🧪 Section 1: Dart Basics

🎯 Objectives

By the end of this lab, students will be able to:

- Declare and use variables of different types.
- Understand null safety and nullable variables.
- Apply control structures (if, loops, switch).
- Define and call functions, including optional parameters and arrow syntax.

🔹 1. Variables, Types, and Null Safety

🧠 Concepts

- Dart is statically typed, but supports **type inference** with `var`.
- Variables can be **nullable** (`String?`) or **non-nullable** (`String`).
- Once declared, non-nullable variables cannot hold `null` values.

💡 Example

```
void main() {
  int age = 25;                // Non-nullable
  double height = 1.75;
  String name = 'Alice';
  bool isStudent = true;

  // Nullable variable
  String? nickname;
  nickname = null;

  print('$name is $age years old, height: $height m');
  print('Nickname: $nickname');
}
```

Explanation:
Here `nickname` is declared as `String?`, which allows it to be `null`.
If you remove the `?`, the compiler will warn you.

🧩 Exercise 1.1

Create variables for:

- A product’s name (`String`)
- Its price (`double`)
- Its quantity (`int`)
- A nullable description (`String?`)

Then print a message like:

Product: Laptop (Qty: 2, Price: 1500.0) – Description: High-end gaming laptop

---

🔷 2. Control Structures

🧠 Concepts

Dart supports:

- **if / else** for conditional execution
- **for** and **while** loops for repetition
- **switch / case** for multi-way branching

💡 Example

```
void main() {
  int score = 82;

  if (score >= 90) {
    print('Excellent');
  } else if (score >= 75) {
    print('Good');
  } else {
    print('Needs improvement');
  }

  for (int i = 1; i <= 3; i++) {
    print('Attempt $i');
  }

  switch (score ~/ 10) {
    case 10:
    case 9:
      print('Grade: A');
      break;
    case 8:
      print('Grade: B');
      break;
    default:
      print('Grade: C or below');
  }
}
```

Explanation:

- `~/` does integer division (e.g., 82 ~/ 10 = 8).
- `switch` can group cases (9 and 10 → Grade A).
- Loops use standard C-style syntax.

🧩 Exercise 2.1

Write a Dart program that:

1. Declares a variable `number = 7`.
2. Prints:
   - `"Even number"` if it's even,
   - "Odd number" otherwise.
3. Prints all numbers from `1` to `number` using a loop.

💡 *Hint:* Use for (int i = 1; i <= number; i++) { ... }

🧩 Exercise 2.2 (Challenge)

Ask the user (simulate using a variable) for a grade between 0–100.
Use a `switch` statement to print:

- "A" for 90–100
- "B" for 80–89
- "C" for 70–79
- "F" for anything else

💡 Use `score ~/ 10` as in the example.

🔷 3. Functions, Optional Parameters, and Arrow Syntax

🧠 Concepts

- Functions can have **positional** or **named** optional parameters.
- You can assign **default values**.
- Arrow syntax (`=>`) is shorthand for single-expression functions.

💡 Example

```
int add(int a, [int b = 0]) => a + b; // optional positional

void greet(String name, {String title = 'Mr./Ms.'}) {
  print('Hello $title $name');
}
```

```
void main() {
  print(add(5, 3));    // 8
  print(add(5));       // 5
  greet('Alice');      // Hello Mr./Ms. Alice
  greet('Bob', title: 'Dr.'); // Hello Dr. Bob
}
```

Explanation:

- `[int b = 0]` → optional positional parameter.
- `{String title = 'Mr./Ms.'}` → optional named parameter.
- => replaces { return expression; }.

🧩 Exercise 3.1

Write a function `calculateArea` with **two optional parameters**:
`width` and `height`, both default to `1`.
Return `width * height` using **arrow syntax**.

In `main()`, call:

```
print(calculateArea());         // 1
print(calculateArea(5));        // 5
print(calculateArea(5, 3));     // 15
```

🧩 Exercise 3.2 (Challenge)

Create a function greetUser(String name, {int age = 0}) that:

- Prints "Hello <name>".
- If age > 0, prints "You are <age> years old."

Try calling it with and without the `age` argument.

🏁 Summary

In this lab, you learned how to:
✅ Use variables and null safety
✅ Apply control flow and loops
✅ Create functions with flexible parameters