DEPARTMENT OF COMPUTER SCIENCE
RAJAGIRI COLLEGE OF SOCIALSCIENCES
(Autonomous)

MASTER OF COMPUTER APPLICATIONS

Data Analytics using Python
MCA 306

LAB RECORD

NAME : BENJAMIN KOSHY BIJU

SEMESTER : THIRD

REGISTER NO : 2124017

DEPARTMENT OF COMPUTER SCIENCE
RAJAGIRI COLLEGE OF SOCIAL SCIENCES
(Autonomous)

MASTER OF COMPUTER APPLICATIONS

CERTIFICATE

NAME : BENJAMIN KOSHY BIJU

SEMESTER : THIRD

REGISTER NO : 2124017

Certified that this is a bonafide record of work done by the student in the Software Laboratory of Rajagiri Department of Computer Science, Kalamassery.

Neethu Narayanan
Faculty in Charge

Dr. Bindiya M Varghese
Dean, ComputerScience

Internal Examiner

External Examiner

Place : Kalamassery
Date :

Table of Contents

Page No

Program 1	**Date:13/06/2022**

Write a program to perform different arithmetic operations on numbers in python.

```
n1=input("Enter first number:")
n2=input("Enter second number:")
sum=float(n1)+float(n2)
min = float(n1) - float(n2)
mul=float(n1)*float(n2)
div=float(n1)/float(n2)
print("Sum of ",n1," and ",n2," is ",sum)
print("Difference of ",n1," and ",n2," is ",min)
print("Product of ",n1," and ",n2," is ",mul)
print("Division of ",n1," and ",n2," is ",div)
```

Output

```
========================= RESTART: E:/python_prgm
Enter first number:2
Enter second number:3
Sum of  2  and  3  is  5.0
Difference of  2  and  3  is  -1.0
Product of  2  and  3  is  6.0
Division of  2  and  3  is  0.6666666666666666
```

Program 2	Date:13/06/2022

Write a program to create, concatenate, find range, slice, check the membership and print a string and accessing sub string from a given string.

```
str1=input("Enter String 1:")
str2=input("Enter String 2:")
str3=str1+str2
print("Concatenated string:",str3)
print("Range of String")
for i in range(len(str1)):
    print(i,str1[i])
s1=slice(3)
print("Slicing:",str1[s1])
print("Slicing:",str2[s1])
str4=input("Enter a string to check membership:")
if str4 in str1:
    print(str4+" found in the string",str1)
else:
    print(str4+"not in the string",str1)
print("Substring of",str2,"is",str2[0:3])
```

Output

```
Enter String 1:Good
Enter String 2:Morning
Concatenated string: GoodMorning
Range of String
0 G
1 o
2 o
3 d
Slicing: Goo
Slicing: Mor
Enter a string to check membership:oo
oo found in the string Good
Substring of Morning is Mor
```

Program 3	**Date:13/06/2022**

Write a python program
i)To create list with college names,
ii)Append a new college into the list,
iii)Add a new college at first positon and
iv)Remove a name from colleges lists.

```
college=['RCMAS','RCSS','RSET']

print(college)

college.append("SJCET")

print(college)

college.insert(0, "IIT")

print(college)

college.remove("SJCET")

print(college)
```

Output

```
========================= RESTART: E:/python_pr
['RCMAS', 'RCSS', 'RSET']
['RCMAS', 'RCSS', 'RSET', 'SJCET']
['IIT', 'RCMAS', 'RCSS', 'RSET', 'SJCET']
['IIT', 'RCMAS', 'RCSS', 'RSET']
```

Program 4	**Date:15/06/2022**

Write a program to demonstrate working with tuples in python

i. Create tuples with hardware's of computer: -
Hardware = ("Monitor", "RAM", "Expansion cards", "HDD")

ii. Check whether 'HDD' is present in the tuple or not

```
Hardware = ("Monitor", "RAM", "Expansion cards", "HDD")
print(Hardware)
N=input("Enter the element to search:")
my_result = False
for elem in Hardware :
 if N == elem :
   my_result = True
   break
print("Does the tuple contain the value mentioned ?")
print(my_result)
```

Output

```
======================== RESTART: E:/python_pro
('Monitor', 'RAM', 'Expansion cards', 'HDD')
Enter the element to search:RAM
Does the tuple contain the value mentioned ?
True
```

Program 5	**Date:15/06/2022**

Write a program to demonstrate working with dictionaries in python

i) Create a book dictionary
ii) Add items to dictionary
iii) Change values of a key
iv) Find the length of the made dictionary

```
books={"name":"PYTHON",
      "price":200,
      "year":1900
   }
print(books)
books["author"]="ABCD"
print("After adding",books)
books.update({"price":199})
print("After changing value",books)
print("Length of this dictionary",len(books))
```

Output

```
{'name': 'PYTHON', 'price': 200, 'year': 1900}
After adding {'name': 'PYTHON', 'price': 200, 'year': 1900, 'author': 'ABCD'}
After changing value {'name': 'PYTHON', 'price': 199, 'year': 1900, 'author': 'A
BCD'}
Length of this dictionary 4
```

Program 6	**Date:20/06/2022**

Write a python program to convert temperature to and from Celsius to Fahrenheit

```
cel=int(input("Enter in Celcius"))

to_far=cel*(9/5)+32

print("Farenheit is ",to_far)

far=int(input("Enter in Farenheit"))

to_cel=far-32*(5/9)

print("Farenheit is ",to_cel)
```

Output

```
------------------------- RESTART
Enter in Celcius 40
Farenheit is  104.0
Enter in Farenheit 23
Farenheit is  5.222222222222221
```

Program 7	**Date:20/06/2022**

Write a python program to construct the following pattern using nested for loop:

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

```
for i in range(6):
    for j in range(i):
        print("*",end='')
    print(" ")
for i in range(6):
    for j in range(6-i):
        print("*",end='')
    print(" ")
```

Output

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

Program 8	**Date:22/06/2022**

Write a python program to that accepts length of three sides of a triangle as inputs. The program should indicate whether or not the triangle is a right angled triangle (use Pythagorean theorem).

```
a=int(input("Enter Base"))

b=int(input("Enter Altitude"))

c=int(input("Enter Hypotenuse"))

hsq=c*c;

s=(a*a)+(b*b)

if hsq==s:

   print("It is right angled triangle")

else:

   print("It is not a right angled triangle")
```

Output

```
========================= RESTART: E:
Enter Base 2
Enter Altitude 3
Enter Hypotenuse 4
It is not a right angled triangle
```

Program 9	Date:22/06/2022
Python program to check whether the given integer is a multiple of both 5 and 7	

```
a=int(input("Enter a Number"))

if a%5==0 and a%7==0:

    print("Given integer is a multiple of both 5 and 7 ")

else:

    print("Given integer is not a multiple of both 5 and 7 ")
```

Output

```
======================== RESTART: E:/python_prgms/Q12.py ====
Enter a Number 7
Given integer is not a multiple of both 5 and 7
```

Program 10	**Date:22/06/2022**
Python program to display all integers within the range 100-200 whose sum of digits is an even number	

```
for i in range(100,200):
    num = i
    sum = 0
    while(num!=0):
        digit = num%10
        sum = sum + digit
        num = num//10
    if(sum%2==0):
        print(i)
```

