`while` Loops

Get started with Kotlin's loop constructs for repeating blocks of code.

WE'LL COVER THE FOLLOWING

- `while` Loops
- `do-while` Loops
- Quiz
- Summary

Kotlin offers the three standard types of loops:

- `while` loops
- `do-while` loops
- `for` loops

As a programmer, loops in Kotlin will feel familiar quickly. Let's look at a few examples, starting with `while` and `do-while` loops.

`while` Loops

Kotlin's `while` loops have exactly the same syntax and semantics as `while` loops in Java or C:

```
val input = 17
var sqrtApprox = 1.0
var delta = 1.0

while (delta > 0.001) {  // `while` block is repeated until error gets below threshold
  sqrtApprox = 0.5 * (sqrtApprox + input / sqrtApprox)
  delta = Math.abs((input - sqrtApprox*sqrtApprox) / (2*sqrtApprox))
}

println("sqrt($input) is approximately $sqrtApprox")
```

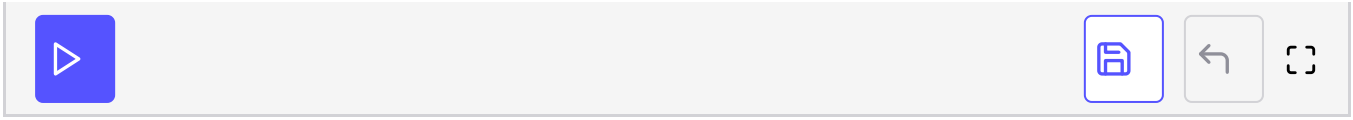
The `while` keyword is followed by a condition. The loop will repeat its code block until this condition evaluates to `false`.

This code example approximates the square root of the `input` up to an error margin of at most `0.001`.

`do-while` Loops

The `do-while` loop checks its condition *after* each iteration, whereas the regular `while` loop checks it *before* each iteration. Consequently, a `do-while` loop always performs at least one iteration. This is why it's typically used in cases where that first iteration gets some initial data, e.g. user input:

```
do {
  print("> ")
  val input = getUserInput()
  println("User typed: $input")
} while (input != ":q")
```



The advantage here is that you can limit the scope of the `input` variable to the loop block, which is not possible using a regular `while` loop. It's good practice to minimize the scopes of your variables.

Quiz

Kotlin's `while` Loops

1

What's the output of the following program?

```
println("Output: ")
while(false) {
  println("#")
}
```

> **Note:** Exercises follow in the next lesson on `for` loops.

Summary

Kotlin provides two slightly different `while` loop structures:

- The regular `while` loop checks its condition *before* each iteration.
- The `do-while` loop checks its condition *after* each iteration.
- Generally, `while` loops are used when the number of iterations is not fixed.

In the next lesson, you will explore Kotlin's `for` loops as an alternative to `while` loops, and learn when to prefer which type of loop.