LESSON 4

main.go

```
v, err := f()
if err != nil {
   return err
}
```

Go

if err != nil

Learn it all @MyWebUniversity



A FREE MYWEBUNIVERSITY COURSE

GO PROGRAMMING

LESSON 4

Functions and Errors as Values

Multiple returns, named results, variadic functions, closures and if err != nil

Defining Functions

funcs.go

```go
func add(a, b int) int {
    return a + b
}

func Average(scores []float64) float64 {    // exported
    sum := 0.0
    for _, s := range scores { sum += s }
    return sum / float64(len(scores))
}
```

- Write func, the name, parameters with their types after the names, and the return type …
- Names that start with a capital letter are exported, meaning other packages can use them. …

01

Multiple Return Values

divide.go

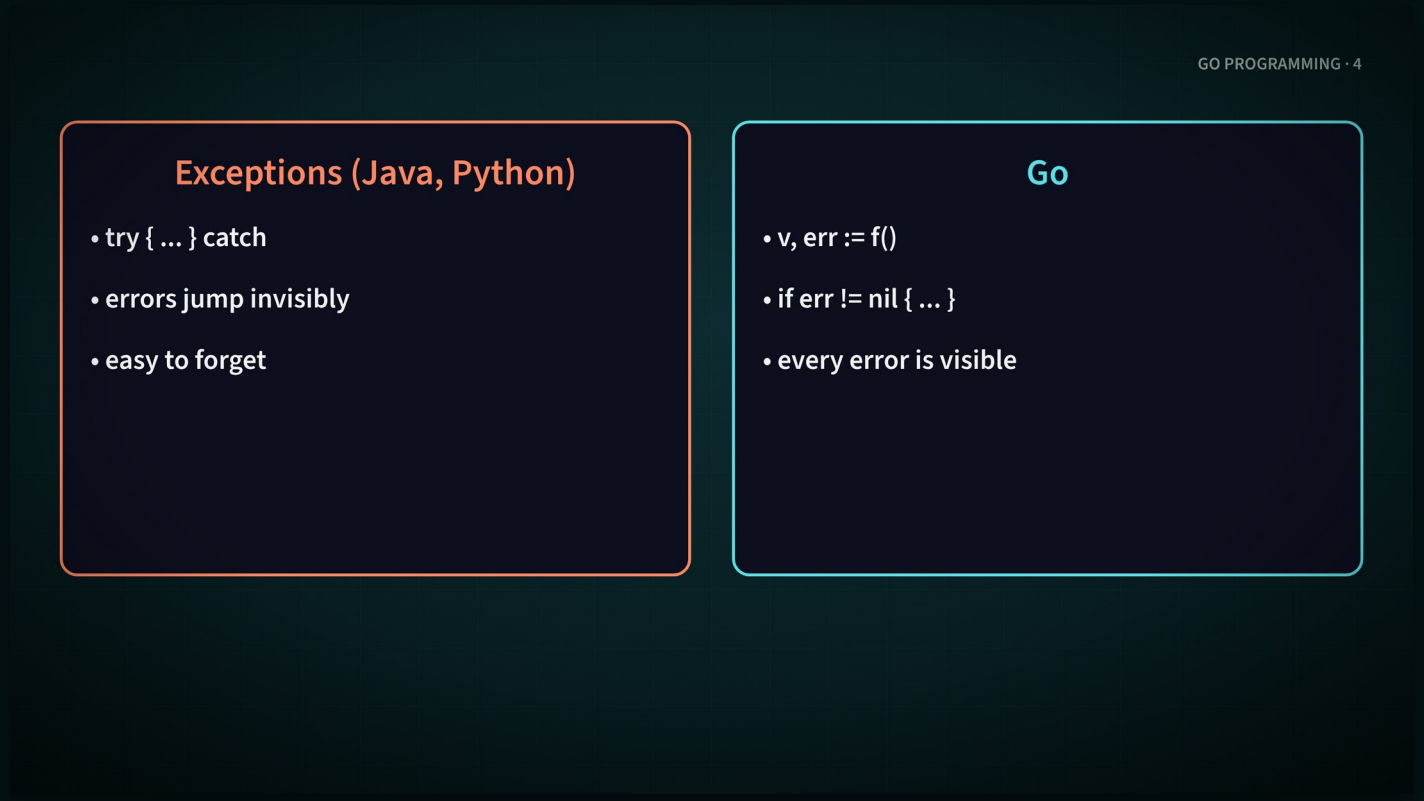
```go
func divide(a, b float64) (float64, error) {
    if b == 0 {
        return 0, errors.New("division by zero")
    }
    return a / b, nil
}

q, err := divide(10, 0)
if err != nil { fmt.Println("error:", err); return }
```

- A Go function can return several values. The standard pattern returns a result and an error …
- The caller receives both and checks the error right away.

02

Errors Are Values



- Go has no exceptions for normal errors. An error is just a value you return and check with ...
- It feels repetitive at first, but every failure point is visible in the code, and nothing ...

03

Variadic Functions

variadic.go

```go
func sum(nums ...int) int {
    total := 0
    for _, n := range nums { total += n }
    return total
}
sum(1, 2, 3)              // 6
scores := []int{90, 85, 70}
sum(scores...)            // 245
```

- Three dots before a type accept any number of arguments, which arrive as a slice. fmt dot …
- Pass an existing slice with three dots after it.

04

Functions as Values and Closures

closure.go

```go
func counter() func() int {
    count := 0
    return func() int {
        count++
        return count
    }
}
next := counter()
next(); next()
fmt.Println(next())       // 3
```

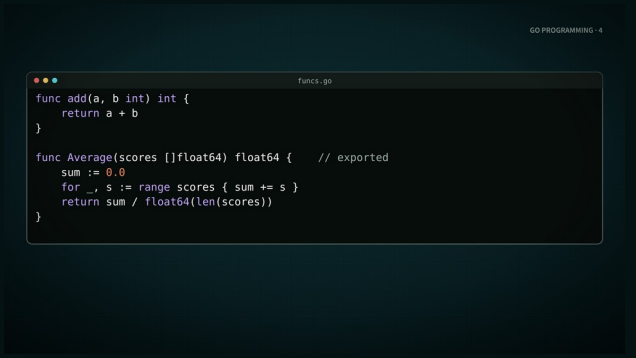
- Functions are values: store them in variables and pass them to other functions. An …
- Here a counter function returns a new function that remembers its own count.

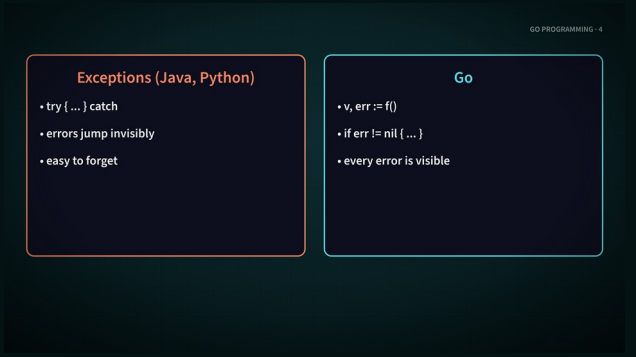
05

Lesson Map


1
Defining Functions
2
Multiple Return Values
3
Errors Are Values
4
Variadic Functions
5
Functions as Values and Closures

Sources and Further Reading





Thank you for watching

MyWebUniversity.com

Free technical education since 2000 — English and Dari

@MyWebUniversity · @MyWebUniversity-Dari