**TASK #02**

**1. Data Types and Variables:**

* **Data Types:** Data types define the kind of data a variable can hold. Dart supports various built-in data types:
  + **Numbers:**
    - int: Whole numbers (e.g., 10, -5)
    - double: Floating-point numbers (e.g., 3.14, -12.5)
  + **Strings:** Sequences of characters enclosed in single (') or double (") quotes (e.g., "Hello, world!", 'This is a string')
  + **Booleans:** Represent logical values (true or false)
  + **Lists:** Ordered collections of items enclosed in square brackets [] (e.g., [1, 2, 3, "apple"])
  + **Maps:** Unordered collections of key-value pairs using curly braces {} (e.g., {"name": "John", "age": 30})
  + **Sets:** Unordered collections of unique items (e.g., {1, 2, 2, "unique"})
  + **Null:** Represents the absence of a value
* **Variables:** Variables are named containers that store data. To declare a variable:
  + Specify the data type (optional, var for automatic type inference)
  + Use a valid identifier name (letters, numbers, underscores; starts with a letter)
  + Assign a value using the = operator

Dart

int age = 25; // Integer variable

String name = "Alice"; // String variable

bool isLoggedIn = true; // Boolean variable

List<int> numbers = [1, 2, 3]; // List of integers

Map<String, String> person = {"name": "Bob", "city": "New York"}; // Map with string keys and values

**2. Operators and Control Flow:**

* **Operators:** Symbols used to perform operations on data:
  + Arithmetic operators (+, -, \*, /, % for modulo)
  + Comparison operators (==, !=, <, >, <=, >=)
  + Logical operators (&& for AND, || for OR, ! for NOT)
  + Assignment operators (=, +=, -=, \*=, etc.)
* **Control Flow Statements:** Control the flow of program execution:
  + if statements: Execute code conditionally based on a boolean expression
  + if...else statements: Provide alternative execution paths
  + switch statements: Execute code based on a matched value
  + for loops: Repeat a block of code a specific number of times
  + while loops: Repeat a block of code as long as a condition is true
  + do...while loops: Execute a block of code at least once, then repeat as long as a condition is true

**3. Functions:**

* Reusable blocks of code that perform a specific task.
* Defined using the func keyword, followed by a name, optional parameters, and a return type (optional, void for no return).
* Functions can call other functions and access variables from their enclosing scope.

Dart

void greet(String name) {

print("Hello, $name!");

}

int add(int a, int b) {

return a + b;

}

**4. Nullability:**

* Dart is null-safe by default, meaning variables must be explicitly initialized with a value or marked as nullable using ?.
* This helps prevent runtime crashes due to null pointer exceptions.
* Use the null-check operator ! to access properties of a nullable variable only if it's not null.

Dart

String? message; // Nullable string variable

if (message != null) {

print(message.toUpperCase()); // Access only if not null

}

**5. Collections:**

* Used to organize and manage groups of data:
  + **Lists:** Order matters, can contain duplicates
  + **Sets:** Unordered collection of unique items
  + **Maps:** Key-value pairs, efficient for lookup by key
* You can iterate over collections using for loops or forEach methods.

**1. Just-In-Time (JIT) compilation:**

* Happens **at runtime**. The Dart VM translates your code into machine code **as it's needed**.
* This allows for **fast development cycles** because changes are reflected immediately (hot reload in Flutter).
* However, there's an initial overhead during startup as the code is compiled on the fly.
* JIT compilation can also be **optimized** based on how the code is actually used, potentially leading to **higher peak performance** in the long run.

**2. Ahead-of-Time (AOT) compilation:**

* Happens **before** the application is run. The Dart code is translated into machine code for the target platform **during the build process**.
* This results in **faster startup times** because there's no need for on-the-fly compilation.
* AOT-compiled code is generally more **predictable** in terms of performance.
* However, AOT compilation might not be able to optimize as well as JIT since it doesn't have access to runtime information.

**What makes Dart special?**

* **Single codebase, multiple platforms:** Dart allows you to write code once and deploy it on various platforms like web, mobile (Flutter), desktop, and servers.
* **Hot reload:** In development mode with JIT, changes in your code are reflected almost instantly in the running application, making development very productive.
* **Strong typing and null safety:** Dart enforces type safety, which helps catch errors early and prevents runtime crashes due to null pointer exceptions.
* **Isolates:** Dart supports isolates, which are lightweight processes that can run concurrently for better performance and memory management.

These features, along with its clean syntax and rich ecosystem, make Dart a versatile and developer-friendly language.