Level 1–2

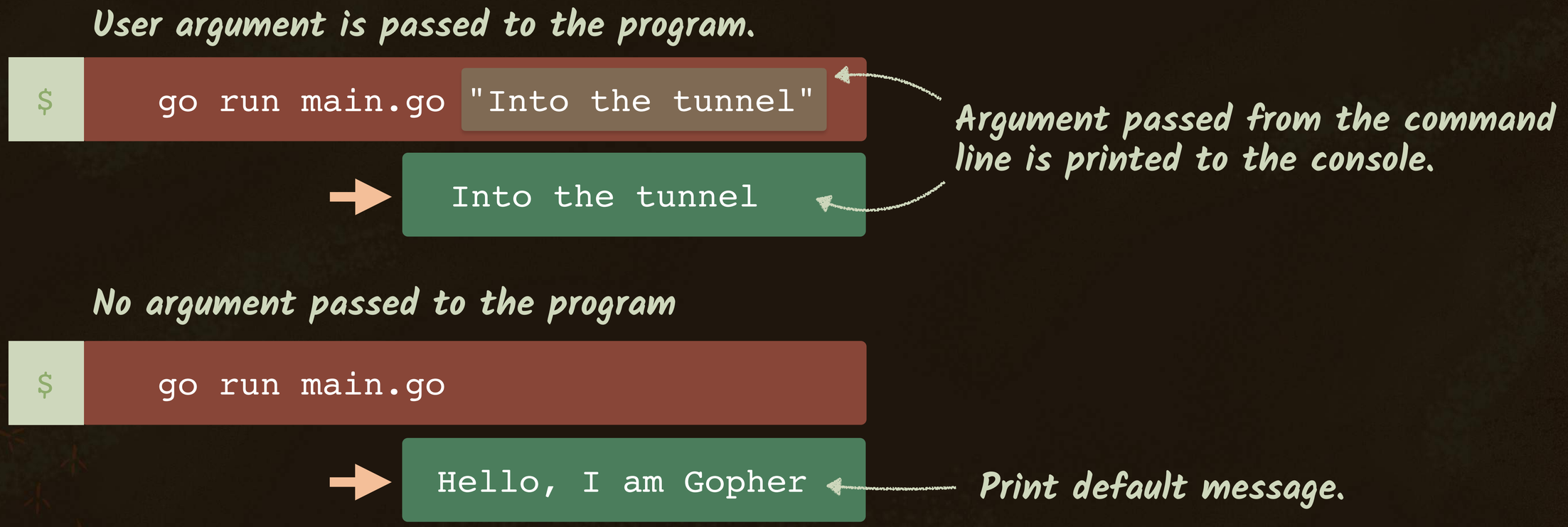
3, 2, 1… Go!

Conditionals, Args & Imports

ON TRACK with GOLANG

Printing Two Different Messages

We will write a program that reads a **user-supplied argument** from the command line and prints it to the console. If no argument is given, then a **default message** is printed.



Using Conditionals

There are no parentheses in if/else statements, and blocks **must be brace-delimited.**

src/hello/main.go

```
package main

import "fmt"

func main() {
  if                    {

  } else {

  }
}
```

Boolean expressions go here, and no parentheses are necessary.

Using Conditionals

The len() built-in function returns the **length of its argument.**

src/hello/main.go

```
package main

import "fmt"

func main() {
  if len(        ) > 1 {

  } else {

  }
}
```

Statement evaluates to true if len() returns a number greater than 1.

Built-in functions are functions that can be invoked directly without us having to import a package.

Reading Arguments

The os.Args array contains all arguments passed to the running program, including user-supplied arguments from the **command line.**

```go
package main

import (
  "fmt"
  "os"
)

func main() {
  if len(os.Args) > 1 {

  } else {

  }
}
```

Import package from the standard library

What a command-line argument looks like

```
$ go run main.go "Into the tunnel"
```

`os.Args[0]`

`os.Args[1]`

```
/var/folders/(...)/command-line-arguments/_obj/exe/main
```

Name of the temporary executable created by the go run command

An array with the program arguments, starting with the name of the executable and followed by any user-supplied arguments

Printing Arguments

We invoke fmt.Println() from both the if and else blocks. First, we pass it the user-supplied command-line argument (os.Args[1]), and, on the second block, a default greeting message.

```
package main

import (
  "fmt"
  "os"
)

func main() {
  if len(os.Args) > 1 {
    fmt.Println(os.Args[1])
  } else {
    fmt.Println("Hello, I am Gopher")
  }
}
```

User-supplied arguments start on index 1 of the array.

If no arguments are passed, then we print a default greeting message.

Running the Program With Arguments

If given an argument, then our program will print this argument to the console.

```
package main

import (
  "fmt"
  "os"
)

func main() {
  if len(os.Args) > 1 {
    fmt.Println(os.Args[1])
  } else {
    fmt.Println("Hello, I am Gopher")
  }
}
```

Returns 2...

...the if block is run...

```
$ go run main.go "Into the tunnel"
```

os.Args[1]

```
Into the tunnel
```

...and the argument is printed to the console.

