**Lab Session #3**

**Exploring NumPy Module in Python**

**Aim:** To Learn NumPy module in python and explore the various functions available in it.

**Problem Definition**: Develop Python Programs to do the following:

1. Create a Numpy array filled with all ones.
2. Find the number of occurrences of a sequence in a NumPy array
3. Interchange two axes of an array
4. Trim the leading and/or trailing zeros from a 1-D array
5. Reverse a numpy array
6. make a NumPy array read-only.
7. Replace NumPy array elements that doesn’t satisfy the given condition
8. Remove columns in Numpy array that contains non-numeric values
9. Create a Numpy array with random values
10. Get the n-largest values of an array using NumPy?
11. get the floor, ceiling and truncated values of the elements of a numpy array
12. Evaluate Einstein’s summation convention of two multidimensional NumPy arrays
13. Compute the variance of the NumPy array
14. Compute the standard deviation of the NumPy array
15. Compute pearson product-moment correlation coefficients of two given NumPy arrays
16. Calculate the mean across dimension in a 2D NumPy array
17. Define a polynomial function. Add, subtract, multiply and divide one polynomial to another using NumPy
18. save a NumPy array to a text file?
19. Load data from a text file
20. Plot line graph from NumPy array
21. Create Histogram using NumPy

**Theory:** NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It is open-source software. It contains various features including these important ones:

* A powerful N-dimensional array object
* Sophisticated (broadcasting) functions
* Tools for integrating C/C++ and Fortran code
* Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

Installation:

Mac and Linux users can install NumPy via pip command:

*pip install numpy*

Windows does not have any package manager analogous to that in linux or mac. Please download the pre-built windows installer for NumPy from here (according to your system configuration and Python version). And then install the packages manually.

NumPy’s main object is the homogeneous multidimensional array.

It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In NumPy dimensions are called axes. The number of axes is rank. NumPy’s array class is called ndarray. It is also known by the alias array.

Array creation: There are various ways to create arrays in NumPy.

* For example, you can create an array from a regular Python list or tuple using the array function. The type of the resulting array is deduced from the type of the elements in the sequences.
* Often, the elements of an array are originally unknown, but its size is known. Hence, NumPy offers several functions to create arrays with initial placeholder content. These minimize the necessity of growing arrays, an expensive operation. For example: np.zeros, np.ones, np.full, np.empty, etc.
* To create sequences of numbers, NumPy provides a function analogous to range that returns arrays instead of lists.
* arange: returns evenly spaced values within a given interval. step size is specified.
* linspace: returns evenly spaced values within a given interval. num no. of elements are returned.
* Reshaping array: We can use reshape method to reshape an array. Consider an array with shape (a1, a2, a3, …, aN). We can reshape and convert it into another array with shape (b1, b2, b3, …, bM). The only required condition is: a1 x a2 x a3 … x aN = b1 x b2 x b3 … x bM . (i.e original size of array remains unchanged.)
* Flatten array: We can use flatten method to get a copy of array collapsed into one dimension. It accepts order argument. Default value is ‘C’ (for row-major order). Use ‘F’ for column major order.

Array Indexing: Knowing the basics of array indexing is important for analysing and manipulating the array object. NumPy offers many ways to do array indexing.

* Slicing: Just like lists in python, NumPy arrays can be sliced. As arrays can be multidimensional, you need to specify a slice for each dimension of the array.
* Integer array indexing: In this method, lists are passed for indexing for each dimension. One to one mapping of corresponding elements is done to construct a new arbitrary array.
* Boolean array indexing: This method is used when we want to pick elements from array which satisfy some condition.

Basic operations: Plethora of built-in arithmetic functions are provided in NumPy.

* Operations on single array: We can use overloaded arithmetic operators to do element-wise operation on array to create a new array. In case of +=, -=, \*= operators, the existing array is modified.
* Unary operators: Many unary operations are provided as a method of ndarray class. This includes sum, min, max, etc. These functions can also be applied row-wise or column-wise by setting an axis parameter.
* Binary operators: These operations apply on array elementwise and a new array is created. You can use all basic arithmetic operators like +, -, /, , etc. In case of +=, -=, = operators, the existing array is modified. And many more.

