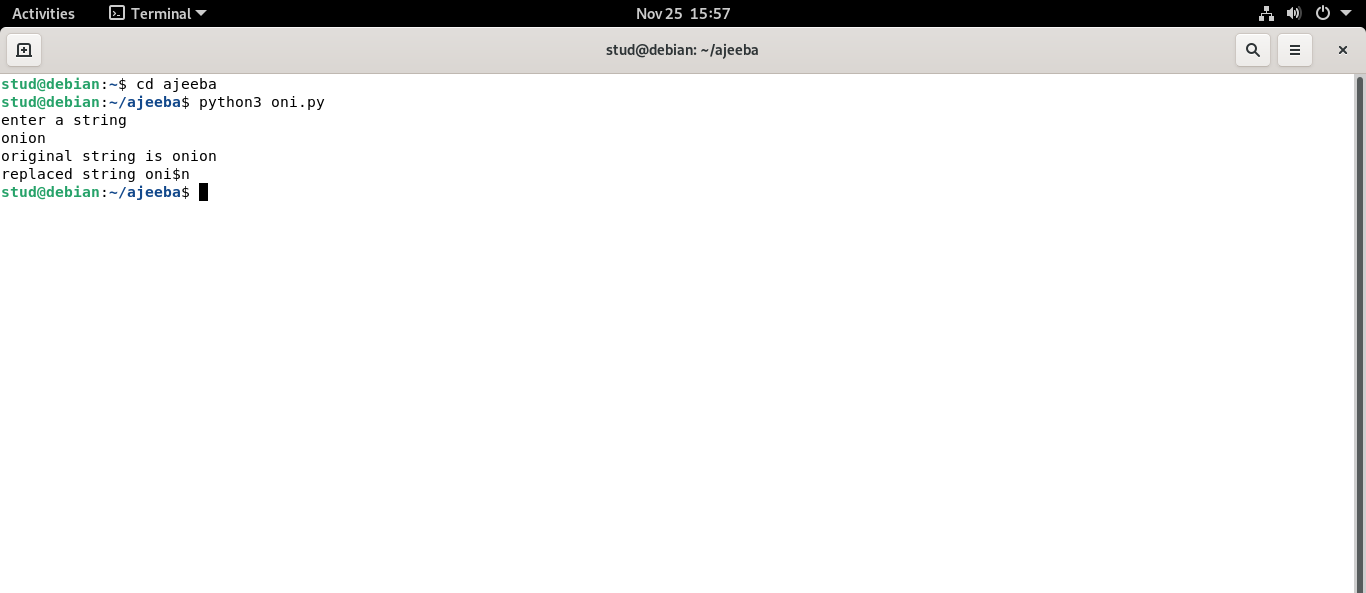
char=str[0]

str=str.replace(char,'$')

str=char+str[1:]

print("String: ",str)

**output**



**AIM**

**8.Create a string from given string where first and last characters exchanged. [eg:python->nythop]**

**CODE**

s=input("Enter a string: ")

t=s[0]

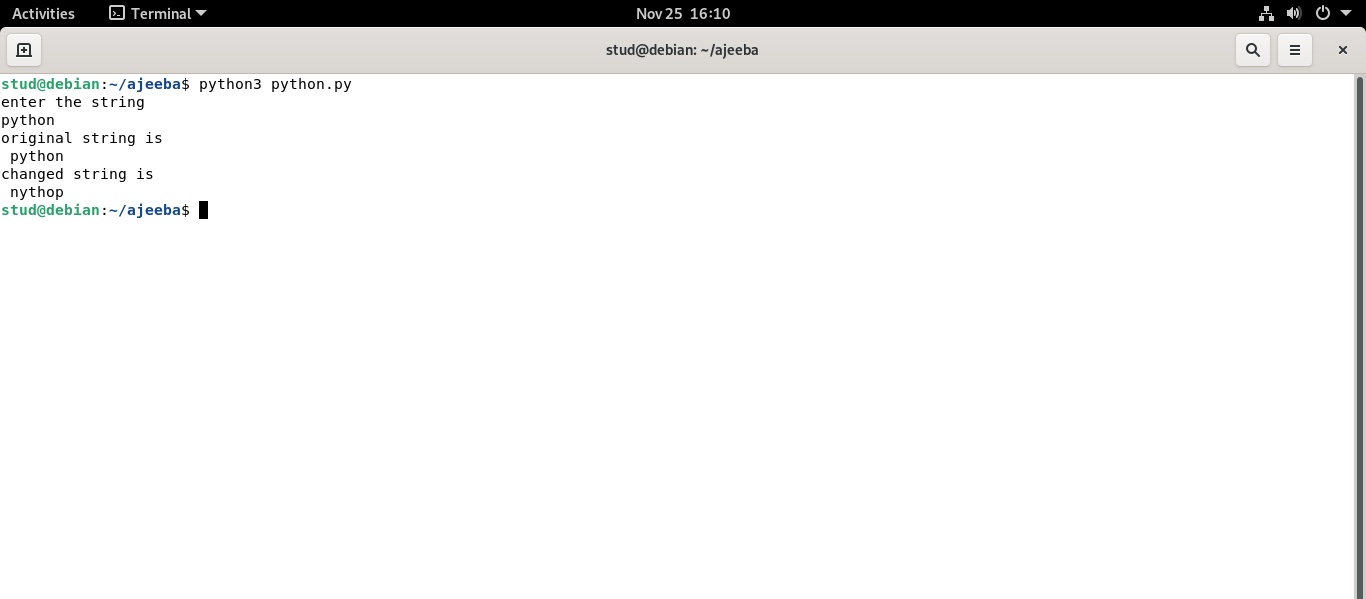
t1=s[-1]

n=len(s)

ns=t1+s[1:n-1]+t

print(ns)

