

Name - Onkar Katkamwar  
Roll No - 630  
PRN - 202201040098  
Batch - F(2)

## Essential of Data Science Lab Assignment No: 3

```
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]])
```

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

Output -

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

## 1. Matrix Operation

### 1.1 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]]
```

## 1.2. 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]]
```

## 1.3. 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]]
```

## 1.4. 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]]
```

## 1.5. Mod

```
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]]
```

## 1.6. dot Product

```
[ ] resultarray=np.dot(array1,array2)
    print("",resultarray)
```

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

## 1.7. Transpose

```
[ ] resultarray=np.transpose(array1)
    print(resultarray)
#Or
resultarray=array1.transpose()
print(resultarray)
```

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

## 2. Horizontal and vertical stacking of Numpy Arrays

### 2.1. Horizontal Stacking

```
[ ]
resultarray=np.hstack((array1,array2))
resultarray
```

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

### 2.2. Vertical Stacking

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

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

## 3. Custom sequence generation

### 3.1. Range

```
[ ] nparray=np.arange(0,12,1).reshape(3,4)
nparray
```

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

### 3.2. Linearly Separable

```
[ ] nparray=np.linspace(start=0,stop=24,num=12).reshape(3,4)
nparray
```

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

### 3.3. Empty Array

```
[ ]
```

```
[ ]
```

```
[ ] nparray=np.empty((3,3),int)
    nparray
```

```
array([[ 11,  24,  39],
       [ 56,  75,  96],
       [119, 144, 171]])
```

### 3.4. Empty Like Some other array

```
[ ] nparray=np.empty_like(array1)
    nparray
```

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

### 3.5. Identity Matrix

```
[ ] nparray=np.identity(3)
    nparray
```

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

## 4. Arithmetic and Statistical Operations, Mathematical Operations, Bitwise Operators

### 4.1. Arithmetic Operation

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

```
[1 2 3 4 5]
[11 12 13 14 15]
```

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

```
[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]
```

## 4.2. Statistical and Mathematical 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))
```

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



## 4.3. Bitwise Operations

```
▶ 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)
```

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

```
[ ] ### You can get Binary Representation of Number #####
    print(np.binary_repr(10,8))
    resultarray=np.left_shift(10,2)
    print(resultarray)
    print(np.binary_repr(np.left_shift(10,2),8))
```

```
00001010
40
00101000
```

## 5.Copying and viewing arrays

### 5.1 Copy

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

[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]
```

### 5.2 View


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


[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]
```

## 6. Sorting

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

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

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

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

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

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

## 7. Searching

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

```
3
```

## 8. 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
```

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

## 9. Data Stacking

```
[ ] array1=np.array(np.arange(1,5).reshape(2,2))
    print(array1)
    array2=np.array(np.arange(11,15).reshape(2,2))
    print(array2)
```

```
[[1 2]
 [3 4]]
[[11 12]
 [13 14]]
```

```
[ ] newarray=np.stack([array1,array2],axis=0)
    print(newarray)
```

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

  [[11 12]
   [13 14]]]
```

```
[ ] newarray=np.stack([array1,array2],axis=1)
    print(newarray)
```

```
[[[ 1  2]
   [11 12]]

  [[ 3  4]
   [13 14]]]
```

## 10. Append

```
[ ] array1=np.arange(1,10).reshape(3,3)
    print(array1)
    array2=np.arange(21,30).reshape(3,3)
    print(array2)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
[[21 22 23]
 [24 25 26]
 [27 28 29]]
```

```
▶ np.append(array1,array2,axis=0)
```

```
👤 array([[ 1,  2,  3],
          [ 4,  5,  6],
          [ 7,  8,  9],
          [21, 22, 23],
          [24, 25, 26],
          [27, 28, 29]])
```

```
[ ] np.append(array1,array2,axis=1)
```

```
array([[ 1,  2,  3, 21, 22, 23],
       [ 4,  5,  6, 24, 25, 26],
       [ 7,  8,  9, 27, 28, 29]])
```

## 11. Concat

```
[ ] array1=np.arange(1,10).reshape(3,3)
print(array1)
array2=np.arange(21,30).reshape(3,3)
print(array2)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
[[21 22 23]
 [24 25 26]
 [27 28 29]]
```

```
[ ] np.concatenate((array1,array2),axis=0)
```

```
array([[ 1,  2,  3],
       [ 4,  5,  6],
       [ 7,  8,  9],
       [21, 22, 23],
       [24, 25, 26],
       [27, 28, 29]])
```

```
[ ] np.concatenate((array1,array2),axis=1)
```

```
array([[ 1,  2,  3, 21, 22, 23],
       [ 4,  5,  6, 24, 25, 26],
       [ 7,  8,  9, 27, 28, 29]])
```

## 12.Broadcasting

```
#Broadcasting
import numpy as np
x = np.array([1,2,3,4,5,7,8,9])
y = np.array([1,23])
x = x.reshape(4,2)
print("x=",x)
print("\n y=",y)
z = x+y
print("x+y=",z)
```

```
x= [[1 2]
     [3 4]
     [5 7]
     [8 9]]
```

```
y= [ 1 23]
x+y= [[ 2 25]
      [ 4 27]
      [ 6 30]
      [ 9 32]]
```

