1. Numpy is a python library

Numpy stands for numerical python.

Travis Oliphant developed numpy library in the year 2005.

Problem faced by poeple when they only used python for data analysis

1. It was very slow when working on large datasets like weather, stock market data

2. People had to use loops to work on this datasets which was time consuming

3. They used lists but the lists are not memory efficient, not suitable for large data

Numpy was built to:

1. To handle large amount of numbers in little to no time (like millions of rows of data).

2. To overcome the slowness of lists and loops, and invented special arrays

Special arrays where super fast and memory efficient.They are advanced lists

Benefits of Numpy:

1. Speed is 50 - 100x more than lists

2. Uses less memory

3. Easy math operations

4. Used in AI, DS and ML, stock market & finance, medical research, and image processing.

full method: parameters to know:

np.full(shape, fill\_value, dtype=None, order='C', \*, like=None)

Parameter Description

shape Tuple indicating the shape of the array (e.g. (3, 3, 2))

(3, 3, 2)

↑ ↑ ↑

| | └── 2 columns in each row

| └───── 3 rows in each block

└──────── 3 blocks (or 3 matrices)

fill\_value The value to fill the array with (e.g. 25)

dtype (Optional) Data type of the output array (e.g. int, float)

order (Optional) Row-major ('C') or column-major ('F') memory layout

like (Optional) Reference object to define the result's type and behavior

Array properties:

1. Numpy array carries shape, size and type of data, which we must know before performing any operations on it.

e.g. assume you are an data ware house manager,

there is lots of data of company products which you have to keep track of:

1. product name

2. Quantity of a product

3. Price of the product

You must know product name, quantity i.e. shape() of the product, type of data of product

shape method: can be used to know the size of the array

e.g. assume we have an excel table, shape tells us the dimensions of it

size method: returns total numbers of elements in array

e.g. to know what is the size of a dataset

MCQ:

Q1. NumPy arrays are faster than lists mainly because:

a) They store data in different types

b) They are written in C and store data in contiguous memory

c) They use loops internally

d) They can store any Python object

ans: b

MCQ:

Q2. What will be the output of:

import numpy as np

arr = np.arange(2, 10, 3)

print(arr)

a) [2 5 8]

b) [2 5 8 11]

c) [2 5 7]

d) [2 4 6]

ans: a

1. Creating NumPy Arrays (Deep Dive)

Theory:

Creating arrays is the first step in almost any numerical task. Arrays store data in a fast, compact way, and NumPy gives many shortcuts (zeros, ones, full, arange, linspace) to set up datasets without typing everything.

Where it’s used in AI/ML & DS:

Zeros & Ones: Initializing weights or bias arrays in ML models before training.

Full(): Setting all sensor readings to a default value before actual readings come in.

arange(): Generating index values for time series like stock prices or sensor data.

linspace(): Creating smooth points for plotting curves (e.g., sigmoid function).

Lists → Arrays: Converting raw dataset from CSV into NumPy arrays for faster math operations.

Example:

import numpy as np

# Image preprocessing: create a black image (all zeros)

black\_img = np.zeros((100, 100, 3), dtype=np.uint8)

# Simulation: array of timestamps

timestamps = np.arange(0, 10, 0.5)

# ML weight initialization: all small values

weights = np.full((3, 3), 0.01)

MCQ:

Q2. np.linspace(0, 1, 5) generates:

a) [0, 1, 2, 3, 4]

b) [0. , 0.25, 0.5 , 0.75, 1.]

c) [0. , 0.2, 0.4, 0.6, 0.8]

d) Error

ans: b