Golang for Jobseekers

Unleash the power of Go programming for career advancement

Hairizuan Bin Noorazman

www.bpbonline.com

First published: 2023

Published by BPB Online
WeWork
119 Marylebone Road
London NW1 5PU

UK | UAE | INDIA | SINGAPORE

ISBN 978-93-55518-538

www.bpbonline.com

Dedicated to

My beloved Parents:

Noorazman Bin Bulat

Jamilah Bte Yusof

&

My wife Nurwidayu

About the Author

Hairizuan Bin Noorazman has a degree in Chemical Engineering but switched over to Information Technology when he entered the workforce. He has worked on various technologies over the years and transitioned between multiple roles before working as a Devops Engineer in Acronis. He has 7+ years of experience in various domains like Software Development, DevOps Automation tools, and Web Analytics tools.

In his spare time he writes blog posts and create videos on technical topics (e.g., usage of Golang/DevOps tools). He is also a Google Developer Expert (GDE) - a program by Google which recognizes individuals for contributing and sharing knowledge about Google technologies – in his case, it is mostly Google Cloud technologies.

About the Reviewers

- **Aakash Handa** is a Full Stack Solution Architect, the crossover between design and programming has always been of interest to him. He has been lucky enough to work alongside some talented teams on a number of high-profile websites. He has a wide range of skills that include back-end development using open source technologies (NodeJs, Python, Go), design (working closely with designers), front-end development (Angular2, ReactJs, HTML5, CSS3, Javascript, Responsive, UX), Server Administrator(AWS, IBM, Azure), database(MongoDB, cassendra, Mysql), load balancing (Varnish and Memcached).
- **Prithvipal Singh** is currently working as a Senior Software Engineer in Infoblox, where he is a core member of a DNS Firewall related project and working on projects developed as a cloud-native distributed system using Golang. He is the author of the Hands-on Go Programming book. He is an expert in distributed systems, cloud computing, and microservice architecture. He has been working in the IT industry for a decade and has vast experience working in Golang, Java, Docker, Kubernetes, Python, and Nodejs. He holds an MCA degree from Pune University.

Acknowledgements

There are a few people I want to thank for the continued and ongoing support they have given me during the writing of this book. First and foremost, I would like to thank my parents and my wife for continuously encouraging me to write the book — I could have never completed this book without their support.

I am grateful to my current company Acronis which provided me with the environment to learn and develop the intuition for developing Golang applications and understand the end-to-end flow of developing applications and getting them into production.

My gratitude also goes to the team at BPB Publication for being supportive during the book's writing. I took a long while to write this book (since this was written during my off-hours outside work). It is particularly helpful to have someone who helps monitor the progress of the book and constantly checks in on the status of the book on a frequent basis.

Preface

After Python, the next programming language that appeals to users, primarily in the domain of Web application and software tooling, is Google's Go language. It consistently ranks among the top 10 programming languages and aspires to be adopted by the larger developer community.

This book is designed for fresh graduates or individuals with previous programming experience using C, Python, or Java. This book teaches the fundamental elements of Go by practicing and writing programs to get strong understanding of the elements. You will learn data types, protocols, data structures, and algorithms.

The book will then introduce the reader to various aspects of the software engineering world, such as understanding REST APIs (one of the more common ways to structure web applications), deployment processes, and engineering practices of monitoring and logging applications.

Chapter 1: Understanding Golang and its Potential – This chapter sets the stage and provides the motivation for the reader why to choose Golang to build their next app or why companies are taking up Golang nowadays

Chapter 2: Golang Fundamentals – This chapter will cover a quick overview of how to get started with Golang. It is assumed that the reader already has experience with another programming language. Hence, a lot of things mentioned in this chapter would utilize common programming terms such as variables, types, functions, structs, and interface

Chapter 3: Exploring Data Structures – This chapter will cover some of the Data Structures that are sometimes asked and tested during interview sessions for software engineering roles. Some of the concepts covered here are generally covered within the school syllabus/bootcamp syllabus, so it will mostly serve as a reminder of what some of these data structures are and how they can potentially be used (if any)

Chapter 4: Understanding Algorithms – This chapter will cover some of the potential algorithm questions that could be covered in interview questions for a software engineer role. The chapter will cover some common ways to solve the algorithms using the Golang programming language as well as the time/space

complexity that is related to the mentioned algorithm. It might not cover most cases but it should cover some of the more common concepts

