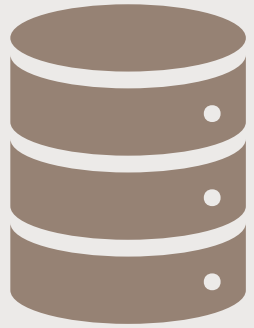
INTRO TO NUMPY

WHAT IS NUMPY

A special tool that helps us work with numbers easily. (data, list, measurements, calculations etc)



Why do we use NumPy instead of normal Python lists? (Speed, efficiency)

QUICK INSTALLATION

```
pip install numpy
```

IMPORT FILES FOR NUMPY

```
import numpy as np
```

COMPARING PYTHON LISTS VS. NUMPY ARRAYS

```
my_list = [1, 2, 3]

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


print("Python List:", my_list)

print("NumPy Array:", my_array)
```

DIFFERENT WAYS TO CREATE ARRAYS

```
np.array([1, 2, 3, 4])  # From a list
np.zeros(5)  # Creates an array of zeros
np.ones((3, 3))  # Creates a 3x3 array of ones
np.arange(0, 10, 2)  # Array with a step
np.linspace(0, 1, 5)  # Evenly spaced numbers
```

HANDS-ON CHALLENGE:

1

Create a NumPy array with numbers from **1 to 20**.

2

Create a **5x5 array** of ones.

3

Create an array of **10 numbers equally spaced out** between 0 and 1.

BASIC ARRAY OPERATIONS

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

print(a + b)  # Add
print(a * b)  # Multiply
print(a ** 2)  # Square
```

FINDING MAX, MIN, AND MEAN:

```
arr = np.array([10, 20, 30, 40])
print(np.max(arr))  # Largest number
print(np.min(arr))  # Smallest number
print(np.mean(arr))  # Average
```

HANDS-ON CHALLENGE:

1

Create two random arrays and multiply them

2

Find the **sum, mean, and maximum** of a random array

INDEXING

- How can we access an element in the array.

```
arr = np.array([10, 20, 30, 40, 50])
print(arr[0])  # First element
print(arr[-1])  # Last element
```

SLICING

```
print(arr[1:4])  # Elements from index 1 to 3
print(arr[:3])   # First three elements
print(arr[-3:])  # Last three elements
```

HANDS-ON CHALLENGE:

1

Create an array of **numbers 1 to 50** and print the **first 10 numbers**.

2

Print only the **even numbers** from the array.

MINI-PROJECT 1: TEMPERATURE DATA ANALYSIS

- Create a NumPy array with **7 temperatures (one for each day of the week)**.
- Find:

  The **highest** and **lowest** temperatures.

  The **average** temperature for the week.

  Convert all temperatures from **Celsius to Fahrenheit** using:

  ```
  F = (C * 9/5) + 32
  ```

MINI-PROJECT 2: RANDOM NUMBER GRID

- Create a **5x5 array** of random numbers.
- Find:

  The **maximum number** in each row.

  The **sum of each column**.

Bonus Challenge:

- Sort each row in **ascending order** using `np.sort()`.

CREATING 2D ARRAYS

```
import numpy as np


# Creating a 2D array (3 rows, 4 columns)
matrix = np.array([[1, 2, 3, 4],
                   [5, 6, 7, 8],
                   [9, 10, 11, 12]])
```

INDEXING 2D ARRAYS

```
import numpy as np

# Creating a 2D array
matrix = np.array([[10, 20, 30],
                   [40, 50, 60],
                   [70, 80, 90]])

# Accessing a single element (row index, column index)
print(matrix[0, 1])  # Output: 20
print(matrix[2, 2])  # Output: 90
```

SLICING 2D ARRAYS

2. Slicing Rows and Columns

- `matrix[start_row:end_row, start_col:end_col]`

python

```python
# Slicing first two rows
print(matrix[0:2, :])

# Slicing first two columns
print(matrix[:, 0:2])

# Extracting a specific section
print(matrix[1:, 1:])
```

3. Selecting an Entire Row or Column

```python
# Selecting an entire row
print(matrix[1, :])  # Second row

# Selecting an entire column
print(matrix[:, 2])  # Third column
```

4. Negative Indexing

```python
# Last row
print(matrix[-1, :])

# Last column
print(matrix[:, -1])
```

HANDS-ON CHALLENGE:

1

Extract the middle 2×2 section of a 4×4 matrix.

2

Reverse the order of rows in matrix using slicing

SORTING & FILTERING WITH NUMPY

```
# Generate random numbers
data = np.random.randint(1, 100, size=(5, 5))
print("Original Data:\n", data)

# Sorting each row
sorted_data = np.sort(data, axis=1)
print("Sorted Rows:\n", sorted_data)

# Filtering values greater than 50
filtered_data = data[data > 50]
print("Filtered Data (values > 50):", filtered_data)
```

HANDS-ON CHALLENGE:

1

Sort the array column-wise.

2

Filter numbers that are **even and greater than 30**.

QUICK INSTALLATION

