|  |
| --- |
| **FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY (FISAT)TM**  **HORMIS NAGAR, MOOKKANNOOR, ANGAMALY-683577**    **FOCUS ON EXCELLENCE**  **20MCA131 PROGRAMMING LAB LABORATORY RECORD**  **Name: ANJANA RAJEEV**  **Branch: MASTER OF COMPUTER APPLICATIONS**  **Semester: 1 Batch: A Roll No: 24**  **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 Anjana Rajeev* *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 of  Internal Examiner External Examiner |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CONTENT**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Sl No** | **Date of Experiment** | **Title of the Experiment** | **Page No:** | **Signature of Staff –In – Charge** | | 1 |  | Display future leap years from current year to a final year entered by user. |  |  | | 2 |  | List comprehensions:   1. Generate positive list of numbers from a given list of integers 2. Square of N numbers 3. Form a list of vowels selected from a given word 4. List ordinal value of each element of a word (Hint: use ord() to get ordinal values) |  |  | | 3 |  | Count the occurrences of each word in a line of text. |  |  | | 4 |  | Prompt the user for a list of integers. For all values greater than 100, store ‘over’ instead. |  |  | | 5 |  | Store a list of first names. Count the occurrences of ‘a’ within the list |  |  | | 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 |  |  | | 7 |  | Get a string from an input string where all occurrences of first character replaced with ‘$’, except first character.  [eg: onion ->oni$n] |  |  | | 8 |  | Create a string from given string where first and last characters exchanged. [eg: python - >nythop] |  |  | | 9 |  | Accept the radius from user and find area of circle. |  |  | | 10 |  | Find biggest of 3 numbers entered. |  |  | | 11 |  | Accept a file name from user and print extension of that. |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Sl No** | **Date of Experiment** | **Title of the Experiment** | **Page No:** | **Signature of Staff –In – Charge** | | 12 |  | Create a list of colors from comma-separated color names entered by user. Display first and last colors. |  |  | | 13 |  | Accept an integer n and compute n+nn+nnn. |  |  | | 14 |  | Print out all colors from color-list1 not contained in color-list2. |  |  | | 15 |  | Create a single string separated with space from two strings by swapping the character at position 1. |  |  | | 16 |  | Sort dictionary in ascending and descending order. |  |  | | 17 |  | Merge two dictionaries. |  |  | | 18 |  | Find gcd of 2 numbers. |  |  | | 19 |  | From a list of integers, create a list removing even numbers. |  |  | | 20 |  | Program to find the factorial of a number. |  |  | | 21 |  | Generate Fibonacci series of N terms. |  |  | | 22 |  | Find the sum of all items in a list |  |  | | 23 |  | Generate a list of four digit numbers in a given range with all their digits even and the number is a perfect square. |  |  | | 24 |  | Display the given pyramid with step number accepted from user.  Eg: N=4  1   1. 4 2. 6 9   8 12 16 |  |  | | 25 |  | Count the number of characters (character frequency) in a string. |  |  | | 26 |  | Add ‘ing’ at the end of a given string. If it already ends with ‘ing’, then add ‘ly’ |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Sl No** | **Date of Experiment** | **Title of the Experiment** | **Page No:** | **Signature of Staff –In – Charge** | | 27 |  | Accept a list of words and return length of longest word. |  |  | | 28 |  | Construct following pattern using nested loop  \*   * \* * \* \* * \* \* \* * \* \* \* \* * \* \* \* * \* \* * \* |  |  | | 29 |  | Generate all factors of a number. |  |  | | 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) |  |  | | 31 |  | Create Rectangle class with attributes length and breadth and methods to find area and perimeter. Compare two Rectangle objects by their area. |  |  | | 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. |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | | Sl No | Date of Experiment | Title of the Experiment | Page No: | Signature of Staff –In – Charge | | 33 |  | Create a class Rectangle with private attributes length and width. Overload ‘<’ operator to compare the area of 2 rectangles. |  |  | | 34 |  | Create a class Time with private attributes hour, minute and second. Overload ‘+’ operator to find sum of 2 time. |  |  | | 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. |  |  | | 36 |  | Write a Python program to read a file line by line and store it into a list. |  |  | | 37 |  | Write a Python program to read each row from a given csv file and print a list of string. |  |  | |  |  |  |  |  | |  |  |  |  |  | |  |  |  |  |  |   **PROGRAM 1**  **AIM** : Display future leap years from current year to a final year entered by user.  **SOURCE CODE**  startyear=2021  endyear=int(input('Enter the end year'))  print('The leap years are:')  for i in range(startyear,endyear):  if(i%4==0 and i %100!=0 or i%400==0):  print(i)  **OUTPUT**  **PROGRAM 2**  **AIM** : 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.  **SOURCE CODE**  (a)  list=[2,3,4-5,0,7,8] for num in list:  if(num>0):  print(num)  **OUTPUT**    **SOURCE CODE**  (b)  numbers = [1, 2, 3, 4, 5]  s= [number \*\* 2 for number in numbers]  print(s)  **OUTPUT**    **SOURCE CODE**  (c)  L=[]  s="India is my country"  for i in s:  if i in ("aeiouAEIOU"):  L.append(i)  print(L)      **OUTPUT**    **SOURCE CODE**  (d)  ordinal=input("Enter a name:")  print("The ASCII value of the letters in the word is")  for letter in ordinal:  n=ord(letter)  print(n)  **OUTPUT**      **PROGRAM 3**  **AIM :** Count the occurence of each word in a line of text.  **SOURCE 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**    **PROGRAM 4**  **AIM :** Prompt the user for a list of integers. For all values greater than 100, store ‘over’ instead.  **SOURCE CODE**  list=[]  while True:  n=int(input("Enter the value:"))  if(n<=100):  list.append(n)  else:  list.append('over')  print(list)  **OUTPUT**    **PROGRAM 5**  **AIM :** Store a list of first names. Count the occurrence of ‘a’ within the list.  **SOURCE CODE**  l = ['a','j','aaa','aa']  print('occurance of a')  count = 0  for i in l:  num=i.count('a')  count=count+num  print(count)  **OUTPUT**    **PROGRAM 6**  **AIM :** Enter two list of integers. Check  (a) Whether they are of same length.  (b) Whether list sums to same value.  (c) Whether any value occur in both.  **SOURCE CODE**  l1=[5,6,3,7]  l2=[2,1,7,10,8]  x=len(l1)  y=len(l2)  if x==y:  print("The list is of same length")  else:  print("The list is of different length")  sum1=0  sum2=0  for i in range(len(l1)):  sum1=sum1+l1[i]  print("The sum of list1 is:",sum1)  for j in range(len(l2)):  sum2=sum2+l2[j]  print("The sum of list2 is:",sum2)  if sum1=sum2:  print("The sum of list1 is equal to list2")  else:  print("The sum of list1 is not equal to list2")  for i in range(x):  for j in range(y):  if l1[i]==l2[j]:  print(l1[i],"and",l2[j],"occur in both")  **OUTPUT**      **PROGRAM 7**  **AIM :** Get a string from an input string where all occurence of first charecter replaced with ‘$’, except first character.  [onion -> oni$n]  **SOURCE CODE**  ch=input("Enter a string:")  f=ch[0]  print(ch[0],end="")  f=f.lower()  for i in range(1,len(ch)):  if ch[i]==f:  print("$", end="")  else:  print(ch[i],end="")  **OUTPUT**    **PROGRAM 8**  **AIM :** Create a string from given string where first and last characters exchanged.  [eg : Python ->nythoP]  **SOURCE CODE**  s="python"  t=s[0]  t1=s[-1]  n=len(s)  ns=t1+s[1:n-1]+t  print(ns)  **OUTPUT**    **PROGRAM 9**  **AIM:** Accept the radius from user and find area of circle.  **SOURCE CODE**  p=int (input("Enter the radius"))  ar=3.14\*p\*p  print("Area=",ar)  **OUTPUT**    **PROGRAM 10**  **AIM :** Find biggest of 3 numbers entered .  **SOURCE CODE**  print("Enter 3 Numbers :")  a=int(input(""))  b=int(input(""))  c=int(input(""))  if (a>b)&(a>c):  print(a,"is biggest")  if (b>a)&(b>c):  print(b,"is biggest")  if (c>a)&(c>b):  print(c,"is biggest")  **OUTPUT**    **PROGRAM 11**  **AIM:** Accept a file name from user and print extension for that.  **SOURCE CODE**  import os  a=input("enter the filename : ")  print("The extension of file",a, "is",os.path.splitext(a))  **OUTPUT**    **PROGRAM 12**  **AIM:** Create a list of colors from comma-separated colour names entered by user. Display first and last colours.  **SOURCE CODE**  n=int(input("Enter the size:"))  l=[]  for i in range(0,n):  color=input("Enter Your Choice:")  l.append(color)  print(l[0])  print(l[n-1])  **OUTPUT**      **PROGRAM 13**  **AIM:** Accept an integer n and compute n+nn+nnn.  **SOURCE CODE**  num=input("Enter a number:")  dum1=num+num+num  dum2=num+num  dum3=num  print(int(dum1)+int(dum2)+int(dum3))  **OUTPUT**    **PROGRAM 14**  **AIM :** Print out all colours from color list1 not contained in color list2.  **SOURCE CODE**  a=['red','green','blue','yellow']  b=['orange','pink','yellow','blue','violet']  for i in a:  if i not in b:  print(i)  **OUTPUT**    **PROGRAM 15**  **AIM :** Create a single string separated with space from two strings by swapping the character at position 1.  **SOURCE CODE**  string1="Fisat"  string2="Ankamaly"  f1=string1[0]  f2=string2[0]  string=f2+string1[1:]+" "+f1+string2[1:]  print("The new string is :",string)  **OUTPUT**      **PROGRAM 16**  **AIM :** Sort dictinary in ascending and descending order.  **SOURCE 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) 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**    **PROGRAM 17**  **AIM :** Merge two dictionaries.  **SOURCE CODE**  D1={"name":"anju","age":"21"}  D2={"sex":"female","qualification":"bsc cs"}  D1.update(D2)  print(D1)  **OUTPUT**        **PROGRAM 18**  **AIM :** Find gcd of two numbers.  **SOURCE CODE**  x=int(input("Enter the first number:"))  y=int(input("Enter the second number:"))  if x>y:  small=y  else:  small=x  for i in range(1,small+1):  if x%i==0 and y%i==0:  hcf=i  print(hcf)  **OUTPUT**    **PROGRAM 19**  **AIM :** Form a list of integers, create a list removing even numbers.  **SOURCE CODE**  list1=[]  list2=[]  n=int(input("Enter the list size :"))  for i in range(0,n):  list1.append(int(input("Enter an element :")))  print("The list is\t",list1)  for i in list1:  if i%2!=0:  list2.append(i)  print("The odd list is\t",list2)  **OUTPUT**    **PROGRAM 20**  **AIM :** Program to find the factorial of a number.  **SOURCE CODE**  n=int(input("Enter a Number :"))  factorial=1  for i in range(1,n+1):  factorial=factorial\*i  print("Factorial of",n,"=",factorial)  **OUTPUT**      **PROGRAM 21**  **AIM :** Generate Fibonacci series of N terms.  **SOURCE CODE**  n=int(input("Enter a Number :"))  print("The first",n,"fibonacci seriers is :")  f1=0  f2=1  for i in range(0,n):  print(f1)  f3=f1  f1=f1+f2  f2=f3  **OUTPUT**    **PROGRAM 22**  **AIM :** Find the sum of all items in a list.  **SOURCE CODE**  list1=[1,2,3,4,5,6,7]  summ=0  for i in list1:  summ=summ+i  print("sum=",summ)  **OUTPUT**    **PROGRAM 23**  **AIM :** Generate a list of four digit numbers in a given range with all their digits even and the number is a perfect square.  **SOURCE 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(list1)  **OUTPUT**    **PROGRAM 24**  **AIM :** Display the given pyramid with step numbers accepted from user.  Eg : 4  1  2 4  3 6 9  4 8 12 16  **SOURCE CODE**  n=int(input("Enter a number :"))  for i in range(1,n+1):  for j in range(i,(i\*i)+1,i):  print(j,"\t",end="")  print("\n")  **OUTPUT**    **PROGRAM 25**  **AIM :** Count the number of characters (character frequency) in a string.  **SOURCE CODE**  string=input("Enter a string :")  ulist=[]  for i in string:  if i not in ulist:  ulist.append(i)  for i in ulist:  count=0  for j in string:  if(i==j):  count+=1  print(i,"\t:",count)  **OUTPUT**    **PROGRAM 26**  **AIM :** Add ‘ing’ at the end of a given string.If it already ends with ‘ing’ , then add ‘ly’.  **SOURCE CODE**  string=input("Enter a string :")  if(string[-3:]=="ing"):  string+="ly"  else:  string+="ing"  print(string)  **OUTPUT**    **PROGRAM 27**  **AIM :** Accept a list of words and return length of longest word.  **SOURCE CODE**  wlist=[]  print("Enter 5 words :")  for i in range(0,5):  wlist.append(input(""))  temp=wlist[0]  for i in range(1,5):  if len(wlist[i])>len(temp):  temp=wlist[i]  print("Length of longest word is",len(temp))  **OUTPUT**    **PROGRAM 28**  **AIM :** Construct following pattern using nested loop.  \*  \*\*  \*\*\*  \*\*\*\*  \*\*\*\*\*  \*\*\*\*  \*\*\*  \*\*  \*  **SOURCE CODE**  for i in range(1,6):  for j in range(0,i):  print("\*",end="")  print("\n")  for i in range(4,0,-1):  for j in range(0,i):  print("\*",end="")  print("\n")  **OUTPUT**    **PROGRAM 29**  **AIM :** Generate all factors of a number.  **SOURCE CODE**  n=int(input("Enter a number :"))  print("The factors are :")  for i in range(1,n+1):  if(n%i)==0:  print(i)  **OUTPUT**        **PROGRAM 30**  **AIM :** Create a package graphics with modules rectangle ,circle and sub-package 3D graphics with module cuboid and sphere. Include methods to find area and perimeter of respective figures in each modules. Write programs that finds area and perimeter of figures by different importing statements.  **Terminal Commands**    **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))  **OUTPUT**    **PROGRAM 31**  **AIM :** Create Rectangle class with attributes length and breadth and methods to find area and perimeter. Compare two rectangle objects by their area.  **SOURCE CODE**  Class Rectangle:  def \_\_init\_\_(self,l,b):  self.l=l  self.b=b  def area(self):  return (self.l\*self.b)  def perimeter(self):  return (2\*(self.l+self.b))  def print(self):  print(self.area)    r1=Rectangle(10,2)  r2=Rectangle(5,8)  x=r1.area()  y=r2.area()  print("area of first rectangle is",x)  print("area of second rectangle is",y)  p=r1.perimeter()  q=r2.perimeter()  print("perimeter of first rectangle is",p)  print("perimeter of second rectangle is",q)  print  if(x>y):  print('Area of first rectangle greater than second rectangle');  else:  print('Area of second rectangle greater than first rectangle');  **OUTPUT**    **PROGRAM 32**  **AIM:** Create a Bank account with members account number, name, type of account and balance. Write constructor and methods to deposite at the bank and withdraw an amount from the bank.  **SOURCE CODE**  class Bank:  def \_\_init\_\_(self,accno,aname,a\_type,bal):  self.accno=accno  self.aname=aname  self.a\_type=a\_type  self.bal=bal  def withdraw(self,x):  self.bal=self.bal-x  def deposit(self,y):  self.bal=self.bal+y  def print(self):  print(self.accno,self.aname,self.a\_type,self.bal)  acc1=Bank(2435,'anju','sbi',10000)  acc2=Bank(5436,'aju','federal',22000)  acc1.withdraw(1000)  acc1.deposit(4000)  acc2.withdraw(2500)  acc1.deposit(8000)  acc1.print()  acc2.print()  **OUTPUT**    **PROGRAM 33**  **AIM :** Create a class Rectangle with private attributes length and width. Overload ‘<’ operator to compare the area of two rectangles.  **SOURCE CODE**  class Rectangle:  def \_\_init\_\_(self,ln,br):  self.ln=ln  self.br=br  def area(self):  p=self.ln\*self.br  return p  def \_\_lt\_\_(self,r1):  if r2.area()<r1.area():  return r2.area()  else:  return r1.area()  def perim(self):  q=2\*(self.ln+self.br)  return q  a=int(input("Enter length of the first rectangle:"))  b=int(input("Enter breadth of the first rectangle:"))  r1=Rectangle(a,b)  a=int(input("Enter length of the second rectangle:"))  b=int(input("Enter breadth of the second rectangle:"))  r2=Rectangle(a,b)  print("Perimeter of first rectangle= ",r1.perim())  print("Perimeter of second rectangle= ",r2.perim())  print("Least one is:",r1<r2)  **OUTPUT**    **PROGRAM 34**  **AIM :** Create a class Time with private attributes hour,minute and second. Overload ‘+’ operator to find sum of two time.  **SOURCE 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**    **PROGRAM 35**  **AIM :** 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 overreading.  **SOURCE CODE**  class Publisher:  def \_\_init\_\_(self,name):  self.name=name  class Book(Publisher):  def \_\_init\_\_(self,name,title,auther):  super().\_\_init\_\_(name)  self.title=title  self.auther=auther  def print\_function(self):  print("This Fuction is a member fuction of class Publisher")  class Python(Book):  def \_\_init\_\_(self,name,title,auther,price,nop):  super().\_\_init\_\_(name,title,auther)  self.price=price  self.nop=nop  def print\_function(self):  print("Name :",self.name)  print("Title :",self.title)  print("Auther :",self.auther)  print("Price :",self.price)  print("Number of Pages :",self.nop)  p1=Python("Text book","Python Programming","Mr.abc",100,500)  p1.print\_function()  p2=Book("a","b","c")  p2.print\_function()  **OUTPUT**    **PROGRAM 36**  **AIM:** Write a 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)  **text.txt**  "Cats, also called domestic cats are small, carnivorous mammals, of the family Felidae.  Domestic cats are often called 'house cats' when kept as indoor pets.  Cats have been domesticated for nearly 10,000 years.  They are one of the most popular pets in the world."  **OUTPUT**    **PROGRAM 37**  **AIM:** Write a Python program to read each row from a given csv file and print a list of strings.  **SOURCE CODE**  import csv  with open('people.csv', 'r') as file:  reader = csv.reader(file)  for row in reader:  print(row)  **text.csv**  Name,Designation,Salary  Jessy,Manager,90000  Tom,Clerk,40000  Alfred,Assistant Manager,70000  **OUTPUT** |