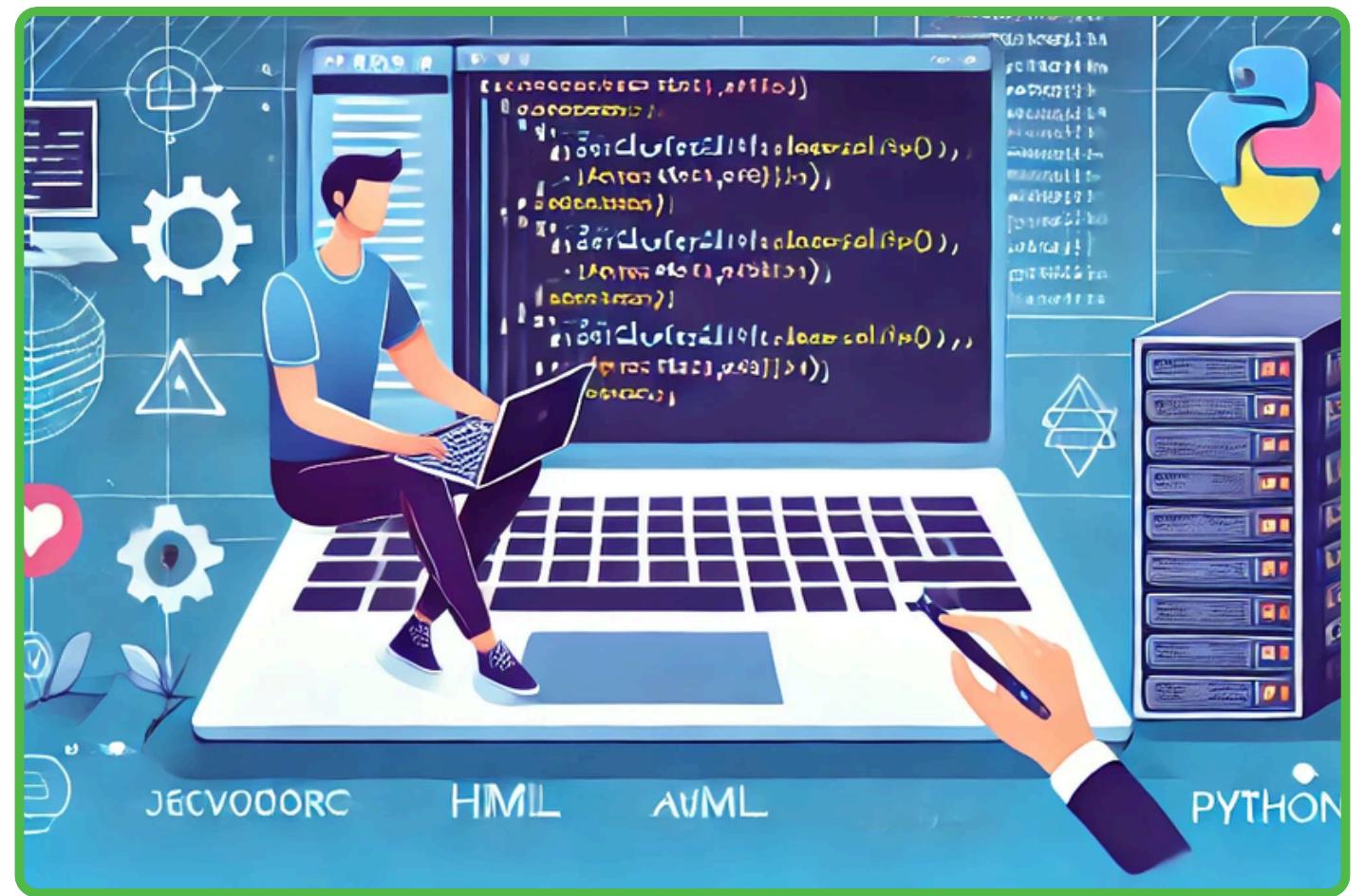


# CURSO TÉCNICO EM INFORMÁTICA



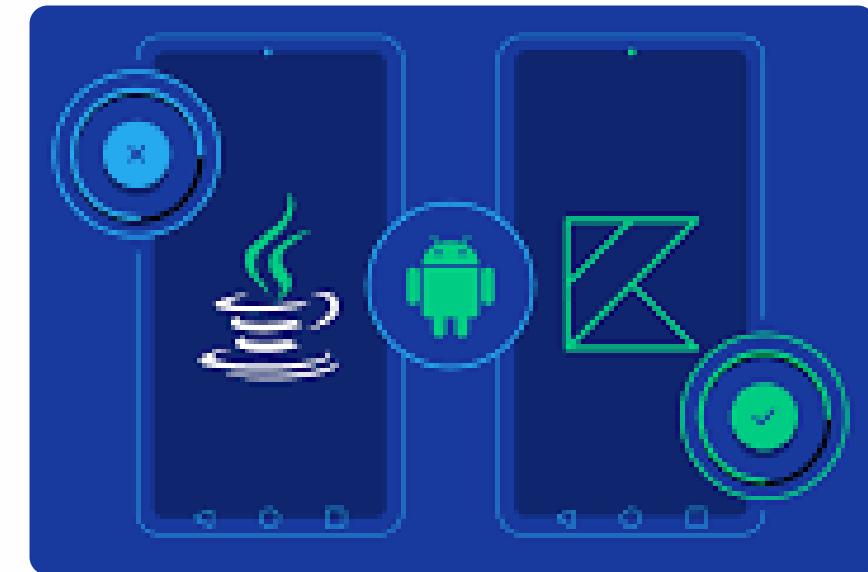
 **Kotlin**



## Introdução ao Kotlin

Kotlin é uma linguagem de programação moderna e em alta no mercado, lançada em 2016 pela empresa JetBrains.

Desde seu lançamento, o **Kotlin** conquistou grande popularidade, principalmente por ser compatível com Java — uma das linguagens de programação mais utilizadas no mundo. Isso significa que códigos e bibliotecas escritas em Java podem ser usados diretamente em programas **Kotlin**, facilitando a adoção da linguagem por desenvolvedores que já trabalham com Java.

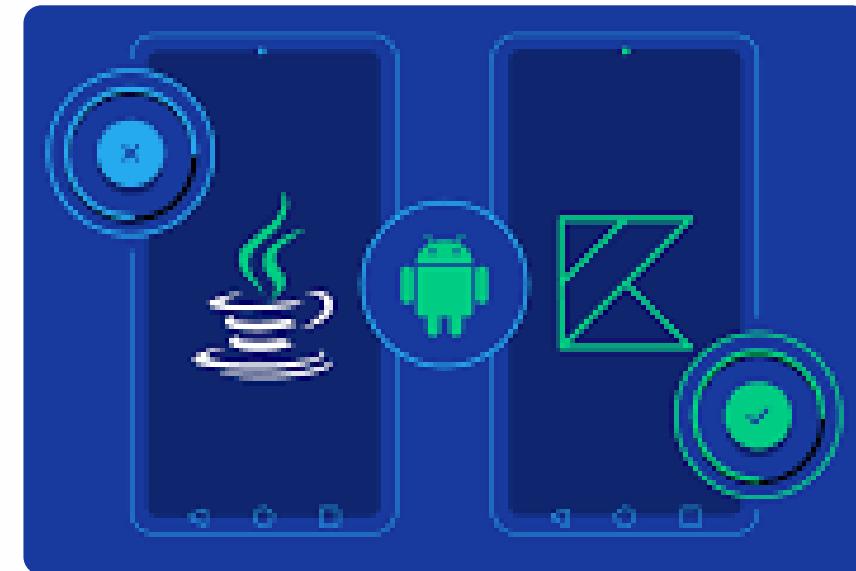


## Introdução ao Kotlin

Onde o Kotlin é utilizado?

