

Name : Shashank patokar

RollNO – 449

D3

```
import numpy as np
```

```
# Load the datasets into arrays
```

```
data1 = np.genfromtxt('testmarks1.csv', delimiter='\t', skip_header=1)
```

```
data2 = np.genfromtxt('testmarks2.csv', delimiter='\t', skip_header=1)
```

```
# Matrix Operations #
```

```
Addition
```

```
matrix_sum = data1 + data2
```

```
# Subtraction
```

```
matrix_diff = data1 - data2
```

```
# Multiplication
```

```
matrix_product = np.matmul(data1[:, 1:], data2[:, 1:].T)
```

```
# Transpose
```

```
matrix_transpose = data1.T
```

```
# Horizontal and Vertical Stacking
```

```
horizontal_stack = np.hstack((data1, data2))
```

```
vertical_stack = np.vstack((data1, data2))
```

```
# Custom Sequence Generation
```

```
custom_sequence = np.arange(10, 51, 10)
```

```
# Arithmetic and Statistical Operations
```

```
# Mean
```

```
mean = np.mean(data1)

# Standard Deviation
std_dev = np.std(data1)


# Minimum
minimum = np.min(data1)


# Maximum
maximum = np.max(data1)


# Mathematical Operations
# Square Root
sqrt = np.sqrt(data1)


# Exponential
exp = np.exp(data1)


# Bitwise Operators
bitwise_and = np.bitwise_and(data1.astype(int),
data2.astype(int))
bitwise_or = np.bitwise_or(data1.astype(int), data2.astype(int))


# Copying and Viewing Arrays
copy_array = data1.copy()
view_array = data1.view()


# Data Stacking
data_stack = np.column_stack((data1, data2))


# Searching
index = np.where(data1 == 40.9)
```

```
# Sorting
sorted_data = np.sort(data1, axis=0)

# Counting
unique_values, counts = np.unique(data1[:, 1], return_counts=True)

# Broadcasting
broadcasted_array = data1 + 10

# Displaying the results
print("Matrix Sum:")
print(matrix_sum)
print("\nMatrix Difference:")
print(matrix_diff)
print("\nMatrix Product:")
print(matrix_product)
print("\nMatrix Transpose:")
print(matrix_transpose)
print("\nHorizontal Stack:")
print(horizontal_stack)
print("\nVertical Stack:")
print(vertical_stack)
print("\nCustom Sequence:")
print(custom_sequence)
print("\nMean:") print(mean)
print("\nStandard Deviation:")
print(std_dev)
print("\nMinimum:")
print(minimum)
print("\nMaximum:")
print(maximum)
print("\nSquare Root:")
```

```

print(sqrt)

print("\nExponential:")

print(exp) print("\nBitwise
AND:") print(bitwise_and)

print("\nBitwise OR:")

print(bitwise_or)

print("\nCopied Array:")

print(copy_array)

print("\nView Array:")

print(view_array)

print("\nData Stack:")

print(data_stack)

print("\nIndex of 40.9 in
data1:") print(index)

print("\nSorted Data:")

print(sorted_data)

print("\nUnique Values and
Counts:")

print(unique_values, counts)

print("\nBroadcasted Array:")

print(broadcasted_array)

```

Output: **Matrix Sum:**

```

[[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]]

```

**Matrix Difference:**

```

[[ 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]]
```

Matrix Product:

```
[[3670.7699 3661.4676 3433.9648 3406.1468 3382.4896 3325.1596 3372.376
 3537.4409 3707.9462 3861.2343]
[3718.4627 3708.7576 3478.0157 3450.2001 3426.2988 3368.0122 3416.1717
 3583.285  3756.0027 3911.6643]
[3595.8285 3585.3246 3360.4967 3335.8215 3312.727  3255.4027 3303.3737
 3464.1376 3631.7204 3783.285 ]
[3392.6904 3384.3192 3174.7776 3148.0944 3126.3816 3073.6692 3116.964
 3270.    3427.0908 3568.878 ]
[3458.1081 3448.9982 3233.9342 3208.7108 3186.342  3131.9908 3176.9399
 3332.01   3493.0276 3637.5752]
[3387.8333 3378.7632 3168.3294 3143.2532 3121.5366 3068.2657 3112.4063
 3264.5992 3421.9367 3564.0835]
[3478.318  3469.046  3252.1663 3227.5485 3204.8906 3150.0459 3195.457
 3351.0376 3513.4454 3658.6088]
[3587.5821 3577.6888 3354.1456 3328.525  3305.425  3248.7103 3295.8567
 3456.5956 3623.6199 3774.1931]
[3782.1961 3772.3736 3537.3438 3509.5092 3485.0318 3425.7029 3474.6919
 3644.3812 3820.4427 3978.3859]
[3915.0043 3904.4672 3660.1961 3632.7021 3607.1972 3545.3782 3596.6185
 3771.6478 3954.5059 4117.9791]]
```

