Experiment No:21 Lab Cycle:03 Date:10-10-2024

FIBONACCI SERIES

AIM:

Write a Python program to generate the Fibonacci series up to N numbers, where N is the limit provided by the user.

ALGORITHM:

1.Accept the input value limit from the user to determine the number of terms in the Fibonacci series.

2. Initialize a list fib with the first two numbers of the Fibonacci series, i.e., [0, 1].

3. Loop from 2 to limit - 1 (because the first two Fibonacci numbers are already present):

Calculate the next Fibonacci number by adding the last two numbers in the fib list.

Append the calculated number to the fib list.

5.Print the entire Fibonacci series stored in the fib list.

SOURCE CODE:

```
limit = int(input("Enter the limit: "))
fib = [0, 1]
for i in range(2, limit):
fib.append(fib[-1] + fib[-2])
print("Fibonacci series:", fib)
```

OUTPUT:

Enter the limit: 5

Fibonacci series: [0, 1, 1, 2, 3]

RESULT:

The program ran successfully and output is verified.

Experiment No:22 Lab Cycle:03 Date:10-10-2024

SUM OF LIST

AIM:

To find the sum of all items in a list where the list is provided by the user.

ALGORITHM:

1. Accept the list of numbers as input from the user.
2. Use the sum() function to calculate the sum of all items in the list.
3. Display the sum of the list.

SOURCE CODE:

```
list_input = input("Enter numbers separated by spaces: ").split()
list_numbers = [int(i) for i in list_input]
print(f"Sum of Lists: {sum(list_numbers)}")
```

OUTPUT:

```
Enter numbers separated by spaces: 1 2 3 4 5 6 7 8 9 10
Sum of Lists: 55
```

RESULT:

The program ran successfully and output is verified

Experiment No:23 Lab Cycle:03 Date:10-10-2024

PERFECT SQUARE

AIM:

Write a Python program to generate a list of four-digit numbers within a given range where all the digits are even, and the number is a perfect square.

ALGORITHM:

1. Accept the starting and ending range values as input from the user.

2. Initialize an empty list out to store the numbers that meet the conditions.

3. Check if the starting and ending ranges are valid (both positive, and end greater than start)

4. Loop through the numbers in the given range:

   Check if the number is a perfect square.

   Check if all digits of the number are even.

   If both conditions are satisfied, append the number to the out list.

5. Print the list of numbers that satisfy the conditions. If the ranges are invalid, print a message indicating so.

SOURCE CODE:

```
import math

start = int(input("Enter the starting range: "))

end = int(input("Enter the ending range: "))

out = []

if start >= 1 and end >= 1 and end > start:

        for i in range(start, end + 1):

if int(math.sqrt(i))**2 == i and all(int(digit) % 2 == 0 for digit in str(i)):

        out.append(i)

        print(f"Output: ", out)

else:

        print("Give Valid Ranges")
```

OUTPUT:

Enter the starting range: 1000

Enter the ending range: 9999

Output: [1600, 3600, 6400, 8400]

RESULT:

The program ran successfully and output is verified

Experiment No:24 Lab Cycle:03 Date:10-10-2024

NUMBER PYRAMID

AIM:

Write a Python program to display the given pyramid with step number accepted by user

eg: n=4

1

24

369

481216

ALGORITHM:

1. Accept the limit value n from the user to determine the number of rows in the pyramid.

2. Loop through each row (from 1 to n):

For each row, initialize count = 1.

Loop through the number of elements in the row (equal to the row number) and print the product of the current row number and the count value.

Increment count after each element is printed.

3.Print the pyramid structure.

SOURCE CODE:

```
n = int(input("Enter the limit: "))


for i in range(1, n + 1):
    count = 1
    for j in range(0, i):
        print(i * count, end=" ")
        count += 1
    print("\n")
```

OUTPUT:

Enter the limit: 4

1

24

369

481216

RESULT:

The program ran successfully and output is verified

Experiment No:25 Lab Cycle:03 Date:10-10-2024

CHARACTER FREQUENCY

AIM:

Write a Python program to count the number of character (character frequency) in a string.