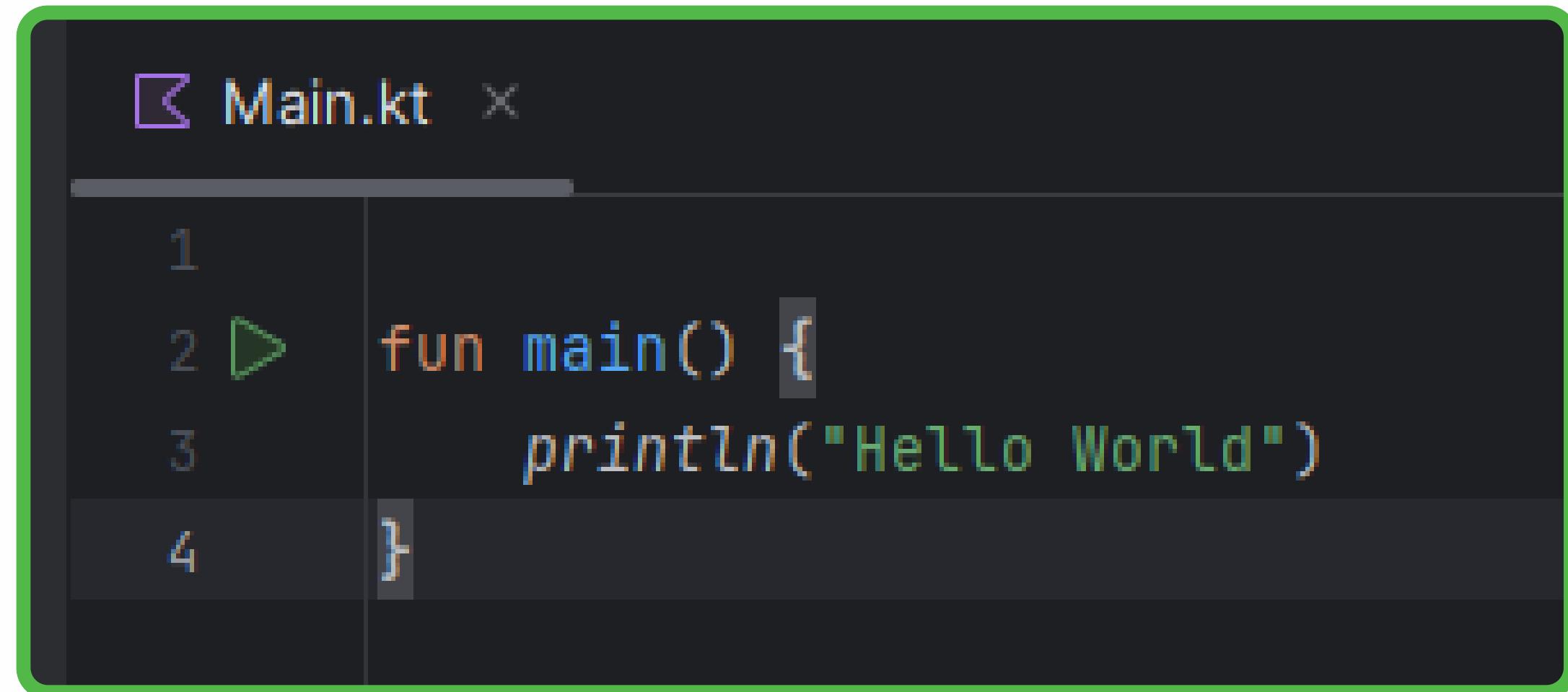
O Kotlin é extremamente versátil e pode ser aplicado em diversas áreas, como:

- Aplicações móveis (especialmente no desenvolvimento de aplicativos Android)
- Desenvolvimento web
- Aplicações no servidor (back-end)
- Ciência de dados



## Sintaxe do Kotlin

Criamos um arquivo **Kotlin** chamado **Main.kt** e utilizamos o seguinte código para imprimir "Hello World" na tela:

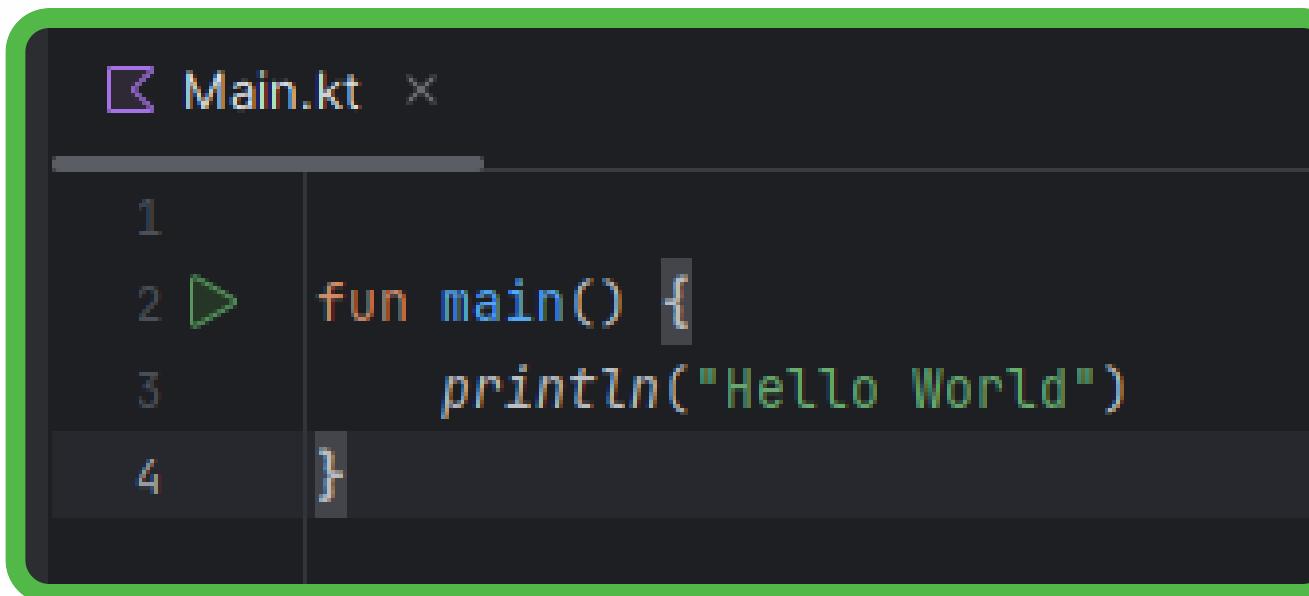


```
>Main.kt
```

```
1
2 fun main() {
3     println("Hello World")
4 }
```



