ITCP-25/AI-003 (Noor Fatima)

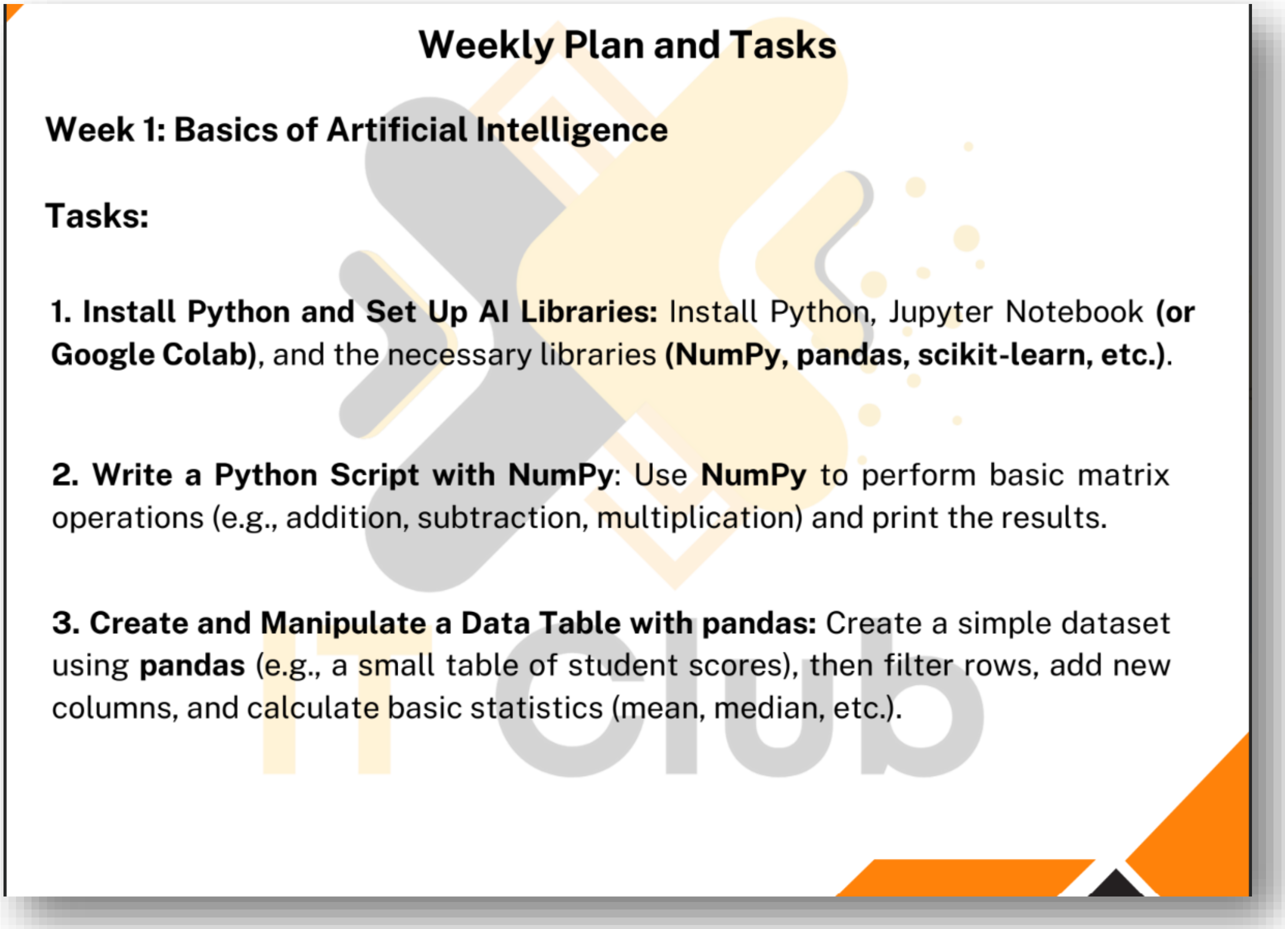
Week 1: Basics of Artificial Intelligence

Tasks:

1. Install Python and Set Up Al Libraries: Install Python, Jupyter Notebook (or Google Colab), and the necessary libraries (NumPy, pandas, scikit-learn, etc.).

