

**M.Sc. (Five Year Integrated) in Computer Science
(Artificial Intelligence & Data Science)**

First Semester

Laboratory Record

21-805-0106: PYTHON PROGRAMMING LAB

*Submitted in partial fulfillment
of the requirements for the award of degree in
Master of Science (Five Year Integrated)
in Computer Science (Artificial Intelligence & Data Science) of
Cochin University of Science and Technology (CUSAT)
Kochi*



Submitted by

**OMAL S.
(80521015)**

**DEPARTMENT OF COMPUTER SCIENCE
COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY (CUSAT)
KOCHI-682022**

MARCH 2022

DEPARTMENT OF COMPUTER SCIENCE
COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY (CUSAT)
KOCHI, KERALA-682022



*This is to certify that the practical laboratory record for **21-805-0106: Python Programming Lab** is a record of work carried out by **OMAL S.(80521015)**, in partial fulfillment of the requirements for the award of degree in **Master of Science (Five Year Integrated)** in **Computer Science (Artificial Intelligence & Data Science)** of Cochin University of Science and Technology (CUSAT), Kochi. The lab record has been approved as it satisfies the academic requirements in respect of the first semester laboratory prescribed for the Master of Science (Five Year Integrated) in Computer Science degree.*

Faculty Member in-charge

Dr. Shailesh S.
Assistant Professor
Department of Computer Science
CUSAT

Dr. Philip Samuel
Professor and Head
Department of Computer Science
CUSAT

Sl. No.	Program	Page No.
1.	Arithmetic Operations on Four Digit Numbers	1
2.	Triangle Area and Contribution	3
3.	Employee Payslip Generation	6
4.	Check for Happy Number	9
5.	String Operations	12
6.	Pairs of Rabbits in 'N' Months	16
7.	Operations on List of Integers	19
8.	Iris'.json' File Handling	22
9.	Box Class for Shapes	26
10.	3D_Shapes Inheritance	31
11.	Tic Tac Toe	33
12.	Principal Component Analysis	41
13.	Data Visualisation	44
14.	Vehicle Details	50
15.	Tkinter UI Application	57

ARITHMETIC OPERATION ON FOUR DIGIT NUMBER

AIM

Develop a program to read a four-digit number and find its

- Sum of digits
- Reverse
- Difference between the product of digits at the odd position and the product of digits at the even position.

THEORY

- input () - 'input()' function is used to take user input. By default, it returns the user input in form of a string.
- Strings - String is a sequence of characters
- Arithmetic operators - Arithmetic operators are used to perform mathematical operations like addition, subtraction, multiplication and division.

PROGRAM

```
def arithmetic_operations():
    num=int(input("Enter the four digit number:"))
    t=num
    sum=0
    reverse=0                                #reverse to store the reverse number

    reminder1=num%10                        #reminder1- store the last digit
    reverse=(reverse*10)+reminder1
    num=num//10

    reminder2=num%10                        #reminder2- store the last second digit
    reverse=(reverse*10)+reminder2
    num=num//10

    reminder3=num%10                        #reminder3- store the second digit
    reverse=(reverse*10)+reminder3
    num=num//10

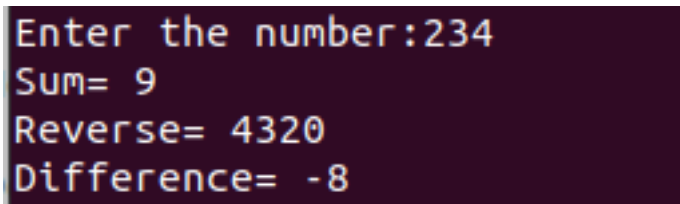
    reminder4=num%10                        #reminder4- store the first digit
    reverse=(reverse*10)+reminder4
```

```
num=num//10

sum=remainder4+remainder3+remainder2+remainder1
difference=(remainder2*remainder4)-(remainder1*remainder3)
print("Sum=",sum)                #print the sum of the digits
print("Reverse = ",end="")      #print the reverse of the input number
if(t%10==0):
    while(t!=0):
        if(t%10==0 and t//10):
            print("0",end="")
        t=t//10
    print(reverse)

print("Difference=",difference)
                                #print the difference between the products
arithmetic_operations()
```

SAMPLE INPUT-OUTPUT



```
Enter the number:234
Sum= 9
Reverse= 4320
Difference= -8
```

TEST CASES

Test case No.	Description	Input	Expected output	Actual output	Result
1	Check for the sum output	1234	10	10	Pass
2	Check for the Reverse output	1234	4321	4321	Pass
3	Check for the Difference of odd and even numbers output	1234	-5	-5	Pass
4	Check output for numbers with zeros	1000	1 0001 0	1 0001 0	Pass
5	Wrong input	abc	error	error	Pass

RESULT

Program executed Successfully and the output is obtained.

GIT LINK

<https://github.com/omals/Python-Lab/tree/main/LabCycle/LabCycle1/1Question>

TRIANGLE AREA AND CONTRIBUTION

AIM

Develop a program to read the three sides of two triangles and calculate the area of both. Define a function to read the three sides and call it. Also, define a function to calculate the area. Print the total area enclosed by both triangles and each triangle's contribution (%) towards it.

$$A = \sqrt{s(s-a)(s-b)(s-c)} \text{ and } s = \frac{a+b+c}{2}$$

THEORY

- Datatype - Data types are the classification or categorization of data items. It represents the kind of value that tells what operations can be performed on a particular data.
- Functions - A function is a block of related statements that performs a specific task which only runs when it is called.
- Expressions - An expression is a combination of operators and operands that is interpreted to produce some other value.
- Built-in functions - There are several Built-in functions that are pre-defined in the programming language's library and are readily available for use.

PROGRAM

```
#function to input the side
def side():
    side=int(input("Enter the side:"))
    return (side)

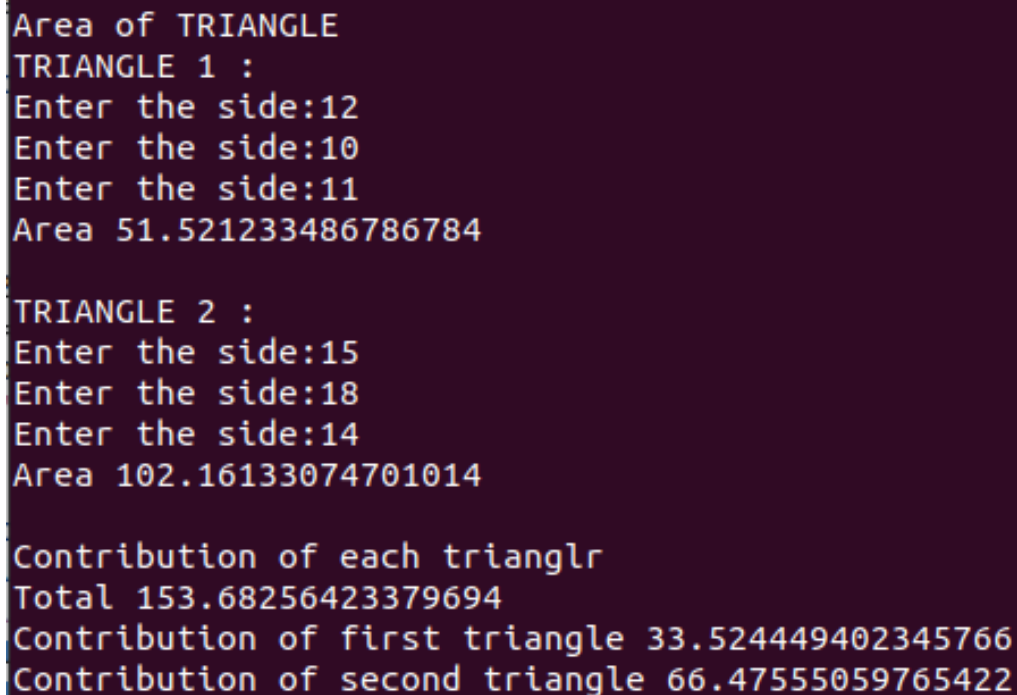
#function to calculate area
def area():
    s1=side()                #s1,s2,s3- three sides of triangle
    s2=side()
    s3=side()
    semip=(s1+s2+s3)*0.5     #semip=semiperimeter of teh triangle
    calculation=semip*(semip-s1)*(semip-s2)*(semip-s3)
    a=calculation**0.5
    print("Area",a)
    return a
```

```
#function to calculate the contribution of each triangle
def contribution(a1,a2):          #a1- area of first triangle
    print("Total",(a1+a2))        #a2- area of second triangle
    con1=(a1/(a1+a2))*100         #con1- contribution of first triangle
    con2=(a2/(a1+a2))*100         #con2- contribution of second triangle
    print("Contribution of first triangle",con1)
    print("Contribution of second triangle",con2,"\n")

#function to be invoked
def main():
    print("\nArea of TRIANGLE")
    print("TRIANGLE 1 : ")
    area1=area()
    print("\nTRIANGLE 2 : ")
    area2=area()
    print("\nContribution of each triangle")
    contribution(area1,area2)

main()                                #function invocation
```

SAMPLE INPUT-OUTPUT



```
Area of TRIANGLE
TRIANGLE 1 :
Enter the side:12
Enter the side:10
Enter the side:11
Area 51.521233486786784

TRIANGLE 2 :
Enter the side:15
Enter the side:18
Enter the side:14
Area 102.16133074701014

Contribution of each triangle
Total 153.68256423379694
Contribution of first triangle 33.524449402345766
Contribution of second triangle 66.47555059765422
```

TEST CASES

Test Case No.	Description	input	Expected output	Actual output	Result
1	Check for area of two triangle	3,4,5	6.0	6.0	Pass
		12,13,5	36.0	36.0	
2	Check for contribution of each triangle	3,4,5	16.66666	16.66666	Pass
		12,13,5	83.33333	83.33333	
3	Wrong input	at	error	error	Pass

RESULT

Program executed Successfully and the output is obtained.

GIT LINK

<https://github.com/omals/Python-Lab/tree/main/LabCycle/LabCycle1/2Question>

EMPLOYEE PAYSIP GENERATION

AIM

Develop a program to read the employee's name, code, and basic pay and calculate the gross salary, deduction, and net salary according to the following conditions. Define a function to find each of the components. Finally, generate a payslip.

Basic Pay (BP)	DA (%)	HRA (%)	MA	PT	PF (%)	IT (%)
<10000	5	2.5	500	20	8	-
<30000		5	2500	60	8	-
<50000	11	7.5	5000	60	11	11
else	25	11	7000	80	12	20

Gross Salary (GS) : $BP + DA + HRA + MA$

Deduction (D): $PT + PF + IT$

Net Salary = $GS - D$

THEORY

- Conditional Branching - A programming instruction that directs the computer to another part of the program based on the results of a comparison

PROGRAM

```
#function to include the details of the employee

def detail():
    name=str(input("Enter the name \t : "))
    code=int(input("Enter the code \t : "))
    basic_pay=float(input("Enter the basic pay:"))
    return (name,code,basic_pay)

#function to calculate the Gross Salary
def gross_salary(bp,da,hra,ma):
    gs=bp+da+hra+ma                                #gs- Gross Salary value
    return gs

#function to calculate Deduction

def deduction(pt,pf,it):
    print("Professional Tax          :",pt)
```

```
print("Provident Fund          :",pf,"%")
print("Income Tax             :",it,"%")
d=pt+pf+it                                #d- Deduction value
print("Deduction              =  ",d)
return d

#function to calculate the Net Salary
def net_salary(BP,DA,HRA,MA,PT,PF,IT):
    print("Dearness Allowance    :",DA,"%")
    print("House Rent Allowance   :",HRA,"%")
    print("Medical Allowance       :",MA)
    #invoke the function gross_salary and return value to GS
    GS=gross_salary(BP,DA,HRA,MA)
    # #invoke the function deduction and return value to D
    D=deduction(PT,PF,IT)
    print("Gross salary            =  ",GS)
    ns= GS-D                                #ns- store value of Net Salary
    print("Net Salary              =  ",ns)

#function to display the input details of employee
def display(nam,cod,basicpay):              #nam-name    cod-code
    print("Name of the employee \t :",nam)
    print("Code of the employee \t :",cod)
    print("Basic pay of the employee:",basicpay)

#function from which all other functions are invoked
def main():
    print("Enter the details of the employee.")
    n,c,bp=detail()                          #n-name    c-code
    print("\tPay list")                       # bp-Basic Pay of the employee
    display(n,c,bp)                          #invoke Display function
    if(bp<10000):
        net_salary(bp,5,2.5,500,20,8,0)      #invoke net_salary function
    elif(bp<30000 and bp>=10000):
        net_salary(bp,7.5,5,2500,60,8,0)
    elif(bp<50000 and bp>=30000):
        net_salary(bp,11,7.5,5000,60,11,11)
    else:
        net_salary(bp,25,11,7000,80,12,20)

main()                                       #invoke the main function
```

SAMPLE INPUT-OUTPUT

```

Enter the details of the employee.
Enter the name      : Manju
Enter the code      : 2032
Enter the basic pay:25000
      Pay list
Name of the employee      : Manju
Code of the employee      : 2032
Basic pay of the employee: 25000.0
Dearness Allowance        : 7.5 %
House Rent Allowance      : 5 %
Medical Allowance         : 2500
Professional Tax           : 60
Provident Fund             : 8 %
Income Tax                : 0 %
Deduction                 = 68
Gross salary              = 27512.5
Net Salary                = 27444.5