**Output**



**AIM**

**9 .Accept the radius from the user and find the area of the circle.**

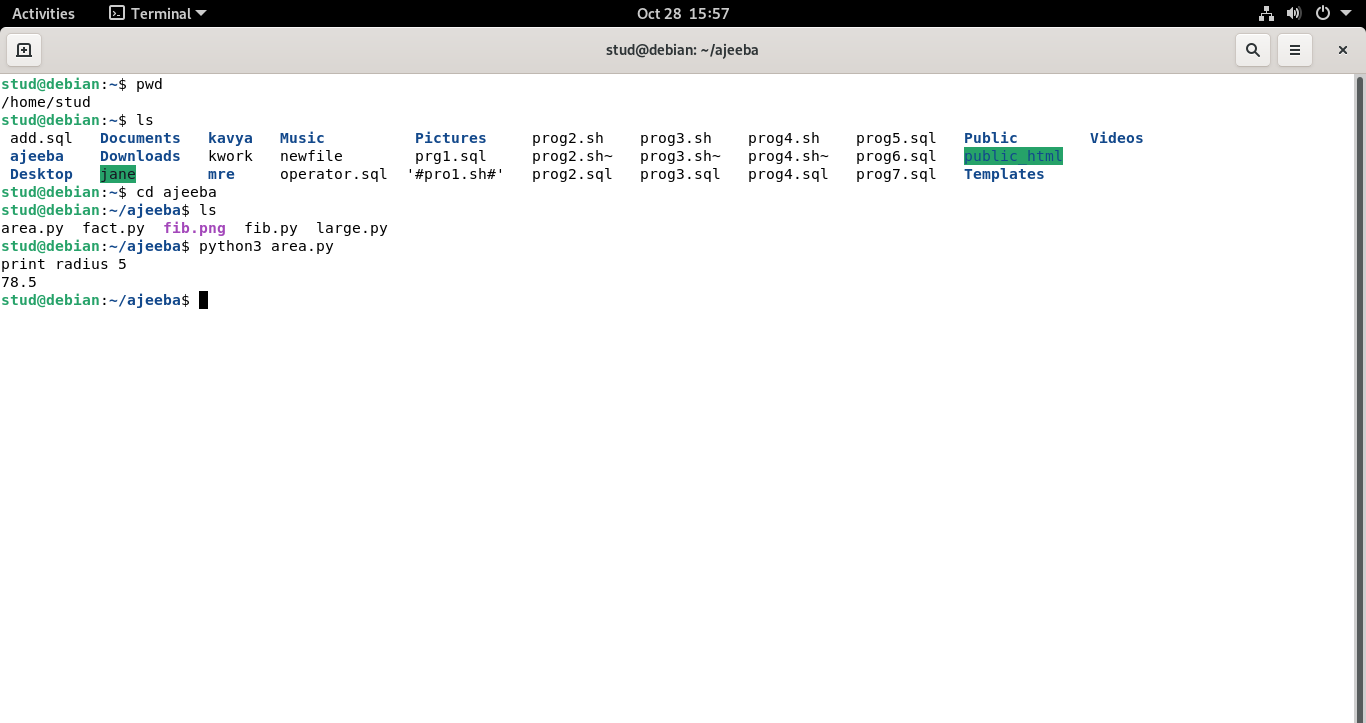
**Source code**

r=int(input('Enter the radius: '))

A=3.14\*r\*r

print(A)

**Output**



**AIM**

**10. Find the biggest of 3 numbers**

**CODE**

a=int(input('Enter first number:'))

b=int(input('Enter second number:'))

c=int(input('Enter third number:'))

if a>b and a>c:

print(a)

if b>a and b>c:

print(b)

if c>a and c>b:

print(c)

**OUPUT**



**AIM**

**11.Accept a file name from user and print extension of that.**

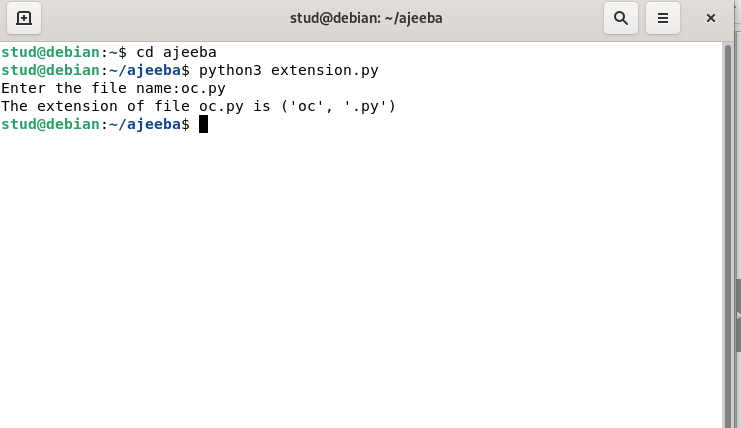
**CODE**

import os

a=input("Enter file name:")

print("The extension of file",a,"is",os.path.splitext(a))

**OUTPUT**



**AIM**

**12.Create a list of colors from comma-separated color names entered by user. Display first and last colors.**

**CODE**

colors=[]

str=(input("Enter color names:"))

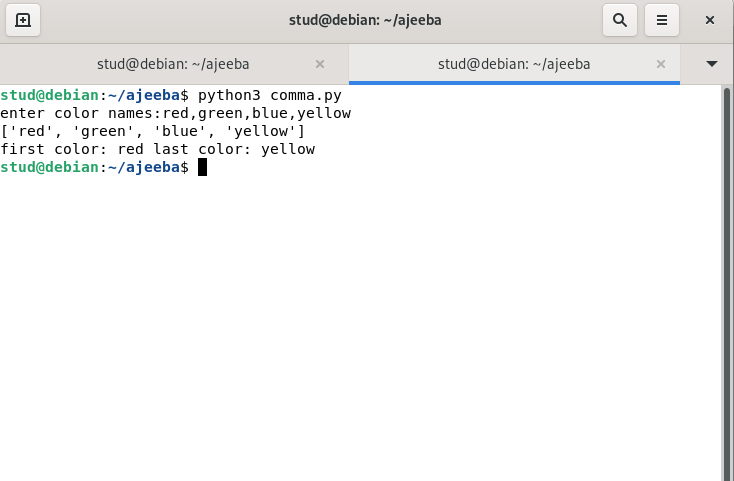
for i in str.split(','):

colors.append(i)

print(colors)

print("first color:",colors[0],"Last color:",colors[-1])

**OUTPUT**



**AIM**

**13.Accept an integer n and compute n+nn+nnn.**

**CODE**

n=int(input("Enter the number:"))

a=n\*1

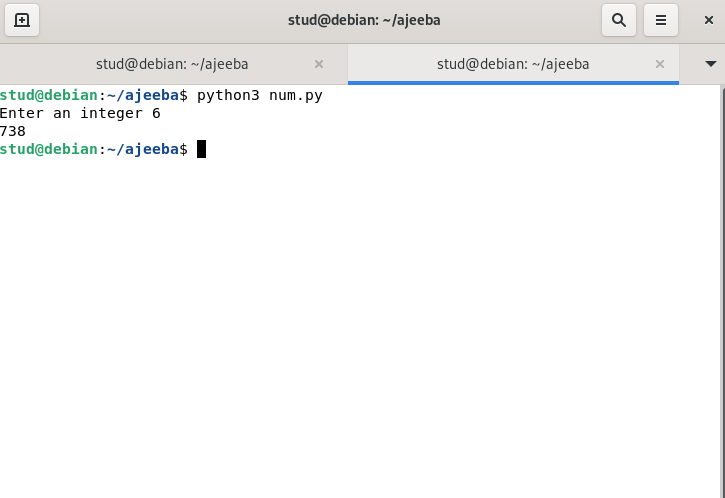
b=n\*11

c=n\*111

s=a+b+c

print(n,"+",n,"\*",n,"+",n,"\*",n,"\*",n,"=",s)

