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

NAZAL NIHAD TT
(80522015)

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

MAY 2023

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 **NAZAL NIHAD TT(80522015)**, 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. Madhu S. Nair
Assistant Professor	Professor and Head
Department of Computer Science	Department of Computer Science
CUSAT	CUSAT

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.

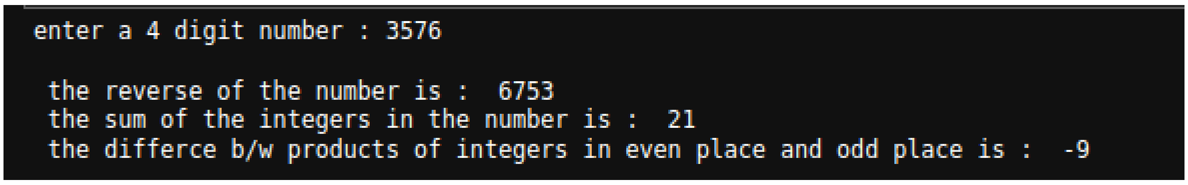
THEORY

- 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

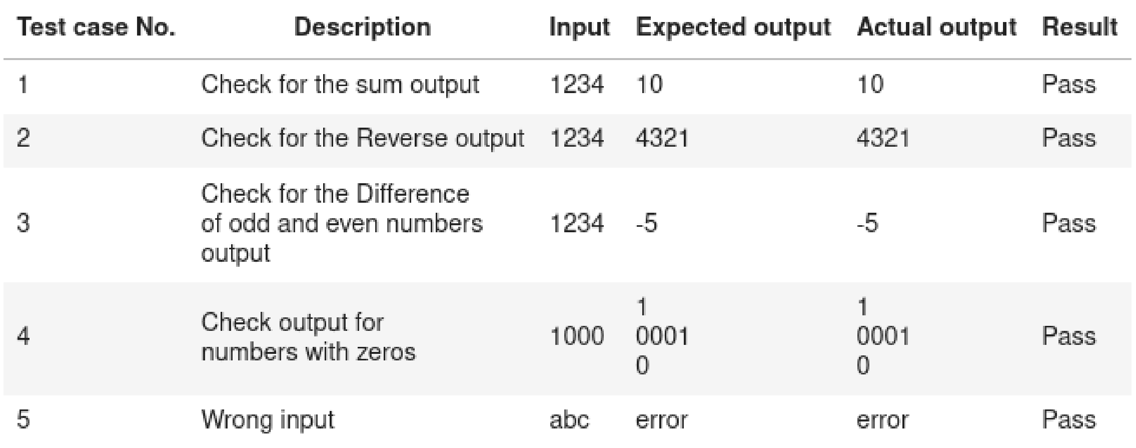
```
num = int(input("enter a 4 digit number : "))
length = len(str(num))

if(length == 4):
  i , reverse , sum = 1,0,0
  odd_place , even_place = 1,1

  while(i<=length):
    #finding the last digit
    a = num%10

    if i%2==0:
      #checking even place , multiplying and storing
      even_place = even_place*a
    else:
      #checking odd place , multiplying and storing
      odd_place = odd_place*a

    #reverse the integer
    reverse = reverse*10 + a
    #find sum of integer
    sum += a
    #change original digit
    num=num//10
    i+=1

  #difference b/w product of numbers at odd and even place
  difference =  even_place-odd_place
  print("the reverse of the number is : ",reverse )
  print(" the sum of the integers in the number is : ",sum)
  print( " the differce b/w products of integers in even place and odd place is
  : ",difference)
else:
  print("Please enter a 4 digit number!!")
```

SAMPLE INPUT-OUTPUT



```
enter a 4 digit number : 3576

 the reverse of the number is :  6753
 the sum of the integers in the number is :  21
 the differce b/w products of integers in even place and odd place is :  -9
```

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 :
$//github.com/nazalnihad/Python_Lab_Cycles/blob/main/Lab_cycle_1/Lab_cycle_qn1/qn1.py$

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

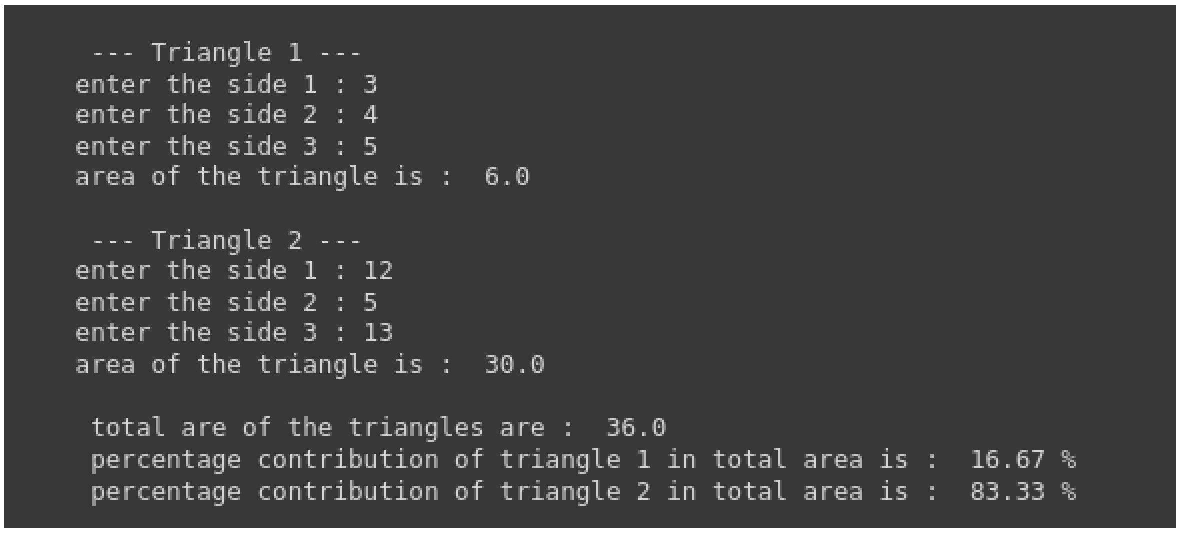
THEORY

- 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

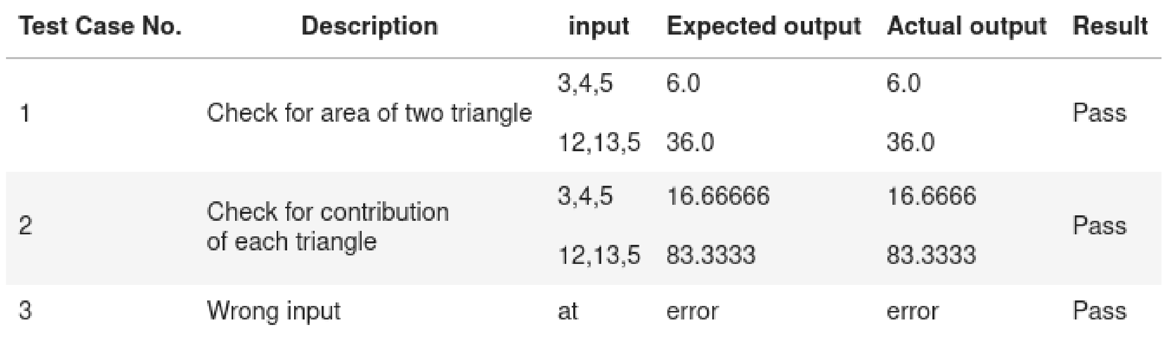
```
import math

#function to get input of 3 sides of a triangle
def sides():
  s1 = int(input("enter the side 1 : "))
  s2 = int(input("enter the side 2 : "))
  s3 = int(input("enter the side 3 : "))
  return s1,s2,s3

#function to find are from the given sides
def area(s1,s2,s3):
  area = 0
  #semi perimeter
  s = (s1+s2+s3)/2
  area = math.sqrt(s*(s-s1)*(s-s2)*(s-s3))
  print("area of the triangle is : ",round(area,2)) #area rounded off to 2 digits
  return area

#function to find contribution of each triangle in total area
def percentage(area1,area2):
  total = area1+area2
  print("\n total are of the triangles are : ",round(total,2))
  #contribution of triangle 1 and round off to 2 digits
  triangle1 = round(area1*100/total,2)
  #contribution of triangle 2 and round off to 2 digits
  triangle2 = round(area2*100/total,2)

  print(" percentage contribution of triangle 1 in total area is : ",triangle1,"%")
  print(" percentage contribution of triangle 2 in total area is : ",triangle2,"%")

print("\n --- Triangle 1 ---")
#call function to get sides
a,b,c = sides()
#call function to find area
area1 = area(a,b,c)

print("\n --- Triangle 2 ---")
a1,b2,c3 = sides()
area2 = area(a1,b2,c3)

#call function to find percentage of each
percentage(area1,area2)
```

SAMPLE INPUT-OUTPUT



```
 --- Triangle 1 ---
enter the side 1 : 3
enter the side 2 : 4
enter the side 3 : 5
area of the triangle is :  6.0

 --- Triangle 2 ---
enter the side 1 : 12
enter the side 2 : 5
enter the side 3 : 13
area of the triangle is :  30.0

 total are of the triangles are :  36.0
 percentage contribution of triangle 1 in total area is :  16.67 %
 percentage contribution of triangle 2 in total area is :  83.33 %
```

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	16.6666	Pass
		12,13,5	83.3333	83.3333	
3	Wrong input	at	error	error	Pass

RESULT

Program executed Successfully and the output is obtained.

GIT LINK

$https : //github.com/nazalnihad/Python_{L}ab_{C}ycles/blob/main/Lab_{c}ycle_{1}/Lab_{c}ycle_{q}n2/qn2.py$

EMPLOYEE PAYSLIP 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

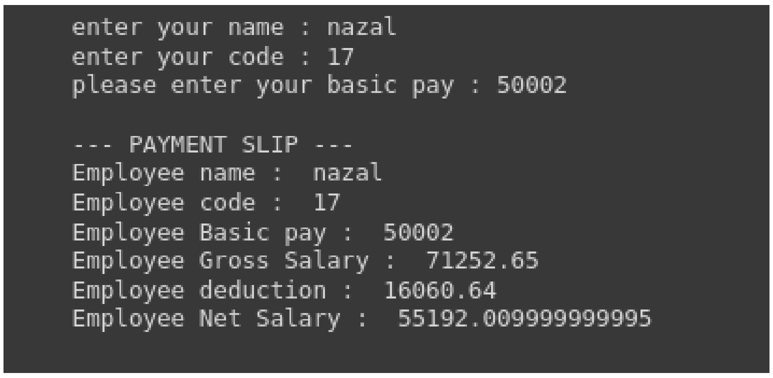
- Conditional Branching - A programming instruction that directs the computer to another part of the program based on the results of a comparison

PROGRAM

```
#function to find gross salary
def gross_salary(bp,da,hra,ma):
  gross_salary = bp+da+hra+ma
  return gross_salary

#function to find deduction
def deduction(pt,pf,it):
  deduction = pt+pf+it
  return deduction

#function to find net salary
def net_salary(gs,d):
  net_salary = (gs - d)
  return net_salary

#function to print payment slip
```

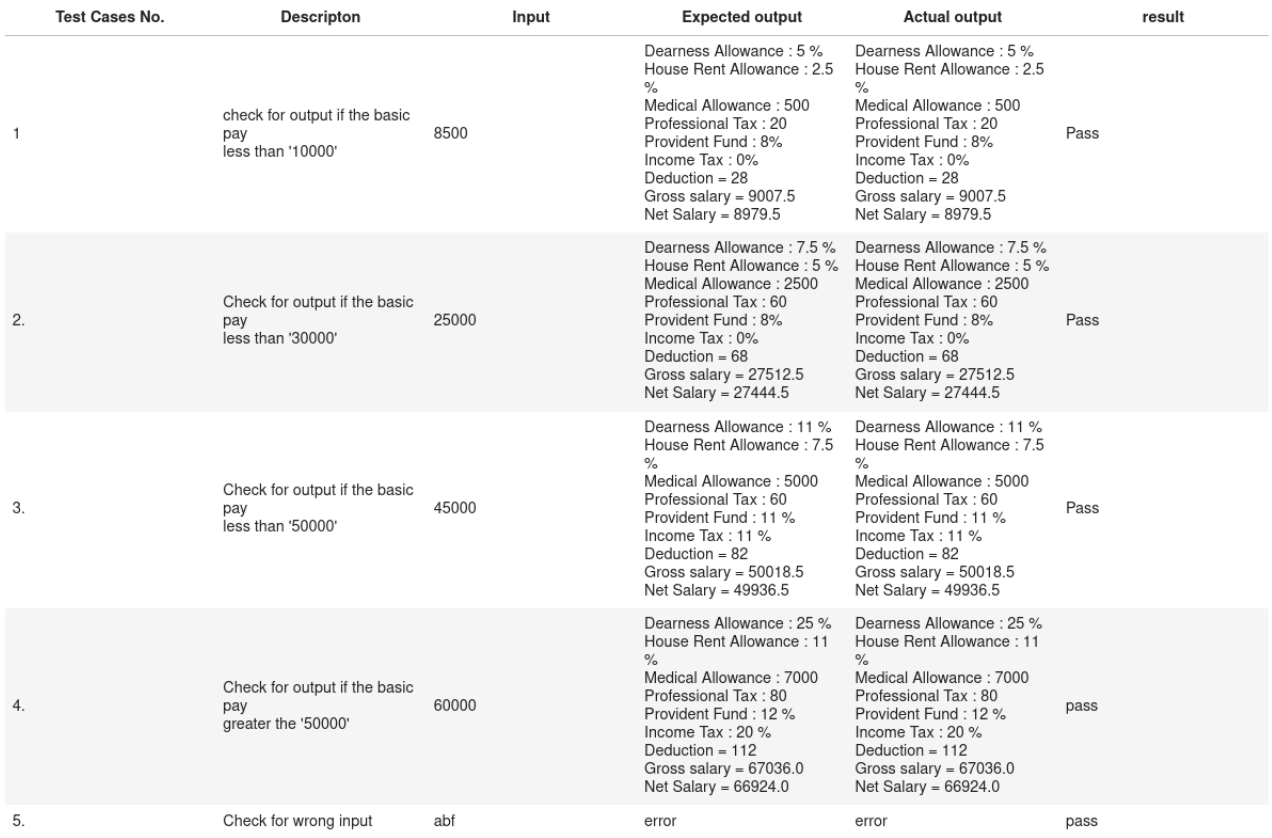
```
def payment_slip(name,code,basic_pay):
  if basic_pay<10000:
    da = (5*basic_pay)/100
    hra = (2.5*basic_pay)/100
    ma = 500
    pt = 20
    pf = (8*basic_pay)/100
    it = 0
    Gross_Salary = gross_salary(basic_pay,da,hra,ma)
    Deduction = deduction(pt,pf,it)
    Net_Salary = net_salary(Gross_Salary,Deduction)

  elif basic_pay<30000:
    da = (7.5*basic_pay)/100
    hra = (5*basic_pay)/100
    ma = 2500
    pt = 60
    pf = (8*basic_pay)/100
    it = 0
    Gross_Salary = gross_salary(basic_pay,da,hra,ma)
    Deduction = deduction(pt,pf,it)
    Net_Salary = net_salary(Gross_Salary,Deduction)

  elif basic_pay<50000:
    da = (11*basic_pay)/100
    hra = (7.5*basic_pay)/100
    ma = 5000
    pt = 60
    pf = (11*basic_pay)/100
    it = (11*basic_pay)/100
    Gross_Salary = gross_salary(basic_pay,da,hra,ma)
    Deduction = deduction(pt,pf,it)
    Net_Salary = net_salary(Gross_Salary,Deduction)

  else:
    da = (25*basic_pay)/100
    hra = (7.5*basic_pay)/100
    ma = 5000
    pt = 60
    pf = (12*basic_pay)/100
    it = (20*basic_pay)/100
    Gross_Salary = gross_salary(basic_pay,da,hra,ma)
    Deduction = deduction(pt,pf,it)
    Net_Salary = net_salary(Gross_Salary,Deduction)

  print("\n--- PAYMENT SLIP ---")
  print("Employee name : ",name)
  print("Employee code : ",code)
  print("Employee Basic pay : ",basic_pay)
  print("Employee Gross Salary : ",Gross_Salary)
  print("Employee deduction : ",Deduction)
  print("Employee Net Salary : ",Net_Salary)


name = str(input("enter your name : "))
code = int(input("enter your code : "))
basic_pay = int(input("please enter your basic pay : "))

payment_slip(name,code,basic_pay)
```

SAMPLE INPUT-OUTPUT



```
enter your name : nazal
enter your code : 17
please enter your basic pay : 50002

--- PAYMENT SLIP ---
Employee name :  nazal
Employee code :  17
Employee Basic pay :  50002
Employee Gross Salary :  71252.65
Employee deduction :  16060.64
Employee Net Salary :  55192.009999999995
```

TEST CASES



Test Cases No.	Descripton	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

$https$:
$//github.com/nazalnihad/Python_Lab_Cycles/blob/main/Lab_cycle_1/Lab_cycle_qn3/qn3.py$

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

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

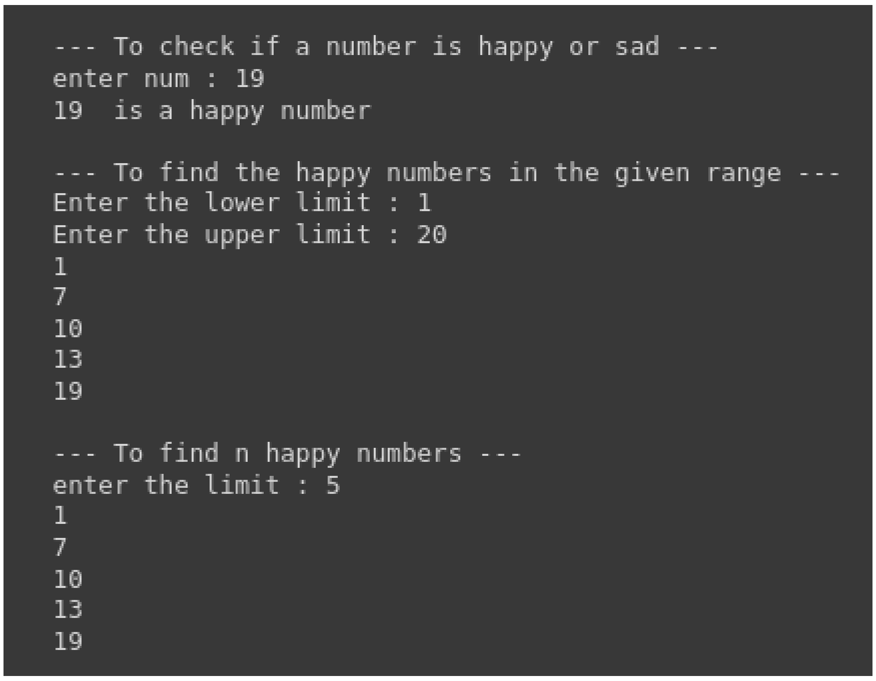
- Loops – for, while
  for loop - A for loop is used for iterating over a sequence
  while loop - While loop is used to execute a block of statements repeatedly until a given condition is satisfied. And when the condition becomes false, the line immediately after the loop in the program is executed.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

