

Introduction to

Golang

The Golang logo, featuring the word "GO" in a stylized blue font with three horizontal lines to the left of the "G".

Advanced Web Topics | WMAD Sr. B

Eduardo Pohl & Yasmin Kobayachi



Today's agenda

What we will cover

History of Go

.....

How Go works

.....

Libraries & Packages

.....

Variables

.....

Arrays & Slices

.....

Functions

.....

Demo

Brief History of Go



2009

Go was developed by
Google developers

2012

Initial version was released

2023

Current version is **go1.20**
released on Feb 14th, 2023

How Go works

Designed to be simple and easy to use, with a focus on developer productivity

Syntax

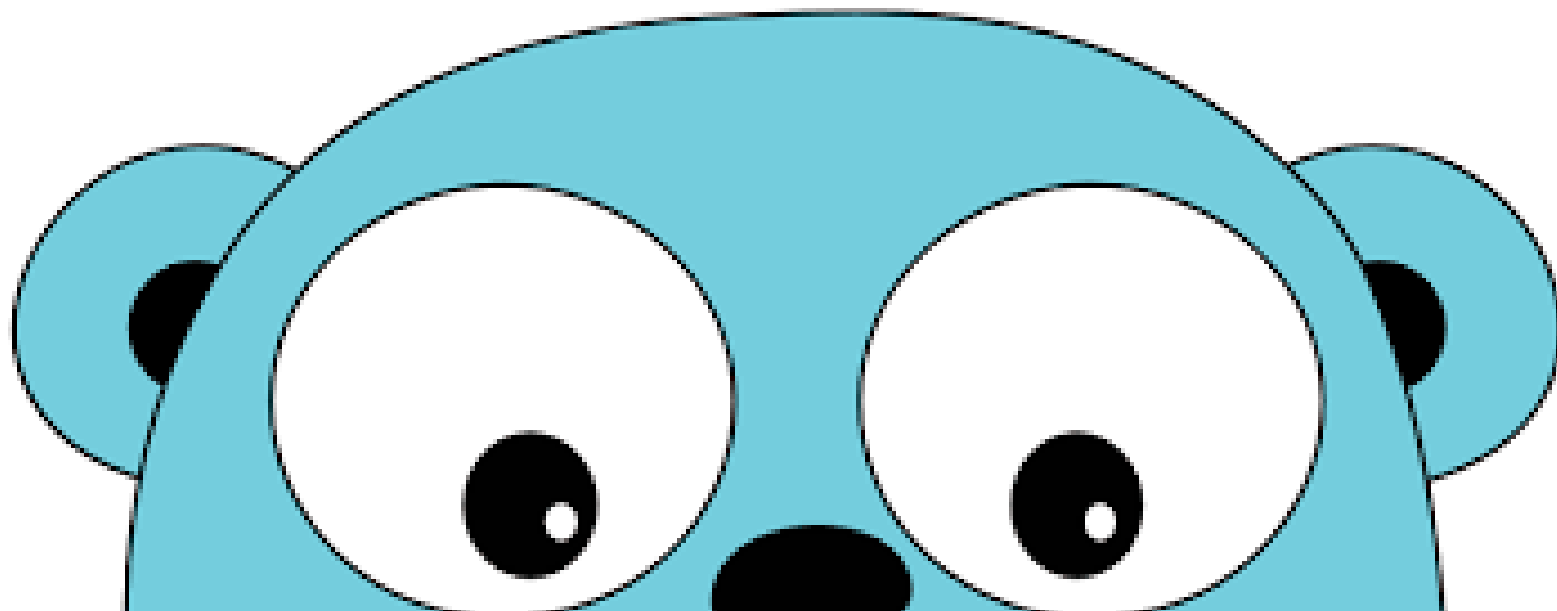
- Inspired by several other programming languages
 - Similar to C in its syntax with a few key differences
 - Around 25 keywords
-

Compilation

- Compiled into executables that can run anywhere, even if the machine doesn't have Go installed
-