**OUTPUT**



**A**

**AIM**

**14.Print out all color from color-list1 not contained in color-list2**

**CODE**

l1=['red','green','blue','yellow','black']

l2=['red','green','yellow']

print(l1)

print(l2)

print("Colors that are not in l1:

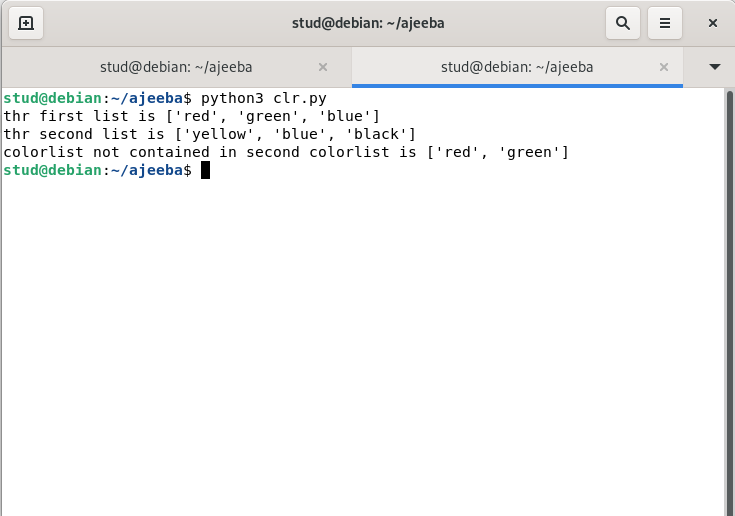
")

for i in l1:

if i not in l2:

print(i)

**OUTPUT**



**AIM**

**15. Create a single string separated with space from two strings by swapping the character at position 1.**

**CODE**

str1=input("Enter first string:")

str2=input("Enter second string:")

str3=str2[0]+str1[1:]+" "+str1[0]+str2[1:]

print(str3)

**OUTPUT**



**AIM**

**16.sort directory in ascending and descending order**

**CODE**

dict1={"a":1,"c":3,"d":2,"b":4}

l=list(dict1.items())

print(l)

l.sort()

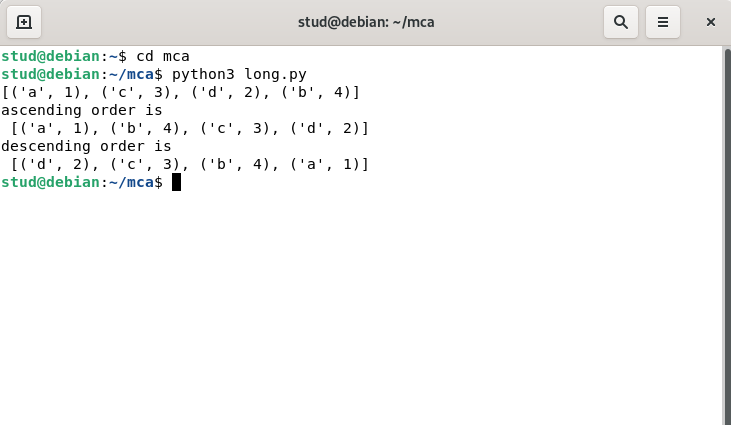
print("ascending order is\n",l)

l=list(dict1.items())

l.sort(reverse=True)

print("descending order is\n",l)

**OUTPUT**



**AIM**

**17.merge two dictionaries**

**CODE**

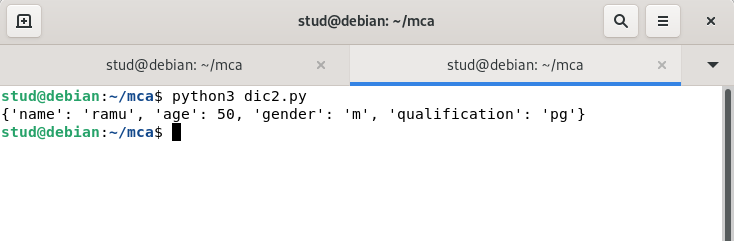
dict1={"name":"ramu","age":50}

dict2={"gender":"m","qualification":"pg"}

dict1.update(dict2)

print(dict1)

**OUTPUT**



**AIM**

**18. Find gcd of 2 numbers**

**CODE**

a=int(input("Enter first number: "))

b=int(input("Enter first number: "))

x=min(a,b)