```
pip install matplotlib
```

IMPORT FILES FOR GRAPHS

```
import matplotlib.pyplot as plt
```

```
import matplotlib.pyplot as plt


# Sample data
x = np.array([1, 2, 3, 4, 5])
y = np.array([2, 4, 6, 8, 10])


# Line plot
plt.plot(x, y, marker='o', linestyle='-
plt.xlabel("X values")
plt.ylabel("Y values")
plt.title("Basic Line Graph")
plt.legend()
plt.show()
```

BASIC DATA VISUALIZATION

- Plotting line graphs
- Creating bar charts
- Adding labels and titles

HANDS-ON CHALLENGE:

1

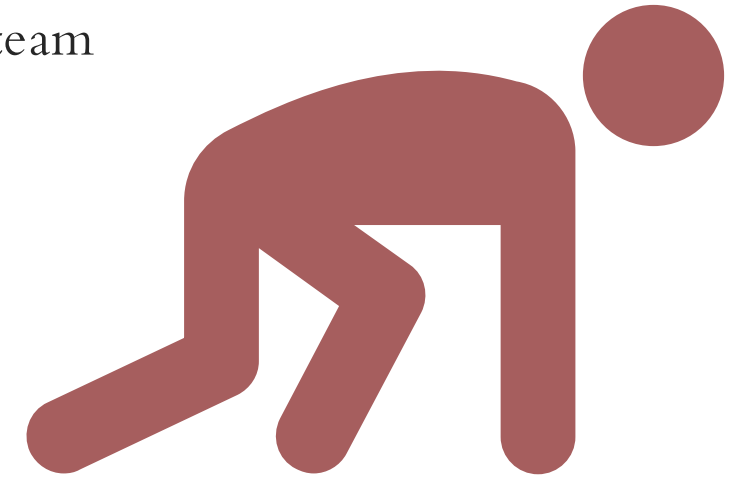
Create a line graph for 5 products with their sales data.

2

Customize the graph (add colors, labels, and gridlines).

- 1. The Pizza Party Budget 🍕 💰
- You're organizing a pizza party for your friends and need to calculate how much it will cost. You have the prices of different pizza sizes and need to find the most cost-effective option.
- Tasks:
- ✅ Store pizza prices in a NumPy array.
- ✅ Find the cheapest and most expensive pizza.
- ✅ Calculate the average cost per pizza.

- 2. The Sports Score Tracker 🏀🏆
- You're tracking the scores of your favorite basketball team over the last 10 games. You want to analyze their performance.
- Tasks:
- ✅ Store the last 10 game scores in a NumPy array.
- ✅ Find the highest and lowest scores.
- ✅ Calculate the average score of the team.
- ✅ Determine if they scored above 100 points in any game.



- 3. The Temperature Analyzer 🌞🌡️
- You're studying the temperature of your city over the past week and want to analyze the data.
- Tasks:
- ✅ Store the temperatures for 7 days in a NumPy array.
- ✅ Find the highest and lowest temperature.
- ✅ Calculate the average temperature.
- ✅ Convert all temperatures from Celsius to Fahrenheit using the formula:
- F = (C * 9/5) + 32

- 4. The Supermarket Price Checker 🛒💳
- You're comparing the prices of 5 grocery items at different supermarkets. You want to find the best deals.
- Tasks:
- ✅ Store the prices of 5 items from 3 different supermarkets in a 2D NumPy array (5*3).
- ✅ Find the cheapest price for each item.
- ✅ Find the most expensive supermarket overall.
- ✅ Calculate the average price of all items.

- 5. The Lucky Lottery Numbers 🎯🎲
- You're generating random lottery numbers and analyzing them.
- Tasks:
- ✅ Create an array of 6 random numbers between 1 and 50.
- ✅ Sort the numbers in ascending order.
- ✅ Output number bigger than 7 in the array

- 6. The Classroom Grades Analyzer 📚🏫
- You're analyzing the grades of students in your class to see how well everyone performed.
- Tasks:
- ✅ Store the grades of 20 students in a NumPy array.
- ✅ Find the highest and lowest grade.
- ✅ Calculate the class average.
- ✅ Count how many students scored above 80.
- ✅ Sort the grades in descending order.

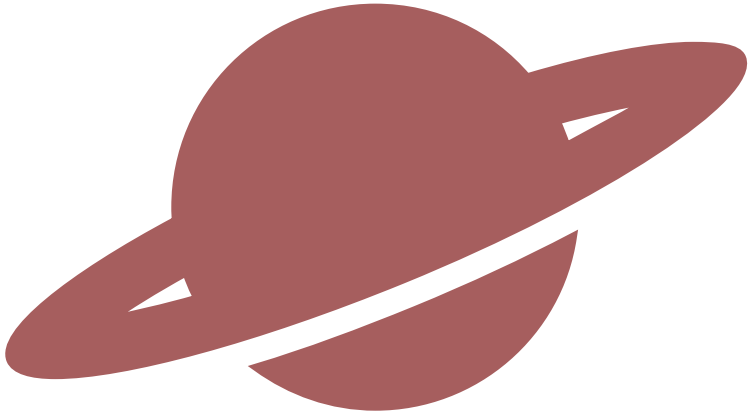
- 7. The Space Travel Calculator 🚀🌌
- You're planning a space journey and need to calculate distances between planets.
- Tasks:
- ✅ Store the distances (in million km) from Earth to different planets in a NumPy array.
- ✅ Find the closest and farthest planet.
- ✅ Calculate the average distance to the planets.
- ✅ Convert all distances to miles (1 km = 0.621 miles).

- 8. The Movie Marathon Data 🎬🎥
- You're tracking how long different movies are to plan a movie marathon.
- Tasks:
- ✅ Store the lengths (in minutes) of 5 movies in a NumPy array.
- ✅ Find the longest and shortest movie.
- ✅ Calculate the total time for the marathon.
- ✅ Convert all movie times to hours and minutes format.

NASA MISSION

The year is **2095**. Earth has sent a group of astronauts to **Planet Xylon**, a newly discovered planet in a distant galaxy. They were supposed to send back **important scientific data**, but their transmission got scrambled due to an **electromagnetic storm**!

