

LABCYCLE 3

EXPERIMENT NO:31

BUILT IN PACKAGES

Date:09/01/2023

AIM: Work with Built-in packages.

ALGORITHM:

Step 1: Start.

Step 2: Using import statement import a built-in package math to perform some operations.

Step 3: Read values for radius.

Step 4: Define two functions to return area and perimeter of a circle.

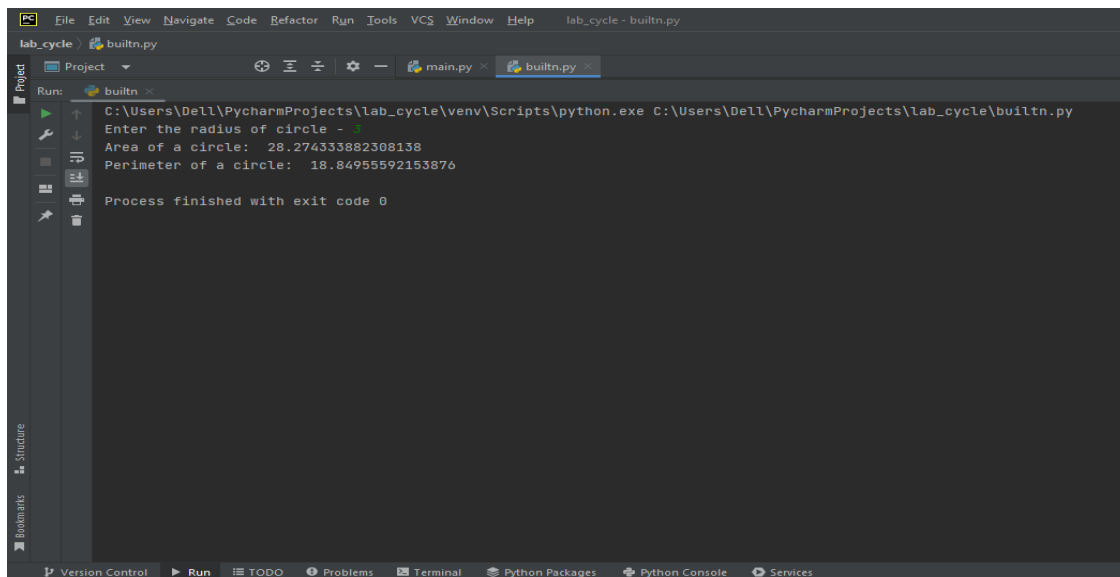
Step 5: Call the methods and print result.

Step 6: Stop.

PROGRAM CODE:

```
import math
def area(r):
    return math.pi*r*r
def perimeter(r):
    return 2*math.pi*r
r=int(input("Enter the radius of circle - "))
print("Area of a circle: ", area(r))
print("Perimeter of a circle: ", perimeter(r))
```

OUTPUT:



```
C:\Users\De11\PycharmProjects\lab_cycle\venv\Scripts\python.exe C:\Users\De11\PycharmProjects\lab_cycle\builtn.py
Enter the radius of circle - 1
Area of a circle:  28.274333882308138
Perimeter of a circle:  18.84955592153876
Process finished with exit code 0
```

RESULT:

Program to implement built-in packages has been executed successfully and output is verified.

LABCYCLE 3

EXPERIMENT NO:32

PACKAGES AND SUB PACKAGES

Date: 09/01/2023

AIM: Create a package with modules rectangle, circle and sub-package 3D-graphice with modules cuboid and sphere. Include methods to find the 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:

Step 1: Start.

Step 2: Create a package(graphics) with two modules (rectangle and circle). Define methods to find area and perimeter rectangle and circle.

Step 3: Create a sub-package (graphics3d) with two modules (sphere and cuboid). Define methods to perform the calculations.

Step 4: In a python file using import statement import the above modules,read the values for dimensions.

Step 5: Call the methods and print result.

Step 6: Stop.

