Assignment No : 3

Name: Shinde Shruti Anil

Division : G(G3)

Roll No : 759

PRN No : 202201040110

Prepare/Take [datasets](#) for any real-life application. Read a [dataset](#) into an array. Perform the following operations on it:

1. **Perform all matrix operations**
2. **Horizontal and vertical stacking of Numpy Arrays**
3. **Custom sequence generation**
4. **Arithmetic and Statistical Operations, Mathematical Operations, Bitwise Operators**
5. **Copying and viewing arrays**
6. **Data Stacking, Searching, Sorting, Counting, Broadcasting**

Solution :

```
Help   All changes saved                                    Comment   Share   S

+ Code  + Text                                                      RAM
                                                                    Disk

[1]  import numpy as np

[3]  #Read dataset into array
     array=np.loadtxt("/content/sample_data/dataset6 (1).csv", delimiter =',',dtype=str)
     array

     array([['Bastman', 'Tot_Run', 'OUT', 'NO_Ball', 'Avg', 'StrikeRate'],
            ['V Kohli', '5426', '152', '4111', '35.69736842', '131.987351'],
            ['SK Raina', '5386', '160', '3916', '33.6625', '137.5383044'],
            ['RG Sharma', '4902', '161', '3742', '30.44720497', '130.9994655'],
            ['DA Warner', '4717', '114', '3292', '41.37719298', '143.2867558'],
            ['S Dhawan', '4601', '137', '3665', '33.58394161', '125.5388813'],
            ['CH Gayle', '4525', '110', '2972', '41.13636364', '152.2543742'],
            ['MS Dhoni', '4450', '118', '3206', '37.71186441', '138.8022458'],
            ['RV Uthappa', '4420', '156', '3381', '28.33333333',
             '130.7305531'],
            ['AB de Villiers', '4414', '104', '2902', '42.44230769',
             '152.1019986'],
            ['G Gambhir', '4219', '134', '3400', '31.48507463', '124.0882353'],
            ['AM Rahane', '3834', '117', '3133', '32.76923077', '122.3747207'],
            ['KD Karthik', '3669', '138', '2813', '26.58695652',
             '130.4301458'],
            ['SR Watson', '3590', '115', '2566', '31.2173913', '139.9064692']],
           dtype='<U14')
```

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

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

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

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

[13] #1]All matrix operations
     #matrix addition
     array3=array1+array2
     array3

     array([[12, 14, 16],
            [18, 20, 22],
            [24, 26, 28]])
```

```
[7]  #matrix substraction
     array3=array1-array2
     array3

     array([[-10, -10, -10],
            [-10, -10, -10],
            [-10, -10, -10]])

[8]  #matrix multiplication
     array3=array1*array2
     array3

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

[9]  #matrix division
     array3=array1/array2
     array3

     array([[0.09090909, 0.16666667, 0.23076923],
            [0.28571429, 0.33333333, 0.375     ],
            [0.41176471, 0.44444444, 0.47368421]])
```

```
[10]  #Transpose
      resultarray=np.transpose(array1)
      print(resultarray)
      resultarray=np.transpose(array2)
      print(resultarray)

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

```
  ▶  #Dot product
      resultarray=np.dot(array1,array2)
      print("",resultarray)
      #Cross product
      resultarray=np.cross(array1,array2)
      print("",resultarray)
```

```
  ⤷   [[ 90  96 102]
       [216 231 246]
       [342 366 390]]
      [[-10  20 -10]
       [-10  20 -10]
       [-10  20 -10]]
```

```
[14]  #2]
      #Horizontal and vertical stacking of Numpy Arrays
      resultarray=np.hstack((array1,array2))
      print("",resultarray)
      resultarray=np.vstack((array1,array2))
      print("",resultarray)

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

```
  ▶  #3]Custom sequence generation
      #Range
      nparray=np.arange(0,12,1).reshape(3,4)
      nparray
```

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

```python
#Linear 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.        ]])
```

```python
[21] #empty array
nparray=np.empty((3,3),int)
nparray
```

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

```python
[22] #empty like some other array
nparray=np.empty_like(array1)
nparray
```

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

```python
[23] #identity matrix
nparray=np.identity(3)
nparray
```

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

```python
#4
#Arithmetic and Statistical Operations, Mathematical Operations, Bitwise Operators
#Arithmetic operations

#addition
print(np.add(array1,array2))
#multiplication
print(np.multiply(array1,array2))
#division
print(np.divide(array1,array2))
#multiplication
print(np.multiply(array1,array2))
```

```
[[12 14 16]
 [18 20 22]
 [24 26 28]]
[[ 11  24  39]
 [ 56  75  96]
 [119 144 171]]
[[0.09090909 0.16666667 0.23076923]
 [0.28571429 0.33333333 0.375     ]
 [0.41176471 0.44444444 0.47368421]]
[[ 11  24  39]
```

```python
[44] # Statistical Operations, Mathematical Operations
     #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))
     #variance
     print(np.var(array1))
```

```
2.581988897471611
1
45
5.0
5.0
6.666666666666667
```

```python
[15] #Bitwise operation
     resultarray=np.bitwise_and(array1,array2)
     print(resultarray)
     resultarray=np.bitwise_or(array1,array2)
     print(resultarray)
     resultarray=np.left_shift(array1,array2)
     print(resultarray)
     resultarray=np.right_shift(array1,array2)
     print(resultarray)
```

```
[15]  [[1 0 1]
      [4 5 0]
      [1 0 1]]
     [[11 14 15]
      [14 15 22]
      [23 26 27]]
     [[   2048    8192   24576]
      [  65536  163840  393216]
      [ 917504 2097152 4718592]]
     [[0 0 0]
      [0 0 0]
      [0 0 0]]
```

```python
#5
#Copying and viewing arrays
#Copy
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]]
[[100 100 100]
 [  4   5   6]
 [  7   8   9]]
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

```
[48]  #View
      newarray=array1.view()
      print(newarray)
      #modification in original array
      array1[0]=100
      print(array1)
      print(newarray)

      [[100 100 100]
       [  4   5   6]
       [  7   8   9]]
      [[100 100 100]
       [  4   5   6]
       [  7   8   9]]
      [[100 100 100]
       [  4   5   6]
       [  7   8   9]]

  ▶   #6
      #Data Stacking, Searching, Sorting, Counting, Broadcasting
      #Data Staking
      newarray=np.stack([array1,array2],axis=0)
      print(newarray)
      newarray=np.stack([array1,array2],axis=1)
      print(newarray)
```

```
[50]
0s        [[[100 100 100]
           [  4   5   6]
           [  7   8   9]]

          [[ 11  12  13]
           [ 14  15  16]
           [ 17  18  19]]]
         [[[100 100 100]
           [ 11  12  13]]

          [[  4   5   6]
           [ 14  15  16]]

          [[  7   8   9]
           [ 17  18  19]]]
```

```
  ▶   #Searching
      np.sort(array1,axis=0)#Horizontal sort

  ⤷   array([[  4,   5,   6],
             [  7,   8,   9],
             [100, 100, 100]])
```

```
[54]  np.sort(array1,axis=1)#Vertical sort
0s
         array([[100, 100, 100],
                [  4,   5,   6],
                [  7,   8,   9]])
```

```
[59]  #Counting
0s      print(np.count_nonzero(array1))#Return all nonzero elements#
        print(np.nonzero(array1))#return index
        print(array1.size)#total elements

        9
        (array([0, 0, 0, 1, 1, 1, 2, 2, 2]), array([0, 1, 2, 0, 1, 2, 0, 1, 2]))
        9
```