CBL301 COMPUTATIONAL STATISTICS LAB

LABORATORY RECORD

23CSBS MDL23CSBS

B. Tech. Computer Science & Business Systems

Department of Computer Engineering

Model Engineering College Thrikkakara, Kochi 682021

Phone: +91.484.2575370

http://www.mec.ac.in, hodcs@mec.ac.in

CERTIFIED THAT THIS IS A BONAFIDE RECORD OF THE WORKDONE IN THE LABORATORY/WORKSHOP OF MODEL ENGINEERING COLLEGE, THRIKKAKARA

Staff in Charge **Head of the Department**

Submitted for the Practical Examination in

..... held on .....

Internal Examiner **External Examiner**

INDEX

Program

```
num = int(input("Enter a number: "))

if num % 2 == 0:
    print(f"{num} is Even")
else:
    print(f"{num} is Odd")
```

Output

```
Enter a number: 10
10 is Even

=== Code Execution Successful ===
Enter a number: 5
5 is Odd

=== Code Execution Successful ===
```

Experiment No.: 1 Date: _____

EVEN/ODD CHECK

Aim:

Ask the user for a number. Depending on whether the number is even or odd, print out an appropriate message to the user.

Algorithm:

1. Start
2. Take an integer input from the user
3. If the number is divisible by 2 print "Even"
4. Else print "Odd"
5. Stop

Result:

Program has been executed successfully and obtained the output.

Program

```
inputVal = int(input("Enter a number:- "))

print("\nDivisors of ", inputVal, ":")

for i in range(1, inputVal + 1):
 if inputVal % i == 0:
 print(f"{i} is a divisor of {inputVal}")
```

Output

```
Enter a number:- 10

Divisors of  10 :
1 is a divisor of 10
2 is a divisor of 10
5 is a divisor of 10
10 is a divisor of 10
```

Experiment No.: 2 Date: _____

DIVISORS OF A NUMBER

Aim:

To create a program that takes an integer input from the user and prints all the divisors of that number

Algorithm:

1. Start
2. Ask the user to enter an integer number and store it in a variable inputVal
3. Use a loop to check each number from 1 to inputVal
4. If the current number divides inputVal evenly (no remainder), print it as a divisor
5. Stop

Result:

Program has been executed successfully and obtained the output

Program

```
decimal_number = int(input("Enter a decimal number: "))

binary = bin(decimal_number)
octal = oct(decimal_number)
hexadecimal = hex(decimal_number)


print(f"Binary: {binary[2:]}")
print(f"Octal: {octal[2:]}")
print(f"Hexadecimal: {hexadecimal[2:].upper()}")
```

Output

```
Enter a decimal number: 11
Binary: 1011
Octal: 13
Hexadecimal: B
```

Experiment No.: 3 Date: _____

DECIMAL TO OTHER BASES CONVERSION

Aim:

To create a program that takes a decimal input from the user and prints the Decimal, Binary and Hexadecimal equivalent.

Algorithm:

1. Start
2. Prompt the user to enter a number in decimal
3. Read and convert the input to an integer
4. Convert the decimal number to:

```
->Binary using bin()

->Octal using oct()

->Hexadecimal using hex()
```

5. Remove prefixes (0b, 0o, 0x) if needed for display
6. Print the converted values
7. Stop

Result:

Program has been executed successfully and obtained the output.

Program

```
number=int(input("Enter a number: "))
i=1
print(f"Multiplication table of {number}:")
while i<=10:
 print(f"{number} x {i} = {number*i}")
 i += 1
```

Output

```
Enter a number: 2
Multiplication table of 2:
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
```

Experiment No.: 4 Date: _____

MULTIPLICATION TABLE

Aim:

Print the multiplication table of a given number

Algorithm:

1. Start
2. Input a number from the user and store it in variable number
3. Initialize a counter variable i=1
4. Display a heading: "Multiplication table of number"
5. Repeat steps 6–8 while i <= 10:
6. Calculate the product: result = number * i
7. Print the multiplication in the format: "number x i = result"
8. Increment i by 1 (i = i + 1)
9. End loop when i > 10
10. Stop

Result:

Program has been executed successfully and obtained the output.

Program

```
def fibo(n):
    if n <= 0:
        return n
    elif n == 1:
        return 1
    else:
        return fibo(n-1) + fibo(n-2)

a = int(input("Enter a number: "))
m = fibo(a)
print("Fibonacci Sequence for", a, "is", m)
```

Output

```
Enter a number: 6
Fibonacci Sequence for 6 is 8
```

Experiment No.: 5 Date: _____

NTH FIBONACCI

Aim:

To write a Python program to generate the nth Fibonacci number using recursion.

Algorithm:

1. Start
2. Define a recursive function fibo(n):

```
if n <= 0:
        return n
elif n == 1:
        return 1
else:
        return fibo(n-1) + fibo(n-2)
```

3. Accept an integer input a from the user.
4. Call fibo(a) and store the result in m.
5. Display the result as the nth Fibonacci number.
6. Stop

Result:

Program has been executed successfully and obtained the output.

Program

```
rows=int(input("Enter no of rows:"))
col=int(input("Enter no of coloums:"))
sparse ={}
matrix = []
print("Enter matrix value row by row")
for i in range(rows):
    row_values=list(map(int,input().split()))
    matrix.append(row_values)
    for j in range(col):
        if row_values[j]!=0:
            sparse[(i,j)]=row_values[j]
print("Matrix :")
for i in range(rows):
    print(matrix[i])
print("Sparse matrix :")
print(sparse)
```

Output

```
Enter no of rows:2
Enter no of coloums:3
Enter matrix value row by row
3 8 0
9 8 1
Matrix :
[3, 8, 0]
[9, 8, 1]
Sparse matrix :
{(0, 0): 3, (0, 1): 8, (1, 0): 9, (1, 1): 8, (1, 2): 1}
```

Experiment No.: 6 Date: _____

SPARSE MATRIX.

Aim:

To create a program that reads and displays a Sparse Matrix using Dictionary.

Algorithm:

1. Start.
2. Input the number of rows (rows) and number of columns (col).
3. Initialize:

```
sparse={}
matrix=[]
```

4. Input Matrix Elements:

```
For each row i from 0 to rows-1
     Read a list of integers as row_values
     Append row_values to matrix
     For each column j from 0 to col-1:
          If row_values[j] !=0:
               sparse[(i, j)] = row_values[j]
```

5. Print the full matrix.
6. Print the sparse dictionary.
7. Stop.

Result:

Program has been executed successfully and obtained the output.

Program

```
student_grades = {
"Maths" : 100,
"Science" : 98,
"History" : 100,
"Geography" : 100,
"English" : 98
}

def average_grade(grades):
 if not grades:
 return 0
 total = sum(grades.values())
 count = len(grades)
 average = total / count
 return average


average = average_grade(student_grades)
print(f"The average grade of student : {average} ")
```

Output

```
The average grade of student : 99.2
```

Experiment No.: 7 Date: _____

AVERAGE GRADE

Aim:

Create a Python dictionary representing a student's grades in different subjects (e.g., Math, Science, History). Write a function to calculate the average grade of the student across all subjects.

Algorithm:

1. Start
2. Store subjects and marks in a dictionary.
3. If dictionary is empty -> return 0.
4. Else:

```
Compute the sum of all marks using sum(grades.values()).

Count the number of subjects using len(grades).

Calculate the average as total / count.

Return the average.
```

5. Print average grade.
6. Stop

Result:

Program has been executed successfully and obtained the output.

Program

```
def is_valid(s):
    stack = []
    for char in s:
        if char == '(' or char == '{' or char == '[':
            stack.append(char)
        elif char == ')':
            if not stack or stack[-1] != '(':
                return False
            stack.pop()
        elif char == '}':
            if not stack or stack[-1] != '{':
                return False
            stack.pop()
        elif char == ']':
            if not stack or stack[-1] != '[':
                return False
            stack.pop()
        else:
            return False
    if len(stack) == 0:
        return True
    else:
        return False

while True:
    s = input("Enter the input string: ")
    b = is_valid(s)
    if b:
        print("Valid String")
    else:
        print("Invalid String")
```

Output

```
Enter the input string: ()
Valid String

Enter the input string: {[]}
Valid String

Enter the input string: ([)]
Invalid String

Enter the input string: {[(())]}
Valid String

Enter the input string: {[(])}
Invalid String
```

Experiment No.: 8 Date: _____

VALID PARENTHESIS

Aim:

To write a program that determines whether a given string consisting of parentheses '()', curly braces '{}', and square brackets '[]' is valid or not, ensuring that

(a) Every opening bracket has a corresponding closing bracket of the same type.

