Assignment 3

Name: Yash Gunjal

PRN No: 202201040106

Division: G

Roll No: 724

Question: Perform all the Numpy operations in Python.

Code

```
import numpy as np
array1=np.array([[1,2,3],[4,5,6],[7,8,9]])
array1
```

Output

```
array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
```

Code

```
array2=np.array([[11,12,13],[14,15,16],[17,18,19]])
array2
```

Output

```
array([[11, 12, 13], [14, 15, 16], [17, 18, 19]])
```

A. Performance of all matrix operations

Code

```
#Addition

resultarray=array1+array2
print("\nUsing Operator:\n",resultarray)
resultarray=np.add(array1,array2)
print("\nUsing Numpy Function:\n",resultarray)
```

Output

```
Using Operator:

 [[12 14 16]
 [18 20 22]
 [24 26 28]]

Using Numpy Function:
 [[12 14 16]
 [18 20 22]
 [24 26 28]]
```

Code

```
# Subtraction
resultarray=array1-array2
print("\nUsing Operator:\n",resultarray)
resultarray=np.subtract(array1,array2)
print("\nUsing Numpy Function:\n",resultarray)
```

Output

```
Using Operator:
 [[-10 -10 -10]
 [-10 -10 -10]
 [-10 -10 -10]]

Using Numpy Function:
 [[-10 -10 -10]
 [-10 -10 -10]
 [-10 -10 -10]]
```

Code

```
# Multiplication
resultarray=array1*array2
print("\nUsing Operator:\n",resultarray)
resultarray=np.multiply(array1,array2)
print("\nUsing Numpy Function:\n",resultarray)
```

Output

```
Using Operator:
 [[ 11  24  39]
 [ 56  75  96]
 [119 144 171]]

Using Numpy Function:
 [[ 11  24  39]
 [ 56  75  96]
 [119 144 171]]
```

Code

```
# Division
resultarray=array1/array2
print("\nUsing Operator:\n",resultarray)
resultarray=np.divide(array1,array2)
print("\nUsing Numpy Function:\n",resultarray)
```

Output

```
Using Operator:
 [[0.09090909 0.16666667 0.23076923]
 [0.28571429 0.33333333 0.375     ]
 [0.41176471 0.44444444 0.47368421]]

Using Numpy Function:
 [[0.09090909 0.16666667 0.23076923]
 [0.28571429 0.33333333 0.375     ]
 [0.41176471 0.44444444 0.47368421]]
```

Code

```
#Modulus
resultarray=array1%array2
print("\nUsing Operator:\n",resultarray)
resultarray=np.mod(array1,array2)
print("\nUsing Numpy Function:\n",resultarray)
```

Output

```
Using Operator:

 [[1 2 3]
 [4 5 6]
 [7 8 9]]

Using Numpy Function:
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
```

Code

```
# Dot Product
resultarray=np.dot(array1,array2)
print("",resultarray)
```

Output

```
[[ 90  96 102]
 [216 231 246]
 [342 366 390]]
```

Code

```
# Transpose of Matrix
resultarray=np.transpose(array1)
print(resultarray)
#Or
resultarray=array1.transpose()
print(resultarray)
```

Output

```
[[1 4 7]
 [2 5 8]
 [3 6 9]]
[[1 4 7]
 [2 5 8]
 [3 6 9]] [216 231 246]
 [342 366 390]]
```

B. Horizontal and vertical stacking of Numpy Arrays

Code

```
# Horizontal Stacking
resultarray=np.hstack((array1,array2))
resultarray
```

Output

```
array([[ 1,  2,  3, 11, 12, 13],
       [ 4,  5,  6, 14, 15, 16],
       [ 7,  8,  9, 17, 18, 19]])
```

Code

```
# Vertical Stacking array([[ 1,  2,  3, 11, 12, 13],
       [ 4,  5,  6, 14, 15, 16],
       [ 7,  8,  9, 17, 18, 19]])

resultarray=np.vstack((array1,array2))
resultarray
```

Output

```
array([[ 1, 2, 3], [ 4, 5, 6], [ 7, 8, 9], [11, 12, 13], [14, 15, 16], [17,
18, 19]])
```

C. Custom sequence generation

Code

```
#  Range
nparray=np.arange(0,12,1).reshape(3,4)
nparray
```

Output

```
array([[ 0,  1,  2,  3], [ 4,  5,  6,  7], [ 8,  9, 10, 11]])
```

Code

```
# Linear Seperable
nparray=np.linspace(start=0,stop=24,num=12).reshape(3,4)
nparray
```

Output

```
array([[ 0. ,  2.18181818,  4.36363636,  6.54545455], [ 8.72727273,
10.90909091, 13.09090909, 15.27272727], [17.45454545, 19.63636364,
21.81818182, 24. ]])
```

