**Reg No: FA22-BCS-090**

Table of Contents

[1 Dart Introduction: 3](#_Toc193021756)

[1.1 Important Points of Dart: 3](#_Toc193021757)

[1.2 Advantages of Dart 3](#_Toc193021758)

[2 Variables: 3](#_Toc193021759)

[2.1 Data Types in Dart 4](#_Toc193021760)

[**2.1.1** Numeric Types (int, double): 4](#_Toc193021761)

[2.1.2 String Type (String):. 4](#_Toc193021763)

[2.1.3 Boolean Type (bool): 5](#_Toc193021764)

[2.1.4 User Input 5](#_Toc193021765)

[2.1.5 Dynamic Type (dynamic, var):. 6](#_Toc193021766)

[3 Scope of Variables: 6](#_Toc193021767)

[3.1 Global Scope: 6](#_Toc193021768)

[3.2 Local Scope (Function Scope): 7](#_Toc193021769)

[3.3 Block Scope: 7](#_Toc193021770)

[4 Comments: 8](#_Toc193021771)

[4.1 Single-line Comments (//): 8](#_Toc193021772)

[4.2 Multi-line Comments (/\* ... \*/) 9](#_Toc193021773)

[5 Collections: 9](#_Toc193021774)

[5.1 Queue: 9](#_Toc193021775)

[5.2 Enum: 10](#_Toc193021776)

[5.3 Map Type (Map): 10](#_Toc193021777)

[5.4 List Type (List) 11](#_Toc193021778)

[6 Control Flow: 11](#_Toc193021779)

[6.1 If: 11](#_Toc193021780)

[6.2 Else: 12](#_Toc193021781)

[6.3 else if: 12](#_Toc193021782)

[6.4 Switch: 13](#_Toc193021783)

[7 Loops: 13](#_Toc193021784)

[7.1 While Loop 13](#_Toc193021785)

[7.2 do-while loop: 13](#_Toc193021786)

[7.3 for loop: 14](#_Toc193021787)

[7.4 For Each Loop: 15](#_Toc193021788)

[7.5 for-in loop: 15](#_Toc193021789)

[8 Functions: 15](#_Toc193021790)

[8.1 Function Definition: 15](#_Toc193021791)

[8.2 **Positional parameters**. 16](#_Toc193021792)

[8.3 Optional Positional Parameters: 17](#_Toc193021793)

[8.4 List Parameter (Dynamic Type): 17](#_Toc193021794)

[8.5 List Parameter (Fixed Type) 18](#_Toc193021795)

[8.6 Required Positional Parameter: 18](#_Toc193021796)

[8.7 Nullable Parameter with Default Value: 19](#_Toc193021797)

[9 Arithmetic Operations in Dart: 19](#_Toc193021798)

[9.1 Addition (+): 19](#_Toc193021799)

[9.2 Subtraction (-): 20](#_Toc193021800)

[9.3 Multiplication (\*): 20](#_Toc193021801)

[9.4 Division (/): 21](#_Toc193021802)

[9.5 Integer Division (~/): 21](#_Toc193021803)

[9.6 Modulus (%): 22](#_Toc193021804)

[10 Exception Handling: 22](#_Toc193021805)

[10.1 try block: 23](#_Toc193021806)

[10.2 catch Block: 23](#_Toc193021807)

[10.3 finally Block 24](#_Toc193021808)

[11 Practice Problems: 24](#_Toc193021809)

[11.1 Find Factorial: 24](#_Toc193021810)

[11.2 Check if a Number is Even or Odd: 26](#_Toc193021811)

[11.3 Sum of First N Natural Number: 27](#_Toc193021812)

[11.4 Try-Catch for Invalid Input: 27](#_Toc193021813)

[11.5 Print Numbers from 1 to 10: 28](#_Toc193021814)

[11.6 Check if a Number is Positive, Negative, or Zero: 28](#_Toc193021815)

[11.7 Find the Sum of Two Numbers: 29](#_Toc193021816)

[11.8 Function to Find the Maximum of Two Numbers: 30](#_Toc193021817)

[11.9 Function to Check if a Number is Prime: 31](#_Toc193021818)

# Dart Introduction:

Dart is a programming language developed by Google, mainly used for building mobile, web, and desktop applications. It is the primary language for Flutter, a popular framework for creating cross-platform mobile apps. Dart is easy to learn, especially for those familiar with languages like Java, JavaScript, or C#. It is an object-oriented language, meaning it organizes code into objects and classes. Dart supports both just-in-time (JIT) and ahead-of-time (AOT) compilation, making it fast and efficient. One of Dart’s key features is its strong and flexible type system, which helps catch errors early. It also has a garbage collector to manage memory automatically. Dart’s syntax is clean and simple, making coding easier. Dart is used in Flutter to create beautiful, high-performance mobile apps for Android and iOS with a single codebase. It also supports web and desktop applications. Overall, Dart is a powerful and modern programming language that helps developers build smooth, fast, and reliable applications. Whether you are a beginner or an experienced developer, learning Dart can be a great choice for app development.

## Important Points of Dart:

* **Object-Oriented** – Dart supports object-oriented programming with classes and inheritance.
* **Strongly Typed** – Dart uses both static and dynamic typing, allowing flexible type definitions.
* **Null Safety** – Prevents null reference errors with sound null safety features.
* **Concurrency with Isolates** – Uses isolates instead of threads for better parallel execution.
* **JIT & AOT Compilation** – JIT enables hot reload during development, while AOT improves app performance.
* **Garbage Collection** – Dart manages memory automatically, preventing memory leaks.
* **Platform Independence** – Can be used for mobile, web, and desktop applications.
* **Asynchronous Programming** – Uses async and await for handling asynchronous operations efficiently.
* **Rich Standard Library** – Offers extensive built-in libraries for handling collections, math, and I/O operations.
* **Used in Flutter** – Dart is the primary language for developing Flutter applications.

## Advantages of Dart

* **Fast Development** – Supports hot reload, allowing instant UI updates.
* **Optimized Performance** – AOT compilation makes Dart applications run faster.
* **Cross-Platform** – Develop apps for iOS, Android, web, and desktop using a single codebase.

**Easy to Learn** – Syntax is simple and familiar to developers with JavaScript or Java experience.

**Better UI Development** – Used with Flutter, providing smooth animations and rich UI components.

**Secure and Reliable** – Features like null safety enhance app stability and prevent runtime errors.

**Open-Source** – Free to use and backed by Google with an active community.

# Variables:

A variable in Dart is a named storage location that holds a value. It can store different data types like numbers, strings, and objects. Variables can be mutable (changeable) or immutable (fixed using final or const).

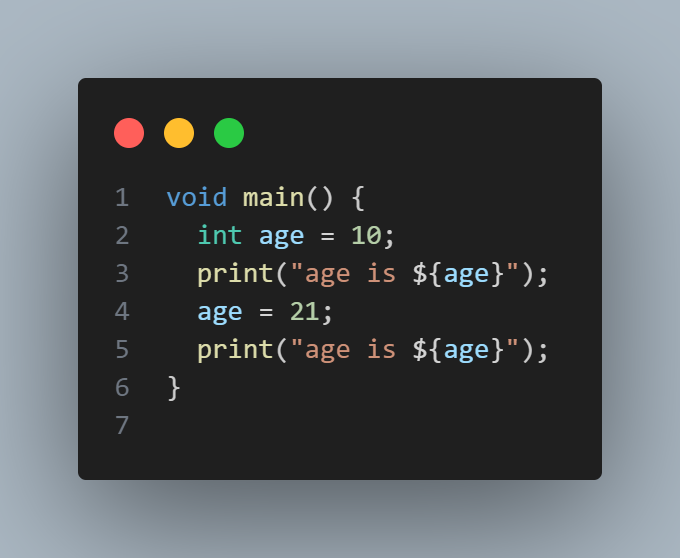


Figure 1 Variables

## Data Types in Dart

### Numeric Types (int, double):

### int stores whole numbers, for example, int num1 = ;. double stores decimal numbers, for example, double num2 = 1.5.

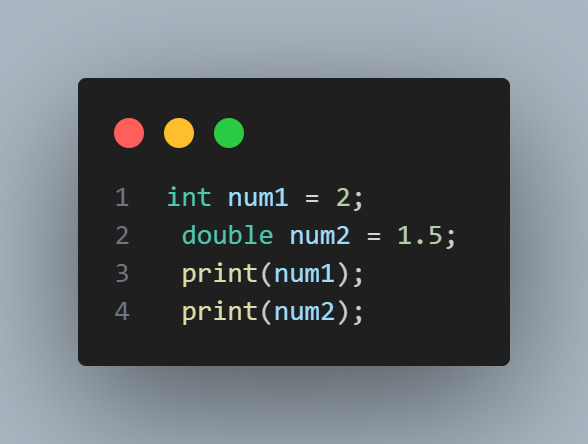
****

Figure 2 numeric Data Type

# 2.1.2 String Type (String) Stores text values, for example, String name = "Zain".



Figure 3 String Data Type

# 2.1.3 Boolean Type (bool): Holds true or false values, for example, bool is Active = true.



Figure 4 bool Data Type

### 2.1.4 User Input

User input in Dart is collected using the stdin. readLineSync() function from the dart: io library. This function reads input as a string from the console.

