Programming in Go

Matt Holiday
Christmas 2020

A Simple Example

Read a name from the command line

```
package main

import (
    "fmt"
    "os"
)

func main() {
    fmt.Printf("Hello, %s!\n", os.Args[1])
}
```

Running a program with a bug

From the command line:

```
$ go run .   ## no name
panic: runtime error: index out of range [1] with length 1

goroutine 1 [running]:
main.main()
    /Users/mholiday/go/src/hello/main.go:9 +0xce
exit status 2
```

What went wrong? We read past the end of `os.Args`!

Build a unit test

```go
//file: cmd/main.go
package main

import (
    "fmt"
    "hello"
    "os"
)

func main() {
    if len(os.Args) > 1 {
        fmt.Println(hello.Say(os.Args[1]))
    } else {
        fmt.Println(hello.Say("world"))
    }
}
```

Build a unit test

```go
//file: hello.go
package hello

import "fmt"

func Say(name string) string {
    return fmt.Sprintf("Hello, %s!", name)
}
```

Build a unit test

```go
//file: hello_test.go
package hello

import "testing"

func TestSayHello(t *testing.T) {
    want := "Hello, test!"
    got := Say("test")

    if want != got {
        t.Errorf("wanted %s, got %s", want, got)
    }
}
```

Running a unit test

From the command line:

```
$ go test ./...

ok      hello   0.003s
?       hello/cmd   [no test files]
```

Something a little fancier

```go
//file: hello.go
package hello

import "strings"

func Say(names []string) string {
    if len(names) == 0 {
        names = []string{"world"}
    }

    return "Hello, " + strings.Join(names, ", ") + "!"
}
```

Find it on GitHub: matt4biz/go-class-hello

Something a little fancier

```
//file: hello_test.go
package hello

import "testing"

func TestSay(t *testing.T) {
    subtests := []struct {
        items  []string
        result string
    }{
        {
            result: "Hello, world!",
        },
        {
            items:  []string{"Matt"},
            result: "Hello, Matt!",
        },
        . . .
```

Something a little fancier

```
        . . .

        {
            items:  []string{"Matt", "Cary", "Anne"},
            result: "Hello, Matt, Cary, Anne!",
        },
    }

    for _, st := range subtests {
        if s := Say(st.items); s != st.result {
            t.Errorf("got %s, gave %v, wanted %s", s,
                st.items, st.result)
        }
    }
}
```

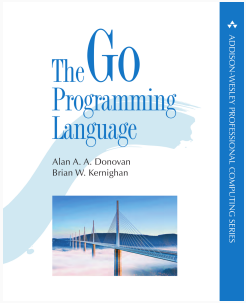
Something a little fancier

```
//file: cmd/main.go
package main

import (
    "fmt"
    "hello"
    "os"
)

func main() {
    fmt.Println(hello.Say(os.Args[1:]))
}
```

Significant changes since the book

Language:

1. struct conversion with mismatched tags (1.8)
2. type aliases (1.9)
3. **Go modules** (1.11+)
4. improved numeric literals and error handling (1.13)
5. overlapping interfaces (1.14)



Runtime:

1. performance improvements in every release
2. full (async) preemption of goroutines (1.14)

Go modules

Go modules are the one really big change from the *GOPL* book (instead of using `$GOPATH`)

You create a root directory with the module name, e.g., `hello`

Run `go mod init hello` to create the file `go.mod`:

```
module hello

go 1.14
```

This file will also end up with a list of 3rd-party dependencies (later)