Output

```
176
178
181
183
185
187
189
190
192
194
196
198
```

Program 11	Date:27/06/2022
Python program to implement a calculator to do basic operations	

```
operation = input('''
Please type in the math operation you would like to complete:
+ for addition
- for subtraction
* for multiplication
/ for division
''')
number_1 = int(input('Enter your first number: '))
number_2 = int(input('Enter your second number: '))
if operation == '+':
    print('{} + {} = '.format(number_1, number_2))
    print(number_1 + number_2)
elif operation == '-':
    print('{} - {} = '.format(number_1, number_2))
    print(number_1 - number_2)
elif operation == '*':
    print('{} * {} = '.format(number_1, number_2))
    print(number_1 * number_2)
elif operation == '/':
    print('{} / {} = '.format(number_1, number_2))
    print(number_1 / number_2)
else:
    print('You have not typed a valid operator, please run the program again.')
```

Output

```
Please type in the math operation you would like to complete:
+ for addition
- for subtraction
* for multiplication
/ for division
+
Enter your first number: 4
Enter your second number: 5
4 + 5 =
9
```

Program 12	Date:27/06/2022
Python program to implement matrix multiplication	

```
X = [[1,2,3],
   [4 ,4,5],
   [2 ,1,2]]
Y = [[6,5,3],
   [6,4,4],
   [3,2,9]]
result = [[0,0,0],
        [0,0,0],
        [0,0,0]]
for i in range(len(X)):
  for j in range(len(Y[0])):
    for k in range(len(Y)):
        result[i][j] += X[i][k] * Y[k][j
for r in result:
  print(r)
```

Output

```
====================
[27, 19, 38]
[63, 46, 73]
[24, 18, 28]
```

Program 13	Date:29/06/2022
Python program to print Fibonacci series using iteration	

```
n=int(input("Enter the limit"))
a=0
b=1
print(a)
print(b)
for i in range(2,n):
    c=a+b
    a=b
    b=c
    print(c)
```

Output

```
========================
Enter the limit 5
0
1
1
2
3
```

Program 14	Date:29/06/2022
Write a Python program to solve (x + y) * (x + y)	

```
x=int(input("Enter the x Value"))
y=int(input("Enter the y Value"))
s=x+y
sq=s*s
print("{}+{} * {}+{} ={}".format(x,y,x,y,sq))
```

Output

```
========================
Enter the  x  Value 4
Enter the y Value 5
4+5 * 4+5 =81
```

Program 15	**Date:04/07/2022**

A program to compute distance between two points taking input from the user Write a program add.py that takes 2 numbers as command line arguments and prints its sum.

```
x1=int(input("enter x1 : "))

x2=int(input("enter x2 : "))

y1=int(input("enter y1 : "))

y2=int(input("enter y2 : "))

result= ((((x2 - x1 )**2) + ((y2-y1)**2) )**0.5)

print("distance between",(x1,x2),"and",(y1,y2),"is : ",result)
```

Output

```
======================== RESTART: E:/python_prgms/Q20.py ====
enter x1 : 2
enter x2 : 3
enter y1 : 4
enter y2 : 5
distance between (2, 3) and (4, 5) is :  1.4142135623730951
```

Program 16	Date:04/07/2022

Python Program to implement sorting techniques: -

a. Bubble Sort

b. Quick Sort

Bubble Sort

```
def bubble_sort(list1):
    for i in range(0,len(list1)-1):
        for j in range(len(list1)-1):
            if(list1[j]>list1[j+1]):
                temp = list1[j]
                list1[j] = list1[j+1]
                list1[j+1] = temp
    return list1
list1 = [5, 3, 8, 6, 7, 2]
print("The unsorted list is: ", list1)
print("The sorted list is: ", bubble_sort(list1))
```

Output

```
=========== RESTART: C:/Users/H P/Desktop/MCA/Sem 3/Python/pgm22a.py ===========
The unsorted list is:  [5, 3, 8, 6, 7, 2]
The sorted list is:  [2, 3, 5, 6, 7, 8]
```

Quick Sort

```
def partition(array, low, high):
pivot = array[high]
  i = low - 1
  for j in range(low, high):
    if array[j] <= pivot:
      i = i + 1
      (array[i], array[j]) = (array[j], array[i])
```

```
  (array[i + 1], array[high]) = (array[high], array[i + 1])
  return i + 1
def quickSort(array, low, high):
  if low < high:
    pi = partition(array, low, high)
    quickSort(array, low, pi - 1)
    quickSort(array, pi + 1, high)
data = [8, 7, 2, 1, 0, 9, 6]
print("Unsorted Array")
print(data)
size = len(data)
quickSort(data, 0, size - 1)
print('Sorted Array in Ascending Order:')
print(data)
```

Output

Program 17	Date:06/07/2022

Python Program to implement searching techniques: -

a. Linear search

b. Binary search

a. Linear search

```
def linear_search(arr, a, b):
    for i in range(0, a):
        if (arr[i] == b):
            return i
    return -1
arr = [9, 7, 5, 3, 1]
print("The array given is ", arr)
b = 5
print("Element to be found is ", b)
a = len(arr)
index = linear_search(arr, a, b)
if(index == -1):
    print("Element is not in the list")
else:
    print("Index of the element is: ", index)
```

Output

```
======================== RESTART: E:/pyt
The array given is  [9, 7, 5, 3, 1]
Element to be found is  5
Index of the element is:  2
```

b. Binary search

```
def binary_search(arr, a, low, high):
while low <= high:
mid = low + (high - low)//2
if arr[mid] == a:
return mid
```

```
elif array[mid] < a:
low = mid + 1
else:
high = mid - 1
return -1
arr = [1, 2, 3, 4, 5, 6, 7]
a = 4
print("The given array is", arr)
print("Element to be found is ", a)
index = binary_search(arr, a, 0, len(arr)-1)
if index != -1:
print("The Index of the element is " + str(index))
else:
print("Element Not found")
```

Output

```
The given array is [1, 2, 3, 4, 5, 6, 7]
Element to be found is  4
The Index of the element is 3
```

Program 18	**Date:06/07/2022**
Write a recursive function to calculate the sum of numbers from 0 to 10	

```
def recur_sum(n):
  if n <= 1:
     return n
  else:
     return n + recur_sum(n-1)
num = 10
if num < 0:
  print("Enter a positive number")
else:
  print("The sum is",recur_sum(num))
```

Output

```
======================== R
The sum is 55
```

Program 19	Date:06/07/2022

Write a Python function student_data () which will print the id of a student (student_id). If the user passes an argument student_name or student_class the function will print the student name and class.

```
def student_data(student_id, **arr):
    print(f'\nStudent ID: {student_id}')
    if 'student_name' in arr:
        print(f"Student Name: $ {arr['student_name']}")
        if 'student_name' and 'student_class' in arr:
            print(f"\nStudent Name: $ {arr['student_name']}")
            print(f"Student Class: $ {arr['student_class']}")
student_data(student_id='SV12', student_name='Jean Garner')
student_data(student_id='SV12', student_name='Jean Garner', student_class ='V')
```

Output

```
Student ID: SV12
Student Name: $ Jean Garner

Student ID: SV12
Student Name: $ Jean Garner

Student Name: $ Jean Garner
Student Class: $ V
```

Program 20	Date:11/07/2022
Write a Python program to reverse the digits of a given number and add it to the original, If the sum is not a palindrome repeat this procedure	

```
def rev_number(n):
  s = 0
  while True:
    k = str(n)
    if k == k[::-1]:
      break
    else:
      m = int(k[::-1])
      n += m
      s += 1
  return n
print(rev_number(1234))
print(rev_number(1473))
```