```

TEST CASES

Test Cases No.	Description	Input	Expected output	Actual output	result
1	check for output if the basic pay less than '10000'	8500	Dearness Allowance : 5 % House Rent Allowance : 2.5 % Medical Allowance : 500 Professional Tax : 20 Provident Fund : 8% Income Tax : 0% Deduction = 28 Gross salary = 9007.5 Net Salary = 8979.5	Dearness Allowance : 5 % House Rent Allowance : 2.5 % Medical Allowance : 500 Professional Tax : 20 Provident Fund : 8% Income Tax : 0% Deduction = 28 Gross salary = 9007.5 Net Salary = 8979.5	Pass
2.	Check for output if the basic pay less than '30000'	25000	Dearness Allowance : 7.5 % House Rent Allowance : 5 % Medical Allowance : 2500 Professional Tax : 60 Provident Fund : 8% Income Tax : 0% Deduction = 68 Gross salary = 27512.5 Net Salary = 27444.5	Dearness Allowance : 7.5 % House Rent Allowance : 5 % Medical Allowance : 2500 Professional Tax : 60 Provident Fund : 8% Income Tax : 0% Deduction = 68 Gross salary = 27512.5 Net Salary = 27444.5	Pass
3.	Check for output if the basic pay less than '50000'	45000	Dearness Allowance : 11 % House Rent Allowance : 7.5 % Medical Allowance : 5000 Professional Tax : 60 Provident Fund : 11 % Income Tax : 11 % Deduction = 82 Gross salary = 50018.5 Net Salary = 49936.5	Dearness Allowance : 11 % House Rent Allowance : 7.5 % Medical Allowance : 5000 Professional Tax : 60 Provident Fund : 11 % Income Tax : 11 % Deduction = 82 Gross salary = 50018.5 Net Salary = 49936.5	Pass
4.	Check for output if the basic pay greater the '50000'	60000	Dearness Allowance : 25 % House Rent Allowance : 11 % Medical Allowance : 7000 Professional Tax : 80 Provident Fund : 12 % Income Tax : 20 % Deduction = 112 Gross salary = 67036.0 Net Salary = 66924.0	Dearness Allowance : 25 % House Rent Allowance : 11 % Medical Allowance : 7000 Professional Tax : 80 Provident Fund : 12 % Income Tax : 20 % Deduction = 112 Gross salary = 67036.0 Net Salary = 66924.0	pass
5.	Check for wrong input	abf	error	error	pass

RESULT

Program executed Successfully and the output is obtained.

GIT LINK

<https://github.com/omals/Python-Lab/tree/main/LabCycle/LabCycle1/3Question>

HAPPY NUMBER

AIM

Develop a program to perform the following task:

- Define a function to check whether a number is happy or not.
- Define a function to print all happy numbers within a range
- Define a function to print first N happy numbers

A happy number is a number defined by the following process:

- Starting with any positive integer, replace the number with the sum of the squares of its digits.
- Repeat the process until the number equals 1 (where it will stay), or it loops endlessly in a cycle which does not include 1.
- Those numbers for which this process ends in 1 are happy.

THEORY

- Loops – for, while
for loop - A for loop is used for iterating over a sequence
while loop - While loop is used to execute a block of statements repeatedly until a given condition is satisfied. And when the condition becomes false, the line immediately after the loop in the program is executed.
- Nested loops - A nested loop is a loop inside the body of the outer loop.

PROGRAM

```
#function to check whether happy or sad
def happy(n):
    for i in range(1,101):
        sum=0
        while n!=0:
            digit=n%10                #digit-each extracted digits stored
            square=digit**2
            n=n//10
            sum=sum+square
        n=sum
    return (sum)
```

```
#function to print the happy number within range
def happynumber(l,u):          #l-lowerlimit  u-upperlimit
    count=0
    for i in range(l+1,u):
        s=happy(i)            #invoke happy() function
        if s==1:
            print(i,end=" ")
            count=1           #count to check whether there
                                #is happy number in the range
    print()
    if count==0:
        print("There is no happy numbers between this range.")

#Function to print 'n' number of happy numbers
def printnumbers(n):
    count=1                    #count- to check the number of happy
                                #numbers printed
    i=1
    while count<=n:
        total=happy(i)         #invoke happy function
        if total==1:           #total- condition variable
            print(i,end=" ")
            count+=1
        i=i+1
    print('')

#function to invoke all above mentioned functions

def main():
    print("\t HAPPY NUMBER FUNCTIONS")
    print("\tA number is Happy or Sad")
    num=int(input("Enter the number to check:"))

    if happy(num)==1:
        print("Its a Happy number.")
    else:
        print("Its a Sad number.")

    print("\tHappy numbers within a range.")

    lowerlimit=int(input("Enter the Lower limit:"))
```

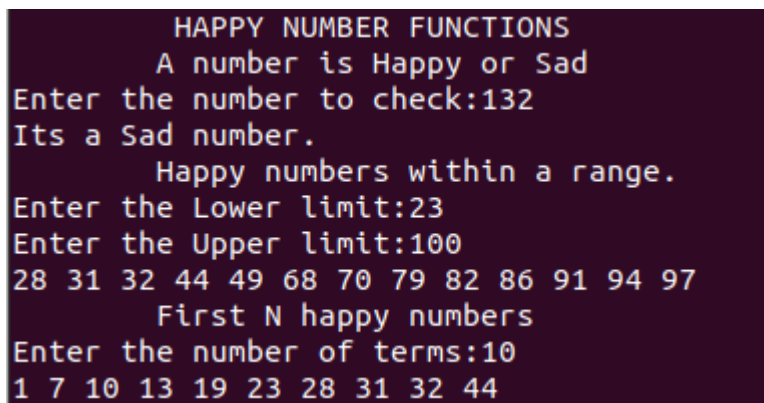
```
upperlimit=int(input("Enter the Upper limit:"))
happynumber(lowerlimit,upperlimit)

print("\tFirst N happy numbers")
N=int(input("Enter the number of terms:"))

printnumbers(N)                                #N-number of terms to print

main()                                          #function invokation
```

SAMPLE INPUT-OUTPUT



```
HAPPY NUMBER FUNCTIONS
A number is Happy or Sad
Enter the number to check:132
Its a Sad number.
Happy numbers within a range.
Enter the Lower limit:23
Enter the Upper limit:100
28 31 32 44 49 68 70 79 82 86 91 94 97
First N happy numbers
Enter the number of terms:10
1 7 10 13 19 23 28 31 32 44
```

TEST CASES

Test Cases No.	Description	Input	Expected output	Actual output	result
1	Check whether output Sad	45	Sad number	Sad number	Pass
2	Print happy numbers within the range	19 23	No happy numbers	No happy numbers	Pass
3	Print N terms of happy numbers	0	No output	No output	Pass
4	Check whether output Happy	100	Happy number	Happy number	Pass
5	Print happy numbers within the range	30 45	31 32 44	31 32 44	Pass
6	Print N terms of happy numbers	20	1 7 10 13 19 23 28 31 32 44 4 9 68 70 79 82 86 91 94 97 100	1 7 10 13 19 23 28 31 32 44 4 9 68 70 79 82 86 91 94 97 100	Pass
7	Wrong input	ad	error	error	pass

RESULT

Program executed Successfully and the output is obtained.

GIT LINK

<https://github.com/omals/Python-Lab/tree/main/LabCycle/LabCycle1/4Question>

STRING OPERATIONS

AIM

Develop a program to read a string and perform the following operations:

- Print all possible substring
- Print all possible substrings of length K
- Print all possible substrings of length K with N distinct characters
- Print all palindrome substrings

THEORY

- Strings - Strings are arrays of bytes representing Unicode characters.
- String functions - String functions are built-in functions that can be called by string objects to perform various actions.
- Slicing - Slicing in Python is a feature that enables accessing parts of sequences like strings, tuples, and lists.

PROGRAM

```
#Throughout the program 's' is the string
#k-length of substring
#n-number of distinct characters

#function to generate substring
def substring(s):
    print("\nSubstrings '",s,"'")
    for i in range(len(s)+1):
        for j in range(i+1,len(s)+1):
            print(s[i:j])
    print("\t")

#function to generate substring of particular length
def lengthsubstring(s,k):
    print("\nSubstrings of length :",k,"\n\t",end="")
    for i in range(len(s)+1):
        for j in range(len(s)+1):
            if len(s[i:j])==k:
                print(s[i:j],end=" ")
```

```
print()

#function to determine the substring with distinct characters
def substringdistinct(s,k,n):
    print("\nSubstrings of length ",k," with ",n,end="")
    print(" distinct characters\n\t",end="")
    count=0
    for i in range(len(s)+1):
        for j in range(len(s)+1):
            if len(s[i:j])==k:
                sets=set(s[i:j])                #sets-set string
                if len(sets)==n:
                    print(s[i:j],end=" ")
                    count=1
    if count==0:
        print("There no substrings with ",n," distinct characters in ",end=" ")
        print(k,"length substring")
    print()

#function to find the maximum length substring with n distinct characters
def substringmaxlength(s,n):
    count=0
    temp2=0
    string=[]
    for i in range(len(s)+1):
        for j in range(i+1,len(s)+1):
            if len(set(s[i:j]))==n:
                string.append(s[i:j])
    for i in range(len(string)):
        if(len(string[i])>len(string[-1])):
            print(string[i],end=" ")
            count=1
    if count==0:
        print("There no substrings with ",n," distinct characters ",end="")
        print("in length substring")
    print()

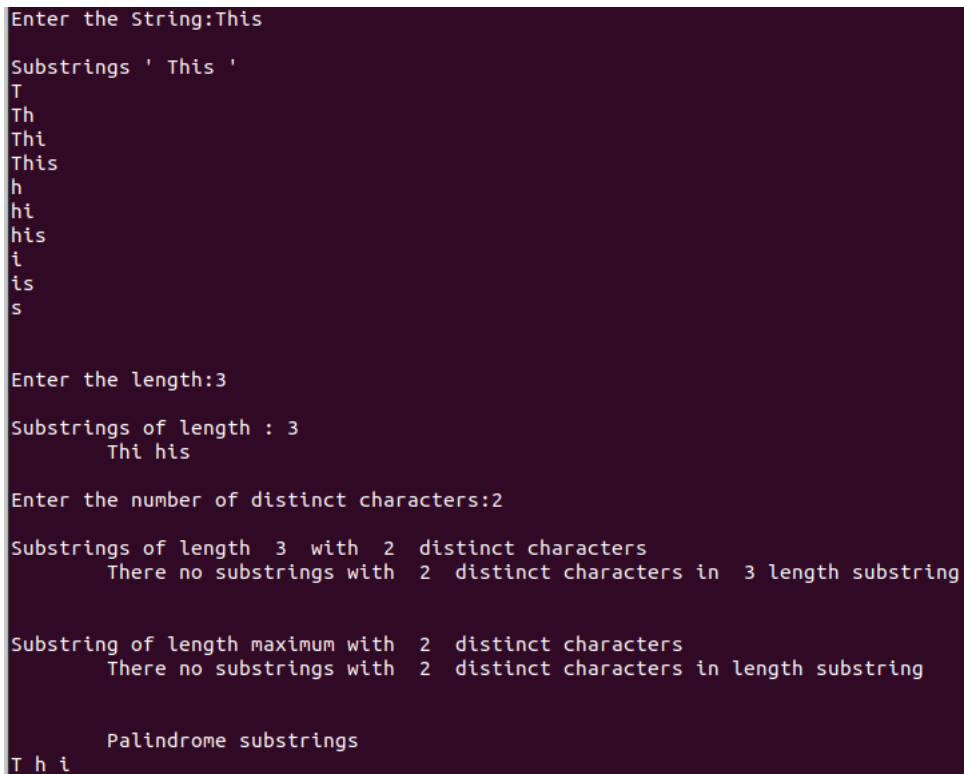
#function to print for palindrome of the string
def palindrome(s):
    print("\n\tPalindrome substrings")
    for i in range(len(s)+1):
```



```
        for j in range(i+1,len(s)):
            sub=s[i:j]                #sub- sub string
            subin=sub[::-1]           #subin- sub string inverse
            if sub==subin:
                print(subin,end=" ")
        print()

#function where main calling for all function happenning
def main():
    string=str(input("Enter the String:"))
    substring(string)
    length=int(input("\nEnter the length:"))
    lengthsubstring(string,length)
        #terms-to store the numbbber of distinct characters
    terms=int(input("\nEnter the number of distinct characters:"))
    substringdistinct(string,length,terms)
    print("\nSubstring of length maximum with ",terms,end="")
    print(" distinct characters\n\t",end="")
    substringmaxlength(string,terms)
    palindrome(string)
main()                #functions invokation
```

SAMPLE INPUT-OUTPUT



```
Enter the String:This
Substrings ' This '
T
Th
Thi
This
h
hi
his
i
is
s

Enter the length:3
Substrings of length : 3
    Thi his

Enter the number of distinct characters:2
Substrings of length 3 with 2 distinct characters
    There no substrings with 2 distinct characters in 3 length substring

Substring of length maximum with 2 distinct characters
    There no substrings with 2 distinct characters in length substring

    Palindrome substrings
T h i
```

TEST CASES

Test Cases No.	Descripton	Input	Expected output	Actual output	result
1	Print sub strings	abaca	a ab aba abaca b ba bac baca a ac aca c ca a	a ab aba abaca b ba bac baca a ac aca c ca a	Pass
2	Print Sub Strings with length k	3	aba bac aca	aba bac aca	Pass
3	Print substring of length k with n distinct characters	2	aba aca	aba aca	Pass
4	Print substring of maximum length with n distinct characters	2	aba aca	aba aca	Pass
5	Palindromic sub strings	abaca	a aba b a aca c a	a aba b a aca c a	Pass
6	Input for numbers Print substrings	12134	1 12 121 1213 12134 2 21 213 2134 1 13 134 3 34 4	1 12 121 1213 12134 2 21 213 2134 1 13 134 3 34 4	pass
7	Print Sub Strings with length k	3	121 213 134	121 213 134	Pass
8	Print substring of length k with n distinct characters	1	There no substrings with 1 distinct characters in 3 length substring	There no substrings with 1 distinct characters in 3 length substring	Pass
9	Print palidromic sub strings	12134	1 121 2 1 3 4	1 121 2 1 3 4	Pass

RESULT

Program executed Successfully and the output is obtained.