## Sintaxe do Kotlin



```
Main.kt
1
2 fun main() {
3     println("Hello World")
4 }
```

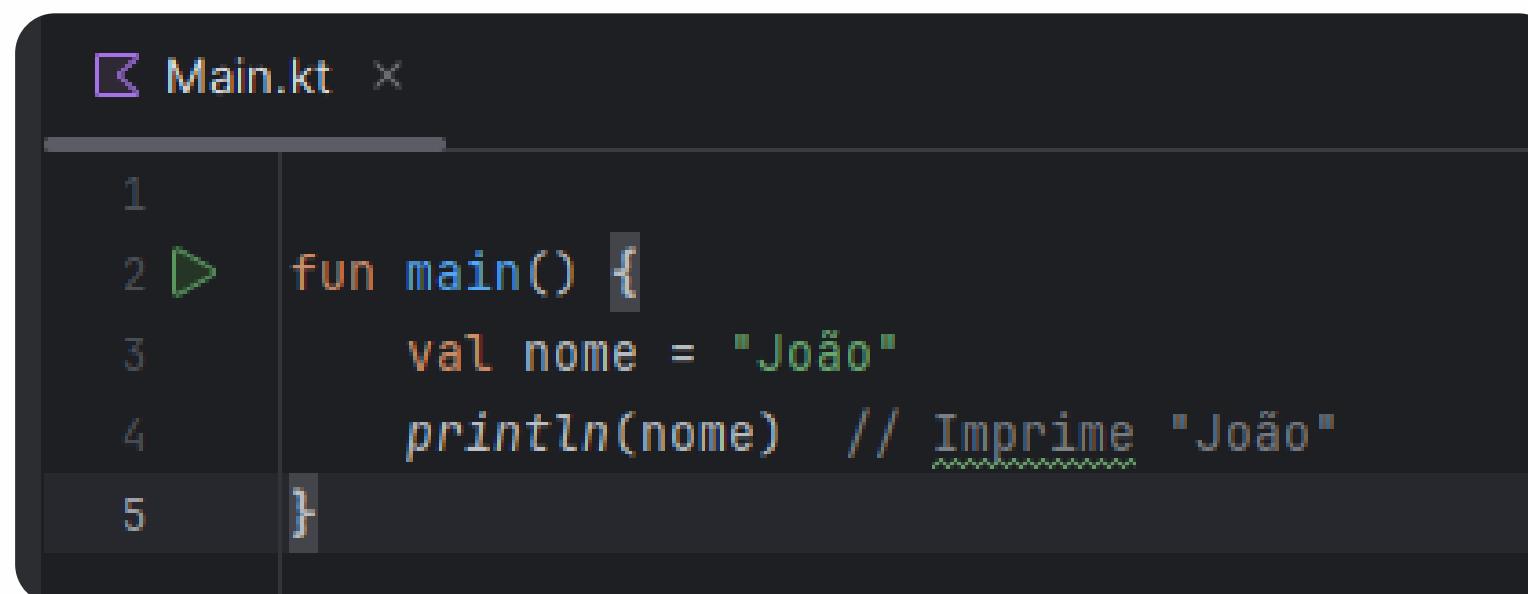
- A palavra-chave **fun** é usada para declarar uma função.
- Uma função é um bloco de código projetado para realizar uma tarefa específica. No exemplo acima, estamos declarando a função **main()**.
- A função **main()** é algo que você verá em todo programa Kotlin.
- Ela é a porta de entrada da aplicação, ou seja, é onde o programa começa a ser executado.
- Todo o código dentro das chaves **{ }** da função **main()** será executado quando o programa for iniciado.
- Dentro da função **main()**, usamos a função **println()**.
- Essa função serve para exibir (imprimir) um texto no console. No nosso exemplo, ela imprime "Hello World".



## Declarando Variáveis

Existem dois tipos principais de variáveis em **Kotlin**:

- **Variáveis imutáveis (val)**: Uma variável declarada com a palavra-chave val não pode ter seu valor alterado após ser inicializada. Ou seja, é uma variável constante.



```
Main.kt
1
2 fun main() {
3     val nome = "João"
4     println(nome) // Imprime "João"
5 }
```

Nesse exemplo, a variável nome é inicializada com o valor "João", e não podemos alterar esse valor depois que ele foi atribuído.

