**Training Day - 2**

**Give me some idea on what is a Data type. What are Primitive Data types and Non-Primitive Data types. Explain with some examples.**

**What is a Data Type?**

A **data type** is an attribute of data that tells the computer how to interpret and store that data. It defines the kind of value a variable can hold and what operations can be performed on that value. Data types determine the size, range, and operations that can be performed on a variable.

**Types of Data Types**

There are two main categories of data types:

1. **Primitive Data Types**
2. **Non-Primitive Data Types**

**1. Primitive Data Types**

Primitive data types are the most basic data types in programming languages. They represent simple values and have a fixed size. These data types are built-in, and they are not created by the user.

**Examples of Primitive Data Types:**

1. **Integer (int)**:
   * Used to represent whole numbers, both positive and negative.
   * Example: int age = 25;
2. **Float (float)**:
   * Used to represent decimal numbers or real numbers.
   * Example: float temperature = 36.6;
3. **Double (double)**:
   * Similar to float, but it has a higher precision for representing decimal values.
   * Example: double pi = 3.14159265359;
4. **Character (char)**:
   * Represents a single character or symbol.
   * Example: char grade = 'A';
5. **Boolean (bool)**:
   * Represents a value of either true or false.
   * Example: bool isOpen = true;
6. **String (in some languages)**:
   * Represents a sequence of characters, but it can be considered as a primitive type in some languages (like Python) or as a non-primitive type in others (like Java).
   * Example: String name = "John"; (Java)

**2. Non-Primitive Data Types**

Non-primitive data types (also called **reference types**) are more complex and are created by the programmer. They can store multiple values and have more functionality. These types can vary in size and behavior, depending on how they're structured and used.

**Examples of Non-Primitive Data Types:**

1. **Array**:
   * A collection of elements, typically of the same data type, stored in contiguous memory locations.
   * Example: int[] numbers = {1, 2, 3, 4};
2. **Class**:
   * A blueprint for creating objects that hold data and methods. Classes can represent real-world entities.
   * Example:
   * class Person {
   * String name;
   * int age;
   * }
   * Person p1 = new Person();
3. **Interface**:
   * Defines a contract that a class must follow. It specifies methods without implementing them, and the implementing class provides the logic for those methods.
   * Example:
   * interface Animal {
   * void sound();
   * }
   * class Dog implements Animal {
   * public void sound() {
   * System.out.println("Bark");
   * }
   * }
4. **String (in some languages)**:
   * Though **String** is considered a primitive in some languages like Python, it’s a non-primitive in languages like Java, where it is an object with more functionality (methods for string manipulation, etc.).
   * Example: String message = "Hello, World!"; (Java)
5. **List, Set, Map (Collections)**:
   * Data structures that can store multiple values, with varying ways of handling the data (e.g., lists allow duplicates, maps store key-value pairs).
   * Example:
   * List<String> fruits = new ArrayList<>();
   * fruits.add("Apple");
   * fruits.add("Banana");