

**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*

**PRUTHVIKANTH AC  
(80522017)**

**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 **PRUTHVIKANTH AC (80522017)**, in partial fulfillment of the requirements for the award of degree in **Master of Science (Five Year Integrated) in Computer Science (Artificial Intelligence & Data Science)** of Cochin University of Science and Technology (CUSAT), Kochi. The lab record has been approved as it satisfies the academic requirements in respect of the first semester laboratory prescribed for the Master of Science (Five Year Integrated) in Computer Science degree.*

**Faculty Member in-charge**

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

Dr. Madhu S. Nair  
Professor and Head  
Department of Computer Science  
CUSAT

Sl. No.	Program	Page No.
1.	Arithmetic Operations on Four Digit Numbers	1
2.	Triangle Area and Contribution	3
3.	Employee Payslip Generation	5
4.	Check for Happy Number	8
5.	String Operations	12
6.	Pairs of Rabbits in 'N' Months	18
7.	Operations on List of Integers	20
8.	Iris'.json' File Handling	23
9.	Box Class for Shapes	28
10.	3D_Shapes Inheritance	32
11.	Tic Tac Toe	35
12.	Principal Component Analysis	41
13.	Data Visualisation	44
14.	Vehicle Details	49
15.	Tkinter UI Application	55

## ARITHMETIC OPERATION ON FOUR DIGIT NUMBER

### AIM

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

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

### THEORY

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

### PROGRAM

```
def sum_digits(num): # function to find the sum of the digits
    sum = 0
    while num != 0:
        sum += num % 10 # extracting the digits and adding them to sum
        num = num//10 # floor division to remove the extracted digit
    return sum

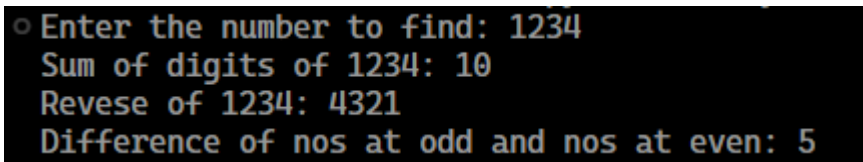
def reverse(num): # function to find the reverse of the number
    rev = ""
    while num != 0:
        rev += str(num % 10)
        num = num // 10
    return rev

def sum_places(num): # function to find the diff b/w the odd num and even num
    prododd = 1
    prodeven = 1
    pos = 1 # variable for keeping track of the position
    while num != 0:
        if pos % 2 == 0: # position is even,multitplied to even
```

```
        prodeven *= num % 10
    else:
        prododd *= num % 10 # position is odd,multitplied to odd
    num = num//10
    pos += 1 # updating the position
    return prodeven-prododd # returning the osubtracted value

num = int(input("Enter the number to find: "))
print(f"Sum of digits of {num}: {sum_digits(num)}")
print(f"Revese of {num}: {(reverse(num))}") # function call
print(f"Difference of nos at odd and nos at even: {sum_places(num)}")
```

### SAMPLE INPUT-OUTPUT



```
Enter the number to find: 1234
Sum of digits of 1234: 10
Revese of 1234: 4321
Difference of nos at odd and nos at even: 5
```

### 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

[Click Here for the Code](#)

## TRIANGLE AREA AND CONTRIBUTION

### AIM

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

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

### THEORY

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

### PROGRAM

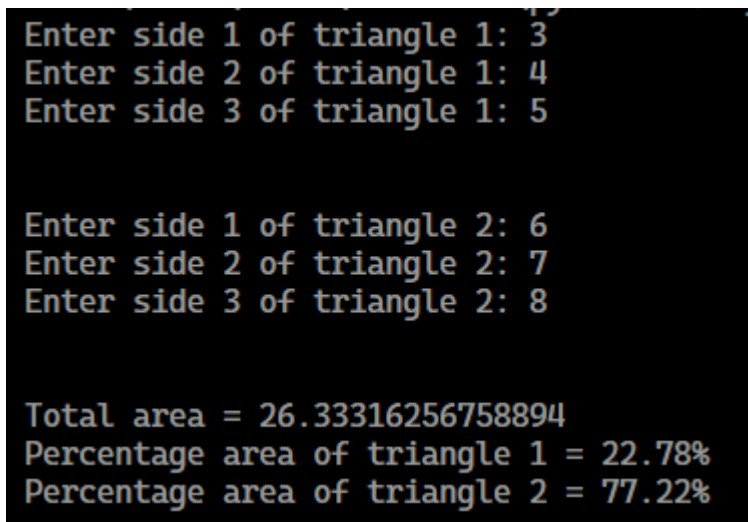
```
import math #importing math module for sqrt

def read(no): #function for reading the sides
    side1 = int(input(f"Enter side 1 of triangle {no}: "))
    side2 = int(input(f"Enter side 2 of triangle {no}: "))
    side3 = int(input(f"Enter side 3 of triangle {no}: "))
    print("\n")
    return (area(side1, side2, side3)) #area function call and passing sides

def area(side1, side2, side3):
    sp = (side1+side2+side3)/2 #finding the semi-perimeter
    area = math.sqrt(sp*(sp-side1)*(sp-side2)*(sp-side3)) #calculating area
    return area
```

```
area1 = read(1) #function calling and accepting into respective variable
area2 = read(2)
total = area1+area2
print(f"Total area = {total}\n" #displaying area
      f"Percentage area of triangle 1 = {((area1/total)*100):.4}%\n"
      f"Percentage area of triangle 2 = {((area2/total)*100):.4}%\n")
```

### SAMPLE INPUT-OUTPUT



```
Enter side 1 of triangle 1: 3
Enter side 2 of triangle 1: 4
Enter side 3 of triangle 1: 5

Enter side 1 of triangle 2: 6
Enter side 2 of triangle 2: 7
Enter side 3 of triangle 2: 8

Total area = 26.33316256758894
Percentage area of triangle 1 = 22.78%
Percentage area of triangle 2 = 77.22%
```

### 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

[Click Here for the Code](#)

## EMPLOYEE PAYSIP GENERATION

### AIM

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

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

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

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

Net Salary =  $GS - D$

### THEORY

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

### PROGRAM

```
def employee_details(): #function for taking in employee details
    name = input("Enter the Employee Name: ")
    code = input("Enter the Employee Code: ")
    basic_pay = int(input("Enter the Basic Pay: "))
    list = [name, code, main(basic_pay)] #puts them inside a list and returns it
    return list

def main(basic_pay): #checking basic pay and assiging required values
    if basic_pay < 10000:
        DA = 0.05
        HRA = 0.025
        MA = 500
        PT = 20
        PF = 0.08
        IT = 0

    elif basic_pay > 10000 and basic_pay < 30000:
```



```
    DA = 0.075
    HRA = 0.05
    MA = 2500
    PT = 60
    PF = 0.08
    IT = 0

elif basic_pay > 30000 and basic_pay < 50000:
    DA = 0.11
    HRA = 0.075
    MA = 5000
    PT = 60
    PF = 0.11
    IT = 0.11

else:
    DA = 0.25
    HRA = 0.11
    MA = 7000
    PT = 80
    PF = 0.12
    IT = 0.20

return calc(basic_pay, DA, HRA, MA, PT, PF, IT) #returns the final amount


def calc(basic_pay, DA, HRA, MA, PT, PF, IT): #calculates GS AND DED
    GS = basic_pay+basic_pay*DA+basic_pay*HRA+MA
    DED = PT+basic_pay*PF+basic_pay*IT
    return net_Salary(GS, DED) #calculate the net amount


def net_Salary(GS, DED): #function for calculating net salary
    net = GS-DED
    return net


list = employee_details() #takes the users input
print("\nEmployee Name:", list[0], "\nEmployee Code: ", #prints the details
      list[1], "\nNet Salary: ", list[2], "\n")
```

## SAMPLE INPUT-OUTPUT

```

Enter the Employee Name: JAMES
Enter the Employee Code: #4567
Enter the Basic Pay: 430000

Employee Name: JAMES
Employee Code: #4567
Net Salary: 454120.0

```

## TEST CASES

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

## RESULT

Program executed Successfully and the output is obtained.

