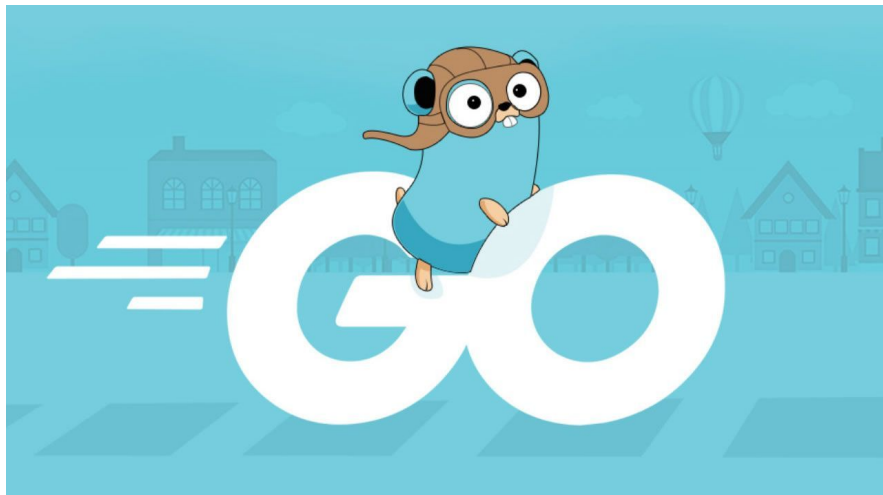


# Learn by Doing

Greg Horie

# Overview

- Quick Background - Go Design
- Learn By Doing Format
- Learn By Doing Examples
- Summary
- Future Discussions
- Feedback



# Go Design Inspirations

- Designed as a next-generation C
- Borrows some syntax from C
- Borrows from Pascal, Modula, and Oberon



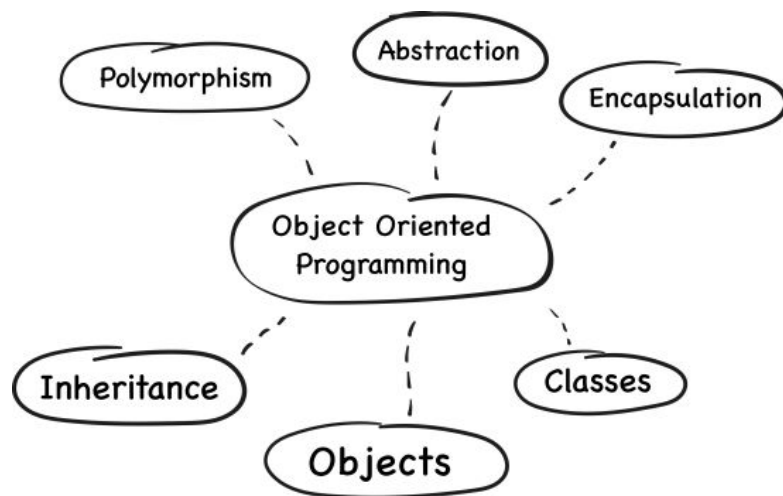
# Go Design Choices

- Compiled, statically typed language
- Compiled executables are operating system specific
- Compiled applications contain a statically-linked run-time
- Provides the illusion of an interpreted language
- No virtual-machine
- Garbage collection is a feature



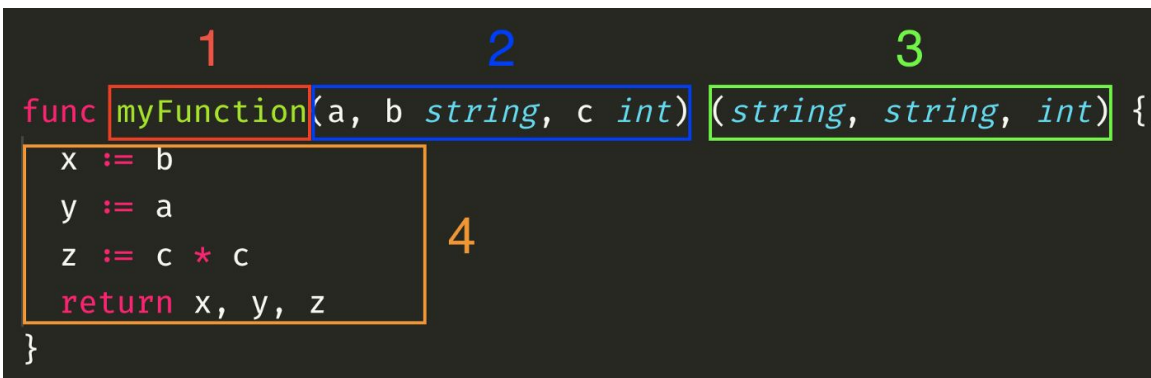
# Is Go Object-Oriented?

