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

ABDUL HAKKEEM P A
(80521001)

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 software laboratory record for ***21-805-0106: Python Programming Lab*** *is a record of work carried out by* ***Abdul Hakkeem P A (80521001)****, 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	Dr. Philip Samuel
Assistant Professor	Professor and Head
Department of Computer Science	Department of Computer Science
CUSAT	CUSAT

Table of Contents

OPERATIONS ON 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

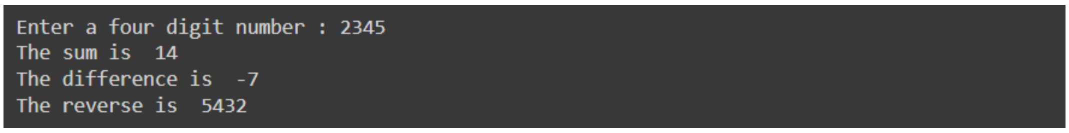
input () , Strings , Arithmetic operators

PROGRAM

```
#function to extract digits of the number.
def extractDigits(num):
  #extracting each digits into variables digit_1 to digit_4 respectively
  digit_4 = num%10
  num = num//10
  digit_3 = num%10
  num = num//10
  digit_2 = num%10
  num = num//10
  digit_1 = num%10
  num = num//10
  return digit_1,digit_2,digit_3,digit_4

#function to calculate the sum of digits.
def sumOfDigits(digit_1,digit_2,digit_3,digit_4):
  sum = 0
  #sum of each digits stored to variable sum
  sum = digit_1 + digit_2 + digit_3 + digit_4
  print("The sum is ",sum)

#function to reverse the number.
def reverseNumber(digit_1,digit_2,digit_3,digit_4):
  #reverse of the number stored to variable reverse
  reverse = (digit_4*1000) + (digit_3*100) + (digit_2*10) + digit_1
  print("The reverse is ",reverse)

#function to the compute the difference between the products.
def diffOfProductofDigits(digit_1,digit_2,digit_3,digit_4):
  #product of digits at odd positions are stored to product_of_odd
```

```
    product_of_odd = digit_1*digit_3
    #product of digits at even positions are stored to product_of_even
    product_of_even = digit_2*digit_4
    #the difference of product of odd and even is stored to difference
    difference = product_of_odd - product_of_even
    print("The difference is ",difference)

num = int(input("Enter a four digit number : "))
digit1,digit2,digit3,digit4 = extractDigits(num)
sumOfDigits(digit1,digit2,digit3,digit4)
diffOfProductofDigits(digit1,digit2,digit3,digit4)
reverseNumber(digit1,digit2,digit3,digit4)
```

SAMPLE INPUT-OUTPUT



```
Enter a four digit number : 2345
The sum is  14
The difference is  -7
The reverse is  5432
```

TEST CASES

Test Case No:	Test Case Description	Test Data	Expected Result	Actual Result	Pass/Fail
1	Checking with random number	num = 1547	Sum is 17 Difference is -31 Reverse is 7451	Sum is 17 Difference is -31 Reverse is 7451	Pass

GITHUB LINK

Click Here for the Code.

AREA AND PERCENTAGE CALCULATION

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.

THEORY

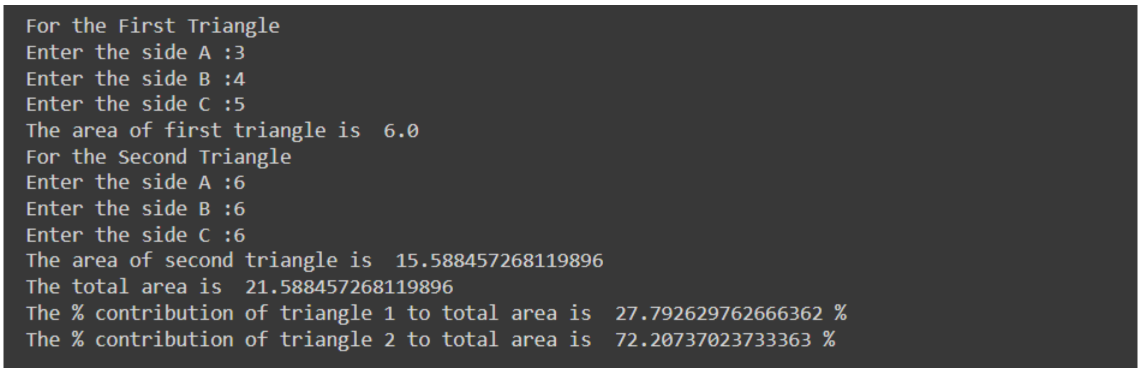
Datatype , Functions , Expressions ,Built-in functions

PROGRAM

```
#function - to get the sides from the user.
def get_sides():
  a = float(input("Enter the side A :"))
  b = float(input("Enter the side B :"))
  c = float(input("Enter the side C :"))
  return a,b,c

#function - to calculate the area of the triangle.
def calculate_area(a,b,c):
  s = (a+b+c)/2
  area = (s*(s-a)*(s-b)*(s-c))**(1/2)
  return area

#function - to calculate the total area and % contribution of each triangle.
def totalAreaAndContribution(area_1,area_2):
  #total area calculation
  total_area = area_1 + area_2
  #calculating the contribution of each triangle to area
  contribution_of_triangle1 = (100*area_1)/total_area
  contribution_of_triangle2 = (100*area_2)/total_area
  print("The total area is ",total_area)
  print("The % contribution of triangle 1 to total area is ",round
  (contribution_of_triangle1,2),"%")
  print("The % contribution of triangle 2 to total area is ",round
  (contribution_of_triangle2,2),"%")

print("First Triangle")
a1,b1,c1 = get_sides()
```

```
area_1 = calculate_area(a1,b1,c1)
print("The area of first triangle is ",area_1)

print("Second Triangle")
a2,b2,c2 = get_sides()
area_2 = calculate_area(a2,b2,c2)
print("The area of second triangle is ",area_2)
totalAreaAndContribution(area_1,area_2)
```

SAMPLE INPUT-OUTPUT



```
For the First Triangle
Enter the side A :3
Enter the side B :4
Enter the side C :5
The area of first triangle is  6.0
For the Second Triangle
Enter the side A :6
Enter the side B :6
Enter the side C :6
The area of second triangle is  15.588457268119896
The total area is  21.588457268119896
The % contribution of triangle 1 to total area is  27.792629762666362 %
The % contribution of triangle 2 to total area is  72.20737023733363 %
```

TEST CASES

Test Case No:	Test Case Description	Test Data	Expected Result	Actual Result	Pass /Fail
1	Checking with random measurments	a1 =3,b1=4, c1 =5 a2 = 6, b2=6, c2 = 6	Area 1 = 6 Area 2 = 15.58 Total Area = 21.58 % Contribution of Triangle 1 = 27.79% % Contribution of Triangle 2 = 72.21%	Area 1 = 6 Area 2 = 15.58 Total Area = 21.58 % Contribution of Triangle 1 = 27.79% % Contribution of Triangle 2 = 72.21%	Pass

GITHUB LINK

Click Here for the Code

SALARY SLIP 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.

THEORY

Conditional Branching

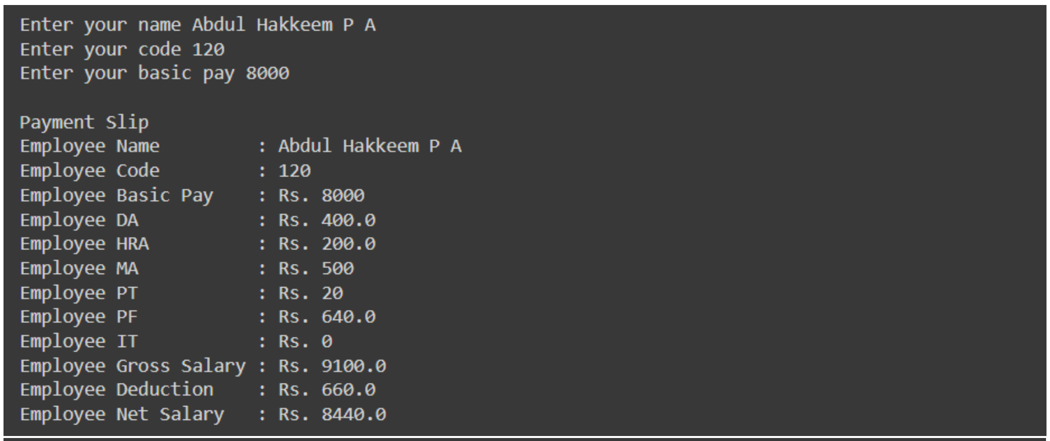
PROGRAM

```
#function - to calculate gross salary , arguements : Basic Pay , DA ,HRA , MA
def gross_salay(bp,da,hra,ma):
  grossSalary = bp+da+hra+ma
  return grossSalary

#function - to calculate deduction , arguements : Pf, Pt,It
def deduction(pf,pt,it):
  deduct = pf+pt+it
  return deduct

#function - to calculate net salary , arguements : Gross Salary and Deduction
def net_salary(GS,D):
  net_sal = GS - D
  return net_sal

#function to generate the payment slip.
def generatePaymentSlip(name,code,basic_pay):
  #if basic pay is less than 10,000.
  if basic_pay<10000:
    DA = (basic_pay*5)/100
    HRA = (basic_pay*2.5)/100
    MA = 500
    PT = 20
    PF = (basic_pay*8)/100
    IT = 0
    Gross_Salary = gross_salay(basic_pay,DA,HRA,MA)
    Deduction = deduction(PF,PT,IT)
    Net_Salary = net_salary(Gross_Salary,Deduction)
```

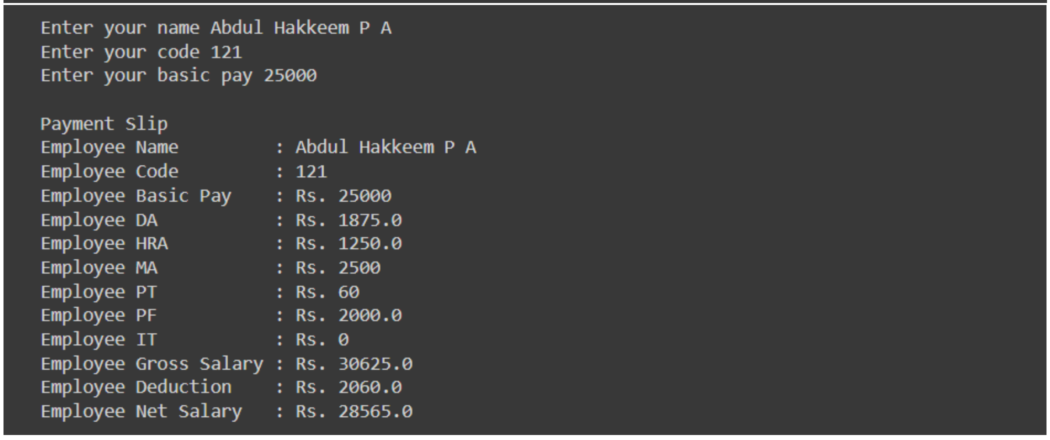
```
#if basic pay is greater than 10,000 and less than 30,000.
elif basic_pay>=10000 and basic_pay<30000:
  DA = (basic_pay*7.5)/100
  HRA = (basic_pay*5)/100
  MA = 2500
  PT = 60
  PF = (basic_pay*8)/100
  IT = 0
  Gross_Salary = gross_salay(basic_pay,DA,HRA,MA)
  Deduction = deduction(PF,PT,IT)
  Net_Salary = net_salary(Gross_Salary,Deduction)

#if basic pay is greater than 30,000 and less than 50,000.
elif basic_pay>=30000 and basic_pay<50000:
  DA = (basic_pay*11)/100
  HRA = (basic_pay*7.5)/100
  MA = 5000
  PT = 60
  PF = (basic_pay*11)/100
  IT = (basic_pay*11)/100
  Gross_Salary = gross_salay(basic_pay,DA,HRA,MA)
  Deduction = deduction(PF,PT,IT)
  Net_Salary = net_salary(Gross_Salary,Deduction)

#if the basic pay is above 50,000.
else:
  DA = (basic_pay*25)/100
  HRA = (basic_pay*11)/100
  MA = 7000
  PT = 80
  PF = (basic_pay*12)/100
  IT = (basic_pay*20)/100
  Gross_Salary = gross_salay(basic_pay,DA,HRA,MA)
  Deduction = deduction(PF,PT,IT)
  Net_Salary = net_salary(Gross_Salary,Deduction)

#printing the payment slip
print("\nPayment Slip")
print("Employee Name         : "+name)
print("Employee Code         :",code)
print("Employee Basic Pay    : Rs.",basic_pay)
```

```
  print("Employee DA           : Rs.",DA)
  print("Employee HRA          : Rs.",HRA)
  print("Employee MA           : Rs.",MA)
  print("Employee PT           : Rs.",PT)
  print("Employee PF           : Rs.",PF)
  print("Employee IT           : Rs.",IT)
  print("Employee Gross Salary : Rs.",Gross_Salary)
  print("Employee Deduction    : Rs.",Deduction)
  print("Employee Net Salary   : Rs.",Net_Salary)


EmployeeName = input("Enter your name ")
EmployeeCode = int(input("Enter your code "))
basic_pay = int(input("Enter your basic pay "))
generatePaymentSlip(EmployeeName,EmployeeCode,basic_pay)
```

