**Lab Session #4**

Title: Matrices and Vectors using NumPy

Aim: To implement the basic operations on Matrices and Vectors

Problem Definition: Develop Python Programs to do the following:

1. Add, subtract, multiply and divide two vectors.
2. Compute the vector dot product.
3. Find the sum of values in a matrix.
4. Calculate the sum of the diagonal elements of a NumPy array
5. Add and Subtract Matrices in Python
6. Add row/columns in numpy array
7. Matrix Multiplication in NumPy
8. Inverse a matrix using NumPy
9. Count the frequency of unique values in NumPy array
10. Multiply matrices of complex numbers using NumPy in Python.
11. Program to find matrix Determinant, Trace, Eigenvalues, Eigenvectors, Singular value decomposition of a Matrix.

**Code:**

import numpy as np

def acc\_vector():

n = int(input("Enter size of vector: "))

print("Enter elements of vector: ")

inputs = list(map(int, input().split()))

arr = []

k = 0

for i in range(n):

elem = inputs[k]

arr.append(elem)

k += 1

arr = np.array(arr)

return arr

def acc\_matrix():

r, c = map(int, input("Enter rows and columns of matrix: ").split())

print("Enter elements of matrix: ")

inputs = list(map(int, input().split()))

arr = []

mat = []

k = 0

for i in range(r):

for j in range(c):

elem = inputs[k]

arr.append(elem)

k += 1

mat.append(arr)

arr = []

mat = np.array(mat)

return mat

def acc\_complex\_matrix():

r, c = map(int, input("Enter rows and columns of matrix: ").split())

print("Enter elements of matrix in a+bj format: ")

inputs = list(map(complex, input().split()))

arr = []

mat = []

k = 0

for i in range(r):

for j in range(c):

elem = inputs[k]

arr.append(elem)

k += 1

mat.append(arr)

arr = []

mat = np.array(mat)

return mat

while (1):

print("1:Add, subtract, multiply and divide two vectors.")

print("2:Compute the vector dot product.")

print("3:Find the sum of values in a matrix.")

print("4:Calculate the sum of the diagonal elements of a NumPy array.")

print("5:Add and Subtract Matrices in Python.")

print("6:Add row/columns in numpy array.")

print("7:Matrix Multiplication in NumPy.")

print("8:Inverse a matrix using NumPy.")

print("9:Count the frequency of unique values in NumPy array.")

print("10:Multiply matrices of complex numbers using NumPy in Python.")

print("11:Program to find matrix Determinant, Trace, Eigenvalues, Eigenvectors, Singular value decomposition of a Matrix.")

print("12:Exit.")

choice = int(input("Enter choice."))

match choice:

case 1:

a = acc\_vector()

b = acc\_vector()

print()

print("Sum ", np.add(a, b))

print("Difference ", np.subtract(a, b))

print("Multiplication ", np.multiply(a, b))

print("Division ", np.divide(a, b))

print()

case 2:

a = acc\_vector()

b = acc\_vector()

print()

print("Vector dot product: ", np.dot(a, b))

print()

case 3:

a = acc\_matrix()

print()

print("Sum of all values: ", np.sum(a))

print()

case 4:

a = acc\_matrix()

print()

print("Sum of diagonal elements: ", np.trace(a))

print()

case 5:

a = acc\_matrix()

b = acc\_matrix()

print()

print("Sum\n", np.add(a, b))

print("Difference\n", np.subtract(a, b))

print()

case 6:

a = acc\_matrix()

r = a.shape[0]

c = a.shape[1]

columns = []

temp = []

print("Enter the column to be added:")

inputs = list(map(int, input().split()))

for i in range(r):

temp.append(inputs[i])

columns.append(temp)

temp = []

columns = np.array(columns)

print("Enter the row to be added:")

row = list(map(int, input().split()))

row = np.array(row)

arrc = np.concatenate([a, columns], axis=1)

arrr = np.r\_[a, [row]]

print()

print("Column added\n", arrc)

print("Row added\n", arrr)

print()

case 7:

a = acc\_matrix()

b = acc\_matrix()

print()

print("Multiplied matrix:\n", np.matmul(a, b))

print()

case 8:

a = acc\_matrix()

print()

if (np.linalg.det(a) == 0):

print("Inverse matrix does not exist.")