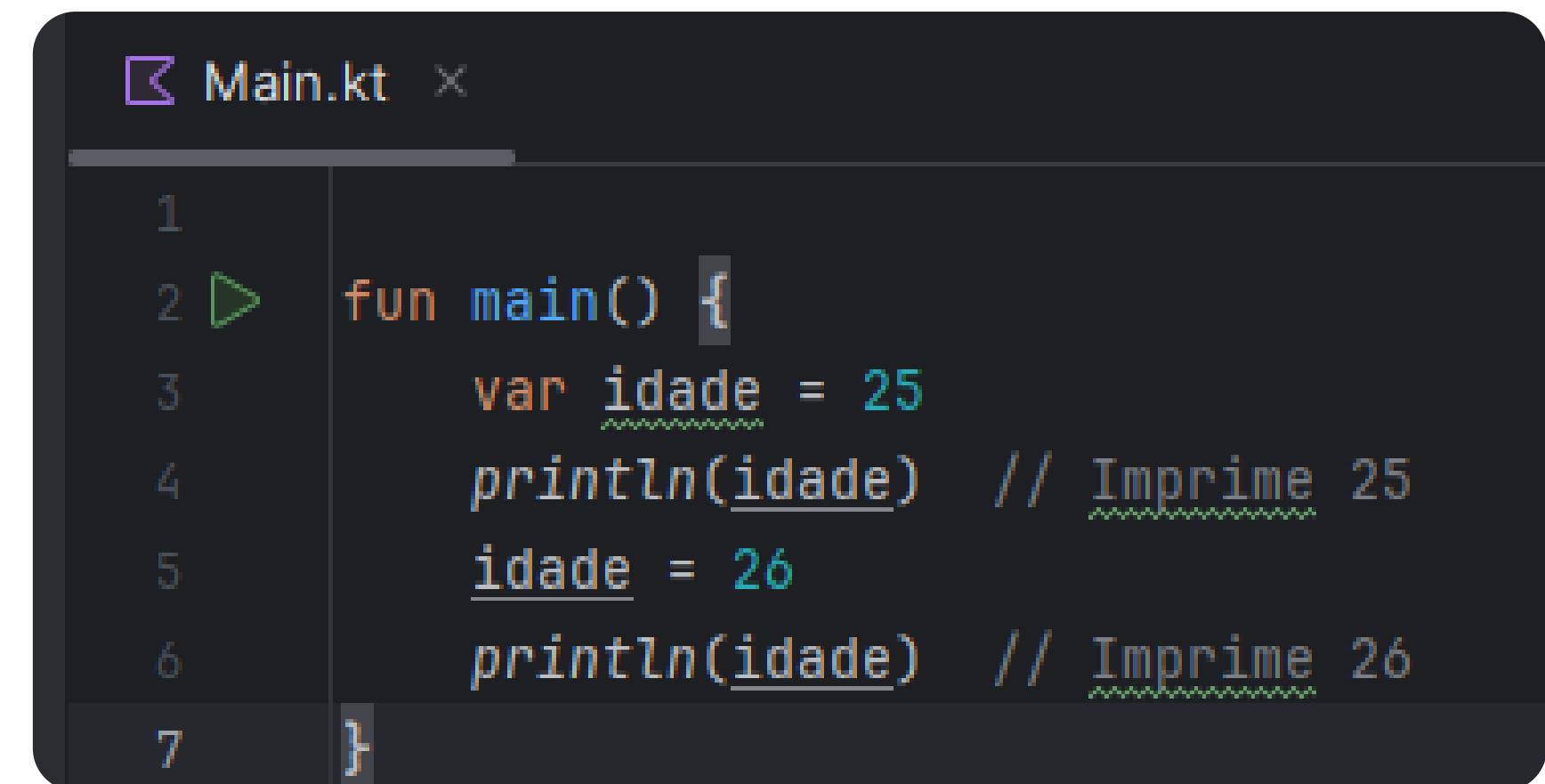


## Declarando Variáveis

Existem dois tipos principais de variáveis em **Kotlin**:

- **Variáveis mutáveis (var)**: Já as variáveis declaradas com a palavra-chave var podem ter seus valores alterados durante a execução do programa. Ou seja, são variáveis que podem ser modificadas. .

Aqui, a variável idade começa com o valor 25 e, em seguida, seu valor é alterado para 26.



```
Main.kt
1
2 fun main() {
3     var idade = 25
4     println(idade) // Imprime 25
5     idade = 26
6     println(idade) // Imprime 26
7 }
```



## Tipos de Dados em Kotlin

Existem dois tipos principais de variáveis em **Kotlin**:

- **Inteiros (Int)**

**Int** é o tipo mais comum para armazenar números inteiros (sem casas decimais), e seu valor varia de -2.147.483.648 a 2.147.483.647.

- **Números de Ponto Flutuante (Double, Float):**

**Double** é utilizado para armazenar números com casas decimais de precisão dupla, o que significa que ele oferece maior precisão em números de ponto flutuante.

**Float** é similar ao Double, mas com precisão simples. Para declarar um número **Float**, é necessário adicionar o sufixo **F** no valor.



## Tipos de Dados em Kotlin

Existem dois tipos principais de variáveis em **Kotlin**:

- **Texto (String)**

**String** é o tipo utilizado para armazenar sequências de caracteres (textos).

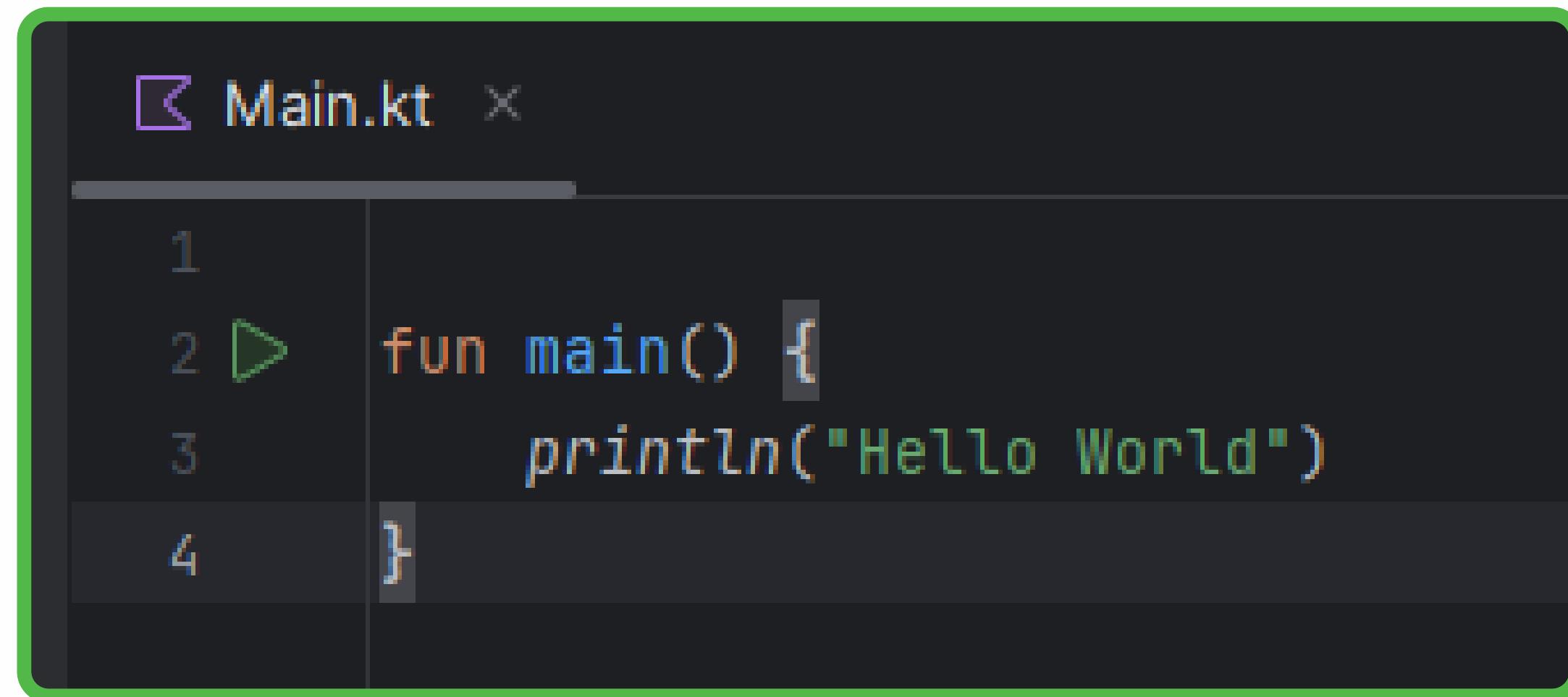
- **Booleanos (Boolean)**

Boolean armazena valores lógicos: **true** ou **false**.



## Saída de Dados

Para imprimir dados no console, usamos a função **println( )**. Esta função exibe uma linha de texto seguida por uma **quebra de linha**. Caso você queira imprimir algo sem pular uma linha, pode usar **print()**, que apenas imprime o texto sem adicionar a quebra de linha no final.