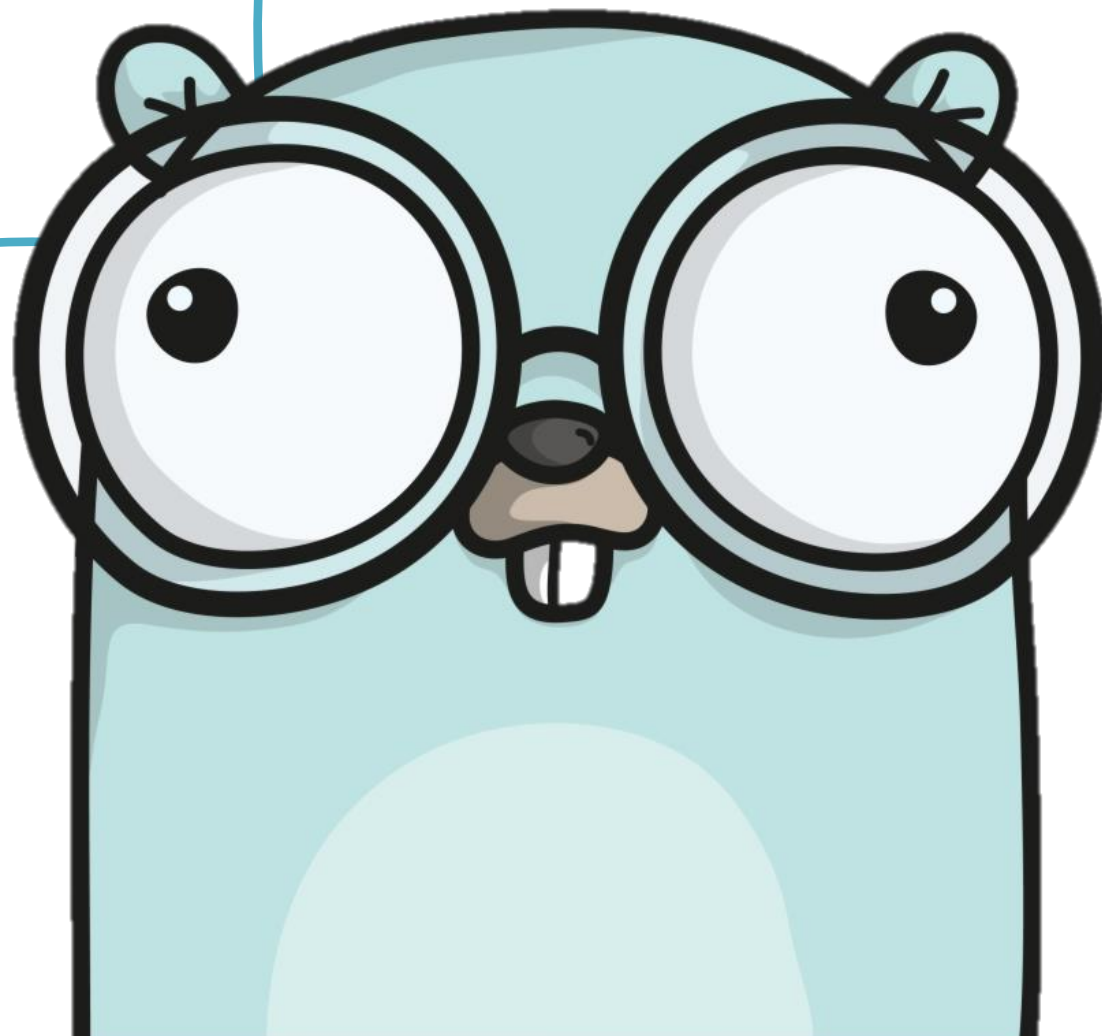
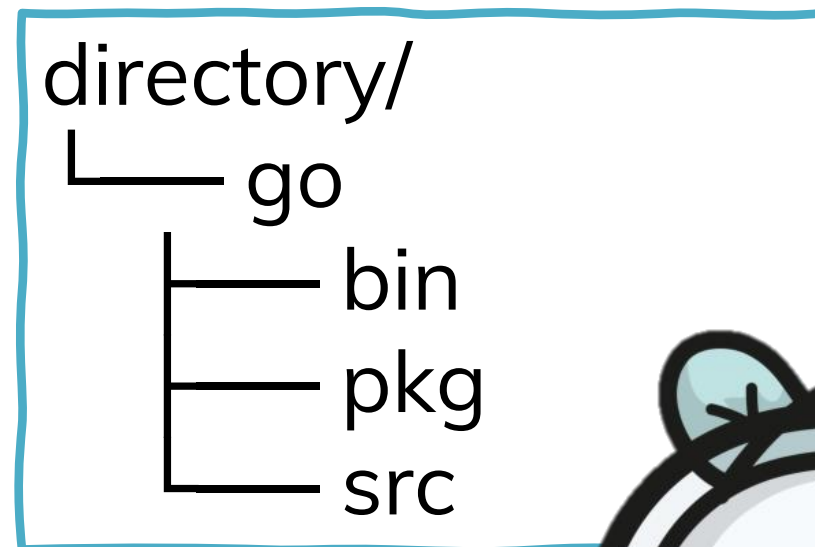
Concurrency

- Go's concurrency model is based on lightweight threads called goroutines and they can execute independently of each other



How Go works

Folders in Go needs to follow specific names



bin

- It holds all compiled code, the executable files (binary files)
-

pkg

- It holds all shared packages of the applications
-

src

- Where we write our code for each application
- Hold the third-party packages

Libraries & Packages



Standard Library

Go already comes with a rich standard library, providing a wide range of functionality



