

# Conditions with `if`

Learn how to use Kotlin's `if` conditions as well as comparison and logical operators.

## WE'LL COVER THE FOLLOWING



- Conditions using `if`
  - Equality and Comparison Operators
  - Logical Operators
- Quiz
- Exercise
- Summary

For conditional control flow, Kotlin uses the `if` and `when` keywords. The syntax of `if`-blocks is the same as in C and Java. Similarly, the `when` keyword is similar to `switch` in these languages but more powerful, as you will learn.

## Conditions using `if` #

Conditions with `if` can be written as follows:

```
if (planet == "Jupiter") {  
    println("Radius of Jupiter is 69,911km")  
} else if (planet == "Saturn") {  
    println("Radius of Saturn is 58,232km")  
} else {  
    println("No data for planet $planet")  
}
```



Each condition is followed by a block of code in curly braces. There can be any number of else-if blocks (including zero) and up to one else-block.

**Note:** In the `else`-block, the `planet` variable is included in the output string using *string interpolation*. To do so, you just prepend the variable with a `$`. More complex expressions must be wrapped with curly braces: `"${user.name} logged in"`.

## Equality and Comparison Operators #

Kotlin's operators to formulate conditions are the following:

Operator	Meaning
<code>==</code>	Structural equality ("equals")
<code>!=</code>	Structural inequality ("!=equals")
<code>===</code>	Referential equality (same memory address)
<code>!==</code>	Referential inequality
<code>&gt;</code>	Greater than
<code>&lt;</code>	Less than
<code>&gt;=</code>	Greater or equal than
<code>&lt;=</code>	Less than or equal

Note that the comparison operators work on any `Comparable` by using its `compareTo` method.

**Note for Java developers:** In Kotlin, you use `==` where you would use `equals` in Java and `===` where you would use Java's `==`.

## Logical Operators #

To build up more complex conditions from primitive conditions (using the operators above), Kotlin provides the standard logical operators:

<i>Operator</i>	<i>Meaning</i>
<code>&amp;&amp;</code>	Logical “and”
<code>  </code>	Logical “or”
<code>!</code>	Logical “not”

Using these, you can combine multiple primitive conditions:

```
if (planet == "Jupiter" || planet == "Saturn" || planet == "Uranus" || planet == "Neptune") {  
    println("Your chosen planet has rings, typically made of rocks and ice boulders")  
}
```



## Quiz #

Conditional control flow using `if`

Q

How would you express the condition “the text is empty or at least 3 characters long”?

COMPLETED 0%

1 of 1



## Exercise #

Complete the following code snippet to check whether...

1. the `age` variable is between 18 and 21 (both inclusive).
2. the `username` variable equals `"admin"` or `"system"`
3. the `number` variable is equal to neither `17` nor `42`.

 Problem

 Solution

```
// Note: In this code widget, you have access to the predefined variables age, username, and  
  
// Check age  
  
// Check username  
  
// Check number
```



## Summary #

- Kotlin's `if` conditions work the exact same way as in other languages like Java or C.
- The logical and comparison operators are also known from some other languages.
  - But in contrast to Java, Kotlin offers `==` for structural equality and `===` for referential equality checks.

In the next lesson, you will use the `when` statement for conditional control

flow and learn when to prefer it over `if`.