ALGORITHM:

1. Start
2. Store a string(str) and initialise an empty frequency
3. For each character in string

if char in freq:

freq[char]+=1, increment the count of character

else:

Add a character and count to 1

4. Print the dictionary
5. Stop

SOURCE CODE:

```
str=input("Enter a string:")
freq={}
for char in str:
    if char in freq:
        freq[char]+=1
    else:
        freq[char]=1
print("Character frequencies:")
for char,count in freq.items():
    print(f"'{char}':{count}")
```

OUTPUT:

Enter a string:hello

Character frequencies:

'h':1

'e':1

'l':2

'o':1

RESULT:

The program ran successfully and output is verified

Experiment No:26 Lab Cycle:03 Date:10-10-2024

END OF A STRING

AIM:

Write a Python program to add "ing" at the end of a given string. If the string already ends with "ing", add "ly" instead.

ALGORITHM:

1.Accept a string as input from the user.

2.Check if the length of the string is greater than 3. If not, do nothing.

3.If the string ends with "ing", append "ly" to the string.

4. If the string does not end with "ing", append "ing" to the string.

5. Display the modified string.

SOURCE CODE:

```
str = input("Enter the string: ")
print("Input string:", str)

if len(str) > 3:
    if str[-3:] == "ing":
        str += "ly"
    else:
        str += "ing"

print("Formatted String:", str)
```

OUTPUT:

Enter the string: play

Input string: play

Formatted String: playing

Enter the string: swimming

Input string: swimming

Formatted String: swimmingly

RESULT:

The program ran successfully and output is verified

Experiment No:27 Lab Cycle:03 Date:10-10-2024

LONGEST WORD

AIM:

Write a Python program to accept a list of words and return length of longest word.

ALGORITHM:

1.Start

2.Input list of words

3.Find length and check if it is largest if true then store in l

4.Print the largest word length

5.Stop

SOURCE CODE:

```
a=[]
n=int(input("enter size of list:"))
print("enter list elements:")
for i in range(n):
  c=input()
  a.append(c)
print("List:",a)
l=0
for i in a:
   ln=len(i)
   if ln>l:
      l=ln
      w=i
print("longest word:" ,w ," and length is",l)
```

OUTPUT:

List:[‘one’,’two’,’three’,’four’]

Longest word: three and length is 5

RESULT:

The program ran successfully and output is verified

Experiment No:28 Lab Cycle:03 Date:10-10-2024

PATTERN

AIM:

Write a Python program to construct the following pattern using nested loops:

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

ALGORITHM:

1. Accept the value n (number of rows) from the user.
2. Use a nested loop to print the first half of the pattern (increasing number of stars).
3. Use another nested loop to print the second half of the pattern (decreasing number of stars).
4. In each iteration, print stars (*) according to the row number.

SOURCE CODE:

```
n = int(input("Enter the number of rows (n): "))
for i in range(1, n + 1):
    for j in range(i):
```

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


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

OUTPUT:

Enter the number of rows (n): 6

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

RESULT:

The program ran successfully and output is verified

Experiment No:29 Lab Cycle:03 Date:10-10-2024

FACTORS

AIM:

Write a Python program to generate all factors of a number.

ALGORITHM:

1. Ask the user to enter a number.
2. Initialize an empty list called factors.
3. Use a loop to iterate from 1 to number.
4. If ‘number' is divisible by current number, add the current number to factors list.
5. After the loop, print the list.

SOURCE CODE:

```
number = int(input("Enter number: "))
factors = []
for i in range(1, number + 1):
 if number % i == 0:
 factors.append(i)
print(f"Factors of {number} are: {factors}")
```

OUTPUT:

```
Enter number: 20
Factors of 20 are: [1, 2, 3, 5, 6, 10, 15, 30]
```

RESULT:

The program ran successfully and output is verified

Experiment No:30 Lab Cycle:03 Date:10-10-2024

AREA USING LAMBDA

AIM:

Write a Python program to find area of square, rectangle and triangle using lambda functions.

ALGORITHM:

1. Start
2. Input the side length of the square, base and height of the triangle, and length and breadth of the rectangle from the user.
3. Calculate the area of the square using a lambda function.
4. Calculate the area of the triangle using a lambda function.
5. Calculate the area of the rectangle using a lambda function.
6. Print the areas of the square, triangle, and rectangle to the user.
7. Stop.

