# [Tomada de decisão no seu código](https://developer.android.com/codelabs/basic-android-kotlin-training-conditional-dice-roll-behavior?hl=pt-br&continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-kotlin-four%3Fhl%3Dpt-br%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-training-conditional-dice-roll-behavior" \l "1)

O programa Lucky Dice Roller precisa determinar se o usuário tirou um número da sorte e receberá um parabéns ou se receberá uma mensagem para tentar novamente.

Como desenvolvedor de apps, você precisa tomar decisões sobre o comportamento do app e criar resultados diferentes para os usuários.

Ao criar um app de compras, você pode mostrar telas diferentes com base nas opções de entrega escolhidas pelo usuário. Em um jogo de perguntas e respostas, telas diferentes são exibidas com base na resposta do jogador. Dependendo do app, pode haver muitos resultados possíveis que precisam ser considerados no código.

No programa Lucky Dice Roller, o app precisa processar diferentes casos, como:

* **Se** o valor resultante do dado for o número da sorte, exiba uma mensagem de parabéns.
* **Caso contrário, se** o valor resultante do dado não for o número da sorte, exiba uma mensagem para tentar novamente.

Para adicionar essa lógica ao código, use palavras-chave especiais de Kotlin, como if, else e when.

Vejamos alguns exemplos.

### **Usar uma instrução if para configurar uma condição que é atendida**

1. Analise o código abaixo. Você consegue adivinhar qual será o resultado?

fun main() {  
   val num = 5  
   if (num > 4) {  
       println("The variable is greater than 4")  
   }  
}

1. Copie e cole o código no editor de programas Kotlin e execute o programa para observar o resultado.

The variable is greater than 4

O processo de tomada de decisão para esse programa é:

1. Criar uma variável num e defini-la como 5..
2. Caso seja verdade que num é maior que 4, exibir "The variable is greater than 4".
3. Em todas as outras situações, não fazer nada.

No exemplo acima, num foi definido como 5. A instrução if compara se a variável é maior que 4. Como isso é verdade, o sistema executa as instruções nas chaves e exibe a mensagem.

**Resumo**: o símbolo **>** é um operador que compara dois valores, para verificar se o primeiro é maior que o segundo, e retorna um resultado **true** (verdadeiro) ou **false** (falso). Outros operadores comuns são: **<** para "menor que", **==** para "igual a", **>=** para "maior ou igual a" e **<=** para "menor ou igual a".

Observe o formato geral da instrução if:

* Comece com a palavra-chave if.
* Continue com dois parênteses (). Insira a condição dentro dos parênteses. A condição é qualquer coisa que possa ser true ou false. Por exemplo, se um número é maior que outro.
* Continue com duas chaves {}. Dentro das chaves, coloque o código que será executado se a condição for true.

if (condition-is-true) {

execute-this-code

}

**Tipo de dados booleanos para valores verdadeiros e falsos:**

* O processo em que um programa testa se uma condição foi atendida é chamado de "avaliação da condição". O resultado da avaliação de uma condição será **true** se ela for atendida ou **false** se não for. Os desenvolvedores também dizem: "a condição foi avaliada como **true**" ou "a condição foi avaliada como **false**".
* Condições só podem ser avaliadas como **true** ou **false**.
* Assim como existe um tipo de dados **Int** para números inteiros e **IntRange** para intervalos, existe um tipo de dados para **true** e **false**, chamado **Boolean**. Você verá variáveis do tipo **Boolean** mais adiante neste curso.

### **Usar uma instrução if para configurar uma condição que não é atendida**

1. Mude o valor de num para 3, conforme mostrado abaixo. O que você espera que aconteça ao executar este código?

fun main() {  
    val num = 3  
    if (num > 4) {  
        println("The variable is greater than 4")  
    }  
}

1. Copie e cole o código no editor de programas Kotlin e execute o programa para ver o resultado.

Com num definido como 3, nada é exibido, porque o valor de num é menor que 4. Dessa forma, a condição em que num é maior que 4 é false, e o código entre as chaves não é executado, não exibindo nada.

### **Usar else para criar uma alternativa para condições com falha**

Em vez de não fazer nada, você pode oferecer uma alternativa ao usuário quando uma condição não for atendida. Assim como acontece com a linguagem normal, você pode fazer isso com uma instrução else.