(b) Brackets are closed in the correct order.

(c) No unmatched or misordered brackets remain in the string.

Algorithm:

1. Start
2. Read the input string containing brackets.
3. Initialize an empty stack to store opening brackets.
4. Traverse each character in the string.
5. If the character is an opening bracket (, {, or [, push it onto the stack.
6. If the character is a closing bracket), }, or], check the top of the stack:
7. If the stack is empty or the top does not match the corresponding opening bracket, return False. Otherwise, pop the top element from the stack.
8. If any other character is encountered, return False.
9. After traversing the string, if the stack is empty, return True, otherwise, return False
10. Display the result as "Valid String" or "Invalid String".
11. Stop

Result:

Program has been executed successfully and obtained the output.

Program

```
fruits = ("apple", "banana", "mango", "orange", "grape")

user_input = input("Enter the name of a fruit: ").strip().lower()

if user_input in fruits:
 print(f"Yes, {user_input} is in the list of fruits.")
else:
 print(f"No, {user_input} is not in the list of fruits.")
```

Output

```
Enter the name of a fruit: Mango
Yes, mango is in the list of fruits.

============================================================
Enter the name of a fruit: Pineapple
No, pineapple is not in the list of fruits.
```

Experiment No.: 9 Date: _____

TUPLE ITEM CHECK

Aim:

To check whether a fruit entered by the user exists in the predefined tuple.

Algorithm:

1. Start
2. Create a tuple with different fruit names.
3. Prompt the user to input a fruit name.
4. Check if the entered fruit name is present in the tuple.
5. Print a message indicating whether it is present or not.
6. Stop

Result:

Program has been executed successfully and obtained the output.

Program

```
def check(s):
    vowels='aeiou'
    v=0
    c=0
    q=0
    words=s.split()
    for i in s:
        charlower=i.lower()
        if charlower in vowels:
            v+=1
        elif charlower.isalpha():
            c+=1
        elif charlower=='?':
            q+=1
    print("Count of vowels: ",v)
    print("Count of consonants: ",c)
    print("Number of words: ",len(words))
    print("Count of ?: ",q)
while True:
    s=input("Enter a string: ")
    if s=="":
        break
    check(s)
```

Output

```
Enter a string: What is your name ? My name is Sanju Samson.
Count of vowels:  13
Count of consonants:  20
Number of words:  10
Count of ?:  1
Enter a string:

=== Code Execution Successful ===
```

Experiment No.: 10 Date: _____

COUNT VOWELS, CONSONANTS, WORDS AND '?'

Aim:

To write a program that counts the number of vowels, consonants, words, and question marks in a given string.

Algorithm:

1. Start
2. Read the input string s.
3. If s is an empty string, then go to STOP.
4. Initialize the counters: Set v = 0 (vowels) Set c = 0 (consonants) Set q = 0 (question marks)
5. Define the set of vowels as {a, e, i, o, u}.
6. Split the string into words using whitespace and find the number of words.
7. Repeat for each character in the string:

```
-> Convert the character to lowercase.
-> If the character is a vowel, increase v by 1.
-> Else if the character is an alphabet, increase c by 1.
-> Else if the character is '?', increase q by 1.
-> Otherwise ignore the character.
```

8. Display the results:

```
-> Count of vowels
-> Count of consonants
-> Number of words
-> Number of question marks
```

9. Repeat from Step 2 if another string is to be processed.
10. Stop

Result:

Program has been executed successfully and obtained the output.

Program

```
import pandas as pd


data={
'Student':['Alice','Bob','Charlie','Daviid','Eva','Frank','Grace','Helen'],
'Class':['A','A','B','B','A','B','A','B'],
'Subject':['math','math','math','math','Science','Science','Science','Science'],
'Marks':[85,78,92,88,90,75,80,85]
}

df=pd.DataFrame(data)
print("Original Data:")
print(df)

pivot_table=pd.pivot_table(df,values='Marks',index='Class',columns='Subject',agg-
func='mean')
print("Pivot table(Average Marks per Class and Subject):")
print(pivot_table)

cross_tab=pd.crosstab(df['Class'],df['Subject'])

print("Cross Tabulation (Number of Students per clas and subject):")

print(cross_tab)
```

Output

```
Original Data:
 Student Class  Subject  Marks
0    Alice     A     math     85
1      Bob     A     math     78
2  Charlie     B     math     92
3   Daviid     B     math     88
4      Eva     A  Science     90
5    Frank     B  Science     75
6    Grace     A  Science     80
7    Helen     B  Science     85
Pivot table(Average Marks per Class and Subject):
Subject  Science  math
Class
A           85.0  81.5
B           80.0  90.0
Cross Tabulation (Number of Students per clas and subject):
Subject  Science  math
Class
A              2     2
B              2     2
```

Experiment No.: 11 Date: _____

PIVOT TABLES AND CROSS TABULATIONS

Aim:

To Write a Python program that demonstrates both Pivot Tables and Cross Tabulations using the pandas library.

Algorithm:

1. Start
2. Import Pandas Library
3. Create a sample dataset with students, their class, subject, and marks
4. Convert the dictionary into a pandas DataFrame
5. Display the original dataset
6. Create a pivot table using table() to calculate average marks per class and subject
7. Create a cross tabulation using crosstab() to count number of students per class and subject
8. Display the Result
9. Stop

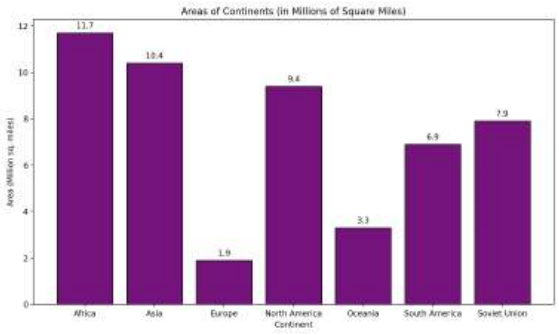
Result:

Program has been executed successfully and obtained the output.

Program

```
import matplotlib.pyplot as plt

continents = ['Africa', 'Asia', 'Europe', 'North America', 'Oceania', 'South America',
'Soviet Union']
areas = [11.7, 10.4, 1.9, 9.4, 3.3, 6.9, 7.9]

plt.figure(figsize=(10, 6))
plt.bar(continents, areas, color='purple', edgecolor='black')

plt.title('Areas of Continents (in Millions of Square Miles)')
plt.xlabel('Continent')
plt.ylabel('Area (Million sq. miles)')

for i, area in enumerate(areas):
 plt.text(i, area + 0.2, str(area), ha='center', fontsize=10)

plt.tight_layout()
plt.show()
```

Output



Experiment No.: 12 Date: _____

BAR CHART OF CONTINENTS

Aim:

To write a Python program that plots a bar chart showing the areas of different continents (in million square miles) using Matplotlib.

