###1

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

i) Create a one-dimensional array and perform all operations on it.

]:

```
import numpy as np
#creation
a=np.array([3,4,2,6,9])
b=np.array([1,0,3,7,2])
print("1-D Array 1:",a)
print("1-D Array 2:",b)

#Basic Operation
print("Addition:",a+b)
print("Subtraction:",a-b)
print("Multiplication:",np.dot(a,b))
print("Division:",a/b)
print("Reshaping Array 1\n",a.reshape(5,1))
print("Transposing Array 1\n",np.transpose(a))
print("Mean of array 1:",np.mean(a))
print("Median of array 1:",np.median(a))
print("Squares of array 1:",np.square(a))
print("Roots of array 1:",np.sqrt(a))
```

]:

```
1-D Array 1: [3 4 2 6 9]
1-D Array 2: [1 0 3 7 2]
Addition: [ 4  4  5 13 11]
Subtraction: [ 2  4 -1 -1  7]
Multiplication: 69
Division: [3.                inf 0.66666667 0.85714286 4.5       ]
Reshaping Array 1
 [[3]
 [4]
 [2]
 [6]
 [9]]
Transposing Array 1
 [3 4 2 6 9]
Mean of array 1: 4.8
Median of array 1: 4.0
Squares of array 1: [ 9 16  4 36 81]
Roots of array 1: [1.73205081 2.         1.41421356 2.44948974 3.        ]
```

]:

```
<ipython-input-18-79bdff051783>:12: RuntimeWarning: divide by zero encountered in divide
  print("Division:",a/b)
```

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

]:

```

import numpy as np
#creation
a=np.array([[1,2,3],[4,5,6],[7,8,9]])
print("Array:",a)
print("Shape:",np.shape(a))
print("Dimension:",np.ndim(a))
```

]:

```
Array: [[1 2 3]
 [4 5 6]
 [7 8 9]]
Shape: (3, 3)
Dimension: 2
```

iii) Create a matrix full of zeros and ones

]:

```
import numpy as np
a=np.zeros((2,2))
b=np.ones((2,3))
print(a,"\n",b)
```

]:

```
[[0. 0.]
 [0. 0.]]
 [[1. 1. 1.]
 [1. 1. 1.]]
```

iv) Reshape and flatten data in the array

]:

```
a=np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])
b=a.flatten()
print("Array:\n",a)
print("Reshaped Array :\n",a.reshape(4,3))
print("Flatted Array:",b)
```

```
Array:
 [[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
Reshaped Array :
 [[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
Flatted Array: [ 1  2  3  4  5  6  7  8  9 10 11 12]
```

v) Perform arithmetic operations on multi-dimensional arrays

```
a=np.array([[1,2,3],[4,5,6],[7,8,9]])
b=np.array([[10,3,4],[1,4,2],[6,2,4]])
print("Array 1:",a)
print("Array 2:",b)
print("Arithmetic Operations")
print("Addition:\n",a+b)
print("Subtractoin:\n",a-b)
print("Multiplication:\n",a*b)
print("Division:\n",a/b)
```

```
Array 1: [[1 2 3]
 [4 5 6]
 [7 8 9]]
Array 2: [[10  3  4]
 [ 1  4  2]
 [ 6  2  4]]
Arithmetic Operations
Addition:
 [[11  5  7]
 [ 5  9  8]
 [13 10 13]]
Subtractoin:
 [[-9 -1 -1]
 [ 3  1  4]
 [ 1  6  5]]
Multiplication:
 [[10  6 12]
 [ 4 20 12]
 [42 16 36]]
Division:
 [[0.1        0.66666667 0.75      ]
 [4.         1.25       3.        ]
 [1.16666667 4.         2.25      ]]
```

vi) Append data vertically and horizontally

```
#stacking arrays
a1=np.array([[1,1],[2,2]])
a2=np.array([[3,3],[4,4]])
print(a1,a2)
print('\n vstack \n')
print(np.vstack((a1,a2)))
print('\n hstack \n')
print(np.hstack((a1,a2)))
```

```
[[1 1]
 [2 2]] [[3 3]
 [4 4]]

 vstack

[[1 1]
 [2 2]
 [3 3]
 [4 4]]

 hstack

[[1 1 3 3]
 [2 2 4 4]]
```