Output

```
--------------
5555
9339
```

Program 21	Date:13/07/2022

Write a menu-driven program that creates a Phonebook Directory using the different functions. We will addthe following features tothe Phonebook Directory:

a. **Storing the Contact Numbers of People**
b. **Searching for the Contact Number using the person'sname**

```
print( "WELCOME TO THE PHONEBOOK DIRECTORY")
  filename = "myphonebook.txt"
myfile = open(filename, "a+")
myfile.close
def main_menu():
    print( "\nMAIN MENU\n")
    print( "1. Show all existing Contacts")
    print( "2. Add a new Contact")
    print( "3. Search the existing Contact")
    print( "4. Exit")
    choice = input("Enter your choice: ")
    if choice == "1":
        myfile = open(filename, "r+")
        filecontents = myfile.read()
        if len(filecontents) == 0:
            print( "There is no contact in the phonebook.")
        else:
            print(filecontents)
        myfile.close
        enter = input("Press Enter to continue ...")
        main_menu()
    elif choice == "2":
        newcontact()
        enter = input("Press Enter to continue ...")
        main_menu()
```

```
  elif choice == "3":
      searchcontact()
      enter = input("Press Enter to continue ...")
      main_menu()
  elif choice == "4":
      print("Thank you for using Phonebook!")
  else:
      print( "Please provide a valid input!\n")
      enter = input( "Press Enter to continue ...")
      main_menu()
def searchcontact():
  searchname = input( "Enter First name for Searching contact record: ")
  remname = searchname[1:]
  firstchar = searchname[0]
  searchname = firstchar.upper() + remname
  myfile = open(filename, "r+")
  filecontents = myfile.readlines()

  found = False
  for line in filecontents:
      if searchname in line:
          print( "Your Required Contact Record is:", end = " ")
          print( line)
          found = True
          break
  if found == False:
      print( "The Searched Contact is not available in the Phone Book", searchname)
def input_firstname():
  first = input( "Enter your First Name: ")
```

```
    remfname = first[1:]
    firstchar = first[0]
    return firstchar.upper() + remfname
  def input_lastname():
    last = input( "Enter your Last Name: ")
    remlname = last[1:]
    firstchar = last[0]
    return firstchar.upper() + remlname
  def newcontact():
    firstname = input_firstname()
    lastname = input_lastname()
    phoneNum = input( "Enter your Phone number: ")
    emailID = input( "Enter your E-mail Address: ")
    contactDetails = ("[" + firstname + " " + lastname + ", " + phoneNum + ", " + emailID +
"]\n")
    myfile = open(filename, "a")
    myfile.write(contactDetails)
    print( "The following Contact Details:\n " + contactDetails + "\nhas been stored
successfully!")
  main_menu()
```

Output

```
WELCOME TO THE PHONEBOOK DIRECTORY

MAIN MENU

1. Show all existing Contacts
2. Add a new Contact
3. Search the existing Contact
4. Exit
Enter your choice: 2
Enter your First Name: Merin
Enter your Last Name: Thomas
Enter your Phone number: 4566738654
Enter your E-mail Address: merin@gmail.com
The following Contact Details:
 [Merin Thomas, 4566738654, merin@gmail.com]

has been stored successfully!
Press Enter to continue ...
```

Program 22	Date:13/07/2022

Perform various set operations

Set Union
Set Intersection
Set Difference

```
A = {0, 2, 4, 6, 8};

B = {1, 2, 3, 4, 5};

print("Union :", A | B)

print("Intersection :", A & B)

print("Difference :", A - B)

print("Symmetric difference :", A ^ B)
```

Output

```
Union : {0, 1, 2, 3, 4, 5, 6, 8}
Intersection : {2, 4}
Difference : {0, 8, 6}
Symmetric difference : {0, 1, 3, 5, 6, 8}
```

Program 23	Date:18/07/2022

Create a dictionary to store the name, roll_no, total_mark of N students. Now print the details of the student who has got the highest total_mark

```
n=int(input("Enter number of students:"))
result={}
for i in range(n):
    print("Enter Details of student No.",i+1)
    rno=int(input("RollNo:"))
    name=input("Name:")
    marks=int(input("Marks:"))
    result[rno]=[name,marks]
print(result)
for student in result:
    if result[student][1]>75:
        print(result[student][0])
```

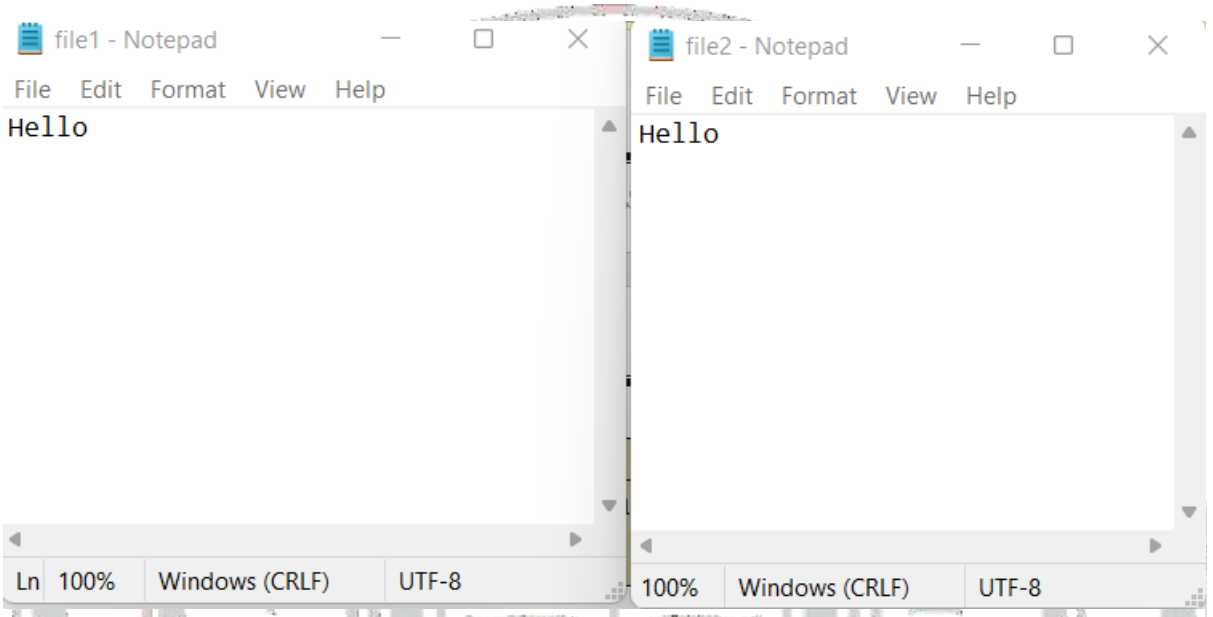
Output

```
Enter number of students:2
Enter Details of student No. 1
RollNo:1
Name:joel
Marks:70
Enter Details of student No. 2
RollNo:2
Name:justin
Marks:89
{1: ['joel', 70], 2: ['justin', 89]}
justin
```

Program 24	Date:18/07/2022
Write a Python program to copy the contents of a file into another file line by line	

```
with open('file1.txt','r') as firstfile, open('file2.txt','a') as secondfile:
    for line in firstfile:
        secondfile.write(line)
```

Output



Program 25	Date:25/07/2022

Use os module to perform

A. **Create a directory**
B. **Directory listing**
C. **Search for ".py" files**
D. **Remove a particular file**

A. Create a directory

```
import os
directory = "MyDirectory"
parent_dir = "E:\Rosu\python_prgms"
path = os.path.join(parent_dir, directory)
os.mkdir(path)
print("Directory '% s' created" % directory)
directory = "Dir"
parent_dir = "E:\Rosu\python_prgms"
mode = 0o666
path = os.path.join(parent_dir, directory)
os.mkdir(path, mode)
print("Directory '% s' created" % directory)
```

Output

```
Directory 'MyDirectory' created
Directory 'Dir' created
```

B. Directory Listing

```
import os
path = "/"
dir_list = os.listdir(path)
print("Files and directories in '", path, "' :")
print(dir_list)
```

Output

```
Files and directories in ' / ' :
['$RECYCLE.BIN', 'Intership', 'MAJOR PROJECT', 'MBA DOCUMENTS',
'UK process']
```

