import numpy as np

a. Using Numpy, write a basic array of operations on single array to add x to each element of array and subtract y from each element of array.

```
arr = np.array([1,2,3,4,5])

print(arr+1)
```

Output:

```
array([2, 3, 4, 5, 6])
```

```
print(arr-1)
```

Output:

```
array([0, 1, 2, 3, 4])
```

b. Using Numpy, write a program to add, subtract and multiply two matrices.

```
arr1 = np.array([1,2,3,4])

arr2 = np.array([4,3,2,1])

print(arr1+arr2)

print(arr1-arr2)

print(arr1*arr2)
```

Output:

```
[5 5 5 5]
[-3 -1  1  3]
[4 6 6 4]
```

c. Write a Python program to do the following operations: Library: NumPy

i) Create multi-dimensional arrays and find its shape and dimension

```
matrix = np.ones([2,3,3])

print(matrix.shape)

print(matrix.ndim)
```

Output:

```
(2, 3, 3)
3
```

ii) Create a matrix full of zeros and ones

```
matrix = np.identity(3,dtype=int)

print(matrix)
```

Output:

```
array([[1, 0, 0],
       [0, 1, 0],
       [0, 0, 1]])
```

iii) Reshape and flatten data in the array

```
matrix.flatten()
```

```
array([1, 0, 0, 0, 1, 0, 0, 0, 1])
matrix.reshape([1,9])
```

Output:

```
array([[1, 0, 0, 0, 1, 0, 0, 0, 1]])
```

iv) Append data vertically and horizontally

```
row = np.array([1,2,3])
col = np.array([1,2,3])
print(np.vstack((matrix,row)))
print(np.hstack((matrix,col.reshape([len(col),1]))))
```

Output:

```
[[1 0 0]
 [0 1 0]
 [0 0 1]
 [1 2 3]]
[[1 0 0 1]
 [0 1 0 2]
 [0 0 1 3]]
```

v) Apply indexing and slicing on array

```
matrix[1:,1:]
```

Output:

```
array([[1, 0], [0, 1]])
```

vi) Use statistical functions on array - Min, Max, Mean, Median and Standard Deviation

```
arr = np.array([1,2,3,4,5,6])

print(np.max(arr))

print(np.min(arr))

print(np.mean(arr))
```

print(np.median(arr))

print(np.std(arr))

Output:

```
6
1
3.5
3.5
1.707825127659933
```

vii) Dot and matrix product of two arrays

mat1 = np.array([[1,2],[4,5]])

mat2 = np.array([[6,5],[3,2]])

print(np.dot(mat1,mat2))

print(np.multiply(mat1,mat2))

Output:

```
[[12  9]
 [39 30]]
[[ 6 10]
 [12 10]]
```

viii) Compute the Eigen values of a matrix

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

np.linalg.eig(matrix)

Output:

```
(array([16.58623849+0.j        , -0.29311924+0.79848134j,
        -0.29311924-0.79848134j]),
 array([[ 0.22456533+0.j        , -0.16240098+0.43130149j,
         -0.16240098-0.43130149j],
        [ 0.60905077+0.j        ,  0.692036  +0.j        ,
          0.692036  -0.j        ],
        [ 0.76067573+0.j        , -0.50615145-0.2291328j ,
         -0.50615145+0.2291328j ]]))
```

ix) Solve a linear matrix equation such as 3 * x0 + x1 = 9, x0 + 2 * x1 = 8

D = np.linalg.det([[3,1],[1,2]])

d1 = np.linalg.det([[9,1],[2,2]])

d2 = np.linalg.det([[3,1],[1,2]])

```
print(f'X0 = {round(d1/D)} and x1 = {round(d2/D)}')
```

Output:

X0 = 3 and x1 = 1

Output:

a = [[1,2,3],[4,5,6],[7,8,9]]

x) Compute the multiplicative inverse of a matrix

```
np.linalg.inv(a)
```

Output:

```
array([[ 3.15251974e+15, -6.30503948e+15,  3.15251974e+15],
       [-6.30503948e+15,  1.26100790e+16, -6.30503948e+15],
       [ 3.15251974e+15, -6.30503948e+15,  3.15251974e+15]])
```

xi) Compute the rank of a matrix

```
np.linalg.matrix_rank(a)
```

Output:

2

xii) Compute the determinant of an array

```
np.linalg.det(a)
```

Output:

-9.51619735392994e-16