Practical No: 03

Shivshankar J Ghyar

Roll no: 836 H DIV

Prn:202201040031

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("Testmarks1 is :")

print(a)


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

print("Testmarks2 is:")

print(b)
```

```
#1 Arithematic operation
#1 Addition of testmarks
print("\n Addition of Testmarks1 & Testmarks2 is :")
print(np.add(a,b))

#2 Substraction of a & b
print("Substraction of Testmarks1 & Testmarks2 is:")
print(np.subtract(a,b))

#3 Multiplication of a & b
print("\n Multiplication of testmarks1 & testmarks2 is:")
print(np.multiply(a,b))

#4 Transpose of matrices
print("\n Transpose of matrix testmarks1 is:")
print(np.transpose(a))
print("\n Transpose of matrix testmarks2 is:")
print(np.transpose(b))

#2 stastical operations
#1 Mean of matrices
print("Mean of matrix a is:")
print(np.mean(a,0))

print("Mean of matrix b is :")
print(np.mean(b,0))
```

```
#2 Max of columns in a
print("Max of columns in a is:")
c=np.max(a,0)
print(c)

#3 standard Deviation
sd=np.std(a[0])
print("Standard deviation is:")
print(sd)



#Horizontal & Vertical stacking of arrays
result=np.hstack((a,b))
print("The horizontal stacking of the datasets is:")
print(result)

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

#Custom Sequence generation
seq_gen=[]
for i in range(len(a)):
```

```
    seq_gen.append(a[i])
    seq_gen.append(b[i])
print(seq_gen)

# Bitwise Operators
#Bitwise and
print("a[0][2]",a[0][2])
print("a[0][3]",a[0][3])

bitwise_and=np.bitwise_and(int(a[0][3]), int(a[0][2]))
print("Bitwise and of ",a[0][2],"&",a[0][3],"is:")
print(bitwise_and)

#Bitwise or
bitwise_or=np.bitwise_or(int(a[0][3]),int(a[0][2]))
print("Bitwise or: ",bitwise_or)



#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 operator
#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))
```

```
#Mathematical Operations
#1 Square root
sqrt=np.sqrt(a)
print("Square root of testmarks1 elements is:")
print(sqrt)

#2 Exponential
exp=np.exp(a)
print("Exponential od testmarks1 elements is:")
print(exp)

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))


#Stacking along axis=-1
stacked = np.stack((a,b),axis=-1)
print("Stacking along axis=-1 i.e.last dimension is:")
print(stacked)

#Broadcasting
print("Testmarks1 804 ")
```

```
print("Before Broadcasting:")
print(a[3])
mul_arr=a[3]*3
print("After broadcasting:")
print(mul_arr)

#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))

#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:

```
*** Remote Interpreter Reinitialized ***
Testmarks1 is :
[[801.    43.05  27.79  28.7   27.79]
 [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]]

Testmarks2 is:

[[801. 28.48 34.18 30.56 22.23]

[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]]

Addition of Testmarks1 & Testmarks2 is :

[[1602. 71.53 61.97 59.26 50.02]

[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 Testmarks1 & Testmarks2 is:

[[0. 14.57-6.39-1.86 5.56]

[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 testmarks1 & testmarks2 is:

[[6.4160100e+05 1.2260640e+03 9.4986220e+02 8.7707200e+02
6.1777170e+02]

[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]]

Transpose of matrix testmarks1 is:

[[801. 802. 803. 804. 805. 806. 807. 808. 809. 810.]

[43.05 43.47 42.24 39.24 40.9 39.47 41.68 42.19 44.75 46.95]

[27.79 28.52 28.16 26.16 26.03 26.31 25.63 27.61 28.35 28.88]

[28.7 28.98 28.16 26.16 27.27 26.31 27.79 28.13 29.83 31.3]

[27.79 27.89 25.63 26.16 25.65 25.21 25.46 26.21 28.21 28.53]]

Transpose of matrix testmarks2 is:

[[801. 802. 803. 804. 805. 806. 807. 808. 809. 810.]

[28.48 28.1 26.16 26.16 26.1 25.45 26.16 27.44 28.63 30.35]

[34.18 33.72 31.39 31.39 31.32 30.54 31.39 32.93 34.35 36.42]

[30.56 30.68 28.2 28.78 28.22 27.73 28.01 28.83 31.03 31.38]

[22.23 22.82 22.53 20.93 20.82 21.05 20.51 22.08 22.68 23.1]]

Mean of matrix a is:

[805.5 42.394 27.344 28.263 26.674]

Mean of matrix b is :

[805.5 27.303 32.763 29.342 21.875]

Max of columns in a is:

[810. 46.95 28.88 31.3 28.53]

Standard deviation is:

307.72170639069327

The horizontal stacking of the datasets is:

[[801. 43.05 27.79 28.7 27.79 801. 28.48 34.18 30.56 22.23]
 [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:

[[801. 43.05 27.79 28.7 27.79]
 [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]

[801. 28.48 34.18 30.56 22.23]

[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]]

[array([801. , 43.05, 27.79, 28.7 , 27.79]), array([801. , 28.48, 34.18, 30.56, 22.23]), array([802. , 43.47, 28.52, 28.98, 27.89]), array([802. , 28.1 , 33.72, 30.68, 22.82]), array([803. , 42.24, 28.16, 28.16, 25.63]), array([803. , 26.16, 31.39, 28.2 , 22.53]), array([804. , 39.24, 26.16, 26.16, 26.16]), array([804. , 26.16, 31.39, 28.78, 20.93]), array([805. , 40.9 , 26.03, 27.27, 25.65]), array([805. , 26.1 , 31.32, 28.22, 20.82]), array([806. , 39.47, 26.31, 26.31, 25.21]), array([806. , 25.45, 30.54, 27.73, 21.05]), array([807. , 41.68, 25.63, 27.79, 25.46]), array([807. , 26.16, 31.39, 28.01, 20.51]), array([808. , 42.19, 27.61, 28.13, 26.21]), array([808. , 27.44, 32.93, 28.83, 22.08]), array([809. , 44.75, 28.35, 29.83, 28.21]), array([809. , 28.63, 34.35, 31.03, 22.68]), array([810. , 46.95, 28.88, 31.3 , 28.53]), array([810. , 30.35, 36.42, 31.38, 23.1])]

a[0][2] 27.79

a[0][3] 28.7

Bitwise and of 27.79 & 28.7 is:

24

Bitwise or: 31

a[0] is: [172. 43.05 27.79 28.7 27.79]

copy of array is: [801. 43.05 27.79 28.7 27.79]

b[0] [182. 28.48 34.18 30.56 22.23]