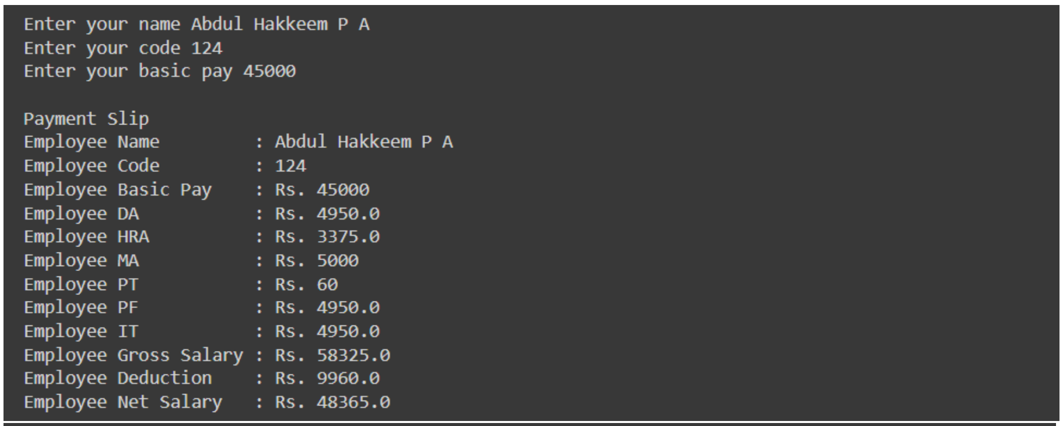
SAMPLE INPUT-OUTPUT



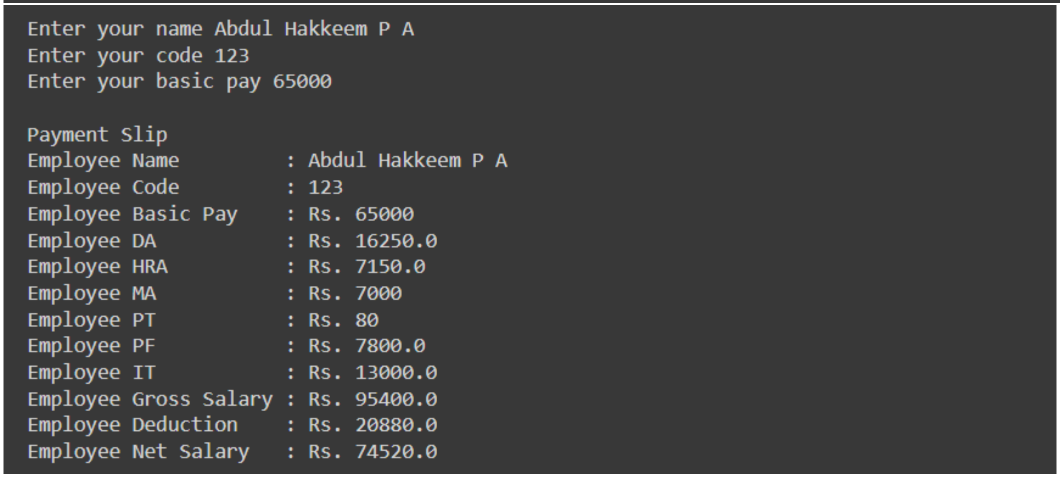
```
Enter your name Abdul Hakkeem P A
Enter your code 120
Enter your basic pay 8000

Payment Slip
Employee Name         : Abdul Hakkeem P A
Employee Code         : 120
Employee Basic Pay    : Rs. 8000
Employee DA           : Rs. 400.0
Employee HRA          : Rs. 200.0
Employee MA           : Rs. 500
Employee PT           : Rs. 20
Employee PF           : Rs. 640.0
Employee IT           : Rs. 0
Employee Gross Salary : Rs. 9100.0
Employee Deduction    : Rs. 660.0
Employee Net Salary   : Rs. 8440.0
```



```
Enter your name Abdul Hakkeem P A
Enter your code 121
Enter your basic pay 25000

Payment Slip
Employee Name         : Abdul Hakkeem P A
Employee Code         : 121
Employee Basic Pay    : Rs. 25000
Employee DA           : Rs. 1875.0
Employee HRA          : Rs. 1250.0
Employee MA           : Rs. 2500
Employee PT           : Rs. 60
Employee PF           : Rs. 2000.0
Employee IT           : Rs. 0
Employee Gross Salary : Rs. 30625.0
Employee Deduction    : Rs. 2060.0
Employee Net Salary   : Rs. 28565.0
```



```
Enter your name Abdul Hakkeem P A
Enter your code 124
Enter your basic pay 45000

Payment Slip
Employee Name         : Abdul Hakkeem P A
Employee Code         : 124
Employee Basic Pay    : Rs. 45000
Employee DA           : Rs. 4950.0
Employee HRA          : Rs. 3375.0
Employee MA           : Rs. 5000
Employee PT           : Rs. 60
Employee PF           : Rs. 4950.0
Employee IT           : Rs. 4950.0
Employee Gross Salary : Rs. 58325.0
Employee Deduction    : Rs. 9960.0
Employee Net Salary   : Rs. 48365.0
```



```
Enter your name Abdul Hakkeem P A
Enter your code 123
Enter your basic pay 65000

Payment Slip
Employee Name         : Abdul Hakkeem P A
Employee Code         : 123
Employee Basic Pay    : Rs. 65000
Employee DA           : Rs. 16250.0
Employee HRA          : Rs. 7150.0
Employee MA           : Rs. 7000
Employee PT           : Rs. 80
Employee PF           : Rs. 7800.0
Employee IT           : Rs. 13000.0
Employee Gross Salary : Rs. 95400.0
Employee Deduction    : Rs. 20880.0
Employee Net Salary   : Rs. 74520.0
```

TEST CASES

Test Case No:	Test Case Description	Test Data	Expected Result	Actual Result	Pass /Fail
1	Basic Pay below 10,000	Basic Pay = 8000	DA = 400, HRA = 200, MA =500 PT = 20, PF = 640 IT = 0 , GS = 9100 Deduction = 660 Net Salary = 8440	DA = 400, HRA = 200, MA =500 PT = 20, PF = 640 IT = 0 , GS = 9100 Deduction = 660 Net Salary = 8440	Pass
2	Basic Pay below 30,000 and above 10,000	Basic Pay = 25000	DA = 1875, HRA = 1250, MA =2500 PT = 60, PF = 2000 IT = 0 , GS = 30625 Deduction = 2060 Net Salary = 28565	DA = 1875, HRA = 1250, MA =2500 PT = 60, PF = 2000 IT = 0 , GS = 30625 Deduction = 2060 Net Salary = 28565	Pass

3	Basic Pay below 50,000 and above 30,000	Basic Pay = 45000	DA = 4950, HRA = 3375, MA =5000 PT = 600, PF = 4950 IT = 4950 ,GS = 58325 Deduction = 9960 Net Salary = 48365	DA = 4950, HRA = 3375, MA =5000 PT = 600, PF = 4950 IT = 4950 ,GS = 58325 Deduction = 9960 Net Salary = 48365	**Pass**
4	Basic Pay above 50,000	Basic Pay = 65000	DA = 16250, HRA = 7150, MA =7000 PT = 80, PF = 7800 IT =13000 ,GS = 95400 Deduction = 20880 Net Salary = 74520	DA = 16250, HRA = 7150, MA =7000 PT = 80, PF = 7800 IT =13000 ,GS = 95400 Deduction = 20880 Net Salary = 74520	**Pass**

GITHUB LINK

Click Here for the Code

HAPPY NUMBER

AIM

Develop a program to perform the following task: a. Define a function to check whether a number is happy or not. b. Define a function to print all happy numbers within a range. c. Define a function to print first N happy numbers.

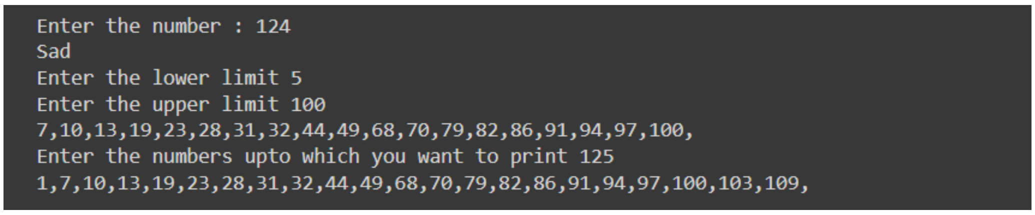
THEORY

Loops – for, while , Nested loops

PROGRAM

```
#function - to print happy number's between a range
def print_in_a_range(l,u):
  for i in range(l,u+1):
    #function - check_happy is used to check whether i is a happy number or not
    isHappy = check_happy(i)
    if isHappy:
      print(i,end=",")

#function - to print happy number's upto to a number
def print_upto_n(n):
  for i in range(1,n+1):
    #function - check_happy is used to check whether i is a happy number or not
    isHappy = check_happy(i)
    if isHappy:
      print(i, end=",")



#function - check whether the number is happy or not
def check_happy(num):
  for i in range(0,101):
    happy_sum = 0
    while num>0:
      remainder = num % 10
      num = num //10
      happy_sum = happy_sum + remainder**2

    if(happy_sum == 1):
      #returns true if the sum of the squares of its digit is 1
      return True
    else:
      #if the sum is not 1 , then the sum is again passed to the loop
      num = happy_sum
      #after 100 iterations , the number is declared as sad
      if(i == 100):
        return False



#to - check whether a number is happy or not
num = int(input("Enter the number : "))
result = check_happy(num)
if result:
  print("Happy")
else:
  print("Sad")

#to print happy numbers between a range
lower_range = int(input("Enter the lower limit "))
upper_range = int(input("Enter the upper limit "))
print_in_a_range(lower_range,upper_range)

#to print happy numbers upto to n value
limit = int(input("\nEnter the numbers upto which you want to print "))
print_upto_n(limit)
```

SAMPLE INPUT-OUTPUT



```
Enter the number : 124
Sad
Enter the lower limit 5
Enter the upper limit 100
7,10,13,19,23,28,31,32,44,49,68,70,79,82,86,91,94,97,100,
Enter the numbers upto which you want to print 125
1,7,10,13,19,23,28,31,32,44,49,68,70,79,82,86,91,94,97,100,103,109,
```

TEST CASES

Test Case No:	**Test Case Description**	**Test Data**	**Expected Result**	**Actual Result**	**Pass /Fail**
1	Checking with random inputs	number =11	Sad	Sad	**Pass**
2	Checking with random inputs	number =13	Happy	Happy	**Pass**

GITHUB LINK

Click Here for the Code

STRING OPERATIONS

AIM

Develop a program to read a string and perform the following operations: • Print all possible substrings. • Print all possible substrings of length K. • Print all possible substrings of length K with N distinct characters. • Print substring(s) of length maximum length with N distinct characters. • Print all palindrome substrings.