GIT LINK

<https://github.com/omals/Python-Lab/tree/main/LabCycle/LabCycle1/5Question>

PAIR OF RABBITS IN 'N' MONTHS

AIM

Suppose a newly born pair of rabbits, one male and one female, are put in a field. Rabbits can mate at the age of one month so that at the end of its second month, a female has produced another pair of rabbits. Suppose that our rabbits never die and that the female always produces one new pair every month from the second month.

Develop a program to show a table containing the number of pairs of rabbits in the first N months.

THEORY

- Critical thinking - Critical thinking involves approaching a problem or situation analytically and breaking it into separate components for more efficient problem-solving.
- Loops - A loop is a sequence of instructions that is continually repeated until a certain condition is reached.
- formatted io - Formatted I/O functions are used to take various inputs from the user and display multiple outputs to the user.

PROGRAM

```
#function to calculate the number of rabbits

def calculate(a):
    n1=1                      #a-months entered by the user
    n2=1

    if a==1:
        display(1,n1)
        return n1
    elif a==2:
        display(1,n1)
        display(2,n2)
        return n2

    else:
        if a!=0:
```

```
    display(1,n1)
    display(2,n2)

    for i in range(3,a+1):
        n3=n1+n2
        display(i,n3)
        n1=n2
        n2=n3
    return n3

else:
    print("\nThere is no pair of rabbits are put into the field\n")
    return 0;

#function to display the values in table format

def display(j,n3):
    print("_"*50)
    print ("\t",j,"\t|\t",n3)

#function take input and give output

def main():
    a=int(input("Enter the months: "))

    print("\n",end='')
    print("-"*50)
    print("\tTABLE OF RABBIT PAIRS")
    print("-"*50)
    print("\tMONTH \t| Number of pair of Rabbits")

    n=calculate(a)          # n- number of pairs at the end of a months

    print("-"*50)
    print("\nTotal number of Rabbit Pairs at the end of ",end="")
    print(a," months is: ",n,"\n")

main()                      #main function to begin the execution
```

SAMPLE INPUT-OUTPUT

```
Enter the months: 20
-----
      TABLE OF RABBIT PAIRS
-----
MONTH | Number of pair of Rabbits
-----
    1 |          1
    2 |          1
    3 |          2
    4 |          3
    5 |          5
    6 |          8
    7 |         13
    8 |         21
    9 |         34
   10 |         55
   11 |         89
   12 |        144
   13 |       233
   14 |       377
   15 |      610
   16 |     987
   17 |    1597
   18 |   2584
   19 |  4181
   20 |  6765
-----
Total number of Rabbit Pairs at the end of  20 months is: 6765
```

TEST CASES

Test Cases	Description	input	Expected output	Actual output	Result
1	check the output	20	list of pairs for 20 months	list of pairs for 20 months	pass
2	check the formatted table	15	Table lines	Table lines	pass
3	check for wrong input	-1	error	error	pass
4	check for wrong input	a	error	error	pass

RESULT

Program executed Successfully and the output is obtained.

GIT LINK

<https://github.com/omals/Python-Lab/tree/main/LabCycle/LabCycle2/1Question>

OPERATIONS ON LIST OF INTEGERS

AIM

Write a program to read a string containing numbers separated by a space and convert it as a list of integers. Perform the following operations on it.

1. Rotate elements in a list by 'k' position to the right
2. Convert the list into a tuple using list comprehension
3. Remove all duplicates from the tuple and convert them into a list again.
4. Create another list by putting the results of the evaluation of the function $f(x) = x^2 - x$ with each element in the final list
5. After sorting them individually, merge the two lists to create a single sorted list.

THEORY

- List - A list in Python is used to store the sequence of various types of data. It is created by placing elements inside square brackets `[]`, separated by commas.
- tuple - A Tuple is a collection of Python objects separated by commas. They are used to store multiple items in a single variable.
- set - A Set is an unordered collection data type that is iterable, mutable and has no duplicate elements
- list comprehension - List comprehension offers a shorter syntax when you want to create a new list based on the values of an existing list.

PROGRAM

```
#function to merge and sort the list
def SortMerge(temp1,temp2):
    listfinal=temp1+temp2
    listfinal.sort()
    return listfinal
#function to sort the individual list
def final_sort(l1,l2):
    l1.sort()
    l2.sort()
    final_list=SortMerge(l1,l2)
    print("5. Single Sorted list          :",final_list)
    print("_"*70,"\n")
```

```
#function to evaluate the mathematical function
def function(l):
    l_function=[]          # l_function - list of result of the expression
    for i in range(0,len(l)):      #f(x)=x{2}-x
        l_function.append(l[i]**2 -l[i])
    print("4. Values of functions      :",l_function)
    final_sort(l,l_function)

#function to rewrite the list without duplication
def duplication(l):
    l_ndupli=tuple(set(l))
    l_ndupli=list(l_ndupli)
    print("3. List without duplication  :",l_ndupli)
    function(l_ndupli)

#function to convert the list to tuple
def list_tuple(l):
    l_tuple=tuple(i for i in l)
    print("2.Tuple by list Comprehension :",l_tuple)
    duplication(l_tuple)

#function to rotate the list
def rotate(l):
    n= int(input("\n1. Enter Number of rotation: "))
    temp=[]
    if n>len(l):          #Condition to avoid repeated looping
        n = int(n%len(l))
    for i in range(len(l)-n,len(l)):
        #print(len(l),n,i)
        temp.append(l[i])
    for i in range(0,len(l)-n):
        temp.append(l[i])
    print("\tRotated List          :",temp)

#main function to get the list
def main():
    l_string=[]
    print("\n\tList of String to list of Integers and operations.")
    print("_"*70)
    l_string=input("Enter Numbers separated by space : ")
    l_string=list(l_string.split(" "))    #l_string - list of string
    print("\nList of string      :",l_string)
    l_int=[]                          #l_int - list of integers
    for i in l_string:
        l_int.append(int (i))
```

```

    print("List of Integers :",l_int)
    print("_"*70)
    rotate(l_int)                                #function call
    list_tuple(l_int)                            #function call
main()                                           #program execution beginning

```

SAMPLE INPUT-OUTPUT

```

List of String to list of Integers and operations.
-----
Enter Numbers separated by space : 23 4 5 6 234 4 3 1

List of string   : ['23', '4', '5', '6', '234', '4', '3', '1']
List of Integers : [23, 4, 5, 6, 234, 4, 3, 1]
-----

1. Enter Number of rotation: 5
   Rotated List           : [6, 234, 4, 3, 1, 23, 4, 5]
2. Tuple by list Comprehension : (23, 4, 5, 6, 234, 4, 3, 1)
3. List without duplication  : [1, 3, 4, 5, 6, 234, 23]
4. Values of functions       : [0, 6, 12, 20, 30, 54522, 506]
5. Single Sorted list        : [0, 1, 3, 4, 5, 6, 6, 12, 20, 23, 30, 234, 506, 54522]
-----

```

TEST CASES

Test Cases	Description	input	Expected output	Actual output	Result
1	check output from list of number input	23 4 5 6 234 4 3 1	list of string	list of string	pass
2		23 4 5 6 234 4 3 1	list of integers	list of integers	pass
3	check output of rotation	5	rotated list	rotated list	pass
4	Check value of function	$f(x)=x^2-x$	list of integers	list of integers	pass
5	check for Sorted list	23 4 5 6 234 4 3 1 0 6 12 20 30 54522 506	sorted list	sorted list	pass
6	check string input	e w r r	error	error	pass

RESULT

Program executed Successfully and the output is obtained.

GIT LINK

<https://github.com/omals/Python-Lab/tree/main/LabCycle/LabCycle2/2Question>

IRIS.JSON FILE HANDLING

AIM

Read the file 'iris.json' as a text file :

1. Create a list having each line of the file as an element
2. Convert it into a list of dictionary objects.
3. Show the details of all flowers whose species is "setosa".
4. Print the minimum petal area and max sepal area in each species
5. Sort the list of dictionaries according to the total area are sepal and petal.

THEORY

- JSON - JavaScript Object Notation (JSON) is a standard text-based format for representing structured data based on JavaScript object syntax. It is commonly used for transmitting data in web applications.
- dictionary - A dictionary is a collection which is ordered, changeable and do not allow duplicates. Dictionaries are used to store data values in key:value pairs.

PROGRAM

```
import json
def readAsList(filepath):
    fp = open(filepath,'r')    #list having each line of json as elements
    jsonList = fp.readlines()
    fp.close()
    return jsonList
def readAsListOfDict(filepath):
    fp = open(filepath,'r')    #list having dictionary of objects.
    jsonData = json.load(fp)
    fp.close()
    return jsonData
def printSetosa(jsonList):    #printing the details of only setosa.
    print("\nDetails of flowers of species setosa")
    for i in jsonList:
        if(i['species']=='setosa'):
            print(i)
def sepalAreaAndPetalArea(jsonList):    #list to store species names.
    listOfSpeciesName = list()
```

```
                                #appending the different species name to the list.
for i in jsonList:
    listOfSpeciesName.append(i['species'])
                                #removing duplicates to get unique speices.
listOfSpeciesName = list(set(listOfSpeciesName))
                                #list to store sepal and petal area.

sepalArea = list()
petalArea = list()

                                #species sepal area and petal area
for i in listOfSpeciesName:
    for j in jsonList:
        if(j['species']==i):
            sepalArea.append(j['sepalLength']*j['sepalWidth'])
            petalArea.append(j['petalLength']*j['petalWidth'])
    print()
    print(i.capitalize())

                                #printing minimum and maximum areas.
    print("Maximum Sepal Area in ",i.capitalize()," is ",end="")
    print(round(max(sepalArea),2))
    print("Minimum Petal Area in ",i.capitalize()," is ",end="")
    print(round(min(petalArea),2))
    sepalArea.clear()
    petalArea.clear()

def sortTotalArea(jsonList):
    for i in jsonList:
        #adding total area to the each dictionary
        petal=i['petalLength']*i['petalWidth']
        sepal=i['sepalLength']*i['sepalWidth']
        totalArea = (petal)+(sepal)
        i.update({'totalArea':round(totalArea,2)})

        #list sorted according to total area
    sortedList = sorted(jsonList,key=lambda i:i['totalArea'])
    print("\nList sorted on the basis of total area")
    for i in sortedList:
        print(i)

filePath = 'iris.json'
#saving the file
jsonList = readAsList(filePath)
print("List with each line as element\n")
for line in jsonList:
    print(line)

jsonData = readAsListOfDict(filePath)
print("\nList of Dictionaries")
```

```
for i in jsonData:
    for key, values in i.items():
        print(key.capitalize()+" : ",values,end=" , ")
    print()
printSetosa(jsonData)          #function call for displaying setosa series
                               #function call for find total sepal and petal area
sepalAreaAndPetalArea(jsonData)
                               #function to find the total area
sortTotalArea(jsonData)
```