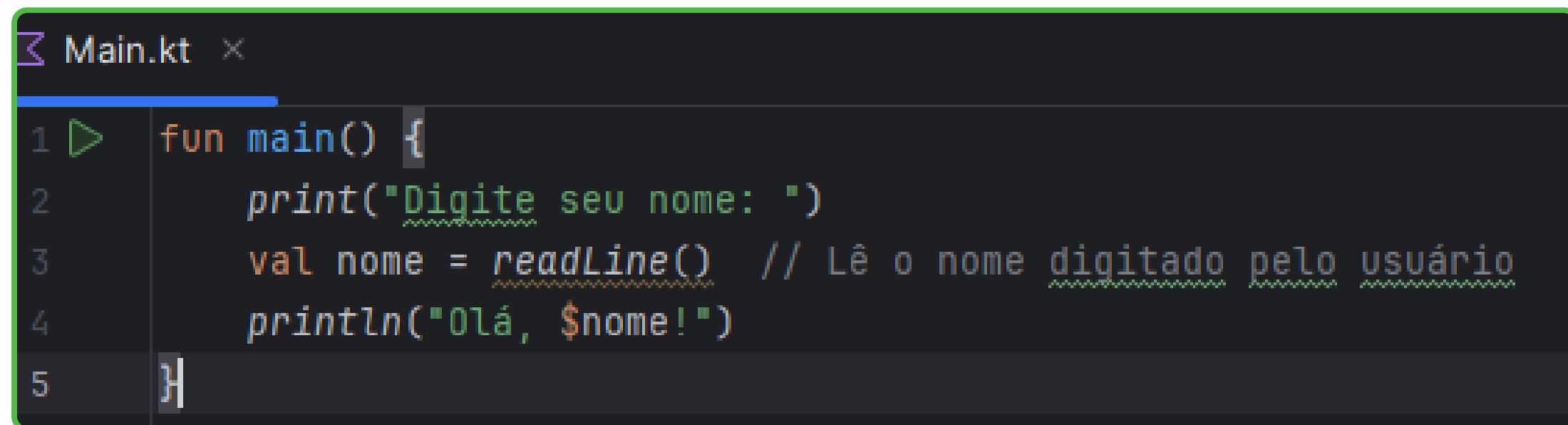


```
Main.kt
1
2 fun main() {
3     println("Hello World")
4 }
```



## Entrada de Dados

Para ler dados inseridos pelo usuário, utilizamos a função **readLine()**, que lê uma linha de texto e retorna esse valor como uma **string**.

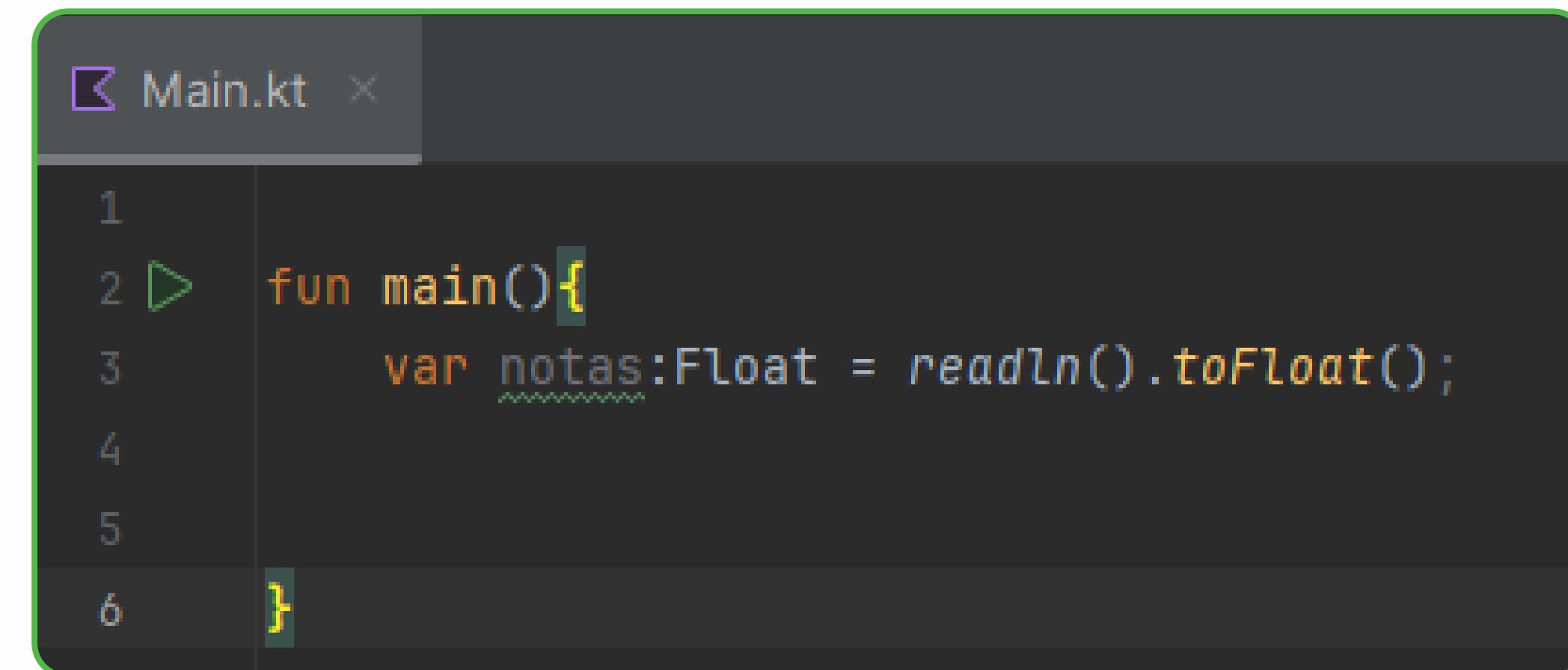


```
Main.kt
1 fun main() {
2     print("Digite seu nome: ")
3     val nome = readLine() // Lê o nome digitado pelo usuário
4     println("Olá, $nome!")
```



## Conversão de Entrada com `readIn()` em **Kotlin**

No **Kotlin**, a função `readIn()` é utilizada para ler dados digitados pelo usuário diretamente do console. Essa função sempre retorna os dados como uma **String**. No entanto, em muitos casos, especialmente quando trabalhamos com valores numéricos, precisamos converter essa String para outro tipo de dado, como **Int**, **Float**, **Double**, entre outros.