## GIT LINK

[Click Here for the Code](#)

## HAPPY NUMBER

### AIM

Develop a program to perform the following task:

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

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

```
def check_happy(num): #function for checking is happy
    count = 0 #for checking how many times it looped
    while num != 1 and count < 100: #if number = 1, then its happy
        temp, sum = num, 0
        while temp > 0: #runs loop until temp reaches 0
            sum += (temp % 10)**2 #adds the squared reminder to sum
            temp = temp//10
        num = sum #gives the final sum to num
        count += 1 #increases the count for loop
    if num == 1 and count < 100: #condition for happy number
        return True # Return true if it is a happy number
    else:
```

```
        return False # Return false if the number is a sad number

def range_happy(start, end): #function for finding happy number within a range
    while start < end: #loop runs till the start value is lower than end
        if check_happy(start) is True: #checks start value is happy and prints
            print(start, end=" ")
        start += 1 #goes to the second number

def n_happy(end): #function for finding n happy numbers
    count = 0 #variable for keeping track of how many numbers printed
    while count < end:
        if check_happy(count) is True: #checks happy and prints
            print(count, end=" ")
        count += 1 #goes to the second number

while True: #main menu runs forever until user exits
    print("\n\nMAIN MENU\n",
          "1.Check Happy\n",
          "2.Print happy within range\n",
          "3.Print N Happy Numbers\n",
          "4.Exit!")
    choice = int(input("Enter the Choice:"))

    if choice == 1:
        num = int(input("Enter the number to check if Happy: ")) #input
        if check_happy(num) is True:
            print("\nHappy Number!")
        else:
            print("\nSad Number!")
    elif choice == 2:
        start = int(input("Enter the Starting No: ")) #starting value
        end = int(input("Enter the Stopping No: ")) #ending value
        range_happy(start, end)
    elif choice == 3:
        end = int(input("Enter the Limit: ")) #variable for ending value
        n_happy(end)
    else:
        break #breaks out the while loop and exits the program
```

## SAMPLE INPUT-OUTPUT

```
MAIN MENU
1.Check Happy
2.Print happy within range
3.Print N Happy Numbers
4.Exit!
Enter the Choice:1
Enter the number to check if Happy: 33

Sad Number!
```

```
MAIN MENU
1.Check Happy
2.Print happy within range
3.Print N Happy Numbers
4.Exit!
Enter the Choice:1
Enter the number to check if Happy: 32

Happy Number!
```

```
MAIN MENU
1.Check Happy
2.Print happy within range
3.Print N Happy Numbers
4.Exit!
Enter the Choice:2
Enter the Starting No: 5
Enter the Stopping No: 50
7 10 13 19 23 28 31 32 44 49
```

```
MAIN MENU
1.Check Happy
2.Print happy within range
3.Print N Happy Numbers
4.Exit!
Enter the Choice:3
Enter the Limit: 70
1 7 10 13 19 23 28 31 32 44 49 68
```

## 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**

[Click Here for the Code](#)

## STRING OPERATIONS

### AIM

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

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

### THEORY

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

### PROGRAM

```
def subs(string, n): # function for finding all possible substrings
    i, j, list = 0, 0, []
    for i in range(0, n+1): # two nested loops to find all possible substrings
        for j in range(i+1, n+1):
            # slices the strings to extract the substring and appends to list
            list.append(string[i:j])
    return list

def subs_length(string, l): #function for finding substring with specific length
    i, j, list = 0, 0, []
    for i in range(0, len(string)+1): #two nested loops to find substrings
        for j in range(i+1, len(string)+1):
            # checks if the sliced substring has length and appends into a list
            if len(string[i:j]) == l:
                list.append(string[i:j])
    return list
```

```
#finding substring with a specific length and has x distinct characters
def subs_dist_length(string, l, n):
    i, j, list = 0, 0, []
    for i in range(0, len(string)+1):
        # uses two nested loops to find all the possible substrings
        for j in range(i+1, len(string)+1):
            sub = string[i:j]
            if len(sub) == l: # checks if substring has required length
                #compares the no of distinct chars in the substring
                if len(set(sub)) == n:
                    list.append(sub)
    return list

#finding substring with max length and has x distinct characters
def subs_dist_max_length(string, n):
    i, j, list = 0, 0, []
    for i in range(0, len(string)+1):
        #two nested loops to find the possible substrings
        for j in range(i+1, len(string)+1):
            temp = string[i:j]
            dist = set(temp)
            if len(dist) == n: #compare the no of distinct chars in the substring
                list.append(temp)

    #max inside the list and finds the longest substring and stores the len
    long = len(max(list, key=len))
    for i in list:
        if len(i) == long: # checks if the longest string and print
            print(i, end=" ")

def palindrome_subs(string): # function for finding palindrome substrings
    i, j, list = 0, 0, []
    for i in range(0, len(string)+1):
        # uses two nested loops to find all the possible substrings
        for j in range(i+1, len(string)+1):
            sub = string[i:j]
            # compares substring to the reverse
            if sub == sub[::-1]:
                list.append(sub) # appends to list if so
```



```
    return list

while True: # main menu loop
    print("\n\nMAIN MENU\n",
          "1.Print all possible substrings\n",
          "2.Print substrings of length L\n",
          "3.Print substrings of length L with N distinct letters\n",
          "4.Print substrings of max length with N distinct letters\n",
          "5.Print Palindrome substrings\n",
          "6.Exit!")

    choice = int(input("\nEnter the Choice:"))
    if choice in [1, 2, 3, 4, 5]:
        string = input("Enter the String: ")

        if len(string) > 2: # checks if length is 2
            if choice == 1:
                print("\nAll possible substrings are: ")
                list = subs(string, len(string))
                for i in list:
                    print(i, end=" ")

            elif choice == 2:
                l = int(input("Enter the length of substring: "))
                print(f"\nSubstrings of length {l} are: ")
                list = subs_length(string, l)
                for i in list:
                    print(i, end=" ")

            elif choice == 3:
                n = int(input("Enter the number of distinct letters: "))
                l = int(input("Enter the length of substring: "))
                print(
                    f"\nSubstrings of length {l} and {n} distinct letters are: ")
                list = subs_dist_length(string, l, n)
                for i in list:
                    print(i, end=" ")

            elif choice == 4:
                n = int(input("Enter the number of distinct letters: "))
```

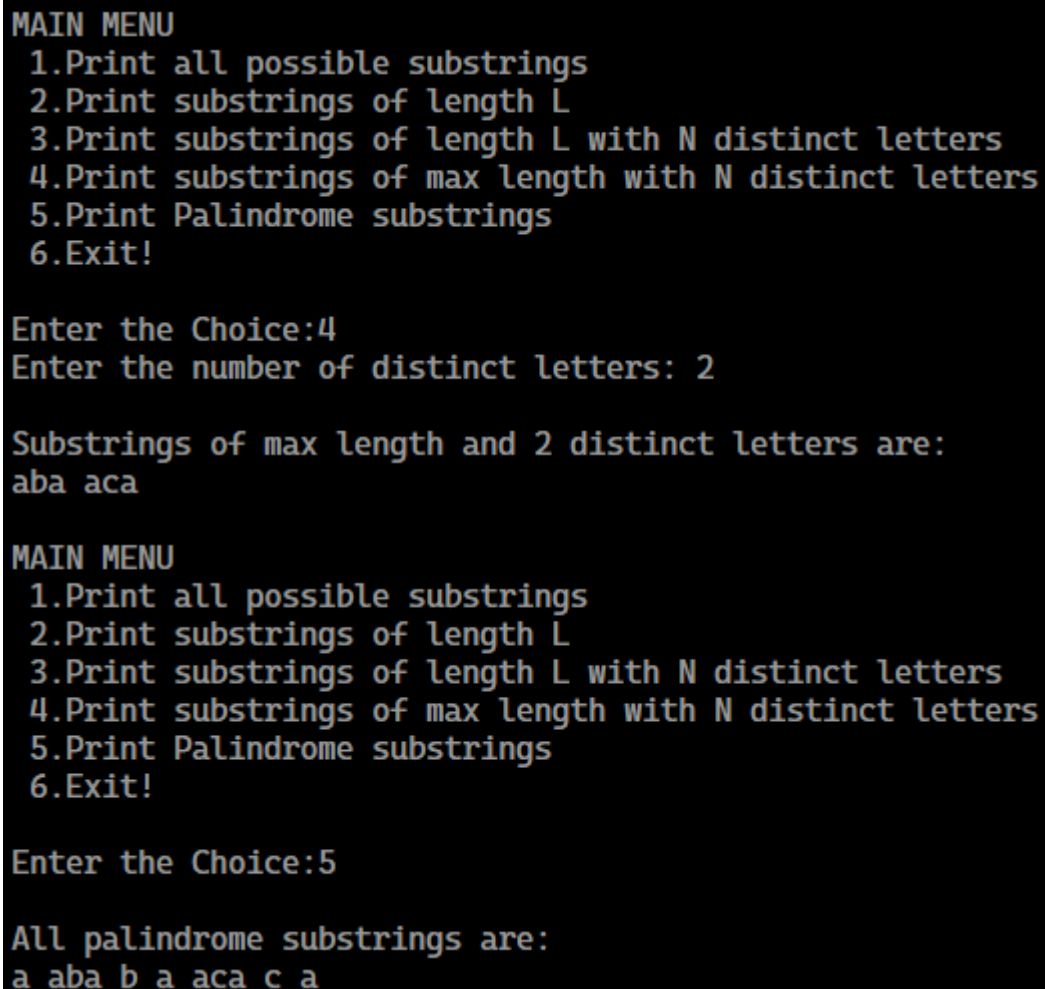
```
        print(
            f"\nSubstrings of max length and {n} distinct letters are: ")
        subs_dist_max_length(string, n)

    elif choice == 5:
        print("\nAll palindrome substrings are: ")
        list = palindrome_subs(string)
        for i in list:
            print(i, end=" ")

    else:
        print("Error!, Please enter atleast 1 letter string!!")
        break

else:
    print("Come Back Later!")
    break
```

