Day 1: Introduction to NumPy

1. What is NumPy?

NumPy (Numerical Python) is a powerful library for numerical computing in Python. It provides:

- A high-performance **n-dimensional array object** (`ndarray`)
- Tools for integrating with C/C++ and Fortran
- Functions for **linear algebra**, **Fourier transforms**, **random number generation**, and more

> **Why Data Engineers should learn NumPy:**
> Data engineering often requires working with large datasets, doing fast matrix operations, and preparing data for ML. NumPy is **10x to 100x faster** than native Python lists due to vectorization and C-based backend.

2. Installing NumPy

You can install NumPy via pip or conda:

```
pip install numpy
```

Or, using Anaconda:

```
conda install numpy
```

3. Importing NumPy

The standard convention is:

```
import numpy as np
```

4. Why Not Just Use Python Lists?

Feature	Python List	NumPy Array
Type	Heterogeneous	Homogeneous
Speed	Slow	Very fast (compiled C backend)
Memory Usage	High	Efficient (contiguous memory)
Operations	Element-wise not easy	Vectorized and built-in

> **Real-time Scenario:**
> When processing millions of machine sensor readings in real-time, NumPy arrays help you perform transformations (e.g., scaling, offset correction) with minimal delay.

5. Creating Your First NumPy Array

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5])
print(arr)
```

Output:

```
[1 2 3 4 5]
```

Check Array Properties

```
print("Type:", type(arr))         # <class 'numpy.ndarray'>
print("Shape:", arr.shape)        # (5,)
print("Data Type:", arr.dtype)    # int64 (or int32 depending on your system)
```

6. Data Engineer Use Case

Scenario: You have a machine producing 1 reading per second for 5 sensors.

```
sensor_readings = np.array([
    [20.5, 30.2, 40.1, 21.3, 25.5],
    [20.7, 30.1, 40.4, 21.5, 25.7],
    [20.9, 30.0, 40.6, 21.6, 25.9]
])

print("Shape:", sensor_readings.shape)  # (3, 5)
```

This represents 3 seconds of readings for 5 sensors. You can now:

- Calculate averages
- Detect anomalies
- Reshape for time-based processing

---

7. Summary

Concept	Description
`np.array()`	Convert list to NumPy array
`.dtype`	Data type of array
`.shape`	Shape of the array (rows, columns)
`.ndim`	Number of dimensions
`type()`	Object type (should be ndarray)

---

8. Day 1 Task

1. Install NumPy and Jupyter Notebook (if not already done).

2. Convert the following list of hourly temperature readings into a NumPy array:

temps = [72.4, 73.0, 72.5, 74.2, 75.1, 76.5, 74.0]

3. Print:
    - Mean temperature
    - Minimum and maximum
    - Total number of readings
    - Data type and shape
4. BONUS: Create a 3x3 matrix of random machine speeds and print its shape and data type.

---

9. Real-Time Interview Tip

Q: Why would you prefer NumPy over Pandas for initial data cleaning in batch pipelines?

A: NumPy offers faster vectorized operations for raw numerical data. In the ETL layer, we often clean or normalize structured arrays before converting them into DataFrames for downstream analytics.