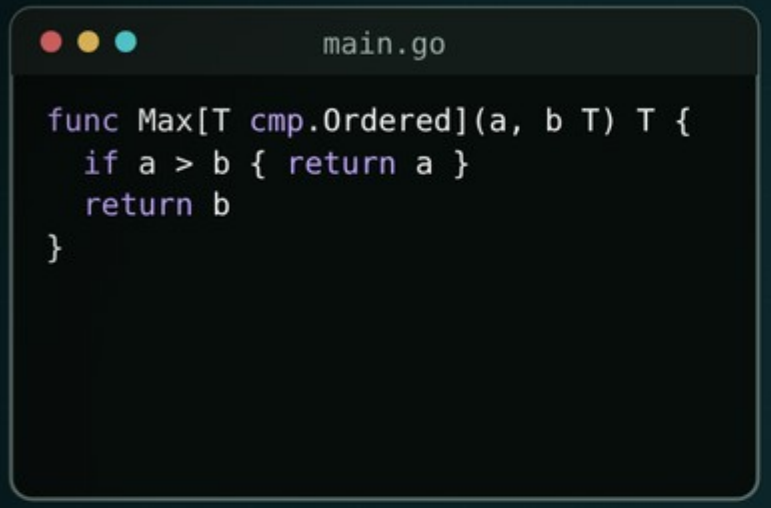
LESSON 11

Go

Tests & Generics



Learn it all @MyWebUniversity



A FREE MYWEBUNIVERSITY COURSE

GO PROGRAMMING

LESSON 11

Testing and Generics

go test, table-driven tests, benchmarks, and type parameters

Writing Tests

grade_test.go

```go
package grade

import "testing"

func TestLetter(t *testing.T) {
    got := Letter(92)
    if got != "A" {
        t.Errorf("Letter(92) = %q, want %q", got, "A")
    }
}
```

- Tests live next to your code in files ending with underscore test dot go. A test function …
- go test runs them all. No framework is needed; it is built into the toolchain.

01

Table-Driven Tests

grade_test.go

```
tests := []struct {
    name  string
    score int
    want  string
}{
    {"top", 95, "A"}, {"edge", 90, "A"}, {"mid", 80, "B"},
}
for _, tc := range tests {
    t.Run(tc.name, func(t *testing.T) {
        if got := Letter(tc.score); got != tc.want { t.Errorf("got %q", got) }
    })
}
```

- The Go way is a table of cases: a slice of structs with inputs and expected outputs, and …
- t dot Run gives each case a name, so failures say exactly which case broke.

02

Coverage and Benchmarks

terminal

```
$ go test -cover ./...
ok  gradebook/grade  0.003s  coverage: 94.1% of statements
$ go test -bench=. -benchmem
BenchmarkLetter-8   500000000   2.31 ns/op   0 B/op   0 allocs/op
```

- go test dash cover shows what percent of your code the tests run. Benchmark functions start ...
- Run them with dash bench, and Go reports nanoseconds per operation and memory allocations.

03

Generics: Type Parameters

generic.go

```
func Max[T cmp.Ordered](a, b T) T {
    if a > b { return a }
    return b
}
Max(3, 7)                 // 7
Max(2.5, 1.5)             // 2.5
Max("Herat", "Kabul")     // Kabul
```

- Since Go 1.18, functions and types can take type parameters in square brackets. One Max …
- A constraint like cmp dot Ordered says which types are allowed.

04

Generic Types

stack.go

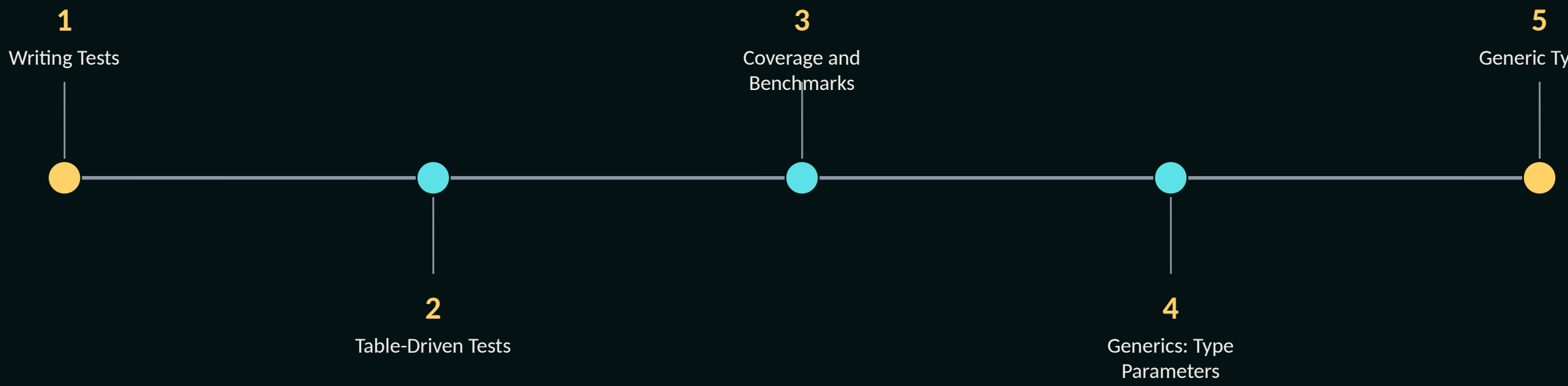
```go
type Stack[T any] struct{ items []T }

func (s *Stack[T]) Push(v T) { s.items = append(s.items, v) }
func (s *Stack[T]) Pop() (T, bool) {
    var zero T
    if len(s.items) == 0 { return zero, false }
    v := s.items[len(s.items)-1]
    s.items = s.items[:len(s.items)-1]
    return v, true
}
```

- Types can be generic too. A Stack of T works for any element type with full type safety and …
- Use generics for containers and algorithms. For everything else, plain types and interfaces …

05

Lesson Map


1
Writing Tests
2
Table-Driven Tests
3
Coverage and Benchmarks
4
Generics: Type Parameters
5
Generic Types

Sources and Further Reading

grade_test.go

```go
package grade

import "testing"

func TestLetter(t *testing.T) {
    got := Letter(92)
    if got != "A" {
        t.Errorf("Letter(92) = %q, want %q", got, "A")
    }
}
```

GO PROGRAMMING · 11

terminal

```
$ go test -cover ./...
ok  gradebook/grade  0.003s  coverage: 94.1% of statements
$ go test -bench=. -benchmem
BenchmarkLetter-8   500000000   2.31 ns/op   0 B/op   0 allocs/op
```

Thank you for watching

MyWebUniversity.com

Free technical education since 2000 — English and Dari

@MyWebUniversity · @MyWebUniversity-Dari