gcd=0

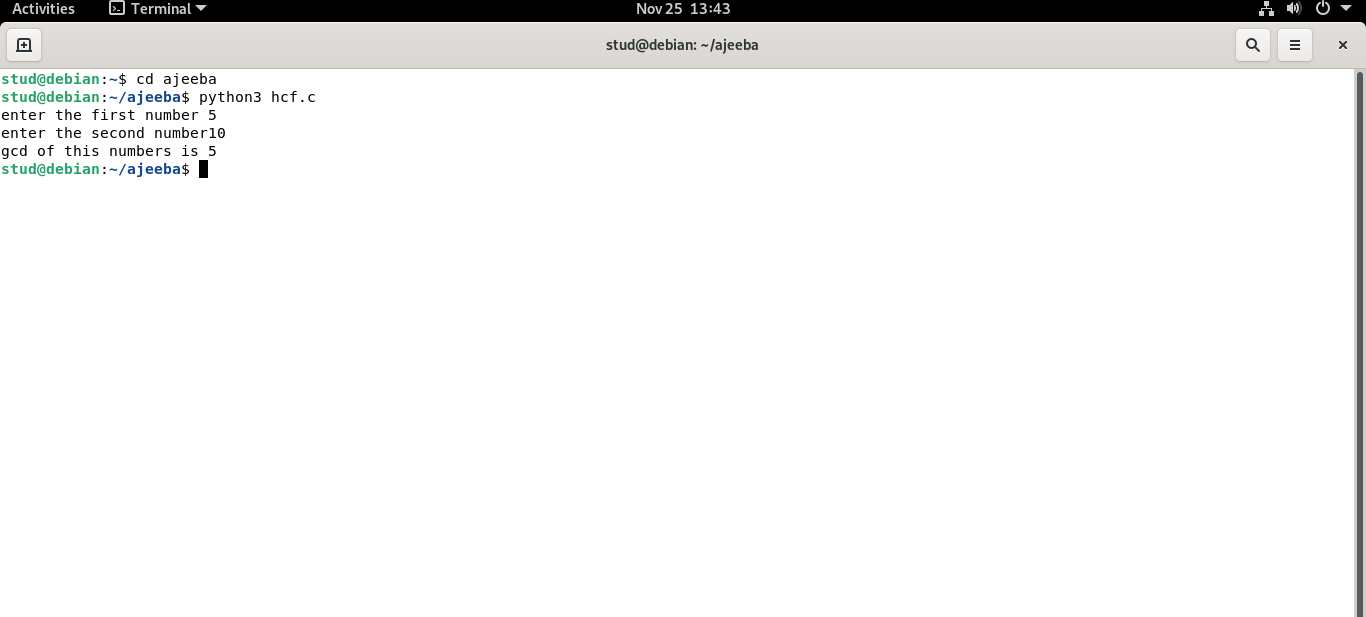
for i in range (1,x+1):

if((a%x==0) and (b%x==0)):

gcd=i

print("GCD is",i)

**OUTPUT**



**AIM**

**19.From a list of integers,create a list removing even numbers.**

**CODE**

l1=[1,2,3,4,5,6,7,8,9,10]

print(l1)

l2=[]

for i in range(len(l1)):

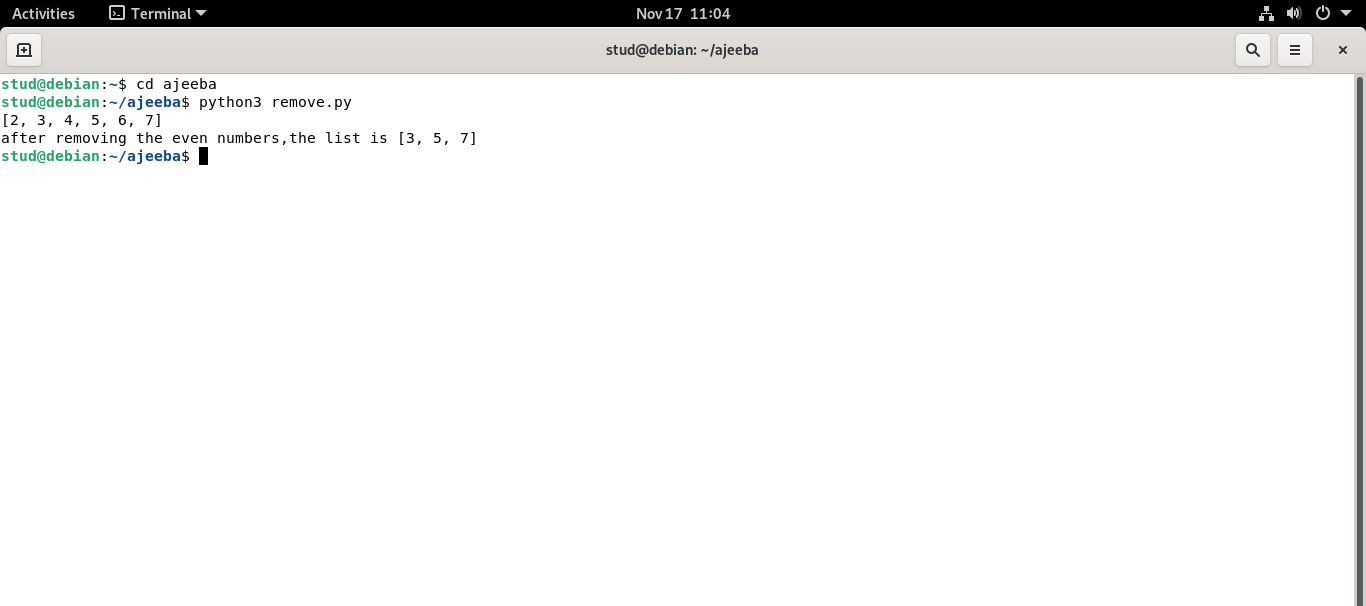
if l1[i]%2!=0:

l2.append(l1[i])

print("List after removing even elements")

print(l2)

**OUTPUT**



**AIM**

**20.Program to find the factorial of a number.**

**CODE**

n=int(input('Enter a number:'))

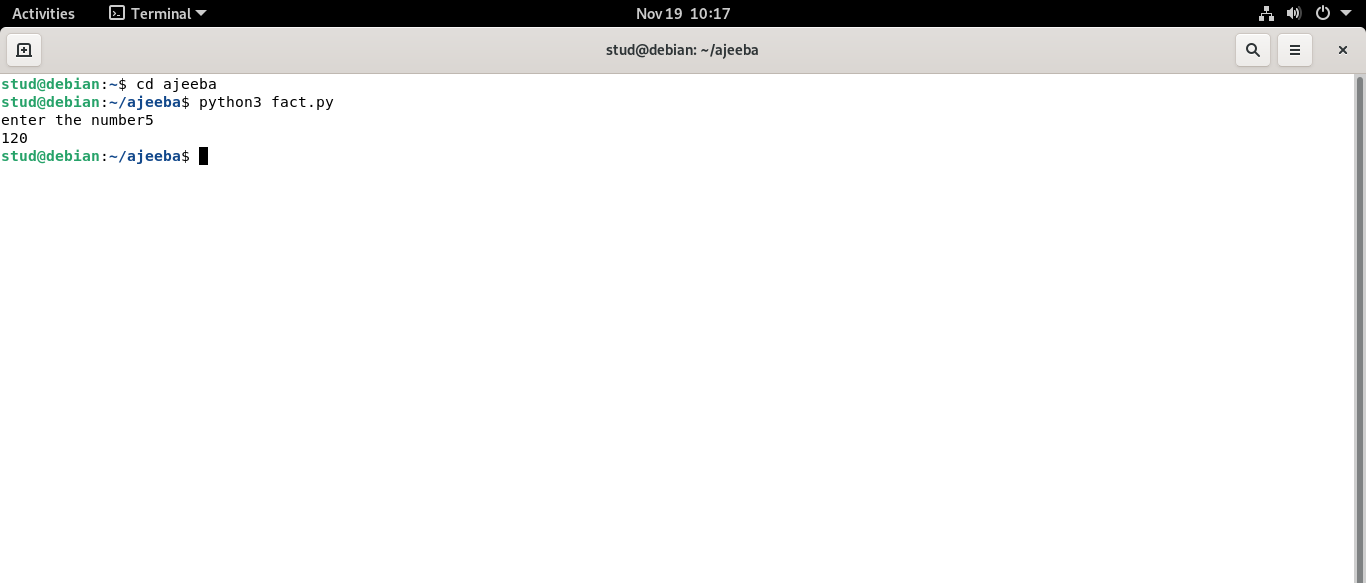
fact=1

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

fact=fact\*i

print(fact)