2. Write a Python Script with NumPy: Use NumPy to perform basic matrix operations (e.g., addition, subtraction, multiplication) and print the results.

3. Create and Manipulate a Data Table with pandas: Create a simple dataset using pandas (e.g., a small table of student scores), then filter rows, add new columns, and calculate basic statistics (mean, median, etc.). statistics(mean, median, etc.).



Weekly Plan and Tasks

Week 1: Basics of Artificial Intelligence

Tasks:

1. Install Python and Set Up AI Libraries: Install Python, Jupyter Notebook **(or Google Colab)**, and the necessary libraries **(NumPy, pandas, scikit-learn, etc.)**.

2. Write a Python Script with NumPy: Use **NumPy** to perform basic matrix operations (e.g., addition, subtraction, multiplication) and print the results.

3. Create and Manipulate a Data Table with pandas: Create a simple dataset using **pandas** (e.g., a small table of student scores), then filter rows, add new columns, and calculate basic statistics (mean, median, etc.).

Task 1:

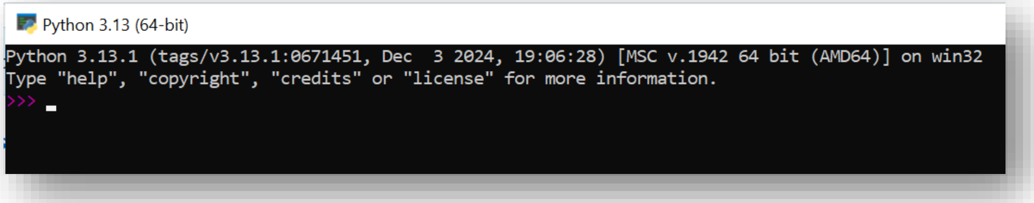
Install Python and Set Up AI Libraries: Install Python, Jupyter Notebook (or Google Colab), and the necessary libraries (NumPy, pandas, scikit-learn, etc.).

Installation and Setup of Python and Essential Libraries

1. Introduction I was assigned the task of installing Python and setting up essential libraries, including Jupyter Notebook (or Google Colab), NumPy, pandas, and scikit-learn. This document outlines the step-by-step process followed to complete this task successfully.