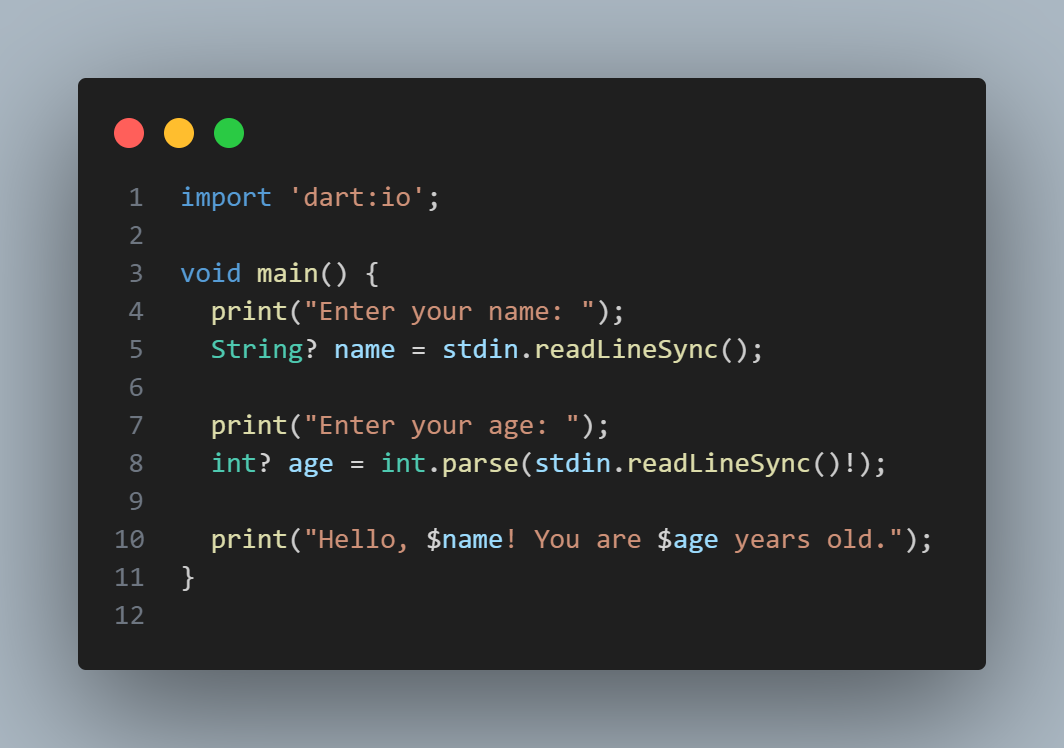


Figure user input

# 2.1.5 Dynamic Type (dynamic, var): var automatically detects the type, for example, var x = "Hello";. dynamic allows the type to change at runtime, for example, List<dynamic> number = [1, 2, 3,"Zain"].

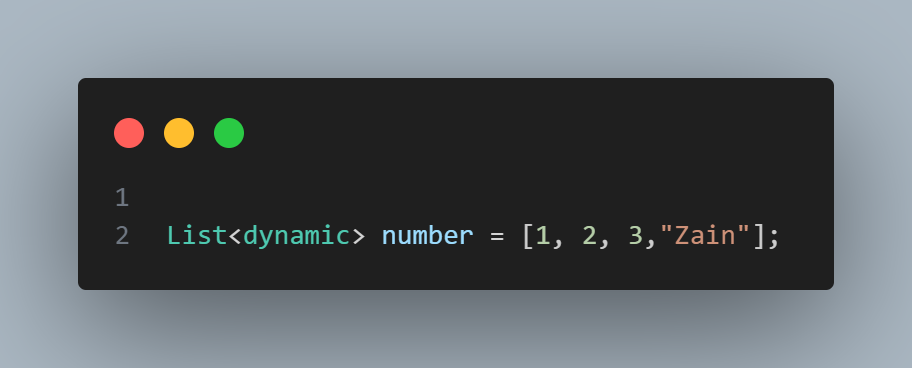


Figure 6 Dynamic Data Type

# Scope of Variables:

The **scope of a variable** refers to the region of the code where the variable can be accessed and used. In Dart, there are three main types of scope

## Global Scope:

Variables declared outside all functions can be accessed anywhere in the program.

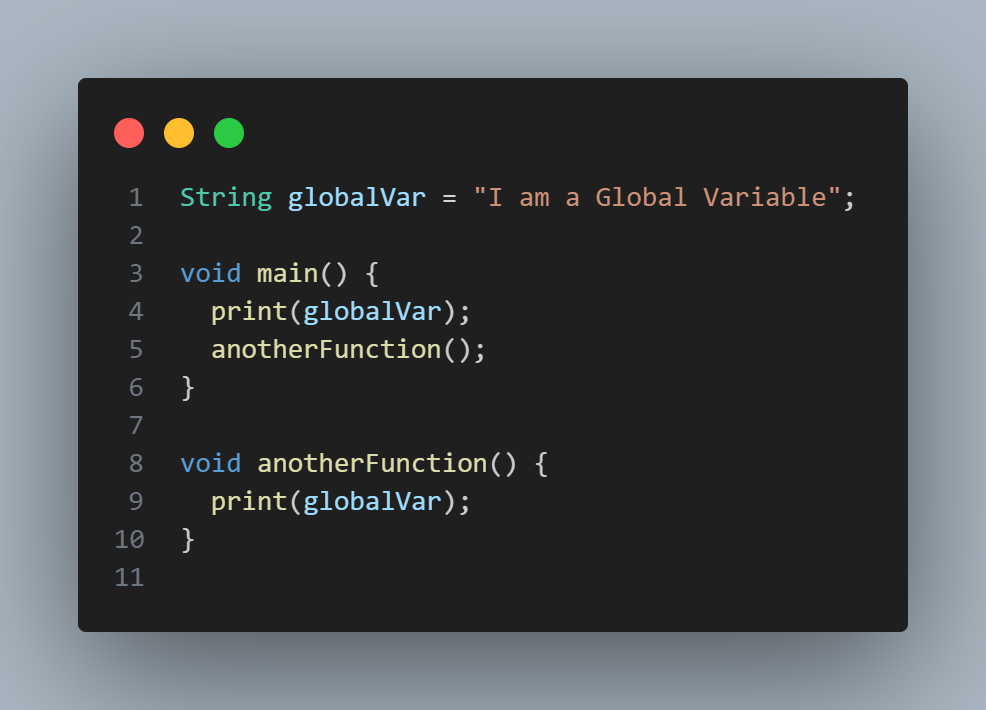


Figure Global Variable

## Local Scope (Function Scope):

variables declared inside a function can only be accessed within that function.