#### SAMPLE INPUT-OUTPUT



```
MAIN MENU
1.Print all possible substrings
2.Print substrings of length L
3.Print substrings of length L with N distinct letters
4.Print substrings of max length with N distinct letters
5.Print Palindrome substrings
6.Exit!

Enter the Choice:4
Enter the number of distinct letters: 2

Substrings of max length and 2 distinct letters are:
aba aca

MAIN MENU
1.Print all possible substrings
2.Print substrings of length L
3.Print substrings of length L with N distinct letters
4.Print substrings of max length with N distinct letters
5.Print Palindrome substrings
6.Exit!

Enter the Choice:5

All palindrome substrings are:
a aba b a aca c a
```

Enter the String: abaca

MAIN MENU

- 1.Print all possible substrings
- 2.Print substrings of length L
- 3.Print substrings of length L with N distinct letters
- 4.Print substrings of max length with N distinct letters
- 5.Print Palindrome substrings
- 6.Exit!

Enter the Choice:1

All possible substrings are:

a ab aba abac abaca b ba bac baca a ac aca c ca a

MAIN MENU

- 1.Print all possible substrings
- 2.Print substrings of length L
- 3.Print substrings of length L with N distinct letters
- 4.Print substrings of max length with N distinct letters
- 5.Print Palindrome substrings
- 6.Exit!

Enter the Choice:2

Enter the length of substring: 3

Substrings of length 3 are:

aba bac aca

MAIN MENU

- 1.Print all possible substrings
- 2.Print substrings of length L
- 3.Print substrings of length L with N distinct letters
- 4.Print substrings of max length with N distinct letters
- 5.Print Palindrome substrings
- 6.Exit!

Enter the Choice:3

Enter the number of distinct letters: 2

Enter the length of substring: 3

Substrings of length 3 and 2 distinct letters are:

aba aca

**TEST CASES**

Test Cases No.	Description	Input	Expected output	Actual output	result
1	Print sub strings	abaca	a ab aba abaca b ba bac bac a ac aca c ca a	a ab aba abaca b ba bac bac 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**

[Click Here for the Code](#)

## PAIR OF RABBITS IN 'N' MONTHS

### AIM

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

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

### THEORY

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

### PROGRAM

```
from tabulate import tabulate # module to print table

# function for finding the number of rabbit pairs
def count_rabbits(n):
    if n == 1 or n == 2:
        return 1
    else:
        return count_rabbits(n-1) + count_rabbits(n-2)

N = int(input("Enter the number of months: ")) # input for number of months
heading = ["Month", "Rabbit Pairs"]
data = []
for i in range(1, N+1): # loop to append the data to the list
    data.append([i, count_rabbits(i)])

print(tabulate(data, headers=heading, tablefmt="fancy_grid")) # prints the table
```

**SAMPLE INPUT-OUTPUT**

Enter the number of months: 12

Month	Rabbit Pairs
1	1
2	1
3	2
4	3
5	5
6	8
7	13
8	21
9	34
10	55
11	89
12	144

**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**

[Click Here for the Code](#)

## OPERATIONS ON LIST OF INTEGERS

### AIM

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

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

### THEORY

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

### PROGRAM

```
# Function to rotate the list by k positions
def rotate_list(l, k):
    return l[-k:] + l[:-k] # Returns rotated list

# Function to remove duplicates from the list
def remove_duplicates(l):
    # Using set() to remove duplicates
    return list(set(l))

# Function to evaluate function
def evaluate_function(l):
    return [x**2 - x for x in l] #Returns evaluated list
```

```
# Function to merge two lists
def merge_lists(l1, l2):
    # Using sorted() to sort the lists
    l1 = sorted(l1)
    l2 = sorted(l2)
    # Using extend() to merge the lists
    l1.extend(l2)
    return l1

# Main function
def main():
    # Taking input from the user
    l = input("Enter the list of numbers: ").split()
    k = int(input("Enter the number of positions to rotate: "))

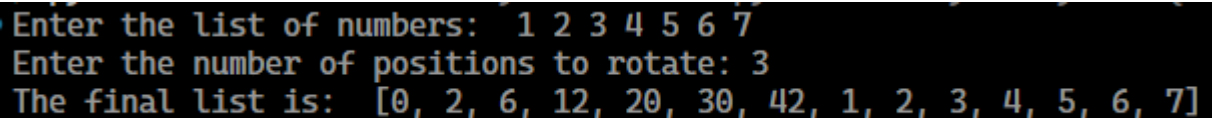
    # Converting the list to integers
    l = [int(x) for x in l]

    # Calling the functions
    l = rotate_list(l, k)
    t = tuple(l)
    l = remove_duplicates(l)
    l = evaluate_function(l)
    l = merge_lists(l, t)

    # Printing the final list
    print("The final list is: ", l)

# Calling the main function
main()
```

#### SAMPLE INPUT-OUTPUT



```
Enter the list of numbers: 1 2 3 4 5 6 7
Enter the number of positions to rotate: 3
The final list is: [0, 2, 6, 12, 20, 30, 42, 1, 2, 3, 4, 5, 6, 7]
```



**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 w r r	error	error	pass

**RESULT**

Program executed Successfully and the output is obtained.

**GIT LINK**

[Click Here for the Code](#)

## 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

# Function to read the file 'iris.json' line by line
def read_as_lines(filename):
    with open(filename, "r") as text_file:
        text_content = text_file.readlines()
    return text_content

# Function to read the file 'iris.json' as a dictionary
def read_as_dict(filename):
    with open(filename, "r") as json_file:
        json_content = json.load(json_file)
    return json_content

# Function to get the details of all flowers whose species is "setosa"
def get_flowers(dictionary_list , flower_species):
    flowers = []
    for flower in dictionary_list:
```

```
        if flower["species"] == flower_species:
            flowers.append(flower)
    return flowers

# Function to get the minimum petal area and max sepal area in each species
def get_sepal_and_petal_area(dictionary_list):
    species_list = sorted({flower["species"] for flower in dictionary_list})
    results = {}
    for species in species_list:
        sepal_areas = []
        petal_areas = []
        for flower in dictionary_list:
            if flower["species"] == species:
                sepal_areas.append(flower["sepalLength"] * flower["sepalWidth"])
                petal_areas.append(flower["petalLength"] * flower["petalWidth"])
        results[species] = {"min_sepal_area": min(sepal_areas), "max_petal_area": max(petal_areas)}
    return results

# Function to sort the list of dictionaries according to the total area are sepal and petal
def sort_by_total_area(dictionary_list):
    for flower in dictionary_list:
        total_area = flower["sepalLength"] * flower["sepalWidth"]
        + flower["petalLength"] * flower["petalWidth"]
        total_area = round(total_area, 2)
        flower["totalArea"] = total_area
    return sorted(dictionary_list, key=lambda flower: flower["totalArea"])

# Main program
print("Welcome to the program!\n",
      "1. Read the file 'iris.json' as lines\n",
      "2. Read the file 'iris.json' as json\n",
      "3. Get the details of all flowers whose species is 'setosa'\n",
      "4. Get the minimum petal area and max sepal area in each species\n",
      "5. Sort the list of dictionaries according to the total area are sepal and petal\n",
      "6. Exit the program\n")

while True:
    choice = input("Enter your choice: ")
    print("\n-----\n")
    if choice == "1":
        print("Contents of the file 'iris.json' read as lines: ")
```

