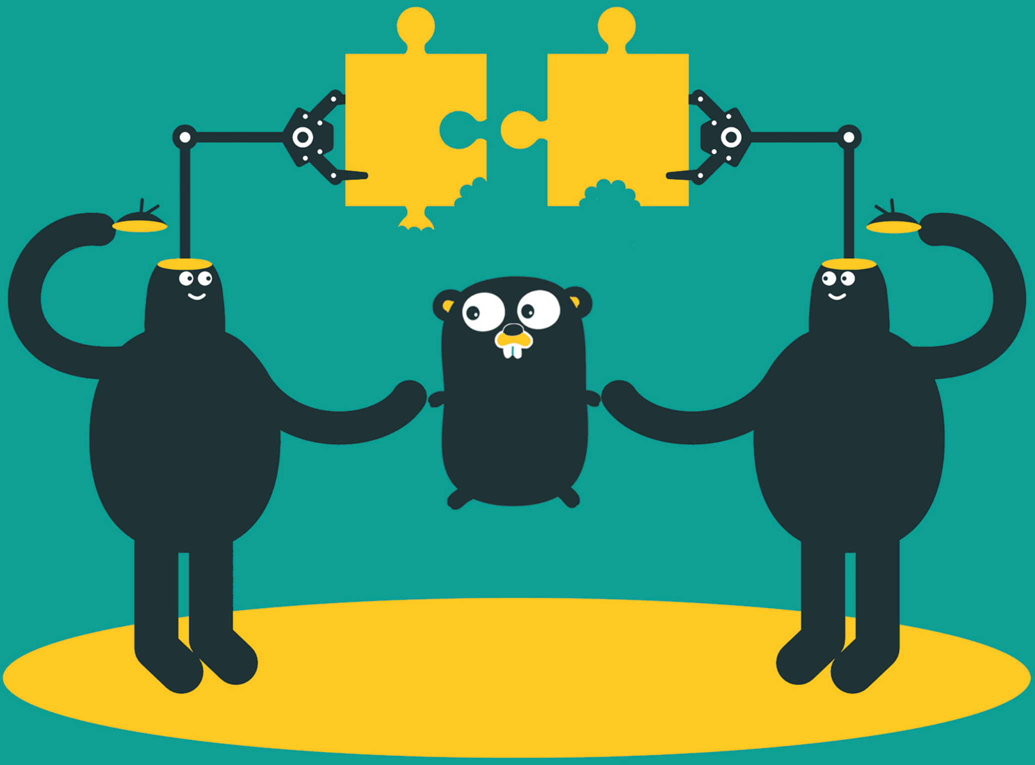
GET PROGRAMMING WITH GO



Nathan Youngman
Roger Peppé

MANNING

SAMPLE LESSON

Get Programming with Go

by Nathan Youngman

Roger Peppé

Lesson 1

Contents

UNIT 0

Getting started

Traditionally, the first step to learning a new programming language is to set up the tools and environment to run a simple "Hello, world" application. With the Go Playground, this age-old endeavor is reduced to a single click.

With that out of the way, you can begin learning the syntax and concepts needed to write and modify a simple program.

LESSON 1

GET READY, GET SET, GO

After reading lesson 1, you'll be able to

- Know what sets Go apart
- Visit the Go Playground
- Print text to the screen
- Experiment with text in any natural language

Go is the contemporary programming language of *cloud computing*. Amazon, Apple, Canonical, Chevron, Disney, Facebook, General Electric, Google, Heroku, Microsoft, Twitch, Verizon, and Walmart are among the companies adopting Go for serious projects (see thenewstack.io/who-is-the-go-developer/ and golang.org/wiki/GoUsers). Much of the infrastructure underlying the web is shifting to Go, driven by companies like CloudFlare, Cockroach Labs, DigitalOcean, Docker, InfluxData, Iron.io, Let's Encrypt, Light Code Labs, Red Hat CoreOS, SendGrid, and organizations like the Cloud Native Computing Foundation.

Go excels in the data center, but its adoption extends beyond the workplace. Ron Evans and Adrian Zankich created Gobot (gobot.io), a library to control robots and hardware. Alan Shreve created the development tool ngrok (ngrok.com) as a project to learn Go, and has since turned it into a full-time business.

The community of people who have adopted Go call themselves *gophers*, in honor of Go's lighthearted mascot (figure 1.1). Programming is challenging, but with Go and this book, we hope you discover the joy of coding.

Figure 1.1 Go gopher mascot designed by Renée French

In this lesson, you'll experiment with a Go program in your web browser.