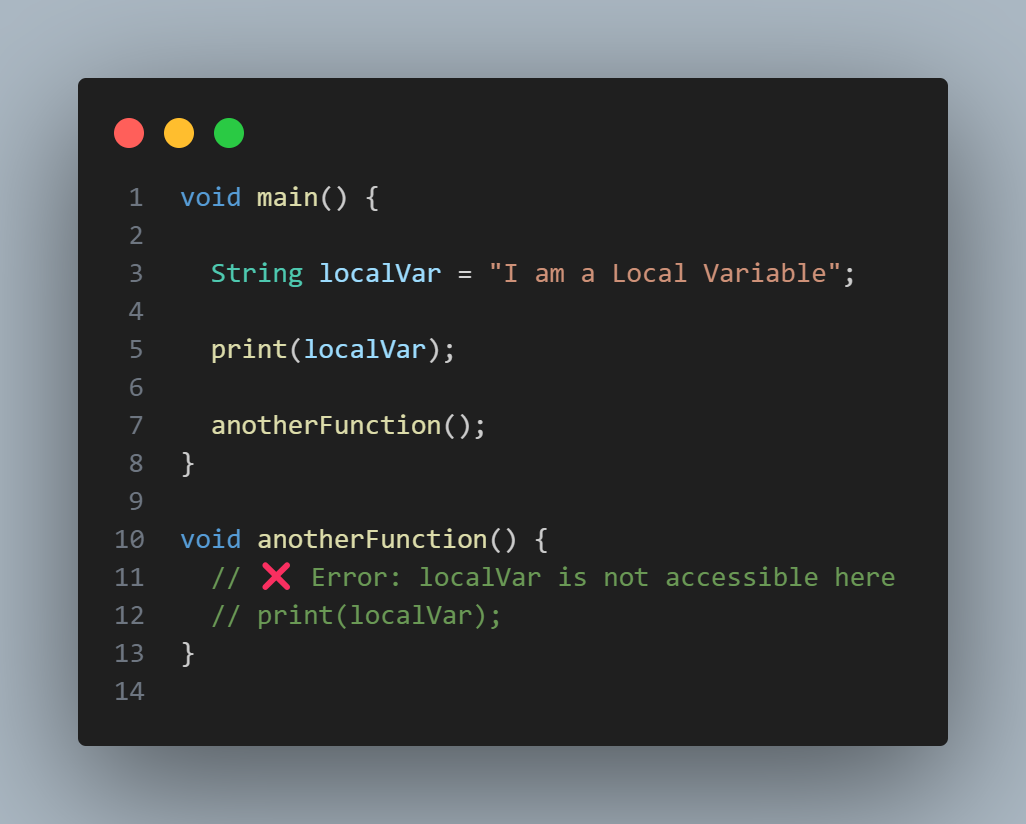


Figure Local Variable

## Block Scope:

A variable declared inside a {} block, accessible only within that block.



Figure Block Scope

# Comments:

## 4.1 Single-line Comments (//):

Used for short comments on a single line.

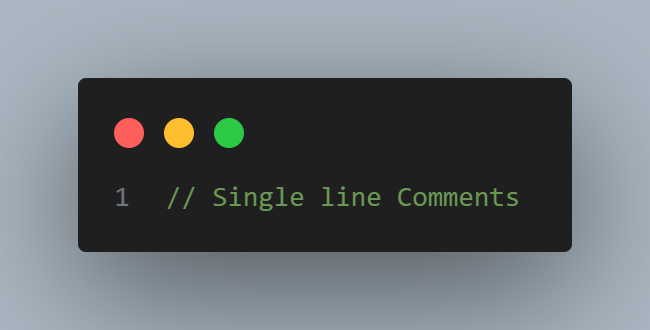


Figure 10 Single Line Comments

## 4.2 Multi-line Comments (/\* ... \*/)

Used for longer comments that span multiple lines.



Figure 11 Multiple Line Comments

# Collections:

## Queue:

A FIFO (First-In-First-Out) collection for task scheduling.

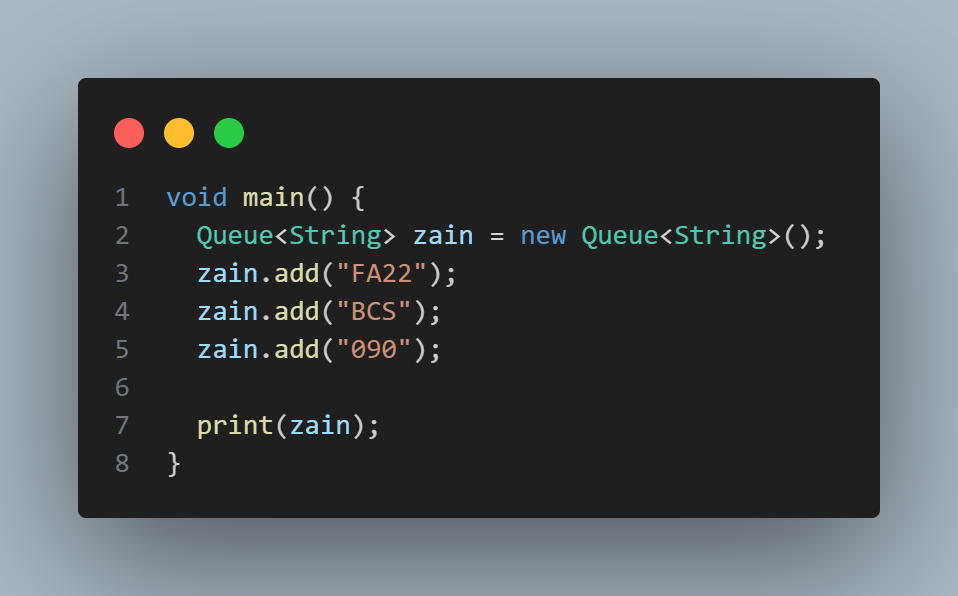
****

Figure 12 Queue Collection

## 5.2 Enum:

An **Enum** is a special data type that defines a fixed set of named constants for better readability.

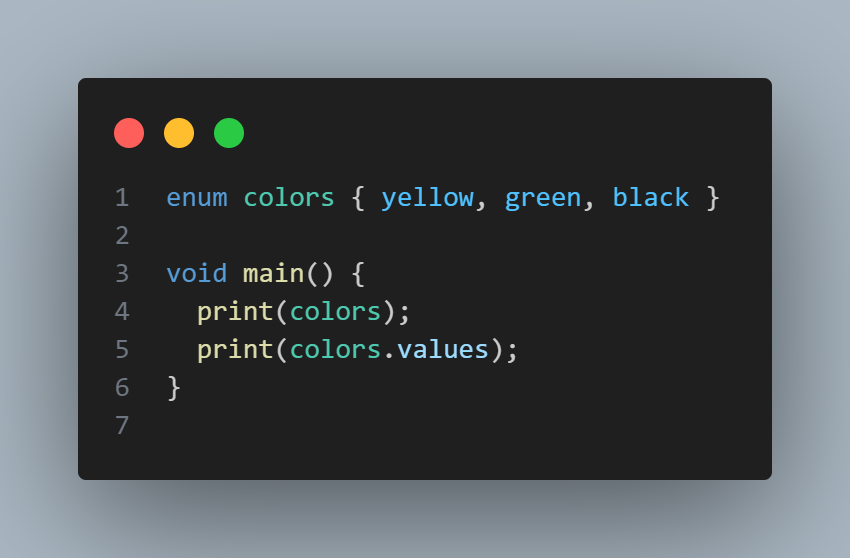


Figure 13 Enum Collection

5.3 Map Type (Map):  
Stores key-value pairs, for example, Map<String, int> scores = {"Alice": 90, "Bob": 85}



Figure 14 Map Data Type

5.4 List Type (List)  
Stores multiple values in an ordered collection, for example, List<String> Fruits = [“Apple”, “Banna”]

A screen shot of a computer code

AI-generated content may be incorrect.

Figure 15 List Data Type

# Control Flow:

## If:

Executes a block of code if a condition is true.

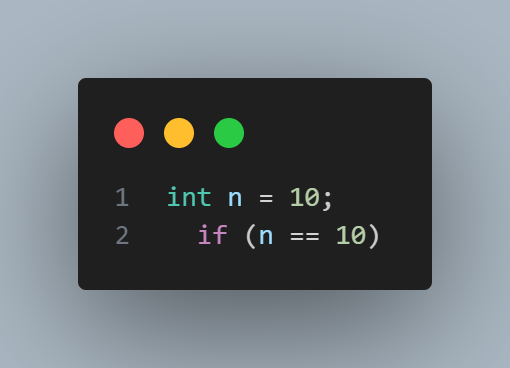


Figure 16 if Control Flow

## 6.2 Else:Executes a block of code if the if condition is false.A computer screen shot of a black rectangular object with colorful text AI-generated content may be incorrect. Figure 17 else Control Flow

## else if:

Checks multiple conditions when if is false.

A screen shot of a computer program

AI-generated content may be incorrect.

Figure 18 Elseif Control Flow

## Switch:

Selects a block of code to execute based on a value.A screenshot of a computer program

AI-generated content may be incorrect.

Figure Switch Control Flow

# Loops:

## While Loop

Executes code repeatedly while a condition is true.

