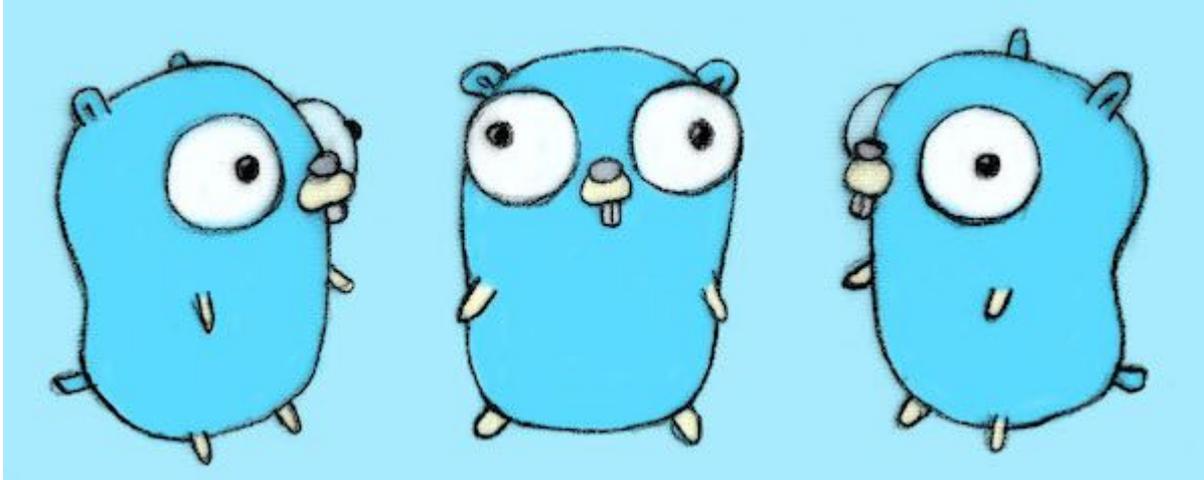


# Programmieren mit Go - Eine Einführung



# Ablauf

Ein paar Slides

Go By Examples (Code lesen)

Ein paar Slides

Beispiele selber umsetzen

# Thomas Güttler

Geb: 1976

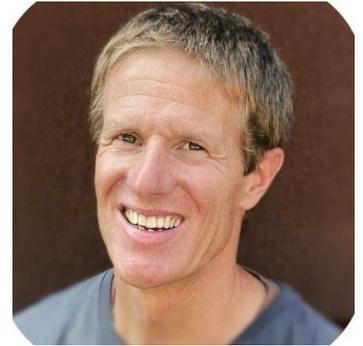
BASIC/Pascal: ..1995

Studium: 1996-2001: C, C++, Prolog, Java, Visual Basic

Beruflich: Ab 2001 Python, Bash und etwas C.

Bis 2023: Python, Django, HTML/CSS/(JS), Linux-Server.

Ab 2023: Kubernetes & Go ([Syself](#), [Cluster API Provider Hetzner](#))



## Woher?

Bisherige Programmiererfahrung?

Welcher Bereich? (Web, Embedded, DevOps, ... )

Fachgebiet? (Banken, Elektronik, ...)

## Wohin / Ziele?

Warum wollt ihr Go lernen?

# Go is noch jung

1972 C

1974 SQL

1985 C++

1991 Python

1995 Java, JavaScript, PHP

2009 Go, Go 1.0 in 2012

Feb 2024	Feb 2023	Change	Programming Language		Ratings	Change
1	1			Python	15.16%	-0.32%
2	2			C	10.97%	-4.41%
3	3			C++	10.53%	-3.40%
4	4			Java	8.88%	-4.33%
5	5			C#	7.53%	+1.15%
6	7	▲		JavaScript	3.17%	+0.64%
7	8	▲		SQL	1.82%	-0.30%
8	11	▲		Go	1.73%	+0.61%
9	6	▼		Visual Basic	1.52%	-2.62%
10	10			PHP	1.51%	+0.21%

Tiobe Index

Kompiliert schnell

Nebenläufigkeit (Goroutines, Channels)

automatische Speicherbereinigung (Garbage Collection)

Typisierung: stark, statisch (Autocomplete everywhere)

Pointer

Wenige Schlüsselwörter (“einfach”)

Generics

Statisches Binary, leichtes cross-compiling

BSD-Lizenz

Mehrere professionelle Vollzeitstellen für die Weiterentwicklung

# Go ist langsam im Vergleich zu

C

C++

Rust

Zig

.... Speichersicherheit und Garbage Collection haben ihren Preis.

Sobald Daten über das Netzwerk gehen ist das der Flaschenhals und nicht die Programmiersprache.

Was ist dir wichtig?

**“Mit der Aufgabe fertig werden”** vs **“100% Performance”** ?

# Go ist **\*\*langweilig\*\***

Kompatibilitätsversprechen: Programme, die heute funktionieren werden auch in Zukunft funktionieren.