PROGRAM

```
#check if the number is happy
def happy(num):
  for i in range(1,101):
    sum = 0
    while num>0:
      remainder = num%10
      sum = sum + remainder**2
      num = num//10
#iterate 100 times if sum still not 0 returns false
    if sum==1:
      return True
    elif num<1:
      num = sum
    if i==100 :
      return False

#function to print happy numbers in the given range
def range_happy(lower , upper):
  happy_count = 0
  for i in range(lower , upper):
    if happy(i) == True:
      happy_count += 1
      print(i)
  if happy_count == 0:
      print("there are no happy numbers in the given range")

#function to print n happy numbers
def n_happy(limit):
  n = 0
  i=1
  while i>=1:
    if happy(i) == True:
      print(i)
      n+=1
    i+=1
    if n == limit:
      break

print("\n--- To check if a number is happy or sad ---")
num = int(input("enter num : "))

#check if happy number function is true or not
if happy(num) == True:
  print(num , " is a happy number")
else:
  print(num , " is a sad number ")
print("\n--- To find the happy numbers in the given range ---")

lower = int(input("Enter the lower limit : "))
upper = int(input("Enter the upper limit : "))
range_happy(lower,upper)

print("\n--- To find n happy numbers ---")
```

```
limit =  int(input("enter the limit : "))
n_happy(limit)
```

SAMPLE INPUT-OUTPUT



```
--- To check if a number is happy or sad ---
enter num : 19
19  is a happy number

--- To find the happy numbers in the given range ---
Enter the lower limit : 1
Enter the upper limit : 20
1
7
10
13
19

--- To find n happy numbers ---
enter the limit : 5
1
7
10
13
19
```

TEST CASES

Test Cases No.	Descripton	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

$https$:
$//github.com/nazalnihad/Python_Lab_Cycles/blob/main/Lab_cycle_1/Lab_cycle_qn4/qn4.py$

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

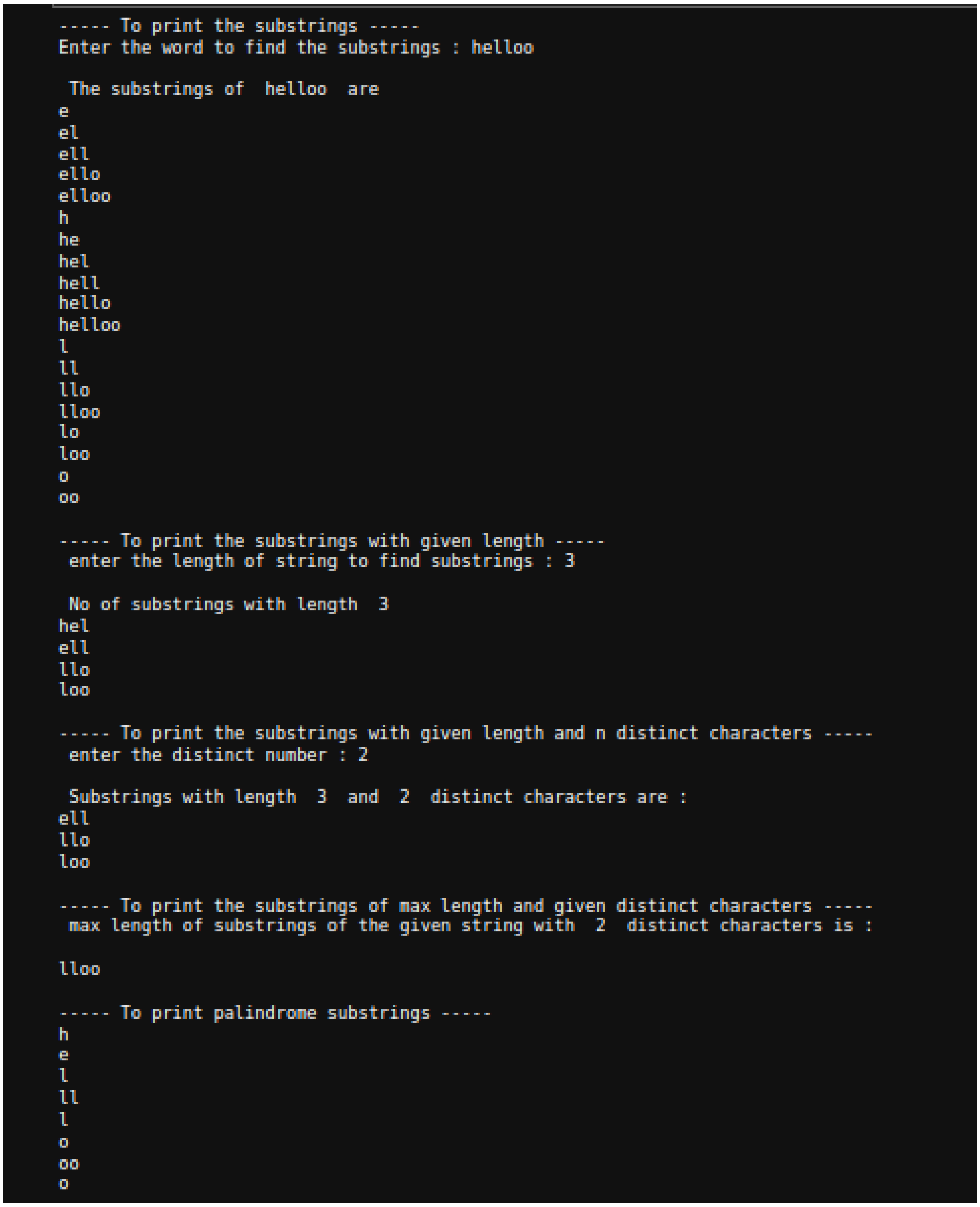
THEORY

- Strings - Strings are arrays of bytes representing Unicode characters.
- String functions - String functions area 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

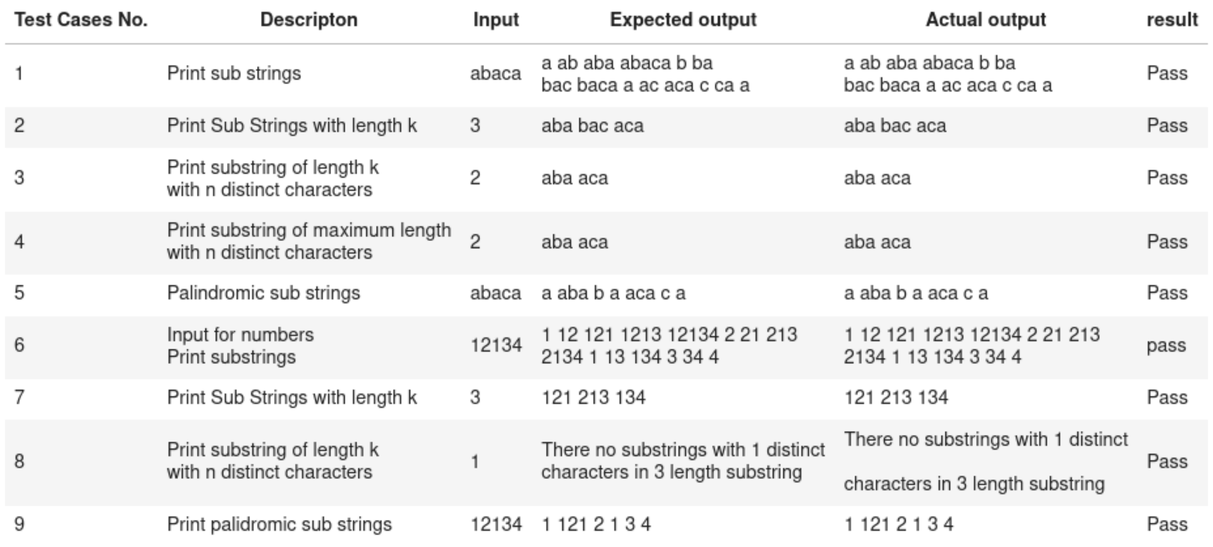
```
def sub_strings(text):
  #function to iterate through the string and find substrings
  sub_list = []
  for i in range(len(text)+1):
    for j in range(i+1,len(text)+1):
      sliced = text[i:j]
      sub_list.append(sliced)
  return sub_list

#function to find substrings from the string and print
def find_substrings(text):
  sub_list = sub_strings((text))
  #list is sorted and made into set to remove repetetion
  for i in sorted(set(sub_list)):
    print(i)

#function to find substrings with specified length
def k_sub(length,text):
  sub_list = sub_strings((text))
  for i in sub_list:
    if len(i)==length:
      print(i)

#function to find substrings with specified length and no of distinct characters
def n_distinct_and_k_length(distinct,length,text):
  sub_list = sub_strings((text))
  for i in sub_list:
    if len(i)==length:
      #make i to set to remove duplicate letters and check with given distinct number
      if len(set(i)) == distinct:
        print(i)

#function to find the substrings with max length and given no of distinct characters
def n_distinct_max_k_length(distinct,text):
  sub_list = sub_strings(text)

  #finds the largest string from list and find it's length
  max_length = max(sub_list,key = len)
  check_length = len(max_length)

  for i in sub_list:
    #iterate through list and check if an element matches the requirements and
    temporarily store it
      if len(i) == check_length:
        if len(set(i))==distinct:
            temp = i
        #if no element matches , max length is reduced , the process is
        repeated until an element becomes a match
        elif check_length>len(set(i)):
          check_length -= 1

  #check if elements of original list matches the max length and the distinct
  character condition
  for i in sub_list:
    if len(i) == len(temp):
      if len(set(i)) == distinct:
        print(i)
#function to print palindrome substrings
def palindrome_substrings(text):
  sub_list = sub_strings(text)
  for i in sub_list:
    if i == i[::-1]: #check if reverse of word is same
      print(i)

#print all the substrings of the given string
print("\n----- To print the substrings -----")
text = str(input("Enter the word to find the substrings : "))
print("\n The substrings of ",text , " are ")
find_substrings(text)

#print substrings of the given length
print("\n----- To print the substrings with given length -----")
length = int(input(" enter the length of string to find substrings : "))
print("\n No of substrings with length ", length,)
k_sub(length,text)

#print substrings of given length and distinct characters
print("\n----- To print the substrings with given length and n distinct
characters -----")
distinct = int(input(" enter the distinct number : "))
print("\n Substrings with length ",length," and ",distinct," distinct
characters are : ")
n_distinct_and_k_length(distinct,length,text)

#print substrings of max length and distinct characters
print("\n----- To print the substrings of max length and given distinct
characters -----")
print(" max length of substrings of the given string with ",distinct," distinct
characters is : \n")
n_distinct_max_k_length(distinct,text)

#print palindrome substrings
print("\n----- To print palindrome substrings -----")
palindrome_substrings(text)
```

SAMPLE INPUT-OUTPUT



```
----- To print the substrings -----
Enter the word to find the substrings : helloo

 The substrings of  helloo  are
e
el
ell
ello
elloo
h
he
hel
hell
hello
helloo
l
ll
llo
lloo
lo
loo
o
oo

----- To print the substrings with given length -----
 enter the length of string to find substrings : 3

 No of substrings with length  3
hel
ell
llo
loo

----- To print the substrings with given length and n distinct characters -----
 enter the distinct number : 2

 Substrings with length  3  and  2  distinct characters are :
ell
llo
loo

----- To print the substrings of max length and given distinct characters -----
 max length of substrings of the given string with  2  distinct characters is :

lloo

----- To print palindrome substrings -----
h
e
l
ll
l
o
oo
o
```

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

$https$:
$//github.com/nazalnihad/Python_{L}ab_{C}ycles/blob/main/Lab_{c}ycle_{1}/Lab_{c}ycle_{q}n5/qn5.py$

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.

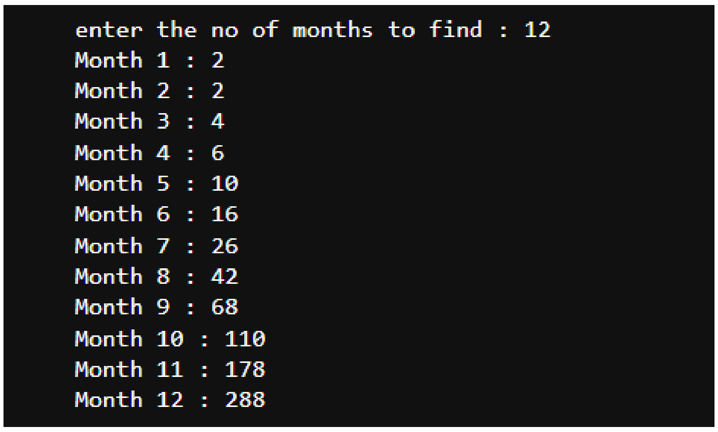
THEORY

- 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 instruction s 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

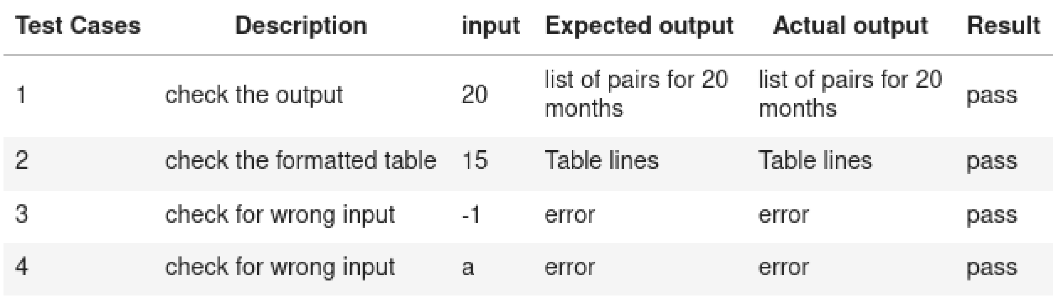
```
try:
    ask = int(input("Enter the number of months to find: "))
except ValueError:
    print("Error")

if ask==1:
  print(2)
elif ask>1:
  month1 = 2
  month2 = 2
  total,i=0,0
  rabbits = [2,2]
  for i in range(ask-2):
    total = month2+month1
    rabbits.append(total)
    month1 = month2
    month2 = total
    i+=1
else:
    print("error")

  count=0
  for k in rabbits:
    print("Month ",rabbits.index(k)+1," : ",k)
    if k==2:
      count+=1
    if count>0:
      rabbits[rabbits.index(k)]=-1
```

SAMPLE INPUT-OUTPUT



```
enter the no of months to find : 12
Month 1 : 2
Month 2 : 2
Month 3 : 4
Month 4 : 6
Month 5 : 10
Month 6 : 16
Month 7 : 26
Month 8 : 42
Month 9 : 68
Month 10 : 110
Month 11 : 178
Month 12 : 288
```

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

$https://github.com/nazalnihad/Python_{L}ab_{C}ycles/blob/main/Lab_{c}ycle_{2}/QN1/qn1.py$

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

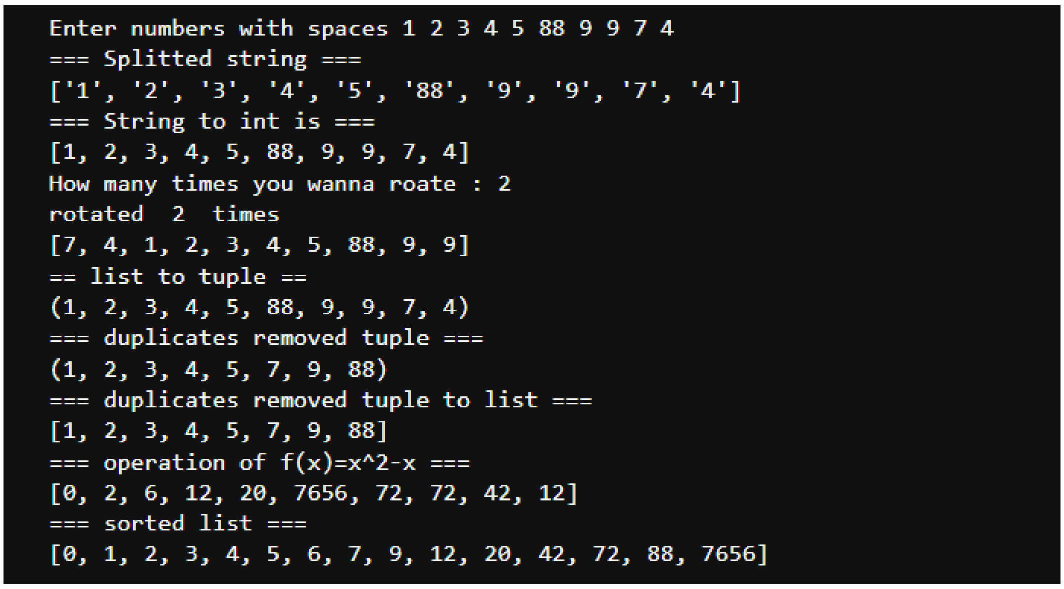
- List - A list in Python is used to store the sequence of various types of data.It is created by placing elements inside square brackets [] , separated by commas.
- tuple - A Tuple is a collection of Python objects separated by commas.They are used to store multiple items in a single variable.
- set - 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

```
string_int = str(input("Enter numbers with spaces "))
string_split = string_int.split()
print("=== Splitted string ===")
print(string_split)
string_to_int = [int(i) for i in string_split]
print("=== String to int is ===")
print(string_to_int)

ask = int(input("How many times you wanna roate : "))
rotate = (string_to_int[-ask:])
left = (string_to_int[0:len(string_to_int)-ask])
```

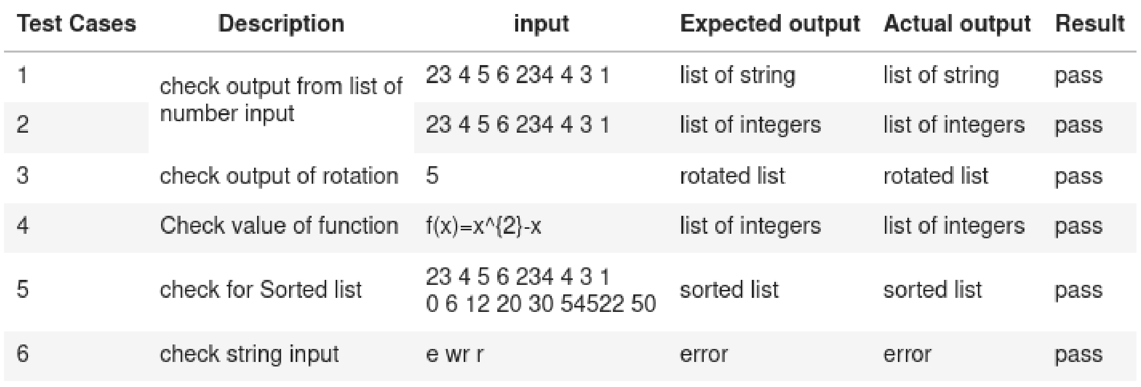
```
rotated_list = rotate+left
print("rotated ",ask," times ")
print(rotated_list)

print("== list to tuple ==")
tupled_list = tuple(string_to_int)
print(tupled_list)

print("=== duplicates removed tuple ===")
rm_dupe = tuple(set(tupled_list))
print(rm_dupe)

print("=== duplicates removed tuple to list ===")
no_dupe_list = list(rm_dupe)
print(no_dupe_list)

print("=== operation of f(x)=x^2-x ===")
op_list = [i*i - i for i in string_to_int]
print(op_list)

print("=== sorted list ===")
sorted_list = sorted(set(no_dupe_list+op_list))
print(sorted_list)
```

SAMPLE INPUT-OUTPUT



```
Enter numbers with spaces 1 2 3 4 5 88 9 9 7 4
=== Splitted string ===
['1', '2', '3', '4', '5', '88', '9', '9', '7', '4']
=== String to int is ===
[1, 2, 3, 4, 5, 88, 9, 9, 7, 4]
How many times you wanna roate : 2
rotated  2  times
[7, 4, 1, 2, 3, 4, 5, 88, 9, 9]
== list to tuple ==
(1, 2, 3, 4, 5, 88, 9, 9, 7, 4)
=== duplicates removed tuple ===
(1, 2, 3, 4, 5, 7, 9, 88)
=== duplicates removed tuple to list ===
[1, 2, 3, 4, 5, 7, 9, 88]
=== operation of f(x)=x^2-x ===
[0, 2, 6, 12, 20, 7656, 72, 72, 42, 12]
=== sorted list ===
[0, 1, 2, 3, 4, 5, 6, 7, 9, 12, 20, 42, 72, 88, 7656]
```

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 50	sorted list	sorted list	pass
6	check string input	e wr r	error	error	pass

RESULT

Program executed Successfully and the output is obtained.