SOURCE CODE:

```
side = int(input("Enter the side length of the square: "))
height=int(input("Enter the height of the triangle: "))
base = int(input("Enter the base of the triangle: "))
length = int(input("Enter the length of the rectangle: "))
breadth = int(input("Enter the breadth of the rectangle: "))
square = lambda x: x**2
rectangle = lambda l, b: l * b
triangle = lambda ht, bs: 0.5 * ht * bs
print("Area of Square = ", square(side))
print("Area of Triangle = ", triangle(height, base))
print("Area of Rectangle = ", rectangle(length, breadth))
```

OUTPUT:

Enter the side length of the square:4

Enter the height of the triangle: 10

Enter the base of the triangle: 5

Enter the length of the rectangle: 12

Enter the breadth of the rectangle: 4

Area of Square = 16

Area of Triangle = 25.0

Area of Rectangle = 48

RESULT:

The program ran successfully and output is verified

Experiment No:31 Lab Cycle:03 Date:10-10-2024

BUILT IN PACKAGES

AIM:

Write a Python program to demonstrate the usage of built-in Python packages..

ALGORITHM:

1. Import the necessary built-in Python packages (like math, random, datetime, etc.).
2. Use functions from each package to perform basic operations.
3. Display the results to show how the built-in packages work.

SOURCE CODE:

```
number = 16
sqrt_value = math.sqrt(number)
print(f"Square root of {number} is: {sqrt_value}")

random_number = random.randint(1, 100)
print(f"A random number between 1 and 100 is: {random_number}")

current_datetime = datetime.datetime.now()
print(f"Current date and time: {current_datetime}"
```

OUTPUT:

```
Square root of 16 is: 4.0
A random number between 1 and 100 is: 57
Current date and time: 2025-01-05 10:45:12.123456
```

RESULT:

The program ran successfully and output is verified

Experiment No:32 Lab Cycle:03 Date:10-10-2024

PACKAGES AND SUB PACKAGES

AIM:

Write a program to 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)

ALGORITHM:

1.Start

2.Create a package graphics and inside this create init.py, rectangle.py, circle.py

3.Create a sub-package TD graphics and inside this create init.py, cuboid.py, sphere.py

4.In Main.py import rectangle and circle using import statement.

5. Also use import * to import methods of cuboid.py, sphere.py from TD graphics

6.Display the area and perimeter of rectangle, circle, cuboid and sphere

7.Stop

SOURCE CODE:

Main.py:

```
from graphics import rectangle,circle
from graphics.TDgraphics.cuboid import *
from graphics.TDgraphics.sphere import *

print("Area of rectangle:",rectangle.rarea(2,3))
print("Perimeter of rectangle:",rectangle.rperimeter(2,3))

print("Area of circle:",circle.carea(2))
print("Perimeter of circle:",circle.cperimeter(2))
```

```
print("Area of cuboid:",qarea(1,2,3))
print("Perimeter of cuboid:",qperimeter(1,2,3))
print("Area of sphere:",sarea(2))
print("Perimeter of sphere:",sperimeter(2))
```

graphics(package)

rectangle.py:

```
def rarea(l,b):
    return l*b
def rperimeter(l,b):
    return 2*(l+b)
```

circle.py:

```
from math import pi
def carea(r):
    return pi*r*r
def cperimeter(r):
    return 2*pi*r
```

init.py

graphics/TDgraphics(package)

cuboid.py:

```
def qarea(l,b,h):
    return 2*(l*b+b*h+l*h)
def qperimeter(l,b,h):
    return 4*(l+b+h)
```

sphere.py:

```
from math import pi
def sarea(r):
    return 4*pi*r*r
def sperimeter(r):
    return 2*pi*r
```

init.py

OUTPUT:

Area of rectangle: 6

Perimeter of rectangle: 10

Area of circle: 12.566370614359172

Perimeter of circle: 12.566370614359172

Area of cuboid: 22

Perimeter of cuboid: 24

Area of sphere: 50.26548245743669

Perimeter of sphere: 12.566370614359172

RESULT:

The program ran successfully and output is verified

Experiment No:33 Lab Cycle:04 Date:10-10-2024

COMPARE TWO RECTANGLES

AIM:

Write a Program to create a Rectangle class with attributes length and breadth and methods to find area and perimeter. Also, compare two Rectangle objects by their area.

