NUMPY CONTINUATION:

Data Science Foundations — NumPy & Pandas

NumPy (Numerical Python)

NumPy is the **core library for numerical and scientific computing** in Python. It provides high-performance **multi-dimensional arrays** and tools to operate on them efficiently — making it faster than native Python lists.

1. NumPy Arrays

➤ **Creating Arrays**

NumPy arrays are similar to Python lists but much faster and more memory-efficient.

```
import numpy as np


# Creating a 1D array
arr = np.array([1, 2, 3, 4, 5])
print("1D Array:", arr)


# Creating a 2D array
matrix = np.array([[1, 2, 3], [4, 5, 6]])
print("\n2D Array:\n", matrix)


# Array of zeros
zeros = np.zeros((2, 3))
print("\nZeros Array:\n", zeros)


# Array of ones
ones = np.ones((3, 3))
```

```
print("\nOnes Array:\n", ones)


# Array with random numbers
randoms = np.random.rand(2, 2)
print("\nRandom Array:\n", randoms)
```

2. Array Indexing & Slicing

```
# Accessing elements
print(arr[0])      # First element
print(matrix[1, 2]) # Row 1, Column 2


# Slicing
print(arr[1:4])    # Elements from index 1 to 3


# Modifying elements
arr[2] = 10
print("Modified Array:", arr)
```

3. Vectorization (Fast Computations)

Vectorization means performing operations on entire arrays without using loops — much faster!

```
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])


# Element-wise operations
print(a + b)    # Addition
```

```
print(a * b)     # Multiplication

print(a ** 2)    # Power

print(np.sqrt(a)) # Square root
```

Why it's powerful:
Instead of looping through elements, NumPy performs operations at **C speed** internally.

4. Matrix Operations

```
# Matrix creation

A = np.array([[1, 2, 3],

        [4, 5, 6]])


B = np.array([[7, 8],

        [9, 10],

        [11, 12]])


# Matrix multiplication (dot product)

C = np.dot(A, B)

print("Matrix Product:\n", C)


# Transpose

print("Transpose:\n", A.T)


# Determinant

square_matrix = np.array([[2, 3], [1, 4]])

print("Determinant:", np.linalg.det(square_matrix))
```

🧠 Real-Time Example

Imagine you're working for a logistics company and need to calculate **total delivery costs** across multiple regions.

```
# Cost per delivery (₦)
cost_per_delivery = np.array([1000, 1200, 800, 950])


# Number of deliveries per region
deliveries = np.array([20, 35, 40, 25])


# Total cost
total_cost = cost_per_delivery * deliveries
print("Total cost per region:", total_cost)
print("Overall cost:", np.sum(total_cost))
```

📝 NumPy Quiz

1. What does NumPy stand for?
2. Difference between a list and a NumPy array?
3. What does np.dot() do?
4. How do you find the mean and standard deviation of an array?
5. Write code to create a 3×3 identity matrix.

💼 NumPy Assignment

1. Create a 4×4 random matrix and find:
    - Its transpose

- Mean and standard deviation of elements

2. Create two arrays and compute element-wise addition, subtraction, and multiplication.
3. Write a program to calculate the monthly profit for 12 months using revenue and cost arrays.

Pandas (Python Data Analysis Library)

Pandas is used for **data cleaning, transformation, and analysis**.
It introduces two main data structures:

- **Series** → 1D labeled array
- **DataFrame** → 2D labeled data table (rows & columns)

1. Series

```
import pandas as pd


# Creating a Series

data = pd.Series([10, 20, 30, 40], index=['Q1', 'Q2', 'Q3', 'Q4'])

print(data)


# Accessing elements

print(data['Q2'])


# Applying operations

print(data.mean())
```

2. DataFrame

```
# Creating a DataFrame
data = {
    'Product': ['Phone', 'Laptop', 'Tablet', 'Camera'],
    'Price': [200, 800, 300, 450],
    'Quantity': [10, 5, 8, 6]
}

df = pd.DataFrame(data)
print(df)

# Accessing columns
print(df['Product'])

# Adding a new column
df['Total'] = df['Price'] * df['Quantity']
print(df)
```

3. GroupBy (Data Aggregation)

```
sales = pd.DataFrame({
    'Region': ['West', 'East', 'West', 'North', 'East'],
    'Sales': [2500, 3000, 2000, 4000, 1500]
})
```

```
# Group by Region and calculate total sales
grouped = sales.groupby('Region')['Sales'].sum()
print(grouped)
```

4. Merge (Combine Datasets)

```
products = pd.DataFrame({
    'ProductID': [1, 2, 3],
    'Name': ['Phone', 'Laptop', 'Tablet']
})

prices = pd.DataFrame({
    'ProductID': [1, 2, 3],
    'Price': [200, 800, 300]
})

merged = pd.merge(products, prices, on='ProductID')
print(merged)
```

