PROGRAMMER'S ROADMAP

Hi!

If you're an experienced developer, you may want to watch only the following lectures. If you follow this roadmap, you will need to watch about 50 lectures instead of 180 lectures.

This course starts from the most basics than advances toward the end, step by step. So, the complexity of the topics increases on each level. I've intentionally designed it so to make it easy for everyone.

If you think some of the topics are easy for you, then watch the recap lectures, take the quizzes and exercises, and even skip the lectures in that section altogether, you can always come back to them later.

Enjoy!

LECTURES

- **Write Your First Go Program**
    - Please watch all the lectures.
- **Master the Type System of Go**
    - What is Go Doc?
    - The lectures under "Write a Library Package".
    - Every Go type has a zero value
    - What is a blank identifier?
    - Let's declare a couple of variables!
    - What is type inference?
    - How to short declare multiple variables?
    - Why can't you short declare a variable in the package-level?
    - What is redeclaration?
    - When to use a short declaration?
    - Get input from command-line and learn about slices
    - Learn the basics of os.Args
    - Greet people using os.Args
    - The lectures under "Print Formatted Output Using Printf".
    - Convert Celsius to Fahrenheit
    - Convert Feet to Meters
    - What is a Raw String Literal?
    - How to get the length of a string?
    - The lectures after "What is a Predeclared Type?" and to the end of the section.
    - The following lectures under "Understand Untyped Constants"
        - Learn the rules of constants
        - Recap: Constants
        - How untyped constants work under the hood?
        - What is a Default Type?
        - Example: time.Duration

        - What is iota?
        - Naming Things: Recommendations

- **Control Flow and Error Handling**
    - Watch all the lectures under "Pass Me: Create a Password-Protected Program"
    - Watch all the lectures under "Understand Go's Error Handling"
    - Use multiple values in case conditions
    - How does the fallthrough statement work?
    - Solution: Parts of a Day
    - Recap: Switch Statement
    - How to continue a loop? (+BONUS: Debugging)
    - Create a multiplication table
    - How to loop over a slice?
    - For Range: Learn the easy way!

- **Projects: For Beginners**
    - Please watch all the lectures.

- **Remaining Sections**
    - You may watch all the remaining lectures from this point. They are intermediate to advanced level lectures.

That's all! Enjoy! 🤩

BONUS: Why should you learn Go?

In summary: Go is easy as Python and Javascript , and it's as fast as C/C++. It's more enjoyable to work with Go than C/C++. You can go low-level, or you can stay high-level.

What Go is used for?

Go is used mostly by web companies: Google, Facebook, Twitter, Uber, Apple, Dropbox, Soundcloud, Medium, Mozilla Firefox, Github, Docker, Kubernetes, and Heroku.

Go is best for: Cross-Platform Command-line Tools, Distributed Network Applications, Cloud technologies like Microservices and Serverless, Web APIs, Database Engines, Big-Data Processing Pipelines, Embedded Development, and so on.

Go is not best for (but doable): Desktop Apps, Writing Operating Systems, Kernel Drivers, Game Development, etc.

Who Designed Go?

Go designed by one of the most influential people in the industry:

- Unix: Ken Thompson

- UTF-8, Plan 9: Rob Pike
- Hotspot JVM (Java Virtual Machine): Robert Griesemer

HOW MUCH CAN YOU EARN?

* Go Salaries

From Eight years of Go post:

> Today, **every single cloud company has critical components of their cloud infrastructure implemented in Go** including Google Cloud, AWS, Microsoft Azure, Heroku, and many others. Go is a key part of cloud companies like Alibaba, Cloudflare, and Dropbox. Go is a critical part of open infrastructure including Kubernetes, Cloud Foundry, Openshift, NATS, Docker, Istio, Etcd, Consul, Juju, and many more. Companies are increasingly choosing Go, to build cloud infrastructure solutions.

What Can You Accomplish with Go?

- A network Driver written in Go (*only 10% penalty compared to C driver*)
- Google gVisor (*Userspace kernel written in Go*)
- Multi-platform Nintendo emulator
- Docker: Container system
- Kubernetes: Container scheduling and management
- VM image deduplication utility
- Chat server
- RUM beacon collector
- Time-series database engine, a client for it, command-line tools, etc.
- Map-reduce library
- Clustered front-end-optimizing reverse proxy with on the fly content rewriting, image resizing, caching, Lua event handler execution (all multi-tenant)
- Geographically distributed reverse proxy CDN nodes
- Health management daemon with event handlers and peer to peer reporting
- Pure Go DNS server
- API backend that interfaces with MySQL
- Linux process capture/restore utility
- Reverse Proxy to mask our asset server from clients.
- HTML -> PDF converter for invoice generation.
- URL shortener like tinyurl.com and goo.gl
- SMS messaging service.
- Credit Card payment gateway
- JSON Web Token package
- On the fly image processing services
- 3d render farm/content production pipeline (pretty large project)
- Production lxc container deployment
- Automated testing management

Reference: This Reddit post.

CHECK OUT FOR MORE INFORMATION:

- About Go: An Overview
- Why should you learn Go?
- Emerging language of cloud Infrastructure
- Companies using Go
- Eight years of Go
- Twitter: Handling Five Billion Session in a Day with Go
- A C++ developer looks at Go