C. Search for .py files

```
import os
dir_path = os.path.dirname(os.path.realpath(_file_))
for root, dirs, files in os.walk(dir_path):
    for file in files:
        if file.endswith('.py'):
            print (root+'/'+str(file))
```

Output

```
E:\Rosu\python_prgms/20.py
E:\Rosu\python_prgms/24.py
E:\Rosu\python_prgms/add_two_numbers.py
E:\Rosu\python_prgms/binary.py
E:\Rosu\python_prgms/Q1.py
E:\Rosu\python_prgms/Q10.py
E:\Rosu\python_prgms/Q11.py
E:\Rosu\python_prgms/Q12.py
E:\Rosu\python_prgms/Q13.py
E:\Rosu\python_prgms/Q14.py
E:\Rosu\python_prgms/Q15.py
E:\Rosu\python_prgms/Q16.py
E:\Rosu\python_prgms/Q17.py
E:\Rosu\python_prgms/Q18.py
E:\Rosu\python_prgms/Q19.py
E:\Rosu\python_prgms/Q19new.py
```

D. Remove a particular file

```
import os
file = 'file1.txt'
    location = "E:\Rosu\python_prgms"
    path = os.path.join(location, file)
    os.remove(path)
```

Output

```
===================== RESTART: E
```

Program 26	**Date:25/07/2022**
Create a simple banking application by using inheritance.	

```
class Bank_Account:
        def_init_(self):
                self.balance=0
                print("Hello!!! Welcome to the Deposit & Withdrawal Machine")
def deposit(self):
                amount=float(input("Enter amount to be Deposited: "))
                self.balance += amount
                print("\n Amount Deposited:",amount)
def withdraw(self):
                amount = float(input("Enter amount to be Withdrawn: "))
                if self.balance>=amount:
                        self.balance-=amount
                        print("\n You Withdrew:", amount)
                else:
                        print("\n Insufficient balance ")
def display(self):
                print("\n Net Available Balance=",self.balance)
s = Bank_Account()
s.deposit()
s.withdraw()
s.display()
```

Output

```
===================== RESTART: E:/Rosu/python_prgms/Q26.
Hello!!! Welcome to the Deposit & Withdrawal Machine
Enter amount to be Deposited: 5000

 Amount Deposited: 5000.0
Enter amount to be Withdrawn: 100

 You Withdrew: 100.0

 Net Available Balance= 4900.0
```

Program 27	Date:27/07/2022

Implement the concept of polymorphism while creating class methods

```
from math import pi
class Rectangle:
  def_init_(self, length, breadth):
    self.l = length
    self.b = breadth
  def perimeter(self):
    return 2*(self.l + self.b)
  def area(self):
    return self.l * self.b
class Circle:
  def_init_(self, radius):
    self.r = radius
  def perimeter(self):
    return 2 * pi * self.r
  def area(self):
    return pi * self.r ** 2
# Initialize the classes
rec = Rectangle(5,3)
cr = Circle(4)
print("Perimter of rectangel: ",rec.perimeter())
print("Area of rectangel: ",rec.area())
print("Perimter of Circle: ",cr.perimeter())
print("Area of Circle: ",cr.area())
```

Output

```
Perimter of rectangel:  16
Area of rectangel:  15
Perimter of Circle:  25.132741228718345
Area of Circle:  50.26548245743669
```

Program 28	Date:01/08/2022

Create a MySQL database to perform the following operations using python.

1. **Create a STUDENTS table with 4 columns**
2. **Perform,**
   - **SELECT**
   - **INSERT**
   - **UPDATE**
   - **DELETE**

//Creating database mydatabase

```
import mysql.connector
mydb = mysql.connector.connect(
  host="localhost",
  user=”root”,
  password=""
)
mycursor = mydb.cursor()
mycursor.execute("CREATE DATABASE mydatabase")
```

Output

```
D:\Python\sqlpython>python show.py
('and_ems',)
('carservice',)
('college',)
('information_schema',)
('mydatabase',)
('mysql',)
('performance_schema',)
('phpmyadmin',)
('test',)
```

//1)Creating table

```
import mysql.connector
mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="",
  database="mydatabase"
)
mycursor = mydb.cursor()
```

mycursor.execute("CREATE TABLE student (id VARCHAR(25), name VARCHAR(255),age varchar(22),course varchar(50))")

Output

```
D:\Python\sqlpython>python checktable.py
('student',)
```

//2)a)Insert

```
import mysql.connector
mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="",
  database="mydatabase"
)
mycursor = mydb.cursor()
sql = "INSERT INTO student (id, name,age,course) VALUES (%s, %s,%s,%s)"
val = ("s01", “John",”23”,”CS”)
mycursor.execute(sql, val)
mydb.commit()
print(mycursor.rowcount, "record inserted.")
```

Output

```
D:\Python\sqlpython>python insert.py
1 record inserted.
```

//b)Select

```
import mysql.connector
mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="",
  database="mydatabase"
)
```

```
mycursor = mydb.cursor()
mycursor.execute("SELECT * FROM student")
myresult = mycursor.fetchall()
for x in myresult:
  print(x)
```

Output

```
('s01', 'John','23','CS')
```

//c)Update

```
import mysql.connector
mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="",
  database="mydatabase"
)
mycursor = mydb.cursor()
sql = "UPDATE student SET course = 'EE' WHERE id = 's01'"
mycursor.execute(sql)
mydb.commit()
print(mycursor.rowcount, "record(s) affected")
```

Output

```
1 record(s) affected
```

//d)Delete

```
import mysql.connector
mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="",
```

```
  database="mydatabase"
)
mycursor = mydb.cursor()
sql = "DELETE FROM student WHERE id = 's01'"
mycursor.execute(sql)
mydb.commit()
print(mycursor.rowcount, "record(s) deleted")
```

Output

```
1 record(s) deleted
```

Program 29	Date:01/08/2022
Create a simple application form using a CGI program and pass information using the POST method.	

```
#!C:\\Users\\hp\\AppData\\Local\\Programs\\Python\\Python39\\python.exe

# Importing the 'cgi' module

import cgi

print("Content-type: text/html\r\n\r\n")

print("<html><body>")

print("<h1> Hello Program! </h1>")

# Using the inbuilt methods


form = cgi.FieldStorage()

if form.getvalue("name"):

        name = form.getvalue("name")

        print("<h1>Hello" +name+"! Thanks for using my script!</h1><br />")

if form.getvalue("happy"):

        print("<p> Yayy! I'm happy too! </p>")

if form.getvalue("sad"):

        print("<p> Oh no! Why are you sad? </p>")


# Using HTML input and forms method

print("<form method='post' action='first.py'>")

print("<p>Name: <input type='text' name='name' /></p>")

print("<input type='checkbox' name='happy' /> Happy")

print("<input type='checkbox' name='sad' /> Sad")

print("<input type='submit' value='Submit' />")

print("</form")

print("</body></html>"
```

Output


Hello Program!
Name: Benjamin
Happy Sad Submit
Hello Program!
HelloBenjamin! Thanks for using my script!
Yayy! I'm happy too!
Name:
Happy Sad Submit

Program 30	Date:03/08/2022

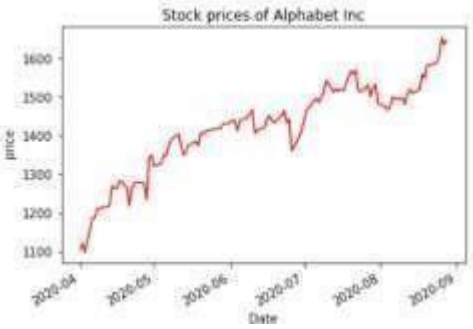
Visualize the following using the given dataset(alphabet_stock_data.csv),

a. **create a line plot of the historical stock prices of Alphabet Inc. between two specific dates.**
b. **create a bar plot of the trading volume of Alphabet Inc. stock between two specific dates.**
c. **create a stacked histograms plot with more bins of opening, closing, high, low stock prices of Alphabet Inc. between two specific dates.**
d. **create a scatter plot of the trading volume/stock prices of Alphabet Inc. stock between two specific dates.**

a.

```
import pandas as pd

import matplotlib.pyplot as plt

df = pd.read_csv("alphabet_stock_data.csv")

start_date = pd.to_datetime('2020-3-1')

end_date = pd.to_datetime('2020-08-30')

df['Date'] = pd.to_datetime(df['Date'])

new_df = (df['Date']>= start_date) & (df['Date']<= end_date)

df1 = df.loc[new_df]

df2 = df1.set_index('Date')

plt.title("Stock prices of Alphabet Inc")

plt.xlabel("Date")

plt.ylabel(" price")

df2['Close'].plot(color='red');

plt.show()
```

Output



b.