5. Pivot Table

```
sales_data = pd.DataFrame({
    'Month': ['Jan', 'Jan', 'Feb', 'Feb', 'Mar', 'Mar'],
    'Product': ['Phone', 'Laptop', 'Phone', 'Laptop', 'Phone', 'Laptop'],
    'Revenue': [5000, 8000, 6000, 7500, 5500, 9000]
})
```

```
pivot = sales_data.pivot_table(values='Revenue', index='Month', columns='Product', aggfunc='sum')

print(pivot)
```

Real-Time Example

Let's say you're analyzing **monthly sales data** for an e-commerce business.

```
data = {
    'Month': ['Jan', 'Jan', 'Feb', 'Feb', 'Mar', 'Mar'],
    'Category': ['Electronics', 'Fashion', 'Electronics', 'Fashion', 'Electronics', 'Fashion'],
    'Revenue': [12000, 8000, 15000, 9000, 18000, 10000]
}

df = pd.DataFrame(data)

# Total revenue per category
print(df.groupby('Category')['Revenue'].sum())

# Monthly revenue summary
print(df.pivot_table(values='Revenue', index='Month', columns='Category', aggfunc='sum'))
```

Pandas Quiz

1. What's the difference between a Series and a DataFrame?
2. How do you merge two DataFrames in Pandas?
3. What does groupby() do?

4. How can you handle missing data?
5. What is a pivot table used for?

Pandas Assignment

1. Load a CSV file of sales data (e.g., sales.csv).
    - Find total revenue per region.
    - Identify the month with highest sales.
    - Add a "Profit" column assuming a 20% margin.
2. Merge two DataFrames: one containing customer info and another containing purchase data.
3. Create a pivot table showing average revenue per category per month.

🎓 **Capstone Projects**

Project 1: E-Commerce Data Analysis

Goal: Analyze and visualize sales data for an online store.
Tasks:

- Load and clean CSV data (Product, Category, Region, Date, Sales)
- Calculate monthly revenue trends
- Find top 5 selling products
- Use NumPy for total profit computation
- Create summary reports with Pandas

Project 2: School Performance Analytics

Goal: Analyze student performance data from different schools.
Tasks:

- Import dataset with student scores

- Group by school and subject
- Find average performance per class
- Identify top and low-performing students
- Present findings using pivot tables

Project 3: Market Insights Dashboard

Goal: Use Pandas and NumPy to analyze marketing data.
Tasks:

- Clean and merge customer and campaign data
- Segment customers by age, gender, region
- Calculate ROI for each marketing campaign
- Summarize data into pivot tables and visual summaries

This part will teach you to go from **raw data → analysis → visual storytelling**, which is the real-world skill every data scientist needs.

🎨 Data Visualization & Reporting with NumPy, Pandas, Matplotlib, and Seaborn

🌟 Overview

Data visualization helps turn data into **insights**.
While **NumPy** and **Pandas** handle computation and data manipulation, **Matplotlib** and **Seaborn** help communicate findings clearly.

We'll use:

- 🧮 **NumPy** → for numerical computation
- 📊 **Pandas** → for data handling
- 🎨 **Matplotlib & Seaborn** → for visualization

🔷 1. Setting Up the Libraries

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns


# Display settings

pd.set_option('display.max_columns', None)

sns.set(style='whitegrid')
```

🔷 2. Create or Load Real-Time Project Data

Let's simulate a **retail sales dataset** (as if from an e-commerce business).

```
# Simulate dataset

np.random.seed(42)
```

```
months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun']
categories = ['Electronics', 'Fashion', 'Home', 'Sports']

# Generate random data
data = {
    'Month': np.random.choice(months, 100),
    'Category': np.random.choice(categories, 100),
    'Units_Sold': np.random.randint(10, 500, 100),
    'Price': np.random.randint(20, 500, 100)
}

df = pd.DataFrame(data)

# Calculate Revenue
df['Revenue'] = df['Units_Sold'] * df['Price']
print(df.head())
```

🔷 3. Analyze Data with Pandas + NumPy

```
# Total revenue
total_revenue = df['Revenue'].sum()

# Average revenue per category
avg_revenue = df.groupby('Category')['Revenue'].mean()

# Monthly revenue trend
```

```
monthly_sales = df.groupby('Month')['Revenue'].sum().sort_index()


# NumPy summary
mean_price = np.mean(df['Price'])
std_units = np.std(df['Units_Sold'])


print("Total Revenue:", total_revenue)
print("\nAverage Revenue per Category:\n", avg_revenue)
print("\nMonthly Revenue Trend:\n", monthly_sales)
```

🔷 4. Visualizing Insights

🟦 a. Bar Chart — Revenue by Category