PROGRAM CODE:

result.py

```
import graphics.circle

import graphics.rectangle

import graphics.graphics3d.cuboid

import graphics.graphics3d.sphere

r=int(input("Enter the radius of circle - "))

r1=int(input("Enter the radius of sphere - "))

leng=int(input("Enter the length of rectangle - "))
```

```
bread=int(input("Enter the breadth of rectangle - "))
length=int(input("Enter the length of cuboid - "))
height=int(input("Enter the height of cuboid - "))
width=int(input("Enter the width of cuboid - "))
print("Area of circle is:",graphics.circle.area(r))
print("Perimeter if circle is:",graphics.circle.perimeter(r))
print("Area of Rectangle is:",graphics.rectangle.area(leng,bread))
print("Perimeter of Rectangle is:",graphics.rectangle.perimeter(leng,bread))
print("Area of Cuboid is:",graphics.graphics3d.cuboid.area(length,width,height))
print("Perimeter of Cuboid is:",graphics.graphics3d.cuboid.perimeter(length,width,height))
print("Area of Sphere is :",graphics.graphics3d.sphere.area(r1))
print("Perimeter of sphere is:",graphics.graphics3d.sphere.perimeter(r1))
```

graphics package

rectangle.py

```
def area(leng,bread):
    return (leng*bread)

def perimeter(leng,bread):
    return 2*(leng+bread)
```

circle.py

```
import math

def area(r):
    return math.pi*r*r

def perimeter(r):
    return 2*math.pi*r
```

graphics3d(package inside graphics)

sphere.py

```
import math
```

```
def area(r):
```

```
    return 4*math.pi*r
```

```
def perimeter(r):
```

```
    return 2*math.pi*r
```

cuboid.py

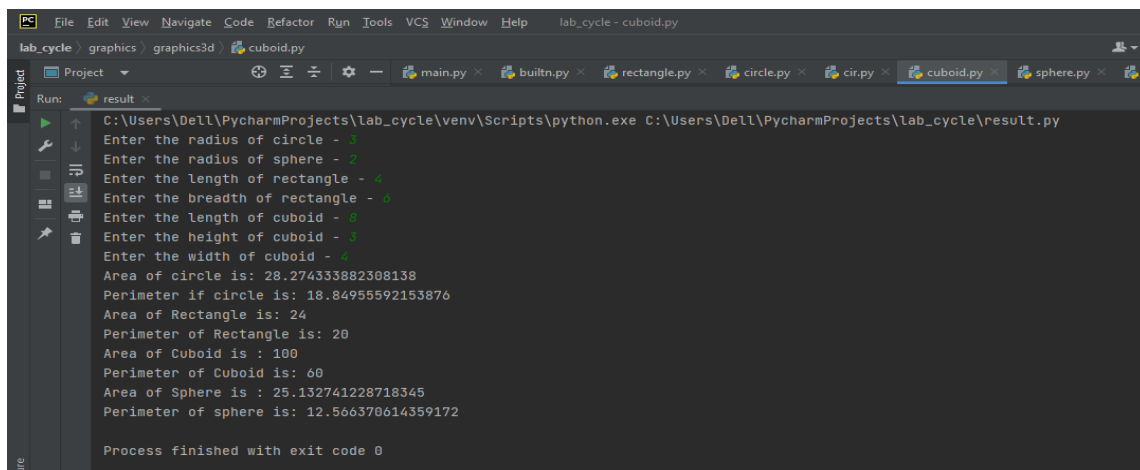
```
def area(length,width,height):
```

```
    return 2*(length*width)+(length*height)+(width*height)
```

```
def perimeter(length,width,height):
```

```
    return 4*(length+width+height)
```

OUTPUT:



```
C:\Users\Dell\PycharmProjects\lab_cycle\venv\Scripts\python.exe C:\Users\Dell\PycharmProjects\lab_cycle\result.py
Enter the radius of circle - 3
Enter the radius of sphere - 3
Enter the length of rectangle - 4
Enter the breadth of rectangle - 4
Enter the length of cuboid - 4
Enter the height of cuboid - 3
Enter the width of cuboid - 4
Area of circle is: 28.274333882308138
Perimeter of circle is: 18.84955592153876
Area of Rectangle is: 24
Perimeter of Rectangle is: 20
Area of Cuboid is : 100
Perimeter of Cuboid is: 60
Area of Sphere is : 25.132741228718345
Perimeter of sphere is: 12.566370614359172

Process finished with exit code 0
```

RESULT:

Program to create package and sub-package has been executed successfully and output is verified.

LABCYCLE 4

EXPERIMENT NO:33

COMPARE TWO RECTANGLES

Date:09/01/2023

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

ALGORITHM:

Step 1: Start.

Step 2: Define a class.

Step 3: Define method to calculate the area.

Step 4: Read the length and breadth for two rectangles.

Step 5: Create two object to invoke the class members and print the area.

Step 6: Check if obj1>obj2 then rectangle one has greater area.

