

Programming in Go

Matt Holiday
Christmas 2020



Strings

Strings

Types related to strings:

- `byte`: a synonym for `uint8`
- `rune`: a synonym for `int32` for characters
- `string`: an **immutable** sequence of “characters”
 - physically a sequence of `bytes` (UTF-8 encoding)
 - logically a sequence of (unicode) `runes`

Runes (characters) are enclosed in single quotes: `'a'`

“Raw” strings use backtick quotes: ``string with "quotes"``

They also don't evaluate escape characters such as `\n`

String-related types

Let's see rune vs byte in a string:

```
package main
import "fmt"

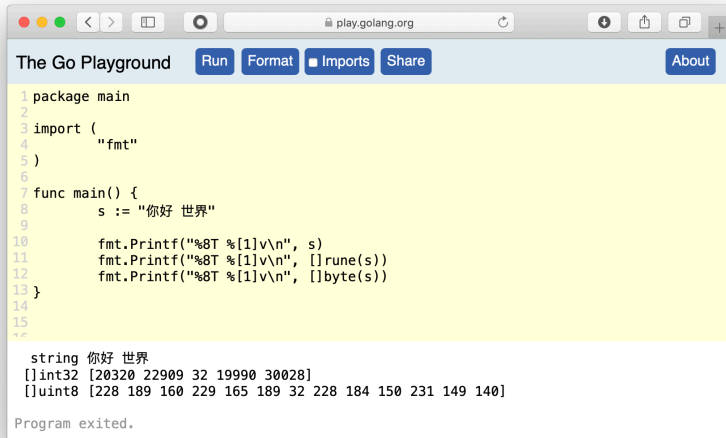
func main() {
    s := "élite"
    fmt.Printf("%8T %[1]v\n", s)
    fmt.Printf("%8T %[1]v\n", []rune(s))
    fmt.Printf("%8T %[1]v\n", []byte(s))
}
```

é is one rune (character) but two bytes in UTF-8 encoding:

```
string élite
[]int32 [233 108 105 116 101]
[]uint8 [195 169 108 105 116 101]
```

String-related types, in Chinese

I can't do this in the slides:



The screenshot shows the Go Playground web interface. The browser address bar displays 'play.golang.org'. The interface includes buttons for 'Run', 'Format', 'Imports', 'Share', and 'About'. The code editor contains the following Go code:

```
1 package main
2
3 import (
4     "fmt"
5 )
6
7 func main() {
8     s := "你好 世界"
9
10    fmt.Printf("%8T %[1]v\n", s)
11    fmt.Printf("%8T %[1]v\n", []rune(s))
12    fmt.Printf("%8T %[1]v\n", []byte(s))
13 }
14
15
```

The output of the program is displayed below the code editor:

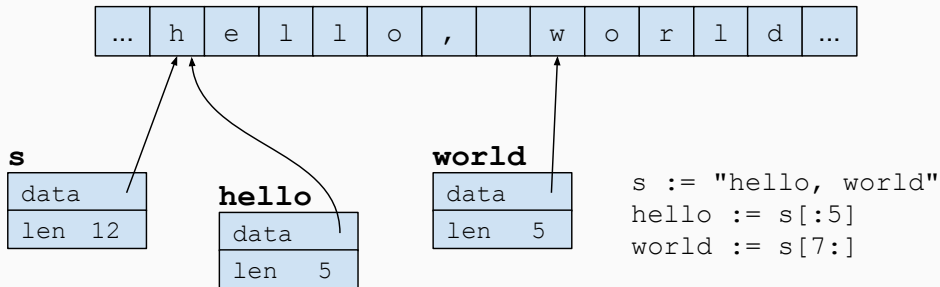
```
string 你好 世界
[]int32 [20320 22909 32 19990 30028]
[]uint8 [228 189 160 229 165 189 32 228 184 150 231 149 140]
```

At the bottom, it says 'Program exited.'

String structure

The internal string representation is a pointer and a length

Strings are **immutable** and can share the underlying storage



Strings

Strings are a sequence of characters and are **immutable**

The built-in `len` function calculates the length

Strings overload the addition operator (+ and +=)

```
s := "the quick brown fox"

a := len(s)           // 19
b := s[:3]            // "the"
c := s[4:9]           // "quick"
d := s[:4] + "slow" + s[9:] // replaces "quick"

s[5] = 'a'            // SYNTAX ERROR
s += "es"             // now plural (copied)
```

Strings are passed *by reference*, thus they aren't copied

String functions

Package strings has many functions on strings

```
s := "a string"

x := len(s)           // built-in, = 8

strings.Contains(s, "g") // returns true
strings.Contains(s, "x") // returns false

strings.HasPrefix(s, "a") // returns true
strings.Index(s, "string") // returns 2

s = strings.ToUpper(s)   // returns "A STRING"
```

Note that we assign the result of ToUpper back to s