SAMPLE INPUT-OUTPUT

```
List with each line as element
[
{"sepalLength": 5.1, "sepalWidth": 3.5, "petalLength": 1.4, "petalWidth": 0.2, "species": "setosa"},
{"sepalLength": 4.9, "sepalWidth": 3.0, "petalLength": 1.4, "petalWidth": 0.2, "species": "setosa"},
{"sepalLength": 4.7, "sepalWidth": 3.2, "petalLength": 1.3, "petalWidth": 0.2, "species": "setosa"},
{"sepalLength": 4.6, "sepalWidth": 3.1, "petalLength": 1.5, "petalWidth": 0.2, "species": "setosa"},
{"sepalLength": 5.0, "sepalWidth": 3.6, "petalLength": 1.4, "petalWidth": 0.2, "species": "setosa"},
{"sepalLength": 5.4, "sepalWidth": 3.9, "petalLength": 1.7, "petalWidth": 0.4, "species": "setosa"},
{"sepalLength": 4.6, "sepalWidth": 3.4, "petalLength": 1.4, "petalWidth": 0.3, "species": "setosa"},
{"sepalLength": 5.0, "sepalWidth": 3.4, "petalLength": 1.5, "petalWidth": 0.2, "species": "setosa"},
{"sepalLength": 4.4, "sepalWidth": 2.9, "petalLength": 1.4, "petalWidth": 0.2, "species": "setosa"},
{"sepalLength": 4.9, "sepalWidth": 3.1, "petalLength": 1.5, "petalWidth": 0.1, "species": "setosa"},
{"sepalLength": 5.4, "sepalWidth": 3.7, "petalLength": 1.5, "petalWidth": 0.2, "species": "setosa"},
{"sepalLength": 4.8, "sepalWidth": 3.4, "petalLength": 1.6, "petalWidth": 0.2, "species": "setosa"},
{"sepalLength": 4.8, "sepalWidth": 3.0, "petalLength": 1.4, "petalWidth": 0.1, "species": "setosa"},
{"sepalLength": 4.3, "sepalWidth": 3.0, "petalLength": 1.1, "petalWidth": 0.1, "species": "setosa"},
{"sepalLength": 5.8, "sepalWidth": 4.0, "petalLength": 1.2, "petalWidth": 0.2, "species": "setosa"},
{"sepalLength": 5.7, "sepalWidth": 4.4, "petalLength": 1.5, "petalWidth": 0.4, "species": "setosa"},
{"sepalLength": 5.4, "sepalWidth": 3.9, "petalLength": 1.3, "petalWidth": 0.4, "species": "setosa"},
{"sepalLength": 5.1, "sepalWidth": 3.5, "petalLength": 1.4, "petalWidth": 0.3, "species": "setosa"},
{"sepalLength": 5.7, "sepalWidth": 3.8, "petalLength": 1.7, "petalWidth": 0.3, "species": "setosa"},
{"sepalLength": 5.1, "sepalWidth": 3.8, "petalLength": 1.5, "petalWidth": 0.3, "species": "setosa"},
{"sepalLength": 5.4, "sepalWidth": 3.4, "petalLength": 1.7, "petalWidth": 0.2, "species": "setosa"},
{"sepalLength": 5.1, "sepalWidth": 3.7, "petalLength": 1.5, "petalWidth": 0.4, "species": "setosa"},
{"sepalLength": 4.6, "sepalWidth": 3.6, "petalLength": 1.0, "petalWidth": 0.2, "species": "setosa"},
{"sepalLength": 5.1, "sepalWidth": 3.3, "petalLength": 1.7, "petalWidth": 0.5, "species": "setosa"},
{"sepalLength": 4.8, "sepalWidth": 3.4, "petalLength": 1.9, "petalWidth": 0.2, "species": "setosa"},
{"sepalLength": 5.0, "sepalWidth": 3.0, "petalLength": 1.6, "petalWidth": 0.2, "species": "setosa"},
{"sepalLength": 5.0, "sepalWidth": 3.4, "petalLength": 1.6, "petalWidth": 0.4, "species": "setosa"},
{"sepalLength": 5.2, "sepalWidth": 3.5, "petalLength": 1.5, "petalWidth": 0.2, "species": "setosa"},
{"sepalLength": 5.2, "sepalWidth": 3.4, "petalLength": 1.4, "petalWidth": 0.2, "species": "setosa"},
{"sepalLength": 4.7, "sepalWidth": 3.2, "petalLength": 1.6, "petalWidth": 0.2, "species": "setosa"}]
```

TEST CASES

Test Cases No.	Description	Input	Expected output	Actual output	result
1	check whether is located and accepted	iris.json	accepted	accepted	pass
2	check the output of '.json' file as string	iris.json	list of string	list of string	pass
3	check the output of '.json' file as dictionary	iris.json	list of dictionary	list of dictionary	pass
4	check output for setosa species	iris.json	list of dictionary with the key value setosa	list of dictionary with the key value setosa	pass
5	check output for Minimum sepal area and Maximum Petal area of all species	iris.json	Versicolor Maximum Sepal Area is 22.4 Minimum Petal Area is 3.3 Virginica Maximum Sepal Area is 30.02 Minimum Petal Area is 7.5 Setosa Maximum Sepal Area is 25.08 Minimum Petal Area is 0.11	Versicolor Maximum Sepal Area is 22.4 Minimum Petal Area is 3.3 Virginica Maximum Sepal Area is 30.02 Minimum Petal Area is 7.5 Setosa Maximum Sepal Area is 25.08 Minimum Petal Area is 0.11	pass
6	check the output for sorted dictionary	iris.json	sorted list of dictionary based on total area	sorted list of dictionary based on total area	pass

RESULT

Program executed Successfully and the output is obtained.

GIT LINK

<https://github.com/omals/Python-Lab/tree/main/LabCycle/LabCycle2/3Question>

BOX CLASS FOR SHAPES

AIM

Write a program to create a class Box with data members length, breadth, height, area, and volume. Provide constructor that enables initialization with one parameter (for cube), two parameters (for square prism) three parameters (rectangular prism). Also, provide functions to calculate area and volume.

Create a list of N boxes with random measurements and print the details of the box with maximum volume: area ratio.

THEORY

- Class - A class is a user-defined blueprint or prototype from which objects are created. Classes provide a means of bundling data and functionality together.
- objects - An Object is an instance of a Class. An object is simply a collection of data (variables) and methods (functions) that act on those data.
- constructor - Constructor allow you to create and properly initialize objects of a given class, making those objects ready to use.

PROGRAM

```
import random

class Box:                                #class box defined
    count=0                               #with the initial value of data members
    length = 0.0
    breadth = 0.0
    height = 0.0
    volume = 0.0
    area=0.0

    def __init__(self,*args):              #Constructor
        if(len(args)==1):
            self.length=args[0]
            self.count=0
        elif(len(args)==2):
            self.length = args[0]
            self.height = args[1]
            self.count=1
        elif(len(args)==3):
            self.length=args[0]
            self.breadth=args[1]
```

```
        self.height=args[2]
        self.count=2
    else:
        print("Constructor out of Scope.")
def area(self):                #member function to calculate area
    if(self.count==0):
        self.area=6*self.length**2
    elif(self.count==1):
        self.area=(2*self.length**2)+(4*self.length*self.height)
    elif(self.count==2):
        self.area=2*(self.breadth*self.length+self.height*#
                        self.length+self.height*self.breadth)
    else:
        print("Something went worry")
    if(self.count==0):
        self.volume=self.length**3
    elif(self.count==1):
        self.volume=(self.length**2)*self.height
    elif(self.count==2):
        self.volume=self.length*self.breadth*self.height
    else:
        print("Something went worry")
def display(self):            #member function to display the calculated values
    print("\tArea   : ",self.area)
    print("\tVolume : ",self.volume)

def ratio(self):
    r=self.volume/self.area
    print("\tRatio   : ",r)
    return r

def maxratio(r):              #function to find and check the maximum Volume:Area ratio
    if len(r["ratio"])==0 :   #exception
        print("Complete")
    else:
        templistratio=list(r["ratio"])
        #'templistratio'-temporary list of ratio from the dictionary
        maximum=max(templistratio)
        #'maximum'-Maximum ratio of the list of ratio 'r'
        tempi=templistratio.index(maximum)
        #'tempi'-Temporary index of maximum ratio in te list
        templistkey=list(r["Key"])
```

```
        # 'templistkey' - Temporary list of key from the dictionary
keyvalue=(int(templistkey[tempi]))
        # 'keyvalue' - to get the key value at that index
print("\nMaximum volume:area ratio for ",end="")
if keyvalue== 1 :
    print ("Cube. Value = ",templistratio[tempi],"\n")
elif keyvalue==2:
    print ("Square Prism. Value = ",templistratio[tempi],"\n")
elif keyvalue==3:
    print ("Rectangular Prism. Value = ",templistratio[tempi],"\n")
else:
    print("Something Wrong","\n")
def cube():
    cube=[]                                #cube list declaration for dimensions
    cube.append(random.randint(1,1000))    #random values assigned
    print("_"*70)
    print("Cube : dimensions = ",cube)
    cube_obj=Box(cube[0])
        #object declaration constructor with one argument is called
    cube_obj.area()
    cube_obj.display()
    return(cube_obj.ratio())
def squareprisum():
    square=[]                            #square prism list declaration for dimensions
    for i in range(2):
        square.append(random.randint(1,1000))    #random values assigned
    print("_"*70)
    print("square Prism : dimensions = ",square)
    squarep_obj=Box(square[0],square[1])
        #object declaration constructor with two arguments is called
    squarep_obj.area()
    squarep_obj.display()
    return(squarep_obj.ratio())
def rectangularprisum():
    rectangle=[]                        #rectangle list declaration for dimensions
    for i in range(3):
        rectangle.append(random.randint(1,1000))    #random values assigned
    print("_"*70)
    print("Rectangular Prism : dimensions = ",rectangle)
    rectangularp_obj=Box(rectangle[0],rectangle[1],rectangle[2])
        #object declaration with three arguments called
```

```
    rectangularp_obj.area()
    rectangularp_obj.display()
    return(rectangularp_obj.ratio())
def main():
    #main function
    n=int(input("\nEnter the number of BOX required : "))
    ratio={"ratio": [], "Key": []}
    #ratio dictionary of list declaration for storing the values
    if(n<=2):
        for k in range(0,n,2):
#loop to create box of different shapes with random values
            if(k<n):
                ratiov=cube()
                ratio["ratio"].append(ratiov)
                #appending the values to the dictionary
                ratio["Key"].append("1")
                k=k+1
            if(k<n):
                ratiov=squareprisum()
                ratio["ratio"].append(ratiov)
                ratio["Key"].append("2")
                k=k+1
    else:
        for k in range(0,n,3):
            if(k<n):
                ratiov=cube()
                ratio["ratio"].append(ratiov)
                ratio["Key"].append("1")
                k=k+1
            if(k<n):
                ratiov=squareprisum()
                ratio["ratio"].append(ratiov)
                ratio["Key"].append("2")
                k=k+1
            if(k<n):
                ratiov=rectangularprisum()
                ratio["ratio"].append(ratiov)
                ratio["Key"].append("3")
                k=k+1
    print("_"*70)
    maxratio(ratio)
#function call
main()
```


SAMPLE INPUT-OUTPUT

```

Enter the number of BOX required : 4
-----
Cube : dimensions = [140]
      Area  : 117600
      Volume : 2744000
      Ratio  : 23.33333333333332
-----
square Prism : dimensions = [306, 186]
      Area  : 414936
      Volume : 17416296
      Ratio  : 41.97345132743363
-----
Rectangular Prism : dimensions = [108, 525, 417]
      Area  : 641322
      Volume : 23643900
      Ratio  : 36.867439445395604
-----
Cube : dimensions = [59]
      Area  : 20886
      Volume : 205379
      Ratio  : 9.833333333333334
-----
Maximum volume:area ratio for Square Prism. Value = 41.97345132743363

```

TEST CASES

Test Cases No.	Description	Input	Expected output	Actual output	result
1	check the output for the number of boxes created	4	Cube Square Prism Rectangular Prism Cube	Cube Square Prism Rectangular Prism Cube	pass
2	check the random variables are used for values	dimensions	different values in range of (1,1000)	different values in range of (1,1000)	pass
3	check the output of maximum volume : area ratio	area and volume generated	Maximum value from the ration obtained	Maximum value from the ration obtained	pass
4	check for wrong input	a	error	error	pass

RESULT

Program executed Successfully and the output is obtained.

GIT LINK

<https://github.com/omals/Python-Lab/tree/main/LabCycle/LabCycle2/4Question>

3D_SHAPES INHERITANCE

AIM

Write a program to create a parent class, 3D_Shapes, with methods print_Volume() and print_Area(), which prints the Volume and Area, respectively. Create classes Cylinder and Sphere by inheriting 3D_Shapes class. Using these child classes, calculate and print the volume and area of a cylinder and sphere

THEORY

- Inheritance - Inheritance refers to defining a new class with little or no modification to an existing class. The new class is called derived (child) class and the one from which it inherits is called the base (parent) class.

PROGRAM

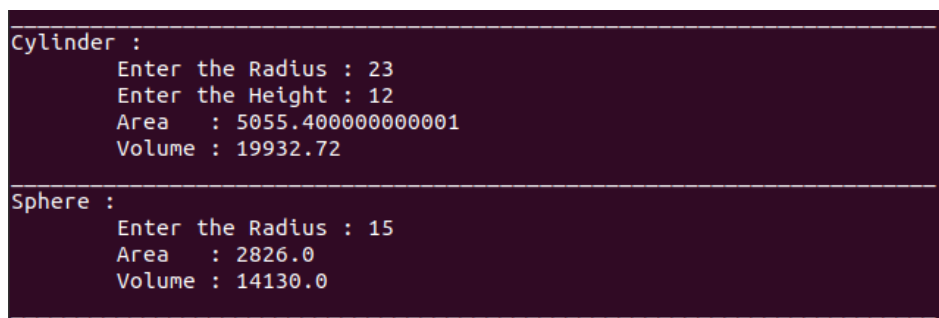
```
class threeD_shape:                #Base class threeD_shape
    def printVolume(self):
        print("\tVolume :",self.volume)
    def printArea(self):
        print("\tArea   :",self.area)
class Cylinder(threeD_shape):      #Derived class Cylinder
    def __init__(self,r,h):        #Constructor
        self.radius=r
        self.height=h
    def calculateA(self):           #member function to calculate area of cylinder
        self.area=(2*3.14*self.radius*self.height)+(2*3.14*self.radius**2)
        threeD_shape.printArea(self)
    def calculateV(self):           #member function to calculate volume of cylinder
        self.volume=3.14*self.radius**2*self.height
        threeD_shape.printVolume(self)    #call to the base class function
class Sphere(threeD_shape):        #Derived class Sphere
    def __init__(self,r):          #Constructor
        self.radius=r
    def calculateA(self):           #member function to calculate area of sphere
        self.area=4*3.14*self.radius**2
        threeD_shape.printArea(self)
    def calculateV(self):           #member function to calculate volume of sphere
        self.volume=4/3*3.14*self.radius**3
        threeD_shape.printVolume(self)    #call to the base class function
def main():
    print("_"*70)
```

```
print("Cylinder : ")
cylinderR=int(input("\tEnter the Radius : "))
cylinderH=int(input("\tEnter the Height : "))

cylinder_obj=Cylinder(cylinderR,cylinderH)      #Object declaration
cylinder_obj.calculateA()
cylinder_obj.calculateV()
print("_"*70)
print("Sphere : ")
sphereR=int(input("\tEnter the Radius : "))

sphere_obj=Sphere(sphereR)                      #Object declaration
sphere_obj.calculateA()
sphere_obj.calculateV()
print("_"*70)
main()                                           #beginning of program execution
```

SAMPLE INPUT-OUTPUT



```
Cylinder :
Enter the Radius : 23
Enter the Height : 12
Area : 5055.400000000001
Volume : 19932.72

Sphere :
Enter the Radius : 15
Area : 2826.0
Volume : 14130.0
```

TEST CASES

Test Cases	Description	input	Expected output	Actual output	Result
1	check the output for cylinder	23 12	Area=5055.400000000001 Volume : 19932.72	Area=5055.400000000001 Volume : 19932.72	pass
2	check output for sphere	15	Area : 2826.0 Volume : 14130.0	Area : 2826.0 Volume : 14130.0	pass
3	check wrong input	a	error	error	pass

RESULT

Program executed Successfully and the output is obtained.