Third-Party Packages

External packages and libraries can be easily installed and used

Variables

- There are only 5 types of variables
- We can't declare a variable and not use it – that will cause a compilation error!
- We don't need to explicitly declare the variable type

TYPE	DEFAULT VALUE
string	""
int	0
float32 or float64	0
bool	false
*	nil



Variables

- Declaring variables

```
var variableName string = "Go1ang" // explicitly declaring a string
```

```
var variableName = "Go1ang" // Go already understands this is a string
```

```
variableName := "Go1ang" // short declaration syntax
```

```
var example float32 = 2.023 // explicitly declaring a float32
```

```
var example = 2.023 // Go is the one who decides which float it  
example := 2023 should be
```

```
const year = 2023
```


Variables

- Declaring variables

```
var firstName, lastName string
```

// we can declare many
variables at the same time

```
var (  
    weight = 10.85  
    name = "Golang"  
    quantity := 3  
)
```

Variables

- To know the variable type, we need to use the function `TypeOf` inside the `reflect` package

```
example := 2023
```

```
typeOfExample := reflect.TypeOf(example)
```

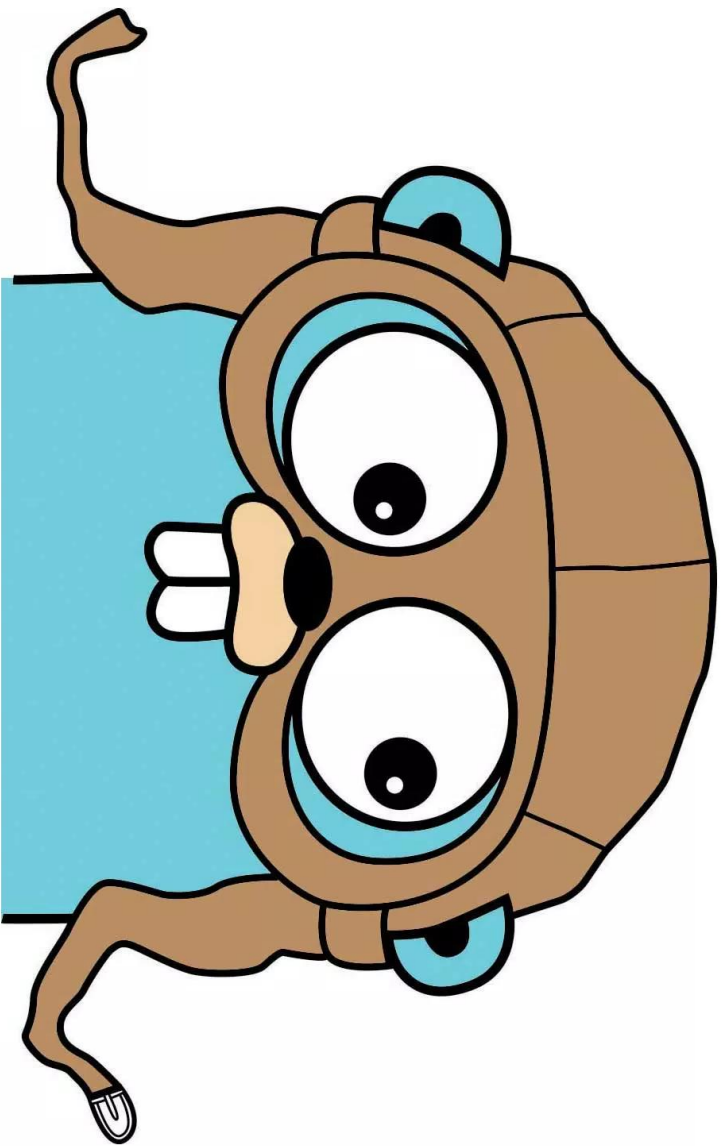
```
fmt.Println(typeOfExample)
```

```
// fmt is another package
```

```
// this will print float32 or float64
```

Arrays & Slices

- Slices are a type of an array
- The difference between arrays and slices is that arrays have a pre-determined size, while slices don't
- When declared, arrays give the default value for its elements – that means that if you don't initialize one of the elements, it will have an initial value based on its type



Arrays

- Declaring an array and changing the value for its elements

```
var arrayName [3]int
```

```
arrayName[0] = 10
```

```
arrayName[2] = 30
```

```
fmt.Println(arrayName)
```

```
// output:
```

```
// [10 0 30]
```

Slices

- Declaring and initializing a slice

```
sliceName := []string{"Red", "Blue"}
```

```
fmt.Println(sliceName)
```

```
// output:
```

```
// [Red Blue]
```

