Preface

In the preface of the first edition, I wrote:

> My first choice for a book title was *Boring Go* because, properly written, Go is boring….
>
> Boring does not mean trivial. Using Go correctly requires an understanding of how its features are intended to fit together. While you can write Go code that looks like Java or Python, you're going to be unhappy with the result and wonder what all the fuss is about. That's where this book comes in. It walks through the features of Go, explaining how to best use them to write idiomatic code that can grow.

Go remains a small language with a small feature set. It still lacks inheritance, aspect-oriented programming, function overloading, operator overloading, pattern matching, named parameters, exceptions, and many additional features that complicate other languages. So why does a book on a boring language need an update?

There are a few reasons for this edition. First, just as boring doesn't mean *trivial*, it also does not mean *unchanging*. Over the past three years, new features, tools, and libraries have arrived. Improvements like structured logging, fuzzing, workspaces, and vulnerability checking help Go developers create reliable, lasting, maintainable code. Now that Go developers have several years of experience with generics, the standard library has started to include type constraints and generic functions to reduce repetitive code. Even the unsafe package has been updated to make it a little safer. Go developers need a resource that explains how to best use these new features.

Secondly, some aspects of Go weren't done justice by the first edition. The introductory chapter didn't flow as well as I'd like. The rich Go tool ecosystem wasn't explored. And first-edition readers asked for exercises and additional sample code. This edition attempts to address those limitations.

Finally, the Go team has introduced something new, and, dare I say, *exciting*. There's now a strategy that allows Go to keep the backward compatibility required for long-term software engineering projects while providing the ability to introduce

backward-breaking changes to address long-standing design flaws. The new `for` loop variable scoping rules introduced in Go 1.22 are the first feature to take advantage of this approach.

Go is still boring, it's still fantastic, and it's better than ever. I hope you enjoy this second edition.

Who Should Read This Book

This book is targeted at developers who are looking to pick up a second (or fifth) language. The focus is on people who are new to Go. This ranges from those who don't know anything about Go other than it has a cute mascot, to those who have already worked through a Go tutorial or even written some Go code. The focus for *Learning Go* isn't just how to write programs in Go; it's how to write Go *idiomatically*. More experienced Go developers can find advice on how to best use the newer features of the language. The most important thing is that the reader wants to learn how to write Go code that looks like Go.

Experience is assumed with the tools of the developer trade, such as version control (preferably Git) and IDEs. Readers should be familiar with basic computer science concepts like concurrency and abstraction, as the book explains how they work in Go. Some of the code examples are downloadable from GitHub, and dozens more can be tried out online on The Go Playground. While an internet connection isn't required, it is helpful when reviewing executable examples. Since Go is often used to build and call HTTP servers, some examples assume familiarity with basic HTTP concepts.

While most of Go's features are found in other languages, Go makes different trade-offs, so programs written in it have a different structure. *Learning Go* starts by looking at how to set up a Go development environment, and then covers variables, types, control structures, and functions. If you are tempted to skip over this material, resist the urge and take a look. It is often the details that make your Go code idiomatic. Some of what seems obvious at first glance might actually be subtly surprising when you think about it in depth.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic
: Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`
: Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

`Constant width bold`
: Shows commands or other text that should be typed literally by the user.

`Constant width italic`
: Shows text that should be replaced with user-supplied values or by values determined by context.

This element signifies a tip or suggestion.

This element signifies a general note.

This element indicates a warning or caution.

Using Code Examples

Supplemental material (code examples, exercises, etc.) is available for download at *https://github.com/learning-go-book-2e*.

If you have a technical question or a problem using the code examples, please send email to *bookquestions@oreilly.com*.

This book is here to help you get your job done. In general, if example code is offered with this book, you may use it in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant

amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Learning Go* by Jon Bodner (O'Reilly). Copyright 2024 Jon Bodner, 978-1-098-13929-2."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

O'Reilly Online Learning

For more than 40 years, *O'Reilly Media* has provided technology and business training, knowledge, and insight to help companies succeed.