THEORY

Strings, String functions, Slicing

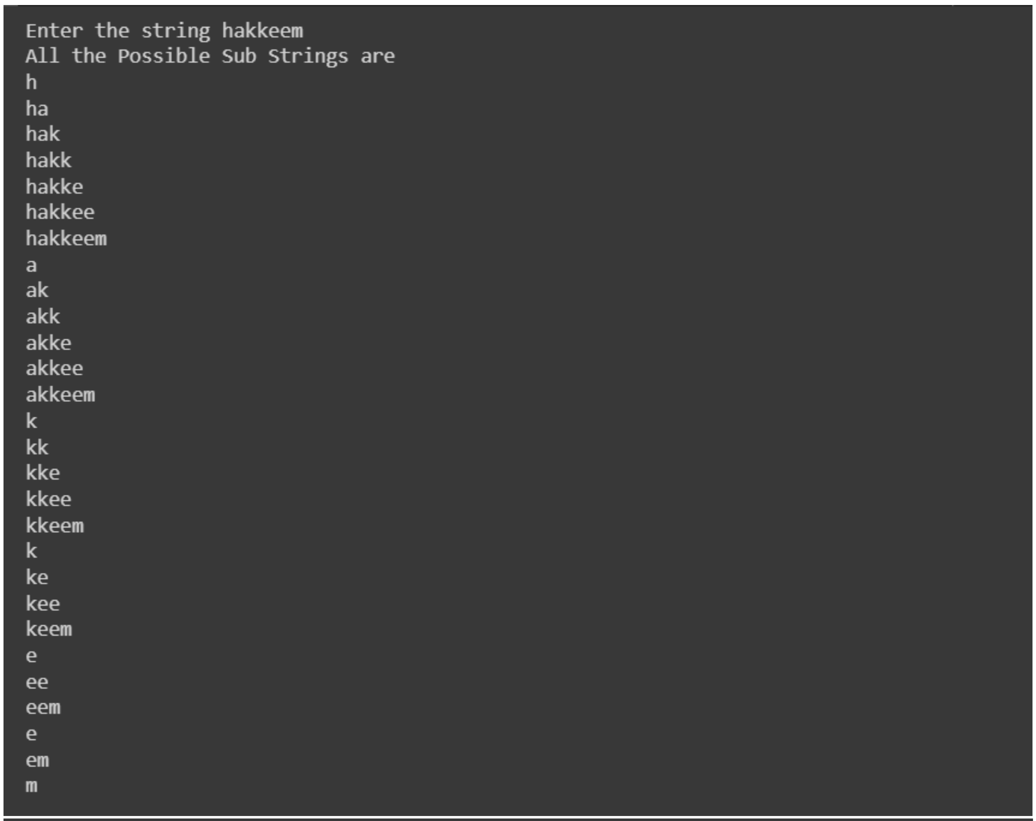
PROGRAM

```
#function - to print all possible substrings
def sub_strings(name):
  for i in range(0,len(name)+1):
    for j in range(i+1,len(name)+1):
      s = name[i:j]
      print(s, end=",")

#function - to print all possible substrings with length specified
def sub_strings_with_length(name,size):
  for i in range(0,len(name)+1):
    for j in range(i+1,len(name)+1):
      s = name[i:j]
      if len(s)==size:
        print(s,end=" , ")

#function - to print all possible substrings with length and no of distinct
characters specified
def sub_strings_with_length_with_N_characters(name,size,N):
  for i in range(0,len(name)+1):
    for j in range(i+1,len(name)+1):
      s = name[i:j]
      if len(s)==size:
        distinct = set(s)
        if len(distinct) == N:
          print(s,end=" , ")

#function - to print all possible substrings with max length and N no of
distinct characters
```

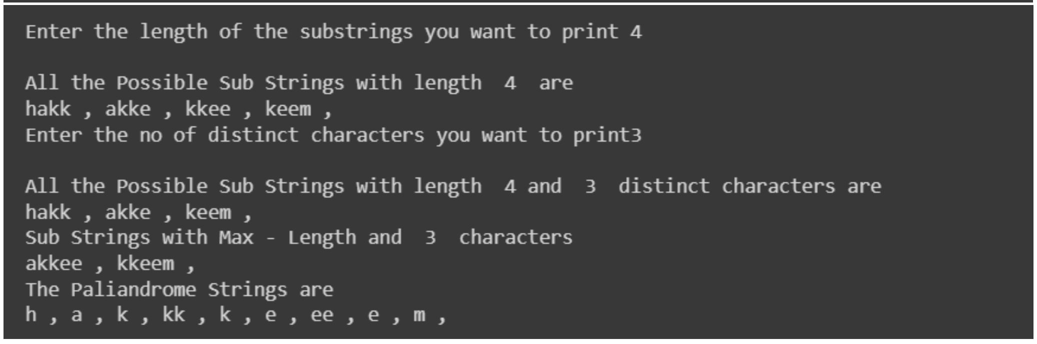
```
def sub_strings_with_max_length_with_N_characters(name,N):
  string_list = []
  for i in range(0,len(name)+1):
    for j in range(i+1,len(name)+1):
      s = name[i:j]
      distinct = set(s)
      if len(distinct) == N:
        string_list.append(s)

  length = len(max(string_list,key = len))
  for i in string_list:
    if len(i)==length:
      print(i,end=" , ")

#function - to print all the paliandrome substrings
def print_paliandrome(name):
  for i in range(0,len(name)+1):
    for j in range(i+1,len(name)+1):
      s = name[i:j]
      reverse = s[::-1]
      if reverse == s:
        print(s , end = " , ")



name = input("Enter the string ")
print("All the Possible Sub Strings are")
sub_strings(name) #prints all possible sub-strings

length = int(input("\nEnter the length of the substrings you want to print "))
#prints all possible sub-strings with length specified
print("\nAll the Possible Sub Strings with length ",length," are")
sub_strings_with_length(name,length)

num_of_distinct = int(input("\nEnter the no of distinct characters you want to
print "))
#prints all possible sub-strings with length and no of distinct characters
print("\nAll the Possible Sub Strings with length ",length,"and ",num_of_distinct,"
distinct characters are")
sub_strings_with_length_with_N_characters(name,length,num_of_distinct)

#prints all sub-strings with max-length and no of distinct characters given
```

```
print("\nSub Strings with Max - Length and ",num_of_distinct," characters")
sub_strings_with_max_length_with_N_characters(name,num_of_distinct)

#prints all the paliandrome sub-strings
print("\nThe Paliandrome Strings are ")
print_paliandrome(name)
```

SAMPLE INPUT-OUTPUT



```
Enter the string hakkeem
All the Possible Sub Strings are
h
ha
hak
hakk
hakke
hakkee
hakkeem
a
ak
akk
akke
akkee
akkeem
k
kk
kke
kkee
kkeem
k
ke
kee
keem
e
ee
eem
e
em
m
```



```
Enter the length of the substrings you want to print 4

All the Possible Sub Strings with length  4  are
hakk , akke , kkee , keem ,
Enter the no of distinct characters you want to print3

All the Possible Sub Strings with length  4 and  3  distinct characters are
hakk , akke , keem ,
Sub Strings with Max - Length and  3  characters
akkee , kkeem ,
The Paliandrome Strings are
h , a , k , kk , k , e , ee , e , m ,
```

TEST CASES

Test Case No:	Test Case Description	Test Data	Expected Result	Actual Result	Pass /Fail
1	Checking with random strings	string = ”abcde”	Sub Strings with length 3 = abc,bcd cde	Sub Strings with length 3 = abc,bcd cde	**Pass**

GITHUB LINK

Click Here for the Code

RABBIT PAIR

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.

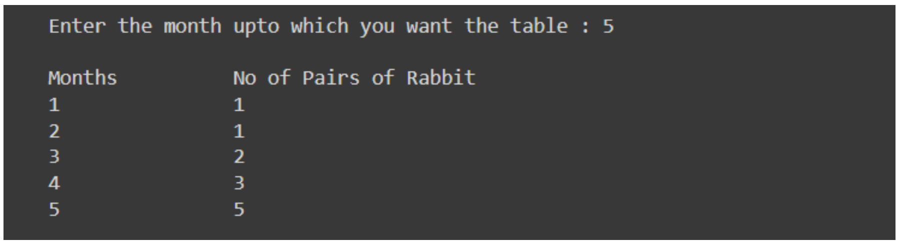
THEORY

Critical thinking, Loops, formatted io

PROGRAM

```
#function to print rabbit pairs.
def printRabbitPairs(n):
  #adding the first no of pairs of rabbit to the list
  rabbit_pair = [1,1]
  print("\nMonths\t\tNo of Pairs of Rabbit")
  #loop used to iterate upto n-months
  for i in range(0,n):
    #printing the month
    print(i+1,end="\t\t")
    #printing the pair of rabbits
    print(rabbit_pair[i])
    rabbit_pair.append(rabbit_pair[i]+rabbit_pair[i+1])

noOfMonths = int(input("Enter the month upto which you want the table : "))
printRabbitPairs(noOfMonths)
```

SAMPLE INPUT-OUTPUT



```
Enter the month upto which you want the table : 5

Months          No of Pairs of Rabbit
1               1
2               1
3               2
4               3
5               5
```

TEST CASES

Test Case No:	**Test Case Description**	**Test Data**	**Expected Result**	**Actual Result**	**Pass /Fail**
1	Checking with random input	month = 8	rabbit pairs = 21	rabbit pairs = 21	**Pass**

GITHUB LINK

Click Here for the Code

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$\hat{2}$-x with each element in the final list 5. After sorting them individually, merge the two lists to create a single sorted list.

