Functions

Nigel Poulton

@nigelpoulton | www.nigelpoulton.com

Module Overview

Why do we use functions?

Defining functions in Go

The **main** function

Custom functions

Introduce some advanced topics

This stuff is **IMPORTANT**

Why Use Functions

Functions enable to us to write clean, tidy, modular code.

Program (150 LOC)

Program (150 LOC)

Program (150 LOC)

Function

Program (150 LOC)

Function

Calls/references to function
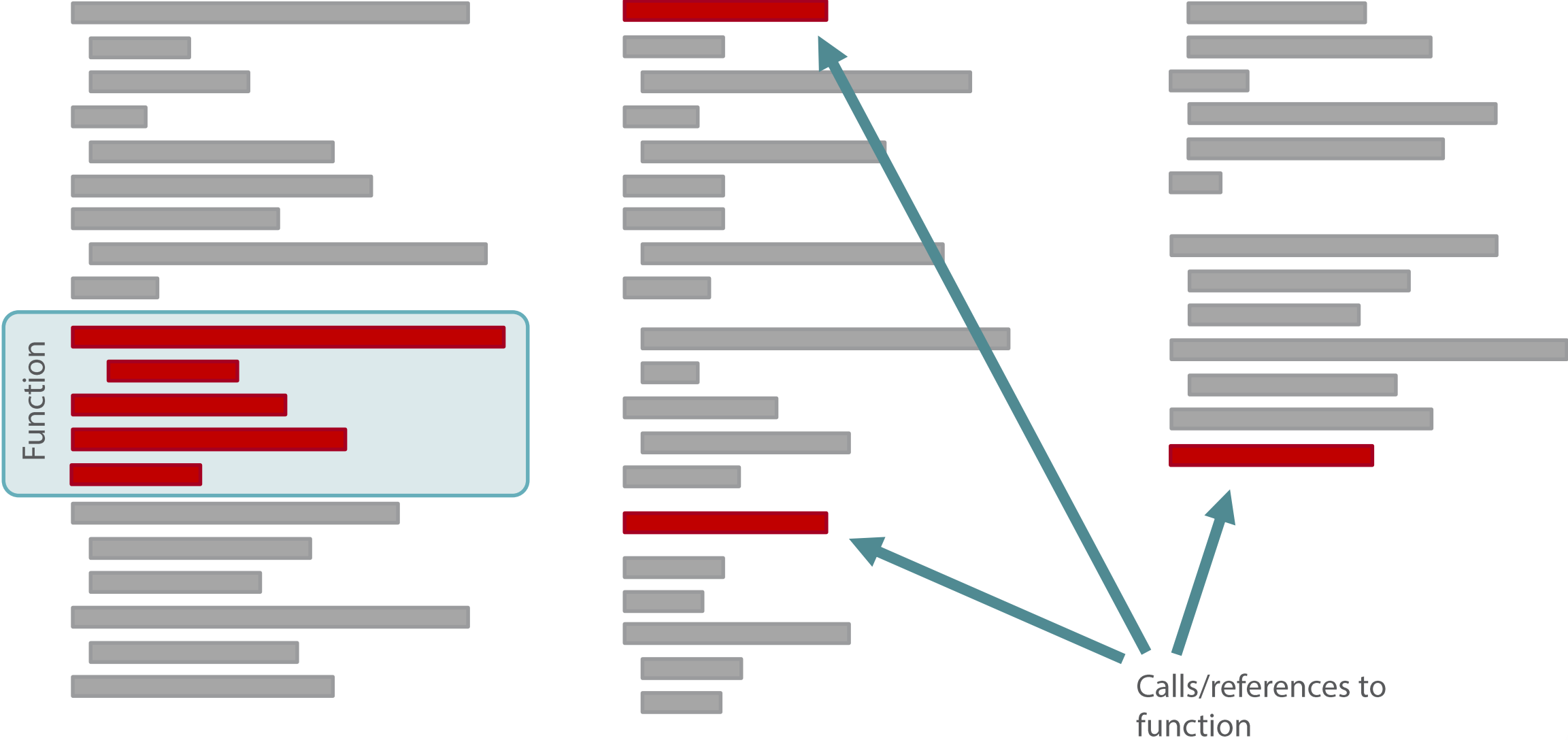
How Functions Work

At a high level…

titleCase()

hello pluralsight
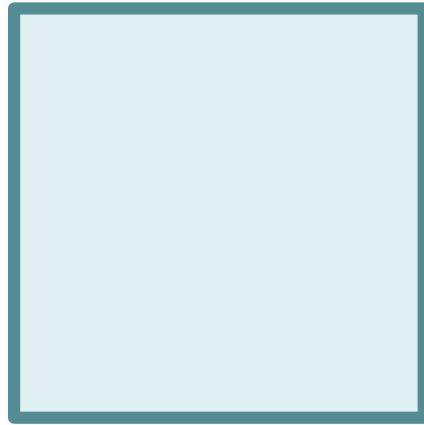
titleCase()
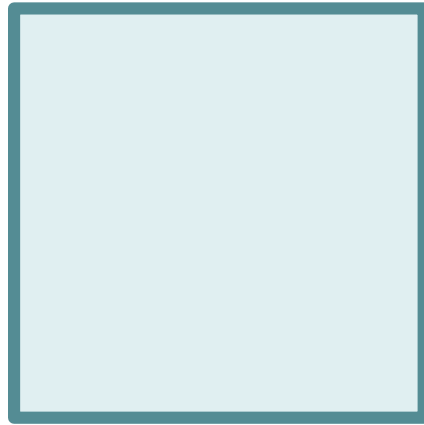
hello pluralsight

Hello Pluralsight

(Parameters)

(return values)

Function Syntax

```
func titlecase() {
    <code>
}
```

func keyword

All statements within a function enclosed within { }

```
func titlecase(text string) {
fmt.Println(titlecase(text))
}
```

func keyword

All statements within a function enclosed within **{ }**

```go
text := "hello pluralsight"
fmt.Println(titlecase(text))

func titlecase(text string) string {
    <code>
}    return text
}
```

func keyword

All statements within a function enclosed within **{ }**

Parameters and returns are defined within the function signature

The **return** keyword exits a function

Function Basics

Variadic Functions

Summary

This stuff is **IMPORTANT!**

- Make sure you understand it!

There is **sooooo much more** to functions!

Summary

Why Functions

- Concise code
- re-usable code
- Easy code to test
- Easy code to debug

Variadic Functions

```
(value1 ...string) {}
```

Function Syntax

```
func funcName(value1 string) string {
<code>
}
```