Slices

- Slices have lengths and capacities:
 - Length is the number of items inside a slice – we can check it using `len(slice)`
 - Capacity is how many elements a slice can hold – we can check it using `cap(slice)`

```
colors := []string{"Red", "Blue", "Yellow"}
```

```
fmt.Println(len(colors)) // output: 3
```

```
fmt.Println(cap(colors)) // output: 3
```

Slices

- To add a new element to a slice, we use `append(slice, element)`
 - In this case, the capacity will double from its original capacity automatically

```
colors = append(colors, "Black")
```

```
fmt.Println(colors)           // output: [Red Blue Yellow Black]
```

```
fmt.Println(len(colors))      // output: 4
```

```
fmt.Println(cap(colors))      // output: 6
```

Functions

- We use the keyword **func** to declare a function
- **func main()** is the primary function
 - It doesn't return anything and doesn't receive any parameters
 - When it ends, the application stops running
- A function can return one or more parameters
- Go doesn't have access modifiers. What will define the access control is the first letter case of a field , function or method name



camelCase	Private (unexported)
PascalCase	Public (exported)

A large blue circle is centered on the page. To its left, there is a black curly brace '{' positioned vertically in the middle of the circle's height.

- 

```
func sum(num1, num2 int) int {           // Function to sum up 2 numbers
    return num1 + num2                    // This will return an integer
}
```

```
// This will return an integer
```



Functions

- Variadic function
 - Receives an undetermined number of parameters
 - We use ... to indicate it is a variadic function

```
func sum(numbers ...int) int {  
    total := 0  
    for _, num := range numbers {  
        total += num  
    }  
    return total  
}
```

```
// Range returns two values:  
the index and the value for each iteration  
  
// We won't need the index, so we name it _  
  
// This way, Go won't complain we are not  
using this value
```

Functions

- Considering the previous variadic function

<code>fmt.Println(sum())</code>	<code>// output: 0</code>
<code>fmt.Println(sum(1, 2, 3))</code>	<code>// output: 6</code>
<code>fmt.Println(sum(10, 20, 30, 40, 50))</code>	<code>// output: 150</code>

Let's code!



References

Documentation

- *Documentation - The Go Programming Language*. (n.d.). <https://go.dev/doc/>
- Wikipedia contributors. (2023, March 3). *Go (programming language)*. Wikipedia. [https://en.wikipedia.org/wiki/Go_\(programming_language\)](https://en.wikipedia.org/wiki/Go_(programming_language))

.....

Courses

- *Cursos da Formação Linguagem Go*. (n.d.). Alura. <https://cursos.alura.com.br/formacao-go>
- The Net Ninja. (2021, May 19). *Go (Golang) Tutorial #1 - Introduction & Setup* [Video]. YouTube. https://www.youtube.com/watch?v=etSN4X_fCnM
- freeCodeCamp.org. (2019, June 20). *Learn Go Programming - Golang Tutorial for Beginners* [Video]. YouTube. <https://www.youtube.com/watch?v=YS4e4q9oBaU>

References

Images and others

- *Presentation template*. (n.d.). Canva. Retrieved March 1, 2023, from <https://www.canva.com/>
- Flaticon. (2019, July 15). *Folder Icon - 1975660*. https://www.flaticon.com/free-icon/folder_1975660?term=file&related_id=1975660
- *File:Go gopher favicon.svg - Wikimedia Commons*. (n.d.). https://commons.wikimedia.org/wiki/File:Go_gopher_favicon.svg
- *Gopher*. (n.d.). Retrieved March 6, 2023, from <https://golangforall.com/en/>
- *Go Gopher coding it up Cloudflare and Kubernetes*. (n.d.). Dribbble. <https://dribbble.com/shots/3878129-Go-Gopher-coding-it-up-Cloudflare-and-Kubernetes>
- *Gopher with Laptop*. (n.d.). Dribbble. <https://dribbble.com/shots/18090283-Gopher-with-Laptop>
- C. (n.d.). *GitHub - cilium/pwru: Packet, where are you? -- eBPF-based Linux kernel networking debugger*. GitHub. <https://github.com/cilium/pwru>
- *Datei Anzeigen - Golang Gopher China Transparent PNG - 2838x2716 - Free Download on NicePNG*. (n.d.). NicePNG.com. https://www.nicepng.com/ourpic/u2e6t4e6t4a9u2e6_datei-anzeigen-golang-gopher-china/
- Cardoza, C. (2021, March 9). *Why developers love Go*. SD Times. <https://sdtimes.com/softwaredev/why-developers-love-go/>
- *CodePen Home Bootstrap Crud Data Table For Database with Modal Form*. (n.d.). Codepen. Retrieved March 7, 2023, from <https://codepen.io/naikjavaaid/pen/XPrpjr>