Algorithm:

1. Start
2. Import the matplotlib.pyplot library.
3. Define a list of continent names.
4. Define a list of corresponding continent areas (in million square miles).
5. Create a figure window with appropriate size

```
plt.figure(figsize=(10, 6))
```

6. Plot a bar chart with continents on the X-axis and areas on the Y-axis.

```
plt.bar(continents, areas, color='purple', edgecolor='black')
```

7. Set the title, xlabel and ylabel of the chart

```
plt.title('Areas of Continents (in Millions of Square Miles)')
plt.xlabel('Continent')
plt.ylabel('Area (Million sq. miles)')
```

8. Display the chart on the screen.

```
plt.show()
```

9. Stop

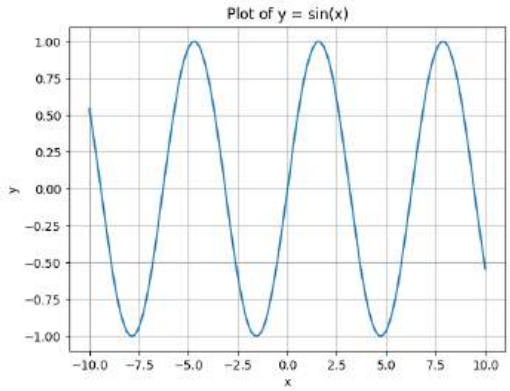
Result:

Program has been executed successfully and obtained the output.

Program

```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-10, 10, 200)
y = np.sin(x)

plt.plot(x, y)
plt.title("Plot of y = sin(x)")
plt.xlabel("x")
plt.ylabel("y")
plt.grid(True)
plt.show()
```

Output



Experiment No.: 13 Date: _____

PLOT A GRAPH Y = F(X)

Aim:

To write a program to plot the graph of y = f(x) and visualize the relationship between x and y.

Algorithm:

1. Start
2. Import the libraries numpy as np and matplotlib.pyplot as plt.
3. Generate x values using np.linspace from -10 to 10 with 200 points.
4. Calculate y for each x using the equation y = np.sin(x).
5. Plot y versus x using plt.plot(x, y).
6. Set the title of the graph to "Plot of y = sin(x)".
7. Label the x-axis as "x" and the y-axis as "y".
8. Display the graph using plt.show().
9. Stop

Result:

Program has been executed successfully and obtained the output.

Program

```
import numpy as np

A = np.arange(27).reshape(3,3,3)
print("Original 3D Array: \n",A)
diagonals = A.diagonal(axis1 = -2, axis2 = -1)
print("\nDiagonals of each 2D slice :\n",diagonals)
```

Output

```
Original 3D Array:
[[[ 0  1  2]
[ 3  4  5]
[ 6  7  8]]

[[ 9 10 11]
[12 13 14]
[15 16 17]]

[[18 19 20]
[21 22 23]
[24 25 26]]]

Diagonals of each 2D slice :
[[ 0  4  8]
[ 9 13 17]
[18 22 26]]
```

Experiment No.: 14 Date: _____

GET 2D DIAGONALS FROM 3D ARRAY

Aim:

To extract the diagonals of each 2D slice from a 3D NumPy array.

Algorithm:

1. Start
2. Import the NumPy library.
3. Create a 3D NumPy array using np.arange() and reshape it to dimensions (3, 3, 3).
4. Display the original 3D array.
5. Extract the diagonals from each 2D slice of the 3D array using the .diagonal() method with axis1=-2 and axis2=-1.
6. Display the diagonals of each 2D slice.
7. Stop

Result:

Program has been executed successfully and obtained the output.

Program

```
import numpy as np
arr=np.array([[1,2],[3,4]])
print("Original 2D array")
print(arr)
flat=arr.flatten()
print("Flatten 1D array")
print(flat)
```

Output

```
Original 2D array
[[1,2]
[3,4]]
Flatten 1D array
[1,2,3,4]
```

Experiment No.: 15 Date: _____

FLATTEN 2D ARRAY TO 1D

Aim:

Flatten a 2D numpy array into 1D array.

Algorithm:

1. Start.
2. Import the numpy library.
3. Create a sample 2D Numpy array and display it.
4. Use the .flatten() method on 2D array. This method returns a new 1D array containing all the elements of original array.
5. Display the resulting flattened 1D array.
6. Stop.

Result:

Program has been executed successfully and obtained the output.

Program

```
import numpy as np

n = 10
phi = (1 + np.sqrt(5)) / 2
fib = np.round((phi**np.arange(n)  -  (-1/phi)**np.arange(n)) / np.sqrt(5))
print("Fibonacci Series:", fib.astype(int))
```

Output

```
Fibonacci Series: [0  1   1   2  3  5  8  13  21  34]
```

Experiment No.: 16 Date: _____

FIBONACCI SERIES USING BINET'S ARRAY

Aim:

To generate the Fibonacci series using Binet's formula with NumPy array operations for efficient computation.

Algorithm:

1. Start
2. Import the NumPy library.
3. Set n <- 10.
4. Compute the golden ratio: phi equals one plus the square root of five, divided by two.
5. Generate an array of indices from 0 to n-1: indices•np.arange(n)
6. Apply Binet's formula to compute Fibonacci values: Fibonacci is equal to phi raised to the power of indices minus negative one divided by phi raised to the power of indices, all divided by the square root of five
7. Round the values of fib to the nearest integer using np.round().
8. Convert the values of fib into integer type using astype(int).
9. Print "Fibonacci Series:" followed by the values of fib.
10. Stop

Result:

Program has been executed successfully and obtained the output.

Program

```
import numpy as np

A=np.array([[1,2],[3,4]])
print("Matrix A : ")
print(A)

A_inv=np.linalg.inv(A)
print("Inverse of A is :\n",A_inv)
```

Output

```
Matrix A :
[[1 2]
[3 4]]

Inverse of A is :
[[-2.    1. ]
[ 1.5   -0.5]]
```

Experiment No.: 17

Date: _____

INVERSE A MATRIX

Aim:

To create a program to find the inverse of a matrix using NumPy.

Algorithm:

1. Start
2. Import the numpy library.

```
import numpy as np
```

3. Create a square matrix using numpy.array
4. Find the inverse of the matrix using the function numpy.linalg.inv(matrix)

```
A_inv=np.linalg.inv(A)
```

5. Display the original matrix and its inverse.
6. Stop

Result:

Program has been executed successfully and obtained the output.

Program

```
import numpy as np
a=np.array([1,2,3])
b=np.array([4,5,6])
inner=np.inner(a,b)
out=np.outer(a,b)
cro=np.cross(a,b)
print("Vector A",a)
print("Vector B",b)
print("Inner Product:",inner)
print("Outer Product:\n",out)
print("Cross Product:",cro)
```

Output

```
Vector A [1 2 3]
Vector B [4 5 6]
Inner Product: 32
Outer Product:
 [[ 4  5  6]
 [ 8 10 12]
 [12 15 18]]
Cross Product: [-3  6 -3]
```

Experiment No.: 18 Date: _____

INNER, OUTER, AND CROSS PRODUCTS

Aim:

To find the inner product, outer product, and cross product of two vectors using NumPy.

Algorithm:

1. Start
2. Import the numpy library as np.
3. Define two 1-D arrays (vectors) a and b using np.array().
4. Compute the inner product using np.inner(a, b).
5. Compute the outer product using np.outer(a, b).
6. Compute the cross product using np.cross(a, b).
7. Print the original vectors and the computed results.
8. Stop

Result:

Program has been executed successfully and obtained the output.

Program

```
import numpy as np

a = np.array([[1,2],[3,4]])
b = np.array([[0,5],[6,7]])

kron = np.kron(a,b)
print("Kronecker product:\n",kron)
```

Output

```
Kronecker product:
[[ 0  5  0 10]
 [ 6  7 12 14]
 [ 0 15  0 20]
 [18 21 24 28]]
```

Experiment No.: 19 Date: _____

KRONECKER PRODUCT OF ARRAYS

Aim:

To find the Kronecker product of two arrays using numpy.

Algorithm:

1. Start
2. Import the numpy library.
3. Define two matrices, a and b.
4. Compute the Kronecker product of a and b using np.kron(a,b).
5. Store the result in kron and print it.
6. Stop

Result:

Program has been executed successfully and obtained the output.

Program

```
import numpy as np

rows = int(input("Enter number of rows: "))
cols = int(input("Enter number of columns: "))

print(f"Enter {rows * cols} elements separated by spaces:")
elements = list(map(int, input().split()))

matrix = np.array(elements).reshape(rows, cols)

print("Matrix:")
print(matrix)

flat_list = matrix.flatten().tolist()

print("Flattened list:", flat_list)
```

Output

```
Enter number of rows: 2
Enter number of columns: 3
Enter 6 elements separated by spaces:
1 2 3 4 5 6
Matrix:
[[1 2 3]
[4 5 6]]
Flattened list: [1, 2, 3, 4, 5, 6]
```

Experiment No.: 20 Date: _____

CONVERT MATRIX TO LIST

Aim:

To convert a matrix into a list by arranging all its elements.