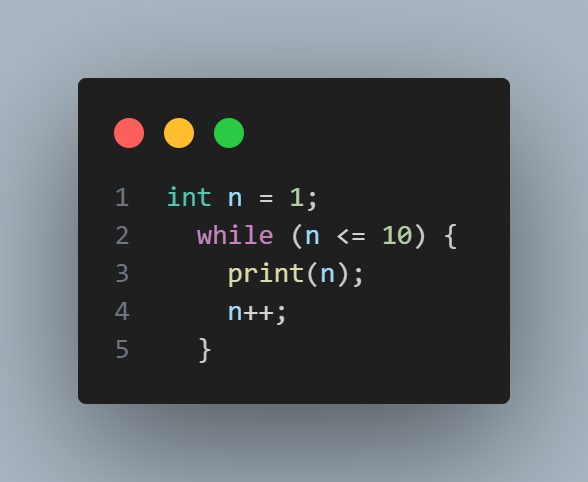


Figure 20 While Loop

## do-while loop:

Runs at least once, then repeats while a condition is true.

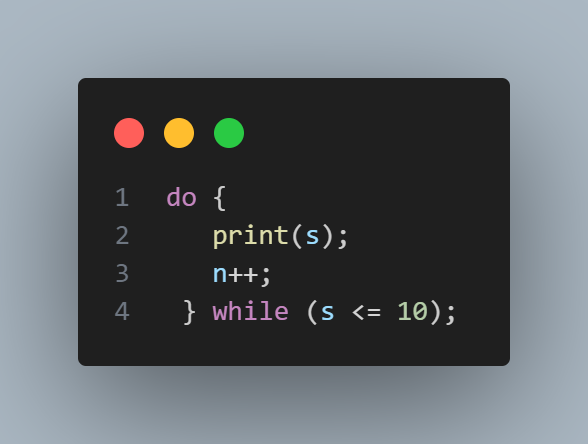


Figure 21 Do While Loop

## for loop:

Repeats a block of code a fixed number of times. Figure For Loop

## For Each Loop:

Iterates over each element in a collection.

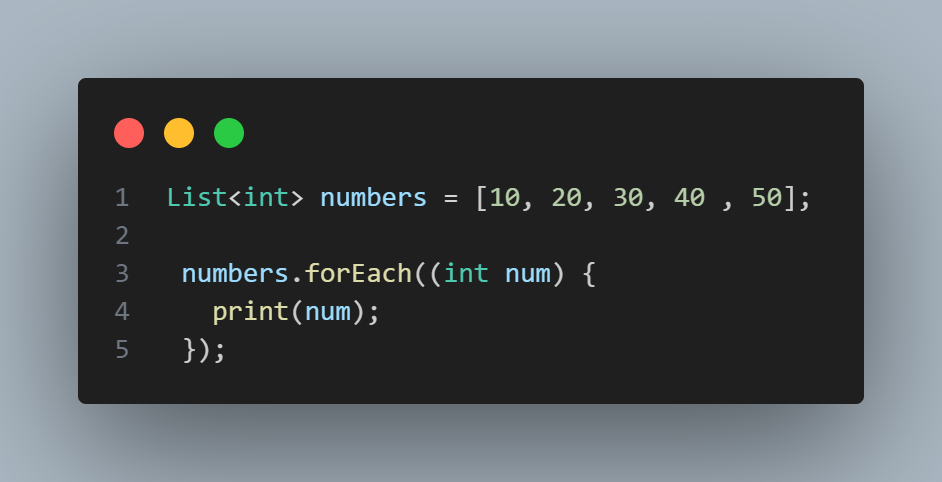


Figure For Each Loop

## for-in loop:

Iterates over elements in an inerrable like lists or sets.



Figure for in loop

# Functions:

## Function Definition:

A **function** is a reusable block of code that performs a specific task when called.

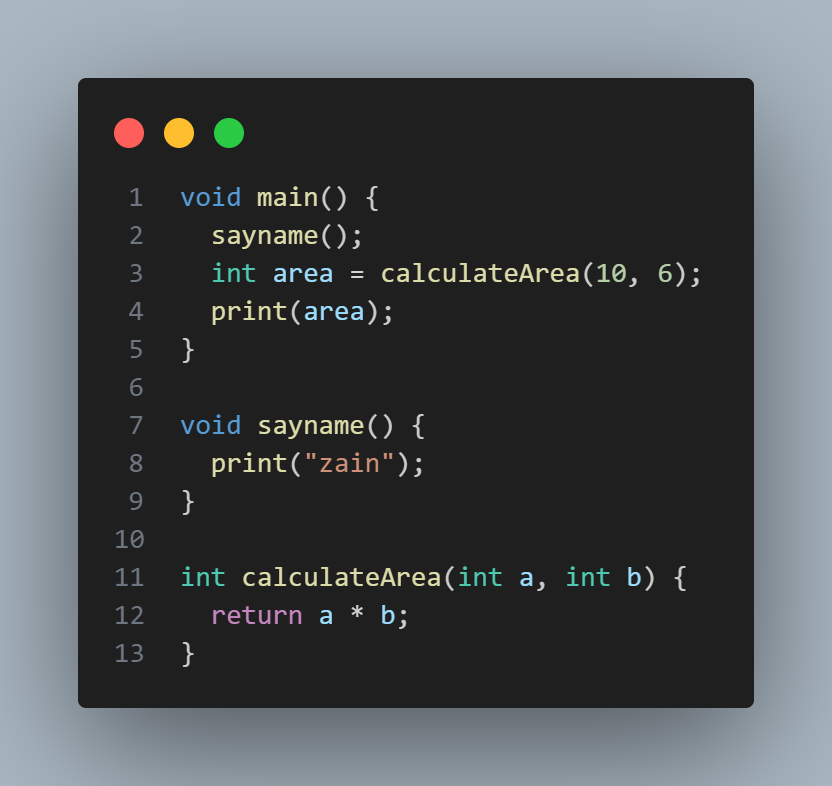


Figure Basic Function

## **Positional parameters**.

**Positional parameters** are function parameters that must be passed in the correct order.



Figure Positional parameters

## Optional Positional Parameters:

Parameters in square brackets [] are optional. A default value (false for open) is provided if no argument is passed.

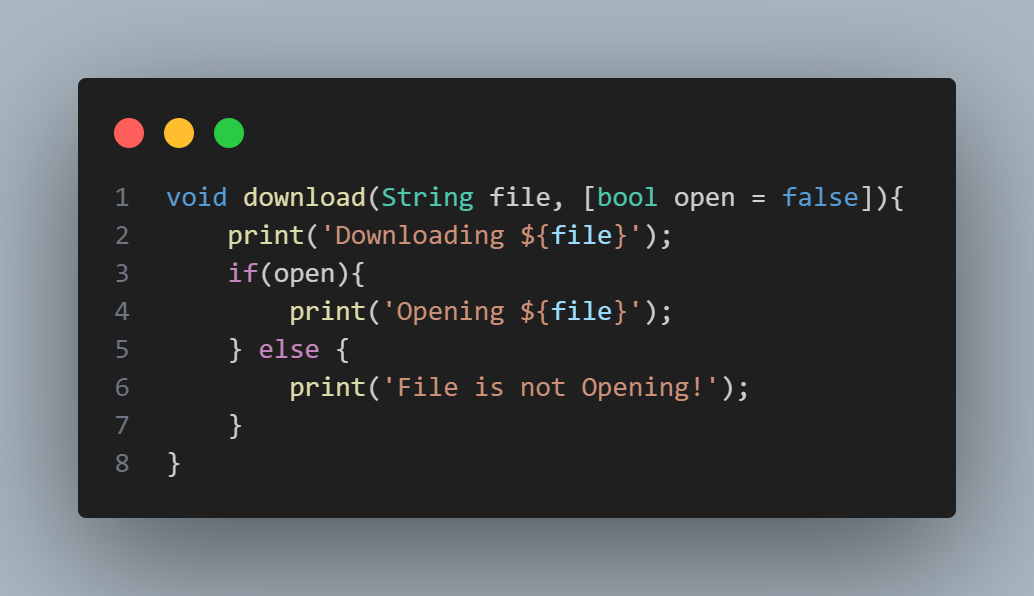
****

Figure Optional Positional Parameters

## List Parameter (Dynamic Type):

Accepts a list of mixed data types (List<dynamic>). Useful when handling different types of values.

