Wrapping Up

Kotlin doesn't force you to create methods; you can create top-level functions as well. This opens a few more design choices in Kotlin than Java has—applications aren't required to comprise of objects only, they can be composed of functions too. This allows you to create procedural, object-oriented, or functional-style code, whichever is a better choice in a given context. The compiler can infer return types for single-expression, non-block functions. The types of parameters are always required, and that's good.

Default arguments make it a lot easier to extend functions in Kotlin and reduce the need to overload functions. `vararg` parameters offer the flexibility to pass a discrete number of arguments with type safety, and the spread operator gives you a nice way to explode an array argument into a `vararg` parameter. Using named arguments is a great way to make code readable; it's a way to write self-documenting code. Finally, destructuring is a capability that can reduce noise in code and make the code highly concise.

In the next chapter, we'll learn about iterating over a range of values and processing data using the argument-matching facility of Kotlin.