Beginner Go Workshop

by @kinbiko
on 2025-06-21

» About this workshop

- ✊ Hands-on!
  - After the initial slides I expect you to get hands-on with your own computer.
  - You'll probably get stuck at some point. Don't worry, there'll be people around to help.
- 🤓 This workshop is an introduction to Go, but not to programming in general.
- 💨 Quickly cover Go syntax basics.
  - If you get stuck on this, you're encouraged to Google or LLM-it.
- 🏄 Elaborate on Go specific features.
  - If you get stuck on this, you're encouraged to interrupt and ask.

» Who this is for

- 🙋 Complete Go beginners or juniors.
  - I only assume you've installed Go on your computer. (Do so during the hands-off section if you haven't already)
- 🤓 New language, open-mindedness required.
  - You'll probably disagree with Go's design decisions, or think another language does something better. Aim to understand **why** the decision was made in your own time.
- 📈 Experienced Gophers might be bored during the workshop.
  - If you're an experienced gopher please help me help others during the hands-on periods.
  - Please hang around for the second half, it'll be much more interesting for you.

» Raise your hand / keep your hand raised if...

* You've never used Go.
* You've edited Go code slightly, but can't write from scratch.
* You've used the go command-line tool locally to quickly install something.
* You've written `if err != nil` more than 10 times.
* You're confident in how to structure backend applications in Go.
* You know how to do dependency injection in Go without 3rd party packages.
* You've used build tags in your projects.
* You've used build tags in your projects, and regretted it.
* You're confident in writing advanced concurrency patterns.
* You've made contributions to the Go language or ecosystem itself.

» Go Selling Points

* 💿 Super fast compiler. Feels interpreted.
* 😌 Simple. Read language spec.
* 1️⃣ Single binary = simple deployment.
* 📚 Excellent std lib.
* 🧰 Excellent built-in tooling under the go CLI.
* 💨 Fast.
* 🤖 Great for AI(?).

» Compared to Python

* Go is much faster.
* Go is much better for concurrency like web-servers.
* Go has much more boilerplate.
* You don't use Go for machine learning, data science, etc.

» Compared to JavaScript

* Go is server only, and faster.
* Go has static typing.
* Concurrency models very different.
* Go compiles to a single binary.

» Compared to Rust

* Rust is faster in general.
* Go is easier to learn and faster to write.
* Go is garbage collected.

» Who uses Go?

* Backend microservices.
* Cloud infrastructure.
* Developer/system tooling.

Examples

Docker, Kubernetes, Terraform, Prometheus, Google, Uber, Monzo, Mercari, etc.

» You're about to join the Go community

* We're pretty friendly. Like the Ruby community.
* If you write Go you're known as a Gopher.
* "Go" is the name of the language.
    * "golang" should only be used when search queries are ambiguous.

» Let's get hands-on

gobyexample.com

» gobyexample.com

* Follow the exercises from top to bottom.
* Aim to reach "Closing channels" inclusive.
* You may **skip**: (in the interest of time)
    * Closures
    * Recursion
    * Strings and Runes
    * Struct Embedding
    * Generics
    * Non-Blocking Channel Operations
* We'll do the first together, then in groups.
* Ask an expert if you're confused.
* Common points of interest discussed together.

» gobyexample.com (if more time)

* Continue the exercises from top to bottom.
* Aim to reach the bottom.
* You may **skip**: (in the interest of time)
  * Timers
  * Rate Limiting
  * Atomic Counters
  * Stateful Goroutines
  * Sorting
  * Sorting by Functions
  * Text Templates
  * Regular Expressions
  * XML
  * Epoch
  * Random Numbers
  * URL Parsing
  * SHA256 Hashes
  * Base64 Encoding
  * Reading Files
  * Writing Files
  * Line Filters
  * Directories
  * Temporary Files and Directories
  * Spawning Processes
  * Exec'ing Processes