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv("alphabet_stock_data.csv")
start_date = pd.to_datetime('2020-4-1')
end_date = pd.to_datetime('2020-4-30')
df['Date'] = pd.to_datetime(df['Date'])
new_df = (df['Date']>= start_date) & (df['Date'] <= end_date)
df1 = df.loc[new_df]
df2 = df1.set_index('Date')
plt.title('Opening and closing stock price')
plt.xlabel('Date')
plt.ylabel('Trading Volume')
df2['Volume'].plot(kind='bar')
plt.show()
```

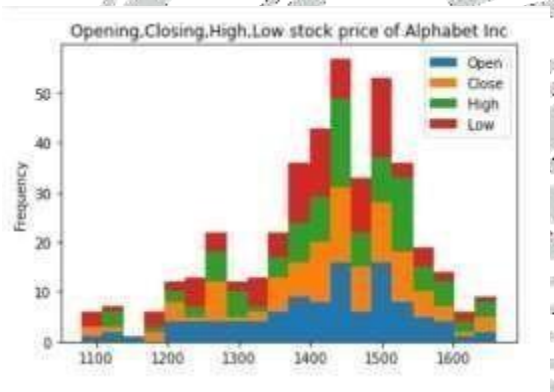
Output

c.

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv("alphabet_stock_data.csv")
start_date = pd.to_datetime('2020-03-1')
```

```
end_date = pd.to_datetime('2020-08-30')
df['Date'] = pd.to_datetime(df['Date'])
new_df = (df['Date']>= start_date) & (df['Date'] <= end_date)
df1 = df.loc[new_df]
df2 = df1[['Open','Close','High','Low']]
df2.plot.hist(stacked=True, bins=20)
plt.title('Opening,Closing,High,Low stock price of Alphabet Inc')
plt.show()
```

Output



d.

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv("alphabet_stock_data.csv")
start_date = pd.to_datetime('2020-4-1')
end_date = pd.to_datetime('2020-9-30')
df['Date'] = pd.to_datetime(df['Date'])
new_df = (df['Date']>= start_date) & (df['Date']<= end_date)
df1 = df.loc[new_df]
df2 = df1.set_index('Date')
x = ['Close']
y = ['Volume']
```

```
df2.plot.scatter(x,y,s=40)
plt.grid(True)
plt.title('Opening,Closing,High,Low stock price of Alphabet Inc')
plt.xlabel("Stock Price")
plt.ylabel("Trading Volume")
plt.show()
```

Output

Program 31	Date:29/08/2022

Handle the given dataset (Data.csv) with adequate preprocessing steps mentioned and visualize the dataset with appropriate graphs.

A. **Handle Missing Data Values**
B. **Encode the categorical data**
C. **Scale your features**

A.

```
import pandas as pd
dataset = pd.read_csv('Data.csv')
dataset.head()
dataset.shape
dataset.isna().sum()
```

B.

```
import pandas as pd
dataset = pd.read_csv('Data.csv')
dataset.head()
dataset.shape
dataset['Age'].fillna(dataset['Age'].median(),inplace=True)
dataset['Salary'].fillna(dataset['Salary'].median(),inplace=True)
dataset.isnull().sum()
```

C.

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.2, random_state=0)
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
```

Output

```
Out[3]: Country      0
        Age          1
        Salary       1
        Purchased    0
        dtype: int64
```

```
Out[4]: Country      0
        Age          0
        Salary       0
        Purchased    0
        dtype: int64
```

	0	1	2	3	4
0	-1	1.73205	-0.57735	-0.294687	0.133962
1	1	-0.57735	-0.57735	-0.938959	1.22637
2	1	-0.57735	-0.57735	0.341745	-1.7415
3	-1	1.73205	-0.57735	-0.0589215	-0.990562
4	1	-0.57735	-0.57735	1.63449	1.41175
5	1	-0.57735	-0.57735	1.40233	0.113352
6	-1	-0.57735	1.73205	-0.718042	0.391581
7	-1	-0.57735	1.73205	-1.35519	-0.535948

	0	1	2	3	4
0	-1	1.73205	-0.57735	-2.41578	-0.906819
1	-1	-0.57735	1.73205	1.82057	2.24644

Program 32	**Date:29/08/2022**

Using the given dataset (dirtydata.csv),

a. **Handle the data with empty cells (Use dropna() and fillna())**
b. **Replace the empty cells using mean, median, and mode.**
c. **Handle the data in the wrong format.**
d. **Handle the wrong data from the dataset.**
e. **Discover and remove duplicates.**

1)

```
import pandas as pd

df = pd.read_csv('dirtydata.csv')

df.dropna(inplace = True)

df.fillna(130, inplace = True)

print(df.to_string())
```

Output

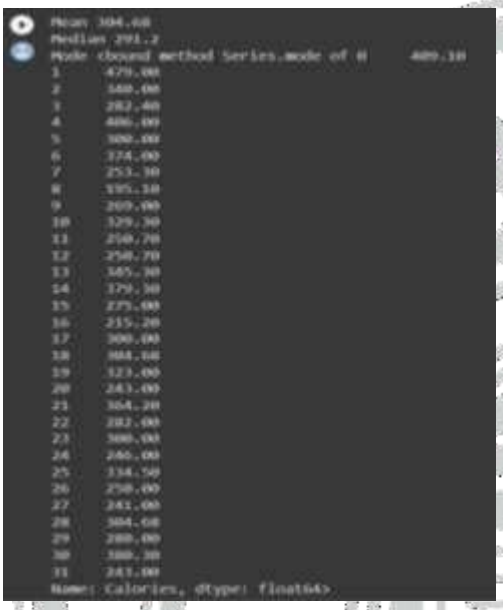
	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2020/12/01'	110	130	409.1
1	60	'2020/12/02'	117	145	479.0
2	60	'2020/12/03'	103	135	340.0
3	45	'2020/12/04'	109	175	282.4
4	45	'2020/12/05'	117	148	406.0
5	60	'2020/12/06'	102	127	300.0
6	60	'2020/12/07'	110	136	374.0
7	450	'2020/12/08'	104	134	253.3
8	30	'2020/12/09'	109	133	195.1
9	60	'2020/12/10'	98	124	269.0
10	60	'2020/12/11'	103	147	329.3
11	60	'2020/12/12'	100	120	250.7
12	60	'2020/12/12'	100	120	250.7
13	60	'2020/12/13'	106	128	345.3
14	60	'2020/12/14'	104	132	379.3
15	60	'2020/12/15'	98	123	275.0
16	60	'2020/12/16'	98	120	215.2
17	60	'2020/12/17'	100	120	300.0
18	60	'2020/12/18'	103	123	323.0
19	45	'2020/12/19'	97	125	243.0
20	60	'2020/12/20'	108	131	364.2
21	60	'2020/12/21'	130	101	300.0
22	45	'2020/12/22'	105	132	246.0
23	60	'2020/12/23'	102	126	334.5
24	60	'2020/12/24'	100	120	250.0
25	60	'2020/12/25'	102	129	380.3
26	60	20201226	100	120	250.0
27	60	'2020/12/27'	92	118	241.0
28	60	'2020/12/28'	103	132	280.0
29	60	'2020/12/29'	100	132	280.0
30	60	'2020/12/30'	102	129	380.3
31	60	'2020/12/31'	92	115	243.0

