Introduction to Kotlin Essentials

> *Kotlin is a language that scales—you can create scripts with a few lines, or an entire enterprise application with many classes and libraries. Our objective is to deliver value, not to write a lot of code. From that point of view, the conciseness and effectiveness of every line of code matters.*
>
> *In this part, you'll learn about type inference, defining variables, using string templates, and creating multiline strings. Then we'll look at how to create functions to elegantly iterate over ranges of values and to process data using the argument-matching capabilities of Kotlin.*
>
> *You'll learn to create scripts, but that knowledge will carry you forward to creating OO code, writing functional code...all the way to creating enterprise and Android applications.*

Simple things should be easy to create, and complicated things should be affordable. Quick, how many lines of code do you need to programmatically find the number of cores on a machine? How about the following:

```
println(Runtime.getRuntime().availableProcessors())
```

That's the entire program in Kotlin—with only half a line of code, no need for a semicolon, no imports, and using the Java JDK but with far less code and ceremony. That's Kotlin.

Kotlin is about getting your work done; it doesn't impose ceremony on you. You can start small and scale up. Programming is a series of mini experiments; you often prototype solutions to see if things make sense, and write code to get a feel and to tease out design ideas. Kotlin scripts are a great way to achieve that quickly without writing wasteful code.

In this chapter, you'll learn a few essentials—the building blocks that typically go into the body of functions and methods but which you can drop directly into scripts, as well, in Kotlin. These include defining variables of numbers and strings, creating constants, specifying the type information, creating a string expression with embedded values, creating multiline strings—a hodgepodge of features that we'll use often when programming with Kotlin, whether it's writing scripts, creating classes, or writing functional-style code.

Experienced Java programmers who begin to program in Kotlin have to unlearn a few of their Java practices while they learn the nuances of Kotlin. In this chapter, we'll look at things that eyes very familiar with Java have to adjust to when transitioning to Kotlin. Just like wearing a new pair of glasses or contacts, it takes some getting used to, but Kotlin's elegance will make you feel comfortable in no time.

We'll start with an appreciation for conciseness in Kotlin. Every line of code matters you can't make an entire application concise when the most fundamental operations are verbose. Whether you write imperative-style, functional-style, or object-oriented code, expressions and statements are the building blocks. Kotlin removes ceremony, noise, and clutter from every single line of code, so your overall program can be shorter, more expressive, and easier to maintain.

Kotlin makes optional a number of things that are required in other languages—semicolons, explicit type specifications, and classes, to mention a few. Kotlin requires you to make a conscious decision at the time of creating a variable if it should be mutable or immutable. It provides string templates to ease the pain of creating string expressions and multiline strings. And it favors expressions over statements, to reduce the need for mutating variables.

Let's dive in and see the root of Kotlin's design philosophy at the most rudimentary part of code, a single line of statement or expression. Along the way, we'll look at how Kotlin reduces noise in code and how you can define variables, use type inference, rely on sensible warnings, easily embed expressions in strings, and create multiline strings. Learning these fundamental and essential concepts first will help to influence every line of code you write in Kotlin.

The next lesson explores how less code in Kotlin does more work.