Lesson 1: Introduction to Dart and Variables

Goal: Understand the basic syntax and declare variables in Dart.

- **Mentor Task:**
    1. Explain what Dart is and why it's used (e.g., for building Flutter apps).
    2. Show how to declare variables using `var`, `int`, `double`, and `String`.
    3. Demonstrate initializing and printing variables.
    4. Guide the learner in declaring and printing their own variables.
    5. Ensure the learner can change and print new variable values.

Lesson 2: Data Types and Type Inference

Goal: Learn about Dart's data types and how type inference works.

- **Mentor Task:**
    1. Explain common data types: `int`, `double`, `String`, `bool`.
    2. Show how Dart uses type inference (e.g., using `var`).
    3. Have the learner write examples of different data types.
    4. Guide them in observing type inference in action by printing out types.
    5. Ensure the learner understands why and when to use specific types.

Lesson 3: Control Flow (If/Else)

Goal: Use conditional statements to control program flow.

- **Mentor Task:**
    1. Explain basic `if`, `else if`, and `else` syntax.
    2. Show how conditions can be based on boolean expressions.
    3. Write a simple program that makes decisions based on user input (e.g., age check for drinking).
    4. Let the learner modify the program with new conditions.
    5. Help them test and debug any issues in the logic.

Lesson 4: Loops (For and While)

Goal: Implement `for` and `while` loops in Dart.

- **Mentor Task:**
    1. Explain how a `for` loop works with a counting example.
    2. Guide the learner in creating their own `for` loop (e.g., printing numbers 1 to 10).
    3. Demonstrate how a `while` loop works with a condition.
    4. Let the learner write a `while` loop that runs until a condition is met.
    5. Ensure they understand the difference between `for` and `while` loops.

Lesson 5: Functions

Goal: Define and call functions in Dart.

- **Mentor Task:**
    1. Explain what functions are and why they're useful (for reusability).
    2. Show how to define a simple function that takes parameters and returns a value.
    3. Guide the learner in writing their own function (e.g., a function to calculate the area of a rectangle).
    4. Help them call the function with different inputs and print results.
    5. Ensure the learner understands function signatures and return types.

Lesson 6: Lists and Collections

Goal: Work with lists to store and manipulate collections of data.

- **Mentor Task:**
    1. Explain what lists are and how to declare them in Dart (e.g., `List<int>`).
    2. Show how to add, access, and remove elements from a list.
    3. Guide the learner in creating a list and iterating over it using a `for` loop.
    4. Let them practice modifying the list (e.g., adding and removing elements).
    5. Ensure they understand common list methods (`add`, `remove`, `length`, etc.).

Lesson 7: Map Data Structures

Goal: Use maps to store key-value pairs.

- **Mentor Task:**
    1. Introduce maps and explain when to use them.
    2. Show how to declare a map and access values using keys.

3. Guide the learner in creating a map (e.g., a phone book with names and numbers).
4. Have them practice adding, updating, and removing key-value pairs.
5. Let them iterate over the map and print key-value pairs.

Lesson 8: Object-Oriented Programming (Classes and Objects)

Goal: Understand how to define and use classes in Dart.

- **Mentor Task:**
    1. Explain the basics of object-oriented programming (OOP) and the importance of classes.
    2. Show how to create a simple class with properties and methods (e.g., a `Car` class).
    3. Guide the learner in creating an object from the class and using its methods.
    4. Have them modify the class by adding new methods or properties.
    5. Ensure they understand how to create multiple objects and interact with them.

Lesson 9: Constructors and Named Parameters

Goal: Learn about constructors and how to use named parameters in Dart.

- **Mentor Task:**
    1. Explain the purpose of constructors in initializing objects.
    2. Show how to define a constructor and use named parameters for clarity.
    3. Guide the learner in modifying a class to include a constructor (e.g., a `Person` class with name and age).
    4. Let them instantiate the class with different values using named parameters.
    5. Ensure they understand the benefits of named parameters for readability and safety.

Lesson 10: Error Handling (Try, Catch, Finally)

Goal: Handle exceptions and errors gracefully in Dart programs.

- **Mentor Task:**
    1. Explain the `try`, `catch`, and `finally` blocks for error handling.
    2. Show how to catch an error (e.g., dividing by zero) and print an error message.
    3. Guide the learner in writing their own `try-catch` block for common errors.

4. Let them test different error scenarios and ensure the program doesn’t crash.
5. Review the importance of error handling in making programs robust.

Teaching Instructions