**Code:**

import numpy as np

import math

import matplotlib.pyplot as plt

def SequenceDetector():

# inputing arrays using list comprehension, then

# converting to a numpy array

array = np.array([int(i) for i in input("Enter elements: ").split()])

sequence = np.array([int(i) for i in input(

"Enter sequence to search for: ").split()])

count = 0

for i in range(len(array)):

if (array[i] == sequence[0]):

j = i + 1

k = 1

while (j < len(sequence)):

if (array[j] == sequence[k]):

k += 1

j += 1

i += 1

if (k == len(sequence)-1):

count += 1

print(count)

def polynomial(x, y):

print("sum = ", np.polyadd(x, y))

print("difference = ", np.polysub(x, y))

print("product = ", np.polymul(x, y))

print("quotient = ", np.polydiv(x, y))

print("1: Create a Numpy array filled with all ones.")

print("2: Find the number of occurrences of a sequence in a NumPy array")

print("3: Interchange two axes of an array")

print("4: Trim the leading and/or trailing zeros from a 1-D array")

print("5: Reverse a numpy array")

print("6: make a NumPy array read-only.")

print("7: Replace NumPy array elements that doesn't satisfy the given condition")

print("8: Remove columns in Numpy array that contains non-numeric values")

print("9: Create a Numpy array with random values")

print("10: Get the n-largest values of an array using NumPy.")

print("11: Get the floor, ceiling and truncated values of the elements of a numpy array")

print("12: Evaluate Einstein's summation convention of two multidimensional NumPy arrays")

print("13: Compute the variance of the NumPy array")

print("14: Compute the standard deviation of the NumPy array")

print("15: Compute pearson product-moment correlation coefficients of two given NumPy arrays")

print("16: Calculate the mean across dimension in a 2D NumPy array")

print("17: Define a polynomial function. Add, subtract, multiply and divide one polynomial to another using NumPy")

print("18: Save a NumPy array to a text file?")

print("19: Load data from a text file")

print("20: Plot line graph from NumPy array")

print("21: Create Histogram using NumPy")

print("0:Exit.")

while (1):

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

match choice:

case 1:

n = int(input("Enter length: "))

array = np.array([1]\*n)

print(array)

case 2:

SequenceDetector()

case 3:

m = int(input("Enter dimension m: "))

n = int(input("Enter dimension n: "))

array = np.array([[int(i) for i in input(

f"Enter elements of row {j}: ").split()] for j in range(m)])

print("original matrix: ", array)

array = np.swapaxes(array, 0, 1)

print("matrix with axes changed : ", array)

case 4:

array = np.array([int(i)

for i in input("Enter elements: ").split()])

array = np.trim\_zeros(array)

print(array)

case 5:

array = np.array([int(i)

for i in input("Enter elements: ").split()])

array = np.flip(array)

print(array)

case 6:

array = np.array([int(i)

for i in input("Enter elements: ").split()])

array.flags.writeable = False

array[0] = 2

case 7:

array = np.array([int(i)

for i in input("Enter elements: ").split()])

print("if element is not divisible by 2 replace it by 0")

array[array % 2 != 0] = 0

print(array)

case 8:

array = [[math.nan, 2, 3], [math.nan, 3, 4]]

# inputing the 2 matrices

array = np.array(array)

print("Given array:")

print(array)

print("\nRemove all columns containing non-numeric elements ")

print(array[:, ~np.isnan(array).any(axis=0)])

case 9:

n = int(input("Enter size of array."))

array = np.random.rand(1, n)

print(array)

case 10:

array = np.array([int(i)

for i in input("Enter elements: ").split()])

array = np.sort(array)

n = int(input("Enter value of n: "))

print(array[-n:])

case 11:

array = np.array([float(i)

for i in input("Enter elements: ").split()])

print("floor-ified array: ", np.floor(array))

print("Ceiling-ified array: ", np.ceil(array))

print("truncated array: ", np.trunc(array))

case 12:

# to enter a 2d numpy matrix

m = int(input("Enter dimension m: "))

n = int(input("Enter dimension n: "))

# inputing the 2 matrices

array1 = np.array([[int(i) for i in input(

f"Enter elements of row {j} of array 1: ").split()] for j in range(m)])

array2 = np.array([[int(i) for i in input(

f"Enter elements of row {j} of array 2: ").split()] for j in range(m)])

print("Original matrices: ")

print(array1)

print(array2)