```
view array  [182.    28.48  34.18  30.56  22.23]
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
Square root of testmarks1 elements is:
[[13.11487705  6.56124988  5.27162214  5.35723809  5.27162214]
```

[28.31960452 6.59317829 5.34041197 5.38330753 5.28109837]

[28.33725463 6.49923072 5.30659966 5.30659966 5.06260802]

[28.35489376 6.26418391 5.11468474 5.11468474 5.11468474]

[28.37252192 6.39531078 5.10196041 5.22206856 5.0645829]

[28.39013913 6.28251542 5.12932744 5.12932744 5.02095608]

[28.40774542 6.45600496 5.06260802 5.27162214 5.04579032]

[28.42534081 6.49538298 5.25452186 5.30377224 5.11957029]

[28.44292531 6.68954408 5.3244718   5.46168472 5.31130869]

[28.46049894 6.85200701 5.37401154 5.59464029 5.34134814]]

C:\Users\Shivshankar\Downloads\python shiv\practical3 testmarks1.py:124: RuntimeWarning: overflow encountered in exp

exp=np.exp(a)

Exponential od testmarks1 elements is:

[[4.99632738e+74 4.97024098e+18 1.17231319e+12 2.91240408e+12

1.17231319e+12]

[inf 7.56451570e+18 2.43264437e+12 3.85348866e+12

1.29560645e+12]

[inf 2.21105179e+18 1.69719839e+12 1.69719839e+12

1.35197161e+11]

[inf 1.10081787e+17 2.29690824e+11 2.29690824e+11

2.29690824e+11]

[inf 5.78954335e+17 2.01690463e+11 6.96964281e+11

1.37928325e+11]

[inf 1.38548938e+17 2.66862665e+11 2.66862665e+11

8.88308645e+10]

[inf 1.26297282e+18 1.35197161e+11 1.17231319e+12

1.14061088e+11]

[inf 2.10321752e+18 9.79198288e+11 1.64703859e+12
2.41467325e+11]
[inf 2.72068377e+19 2.05233647e+12 9.01580262e+12
1.78421561e+12]
[inf 2.45542077e+20 3.48678073e+12 3.92118456e+13
2.45709285e+12]]
Original array:
[[172. 43.05 27.79 28.7 27.79]
[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:
[[172. 43. 28. 29. 28.]
[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:

[[182.   28.48 34.18 30.56 22.23]

[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:

[[182. 28. 34. 30. 22.]

[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:

[[182. 29. 35. 31. 23.]

[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.]]

Stacking along axis=-1 i.e.last dimension is:

[[[172. 182.]

[43.05 28.48]

[27.79 34.18]

[28.7   30.56]

[27.79 22.23]]

[[802. 802.]

[43.47 28.1]

[28.52 33.72]

[28.98 30.68]

[27.89 22.82]]

[[803. 803.]

[42.24 26.16]

[28.16 31.39]

[28.16 28.2]

[25.63 22.53]]

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

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

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

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

[[808. 808.]

[42.19 27.44]

[27.61 32.93]

[28.13 28.83]

[26.21 22.08]]

[[809. 809.]

[44.75 28.63]

[28.35 34.35]

[29.83 31.03]

[28.21 22.68]]

[[810. 810.]

[46.95 30.35]

[28.88 36.42]

[31.3   31.38]

[28.53 23.1]]]

Testmarks1 804

Before Broadcasting:

[804.   39.24 26.16 26.16 26.16]

After broadcasting:

[2412.   117.72   78.48   78.48   78.48]

Sort along column:

[[172.   39.24 25.63 26.16 25.21]

[802.   39.47 26.03 26.31 25.46]

[803.   40.9   26.16 27.27 25.63]

[804. 41.68 26.31 27.79 25.65]

[805. 42.19 27.61 28.13 26.16]

[806. 42.24 27.79 28.16 26.21]

[807. 43.05 28.16 28.7   27.79]

[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, 9], dtype=int64), array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64))

counting elements greater than 800 9

matrix a

[[172. 43.05 27.79 28.7   27.79]

[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]]

a1

[1 2 3 4 5]

addition of a and a1 with broadcasting a1 :

[[173. 45.05 30.79 32.7   32.79]

[803. 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]]
>>>
```