DATA 30

WEEK 1

Day 1: Python Basics

📌 **Topics to Cover**

- **Python Introduction** – What is Python? Why it's popular (simplicity, libraries, wide usage in ML/AI, web dev, etc.).
- **Installation & Setup** – Install Python, Jupyter Notebook/VS Code, and basic setup.
- **Data Types** – Integers, Floats, Strings, Booleans. Learn type checking with `type()`.
- **Variables & Type Casting** – Variable naming rules, assigning values, type conversion (e.g., `int("10")`).
- **Input & Output** – `input()`, `print()`, and f-strings for formatted output.
- **Basic Operators** – Arithmetic (`+ - * / // % **`), Comparison (`== != > <`), Logical (`and, or, not`).

👉 **Task:**
Write a program that takes user input for name and age and prints:

```
Hello {name}, you are {age} years old!
```

Day 2: Control Flow

📌 **Topics to Cover**

- **Conditional Statements** – if, elif, else (decision making).
- **Loops** – for loop (iterating over ranges & collections), while loop (runs until condition breaks).

- **Loop Control** – `break`, `continue`, `pass` with examples.
- **Range Function** – Generate sequences (`range(start, stop, step)`).

👉 **Task:**
Write a program that prints the multiplication table of a number (entered by the user) using both **for** and **while** loops.

Day 3: Data Structures

📌 **Topics to Cover**

- **Lists** – Creating lists, indexing, slicing, common methods (`append`, `remove`, `sort`, `reverse`).
- **Tuples** – Immutable lists, tuple unpacking, when to use.
- **Sets** – Storing unique items, set operations (union, intersection, difference).
- **Dictionaries** – Key-value pairs, methods (`keys`, `values`, `items`, `get`).

👉 **Task:**
Create a dictionary with student names as keys and marks as values. Print the name of the student with the highest marks.

Day 4: Functions & File Handling

📌 **Topics to Cover**

- **Functions** – Defining, calling, passing parameters, returning values.
- **Default & Keyword Arguments** – Flexible ways to call functions.
- **Lambda Functions** – Small anonymous functions.
- **File Handling** – Reading & writing text files using `with open("file.txt", "r/w/a")`.

👉 **Task:**
Write a function to check if a number is prime. Use it to generate prime numbers between 1–100 and save them in a file.

Day 5: OOPs in Python (imp)

📌 **Topics to Cover**

- **Introduction to OOP** – Why OOP is used (real-world modeling, code reusability).
- **Classes & Objects** – Defining a class, creating objects.
- **`__init__` Method** – Constructor for initializing objects.
- **Methods** – Instance methods, Class methods (`@classmethod`), Static methods (`@staticmethod`).
- **Encapsulation** – Public, Protected (`_var`), and Private (`__var`) variables.

👉 **Task:**
Create a `Student` class with attributes `name` and `marks`. Add a method to calculate the average marks.

Day 6: OOPs Advanced

📌 **Topics to Cover**

- **Inheritance** – Reusing code from parent class. Types: Single, Multiple, Multilevel.
- **Method Overriding** – Redefining a parent method in a child class. Use `super()` to call parent version.
- **Polymorphism** – Same method behaving differently depending on context. (Duck typing, operator overloading).
- **Abstraction** – Hiding implementation using abstract classes (`abc` module).

👉 **Task:**
Create a base class `Shape` with a method `area()`. Inherit `Rectangle` and `Circle` classes from it and override the `area()` method to calculate respective areas.

Day 7: NumPy Basics & Operations

📌 **Topics to Cover**

- **What is NumPy?** – Fast numerical computation library. Works with arrays, unlike slow Python lists.
- **Installing NumPy** – `pip install numpy`.
- **Creating Arrays** – `np.array()`, `np.arange()`, `np.linspace()`, `np.zeros()`, `np.ones()`.
- **Array Attributes** – `shape`, `size`, `ndim`, `dtype`.
- **Indexing & Slicing** – Accessing elements, subarrays.
- **Reshaping** – `reshape()`, `flatten()`.
- **Mathematical Operations** – Element-wise + broadcasting.
- **Aggregate Functions** – `sum`, `min`, `max`, `mean`, `std`.
- **Random Module** – `np.random.rand()`, `np.random.randint()`, `np.random.normal()`.

👉 **Tasks:**

1. Create a 3x3 matrix using `np.arange()` and print its shape, size, and dimension.
2. Generate 100 random numbers using `np.random.normal()` and calculate mean & standard deviation.

⚡ **End Goal:** After this plan, you'll have a **solid Python foundation** (syntax, logic, data structures, file handling), **clear understanding of OOPs** (essential for advanced coding & interviews), and **hands-on NumPy** (step toward data science, ML, and DL).

Reference Links:
https://youtu.be/UrsmFxEIp5k?si=PMKegP8R8ADd6wDt
https://youtu.be/nLRL_NcnK-4?si=1cjy3MsaOq8OVY_U