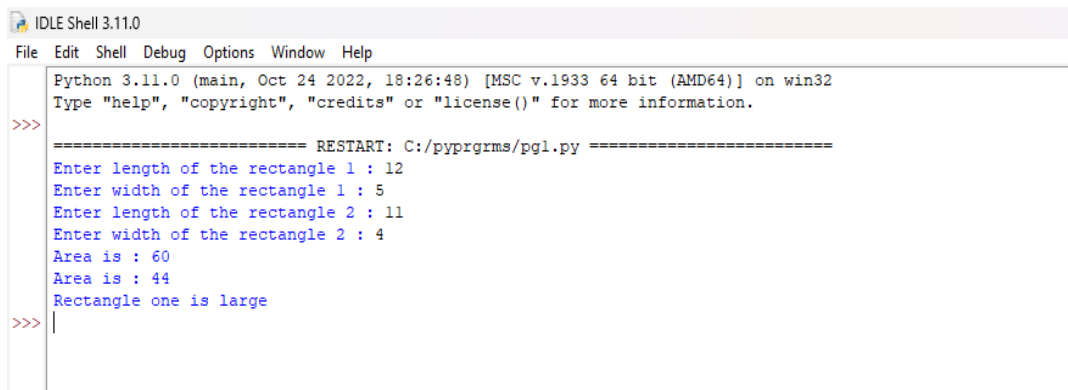
Step 7: Stop.

PROGRAM 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(length1, width1)
obj2 = rectangle(length2, width2)
obj1.calc_area()
obj2.calc_area()
if obj1 < obj2:
    print("Rectangle two is large")
else:
    print("Rectangle one is large ")
```

OUTPUT:



```
IDLE Shell 3.11.0
File Edit Shell Debug Options Window Help
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/pyprgrms/pgl.py =====
Enter length of the rectangle 1 : 12
Enter width of the rectangle 1 : 5
Enter length of the rectangle 2 : 11
Enter width of the rectangle 2 : 4
Area is : 60
Area is : 44
Rectangle one is large
>>> |
```

RESULT:

Program to compare two rectangles has been executed successfully and output is verified.

LABCYCLE 4

EXPERIMENT NO:34

BANK ACCOUNT DETAILS

Date:16/01/2023

AIM: 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.

ALGORITHM:

Step 1: Start.

Step 2: Define a class.

Step 3: Declare the member of class

Step 4: Read the members. Define methods to calculate balance amount.

Step 5: Create an object to invoke the class members and print the account details.

Step 6: Check if withdraw amount > deposit amount then print "INSUFFICIENT BALANCE"

Step 7: Stop.

PROGRAM CODE:

```
class bank:
    def __init__(self, accno, name, accty, bal):
        self.accno = accno
        self.name = name
        self.accty = accty
        self.bal = 0
    def showamt(self):
        print("Account Holder Name:", self.name)
        print("Account Number:", self.accno)
        print("Account Type:", self.accty)
        print("Your Balance Amount:", self.bal)
    def depo(self, d1):
        self.bal = self.bal + d1
```



```
        return self.bal
    def withd(self, w1):
        self.bal = self.bal - w1
        return self.bal
n = input("Enter your name: ")
a = int(input("Enter your Account number: "))
t = input("Enter your account type: ")
o = bank(a, n, t, bal=0)
o.showamt()
while (True):
    print("\nMenu")
    print("\n1.Deposit")
    print("\n2.Withdraw")
    c = int(input("Enter your choice:"))
    if (c == 1):
        d = int(input("Enter the amount to deposit"))
        print("Your total balance is :", o.depo(d))
    elif(c==2):
        w = int(input("Enter the amount to be withdrawn:"))
        if (w > d):
            print("INSUFFICIENT BALANCE")
        else:
            print("Your total balance Amount is", o.withd(w))
    else:
        print("Enter a valid choice.")
```

OUTPUT:

```

Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/pypxgrms/pg2.py =====
Enter your name: Becky
Enter your Account number: 67852443309
Enter your account type: 0
Account Holder Name: Becky
Account Number: 67852443309
Account Type: 0
Your Balance Amount: 0
|
Menu
1.Deposit
2.Withdraw
Enter your choice:1
Enter the amount to deposit:500
Your total balance is : 500
Menu
1.Deposit
2.Withdraw
Enter your choice:1
Enter the amount to deposit:200
Your total balance is : 700
Menu
1.Deposit
2.Withdraw
Enter your choice:2
Enter the amount to be withdrawn:200
Your total balance Amount is 500
Menu

```