ALGORITHM:

1. Define a Rectangle class.
2. Add attributes length and breadth to represent the dimensions of the rectangle.
3. Define the method area() to calculate and return the area of the rectangle (length * breadth).
4. Define the method perimeter() to calculate and return the perimeter of the rectangle (2 * (length + breadth)).
5. Create two Rectangle objects by taking user inputs for length and breadth.
6. Compare the areas of the two Rectangle objects and print which one has a larger area, or if they are equal.

SOURCE 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)
length1 = int(input("Enter length of first rectangle: "))
breadth1 = int(input("Enter breadth of first rectangle: "))
```

```
rect1 = Rectangle(length1, breadth1)
length2 = int(input("Enter length of second rectangle: "))
breadth2 = int(input("Enter breadth of second rectangle: "))
rect2 = Rectangle(length2, breadth2)
area1 = rect1.area()
area2 = rect2.area()
print(f"Area of first rectangle: {area1}")
print(f"Area of second rectangle: {area2}")
if area1 > area2:
   print("The first rectangle has a larger area.")
elif area1 < area2:
   print("The second rectangle has a larger area.")
else:
            print("Both rectangles have equal areas.")
```

OUTPUT:

```
Enter length of first rectangle: 5
Enter breadth of first rectangle: 4
Enter length of second rectangle: 6
Enter breadth of second rectangle: 3

Area of first rectangle: 20
Area of second rectangle: 18
The first rectangle has a larger area
```

RESULT:

The program ran successfully and output is verified

Experiment No:34 Lab Cycle:04 Date:10-10-2024

BANK ACCOUNT DETAILS

AIM:

Write a Python program to create a Bank Account class with attributes such as account number, account holder's name, account type, and balance, and implement methods to deposit, withdraw, and display account details.

ALGORITHM:

1.Define a class Bank with a constructor init to initialize account details: account number, name, type, and balance.

2.Define a method display to print the account details.

3.Define a method deposit to add funds to the account balance.

4.Define a method withdraw to subtract funds from the account balance, ensuring sufficient balance exists.

5.Create an object of the Bank class and use the methods to deposit, withdraw, and display account information.

SOURCE CODE:

```
class Bank:
    def init(self, no, name, type, bal):
        self.acc_no = no
        self.acc_name = name
        self.acc_type = type
        self.acc_bal = bal

    def display(self):
        print("Account number:", self.acc_no)
        print("Account holder name:", self.acc_name)
```

```
        print("Account type:", self.acc_type)
        print("Account balance is:", self.acc_bal, "\n")

    def deposit(self, val):
        if val > 0:
            print("Current balance is", self.acc_bal)
            self.acc_bal += val
            print(val, "deposited and current balance is:", self.acc_bal, "\n")
        else:
            print("Enter a valid amount \n")

    def withdraw(self, val):
        if val > 0:
            if self.acc_bal >= val:
                print("Current balance is", self.acc_bal)
                self.acc_bal -= val
                print(val, "withdrawn and current balance is:", self.acc_bal, "\n")
            else:
                print("Insufficient balance \n")
        else:
            print("Enter a valid amount \n")


ob1 = Bank(111, "ABCD", "Savings", 100000)
ob1.display()
ob1.withdraw(20000000)
ob1.deposit(50000)
ob1.display()
```

OUTPUT:

```
Account number: 111
Account holder name: ABCD
```

Account type: Savings

Account balance is: 100000

Insufficient balance

Current balance is 100000

50000 deposited and current balance is: 150000

Account number: 111

Account holder name: ABCD

Account type: Savings

Account balance is: 150000

RESULT:

The program ran successfully and output is verified

Experiment No:35 Lab Cycle:04 Date:10-10-2024

OPERATOR OVERLOADING

AIM:

Write a Python program to create a class Rectangle with private attributes length and width.Overload '<' operator to compare the area of 2 rectangles

ALGORITHM:

1. Create a class named 'Rectangle
2. Initialize private attributes '__length','__width' and '__area"
3. Calculate and set the area during initialization.
4. Update the '__area' attribute
5. Compare the areas of two rectangle.
6. Take user input for length and width of two rectangles
7. Call 'cale area' method
8. Use the '<' operator to compare the areas of two rectangles.
9. Print which input is larger

SOURCE CODE:

```
class rectangle:
    __area=0
    __perimeter=0
    def__init__(self,length,width):
        self.__length=length
        self.__width=width
    def calc_area(self):
        self.__area=self.__length*self.__width
        print("Area is:" self.__area)
```

