EDS Practical - 3

Name: Tushar Najawan

Batch : B4

Roll no. : 266

PRN: 202201070092

Aim : 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**

CSV file:

testmarks1.csv

	A	B	C	D	E
1	RollNo	EDS	SON	DT	ET
2	801	43.05	27.79	28.7	27.79
3	802	43.47	28.52	28.98	27.89
4	803	42.24	28.16	28.16	25.63
5	804	39.24	26.16	26.16	26.16
6	805	40.9	26.03	27.27	25.65
7	806	39.47	26.31	26.31	25.21
8	807	41.68	25.63	27.79	25.46
9	808	42.19	27.61	28.13	26.21
10	809	44.75	28.35	29.83	28.21
11	810	46.95	28.88	31.3	28.53

Code:

```
import numpy as np
import pandas as pd

# Read the CSV file into a Pandas DataFrame
filename = "/content/drive/MyDrive/testmarks1.csv"
data = pd.read_csv(filename)
print("Read csv file")
print(data)
print()

# Convert the DataFrame to a Numpy array
array_data  =  data.to_numpy()
print("Read the dataset into an array")
print(array_data)
print()

# 1. Perform all matrix operations
matrix_operations = np.array(array_data[:, 1:], dtype=float)
print("Matrix Operations:")
print(matrix_operations)
print()

# 2. Horizontal and vertical stacking of Numpy Arrays
stacked_horizontal = np.hstack((matrix_operations, matrix_operations))
stacked_vertical = np.vstack((matrix_operations, matrix_operations))
print("Horizontal Stacking:")
print(stacked_horizontal)
print()
print("Vertical Stacking:")
print(stacked_vertical)
print()

# 3. Custom sequence generation
sequence = np.arange(1, 11)
print("Custom Sequence Generation:")
print(sequence)
print()

# 4. Arithmetic and Statistical Operations, Mathematical Operations,
Bitwise Operators
#  Arithmetic  Operations
addition = matrix_operations + 5
subtraction = matrix_operations - 2
multiplication = matrix_operations * 3
```

```
division = matrix_operations / 4

# Statistical Operations
mean = np.mean(matrix_operations)
median = np.median(matrix_operations)
std_dev = np.std(matrix_operations)

# Mathematical Operations
square_root = np.sqrt(matrix_operations)
exponential = np.exp(matrix_operations)

# Bitwise Operators
bitwise_and = matrix_operations.astype(int) & 2
bitwise_or = matrix_operations.astype(int) | 2
bitwise_xor = matrix_operations.astype(int) ^ 2

print("Arithmetic and Statistical Operations:")
print("Addition:")
print(addition)
print()
print("Subtraction:")
print(subtraction)
print()
print("Multiplication:")
print(multiplication)
print()
print("Division:")
print(division)
print()
print("Statistical Operations:")
print("Mean:",  mean)
print("Median:", median)
print("Standard Deviation:", std_dev)
print()
print("Mathematical Operations:")
print("Square Root:")
print(square_root)
print()
print("Exponential:")
print(exponential)
print()
print("Bitwise Operators:")
print("Bitwise AND:")
print(bitwise_and)
print()
print("Bitwise OR:")
print(bitwise_or)
print()
```

```
print("Bitwise XOR:")
print(bitwise_xor)
print()

# 5. Copying and viewing arrays
copied_array = matrix_operations.copy()
view_array = matrix_operations.view()

print("Copying and Viewing Arrays:")
print("Copied Array:")
print(copied_array)
print()
print("View Array:")
print(view_array)
print()

# 6. Data Stacking, Searching, Sorting, Counting, Broadcasting
# Data Stacking
stacked_data = np.hstack((matrix_operations, copied_array))

# Searching
indices = np.where(matrix_operations == 28.16)

# Sorting
sorted_data = np.sort(matrix_operations)

# Counting
count = np.count_nonzero(matrix_operations > 27.5)

# Broadcasting
broadcasted_data = matrix_operations + np.array([1, 2, 3, 4])

print("Data Stacking, Searching, Sorting, Counting, Broadcasting:")
print("Stacked Data:")
print(stacked_data)
print()
print("Indices of 28.16:")
print(indices)
print()
print("Sorted Data:")
print(sorted_data)
print()
print("Count:", count)
print()
print("Broadcasted Data:")
print(broadcasted_data)
```