```

1.Deposit
2.Withdraw
Enter your choice:2
Enter the amount to be withdrawn:500
INSUFFICIENT BALANCE
Menu
1.Deposit
2.Withdraw
Enter your choice:

```

RESULT:

Program to display bank account details has been executed successfully and output is verified.

LABCYCLE 4

EXPERIMENT NO:35

OPERATOR OVERLOADING

Date: 09/01/2023

AIM: Create a class Rectangle with private attributes length and width. Overload '<' operator to compare the area of 2 rectangles.

ALGORITHM:

Step 1: Start.

Step 2: Define a class.

Step 3: Define method to calculate the area.

Step 4: Read the length and breadth for two rectangles.

Step 5: Create two object to invoke the class members.

Step 6: Call the object r1 in r2 and compare.

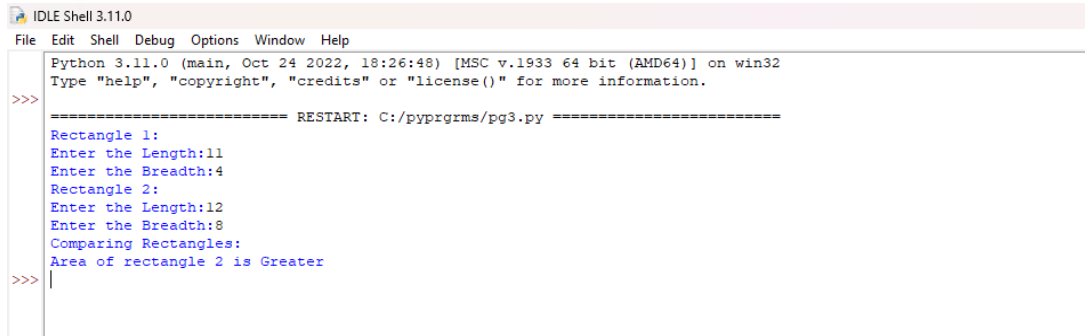
Step 7: Stop.

PROGRAM CODE:

```
class rectangle:
    def __init__(self):
        self._length = int(input("Enter the Length:"))
        self._breadth = int(input("Enter the Breadth:"))
        self.area = self._length * self._breadth
    def greaterThan(self, rectangle):
        if self.area < rectangle.area:
            print("Area of rectangle 1 is Greater")
        else:
            print("Area of rectangle 2 is Greater")
print("Rectangle 1:")
r1 = rectangle()
print("Rectangle 2:")
r2 = rectangle()
print("Comparing Rectangles:")
```

r2.greaterThan(r1)

OUTPUT:



```
IDLE Shell 3.11.0
File Edit Shell Debug Options Window Help
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/pyprgrms/pg3.py =====
Rectangle 1:
Enter the Length:11
Enter the Breadth:4
Rectangle 2:
Enter the Length:12
Enter the Breadth:8
Comparing Rectangles:
Area of rectangle 2 is Greater
>>> |
```

RESULT:

Program has been executed successfully and output is verified.

LABCYCLE 4

EXPERIMENT NO:36

OPERATOR OVERLOADING (SUM OF TIME)

Date: 16/01/2023

AIM: Create a class Time with private attributes hour, minute and second. Overload '+' operator to find the sum of 2 time.

ALGORITHM:

Step 1: Start.

Step 2: Define a class.

Step 3: Initialize members (hour,minute,second) using constructor.

Step 4: Define method to calculate sum of time.

Step 5: Create two object to invoke the class members.

Step 6: Print the sum of time.

Step 7: Stop.

PROGRAM CODE:

