Learn by Doing

Greg Horie

Overview

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

Go Design Inspirations

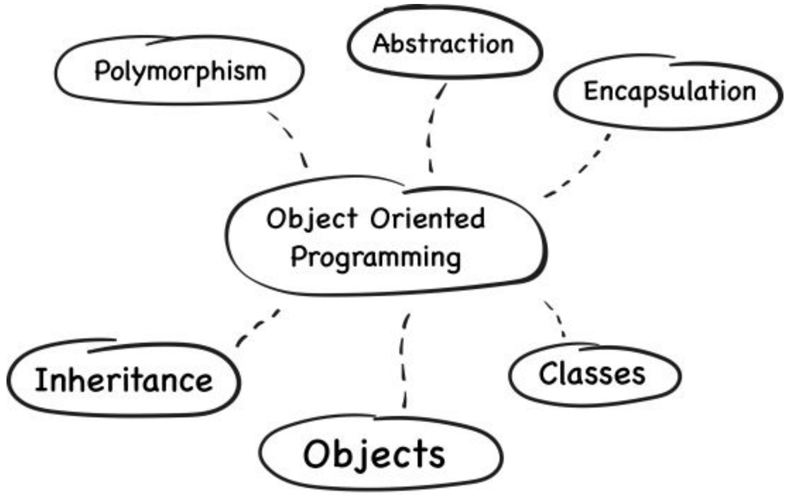
- Designed as a next-generation C
- Borrows some syntax from C
- Borrows from Pascal, Modula, and Oberon

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

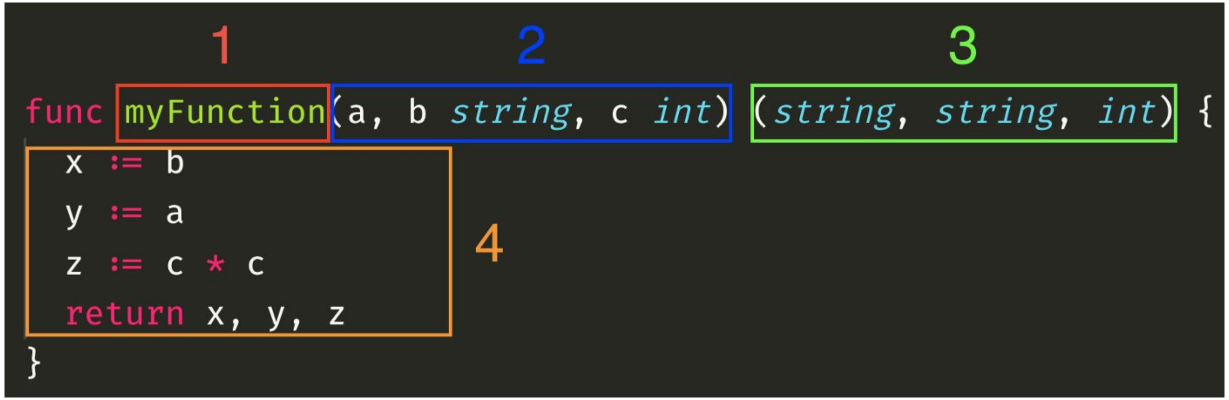
Is Go Object-Oriented?

- 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



```go
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 examples plus discussion
- Iterate
- I'll ask for feedback at the end of the presentation

Hello World

- https://github.com/netserf/vicpimakers-presentation-go-learn-by-doing/blob/main/examples/01_hello_world.go
-

Package vs Module

Go Package

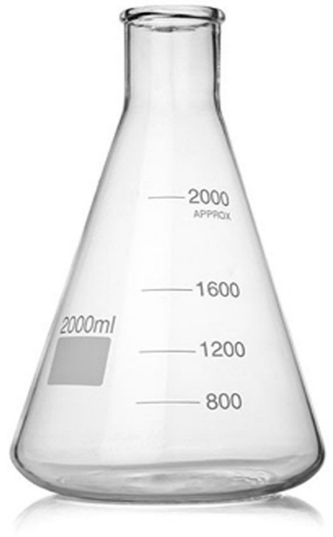
- A directory of .go files.
- Basic building block of a Go program.
- Help to organize code into reusable components.

Go Module

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

Summary

-



Possible Future Discussions

- Go - Learn by Doing 2
- GitHub Actions
- Python Click for CLI tools
- Idiomatic Python
- Kubernetes
- Service Meshes

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

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/