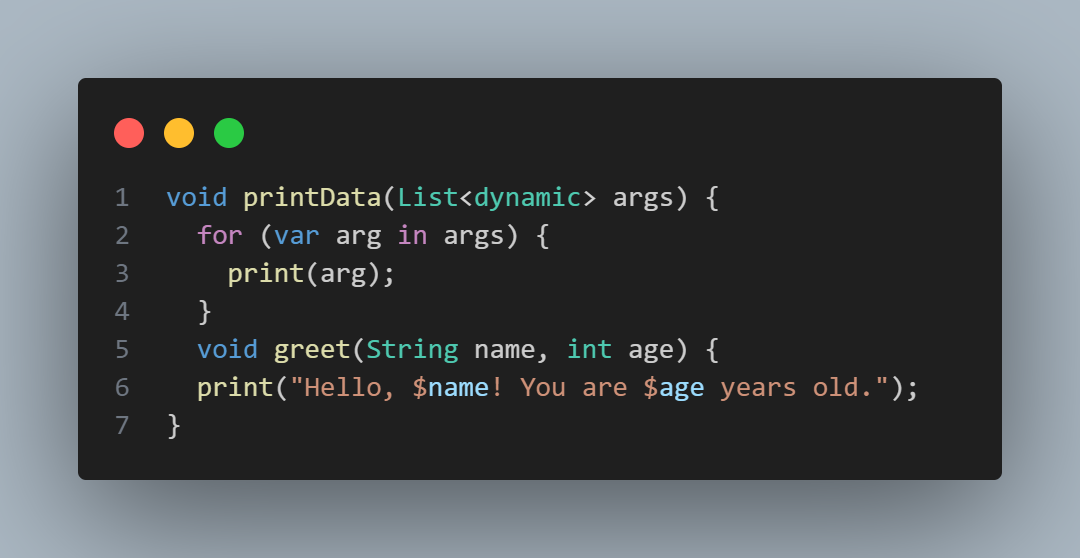


Figure List Parameters Dynamic

## List Parameter (Fixed Type)

Accepts a list of integers (List<int>). Iterates over the list and prints each number.

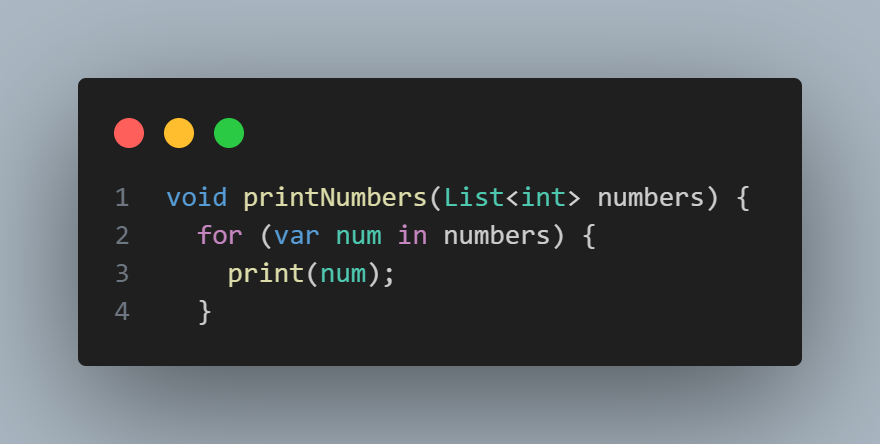


Figure List Perimeters Fixed

## Required Positional Parameter:

The parameter **must** be passed when calling the function. Does not allow null values.



Figure Required Positional Parameter

## Nullable Parameter with Default Value:

The parameter is **optional** and can be null.

Uses the **Null-Coalescing Operator (??)** to assign 'Guest' if null.



Figure Nullable Parameter with Default Valu

# Arithmetic Operations in Dart:

## Addition (+):

Adds two numbers.



Figure Addition

## Subtraction (-):

Subtracts one number from another.

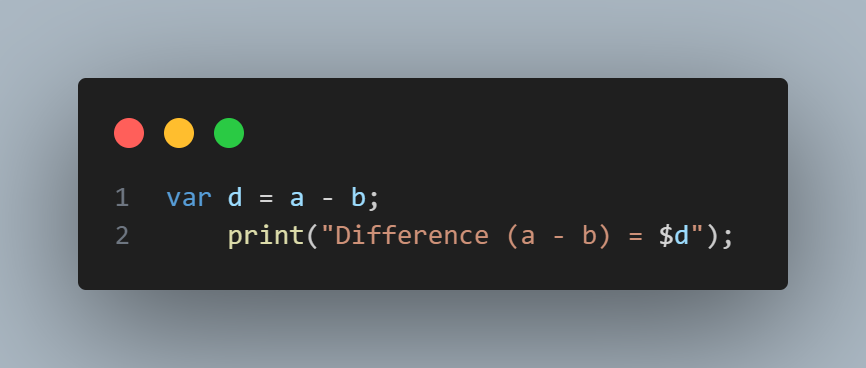


Figure Subtraction

## Multiplication (\*):

Multiplies two numbers.

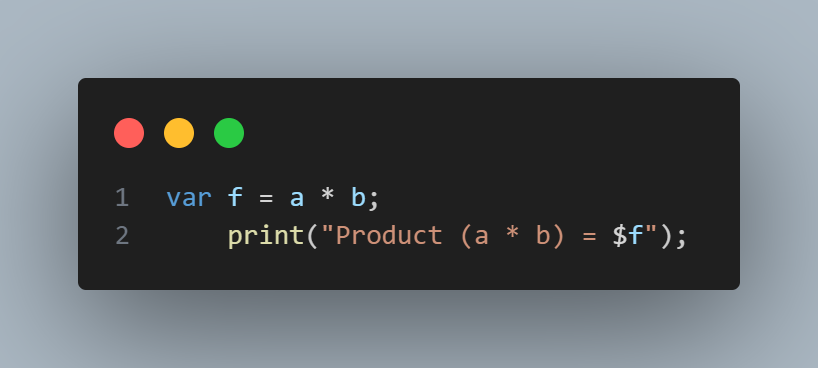


Figure product

## Division (/):

Divides two numbers and returns a double.

A black rectangular with green and blue text

AI-generated content may be incorrect.

Figure Division

## Integer Division (~/):

Divides two numbers and returns an integer.

A black rectangular with colorful text

AI-generated content may be incorrect.

Figure Integer Division

## Modulus (%):

**returns the remainder of division.**

**A black rectangular with text and green circles

AI-generated content may be incorrect.**

Figure Modulus

# Exception Handling:

Exception handling in Dart is done using try, catch, and finally blocks to handle runtime errors and prevent program crashes.

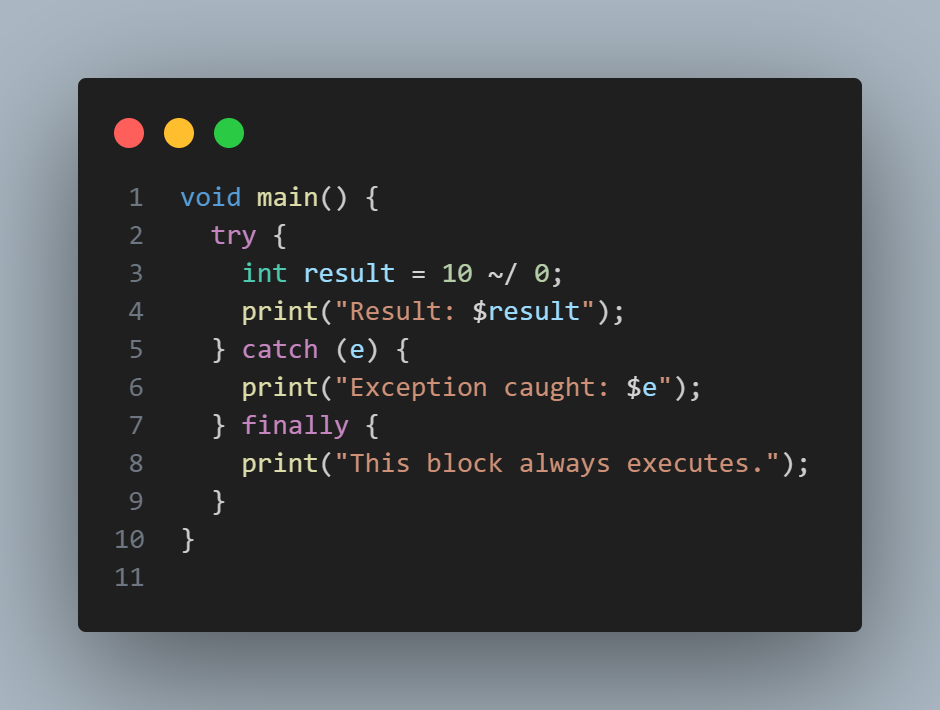


Figure Exception Handling

## try block:

The try block contains the code that may throw an exception.

