# Practical No: 03

Rushikesh Sawant

Roll no: 849 H DIV

Prn:202201040134

## Questions:

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

Code:

```
import numpy as np

a=np.loadtxt('testmarks1.csv',delimiter=',',skiprows=1,dtype=float)

print('matrix of dataset 1:\n',a)

b=np.loadtxt('testmarks2.csv',delimiter=',',skiprows=1,dtype=float)

print('matrix of dataset 2:\n',b)

#sorting,searching and counting

print('Sort along column:\n',np.sort(a, axis = 0))

print(' searching of Indices of elements > 100 \n',np.where(a > 100))
```

```
print('counting elements greater than 800',np.count_nonzero(
a>800))
#mathematical
print('Original array:\n',a)
print('After rounding:\n',np.around(a))
print('The original array:\n' ,b)
print('The floor function array:\n', np.floor(b))
print('The ceil function array:\n',np.ceil(b))
#arithmetic
print("Addition of a & b is :\n",np.add(a,b))
print("Substraction of a & b is:\n",np.subtract(a,b))
print("Multiplication of a & b is:\n",np.multiply(a,b))
print('division f matrix a & b :\n',np.divide(a,b))

print("Transpose of matrix a is:\n",np.transpose(a))
print("Transpose of matrix b is:\n",np.transpose(b))
print("Mean of matrix a is:\n",np.mean(a,0))
print("Mean of matrix b is :\n",np.mean(b,0))
#statatical
print("Max of columns in a is:\n",np.max(a,0))
print("Min of rows  in b is:\n",np.min(b,1))
print("average in b is:\n",np.average(b))
#Horizontal & Vertical stacking of
result=np.hstack((a,b))
```

```
print("The horizontal stacking of the datasets is:\n",result)

result=np.vstack((a,b))
print("The vertical stacking of the datasets is:\n",result)

a1=np.arange(1,10)
b1=np.arange(10,100,10)
print('matrix a by custom sequence gen is :\n',a1)
print('matrix b by custom sequence gen is :\n',b1)
#Copying & Viewying of arrays
cparr=np.copy(a[0])
a[0,0]=172
print("a[0] is:",a[0])
print("copy of array is:",cparr)
viwarr=b[0].view()
b[0,0]=182
print("b[0]",b[0])
print("view array ",viwarr)
a1=np.arange(1,10)
b1=np.arange(10,100,10)
print('matrix a1 is :\n',a1)
print('matrix b1 is :\n',b1)
c=a1.reshape(3,3)
d=b1.reshape(3,3)
```

```
print('matrix a1 after reshape :\n',c)
print('matrix b1 after reshape :\n',d)
#bitwise
print('two elemets of a1 and b1-',a1[0],b1[0])
print('bitwise and of two elements: \n',np.bitwise_and(a1[0],b1[0]))
print('bitwise or of two elements: \n',np.bitwise_or(a1[0],b1[0]))
print('bitwise not of  element: \n',a1[0],'-',np.invert(a1[0]))
print('left shift of  element: \n',a1[0],'-',np.left_shift(a1[0],3))
print('right shift of  element: \n',a1[0],'-',np.right_shift(a1[0],4))

#broadcasting
a1=np.array([1,2,3,4,5])
c1=np.add(a,a1)
print('matrix a \n',a,'\n a1 \n',a1)
print('addition of a and a1 with broadcasting a1 :\n',c1)
```

Output:

```
= RESTART: E:\coding files\python files\activity no 03==
matrix of dataset 1:
 [[802.    43.47  28.52  28.98  27.89]
 [803.    42.24  28.16  28.16  25.63]
 [804.    39.24  26.16  26.16  26.16]
 [805.    40.9   26.03  27.27  25.65]
```

[806.  39.47  26.31  26.31  25.21]

[807.  41.68  25.63  27.79  25.46]

[808.  42.19  27.61  28.13  26.21]

[809.  44.75  28.35  29.83  28.21]