GIT LINK

<https://github.com/omals/Python-Lab/tree/main/LabCycle/LabCycle2/5Question>

TIC TAC TOE

AIM

Develop a two-player tic-tac-toe game using pygame

THEORY

- Pygame library - Pygame is a cross-platform set of Python modules designed for writing video games.

PROGRAM

```
from itertools import count
from timeit import repeat
import pygame,sys
import numpy as np

pygame.init()

WIDTH=600
HEIGHT=800
LINE_WIDTH=25
WINDOW_COL=1
WINDOW_ROW=4
BOARD_ROWS = 3
BOARD_COLS = 3
CIRCLE_RADIUS = 50
CIRCLE_WIDTH = 10
CROSS_WIDTH = 15
SPACE = 55
RED = (255,0,0)
CIRCLE_COLOUR = (120,120,120)
LINE_COLOUR=(232,232,232)
BG_COLOUR=(200,200,200)

screen=pygame.display.set_mode((WIDTH,HEIGHT))
pygame.display.set_caption('TIC TAC TOE')
screen.fill( BG_COLOUR )

board = np.zeros((BOARD_ROWS,BOARD_COLS))
window=np.zeros((WINDOW_ROW,WINDOW_COL))
global win
```

```
def draw_lines():
    #horizontal
    pygame.draw.line(screen, LINE_COLOUR, (20,200), (580,200),LINE_WIDTH )
    pygame.draw.line(screen, LINE_COLOUR, (20,400), (580,400),LINE_WIDTH )
    #vertical
    pygame.draw.line(screen, LINE_COLOUR, (200,15), (200,590),LINE_WIDTH )
    pygame.draw.line(screen, LINE_COLOUR, (400,15), (400,590),LINE_WIDTH )
    #borders
    pygame.draw.line(screen, LINE_COLOUR, (0,10), (600,10),25)
    pygame.draw.line(screen, LINE_COLOUR, (10,0), (10,600),25 )

    pygame.draw.line(screen, LINE_COLOUR, (590,10), (590,600),25)
    pygame.draw.line(screen, LINE_COLOUR, (0,600), (600,600),25)
    rec=pygame.draw.rect(screen, CIRCLE_COLOUR, pygame.Rect(50, 650, 200, 100))
    font = pygame.font.SysFont('Arial', 35)
    screen.blit(font.render('Restart', True, (0,0,0)), ((85,685) ))
    pygame.display.update()

def draw_figures():
    for row in range(BOARD_ROWS):
        for col in range(BOARD_COLS):
            if board[row][col] == 1:
                pygame.draw.circle( screen,CIRCLE_COLOUR,(int( col*200+100),#
                    int(row*200+110)),CIRCLE_RADIUS,CIRCLE_WIDTH)
            elif board[row][col] == 2:
                pygame.draw.line( screen,CIRCLE_COLOUR,(col*200+SPACE,#
                    row*200+200-SPACE),(col*200+200-SPACE,#
                    row*200+SPACE),CROSS_WIDTH)
                pygame.draw.line( screen,CIRCLE_COLOUR,(col*200+SPACE,#
                    row*200+SPACE),(col*200+200-SPACE,#
                    row*200+200-SPACE),CROSS_WIDTH)

def mark_square(row,col,player):
    board[row][col] = player

def available_square(row,col):
    #print(board[row][col])
    if board[row][col] == 0:
        return True
```

```
    else:
        return False

def is_board_full():
    for row in range(BOARD_ROWS):
        for col in range(BOARD_COLS):
            if board[row][col] == 0:
                return False
    #status()
    return True

def check_win(player):
    #vertical check
    for col in range(BOARD_COLS):
        if board[0][col] == player and board[1][col] == player and #
            board[2][col] == player:
            draw_vertical_winning_line(col,player)
            return True

    #horizontal check
    for row in range(BOARD_ROWS):
        if board[row][0] == player and board[row][1] == player and #
            board[row][2] == player:
            draw_horizontal_winning_line(row,player)
            return True

    #asc diagonla check
    if board[2][0] == player and board[1][1] == player and board[0][2] == player:
        draw_asc_diagonal(player)
        return True

    #desc Diagonal check
    if board[0][0] == player and board[1][1] == player and board[2][2] == player:
        draw_desc_diagonal(player)
        return True

    return False

def draw_vertical_winning_line(col,player):
    posX = col * 200 + 100
```

```
    if player == 1:
        colour = RED
        print("O WINS")
    elif player == 2:
        colour = RED
        print("X WINS")
    win=1
    pygame.draw.line(screen, CIRCLE_COLOUR, (posX,35), (posX,600-30),10)

def draw_horizontal_winning_line(row,player):
    posY = row * 200 + 100

    if player == 1:
        print("O WINS")
    elif player == 2:
        print("X WINS")
    win=1

    pygame.draw.line(screen,CIRCLE_COLOUR,(35,posY),(600-30,posY),10)

def draw_asc_diagonal(player):
    if player == 1:
        print("O WINS")
    elif player == 2:
        print("X WINS")
    pygame.draw.line(screen,CIRCLE_COLOUR,(35,600-35),(600-35,35),10)
    win=1

def draw_desc_diagonal(player):
    if player == 1:
        print("O WINS")
    elif player == 2:
        print("X WINS")
    pygame.draw.line(screen,CIRCLE_COLOUR,(35,35),(600-35,600-35),10)
    win=1

def status():
```

```
    if win == "X":
        print("X WINS")
    elif win == "O":
        print("O WINS")

def check_outside(row,col):
    if (row>=50 and row<=250)and (col>=650 and col<=750):
        #print("Entered2")
        return True
    else:
        return False

def restart():
    screen.fill( BG_COLOUR)
    draw_lines()
    player = 1
    for row in range(BOARD_ROWS):
        for col in range(BOARD_COLS):
            board[row][col] = 0

draw_lines()

player = 1
win=0
game_over = False

mouseX = 0
mouseY = 0
count = 0
while True:
    for event in pygame.event.get():

        if event.type == pygame.MOUSEBUTTONDOWN and not game_over:
```



```
mouseX = event.pos[0]
mouseY = event.pos[1]

clicked_row = int(mouseY // 200)
clicked_col = int(mouseX // 200)
#print(clicked_row)
#print(WINDOW_ROW)
#print(clicked_row != WINDOW_ROW-1)

if clicked_row != WINDOW_ROW-1:
    if available_square(clicked_row, clicked_col):

        if player == 1:
            mark_square(clicked_row, clicked_col, 1)
            if check_win(player):
                game_over = True
            player = 2
        elif player == 2:
            mark_square(clicked_row, clicked_col, 2)
            if check_win(player):
                game_over = True
            player = 1

        draw_figures()
        if game_over == True and count == 0:
            status()

elif event.type == pygame.MOUSEBUTTONDOWN :
    #print("Entered")
    #print(mouseX, " ", mouseY)
    if check_outside(mouseX, mouseY):
        #print("Get restart")
        restart()
        game_over=False

pygame.display.update()

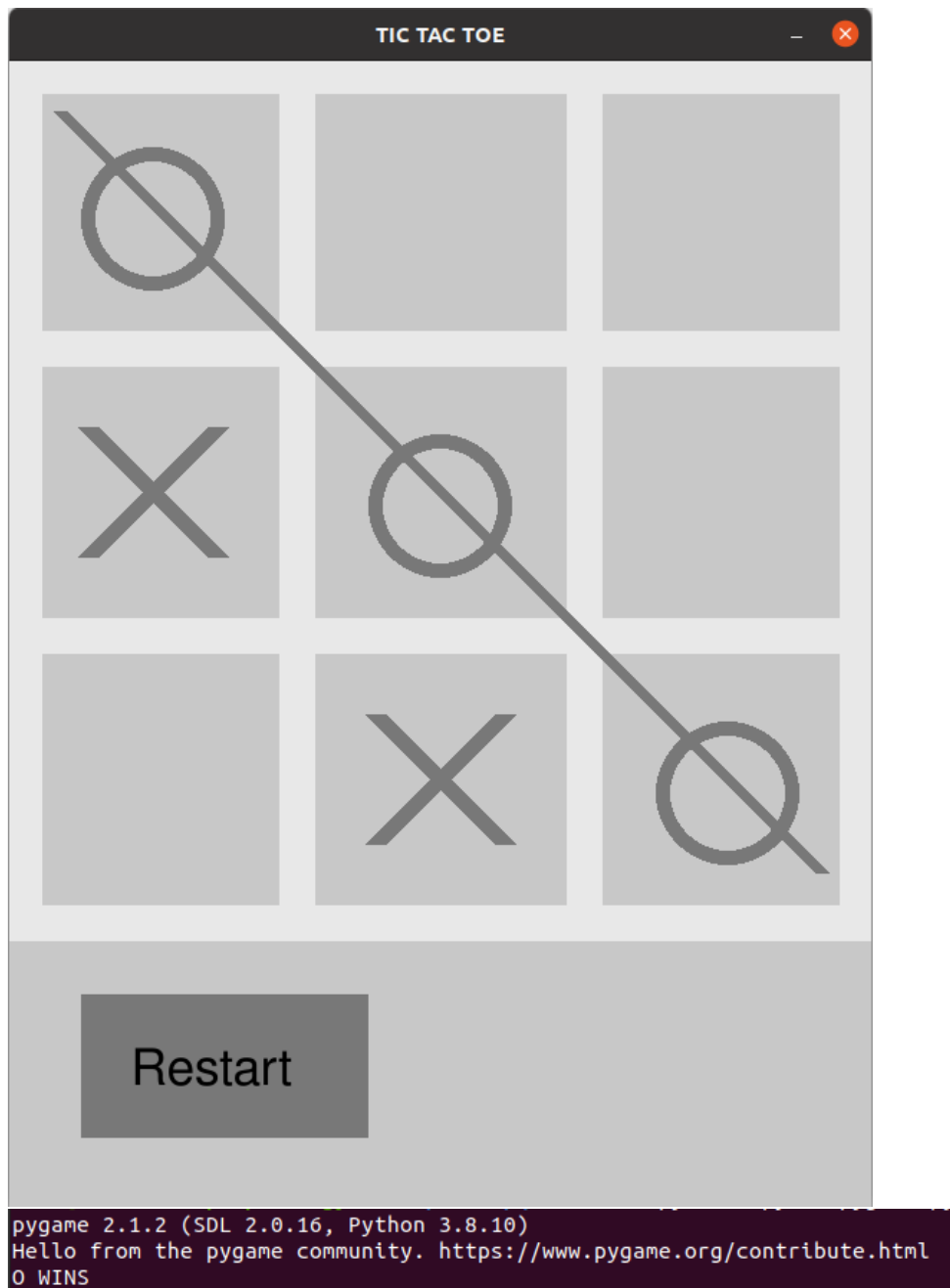
if event.type == pygame.KEYDOWN:
```

```
        if event.key == pygame.K_r:
            restart()
            game_over = False

    if event.type == pygame.QUIT:
        sys.exit()

    pygame.display.update()
```

SAMPLE INPUT-OUTPUT



TEST CASES

Test Cases No.	Description	Input	Expected output	Actual output	result
1	Check for the display of 'O' and 'X' in the screen	Mouse click	Mark corresponding to the player 1(O) or player 2(X)	Mark corresponding to the player 1(O) or player 2(X)	Pass
2	Check for the line on winning	Same pattern in same line	line over the winning pattern	line over the winning pattern	Pass
3	Check for winner display in the command terminal window	Same pattern in the line	Display "O WINS" or "X WINS"	Display "O WINS" or "X WINS"	Pass
4	Check for reset option in between game	Click on the reset button on the window	Overall reset of the window to the default state	Overall reset of the window to the default state	Pass
5	Check for reset window after the game over	click on keyboard key "r"	Overall reset of the window to the default state	Overall reset of the window to the default state	Pass
6	Check for quit option by clicking the close window button	mouse click	window closes program ends execution	window closes program ends execution	

RESULT

Program executed Successfully and the output is obtained.

GIT LINK

<https://github.com/omals/Python-Lab/tree/main/LabCycle/LabCycle3/question1>

PRINCIPAL COMPONENT ANALYSIS ON MATRIX

AIM

Implement Principle Component Analysis(PCA) of a matrix.