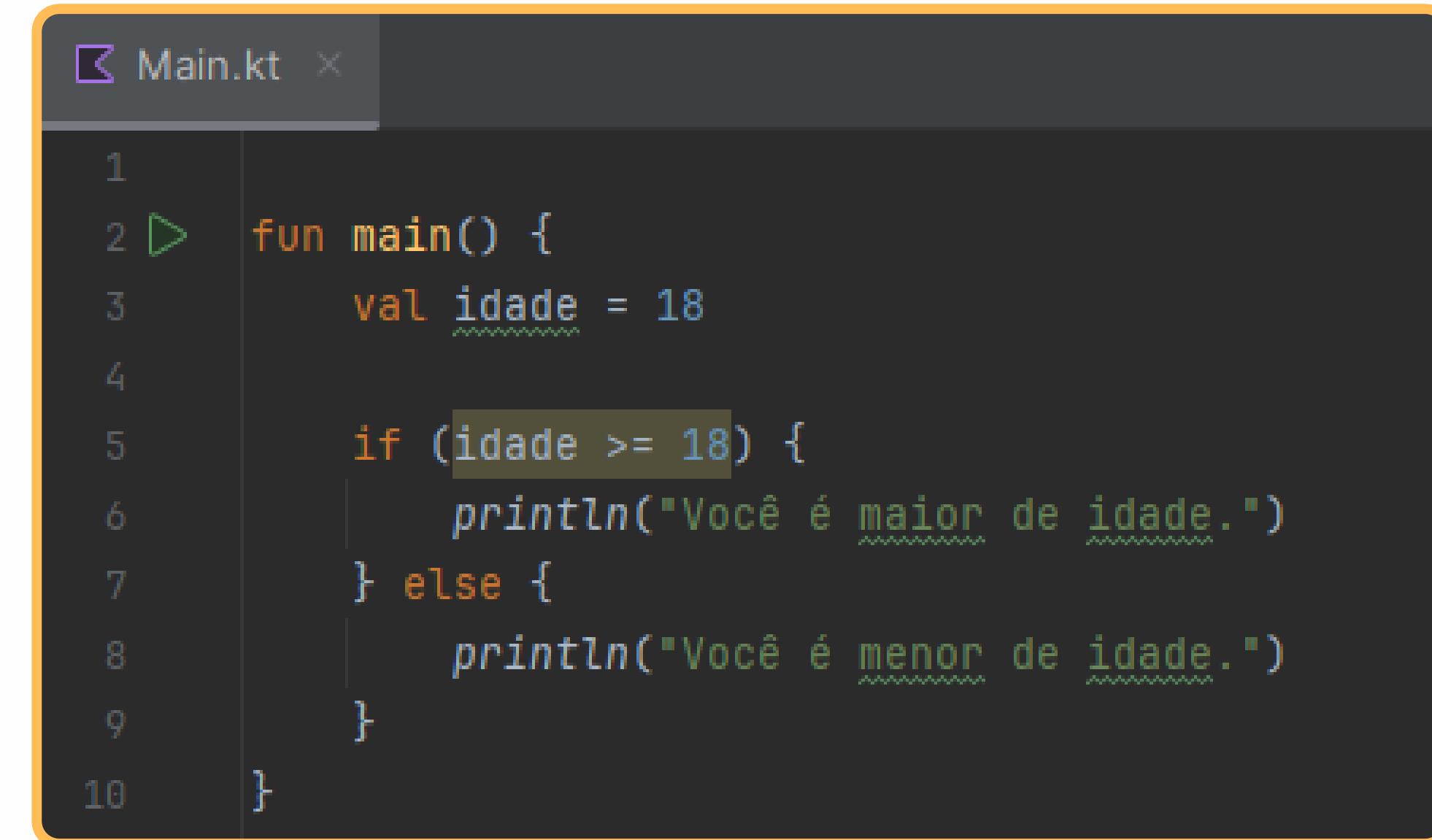
```
>Main.kt
1
2 fun main(){
3     var notas:Float = readIn().toFloat();
4
5
6 }
```

## Conversão de Entrada com readIn() em Kotlin

Tipo de Dado Desejado	Conversão Usada	Exemplo
Inteiro ( Int )	<code>readln().toInt()</code>	<code>val idade = readln().toInt()</code>
Número com ponto ( Float )	<code>readln().toFloat()</code>	<code>val nota = readln().toFloat()</code>
Número com ponto duplo ( Double )	<code>readln().toDouble()</code>	<code>val preco = readln().toDouble()</code>
Número longo ( Long )	<code>readln().toLong()</code>	<code>val populacao = readln().toLong()</code>
Booleano ( Boolean )	<code>readln().toBoolean()</code>	<code>val ativo = readln().toBoolean()</code>

## Estruturas Condicionais: **if**, **else** e o Ternário em **Kotlin**

Em **Kotlin**, as estruturas condicionais são utilizadas para tomar decisões no fluxo do programa com base em condições. A mais comum delas é a instrução **if**, que pode ser combinada com **else** para lidar com diferentes cenários. Sintaxe Básica do **if** e **else**

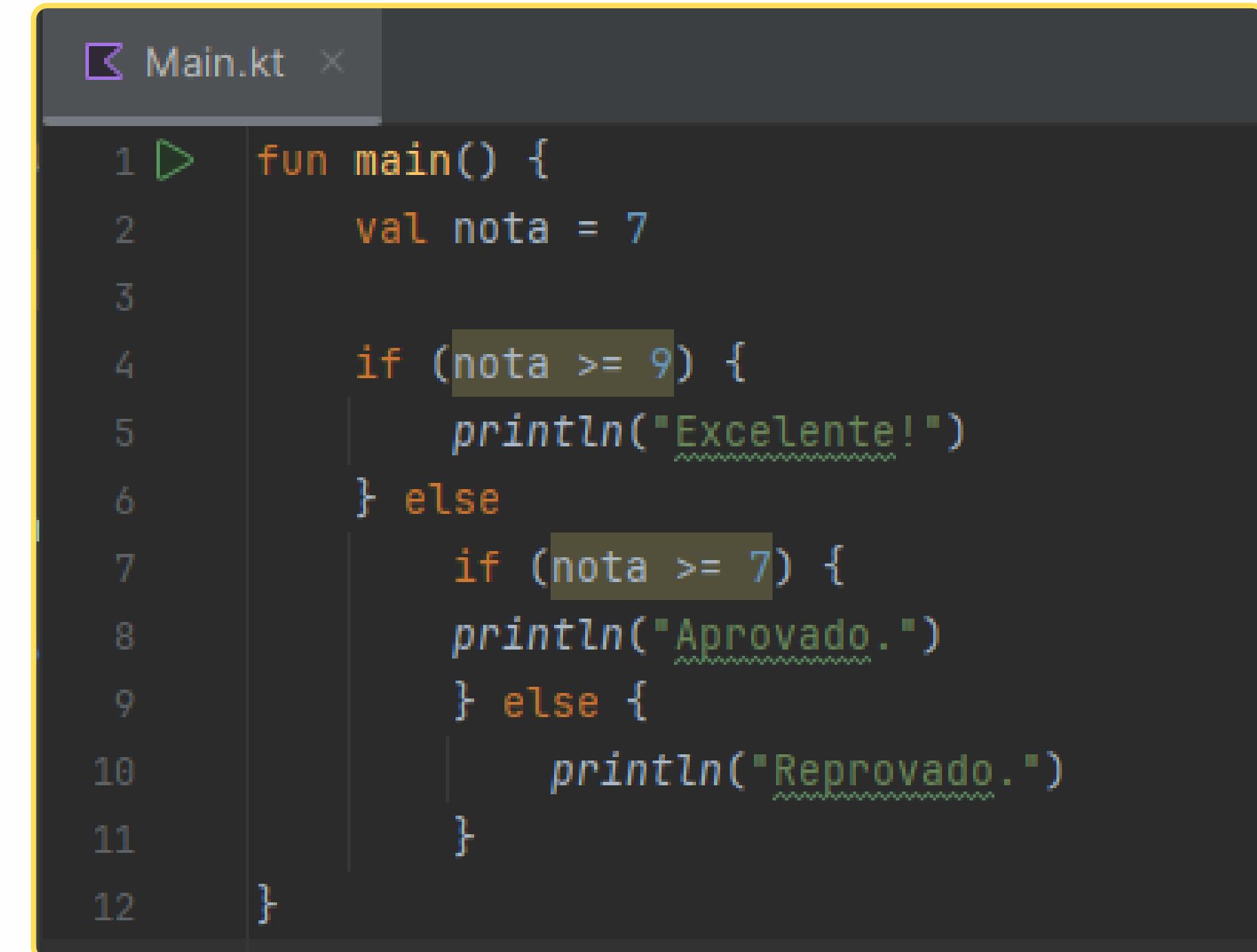


```
1
2 > fun main() {
3     val idade = 18
4
5     if (idade >= 18) {
6         println("Você é maior de idade.")
7     } else {
8         println("Você é menor de idade.")
9    }
10 }
```

## Estruturas Condicionais: **if**, **else** e o Ternário em **Kotlin**

Neste exemplo, o programa verifica se a variável **nota** é maior ou igual a 9. Caso seja, executa o bloco **if**; senão, executa o bloco **else**.