vii) Apply indexing and slicing on array

```
import numpy as np

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

#indexing
print("Element at index 2:", arr[2])
print("Element at index 5:", arr[5])
```

```
#Slicing
print("Slice from index 2 to 5:", arr[2:6])
print("Slice from index 3 to the end:", arr[3:])
print("Slice from the beginning to index 4:", arr[:5])

#Negative indexing and slicing
print("Element at index -1 (last element):", arr[-1])
print("Slice from the end to index -3:", arr[-3:])
print("Slice from index -5 to index -2:", arr[-5:-2])
```

]:

```
Element at index 2: 3
Element at index 5: 6
Slice from index 2 to 5: [3 4 5 6]
Slice from index 3 to the end: [4 5 6 7 8 9]
Slice from the beginning to index 4: [1 2 3 4 5]
Element at index -1 (last element): 9
Slice from the end to index -3: [7 8 9]
Slice from index -5 to index -2: [5 6 7]
```

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

]:

```
#creation
a=np.array([[1,2,3],[4,5,6]])
print("Array :",a)
#min
print("Minimum element :",np.min(a))
#max
print("Maximum element : ",np.max(a))
#mean
print("Mean : {}".format(np.mean(a)))
#median
print("Median : {}".format(np.median(a)))
#standard deviation
print("Standard Deviation : {}".format(np.std(a)))
```

]:

```
Array : [[1 2 3]
 [4 5 6]]
Minimum element : 1
Maximum element :  6
Mean : 3.5
Median : 3.5
Standard Deviation : 1.707825127659933
```

ix) Dot matrix product of two arrays

]:

```
#creation
a=np.array([[1,2,3],[4,5,6]])
b=np.array([[1,2],[3,4],[5,6]])
print("Array 1:",a)
print("Array 2:",b)
#dot product
print("Dot Product :",np.dot(a,b))
```

]:

```
Array 1: [[1 2 3]
 [4 5 6]]
Array 2: [[1 2]
 [3 4]
 [5 6]]
Dot Product : [[22 28]
 [49 64]]
```

x) Compute the Eigen values of a matrix

]:

```
#creation
a=np.array([[1,2,3],[4,5,6],[7,8,9]])
print("Array :",a)
c=np.array(np.linalg.eigvals(a))
print("Eigen values are:\n",c)
```

]:

```
Array : [[1 2 3]
 [4 5 6]
 [7 8 9]]
Eigen values are:
 [ 1.61168440e+01 -1.11684397e+00 -1.30367773e-15]
```

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

]:

```
#input
e=int(input("Enter no.of linear equations "))
n=int(input("Enter no.of variables "))

cof=[]
cons=[]
```

```
for i in range(e):
  print("Enter equation {}".format(i+1))
  l=[]
  for j in range(n):
    c=int(input("Enter coefficent of x{}".format(j)))
    l.append(c)
  cof.append(l)
  k=int(input("Enter constant of equation {}".format(i+1)))
  cons.append(k)
cof=np.array(cof)
cons=np.array(cons)
x = np.linalg.solve(cof, cons)
print("Solutions are")
for i in range(len(x)):
  print("x{}: {}".format(i,x[i]))
```

```
Enter no.of linear equations 2
Enter no.of variables 2
Enter equation 1
Enter coefficent of x03
Enter coefficent of x11
Enter constant of equation 19
Enter equation 2
Enter coefficent of x01
Enter coefficent of x12
Enter constant of equation 28
Solutions are
x0: 2.0
x1: 3.0
```

xii) Compute the multiplicative inverse of a matrix

```
import numpy as np
#creation
a=eval(input("Enter matrix "))
a=np.array(a)
#check for square matrix
if(a.shape[0]==a.shape[1]):
  det=np.linalg.det(a)
  #check for non-singular
  if(det!=0):
    inver=np.linalg.inv(a)
    print("Matrix:\n",a)
    print("Inverse Matrix:\n",inver)
  else:
    print("Matrix is singular")
else:
  print("Matrix is not square")
```

```
Enter matrix [[2,3],[1,4]]
Matrix:
 [[2 3]
 [1 4]]
Inverse Matrix:
 [[ 0.8 -0.6]
 [-0.2  0.4]]
```

