

Getting Started with Google GoLang

...

Week 4



Error handling in Go

Go has an inbuilt **error** type.

An error variable is returned by lots of functions in case of failure.



Error is **nil** on successful execution.




Detecting errors

errors.go

```
1 package main
2
3 import (
4     "fmt"
5     "os"
6     "strconv"
7 )
8
9 func main() {
10     x, y := "10", "20"
11
12     a, err := strconv.Atoi(x)
13     if err != nil {
14         fmt.Println("Error in converting x!")
15         os.Exit(1)
16     }
17
18     b, err := strconv.Atoi(y)
19     if err != nil {
20         fmt.Println("Error in converting y!")
21         os.Exit(1)
22     }
23
24     fmt.Println("Sum is", a+b)
25 }
26
```

 Week-4 > go run errors.go
Sum is 30
 Week-4 > |

```
9 func main() {
10     x, y := "10", "a"
11 }
```

 Week-4 > go run errors.go
Error in converting y!
exit status 1
?1 Week-4 > |

Making use of err

errors.go

```
1 package main
2
3 import (
4     "fmt"
5     "log"
6     "strconv"
7 )
8
9 func main() {
10     x, y := "10", "a"
11
12     a, err := strconv.Atoi(x)
13     if err != nil {
14         log.Fatal(err)
15     }
16
17     b, err := strconv.Atoi(y)
18     if err != nil {
19         log.Fatal(err)
20     }
21
22     fmt.Println("Sum is", a+b)
23 }
24
```

```
?1 Week-4 > go run errors-log.go
2021/02/16 13:39:15 strconv.Atoi: parsing "a": invalid syntax
exit status 1
?1 Week-4 > |
```

Creating custom errors

custom-errors.go

```
3 import (
4     "errors"
5     "fmt"
6     "log"
7 )
8
9 func divide(a, b int) (int, error) {
10     if b == 0 {
11         return 0, errors.New("Cannot divide by 0!")
12     }
13
14     return a / b, nil
15 }
16
17 func main() {
18     x, y := 10, 2
19     z, err := divide(x, y)
20     if err != nil {
21         log.Fatal(err)
22     }
23
24     fmt.Println("Quotient:", z)
25
26     a, b := 15, 0
27     c, err := divide(a, b)
28     if err != nil {
29         log.Fatal(err)
30     }
31
32     fmt.Println("Quotient:", c)
33 }
```

```
?1 Week-4 > go run custom-errors.go
Quotient: 5
2021/02/16 13:51:27 Cannot divide by 0!
exit status 1
?1 Week-4 > |
```

Creating custom errors

```
func divide(a, b int) (int, error) {  
    if b == 0 {  
        return 0, errors.New("Cannot divide by 0!")  
        // Alternate  
        return 0, fmt.Errorf("Cannot divide %v by %v", a, b)  
    }  
  
    return a / b, nil  
}
```

For more about errors, look at:

blog.golang.org/error-handling-and-go

Packages in Go

How to download external packages? We use **go get**
We then import them into our code.

Documentation for packages are available through:

- golang.org/pkg/
- pkg.go.dev/



Importing custom packages



```
Week-4 > go get github.com/cheggaaa/pb/v3
Week-4 > |
```

Package:

github.com/cheggaaa/pb

Using custom packages

```
custom-bar.go
1  package main
2
3  import (
4      "time"
5
6      "github.com/cheggaaa/pb/v3"
7  )
8
9  func main() {
10     count := 100000
11
12     // create and start new bar
13     bar := pb.StartNew(count)
14
15     for i := 0; i < count; i++ {
16         bar.Increment()
17         time.Sleep(time.Millisecond)
18     }
19     bar.Finish()
20 }
21
```

```
👾 Week-4 > go run custom-bar.go
44113 / 100000 [----->-----] 44.11%
```

More resources on imports

- digitalocean.com/community/tutorials/importing-packages-in-go
- callicoder.com/golang-packages/



Go Modules

How do we create our own packages and modules?

How do we write code spanning multiple packages?



Creating Modules

```
Week-4 > mkdir mods
Week-4 > cd mods
mods > |
```

```
mods > go mod init github.com/Gituser143/mods
go: creating new go.mod: module github.com/Gituser143/mods
mods > ls
go.mod
mods > |
```

```
mods > tree
.
├── farewell
│   └── bye.go
├── go.mod
├── greeting
│   └── greet.go
└── main.go

2 directories, 4 files
mods > |
```

Creating Modules

```
main.go
3  import (
4      "fmt"
5
6      "github.com/Gituser143/mods/farewell"
7      "github.com/Gituser143/mods/greeting"
8  )
9
10 func main() {
11     var name string
12
13     fmt.Printf("Enter your name: ")
14     fmt.Scanf("%s", &name)
15
16     greeting.SayHello(name)
17     farewell.SayBye(name)
18 }
```

```
greet.go
1  package greeting
2
3  import "fmt"
4
5  // Export functions you intend
6  // to use outside the package
7  func SayHello(name string) {
8      fmt.Printf("Hello %v!\n", name)
9  }
10
bye.go
1  package farewell
2
3  import "fmt"
4
5  // SayBye goodbye
6  func SayBye(name string) {
7      fmt.Printf("Good Bye %v!\n", name)
8  }
9
```

```
👾 mods > go build main.go
👾 mods > ./main
Enter your name: Bhargav
Hello Bhargav!
👾 mods > |
```

src > tree

```
.
├── display
│   ├── general
│   │   ├── init.go
│   │   └── overallGraphs.go
│   ├── misc
│   │   ├── help.go
│   │   └── keybinds.go
│   └── process
│       ├── allProcs.go
│       ├── init.go
│       └── procGraphs.go
├── export
│   ├── general
│   │   └── generalExport.go
│   └── proc
│       └── procExport.go
├── general
│   ├── cpuInfo.go
│   ├── errors.go
│   ├── generalStats.go
│   └── serveStats.go
├── process
│   ├── process.go
│   ├── serveData.go
│   └── updateProcess.go
└── utils
    ├── barGraph.go
    ├── dataFormat.go
    ├── data.go
    ├── errorArt.go
    ├── testUtils.go
    ├── tickutils.go
    └── utils_test.go
```

10 directories, 23 files

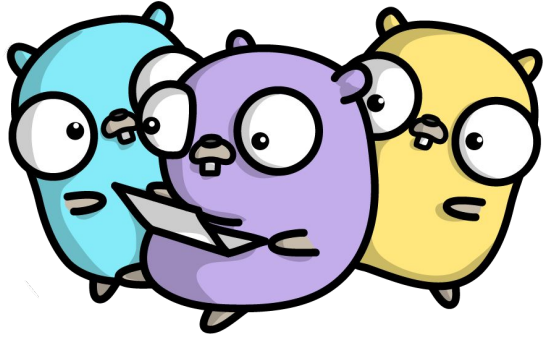
src > |

Go Modules

Resources:

- golang.org/doc/tutorial/create-module
- golangbyexample.com/go-module-sum-module/





Thank You!

Source Code and Slides available at:
github.com/Gituser143/PESU-IO-Go