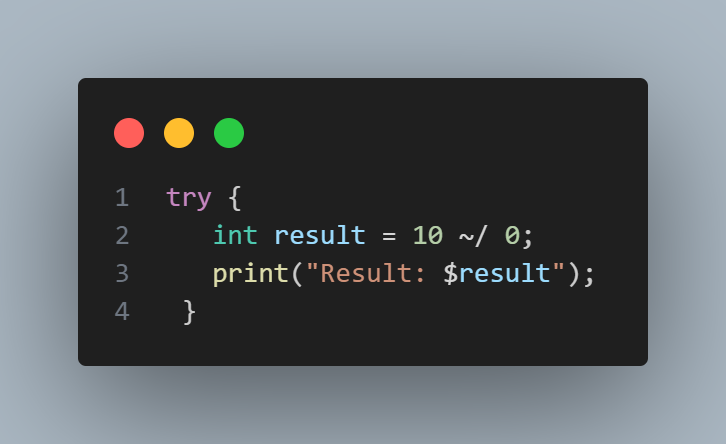


Figure 39 Try Block

## catch Block:

The catch block is used to handle the exception that occurs inside the try block.

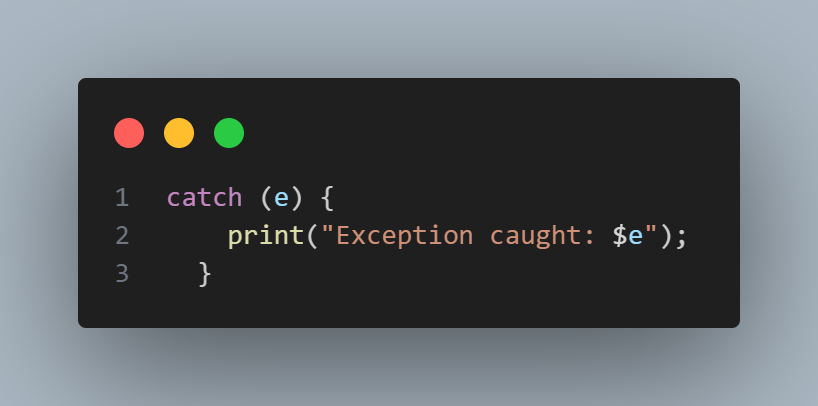
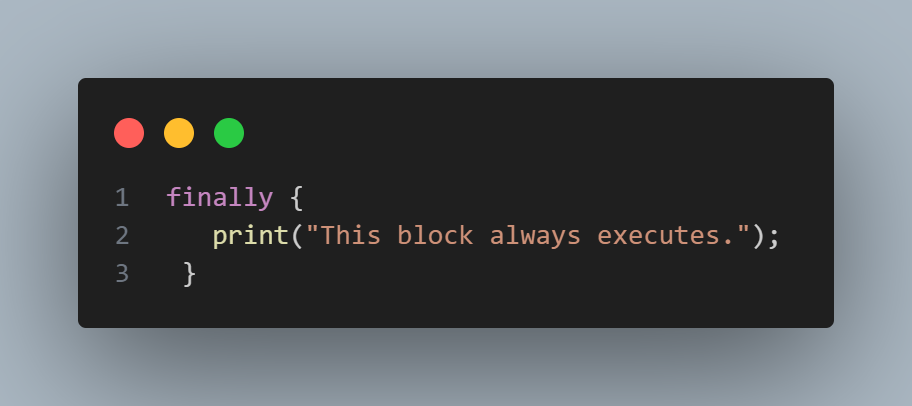


Figure 40 catch Block

## finally Block

The finally block is always executed, regardless of whether an exception occurs or not. It is useful for cleanup operations.



# Practice Problems:

Solving Problem For practice

## Find Factorial:

Find Factorial using Function and loop.

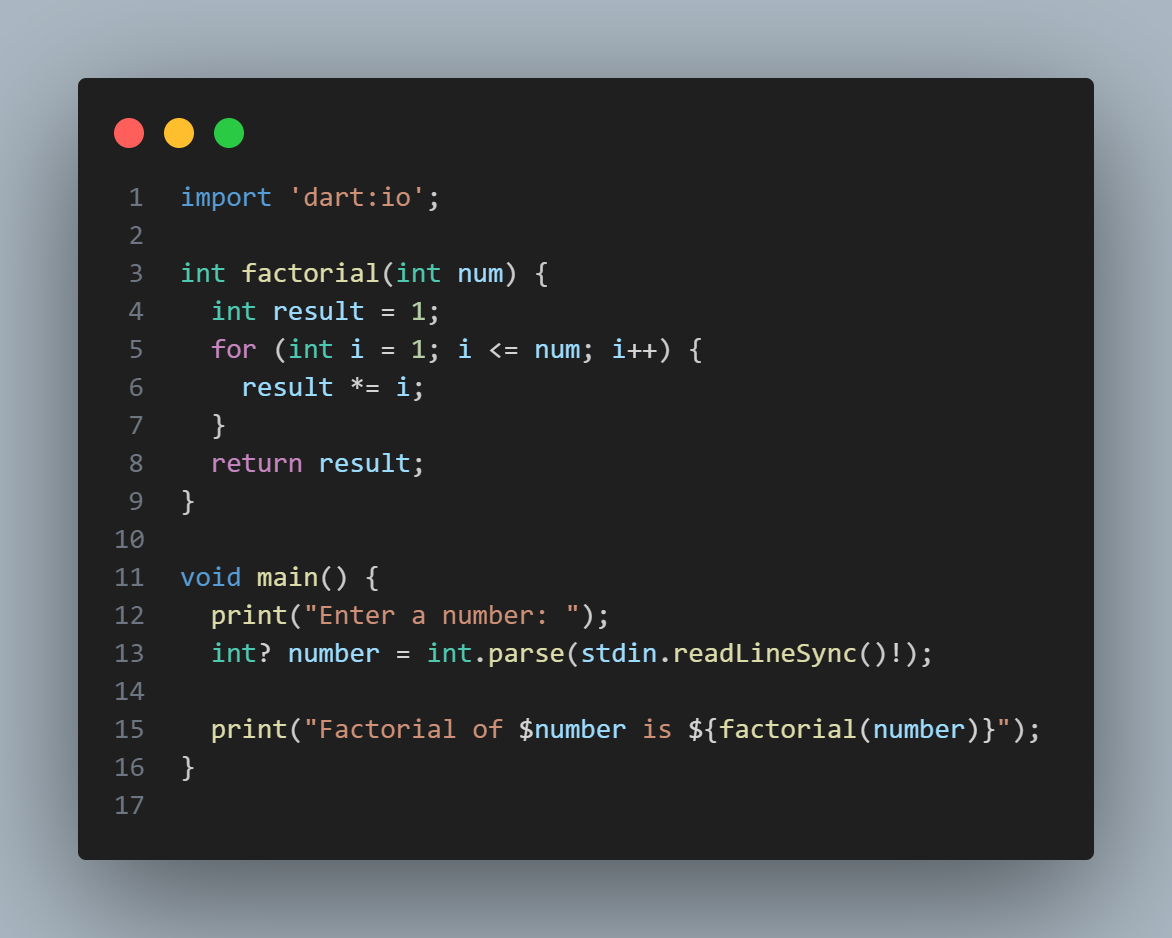


Figure Find Factorial

## Check if a Number is Even or Odd:

Check even or odd number using function and condition.