Algorithm:

1. Start
2. Import the Numpy library.
3. Ask the user to input the number of rows and columns.
4. Prompt the user to enter rows × cols elements separated by spaces.
5. Convert elements into a NumPy array and reshape it to (rows, cols) and call this matrix.
6. Print the matrix
7. Use np.flatten() method on the matrix to convert it into a 1D array.
8. Convert the 1D NumPy array to a Python list and store it in flat_list .
9. Print flat_list .
10. Stop.

Result:

Program has been executed successfully and obtained the output.

Program

```
import numpy as np

rows = int(input("Enter number of rows: "))
cols = int(input("Enter number of columns: "))

print(
 f"Enter the matrix values row by row (each row should have {cols} space-separated
numbers):")

A = []
for i in range(rows):
 row = list(map(float, input(f"Row {i + 1}: ").split()))
 if len(row) != cols:
 raise ValueError(f"Each row must have exactly {cols} values.")

 A.append(row)

A = np.array(A)

Q, R = np.linalg.qr(A)


print("Matrix Q(orthogonal):")

print(Q)

print("\nMatrix R(upper triangular):")

print(R)

print("\nCheck if Q @ R equals A:", np.allclose(A, Q @ R))
```

Output

```
Enter number of rows: 3
Enter number of columns: 3
Enter the matrix values row by row (each row should have 3 space-separated numbers):
Row 1: 12 -51 4
Row 2: 6 167 -68
Row 3: -4 24 -24
Matrix Q(orthogonal):
[[-0.85714286  0.39428571  0.33142857]
[-0.42857143 -0.90285714 -0.03428571]
[ 0.28571429 -0.17142857  0.94285714]]

Matrix R(upper triangular):
[[ -14.          -21.           18.85714286]
[   0.         -175.           67.08571429]
[   0.            0.          -18.97142857]]

Check if Q @ R equals A: True
```

Experiment No.: 21 Date: _____

QR DECOMPOSITION OF MATRIX

Aim:

To perform the QR decomposition of a given matrix.

Algorithm:

1. Start
2. Input matrix dimensions

```
Read number of rows rows

Read number of columns cols
```

3. Read the matrix

```
For each row i from 1 to rows:

Accept cols space-separated values

Store the values in a list

Combine all rows to form matrix A
```

4. Perform QR decomposition

```
Use NumPy's function: Q, R = np.linalg.qr(A)

Here,

Q = orthogonal matrix

R = upper triangular matrix
```

5. Display results

```
Print matrix Q

Print matrix R
```

6. Verification

```
Compute Q @ R

Compare with original matrix A using np.allclose(A, Q @ R)

Print whether the decomposition is correct.
```

7. Stop

Result:

Program has been executed successfully and obtained the output.

Program

```
import numpy as np

def get_matrix():
 n=int(input("Enter the size of the square matrix(n x n):"))
 print(f"Enter the elements row wise:")

 matrix=[]
 for i in range(n):
 row=list(map(float,input(f"Row {i+1}:").split()))
 if len(row)!=n:
 raise ValueError("Each row must have exaclty n elements")
 matrix.append(row)

 return np.array(matrix)

A = get_matrix()

eigenvalues, eigenvectors = np.linalg.eig(A)

print("\nMatrix A:", A)
print("\nEigenvalues:\n", eigenvalues)
print("\nEigenvectors (column-wise):\n", eigenvectors)

for i in range(len(eigenvalues)):
 Av = np.dot(A, eigenvectors[:, i])
 lv = eigenvalues[i] * eigenvectors[:, i]
 print(f"\nCheck for eigenvalue {eigenvalues[i]}:")
 print("A @ v =", Av)
 print("»* v =", lv)
```

Output

```
Enter the size of the square matrix(n x n):2
Enter the elements row wise:
Row 1:4 1
Row 2:2 1

Matrix A: [[4. 1.]
[2. 1.]]

Eigenvalues:
[4.56155281 0.43844719]

Eigenvectors (column-wise):
[[ 0.87192821 -0.27032301]
[ 0.48963374  0.96276969]]

Check for eigenvalue 4.561552812808831:
A @ v = [3.97734658 2.23349016]
»* v = [3.97734658 2.23349016]

Check for eigenvalue 0.4384471871911697:
A @ v = [-0.11852236  0.42212366]
»* v = [-0.11852236  0.42212366]
```

Experiment No.: 22 Date: _____

EIGENVALUES AND EIGENVECTORS

Aim:

To develop a Python program that accepts a square matrix as input, computes its eigenvalues and eigenvectors using NumPy, and verifies the eigenvalue–eigenvector relationship .

Algorithm:

1. Start
2. Ask the user to enter the size n of the square matrix.
3. Initialize an empty list to store the matrix.
4. For each row from 1 to n:

```
Take the row elements as input.

Check if the number of elements is equal to n.

If not, show an error.

Otherwise, add the row to the matrix.
```

5. Convert the list of rows into a NumPy array.
6. Use NumPy's linalg.eig() function to compute eigenvalues and eigenvectors of the matrix.
7. Display the matrix, eigenvalues, and eigenvectors.
8. For each eigenvalue and its corresponding eigenvector:

```
Multiply the matrix with the eigenvector (A @ v).

Multiply the eigenvalue with the eigenvector (»* v).

Display both results to verify that they are the same.
```

9. Stop

Result:

Program has been executed successfully and obtained the output.

Program

```
import numpy as np


arr = np.array(list(map(int, input("Enter the elements of the array (space separated):
").split())))
n = int(input("Enter the order of difference (n): "))
diff_arr = np.diff(arr, n=n)

print("\nOriginal Array:", arr)
print(f"{n}-th Order Difference:", diff_arr)
```

Output

```
Enter the elements of the array (space separated): 4 14 7 34 19
Enter the order of difference (n): 2

Original Array: [ 4 14  7 34 19]
2-th Order Difference: [-17  34 -42]
```

Experiment No.: 23 Date: _____

N-TH ORDER DISCRETE DIFFERENCE

Aim:

To create a program that takes an array input from the user along with an integer n and computes the n-th order discrete difference of the array.

Algorithm:

1. Start
2. Prompt the user to enter the elements of the array (space separated)
3. Read the input and convert it into a NumPy array
4. Prompt the user to enter the order of difference (n)
5. Use np.diff() function with parameter n to calculate the n-th order discrete difference
6. Print the original array and the resulting difference array
7. Stop

Result:

Program has been executed successfully and obtained the output.

Program

```
import numpy as np

# Function to take user input for a matrix
def get_matrix(name):
    rows = int(input(f"Enter number of rows for {name}: "))
    cols = int(input(f"Enter number of columns for {name}: "))
    print(f"Enter elements of {name} row-wise (space separated):")

    matrix = []
    for i in range(rows):
        row = list(map(float, input(f"Row {i+1}: ").split()))
        if len(row) != cols:
            raise ValueError("Each row must have exactly the specified num-
ber of columns.")
        matrix.append(row)
    return np.array(matrix)

# Input two arrays
A = get_matrix("Array A")
B = get_matrix("Array B")

# Input Einstein summation expression
expr = input("Enter Einstein summation expression (e.g. 'ij,jk->ik'): ")

# Compute Einstein summation
result = np.einsum(expr, A, B)

print("\nArray A:\n", A)
print("\nArray B:\n", B)
print(f"\nEinstein Summation Result for '{expr}':\n", result)
```

Output

```
Enter number of rows for Array A: 2
Enter number of columns for Array A: 2
Enter elements of Array A row-wise (space separated):
Row 1: 1 2
Row 2: 3 4

Enter number of rows for Array B: 2
Enter number of columns for Array B: 2
Enter elements of Array B row-wise (space separated):
Row 1: 5 6
Row 2: 7 8

Enter Einstein summation expression (e.g. 'ij,jk->ik'): ij,jk->ik

Array A:
 [[1. 2.]
  [3. 4.]]

Array B:
 [[5. 6.]
  [7. 8.]]

Einstein Summation Result for 'ij,jk->ik':
 [[19. 22.]
  [43. 50.]]
```

Experiment No.: 24 Date: _____

EINSTEIN'S SUMMATION CONVENTION

Aim:

To evaluate the Einstein summation convention on two multidimensional NumPy arrays using Python.

