Practical 3

Name: Dnyaneshwar Fulchand Sirsat

Roll no: 360

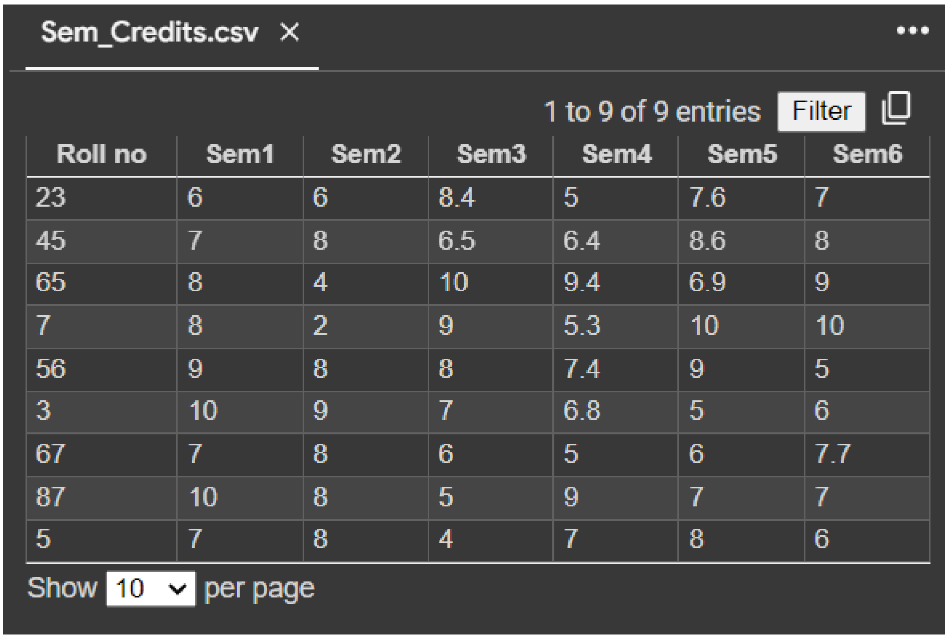
Batch: C3

PRN: 202201050014

Problem Statement:

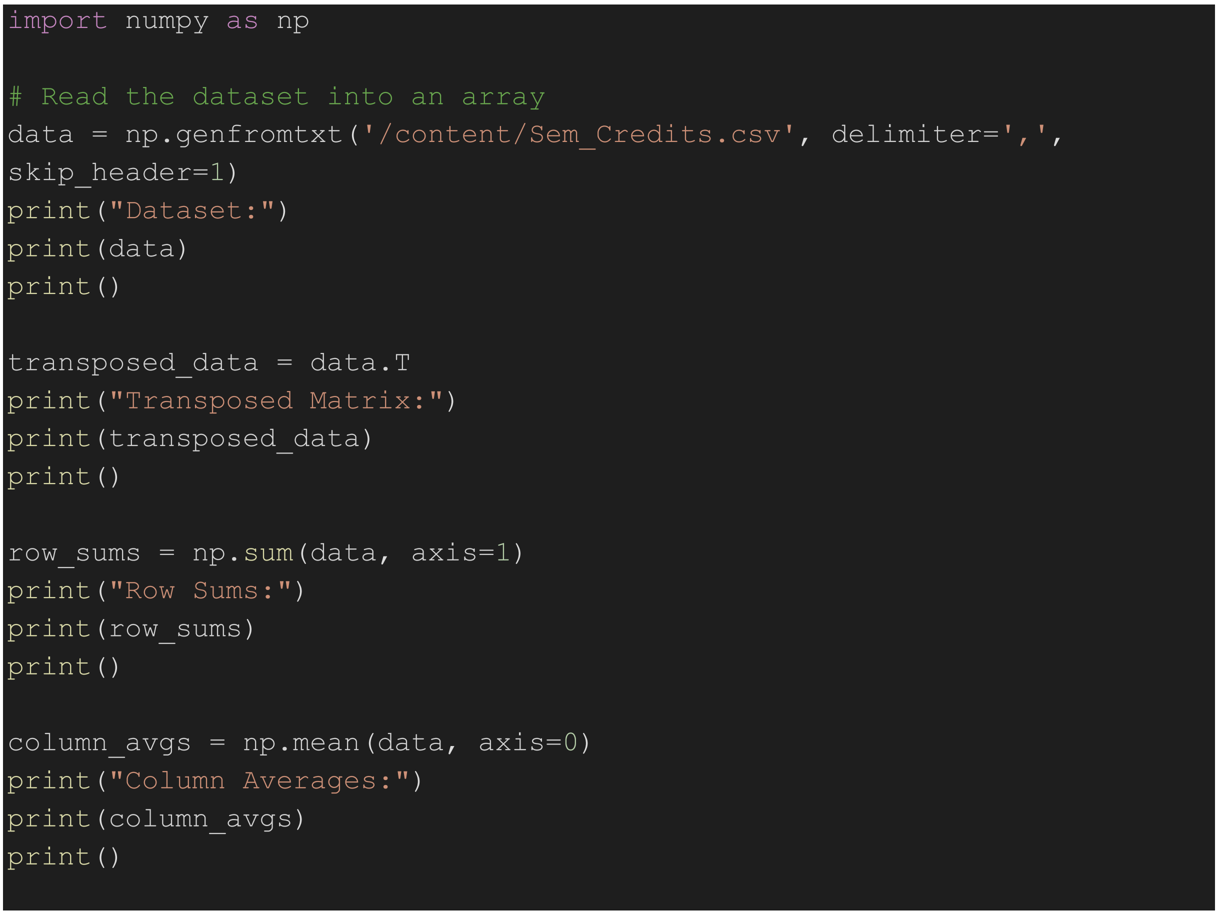
1. Read this dataset into an array
2. Perform all matrix operations on it
3. Horizontal and vertical stacking of numpy arrays
4. Custom sequence generations
5. Arithmetic and statistical operations
6. Mathematical operations
7. Bitwise operations
8. Copying and viewing arrays
9. Data stacking
10. Data Searching
11. Data sorting
12. Data counting
13. Data broadcasting

File: /content/Sem_Credits.csv



Roll no	Sem1	Sem2	Sem3	Sem4	Sem5	Sem6
23	6	6	8.4	5	7.6	7
45	7	8	6.5	6.4	8.6	8
65	8	4	10	9.4	6.9	9
7	8	2	9	5.3	10	10
56	9	8	8	7.4	9	5
3	10	9	7	6.8	5	6
67	7	8	6	5	6	7.7
87	10	8	5	9	7	7
5	7	8	4	7	8	6

Code:



```
import numpy as np

# Read the dataset into an array
data = np.genfromtxt('/content/Sem_Credits.csv', delimiter=',',
skip_header=1)
print("Dataset:")
print(data)
print()

transposed_data = data.T
print("Transposed Matrix:")
print(transposed_data)
print()

row_sums = np.sum(data, axis=1)
print("Row Sums:")
print(row_sums)
print()

column_avgs = np.mean(data, axis=0)
print("Column Averages:")
print(column_avgs)
print()
```

```python
scaled_data = 2 * data
print("Scaled Matrix:")
print(scaled_data)
print()

elementwise_scaled_data = data * 2
print("Element-wise Scaled Matrix:")
print(elementwise_scaled_data)
print()

matrix_product = np.dot(data, data.T)
print("Matrix Product:")
print(matrix_product)
print()

print("Horizontal stacking:")
stacked_horizontal = np.hstack((data, data))
print(stacked_horizontal)

print("Vertical stacking:")
stacked_vertical = np.vstack((data, data))
print(stacked_vertical)

custom_sequence = np.arange(0, 10, 2)
print("Custom sequence:", custom_sequence)

print("Sum of each row:", np.sum(data, axis=1))
print("Mean of each column:", np.mean(data, axis=0))

print("Square root of each element:", np.sqrt(data))
print("Exponential of each element:", np.exp(data))

copy_data = np.copy(data)
view_data = data.view()
print("Copied array:\n", copy_data)
print("Viewed array:\n", view_data)

data_stack = np.stack((data, data))
print("Data stacking:\n", data_stack)

indices = np.where(data > 70)
print("Indices where data > 70:\n", indices)

sorted_data = np.sort(data, axis=0)
print("Sorted data:\n", sorted_data)

unique_elements, counts = np.unique(data, return_counts=True)
print("Unique elements:", unique_elements)
print("Counts:", counts)

broadcasted_data = data + 10
print("Broadcasted data:\n", broadcasted_data)
```