GIT LINK

$https : //github.com/nazalnihad/Python_{L}ab_{C}ycles/blob/main/Lab_{c}ycle_{2}/QN2/qn2.py$

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

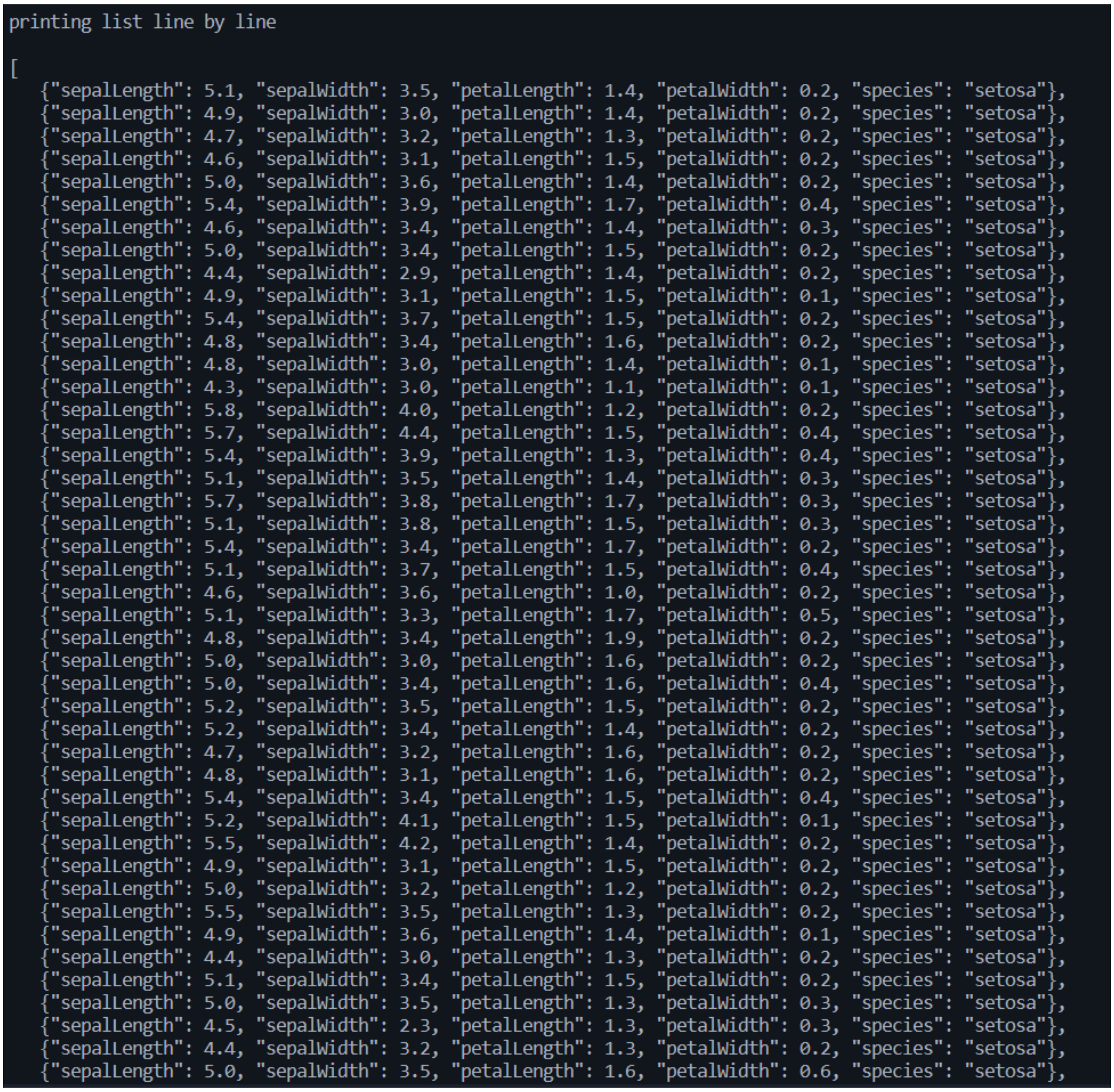
- JSON - JavaScript Object Notation (JSON) is a standard text-based format for representing structured data based on JavaScript object syntax. It is commonly used for transmitting data in web applications.
- dictionary - A dictionary is a collection which is ordered, changeable and do not allow duplicates.Dictionaries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

PROGRAM

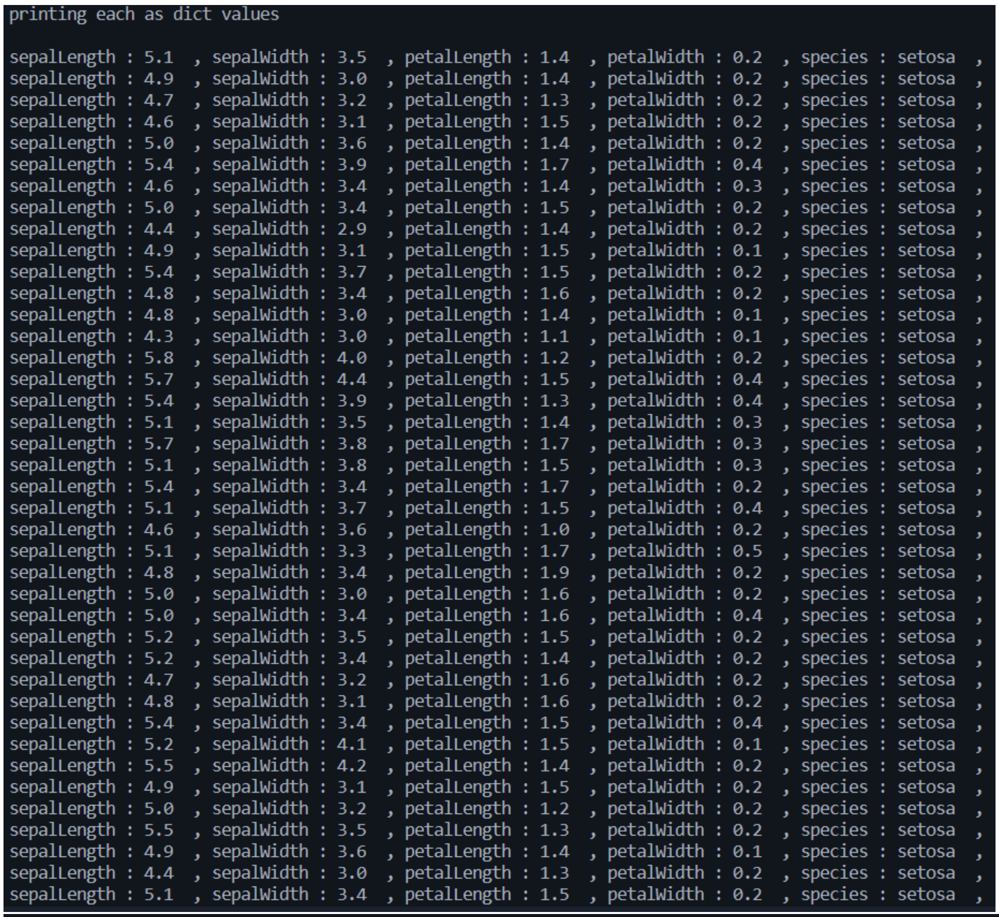
```
import json
file_path = input("Enter file path : ")

#read as list
def list_read(file_path):
    print("\nprinting list line by line \n")
    list_file = open(file_path,'r')
    list_done = list_file.readlines()
    for k in list_done:
        print(k,end=" ")


#read as dict
def list_to_dict(file_path):
    print("\nprinting each as dict values \n")
    list_dict = open(file_path,'r')
    #makes to python file so we can use python operations
    dicts = json.load(list_dict)
    for i in dicts:
        for key,value in i.items():
            print("{} : {} ".format(key,value),end=" , ")
        print("")



#make list if species is setosa and display details
def setosa(path):
    print("\n Details of setosa \n")
    with open(path,'r') as f:
        data = json.load(f)
    a = []
    for flowers in data:
        if (flowers['species']=="setosa"):
            a.append(flowers)
    for m in a:
        print(m)

#to print min petal area and max sepal area in each species
def min_max_area(path):
    print("\nMax and Min area of petal and sepal in each species \n")
    with open(path,'r') as f:
        data = json.loads(f.read())
    species_list = []
    for area in data:
        species_list.append(area['species'])
    species_list = set(species_list)

    petal_list = []
    sepal_list = []
    # print(species_list)
    for i in species_list:
        for j in data:
            if i==j['species']:
                petal_list.append(j['petalLength']*j['petalWidth'])
                sepal_list.append(j['sepalLength']*j['sepalWidth'])
        print("\nMax and min areas of ",i,"\n")
        print("max are of ",i," sepals = ",round(max(sepal_list),2))
        print("min area of ",i," petals = ",round(min(petal_list),2),"\n")
        sepal_list.clear()
        petal_list.clear()
```

```
# Sort the list of dictionaries according to the total area
def sort_with_area(file_path):
    print("\nAdding new key TotalArea and sorting according to it \n")
    area_list = open(file_path,'r')
    area_sort = json.load(area_list)
    for i in area_sort:
        i['TotalArea'] =
        round(i['sepalLength']*i['sepalWidth']+i['petalLength']
        *i['petalWidth'],2)
        # round(i['TotalArea'],2)
    area_sort.sort(key = lambda x:x['TotalArea'])
    for k in area_sort:
        print(k)

list_read(file_path)
list_to_dict(file_path)
setosa(file_path)
min_max_area(file_path)
sort_with_area(file_path)
```

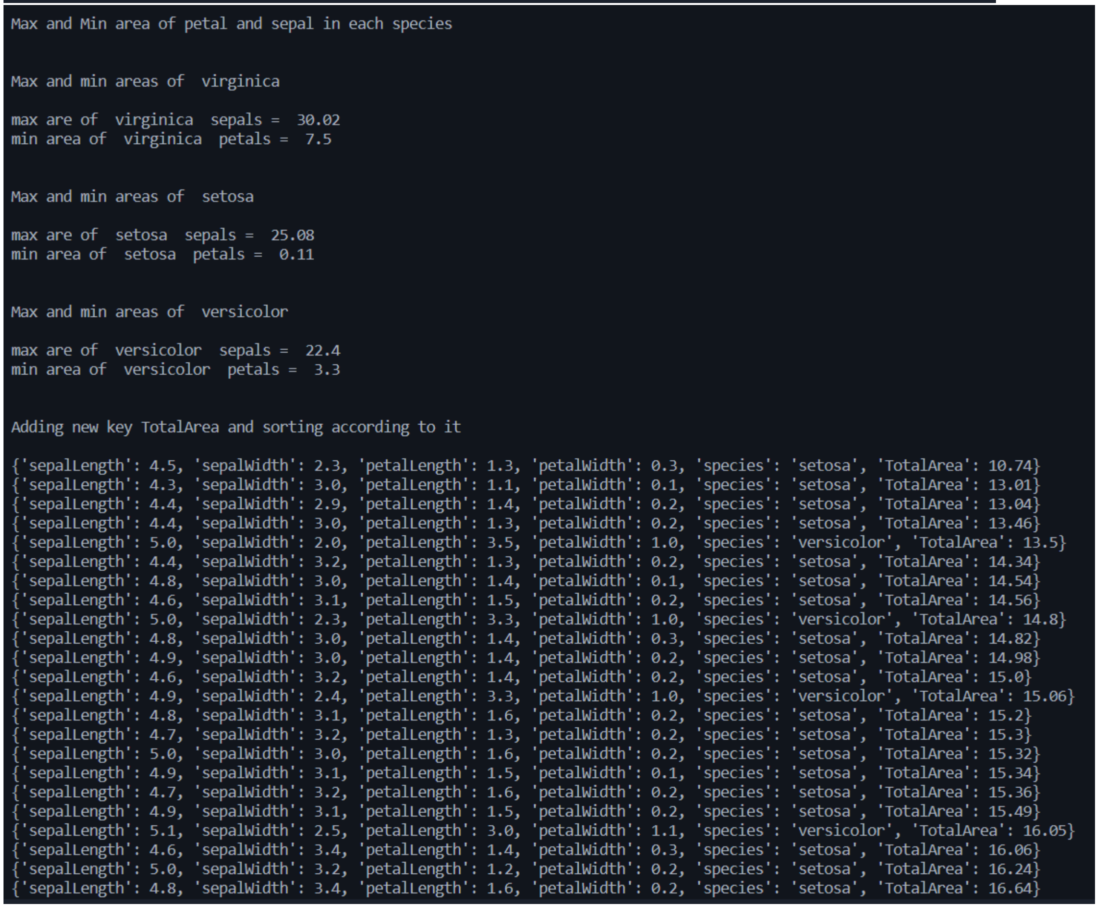
SAMPLE INPUT-OUTPUT



```
printing list line by line

[
   {"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"},
   {"sepalLength": 4.8, "sepalWidth": 3.1, "petalLength": 1.6, "petalWidth": 0.2, "species": "setosa"},
   {"sepalLength": 5.4, "sepalWidth": 3.4, "petalLength": 1.5, "petalWidth": 0.4, "species": "setosa"},
   {"sepalLength": 5.2, "sepalWidth": 4.1, "petalLength": 1.5, "petalWidth": 0.1, "species": "setosa"},
   {"sepalLength": 5.5, "sepalWidth": 4.2, "petalLength": 1.4, "petalWidth": 0.2, "species": "setosa"},
   {"sepalLength": 4.9, "sepalWidth": 3.1, "petalLength": 1.5, "petalWidth": 0.2, "species": "setosa"},
   {"sepalLength": 5.0, "sepalWidth": 3.2, "petalLength": 1.2, "petalWidth": 0.2, "species": "setosa"},
   {"sepalLength": 5.5, "sepalWidth": 3.5, "petalLength": 1.3, "petalWidth": 0.2, "species": "setosa"},
   {"sepalLength": 4.9, "sepalWidth": 3.6, "petalLength": 1.4, "petalWidth": 0.1, "species": "setosa"},
   {"sepalLength": 4.4, "sepalWidth": 3.0, "petalLength": 1.3, "petalWidth": 0.2, "species": "setosa"},
   {"sepalLength": 5.1, "sepalWidth": 3.4, "petalLength": 1.5, "petalWidth": 0.2, "species": "setosa"},
   {"sepalLength": 5.0, "sepalWidth": 3.5, "petalLength": 1.3, "petalWidth": 0.3, "species": "setosa"},
   {"sepalLength": 4.5, "sepalWidth": 2.3, "petalLength": 1.3, "petalWidth": 0.3, "species": "setosa"},
   {"sepalLength": 4.4, "sepalWidth": 3.2, "petalLength": 1.3, "petalWidth": 0.2, "species": "setosa"},
   {"sepalLength": 5.0, "sepalWidth": 3.5, "petalLength": 1.6, "petalWidth": 0.6, "species": "setosa"},
```

