Assignment: Understanding and Applying NumPy for Big Data Analysis

---

Objective

The goal of this assignment is to provide you with hands-on experience using the NumPy library for numerical computations in Python. You will explore array creation, mathematical operations, slicing, and aggregations, which are foundational skills for analyzing and processing large datasets in a Big Data environment.

Scenario: Data Preparation for Retail Analytics

You are a data scientist working for a retail company. The company collects massive amounts of sales data daily, which you need to preprocess and analyze efficiently. This assignment will focus on tasks such as:

1. Converting raw sales data into structured arrays.
2. Performing basic mathematical operations for sales insights.
3. Aggregating sales data across multiple dimensions.
4. Using slicing and indexing to extract subsets of data for analysis.

Tasks

Task 1: Array Creation

1. Create a 2D NumPy array to represent the sales data for 7 days across 3 products. Use the following sales values:
    - Product A: [100, 150, 200, 250, 300, 350, 400]
    - Product B: [120, 170, 220, 270, 320, 370, 420]
    - Product C: [90, 140, 190, 240, 290, 340, 390]
    - **Steps:**
        - Create the array using `np.array`.
        - Print the shape, dimensions, and total size of the array.
    - **Deliverable Example Output:**

```
Shape: (3, 7)
Dimensions: 2
Total elements: 21
```

Task 2: Mathematical Operations

1. Calculate the following for the sales data array:
    - Square of each element.
    - Sales data increased by 10% across all products.
    - Boolean array indicating sales greater than 250.
    - **Steps:**
        - Use elementwise operations to compute the results.
        - Use logical operations to generate the boolean array.
    - **Deliverable Example Output:**

```
Sales increased by 10%:
[[110. 165. 220. ... ]
 ...
]
```

Task 3: Slicing and Indexing

1. Extract the following subsets from the sales array:
    - Sales data for the first 3 days across all products.
    - Sales data for Product B only.
    - Modify the sales data for Product C on Day 7 to 500.
    - **Deliverable Example Output:**

```
Sales for the first 3 days:
[[100 150 200]
 [120 170 220]
 [90 140 190]]
Updated array:
[[100 ...]
 [120 ...]
 [90 ... 500]]
```

Task 4: Aggregation Functions

1. Perform the following aggregations on the sales array:
    - Total sales across all products and days.
    - Average daily sales for each product.
    - Maximum sales for any product on a single day.
    - **Deliverable Example Output:**

```
Total sales: 9450
Average daily sales:
Product A: 250.0
...
Max sales on a single day: 420
```

Task 5: Placeholder Arrays

1. Create the following placeholder arrays for future data:
    - A 3x3 array of zeros to represent sales adjustments.
    - A 4x4 identity matrix for matrix computations.
    - A 2x3 random array for forecasting future sales.
    - **Deliverable Example Output:**

```
Zeros:
[[0. 0. 0.]
 [0. 0. 0.]
 ...
Random:
[[0.1 0.5 ...]
```

Task 6: Real-World Application

1. Using `np.linspace`, generate 12 evenly spaced timestamps representing monthly sales data.
2. Compute the sine transformation of the sales data for Product A (hint: use `np.sin`).
    - **Deliverable Example Output:**

```
Timestamps: [1, 2, 3, ...]
Sine-transformed sales:
[sin(100), sin(150), ...]
```

Submission Guidelines

1. **Code**: Submit a Python script or Jupyter Notebook (.ipynb) with all tasks completed.
2. **Outputs**: Ensure that each task's output is printed clearly with comments explaining the code.
3. **Deadline**: mentioned in moodle
4. **Evaluation Criteria**:
    - Correctness of implementation.
    - Clarity of code and comments.
    - Proper usage of NumPy functions.

Bonus Task

- Using `np.dot` or `@`, compute the dot product of sales data (Product A, B) with a transformation matrix:

```
[[1.1, 0.9],
[0.8, 1.2]]
```

- Interpret the result in the context of adjusting sales data for a discount scenario.