```
    def __lt__(self,second):
        if self.__area<second.__area:
            return True
        else:
            return False
length1= int(input("Enter length of the rectangle 1: "))
width1=int(input("Enter width of the rectangle 1: "))
length2=int(input("Enter length of the rectangle 2: "))
width2=int(input("Enter width of the rectangle 2: "))

obj1=rectangle(length 1, width1)
obj2=rectangle(length2, width2)
objl.calc_area()
obj2.calc_area()

if objl < obj2:
        print("Rectangle two is large")
else:
        print("Rectangle one is large or these are equal")
```

OUTPUT:

```
Enter length of rectangle1:6
Enter length of rectangle1:4
Enter length of rectangle2:5
Enter length of rectangle2:3

Area of 1 is 24
Area of 2 is 15
Rectangle one is larger
```

RESULT:

The program ran successfully and output is verified

Experiment No:36 Lab Cycle:04 Date:10-10-2024

SUM OF TIMES

AIM:

Write a Python program to create a class Time with private attributes hour, minute and second. Overload '+' operator to find sum of 2 time.

ALGORITHM:

1. Create a class Time with attributes hour, minute and second.
2. Define a function add with attribute other which calculates totalhour, totalminute, totalsecond and returns Time (totalhour, totalminute, totalsecond).
3. Define a function display which displays the time.
4. Create two objects t1 and t2 with values.
5. Set t3=t1+t2
6. Display t1, t2 and t3.

SOURCE CODE:

Class Time:

```
def __init__(self, hour=0, minute=0, second=0):
    self.__hour=hour
    self.__minute=minute
    self.__second=second
def __add__(self, other):
    totalhour=self.__hour + other.__hour
    totalminute=self.__minute + other.__minute
    totalsecond=self.__second + other.__second
    totalminute+=totalsecond//60
```

```
                totalsecond%=60
                totalhour+=totalminute//60
                totalminute%=60
                return Time(totalhour, totalminute, totalsecond)
        def display(self):
                return str(self.__hour) + ":" + str(self.__minute) + ":" +str(self.__second)
t1=Time(2,5,50)
t2=Time(1,20,30)
t3=t1 + t2
print("Time 1:", t1.display())
print("Time 2:", t2.display())
print("Sum of time:", t3.display())
```

OUTPUT:

Time 1: 2:5:50

Time 2: 1:20:30

Sum of time: 3:26:20

RESULT:

The program ran successfully and output is verified

Experiment No:37 Lab Cycle:04 Date:10-10-2024

METHOD OVERLOADING IN CLASS

AIM:

Write a Python program to 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

ALGORITHM:

1. Start

2. Define a super class Publisher with constructor to assign publisher name and a method display() to return the name.

3. Define derived class Book inherit from Publisher with attributes title and author, constructors to assign the values and a method display() to return the values

4. Define derived class Python inherit from Book with attributes price and no_of_pages, constructors to assign the values and a method display() to return the values.

5: Create an object for Python and invoke the method display() to return the result.

6: Stop

SOURCE CODE:

```
class Publisher:
    def __init__(self,name):
        self.name=name
    def display(self):
    print(f ” Publisher: {self.name}”)
class Book(Publisher):
    def __init__(self,name,title,author):
        super().__init__(name)
```

```
        self.title=title
        self.author=author
    def display(self):
        super().display()
        print(f "Title: {self.title}")
        print(f "Author: {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 display(self):
        super().display()
        print(f "Price: {self.price}")
        print(f "No of pages: {self.no_of_pages}")
publisher = Python("O' Reilly Media", "Think Python", "Allen B Downey",750,1250)
publisher.display()
```

OUTPUT:

Publisher : O' Reilly Media

Title : Think Python

Author : Allen B Downey

Price : ₹750

No of pages : 1250

RESULT:

The program ran successfully and output is verified

Experiment No:38 Lab Cycle:05 Date:10-10-2024

READ FILE LINE BY LINE

AIM:

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

ALGORITHM:

1.Open the file

2.Initialize an empty list

3.Read each line

4.Strip whitespace

5.Append to list

6.Print the list

SOURCE CODE:

```
number = int(input("Enter number: "))
factors = []
for i in range(1, number + 1):
 if number % i == 0:
 factors.append(i)
print(f"Factors of {number} are: {factors}")
```

OUTPUT:

```
files=open('file1.txt','r')
list1=[]
for x in files:
    list1.append(x.strip())
print(list1)
```

RESULT:

The program ran successfully and output is verified

Experiment No:39 Lab Cycle:05 Date:10-10-2024

COPY ODD LINES

AIM:

Write a Python program to copy odd lines of one file to other.

ALGORITHM:

1.Open the file1.txt for reading

2.Write every other line to file2.txt

3.Read and clean data from file2.txt

4.Print the cleaned lines

SOURCE CODE:

```
with open("file1.txt","r") as file:
   data=file.readlines()
with open("file2.txt","w") as outfile:
   for i in range(0, len(data),2):
       outfile.write(data[i])
with open("file2.txt","r") as outfile:
   out=[line.strip() for line in outfile.readlines()]
   print(out)
```

OUTPUT:

['1.apple', '3.blueberry', '5.grapes']

RESULT:

The program ran successfully and output is verified

Experiment No:40 Lab Cycle:05 Date:10-10-2024

READ EACH ROW FROM CSV FILE

AIM:

Write a Python to read each row from a given CSV file and print a list of strings..

ALGORITHM:

1.Import the csv module to handle CSV files.

2.Open the given CSV file in read mode.

3.Use csv.reader() to read the file row by row.

4.Loop through each row of data and print it as a list of strings.

SOURCE CODE:

```
import csv
filename = "data.csv"
with open(filename, "r") as file:
    data = csv.reader(file)
    for i in data:
    print(i)
```

OUTPUT:

data.csv

```
Name, Age, Country
John, 25, USA
Jane, 30, Canada
['Name', ' Age', ' Country']
['John', ' 25', ' USA']
['Jane', ' 30', ' Canada']
```

RESULT:

The program ran successfully and output is verified

Experiment No:41 Lab Cycle:05 Date:10-10-2024

DISPLAY CONTENT OF COLUMN FROM CSV FILE

AIM:

Write a Python program to read a specific column (e.g., "Username") from a given CSV file and print its content.

ALGORITHM:

1.Import the csv module to handle CSV files.

2.Open the given CSV file in read mode. .

3.Use csv.DictReader() to read the file, which allows accessing columns by name.

4.Loop through each row of data and print the content of the specified column (e.g., 'Username').

SOURCE CODE:

```
import csv
filename = "data.csv"
with open(filename, "r") as file:
  data = csv.DictReader(file)
    for i in data:
      print(i['Username'])
```

OUTPUT:

data.csv

Username, Email, Age

john_doe, john@example.com, 28

jane_doe, jane@example.com, 32

john_doe

jane_doe

RESULT:

The program ran successfully and output is verified

Experiment No:42 Lab Cycle:05 Date:10-10-2024

DICTIONARY TO A CSV FILE

AIM:

Write a Python program to write a Python dictionary to a CSV file, then read and display the content of the CSV file.

ALGORITHM:

1. Define a list of dictionaries that contains data.
2. Open a CSV file in write mode.
3. Write the dictionary keys (column names) as the header in the CSV file.
4. Loop through the list of dictionaries and write the values to the CSV file.
5. Open the same CSV file in read mode.
6. Use csv.reader() to read the content of the file.
7. Display the content row by row.

SOURCE CODE:

```
import csv

data = [{'name': 'murali', 'role': 'student', 'college': 'cec'},
    {'name': 'Ramu', 'role': 'teacher', 'college': 'cec'},
    {'name': 'raju', 'role': 'lab', 'college': 'cec'}]

with open("dict.csv", 'w') as file:
        file.write(','.join(data[0].keys()))
        file.write('\n')
        for i in data:
        file.write(','.join(str(x) for x in i.values()))
        file.write('\n')
```

```
with open("dict.csv", 'r') as f:
    d = csv.reader(f)
    for i in d:
 print(i)
```

OUTPUT:

dict.csv

name,role,college

Keshu,student,cec

Ramu,teacher,cec

raju,lab tech,cec

['name', 'role', 'college']

['Keshu', 'student', 'cec']

['Ramu', 'teacher', 'cec']

['raju', 'lab tech', 'cec']

RESULT:

The program ran successfully and output is verified