

Problem 1: Write a Python program to calculate the sum of each row in a 4x4 matrix.(10 points)

Input: A 4x4 matrix.

Output: The sum of each row.

Example:

```
matrix = [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]]
```

Expected Output:

Sum of row 1: 10

Sum of row 2: 26

Sum of row 3: 42

Sum of row 4: 58

```
matrix = [
    [1, 2, 3, 4],
    [5, 6, 7, 8],
    [9, 10, 11, 12],
    [13, 14, 15, 16]
]

for i in range(len(matrix)):
    sum_of_row = sum(matrix[i])
    print(f"Sum of row {i+1}: {sum_of_row}")
```

```
↩ Sum of row 1: 10
   Sum of row 2: 26
   Sum of row 3: 42
   Sum of row 4: 58
```

Problem 2: From a list of tuples containing student names, scores, grades, and departments, identify and print students scoring above 75 in the "Computer Science" department along with their names and grades. (10 points)

Input:

```
students = [ ("Alice", 70, "B", "Mathematics"), ("Bob", 80, "A", "Computer Science"), ("Charlie", 60, "C", "Physics"), ("David", 85, "A+", "Computer Science"), ("Emma", 90, "A+", "Mathematics"), ("Frank", 75, "B", "Computer Science"), ("Grace", 65, "C", "Physics"), ("Hannah", 78, "A", "Computer Science")]
```

Expected Output:

Computer Science students scoring above 75:

Bob (Grade: A)

David (Grade: A+)

Hannah (Grade: A)

```
students = [ ("Alice", 70, "B", "Mathematics"), ("Bob", 80, "A", "Computer Science"), ("Charlie", 60, "C", "Physics"), ("David", 85, "A+", "Computer Science"), ("Emma", 90, "A+", "Mathematics"), ("Frank", 75, "B", "Computer Science"), ("Grace", 65, "C", "Physics"), ("Hannah", 78, "A", "Computer Science")]

print("Computer Science Students scoring above 75: ")
for student in students:
    name, score, grade, department = student
    if department == "Computer Science" and score > 75:
        print(f"{name} (Grade: {grade})")
```

```
↩ Computer Science Students scoring above 75:
   Bob (Grade: A)
   David (Grade: A+)
   Hannah (Grade: A)
```

Problem 3: (20 points)

You are given a text file named **numbers_with_na.txt** containing 100 entries. The text is attached in the BrightSpace. Each line of the file contains either a number between 1 and 100 (inclusive) or an 'NA' value representing missing data. Write a Python program using pandas to:

- Read the data from the file into a DataFrame.
- Identify and remove rows that contain 'NA' values.

- Normalize the remaining numbers to a 0-to-1 scale.
- Sort the normalized values in ascending order.
- Print the first 10 normalized values after sorting.
- Save these 10 normalized values to a new CSV file called normalized_values.csv.

Input:

A text file numbers_with_na.txt with 100 lines, each containing a single value: either an integer between 1 and 100, or 'NA'.

Output:

The first 10 normalized values printed to the screen. A CSV file normalized_values.csv containing these 10 values.

```
import pandas as pd

# Reading the data from the file into a DataFrame
file_name = "numbers_with_na.txt"
data = pd.read_csv(file_name, header=None, names=["Values"], na_values="NA")

# Removing rows containing 'NA' values and explicitly create a copy
data_cleaned = data.dropna().copy()

#normalization
min_value = data_cleaned["Values"].min()
max_value = data_cleaned["Values"].max()
data_cleaned["Normalized"] = (data_cleaned["Values"] - min_value) / (max_value - min_value)

#sorting
sorted_data = data_cleaned.sort_values(by="Normalized")

first_10_values = sorted_data["Normalized"].head(10)
print("First 10 normalized values:", first_10_values)

# Saving these 10 normalized values to a new CSV file
first_10_values.to_csv("normalized_values.csv", index=False, header=False)
```

```
First 10 normalized values: 18    0.000000
9      0.010204
64     0.020408
39     0.030612
48     0.040816
96     0.051020
34     0.051020
17     0.051020
46     0.071429
71     0.081633
Name: Normalized, dtype: float64
```

Problem 4: (20 points)

You have a 2D tuple containing student information. Each row in the tuple represents a single student and includes the following 5 fields:

ID: A unique identifier for the student (integer).

Department: The department the student belongs to (string).

Class: The class or year of the student (e.g., Freshman, Sophomore, etc.).

Name: The full name of the student (string).

GPA: The GPA of the student (float).\

Your tasks are:

- Store the dataset in a 2D tuple with 10 rows and 5 columns as described.
- Define the column names in a separate list.
- Convert the tuple into a pandas DataFrame and set the column names accordingly.
- Filter out students who have a GPA lower than 3.2.
- Sort the filtered data by GPA in descending order.
- Add a new computed column Honors that is True if the GPA is above 3.5, otherwise False. example:
df_sorted['Honors'] = df_sorted['GPA'] > 3.5
- Export the resulting DataFrame to a CSV file named "student_data_filtered.csv" without an index column. Attach the csv file in the BrightSpace.
- Print a message confirming successful export.

```
import pandas as pd
```

Define the 2D tuple with 10 rows and 5 columns

```
data = ( (101, "Computer Science", "Sophomore", "Alice Johnson", 3.6), (102, "Mathematics", "Junior", "Bob Smith", 3.2), (103, "Physics",
"Freshman", "Charlie Brown", 3.8), (104, "Chemistry", "Senior", "Daisy Ridley", 3.5), (105, "Biology", "Sophomore", "Ethan Clarke", 3.7), (106,
"Engineering", "Junior", "Fiona Gallagher", 3.3), (107, "Law", "Senior", "George Bell", 3.1), (108, "Economics", "Freshman", "Hannah Adams", 3.9),
(109, "Philosophy", "Sophomore", "Ian Wright", 3.4), (110, "Psychology", "Junior", "Jack White", 3.5),)
```

Expected Output

A	B	C	D	E	F
ID	Departme	Class	Name	GPA	Honors
108	Economic	Freshman	Hannah A	3.9	TRUE
103	Physics	Freshman	Charlie Br	3.8	TRUE
105	Biology	Sophomor	Ethan Clar	3.7	TRUE
101	Computer	Sophomor	Alice John	3.6	TRUE
104	Chemistry	Senior	Daisy Ridl	3.5	FALSE
110	Psycholog	Junior	Jack White	3.5	FALSE
109	Philosoph	Sophomor	Ian Wright	3.4	FALSE
106	Engineeri	Junior	Fiona Gall	3.3	FALSE
102	Mathema	Junior	Bob Smith	3.2	FALSE

```
import pandas as pd
```

```
data = ( (101, "Computer Science", "Sophomore", "Alice Johnson", 3.6), (102, "Mathematics", "Junior", "Bob Smith", 3.2), (103, "
```

```
columns = ["ID", "Department", "Class", "Name", "GPA"]
```

```
df = pd.DataFrame(data, columns=columns)
```


```
df_filtered_std = df[df['GPA'] >= 3.2]
```

```
df_sorted_values = df_filtered_std.sort_values(by='GPA', ascending=False)
```

```
df_sorted_values['Honors'] =df_sorted_values['GPA'] > 3.5
```

```
df_sorted_values.to_csv('student_data_filtered.csv', index=False)
```

```
print("DataFrame has been successfully exported to student_data_filtered.csv.")
```

 DataFrame has been successfully exported to student_data_filtered.csv.