1. Adicione uma instrução else para exibir uma mensagem quando num não for maior que 4, conforme mostrado abaixo. O que você espera que aconteça ao executar este código?

fun main() {  
    val num = 3  
    if (num > 4) {  
        println("The variable is greater than 4")  
    } else {  
        println("The variable is less than 4")  
    }  
}

1. Copie e cole o código no editor de programas Kotlin e execute o programa para observar o resultado.

The variable is less than 4

1. Quando num tiver um valor de 3, o programa exibirá a mensagem "The variable is less than 4" associada à instrução else, porque num não é maior que 4.
2. Mude num para 5 e execute o programa novamente. Agora, é verdade que num é maior que 4 e o programa exibirá "The variable is greater than 4".
3. Mude num para 4 e execute o programa. Agora 4 não é maior que 4, e o programa exibe "The variable is less than 4".

Embora "The variable is less than 4" seja o resultado correto para as condições definidas no código, a instrução exibida não está correta, já que 4 não é menor que 4. É possível adicionar outra condição que verifica uma terceira possibilidade, se num é igual a 4, e exibe uma instrução correta quando a condição é verdadeira.

### **Usar uma combinação de else + if para adicionar condições alternativas**

É possível ter mais de uma condição. No exemplo, veja como é possível abranger todas as possibilidades de num:

* **Se** num for maior que 4, exibir "The variable is greater than 4".
* **Além disso, se** num for igual a 4, exibir "The variable is equal to 4".
* **Caso contrário**, exibir "The variable is less than 4".

Esses são casos diferentes da instrução if-else. Há três casos listados.

O código atualizado ficará assim:

fun main() {  
    val num = 4  
    if (num > 4) {  
        println("The variable is greater than 4")  
    } else if (num == 4) {  
        println("The variable is equal to 4")  
    } else {  
        println("The variable is less than 4")  
    }  
}

Observe as seguintes mudanças:

* O valor de num agora está definido como 4. Portanto, você pode testar a nova condição.
* Entre as instruções originais if e else, há uma nova instrução else if para o caso em que num é igual a 4.

1. Copie e cole o código acima no editor de programas Kotlin e execute o programa para observar o resultado.

The variable is equal to 4

1. Tente mudar o valor de num e ver como isso afeta o resultado. Mude num para 2 e 6. Assim, você poderá ver cada uma das condições retornar true.

### **Fluxo de controle**

Observando as instruções if-else acima, o código será executado, ou terá um fluxo, de acordo com as condições. A forma como essa execução é orientada pelas condições é chamada de "fluxo de controle" do programa.

* Digamos que o num resultante do dado seja 3. O programa verifica a primeira condição (num > 4). Ela é falsa. Então, o programa verifica a próxima condição (num == 4), que também é falsa. Em seguida, o programa executa o código da instrução else, que é a última opção.
* Se o valor resultante do dado for igual a 6, a primeira condição (num > 4) será verdadeira. O programa exibe a mensagem "The variable is greater than 4". Como essa condição é verdadeira, ele não precisa verificar o restante. Portanto, ele não usa a instrução if-else.
* Para adicionar condições alternativas, use uma combinação else + if.

# [Criar o jogo Lucky Dice Roll](https://developer.android.com/codelabs/basic-android-kotlin-training-conditional-dice-roll-behavior?hl=pt-br&continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-kotlin-four%3Fhl%3Dpt-br%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-training-conditional-dice-roll-behavior#2)

Você pode editar a função main() no código anterior para que fique igual a esta ou copiar e colar o código abaixo para começar.

fun main() {  
    val myFirstDice = Dice(6)  
    val rollResult = myFirstDice.roll()  
    println("Your ${myFirstDice.numSides} sided dice rolled ${rollResult}!")  
}  
  
class Dice (val numSides: Int) {  
  
    fun roll(): Int {  
        return (1..numSides).random()  
    }  
}

**Observação**: toda vez que você executar o programa acima no playground, a função **main()** será chamada. Isso cria uma nova instância de **Dice** e chama um método **roll()** nela. Por isso, toda vez que você executar o programa, o valor do dado poderá ser diferente.

Verificar se o número da sorte foi tirado

Primeiro, crie um número de sorte e, então, compare o valor resultante do dado com esse número.

.