```

        contents_as_lines = read_as_lines("iris.json")
        for line in contents_as_lines:
            print(line.replace("\n", "\\n"))
    elif choice == "2":
        print("Contents of the file 'iris.json' read as json: \n")
        contents_as_json = read_as_dict("iris.json")
        for dictionary in contents_as_json:
            print(dictionary)
    elif choice == "3":
        setosa = get_flowers(read_as_dict("iris.json"), "setosa")
        for flower in setosa:
            print(flower)
    elif choice == "4":
        areas = get_sepal_and_petal_area(read_as_dict("iris.json"))
        for species in areas:
            print(f"{species}: {areas[species]}")
    elif choice == "5":
        sorted_by_area = sort_by_total_area(read_as_dict("iris.json"))
        for flower in sorted_by_area:
            print(flower)
    elif choice == "6":
        print("Exiting the program...")
        break
    else:
        print("Enter a valid choice!")

```

## SAMPLE INPUT-OUTPUT

```

Welcome to the program!
1. Read the file 'iris.json' as lines
2. Read the file 'iris.json' as json
3. Get the details of all flowers whose species is 'setosa'
4. Get 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
6. Exit the program

Enter your choice: 1

-----

Contents of the file 'iris.json' read as lines:
[\n
{ "sepalLength": 5.1, "sepalWidth": 3.5, "petalLength": 1.4, "petalWidth": 0.2, "species": "setosa" },\n
{ "sepalLength": 4.9, "sepalWidth": 3.0, "petalLength": 1.4, "petalWidth": 0.2, "species": "setosa" },\n
{ "sepalLength": 4.7, "sepalWidth": 3.2, "petalLength": 1.3, "petalWidth": 0.2, "species": "setosa" },\n
{ "sepalLength": 4.6, "sepalWidth": 3.1, "petalLength": 1.5, "petalWidth": 0.2, "species": "setosa" },\n
{ "sepalLength": 5.0, "sepalWidth": 3.6, "petalLength": 1.4, "petalWidth": 0.2, "species": "setosa" },\n
{ "sepalLength": 5.4, "sepalWidth": 3.9, "petalLength": 1.7, "petalWidth": 0.4, "species": "setosa" },\n
{ "sepalLength": 4.6, "sepalWidth": 3.4, "petalLength": 1.4, "petalWidth": 0.3, "species": "setosa" },\n
{ "sepalLength": 5.0, "sepalWidth": 3.4, "petalLength": 1.5, "petalWidth": 0.2, "species": "setosa" },\n
{ "sepalLength": 4.4, "sepalWidth": 2.9, "petalLength": 1.4, "petalWidth": 0.2, "species": "setosa" },\n
{ "sepalLength": 4.9, "sepalWidth": 3.1, "petalLength": 1.5, "petalWidth": 0.1, "species": "setosa" },\n
{ "sepalLength": 5.4, "sepalWidth": 3.7, "petalLength": 1.5, "petalWidth": 0.2, "species": "setosa" },\n
{ "sepalLength": 4.8, "sepalWidth": 3.4, "petalLength": 1.6, "petalWidth": 0.2, "species": "setosa" },\n

```

Enter your choice: 2

Contents of the file 'iris.json' read as json:

```
{'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'}
```

Enter your choice: 3

```
{'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'}
```

Enter your choice: 4

```
setosa: {'min_sepal_area': 10.35, 'max_petal_area': 0.96}
versicolor: {'min_sepal_area': 10.0, 'max_petal_area': 8.64}
virginica: {'min_sepal_area': 12.25, 'max_petal_area': 15.87}
```

Enter your choice: 5

```
{'sepalLength': 5.0, 'sepalWidth': 2.0, 'petalLength': 3.5, 'petalWidth': 1.0, 'species': 'versicolor', 'totalArea': 10.0}
{'sepalLength': 4.5, 'sepalWidth': 2.3, 'petalLength': 1.3, 'petalWidth': 0.3, 'species': 'setosa', 'totalArea': 10.35}
{'sepalLength': 5.0, 'sepalWidth': 2.3, 'petalLength': 3.3, 'petalWidth': 1.0, 'species': 'versicolor', 'totalArea': 11.5}
{'sepalLength': 4.9, 'sepalWidth': 2.4, 'petalLength': 3.3, 'petalWidth': 1.0, 'species': 'versicolor', 'totalArea': 11.76}
{'sepalLength': 4.9, 'sepalWidth': 2.5, 'petalLength': 4.5, 'petalWidth': 1.7, 'species': 'virginica', 'totalArea': 12.25}
{'sepalLength': 5.5, 'sepalWidth': 2.3, 'petalLength': 4.0, 'petalWidth': 1.3, 'species': 'versicolor', 'totalArea': 12.65}
{'sepalLength': 5.1, 'sepalWidth': 2.5, 'petalLength': 3.0, 'petalWidth': 1.1, 'species': 'versicolor', 'totalArea': 12.75}
{'sepalLength': 4.4, 'sepalWidth': 2.9, 'petalLength': 1.4, 'petalWidth': 0.2, 'species': 'setosa', 'totalArea': 12.76}
{'sepalLength': 4.3, 'sepalWidth': 3.0, 'petalLength': 1.1, 'petalWidth': 0.1, 'species': 'setosa', 'totalArea': 12.9}
{'sepalLength': 4.4, 'sepalWidth': 3.0, 'petalLength': 1.3, 'petalWidth': 0.2, 'species': 'setosa', 'totalArea': 13.2}
{'sepalLength': 6.0, 'sepalWidth': 2.2, 'petalLength': 4.0, 'petalWidth': 1.0, 'species': 'versicolor', 'totalArea': 13.2}
{'sepalLength': 5.5, 'sepalWidth': 2.4, 'petalLength': 3.8, 'petalWidth': 1.1, 'species': 'versicolor', 'totalArea': 13.2}
{'sepalLength': 5.5, 'sepalWidth': 2.4, 'petalLength': 3.7, 'petalWidth': 1.0, 'species': 'versicolor', 'totalArea': 13.2}
{'sepalLength': 6.0, 'sepalWidth': 2.2, 'petalLength': 5.0, 'petalWidth': 1.5, 'species': 'virginica', 'totalArea': 13.2}
{'sepalLength': 6.2, 'sepalWidth': 2.2, 'petalLength': 4.5, 'petalWidth': 1.5, 'species': 'versicolor', 'totalArea': 13.64}
{'sepalLength': 5.5, 'sepalWidth': 2.5, 'petalLength': 4.0, 'petalWidth': 1.3, 'species': 'versicolor', 'totalArea': 13.75}
{'sepalLength': 5.6, 'sepalWidth': 2.5, 'petalLength': 3.9, 'petalWidth': 1.1, 'species': 'versicolor', 'totalArea': 14.0}
{'sepalLength': 5.2, 'sepalWidth': 2.7, 'petalLength': 3.9, 'petalWidth': 1.4, 'species': 'versicolor', 'totalArea': 14.04}
```

## TEST CASES

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

[Click Here for the Code](#)

## BOX CLASS FOR SHAPES

### AIM

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

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

### THEORY

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

### PROGRAM

```
import random
class Box: # Box class
    length=0
    breadth=0
    height=0
    area=0
    volume=0
    shape="Cube"
    # *args is used to pass variable number of arguments
    def __init__(self,*args):
        if len(args)==1:
            self.length=args[0]
            self.shape
        elif len(args)==2:
            self.length=args[0]
            self.height=args[1]
            self.shape="Square Prism"
        elif len(args)==3:
            self.length=args[0]
```

```
        self.breadth=args[1]
        self.height=args[2]
        self.shape="Rectangular Prism"
# area and volume methods are defined here
def area(self):
    if self.breadth==0 and self.height==0:
        self.area=6*self.length**2
    elif self.breadth==0:
        self.area=(2*self.length**2)+(4*self.length*self.height)
    else:
        self.area=2*(self.breadth*self.length+
        self.height*self.length+self.height*self.breadth)
def volume(self):
    if self.breadth==0 and self.height==0:
        self.volume=self.length**3
    elif self.breadth==0:
        self.volume=self.length**2*self.height
    else:
        self.volume=self.breadth*self.length*self.height
# display method is defined here
def display(self):
    print(f"Area    : {self.area}")
    print(f"Volume  : {self.volume}")
    print(f"Ratio   : {self.ratio()}\n")
# ratio method is defined here
def ratio(self):
    return self.volume/self.area
# cube, square prism and rectangular prism are generated here
def generate_cube():
    length=random.randint(1,1000)
    return Box(length)
def generate_square_prism():
    length=random.randint(1,1000)
    height=random.randint(1,1000)
    return Box(length,height)
def generate_rectangular_prism():
    length=random.randint(1,1000)
    breadth=random.randint(1,1000)
    height=random.randint(1,1000)
    return Box(length,breadth,height)
# main function
```



