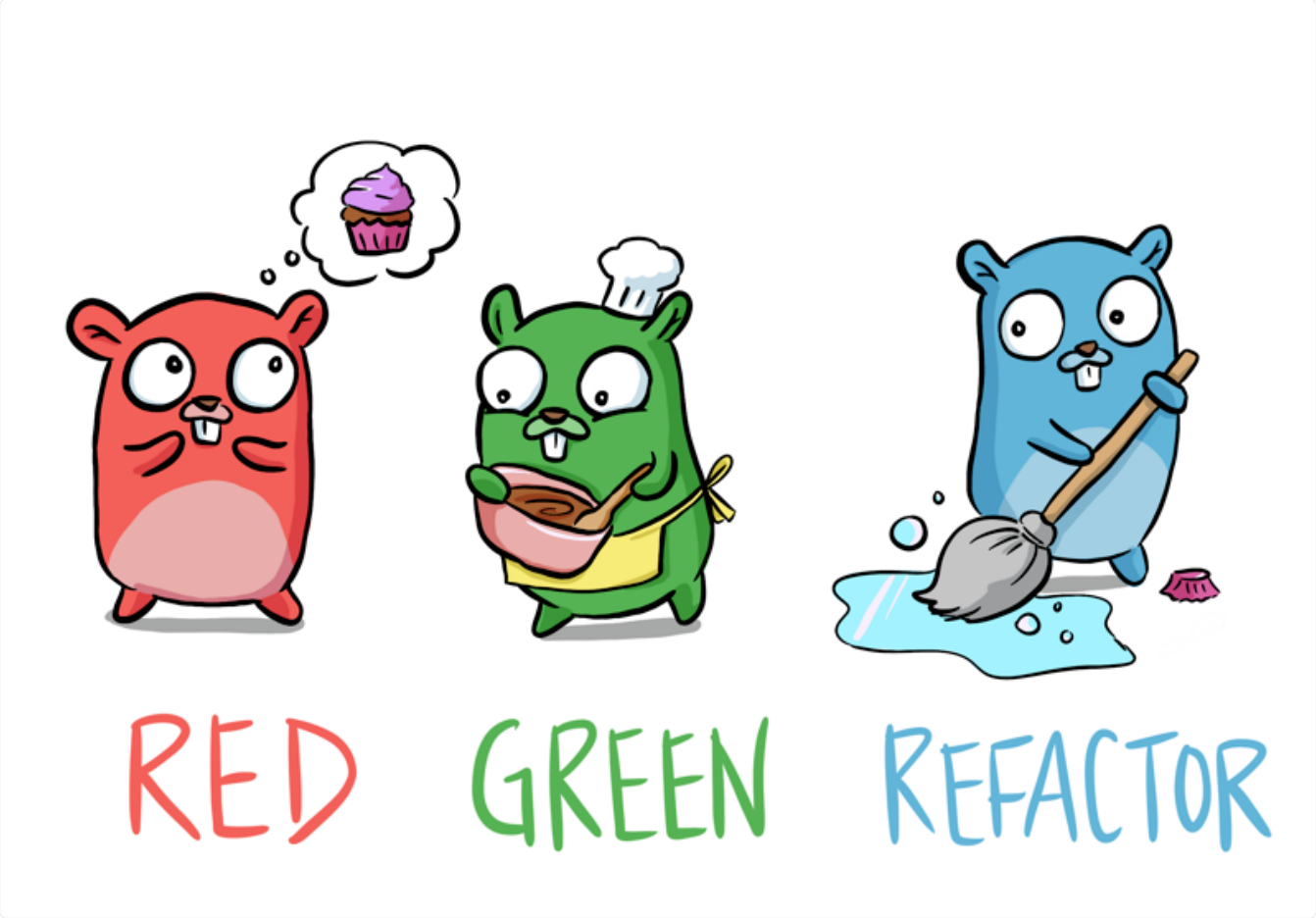
Learn Go with Tests



Art by Denise

Support me

I am proud to offer this resource for free, but if you wish to give some appreciation

- Tweet me @quii
- Mastodon
- Buy me a coffee
- Sponsor me on GitHub

Learn test-driven development with Go

> Explore the Go language by writing tests

> **Get a grounding with TDD**. Go is a good language for learning TDD because it is a simple language to learn and testing is built-in

> Be confident that you'll be able to start writing robust, well-tested systems in Go

Translations:

> 中文

> Português

> 日本語

> 한국어

> Türkçe

Background

I have some experience introducing Go to development teams and have tried different approaches as to how to grow a team from some people curious about Go into highly effective writers of Go systems.

What didn't work

Read *the* book

An approach we tried was to take the blue book and every week discuss the next chapter along with the exercises.

I love this book but it requires a high level of commitment. The book is very detailed in explaining concepts, which is obviously great but it means that the progress is slow and steady - this is not for everyone.

I found that whilst a small number of people would read chapter X and do the exercises, many people didn't.

Solve some problems

Katas are fun but they are usually limited in their scope for learning a language; you're unlikely to use goroutines to solve a kata.

Another problem is when you have varying levels of enthusiasm. Some people just learn way more of the language than others and when demonstrating what they have done end up confusing people with features the others are not familiar with.

This ends up making the learning feel quite *unstructured* and *ad hoc*.

What did work

By far the most effective way was by slowly introducing the fundamentals of the language by reading through go by example, exploring them with examples and discussing them as a group. This was a more interactive approach than "read chapter x for homework".

Over time the team gained a solid foundation of the *grammar* of the language so we could then start to build systems.

This to me seems analogous to practicing scales when trying to learn guitar.

It doesn't matter how artistic you think you are, you are unlikely to write good music without understanding the fundamentals and practicing the mechanics.

What works for me

When *I* learn a new programming language I usually start by messing around in a REPL but eventually, I need more structure.

What I like to do is explore concepts and then solidify the ideas with tests. Tests verify the code I write is correct and documents the feature I have learned.

Taking my experience of learning with a group and my own personal way I am going to try and create something that hopefully proves useful to other teams. Learning the fundamentals by writing small tests so that you can then take your existing software design skills and ship some great systems.

Who this is for

- People who are interested in picking up Go
- People who already know some Go, but want to explore testing more

What you'll need

- A computer!
- Installed Go
- A text editor
- Some experience with programming. Understanding of concepts like `if`, variables, functions etc.
- Comfortable using the terminal

Feedback

- Add issues/submit PRs here or tweet me @quii

Next
Install Go

Last updated 6 months ago