LESSON 5

main.go

```go
s := []int{1, 2}
s = append(s, 3)
m := map[string]int{}
```

# Go

## Slices & Maps

Learn it all @MyWebUniversity




A FREE MYWEBUNIVERSITY COURSE

# GO PROGRAMMING

LESSON 5

## Arrays, Slices and Maps

Fixed arrays, dynamic slices, append, len and cap, and maps

# Arrays



arrays.go

```
var scores [5]int                    // [0 0 0 0 0]
primes := [4]int{2, 3, 5, 7}
grid := [3][3]int{}                  // 2-D array
grid[1][1] = 5
copyOf := primes                     // full copy
```

- A Go array has a fixed length that is part of its type: an array of five ints. Arrays are …
- Because of that, you rarely use arrays directly. Slices, built on top of arrays, are what …

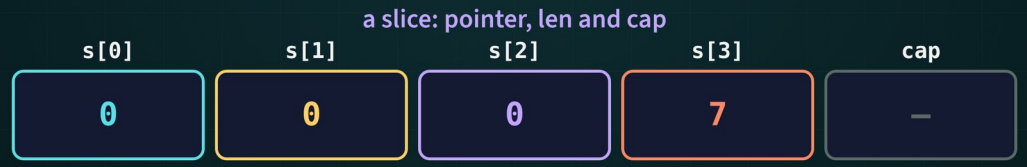
01

# Slices



main.go

```
s := make([]int, 3, 5)       // len 3, cap 5
s = append(s, 7)             // len 4, cap 5
```




- A slice is a flexible view into an array: a pointer, a length and a capacity. append adds …
- Always assign the result of append back to the slice, because it may return a new, bigger …

02

# Slicing and Sharing



slicing.go

```
nums := []int{10, 20, 30, 40, 50}
mid := nums[1:4]            // [20 30 40]
mid[0] = 99                 // nums is now [10 99 30 40 50]!
own := slices.Clone(nums[1:4])
slices.Sort(nums); slices.Contains(nums, 30)
```

- s colon from colon to takes a part of a slice, from the first index up to but not including …
- Changing one changes the other. Use copy, or slices dot Clone, when you need an independent …

03

# Maps



maps.go

```go
ages := map[string]int{"Mariam": 21, "Ahmad": 25}
ages["Zahra"] = 19
delete(ages, "Ahmad")
if age, ok := ages["Omid"]; ok {
    fmt.Println(age)
} else {
    fmt.Println("not found")
}
```

- A map links keys to values: map of string to int. Create it with a literal or make, read …
- The comma ok form tells you whether a key exists, because a missing key returns the zero …

04

# Counting Words



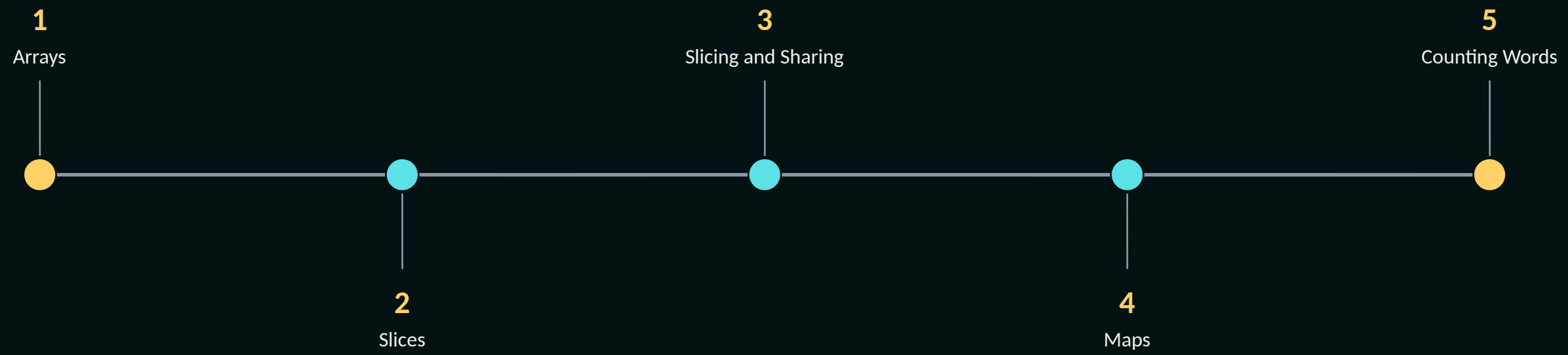
wordcount.go

```
counts := map[string]int{}
for _, w := range strings.Fields(text) {
    counts[strings.ToLower(w)]++
}
words := slices.Collect(maps.Keys(counts))
slices.SortFunc(words, func(a, b string) int { return counts[b] - counts[a] })
fmt.Println(words[:3])
```

- Maps and slices together solve real problems. Here we count words in a text, then sort them …
- Map iteration order is random by design, so sort the keys when order matters.

05

# Lesson Map


1
Arrays
2
Slices
3
Slicing and Sharing
4
Maps
5
Counting Words

# Sources and Further Reading



arrays.go

```go
var scores [5]int                       // [0 0 0 0 0]
primes := [4]int{2, 3, 5, 7}
grid := [3][3]int{}                     // 2-D array
grid[1][1] = 5
copyOf := primes                        // full copy
```

GO PROGRAMMING · 5

slicing.go

```go
nums := []int{10, 20, 30, 40, 50}
mid := nums[1:4]            // [20 30 40]
mid[0] = 99                 // nums is now [10 99 30 40 50]!
own := slices.Clone(nums[1:4])
slices.Sort(nums); slices.Contains(nums, 30)
```

Thank you for watching

# MyWebUniversity.com

*Free technical education since 2000 — English and Dari*

@MyWebUniversity · @MyWebUniversity-Dari