```
def main():
    num_boxes=int(input("Enter the number of boxes to generate: "))
    boxes=[]
    for i in range(num_boxes):
        if i%3==0:
            box=generate_cube()
        elif i%3==1:
            box=generate_square_prism()
        else:
            box=generate_rectangular_prism()
        print(f"{box.shape} : Dimensions = [{box.length},{box.breadth},{box.height}]")
        box.area()
        box.volume()
        box.display()
        boxes.append(box)

    # max_ratio_box is the box with maximum volume:area ratio
    max_ratio_box=max(boxes,key=lambda box:box.ratio())
    max_ratio=max_ratio_box.ratio()
    if isinstance(max_ratio_box,Box):
        if max_ratio_box.length>0 and max_ratio_box.breadth==0
        and max_ratio_box.height==0:
            box_type="Cube"
        elif max_ratio_box.length>0 and max_ratio_box.breadth>0:
            box_type="Rectangular Prism"
        else:
            box_type="Square Prism"
        print(f"Maximum volume:area ratio for {box_type}. Value = {max_ratio}")
    else:
        print("Something went wrong")

main()
```

## SAMPLE INPUT-OUTPUT

```
Enter the number of boxes to generate: 4
Cube : Dimensions = [268,0,0]
Area  : 430944
Volume : 19248832
Ratio  : 44.666666666666664

Square Prism : Dimensions = [351,0,995]
Area  : 1643382
Volume : 122584995
Ratio  : 74.5931225971807

Rectangular Prism : Dimensions = [459,867,410]
Area  : 1883226
Volume : 163160730
Ratio  : 86.63895358284135

Cube : Dimensions = [419,0,0]
Area  : 1053366
Volume : 73560059
Ratio  : 69.83333333333333

Maximum volume:area ratio for Rectangular Prism. Value = 86.63895358284135
```

## TEST CASES

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

## RESULT

Program executed Successfully and the output is obtained.

## GIT LINK

[Click Here for the Code](#)

## 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 _3D_Shapes: # Abstract class
    def print_Volume(self):
        print("Volume: ", self.volume)

    def print_Area(self):
        print("Area: ", self.area)

class Cylinder(_3D_Shapes): # Cylinder class
    def __init__(self, radius, height):
        self.radius = radius
        self.height = height

    def calculate_Volume(self): # Volume of cylinder
        self.volume = 3.14 * self.radius * self.radius * self.height
        _3D_Shapes.print_Volume(self)

    def calculate_Area(self): # Surface area of cylinder
        self.area = 2 * 3.14 * self.radius * (self.radius + self.height)
        _3D_Shapes.print_Area(self)

class Sphere(_3D_Shapes): # Sphere class
    def __init__(self, radius):
        self.radius = radius
```

```
def calculate_Volume(self): # Volume of sphere
    self.volume = (4/3) * 3.14 * self.radius * self.radius * self.radius
    _3D_Shapes.print_Volume(self)

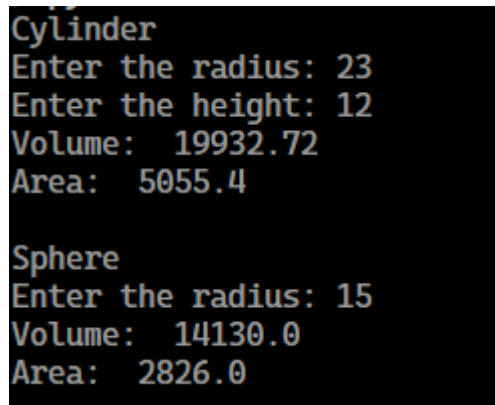
def calculate_Area(self): # Surface area of sphere
    self.area = 4 * 3.14 * self.radius * self.radius
    _3D_Shapes.print_Area(self)

def main(): # main function
    print("Cylinder")
    cylindr= int(input("Enter the radius: "))
    cylindh= int(input("Enter the height: "))
    cylin = Cylinder(cylindr, cylindh)
    cylin.calculate_Volume()
    cylin.calculate_Area()

    print()
    print("Sphere")
    sphr = int(input("Enter the radius: "))
    sph = Sphere(sphr)
    sph.calculate_Volume()
    sph.calculate_Area()

main() # main function call
```

### SAMPLE INPUT-OUTPUT



```
Cylinder
Enter the radius: 23
Enter the height: 12
Volume: 19932.72
Area: 5055.4

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

**TEST CASES**

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

**RESULT**

Program executed Successfully and the output is obtained.

**GIT LINK**

[Click Here for the Code](#)

# 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
import sys
import numpy as np

pygame.init()

# Constants
WIDTH = 600
HEIGHT = 600
LINE_WIDTH = 15
BOARD_ROWS = 3
BOARD_COLS = 3
SQUARE_SIZE = 200
CIRCLE_RADIUS = 60
CIRCLE_WIDTH = 15
CROSS_WIDTH = 25
SPACE = 55
RED = (255, 0, 0)
BG_COLOR = (28, 170, 156)
LINE_COLOR = (23, 145, 135)
CIRCLE_COLOR = (239, 231, 200)
CROSS_COLOR = (66, 66, 66)

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

# Board
board = np.zeros((BOARD_ROWS, BOARD_COLS))
```

```
# Functions
def draw_lines():
    # 1st horizontal line
    pygame.draw.line(screen, LINE_COLOR,
                     (0, SQUARE_SIZE), (WIDTH, SQUARE_SIZE), LINE_WIDTH)
    # 2nd horizontal line
    pygame.draw.line(screen, LINE_COLOR,
                     (0, 2 * SQUARE_SIZE), (WIDTH, 2 * SQUARE_SIZE), LINE_WIDTH)
    # 1st vertical line
    pygame.draw.line(screen, LINE_COLOR,
                     (SQUARE_SIZE, 0), (SQUARE_SIZE, HEIGHT), LINE_WIDTH)
    # 2nd vertical line
    pygame.draw.line(screen, LINE_COLOR,
                     (2 * SQUARE_SIZE, 0), (2 * SQUARE_SIZE, HEIGHT), LINE_WIDTH)

def draw_figures():
    for row in range(BOARD_ROWS):
        for col in range(BOARD_COLS):
            if board[row][col] == 1:
                pygame.draw.circle(screen, CIRCLE_COLOR,
                                   (int(col * SQUARE_SIZE + SQUARE_SIZE // 2),
                                    int(row * SQUARE_SIZE + SQUARE_SIZE // 2)),
                                   CIRCLE_RADIUS, CIRCLE_WIDTH)
            elif board[row][col] == 2:
                pygame.draw.line(screen, CROSS_COLOR,
                                 (col * SQUARE_SIZE + SPACE,
                                  row * SQUARE_SIZE + SQUARE_SIZE - SPACE),
                                 (col * SQUARE_SIZE + SQUARE_SIZE - SPACE,
                                  row * SQUARE_SIZE + SPACE),
                                 CROSS_WIDTH)
                pygame.draw.line(screen, CROSS_COLOR,
                                 (col * SQUARE_SIZE + SPACE, row * SQUARE_SIZE + SPACE),
                                 (col * SQUARE_SIZE + SQUARE_SIZE - SPACE,
                                  row * SQUARE_SIZE + SQUARE_SIZE - SPACE),
                                 CROSS_WIDTH)

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

def available_square(row, col):
    return board[row][col] == 0
```

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

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

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

    # Ascending diagonal win check
    if board[2][0] == player and board[1][1] == player and board[0][2] == player:
        draw_asc_diagonal(player)
        return True

    # Descending diagonal win check
    if board[0][0] == player and board[1][1] == player and board[2][2] == player:
        draw_desc_diagonal(player)
        return True
    return False

def draw_vertical_winning_line(col, player):
    posX = col * SQUARE_SIZE + SQUARE_SIZE // 2
    if player == 1:
        color = CIRCLE_COLOR
    elif player == 2:
        color = CROSS_COLOR
    pygame.draw.line(screen, color, (posX, 15), (posX, HEIGHT - 15), 15)

def draw_horizontal_winning_line(row, player):
    posY = row * SQUARE_SIZE + SQUARE_SIZE // 2
```