Consider this If you tell a digital assistant, "Call me a cab," does it dial a taxi company? Or does it assume you changed your name to *a cab*? Natural languages like English are full of ambiguity.

Clarity is paramount in programming languages. If the language's grammar or syntax allows for ambiguity, the computer may not do what you say. That rather defeats the point of writing a program.

Go isn't a perfect language, but it strives for clarity more so than any other language we've used. As you go through this lesson, there will be some abbreviations to learn and jargon to overcome. Not everything will be clear at first glance, but take the time to appreciate how Go works to reduce ambiguity.

1.1 What is Go?

Go is a *compiled* programming language. Before you run a program, Go uses a *compiler* to translate your code into the 1s and 0s that machines speak. It compiles all your code into a single *executable* for you to run or distribute. During this process, the Go compiler can catch typos and mistakes.

Not all programming languages employ this approach. Python, Ruby, and several other popular languages use an *interpreter* to translate one statement at a time as a program is running. That means bugs may be lurking down paths you haven't tested.

On the other hand, interpreters make the process of writing code fast and interactive, with languages that are dynamic, carefree, and fun. Compiled languages have a reputation for being static, inflexible robots that programmers are forced to appease, and compilers are derided for being slow. But does it need to be this way?

> *We wanted a language with the safety and performance of statically compiled languages such as C++ and Java, but the lightness and fun of dynamically typed interpreted languages such as Python.*
>
> —Rob Pike, *Geek of the Week*
> (see mng.bz/jr8y)

Go is crafted with a great deal of consideration for the *experience* of writing software. Large programs compile in seconds with a single command. The language omits features that lead to ambiguity, encouraging code that is predictable and easily understood. And Go provides a lightweight alternative to the rigid structure imposed by classical languages like Java.

> *Java omits many rarely used, poorly understood, confusing features of C++ that in our experience bring more grief than benefit.*
>
> —James Gosling, *Java: an Overview*

Each new programming language refines ideas of the past. In Go, using memory efficiently is easier and less error-prone than earlier languages, and Go takes advantage of every core on multicore machines. Success stories often cite improved efficiency as a reason for switching to Go. Iron.io was able to replace 30 servers running Ruby with 2 servers using Go (see mng.bz/Wevx and mng.bz/8yo2). Bitly has "seen consistent, measurable performance gains" when rewriting Python apps in Go, and subsequently replaced its C apps with a Go successor (see mng.bz/EnYl).

Go provides the enjoyment and ease of interpreted languages, with a step up in efficiency and reliability. As a small language, with only a few simple concepts, Go is relatively quick to learn. These three tenets form the motto for Go:

Go is an open source programming language that enables the production of **simple**, **efficient**, *and* **reliable** *software at scale.*

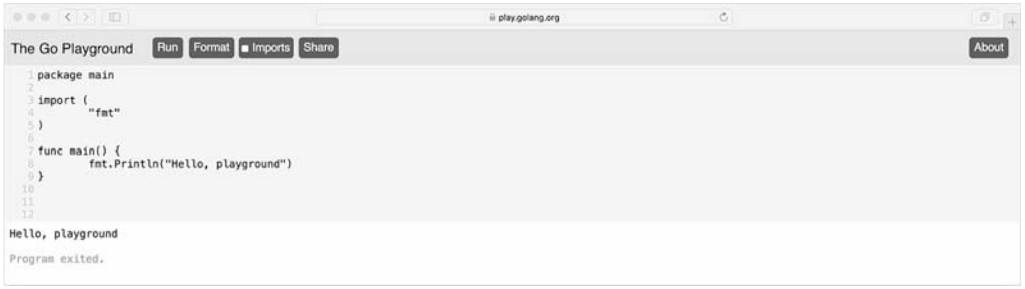
—Go Brand Book

TIP When searching the internet for topics related to Go, use the keyword golang, which stands for Go language. The -lang suffix can be applied to other programming languages as well: Ruby, Rust, and so on.

Quick check 1.1 What are two benefits of the Go compiler?

1.2 The Go Playground

The quickest way to get started with Go is to navigate to play.golang.org. At the Go Playground (figure 1.2) you can edit, run, and experiment with Go without needing to install anything. When you click the Run button, the playground will compile and execute your code on Google servers and display the result.



```
package main

import (
	"fmt"
)

func main() {
	fmt.Println("Hello, playground")
}
```

```
Hello, playground

Program exited.
```

Figure 1.2 The Go Playground

If you click the Share button, you'll receive a link to come back to the code you wrote. You can share the link with friends or bookmark it to save your work.

QC 1.1 answer Large programs compile in seconds, and the Go compiler can catch typos and mistakes before running a program.