```
class Time:
```

```
    def __init__(self, hour, minute, second):
```

```
        self.hour = hour
```

```
        self.minute = minute
```

```
        self.second = second
```

```
    def __add__(self, other):
```

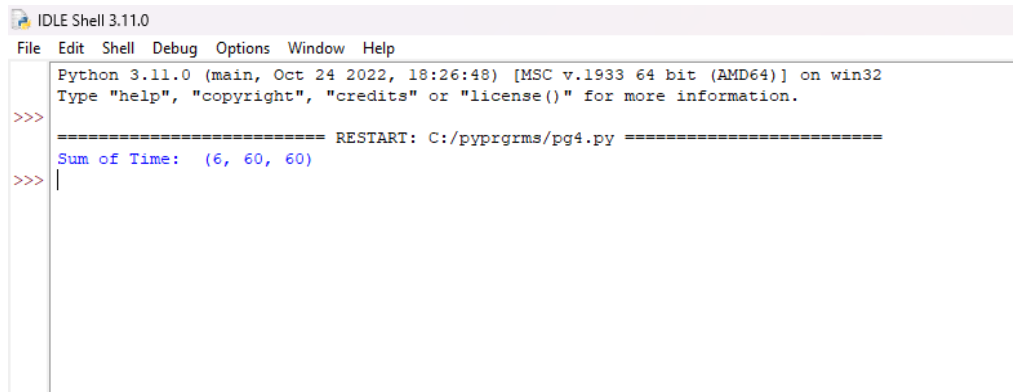
```
        return self.hour + other.hour, self.minute + other.minute, self.second + other.second
```

```
o1 = Time(2, 40, 10)
```

```
o2 = Time(4, 20, 50)
```

```
print("Sum of Time: ",o1 + o2)
```

OUTPUT:

A screenshot of the IDLE Shell 3.11.0 window. The title bar reads 'IDLE Shell 3.11.0'. The menu bar includes 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The shell area shows the following text: 'Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32', 'Type "help", "copyright", "credits" or "license()" for more information.', a prompt '>>>', a separator line '===== RESTART: C:/pyprgrms/pg4.py =====', and the output 'Sum of Time: (6, 60, 60)'. A second prompt '>>>' is visible on the next line with a cursor character '|'.

```
IDLE Shell 3.11.0
File Edit Shell Debug Options Window Help
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/pyprgrms/pg4.py =====
Sum of Time: (6, 60, 60)
>>> |
```

RESULT:

Program to print sum of two times has been executed successfully and output is verified.

LABCYCLE 4

EXPERIMENT NO:37

METHOD OVERRIDING IN CLASS

Date: 16/01/2023

AIM: Create a class Publisher(name). Derive class Book from Publisher with attributes title and author. Derive class Python from Book from Publisher with attributes price and no_of_pages. Write a program that displays information about a Python book. Use base class constructor invocation and method overloading.

ALGORITHM:

Step 1: Start.

Step 2: Define a super class Publisher with constructors and a method display to return the values.

Step 3: Define derived class Book inherit from Publisher with constructors and a method display() display to return the values.

Step 4: Define derived class Python inherit from Book with constructors and a method display() display to return the values.

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

Step 6: Stop.

PROGRAM CODE:

```
class Publisher:
    def __init__(self, pbname):
        self.pbname = pbname
    def display(self):
        return (self.pbname)
class Book(Publisher):
    def __init__(self, title, author, pbname):
        self.title = title
        self.author = author
        Publisher.__init__(self, pbname)
    def display(self):
```

```
        return (self.pbname, self.title, self.author)
class Python(Book):
    def __init__(self, price, pages, title, author, pbname):
        self.price = price
        self.pages = pages
        Book.__init__(self, title, author, pbname)

    def display(self):
        print("Publisher :", self.pbname)
        print("Title :", self.title)
        print("Author :", self.author)
        print("Price :", self.price)
        print("Pages :", self.pages)

pbname=input("Enter the publisher name - ")
title=input("Enter the title name - ")
author=input("Enter the author name - ")
price=int(input("Enter the price - "))
pages=int(input("Enter the number of pages - "))
print("Book Details:")
obj=Python(price, pages, title, author, pbname)
#obj = Python(179, 456, "Wings of Fire ", "A P J Abdul Kalam", "DC Books")
obj.display()
```


OUTPUT:

A screenshot of the IDLE Shell 3.11.0 interface. The window title is 'IDLE Shell 3.11.0'. The menu bar includes 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The shell area shows the following text: 'Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32', 'Type "help", "copyright", "credits" or "license()" for more information.', a prompt '>>>', a separator line '===== RESTART: C:\pyprgrms\pg5.py =====', and the output 'Book Details:', 'Publisher : DC Books', 'Title : Wings of Fire', 'Author : A P J Abdul Kalam', 'Price : 179', and 'Pages : 456'. The prompt '>>>' is followed by a vertical bar '|'.