2)

```
import pandas as pd

df = pd.read_csv(' dirtydata.csv')

x = df["Calories"].mean()

y =  df["Calories"].median()

z = df["Calories"].mode

df["Calories"].fillna(x, inplace = True)

df["Calories"].fillna(y,  inplace  =  True)
```

```
df["Calories"].fillna(z, inplace = True)

print("Mean", x)

print("Median", y)

print("Mode", z)
```

Output



3)

```
import pandas as pd

df = pd.read_csv(' dirtydata.csv')

df['Date'] = pd.to_datetime(df['Date'])

print(df.to_string())
```

Output

	Duration	Date	Pulse	Maxpulse	Calories
0	60	2020-12-01	110	130	409.1
1	60	2020-12-02	117	145	479.0
2	60	2020-12-03	103	135	340.0
3	45	2020-12-04	109	175	282.4
4	45	2020-12-05	117	148	406.0
5	60	2020-12-06	102	127	300.0
6	60	2020-12-07	110	136	374.0
7	450	2020-12-08	104	134	253.3
8	30	2020-12-09	109	133	195.1
9	60	2020-12-10	98	124	269.0
10	60	2020-12-11	103	147	329.3
11	60	2020-12-12	100	120	250.7
12	60	2020-12-12	100	120	250.7
13	60	2020-12-13	106	128	345.3
14	60	2020-12-14	104	132	379.3
15	60	2020-12-15	98	123	275.0
16	60	2020-12-16	98	120	215.2
17	60	2020-12-17	100	120	300.0
18	45	2020-12-18	90	112	NaN
19	60	2020-12-19	103	123	323.0
20	45	2020-12-20	97	125	243.0
21	60	2020-12-21	108	131	364.2
22	45	NaT	100	119	282.0
23	60	2020-12-23	130	101	300.0
24	45	2020-12-24	105	132	246.0
25	60	2020-12-25	102	126	334.5
26	60	2020-12-26	100	120	250.0
27	60	2020-12-27	92	118	241.0
28	60	2020-12-28	103	132	NaN
29	60	2020-12-29	100	132	280.0
30	60	2020-12-30	102	129	380.3
31	60	2020-12-31	92	115	243.0

4)

```
import pandas as pd
df =pd.read_csv(' dirtydata.csv')
df.loc[7, 'Duration'] = 45
print(df)
```

Output

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2020/12/01'	110	130	409.1
1	60	'2020/12/02'	117	145	479.0
2	60	'2020/12/03'	103	135	340.0
3	45	'2020/12/04'	109	175	282.4
4	45	'2020/12/05'	117	148	406.0
5	60	'2020/12/06'	102	127	300.0
6	60	'2020/12/07'	110	136	374.0
7	45	'2020/12/08'	104	134	253.3
8	30	'2020/12/09'	109	133	195.1
9	60	'2020/12/10'	98	124	269.0
10	60	'2020/12/11'	103	147	329.3
11	60	'2020/12/12'	100	120	250.7
12	60	'2020/12/12'	100	120	250.7
13	60	'2020/12/13'	106	128	345.3
14	60	'2020/12/14'	104	132	379.3
15	60	'2020/12/15'	98	123	275.0
16	60	'2020/12/16'	98	120	215.2
17	60	'2020/12/17'	100	120	300.0
18	45	'2020/12/18'	90	112	NaN
19	60	'2020/12/19'	103	123	323.0
20	45	'2020/12/20'	97	125	243.0
21	60	'2020/12/21'	108	131	364.2
22	45	NaN	100	119	282.0
23	60	'2020/12/23'	130	101	300.0
24	45	'2020/12/24'	105	132	246.0
25	60	'2020/12/25'	102	126	334.5
26	60	20201226	100	120	250.0
27	60	'2020/12/27'	92	118	241.0
28	60	'2020/12/28'	103	132	NaN
29	60	'2020/12/29'	100	132	280.0
30	60	'2020/12/30'	102	129	380.3
31	60	'2020/12/31'	92	115	243.0

5)

```
import pandas as pd
df = pd.read_csv(' dirtydata.csv')
```

```
print(df.duplicated())

df.drop_duplicates(inplace = True)

print(df)
```

Output

```
7      False
8      False
9      False
10     False
11     False
12      True
13     False
14     False
15     False
16     False
17     False
18     False
19     False
20     False
21     False
22     False
23     False
24     False
25     False
26     False
27     False
28     False
29     False
30     False
31     False
dtype: bool
```

Program 33	**Date:31/08/2022**

Create a cricketers dataset using a dictionary of lists, and create a new attribute 'Experience Category' using 'Age' as the binning factor.

```
import numpy as np

import pandas as pd

cricketers = {

 'Name': ['Virat Kohli', 'Rohit Sharma', 'Shahid Afridi', 'Anderson', 'Jadeja', 'Fareed Ahmed',
'Hamid Hassan', 'James Anderson'],

 'Age' : [31, 33, 25, 31, 26, 19, 35, 29],

 'Grade' : ['Best', 'Better', 'Good', 'Best', 'Better', 'Good', 'Best', 'Best'],

 'Role' : ['Batsmen', 'Batsmen', 'Batsmen', 'Bowler', 'Bowler', 'Bowler','All Rounder', 'All
Rounder'],

 'Rating' : [871, 869, 829, 722, 719, 701, 765, 858],

 'Country' : ['IND', 'IND', 'PAK', 'NZ', 'IND', 'AFG', 'AFG', 'ENG']

}

df_cricketers = pd.DataFrame(cricketers)

df_cricketers.head()

bins = [0,20,30, 40]

group_names = ['Young', 'Senior', 'Super Senior']

df_cricketers['Experience Category'] = pd.cut(df_cricketers['Age'], bins,
labels=group_names)

df_cricketers.head()
```

Output

	Name	Age	Grade	Role	Rating	Country	Experience Category
0	Virat Kohli	31	Best	Batsmen	871	IND	Super Senior
1	Rohit Sharma	33	Better	Batsmen	869	IND	Super Senior
2	Shahid Afridi	25	Good	Batsmen	829	PAK	Senior
3	Anderson	31	Best	Bowler	722	NZ	Super Senior
4	Jadeja	26	Better	Bowler	719	IND	Senior

Program 34 **Date:31/08/2022**

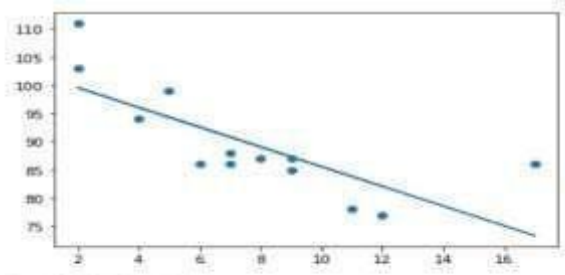
a. car_age = [5,7,8,7,2,17,2,9,4,11,12,9,6]
b. car_speed = [99,86,87,88,111,86,103,87,94,78,77,85,86]