Matrix Transpose:

```
[ 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]]
```

Horizontal Stack:

```
[[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
]]

```

Vertical Stack:

```

[[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 ]]

```

Custom Sequence:

```
[10 20 30 40 50]
```

Mean:

```
186.03499999999997
```

Standard Deviation:

```
309.7929965912722
```

Minimum:

```
25.21
```

Maximum:

```
810.0
```

Square Root:

```

[[28.3019434  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]]
```

Exponential:

```
[[ inf 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]]
```

Bitwise AND:

```
[[801 8 2 28 18]
 [802 8 0 28 18]
 [803 10 28 28 16]
 [804 2 26 24 16]
 [805 8 26 24 16]
 [806 1 26 26 17]
 [807 8 25 24 16]
 [808 10 0 28 18]
 [809 12 0 29 20]
 [810 14 4 31 20]]
```

Bitwise OR:

```
[[801 63 59 30 31]
 [802 63 61 30 31]
 [803 58 31 28 31]
 [804 63 31 30 30]
 [805 58 31 31 29]
 [806 63 30 27 29]
 [807 59 31 31 29]
 [808 59 59 28 30]
 [809 60 62 31 30]
 [810 62 60 31 31]]
```

Copied 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]]
```

View 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]]
```

Data Stack:

```
[[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  28.93
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
]]
```

Index of 40.9 in data1:

```
(array([4]), array([1]))
```

Sorted Data:

```
[[801.    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]]
```

Unique Values and Counts:

```
[39.24 39.47 40.9  41.68 42.19 42.24 43.05 43.47 44.75 46.95] [1 1 1 1
```



1 1 1 1 1 1]

Broadcasted Array:

```
[[811.    53.05  37.79  38.7   37.79]
 [812.    53.47  38.52  38.98  37.89]
 [813.    52.24  38.16  38.16  35.63]
 [814.    49.24  36.16  36.16  36.16]
 [815.    50.9   36.03  37.27  35.65]
 [816.    49.47  36.31  36.31  35.21]

 [817.    51.68  35.63  37.79  35.46]
 [818.    52.19  37.61  38.13  36.21]
 [819.    54.75  38.35  39.83  38.21]
 [820.    56.95  38.88  41.3   38.53]]
```

The screenshot shows a Google Colab notebook interface. The main code cell contains the following Python code:

```
print(broadcasted_array)
```

The output of the code is displayed in the console, showing the result of the matrix operations:

```
Matrix Sum:
[[1602.    71.53  61.97  59.26  50.02]
 [1604.    71.57  62.24  59.66  50.71]
 [1606.    68.4   59.55  56.96  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]]

Matrix Difference:
[[ 0.    14.57 -6.39 -1.86  5.56]
 [ 0.    15.37 -5.2   -1.7   5.07]
 [ 0.    16.08 -3.23 -0.84  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   -7.34 -0.88  5.43]]

Matrix Product:
[[3670.7699 3661.4676 3433.9648 3406.1468 3382.4896 3325.1596 3372.376
 3537.4409 3707.9462 3861.2343]
 [3718.4627 3708.7576 3478.0157 3450.2001 3426.2988 3368.0122 3416.1717
 3583.285  3756.0027 3911.6643]
 [3595.8285 3585.3246 3360.4967 3335.8215 3312.727  3255.4027 3303.3737]]
```

On the right side of the notebook, there is a file viewer showing two CSV files: 'testmarks1.csv' and 'testmarks2.csv'. The 'testmarks1.csv' file is selected, and its contents are displayed in a table with 10 rows and 5 columns: RollNo, EDS, SON, DT, and ET.

RollNo	EDS	SON	DT	ET
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	29.63	26.21
810	46.95	28.88	31.3	26.53

The bottom status bar shows the system clock as 11:12 on 10-06-2023, and the Colab session is completed at 11:08 AM.