**OUTPUT**



**AIM**

**21.Generate fibonacci series of N terms.**

**Source code**

n=int(input('Enter a limit:'))

a=0

b=1

print(a)

print(b)

for i in range (2,n):

c=a+b

print(c)

a=b

b=c

**Output**



**AIM**

**22.Find the sum of all items in a list.**

**CODE**

l=[1,2,3,5]

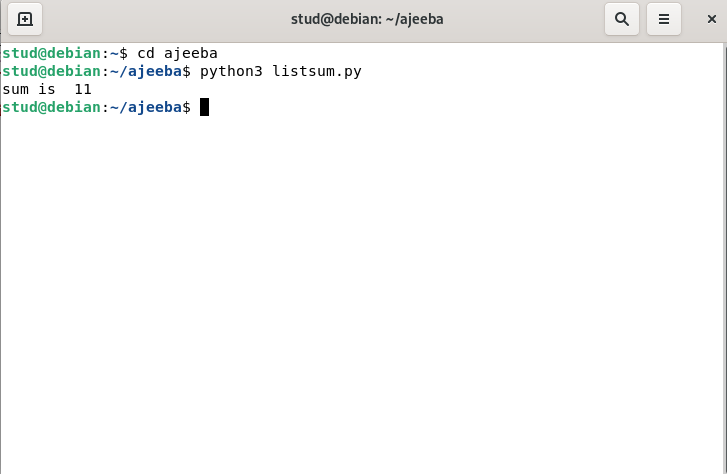
s=0;

for i in range(len(l)):

s=s+l[i]

print("sum is ",s)

**OUTPUT**



**AIM**

**23.Generate a list of four digit numbers in a given range with all their digits even and the number is a perfect square.**

**CODE**

limit1=1000

limit2=9999

list1=[]

for i in range(limit1,limit2):

j=i

digit=[]

while(i!=0):

digit.append(i%10)

i=int(i/10)

count=0

for n in digit:

if n%2==0:

count=count+1

if count==4:

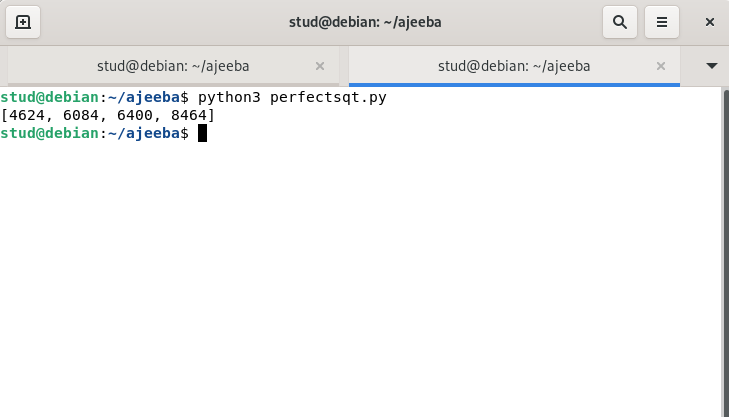
for k in range(31,100):

if((k\*\*2)==j):

list1.append(j)

print(k)

print(list1)

**OUTPUT**

**AIM**

**24.Display the given pyramid with step number accepted from user.**

**CODE**

n=int(input("Enter a number:"))

for j in range(0,n+1):

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

i=j\*i

print(i,end=" ")

print("\n")

**OUTPUT**



**AIM**

**25.count the number of characters (character frequency) in a string.**

**CODE**

string=input("Enter a string:")

list1=[]

for i in string:

if i not in list1:

list1.append(i)

for i in list1:

count=0

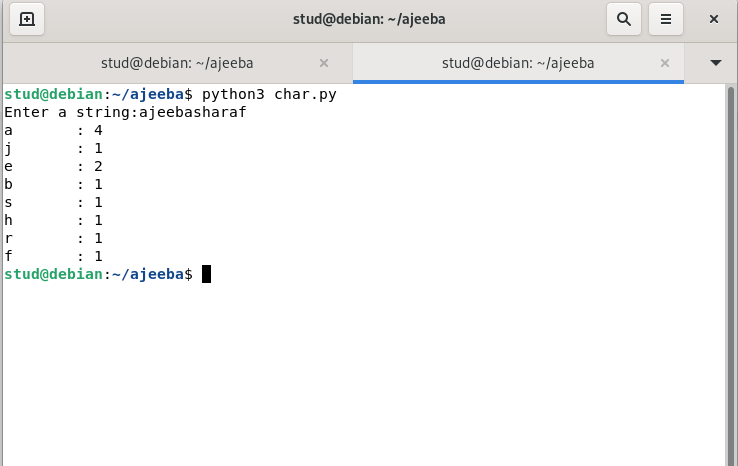
for j in string:

if(i==j):

count=count+1

print(i,"\t:",count)

**OUTPUT**



**26.Add ‘ing’ at the end of a given string. If it already ends with ‘ing’, then add ‘ly’.**

**CODE**

string=input("Enter a string:")

if(string[-3:]=="ing"):

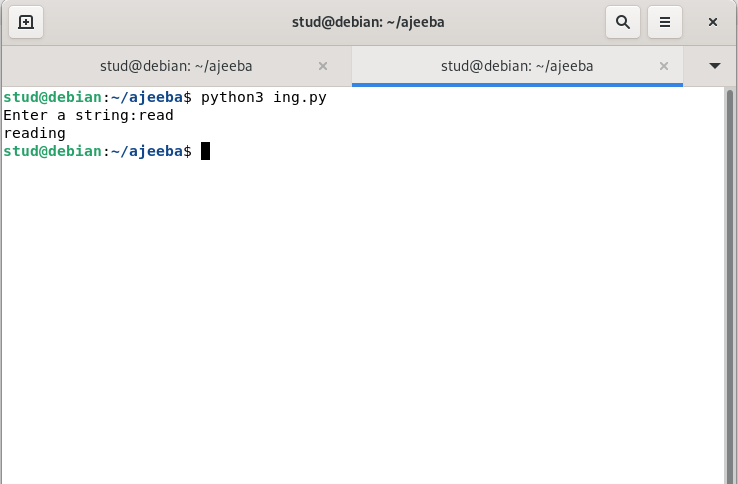
string+="ly"

else:

string+="ing"

print(string)