```
IDLE Shell 3.11.0
File Edit Shell Debug Options Window Help
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\pyprgrms\pg5.py =====
Book Details:
Publisher : DC Books
Title : Wings of Fire
Author : A P J Abdul Kalam
Price : 179
Pages : 456
>>> |
```

RESULT:

Program to implement method overriding has been executed successfully and output is verified.

LABCYCLE 5

EXPERIMENT NO:38

READ FILE LINE BY LINE

Date: 16/01/2023

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

ALGORITHM:

Step 1: Start.

Step 2: Using with statement open the file 'mytxt.txt'.

Step 3: Print the original file.

Step 4: Read the file and remove the new line.

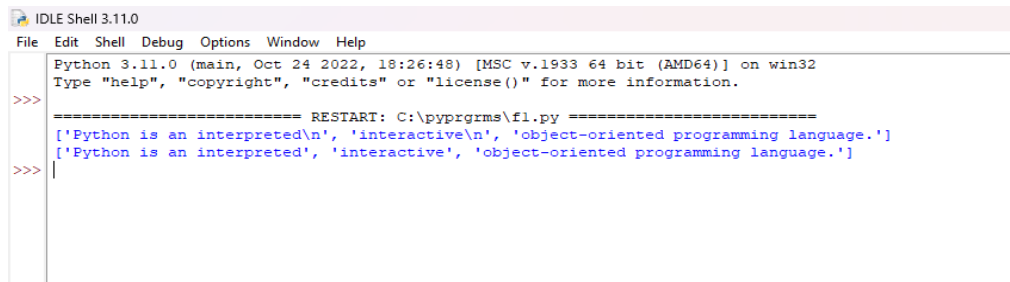
Step 5: Print the list of file content.

Step 6: Stop.

PROGRAM CODE:

```
with open('mytxt.txt') as f:
    lines = [line for line in f]
print(lines)
# removing the new line characters
with open('mytxt.txt') as f:
    lines = [line.rstrip() for line in f]
print(lines)
```

OUTPUT:

A screenshot of the IDLE Shell 3.11.0 window. The title bar reads "IDLE Shell 3.11.0". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The main text area shows the following output:

```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>>
===== RESTART: C:\pyprgrms\fl.py =====
['Python is an interpreted\n', 'interactive\n', 'object-oriented programming language.']
['Python is an interpreted', 'interactive', 'object-oriented programming language.']
>>>
```

RESULT:

Program to copy file has been executed successfully and output is verified.

LABCYCLE 5

EXPERIMENT NO:39

COPY ODD LINES

Date: 16/01/2023

AIM: Python program to copy odd lines of one file to other.

ALGORITHM:

Step 1: Start.

Step 2: Assign the required file name to a variable.

Step 3: Open the file in read mode and assign to a variable.

Step 4: Read the file and remove the new line.

Step 5: Assign the required file name to which the result should be displayed into a variable.

Step 6: Open the file in write mode and assign to a variable.

Step 7: Traverse in each line of the read text file.

Step 8: Check the odd lines of file. Then write those lines into write text file.

Step 9: Print the result.

Step 10: Close both file.

Step 11: Stop.

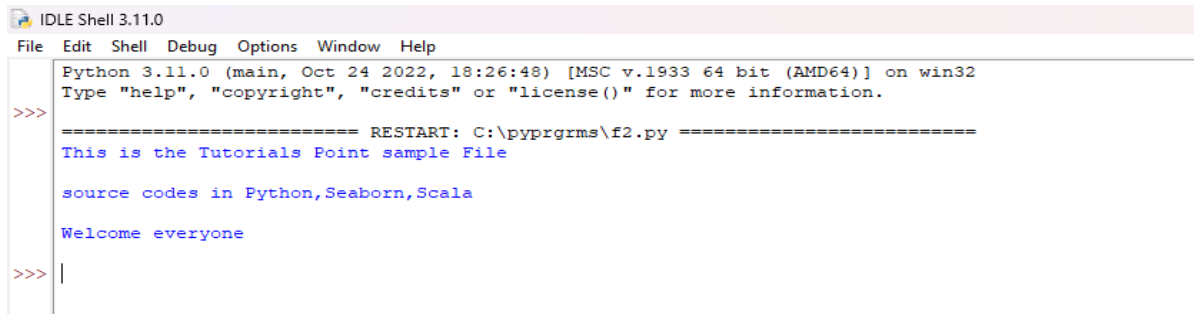
PROGRAM CODE:

```
inputFile = "examplefile.txt"
readFile = open(inputFile, "r")
# output text file path
outputFile = "printodddline.txt"
writeFile = open(outputFile, "w")
ReadFileLines = readFile.readlines()
# Traverse in each line of the read text file
for x in range(0, len(ReadFileLines)):
    if(x % 2 != 0):
        writeFile.write(ReadFileLines[x])
        # printing the odd line
        print(ReadFileLines[x])
```