OUTPUT:

```
Read csv file
   RollNo    EDS    SON     DT     ET
0     801  43.05  27.79  28.70  27.79
1     802  43.47  28.52  28.98  27.89
2     803  42.24  28.16  28.16  25.63
3     804  39.24  26.16  26.16  26.16
4     805  40.90  26.03  27.27  25.65
5     806  39.47  26.31  26.31  25.21
6     807  41.68  25.63  27.79  25.46
7     808  42.19  27.61  28.13  26.21
8     809  44.75  28.35  29.83  28.21
9     810  46.95  28.88  31.30  28.53

Read the dataset into an array
[[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]]

Matrix Operations:
[[43.05 27.79 28.7  27.79]
 [43.47 28.52 28.98 27.89]
 [42.24 28.16 28.16 25.63]
 [39.24 26.16 26.16 26.16]
 [40.9  26.03 27.27 25.65]
 [39.47 26.31 26.31 25.21]
 [41.68 25.63 27.79 25.46]
 [42.19 27.61 28.13 26.21]
 [44.75 28.35 29.83 28.21]
 [46.95 28.88 31.3  28.53]]

Horizontal Stacking:
[[43.05 27.79 28.7  27.79 43.05 27.79 28.7  27.79]
 [43.47 28.52 28.98 27.89 43.47 28.52 28.98 27.89]
 [42.24 28.16 28.16 25.63 42.24 28.16 28.16 25.63]
 [39.24 26.16 26.16 26.16 39.24 26.16 26.16 26.16]
 [40.9  26.03 27.27 25.65 40.9  26.03 27.27 25.65]
 [39.47 26.31 26.31 25.21 39.47 26.31 26.31 25.21]
 [41.68 25.63 27.79 25.46 41.68 25.63 27.79 25.46]
 [42.19 27.61 28.13 26.21 42.19 27.61 28.13 26.21]
 [44.75 28.35 29.83 28.21 44.75 28.35 29.83 28.21]
 [46.95 28.88 31.3  28.53 46.95 28.88 31.3  28.53]]

Vertical Stacking:
[[43.05 27.79 28.7  27.79]
 [43.47 28.52 28.98 27.89]
 [42.24 28.16 28.16 25.63]
 [39.24 26.16 26.16 26.16]
 [40.9  26.03 27.27 25.65]
 [39.47 26.31 26.31 25.21]
 [41.68 25.63 27.79 25.46]
 [42.19 27.61 28.13 26.21]
```

```
 [44.75 28.35 29.83 28.21]
 [46.95 28.88 31.3  28.53]
 [43.05 27.79 28.7  27.79]
 [43.47 28.52 28.98 27.89]
 [42.24 28.16 28.16 25.63]
 [39.24 26.16 26.16 26.16]
 [40.9  26.03 27.27 25.65]
 [39.47 26.31 26.31 25.21]
 [41.68 25.63 27.79 25.46]
 [42.19 27.61 28.13 26.21]
 [44.75 28.35 29.83 28.21]
 [46.95 28.88 31.3  28.53]]

Custom Sequence Generation:
[ 1  2  3  4  5  6  7  8  9 10]

Arithmetic and Statistical Operations:
Addition:
[[48.05 32.79 33.7  32.79]
 [48.47 33.52 33.98 32.89]
 [47.24 33.16 33.16 30.63]
 [44.24 31.16 31.16 31.16]
 [45.9  31.03 32.27 30.65]
 [44.47 31.31 31.31 30.21]
 [46.68 30.63 32.79 30.46]
 [47.19 32.61 33.13 31.21]
 [49.75 33.35 34.83 33.21]