```
    if player == 1:
        color = CIRCLE_COLOR
    elif player == 2:
        color = CROSS_COLOR
    pygame.draw.line(screen, color, (15, posY), (WIDTH - 15, posY), 15)

def draw_asc_diagonal(player):
    if player == 1:
        color = CIRCLE_COLOR
    elif player == 2:
        color = CROSS_COLOR
    pygame.draw.line(screen, color, (15, HEIGHT - 15), (WIDTH - 15, 15), 15)

def draw_desc_diagonal(player):
    if player == 1:
        color = CIRCLE_COLOR
    elif player == 2:
        color = CROSS_COLOR
    pygame.draw.line(screen, color, (15, 15), (WIDTH - 15, HEIGHT - 15), 15)

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

# Variables
player = 1
game_over = False

# Main loop
draw_lines()

while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
        if event.type == pygame.MOUSEBUTTONDOWN and not game_over:
            mouseX = event.pos[0]
```

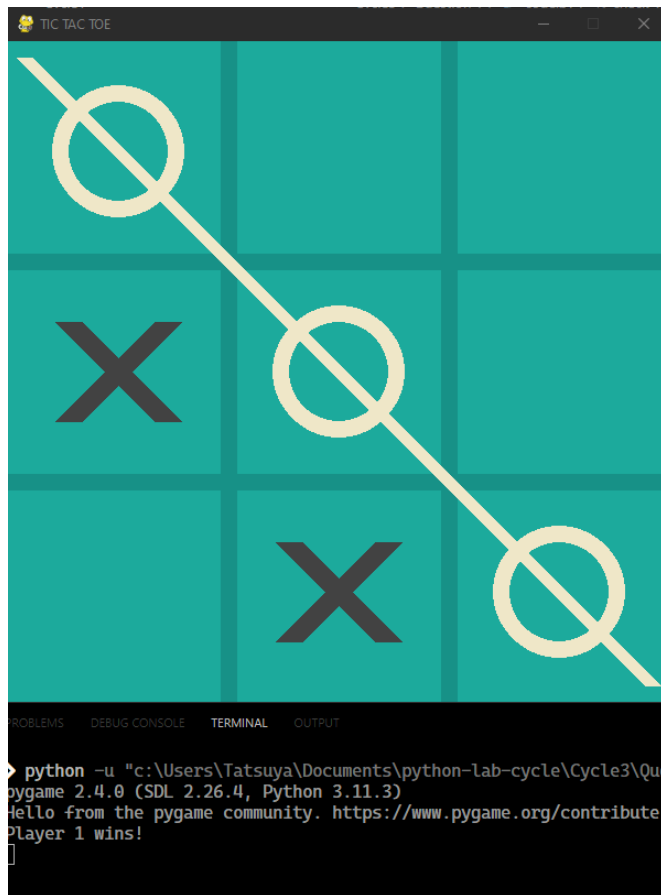
```
mouseY = event.pos[1]
clicked_row = int(mouseY // SQUARE_SIZE)
clicked_col = int(mouseX // SQUARE_SIZE)
if available_square(clicked_row, clicked_col):
    mark_square(clicked_row, clicked_col, player)
    if check_win(player):
        game_over = True
        print("Player {} wins!".format(player))

    elif is_board_full():
        game_over = True
        print("It's a tie!")

    player = player % 2 + 1
    draw_figures()
if event.type == pygame.KEYDOWN:
    if event.key == pygame.K_r:
        restart()
        game_over = False

pygame.display.update()
```

## **SAMPLE INPUT-OUTPUT**



## TEST CASES

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

## RESULT

Program executed Successfully and the output is obtained.

## GIT LINK

Click Here for the Code

# 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 input_data(): # Function to take input from user
    print("Enter the number of rows and columns of the matrix: ")
    rows, columns = map(int, input().split())
    print("Enter the elements of the matrix: ")
    matrix = []
    for i in range(rows):
        row = list(map(int, input().split()))
        matrix.append(row)
    return np.array(matrix)

# Function to calculate mean of a matrix
def calculate_mean(matrix):
    return np.mean(matrix, axis=0)

# Function to center a matrix
def center_matrix(matrix, mean):
    return matrix - mean

# Function to calculate covariance of a matrix
def calculate_covariance(centered_matrix):
    return np.cov(centered_matrix.T)

# Function to perform eigendecomposition of a matrix
def perform_eigendecomposition(covariance_matrix):
```

```
        return np.linalg.eig(covariance_matrix)

# Function to compute PCA of a matrix
def compute_PCA(matrix):
    mean_matrix = calculate_mean(matrix)
    centered_matrix = center_matrix(matrix, mean_matrix)
    covariance_matrix = calculate_covariance(centered_matrix)
    eigenvalues, eigenvectors = perform_eigendecomposition(covariance_matrix)
    projection_matrix = (eigenvectors.T[:][:2]).T
    projected_data = projection_matrix.T.dot(centered_matrix.T)
    return projected_data.T

# Main function
input_matrix = input_data()
print("Input Matrix:")
print(input_matrix)
pca_result = compute_PCA(input_matrix)
print("PCA Result:")
print(pca_result)
```

### SAMPLE INPUT-OUTPUT

```
Enter rows and columns separated by a space: 3 3
Enter the values for matrix A of shape 3x3
Enter elements row by row separated by a space: 1 2 3
Enter elements row by row separated by a space: 4 5 6
Enter elements row by row separated by a space: 7 8 9
Input Matrix:
[[1 2 3]
 [4 5 6]
 [7 8 9]]
PCA Result:
[[ 2.22044605e-16 -5.19615242e+00]
 [ 0.00000000e+00  0.00000000e+00]
 [-2.22044605e-16  5.19615242e+00]]
```

### 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**

[Click Here for the Code](#)

## DATA VISUALIZATION

### AIM

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

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

### THEORY

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

### PROGRAM

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

# Read the CSV file into a pandas dataframe
df = pd.read_csv('Iris.csv')

# Define a function to create a bar chart of the frequency of each species
```

```
def frequencyBar():
    df.Species.value_counts().plot(figsize=(8,6), kind='bar', color=['r','g','b'], xlabel=
    plt.title("Frequency Bar Graph")
    plt.show()

# Define a function to apply PCA to the data and create a scatter plot of the first two pr
def pcaAppliedGraph():
    print("\nPCA Graph")
    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
    finalDf = pd.concat([principalDf, df['Species']], axis = 1)

    fig = plt.figure(figsize = (10,8))
    ax = fig.add_subplot(1,1,1)
    ax.set_xlabel('First Principle Component')
    ax.set_ylabel('Second Principal Component')
    ax.set_title('PCA Graph')
    targets = ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']
    colors = ['r', 'g', 'b']
    for target, color in zip(targets,colors):
        indicesToKeep = finalDf['Species'] == target
        ax.scatter(finalDf.loc[indicesToKeep, 'principal component 1'], finalDf.loc[indice
    ax.legend(targets)
    plt.show()

