# Introduction to Functions in Kotlin

Kotlin doesn't insist that you create classes for everything. No one gets praise for duplicating code, but to reuse doesn't mean building a class hierarchy. Unlike Java where a class is the smallest reusable piece, in Kotlin both standalone functions and classes can be reused.

Kotlin takes a highly pragmatic approach to creating good quality code—create small simple standalone functions where they suffice and roll your code into methods of classes only when necessary. In this chapter, we'll focus on standalone functions for two reasons. First, because we can—that is, in Kotlin we can create top-level standalone functions to reuse code and don't have to waste our time and effort with classes if there's little value. Second, all the capabilities of functions directly carry over to creating methods of classes—after all, methods are simply functions that run in the context of classes or objects. So what you learn here is useful when working with scripts, procedural code, and functional code, as well as when building complex object-oriented code.

With Kotlin you don't have to masquerade standalone functions as static methods of a class—that is, you don't have to pretend to do OO to please the language. You can create global top-level functions, as in languages like C and C++, if that's right for your application. Functions may reside at the top level or directly within packages—you decide where to place them.

Kotlin requires that you specify the types of the parameters to functions, but you can ask it to infer the return type for single-expression functions. When calling functions, you're not required to pass an argument for every parameter; instead you may choose to use default arguments. You can use this feature to easily evolve functions and methods. To make the call to methods expressive, Kotlin gives you the power to name your arguments. This greatly increases the readability of the code. In addition, you may pass a variable number of arguments to functions without losing compile-time safety. Kotlin also has the capability for destructuring, which provides a highly concise way to extract properties from objects into standalone variables.

In this chapter, you'll learn how to work with global or standalone functions. We'll start with Kotlin’s rules for defining functions, look at how it treats functions as expressions, and then examine many useful features, including default arguments, named arguments, defining variable number of arguments, the spread operator, and using destructuring. Using these features, you can create highly expressive code that’s easy to read and also more flexible to maintain. If you’re more interested in writing object-oriented code, be assured that the concepts you learn here apply to methods of classes as well.

Let’s have some fun with functions in the coming lessons.