xiii) Compute the rank of a matrix

```
#creation
a=eval(input("Enter matrix "))
a=np.array(a)
#rank
rank=np.linalg.matrix_rank(a)
print("Rank of matrix is ",rank)
```

```
Enter matrix [[1,2,3],[4,5,6],[7,8,9]]
Rank of matrix is  2
```

xiv) Compute the determinant of an array

```
#creation
a=eval(input("Enter matrix "))
a=np.array(a)
#determinant
det=np.linalg.det(a)
print("Array :\n",a)
print("Determinant:",det)
```

```
Enter matrix [[1,2,3],[4,5,6],[7,8,9]]
Array :
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
Determinant: 0.0
```

xv) Perform transpose and change of axes operations on arrays

]:

```
#creation
a=eval(input("Enter matrix "))
a=np.array(a)
#transpose
trans=np.transpose(a)
print("Array :\n",a)
print("Transpose:\n",trans)
print("Shape of given array is:",np.shape(a))
#change of axes
print(a.swapaxes(1,1))
```

]:

```
Enter matrix [[1,2,3],[4,5,6]]
Array :
 [[1 2 3]
 [4 5 6]]
Transpose:
 [[1 4]
 [2 5]
 [3 6]]
Shape of given array is: (2, 3)
[[1 2 3]
 [4 5 6]]
```

xvi) Perform splitting operations on arrays.

]:

```
#splitting arrays
import numpy as np
a=np.arange(1,25).reshape(12,2)
print("Array:",a)
print("After Splitting")
print('\n  VSPLIT  ')
#vsplit
v1=np.array(np.vsplit(a,3)) #splits along rows
print(v1)
#hsplit
print('\n  HSPLIT  ')
v2=np.array(np.hsplit(a,2)) #splits along columns
print(v2)
```

]:

```
Array: [[ 1  2]
 [ 3  4]
 [ 5  6]
 [ 7  8]
 [ 9 10]
 [11 12]
 [13 14]
 [15 16]
 [17 18]
 [19 20]
 [21 22]
 [23 24]]
After Splitting

  VSPLIT  
[[[ 1  2]
  [ 3  4]
  [ 5  6]
  [ 7  8]]

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

 [[17 18]
  [19 20]
  [21 22]
  [23 24]]]

  HSPLIT  
[[[ 1]
  [ 3]
  [ 5]
  [ 7]
  [ 9]
  [11]
  [13]
  [15]
  [17]
  [19]
  [21]
  [23]]

 [[ 2]
  [ 4]
  [ 6]
  [ 8]
  [10]
  [12]
  [14]
  [16]
  [18]
  [20]
  [22]
  [24]]]
```

###2

1.How to convert an array of strings to an array of floats in numpy?

```
import numpy as np
string_arr = np.array(['1.1', '2.2', '3.3'])
float_arr = string_arr.astype(np.float64)
print(float_arr)
```

```
[1.1 2.2 3.3]
```

2. Extracting first n columns of a Numpy matrix

```
import numpy as np

the_arr = np.array([[0, 1, 2, 3, 5, 6, 7, 8],
                    [4, 5, 6, 7, 5, 3, 2, 5],
                    [8, 9, 10, 11, 4, 5, 3, 5]])


print(the_arr[:, 1:5])
```

```
[[ 1  2  3  5]
 [ 5  6  7  5]
 [ 9 10 11  4]]
```

3.How to calculate the sum of every row in a NumPy array in Python?

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])

newarr = arr.reshape(4, 3)
print(newarr)

column_sums = newarr.sum(axis=1)
print(column_sums)
```

```
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
[ 6 15 24 33]
```

4. How to check all elements are NaN in a NumPy Array in Python?

```
import numpy as np


the_array = np.array([np.nan, 2, 3, 4])
array_has_nan = np.isnan(the_array).all()
print(array_has_nan)


the_array = np.array([np.nan, np.nan, np.nan, np.nan])
array_has_nan = np.isnan(the_array).all()
print(array_has_nan)
```

```
False
True
```

5. How to multiply each element of Numpy array in Python?