```
writeFile.close()
```

```
readFile.close()
```

OUTPUT:



```
IDLE Shell 3.11.0
File Edit Shell Debug Options Window Help
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\pyprgrms\f2.py =====
This is the Tutorials Point sample File
source codes in Python,Seaborn,Scala
Welcome everyone
>>> |
```

RESULT:

Program to print odd lines of files has been executed successfully and output is verified.

LABCYCLE 5

EXPERIMENT NO:40

READ EACH ROW FROM CSV FILE

Date: 16/01/2023

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

ALGORITHM:

Step 1: Start.

Step 2: Using import statement import the csv module.

Step 3: Using with statement open the required file as csv.

Step 4: Read the file and remove the new line.

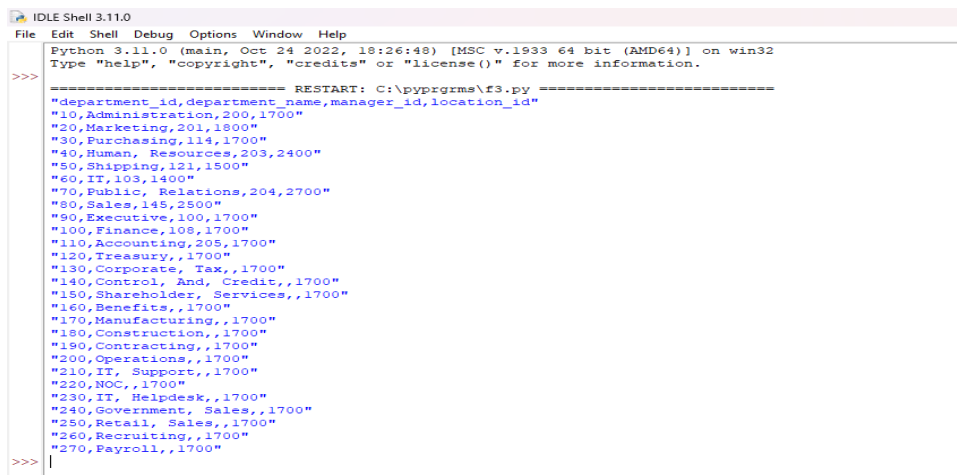
Step 5: Print the list of file content.

Step 6: Stop.

PROGRAM CODE:

```
import csv
with open('departments.csv', newline='') as csvfile:
    data = csv.reader(csvfile, delimiter=' ', quotechar='|')
    for row in data:
        print(' '.join(row))
```

OUTPUT:



```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
----- RESTART: C:\pyprgrms\fs.py -----
"department_id,department_name,manager_id,location_id"
"10,Administration,200,1700"
"20,Marketing,201,1800"
"30,Purchasing,114,1700"
"40,Human, Resources,203,2400"
"50,Shipping,121,1500"
"60,IT,103,1400"
"70,Public, Relations,204,2700"
"80,Sales,145,2500"
"90,Executive,100,1700"
"100,Finance,108,1700"
"110,Accounting,205,1700"
"120,Treasury,,1700"
"130,Corporate, Tax,,1700"
"140,Control, And, Credit,,1700"
"150,Shareholder, Services,,1700"
"160,Benefits,,1700"
"170,Manufacturing,,1700"
"180,Construction,,1700"
"190,Contracting,,1700"
"200,Operations,,1700"
"210,IT, Support,,1700"
"220,NOC,,1700"
"230,IT, Helpdesk,,1700"
"240,Government, Sales,1700"
"250,Retail, Sales,,1700"
"260,Recruiting,,1700"
"270,Payroll,,1700"
>>> |
```

RESULT:

Program read each row from csv file has been executed successfully and output is verified.

LABCYCLE 5

EXPERIMENT NO:41

DISPLAY CONTENT OF COLUMN FROM CSV FILE

Date: 16/01/2023

AIM: Write a Python program to read specific columns of a given CSV file and print the content of the columns.

ALGORITHM:

Step 1: Start.

Step 2: Using import statement import the csv module.

Step 3: Using with statement open the required file as csv.

Step 4: Read the file and remove the new line.

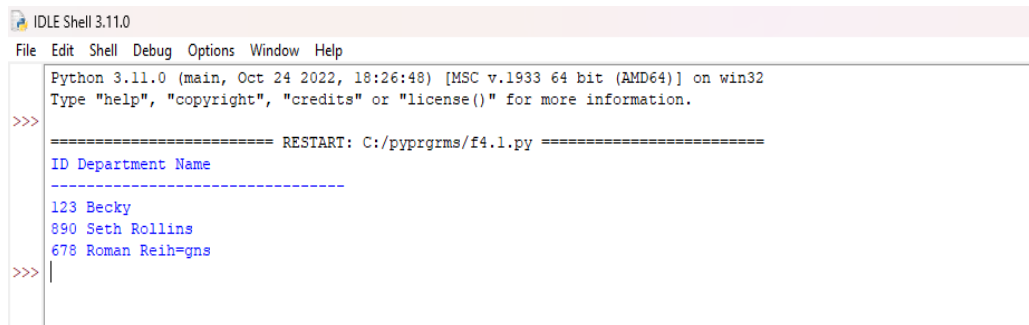
Step 5: Print the list of file content.

Step 6: Stop.

PROGRAM CODE:

```
import csv
with open('departments.csv', newline='') as csvfile:
    data = csv.DictReader(csvfile)
    print("ID Department Name")
    print("-----")
    for col in data:
        print(col['department_id'], col['department_name'])
```


OUTPUT:

A screenshot of the IDLE Shell 3.11.0 interface. The title bar reads 'IDLE Shell 3.11.0'. The menu bar includes 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main text area shows the following content: 'Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32', 'Type "help", "copyright", "credits" or "license()" for more information.', a prompt '>>>', a separator line '===== RESTART: C:/pyprgrms/f4.1.py =====', a header 'ID Department Name' followed by a dashed line, and three lines of data: '123 Becky', '890 Seth Rollins', and '678 Roman Reih=gns'. The prompt '>>>' is followed by a vertical cursor bar.

```
IDLE Shell 3.11.0
File Edit Shell Debug Options Window Help
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/pyprgrms/f4.1.py =====
ID Department Name
-----
123 Becky
890 Seth Rollins
678 Roman Reih=gns
>>> |
```

RESULT:

Program display column content from csv file has been executed successfully and output is verified.

LABCYCLE 5

EXPERIMENT NO:42

DICTIONARY TO A CSV FILE

Date: 16/01/2023

AIM: Write a Python program to write a Python dictionary to a csv file. After writing the CSV file read the CSV file and display the content.

ALGORITHM:

Step 1: Start.

Step 2: Using import statement import the csv module.

Step 3: Using with statement open the required file.

Step 4: Read the file.

Step 5: Traverse the file and print row.

Step 6: Stop.

PROGRAM CODE:

```
import csv
```

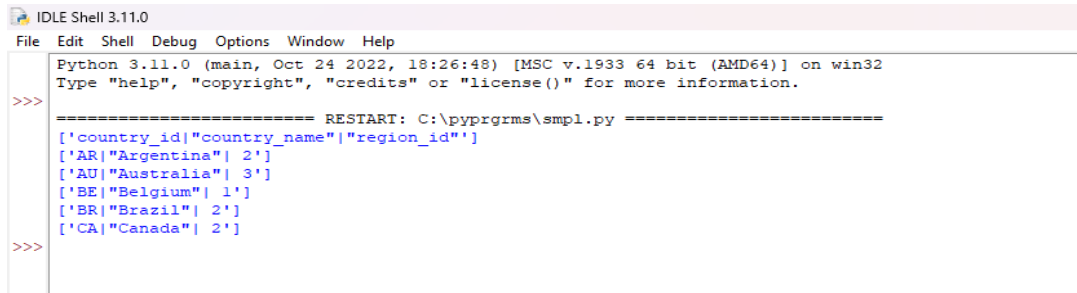
```
with open('mytemp.csv','r') as file:
```

```
    reader = csv.reader(file)
```

```
    for row in reader:
```

```
        print(row)
```

OUTPUT:



```
IDLE Shell 3.11.0
File Edit Shell Debug Options Window Help
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\pyprgrms\smp1.py =====
[{'country_id': 'country_name', 'region_id': 1}]
[{'AR': 'Argentina', '2'}]
[{'AU': 'Australia', '3'}]
[{'BE': 'Belgium', '1'}]
[{'BR': 'Brazil', '2'}]
[{'CA': 'Canada', '2'}]
>>>
```

RESULT:

Program to display dictionary to a csv file has been executed successfully and output is verified.