THEORY

- Numpy - NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays.
- Linear Algebra - The NumPy linear algebra functions rely on BLAS and LAPACK to provide efficient low level implementations of standard linear algebra algorithms.

PROGRAM

```
import numpy as np
import pandas as pd
def PCA(X , num_components):

    # mean Centering the data
    X_meaned = X - np.mean(X , axis = 0)

    # calculating the covariance matrix of the mean-centered data.
    cov_mat = np.cov(X_meaned, rowvar = False)
    print("\nCovariance Matrix :\n",cov_mat)

    #Calculating Eigenvalues and Eigenvectors of the covariance matrix
    eigen_values , eigen_vectors = np.linalg.eigh(cov_mat)
    print("\nEigen Value :\n",eigen_values)
    print("\nEigen Vector:\n",eigen_vectors)

    #sort the eigenvalues and eigenvectors in descending order
    sorted_index = np.argsort(eigen_values)[::-1]
    sorted_eigenvalue = eigen_values[sorted_index]
    sorted_eigenvectors = eigen_vectors[:,sorted_index]

    # select the first n eigenvectors, n is desired dimension
    eigenvector_subset = sorted_eigenvectors[:,0:num_components]
    #Transform the data
    X_reduced = np.dot(eigenvector_subset.transpose() ,#
                       X_meaned.transpose() ).transpose()
```

```
        return X_reduced

r=int(input("Enter the number of row      : "))
c=int(input("Enter the number of columns : "))
print("\nEnter the values in the form of",r,"*",c,"matrix form :");
a=[(input().split()) for j in range(r)]
matrix=np.array(a,float)
print("\nMatrix : \n",matrix)

mat_reduced = PCA(matrix , 2)

#Creating a Pandas DataFrame of reduced Dataset
principal_df = pd.DataFrame(mat_reduced , columns = ['PC1','PC2'])
print("\nPrincipal Component Analysis : \n")
print(principal_df)
```

SAMPLE INPUT-OUTPUT

```
Enter the number of row      : 5
Enter the number of columns : 3

Enter the values in the form of 5 * 3 matrix form :
90 60 90
90 90 30
60 60 60
60 60 90
30 30 30

Matrix :
[[90. 60. 90.]
 [90. 90. 30.]
 [60. 60. 60.]
 [60. 60. 90.]
 [30. 30. 30.]]

Covariance Matrix :
[[630. 450. 225.]
 [450. 450.   0.]
 [225.   0. 900.]]

Eigen Value :
[ 56.02457535 786.38798335 1137.5874413 ]

Eigen Vector:
[[-0.6487899  0.3859988 -0.65580225]
 [ 0.74104991  0.51636642 -0.4291978 ]
 [ 0.17296443 -0.7644414 -0.62105769]]

Principal Component Analysis :

      PC1      PC2
0 -34.370985 -13.669271
1 -9.983457  47.688206
2  3.934814 -2.315993
3 -14.696917 -25.249235
4  55.116546 -6.453707
```

TEST CASES

Test Cases	Description	Input	Expected output	Actual Output	Result
1	Check Matrix input	90 60 90 90 90 30 60 60 60 60 60 90 30 30 30	Inputed matrix is displayed	Inputed matrix is displayed	Pass
2	Check for covariance, Eigen Value,Eigen Vector	Matrix input	Value obtained	value obtained	pass
3	Check for Principal Component Analysis two values	Matrix input	Two P C A Values	Two P C A Values	Pass

RESULT

Program executed Successfully and the output is obtained.

GIT LINK

<https://github.com/omals/Python-Lab/tree/main/LabCycle/LabCycle3/question2>

DATA VISUALIZATION

AIM

Create an account in Kaggle.com Download iris dataset Load it using pandas library Prepare the following charts :

- Bar chart showing the frequency of species column
- Apply PCA to get two principle components and show the data distribution as a scatter plot. (use function from sklearn)
- Show the distribution of each attribute as histogram.

THEORY

- Visualization - Matplotlib and Seaborn are python libraries that are used for data visualization. They have inbuilt modules for plotting different graphs.
- Data processing - Python can handle various encoding processes, and different types of modules need to be imported to make these encoding techniques work. Pandas is a Python language package, which is used for data processing.
- Libraries :
 - pandas - Pandas is an open source Python package that is most widely used for data science/data analysis and machine learning tasks.
 - matplotlib - Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy.
 - seaborn - Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures.
- histogram - A histogram is basically used to represent data provided in a form of some groups.

PROGRAM

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
#import plotly.express as px
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
dataframe = pd.read_csv('Iris.csv')
dataframe.head()
```

```
def frequency_graph():
    y=dataframe['Species'].value_counts()
    y.plot.bar(color=['red','orange','yellow'])
    plt.title('Frequency graph')
    plt.xlabel('Species')
    plt.ylabel('Frequency')
    plt.show()

def pcacomponent():
    X = dataframe[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']]
    pca = PCA(n_components=2)
    components = pca.fit_transform(X)

    principalDf=pd.DataFrame(data=components,columns=['principal component 1'#
                                                    , 'principal component 2'])

    principalDf.head()
    finalDf=pd.concat([principalDf,dataframe[['Species']]],axis=1)
    finalDf.head()
    fig=plt.figure(figsize=(8,8))
    ax=fig.add_subplot(1,1,1)
    ax.set_xlabel('Principal Component 1',fontsize = 15)
    ax.set_ylabel('Principal Component 2',fontsize = 15)
    ax.set_title('Two Principle Components',fontsize=20)
    targets=['Iris-setosa','Iris-versicolor','Iris-virginica']
    colors=['r','g','b']
    for target,color in zip(targets,colors):
        indicesToKeep = finalDf['Species'] == target
        ax.scatter(finalDf.loc[indicesToKeep,'principal component 1'],
                    finalDf.loc[indicesToKeep,'principal component 2'],
                    c=color,
                    s=50)
    ax.legend(targets)
    ax.grid()
    plt.show()

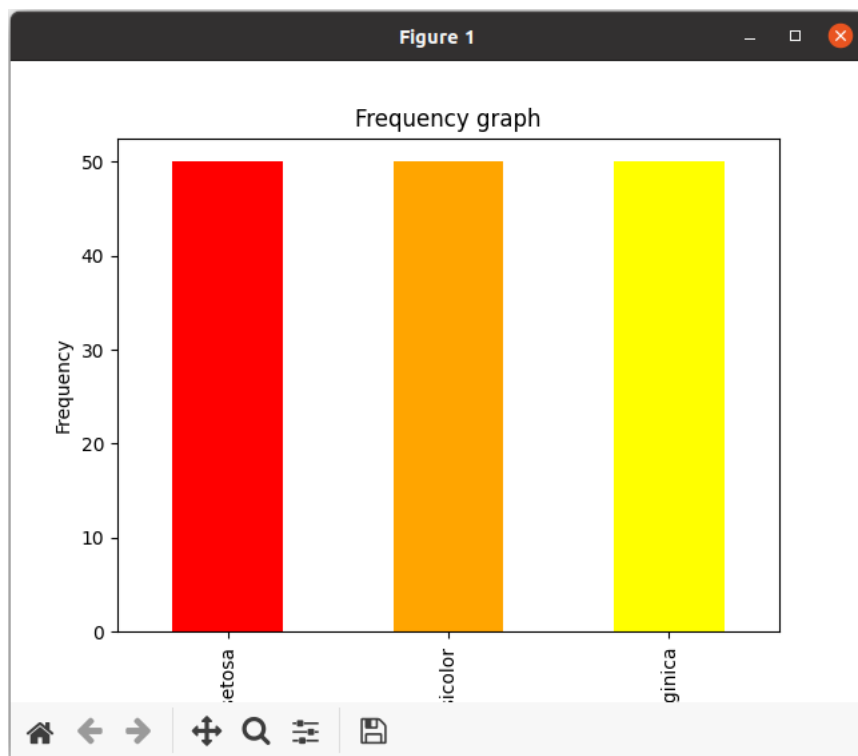
def histogram():
    dataframe['SepalLengthCm'].plot(kind='hist')
    plt.title('Sepal Length Histogram')
    plt.xlabel('Sepal Length')
    plt.ylabel('Distribution')
    plt.show()
```

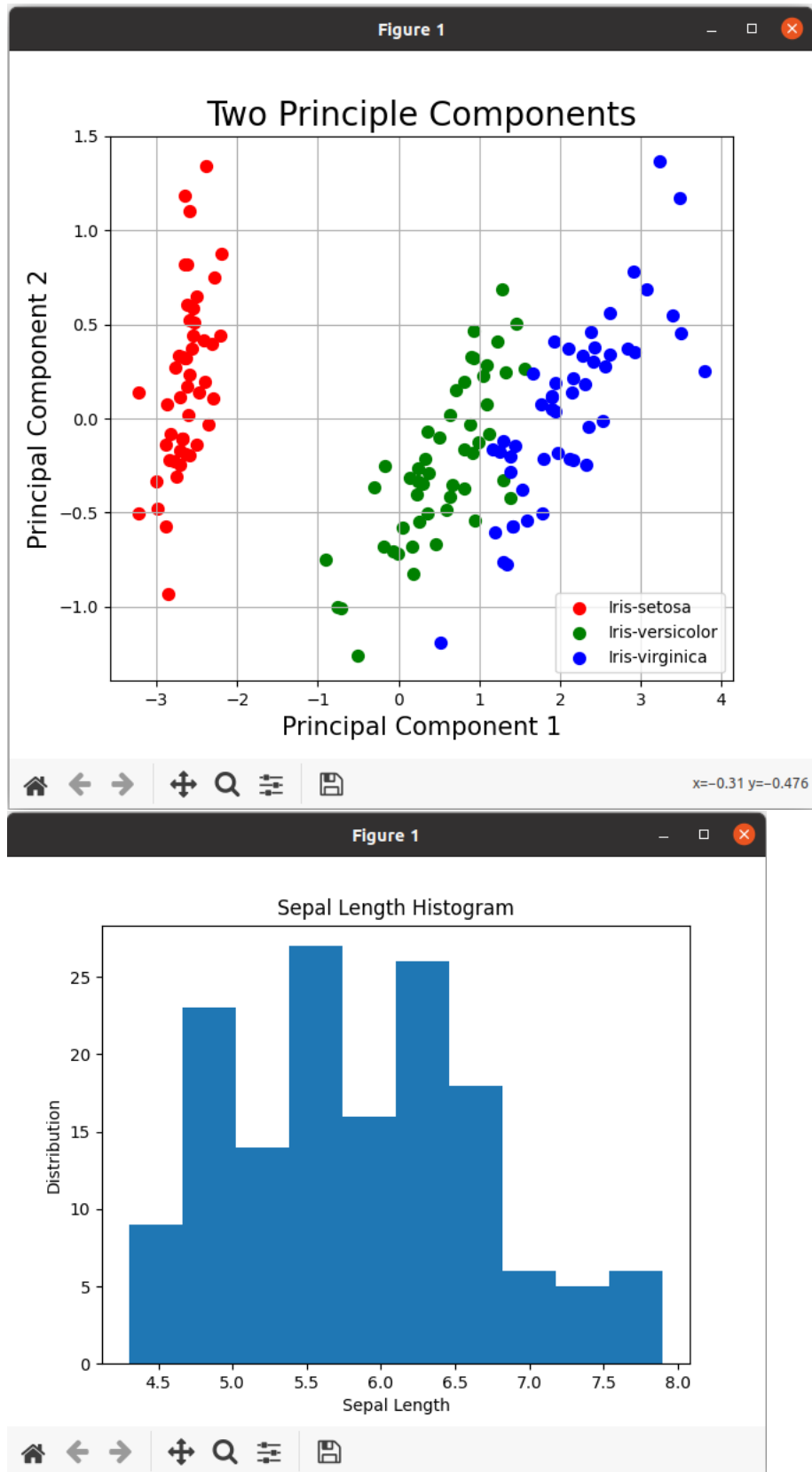


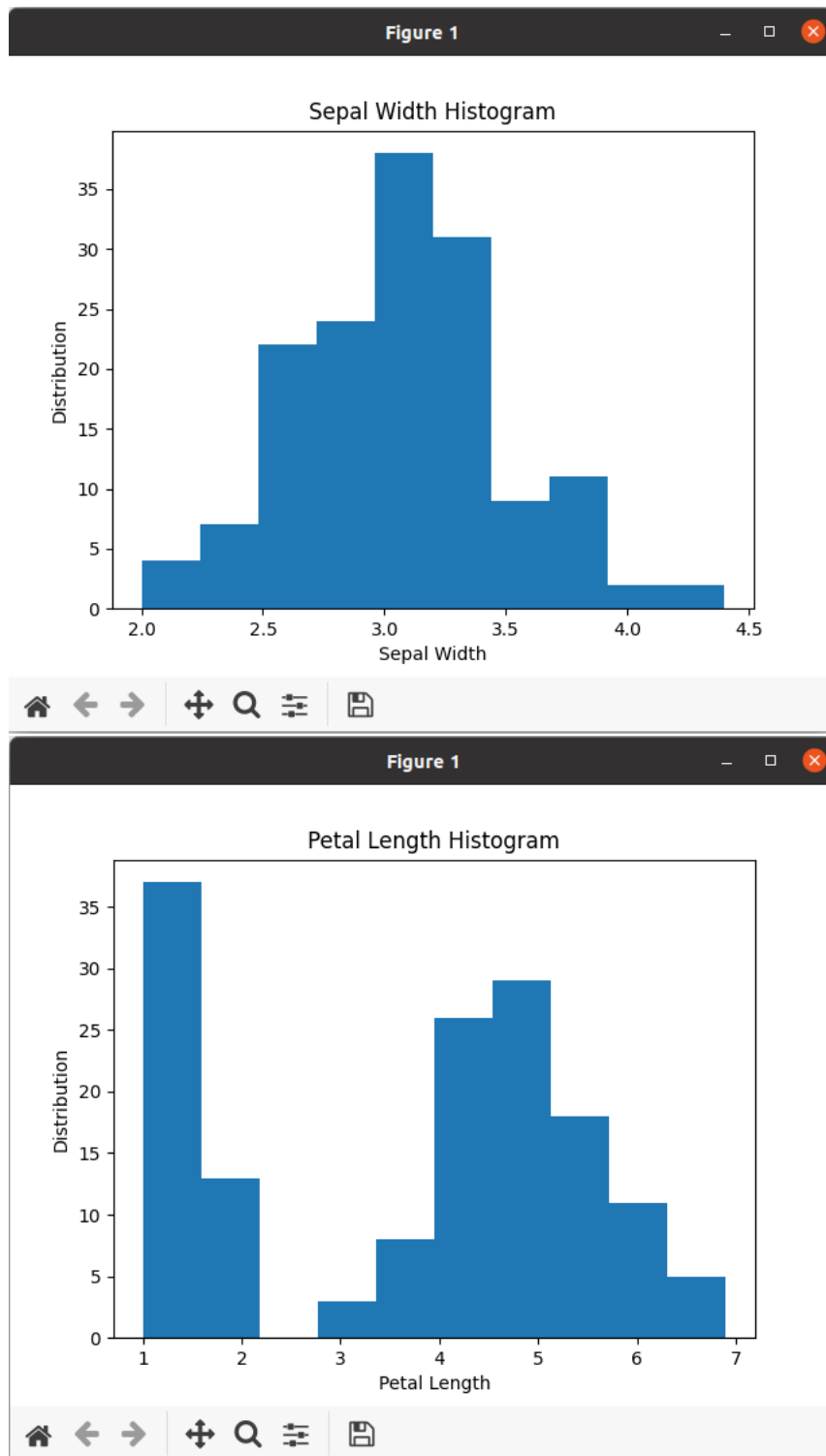
```
dataframe['SepalWidthCm'].plot(kind='hist')
plt.title('Sepal Width Histogram')
plt.xlabel('Sepal Width')
plt.ylabel('Distribution')
plt.show()
dataframe['PetalLengthCm'].plot(kind='hist')
plt.title('Petal Length Histogram')
plt.xlabel('Petal Length')
plt.ylabel('Distribution')
plt.show()
dataframe['PetalWidthCm'].plot(kind='hist')
plt.title('Petal Width Histogram')
plt.xlabel('Petal Width')
plt.ylabel('Distribution')
plt.show()