else:

print("Inverse matrix\n", np.linalg.inv(a))

print()

case 9:

a = acc\_vector()

u, f = np.unique(a, return\_counts=True)

count = np.asarray((u, f))

print()

print("Unique values and frequency:\n", count)

print()

case 10:

a = acc\_complex\_matrix()

b = acc\_complex\_matrix()

print()

print("Multiplication is: ", np.vdot(a, b))

print()

case 11:

a = acc\_matrix()

print()

print("Determinant:", np.linalg.det(a), "\n")

print("Trace:", np.trace(a), "\n")

w, v = np.linalg.eig(a)

print("Eigenvalues:", w, "\n")

print("Eigenvectors:", v, "\n")

print("Singular Value Decomposition:\n")

u, s, vh = np.linalg.svd(a)

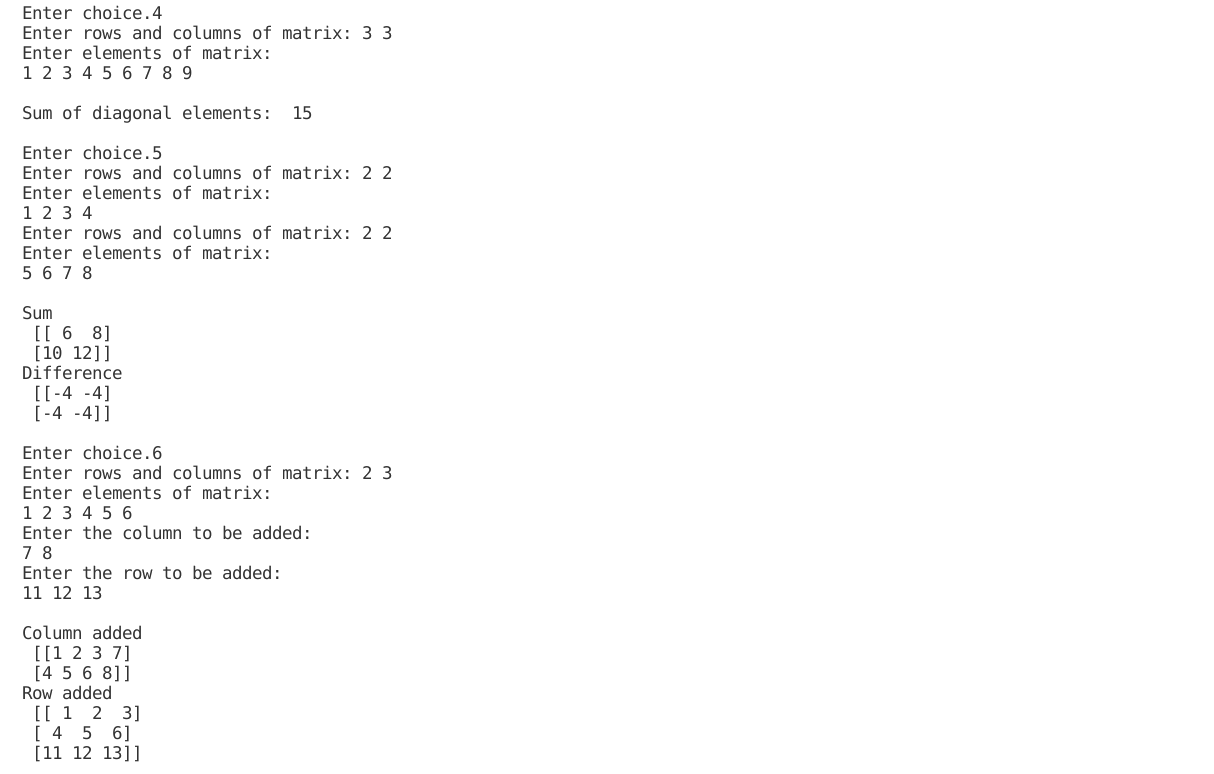
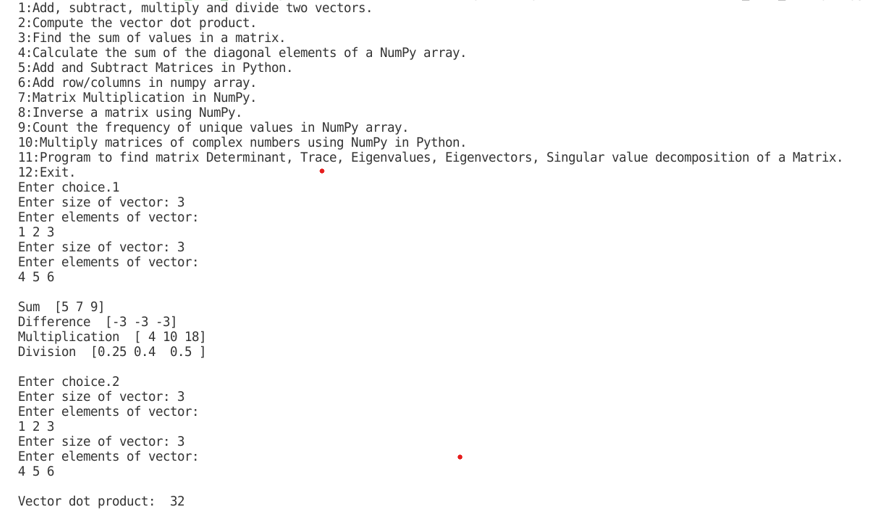
print("\nu=\n", u, "\ns=\n", s, "\nvh=\n", vh)