**OUTPUT**



**AIM**

**27.Accept a list of words and return length of longest word.**

**CODE**

lis=[]

n=int(input("Enter the range:"))

print("Enter the words:")

for i in range(0,n):

lis.append(input(""))

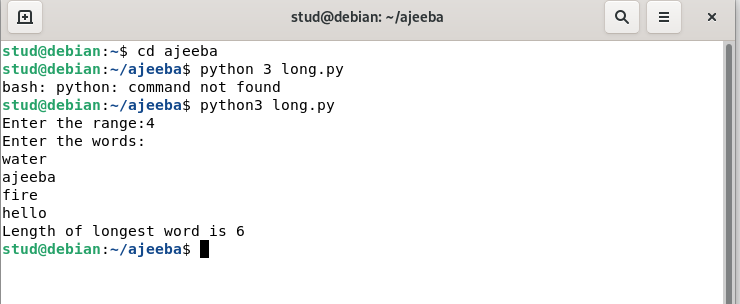
longest=lis[0]

for i in range(1,n):

if(len(lis[i])>len(longest)):

longest=lis[i]

print("Length of longest word is",len(longest))

**OUTPUT**

**AIM**

**28.Construct following pattern using nested loop.**

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

\* \* \* \*

\* \* \*

\* \*

\*

**CODE**

for i in range(1,6):

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

print("\*",end=" ")

print("\n")

for i in range(4,0,-1):

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

print("\*",end=" ")

print("\n")

**OUTPUT**



**AIM**

**29. Generate all factors of a number.**

**CODE**

n=int(input("Enter a number:"))

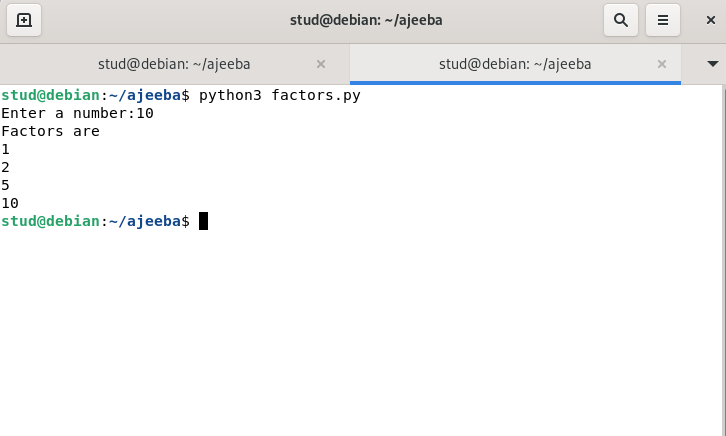
print("Factors are")

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

if(n%i==0):

print(i)

**OUTPUT**



**AIM**

30.**Create a package graphics with modules rectangle, circle and sub-package 3D-graphics with modules cuboid and sphere. Include methods to find area and perimeter of respective figures in each module. Write programs that finds area and perimeter of figures by different importing statements. (Include selective import of modules and import \* statements)**

**Source code**

**Graphice\circle.py**

from math import pi

def area\_circle(radius):

return pi\*radius\*radius

def perimeter\_circle(radius):

return 2\*pi\*radius

**Graphics\rectangle.py**

def area\_rec(length,width):

return length\*width

def perimeter\_rec(length,width):

return 2\*(length+width)

**Graphics\tdgraphics\cuboid.py**

def area\_cuboid(l,b,h):

return 2\*(l\*h + b\*h + l\*b)

def volume\_cuboid(l,b,h):

return l\*b\*h

**Graphics\tdgraphics\sphere.py**

from math import pi

def area\_sphere(radius):

return 4\*(pi\*radius\*radius)

def perimeter\_sphere(radius):

return 2\*pi\*radius

**graphics.py (driver code)**

import Graphics

from Graphics import circle,rectangle

from Graphics.tdgraphics import cuboid,sphere

from Graphics.circle import \*

print("Area of a circle with radius 10 is : ",circle.area\_circle(10))

print("Permeter of a circle with radius 10 is ",circle.perimeter\_circle(10))

print("\n")

print("Area of a Rectangle with length and width 10 is : ",rectangle.area\_rec(10,10))

print("Permeter of a Rectangle with length and width 10 is : ",rectangle.perimeter\_rec(10,10))

print("\n")

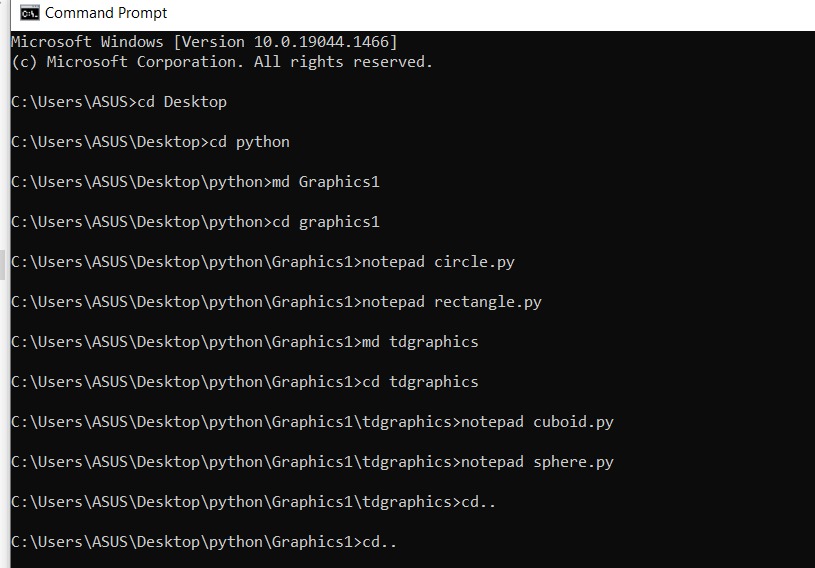
print("Area of a cuboid with length,width,height 10 is : ",cuboid.area\_cuboid(10,10,10))