NOTE You can use the Go Playground for every code listing and exercise in this book. Or, if you're already familiar with a text editor and the command line, you can download and install Go on your computer from golang.org/dl/.

Quick check 1.2 What does the Run button do in The Go Playground?

1.3 Packages and functions

When you visit the Go Playground, you'll see the following code, which is as good a starting point as any.

Listing 1.1 Hello, playground: playground.go

```
package main                      ← Declares the package
                                    this code belongs to
import (
    "fmt"                         ← Makes the fmt (format)
)                                   package available for use

func main() {                     ← Declares a function named main
    fmt.Println("Hello, playground")   ← Prints Hello, playground
}                                        to the screen
```

Though short, the preceding listing introduces three keywords: `package`, `import`, and `func`. Each keyword is reserved for a special purpose.

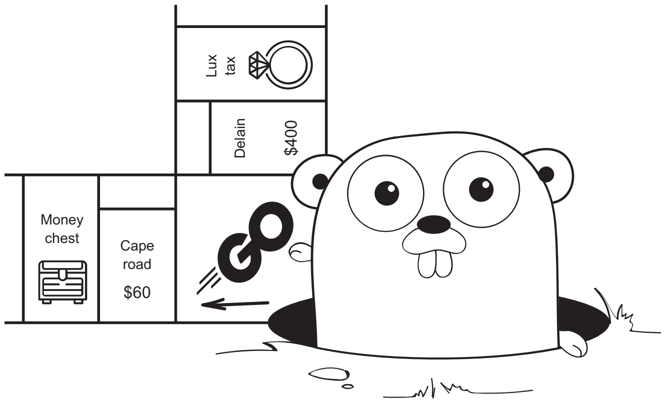
The `package` keyword declares the package this code belongs to, in this case a package named `main`. All code in Go is organized into *packages*. Go provides a standard library comprised of packages for math, compression, cryptography, manipulating images, and more. Each package corresponds to a single idea.

The next line uses the `import` keyword to specify packages this code will use. Packages contain any number of *functions*. For example, the `math` package provides functions like `Sin`, `Cos`, `Tan`, and `Sqrt` (square root). The `fmt` package used here provides functions for *formatted* input and output. Displaying text to the screen is a frequent operation, so this package name is abbreviated `fmt`. Gophers pronounce `fmt` as "FŌŌMT!," as though it were written in the large explosive letters of a comic book.

QC 1.2 answer The Run button will compile and then execute your code on Google servers.

The `func` keyword declares a function, in this case a function named `main`. The *body* of each function is enclosed in curly braces `{}`, which is how Go knows where each function begins and ends.

The `main` *identifier* is special. When you run a program written in Go, execution begins at the `main` function in the `main` package. Without `main`, the Go compiler will report an error, because it doesn't know where the program should start.



To print a *line* of text, you can use the `Println` function (`ln` is an abbreviation for line). `Println` is prefixed with `fmt` followed by a dot because it is provided by the `fmt` package. Every time you use a function from an imported package, the function is prefixed with the package name and a dot. When you read code written in Go, the package each function came from is immediately clear.

Run the program in the Go Playground to see the text *Hello, playground*. The text enclosed in quotes is echoed to the screen. In English, a missing comma can change the meaning of a sentence. Punctuation is important in programming languages too. Go relies on quotes, parentheses, and braces to understand the code you write.

Quick check 1.3

1 Where does a Go program start?
2 What does the `fmt` package provide?

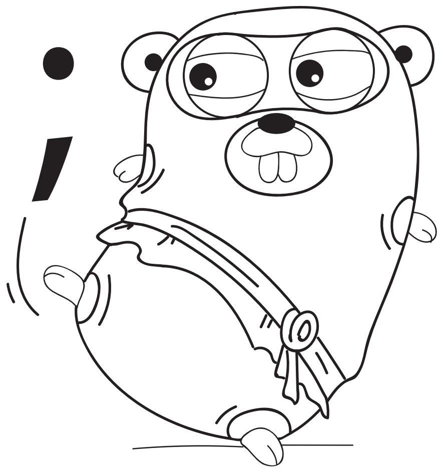
QC 1.3 answer

1 A program starts at the `main` function in the `main` package.
2 The `fmt` package provides functions for formatted input and output.

1.4 The one true brace style

Go is picky about the placement of curly braces {}. In listing 1.1, the opening brace { is on the same line as the `func` keyword, whereas the closing brace } is on its own line. This is the *one true brace style*—there is no other way. See mng.bz/NdE2.