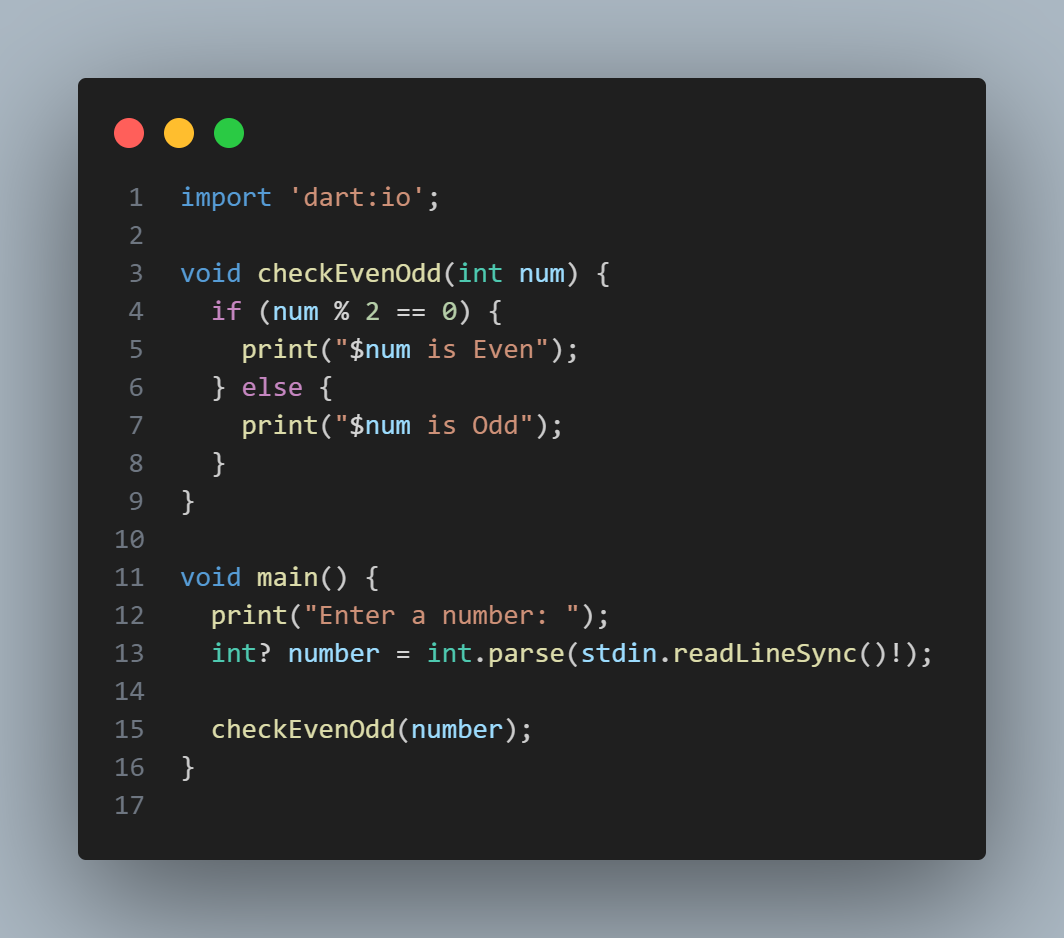


Figure Check number is even or odd

## Sum of First N Natural Number:

Using loop and functions

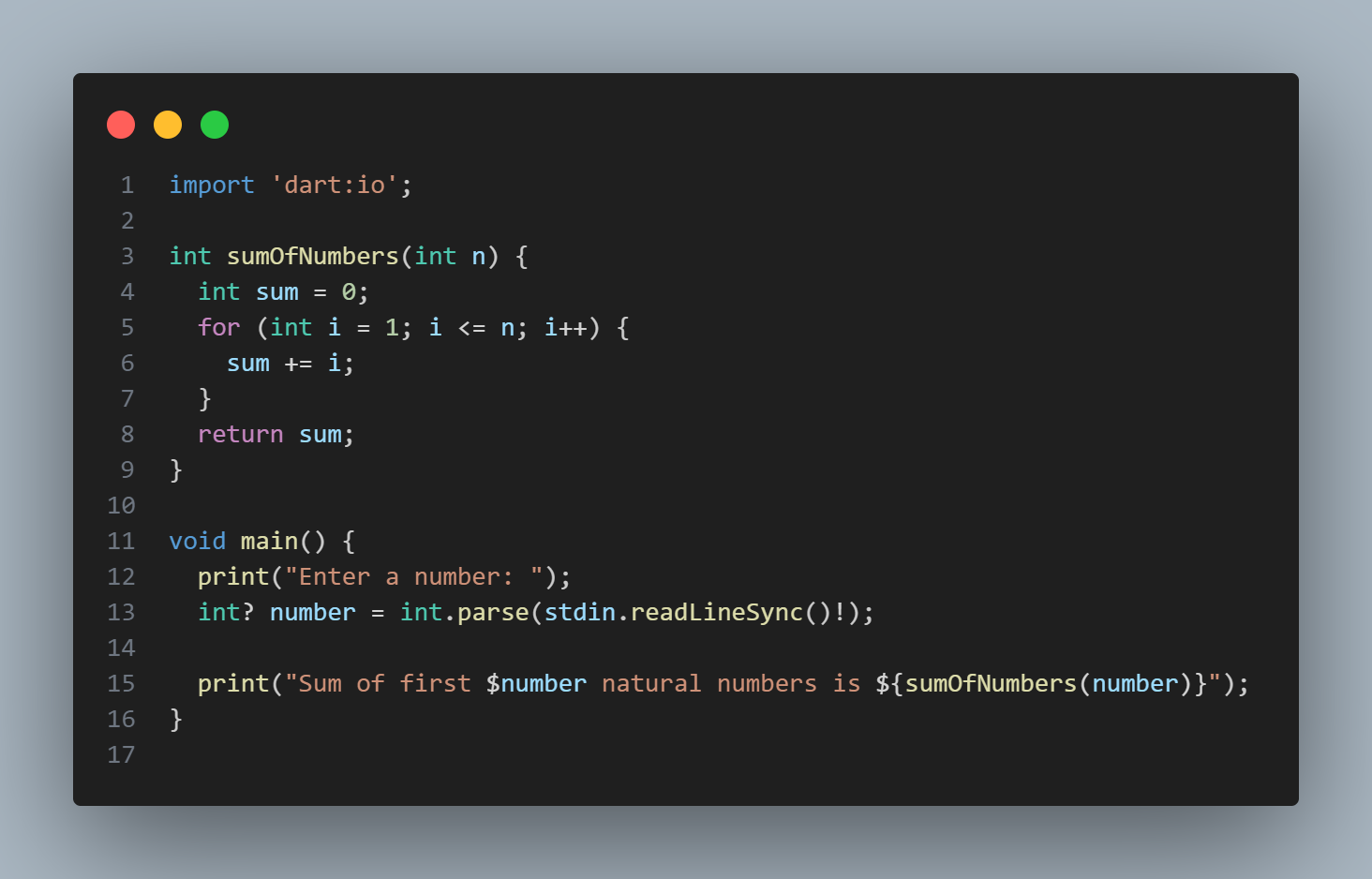


Figure sum of natural numbers

## Try-Catch for Invalid Input:

Using Exception Handling.

