Introduction to External Iteration with Argument Matching

Count the number of times you've used for loops in your code. Shocking. Something that's so widely used and fundamental as iterating should be fluent, concise, easy to write, and effortless to understand. Yet, in C-like languages, `for` loops have rather been primitive and verbose. Not so in Kotlin.

Kotlin provides both external iterators, used in the imperative style of programming, and internal iterators, used in the functional style. With external iteration, you, as the programmer, control the sequence of iteration explicitly, for example, with `i++`, `i--`, and so on. Internal iteration takes charge of the sequencing and lets programmers focus on the actions or computations for each iteration, resulting in less code and fewer errors. We'll explore the benefits of internal iteration in Chapter 11, Functional Programming with Lambdas, but in this chapter we'll focus on the fluency Kotlin provides for working with the common external iteration.

In this chapter, you'll learn the features in Kotlin to concisely and elegantly iterate over ranges of values and through objects in collections. You'll see that code you write to sift through data and make decisions doesn't have to be a series of repeated `if-else` blocks. Kotlin's elegant argument-matching capabilities will remove so much cruft and make the decision logic highly transparent to the reader.

Once you pick up the facilities provided in Kotlin, you'll never want to go back to the old ways of iterating. Concise, pleasant, and expressive syntax will keep you moving, motivated, and focused on the problem at hand. As a result, you'll write less code to implement your business rules.

Since we often work with ranges of values, you can readily use the classes and functions in the `kotlin.ranges` package to nicely iterate, turning the task of writing that code into a pleasant experience. The enhanced for loop is not only for a range of values; you may use it to iterate over values in a collection as well.

We'll also look at the elegant argument-matching syntax and the powerful when

expression, which will save you from drowning in deeply nested code. This will become your go-to syntax for taking different actions on different pieces of data. 1, 2, 3… let's start the iteration.