

# Exercício

## 01 Utilizando Future e async/await

Escreva uma função assíncrona em Dart chamada `fetchData` que simula uma requisição assíncrona a um servidor. A função deve retornar um `Future` que resolve para uma string "Dados obtidos" após um atraso de 3 segundos. Em seguida, escreva um programa principal que chama a função `fetchData` utilizando a palavra-chave `await` e exibe a mensagem de dados obtidos.



# Exercício

## 02 Utilizando Stream e async\*

Escreva uma função assíncrona em Dart chamada `countDownStream` que recebe um número máximo como parâmetro. A função deve retornar um `Stream` que emite contagem regressiva de 5 até 1, com um atraso de 1 segundo entre cada emissão. Em seguida, escreva um programa principal que assina o `Stream` retornado pela função `countDownStream` e exibe os números emitidos.



# Exercício

## 03 Utilizando Future.wait e then

Escreva um programa em Dart que usa a função Future.wait para aguardar a conclusão de três tarefas assíncronas diferentes. Cada tarefa deve ser uma função assíncrona que retorna um Future que resolve algum dado após um tempo aleatório. Após aguardar a conclusão das três tarefas, o programa deve exibir os dados obtidos.

OBS: Use a API de referência:

<https://api.dart.dev/stable/3.0.7/dart-async/Future-class.html>



# Exercício

## 04 Utilizando `Stream.value`, `map` e `toList`

Escreva um programa em Dart que usa um `Stream` para processar uma sequência de nomes. O programa deve criar um `Stream` usando `Stream.value` com uma lista de nomes fornecida pelo usuário. Em seguida, deve usar o método `map` para transformar cada nome em seu comprimento (número de caracteres). Por fim, o programa deve usar o método `toList` para coletar os comprimentos dos nomes em uma lista e exibi-la.