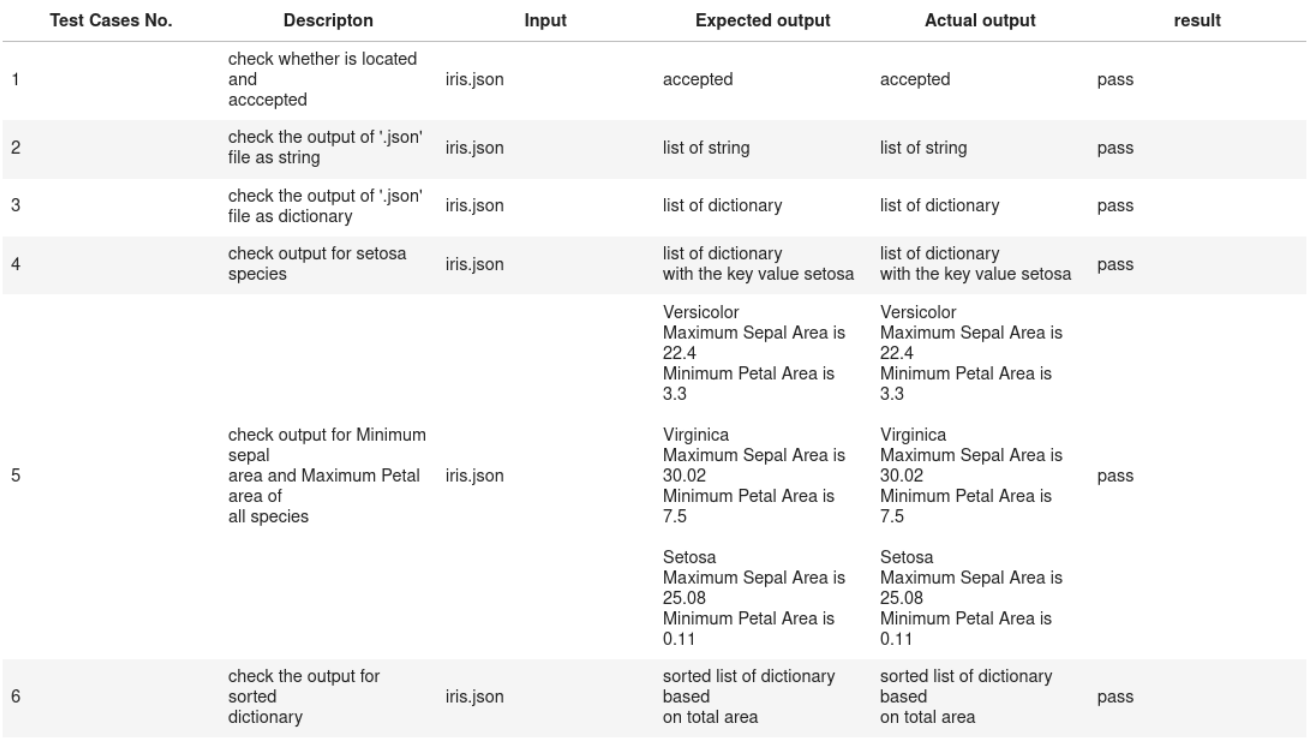


```
printing each as dict values

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  ,
sepalLength : 4.8  , sepalWidth : 3.1  , petalLength : 1.6  , petalWidth : 0.2  , species : setosa  ,
sepalLength : 5.4  , sepalWidth : 3.4  , petalLength : 1.5  , petalWidth : 0.4  , species : setosa  ,
sepalLength : 5.2  , sepalWidth : 4.1  , petalLength : 1.5  , petalWidth : 0.1  , species : setosa  ,
sepalLength : 5.5  , sepalWidth : 4.2  , petalLength : 1.4  , petalWidth : 0.2  , species : setosa  ,
sepalLength : 4.9  , sepalWidth : 3.1  , petalLength : 1.5  , petalWidth : 0.2  , species : setosa  ,
sepalLength : 5.0  , sepalWidth : 3.2  , petalLength : 1.2  , petalWidth : 0.2  , species : setosa  ,
sepalLength : 5.5  , sepalWidth : 3.5  , petalLength : 1.3  , petalWidth : 0.2  , species : setosa  ,
sepalLength : 4.9  , sepalWidth : 3.6  , petalLength : 1.4  , petalWidth : 0.1  , species : setosa  ,
sepalLength : 4.4  , sepalWidth : 3.0  , petalLength : 1.3  , petalWidth : 0.2  , species : setosa  ,
sepalLength : 5.1  , sepalWidth : 3.4  , petalLength : 1.5  , petalWidth : 0.2  , species : setosa  ,
```



```
Max and Min area of petal and sepal in each species


Max and min areas of  virginica

max are of  virginica  sepals =  30.02
min area of  virginica  petals =  7.5


Max and min areas of  setosa

max are of  setosa  sepals =  25.08
min area of  setosa  petals =  0.11


Max and min areas of  versicolor

max are of  versicolor  sepals =  22.4
min area of  versicolor  petals =  3.3


Adding new key TotalArea and sorting according to it

{'sepalLength': 4.5, 'sepalWidth': 2.3, 'petalLength': 1.3, 'petalWidth': 0.3, 'species': 'setosa', 'TotalArea': 10.74}
{'sepalLength': 4.3, 'sepalWidth': 3.0, 'petalLength': 1.1, 'petalWidth': 0.1, 'species': 'setosa', 'TotalArea': 13.01}
{'sepalLength': 4.4, 'sepalWidth': 2.9, 'petalLength': 1.4, 'petalWidth': 0.2, 'species': 'setosa', 'TotalArea': 13.04}
{'sepalLength': 4.4, 'sepalWidth': 3.0, 'petalLength': 1.3, 'petalWidth': 0.2, 'species': 'setosa', 'TotalArea': 13.46}
{'sepalLength': 5.0, 'sepalWidth': 2.0, 'petalLength': 3.5, 'petalWidth': 1.0, 'species': 'versicolor', 'TotalArea': 13.5}
{'sepalLength': 4.4, 'sepalWidth': 3.2, 'petalLength': 1.3, 'petalWidth': 0.2, 'species': 'setosa', 'TotalArea': 14.34}
{'sepalLength': 4.8, 'sepalWidth': 3.0, 'petalLength': 1.4, 'petalWidth': 0.1, 'species': 'setosa', 'TotalArea': 14.54}
{'sepalLength': 4.6, 'sepalWidth': 3.1, 'petalLength': 1.5, 'petalWidth': 0.2, 'species': 'setosa', 'TotalArea': 14.56}
{'sepalLength': 5.0, 'sepalWidth': 2.3, 'petalLength': 3.3, 'petalWidth': 1.0, 'species': 'versicolor', 'TotalArea': 14.8}
{'sepalLength': 4.8, 'sepalWidth': 3.0, 'petalLength': 1.4, 'petalWidth': 0.3, 'species': 'setosa', 'TotalArea': 14.82}
{'sepalLength': 4.9, 'sepalWidth': 3.0, 'petalLength': 1.4, 'petalWidth': 0.2, 'species': 'setosa', 'TotalArea': 14.98}
{'sepalLength': 4.6, 'sepalWidth': 3.2, 'petalLength': 1.4, 'petalWidth': 0.2, 'species': 'setosa', 'TotalArea': 15.0}
{'sepalLength': 4.9, 'sepalWidth': 2.4, 'petalLength': 3.3, 'petalWidth': 1.0, 'species': 'versicolor', 'TotalArea': 15.06}
{'sepalLength': 4.8, 'sepalWidth': 3.1, 'petalLength': 1.6, 'petalWidth': 0.2, 'species': 'setosa', 'TotalArea': 15.2}
{'sepalLength': 4.7, 'sepalWidth': 3.2, 'petalLength': 1.3, 'petalWidth': 0.2, 'species': 'setosa', 'TotalArea': 15.3}
{'sepalLength': 5.0, 'sepalWidth': 3.0, 'petalLength': 1.6, 'petalWidth': 0.2, 'species': 'setosa', 'TotalArea': 15.32}
{'sepalLength': 4.9, 'sepalWidth': 3.1, 'petalLength': 1.5, 'petalWidth': 0.1, 'species': 'setosa', 'TotalArea': 15.34}
{'sepalLength': 4.7, 'sepalWidth': 3.2, 'petalLength': 1.6, 'petalWidth': 0.2, 'species': 'setosa', 'TotalArea': 15.36}
{'sepalLength': 4.9, 'sepalWidth': 3.1, 'petalLength': 1.5, 'petalWidth': 0.2, 'species': 'setosa', 'TotalArea': 15.49}
{'sepalLength': 5.1, 'sepalWidth': 2.5, 'petalLength': 3.0, 'petalWidth': 1.1, 'species': 'versicolor', 'TotalArea': 16.05}
{'sepalLength': 4.6, 'sepalWidth': 3.4, 'petalLength': 1.4, 'petalWidth': 0.3, 'species': 'setosa', 'TotalArea': 16.06}
{'sepalLength': 5.0, 'sepalWidth': 3.2, 'petalLength': 1.2, 'petalWidth': 0.2, 'species': 'setosa', 'TotalArea': 16.24}
{'sepalLength': 4.8, 'sepalWidth': 3.4, 'petalLength': 1.6, 'petalWidth': 0.2, 'species': 'setosa', 'TotalArea': 16.64}
```

TEST CASES



Test Cases No.	Descripton	Input	Expected output	Actual output	result
1	check whether is located and acccepted	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

$https://github.com/nazalnihad/Python_Lab_Cycles/blob/main/Lab_cycle_2/QN3/qn3.py$

BOX CLASS FOR SHAPES

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.

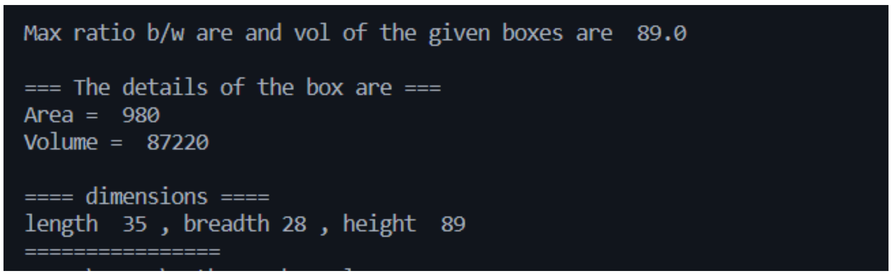
THEORY

- Class - 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

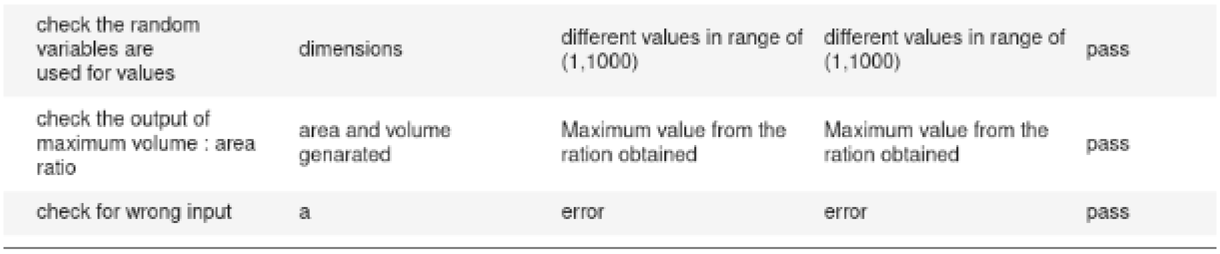
```
import random
class Box:
  def __init__(self , length=1, breadth=1 , height=1):
    if(length==breadth):
        self.length=length
        self.breadth=breadth
        self.height=height
    elif(breadth==height):
      self.length=height
      self.breadth=breadth
      self.height=length
    elif(length==height):
      self.length=length
      self.breadth=height
      self.height=breadth
    else:
      self.length=length
      self.breadth=breadth
      self.height=height


  def area(self):
    self.area = self.length*self.breadth
    return self.area

  def vol(self):
    self.vol = self.length*self.breadth*self.height
    return self.vol
bx =
[Box(random.randint(1,1000),random.randint(1,1000),random.randint(1,1000))
for i in range(10)]

area_box = [_.area() for _ in bx]
vol_box = [_.vol() for _ in bx]
ratio_box = [round(y/x,2) for x,y in zip(area_box,vol_box)]

ind = ratio_box.index(max(ratio_box))
print("\nMax ratio b/w are and vol of the given boxes are ",max(ratio_box))
print("\n=== The details of the box are ===")
print("Area = ", area_box[ind])
print("Volume = ",vol_box[ind])
print("\n==== dimensions ====\nlength ",bx[ind].length,", breadth"
,bx[ind].breadth,", height ",bx[ind].height,"\n================")
```

SAMPLE INPUT-OUTPUT



```
Max ratio b/w are and vol of the given boxes are  89.0

=== The details of the box are ===
Area =  980
Volume =  87220

==== dimensions ====
length  35 , breadth 28 , height  89
================
```

TEST CASES



check the random variables are used for values	dimensions	different values in range of (1,1000)	different values in range of (1,1000)	pass
check the output of maximum volume : area ratio	area and volume genarated	Maximum value from the ration obtained	Maximum value from the ration obtained	pass
check for wrong input	a	error	error	pass

RESULT

Program executed Successfully and the output is obtained.

GIT LINK

$https : //github.com/nazalnihad/Python_{L}ab_{C}ycles/blob/main/Lab_{c}ycle_{2}/QN4/test.py$

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

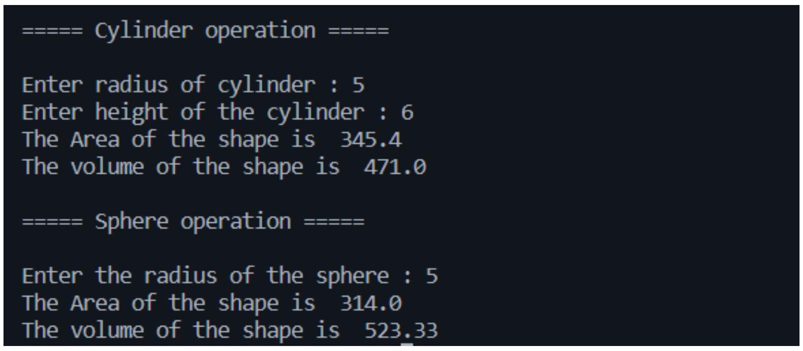
- Inheritance - Inheritance refers to defining a new class with little or no modification to an existing class. The new class is called derived (child) class and the one from which it inherits is called the base (parent) class.

PROGRAM

```
class three_d_shapes(): #base class
    def printVolume(self):
        print("The volume of the shape is ",self.volume)
    def prinArea(self):
        print("The Area of the shape is ",self.area)

class cylinders(three_d_shapes):  #derived class
    def __init__(self,r,h): #initialising values
        self.r = r
        self.h = h

    def area(self):
        self.area = round(2*3.14*self.r*(self.r+self.h),2)
    def volume(self):
        self.volume = round(3.14*self.r*self.r*self.h,2)

class sphere(three_d_shapes): #derived class
    def __init__(self,r):
        self.r = r
    def area(self):
        self.area = round(4*3.14*self.r*self.r,2)
    def volume(self):
        self.volume = round((4/3)*3.14*self.r*self.r*self.r,2)

#cylinders operation
print("\n===== Cylinder operation =====\n")
```

```
cylinder1 = cylinders(int(input("Enter radius of cylinder :
")),int(input("Enter height of the cylinder : ")))
cylinder1.area()
cylinder1.volume()
cylinder1.prinArea()
cylinder1.printVolume()

#sphere operation
print("\n===== Sphere operation =====\n")
sphere1 = sphere(int(input("Enter the radius of the sphere :
")))
sphere1.area()
sphere1.volume()
sphere1.prinArea()
sphere1.printVolume()
```

SAMPLE INPUT-OUTPUT



```
===== Cylinder operation =====

Enter radius of cylinder : 5
Enter height of the cylinder : 6
The Area of the shape is  345.4
The volume of the shape is  471.0

===== Sphere operation =====

Enter the radius of the sphere : 5
The Area of the shape is  314.0
The volume of the shape is  523.33
```

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

$https : //github.com/nazalnihad/Python_Lab_Cycles/blob/main/Lab_cycle_2/QN5/qn5.py$

TIC TAC TOE

AIM

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

THEORY

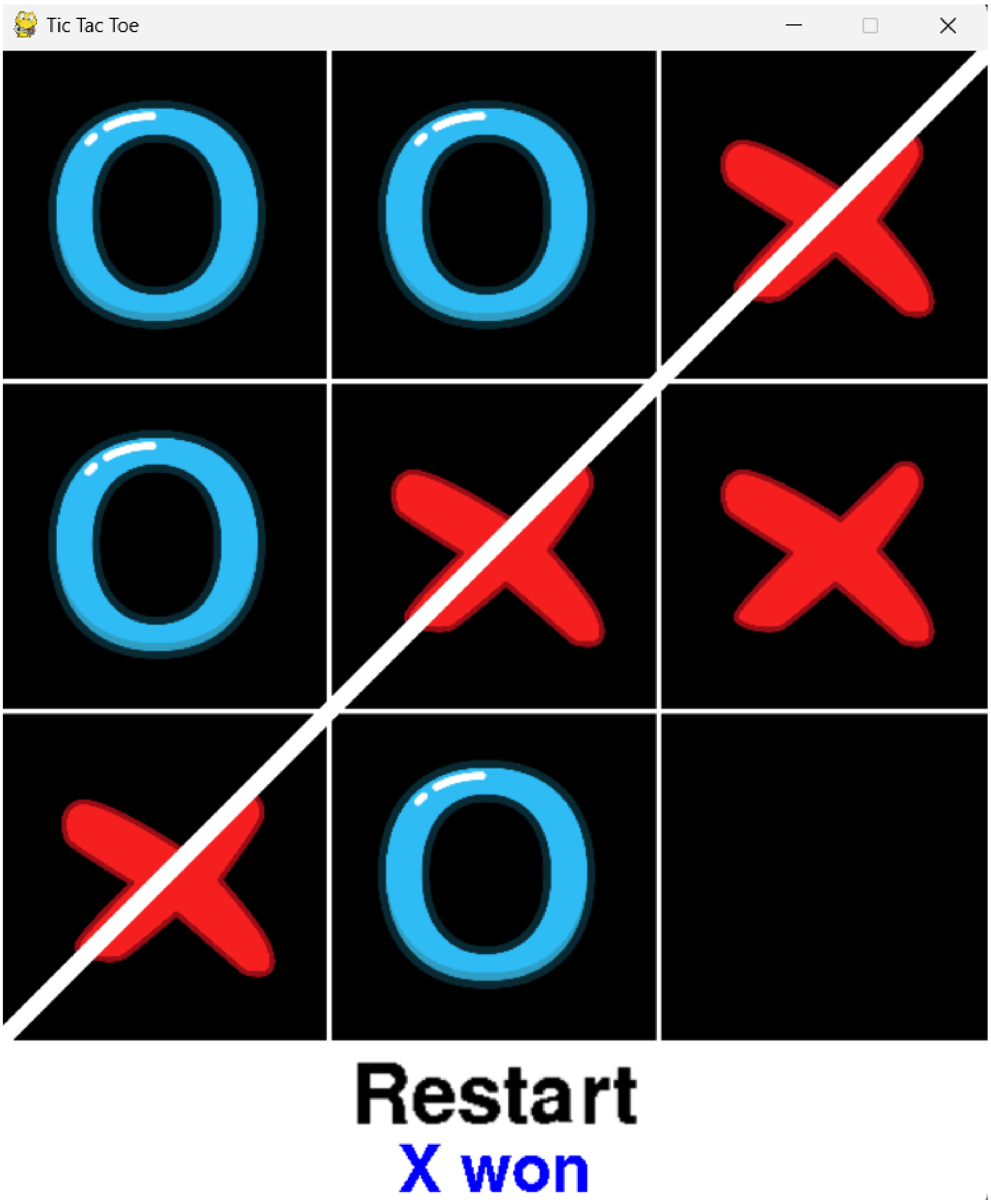
- Pygame library - Pygame is a cross-platform set of Python modules designed for writing video games.