Using the given dataset,

a. Draw the line of linear regression
b. Evaluate, how well the data fit in linear regression?
c. Predict the speed of a 10-year-old car.

```
import matplotlib.pyplot as plt

from scipy import stats

x = [5,7,8,7,2,17,2,9,4,11,12,9,6]

y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

slope, intercept, r, p, std_err = stats.linregress(x, y)

def myfunc(x):

  return slope * x + intercept

mymodel = list(map(myfunc, x))

plt.scatter(x, y)

plt.plot(x, mymodel)

plt.show()

speed = myfunc(10)

print(speed)
```

Output




```
85.59308314937454
-0.758591524376155
```

Program 35	Date:05/09/2022

Using the dataset(cars.csv),

a. **Predict the CO2 emission of a car with a weight 2300Kg and a volume of 1300cm3.**
b. **Print the coefficient values of the regression object.**

```
import pandas

from sklearn import linear_model

df = pandas.read_csv("cars.csv")

X = df[['Weight', 'Volume']]

y = df['CO2']

regr = linear_model.LinearRegression()

regr.fit(X, y)

print('Predicting the CO2 emission of a car with a weight 2300Kg and a volume is 1300cm')

predictedCO2 = regr.predict([[2300, 1300]])

print(predictedCO2)

print('Coefficient values of the regression object')

print(regr.coef_)
```

Output

```
Predicting the CO2 emission of a car with a weight 2300Kg and a volume is 1300cm
[107.2087328]
Coefficient values of the regression object
[0.00755095 0.00780526]
```

Program 36	Date:05/09/2022

Using the insurance dataset (insurance.csv) with adequate preprocessing steps,

a. **Create a regression model**
b. **Visualize the correlation among variables using a heatmap.**
c. **Evaluate the model.**
d. **Predict the charges for a person with age=30, bmi=35.000, and smoker.**

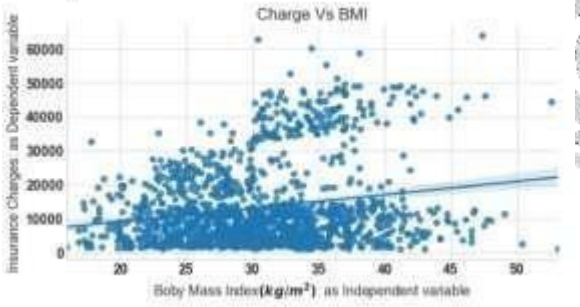
```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.metrics import accuracy_score

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LogisticRegression

plt.rcParams['figure.figsize'] = [8,5]

plt.rcParams['font.size'] =14

plt.rcParams['font.weight']= 'bold'

plt.style.use('seaborn-whitegrid')

df = pd.read_csv('insurance.csv')

x = df.iloc[:, [2, 3]].values

y = df.iloc[:, 4].values

xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size = 0.25, random_state = 0)

print('Dataset')

print(df.head())

print('Linear Regression Model')

sns.lmplot(x='bmi',y='charges',data=df,aspect=2,height=4)

plt.xlabel('Boby Mass Index$(kg/m^2)$: as Independent variable')

plt.ylabel('Insurance Charges: as Dependent variable')

plt.title('Charge Vs BMI')

plt.show()
```

```
print('Visualizing the correlation among variables using a heatmap.')
corr = df.corr()
sns.heatmap(corr, cmap = 'Wistia', annot= True);
#Accuracy
sc_x = StandardScaler()
xtrain = sc_x.fit_transform(xtrain)
xtest = sc_x.transform(xtest)
classifier = LogisticRegression(random_state = 0)
classifier.fit(xtrain, ytrain)
y_pred = classifier.predict(xtest)
print ("Accuracy : ", accuracy_score(ytest, y_pred))
```

Output

```
Dataset
   age     sex     bmi  children smoker     region      charges
0   19  female  27.900         0    yes  southwest  16884.92400
1   18    male  33.770         1     no  southeast   1725.55230
2   28    male  33.000         3     no  southeast   4449.46200
3   33    male  22.705         0     no  northwest  21984.47061
4   32    male  28.880         0     no  northwest   3866.85520
```

```
Linear Regression Model
```




```
Visualizing the correlation among variables using a heatmap.
Accuracy :  0.7880597014925373
```

Program 37	Date:07/09/2022

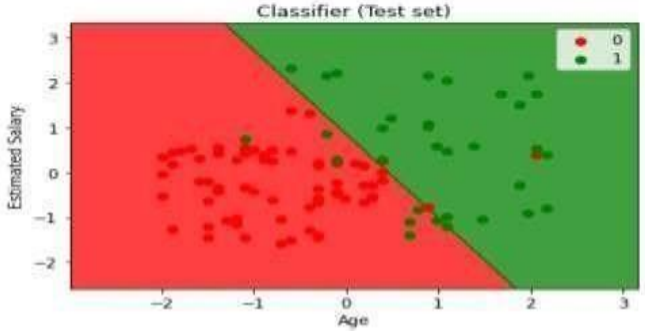
Evaluate the dataset(User_Data.csv) and predict whether a user will purchase the company's product or not. (Use logistic regression). Also print the confusion matrix and generate the classification report.

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LogisticRegression

from matplotlib.colors import ListedColormap

dataset = pd.read_csv('User_Data.csv')

x = dataset.iloc[:, [2, 3]].values

y = dataset.iloc[:, 4].values

xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size = 0.25, random_state = 0)

print('Feature Selection')

sc_x = StandardScaler()

xtrain = sc_x.fit_transform(xtrain)

xtest = sc_x.transform(xtest)

print (xtrain[0:10, :])

classifier = LogisticRegression(random_state = 0)

classifier.fit(xtrain, ytrain)

print('Predicting testing data')

y_pred = classifier.predict(xtest)

print(y_pred)

X_set, y_set = xtest, ytest

X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1,

                     stop = X_set[:, 0].max() + 1, step = 0.01),

                np.arange(start = X_set[:, 1].min() - 1,

                     stop = X_set[:, 1].max() + 1, step = 0.01))
```

```
  plt.contourf(X1, X2, classifier.predict(
            np.array([X1.ravel(), X2.ravel()]).T).reshape(
            X1.shape), alpha = 0.75, cmap = ListedColormap(('red', 'green')))
  plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Classifier (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

Output

```
Feature Selection
[[ 0.58164944 -0.88670699]
 [-0.60673761  1.46173768]
 [-0.01254409 -0.5677824 ]
 [-0.60673761  1.89663484]
 [ 1.37390747 -1.40858358]
 [ 1.47293972  0.99784738]
 [ 0.08648817 -0.79972756]
 [-0.01254409 -0.24885782]
 [-0.21060859 -0.5677824 ]
 [-0.21060859 -0.19087153]]
Predicting testing data
[0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0
 0 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0
 0 0 1 0 1 1 1 1 0 0 1 1 0 1 0 0 0 1 0 0 0 0 0 0 1 1]
```