1. Em main(), exclua a instrução println().
2. Em main(), adicione um val chamado luckyNumber e defina-o como 4. O código ficará assim.

fun main() {  
    val myFirstDice = Dice(6)  
    val rollResult = myFirstDice.roll()  
    val luckyNumber = 4  
}

1. Abaixo, adicione uma instrução if com uma condição dentro dos parênteses (), que verifica se rollResult é igual (==) a luckyNumber. Deixe um pouco espaço entre as chaves {} para adicionar mais códigos.

fun main() {  
    val myFirstDice = Dice(6)  
    val rollResult = myFirstDice.roll()  
    val luckyNumber = 4  
    if (rollResult == luckyNumber) {  
  
    }  
}

1. Dentro das chaves {}, adicione uma instrução println para imprimir "You win!"

fun main() {  
    val myFirstDice = Dice(6)  
    val rollResult = myFirstDice.roll()  
    val luckyNumber = 4  
  
    if (rollResult == luckyNumber) {  
        println("You win!")  
    }  
}

1. Execute seu programa. Pode ser necessário executá-lo várias vezes antes de ter sorte e visualizar a mensagem de vencedor como resultado.

You win!

Responder quando o número da sorte não for tirado

Se o usuário não receber nenhum retorno caso não tenha ganhado, poderá achar que o programa não está funcionando. É recomendável sempre dar uma resposta quando o usuário fizer algo. No programa Lucky Dice Roller, use a instrução else para informar que ele não ganhou.

1. Adicione uma instrução else para exibir "You didn't win, try again!".

fun main() {  
    val myFirstDice = Dice(6)  
    val rollResult = myFirstDice.roll()  
    val luckyNumber = 4  
  
    if (rollResult == luckyNumber) {  
        println("You win!")  
    } else {  
        println("You didn't win, try again!")  
    }  
}

1. Execute o programa e, independentemente do resultado, o usuário sempre será notificado.

Neste ponto, o usuário sabe se ganhou ou não, mas não sabe o porquê. Sempre dê informações ao usuário para que ele entenda o resultado das ações. Imagine que seu programa fosse um aplicativo de empréstimos. "Você não foi aprovado porque seu índice de crédito é ruim" é muito mais informativo que "Não há empréstimos disponíveis para você. Tente novamente." No Lucky Dice Roller, é possível mostrar ao usuário mensagens informativas diferentes sobre cada jogada perdida. Use várias instruções else if para fazer isso.

1. Adicione instruções else if para exibir uma mensagem diferente para cada jogada. Consulte o formato que você aprendeu na tarefa anterior, se necessário.

fun main() {  
    val myFirstDice = Dice(6)  
    val rollResult = myFirstDice.roll()  
    val luckyNumber = 4  
  
    if (rollResult == luckyNumber) {  
        println("You win!")  
    } else if (rollResult == 1) {  
        println("So sorry! You rolled a 1. Try again!")  
    } else if (rollResult == 2) {  
        println("Sadly, you rolled a 2. Try again!")  
    } else if (rollResult == 3) {  
        println("Unfortunately, you rolled a 3. Try again!")  
    } else if (rollResult == 5) {  
        println("Don't cry! You rolled a 5. Try again!")  
    } else {  
        println("Apologies! You rolled a 6. Try again!")  
    }  
}

No código acima:

* Você verifica se o rollResult é o luckyNumber.
* Se rollResult é o luckyNumber, exibe a mensagem vencedora.
* Caso contrário, verifica se o rollResult é 1 e, se for, exibe uma mensagem para nova tentativa.
* Caso contrário, verifica se o rollResult é 2 e, se for, exibe outra mensagem para nova tentativa.
* Caso contrário, continua verificando até chegar ao número 5.
* Caso o resultado não tenha sido nenhum número de 1 a 5, a única opção restante será o 6. Assim, não será necessário fazer outro teste com else if, e você poderá usar a última opção com a instrução else final.

**Dica**: só é possível ter uma instrução **if** com uma instrução **else** em um bloco de código if-else. Porém, é possível ter quantas instruções **else if** forem necessárias entre elas.

Como é muito comum ter vários casos de else if, o Kotlin tem uma forma mais simples de criá-los.