PROGRAM

```
import pygame
from random import randint
from sys import exit
# Initialize Pygame
pygame.init()

size = (600, 700)
line_width = 15
font = pygame.font.Font("freesansbold.ttf", 40)
screen = pygame.display.set_mode(size)
pygame.display.set_caption("Tic Tac Toe")

# main board which is updated later according to conditions
board = [[0, 0, 0],
         [0, 0, 0],
         [0, 0, 0]]

# check empty space



def not_filled(row, col):
    if board[row][col] == 0:
        return True
    else:
        return False

# draw lines on the board
```

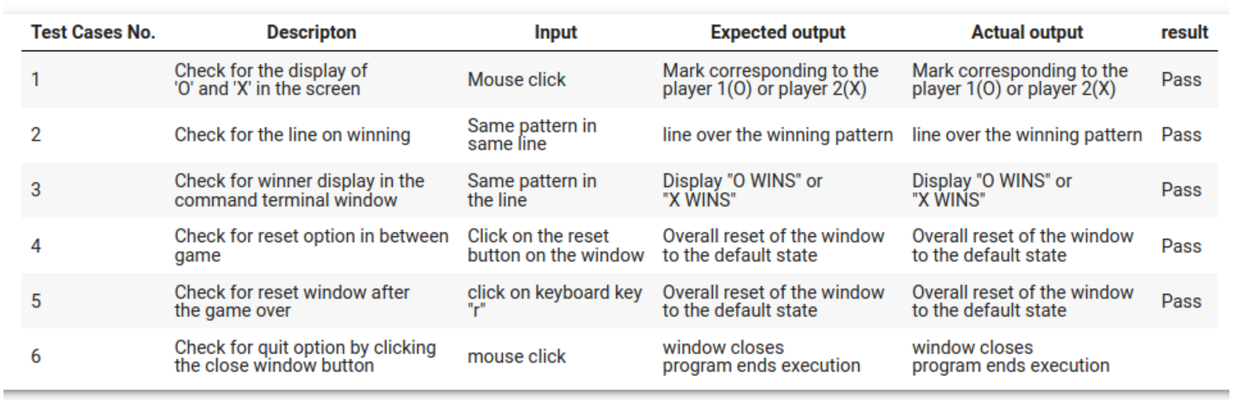
```
def draw_board():
    pygame.draw.line(screen, 'white', (0, 200), (600, 200), 3)
    pygame.draw.line(screen, 'white', (0, 400), (600, 400), 3)
    pygame.draw.line(screen, 'white', (200, 0), (200, 600), 3)
    pygame.draw.line(screen, 'white', (400, 0), (400, 600), 3)

# import x image and draw at position


def drawX(cell, x_pos=0, y_pos=0):
    x = pygame.image.load(
        'Lab_cycle_3/QN1/img/x.png').convert_alpha()
    x = pygame.transform.rotozoom(x, 0, 0.175)
    cell.blit(x, (x_pos, y_pos))

# import o img and draw


def drawO(cell, x_pos=0, y_pos=0):
    o = pygame.image.load(
        'Lab_cycle_3/QN1/img/o.png').convert_alpha()
    o = pygame.transform.rotozoom(o, 0, 0.3)
    cell.blit(o, (x_pos, y_pos))



def checkwin():
    for i in range(3):
        # horizontal condition
        if board[0][i] == board[1][i] == board[2][i] == player:
            pygame.draw.line(screen, 'white', (0, i*200+100),
                             (600, i*200+100), line_width)
            # print(player, "won")
            return True
        # vertical condition
        elif board[i][0] == board[i][1] == board[i][2] == player:
            pygame.draw.line(screen, 'white', (i*200+100, 0),
                             (i*200+100, 600), line_width)
            return True
    # diagonal condition
    if board[0][0] == board[1][1] == board[2][2] == player:
        pygame.draw.line(screen, 'white', (0, 0), (600, 600), line_width)
        return True
    if board[0][2] == board[1][1] == board[2][0] == player:
        pygame.draw.line(screen, 'white', (600, 0), (0, 600), line_width)
        return True


# randomly choose first
player = randint(1, 2)
win_check = False


def check_tie():
    # check if the board is filled
    flag = 0
    for row in range(3):
        for col in range(3):
            if board[row][col] != 0:
                flag += 1
    if flag == 9:
        return True


def end_button():
    # restart button
    game_msg = font.render('Restart', False, 'black')
    game_msg = pygame.transform.rotozoom(game_msg, 0, 1.25)
    game_msg_rect = game_msg.get_rect(center=(300, 635))
    pygame.draw.rect(screen, 'white', (0, 600, 600, 100))
    screen.blit(game_msg, game_msg_rect)
    if not checkwin():
        if player == 1:
            player_msg = font.render('X turn', False, 'black')
        elif player == 2:
            player_msg = font.render('O turn', False, 'black')
    if check_tie():
        player_msg = font.render('TIE', False, 'blue')
    if checkwin():
        if player == 1:
            player_msg = font.render('X won', False, 'blue')
        elif player == 2:
            player_msg = font.render('O won', False, 'blue')
    player_msg_rect = player_msg.get_rect(midbottom=(300, 700))
    screen.blit(player_msg, player_msg_rect)


def restart():
    global board
    board = [[0, 0, 0],
             [0, 0, 0],
             [0, 0, 0]]
    win_check = False
    global player
    player = randint(1, 2)
    screen.fill('black')
    draw_board()


# game loop starts
clock = pygame.time.Clock()
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            exit()
        if event.type == pygame.MOUSEBUTTONDOWN:
            # find mouse click position change to single digit
            and divide to get simplified cooridnates
            x_point = event.pos[0]//100
            y_point = event.pos[1]//100
            m_row = x_point//2
            m_col = y_point//2
            # print(x_point, y_point)
            # print(m_row, m_col)
            if x_point > 5 or y_point > 5:
                restart()
                continue
            win_check = not checkwin()
            if win_check and not_filled(m_row, m_col):
                if player == 1:
                    drawX(screen, (m_row)*200, (m_col)*200)
                    board[m_row][m_col] = 1
                    if checkwin():
                        # checkwin()
                        win_check = False
                        # print("Player 1 won")
                    else:
                        player = 2
                elif player == 2:
                    draw0(screen, (m_row)*200+20, (m_col)*200+20)
                    board[m_row][m_col] = 2
                    if checkwin():
                        # checkwin()
                        win_check = False
                        # print("player 2 won")
                    else:
                        player = 1
                # print(board)

    draw_board()
    end_button()

    # Update the display
    pygame.display.update()
    clock.tick(60)
```

SAMPLE INPUT-OUTPUT



X won

TEST CASES



Test Cases No.	Descripton	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

$https$:
$//github.com/nazalnihad/Python_Lab_Cycles/blob/main/Lab_cycle_3/QN1/QN1_tic_tac_toe.py$

PRINCIPAL COMPONENT ANALYSIS ON MATRIX

AIM

Implement Principle Component Analysis(PCA) of a matrix.

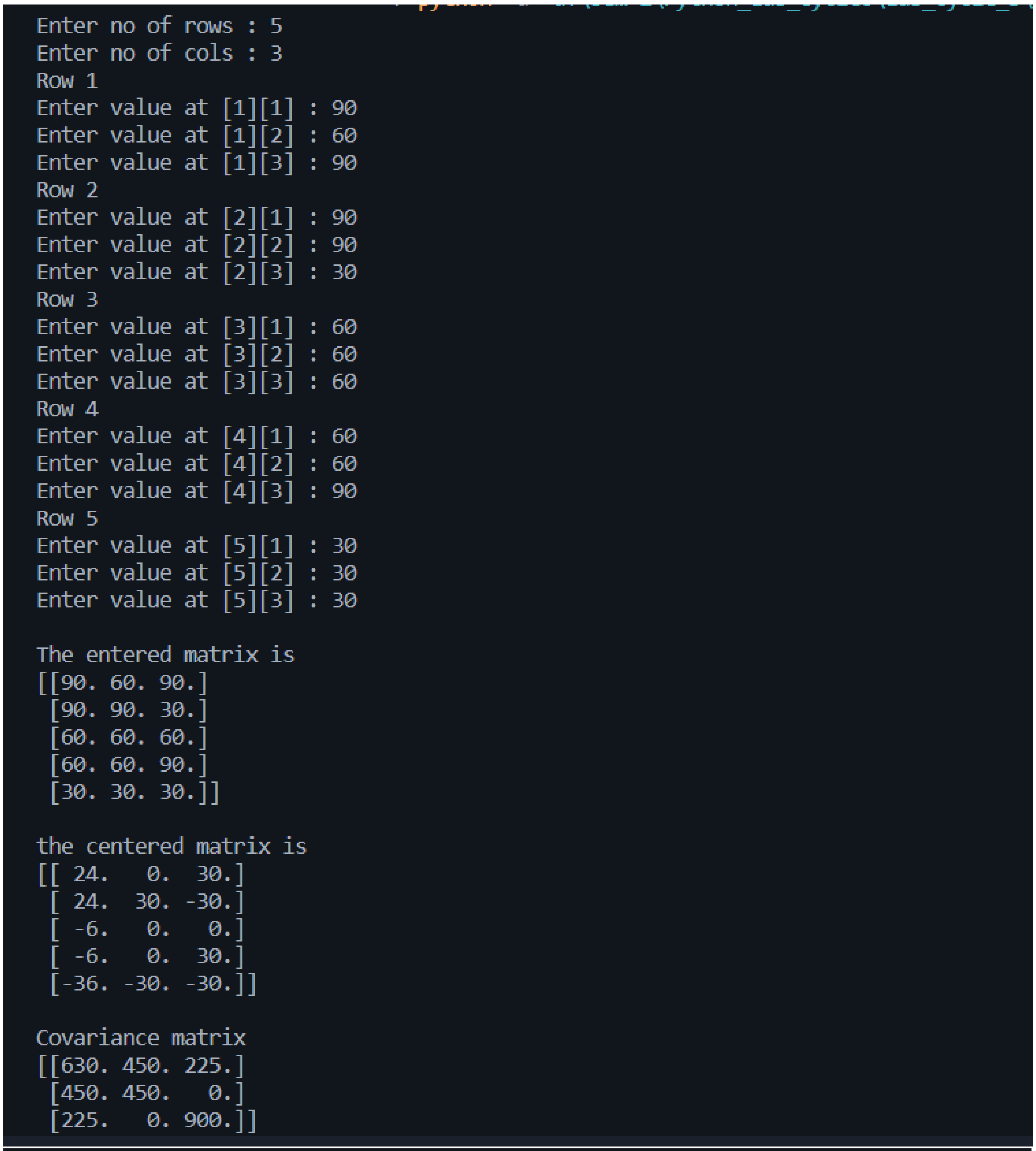
THEORY

- 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.

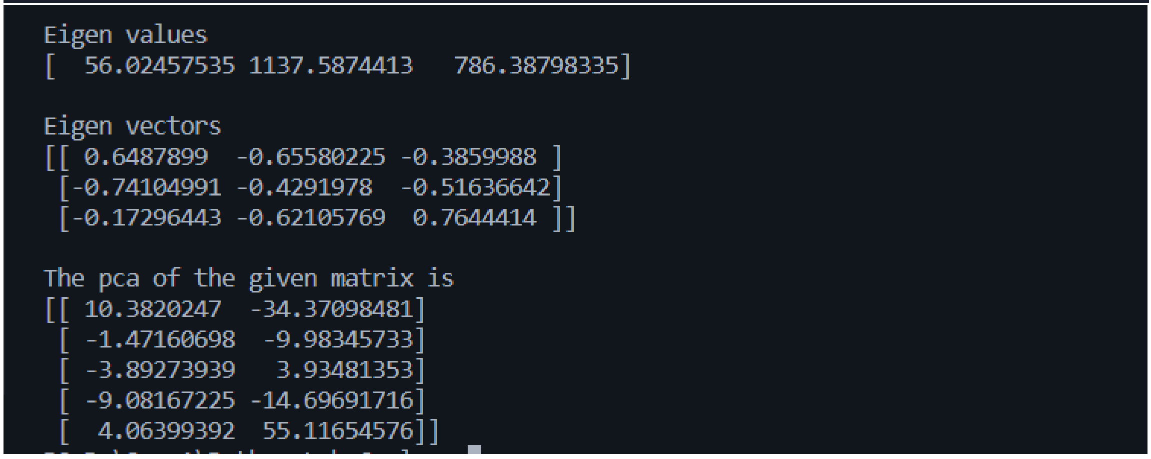
PROGRAM

```
import numpy as np


def getMatrix():
    rows = int(input("Enter no of rows : "))
    cols = int(input("Enter no of cols : "))
    matrix = np.empty((rows, cols))

    # creates an empty matrix with given size and inputs the coordinates
    for i in range(rows):
        print(f"Row {i+1}")
        for j in range(cols):
            value = float(input(f"Enter value at [{i+1}][{j+1}] : "))
            matrix[i, j] = value  # assign values to matrix
    return matrix


def find_pca(matrix):

    # finding mean
    mean = np.mean(matrix, axis=0)

    # subract col wise mean from the matrix
    centered_matrix = matrix - mean
    print("\nthe centered matrix is")
    print(centered_matrix)
    # computing covariance matrix of centered_matrix
    covariance_matrix = np.cov(centered_matrix.T)
    print("\nCovariance matrix")
    print(covariance_matrix)

    # computing eigenvectors and eigenvalues of covariance matrix
    eigenvalues, eigenvectors = np.linalg.eig(covariance_matrix)
    print("\nEigen values")
    print(eigenvalues)

    print("\nEigen vectors")
    print(eigenvectors)

    # compute the projection matrix by finding suitable eigen vectors
    projection_matrix = (eigenvectors.T[:][:2]).T
    P = projection_matrix.T.dot(centered_matrix.T)
    print("\nThe pca of the given matrix is ")
    print(P.T)


mat = getMatrix()
print("\nThe entered matrix is ")
print(mat)
find_pca(mat)
```

SAMPLE INPUT-OUTPUT

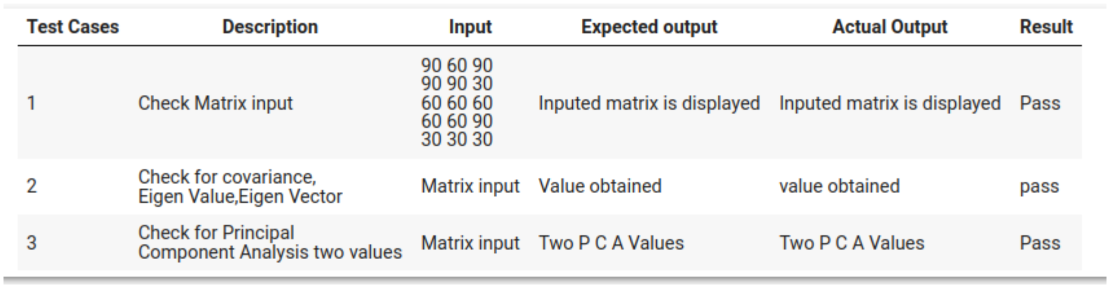


```
Enter no of rows : 5
Enter no of cols : 3
Row 1
Enter value at [1][1] : 90
Enter value at [1][2] : 60
Enter value at [1][3] : 90
Row 2
Enter value at [2][1] : 90
Enter value at [2][2] : 90
Enter value at [2][3] : 30
Row 3
Enter value at [3][1] : 60
Enter value at [3][2] : 60
Enter value at [3][3] : 60
Row 4
Enter value at [4][1] : 60
Enter value at [4][2] : 60
Enter value at [4][3] : 90
Row 5
Enter value at [5][1] : 30
Enter value at [5][2] : 30
Enter value at [5][3] : 30

The entered matrix is
[[90. 60. 90.]
 [90. 90. 30.]
 [60. 60. 60.]
 [60. 60. 90.]
 [30. 30. 30.]]

the centered matrix is
[[ 24.   0.  30.]
 [ 24.  30. -30.]
 [ -6.   0.   0.]
 [ -6.   0.  30.]
 [-36. -30. -30.]]

Covariance matrix
[[630. 450. 225.]
 [450. 450.   0.]
 [225.   0. 900.]]
```



```
Eigen values
[  56.02457535 1137.5874413   786.38798335]

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

The pca of the given matrix is
[[ 10.3820247  -34.37098481]
 [ -1.47160698  -9.98345733]
 [ -3.89273939   3.93481353]
 [ -9.08167225 -14.69691716]
 [  4.06399392  55.11654576]]
```

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

$https$:
$//github.com/nazalnihad/Python_{L}ab_{C}ycles/blob/main/Lab_{c}ycle_{3}/QN2/pca_{o}f_{m}atrix.py$

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 functon from sklearn)
- Show the distribution of each attribute as histogram.

THEORY

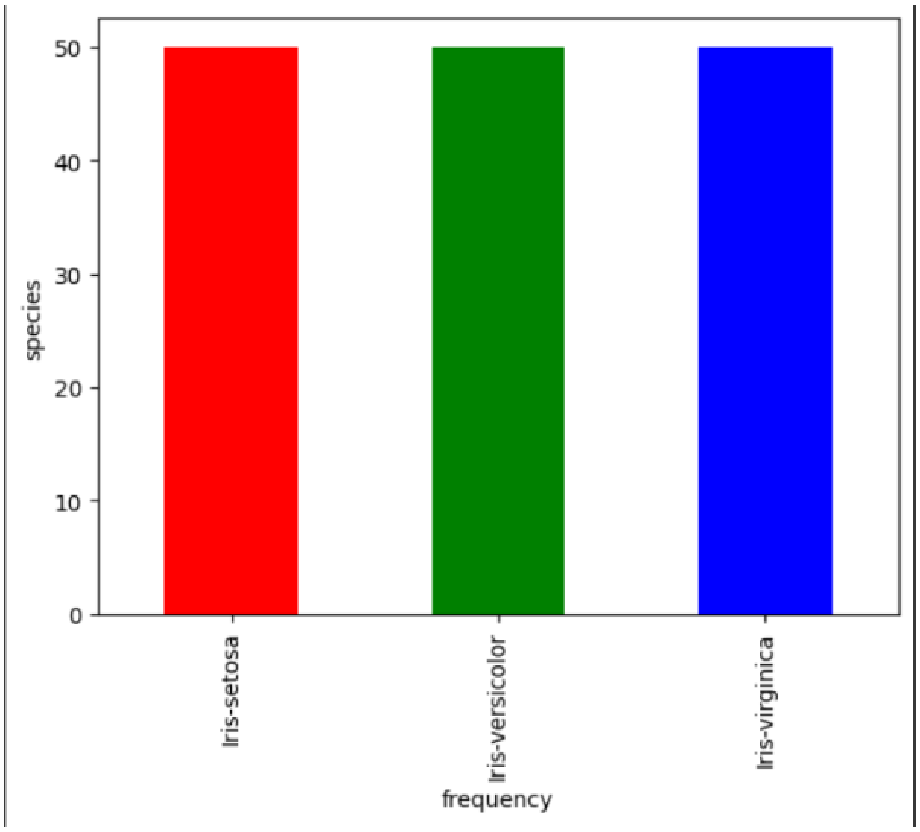
- Visualization - Matplotlib and Seaborn are python libraries that are used for data visualization. They have inbuilt modules for plotting different graphs.
- Data processing - Python can handle various encoding processes, and different types of modules need to be imported to make these encoding techniques 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.