Algorithm:

1. Start
2. Read the dimensions (number of rows and columns) for the first multidimensional array from the user.
3. Read the elements of the first array row-wise from the user and store them in a NumPy array.
4. Read the dimensions for the second multidimensional array from the user.
5. Read the elements of the second array row-wise from the user and store them in a NumPy array.
6. Ask the user to enter a valid Einstein summation expression (for example, 'ij,jk->ik').
7. Compute the Einstein summation of the two arrays using np.einsum() with the given expression.
8. Display the first array, second array, and the resultant array.
9. Stop

Result:

Program has been executed successfully and obtained the output.

Program

```
import numpy as np

x=np.array(list(map(float,input("Enter the elements of the first array:").split())))

y=np.array(list(map(float,input("Enter the elements of the second array:").split())))

if len(x)!=len(y):
 raise ValueError("Both arrays must have same number of elements")
corr_matrix=np.corrcoef(x,y)

print("First Array:",x)

print("Second Array:",y)

print("Correlation Matrix:",corr_matrix)

print("Pearson Correlation Coefficient:",corr_matrix[0,1])
```

Output

```
Enter the elements of the first array: 10 20 30 40 50
Enter the elements of the second array: 5 15 25 35 45

First Array: [10. 20. 30. 40. 50.]
Second Array: [ 5. 15. 25. 35. 45.]
Correlation Matrix: [[1. 1.]
[1. 1.]]
Pearson Correlation Coefficient: 1.0
```

Experiment No.: 25 Date: _____

PEARSON CORRELATION COEFFICIENTS

Aim:

To write a Python program using NumPy that accepts two arrays of equal length from the user and computes the correlation matrix and the Pearson correlation coefficient between the arrays.

Algorithm:

1. Start
2. Import the NumPy library.
3. Take user input for the first array:

```
x=np.array(list(map(float,input("Enter the elements of the first array:").split())))
```

4. Take user input for the second array:

```
 y=np.array(list(map(float,input("Enter the elements of the second ar-
ray:").split())))
```

5. Check if arrays are of the same length:

```
If len(x) != len(y), raise error '"Both arrays must have the same number of elements."
```

6. Compute the Pearson correlation coefficient matrix using NumPy
7. Extract the Pearson correlation coefficient
8. Display outputs:

```
Print x
Print y
Print correlation matrix
Print Pearson correlation coefficient
```

9. Stop

Result:

Program has been executed successfully and obtained the output.

Program

```
import numpy as np

data=list(map(float,input("Enter numbers of the array seperated by space:\n").split()))
arr=np.array(data)

mean=np.mean(arr)
var=np.var(arr)
std_dev=np.std(arr)
print(f" \n Input Data: {arr}")
print(f"\nAverage : {mean}")
print(f"\nVariance : {var}")
print(f"\nStandard Deviation: {std_dev}")
```

Output

```
Enter numbers of the array seperated by space:
132421 324235 436261 575421 345347

 Input Data: [132421. 324235. 436261. 575421. 345347.]

Average : 362737.0

Variance : 21094107678.4

Standard Deviation: 145238.10683976847
```

Experiment No.: 26 Date: _____

AVERAGE, VARIANCE, AND STANDARD DEVIATION

Aim:

To analyze a set of numbers provided by the user and compute key statistical measures: mean, variance, and standard deviation.

Algorithm:

1. Start
2. The user's input is taken and converted into a list of numbers. A NumPy array is created from this list.
3. The program uses the array to calculate the mean, variance, and standard deviation.
4. The original data and the calculated results are prepared for display.
5. Stop

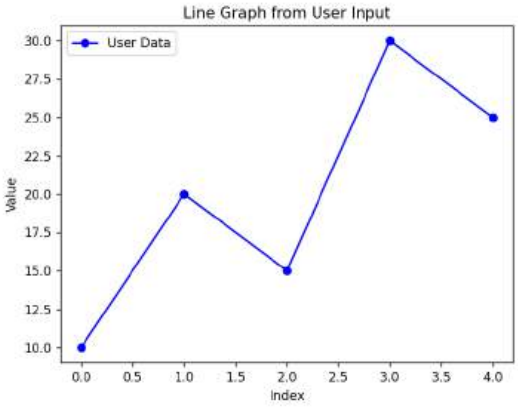
Result:

Program has been executed successfully and obtained the output.

Program

```
import numpy as np
 import matplotlib.pyplot as plt
 data = input("Enter numbers separated by spaces: ")
 y = np.array(list(map(float, data.split())))
 x = np.arange(len(y))
 plt.plot(x, y, marker='o', linestyle='-', color='b', label='User Data')
 plt.xlabel("Index")
 plt.ylabel("Value")
 plt.title("Line Graph from User Input")
 plt.legend()
 plt.show()
```

Output




```
 Enter numbers separated by spaces: 10 20 15 30 25
```

Experiment No.: 27 Date: _____

LINE GRAPH

Aim:

To write a program to plot a line graph

Algorithm:

1. Start
2. Import numpy for numerical operations and matplotlib.pyplot for plotting.
3. Prompt the user to enter numbers separated by spaces as input.
4. Split the input string and convert it into a NumPy array of floating-point numbers.
5. Generate a sequence of index values for the x-axis using np.arange() based on the number of inputs.
6. Plot the index values against the user's data with markers, line style, color, and a label.
7. Assign the label "index" to the x-axis for clarity.
8. Assign the label "Value" to the y-axis and set the title as "Line Graph from User Input".
9. Display a legend to indicate the plotted user data.
10. Stop

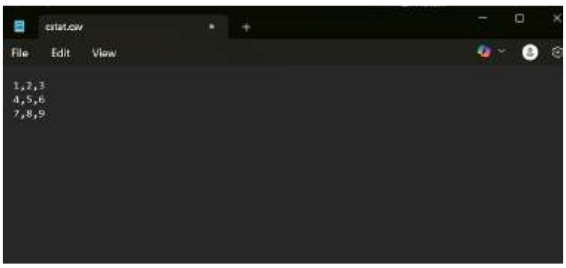
Result:

Program has been executed successfully and obtained the output.

Program

```
import numpy as np

data=np.array([[1, 2, 3],
[4, 5, 6],
[7, 8, 9]])

np.savetxt("C:\\Users\\shaun\\Desktop\\mec\\cycle4\\cstat.csv",
 data,
 delimiter=",",
 fmt="%d")
```

Output



Experiment No.: 28 Date: _____

CONVERT NUMPY ARRAY TO CSV

Aim:

To create a program that converts a NumPy array into a CSV file for storing and accessing data easily.

Algorithm:

1. Start
2. Create a NumPy array.

```
Import numpy as np
Define a 2D array using np.array()
```

3. Save the array into a CSV file.

```
Use NumPy's function:
np.savetxt("file_path.csv", data, delimiter=",", fmt="%d")

Here,
"file_path.csv" = path and name of the CSV file
data = NumPy array
delimiter="," = separates values with commas
fmt="%d" = format specifier for integers
```

4. Stop

Result:

Program has been executed successfully and obtained the output.