# Define a function to create histograms of the sepal and petal measurements
def distributionHistogramSepal():
    print("\nDistribution Histogram\n")
    plt.figure(figsize = (7, 5))
    x = df.SepalLengthCm
    plt.hist(x, color = "r")
    plt.title("Sepal Length Histogram")
    plt.xlabel("Sepal Length cm")
    plt.ylabel("Distribution Count")
    plt.show()

    print()
    plt.figure(figsize = (7, 5))
```



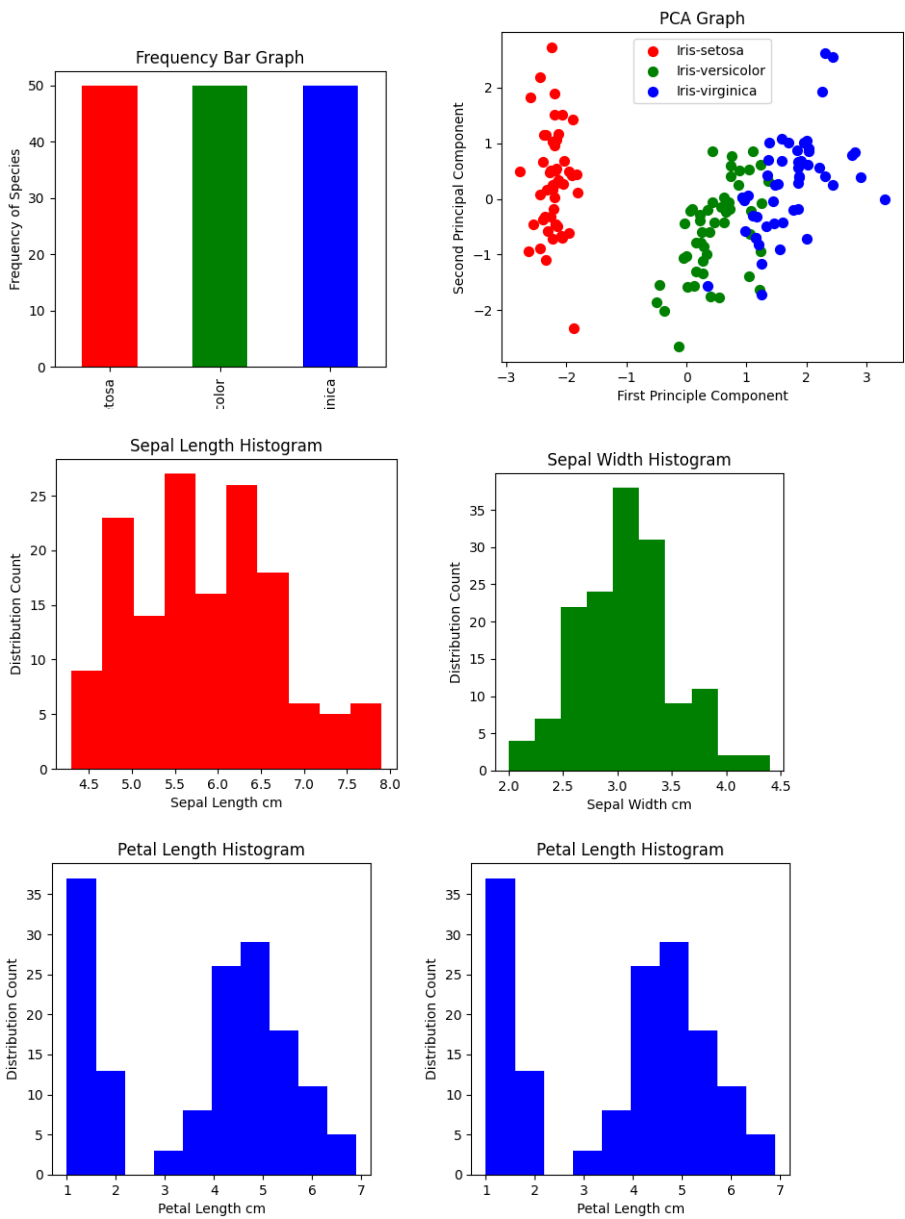
```
x = df.SepalWidthCm
plt.hist(x, color = "g")
plt.title("Sepal Width Histogram")
plt.xlabel("Sepal Width cm")
plt.ylabel("Distribution Count")
plt.show()

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

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

# Call the functions to display the graphs
frequencyBar()
pcaAppliedGraph()
distributionHistogramSepal()
distributionHistogramPetal()
```

SAMPLE INPUT-OUTPUT



TEST CASES

Test Cases	Description	Input	Expected output	Actual Output	Result
1	Check .csv file output of bar graph	Iris.csv file	Bar Graph	Bar Graph	Pass
2	Display Scattered Graph for Principal Component Values	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**

[Click Here for the Code](#)

## VEHICLE DETAILS

### AIM

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

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

### THEORY

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

### PROGRAM

```
import pickle, tabulate
import pandas as pd
from fpdf import FPDF

# Class for Vehicle
class Vehicle:
    def __init__(self, engine_no, model, owner_name, vehicle_type, mileage, vendor, regist.
```

```
        self.mileage = mileage
        self.vendor = vendor
        self.registration_no = registration_no

    def __str__(self):
        s = "{:<15} {:<15} {:<15} {:<15} {:<15} {:<15} {:<15}".format(self.engine_no, self
        print(s)

# Class for Vehicle Collection
class VehicleCollection:
    def __init__(self):
        self.vehicle_list = []
    # Function to add vehicle to the list
    def add_vehicle(self, vehicle):
        self.vehicle_list.append(vehicle)
    # Function to delete vehicle from the list
    def delete_vehicle(self, registration_no):
        for i in self.vehicle_list:
            if i.registration_no == registration_no:
                self.vehicle_list.remove(i)
    # Function to modify vehicle in the list
    def modify_vehicle(self, registration_no, new_vehicle):
        for i in self.vehicle_list:
            if i.registration_no == registration_no:
                self.vehicle_list.remove(i)
                self.vehicle_list.append(new_vehicle)
    # Function to display vehicle list
    def display(self):
        print("{:<15} {:<15} {:<15} {:<15} {:<15} {:<15} {:<15}".format('Engine No', 'Mode
        for vehicle in self.vehicle_list:
            vehicle.__str__()
            print()
    # Function to sort vehicle list by mileage
    def sort_by_mileage(self):
        self.vehicle_list.sort(key=lambda x: x.mileage)
    # Function to store vehicle list as pickle file
    def store_pickle(self):
        with open('vehicle.pickle', 'wb') as f:
            pickle.dump(self.vehicle_list, f)
    # Function to load vehicle list from pickle file
    def load_pickle(self):
```

```
        with open('vehicle.pickle', 'rb') as f:
            self.vehicle_list = pickle.load(f)
# Function to filter vehicle list
def filter(self, attribute, value):
    filtered_list = []
    for i in self.vehicle_list:
        if attribute == 'engine_no':
            if i.engine_no == value:
                filtered_list.append(i)
        elif attribute == 'model':
            if i.model == value:
                filtered_list.append(i)
        elif attribute == 'owner_name':
            if i.owner_name == value:
                filtered_list.append(i)
        elif attribute == 'vehicle_type':
            if i.vehicle_type == value:
                filtered_list.append(i)
        elif attribute == 'mileage':
            if i.mileage == value:
                filtered_list.append(i)
        elif attribute == 'vendor':
            if i.vendor == value:
                filtered_list.append(i)
        elif attribute == 'registration_no':
            if i.registration_no == value:
                filtered_list.append(i)
    return filtered_list
# Function to export vehicle list as pdf
def export_pdf(self, filtered_list):
    pdf = FPDF()
    pdf.add_page()
    pdf.set_font("Arial", size=12)
    pdf.cell(200, 10, txt="Vehicle Details", ln=1, align='C')
    pdf.cell(200, 10, txt="{:<15} {:<15} {:<15} {:<15} {:<15} {:<15} {:<15}".format('E', 'N', 'M', 'V', 'O', 'R', 'D'), ln=2, align='L')
    for i in filtered_list:
        pdf.cell(200, 10, txt="{:<15} {:<15} {:<15} {:<15} {:<15} {:<15} {:<15}".format(i.engine_no, i.model, i.owner_name, i.vehicle_type, i.mileage, i.vendor, i.registration_no), ln=3, align='L')
    pdf.output("vehicle.pdf")

# Main function
def main():
```

```
vehicle_collection = VehicleCollection()
while True:
    print("Vehicle Collection"
          "\n1. Add Vehicle"
          "\n2. Delete Vehicle"
          "\n3. Modify Vehicle"
          "\n4. Display Vehicle"
          "\n5. Sort by Mileage"
          "\n6. Store as pickle file"
          "\n7. Load from pickle file"
          "\n8. Filter"
          "\n9. Export as pdf"
          "\n10. Exit")
    choice = int(input("Enter your choice: "))
    if choice == 1:
        engine_no = int(input("Enter engine no: "))
        model = input("Enter model: ")
        owner_name = input("Enter owner name: ")
        vehicle_type = input("Enter vehicle type: ")
        mileage = float(input("Enter mileage: "))
        vendor = input("Enter vendor: ")
        registration_no = int(input("Enter registration no: "))
        vehicle = Vehicle(engine_no, model, owner_name, vehicle_type, mileage, vendor,
                           registration_no)
        vehicle_collection.add_vehicle(vehicle)
    elif choice == 2:
        registration_no = int(input("Enter registration no of vehicle to be deleted: "))
        vehicle_collection.delete_vehicle(registration_no)
    elif choice == 3:
        registration_no = int(input("Enter registration no of vehicle to be modified: "))
        engine_no = int(input("Enter engine no: "))
        model = input("Enter model: ")
        owner_name = input("Enter owner name: ")
        vehicle_type = input("Enter vehicle type: ")
        mileage = float(input("Enter mileage: "))
        vendor = input("Enter vendor: ")
        registration_no = int(input("Enter registration no: "))
        new_vehicle = Vehicle(engine_no, model, owner_name, vehicle_type, mileage, vendor,
                               registration_no)
        vehicle_collection.modify_vehicle(registration_no, new_vehicle)
    elif choice == 4:
        vehicle_collection.display()
        print()
```