```
import numpy as np

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

prod = np.prod(the_array)
print(prod)
```

```
36
```

6. Scalar Arithmetic Operations on NumPy Array

```python
import numpy as np

array1 = np.array([[10, 20, 30], [40, 50, 60]])

print(array1 + 2)
print("-" * 20)

print(array1 - 5)
print("-" * 20)
print(array1 * 2)
print("-" * 20)

print(array1 / 5)
print("-" * 20)

print(array1 ** 2)
print("-" * 20)
```

```
[[12 22 32]
 [42 52 62]]
--------------------
[[ 5 15 25]
 [35 45 55]]
--------------------
[[ 20  40  60]
 [ 80 100 120]]
--------------------
[[ 2.  4.  6.]
 [ 8. 10. 12.]]
--------------------
[[ 100  400  900]
 [1600 2500 3600]]
--------------------
```

7. How to check for NaN elements in a NumPy Array in Python?

```python
import numpy as np

the_array = np.array([np.nan, 2, 3, 4])
array_has_nan = np.isnan(the_array).any()
print(array_has_nan)

the_array = np.array([1, 2, 3, 4])
array_has_nan = np.isnan(the_array).any()
print(array_has_nan)
```

```
True
False
```

8. NumPy Element Wise Mathematical Operations

```python
import numpy as np

array1 = np.array([[10, 20, 30], [40, 50, 60]])
array2 = np.array([[2, 3, 4], [4, 6, 8]])
array3 = np.array([[-2, 3.5, -4], [4.05, -6, 8]])

print(np.add(array1, array2))
print("-" * 40)

print(np.power(array1, array2))
print("-" * 40)
print(np.remainder((array2), 5))
print("-" * 40)

print(np.reciprocal(array3))
print("-" * 40)

print(np.sign(array3))
print("-" * 40)

print(np.ceil(array3))
print("-" * 40)

print(np.round(array3))
print("-" * 40)
```

```
[[12 23 34]
 [44 56 68]]
----------------------------------------
```

```
[[            100            8000          810000]
 [        2560000     15625000000 167961600000000]]
----------------------------------------
[[2 3 4]
 [4 1 3]]
----------------------------------------
[[-0.5         0.28571429 -0.25      ]
 [ 0.24691358 -0.16666667  0.125     ]]
----------------------------------------
[[-1.  1. -1.]
 [ 1. -1.  1.]]
----------------------------------------
[[-2.  4. -4.]
 [ 5. -6.  8.]]
----------------------------------------
[[-2.  4. -4.]
 [ 4. -6.  8.]]
----------------------------------------
```

9. How to count frequency of unique values in a NumPy array in Python

```
import numpy as np

the_array = np.array([9, 7, 4, 7, 3, 5, 9])

frequencies = np.asarray((np.unique(the_array, return_counts=True))).T
print(frequencies)
```

```
[[3 1]
 [4 1]
 [5 1]
 [7 2]
 [9 2]]
```

10. Write a NumPy program to get the indices of the sorted elements of a given

```
import numpy as np
student_id = np.array([1023, 5202, 6230, 1671, 1682, 5241, 4532])
print("Original array:")
print(student_id)
i = np.argsort(student_id)
print("Indices of the sorted elements of a given array:")
print(i)
```

```
Original array:
[1023 5202 6230 1671 1682 5241 4532]
Indices of the sorted elements of a given array:
[0 3 4 6 1 5 2]
```

11. How to print a full NumPy array without truncation in Python?

```
import numpy as np


np.set_printoptions(threshold=np.inf)

the_array = np.arange(100)
print(the_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 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71
 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95
 96 97 98 99]
```

12. How to get the transpose of a NumPy array in Python?

```
import numpy as np

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

print(the_array.T)
```

```
[[1 2]
 [3 4]]
[[1 3]
 [2 4]]
```

13. How do you replace items that satisfy a condition with another value in Numpy array?

```
import numpy as np
```

```python
the_array = np.array([49, 7, 44, 27, 13, 35, 71])

an_array = np.where(the_array > 30, 0, the_array)
print(an_array)
```

```
]:
[ 0  7  0 27 13  0  0]
```