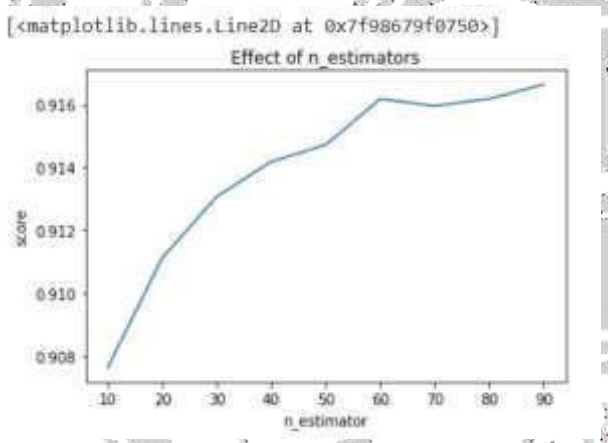
Program 38	Date:12/09/2022

Analyze the given dataset(gym_datUse the Iris dataset and visualize a decision tree with depth=4 and save the plot as a png fila.csv) using RandomForestRegressor and visualize the 'Effect of n_estimators.

```
importnumpy as np
import pandas as pd
from  sklearn.ensemble  import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
from subprocess import check_output
from datetime import time
def time_to_seconds(time):
   returntime.hour* 3600+time.minute* 60+time.second
df = pd.read_csv("gym_data.csv")
df = df.drop("date", axis=1)
noon = time_to_seconds(time(12, 0, 0))
df.timestamp = df.timestamp.apply(lambda t: abs(noon - t))
columns = ["day_of_week", "month", "hour"]
df = pd.get_dummies(df, columns=columns)
df.head(10)
data = df.values
X = data[:, 1:] # all rows, no label
y = data[:, 0] # all rows, label only
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test  =  scaler.transform(X_test)
```

```
model = RandomForestRegressor(n_jobs=-1)
estimators = np.arange(10, 100, 10)
scores = []
for n in estimators:
    model.set_params(n_estimators=n)
    model.fit(X_train, y_train)
    scores.append(model.score(X_test,  y_test))
plt.title("Effect of n_estimators")
plt.xlabel("n_estimator")
plt.ylabel("score")
plt.plot(estimators, scores)
```

Output



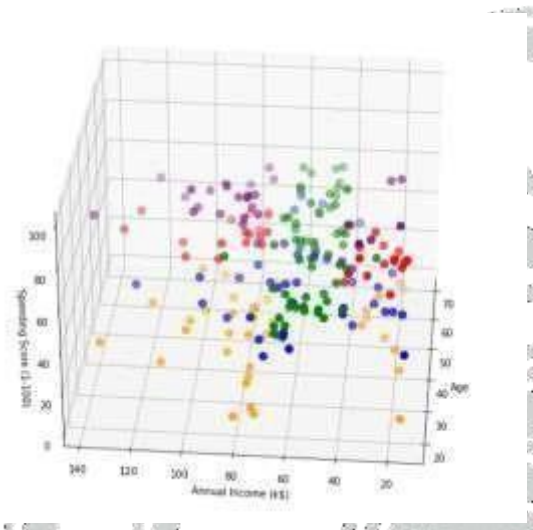
Program 39	Date:14/09/2022

Visualize a 3-Dimensional cluster using the given dataset where no_of_clusters=5. Use Mall_Customers.csv

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from mpl_toolkits.mplot3d import Axes3D

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

from sklearn.cluster import KMeans


df = pd.read_csv("Mall_Customers.csv")

km = KMeans(n_clusters=5)

clusters = km.fit_predict(df.iloc[:,4:])

df["label"] = clusters


fig = plt.figure(figsize=(20,10))

ax = fig.add_subplot(111, projection='3d')

ax.scatter(df.Age[df.label == 0], df["Annual Income (k$)"][df.label == 0], df["Spending
Score (1-100)"][df.label == 0], c='blue', s=60)

ax.scatter(df.Age[df.label == 1], df["Annual Income (k$)"][df.label == 1], df["Spending
Score (1-100)"][df.label == 1], c='red', s=60)

ax.scatter(df.Age[df.label == 2], df["Annual Income (k$)"][df.label == 2], df["Spending
Score (1-100)"][df.label == 2], c='green', s=60)

ax.scatter(df.Age[df.label == 3], df["Annual Income (k$)"][df.label == 3], df["Spending
Score (1-100)"][df.label == 3], c='orange', s=60)

ax.scatter(df.Age[df.label == 4], df["Annual Income (k$)"][df.label == 4], df["Spending
Score (1-100)"][df.label == 4], c='purple', s=60)

ax.view_init(30, 185)
```

```
plt.xlabel("Age")
plt.ylabel("Annual Income (k$)")
ax.set_zlabel('Spending Score (1-100)')
plt.show()
```

Output



Program 40	Date:14/09/2022

Using the dataset provided (Online Retail.xlsx),

a. **Split the data according to the region of the transaction.**
b. **Build the models using the apriori algorithm.**
c. **Develop the association rules.**
d. **Find the frequent items in each region.**

```
import numpy as np

import pandas as pd

from mlxtend.frequent_patterns import apriori, association_rules

data = pd.read_excel('Online Retail.xlsx')

data.head()

# Exploring the different regions of transactions

data.Country.unique()

data['Description'] = data['Description'].str.strip()


# Dropping the rows without any invoice number

data.dropna(axis = 0, subset =['InvoiceNo'], inplace = True)

data['InvoiceNo'] = data['InvoiceNo'].astype('str')


# Dropping all transactions which were done on credit

data = data[~data['InvoiceNo'].str.contains('C')]


basket_France = (data[data['Country'] =="France"].groupby(['InvoiceNo',
'Description'])['Quantity'].sum().unstack().reset_index().fillna(0).set_index('InvoiceNo'))


# Transactions done in the United Kingdom

basket_UK = (data[data['Country'] =="United Kingdom"].groupby(['InvoiceNo',
'Description'])['Quantity'].sum().unstack().reset_index().fillna(0).set_index('InvoiceNo'))


# Transactions done in Portugal

basket_Por = (data[data['Country'] =="Portugal"].groupby(['InvoiceNo',
'Description'])['Quantity'].sum().unstack().reset_index().fillna(0).set_index('InvoiceNo'))
```

```
basket_Sweden = (data[data['Country'] =="Sweden"].groupby(['InvoiceNo',
'Description'])['Quantity'].sum().unstack().reset_index().fillna(0).set_index('InvoiceNo'))

# Defining the hot encoding function to make the data suitable

# for the concerned libraries

def hot_encode(x):

        if(x<= 0):

                return 0

        if(x>= 1):

                return 1


# Encoding the datasets

basket_encoded = basket_France.applymap(hot_encode)

basket_France = basket_encoded


basket_encoded = basket_UK.applymap(hot_encode)

basket_UK = basket_encoded


basket_encoded = basket_Por.applymap(hot_encode)

basket_Por = basket_encoded


basket_encoded = basket_Sweden.applymap(hot_encode)

basket_Sweden = basket_encoded


frq_items = apriori(basket_France, min_support = 0.05, use_colnames = True)


# Collecting the inferred rules in a dataframe
```

```
rules = association_rules(frq_items, metric ="lift", min_threshold = 1)

rules = rules.sort_values(['confidence', 'lift'], ascending =[False, False])

print(rules.head())
```

Output

```
                                        antecedents  \
44                          (JUMBO BAG WOODLAND ANIMALS)
260  (PLASTERS IN TIN CIRCUS PARADE, RED TOADSTOOL ...
272  (RED TOADSTOOL LED NIGHT LIGHT, PLASTERS IN TI...
302  (SET/20 RED RETROSPOT PAPER NAPKINS, SET/6 RED...
300  (SET/6 RED SPOTTY PAPER PLATES, SET/20 RED RET...

                        consequents  antecedent support  consequent support  \
44                        (POSTAGE)            0.076531            0.765306
260                       (POSTAGE)            0.051020            0.765306
272                       (POSTAGE)            0.053571            0.765306
302  (SET/6 RED SPOTTY PAPER PLATES)            0.102041            0.127551
300    (SET/6 RED SPOTTY PAPER CUPS)            0.102041            0.137755

      support  confidence      lift  leverage  conviction
44   0.076531       1.000  1.306667  0.017961         inf
260  0.051020       1.000  1.306667  0.011974         inf
272  0.053571       1.000  1.306667  0.012573         inf
302  0.099490       0.975  7.644000  0.086474   34.897959
300  0.099490       0.975  7.077778  0.085433   34.489796
```