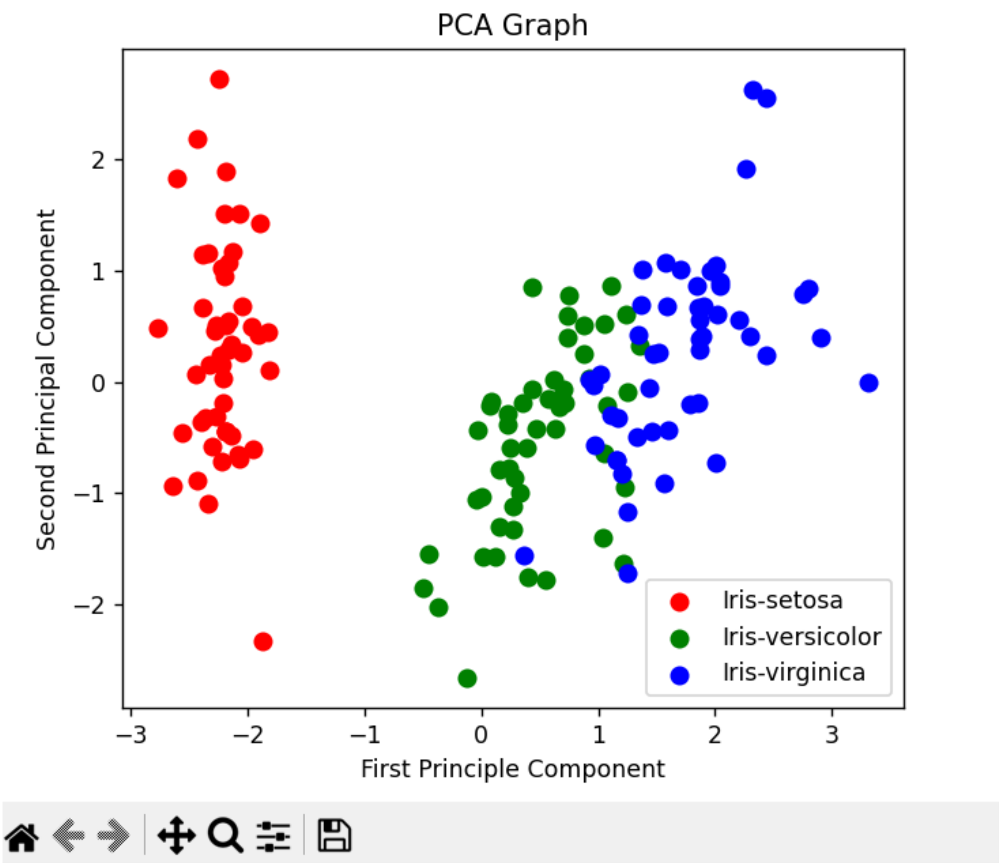
PROGRAM

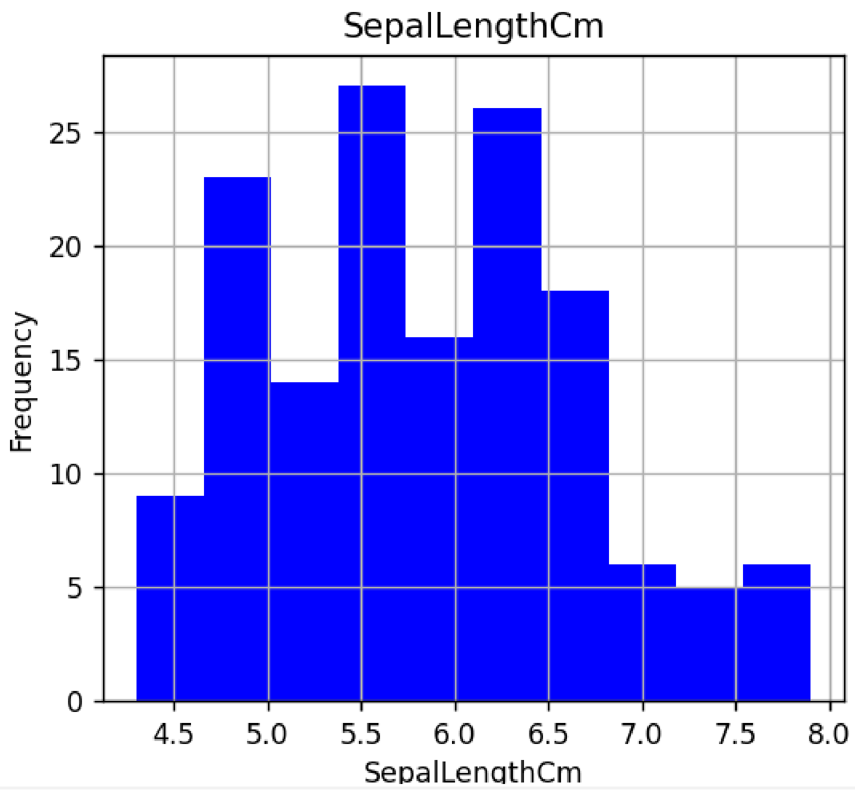
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import random

file_path = input("Enter the file path : ")
df = pd.read_csv(file_path)
```

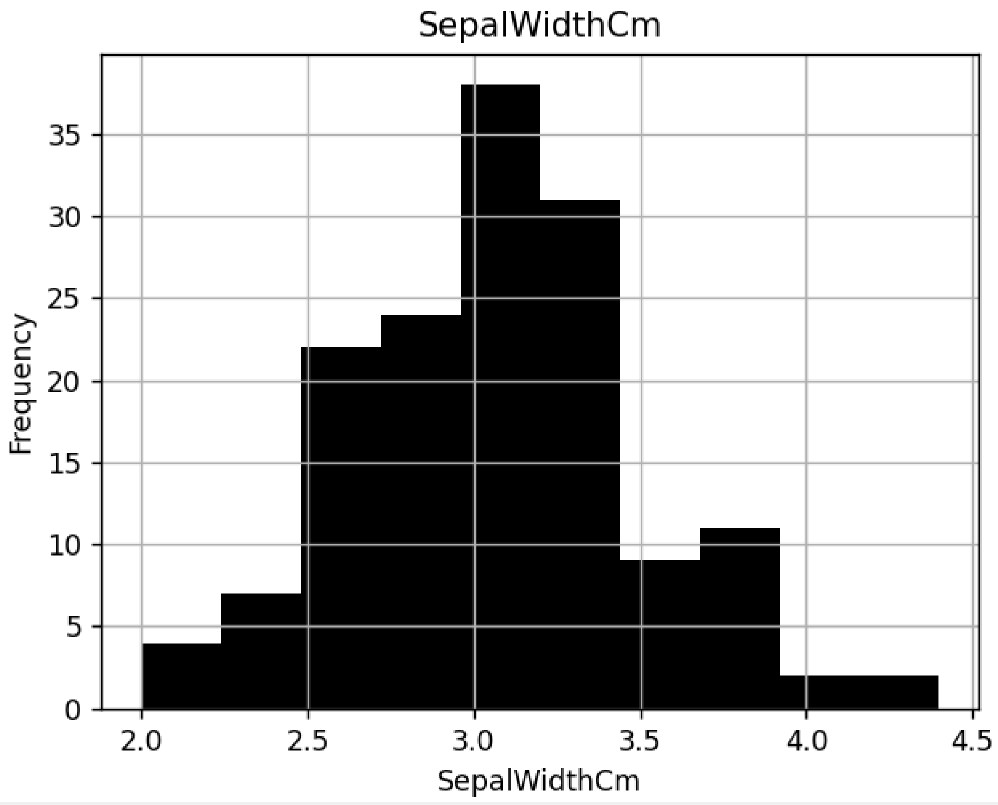
```
# bar plot


def bar_plot():
    categories = df['Species'].value_counts()
    categories.plot(kind="bar", xlabel="frequency",
                    ylabel="species", color=['r', 'g', 'b'])


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))
    figx = fig.add_subplot(1, 1, 1)
    figx.set_xlabel('First Principle Component')
    figx.set_ylabel('Second Principal Component')
    figx.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
        figx.scatter(finalDf.loc[indicesToKeep, 'principal component 1'],
                     finalDf.loc[indicesToKeep, 'principal component 2'], c=color, s=50)
    figx.legend(targets)
    plt.show()


# array to store items to print as histogram
hist_array = []
for i in df.columns:
    if i == "Id" or i == "Species":
        continue
    else:
        hist_array.append(i)
print(hist_array)
color_list = ['r', 'g', 'b', 'black']


def histPlot(category):
    # iterate through color and removes once chosen
    clr = random.choice(color_list)
    color_list.remove(clr)

    # graph details
    df[category].hist(color=clr)
    plt.xlabel(f"{category}")
    plt.ylabel("Frequency")
    plt.title(category)
    plt.show()


bar_plot()
pcaAppliedGraph()

for j in hist_array:
    histPlot(j)
```

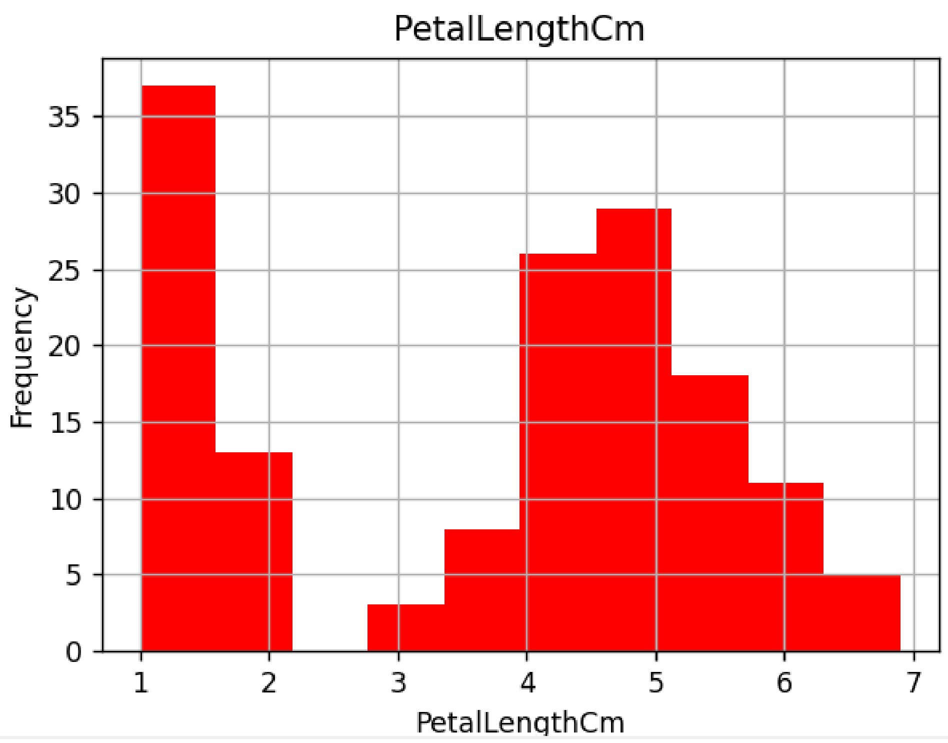
SAMPLE INPUT-OUTPUT

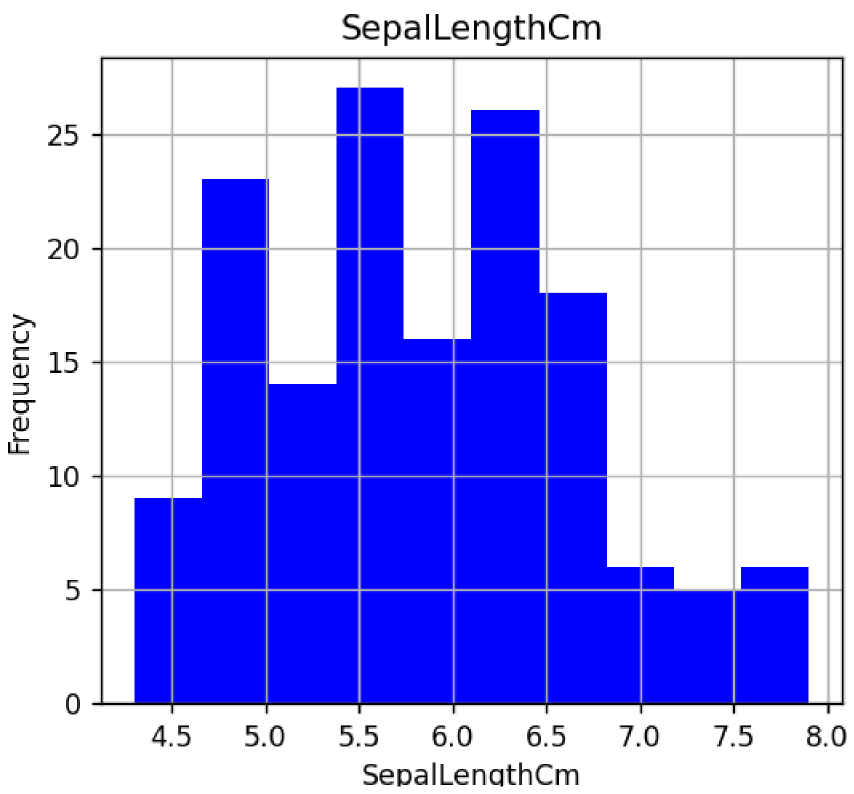




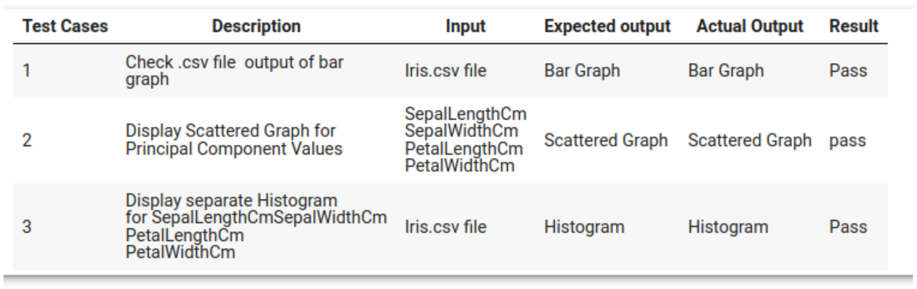

SepalLengthCm
Frequency
SepalLengthCm
0
5
10
15
20
25
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0


SepalWidthCm
Frequency
SepalWidthCm
0
5
10
15
20
25
30
35
2.0
2.5
3.0
3.5
4.0
4.5


PetalLengthCm
Frequency
35
30
25
20
15
10
5
0
1
2
3
4
5
6
7
PetalLengthCm


SepalLengthCm
Frequency
25
20
15
10
5
0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
SepalLengthCm

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	SepalLengthCm SepalWidthCm PetalLengthCm PetalWidthCm	Scattered Graph	Scattered Graph	pass
3	Display separate Histogram for SepalLengthCmSepalWidthCm PetalLengthCm PetalWidthCm	Iris.csv file	Histogram	Histogram	Pass

RESULT

Program executed Successfully and the output is obtained.

GIT LINK

$https$:
$//github.com/nazalnihad/Python_Lab_Cycles/blob/main/Lab_cycle_3/QN3/iris_plotting.py$

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

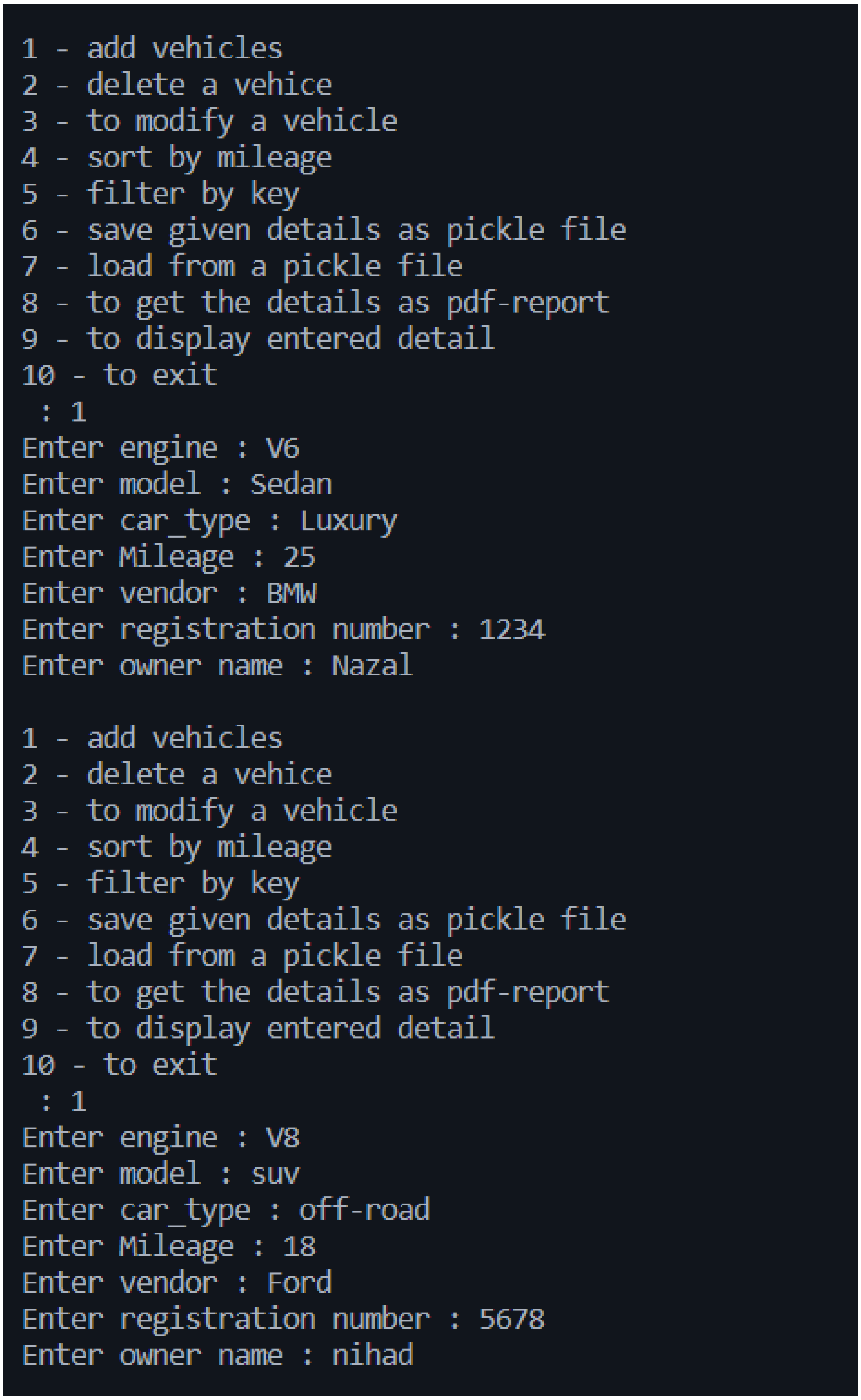
- OOPs - Object-oriented Programming (OOPs) is a programming paradigm that uses objects and classes in programming. It aims to implement real-world entities like inheritance, polymorphisms, encapsulation, etc. in the programming.
- Pickle - Python pickle module is used for serializing and de-serializing 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

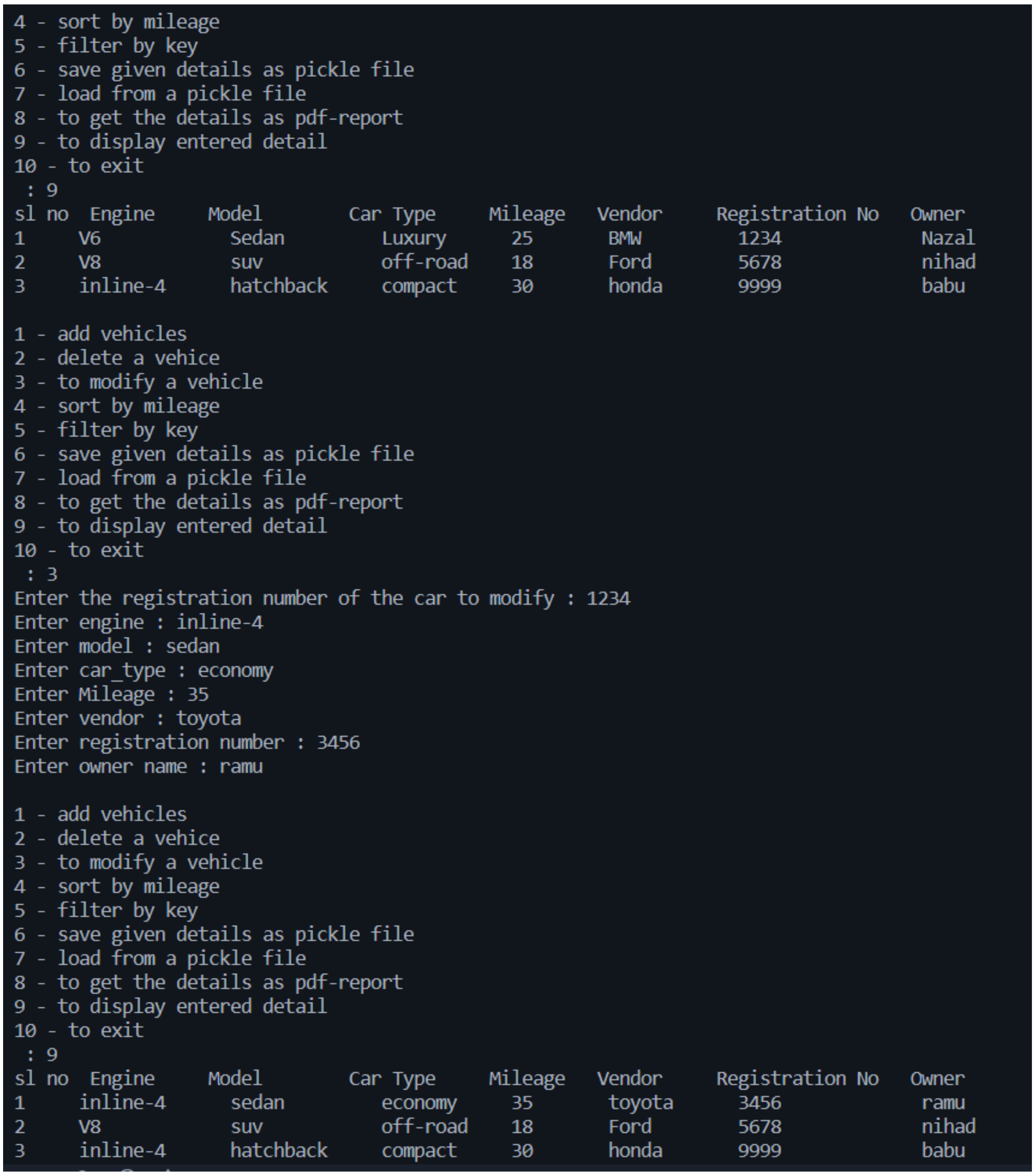
```
import pickle
from fpdf import FPDF


