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

**What is a Data Type?**

A data type is a classification that specifies which type of value a variable can hold. It defines the operations that can be performed on the data, the meaning of the data, and the way values of that type can be stored. Data types are essential in programming because they help the compiler or interpreter understand how to handle the data and ensure that operations are performed correctly.

Types of Data Types

Data types are broadly categorized into two types:

**1. Primitive Data Types**

**2. Non-Primitive Data Types**

1. Primitive Data Types

Primitive data types are the most basic data types in a programming language. They are predefined by the language and are used to represent simple values. These data types are immutable, meaning their values cannot be changed after they are created. They are also stored directly in memory.

**Examples of Primitive Data Types:**

* **Integer (`int`):** Represents whole numbers (positive, negative, or zero).
* **Example:** `int age = 25;`
* **Size:** Typically 4 bytes (32 bits).

**Floating-Point (`float`, `double`):** Represents decimal numbers.

* **`float`:** Single-precision floating-point (e.g., `float pi = 3.14;`).
* **`double`:** Double-precision floating-point (e.g., `double pi = 3.14159;`).
* **Size:** `float` is 4 bytes, `double` is 8 bytes.
* **Character (`char`):** Represents a single character.
* **Example:** `char grade = 'A';`
* Typically 1 byte (8 bits).
* **Boolean (`bool`):** Represents true or false values.
* **Example: `**bool isRaining = true;`
* **Size:** Typically 1 byte.
* **Void (`void`):** Represents the absence of a value. Often used in functions that do not return a value.
* **Example:** `void displayMessage() { ... }`

**Others (language-specific):**

* **`short`:** Smaller integer (e.g., `short x = 10;`).
* **`long`: L**arger integer (e.g., `long population = 8000000000L;`).
* **`byte`:** Very small integer (e.g., `byte b = 127;`).

2. Non-Primitive Data Types

Non-primitive data types are also called reference types or derived data types. They are not predefined; instead, they are created by the programmer. These data types are used to store complex data and are stored as references to memory locations (not directly in memory).

**Key Characteristics of Non-Primitive Data Types:**

* They are mutable (can be modified after creation).
* They can hold multiple values or a collection of values.
* They are stored in the heap memory, and the variable holds a reference (address) to the memory location.

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

* String: Represents a sequence of characters.
* Example: `string name = "John Doe";`
* Strings are technically objects in many languages (e.g., Java, Python).
* Arrays: A collection of elements of the same data type.
* Example: `int[] numbers = {1, 2, 3, 4, 5};`
  + Classes: User-defined data types that encapsulate data and methods.

**Example:**

```java

class Person {

String name;

int age;

}

Person p1 = new Person();

```

* + Interfaces: Define a contract for classes (used in object-oriented programming).

**Example:**

```java

interface Drawable {

void draw();

}

```

* Structures (`struct`): Similar to classes but typically used for lightweight objects (e.g., in C).

**Example:**

```c

struct Point {

int x;

int y;

};

struct Point p1;

```

* + Lists, Sets, Maps: Collections of data (e.g., in Python, Java, or C#).
  + Example: `List<int> numbers = new List<int> {1, 2, 3};`
  + Objects: Instances of classes or structures.
  + Example: `Object obj = new Object();`

**Key Differences Between Primitive and Non-Primitive Data Types**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Primitive Data Types** | **Non-Primitive Data Types** |
| **Definition** | Predefined by the language | Created by the programmer |
| **Storage** | Stored directly in memory | Stored as references in memory |
| **Mutability** | Immutable (cannot be changed) | Mutable (can be changed) |
| **Size** | Fixed size (e.g., int is 4 bytes) | Size depends on the data structure |
| **Examples** | int, float, char, bool | String, Array, Class, List |

**Example in Code**

**Primitive Data Types:**

```java

int age = 30; // Integer

double salary = 50000.75; // Double

char grade = 'A'; // Character

boolean isEmployed = true; // Boolean

```

**Non-Primitive Data Types:**

```java

String name = "Alice"; // String

int[] numbers = {1, 2, 3, 4, 5}; // Array

class Person { // Class

String name;

int age;

}

Person p1 = new Person();

```

**Summary**

* + Primitive Data Types: Basic, immutable, and directly stored in memory. Examples include `int`, `float`, `char`, and `bool`.
  + Non-Primitive Data Types: Complex, mutable, and stored as references. Examples include `String`, `Array`, `Class`, and `List`.

Understanding these data types is crucial for writing efficient and error-free code. Primitive types are used for simple values, while non-primitive types are used for more complex data structures and objects.