```
plt.figure(figsize=(8,5))
sns.barplot(x='Category', y='Revenue', data=df, estimator='sum', ci=None, palette='viridis')
plt.title('Total Revenue by Category')
plt.xlabel('Category')
plt.ylabel('Revenue')
plt.show()
```

🟩 b. Line Chart — Monthly Revenue Trend

```
plt.figure(figsize=(8,5))
monthly_sales.plot(kind='line', marker='o')
plt.title('Monthly Revenue Trend')
plt.xlabel('Month')
plt.ylabel('Revenue')
```

```
plt.grid(True)
plt.show()
```

🟨 **c. Histogram — Distribution of Prices**

```
plt.figure(figsize=(8,5))
sns.histplot(df['Price'], bins=20, kde=True, color='orange')
plt.title('Price Distribution')
plt.xlabel('Price')
plt.ylabel('Frequency')
plt.show()
```

🟥 **d. Scatter Plot — Units vs Revenue**

```
plt.figure(figsize=(8,5))
sns.scatterplot(x='Units_Sold', y='Revenue', hue='Category', data=df)
plt.title('Relationship Between Units Sold and Revenue')
plt.xlabel('Units Sold')
plt.ylabel('Revenue')
plt.show()
```

🟪 **e. Heatmap — Correlation Matrix**

```
plt.figure(figsize=(6,4))
sns.heatmap(df[['Units_Sold','Price','Revenue']].corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

🔷 5. Real-Time Business Scenario

Let's say you're an **Analyst at an e-commerce company** tasked with understanding sales trends.

You discover:

- Electronics generate the **highest revenue**, but also have **high price volatility** (using NumPy standard deviation).
- Fashion sales are **consistent**, offering stable revenue.
- The correlation heatmap shows that Units_Sold and Revenue are **strongly positively correlated** (as expected).

📊 Insight-driven Decision:
➡️ Recommend increasing stock for **Fashion items** (high turnover rate).
➡️ Adjust pricing strategy for **Electronics** to reduce volatility.

🧠 Visualization + Analysis Quiz

1. What's the main difference between Matplotlib and Seaborn?
2. Which NumPy function can find the mean of an array?
3. What does the groupby() method do in Pandas?
4. How can you show the correlation between multiple numeric columns?
5. What does sns.barplot() do by default with duplicate x-values?

💼 Assignments

Assignment 1 — Customer Purchase Insights

- Create a DataFrame containing columns: CustomerID, Age, Region, Purchase_Amount, and Category.
- Use **NumPy** to calculate:
    - Mean and standard deviation of purchase amounts.
    - The total purchase per region using array operations.

- Use **Pandas** to:
    - Group by category and region.
    - Identify the top-spending age group.
- Visualize using:
    - Bar chart (Revenue per Region)
    - Histogram (Age distribution)

Assignment 2 — Employee Productivity Report

- Create or import a CSV with Employee, Department, Hours_Worked, Tasks_Completed, and Salary.
- Use **NumPy** to find correlations between work hours and productivity.
- Use **Pandas** to group by department and calculate average salary per performance.
- Plot:
    - Scatterplot (Hours vs Tasks)
    - Heatmap (correlation between variables)

🎓 **Capstone Projects**

Capstone 1: E-Commerce Performance Dashboard

Goal: Build an analysis pipeline for a real-world e-commerce dataset.

Tasks:

1. Use Pandas to clean and aggregate data (monthly, category-wise).
2. Use NumPy to calculate:
    - Mean, standard deviation, and correlation of numerical columns.
3. Visualize:
    - Monthly revenue trends

- Category performance
- Price distribution and correlation heatmap

4. Present key business recommendations.

Capstone 2: Student Performance Analytics

Goal: Evaluate academic performance using data-driven insights.

Dataset Example Columns:
StudentID, Subject, Score, Attendance, Study_Hours

Tasks:

1. Use Pandas to group by subject and calculate averages.
2. Use NumPy to analyze correlation between study hours and scores.
3. Visualize using:
   - Line chart (average score trend)
   - Scatter plot (Study Hours vs Scores)
   - Heatmap (correlation matrix)

Capstone 3: Market Analysis for Sales Forecasting

Goal: Build a predictive insight report for sales trends.
Dataset Example Columns:
Date, Region, Product, Units_Sold, Price, Revenue

Tasks:

1. Use Pandas to calculate month-over-month sales growth.
2. Use NumPy for mathematical operations (e.g., moving averages, variance).
3. Use Matplotlib & Seaborn for:
   - Line plots (trend analysis)

- Bar plots (region-wise revenue)
- Heatmap (product correlation)