2. Installing Python To install Python, the following steps were taken:

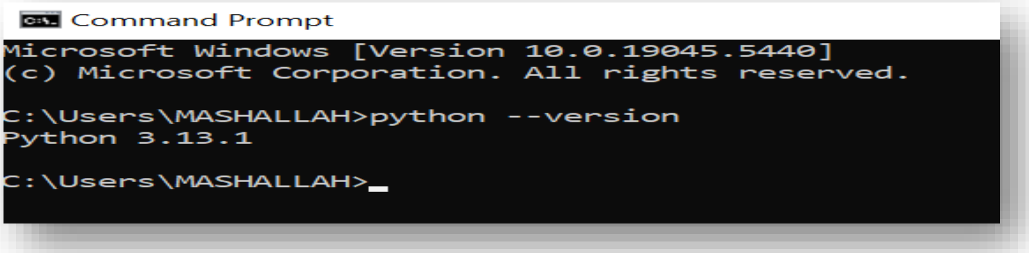
1. Downloaded the latest version of Python from the official website: https://www.python.org/downloads/
2. Installed Python by running the downloaded installer and selecting the option to add Python to the system PATH.
3. Verified the installation by opening the command prompt and running:
4. python –version



```
Python 3.13.1 (tags/v3.13.1:0671451, Dec  3 2024, 19:06:28) [MSC v.1942 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

3. Installing Jupyter Notebook Jupyter Notebook was installed to facilitate working with Python interactively. The following steps were taken:

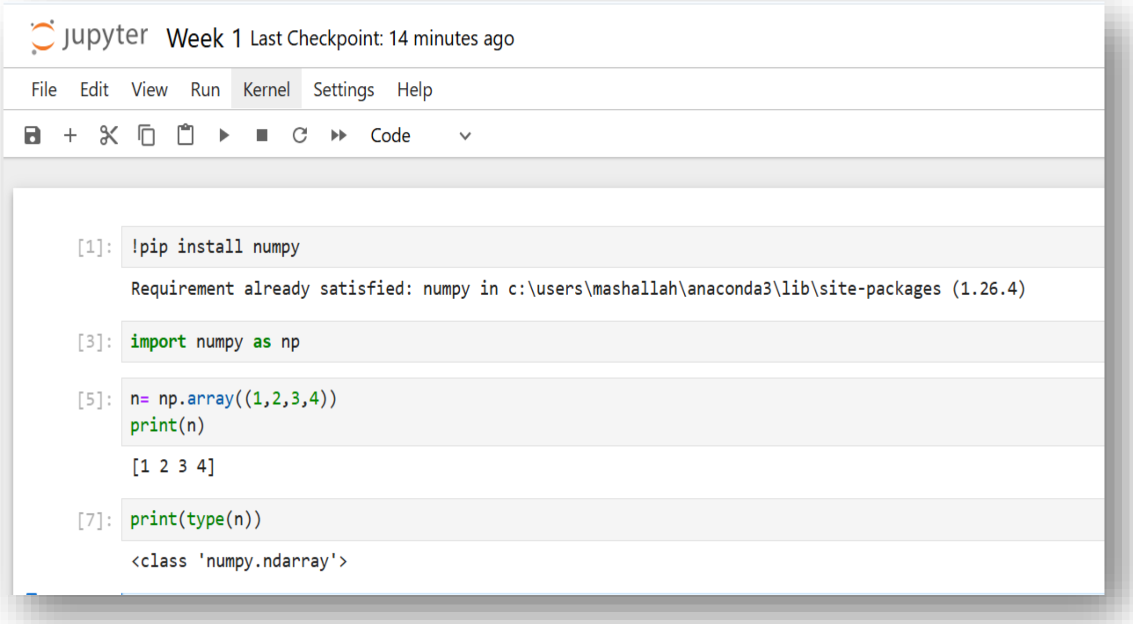
1. Opened the command prompt and installed Jupyter using pip:
2. pip install notebook
3. Verified the installation by running:



```
Microsoft Windows [Version 10.0.19045.5440]
(c) Microsoft Corporation. All rights reserved.

C:\Users\MASHALLAH>python --version
Python 3.13.1

C:\Users\MASHALLAH>_
```

4. Setting Up Google Colab (Alternative to Jupyter Notebook) Google Colab is an online Jupyter environment. The steps to use it are:

1. Opened https://colab.research.google.com/ in a web browser.
2. Signed in with a Google account and created a new notebook.
3. Verified that Python code could be executed within the notebook.


jupyter
Week 1 Last Checkpoint: 14 minutes ago
File Edit View Run Kernel Settings Help
Code