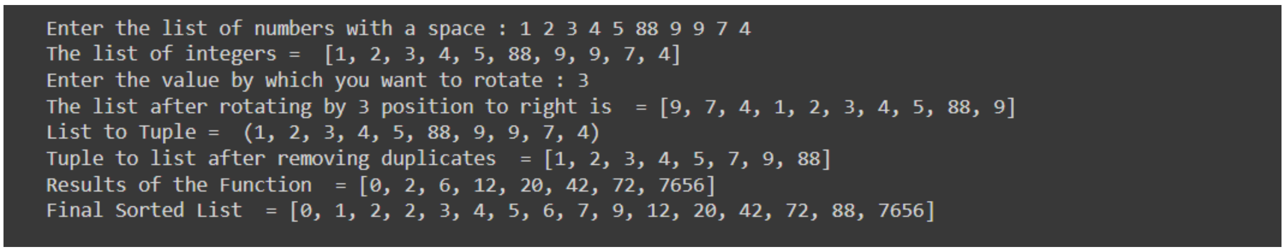
THEORY

List, tuple, set, list comprehension

PROGRAM

```
def convertListToInt(stringList):
  int_list = [int(i) for i in stringList]
  return int_list

def rotateElements(targetList):
  #1.rotate the elements by 'k' position to right
  k = int(input("Enter the value by which you want to rotate : "))
  print("The list after rotating by",k,"position to right is ",end = " = ")
  print(targetList[-k:]+targetList[:-k])

def listToTuple(int_list):
  #2.converting the list into a tuple
  int_tuple = *(i for i in int_list),
  print("List to Tuple = ",int_tuple)
  return int_tuple

def removeDuplicates(int_tuple):
  #3.removing all duplicates
  int_tuple = tuple(set(int_tuple))
  int_list = list(int_tuple)
  return int_list

def mapToFunction(int_list):
  #4.mapping the list to the function f(x) = x^2-x
  square = [(i**2)-i for i in int_list]
  print("Results of the Function f(x) = (x^2-1) = ",square)
  return square

def singleSortedList(int_list,function_list):
  #5.merging the list in 4 and 5 into a single sorted list
  sorted_list = int_list + function_list
  sorted_list.sort()
  print("Final Sorted List = ",sorted_list)

number_string = input("Enter the list of numbers with a space : ")
#list of strings
list_numbers = list(number_string.split(" "))

#converting the list of string to list of integers.
int_list = convertListToInt(list_numbers)
print("The list of integers = ",int_list)
#rotating elements.
rotateElements(int_list)
#converting list to tuple.
int_tuple = listToTuple(int_list)
int_list = removeDuplicates(int_tuple)
print("Tuple to list after removing duplicates = ",int_list)
functionResult = mapToFunction(int_list)
singleSortedList(int_list,functionResult)
```

SAMPLE INPUT-OUTPUT



```
Enter the list of numbers with a space : 1 2 3 4 5 88 9 9 7 4
The list of integers =  [1, 2, 3, 4, 5, 88, 9, 9, 7, 4]
Enter the value by which you want to rotate : 3
The list after rotating by 3 position to right is  = [9, 7, 4, 1, 2, 3, 4, 5, 88, 9]
List to Tuple =  (1, 2, 3, 4, 5, 88, 9, 9, 7, 4)
Tuple to list after removing duplicates  = [1, 2, 3, 4, 5, 7, 9, 88]
Results of the Function  = [0, 2, 6, 12, 20, 42, 72, 7656]
Final Sorted List  = [0, 1, 2, 2, 3, 4, 5, 6, 7, 9, 12, 20, 42, 72, 88, 7656]
```

TEST CASES

Test Case No:	Test Case Description	Test Data	Expected Result	Actual Result	Pass /Fail
1	Checking with random inputs	[1,1,1,2,3 5,7,9]	final sorted list = [0,1,2,2,3,5,6,7, 20,42,72]	final sorted list = [0,1,2,2,3,5,6,7, 20,42,72]	**Pass**

GITHUB LINK

Click Here for the Code

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

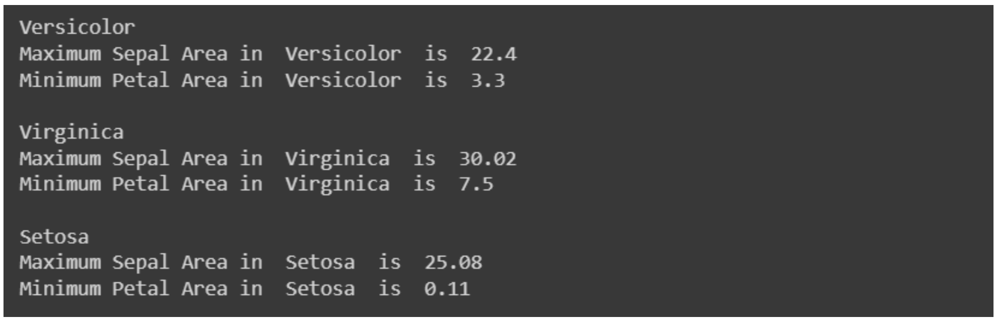
THEORY

JSON, dictionary

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()
  for i in jsonList:
    #appeding the different species name to the list.
    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()
  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 ",round(max(sepalArea),2))
    print("Minimum Petal Area in ",i.capitalize()," is ",round(min(petalArea),2))
    sepalArea.clear()
    petalArea.clear()

def sortTotalArea(jsonList):
  for i in jsonList:
    #adding total area to the each dictionary
    totalArea = (i['petalLength']*i['petalWidth'])+(i['sepalLength']*i['sepalWidth'])
    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)

#path where target file is stored.
filePath = 'iris.json'
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)
sepalAreaAndPetalArea(jsonData)
sortTotalArea(jsonData)
```

SAMPLE INPUT-OUTPUT



```
Versicolor
Maximum Sepal Area in  Versicolor  is  22.4
Minimum Petal Area in  Versicolor  is  3.3

Virginica
Maximum Sepal Area in  Virginica  is  30.02
Minimum Petal Area in  Virginica  is  7.5

Setosa
Maximum Sepal Area in  Setosa  is  25.08
Minimum Petal Area in  Setosa  is  0.11
```

TEST CASES

Test Case No:	Test Case Description	Test Data	Expected Result	Actual Result	Pass /Fail
1	Reading each line as elements of list	iris.json file	List with each line as list element	List with each line as list element	Pass
2	Printing particular details of a json file	Minimum petal area and maximum sepal area in each species	Minimum petal area and maximum sepal area in each species	Minimum petal area and maximum sepal area in each species	Pass
3	Sort the list according to total area	List of dictionary of json data	Sorted list according to total area	Sorted list according to total area	Pass

GITHUB LINK

Click Here for the Code

LIST OF N BOXES

AIM

Write a program to create a class Box with data members length, breadth, height, area, and volume. Provider 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.

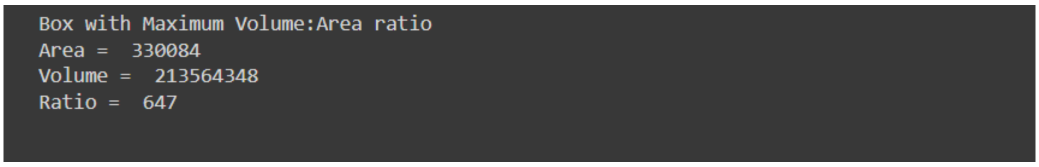
THEORY

Class, objects, constructor

PROGRAM

```
import random

class Box:
  def __init__(self,*args):
    #if all arguements are same,cube prism is initialised.
    if (args[0]==args[1]==args[2]):
      self.__length = args[0]
      self.__breadth = args[0]
      self.__height = args[0]
    #if any two arguements are same, the square prism is initialised
    elif (args[0]==args[1] or args[1]==args[2] or args[0]==args[2]):
      if (args[0]==args[1]):
        self.__length = args[0]
        self.__breadth = args[0]
        self.__height = args[1]
      elif (args[1]==args[2]):
        self.__length = args[1]
        self.__breadth = args[1]
        self.__height = args[0]
      elif (args[0]==args[2]):
        self.__length = args[0]
        self.__breadth = args[0]
        self.__height = args[1]
    #if three arguements are different , rectangular prism is initialised.
    else:
      self.__length = args[0]
      self.__breadth = args[1]
      self.__height = args[2]

  #funtion to calculate area
  def calc_area(self):
    self.__area = self.__length*self.__breadth
    return self.__area
  #funtion to calculate volume
  def calc_volume(self):
    self.__volume = self.__length*self.__breadth*self.__height
    return self.__volume

N = 10
box = [Box(random.randint(1,1000),random.randint(1,1000),random.randint(1,1000)) for i in
area = [i.calc_area() for i in box]
volume = [i.calc_volume() for i in box]
ratio = [x//y for x,y in zip(volume,area)]
index = ratio.index(max(ratio))
print("Box with Maximum Volume:Area ratio")
print("Area = ",area[index])
print("Volume = ",volume[index])
print("Ratio = ",max(ratio))
```

SAMPLE INPUT-OUTPUT



```
Box with Maximum Volume:Area ratio
Area =  330084
Volume =  213564348
Ratio =  647
```

TEST CASES

Test Case No:	Test Case Description	Test Data	Expected Result	Actual Result	Pass /Fail
1	List of Boxes with random measurements of box	Random numbers passed to constructor	Random numbers generated	Random numbers generated	Pass

GITHUB LINK

Click Here for the Code

AREA CALCULATION

AIM

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

THEORY

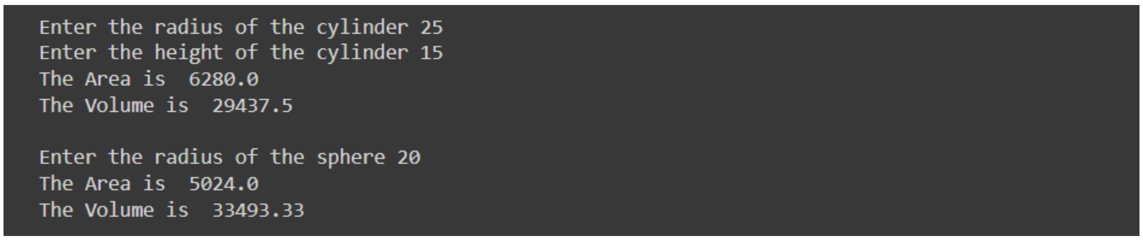
Inheritance

PROGRAM

```
#base - class
class threeD_Shapes:
  def printVolume(self):
    print("The Volume is ",round(self._volume,3))
  def printArea(self):
    print("The Area is ",round(self._area,3))

#derived class - for calculation of area and volume of cylinder
class cylinder(threeD_Shapes):
  #initialisation using constructor
  def __init__(self,r,h):
    self.__radius = r
    self.__height = h
  def calc_area(self):
    self._area = 2*3.14*self.__radius*(self.__radius+self.__height)
  def calc_volume(self):
    self._volume = round((3.14*self.__radius*self.__radius*self.__height),2)

#derived class - for calculation of area and volume of sphere
class sphere(threeD_Shapes):
    #initialisation using constructor
    def __init__(self,r):
      self.__radius = r
    def calc_area(self):
      self._area = 4*3.14*self.__radius*self.__radius
    def calc_volume(self):
      self._volume = round(((4/3)*3.14*(self.__radius**3)),2)
```

```
#object of class cylinder for calculating the area and volume of cylinder
cyl_obj = cylinder(int(input("Enter the radius of the cylinder ")),
int(input("Enter the height of the cylinder ")))
cyl_obj.calc_area()
cyl_obj.printArea()
cyl_obj.calc_volume()
cyl_obj.printVolume()

#object of class sphere for calculating the area and volume of sphere
sph_obj = sphere(int(input("\nEnter the radius of the sphere ")))
sph_obj.calc_area()
sph_obj.calc_volume()
sph_obj.printArea()
sph_obj.printVolume()
```

SAMPLE INPUT-OUTPUT



```
Enter the radius of the cylinder 25
Enter the height of the cylinder 15
The Area is  6280.0
The Volume is  29437.5