Output:

```
[ 5.   7.   8.   4.   7.   8.   6. ]]
Viewed array:
 [[23.   6.   6.   8.4  5.   7.6  7. ]
 [45.   7.   8.   6.5  6.4  8.6  8. ]
 [65.   8.   4.  10.   9.4  6.9  9. ]
 [ 7.   8.   2.   9.   5.3 10.  10. ]
 [56.   9.   8.   8.   7.4  9.   5. ]
 [ 3.  10.   9.   7.   6.8  5.   6. ]
 [67.   7.   8.   6.   5.   6.   7.7]
 [87.  10.   8.   5.   9.   7.   7. ]
 [ 5.   7.   8.   4.   7.   8.   6. ]]
Data stacking:
 [[[23.   6.   6.   8.4  5.   7.6  7. ]
  [45.   7.   8.   6.5  6.4  8.6  8. ]
  [65.   8.   4.  10.   9.4  6.9  9. ]
  [ 7.   8.   2.   9.   5.3 10.  10. ]
  [56.   9.   8.   8.   7.4  9.   5. ]
  [ 3.  10.   9.   7.   6.8  5.   6. ]
  [67.   7.   8.   6.   5.   6.   7.7]
  [87.  10.   8.   5.   9.   7.   7. ]
  [ 5.   7.   8.   4.   7.   8.   6. ]]

 [[23.   6.   6.   8.4  5.   7.6  7. ]
  [45.   7.   8.   6.5  6.4  8.6  8. ]
  [65.   8.   4.  10.   9.4  6.9  9. ]
  [ 7.   8.   2.   9.   5.3 10.  10. ]
  [56.   9.   8.   8.   7.4  9.   5. ]
  [ 3.  10.   9.   7.   6.8  5.   6. ]
  [67.   7.   8.   6.   5.   6.   7.7]
  [87.  10.   8.   5.   9.   7.   7. ]
  [ 5.   7.   8.   4.   7.   8.   6. ]]]
Indices where data > 70:
 (array([7]), array([0]))
Sorted data:
 [[ 3.   6.   2.   4.   5.   5.   5. ]
 [ 5.   7.   4.   5.   5.   6.   6. ]
 [ 7.   7.   6.   6.   5.3  6.9  6. ]
 [23.   7.   8.   6.5  6.4  7.   7. ]
 [45.   8.   8.   7.   6.8  7.6  7. ]
 [56.   8.   8.   8.   7.   8.   7.7]
 [65.   9.   8.   8.4  7.4  8.6  8. ]
 [67.  10.   8.   9.   9.   9.   9. ]
 [87.  10.   9.  10.   9.4 10.  10. ]]
Unique elements: [ 2.   3.   4.   5.   5.3  6.   6.4  6.5  6.8  6.9  7.
7.4  7.6  7.7
  8.   8.4  8.6  9.   9.4 10.  23.  45.  56.  65.  67.  87. ]
Counts: [ 1  1  2  6  1  6  1  1  1  1  9  1  1  1 10  1  1  6  1  5  1
1  1  1
```

```
  1  1]
Broadcasted data:
 [[33.  16.  16.  18.4 15.  17.6 17. ]
 [55.  17.  18.  16.5 16.4 18.6 18. ]
 [75.  18.  14.  20.  19.4 16.9 19. ]
 [17.  18.  12.  19.  15.3 20.  20. ]
 [66.  19.  18.  18.  17.4 19.  15. ]
 [13.  20.  19.  17.  16.8 15.  16. ]
 [77.  17.  18.  16.  15.  16.  17.7]
 [97.  20.  18.  15.  19.  17.  17. ]
 [15.  17.  18.  14.  17.  18.  16. ]]
```