```
[1]: !pip install numpy
     Requirement already satisfied: numpy in c:\users\mashallah\anaconda3\lib\site-packages (1.26.4)
[3]: import numpy as np
[5]: n= np.array((1,2,3,4))
     print(n)
     [1 2 3 4]
[7]: print(type(n))
     <class 'numpy.ndarray'>
```

5. Installing Essential Libraries The necessary libraries were installed using the following commands:

1. Installed NumPy:
2. pip install numpy
3. Installed pandas:
4. pip install pandas
5. Installed scikit-learn:
6. pip install scikit-learn
7. Verified the installations by running:
8. python -c "import numpy, pandas, sklearn; print('Libraries Installed Successfully')"

6. Conclusion The installation and setup of Python, Jupyter Notebook, and essential libraries were successfully completed. This setup enables efficient coding and data analysis for future tasks and projects.

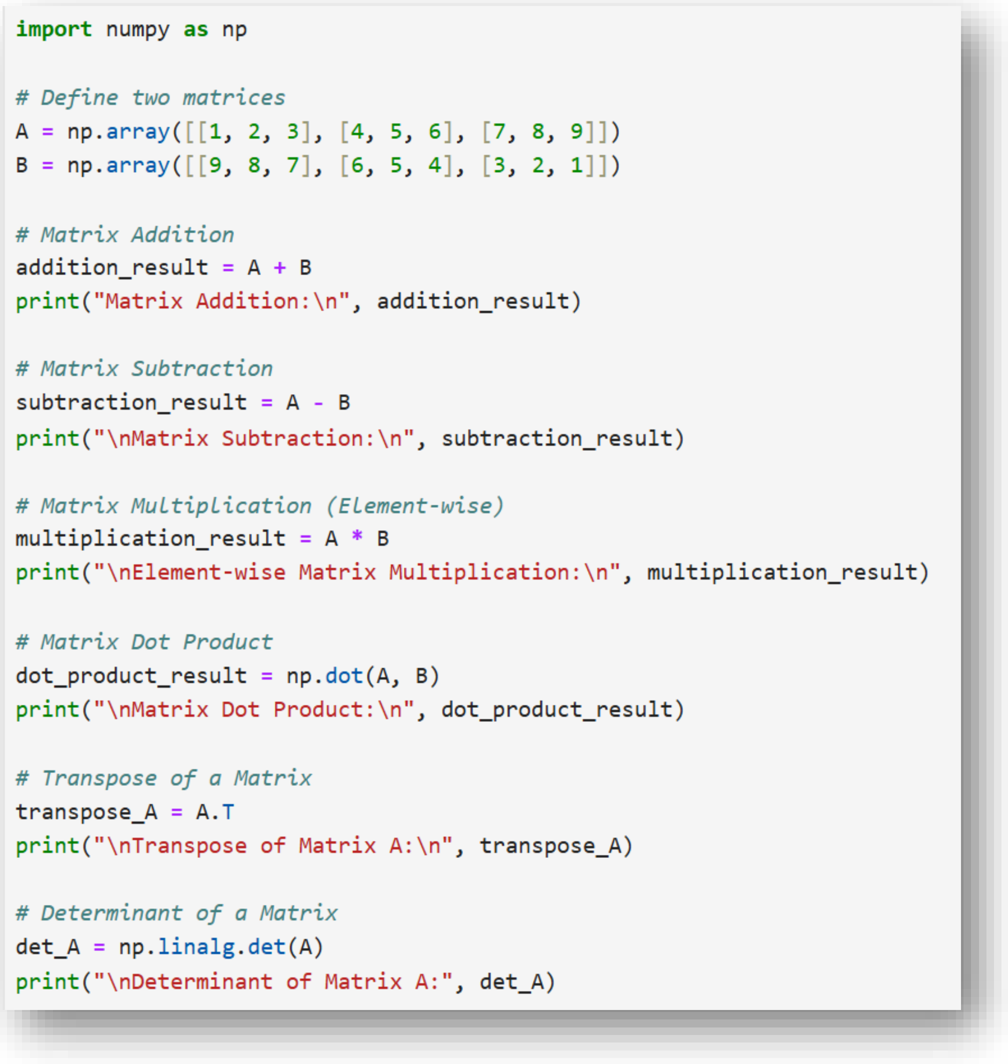
TASK 2:

Write a Python Script with NumPy: Use NumPy to perform basic matrix operations (e.g., addition, subtraction, multiplication) and print the results.

Code:

```
import numpy as np


# Define two matrices

A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

B = np.array([[9, 8, 7], [6, 5, 4], [3, 2, 1]])


# Matrix Addition

addition_result = A + B

print("Matrix Addition:\n", addition_result)


# Matrix Subtraction

subtraction_result = A - B

print("\nMatrix Subtraction:\n", subtraction_result)


# Matrix Multiplication (Element-wise)

multiplication_result = A * B

print("\nElement-wise Matrix Multiplication:\n", multiplication_result)


# Matrix Dot Product

dot_product_result = np.dot(A, B)

print("\nMatrix Dot Product:\n", dot_product_result)


# Transpose of a Matrix

transpose_A = A.T

print("\nTranspose of Matrix A:\n", transpose_A)
```

```
# Determinant of a Matrix

det_A = np.linalg.det(A)

print("\nDeterminant of Matrix A:", det_A)
```



```python
import numpy as np

# Define two matrices
A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
B = np.array([[9, 8, 7], [6, 5, 4], [3, 2, 1]])