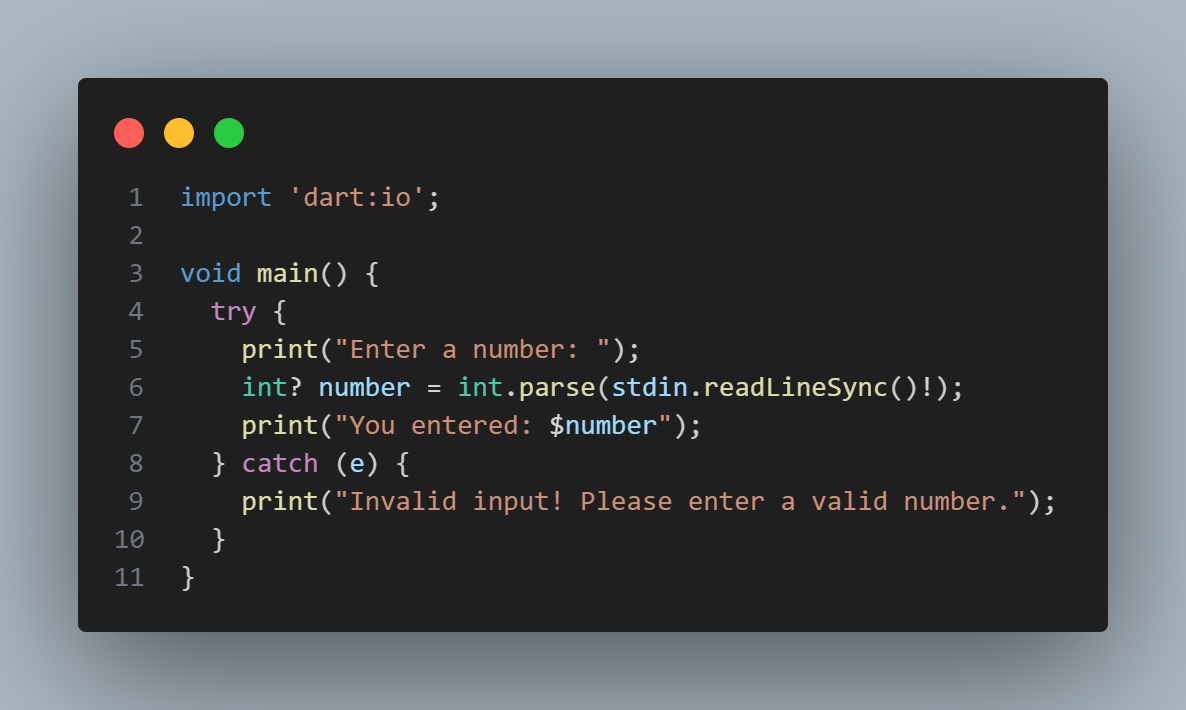


Figure Check invalid or valid input

## Print Numbers from 1 to 10:

Using for loop

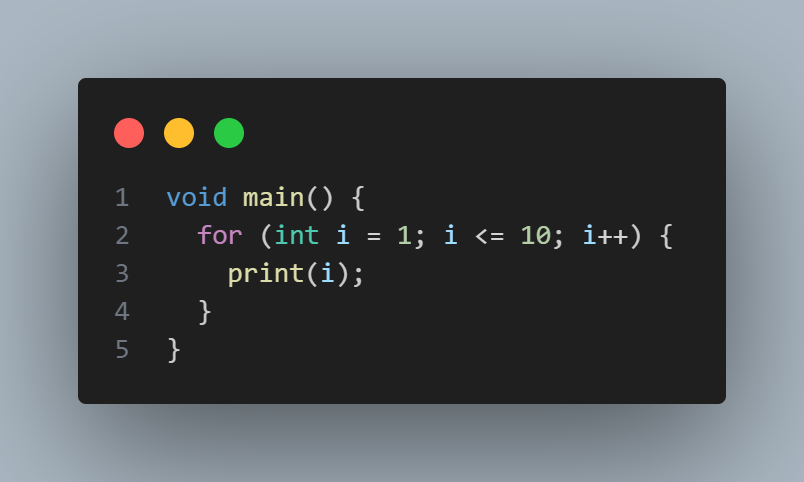


Figure print number 1 to 10

## Check if a Number is Positive, Negative, or Zero:

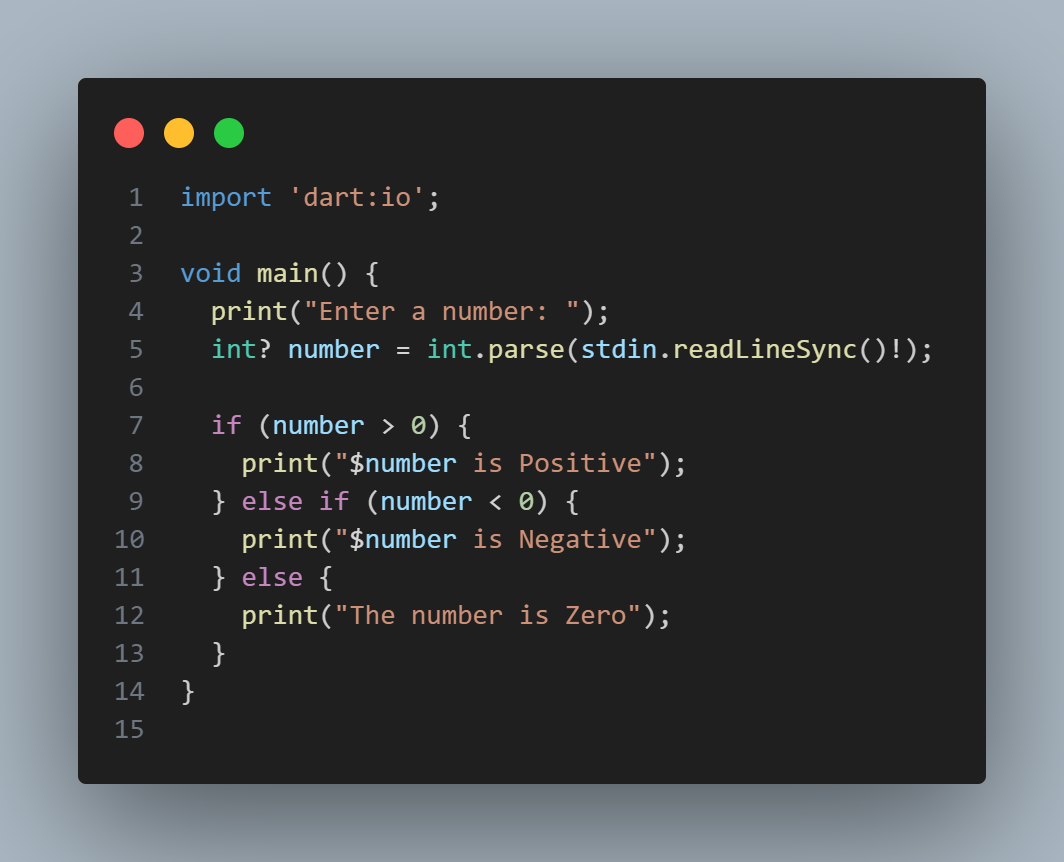


Figure number check

## Find the Sum of Two Numbers:

Used function to add two numbers

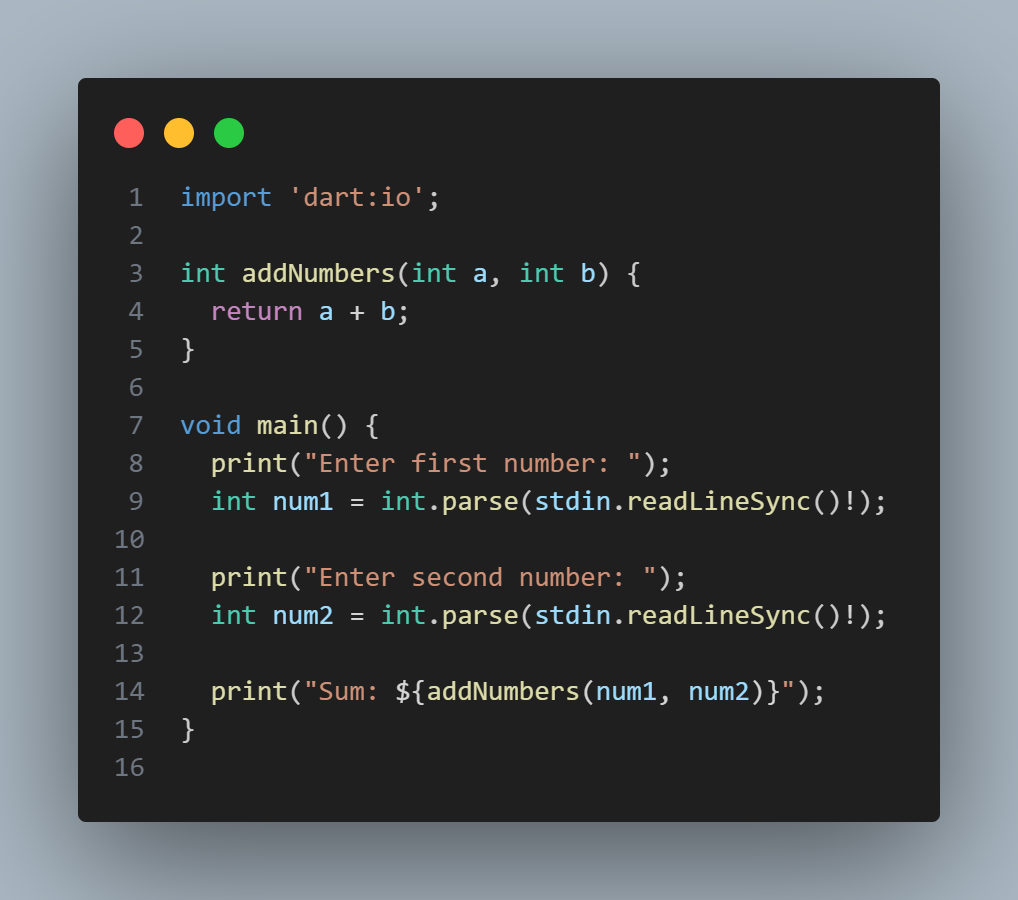


Figure sum of two numbers

## Function to Find the Maximum of Two Numbers:

Used function find max number



Figure find max number function

## Function to Check if a Number is Prime:

Used functions for it

A screen shot of a computer program

AI-generated content may be incorrect.