Our unique network of experts and innovators share their knowledge and expertise through books, articles, and our online learning platform. O'Reilly's online learning platform gives you on-demand access to live training courses, in-depth learning paths, interactive coding environments, and a vast collection of text and video from O'Reilly and 200+ other publishers. For more information, visit *http://oreilly.com*.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-889-8969 (in the United States or Canada)
707-827-7019 (international or local)
707-829-0104 (fax)
support@oreilly.com
https://www.oreilly.com/about/contact.html

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at *https://oreil.ly/learning-go-2ed*.

For news and information about our books and courses, visit *https://oreilly.com*.

Find us on LinkedIn: *https://linkedin.com/company/oreilly-media*

Follow us on Twitter: *https://twitter.com/oreillymedia*

Watch us on YouTube: *https://youtube.com/oreillymedia*

Acknowledgments for the 2nd Edition

I've been humbled and flattered by the response to the first edition of *Learning Go*. It was started in late 2019, completed in late 2020, and released in early 2021. It was, unintentionally, my own pandemic project. Some people baked bread; I explained pointers.

After I finished, people asked me what my next book would be. I told them that I was going to take a well-deserved break and then get to work on a romance novel. But before I could start writing about pirates and princesses, something amazing happened: *Learning Go* was a huge success. It spent the better part of a year as one of the five most popular books on O'Reilly Learning. As sales continued, I saw things in the first edition that I wanted to fix and received letters from readers pointing out errors and omissions. Some were fixed as errata in later printings, but I contacted O'Reilly to see if they would like a new edition, and they were interested.

Returning to *Learning Go* and O'Reilly for the second edition was a pleasant experience. Rita Fernando provided guidance, feedback, and editing. This edition has been made immeasurably better by feedback from Jonathan Amsterdam, Leam Hall, Katie Hockman, Thomas Hunter, Max Horstmann, and Natalie Pistunovich. Chris Hines did an incredibly thorough job finding my mistakes and suggesting better examples. The comments and feedback from Abby Deng and the members of the Go Book Club at Datadog allowed me to makes changes based on feedback from developers new to Go. The (hopefully few) remaining errors are all mine.

My wife and children were kind enough to not mind me skipping out on family movie nights while I figured out the best way to describe new Go features.

I also want to thank the readers of the first edition who reached out to me with their kind words. I thank you all for your support and encouragement.

Acknowledgments for the 1st Edition

Writing a book seems like a solitary task, but it doesn't happen without the help of a great number of people. I mentioned to Carmen Andoh that I wanted to write a book on Go and at GopherCon 2019, and she introduced me to Zan McQuade at O'Reilly. Zan guided me through the acquisition process and continued to provide me advice while I was writing *Learning Go*. Michele Cronin edited the text, gave feedback, and listened during the inevitable rough patches. Tonya Trybula's copyediting and Beth Kelly's production editing made my draft production-quality.

While writing, I received critical feedback (and encouragement) from many people including Jonathan Altman, Jonathan Amsterdam, Johnny Ray Austin, Chris Fauerbach, Chris Hines, Bill Kennedy, Tony Nelson, Phil Pearl, Liz Rice, Aaron Schlesinger, Chris Stout, Kapil Thangavelu, Claire Trivisonno, Volker Uhrig, Jeff Wendling, and

Kris Zaragoza. I'd especially like to recognize Rob Liebowitz, whose detailed notes and rapid responses made this book far better than it would have been without his efforts.

My family put up with me spending nights and weekends at the computer instead of with them. In particular, my wife, Laura, graciously pretended that I didn't wake her up when I'd come to bed at 1 A.M. or later.

Finally, I want to remember the two people who started me on this path four decades ago. The first is Paul Goldstein, the father of a childhood friend. In 1982, Paul showed us a Commodore PET, typed PRINT 2 + 2, and hit the Enter key. I was amazed when the screen said 4 and was instantly hooked. He later taught me how to program and even let me borrow the PET for a few weeks. Second, I'd like to thank my mother for encouraging my interest in programming and computers, despite having no idea what any of it was for. She bought me the BASIC programming cartridge for the Atari 2600, a VIC-20, and then a Commodore 64, along with the programming books that inspired me to want to write my own someday.

Thank you all for helping make this dream of mine come true.