Program

```
import string
def charCount(s):
numWords=0
numSentences=0
numUpper=0
numLower=0
numSpecial=0
try:
with open(file_path, 'r', encoding='utf-8')as file:
 text=file.read()
 words=text.split()
 numWords=len(words)

 sentence_endings={'.','!','?'}
 numSentences=sum(1 for char in text if char in sentence_endings)


 for char in text:
 if char.isupper():
 numUpper+=1
 elif char.islower():
 numLower+=1
 elif char in string.punctuation or not char.isalnum() and not char.isspace():
 numSpecial+=1
print(f"Number of words is :{numWords}")
print(f"Number of sentences is : {numSentences}")
print(f"Number of Upper case letters is : {numUpper}")
print(f"Number of Lower case letters is : {numLower}")
print(f"Number of Special characters is : {numSpecial}")
except FileNotFoundError:
print(f"file '{file_path}' not found.")
except Exception as e:
print(f"An error occurred : {e}")


file_path='sampletext.txt'
charCount(file_path)
```

Output

```
Text file : Hello how are you? Hope you are doing well !
Number of words is : 9
Number of sentences is : 2
Number of upper case letters is : 2
Number of lower case letters is : 31
Number of special characters is : 2
```

Experiment No.: 29 Date: _____

COUNT WORDS, SENTENCES AND CHARACTERS IN A FILE

Aim:

To calculate the number of words, sentences and characters in a given text fle.

Algorithm:

1. Start
2. Initialize counters

```
numWords = 0
numSentences = 0
numUpper = 0
numLower = 0
numSpecial = 0
```

3. Attempt to open the text file

```
try:
 with open(file_path, 'r', encoding='utf-8') as file:
```

4. Read the fle content

```
text = file.read()
```

5. Count the number of words

```
words = text.split()
numWords = len(words)
```

6. Count the number of sentences

```
sentence_endings = {'.', '!', '?'}
numSentences = sum(1 for char in text if char in sentence_endings)
```

7. Count the number of upper case, lower case and special characters

```

```

```
for char in text:
 if char.isupper():
 numUpper += 1
 elif char.islower():
 numLower += 1
 elif char in string.punctuation or not char.isalnum() and not char.isspace():
 numSpecial += 1
```

8. Display the results

```
print(f"Number of words is : {numWords}")
print(f"Number of sentences is : {numSentences}")
print(f"Number of Upper case letters is : {numUpper}")
print(f"Number of Lower case letters is : {numLower}")
print(f"Number of Special characters is : {numSpecial}")
```

9. Handle the errors

```
except FileNotFoundError:
 print(f"file '{file_path}' not found.")
except Exception as e:
 print(f"An error occurred : {e}")
```

10. Stop

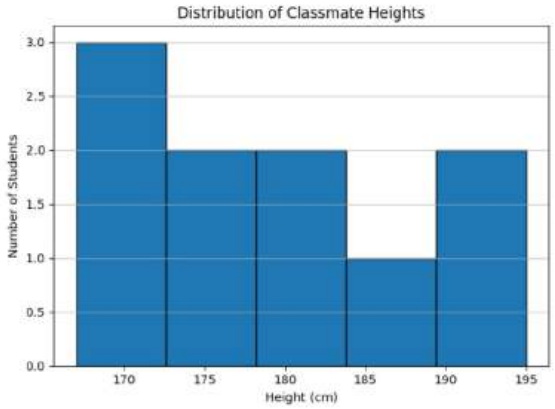
Result:

Program has been executed successfully and obtained the output.

Program

```
import matplotlib.pyplot as plt

heights_input = input("Enter the heights (in cm) separated by spaces: ")

heights = list(map(float, heights_input.split()))

plt.hist(heights, bins=5, edgecolor='black')
plt.title('Distribution of Classmate Heights')
plt.xlabel('Height (cm)')
plt.ylabel('Number of Students')
plt.grid(axis='y', alpha=0.75)
plt.show()
```

Output




```
Enter the heights (in cm) separated by spaces:

167 177 170 187 175 169 180 195 190 182
```

Experiment No.: 30 Date: _____

HISTOGRAM

Aim:

To measure your classmates' height and draw the histogram.

Algorithm:

1. Start
2. import matplotlib
3. Display "Enter the heights (in cm) separated by spaces:"
4. Read the full line of input as a string in 'heights_input'
5. Split the string by spaces and convert them to float and store in 'heights'
6. Use 'hist' function to plot the histogram

```
plt.hist(heights, bins=5, edgecolor='black')
plt.title('Distribution of Classmate Heights')
plt.xlabel('Height (cm)')
plt.ylabel('Number of Students')
plt.grid(axis='y', alpha=0.75)
plt.show()
```

7. Stop

Result:

Program has been executed successfully and obtained the output.

Program

```
import numpy as np

population = [4779736, 710231, 6392017, 2915918, 37253956, 5029196, 3574097, 897934]

mean_population = np.mean(population)
median_population = np.median(population)
variance_population = np.var(population)

print(f"Mean Population: {mean_population: ,.2f}")
print(f"Median Population: {median_population: ,.2f}")
print(f"Variance of Population: {variance_population: ,.2f}")
```

Output

```
Mean Population: 7694135.62
Median Population: 4176916.50
Variance of Population: 128230435522129.22
```

Experiment No.: 31 Date: _____

MEAN, MEDIAN, AND VARIANCE

Aim:

Compute the mean, median, and variance for the population from the given table.

Algorithm:

1. Start
2. Import the numpy library.
3. Initialize a list named population with the given numerical data.
4. Calculate the mean of the population list using the numpy.mean() function and store it in mean_population.
5. Calculate the median of the population list using the numpy.median() function and store it in median_population.
6. Calculate the variance of the population list using the numpy.var() function and store it in variance_population.
7. Print the calculated mean, median, and variance with appropriate labels.
8. Stop

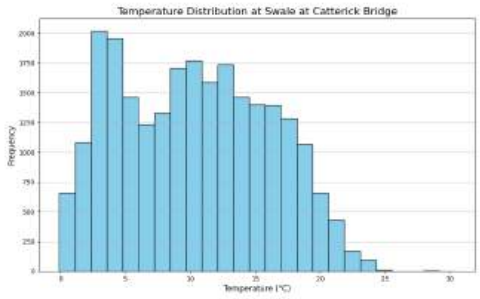
Result:

Program has been executed successfully and obtained the output.

Program

```
import pandas as pd
import matplotlib.pyplot as plt

file_name = 'lois_continuous.csv'

try:
 data = pd.read_csv(file_name, skiprows=1)
except FileNotFoundError:
 print(f"Error: The file '{file_name}' was not found.")
 print("Please make sure the script and the CSV file are in the same directory.")
 exit()

catterick_bridge_data = data[data['SITE_NAME'] == 'Swale at Catterick Bridge'].copy()

catterick_bridge_data['temperature'] = pd.to_numeric(catterick_bridge_data['0476
(Cel)'], errors='coerce')
catterick_bridge_data['dissolved_oxygen'] = pd.to_numeric(catterick_bridge_data['0474
(% satn)'], errors='coerce')

mean_temperature = catterick_bridge_data['temperature'].mean()
median_dissolved_oxygen = catterick_bridge_data['dissolved_oxygen'].median()

print(f"Analysis for 'Swale at Catterick Bridge':")
print(f"Mean Temperature: {mean_temperature:.2f} °C")
print(f"Median Dissolved Oxygen: {median_dissolved_oxygen:.2f} % saturation")

plt.figure(figsize=(12, 7))
plt.hist(catterick_bridge_data['temperature'].dropna(), bins=25, color='skyblue', edge-
color='black')

plt.title('Temperature Distribution at Swale at Catterick Bridge', fontsize=16)
plt.xlabel('Temperature (°C)', fontsize=12)
plt.ylabel('Frequency', fontsize=12)

plt.grid(axis='y', alpha=0.75)

plt.savefig('temperature_histogram.png')
print("\nHistogram plot saved as 'temperature_histogram.png'")
plt.show()
```

Output




```
Analysis for 'Swale at Catterick Bridge':
Mean Temperature: 10.28 °C
Median Dissolved Oxygen: 96.70 % saturation

Histogram plot saved as 'temperature_histogram.png'
```

Experiment No.: 32 Date: _____

ANALYZE RIVER TEMPERATURE DATA

Aim:

This program analyzes river quality data for the "Swale at Catterick Bridge" location to calculate the mean temperature and median dissolved oxygen, and then creates a histogram to visualize the temperature's distribution.