Enter the radius of the sphere 20
The Area is  5024.0
The Volume is  33493.33
```

TEST CASES

Test Case No:	Test Case Description	Test Data	Expected Result	Actual Result	Pass /Fail
1	Checking with random inputs	radius and height of cylinder = 25 , 15 radius of sphere =20	cylinder area= 6280 cylinder volume= 29437.5 sphere area= 5824 sphere volume = 33493.33	cylinder area= 6280 cylinder volume= 29437.5 sphere area= 5824 sphere volume = 33493.33	Pass

GITHUB LINK

Click Here for the Code

TIC TAC TOE

AIM

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

THEORY

Pygame library

PROGRAM

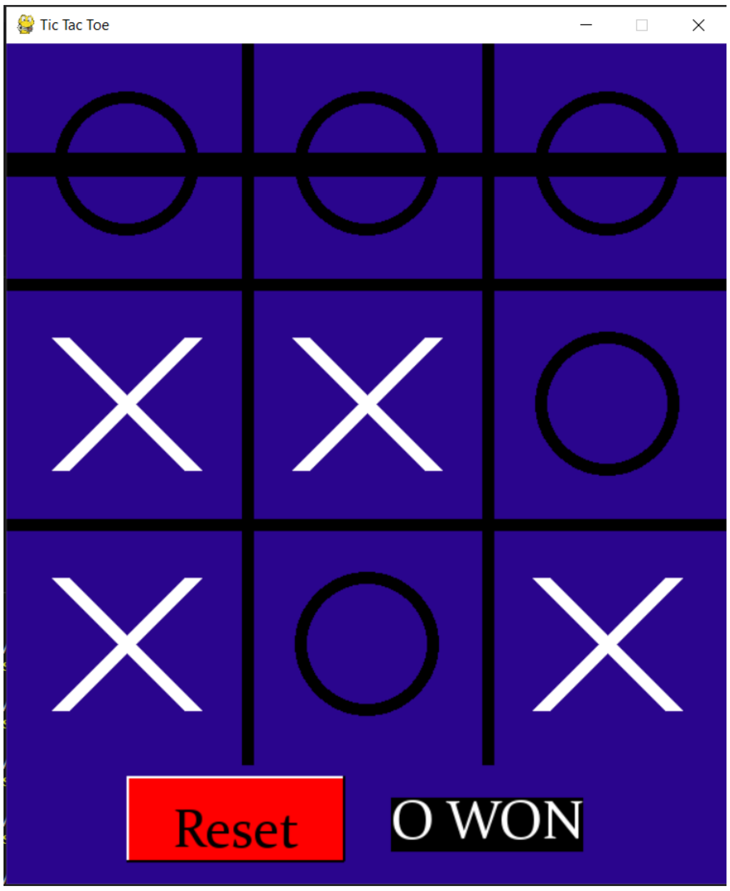
```
import pygame,sys
import numpy as np
from pygame.locals import *

pygame.init()

font = pygame.font.SysFont('Constantia', 45)

#define colours
red = (255, 0, 0)
black = (0, 0, 0)
white = (255, 255, 255)

class button():
     #colours for button and text
     button_col = (255, 0, 0)
     text_col = black
     width = 180
     height = 70

     def __init__(self, x, y, text):
         self.x = x
         self.y = y
         self.text = text

    def draw_button(self):
        button_rect = Rect(self.x, self.y, self.width, self.height)
        pygame.draw.rect(screen, self.button_col, button_rect)
        #add shading to button
        pygame.draw.line(screen, white, (self.x, self.y), (self.x
        + self.width, self.y), 2)
        pygame.draw.line(screen, white, (self.x, self.y),
        (self.x, self.y + self.height), 2)
        pygame.draw.line(screen, black, (self.x, self.y + self.height),
        (self.x + self.width, self.y + self.height), 2)
        pygame.draw.line(screen, black, (self.x + self.width, self.y),
        (self.x + self.width, self.y + self.height), 2)
        #add text to button
        text_img = font.render(self.text, True, self.text_col)
        text_len = text_img.get_width()
        screen.blit(text_img, (self.x + int(self.width / 2) -
        int(text_len / 2), self.y + 25))

WIDTH = 600
boardHeight = 600
HEIGHT = 700
BLACK = (0, 0, 0)
RED = (0, 0, 0)
WHITE = (255,255,255)
BG_COLOR = (42, 5, 141)
LINE_WIDTH = 10
BOARD_ROWS = 3
BOARD_COLUMNS = 3
CIRCLE_RADIUS = 60
CIRCLE_WIDTH = 10
SPACE = 45

screen = pygame.display.set_mode((WIDTH,HEIGHT))
pygame.display.set_caption('Tic Tac Toe')
screen.fill(BG_COLOR)

#board
board = np.zeros((BOARD_ROWS,BOARD_COLUMNS))

def draw_lines():
  #horizontal - lines
  pygame.draw.line(screen,BLACK,(0,200),(600,200),LINE_WIDTH)
  pygame.draw.line(screen,BLACK,(0,400),(600,400),LINE_WIDTH)
  #vertical - lines
  pygame.draw.line(screen,BLACK,(200,0),(200,600),LINE_WIDTH)
  pygame.draw.line(screen,BLACK,(400,0),(400,600),LINE_WIDTH)
```

```
def mark_square(rows,cols,player):
  board[rows][cols] = player

def available_square(rows,cols):
  return board[rows][cols] == 0

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

def check_win(player):
  #vertical-win-check
  for col in range(BOARD_COLUMNS):
    if board[0][col]==player and board[1][col]==player and board[2][col]==player:
      draw_vertical_winning_line(col,player)
      return True
  #horizontal-win-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
  #ascending-diagonal-win-check
  if board[2][0]==player and board[1][1]==player and board[0][2]==player:
    draw_ascending_diagonal_winning_line(player)
    return True
  #descending-diagonal-win-check
  if board[0][0]==player and board[1][1]==player and board[2][2]==player:
    draw_descending_diagonal_winning_line(player)
    return True
  return False

def draw_vertical_winning_line(col,player):
  pygame.draw.line(screen,RED,(col*200+100,0),(col*200+100,600),20)

def draw_horizontal_winning_line(row,player):
  pygame.draw.line(screen,RED,(0,row*200+100),(600,row*200+100),20)
```

```
def draw_ascending_diagonal_winning_line(player):
  pygame.draw.line(screen,RED,(15,boardHeight-15),(WIDTH-15,15),20)

def draw_descending_diagonal_winning_line(player):
  pygame.draw.line(screen,RED,(15,15),(WIDTH-15,boardHeight-15),20)

def restart():
  screen.fill(BG_COLOR)
  draw_lines()
  game_over = False
  player = 1
  for row in range(BOARD_ROWS):
    for col in range(BOARD_COLUMNS):
      board[row][col]=0

def draw_figures():
  for row in range(BOARD_ROWS):
    for col in range(BOARD_COLUMNS):
      if board[row][col]==1:
        pygame.draw.circle(screen,BLACK, (int ((col*200)+100),int((row*200)+100)),
        CIRCLE_RADIUS,CIRCLE_WIDTH)
      elif board[row][col]==2:
        pygame.draw.line(screen,WHITE,(col*200+SPACE,(row*200+200)-SPACE),
        ((col*200+200)-SPACE,(row*200)+SPACE),15)
        pygame.draw.line(screen,WHITE,(col*200+SPACE,(row*200)+SPACE),
        ((col*200+200)-SPACE,(row*200+200)-SPACE),15)

draw_lines()

game_over = False
player = 1

#reset button and won display text.
resetButton = button(100,610,'Reset')
resetButton.draw_button()

text1 = font.render('O WON', True, white,BLACK)
text2 = font.render('X WON', True, white,BLACK)
textRect1 = text1.get_rect()
textRect1.center = (400,650)
textRect2 = text2.get_rect()
```

```
textRect2.center = (400,650)

#main loop
while True:
   for event in pygame.event.get():
     if event.type == pygame.QUIT:
       sys.exit()
     if event.type == pygame.MOUSEBUTTONDOWN and not game_over:
       mouseX  = event.pos[0] #x
       mouseY = event.pos[1] #y
       clicked_row = int(mouseY//200)
       clicked_col = int(mouseX//200)
       if(clicked_row<=2):
        if available_square(clicked_row,clicked_col):
          if player==1:
            mark_square(clicked_row,clicked_col,1)
            if check_win(player):
              game_over = True
              screen.blit(text1, textRect1)
            player = 2
          elif player == 2:
            mark_square(clicked_row,clicked_col,2)
            if check_win(player):
              game_over = True
              screen.blit(text2, textRect2)
            player = 1
          draw_figures()
       else:
         restart()
         player = 1
         game_over = False
         resetButton.draw_button()
     if event.type == pygame.MOUSEBUTTONDOWN:
       mouseY = event.pos[1]
       if(mouseY>600):
         restart()
         player=1
         game_over = False
         resetButton.draw_button()
   pygame.display.update()
```

SAMPLE INPUT-OUTPUT



TEST CASES

Test Case No:	Test Case Description	Test Data	Expected Result	Actual Result	Pass /Fail
1	Testing when Player 1 Wins	When a a row full is filled with X	X WON	X WON	Pass
2	Testing when Player 2 Wins	When a a row full is filled with O	O WON	O WON	Pass

GITHUB LINK

Click Here for the Code

PRINCIPAL COMPONENT ANALYSIS ON MATRIX

AIM

Implement Principle Component Analysis(PCA) of a matrix.

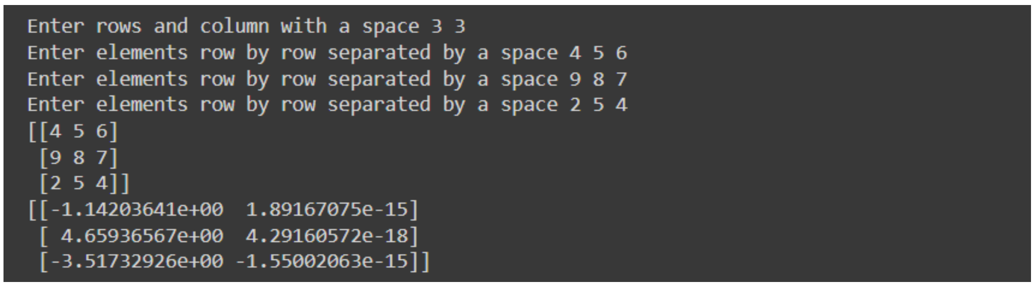
THEORY

Numpy , Linear Algebra

PROGRAM

```
from numpy import array
from numpy import mean
from numpy import cov
from numpy.linalg import eig

def computePCA():
  # taking number of rows and column
  n, m = map(int, input("Enter rows and column with a space ").split())
  #defining the matrix and taking input.
  A = array([input("Enter elements row by row separated by a space ").strip().split()
  for _ in range(n)], int)
  print(A)
  # calculate the mean of each column
  M = mean(A, axis=0)
  # center columns by subtracting column means
  C = A - M
  # calculate covariance matrix of centered matrix
  V = cov(C.T)
  # eigendecomposition of covariance matrix
  values, vectors = eig(V)
  projection_matrix = (vectors.T[:][:2]).T
  # project data
  P = projection_matrix.T.dot(C.T)
  print(P.T)

