📆 DAY 08 — NumPy Basics

Goal : Understand arrays (not lists)

1️⃣ Why NumPy? (Very Simple)

Python lists are:

- Slow for math
- Not optimized for ML

Numpy arrays are:

✔️ Fast

✔️ Memory-efficient

✔️ Built for math & ML

2️⃣ Creating NumPy Arrays

```
import numpy as np

a = np.array([1,2,3])
print(a)

# output
[1,2,3]
```

2D Array(Matrix)

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

3️⃣ Shape (VERY IMPORTANT)

```
print(a.shape)   # (3,)
print(b.shape)   # (2,3)
```

- `(rows, columns)`
- ML models depend heavily on shapes

4️⃣ Array vs List (Big Difference)

```
list = [1,2,3]
arr = np.array([1,2,3])

print(list * 2)   # [1, 2, 3, 1, 2, 3]
print(arr * 2)    # [2, 4, 6]
```

👉 NumPy does **element-wise math**

5️⃣ Common Array Creation Functions

```
np.zeros((2,3))      # all zeros
np.ones((3,3))       # all one
np.eye(3)           # identity matrix
np.arange(0,10, 2)    # range
p.linspace(0, 1, 5)   # evenly spaced
```

6️⃣ Indexing & Slicing

```
arr = np.array([10, 20, 30, 40])

print(arr[0])     # 10
print(arr[1:3])   # [20, 30]
```

2D:

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

print(b[0,1])     # 2
```

7️⃣ Vectorized Operations (🔥 ML Core)

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

print(x + y)
print(x * y)
```

No loops ❌

Fast math ✅

8️⃣ Aggregations

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

arr.sum()
arr.mean()
arr.max()
arr.min()
```

used everywhere in ML evaluation.

9️⃣ Reshaping Arrays

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

reshaped = arr.reshape(2,3)
print(reshaped)

# output
[[1 2 3]
 [4 5 6]]
```