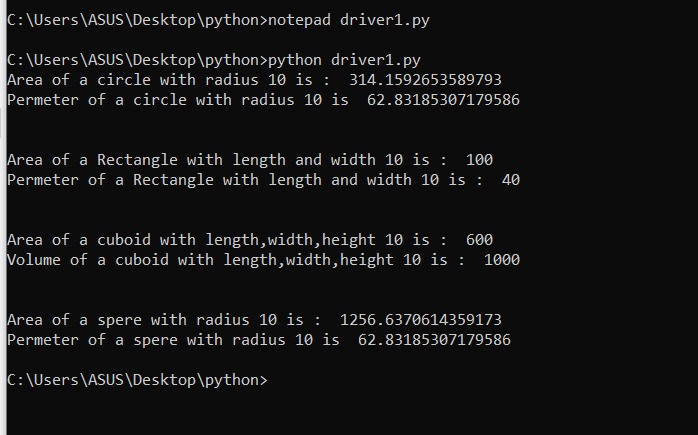
print("Volume of a cuboid with length,width,height 10 is : ",cuboid.volume\_cuboid(10,10,10))

print("\n")

print("Area of a spere with radius 10 is : ",sphere.area\_sphere(10))

print("Permeter of a spere with radius 10 is ",sphere.perimeter\_sphere(10))





**AIM**

**31.Create Rectangle class with attributes length and breadth and methods to find area and perimeter. Compare two Rectangle objects by their area.**

**CODE**

class Rectangle:

def \_\_init\_\_(self,length,breadth):

self.length=length;

self.breadth=breadth;

def area(self):

return self.length\*self.breadth

def perimeter(self):

return 2\*(self.length+self.breadth)

r1=Rectangle(10,2)

r2=Rectangle(5,9)

x=r1.area()

y=r2.area()

z=r1.perimeter()

w=r2.perimeter()

if(x>y):

print("area of rectangle 1 is larger")

else:

print("area of rectangle 2 is smaller")

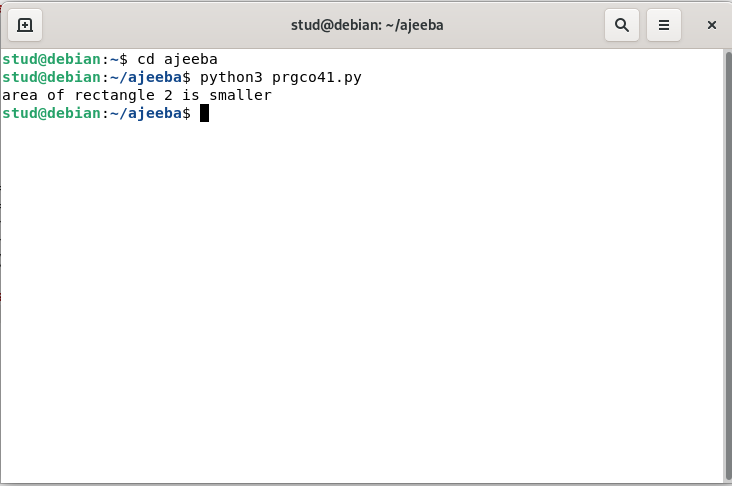
print("area of rectangle r1 is",x)

print("area of rectangle r2 is",y)

print("perimeter of rectangle r1 is",z)

print("perimeter of rectangle r2 is",w)

**OUTPUT**



**AIM**

**32.Create a Bank account with members account number, name, type of account and balance. Write constructor and methods to deposit at the bank and withdraw an amount from the bank.**

**CODE**

class Bank:

def \_\_init\_\_(self,acc\_no,name,acc\_type,balance):

self.acc\_no=acc\_no;

self.name=name;

self.acc\_type=acc\_type;

self.balance=balance;

def withdraw(self,x):

self.balance=self.balance-x

print("balance amount of person after withdrawal:",self.balance)

def deposit(self,y):

self.deposit=self.balance+y

print("balance amount of person after deposite:",self.balance)

def displ(self):

print("account no",self.acc\_no)

print("account holder",self.name)

print("account type",self.acc\_type)

print("account balance",self.balance)

per1=Bank(1234,"anju","savings",70000)

per2=Bank(1235,"adeena","current",5000)

per3=Bank(1236,"ajeeba","savings",76000)

per1.withdraw(1000)

per2.withdraw(2000)

per3.withdraw(4000)

per1.deposit(1000)

per2.deposit(3000)

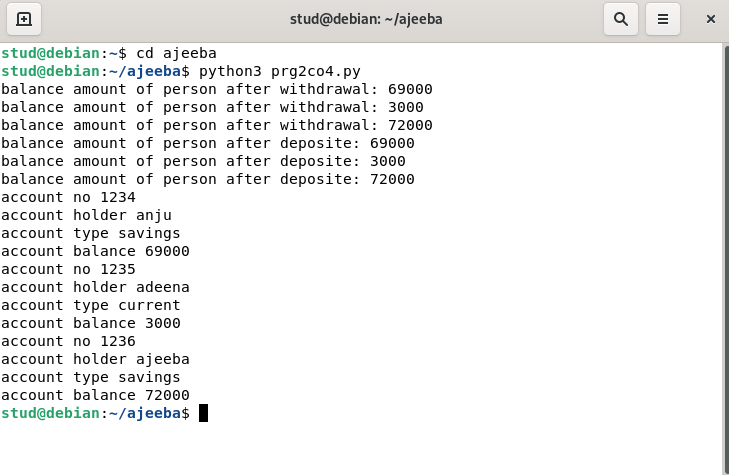
per3.deposit(2000)

per1.displ()

per2.displ()

per3.displ()

**OUTPUT**



**AIM**

**33.Create a class Rectangle with private attributes length and width. Overload ‘<’ operator to compare the area of 2 rectangles.**

**CODE**

class Rectangle:

def \_\_init\_\_(self,length,breadth):

self.length=length;

self.breadth=breadth;

def area(self):

return self.length\*self.breadth

def perimeter(self):

return 2\*(self.length+self.breadth)

def \_\_lt\_\_(self,r2):

if(self.length\*self.breadth<r2.length\*r2.breadth):

return True

else:

return False

r1=Rectangle(10,2)

r2=Rectangle(5,9)

x=r1.area()

y=r2.area()

z=r1.perimeter()

w=r2.perimeter()

print("area of rectangle r1 is",x)

print("area of rectangle r2 is",y)

print("perimeter of rectangle r1 is",z)

print("perimeter of rectangle r2 is",w)