- Has some OOP features
- Can define custom interfaces
- Can define types with member methods
- Can define structs with member fields



# Syntax Rules

- Go is case sensitive
- Variables and package names are in lowercase and mixed case
- Initial character in public field names are uppercase
- Initial uppercase character means symbol is exported
- No semicolons required, but you can use them

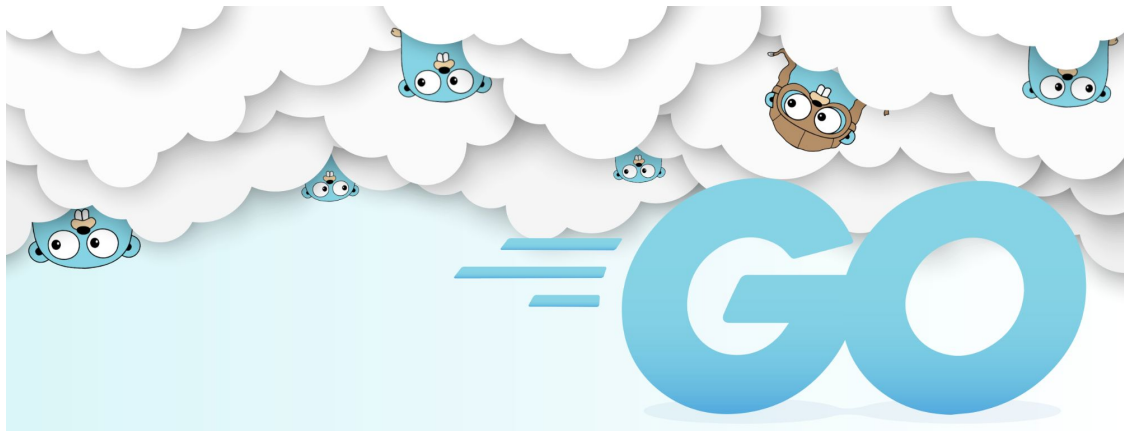


The image shows a Go function definition with four numbered annotations: 1 (red) points to the function name `myFunction`; 2 (blue) points to the parameter list `(a, b string, c int)`; 3 (green) points to the return type `(string, string, int)`; and 4 (orange) points to the function body block. The code is as follows:

```
func myFunction(a, b string, c int) (string, string, int) {  
    x := b  
    y := a  
    z := c * c  
    return x, y, z  
}
```

# Learn By Doing Format

- Try a new experiment
- Code example plus discussion
  - Repeat
- I'll ask for feedback at the end of the presentation



# Examples

- Check out GitHub for the examples:
  - <https://github.com/netserf/vicpimakers-presentation-go-learn-by-doing>



# Summary

- Go is a compiled, statically typed language
- It's easy to learn if you're already familiar with programming
- It compiles down to a single executable!
  - Great for publishing your code, removing many dependency challenges
- It comes with a rich standard library!
  - Encourages building your own over external frameworks and libraries
- If you want a modern language for building server-side components without the challenges of memory management, then try out Go.

# Possible Future Discussions

- Go: Going Deeper (Learn by Doing 2)
- GitHub Actions
- Diagrams as Code
- Python Command Line Tools
- Python Project Templates
- 10 Python Idioms
- Kubernetes

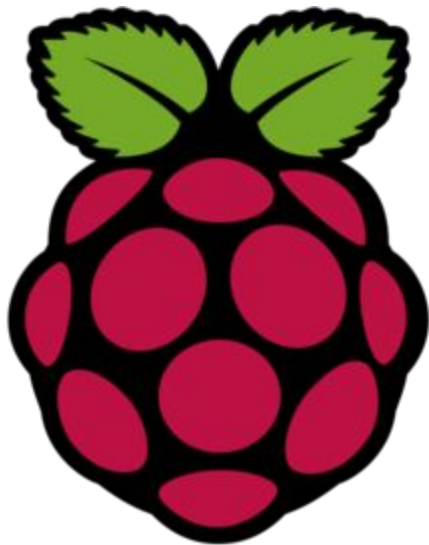


# Feedback

- Anonymous Feedback:
  - <https://forms.gle/EmtLgnfQWQb9q5v86>

# VicPiMakers and Others Slack

- Please let us know if you want an invite to this Slack group



# Backup Slides



# Not Supported in Go

- No type inheritance
- No method or operator overloading
- No structured exception handling
- No implicit numeric conversions



# Package vs Module

## Go Package

- A directory of .go files.
- Basic building block of a Go program.
- Help to organize code into reusable components.

## Go Module

- Collection of packages.
- Includes built-in dependencies and versioning.
- Out of scope for this presentation.

# Syntax Rules - Braces

- Code blocks are wrapped with braces
- Starting brace MUST BE on the same line as preceding statement

```
for i := 0; i < 10; i++ {  
  
    fmt.Println(i)  
  
}
```



# Built-In Functions

- **Link:** <https://golang.org/pkg/builtin>
- Go compiler assumes builtin package is always imported
- **Examples:**
- `len(string)` - return string length
- `panic(error)` - stops execution and displays error message
- `recover()` - manages behavior of a panicking go routine

# Golang.org

- **Link:** <https://golang.org>
- Try the Go language playground on the homepage
- Also try the full page version on <https://play.golang.org>
  - See code samples listed
- **Downloads:** <https://golang.org/dl/>