NASA has intercepted **a series of jumbled numbers** that seem to contain **vital information** about the planet's atmosphere, temperature, and terrain. Your mission as **NASA's top data scientist team** is to **decode the transmission using NumPy** before the signal is lost forever!



MISSION : DECIPHERING THE TRANSMISSION CODE

- **Problem:**
  NASA received a 1D array of numbers representing atmospheric pressure readings, but some numbers are **out of range and incorrect** due to interference. You need to **filter the valid data** and analyze it.
- 📌 **Task:**
- Convert the received list into a **NumPy array**.
- Identify and remove values **below 100** or **above 2000** (faulty readings).
- Find the **average atmospheric pressure** of Planet Xylon.
- **Divide data by zones**: Split the array into "zones" (e.g., Zone A, B, C) and assign each group a zone to clean and analyze.
- **Add faulty sensor IDs**: Add a second array that shows which sensor sent which reading. Kids match cleaned data to good sensors only.
- **Find more stats**: Calculate **median**, min, max, and **visualize** different zoneswith a mini pressure plot using `matplotlib`.

- pressure_readings = [980, 1023, 45, 1980, 2075, 1500, 80, 1999, 3001, 1200,
- 999, 85, 1450, 1760, 90, 2005, 1050, 500, 1890, 2300,
- 30, 1700, 850, 2100, 1950]

MISSION 2: ANALYZING XYLON'S TEMPERATURE READINGS

- **Problem:**
  The astronauts managed to send a week's worth of temperature data, but it's in **Celsius**, and the mission control system reads only **Fahrenheit**. You need to **convert** it!
- 📌 **Task:**
- Store the **7 temperature values** in a NumPy array.
- Convert them to Fahrenheit using the formula:
- `F = (C * 9/5) + 32`
- Find the **highest and lowest temperatures** recorded.
- **Add location tags**: Store temperatures along with time/location labels using a structured array or a 2D array.
- **Trend prediction**: Ask them to predict whether temperatures are **rising or falling** over the week using a simple plot.
- temps_celsius = [-20, -15, -10, -5, 0, 5, 10]

MISSION 3: MAPPING THE SURFACE OF PLANET XYLON

- **Problem:**
  The astronauts used a 5x5 scanner to analyze the terrain of Xylon. The data represents **elevation levels** in meters, but they need it sorted for analysis.
- 📌 **Task:**
- Generate a **5x5 NumPy array** with random elevation values **between 100 and 500 meters**.
- Find the **highest point** on the planet.
- Sort each row in **ascending order** for terrain analysis.
- **Assign rows**: Each team member sorts and analyzes one row. Then combine for the full map.
- Visualise your findings using a line graph
- **Find flat areas**: Challenge them to find any row/column with **minimal elevation difference**.

FINAL MISSION: PREDICTING OXYGEN LEVELS

- **Problem:**
  NASA's life support systems need to know if the astronauts can breathe on Planet Xylon. Scientists predict that oxygen levels are related to **atmospheric pressure**. The formula they use is:
- Oxygen Level=Pressure/50
- 📌 **Task:**
- Take the **clean atmospheric pressure data** from **Mission 1**.
- Use NumPy **array operations** to calculate the **estimated oxygen levels**.
- Find out if **any levels drop below 3%**—if so, the astronauts may need **extra oxygen tanks!**
- **Team oxygen audit**: Assign each group member a chunk of the data and ask them to report risky zones.
- **Write a warning system**: Kids write a function that **alerts** if any reading falls below safe oxygen thresholds.
- **Plot pressure vs oxygen**: Graph the relationship to visualize how pressure affects oxygen.