[810.  46.95  28.88  31.3  28.53]]

matrix of dataset 2:

[[802.  28.1  33.72  30.68  22.82]

[803.  26.16  31.39  28.2  22.53]

[804.  26.16  31.39  28.78  20.93]

[805.  26.1  31.32  28.22  20.82]

[806.  25.45  30.54  27.73  21.05]

[807.  26.16  31.39  28.01  20.51]

[808.  27.44  32.93  28.83  22.08]

[809.  28.63  34.35  31.03  22.68]

[810.  30.35  36.42  31.38  23.1 ]]

Sort along column:

[[802.  39.24  25.63  26.16  25.21]

[803.  39.47  26.03  26.31  25.46]

[804.  40.9  26.16  27.27  25.63]

[805.  41.68  26.31  27.79  25.65]

[806.  42.19  27.61  28.13  26.16]

[807.  42.24  28.16  28.16  26.21]

[808.  43.47  28.35  28.98  27.89]

[809.  44.75  28.52  29.83  28.21]

[810.  46.95  28.88  31.3  28.53]]

searching of Indices of elements > 100

(array([0, 1, 2, 3, 4, 5, 6, 7, 8], dtype=int64), array([0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64))

counting elements greater than 800 9

Original array:

[[802.  43.47  28.52  28.98  27.89]

[803.  42.24  28.16  28.16  25.63]

[804.  39.24  26.16  26.16  26.16]

[805.  40.9  26.03  27.27  25.65]

[806.  39.47  26.31  26.31  25.21]

[807.  41.68  25.63  27.79  25.46]

[808.  42.19  27.61  28.13  26.21]

[809.  44.75  28.35  29.83  28.21]

[810.  46.95  28.88  31.3  28.53]]

After rounding:

[[802.  43.  29.  29.  28.]

[803.  42.  28.  28.  26.]

[804.  39.  26.  26.  26.]

[805.  41.  26.  27.  26.]

[806.  39.  26.  26.  25.]

[807.  42.  26.  28.  25.]

[808.  42.  28.  28.  26.]

[809.  45.  28.  30.  28.]

[810.  47.  29.  31.  29.]]

The original array:

[[802.   28.1  33.72  30.68  22.82]

[803.   26.16  31.39  28.2   22.53]

[804.   26.16  31.39  28.78  20.93]

[805.   26.1   31.32  28.22  20.82]

[806.   25.45  30.54  27.73  21.05]

[807.   26.16  31.39  28.01  20.51]

[808.   27.44  32.93  28.83  22.08]

[809.   28.63  34.35  31.03  22.68]

[810.   30.35  36.42  31.38  23.1 ]]

The floor function array:

[[802. 28. 33. 30. 22.]

[803. 26. 31. 28. 22.]

[804. 26. 31. 28. 20.]

[805. 26. 31. 28. 20.]

[806. 25. 30. 27. 21.]

[807. 26. 31. 28. 20.]

[808. 27. 32. 28. 22.]

[809. 28. 34. 31. 22.]

[810. 30. 36. 31. 23.]]

The ceil function array:

[[802. 29. 34. 31. 23.]

[803. 27. 32. 29. 23.]

[804. 27. 32. 29. 21.]

[805. 27. 32. 29. 21.]

[806. 26. 31. 28. 22.]

[807. 27. 32. 29. 21.]

[808. 28. 33. 29. 23.]

[809. 29. 35. 32. 23.]

[810. 31. 37. 32. 24.]]