Code

```
# Empty Array
nparray=np.empty((3,3),int)
nparray
```

Output

```
array([[ 90,  96, 102], [216, 231, 246], [342, 366, 390]])
```

Code

```
# Empty of some other array
nparray=np.empty_like(array1)
nparray
```

Output

```
array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
```

Code

```
# identity Matrix
nparray=np.identity(3)
nparray
```

Output

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

D. Arithmetic Operations

Code

```
#Arithmatic operations
array1=np.array([1,2,3,4,5])
array2=np.array([11,12,13,14,15])
print(array1)
print(array2)

# Addition
print(np.add(array1,array2))
# Subtraction
print(np.subtract(array1,array2))
# Multiplication
print(np.multiply(array1,array2))
# Division
print(np.divide(array1,array2))
```

Output

```
[1 2 3 4 5]
[11 12 13 14 15]
[12 14 16 18 20]
[-10 -10 -10 -10 -10]
[11 24 39 56 75]
[0.09090909 0.16666667 0.23076923 0.28571429 0.33333333]
```

E. statistical and Mathmatical Operations

Code

```
# statistical and Mathmatical Operations

array1=np.array([1,2,3,4,5,9,6,7,8,9,9])
# Standard Deviation
print(np.std(array1))
#Minimum
print(np.min(array1))
#Summation
print(np.sum(array1))
#Median
print(np.median(array1))
#Mean
print(np.mean(array1))
#Mode
from scipy import stats
print("Most Frequent element=",stats.mode(array1)[0])
print("Number of Occarances=",stats.mode(array1)[1])
# Variance
print(np.var(array1))
```

Output

```
2.7990553306073913
1
63
6.0
5.7272727272727275
Most Frequent element= [9]
Number of Occarances= [3]
7.834710743801653
```

F. Bitwise Operators

Code

```
# Bitwise operators

array1=np.array([1,2,3],dtype=np.uint8)
array2=np.array([4,5,6])
# AND
resultarray=np.bitwise_and(array1,array2)
print(resultarray)
# OR
resultarray=np.bitwise_or(array1,array2)
print(resultarray)
#LeftShift
resultarray=np.left_shift(array1,2)
print(resultarray)
#RightShift
resultarray=np.right_shift(array1,2)
print(resultarray)
```

Output

```
[0 0 2]
[5 7 7]
[ 4  8 12]
[0 0 0]
```

G. Copying and viewing Array

Code

```
# Copy array

array1=np.arange(1,10)
print(array1)
newarray=array1.copy()
print(newarray)
##modification in Original Array
array1[0]=100
print(array1)
print(newarray)
```

Output

```
[1 2 3 4 5 6 7 8 9]
[1 2 3 4 5 6 7 8 9]
[100   2   3   4   5   6   7   8   9]
[1 2 3 4 5 6 7 8 9]
```

Code

```
# view Array

array1=np.arange(1,10)
print(array1)
newarray=array1.view()
print(newarray)
##modification in Original Array
array1[0]=100
print(array1)
print(newarray)
```

Output

```
[1 2 3 4 5 6 7 8 9]
[1 2 3 4 5 6 7 8 9]
[100   2   3   4   5   6   7   8   9]
[100   2   3   4   5   6   7   8   9]
```

Code

```
# view Array

array1=np.arange(1,10)
print(array1)
newarray=array1.view()
print(newarray)
##modification in Original Array
array1[0]=100
print(array1)
print(newarray)
```

Output

```
[1 2 3 4 5 6 7 8 9]
[1 2 3 4 5 6 7 8 9]
[100   2   3   4   5   6   7   8   9]
[100   2   3   4   5   6   7   8   9]
```

H. Searching and Sorting in Array

Code

```
# Searching of array

array1=np.array([[1,2,3,12,5,7],[94,5,6,7,89,44],[7,8,9,11,13,14]])
print(array1)

#Horizontally Sort
np.sort(array1,axis=0)

# Vertically Sort
np.sort(array1,axis=1)

#Perform Search After sorting
array1=np.array([1,2,3,12,5,7])
np.searchsorted(array1,7,side="left")
```

Output

```
[[ 1  2  3 12  5  7]
 [94  5  6  7 89 44]
 [ 7  8  9 11 13 14]]
3
```

I. Counting

Code

```
# counting

array1=np.array([1,2,3,12,5,7,0])
print(np.count_nonzero(array1))#Return total Non Zero element
print(np.nonzero(array1))#Return Index
print(array1.size)#Total Element
```

Output

```
6
(array([0, 1, 2, 3, 4, 5]),)
7
```