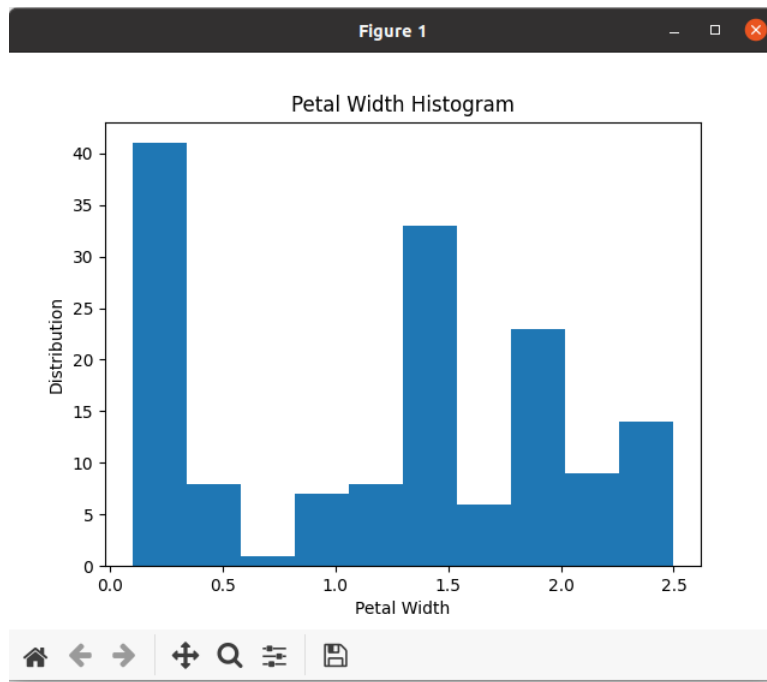
frequency_graph()
pcacomponent()
histogram()
```

SAMPLE INPUT-OUTPUT









TEST CASES

Test Cases	Description	Input	Expected output	Actual Output	Result
1	Check .csv file output of bar graph	Iris.csv file	Bar Graph	Bar Graph	Pass
2	Display Scattered Graph for Principal Component Values	Sepal.LengthCm Sepal.WidthCm Petal.LengthCm Petal.WidthCm	Scattered Graph	Scattered Graph	pass
3	Display separate Histogram for Sepal.LengthCmSepal.WidthCm Petal.LengthCm Petal.WidthCm	Iris.csv file	Histogram	Histogram	Pass

RESULT

Program executed Successfully and the output is obtained.

GIT LINK

<https://github.com/omals/Python-Lab/tree/main/LabCycle/LabCycle3/question3>

VEHICLE DETAILS

AIM

Design a class to store the details of a vehicle such as engine number, model, type, mileage, vendor, registration number, and owner name. Design another class that holds the details of several vehicles and provide functions to

- Display the details of the collection
- the collection according to mileage
- Add, Delete and Modify the entries from the collection
- Store and Load the collection as a pickle file
- Filter the result according to the attributes and export it as a report.

THEORY

- OOPs - Object-oriented Programming (OOPs) is a programming paradigm that uses objects and classes in programming. It aims to implement real-world entities like inheritance, polymorphisms, encapsulation, etc. in the programming.
- Pickle - Python pickle module is used for serializing and de-serializing python object structures.
- PDF report generation - The data input in the program into PDF report by using FPDF module
- Lambda functions for sorting - lambda is used as a function to iterate on each element.
key = lambda x:x[i] here i is the column on which respect to sort the whole list.

PROGRAM

```
import pickle,tabulate
from fpdf import FPDF
vehiclist=[[324942,'Polo_GT','Rahul','Hatchback',19.3,'volkswagen',2456],#
[452313, 'XZ_Plus_LUX', 'Anju', 'SUV', 31.2, 'Tata_Motors', 8734]]
with open('vehicledata.pkl','wb') as vehiclepickle:
    pickle.dump(vehiclist,vehiclepickle)
class Vehicle:
    def __init__(self,EN,MO,ON,TV,MI,V,RN):
        self.enginenumber=EN
        self.model=MO
        self.ownername=ON
```

```
        self.type_v=TV
        self.mileage=MI
        self.vendor=V
        self.registrationNumber=RN
    def display(self):
        s = "{:<9} {:^11} {:^11} {:^10} {:^15} {:^15} {:^15}".#
            format(self.enginenumbr,self.model,self.ownername,self.type_v,#
                self.mileage,self.vendor,self.registrationNumber)
        print(s)

class SeveralVehicles(Vehicle):
    def __init__(self):
        self.vehiclelist=list()
        self.vehicledetails=list()

    def readfile(self,filename):
        vehicledata=open(filename,'rb')
        vehiclereaddata=pickle.load(vehicledata)
        for i in vehiclereaddata:
            vehicle=Vehicle(i[0],i[1],i[2],i[3],i[4],i[5],i[6])
            self.vehiclelist.append(vehicle)
            self.vehicledetails.append(i)
        vehicledata.close()

    def addvehicle(self,vehicle):
        self.vehiclelist.append(vehicle)

    def displaydetails(self):
        print("No.\tEngNo.\t Model\t\tOwner\t Type\t\tMileage#
            \t\tVendor\t\tRegNo.\n")

        No = 1
        for vehicleardetail in self.vehiclelist:
            print(No,end = "\t")
            No+=1
            vehicleardetail.display()
            print()

    def mileagesort(self):
        S_list = sorted(self.vehicledetails,key = lambda x : x[3])
        vehiclelistindex = 0
        self.vehicledetails = S_list
```

```
        for i in S_list:
            A = Vehicle(i[0],i[1],i[2],i[3],i[4],i[5],i[6])
            self.vehiclelist[vehiclelistindex] = A
            vehiclelistindex+=1

def Delete(self,RegNo):
    check = False
    del_index = 0
    for i in self.vehiclelist:
        if i.registrationNumber == RegNo:
            self.vehiclelist.pop(del_index)
            check = True
            del_index+=1
    if not check:
        print(RegNo,"Does not Exist")

def Modify(self,RegNo,Detail_name,Change_value):
    check = False
    for i in self.vehiclelist:
        if i.registrationNumber == RegNo:
            check = True
            if i.Detail_name:
                self.vehiclelist[i].Detail_name = Change_value
            else:
                print(Detail_name,"is not a valid detail")
    if not check:
        print(RegNo,"is not a valid parameter")

def Save(self,filename):
    Details = open(filename,"wb")
    Details.truncate()
    print(self.vehiclelist)
    pickle.dump(self.vehiclelist,Details)
    Details.close()

def filter(self):
    print("1.Owner Name\n2.Vendor name \n3.Model Name \n4.Type\n5.Mileage\n")
    option = int(input("Choose a Data to filter : "))
    filteredList = list()
    if(option==1):
        filterKey = (input("Enter the name you want to filter"))
```

```
        filteredList = [i for i in self.vehiclelist if i['ownerName']== filterKey]
        self.display("Filtered List",filteredList)
    elif (option==2):
        filterKey = (input("Enter the Vendor name you want to filter"))
        filteredList = [i for i in self.vehiclelist if i['vendor']== filterKey]
        self.display("Filtered List",filteredList)
    elif (option==3):
        filterKey = (input("Enter the Model name you want to filter"))
        filteredList = [i for i in self.vehiclelist if i['model']== filterKey]
        self.display("Filtered List",filteredList)
    elif (option==4):
        filterKey = (input("Enter the type you want to filter"))
        filteredList = [i for i in self.vehiclelist if i['type']== filterKey]
        self.display("Filtered List",filteredList)
    elif(option==5):
        filterKey = float(input("Enter the mileage you want to filter"))
        filteredList = [i for i in self.vehiclelist if i['mileage']== filterKey]
        self.display("Filtered List",filteredList)

def report(self,filename):
    pdf = FPDF()
    pdf.add_page()
    pdf.set_font("Arial",size = 10)
    pdf.cell(200,10,ln = 2,align = "C",txt = "No.\tEngNo.\tModel\tType#\n\tMileage\tVendor\tRegNo.\tOwner\n")

    for entries in self.vehiclelist:
        add_text = ""
        print(entries)
        add_text+=str(entries)
        add_text+="\n"
        pdf.cell(200,10,ln = 2,align = "C",txt = add_text)
    pdf.output(filename)

def main():
    vehicle=SeveralVehicles()
    Key=True
    while Key:
        print("\n\tVEHICLE DETAILS COLLECTION\n1.Display \n2.Sort the mileage#\n3.Add \n4. Delete \n5.Modify \n6.Store as a pickle file\n7.#\n\tLoad the pickle file \n8.Filter the data\n9.Export it as a pdf #")
```



```
        report\n10.Exit\n")
choice=int(input("Enter the choice : "))
if choice == 1:
    vehicle.displaydetails()
elif choice == 2:
    vehicle.mileagesort()
    vehicle.displaydetails()
elif choice == 3:
    print("Enter the details : ")
    d1 = int(input("Engine No  :"))
    d2 = input("Model : ")
    d3 = input("Type  : ")
    d4 = int(input("Mileage : "))
    d5 = input("Vendor : ")
    d6 = int(input("Registration No. :"))
    d7 = input("Owner Name : ")
    to_add_vehicle = Vehicle(d1,d2,d7,d3,d4,d5,d6)
    vehicle.addvehicle(to_add_vehicle)
elif choice == 4:
    to_delete_RegNo = int(input("Enter Reg No to delete : "))
    vehicle.Delete(to_delete_RegNo)
elif choice == 5:
    to_modify_RegNo = int(input("Enter Reg No to Modify : "))
    to_modify_detail = int(input("Enter Model name : "))
    to_modify_CV = int(input("Enter Change Value to Modify : "))
    vehicle.Modify(to_modify_RegNo,to_modify_detail,to_modify_CV)
elif choice == 6:
    vehicle.Save("vehicledata.pkl")
elif choice == 7:
    vehicle.readfile("vehicledata.pkl")
elif choice ==8:
    vehicle.filter()
elif choice == 9:
    vehicle.report("report.pdf")
elif choice == 10:
    Key = False
else:
    print("Invalid option try again...")
```

```
main()
```

SAMPLE INPUT-OUTPUT

```
VEHICLE DETAILS COLLECTION
1.Display
2.Sort the mileage
3.Add
4. Delete
5.Modify
6.Store as a pickle file
7.Load the pickle file
8.Filter the data
9.Export it as a pdf report
10.Exit

Enter the choice : 7

VEHICLE DETAILS COLLECTION
1.Display
2.Sort the mileage
3.Add
4. Delete
5.Modify
6.Store as a pickle file
7.Load the pickle file
8.Filter the data
9.Export it as a pdf report
10.Exit