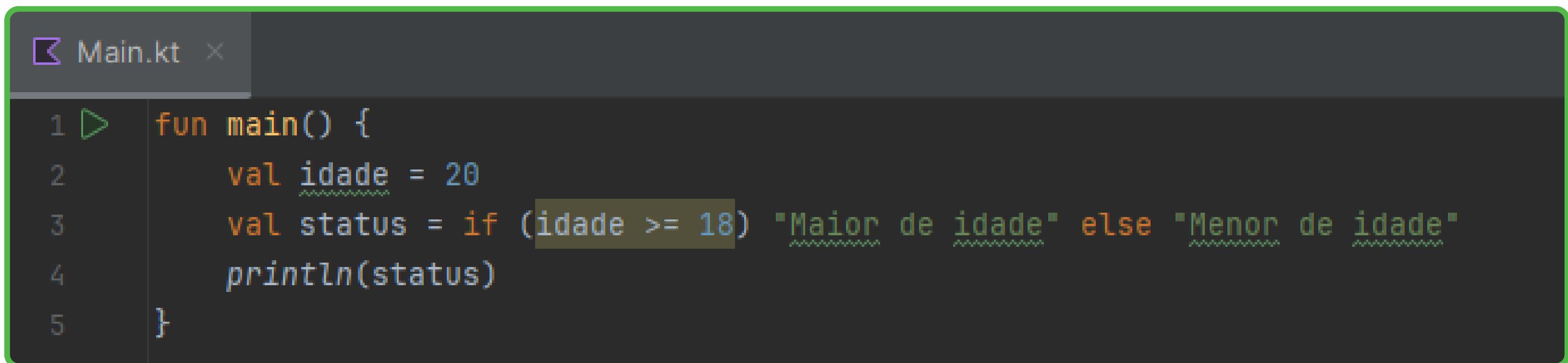
**if com else if** É possível adicionar múltiplas verificações com **else if**:



```
Main.kt
1 fun main() {
2     val nota = 7
3
4     if (nota >= 9) {
5         println("Excelente!")
6     } else
7         if (nota >= 7) {
8             println("Aprovado.")
9         } else {
10            println("Reprovado.")
11        }
12 }
```

## Estruturas Condicionais: if, else e o Ternário em Kotlin

Ao contrário de linguagens como Java ou C, Kotlin não possui o operador ternário (`? :`). Em vez disso, o Kotlin permite que o `if` seja usado como expressão, ou seja, ele retorna um valor. Isso torna o código mais limpo e legível:

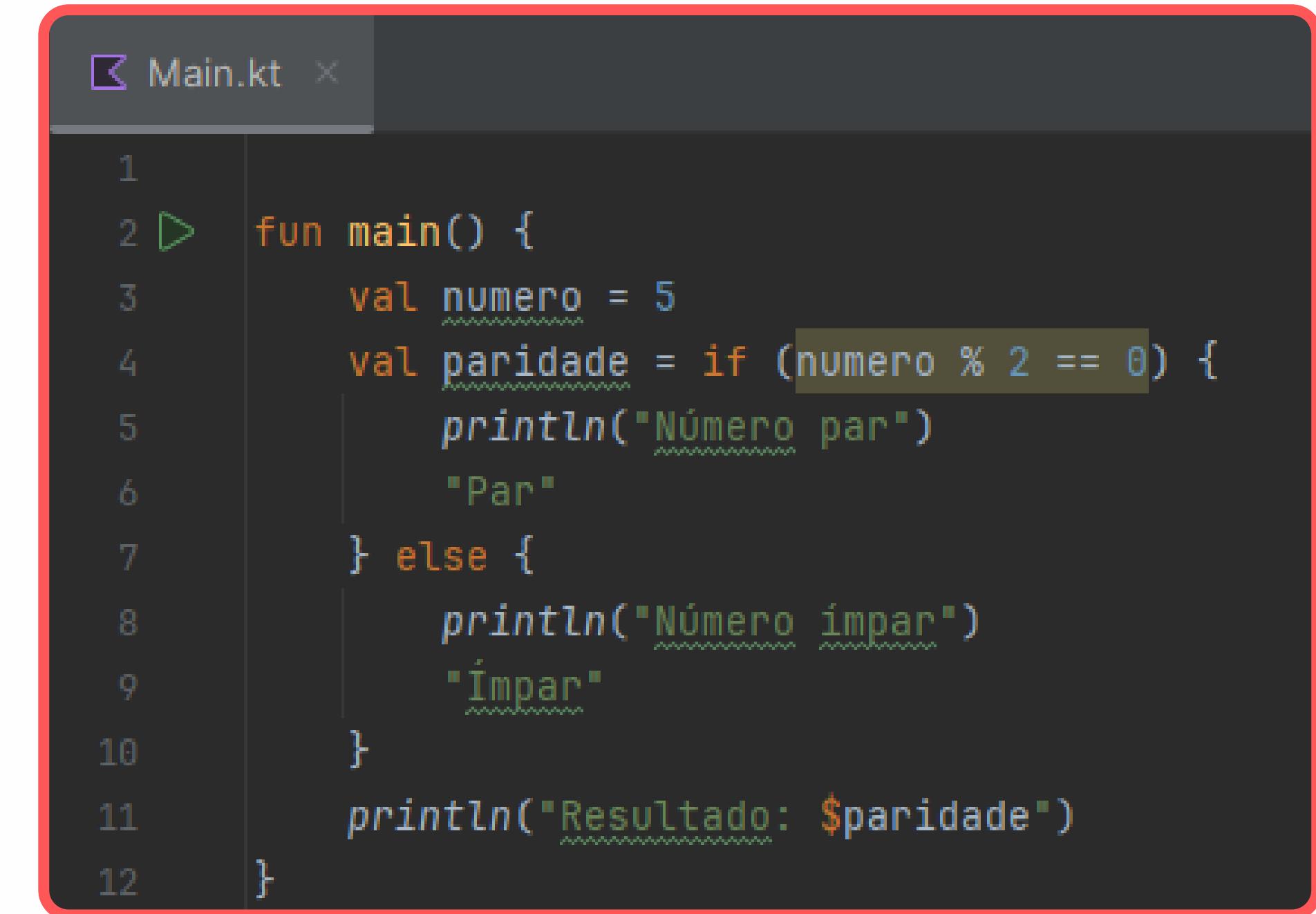


```
Main.kt
1 fun main() {
2     val idade = 20
3     val status = if (idade >= 18) "Maior de idade" else "Menor de idade"
4     println(status)
5 }
```