```
elif choice == 5:
    vehicle_collection.sort_by_mileage()
elif choice == 6:
    vehicle_collection.store_pickle()
elif choice == 7:
    vehicle_collection.load_pickle()
elif choice == 8:
    attribute = input("Enter attribute to be filtered: ")
    value = input("Enter value to be filtered: ")
    filtered_list = vehicle_collection.filter(attribute, value)
    vehicle_collection.export_pdf(filtered_list)
elif choice == 9:
    vehicle_collection.export_pdf(vehicle_collection.vehicle_list)
elif choice == 10:
    exit()
else:
    print("Invalid choice")

# Calling main function
main()
```

### SAMPLE INPUT-OUTPUT

```
Vehicle Collection
1. Add Vehicle
2. Delete Vehicle
3. Modify Vehicle
4. Display Vehicle
5. Sort by Mileage
6. Store as pickle file
7. Load from pickle file
8. Filter
9. Export as pdf
10. Exit
Enter your choice: 7
Vehicle Collection
1. Add Vehicle
2. Delete Vehicle
3. Modify Vehicle
4. Display Vehicle
5. Sort by Mileage
6. Store as pickle file
7. Load from pickle file
8. Filter
9. Export as pdf
10. Exit
Enter your choice: 4
Engine No      Model      Owner Name  Vehicle Type  Mileage  Vendor      Registration No
1234           Audi       Rahul       Hatchback     20.0     Audi India  4568
4736           BMW        Sean        Sedan         17.0     BMW India   4569
3465           Kia        Lucy        SUV           30.0     Kia Motors  6754
```



```
Enter your choice: 1
Enter engine no: 7875
Enter model: McLaren
Enter owner name: Roxy
Enter vehicle type: Hypercar
Enter mileage: 15
Enter vendor: McLaren Motors
Enter registration no: 4572
Vehicle Collection
1. Add Vehicle
2. Delete Vehicle
3. Modify Vehicle
4. Display Vehicle
5. Sort by Mileage
6. Store as pickle file
7. Load from pickle file
8. Filter
9. Export as pdf
10. Exit
Enter your choice: 4
Engine No      Model      Owner Name  Vehicle Type  Mileage      Vendor      Registration No
1234           Audi       Rahul       Hatchback     20.0         Audi India  4568
4736           BMW        Sean        Sedan         17.0         BMW India  4569
3465           Kia        Lucy        SUV           30.0         Kia Motors  6754
7875           McLaren   Roxy        Hypercar      15.0         McLaren Motors  4572
```

## TEST CASES

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

## RESULT

Program executed Successfully and the output is obtained.

## GIT LINK

[Click Here for the Code](#)

## TKINTER UI APPLICATION

### AIM

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

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

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

### THEORY

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

### PROGRAM

```
from tkinter import *
from tkinter import ttk

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

#Constants
scr = Tk()
scr.geometry("990x400+40+50")
scr.configure(bg = "#202225")
scr.title("Vehicle Sale Data")
text= Label(scr, font=('Poppins',15,'bold'),text="VEHICLE DATA",bg="#202225",fg="white")
text.pack()
CarData = VehicleCollection()
DataFrame = LabelFrame(scr,text="Datas",bg = "#2f3136",fg="white")
DataFrame.pack(expand="yes",side=RIGHT)
```

```
CarDisplay = ttk.Treeview(DataFrame)

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

#Creating a frame for the Input details
In_Frame = LabelFrame(scr,text = "Input",bg = "#2f3136",fg="white")
In_Frame.pack(fill="x",expand="yes",padx = 20)
In_EngNo = StringVar(None)
#Creating Labels and respective entry boxes
In_EngNo_Label = Label(In_Frame,text = "Engine No",bg=' #2f3136',fg="white")
In_EngNo_Label.grid(row = 0,column=0,padx=10,pady=10)
In_EngNo = Entry(In_Frame)
In_EngNo.grid(row=0,column=1,padx=10,pady=10)

In_Model_Label = Label(In_Frame,text = "Model",bg=' #2f3136',fg="white")
In_Model_Label.grid(row = 1,column=0,padx=10,pady=10)
In_Model = Entry(In_Frame)
```

```
In_Model.grid(row=1,column=1,padx=10,pady=10)

In_Type_Label = Label(In_Frame,text = "Type",bg='#2f3136',fg="white")
In_Type_Label.grid(row = 2,column=0,padx=10,pady=10)
In_Type = Entry(In_Frame)
In_Type.grid(row=2,column=1,padx=10,pady=10)

In_Mileage_Label = Label(In_Frame,text = "Mileage",bg='#2f3136',fg="white")
In_Mileage_Label.grid(row = 0,column=2,padx=10,pady=10)
In_Mileage = Entry(In_Frame)
In_Mileage.grid(row=0,column=3,padx=10,pady=10)

In_Vendor_Label = Label(In_Frame,text = "Vendor",bg='#2f3136',fg="white")
In_Vendor_Label.grid(row = 1,column=2,padx=10,pady=10)
In_Vendor = Entry(In_Frame)
In_Vendor.grid(row=1,column=3,padx=10,pady=10)

In_Regno_Label = Label(In_Frame,text = "Reg No",bg='#2f3136',fg="white")
In_Regno_Label.grid(row = 2,column=2,padx=10,pady=10)
In_RegNo = Entry(In_Frame)
In_RegNo.grid(row=2,column=3,padx=10,pady=10)

In_Owner_Label = Label(In_Frame,text = "Owner",bg='#2f3136',fg="white")
In_Owner_Label.grid(row = 4,column=0,padx=10,pady=10)
In_Owner = Entry(In_Frame)
In_Owner.grid(row=4,column=1,padx=10,pady=10)

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

def add_to_entry_box():
    clear_inputs()
    #Select the record number
    sel_record = CarDisplay.focus()
```

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

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

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

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

def Load_File():
    scr.filename = filedialog.askopenfilename(initialdir="/",
        title="Select Pickle File",filetypes=(("pickle files","*.dat"),#
        ("All Files","*.*")))
    CarData.load_pickle(scr.filename)
    for i in CarData.vehicle_list:
        show_Values = tuple(i)
        CarDisplay.insert(parent = "",index = 'end',values=show_Values)
```

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

def Sort_mileage():
    #Sorting object data by Mileage
    CarData.sort_by_mileage()
    #Deleting items on window
    for data in CarDisplay.get_children():
        CarDisplay.delete(data)
    for i in CarData.vehicle_list:
        show_Values = tuple(i)
        CarDisplay.insert(parent = "",index = 'end',values=show_Values)

def Save_as_pdf():
    scr.filename = filedialog.askopenfilename(initialdir="/",
        title="Select Pdf File",filetypes=(("pdf files","*.pdf"),("All Files","*..*")))
    CarData.export_pdf(scr.filename)

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

    AddRec_button = Button(ButtonFrame,text = "Add",command=Add_the_record,#
        activebackground='#2f3136',bg='grey',fg="white")
    AddRec_button.grid(row=0,column=0,padx=10,pady=10)

    Modify_button = Button(ButtonFrame,text = "Modify",#
        activebackground='#2f3136',bg='grey',fg="white")
    Modify_button.grid(row=1,column=0,padx=10,pady=10)

    Open_button = Button(ButtonFrame,text = "Open File",command=Load_File,#
        activebackground='#2f3136',bg='grey',fg="white")
    Open_button.grid(row=2,column=0,padx=10,pady=10)

    Sort_button = Button(ButtonFrame,text = "Sort by Mileage",command=Sort_mileage,#
        activebackground='#2f3136',bg='grey',fg="white")
    Sort_button.grid(row=0,column=1,padx=10,pady=10)

    Delete_button = Button(ButtonFrame,text = "Delete Entry",command=Delete_record,#
```



## TEST CASES

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

## RESULT

Program executed Successfully and the output is obtained.

## GIT LINK

[Click Here for the Code](#)