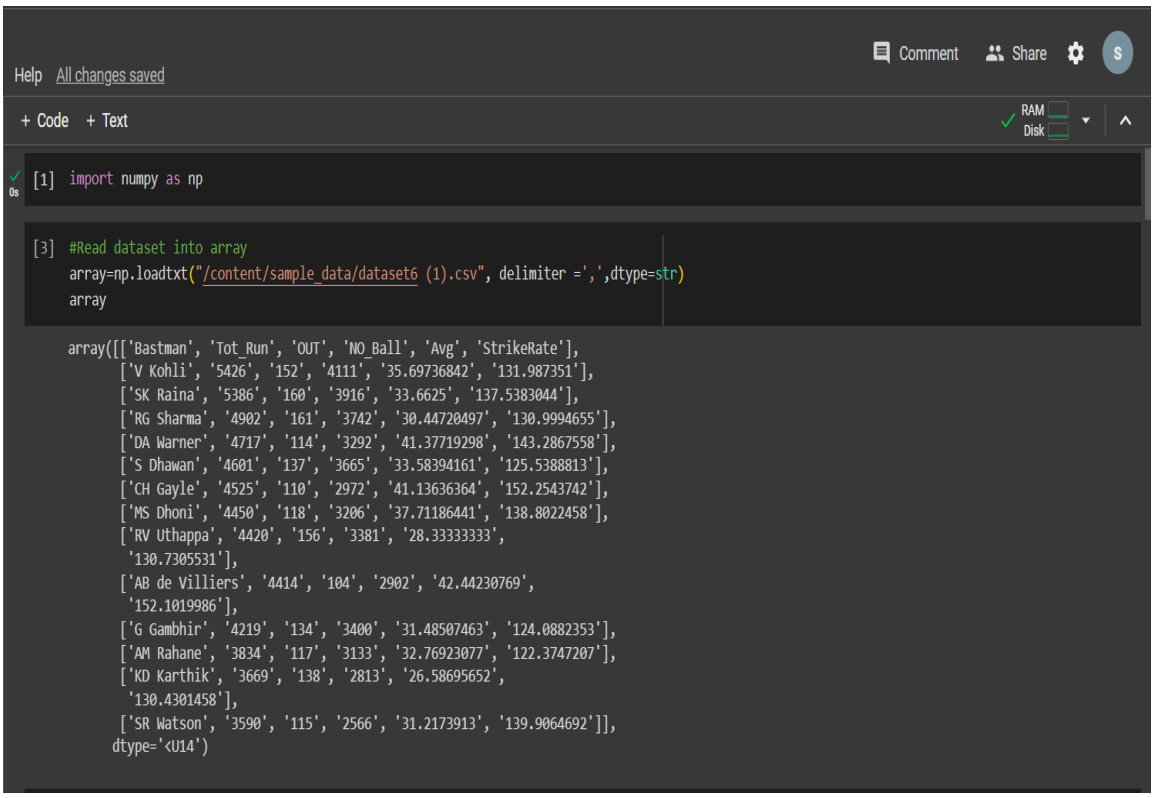
Assignment No : 3

Name : Ajay shinde
Prn: 202201050033
roll no.:758 G3

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 :



```
[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]])
```

```
#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.        ]])
```

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

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

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

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

```
[23] #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
#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]
```

```
[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
```

```
[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]]
```

```
0s #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)
```

```
[[[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
```

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

```
[59] #Counting
     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
```