Addition of a & b is :

[[1604.  71.57  62.24  59.66  50.71]

[1606.  68.4  59.55  56.36  48.16]

[1608.  65.4  57.55  54.94  47.09]

[1610.  67.  57.35  55.49  46.47]

[1612.  64.92  56.85  54.04  46.26]

[1614.  67.84  57.02  55.8  45.97]

[1616.  69.63  60.54  56.96  48.29]

[1618.  73.38  62.7  60.86  50.89]

[1620.  77.3  65.3  62.68  51.63]]

Substraction of a & b is:

[[ 0.  15.37-5.2 -1.7  5.07]

[ 0.  16.08-3.23-0.04  3.1 ]

[ 0.  13.08-5.23-2.62  5.23]

[ 0.  14.8 -5.29-0.95  4.83]

[ 0.  14.02-4.23-1.42  4.16]

[ 0.  15.52-5.76-0.22  4.95]

[ 0.  14.75-5.32-0.7  4.13]

[ 0.  16.12-6.  -1.2  5.53]

[ 0.  16.6 -7.54-0.08  5.43]]

Multiplication of a & b is:

[[6.4320400e+05 1.2215070e+03 9.6169440e+02
8.8910640e+02 6.3644980e+02]

[6.4480900e+05 1.1049984e+03 8.8394240e+02
7.9411200e+02 5.7744390e+02]

[6.4641600e+05 1.0265184e+03 8.2116240e+02
7.5288480e+02 5.4752880e+02]

[6.4802500e+05 1.0674900e+03 8.1525960e+02
7.6955940e+02 5.3403300e+02]

[6.4963600e+05 1.0045115e+03 8.0350740e+02
7.2957630e+02 5.3067050e+02]

[6.5124900e+05 1.0903488e+03 8.0452570e+02
7.7839790e+02 5.2218460e+02]

[6.5286400e+05 1.1576936e+03 9.0919730e+02
8.1098790e+02 5.7871680e+02]

[6.5448100e+05 1.2811925e+03 9.7382250e+02
9.2562490e+02 6.3980280e+02]

[6.5610000e+05 1.4249325e+03 1.0518096e+03
9.8219400e+02 6.5904300e+02]]

division f matrix a & b :

[[1. 1.54697509 0.84578885 0.94458931
1.22217353]

[1. 1.6146789 0.89710099 0.99858156
1.13759432]

[1. 1.5 0.83338643 0.90896456 1.24988055]

[1. 1.56704981 0.83109834 0.96633593 1.23198847]

[1. 1.55088409 0.86149312 0.94879192 1.1976247 ]

[1. 1.59327217 0.81650207 0.99214566 1.24134569]

[1. 1.53753644 0.83844519 0.97571974 1.1870471 ]

[1. 1.56304576 0.82532751 0.96132775 1.24382716]

[1. 1.54695222 0.7929709 0.99745061 1.23506494]]

Transpose of matrix a is:

[[802. 803. 804. 805. 806. 807. 808. 809. 810. ]

[ 43.47 42.24 39.24 40.9 39.47 41.68 42.19 44.75 46.95]

[ 28.52 28.16 26.16 26.03 26.31 25.63 27.61 28.35 28.88]

[ 28.98 28.16 26.16 27.27 26.31 27.79 28.13 29.83 31.3 ]

[ 27.89 25.63 26.16 25.65 25.21 25.46 26.21 28.21 28.53]]

Transpose of matrix b is:

[[802. 803. 804. 805. 806. 807. 808. 809. 810. ]

[ 28.1 26.16 26.16 26.1 25.45 26.16 27.44 28.63 30.35]

[ 33.72 31.39 31.39 31.32 30.54 31.39 32.93 34.35 36.42]

[ 30.68 28.2 28.78 28.22 27.73 28.01 28.83 31.03 31.38]

[ 22.82 22.53 20.93 20.82 21.05 20.51 22.08 22.68 23.1 ]]

Mean of matrix a is:

[806. 42.32111111 27.29444444 28.21444444 26.55 ]

Mean of matrix b is :

[806. 27.17222222 32.60555556 29.20666667 21.83555556]

Max of columns in a is:

[810. 46.95 28.88 31.3 28.53]

Min of rows in b is:

[22.82 22.53 20.93 20.82 21.05 20.51 22.08 22.68 23.1 ]

average in b is:

183.36399999999998

The horizontal stacking of the datasets is:

[[802. 43.47 28.52 28.98 27.89 802. 28.1 33.72 30.68 22.82]

[803. 42.24 28.16 28.16 25.63 803. 26.16 31.39 28.2 22.53]

[804. 39.24 26.16 26.16 26.16 804. 26.16 31.39 28.78 20.93]

[805. 40.9 26.03 27.27 25.65 805. 26.1 31.32 28.22 20.82]

[806. 39.47 26.31 26.31 25.21 806. 25.45 30.54 27.73 21.05]

[807. 41.68 25.63 27.79 25.46 807. 26.16 31.39 28.01 20.51]

[808. 42.19 27.61 28.13 26.21 808. 27.44 32.93 28.83 22.08]

[809. 44.75 28.35 29.83 28.21 809. 28.63 34.35 31.03 22.68]

[810.  46.95  28.88  31.3  28.53 810.  30.35  36.42 31.38  23.1 ]]

The vertical stacking of the datasets is:

[[802.  43.47  28.52  28.98  27.89]

[803.  42.24  28.16  28.16  25.63]

[804.  39.24  26.16  26.16  26.16]

[805.  40.9  26.03  27.27  25.65]

[806.  39.47  26.31  26.31  25.21]

[807.  41.68  25.63  27.79  25.46]

[808.  42.19  27.61  28.13  26.21]

[809.  44.75  28.35  29.83  28.21]

[810.  46.95  28.88  31.3  28.53]

[802.  28.1  33.72  30.68  22.82]

[803.  26.16  31.39  28.2  22.53]

[804.  26.16  31.39  28.78  20.93]

[805.  26.1  31.32  28.22  20.82]

[806.  25.45  30.54  27.73  21.05]

[807.  26.16  31.39  28.01  20.51]

[808.  27.44  32.93  28.83  22.08]

[809.  28.63  34.35  31.03  22.68]

[810.  30.35  36.42  31.38  23.1 ]]

matrix a by custom sequence gen is :

[1 2 3 4 5 6 7 8 9]

matrix b by custom sequence gen is :

[10 20 30 40 50 60 70 80 90]

a[0] is: [172.  43.47  28.52  28.98  27.89]

copy of array is: [802.  43.47  28.52  28.98  27.89]

b[0] [182.  28.1  33.72  30.68  22.82]

view array  [182.  28.1  33.72  30.68  22.82]

matrix a1 is :

[1 2 3 4 5 6 7 8 9]

matrix b1 is :

[10 20 30 40 50 60 70 80 90]

matrix a1 after reshape :

[[1 2 3]

[4 5 6]

[7 8 9]]

matrix b1 after reshape :

[[10 20 30]

[40 50 60]

[70 80 90]]

two elemets of a1 and b1- 1 10

bitwise and of two elements:

0

bitwise or of two elements:

11

bitwise not of element:

1--2

left shift of element:

1- 8

right shift of element:

1- 0

matrix a

[[172. 43.47 28.52 28.98 27.89]

[803. 42.24 28.16 28.16 25.63]

[804. 39.24 26.16 26.16 26.16]

[805. 40.9 26.03 27.27 25.65]

[806. 39.47 26.31 26.31 25.21]

[807. 41.68 25.63 27.79 25.46]

[808. 42.19 27.61 28.13 26.21]

[809.  44.75  28.35  29.83  28.21]

[810.  46.95  28.88  31.3  28.53]]

a1

[1 2 3 4 5]

addition of a and a1 with broadcasting a1 :

[[173.  45.47  31.52  32.98  32.89]

[804.  44.24  31.16  32.16  30.63]

[805.  41.24  29.16  30.16  31.16]

[806.  42.9  29.03  31.27  30.65]

[807.  41.47  29.31  30.31  30.21]

[808.  43.68  28.63  31.79  30.46]

[809.  44.19  30.61  32.13  31.21]

[810.  46.75  31.35  33.83  33.21]

[811.  48.95  31.88  35.3  33.53]]