Algorithm:

1. Start
2. Read the data from the file lois_continuous.csv, making sure to skip the first header row
3. Check if the file was read successfully. If not, print an error and stop
4. Filter the data to create a new list containing only the records from the "Swale at Catterick Bridge" location
5. Convert the temperature and dissolved oxygen data in this new list into a numeric format that can be used for math
6. Calculate the average (mean) of all the temperature values
7. Calculate the middle value (median) of all the dissolved oxygen values
8. Print the final calculated mean temperature and median dissolved oxygen
9. Generate a histogram graph from all the temperature values
10. Save the graph as an image file named temperature_histogram.png
11. Display the graph on the screen
12. Stop

Result:

Program has been executed successfully and obtained the output.

Program

```
import pandas as pd
import matplotlib.pyplot as plt

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

plt.figure(figsize=(8,5))
plt.plot(df['month_number'], df['total_profit'], label="Total Profit")
plt.title("Total Profit per Month")
plt.xlabel("Month")
plt.ylabel("Profit")
plt.legend()
plt.show()

plt.figure(figsize=(8,5))
plt.plot(df['month_number'], df['total_profit'], color='red', linestyle='--', mark-
er='o', linewidth=3, label="Profit")
plt.title("Styled Profit Line Plot")
plt.xlabel("Month")
plt.ylabel("Profit")
plt.legend()
plt.show()

plt.figure(figsize=(10,6))
for col in df.columns[1:-1]:
    plt.plot(df['month_number'], df[col], label=col)
plt.title("Monthly Sales Data of Products")
plt.xlabel("Month")
plt.ylabel("Sales")
plt.legend()
plt.show()

plt.figure(figsize=(8,5))
plt.scatter(df['month_number'], df['toothpaste'], color='green', marker='o')
plt.title("Toothpaste Sales per Month")
plt.xlabel("Month")
plt.ylabel("Toothpaste Units")
plt.show()

plt.figure(figsize=(8,5))
plt.bar(df['month_number']-0.2, df['facecream'], width=0.4, label="Face Cream")
plt.bar(df['month_number']+0.2, df['facewash'], width=0.4, label="Face Wash")
plt.title("Face Cream & Face Wash Sales")
plt.xlabel("Month")
plt.ylabel("Sales")
plt.legend()
plt.show()

plt.figure(figsize=(8,5))
plt.bar(df['month_number'], df['bathingsoap'], color='blue')
plt.title("Bathing Soap Sales")
plt.xlabel("Month")
plt.ylabel("Sales")
plt.savefig("bathingsoap_sales.png")
plt.show()

plt.figure(figsize=(8,5))
plt.hist(df['total_profit'], bins=5, color='purple', edgecolor='black')
plt.title("Profit Ranges Histogram")
plt.xlabel("Profit Range")
plt.ylabel("Frequency")
plt.show()

product_totals = df.drop(columns=['month_number','total_profit']).sum()
plt.figure(figsize=(8,8))
plt.pie(product_totals, labels=product_totals.index, autopct='%1.1f%%')
```

Experiment No.: 33 Date: _____

ANALYZE COMPANY SALES DATA

Aim:

To read the company sales dataset and perform data visualization using Matplotlib by plotting different types of charts such as line plot, scatter plot, bar chart, histogram, pie chart, subplots, and stack plot to analyze monthly and yearly product sales trends

Algorithm:

1. Start
2. Import the required libraries pandas and matplotlib.
3. Load the dataset company_sales_data.csv into a DataFrame
4. Plot total profit per month using a line plot

```
plot(x = df['month_number'], y = df['total_profit'])
show()
```

5. Plot a styled line graph for total profit with color, marker, and line style

```
plot(x = df['month_number'], y = df['total_profit'],
 color='red', linestyle='--', marker='o', linewidth=3)
show()
```

6. Plot multiple line graphs for sales of all products

```
for each product_column in df (excluding month_number, total_profit):
 plot(df['month_number'], df[product_column])
show()
```

7. Plot toothpaste sales using a scatter plot

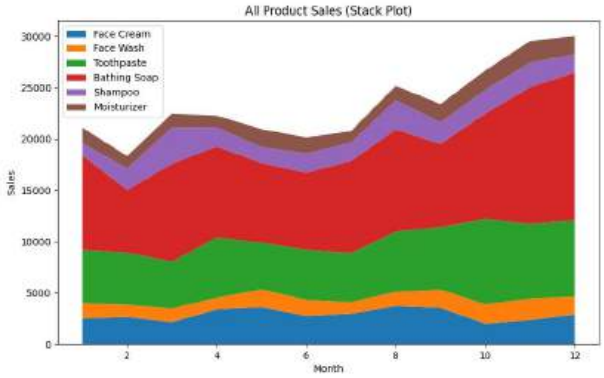
```
scatter(x = df['month_number'], y = df['toothpaste'], color='green')
show()
```

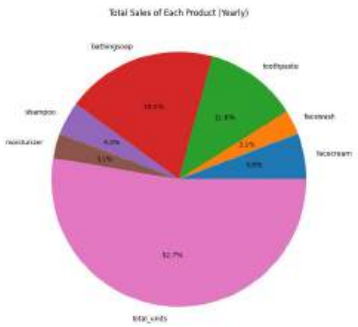
8. Plot bar charts comparing face cream and face wash sales

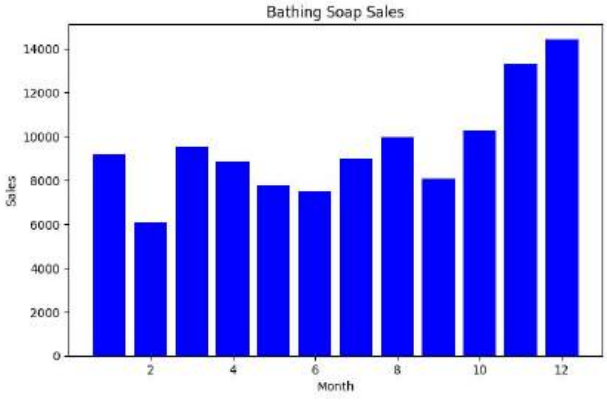
```

```

```
plt.title("Total Sales of Each Product (Yearly)")
plt.show()

plt.figure(figsize=(10,5))
plt.subplot(1,2,1)
plt.plot(df['month_number'], df['bathingsoap'], marker='o', color='blue')
plt.title("Bathing Soap Sales")
plt.subplot(1,2,2)
plt.plot(df['month_number'], df['facewash'], marker='o', color='orange')
plt.title("Face Wash Sales")
plt.show()

plt.figure(figsize=(10,6))
plt.stackplot(df['month_number'],
 df['facecream'], df['facewash'], df['toothpaste'],
 df['bathingsoap'], df['shampoo'], df['moisturizer'],
 labels=['Face Cream','Face Wash','Toothpaste','Bathing Soap','Shampoo','Mois-
turizer'])
plt.title("All Product Sales (Stack Plot)")
plt.xlabel("Month")
plt.ylabel("Sales")
plt.legend(loc='upper left')
plt.show()
```

Output







```
bar(x = df['month_number']-0.2, y = df['facecream'])
bar(x = df['month_number']+0.2, y = df['facewash'])
show()
```

9. Plot bathing soap sales using a bar chart and save the figure

```
bar(x = df['month_number'], y = df['bathingsoap'])
save("bathingsoap_sales.png")
show()
```

10. Plot a histogram of total profit

```
hist(df['total_profit'], bins=5)
show()
```

11. Plot a pie chart of yearly sales of each product

```
product_totals •sum of sales per product
pie(product_totals)
show()
```

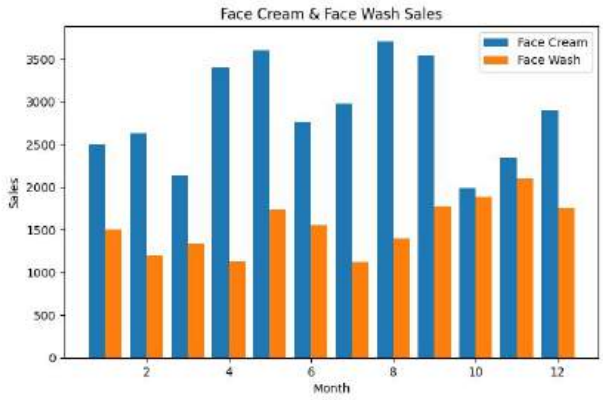
12. Create subplots for bathing soap and face wash sales