## Blocos if com Mais de uma Instrução

Se o bloco if ou else tiver mais de uma linha, o valor retornado será o da última instrução:



```
Main.kt
```

```
1
2 fun main() {
3     val numero = 5
4     val paridade = if (numero % 2 == 0) {
5         println("Número par")
6         "Par"
7     } else {
8         println("Número ímpar")
9         "Ímpar"
10    }
11    println("Resultado: $paridade")
12 }
```

# CURSO TÉCNICO EM INFORMÁTICA

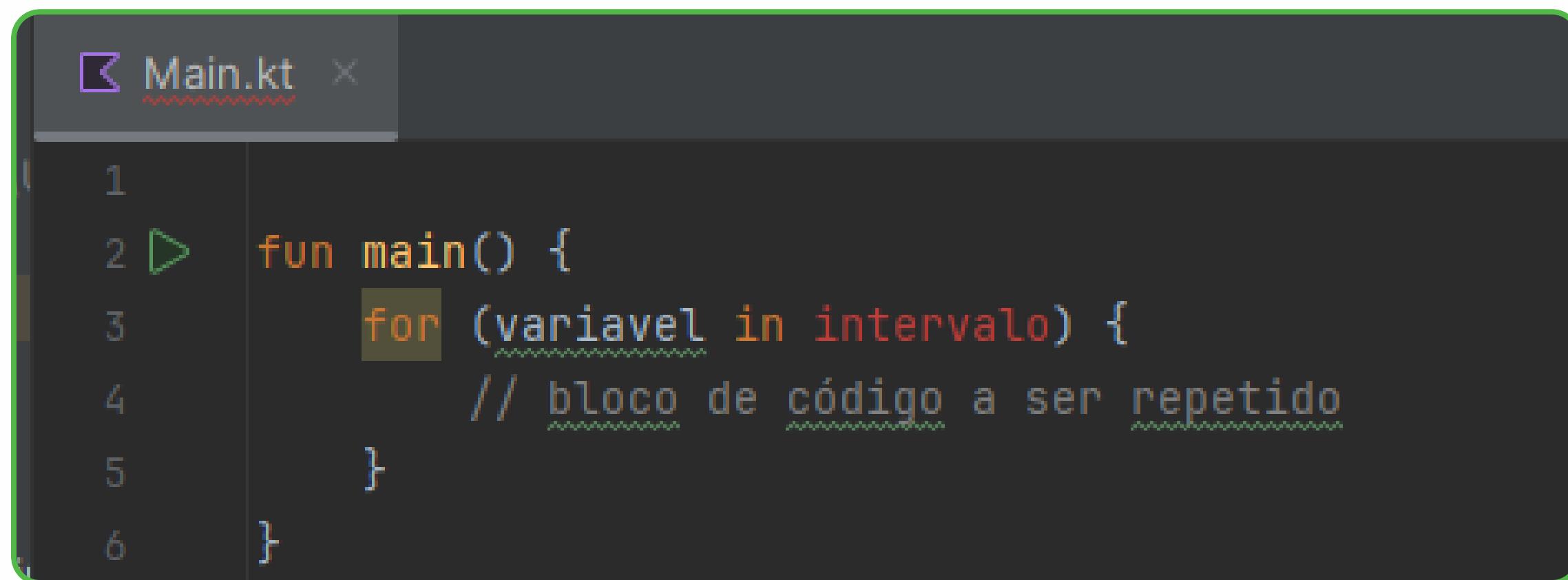
## Vetores



## Estrutura de Repetição for em Kotlin

A estrutura de repetição for é usada para executar um bloco de código várias vezes, geralmente percorrendo uma sequência de valores. Kotlin fornece uma sintaxe poderosa e expressiva para o for, que

Sintaxe Básica do for



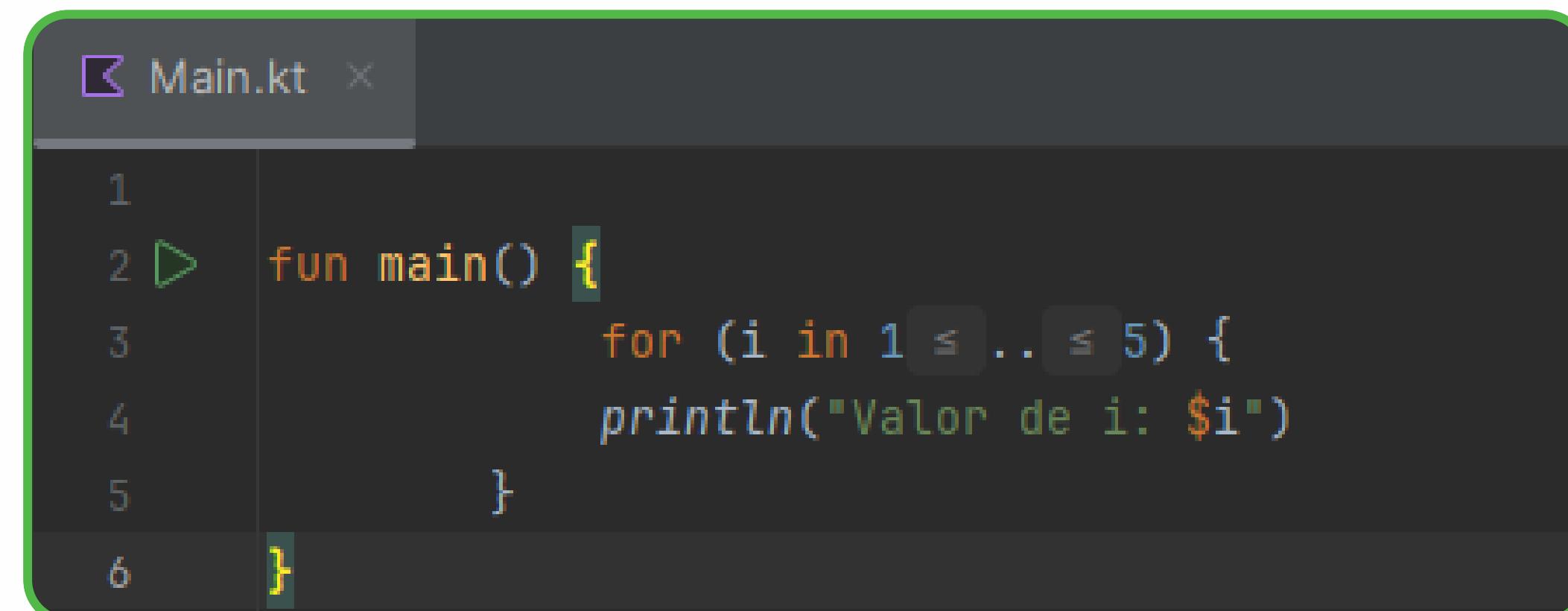
The screenshot shows a code editor window titled "Main.kt". The code is as follows:

```
1
2 > fun main() {
3     for (variavel in intervalo) {
4         // bloco de código a ser repetido
5     }
6 }
```



## Estrutura de Repetição for em Kotlin

Esse laço imprime os números de 1 a 5. O **operador ..** cria um intervalo fechado (inclui o valor final).



```
Main.kt
1
2 ➤ fun main() {
3     for (i in 1 .. 5) {
4         println("Valor de i: $i")
5     }
6 }
```