Running the Program With No Arguments

If **no argument** is supplied, then our program will print the default message to the console.

```
package main

import (
  "fmt"
  "os"
)

func main() {
  if len(os.Args) > 1 {
    fmt.Println(os.Args[1])
  } else {
    fmt.Println("Hello, I am Gopher")
  }
}
```

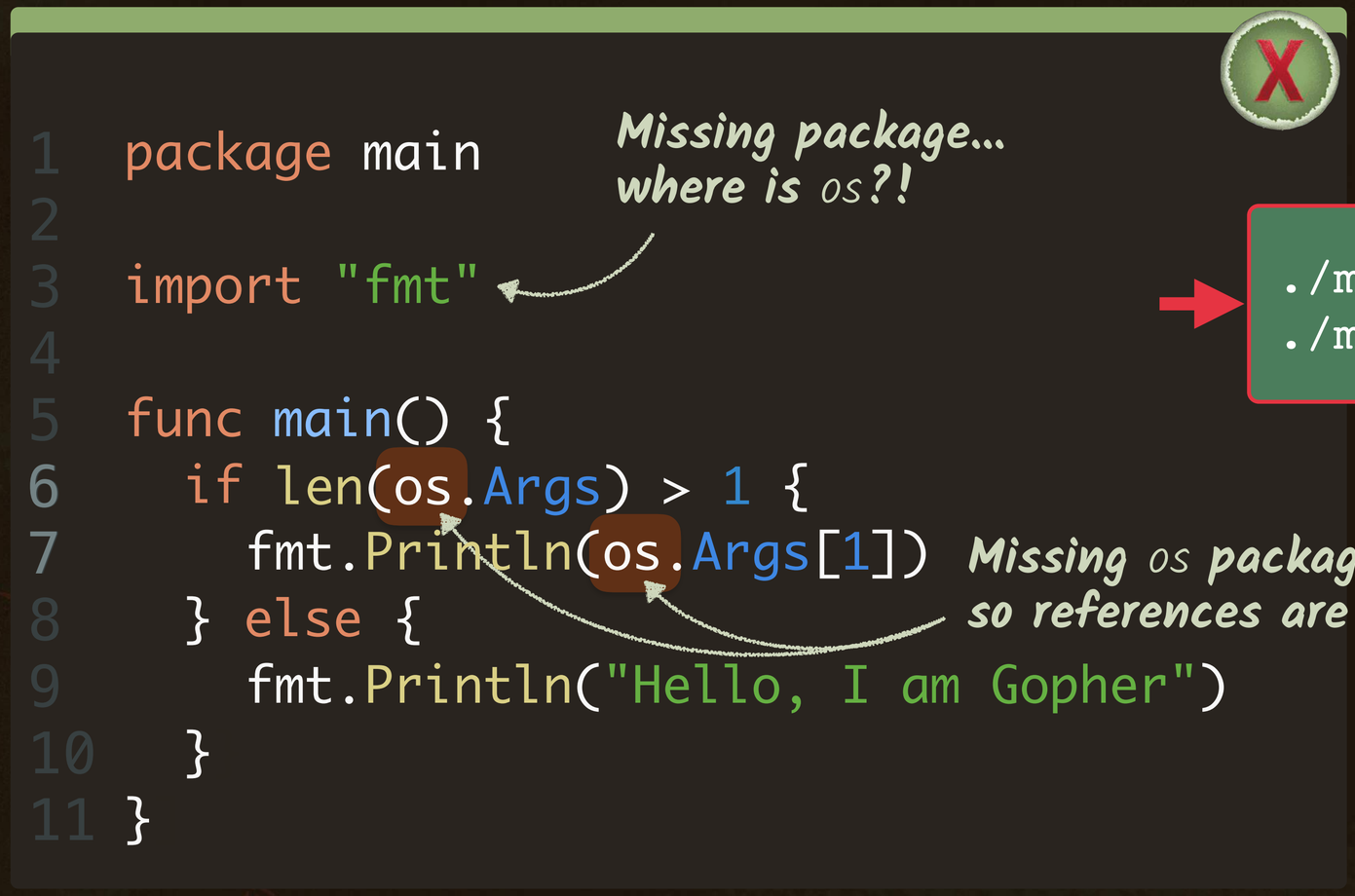
Returns 1...

...*the* else *block is run...*

...and the default message is printed to the console.

Running With Missing Imports

Any missing package imports will **raise an error** during the build process.



```
1  package main
2
3  import "fmt"
4
5  func main() {
6    if len(os.Args) > 1 {
7      fmt.Println(os.Args[1])
8    } else {
9      fmt.Println("Hello, I am Gopher")
10   }
11 }
```

Missing package… where is os?!

Missing os package import, so references are invalid

```
$ go run main.go
```

```
./main.go:6: undefined: os in os.Args
./main.go:7: undefined: os in os.Args
```

The goimports Command

The goimports command ships with Go. It **detects missing packages** and automatically updates import statements in the source code.

The -w flag writes results to the original file instead of printing to the console.

```
$ goimports -w main.go
```

```
package main

import "fmt"

func main() {
  if len(os.Args) > 1 {
    fmt.Println(os.Args[1])
  } else {
    fmt.Println("Hello, Gopher")
  }
}
```

Detects os is being used, but it's not imported

```
package main

import (
  "fmt"
  "os"
)

func main() {
  if len(os.Args) > 1 {
    fmt.Print(os.Args[1])
  } ...
```

Adds missing package and formats code

Running With Fixed Imports

With the necessary packages imported, we can now run our program successfully.

```
package main

import "fmt"

...
```

Fixed imports...

```
$ goimports -w main.go
```

```
package main

import (
  "fmt"
  "os"
)
...
```

Program builds and runs with no errors.

```
$ go run main.go
```

```
Hello, I am Gopher
```

```
$ go run main.go "Into the tunnel"
```

```
Into the tunnel
```