computePCA()
```

SAMPLE INPUT-OUTPUT



```
Enter rows and column with a space 3 3
Enter elements row by row separated by a space 4 5 6
Enter elements row by row separated by a space 9 8 7
Enter elements row by row separated by a space 2 5 4
[[4 5 6]
 [9 8 7]
 [2 5 4]]
[[-1.14203641e+00  1.89167075e-15]
 [ 4.65936567e+00  4.29160572e-18]
 [-3.51732926e+00 -1.55002063e-15]]
```

TEST CASES

Test Case No:	Test Case Description	Test Data	Expected Result	Actual Result	Pass /Fail
1	Giving random values as matrix	[4,5,6; 9,8,7; 2,5,4]	Dimensionality reduced matrix	Dimensionality reduced matrix	Pass

GITHUB LINK

Click Here for the Code

DATA VISUALISASTION

AIM

Create an account in Kaggle.com Download iris dataset from the link https://www.kaggle.com/datasets/saurabh00007/iriscsv 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 functon from sklearn) • Show the distribution of each attribute as histogram. Note: for visualization, you can either use matplotlib or seaborn.

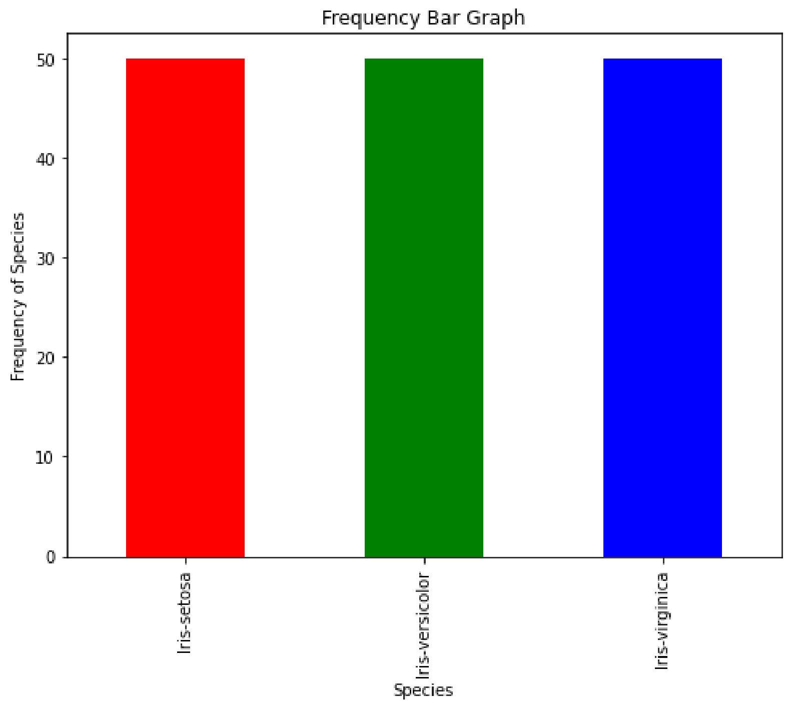
THEORY

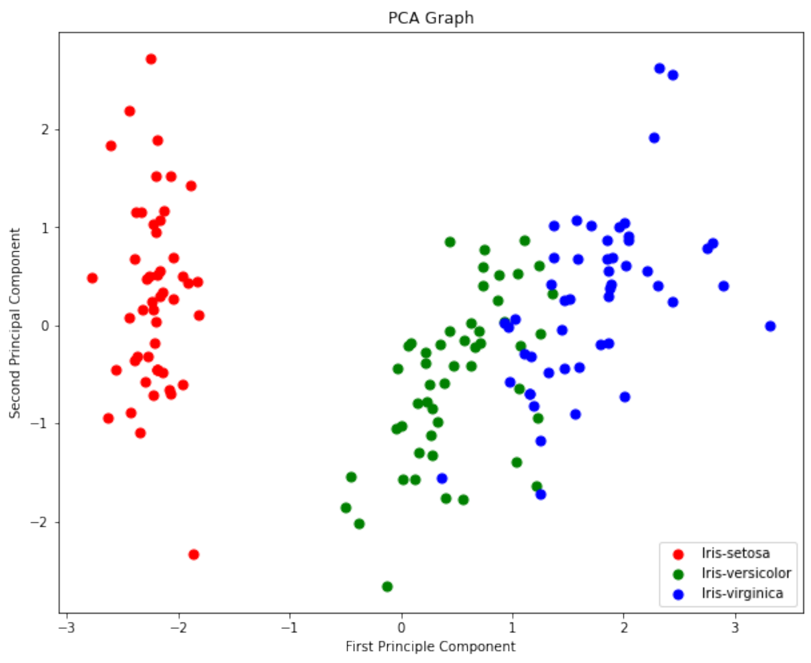
Visualization, Data processing, Libraries : pandas, matplotlib, seaborn, histogram

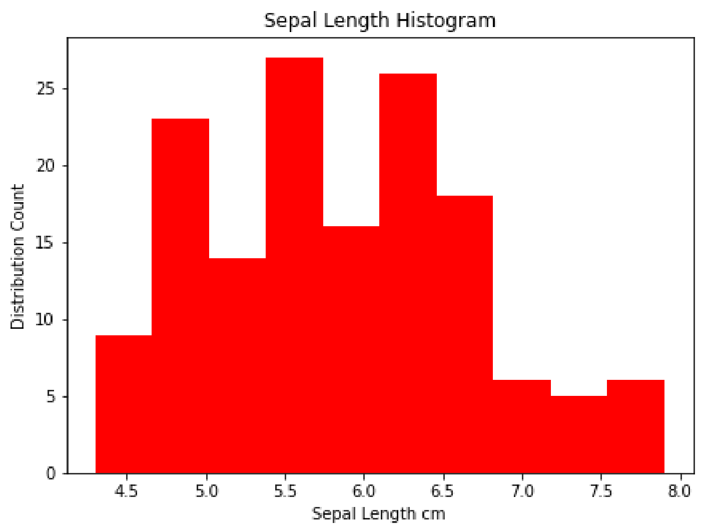
PROGRAM

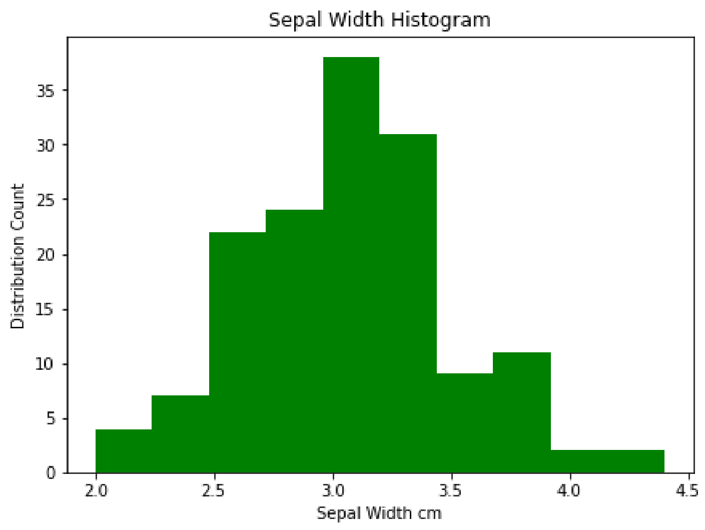
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
%matplotlib inline

#reading the csv file into df.
df = pd.read_csv ('Iris.csv')

def frequencyBar():
  #ploting the bar chart according to the frequency of species.
  df.Species.value_counts().plot(figsize=(8,6),kind='bar',color=['r','g','b'],
  xlabel='Species',ylabel='Frequency of Species')
  plt.title("Frequency Bar Graph")
  plt.show()

def pcaAppliedGraph():
  print("\nPCA Graph")
  #plotting the principal analysis graph for two components
  X = df.iloc[:, 1:5].values
  X_std = StandardScaler().fit_transform(X)
  pca = PCA(n_components=2)
  principalComponents = pca.fit_transform(X_std)
  principalDf = pd.DataFrame(data = principalComponents
              , columns = ['principal component 1', 'principal component 2'])
  finalDf = pd.concat([principalDf, df['Species']], axis = 1)
  fig = plt.figure(figsize = (10,8))
  ax = fig.add_subplot(1,1,1)
  ax.set_xlabel('First Principle Component')
  ax.set_ylabel('Second Principal Component')
  ax.set_title('PCA Graph')
  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)
  plt.show()

def distributionHistogramSepal():
  print("\nDistribution Histogram\n")
  #histogram for sepal length.
  plt.figure(figsize = (7, 5))
  x = df.SepalLengthCm
  plt.hist(x, color = "r")
  plt.title("Sepal Length Histogram")
  plt.xlabel("Sepal Length cm")
  plt.ylabel("Distribution Count")
  plt.show()

  print()
  #histogram for sepal width.
  plt.figure(figsize = (7, 5))
  x = df.SepalWidthCm
  plt.hist(x, color = "g")
  plt.title("Sepal Width Histogram")
  plt.xlabel("Sepal Width cm")
  plt.ylabel("Distribution Count")
  plt.show()

def distributionHistogramPetal():
  print()
  #histogram for petal length.
  plt.figure(figsize = (7, 5))
  x = df.PetalLengthCm
  plt.hist(x, color = "b")
  plt.title("Petal Length Histogram")
  plt.xlabel("Petal Length cm")
  plt.ylabel("Distribution Count")
  plt.show()

  print()
  #histogram for petal width.
  plt.figure(figsize = (7, 5))
  x = df.PetalWidthCm
  plt.hist(x, color = "orange")
  plt.title("Petal Width Histogram")
  plt.xlabel("Petal Width cm")
  plt.ylabel("Distribution Count")
  plt.show()

#invoking the function to display graph.
frequencyBar()
pcaAppliedGraph()
distributionHistogramSepal()
distributionHistogramPetal()
```

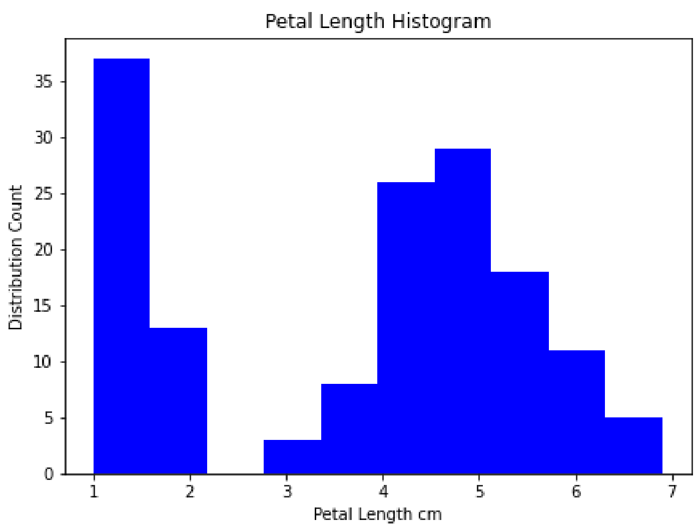
SAMPLE INPUT-OUTPUT

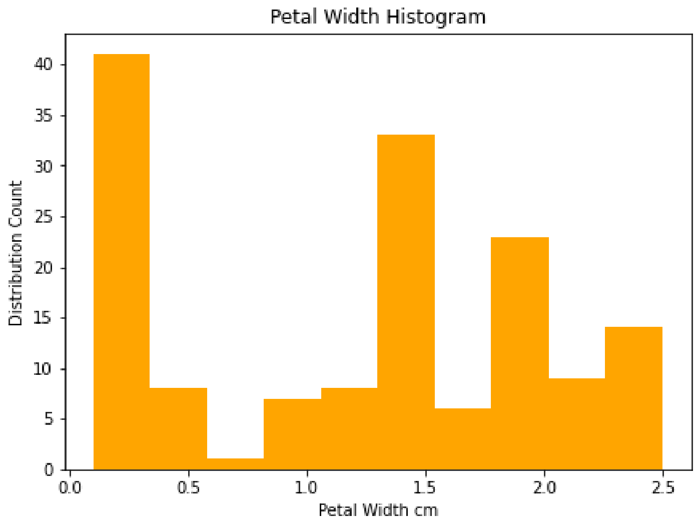



PCA Graph
Second Principal Component
First Principle Component
Iris-setosa
Iris-versicolor
Iris-virginica


Sepal Length Histogram
Distribution Count
Sepal Length cm


Sepal Width Histogram
Distribution Count
Sepal Width cm





TEST CASES

Test Case No:	Test Case Description	Test Data	Expected Result	Actual Result	Pass /Fail
1	Loading CSV file and generating bar graph	Dataframe with 150 rows and 3 different species	Bar Graph for 3 species with same height	Bar Graph for 3 species with same height	Pass

GITHUB LINK

Click Here for the Code

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 , sort 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.