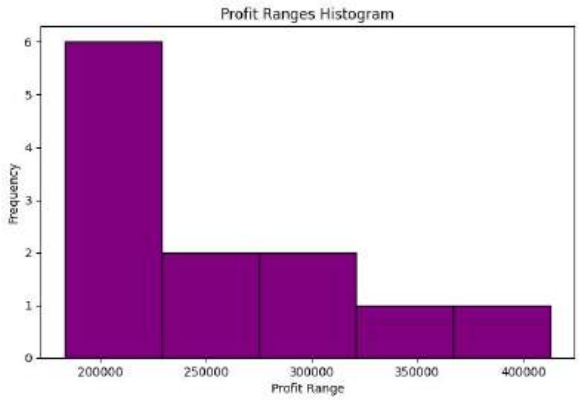
```
subplot(1,2,1) ’plot(df['month_number'], df['bathingsoap'])
subplot(1,2,2) ’plot(df['month_number'], df['facewash'])
show()
```

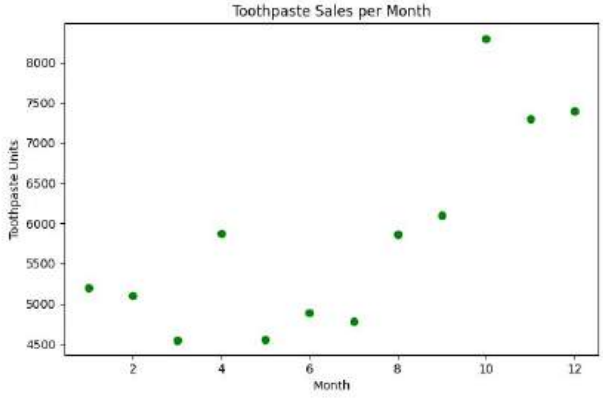
13. Plot a stack plot for all product sales across months

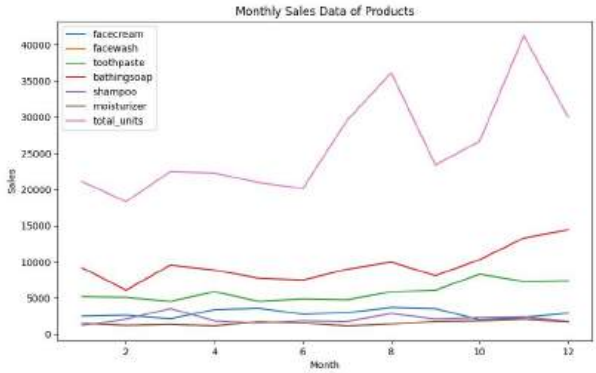
```
stackplot(df['month_number'],
 df['facecream'], df['facewash'], df['toothpaste'],
 df['bathingsoap'], df['shampoo'], df['moisturizer'])
show()
```

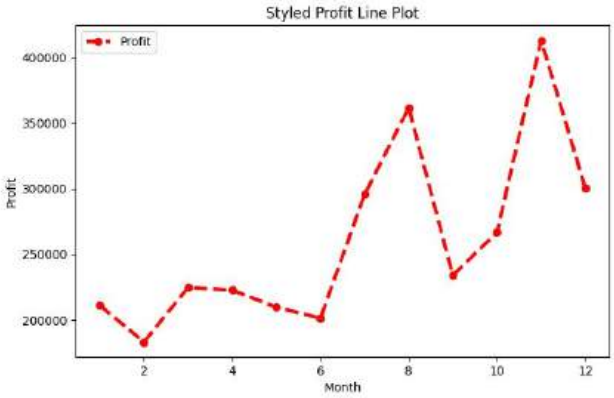
14. Stop


Face Cream & Face Wash Sales
Face Cream
Face Wash
Sales
Month


Profit Ranges Histogram
Frequency
Profit Range


Toothpaste Sales per Month
Toothpaste Units
Month


Monthly Sales Data of Products
facecream
facewash
toothpaste
bathingsoap
shampoo
moisturizer
total_units
Sales
Month


Styled Profit Line Plot
Profit
Month


Total Profit per Month
Total Profit
Profit
Month

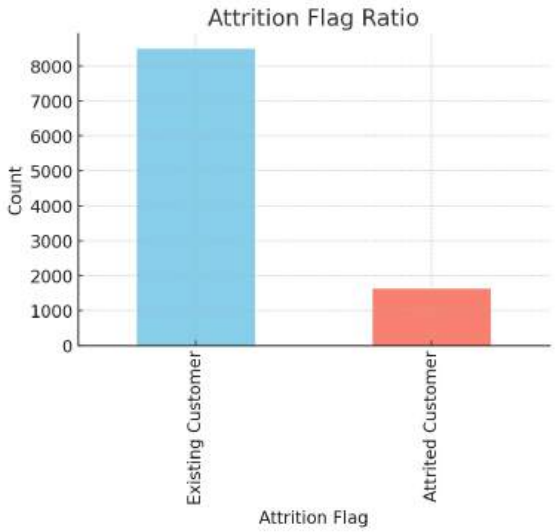
Result:

Program has been executed successfully and obtained the output.

Program

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

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

print(df.isnull().sum())

attrition_ratio = df['Attrition_Flag'].value_counts()
plt.figure(figsize=(6,4))
sns.barplot(x=attrition_ratio.index, y=attrition_ratio.values)
plt.title("Attrition Flag Ratio")
plt.xlabel("Attrition Flag")
plt.ylabel("Count")
plt.show()

categorical_cols = df.select_dtypes(include='object').columns
for col in categorical_cols:
    plt.figure(figsize=(6,4))
    sns.countplot(data=df, x=col)
    plt.title(f"Distribution of {col}")
    plt.xticks(rotation=45)
    plt.show()

print(df.describe())

plt.figure(figsize=(10,8))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()

df = df.drop(columns=[
    'CLIENTNUM',
    'Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Depen-
dent_count_Education_Level_Months_Inactive_12_mon_1',
    'Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Depen-
dent_count_Education_Level_Months_Inactive_12_mon_2'
])

print("Remaining columns:", df.columns)
```

Output



Experiment No.: 34 Date: _____

ANALYZE BANK CUSTOMER ATTRITION DATA

Aim:

To analyze a bank's customer dataset by handling missing values, visualizing attrition ratios, generating statistical summaries, and identifying correlations using Python.

Algorithm:

1. Start
2. Import required libraries

```
Import pandas for data handling.

Import matplotlib.pyplot and seaborn for visualization.
```

3. Load the dataset

```
Use pd.read_csv("BankChurners.csv") to load data into a DataFrame named df.
```

4. Check for missing values

```
Use df.isnull().sum() to display the number of missing values in each column.
```

5. Visualize the ratio of Attrition Flags

```
Count occurrences using value_counts() on the column Attrition_Flag.

Create a bar plot using seaborn.barplot() to visualize the distribution.
```

6. Plot categorical variable distributions

```
Identify categorical columns using df.select_dtypes(include='object').

For each column, plot a count plot (sns.countplot) to visualize the category fre-
quencies.
```

7. Display descriptive statistics

```
Use df.describe() to get the mean, standard deviation, minimum, maximum, and quar-
tile values of numerical features.
```

8. Visualize correlation between numeric features

```
Use df.corr(numeric_only=True) to compute correlations.

Plot a heatmap using sns.heatmap() to identify relationships among numerical vari-
ables.
```

9. Drop unnecessary columns

```
Remove irrelevant or redundant columns such as:

CLIENTNUM (Customer ID)

Long Naive Bayes columns (machine-generated attributes)

Use df.drop(columns=[...]) to perform this operation.
```

10. Print remaining columns

```
Display the names of columns remaining after cleaning using df.columns.
```

11. Stop

Result:

Program has been executed successfully and obtained the output.