class car_details:
    def __init__(self, engine=0, model=0, car_type=0, mileage=0, vendor=0,
    reg_no=0, owner=0):
        self.engine = engine
        self.model = model
        self.car_type = car_type
        self.mileage = mileage
        self.vendor = vendor
        self.reg_no = reg_no
        self.owner = owner

    def display(self):
        print(f"{self.engine}\t{self.model}\t{self.car_type}\t{self.mileage}\t{self.
        vendor}\t{self.reg_no}\t{self.owner}")


class vehicles(car_details):
    def __init__(self):
        self.vehicle_list = list()

    def add(self, vehicle):
        self.vehicle_list.append(vehicle)

    #dupe display function to display sort,filtered.. details
    def dis_dupe(self, d_list):
        sl_no = 1

        print("sl no  Engine     Model        Car Type     Mileage   Vendor
        Registration No   Owner")

        for i in d_list:
            print(
                f"{sl_no:<6}{i.engine:<14}{i.model:<14}{i.car_type:<12}
                {i.mileage:<9}{i.vendor:<12}{i.reg_no:<17}{i.owner}")
            sl_no += 1

    def delete(self, car_reg):
        for i in self.vehicle_list:
            if str(i.reg_no) == str(car_reg):
                self.vehicle_list.remove(i)
                return self.vehicle_list

        print("Registration number not found ")
        return self.vehicle_list

    def modify(self, pos):
        for i in self.vehicle_list:
            if str(i.reg_no) == str(pos):
                i.engine = input("Enter engine : ")
                i.model = input("Enter model : ")
                i.car_type = input("Enter car_type : ")
                i.mileage = int(input("Enter Mileage : "))
                i.vendor = input("Enter vendor : ")
                i.reg_no = int(input("Enter registration number : "))
                i.owner = input("Enter owner name : ")
                return self.vehicle_list
        print("code not found")

    def sort_mileage(self):
        mileage_sorted = sorted(self.vehicle_list, key=lambda x: x.mileage)
        self.dis_dupe(mileage_sorted)

    def filtered(self):
        print("Enter the key to filter with : ")
        print("1 - Engine \n2 - model \n3 - car type \n4 - mileage \n5 -
        vendor \n6 - registration number \n7-owner")
        key = int(input(" : "))

        if key == 1:
            entry = input("Enter engine no : ")
            engine_sorted = [i for i in self.vehicle_list if str(
                i.engine) == str(entry)]
            self.dis_dupe(engine_sorted)

        elif key == 2:
            entry = input("Enter model no : ")
            model_sorted = [i for i in self.vehicle_list if str(
                i.model) == str(entry)]
            self.dis_dupe(model_sorted)

        elif key == 3:
            entry = input("Enter car type : ")
            car_type_sorted = [i for i in self.vehicle_list if str(
                i.car_type) == str(entry)]
            self.dis_dupe(car_type_sorted)

        elif key == 4:
            entry = input("Enter mileage : ")
            mileage_sorted = [i for i in self.vehicle_list if str(
                i.mileage) == str(entry)]
            self.dis_dupe(mileage_sorted)

        elif key == 5:
            entry = input("Enter vendor : ")
            ventor_sorted = [i for i in self.vehicle_list if str(
                i.vendor) == str(entry)]
            self.dis_dupe(ventor_sorted)

        elif key == 6:
            entry = input("Enter registration number : ")
            reg_sorted = [i for i in self.vehicle_list if str(
                i.reg_no) == str(entry)]
            self.dis_dupe(reg_sorted)

        elif key == 7:
            entry = input("Enter owner : ")
            owner_sorted = [i for i in self.vehicle_list if str(
                i.owner) == str(entry)]
            self.dis_dupe(owner_sorted)

        else:
            print("Enter a valid key ")

    def create_pickle(self):
        pickle.dump(self.vehicle_list, open(
            "vehicles_collection.pickle", "wb"))

    def load_pickle(self):
        pickle_path = input("Enter the pickle file's path")
        pickle_loaded_list = pickle.load(open(pickle_path, "rb"))
        self.dis_dupe(pickle_loaded_list)

    def report_as_pdf(self, filename):
        pdf = FPDF()
        pdf.set_title("Car Report")
        pdf.set_font("Arial", size=12)
        pdf.add_page()

        # title
        pdf.cell(0, 10, txt="Car Report", ln=True, align="C")

        # heading added
        pdf.set_font("Arial", "B", size=10)
        pdf.cell(30, 10, txt="Engine", border=1)
        pdf.cell(30, 10, txt="Model", border=1)
        pdf.cell(30, 10, txt="Car Type", border=1)
        pdf.cell(30, 10, txt="Mileage", border=1)
        pdf.cell(30, 10, txt="Vendor", border=1)
        pdf.cell(30, 10, txt="Registration No", border=1)
        pdf.cell(30, 10, txt="Owner", border=1)
        pdf.ln()

        # adding car details
        pdf.set_font("Arial", size=10)
        for vehicle in self.vehicle_list:
            pdf.cell(30, 10, txt=str(vehicle.engine), border=1)
            pdf.cell(30, 10, txt=str(vehicle.model), border=1)
            pdf.cell(30, 10, txt=str(vehicle.car_type), border=1)
            pdf.cell(30, 10, txt=str(vehicle.mileage), border=1)
            pdf.cell(30, 10, txt=str(vehicle.vendor), border=1)
            pdf.cell(30, 10, txt=str(vehicle.reg_no), border=1)
            pdf.cell(30, 10, txt=str(vehicle.owner), border=1)
            pdf.ln()

        # saved as given file name
        pdf.output(filename)

    def disp(self):
        sl_no = 1
        print("sl no  Engine     Model        Car Type     Mileage   Vendor
        Registration No   Owner")
        for i in self.vehicle_list:
            print(
                f"{sl_no:<6}{i.engine:<14}{i.model:<14}{i.car_type:<12}
                {i.mileage:<9}{i.vendor:<12}{i.reg_no:<17}{i.owner}")
            sl_no += 1


def main():
    obj = vehicles()
```

```
start = True
while start:
    print("\n1 - add vehicles \n2 - delete a vehice \n3 - to modify a
    vehicle \n4 - sort by mileage ")
    print("5 - filter by key \n6 - save given details as pickle file
    \n7 - load from a pickle file ")
    print("8 - to get the details as pdf-report \n9 - to display
    entered detail \n10 - to exit")
    choice = int(input(" : "))

    if choice == 1:
        engine = input("Enter engine : ")
        model = input("Enter model : ")
        car_type = input("Enter car_type : ")
        mileage = input("Enter Mileage : ")
        vendor = input("Enter vendor : ")
        reg_no = input("Enter registration number : ")
        owner = input("Enter owner name : ")
        collected = car_details(
            engine, model, car_type, mileage, vendor, reg_no, owner)
        obj.add(collected)

    elif choice == 2:
        del_position = int(
            input("Enter the registration number of the car to delete : "))
        obj.delete(del_position)

    elif choice == 3:
        mod_position = int(
            input("Enter the registration number of the car to modify : "))
        obj.modify(mod_position)

    elif choice == 4:
        obj.sort_mileage()

    elif choice == 5:
        obj.filtered()

    elif choice == 6:
        obj.create_pickle()
        elif choice == 7:
            obj.load_pickle()

        elif choice == 8:
            pdf_name = input("enter the name you want to save as : ")
            pdf_name += ".pdf"
            obj.report_as_pdf(pdf_name)

        elif choice == 9:
            obj.disp()

        elif choice == 10:
            start = False
            return

        else:
            print("enter a valid number")


main()
```

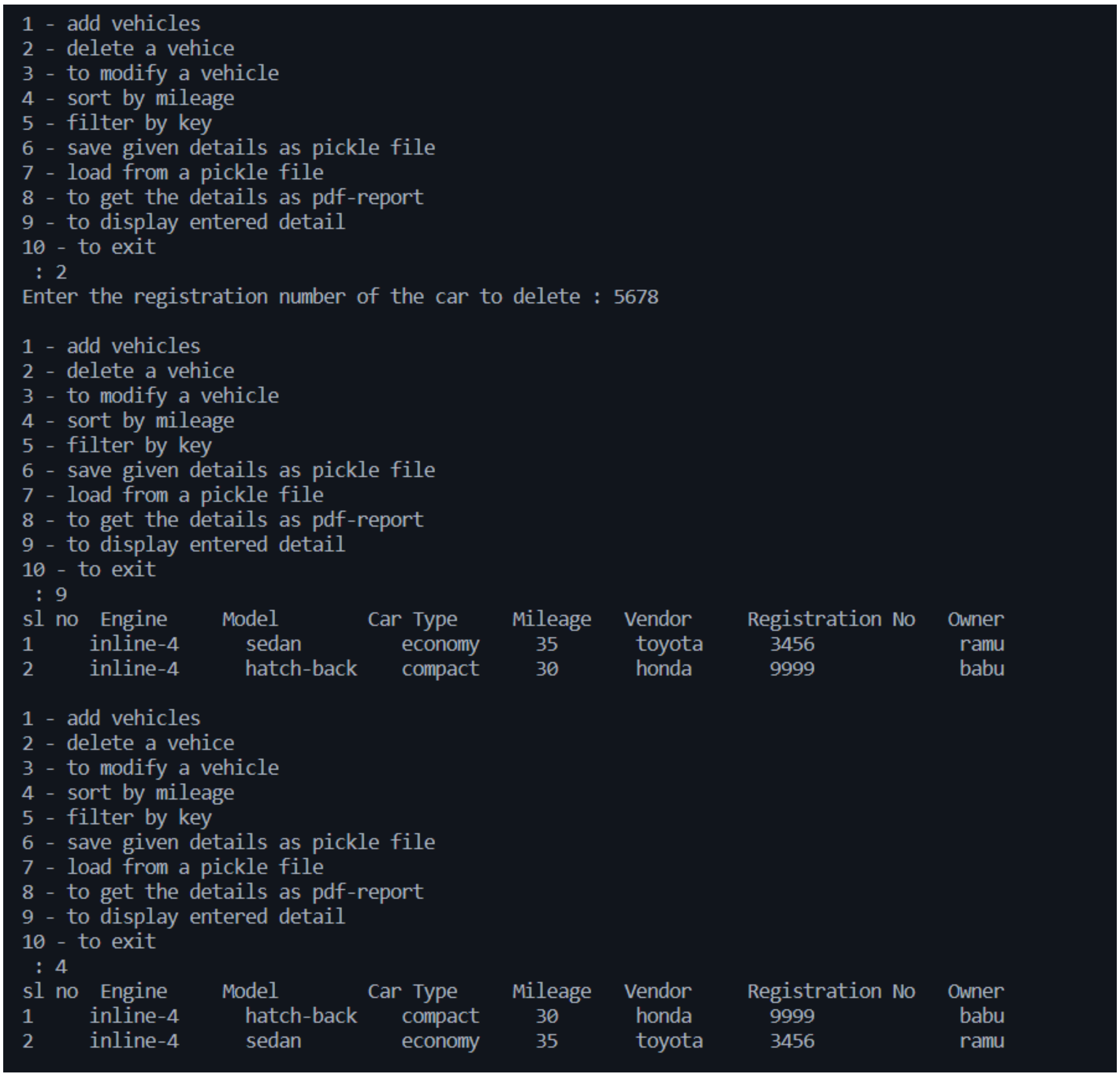
SAMPLE INPUT-OUTPUT



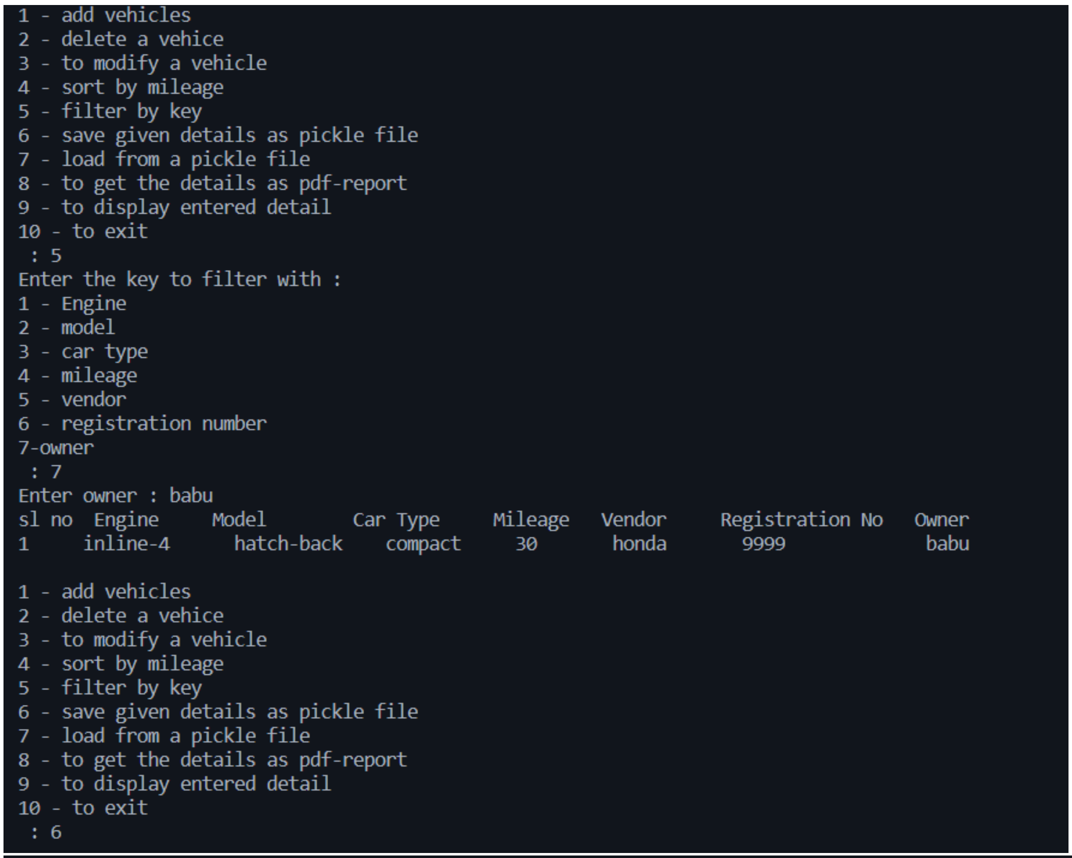
```
1 - add vehicles
2 - delete a vehice
3 - to modify a vehicle
4 - sort by mileage
5 - filter by key
6 - save given details as pickle file
7 - load from a pickle file
8 - to get the details as pdf-report
9 - to display entered detail
10 - to exit
 : 1
Enter engine : V6
Enter model : Sedan
Enter car_type : Luxury
Enter Mileage : 25
Enter vendor : BMW
Enter registration number : 1234
Enter owner name : Nazal