# [Usar uma instrução when](https://developer.android.com/codelabs/basic-android-kotlin-training-conditional-dice-roll-behavior?hl=pt-br&continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-kotlin-four%3Fhl%3Dpt-br%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-training-conditional-dice-roll-behavior#3)

Testar vários resultados diferentes, ou casos, é muito comum em programação. Às vezes, a lista de resultados possíveis pode ser muito longa. Por exemplo, ao lançar um dado de 12 lados, você teria 11 instruções else if entre o resultado correto e o else final. Para facilitar a criação e a leitura desses tipos de instruções, o que ajuda a evitar erros, o Kotlin disponibiliza uma instrução when.

Você mudará o programa para passar a usar uma instrução when. A instrução when começa com a palavra-chave when, seguida de parênteses (). Dentro do parênteses, é inserido o valor a ser testado. Isso é seguido de chaves {} do código que será executado para condições diferentes.

1. No programa, em main(), selecione o código desde a primeira instrução if até a chave } que fecha a última instrução else e exclua esse trecho.

fun main() {  
    val myFirstDice = Dice(6)  
    val rollResult = myFirstDice.roll()  
    val luckyNumber = 4  
}

1. Em main(), abaixo da declaração de luckyNumber, crie uma instrução when. Como when precisa testar o valor resultante do dado, coloque rollResult entre parênteses (). Adicione as chaves {} deixando espaço de sobra, conforme mostrado abaixo.

fun main() {  
    val myFirstDice = Dice(6)  
    val rollResult = myFirstDice.roll()  
    val luckyNumber = 4  
  
    when (rollResult) {  
  
    }  
}

Assim como antes, primeiro teste se rollResult é igual ao luckyNumber.

1. Dentro das chaves {} da instrução when, adicione uma instrução que testa rollResult em comparação ao luckyNumber. Se os dois forem iguais, exiba a mensagem vencedora. A instrução ficará assim:

luckyNumber -> println("You win!")  
Isso significa que:

* Primeiro, insira o valor que está sendo comparado ao rollResult. Ele é o luckyNumber.
* Depois, insira uma seta (->).
* Em seguida, adicione a ação a ser realizada se houver correspondência.

Leia como "Se o rollResult for o luckyNumber, exibir a mensagem "You win!"".

O código main() ficará assim:

fun main() {  
    val myFirstDice = Dice(6)  
    val rollResult = myFirstDice.roll()  
    val luckyNumber = 4  
  
    when (rollResult) {  
        luckyNumber -> println("You win!")  
    }  
}

1. Use o mesmo padrão para adicionar linhas e mensagens para os possíveis resultados (1 a 6, exceto 4), conforme mostrado abaixo. A função main() concluída ficará parecida com esta:

fun main() {  
    val myFirstDice = Dice(6)  
    val rollResult = myFirstDice.roll()  
    val luckyNumber = 4  
  
    when (rollResult) {  
        luckyNumber -> println("You won!")  
        1 -> println("So sorry! You rolled a 1. Try again!")  
        2 -> println("Sadly, you rolled a 2. Try again!")  
        3 -> println("Unfortunately, you rolled a 3. Try again!")  
        5 -> println("Don't cry! You rolled a 5. Try again!")  
        6 -> println("Apologies! you rolled a 6. Try again!")  
    }  
}

1. Execute seu programa. Não há diferença no resultado, mas o código está muito mais compacto e mais fácil de ler.

Parabéns! Você aprendeu duas formas de exibir mensagens dependendo de uma condição. Essa é uma ferramenta eficiente para escrever programas interessantes.

# [Código da solução](https://developer.android.com/codelabs/basic-android-kotlin-training-conditional-dice-roll-behavior?hl=pt-br&continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-kotlin-four%3Fhl%3Dpt-br%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-training-conditional-dice-roll-behavior#4)

fun main() {  
    val myFirstDice = Dice(6)  
    val rollResult = myFirstDice.roll()  
    val luckyNumber = 4  
  
    when (rollResult) {  
        luckyNumber -> println("You won!")  
        1 -> println("So sorry! You rolled a 1. Try again!")  
        2 -> println("Sadly, you rolled a 2. Try again!")  
        3 -> println("Unfortunately, you rolled a 3. Try again!")  
        5 -> println("Don't cry! You rolled a 5. Try again!")  
        6 -> println("Apologies! you rolled a 6. Try again!")  
   }  
}  
  
class Dice(val numSides: Int) {  
    fun roll(): Int {  
        return (1..numSides).random()  
    }  
}