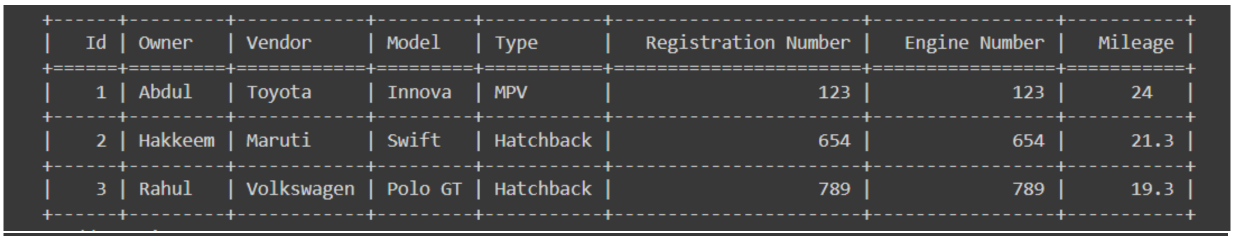
THEORY

OOPs, Pickle, PDF report generation, Lambda functions for sorting

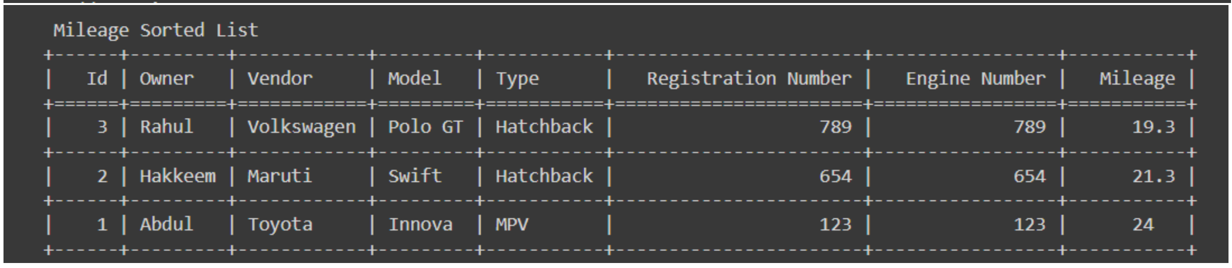
PROGRAM

```
import pickle,tabulate
class vehicleAttributes:
  #list of keys of the attributes of the vehicle.
  _keyList = ["id","ownerName","vendor","model","type","registrationNumber",
  "engineNumber","mileage"]
  #creating a dictionary with the keys and values as None.
  _dataBase = dict.fromkeys(_keyList, None)

class Vehicles(vehicleAttributes):
  #list which will save the dictionary of every vehicles
  __listOfVehicles = list()
  __id = 0
  def addEntries(self):
    option = 1
    while option==1:
      self.__id = self.__id + 1
      entryList = list()
      entryList.append(self.__id)
      entryList.append(input("Owner Name : "))
      entryList.append(input("Vendor Name : "))
      entryList.append(input("Model Name : "))
      entryList.append(input("Type : "))
      entryList.append(int(input("Registration Number : ")))
      entryList.append(int(input("Engine Number : ")))
      entryList.append(float(input("Mileage : ")))
      for i,key in zip(entryList,self._dataBase):
        self._dataBase[key] = i
      self.__listOfVehicles.append(self._dataBase.copy())
      option = int(input("Do you want to add more entries\n1.Add\n2.Exit\n"))

  def deleteEntries(self):
    found = False
    searchKey = int(input("Enter your ID : "))
    for i in range(len(self.__listOfVehicles)):
      if self.__listOfVehicles[i]['id']==searchKey:
        found = True
        del self.__listOfVehicles[i]
        break
    if(not found):
      print("Invalid Id")

  def modifyEntries(self):
    found = False
    searchKey = int(input("Enter your ID : "))
    for i in self.__listOfVehicles:
      if i['id']==searchKey:
        found = True
        print("Choose the attribute you want to change")
        print("1.Owner Name\n2.Vendor Name\n3.Model Name")
        print("4.Type\n5.Registration Number\n6.Engine Number")
        print("7.Mileage")
        option = int(input())
        if option==1:
          i['ownerName'] = input("Owner Name : ")
        elif option==2:
          i['vendor'] = input("Vendor Name : ")
        elif option==3:
          i['model'] = input("Model Name : ")
        elif option==4:
          i['type'] = input("Type : ")
        elif option==5:
          i['registrationNumber'] = int(input("Registration Number : "))
        elif option==6:
          i['engineNumber'] = int(input("Engine Number : "))
        elif option==7:
          i['mileage'] = float(input("Mileage : "))
    if(not found):
      print("Invalid Key")

  def display(self,*args):
    header = ['Id','Owner','Vendor','Model','Type','Registration Number',
    'Engine Number','Mileage']
    if(len(args)==0):
      rows =  [x.values() for x in self.__listOfVehicles]
      print(tabulate.tabulate(rows, header,tablefmt='grid'))
    elif (len(args)==2):
      print("\n",args[0])
      rows = [x.values() for x in args[1]]
      print(tabulate.tabulate(rows, header,tablefmt='grid'))

  def sortMileage(self):
    sortedList = sorted(self.__listOfVehicles,key = lambda i:i['mileage'])
    self.display("Mileage Sorted List",sortedList)

  def createFile(self):
    pickle.dump(self.__listOfVehicles,open("vehicleDetails.pkl","wb"))

  def filterAttributes(self):
    print("Choose the attribute which you want to filter\n1.Owner Name")
    print("2.Vendor\n3.Model Name\n4.Type\n5.Mileage")
    option = int(input("Option : "))
    filteredList = list()
    if(option==1):
      filterKey = (input("Enter the name you want to filter"))
      filteredList = [i for i in self.__listOfVehicles if i['ownerName']== filterKey]
      self.display("Filtered List",filteredList)
    elif (option==2):
      filterKey = (input("Enter the name you want to filter"))
      filteredList = [i for i in self.__listOfVehicles if i['vendor']== filterKey]
      self.display("Filtered List",filteredList)
    elif (option==3):
      filterKey = (input("Enter the name you want to filter"))
      filteredList = [i for i in self.__listOfVehicles if i['model']== filterKey]
      self.display("Filtered List",filteredList)
    elif (option==4):
      filterKey = (input("Enter the name you want to filter"))
      filteredList = [i for i in self.__listOfVehicles if i['type']== filterKey]
      self.display("Filtered List",filteredList)
    elif(option==5):
      filterKey = float(input("Enter the name you want to filter"))
      filteredList = [i for i in self.__listOfVehicles if i['mileage']== filterKey]
      self.display("Filtered List",filteredList)
  def loadFile(self,filePath):
    self.__listOfVehicles = pickle.load(open(filePath,"rb"))
    idList = [self.__listOfVehicles[i]['id'] for i in
    range(len(self.__listOfVehicles))]
    self.__id = max(idList)

def main():
  vehicleObject = Vehicles()
  if(int(input("1.Add Entries\n2.Load Pickle\n"))==1):
    vehicleObject.addEntries()
  else:
    filePath = input("Enter the file name : ")
    vehicleObject.loadFile(filePath)
  vehicleObject.display()
  mainLoopOption=1
  while mainLoopOption==1:
    print("1.Add Entries\n2.Modify Attributes\n3.Delete Attributes
    \n4.Display Entries")
    print("5.Sort According to Mileage\n6.Filter Attributes\n7.Create Pickle File
    \n8.Exit")
    choice = int(input())
    if choice==1:
      vehicleObject.addEntries()
    elif choice==2:
      vehicleObject.modifyEntries()
    elif choice==3:
      vehicleObject.deleteEntries()
    elif choice==4:
      vehicleObject.display()
    elif choice==5:
      vehicleObject.sortMileage()
    elif choice==6:
      vehicleObject.filterAttributes()
    elif choice==7:
      vehicleObject.createFile()
    elif choice==8:
      break
  mainLoopOption = int(input("\n1.Continue\n2.Exit"))

if __name__=="__main__":
  main()
```

SAMPLE INPUT-OUTPUT



```
+------+---------+------------+---------+-----------+----------------------+-----------------+-----------+
|  Id  | Owner   | Vendor     | Model   | Type      |  Registration Number |   Engine Number |  Mileage  |
+======+=========+============+=========+===========+======================+=================+===========+
|    1 | Abdul   | Toyota     | Innova  | MPV       |                  123 |             123 |    24     |
+------+---------+------------+---------+-----------+----------------------+-----------------+-----------+
|    2 | Hakkeem | Maruti     | Swift   | Hatchback |                  654 |             654 |    21.3   |
+------+---------+------------+---------+-----------+----------------------+-----------------+-----------+
|    3 | Rahul   | Volkswagen | Polo GT | Hatchback |                  789 |             789 |    19.3   |
+------+---------+------------+---------+-----------+----------------------+-----------------+-----------+
```



```
 Mileage Sorted List
+------+---------+------------+---------+-----------+----------------------+-----------------+-----------+
|  Id  | Owner   | Vendor     | Model   | Type      |  Registration Number |   Engine Number |  Mileage  |
+======+=========+============+=========+===========+======================+=================+===========+
|    3 | Rahul   | Volkswagen | Polo GT | Hatchback |                  789 |             789 |    19.3   |
+------+---------+------------+---------+-----------+----------------------+-----------------+-----------+
|    2 | Hakkeem | Maruti     | Swift   | Hatchback |                  654 |             654 |    21.3   |
+------+---------+------------+---------+-----------+----------------------+-----------------+-----------+
|    1 | Abdul   | Toyota     | Innova  | MPV       |                  123 |             123 |    24     |
+------+---------+------------+---------+-----------+----------------------+-----------------+-----------+
```

TEST CASES

Test Case No:	Test Case Description	Test Data	Expected Result	Actual Result	Pass /Fail
1	Loading a pickle file with details of Vehicles	3 Vehicle details with unordered mileage	Collection sorted according to mileage	Collection sorted according to mileage	Pass

GITHUB LINK

Click Here for the Code

TKINTER UI APPLICATION

AIM

Convert the task in Question 4 as a UI based application using Tkinter or PyQT

THEORY

GUI using tkinter or PyQT

PROGRAM

```
from tkinter import *
from tkinter.filedialog import askopenfilename,asksaveasfile
from tkinter.messagebox import showinfo
from tkinter.ttk import Style, Treeview
import pickle

global listOfVehicles
global sortedList
listOfVehicles = list()
vehicle_attributes = ["ownerName","vendor","model","type","registrationNumber"
,"engineNumber","mileage"]
vehicleDetails = dict.fromkeys(vehicle_attributes, None)

def addList():
    global listOfVehicles
    treeList.insert(parent='', index='end', text="", values=(owner.get(),
    vendor.get(),model.get(),typeClass.get(),regNumber.get(),engNumber.get(),
    mileage.get()))
    vehicleDetails['ownerName'] = owner.get()
    vehicleDetails['vendor'] = vendor.get()
    vehicleDetails['model'] = model.get()
    vehicleDetails['type'] = typeClass.get()
    vehicleDetails['registrationNumber'] = int(regNumber.get())
    vehicleDetails['engineNumber'] = int(engNumber.get())
    vehicleDetails['mileage'] = float(mileage.get())
    listOfVehicles.append(vehicleDetails.copy())

def filterList():
    global listOfVehicles
    if(owner.get()!="" or vendor.get()!="" or model.get()!="" or
    typeClass.get()!="" or mileage.get()!=""):
        for item in treeList.get_children():
            treeList.delete(item)
    else:
        showinfo(title="Error",message="Give a Filter Key")
    if(owner.get()!=""):
        filterKey = owner.get()
        for i in listOfVehicles:
            if i['ownerName']==filterKey:
                treeList.insert(parent='', index='end', text="", values=
                (i['ownerName'],i['vendor'],i['model'],i['type'],
                i['registrationNumber'],i['engineNumber'],i['mileage']))
    elif (vendor.get()!=""):
        filterKey = vendor.get()
        for i in listOfVehicles:
            if i['vendor']==filterKey:
                treeList.insert(parent='', index='end', text="", values=
                (i['ownerName'],i['vendor'],i['model'],i['type'],
                i['registrationNumber'],i['engineNumber'],i['mileage']))
    elif (model.get()!=""):
        filterKey = model.get()
        for i in listOfVehicles:
            if i['model']==filterKey:
                treeList.insert(parent='', index='end', text="", values=
                (i['ownerName'],i['vendor'],i['model'],i['type'],
                i['registrationNumber'],i['engineNumber'],i['mileage']))
    elif (typeClass.get()!=""):
        filterKey = typeClass.get()
        for i in listOfVehicles:
            if i['type']==filterKey:
                treeList.insert(parent='', index='end', text="", values=
                (i['ownerName'],i['vendor'],i['model'],i['type'],
                i['registrationNumber'],i['engineNumber'],i['mileage']))
    elif (mileage.get()!=""):
        filterKey = float(mileage.get())
        for i in listOfVehicles:
            if i['mileage']==filterKey:
                treeList.insert(parent='', index='end', text="", values=
                (i['ownerName'],i['vendor'],i['model'],i['type'],
                i['registrationNumber'],i['engineNumber'],i['mileage']))
def delete():
   # Get selected item to Delete
```