*“No is temporary, yes is forever.”*

Ein kostspieliges Update wie von Python2 zu Python3 wird es nicht geben.

Es wird auch kein Go 2 geben.

Go is langweilig. Und das ist gut so.

# Das Drumherum

Pakete: ``go get github.com/...``

Tests: ``go test ...``

Language-Server für alle IDEs.

gofmt: Einheitliche Formatierung des Quelltextes.

# Go by Example: Hello World

```
package main    
  
import "fmt"  
  
func main() {  
    fmt.Println("hello world")  
}
```

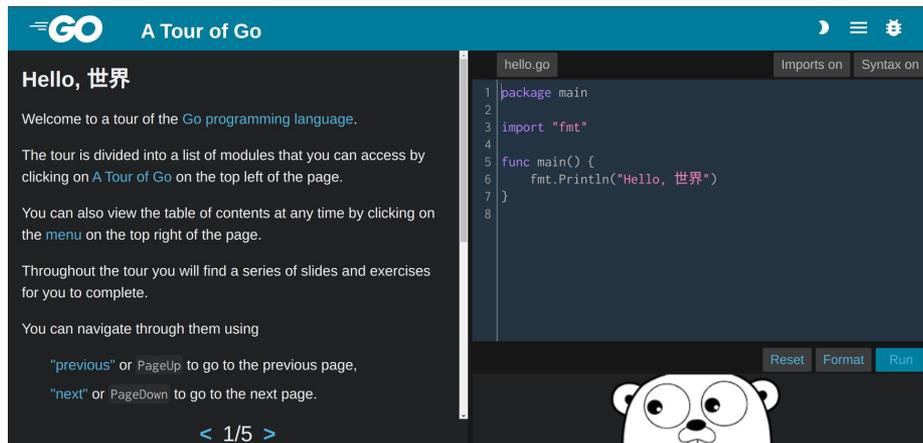
```
$ go run hello-world.go  
hello world  
  
$ go build hello-world.go  
$ ls  
hello-world  hello-world.go  
  
$ ./hello-world  
hello world
```

Next example: [Values](#).

<https://thomas-guettler.de/go/>

# Ich will mehr Beispiele!

## A Tour of Go



The screenshot shows the 'A Tour of Go' website. The top navigation bar includes the Go logo, the title 'A Tour of Go', and icons for a moon, a menu, and a bug. The main content area is titled 'Hello, 世界' and contains the following text:

Welcome to a tour of the [Go programming language](#).

The tour is divided into a list of modules that you can access by clicking on [A Tour of Go](#) on the top left of the page.

You can also view the table of contents at any time by clicking on the [menu](#) on the top right of the page.

Throughout the tour you will find a series of slides and exercises for you to complete.

You can navigate through them using

- "previous" or `PageUp` to go to the previous page.
- "next" or `PageDown` to go to the next page.

At the bottom of the slide, there is a navigation indicator: `< 1/5 >`.

On the right side, there is a code editor window titled 'hello.go' with the following code:

```
1 package main
2
3 import "fmt"
4
5 func main() {
6     fmt.Println("Hello, 世界")
7 }
8
```

Below the code editor are buttons for 'Reset', 'Format', and 'Run'. At the bottom right of the slide, there is a cartoon illustration of a white bear wearing glasses.

## Learn Go with Tests



# Falls Go für dich neu ist ....

“keep it simple”

Verwende Generics (wie zB `slices.Contains()`), aber meist gibt es keinen Grund selber generische Funktionen zu schreiben.

Verwende existierende Interfaces, aber meist musst du keine Interfaces “erfinden”.

## Recommended Packages: pkg.go.dev

golang.org/x/exp/maps: *maps.Keys(), maps.Values()*

testify: *require.Equal(t, expected, actual)*

cobra: *Parse command line arguments.*

zombiezen/go-sqlite: *SQLite without CGO*

kubernetes/client-go: *Connect to the API-Server*

Cluster API: *Manage Kubernetes Cluster with Kubernetes*



# Web Dev: htmx.org and templ.guide

[templ.guide](https://templ.guide) Create HTML with Go. Has own language-server

[htmx.org](https://htmx.org): Not related to Go, but a pragmatic way to create web applications

# ❤️ Du bist nicht alleine ❤️

Answering Machines:

Google, Bing, Stackoverflow, ChatGPT, ...

Q & A:

[reddit.com/r/golang/](https://reddit.com/r/golang/)

[stackoverflow.com/questions/tagged/go](https://stackoverflow.com/questions/tagged/go)

[forum.golangbridge.org/](https://forum.golangbridge.org/)

Podcasts:

[changelog.com/gotime](https://changelog.com/gotime)

[cupogo.dev/](https://cupogo.dev/)

# Viel Spaß mit Go!