 [51.95 33.88 36.3  33.53]]

Subtraction:
[[41.05 25.79 26.7  25.79]
 [41.47 26.52 26.98 25.89]
 [40.24 26.16 26.16 23.63]
 [37.24 24.16 24.16 24.16]
 [38.9  24.03 25.27 23.65]
 [37.47 24.31 24.31 23.21]
 [39.68 23.63 25.79 23.46]
 [40.19 25.61 26.13 24.21]
 [42.75 26.35 27.83 26.21]
 [44.95 26.88 29.3  26.53]]

Multiplication:
[[129.15  83.37  86.1   83.37]
 [130.41  85.56  86.94  83.67]
 [126.72  84.48  84.48  76.89]
 [117.72  78.48  78.48  78.48]
 [122.7   78.09  81.81  76.95]
 [118.41  78.93  78.93  75.63]
 [125.04  76.89  83.37  76.38]
 [126.57  82.83  84.39  78.63]
 [134.25  85.05  89.49  84.63]
 [140.85  86.64  93.9   85.59]]


Division:
[[10.7625  6.9475  7.175   6.9475]
 [10.8675  7.13    7.245   6.9725]
 [10.56    7.04    7.04    6.4075]
```

```
 [ 9.81    6.54    6.54    6.54  ]
 [10.225   6.5075  6.8175  6.4125]
 [ 9.8675  6.5775  6.5775  6.3025]
 [10.42    6.4075  6.9475  6.365 ]
 [10.5475  6.9025  7.0325  6.5525]
 [11.1875  7.0875  7.4575  7.0525]
 [11.7375  7.22    7.825   7.1325]]

Statistical Operations:
Mean: 31.16875
Median: 28.16
Standard Deviation: 6.692269864365901

Mathematical Operations:
Square Root:
[[6.56124988 5.27162214 5.35723809 5.27162214]
 [6.59317829 5.34041197 5.38330753 5.28109837]
 [6.49923072 5.30659966 5.30659966 5.06260802]
 [6.26418391 5.11468474 5.11468474 5.11468474]
 [6.39531078 5.10196041 5.22206856 5.0645829 ]
 [6.28251542 5.12932744 5.12932744 5.02095608]
 [6.45600496 5.06260802 5.27162214 5.04579032]
 [6.49538298 5.25452186 5.30377224 5.11957029]
 [6.68954408 5.3244718  5.46168472 5.31130869]
 [6.85200701 5.37401154 5.59464029 5.34134814]]

Exponential:
[[4.97024098e+18 1.17231319e+12 2.91240408e+12 1.17231319e+12]
 [7.56451570e+18 2.43264437e+12 3.85348866e+12 1.29560645e+12]
 [2.21105179e+18 1.69719839e+12 1.69719839e+12 1.35197161e+11]
 [1.10081787e+17 2.29690824e+11 2.29690824e+11 2.29690824e+11]
 [5.78954335e+17 2.01690463e+11 6.96964281e+11 1.37928325e+11]
 [1.38548938e+17 2.66862665e+11 2.66862665e+11 8.88308645e+10]
 [1.26297282e+18 1.35197161e+11 1.17231319e+12 1.14061088e+11]
 [2.10321752e+18 9.79198288e+11 1.64703859e+12 2.41467325e+11]
 [2.72068377e+19 2.05233647e+12 9.01580262e+12 1.78421561e+12]
 [2.45542077e+20 3.48678073e+12 3.92118456e+13 2.45709285e+12]]

Bitwise Operators:
Bitwise AND:
[[2 2 0 2]
 [2 0 0 2]
 [2 0 0 0]
 [2 2 2 2]
 [0 2 2 0]
 [2 2 2 0]
 [0 0 2 0]
 [2 2 0 2]
 [0 0 0 0]
 [2 0 2 0]]



Bitwise OR:
[[43 27 30 27]
 [43 30 30 27]
 [42 30 30 27]
```

```
 [39 26 26 26]
 [42 26 27 27]
 [39 26 26 27]
 [43 27 27 27]
 [42 27 30 26]
 [46 30 31 30]
 [46 30 31 30]]

Bitwise XOR:
[[41 25 30 25]
 [41 30 30 25]
 [40 30 30 27]
 [37 24 24 24]
 [42 24 25 27]
 [37 24 24 27]
 [43 27 25 27]
 [40 25 30 24]
 [46 30 31 30]
 [44 30 29 30]]

Copying and Viewing Arrays:
Copied Array:
[[43.05 27.79 28.7  27.79]
 [43.47 28.52 28.98 27.89]
 [42.24 28.16 28.16 25.63]
 [39.24 26.16 26.16 26.16]
 [40.9  26.03 27.27 25.65]
 [39.47 26.31 26.31 25.21]
 [41.68 25.63 27.79 25.46]
 [42.19 27.61 28.13 26.21]
 [44.75 28.35 29.83 28.21]
 [46.95 28.88 31.3  28.53]]

View Array:
[[43.05 27.79 28.7  27.79]
 [43.47 28.52 28.98 27.89]
 [42.24 28.16 28.16 25.63]
 [39.24 26.16 26.16 26.16]
 [40.9  26.03 27.27 25.65]
 [39.47 26.31 26.31 25.21]
 [41.68 25.63 27.79 25.46]
 [42.19 27.61 28.13 26.21]
 [44.75 28.35 29.83 28.21]
 [46.95 28.88 31.3  28.53]]

Data Stacking, Searching, Sorting, Counting, Broadcasting:
Stacked Data:
[[43.05 27.79 28.7  27.79 43.05 27.79 28.7  27.79]
 [43.47 28.52 28.98 27.89 43.47 28.52 28.98 27.89]
```

```
 [42.24 28.16 28.16 25.63 42.24 28.16 28.16 25.63]
 [39.24 26.16 26.16 26.16 39.24 26.16 26.16 26.16]
 [40.9  26.03 27.27 25.65 40.9  26.03 27.27 25.65]
 [39.47 26.31 26.31 25.21 39.47 26.31 26.31 25.21]
 [41.68 25.63 27.79 25.46 41.68 25.63 27.79 25.46]
 [42.19 27.61 28.13 26.21 42.19 27.61 28.13 26.21]
 [44.75 28.35 29.83 28.21 44.75 28.35 29.83 28.21]
 [46.95 28.88 31.3  28.53 46.95 28.88 31.3  28.53]]

Indices of 28.16:
(array([2, 2]), array([1, 2]))

Sorted Data:
[[27.79 27.79 28.7  43.05]
 [27.89 28.52 28.98 43.47]
 [25.63 28.16 28.16 42.24]
 [26.16 26.16 26.16 39.24]
 [25.65 26.03 27.27 40.9 ]
 [25.21 26.31 26.31 39.47]
 [25.46 25.63 27.79 41.68]
 [26.21 27.61 28.13 42.19]
 [28.21 28.35 29.83 44.75]
 [28.53 28.88 31.3  46.95]]

Count: 27

Broadcasted Data:
[[44.05 29.79 31.7  31.79]
 [44.47 30.52 31.98 31.89]
 [43.24 30.16 31.16 29.63]
 [40.24 28.16 29.16 30.16]
 [41.9  28.03 30.27 29.65]
 [40.47 28.31 29.31 29.21]
 [42.68 27.63 30.79 29.46]
 [43.19 29.61 31.13 30.21]
 [45.75 30.35 32.83 32.21]
 [47.95 30.88 34.3  32.53]]
```