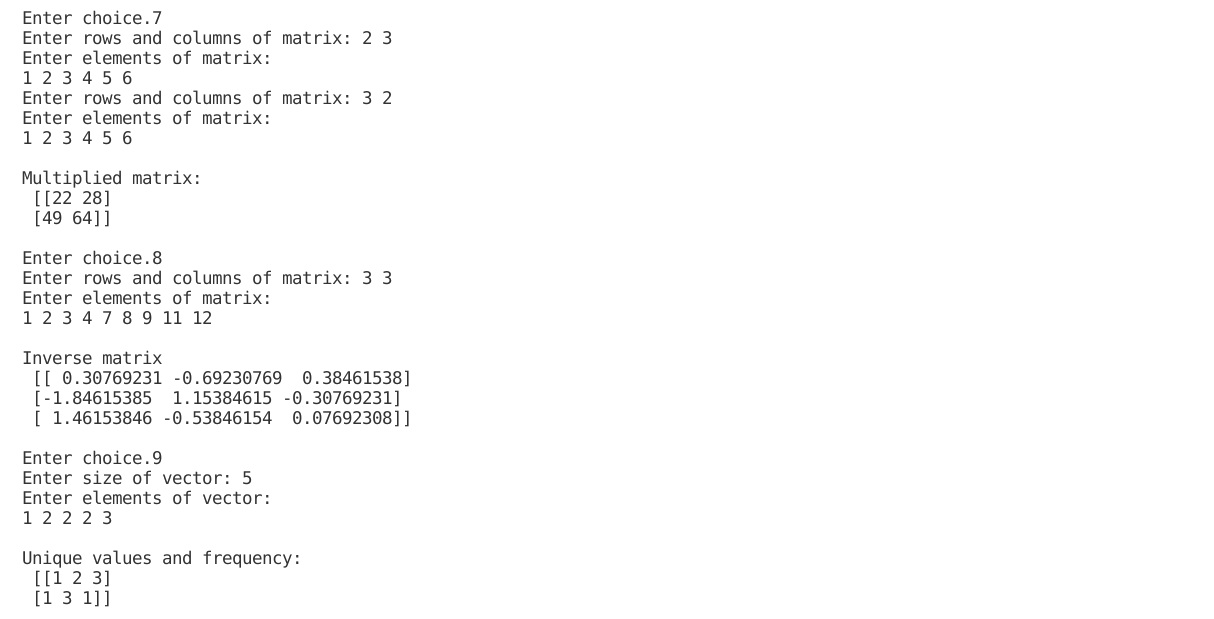
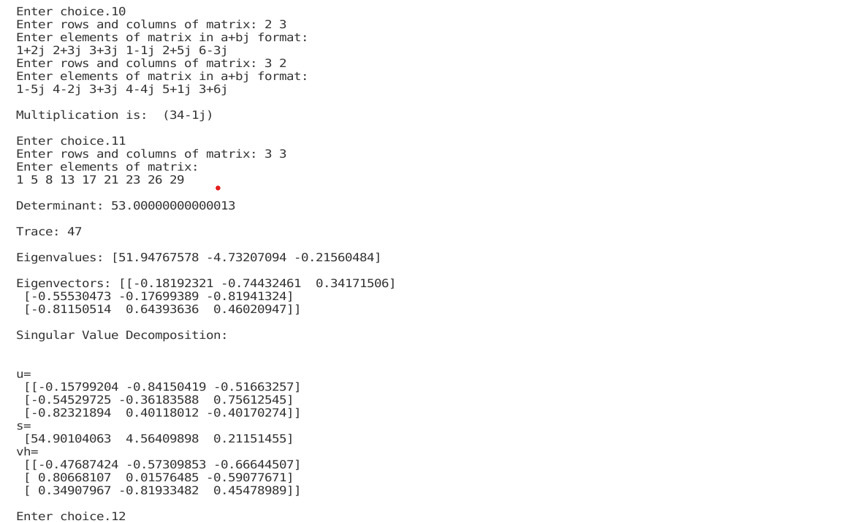
print()

case 12: quit()

**output:**

****





****