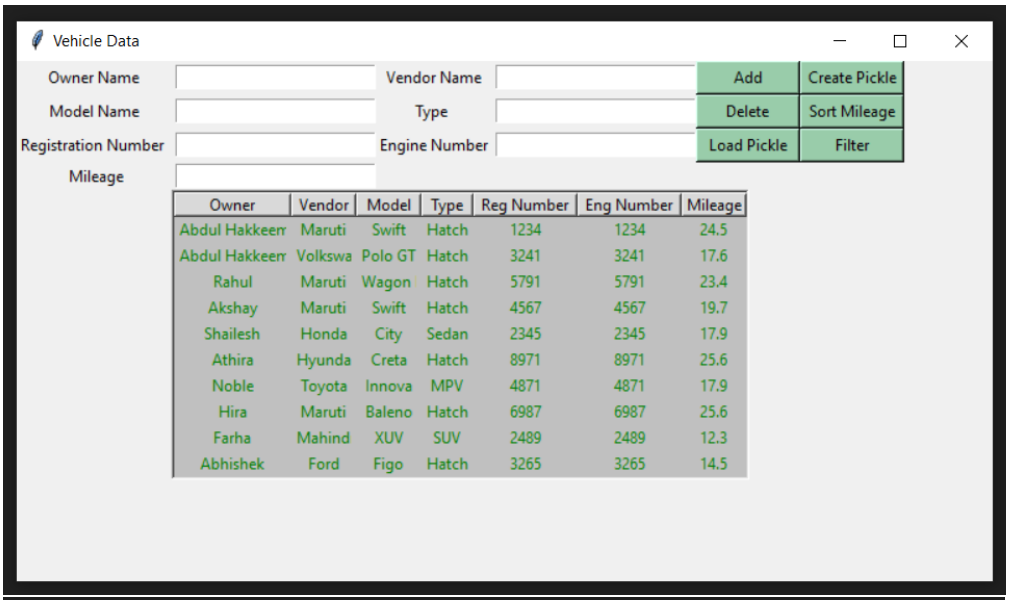
```
    selection=treeList.selection()[0]
    treeList.delete(selection)

def loadFile():
    #Clear the treeview list items
    for item in treeList.get_children():
        treeList.delete(item)
    filetypes = (
        ('Picle files', '*.pkl'),
        ('All files', '*.*')
    )
    global listOfVehicles
    filename = askopenfilename(title="Open Pickle",initialdir='/',filetypes=filetypes)
    listOfVehicles = pickle.load(open(filename,"rb"))
    showinfo(title="Selected File",message=filename)
    for i in listOfVehicles:
        treeList.insert(parent='', index='end', text="", values=
        (i['ownerName'],i['vendor'],i['model'],i['type'],
        i['registrationNumber'],i['engineNumber'],i['mileage']))



def sortMileage():
    #Clear the treeview list items
    for item in treeList.get_children():
        treeList.delete(item)
    global listOfVehicles
    global sortedList
    sortedList = sorted(listOfVehicles,key= lambda i:i['mileage'])
    for i in sortedList:
        treeList.insert(parent='', index='end', text="", values=
        (i['ownerName'],i['vendor'],i['model'],i['type'],
        i['registrationNumber'],i['engineNumber'],i['mileage']))
    showinfo(title="Sorted",message="Sorted Successfully")

def createPickle():
    fileextensions = [('Pickle File', '*.pkl'),('All Files', '*.*')]
    file = asksaveasfile(filetypes = fileextensions, defaultextension = fileextensions)
    pickle.dump(listOfVehicles,open(file,"wb"))
    showinfo(title="Created File",message="Vehicle Pickle File is Created")

#window configuration.
```

```
window = Tk()
window.geometry("750x400")
window.title("Vehicle Data")

#variables to take input from screen.
owner = StringVar()
vendor = StringVar()
model = StringVar()
typeClass = StringVar()
regNumber = StringVar()
engNumber = StringVar()
mileage = StringVar()



#row-0
label1 = Label(window,text="Owner Name ")
label1.grid(row=0,column=0)

entry1 = Entry(window,width=25,textvariable=owner)
entry1.grid(row=0,column=1)

label2 = Label(window,text="Vendor Name ")
label2.grid(row=0,column=2)

entry2 = Entry(window,width=25,textvariable=vendor)
entry2.grid(row=0,column=3)

#row-1
label3 = Label(window,text="Model Name ")
label3.grid(row=1,column=0)

entry3 = Entry(window,width=25,textvariable=model)
entry3.grid(row=1,column=1)

label4 = Label(window,text="Type  ")
label4.grid(row=1,column=2)

entry4 = Entry(window,width=25,textvariable=typeClass)
entry4.grid(row=1,column=3)

#row-2
```

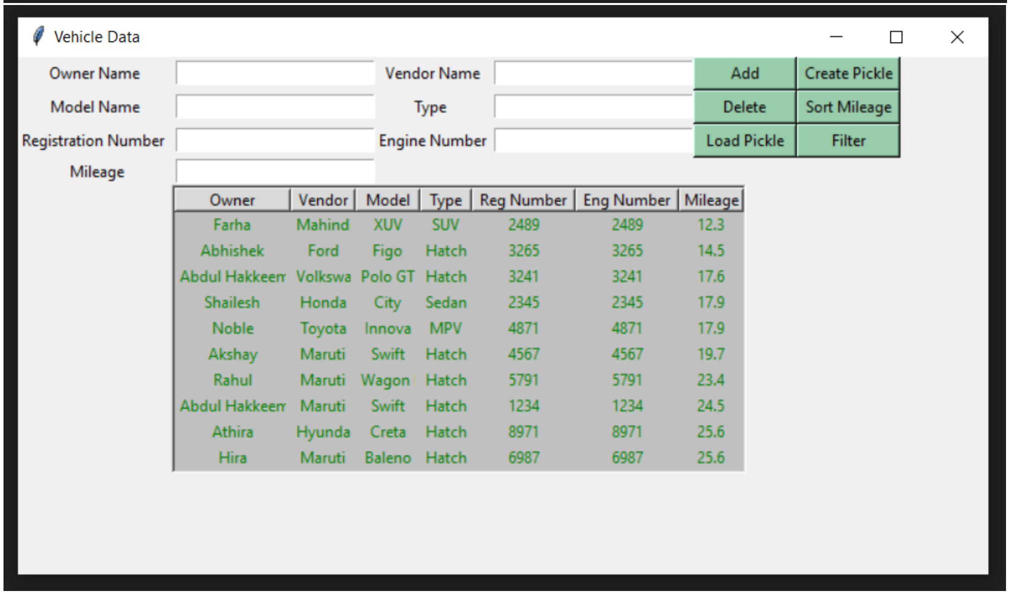
```
label5 = Label(window,text="Registration Number  ")
label5.grid(row=2,column=0)

entry5 = Entry(window,width=25,textvariable=regNumber)
entry5.grid(row=2,column=1)

label6 = Label(window,text="Engine Number ")
label6.grid(row=2,column=2)

entry6 = Entry(window,width=25,textvariable=engNumber)
entry6.grid(row=2,column=3)

#row - 3
label7 = Label(window,text="Mileage")
label7.grid(row=3,column=0)

entry7 = Entry(window,width=25,textvariable=mileage)
entry7.grid(row=3,column=1)

button1=Button(window,width=10,text="Load Pickle",bg='#99CCAA',command=loadFile)
button1.grid(row=2,column=4)

button8=Button(window,width=10,text="Filter",bg='#99CCAA',command=filterList)
button8.grid(row=2,column=5)

#second section
#row - 0
button2=Button(window,width=10,text="Add",bg='#99CCAA',command=addList)
button2.grid(row=0,column=4)

#row - 1
button4=Button(window,width=10,text="Delete",bg='#99CCAA',command=delete)
button4.grid(row=1,column=4)

button5=Button(window,width=10,text="Sort Mileage",bg='#99CCAA',command=sortMileage)
button5.grid(row=1,column=5)

#row - 2
# button6=Button(window,width=10,text="Filter",bg='#99CCAA')
# button6.grid(row=2,column=4)
```

```
button7=Button(window,width=10,text="Create Pickle",bg='#99CCAA',command=createPickle)
button7.grid(row=0,column=5)

style = Style()
style.theme_use("alt")
style.configure("Treeview",
    background="silver",
    foreground='green'
)
style.map('Treeview',background=[('selected','#FF1800')])
treeList = Treeview(columns=("Owner","Vendor","Model","Type","Reg Number",
"Eng Number","Mileage"),show='headings')

treeList.heading("Owner",text="Owner")
treeList.heading("Vendor",text="Vendor")
treeList.heading("Model",text="Model")
treeList.heading("Type",text="Type")
treeList.heading("Reg Number",text="Reg Number")
treeList.heading("Eng Number",text="Eng Number")
treeList.heading("Mileage",text="Mileage")

treeList['show']='headings'

treeList.column("Owner",width=90, anchor="center")
treeList.column("Vendor",width=50, anchor="center")
treeList.column("Model",width=50, anchor="center")
treeList.column("Type",width=40, anchor="center")
treeList.column("Reg Number",width=80, anchor="center")
treeList.column("Eng Number",width=80, anchor="center")
treeList.column("Mileage",width=50, anchor="center")

treeList.grid(row=4,column=0,columnspan=6)
window.mainloop()
```

SAMPLE INPUT-OUTPUT



Owner	Vendor	Model	Type	Reg Number	Eng Number	Mileage
Abdul Hakkeem	Maruti	Swift	Hatch	1234	1234	24.5
Abdul Hakkeem	Volkswa	Polo GT	Hatch	3241	3241	17.6
Rahul	Maruti	Wagon	Hatch	5791	5791	23.4
Akshay	Maruti	Swift	Hatch	4567	4567	19.7
Shailesh	Honda	City	Sedan	2345	2345	17.9
Athira	Hyunda	Creta	Hatch	8971	8971	25.6
Noble	Toyota	Innova	MPV	4871	4871	17.9
Hira	Maruti	Baleno	Hatch	6987	6987	25.6
Farha	Mahind	XUV	SUV	2489	2489	12.3
Abhishek	Ford	Figo	Hatch	3265	3265	14.5



Owner	Vendor	Model	Type	Reg Number	Eng Number	Mileage
Farha	Mahind	XUV	SUV	2489	2489	12.3
Abhishek	Ford	Figo	Hatch	3265	3265	14.5
Abdul Hakkeem	Volkswa	Polo GT	Hatch	3241	3241	17.6
Shailesh	Honda	City	Sedan	2345	2345	17.9
Noble	Toyota	Innova	MPV	4871	4871	17.9
Akshay	Maruti	Swift	Hatch	4567	4567	19.7
Rahul	Maruti	Wagon	Hatch	5791	5791	23.4
Abdul Hakkeem	Maruti	Swift	Hatch	1234	1234	24.5
Athira	Hyunda	Creta	Hatch	8971	8971	25.6
Hira	Maruti	Baleno	Hatch	6987	6987	25.6

TEST CASES

Test Case No:	Test Case Description	Test Data	Expected Result	Actual Result	Pass /Fail
1	Load a Pickle file and sort the collection according to mileage	Details of 10 Vehicles	Sorted list of vehicles according to mileage	Sorted list of vehicles according to mileage	Pass

GITHUB LINK

Click Here for the Code