# calcualting Einstein's summation :

einsteinSummation = np.einsum("mk,kn", array1, array2)

print("Enisteins summation of matrices: ")

print(einsteinSummation)

case 13:

array = np.array([int(i)

for i in input("Enter elements: ").split()])

variance = np.var(array)

print("Variance = ", variance)

case 14:

array = np.array([int(i)

for i in input("Enter elements: ").split()])

stdDeviation = np.std(array)

print("Standard Deviation = ", stdDeviation)

case 15:

array1 = np.array([int(i)

for i in input("Enter elements: ").split()])

array2 = np.array([int(i)

for i in input("Enter elements: ").split()])

pearsonProduct = np.corrcoef(array1, array2)

print("Pearson Product-moment= ", pearsonProduct)

case 16:

m = int(input("Enter dimension m: "))

n = int(input("Enter dimension n: "))

array = np.array([[int(i) for i in input(

f"Enter elements of row {j}: ").split()] for j in range(m)])

# Calculating mean across Rows

row\_mean = np.mean(array, axis=1)

for i in range(len(row\_mean)):

print(f"mean of row {i} = ", row\_mean[i])

# Calculating mean across Columns

column\_mean = np.mean(array, axis=0)

for i in range(len(column\_mean)):

print(f"mean of column {i} = ", column\_mean[i])

case 17:

array1 = np.array([int(i)

for i in input("Enter poly1: ").split()])

array2 = np.array([int(i)

for i in input("Enter poly2: ").split()])

polynomial(array1, array2)

case 18:

array = np.array([int(i)

for i in input("Enter elements: ").split()])

file = open("./Lab Submissions/file.txt", "+w")

for i in array:

file.write(str(i) + "\n")

file.close()

case 19:

array = np.loadtxt("./Lab Submissions/file.txt")

print(array)

case 20:

x = np.array([int(i)

for i in input("Enter x elements: ").split()])

y = np.array([int(i)

for i in input("Enter y elements: ").split()])

plt.title("Line graph")

plt.xlabel("X axis")

plt.ylabel("Y axis")

plt.plot(x, y, color="red")

plt.show()

case 21:

a = np.random.randint(100, size=(50))

bins = np.array([int(i)

for i in input("Enter bins value: ").split()])

np.histogram(a, bins)

hist, bins = np.histogram(a, bins)

# printing histogram

print()

print(hist)

print(bins)

fig = plt.figure(figsize=(10, 7))

plt.hist(a, bins)

plt.title("Numpy Histogram")

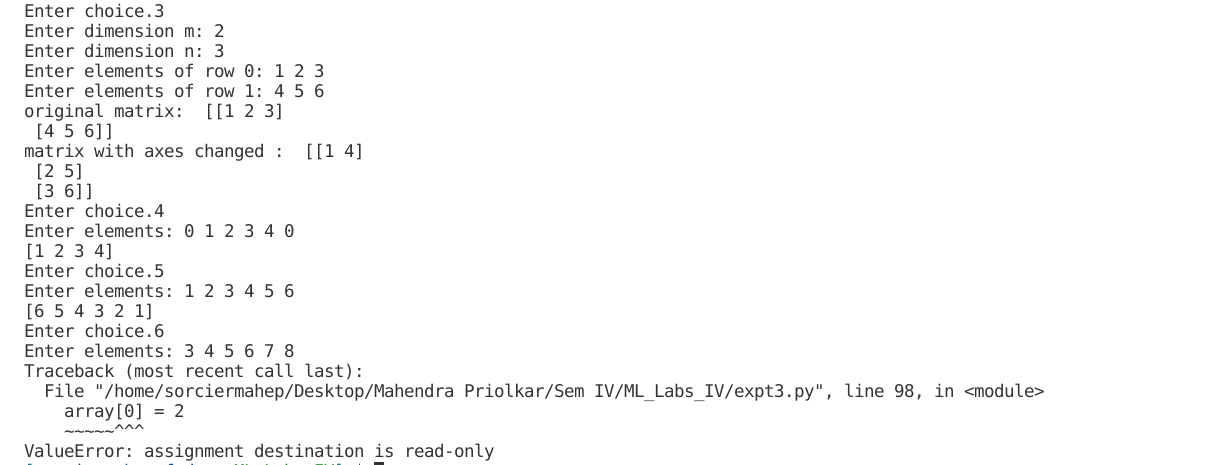
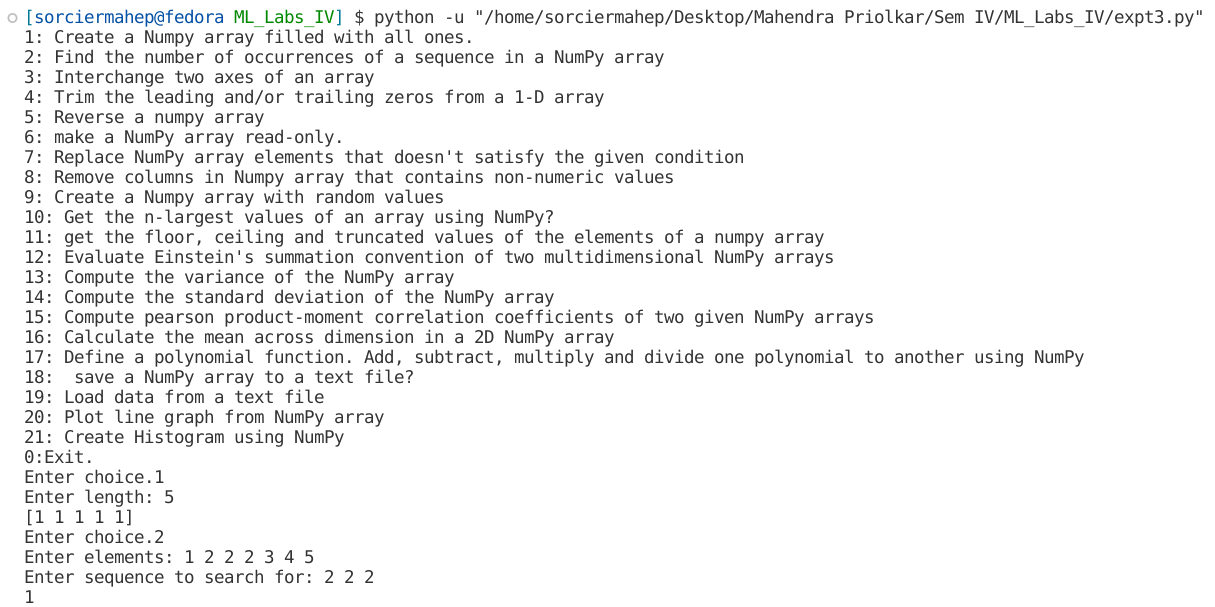
plt.show()