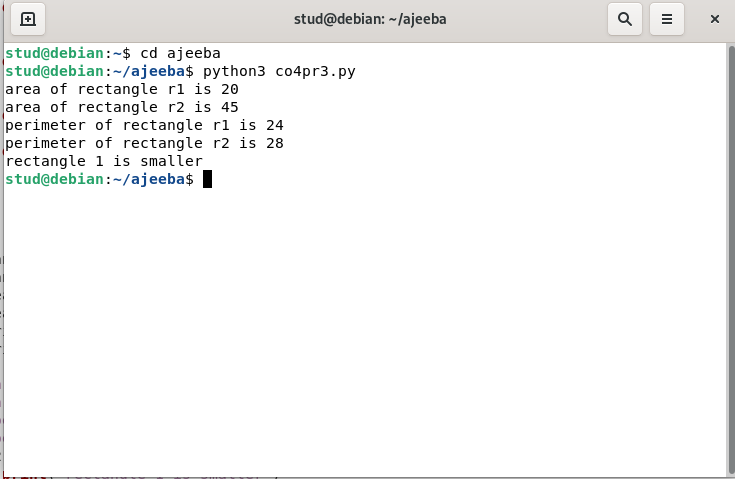
if(r1<r2):

print("rectangle 1 is smaller")

else:

print("rectangle 2 is smaller")

**OUTPUT**



**AIM**

**34.Create a class Time with private attributes hour, minute and second. Overload ‘+’ operator to find sum of 2 time.**

**CODE**

class Time:

def \_\_init\_\_(self,hr,min,sec):

self.\_\_hr=hr

self.\_\_min=min

self.\_\_sec=sec

def \_\_add\_\_(t1,t2):

hr=t1.\_\_hr+t2.\_\_hr

min=t1.\_\_min+t2.\_\_min

sec=t1.\_\_sec+t2.\_\_sec

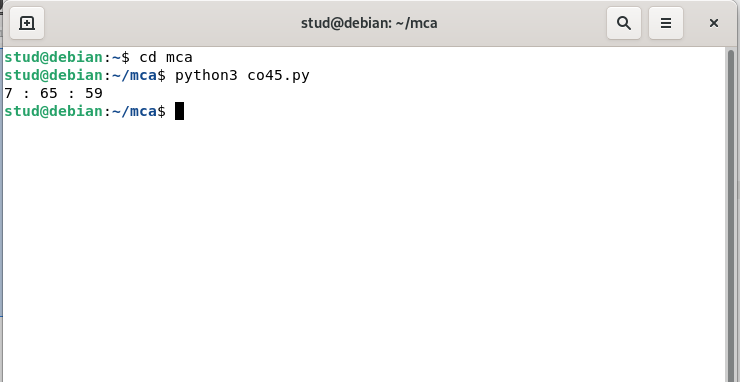
print(hr,":",min,":",sec)

t1=Time(3,45,56)

t2=Time(4,20,3)

t1+t2

**OUTPUT**



**AIM**

**35.Create a class Publisher (name). Derive class Book from Publisher with attributes title and author. Derive class Python from Book with attributes price and no\_of\_pages. Write a program that displays information about a Python book. Use base class constructor invocation and method overriding.**

**Source code**

class Publisher(object):

def \_\_init\_\_(self,name):

self.name=name

def display1(self):

print(self.title)

print(self.author)

class Book(Publisher):

def \_\_init\_\_(self,name,title,author):

super().\_\_init\_\_(name)

self.title=title

self.author=author

def display2(self):

#super().display1()

print(self.title)

print(self.author)

class Python(Book):

def \_\_init\_\_(self,name,title,author,price,no\_of\_pages):

super().\_\_init\_\_(name,title,author)

self.price=price

self.no\_of\_pages=no\_of\_pages

def display3(self):

super().display2()

print(self.price)

print(self.no\_of\_pages)

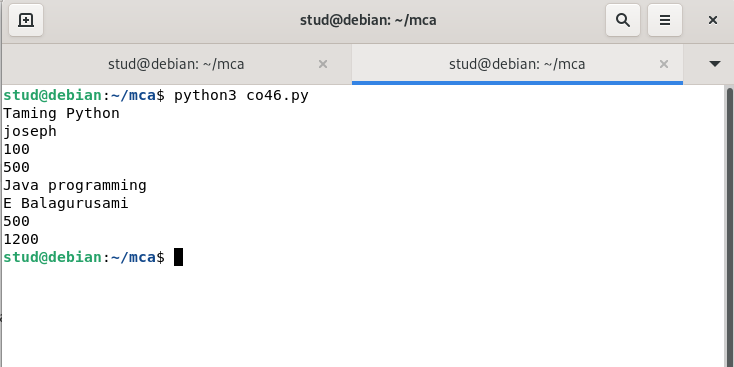
p=Python("ABC Publications","Taming Python","jeeva jose",100,500)

p.display3()

q=Python("XYZ Publications","Java programming","E Balagurusami",500,1200)

q.display3()

**OUTPUT**



**AIM**

36.Write a Python program to read a file line by line and store it into a list.

**Source code**

fp=open("text\_file.txt",'r')

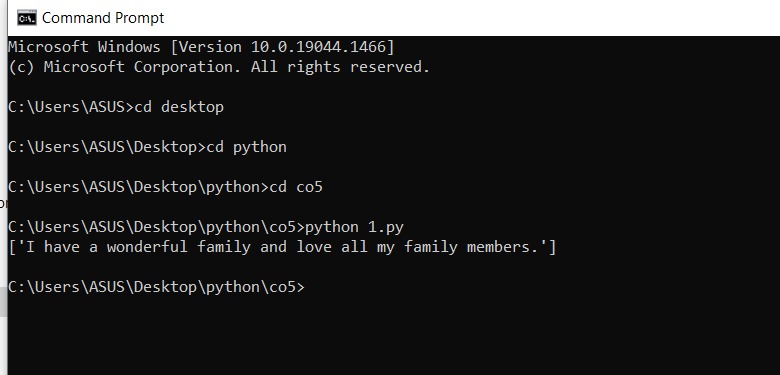
lines=[]

for line in fp:

lines.append(line.strip())

print(lines)

**Output**



**AIM**

**37.Write a Python program to read each row from a given csv file and print a list of strings.**

**CODE**

import csv

with open('people.csv', 'r') as file:

reader = csv.reader(file)

for row in reader:

print(row)

**OUTPUT**