# Matrix Addition
addition_result = A + B
print("Matrix Addition:\n", addition_result)

# Matrix Subtraction
subtraction_result = A - B
print("\nMatrix Subtraction:\n", subtraction_result)

# Matrix Multiplication (Element-wise)
multiplication_result = A * B
print("\nElement-wise Matrix Multiplication:\n", multiplication_result)

# Matrix Dot Product
dot_product_result = np.dot(A, B)
print("\nMatrix Dot Product:\n", dot_product_result)

# Transpose of a Matrix
transpose_A = A.T
print("\nTranspose of Matrix A:\n", transpose_A)

# Determinant of a Matrix
det_A = np.linalg.det(A)
print("\nDeterminant of Matrix A:", det_A)
```

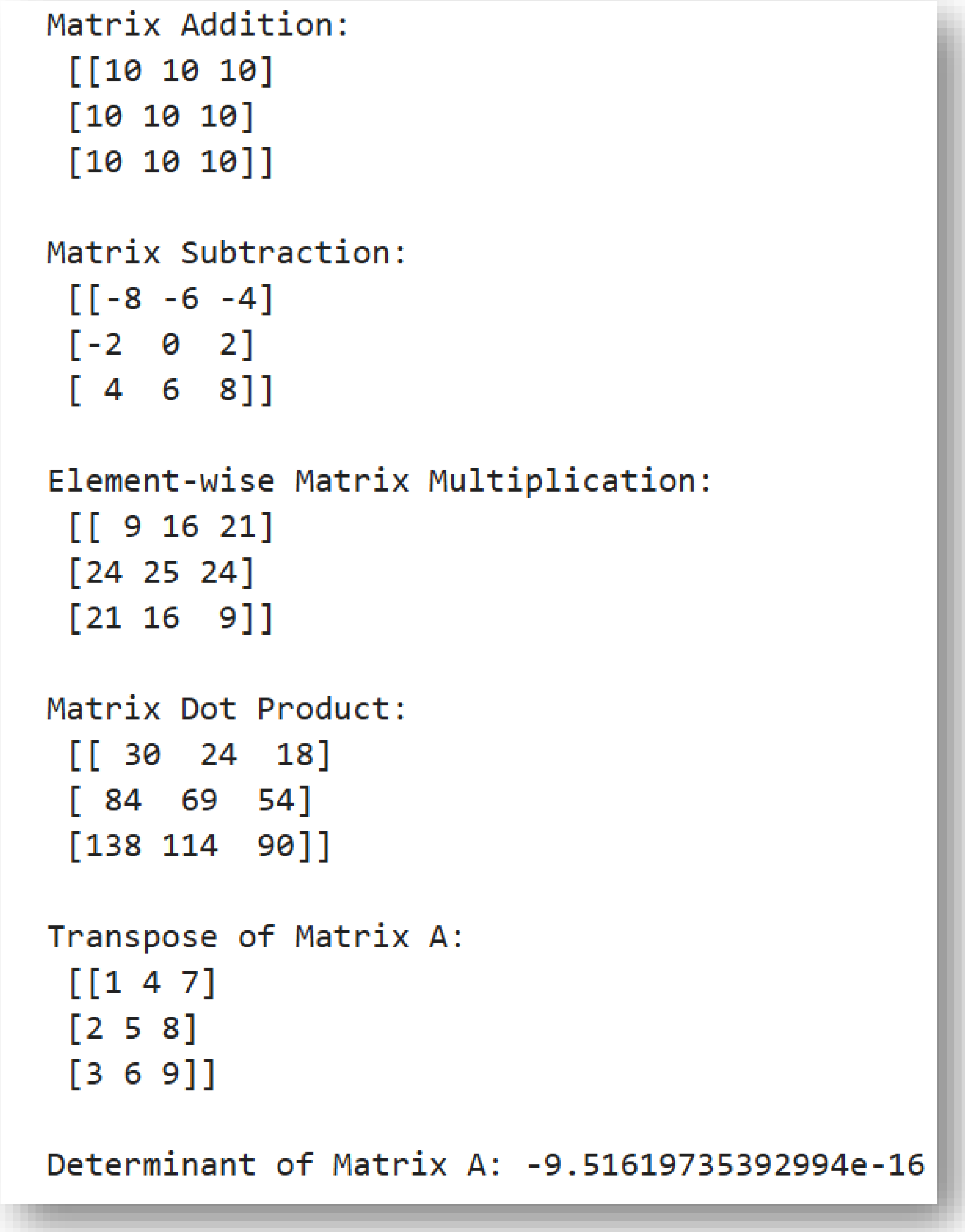
It initializes two 3×3 matrices and performs the following operations:

1. **Matrix Addition:** Adds corresponding elements of the matrices.
2. **Matrix Subtraction:** Subtracts elements of the second matrix from the first.
3. **Element-wise Multiplication:** Multiplies corresponding elements.
4. **Matrix Dot Product:** Computes the dot product of two matrices.

5. **Transpose of a Matrix:** Transposes matrix A.
6. **Determinant Calculation:** Computes the determinant of matrix A.

Each operation's result is printed for verification.

Output:



```
Matrix Addition:
 [[10 10 10]
 [10 10 10]
 [10 10 10]]

Matrix Subtraction:
 [[-8 -6 -4]
 [-2  0  2]
 [ 4  6  8]]

Element-wise Matrix Multiplication:
 [[ 9 16 21]
 [24 25 24]
 [21 16  9]]

Matrix Dot Product:
 [[ 30  24  18]
 [ 84  69  54]
 [138 114  90]]

Transpose of Matrix A:
 [[1 4 7]
 [2 5 8]
 [3 6 9]]

Determinant of Matrix A: -9.51619735392994e-16
```

TASK 3:

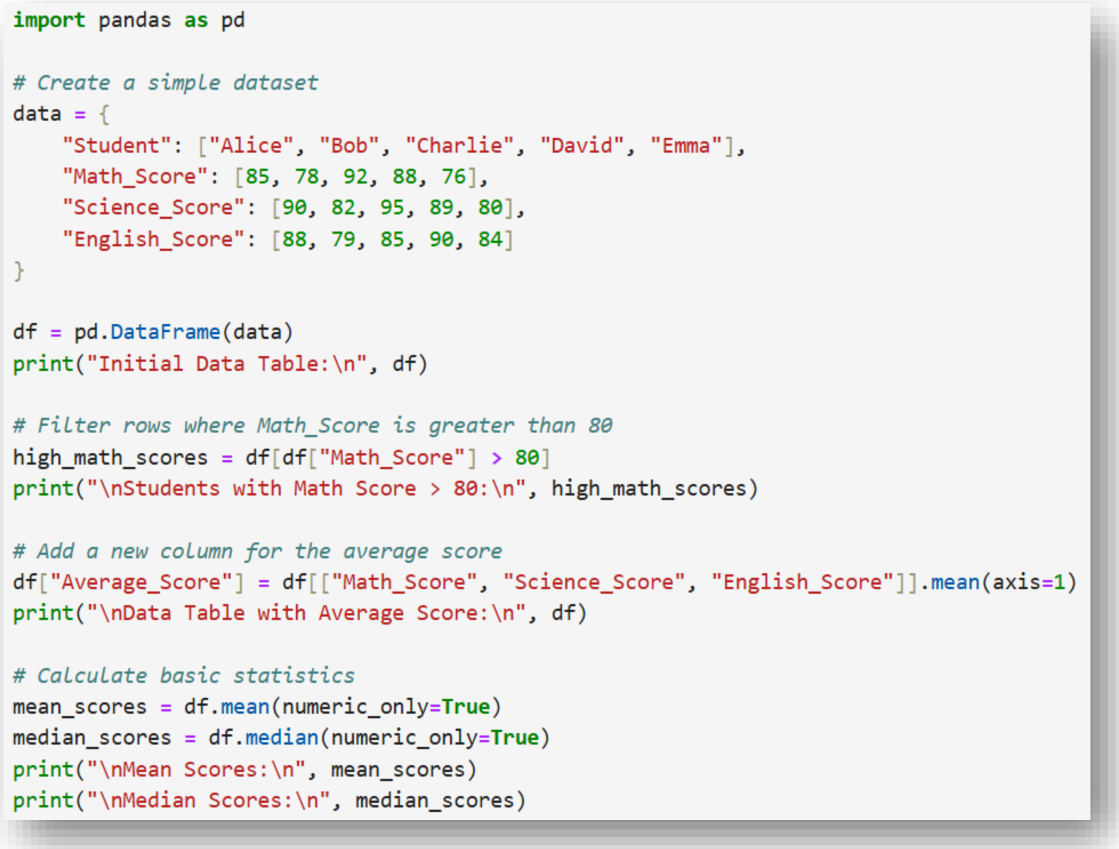
Create and Manipulate a Data Table with pandas: Create a simple dataset using pandas (e.g., a small table of student scores), then filter rows, add new columns, and calculate basic statistics (mean, median, etc.). statistics(mean, median, etc.). import pandas as pd

Code:

```
# Create a simple dataset
data = {
    "Student": ["Alice", "Bob", "Charlie", "David", "Emma"],
    "Math_Score": [85, 78, 92, 88, 76],
    "Science_Score": [90, 82, 95, 89, 80],
    "English_Score": [88, 79, 85, 90, 84]
}

df = pd.DataFrame(data)
print("Initial Data Table:\n", df)

# Filter rows where Math_Score is greater than 80
high_math_scores = df[df["Math_Score"] > 80]
print("\nStudents with Math Score > 80:\n", high_math_scores)

# Add a new column for the average score
df["Average_Score"] = df[["Math_Score", "Science_Score", "English_Score"]].mean(axis=1)
print("\nData Table with Average Score:\n", df)

# Calculate basic statistics
mean_scores = df.mean(numeric_only=True)
median_scores = df.median(numeric_only=True)
```

print("\nMean Scores:\n", mean_scores)

print("\nMedian Scores:\n", median_scores)

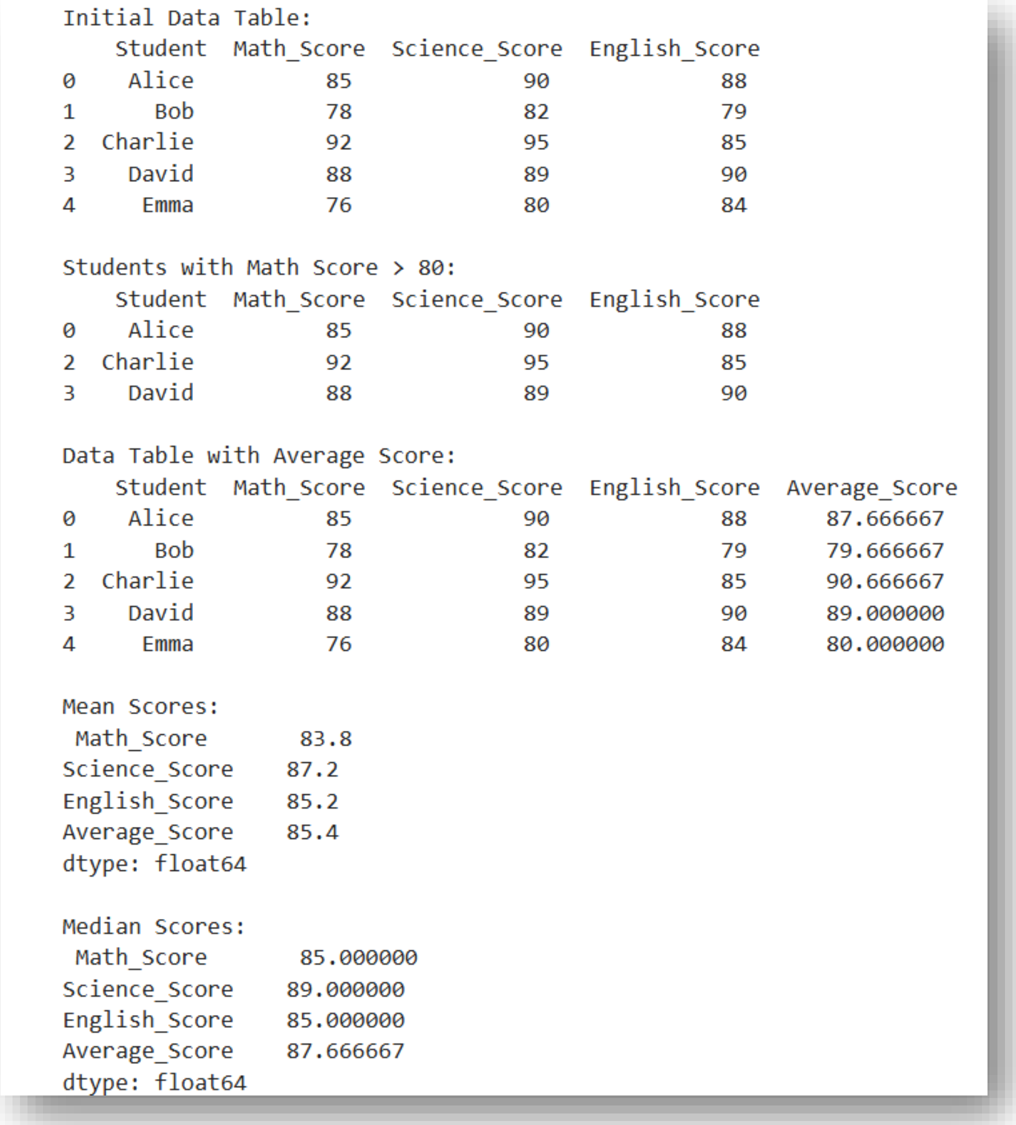