Chapter 5: Getting Comfortable with Go Proverbs – Unlike languages such as Java where consultants have outlined guidelines and design patterns that one could follow to design efficient and easy-to-understand code, Golang has a set of proverbs (thought up by community members) which would prove to be good guiding principles. Refer to the following webpage: **https://go-proverbs.github.io/**

Chapter 6: Building REST APIs – This chapter covers the basics of building web applications that follow REST API principles. Web applications are still one of the most common set of applications that are still being even now and would serve as a good basis before learning other possible protocols that pertain with web applications such as GRPC, thrift, and Graphql

Chapter 7: Testing in Golang – Inform the reader of the importance of writing tests for Golang applications as well as how to do it – particularly important for junior roles especially since in many companies, unit tests/integrated tests are somewhat expected features to write up.

Chapter 8: Deploying a Golang Application in a Virtual Machine – This chapter will cover how to get and run an application in a Virtual Machine. As much as applications are modernized by putting it into containers, not all applications should be containerized. Sometimes, due to security limitations or resource constraints, some of the applications are deployed in a Virtual machine. This chapter covers the various aspect of how to ensure a Golang application is copied over to the VM and then operated

Chapter 9: Deploying a Containerized Golang Application – This chapter focuses less on Golang language specifics but on operations aspect of running the application. The technological have been moving in line with several trends such as containerization and DevOps – where developers should be responsible for containerizing their application. This chapter would cover how to do so and some of the hiccups that can happen for this

Chapter 10: Microservices with Golang Applications – This is a chapter to cover microservices concept – which is a type of structure that many modern big companies would adopt. Generally, the microservices topic is pretty deep so this chapter will cover the topics on the "surface level", enough such that someone would be able to start working in situations where applications are deployed in microservices architectures

Chapter 11: Introduction to Monitoring and Logging – Building an application that provides just the required functionality is nowadays insufficient. We would want to understand and assure ourselves that our application is working fine and in a responsive manner. In order to get such information, we would need to add monitoring and logging to the application. This chapter would cover the details of it so that it would allow us to understand how the application is operating but from an external lens

Chapter 12: Adding Concurrency in Golang Application – One of the main draws of Golang applications is the capability to write up applications that can be concurrent in nature, however, it has not been the focus of this book. One main reason is because YAGNI; if you can avoid using it, you should – it introduces a whole bunch of complexity to the program, making it way harder to debug and understand. However, there are still certain scenarios that might be useful to have such capabilities

Chapter 13: What is Next? – This chapter will cover possible points for the reader to continue exploring in order to understand the Golang language and the technology industry. There are various topics that branch out and require in-depth study in order to understand how to write effective code for such domain spaces. Some of the topics covered such as GRPC, Application Profiling and Generics

Code Bundle and Coloured Images

Please follow the link to download the
Code Bundle and the ***Coloured Images*** of the book:

https://rebrand.ly/ck0bvsw

The code bundle for the book is also hosted on GitHub at **https://github.com/bpbpublications/Golang-for-Jobseekers**. In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at **https://github.com/bpbpublications**. Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Piracy

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at **business@bpbonline.com** with a link to the material.

If you are interested in becoming an author

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit **www.bpbonline.com**. We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit **www.bpbonline.com**.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com

Table of Contents

Chapter 1
Understanding Golang and its Potential

Introduction

Golang is one of the "newer" programming languages that appeared in the block, although it has been around for at least 10 years at this point. The language was birthed and designed in Google by several Google engineers, *Robert Griesemer*, *Rob Pike*, and *Ken Thomson*. Other engineers eventually joined the team, adding new functionality to the language and shaping the language into how it is today. The language was created in order to try to find a way to resolve several issues that were identified with codebases at that time in Google (codebases were written in C++ and Java programming languages). The languages mentioned previously were definitely not designed for the applications that were to be built—Web applications. One can write Web applications with those said languages, but naturally, there will be downsides to using those languages for building Web applications. Refer to the following video here for the full context: **https://www.youtube.com/watch?v=YXV7sa4oM4I**.