Enter the choice : 1
No.    EngNo.    Model      Owner      Type      Mileage    Vendor      RegNo.
1      324942     Polo_GT    Rahul      Hatchback  19.3       volkswagen  2456
2      452313     XZ_Plus_LUX Anju       SUV        31.2       Tata_Motors 8734

VEHICLE DETAILS COLLECTION
1.Display
2.Sort the mileage
3.Add
4. Delete
5.Modify
6.Store as a pickle file
7.Load the pickle file
8.Filter the data
9.Export it as a pdf report
10.Exit

Enter the choice : 3
Enter the details :
Engine No  :324567
Model : MAgnite
Type  : Sport_utility
Mileage : 17
Vendor : Nissan
Registration No. :3290
Owner Name : Anjali

VEHICLE DETAILS COLLECTION
1.Display
2.Sort the mileage
3.Add
4. Delete
5.Modify
6.Store as a pickle file
7.Load the pickle file
8.Filter the data
9.Export it as a pdf report
10.Exit

Enter the choice : 1
No.    EngNo.    Model      Owner      Type      Mileage    Vendor      RegNo.
1      324942     Polo_GT    Rahul      Hatchback  19.3       volkswagen  2456
2      452313     XZ_Plus_LUX Anju       SUV        31.2       Tata_Motors 8734
3      324567     MAgnite    Anjali     Sport_utility 17         Nissan      3290
```

TEST CASES

Test Cases	Description	Input	Expected output	Actual Output	Result
1	Check for the input file vehicledata.pkl	vehicledata.pkl	Successful intake of the file	Successful intake of the file	Pass
2	Check for Load,add,save,report etc of the input file as per the requirement of the user	vehicledta.pkl	Reponding to the input key	Reponding to the input key	pass
3	Check for exit from the program	Choice in the menu	Program execution ends	Program execution ends	Pass

RESULT

Program executed Successfully and the output is obtained.

GIT LINK

[https : //github.com/omals/Python – Lab/tree/main/LabCycle/LabCycle3/question4](https://github.com/omals/Python-Lab/tree/main/LabCycle/LabCycle3/question4)

TKINTER UI APPLICATION

AIM

Design a class to store the details of a vehicle such as engine number, model, type, mileage, vendor, registration number, and owner name. Design another class that holds the details of several vehicles and provide functions to

- Display the details of the collection
- the collection according to mileage
- Add, Delete and Modify the entries from the collection
- Store and Load the collection as a pickle file
- Filter the result according to the attributes and export it as a report.

Convert the above data to UI based application using Tkinter or PyQt

THEORY

- GUI using tkinter or PyQt - Python offers multiple options for developing GUI (Graphical User Interface). Both Tkinter and PyQt are useful for designing acceptable GUI's.

PROGRAM

```
from tkinter import *
from tkinter import ttk

from numpy import delete
from car_class import *
from tkinter import filedialog
import pickle

#Constants
scr = Tk()
scr.geometry("990x400+40+50")
scr.configure(bg = "grey")
scr.title("Vehicle Sale Data")
text= Label(scr, font=('Times New Roman',15,'bold'),text="VEHICLE DATA",bg="grey")
text.pack()
CarData = CarDetails()
DataFrame = LabelFrame(scr,text="Datas",bg = "grey")
DataFrame.pack(expand="yes",side=RIGHT)
```

```
CarDisplay = ttk.Treeview(DataFrame)

def data_screen_frame():
    count=0
    CarDisplay['columns'] = ("EngNo","Model","Type","Mileage","Vendor","RegNo","Owner")
    CarDisplay.column("#0",width=0,minwidth=3)
    CarDisplay.column("EngNo",anchor = W,minwidth=10,width=50)
    CarDisplay.column('Model',anchor = W,minwidth=10,width=50)
    CarDisplay.column('Type',anchor = W,minwidth=10,width=50)
    CarDisplay.column('Mileage',anchor = W,minwidth=20,width=50)
    CarDisplay.column('Vendor',anchor = W,minwidth=20,width=50)
    CarDisplay.column('RegNo',anchor = W,minwidth=20,width=50)
    CarDisplay.column('Owner',anchor = W,minwidth=20,width=50)
    #setting the headings
    CarDisplay.heading("#0",text = " ",anchor=W )
    CarDisplay.heading("EngNo",text = "Engine No.",anchor = W)
    CarDisplay.heading("Model",text = "Model",anchor = W)
    CarDisplay.heading("Type",text ="Type",anchor = W)
    CarDisplay.heading("Mileage",text = "Mileage",anchor = W)
    CarDisplay.heading("Vendor",text = "Vendor",anchor = W)
    CarDisplay.heading("RegNo",text = "Reg. No.",anchor = W)
    CarDisplay.heading("Owner",text = "Owner",anchor = W)
    List_of_cars = CarData.To_write_list
    for i in List_of_cars:
        show_Values = tuple(i)
        CarDisplay.insert(parent = "",index = 'end',values=show_Values)
    CarDisplay.pack()

#Defining Button actions

#Creating a frame for the Input details
In_Frame = LabelFrame(scr,text = "Input",bg = "grey")
In_Frame.pack(fill="x",expand="yes",padx = 20)
In_EngNo = StringVar(None)

#Creating Labels and respective entry boxes
In_EngNo_Label = Label(In_Frame,text = "Engine No",bg='grey')
In_EngNo_Label.grid(row = 0,column=0,padx=10,pady=10)
In_EngNo = Entry(In_Frame)
In_EngNo.grid(row=0,column=1,padx=10,pady=10)

In_Model_Label = Label(In_Frame,text = "Model",bg='grey')
```

```
In_Model_Label.grid(row = 1,column=0,padx=10,pady=10)
In_Model = Entry(In_Frame)
In_Model.grid(row=1,column=1,padx=10,pady=10)

In_Type_Label = Label(In_Frame,text = "Type",bg='grey')
In_Type_Label.grid(row = 2,column=0,padx=10,pady=10)
In_Type = Entry(In_Frame)
In_Type.grid(row=2,column=1,padx=10,pady=10)

In_Mileage_Label = Label(In_Frame,text = "Mileage",bg='grey')
In_Mileage_Label.grid(row = 0,column=2,padx=10,pady=10)
In_Mileage = Entry(In_Frame)
In_Mileage.grid(row=0,column=3,padx=10,pady=10)

In_Vendor_Label = Label(In_Frame,text = "Vendor",bg='grey')
In_Vendor_Label.grid(row = 1,column=2,padx=10,pady=10)
In_Vendor = Entry(In_Frame)
In_Vendor.grid(row=1,column=3,padx=10,pady=10)

In_Regno_Label = Label(In_Frame,text = "Reg No",bg='grey')
In_Regno_Label.grid(row = 2,column=2,padx=10,pady=10)
In_RegNo = Entry(In_Frame)
In_RegNo.grid(row=2,column=3,padx=10,pady=10)

In_Owner_Label = Label(In_Frame,text = "Owner",bg='grey')
In_Owner_Label.grid(row = 4,column=0,padx=10,pady=10)
In_Owner = Entry(In_Frame)
In_Owner.grid(row=4,column=1,padx=10,pady=10)

def clear_inputs():
    #To clear the inputs on screen
    In_EngNo.delete(0,END)
    In_Model.delete(0,END)
    In_Type.delete(0,END)
    In_Mileage.delete(0,END)
    In_Vendor.delete(0,END)
    In_RegNo.delete(0,END)
    In_Owner.delete(0,END)

def add_to_entry_box():
    clear_inputs()
```

```
#Select the record number
sel_record = CarDisplay.focus()
#Selecting values of the record
rec_values = CarDisplay.item(sel_record,'values')
#outputting to entry box
In_EngNo.insert(0,rec_values[0])
In_Model.insert(0,rec_values[1])
In_Type.insert(0,rec_values[2])
In_Mileage.insert(0,rec_values[3])
In_Vendor.insert(0,rec_values[4])
In_RegNo.insert(0,rec_values[5])
In_Owner.insert(0,rec_values[6])

def Delete_record():
    sel_record = CarDisplay.focus()
    rec_values = CarDisplay.item(sel_record,'values')
    #To Remove Data from screen
    to_delete = CarDisplay.selection()[0]
    CarDisplay.delete(to_delete)
    #To remove from CarData
    CarData.Delete_car(rec_values[5])

def Add_the_record():
    add_data = (In_EngNo.get(),In_Model.get(),In_Type.get(),
                int(In_Mileage.get()),In_Vendor.get(),In_RegNo.get(),In_Owner.get())

    to_Add_Car = car(In_EngNo.get(),In_Model.get(),In_Type.get(),int(In_Mileage.get()),
                    ,In_Vendor.get(),In_RegNo.get(),In_Owner.get())
    CarData.add_Car(to_Add_Car)
    CarDisplay.insert(parent = "",index = 'end',values=add_data)
    clear_inputs

def Load_File():
    scr.filename = filedialog.askopenfilename(initialdir="/",
        title="Select Pickle File",filetypes=(("pickle files","*.dat"),#
        ("All Files","*.*")))
    CarData.Load_from_file(scr.filename)
    for i in CarData.To_write_list:
        show_Values = tuple(i)
```

```
        CarDisplay.insert(parent = "",index = 'end',values=show_Values)

def Save_File():
    scr.filename = filedialog.askopenfilename(initialdir="/",
        title="Select pickle File",filetypes=(("File to save","data.dat"),#
            ("All Files","*..*")))
    CarData.Save_Details(scr.filename)

def Sort_mileage():
    #Sorting object data by Mileage
    CarData.Sort_Mileage()
    #Deleting items on window
    for data in CarDisplay.get_children():
        CarDisplay.delete(data)
    for i in CarData.To_write_list:
        show_Values = tuple(i)
        CarDisplay.insert(parent = "",index = 'end',values=show_Values)
def Save_as_pdf():
    scr.filename = filedialog.askopenfilename(initialdir="/",
        title="Select Pdf File",filetypes=(("pdf files","*.pdf"),("All Files","*..*")))
    CarData.Create_report(scr.filename)

def Buttons_Frame():
    ButtonFrame = LabelFrame(scr,text = "Options",bg = "grey")
    ButtonFrame.pack()

    AddRec_button = Button(ButtonFrame,text = "Add",command=Add_the_record,#
        activebackground='#FFA9A9',bg='grey')
    AddRec_button.grid(row=0,column=0,padx=10,pady=10)

    Modify_button = Button(ButtonFrame,text = "Modify",#
        activebackground='#FFA9A9',bg='grey')
    Modify_button.grid(row=1,column=0,padx=10,pady=10)

    Open_button = Button(ButtonFrame,text = "Open File",command=Load_File,#
        activebackground='#FFA9A9',bg='grey')
    Open_button.grid(row=2,column=0,padx=10,pady=10)

    Sort_button = Button(ButtonFrame,text = "Sort by Mileage",command=Sort_mileage,#
        activebackground='#FFA9A9',bg='grey')
    Sort_button.grid(row=0,column=1,padx=10,pady=10)
```



```

Delete_button = Button(ButtonFrame,text = "Delete Entry",command=Delete_record,#
                        activebackground='#FFA9A9',bg='grey')
Delete_button.grid(row=1,column=1,padx=10,pady=10)

Save_button = Button(ButtonFrame,text = "Save Data",command=Save_File,#
                     activebackground='#FFA9A9',bg='grey')
Save_button.grid(row=2,column=1,padx=10,pady=10)

Report_button = Button(ButtonFrame,text = "Save PDF",command=Save_as_pdf,#
                       activebackground='#FFA9A9',bg='grey')
Report_button.grid(row=0,column=3,padx=10,pady=10)

Show_button = Button(ButtonFrame,text = "Show Selection",command=add_to_entry_box,#
                     activebackground='#FFA9A9',bg='grey')
Show_button.grid(row=1,column=3,padx=10,pady=10)

Clear_Button = Button(ButtonFrame,text = "Clear",command=clear_inputs,#
                      activebackground='#FFA9A9',bg='grey')
Clear_Button.grid(row=2,column=3,padx=10,pady=10)

```

```
Buttons_Frame()
```

```
data_screen_frame()
```

```
scr.mainloop()
```

SAMPLE INPUT-OUTPUT

Vehicle Sale Data

VEHICLE DATA

Input

Engine No: 324567 Mileage: 17

Model: Magnite Vendor: Nissan

Type: Sport_utility Reg No: 6790

Owner: Anjali

Options

Add Sort by Mileage Save PDF

Modify Delete Entry Show Selection

Open File Save Data Clear

Datan

Engine	Model	Type	Mileag	Vendo	Reg. N	Owner
32494	Polo_C	Rahul	Hatch	19.3	volksv	2456
45231	XZ_Pl	Anju	SUV	31.2	Tata_M	8734
32456	Magnii	Sport_	17	Nissar	6790	Anjali

TEST CASES

Test Cases	Description	Input	Expected output	Actual Output	Result
1	Check the display of tkinter window	import the tkinter module and the display statements	Successful display of GUI window	Successful display of GUI window	Pass
2	Display and highlight the keys in the window	Display code	Grey color window with the highlight key color of red	Grey color window with the highlight key color of red	pass
3	Check for proper opening of the file from the computer	Selection path	.dat file is opened and listed in the short window	.dat file is opened and listed in the short window	Pass
4	Check for exit from the window	mouse click on the close option	window closed and program execution stops	window closed and program execution stops	Pass

RESULT

Program executed Successfully and the output is obtained.

GIT LINK

<https://github.com/omals/Python-Lab/tree/main/LabCycle/LabCycle3/question5>