print()

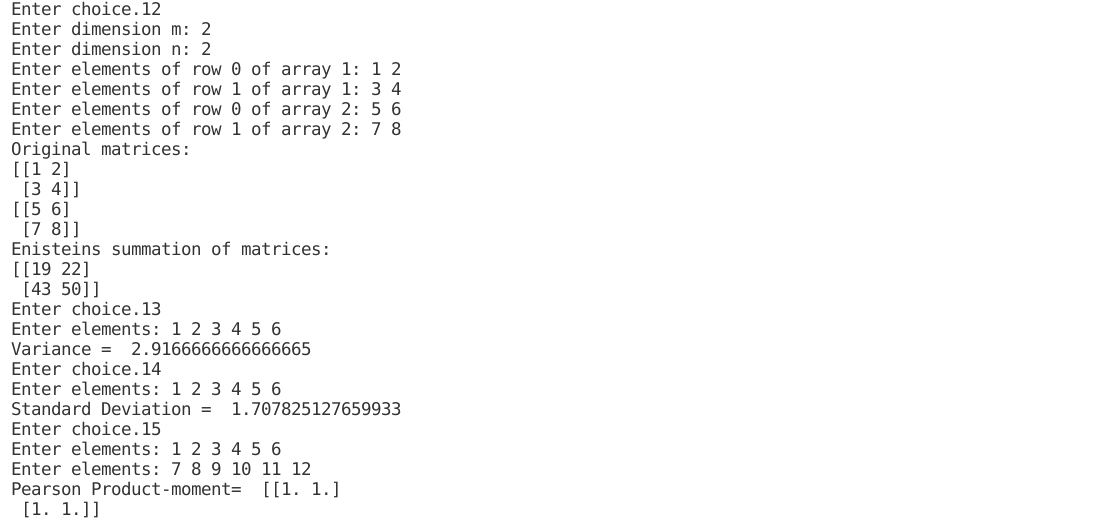
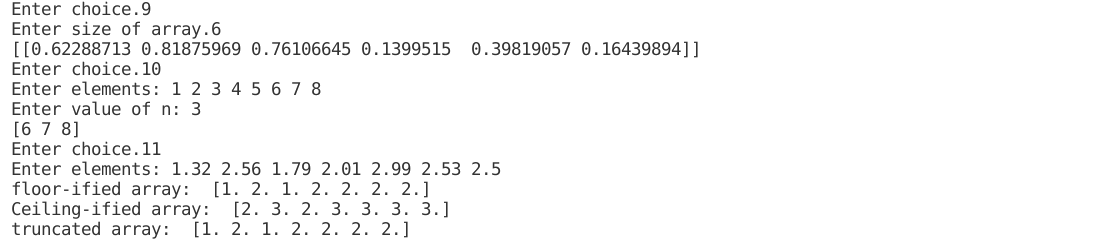
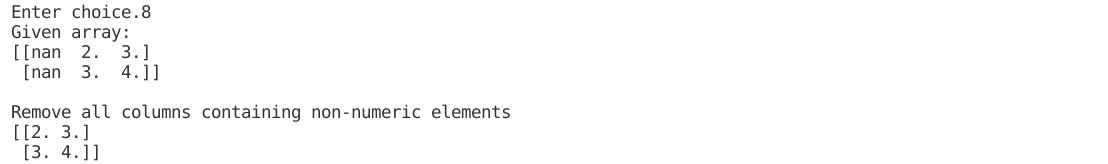
case 0:

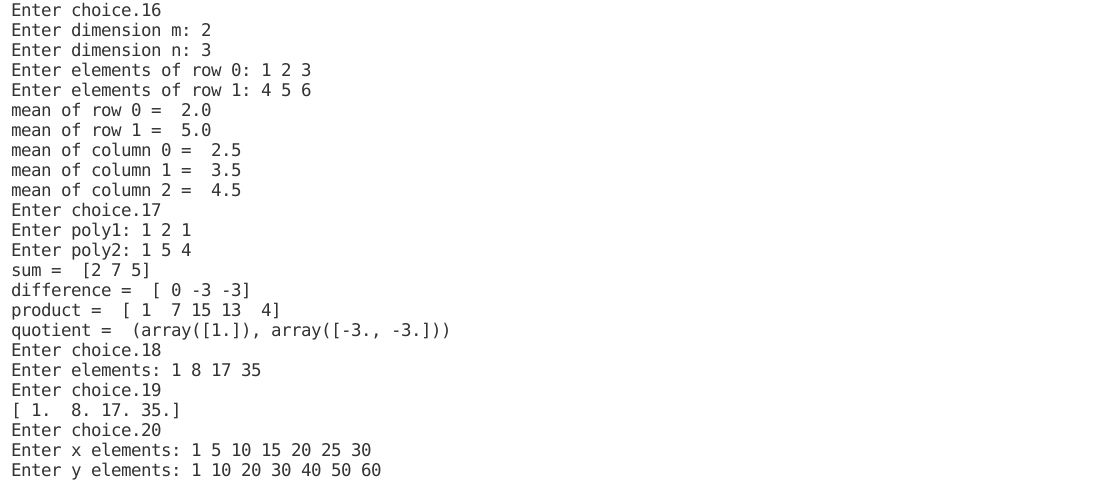
quit()

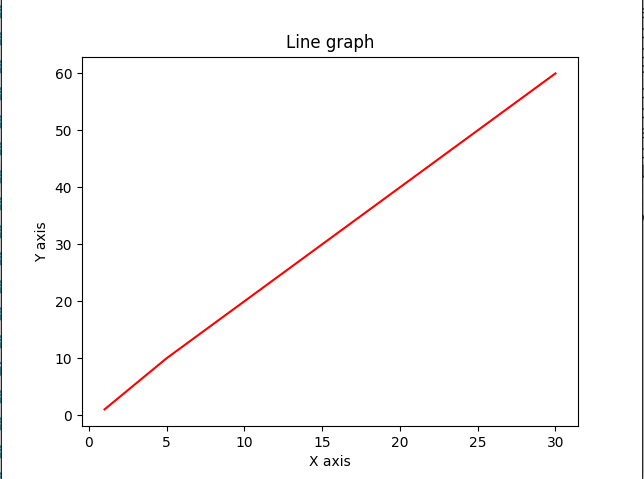
**Output:**

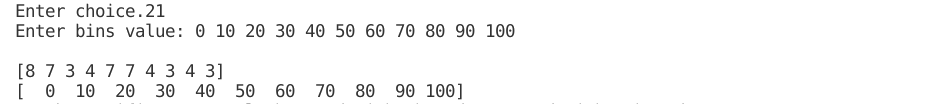










****