Structure

In this chapter, we will discuss the following topics:

- Characteristics of Golang Programming Language
    - Statically typed

    - Garbage collection
    - Cross compilation
    - "Batteries Included" standard library
    - Version guarantees
- What kind of development work is Golang used in?
    - Web applications
    - Command Line Interfaces (CLI)
- Examples of major applications that are built with Golang
    - Docker
    - Kubernetes
    - Hugo
- Overview of some of the companies that use Golang
    - Cloudflare
    - Monzo

Objectives

This chapter serves to provide an initial understanding of the potential of the Golang Programming Language. It will start off with a listing of potential desirable properties of the languages before leading readers to its potential use cases, such as Web applications and command line interface applications. There will also be a short section describing some of the major applications that have been built using the Golang language. The chapter will conclude by covering some examples of companies that have publicly announced their usage of Golang in some major aspects of their company.

Characteristics of Golang Programming Language

Naturally, before choosing to work with a language, you will definitely want to understand the various aspects of the programming language. Certain aspects of a programming language can make the language extremely difficult to use in particular use cases, which is particularly why it is important to understand the problem you are trying to solve before attempting to select the programming language you will want to try creating the code to solve the problem.

In this section, several important characteristics will be discussed—some of them are potential make or break for certain types of applications.

Statically typed

Programming languages generally lie between statically typed or dynamically typed languages. Statically typed languages are languages where one needs to define the type of the variables being used. Type-checking is a part of the language. This is the opposite of dynamically typed languages, where the type for the variable is being inferred or guessed by the language's runtime.

To understand this more easily, examples of other programming languages that you might have probably heard during your programming learning journey. Some examples of dynamically typed programming languages are JavaScript and Python programming language. While defining the variables within that language, you would immediately assign the value to the variable. We do not need to define the type that variable is in. An example of this is as follows:

```
>>> exampleVariable = 12
```

If you have a Python runtime on your computer, you can enter the following short code snippet. What is essentially happening is the variable "**exampleVariable**" is assigned the value **12**. It is also defined to be an integer type. If you try to attempt to combine the number with a piece of text to form a concatenated word, it will error out. An example of this is happening in Python.

```
>>> exampleVariable = 12

>>> exampleVariable + "a"

Traceback (most recent call last):

  File "<stdin>", line 1, in <module>

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Golang is a statically typed language, so you would define the types alongside the variable.

```
var exampleVariableInGolang int = 12
```

The following example assigns the value **12** to the variable "**exampleVariableInGolang**," which is defined as an Integer type. There is a slight code shortcut that is commonly used in most Golang Programming Languages when it comes to defining the variables, but this will definitely be covered in the upcoming chapter, which will go through the various Go lang fundamentals.

Now, before we proceed to the next characteristic, we will need to understand why this matters. Some developers like dynamically typed languages for the following reasons:

- Easier/more convenient to code; do not need to wrangle to ensure the code's correctness but focus on the logic, which might be more interesting than thinking about language's specific syntax requirements.
- Type checking is only done while the program is running; hence, no compile step is needed. Can skip compile type, which for some languages (especially some of the older ones) might take a while.

However, statically typed languages come with their own set of benefits; and if you are interested in Golang, this part might also resonate with you on why this might be a good design decision being made for the language.

- Less resources are spent by the runtime to interpret the type for the variables (less vague for types of variables).
- **Easier type checking**: Code linting tools/IDEs will find it easier to understand the types of variables being defined in the code and be able to suggest the appropriate helper functions.
- **Easier to read the codebase**: In a dynamically typed programming language, nothing stops a developer from using the same variable name over and over again. At the end of a large function, you will not be able to know what is the type in which the variable is. You can attempt to do type casting and so on, but it is still additional code—essentially, doing any type casting and type checking would seem as if you would want to have type checking functionality in the programming language that you are working with.

Garbage collection

Garbage collection is one of those terms that is constantly thrown around on the internet forums that cover programming tutorials. However, this is one of the terms that might prove vital understanding for developers, especially since the choice of the programming language being chosen can affect the development process of the applications that are being developed.

The term Garbage Collection when it pertains to computer science, refers to the strategy of how the programming language runtime manages its memory. In the early days of programming languages, languages were not built with this functionality in mind. An example of such programming languages is C and C++. These languages are still around and behind some of the very vital tools in the programmer's industry. For these mentioned programming languages, you, as the developer, will have to