1 - add vehicles
2 - delete a vehice
3 - to modify a vehicle
4 - sort by mileage
5 - filter by key
6 - save given details as pickle file
7 - load from a pickle file
8 - to get the details as pdf-report
9 - to display entered detail
10 - to exit
 : 1
Enter engine : V8
Enter model : suv
Enter car_type : off-road
Enter Mileage : 18
Enter vendor : Ford
Enter registration number : 5678
Enter owner name : nihad
```



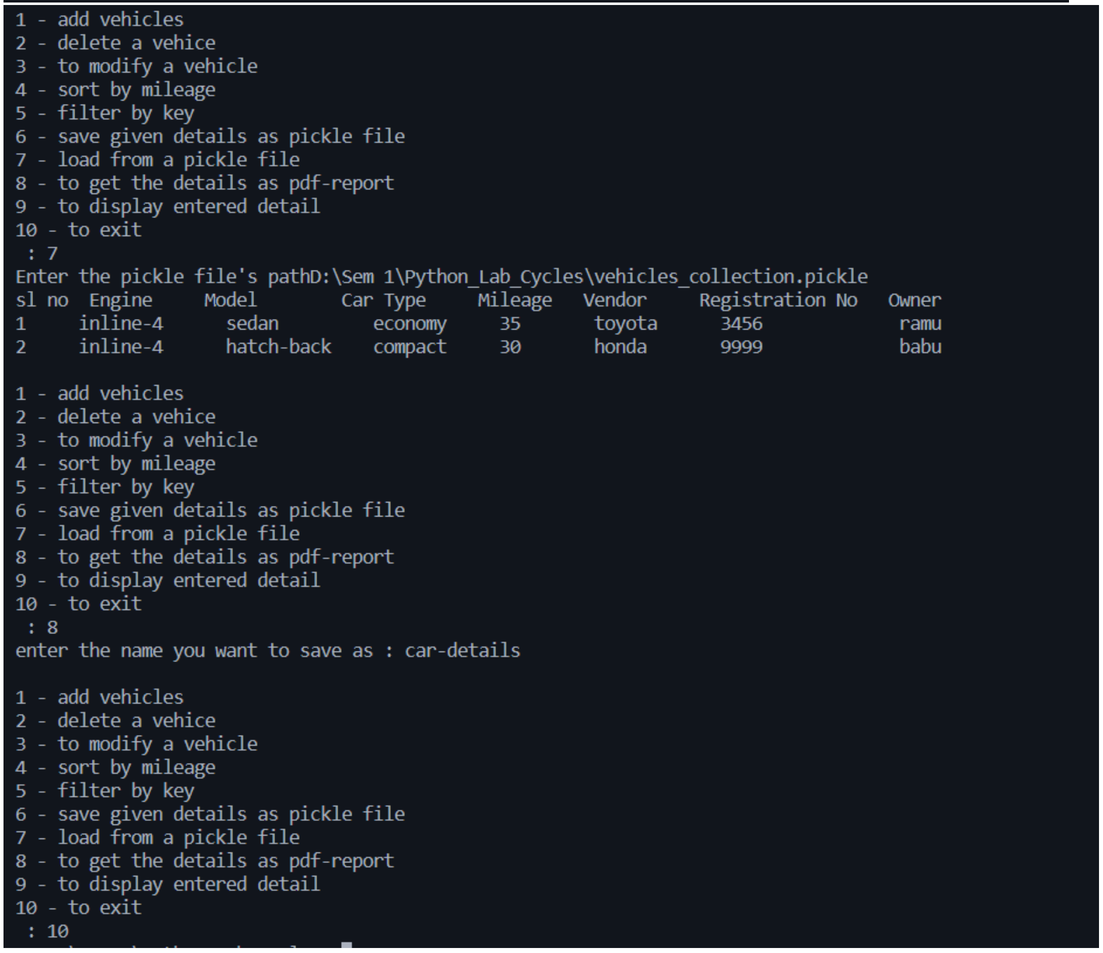
```
4 - sort by mileage
5 - filter by key
6 - save given details as pickle file
7 - load from a pickle file
8 - to get the details as pdf-report
9 - to display entered detail
10 - to exit
 : 9
sl no  Engine     Model       Car Type     Mileage   Vendor     Registration No   Owner
1     V6           Sedan        Luxury      25       BMW         1234              Nazal
2     V8           suv          off-road    18       Ford        5678              nihad
3     inline-4     hatchback    compact     30       honda       9999              babu

1 - add vehicles
2 - delete a vehice
3 - to modify a vehicle
4 - sort by mileage
5 - filter by key
6 - save given details as pickle file
7 - load from a pickle file
8 - to get the details as pdf-report
9 - to display entered detail
10 - to exit
 : 3
Enter the registration number of the car to modify : 1234
Enter engine : inline-4
Enter model : sedan
Enter car_type : economy
Enter Mileage : 35
Enter vendor : toyota
Enter registration number : 3456
Enter owner name : ramu

1 - add vehicles
2 - delete a vehice
3 - to modify a vehicle
4 - sort by mileage
5 - filter by key
6 - save given details as pickle file
7 - load from a pickle file
8 - to get the details as pdf-report
9 - to display entered detail
10 - to exit
 : 9
sl no  Engine     Model       Car Type     Mileage   Vendor     Registration No   Owner
1     inline-4     sedan        economy     35       toyota      3456              ramu
2     V8           suv          off-road    18       Ford        5678              nihad
3     inline-4     hatchback    compact     30       honda       9999              babu
```



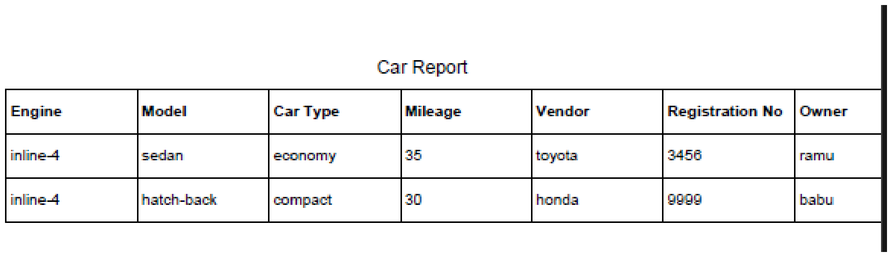
```
1 - add vehicles
2 - delete a vehice
3 - to modify a vehicle
4 - sort by mileage
5 - filter by key
6 - save given details as pickle file
7 - load from a pickle file
8 - to get the details as pdf-report
9 - to display entered detail
10 - to exit
 : 2
Enter the registration number of the car to delete : 5678

1 - add vehicles
2 - delete a vehice
3 - to modify a vehicle
4 - sort by mileage
5 - filter by key
6 - save given details as pickle file
7 - load from a pickle file
8 - to get the details as pdf-report
9 - to display entered detail
10 - to exit
 : 9
sl no  Engine     Model        Car Type     Mileage   Vendor     Registration No   Owner
1     inline-4     sedan         economy     35        toyota      3456               ramu
2     inline-4     hatch-back    compact     30        honda       9999               babu

1 - add vehicles
2 - delete a vehice
3 - to modify a vehicle
4 - sort by mileage
5 - filter by key
6 - save given details as pickle file
7 - load from a pickle file
8 - to get the details as pdf-report
9 - to display entered detail
10 - to exit
 : 4
sl no  Engine     Model        Car Type     Mileage   Vendor     Registration No   Owner
1     inline-4     hatch-back    compact     30        honda       9999               babu
2     inline-4     sedan         economy     35        toyota      3456               ramu
```



```
1 - add vehicles
2 - delete a vehice
3 - to modify a vehicle
4 - sort by mileage
5 - filter by key
6 - save given details as pickle file
7 - load from a pickle file
8 - to get the details as pdf-report
9 - to display entered detail
10 - to exit
 : 5
Enter the key to filter with :
1 - Engine
2 - model
3 - car type
4 - mileage
5 - vendor
6 - registration number
7-owner
 : 7
Enter owner : babu
sl no  Engine     Model        Car Type     Mileage   Vendor     Registration No   Owner
1     inline-4      hatch-back    compact     30       honda       9999              babu

1 - add vehicles
2 - delete a vehice
3 - to modify a vehicle
4 - sort by mileage
5 - filter by key
6 - save given details as pickle file
7 - load from a pickle file
8 - to get the details as pdf-report
9 - to display entered detail
10 - to exit
 : 6
```



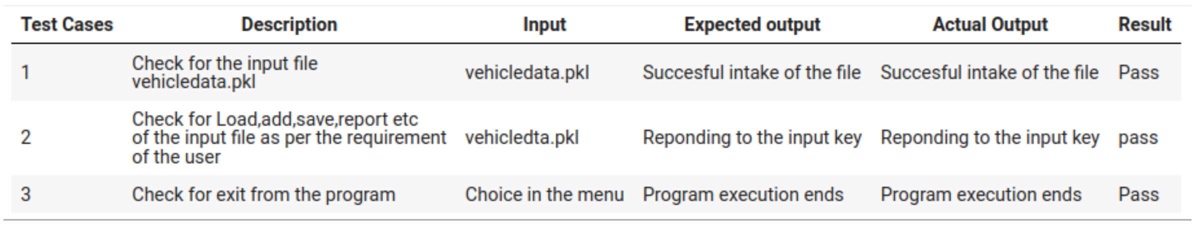
```
1 - add vehicles
2 - delete a vehice
3 - to modify a vehicle
4 - sort by mileage
5 - filter by key
6 - save given details as pickle file
7 - load from a pickle file
8 - to get the details as pdf-report
9 - to display entered detail
10 - to exit
 : 7
Enter the pickle file's pathD:\Sem 1\Python_Lab_Cycles\vehicles_collection.pickle
sl no  Engine     Model        Car Type     Mileage   Vendor     Registration No   Owner
1     inline-4      sedan         economy      35       toyota      3456              ramu
2     inline-4      hatch-back    compact      30       honda       9999              babu

1 - add vehicles
2 - delete a vehice
3 - to modify a vehicle
4 - sort by mileage
5 - filter by key
6 - save given details as pickle file
7 - load from a pickle file
8 - to get the details as pdf-report
9 - to display entered detail
10 - to exit
 : 8
enter the name you want to save as : car-details

1 - add vehicles
2 - delete a vehice
3 - to modify a vehicle
4 - sort by mileage
5 - filter by key
6 - save given details as pickle file
7 - load from a pickle file
8 - to get the details as pdf-report
9 - to display entered detail
10 - to exit
 : 10
```



Car Report

Engine	Model	Car Type	Mileage	Vendor	Registration No	Owner
inline-4	sedan	economy	35	toyota	3456	ramu
inline-4	hatch-back	compact	30	honda	9999	babu

TEST CASES



Test Cases	Description	Input	Expected output	Actual Output	Result
1	Check for the input file vehicledata.pkl	vehicledata.pkl	Succesful intake of the file	Succesful 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

$https$:
$//github.com/nazalnihad/Python_{L}ab_{C}ycles/blob/main/Lab_{c}ycle_{3}/QN4/car_{d}etails_{c}ollect.py$

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 th 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

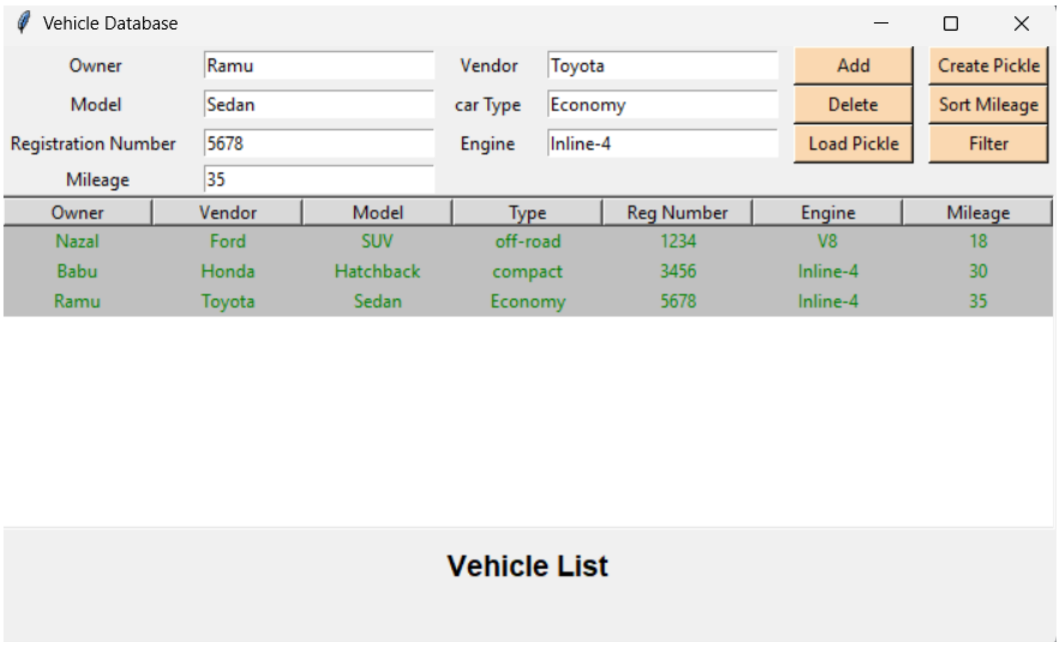
```
from tkinter import *
from tkinter.filedialog import askopenfilename, asksaveasfile
from tkinter.messagebox import showinfo
from tkinter.ttk import Style, Treeview
import pickle

global listOfVehicles
global sortedList
listOfVehicles = list()
vehicle_attributes = ["ownerName", "vendor", "model",
                      "type", "registrationNumber", "engine",
                      "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['engine'] = 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['engine'], 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['engine'], 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['engine'], 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['engine'], 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['engine'], i['mileage']))


def delete():
    selection = treeList.selection()[0]
    treeList.delete(selection)


def loadFile():
    # clear the treeview
    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['engine'], 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['engine'], 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
window = Tk()
window.geometry("705x400")
window.title("Vehicle Database")
```

```
#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 ")
label1.grid(row=0, column=0)

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

label2 = Label(window, text="Vendor ")
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 ")
label3.grid(row=1, column=0)

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

label4 = Label(window, text="car 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  ")
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='#fad8b1', command=loadFile)
button1.grid(row=2, column=4)

button8 = Button(window, width=10, text="Filter",
                 bg='#fad8b1', command=filterList)
button8.grid(row=2, column=5)

# second section
# row - 0
button2 = Button(window, width=10, text="Add", bg='#fad8b1',
command=addList)
button2.grid(row=0, column=4)

# row - 1
button4 = Button(window, width=10, text="Delete", bg='#fad8b1',
command=delete)
button4.grid(row=1, column=4)

button5 = Button(window, width=10, text="Sort Mileage",
                 bg='#fad8b1', command=sortMileage)
button5.grid(row=1, column=5)

button7 = Button(window, width=10, text="Create Pickle",
                 bg='#fad8b1', 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', 'grey')])
treeList = Treeview(columns=("Owner", "Vendor", "Model",
                    "Type", "Reg Number", "Engine", "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("Engine", text="Engine")
treeList.heading("Mileage", text="Mileage")

treeList['show'] = 'headings'

treeList.column("Owner", width=100, anchor="center")
treeList.column("Vendor", width=100, anchor="center")
treeList.column("Model", width=100, anchor="center")
treeList.column("Type", width=100, anchor="center")
treeList.column("Reg Number", width=100, anchor="center")
treeList.column("Engine", width=100, anchor="center")
treeList.column("Mileage", width=100, anchor="center")

list_label = Label(window, text="Vehicle List", font=("Arial",
14, "bold"))
list_label.grid(row=5, column=0, columnspan=6, pady=10)



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

SAMPLE INPUT-OUTPUT



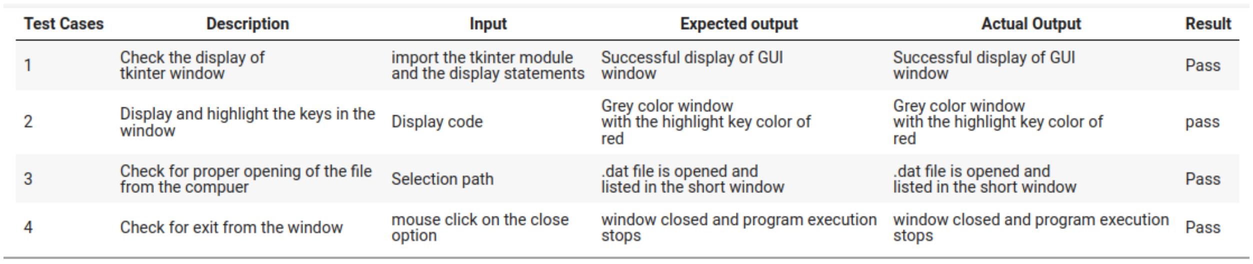
Vehicle Database

Owner	Ramu	Vendor	Toyota
Model	Sedan	car Type	Economy
Registration Number	5678	Engine	Inline-4
Mileage	35		

Owner	Vendor	Model	Type	Reg Number	Engine	Mileage
Nazal	Ford	SUV	off-road	1234	V8	18
Babu	Honda	Hatchback	compact	3456	Inline-4	30
Ramu	Toyota	Sedan	Economy	5678	Inline-4	35

Vehicle List

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 compuer	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

$https : //github.com/nazalnihad/Python_Lab_Cycles/blob/main/Lab_cycle_3/QN5/qn5_ui.py$