To understand why Go became so strict, you need to travel back in time to the birth of Go. In those early days, code was littered with semicolons. Everywhere. There was no escaping them; semicolons followed every single statement like a lost puppy. For example:



```
fmt.Println("Hello, fire hydrant");
```

In December of 2009, a group of ninja gophers expelled semicolons from the language. Well, not exactly. Actually, the Go compiler inserts those adorable semicolons on your behalf, and it works perfectly. Yes, perfectly, but in exchange you must follow the *one true brace style*.

If you put an opening brace on a separate line from the `func` keyword, the Go compiler will report a syntax error:

```
func main()
{
}
```

missing function body

syntax error:unexpected semicolon or newline before {

The compiler isn't upset with you. A semicolon was inserted in the wrong place and it got a little confused.

TIP As you work through this book, it's a good idea to type the code listings yourself. You may see a syntax error if you mistype something, and that's okay. Being able to read, understand, and correct errors is an important skill, and perseverance is a valuable trait.

Quick check 1.4 Where must opening braces { be placed to avoid syntax errors?

Summary

- With the Go Playground you can start using Go without installing anything.
- Every Go program is made up of functions contained in packages.
- To print text on the screen, use the `fmt` package provided by the standard library.
- Punctuation is just as important in programming languages as it is in natural languages.
- You used 3 of the 25 Go keywords: `package`, `import`, and `func`.

Let's see if you got this...

For the following exercise, modify the code in the Go Playground and click the Run button to see the result. If you get stuck, refresh your web browser to get back the original code.

Experiment: playground.go

- Change the text printed to the screen by modifying what appears between quotes. Have the computer greet you by name.
- Display two lines of text by writing a second line of code within the body `{}` of the `main` function. For example:

  ```
  fmt.Println("Hello, world")
  fmt.Println("Hello, 世界")
  ```

- Go supports characters of every language. Print text in Chinese, Japanese, Russian, or Spanish. If you don't speak those languages, you can use Google Translate (translate.google.com) and copy/paste text into the Go Playground.

Use the Share button to get a link to your program and share it with other readers by posting it on the *Get Programming with Go* forums (forums.manning.com/forums/get-programming-with-go).

Compare your solution to the code listing in the appendix.

QC 1.4 answer An opening brace must be on the same line as `func`, rather than on an separate line. This is the one true brace style.

SOFTWARE DEVELOPMENT

GET PROGRAMMING WITH GO

Nathan Youngman | Roger Peppé

Go is a small programming language designed by Google to tackle big problems. Large projects mean large teams with people of varying levels of experience. Go offers a small, yet capable, language that can be understood and used by anyone, no matter their experience.

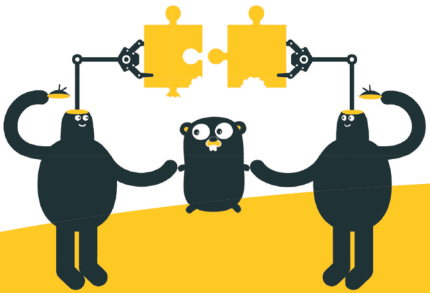
Hobbyists, newcomers, and professionals alike can benefit from a fast, modern language; all you need is the right resource! *Get Programming with Go* provides a hands-on introduction to Go language fundamentals, serving as a solid foundation for your future programming projects. You'll master Go syntax, work with types and functions, and explore bigger ideas like state and concurrency, with plenty of exercises to lock in what you learn.

WHAT'S INSIDE

- **Language concepts like slices, interfaces, pointers, and concurrency**
- **Seven capstone projects featuring spacefaring gophers, Mars rovers, ciphers, and simulations**
- **All examples run in the Go Playground — no installation required!**

This book is for anyone familiar with computer programming, as well as anyone with the desire to learn.

Nathan Youngman *organizes the Edmonton Go meetup and is a mentor with Canada Learning Code.* ***Roger Peppé*** *contributes to Go and runs the Newcastle upon Tyne Go meetup.*



To download their free eBook *in PDF, ePub, and Kindle formats, owners of this book should visit* manning.com/books/get-programming-with-go

MANNING

US $34.99 | Can $46.99 [Including eBOOK]

FREE EBOOK
See first page

"Perfectly organized for learning Go quickly; especially useful for inexperienced programmers."
—MARIO CARRION
MEREDITH CORPORATION

"Learn by doing! Plenty of examples will help you learn the core of the language and expose you to common Go idioms."
—ULISES FLYNN, NAV

"A great book about Go. Written for beginners but useful for seasoned developers too."
—MIKAËL DAUTREY, ISITIX

"The first rung on successfully climbing the Go ladder."
—JEFF SMITH, AGILIFY

ISBN-13: 978-1-617-29309-2
ISBN-10: 1-617-29309-1
53499
9 781617 293092