```python
import pandas as pd

# Create a simple dataset
data = {
    "Student": ["Alice", "Bob", "Charlie", "David", "Emma"],
    "Math_Score": [85, 78, 92, 88, 76],
    "Science_Score": [90, 82, 95, 89, 80],
    "English_Score": [88, 79, 85, 90, 84]
}

df = pd.DataFrame(data)
print("Initial Data Table:\n", df)

# Filter rows where Math_Score is greater than 80
high_math_scores = df[df["Math_Score"] > 80]
print("\nStudents with Math Score > 80:\n", high_math_scores)

# Add a new column for the average score
df["Average_Score"] = df[["Math_Score", "Science_Score", "English_Score"]].mean(axis=1)
print("\nData Table with Average Score:\n", df)

# Calculate basic statistics
mean_scores = df.mean(numeric_only=True)
median_scores = df.median(numeric_only=True)
print("\nMean Scores:\n", mean_scores)
print("\nMedian Scores:\n", median_scores)
```

The following operations are performed:

1. **Creating a Data Table:** A sample dataset of student scores in Math, Science, and English is initialized.
2. **Filtering Data:** Rows where the Math score is greater than 80 are extracted.
3. **Adding a New Column:** The average score for each student is calculated and added as a new column.
4. **Basic Statistical Analysis:** The mean and median of all numerical columns are computed and displayed.

Output:



```
Initial Data Table:
   Student  Math_Score  Science_Score  English_Score
0    Alice          85             90             88
1      Bob          78             82             79
2  Charlie          92             95             85
3    David          88             89             90
4     Emma          76             80             84

Students with Math Score > 80:
   Student  Math_Score  Science_Score  English_Score
0    Alice          85             90             88
2  Charlie          92             95             85
3    David          88             89             90

Data Table with Average Score:
   Student  Math_Score  Science_Score  English_Score  Average_Score
0    Alice          85             90             88      87.666667
1      Bob          78             82             79      79.666667
2  Charlie          92             95             85      90.666667
3    David          88             89             90      89.000000
4     Emma          76             80             84      80.000000

Mean Scores:
 Math_Score       83.8
Science_Score    87.2
English_Score    85.2
Average_Score    85.4
dtype: float64

Median Scores:
 Math_Score       85.000000
Science_Score    89.000000
English_Score    85.000000
Average_Score    87.666667
dtype: float64
```