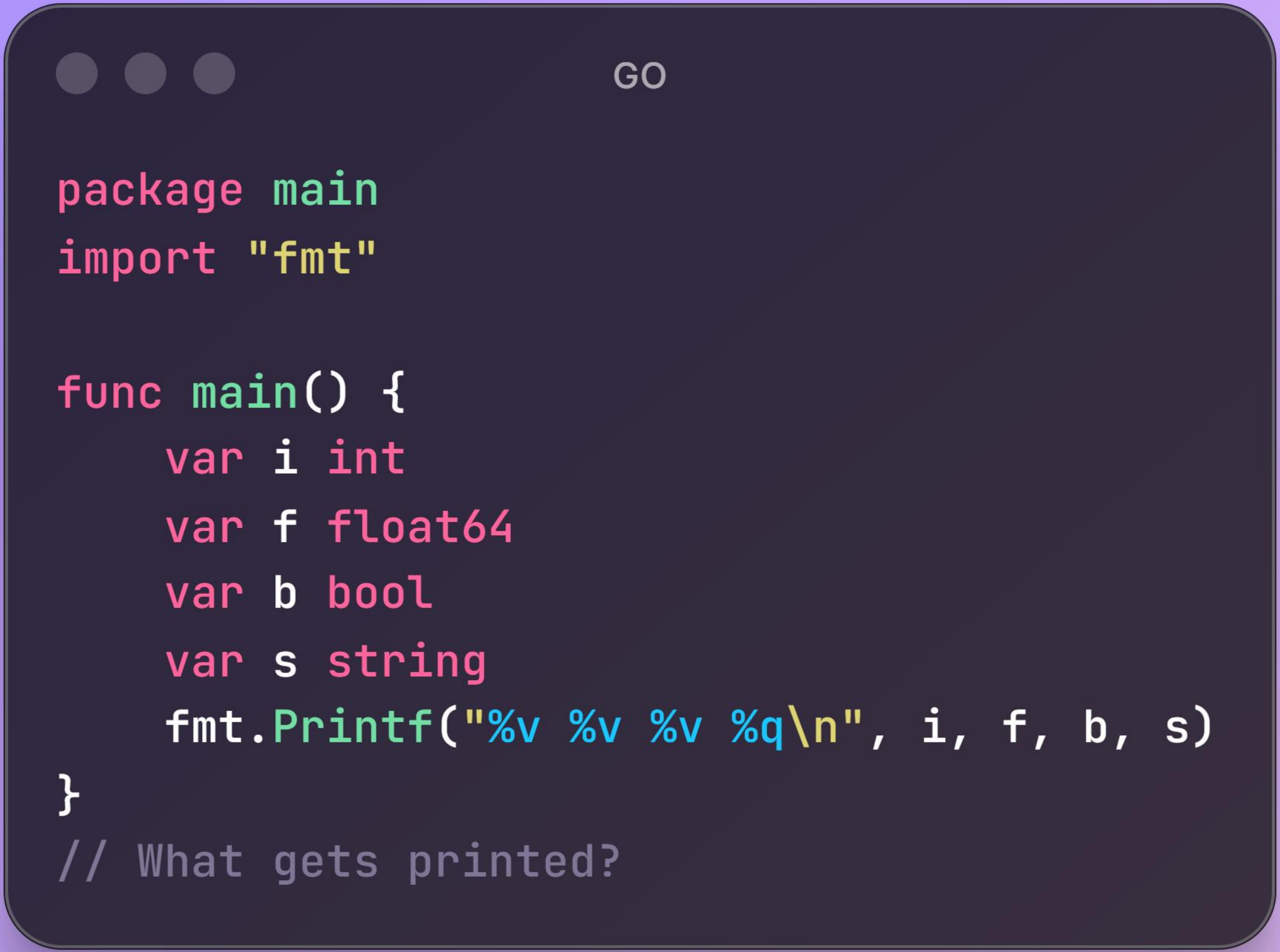
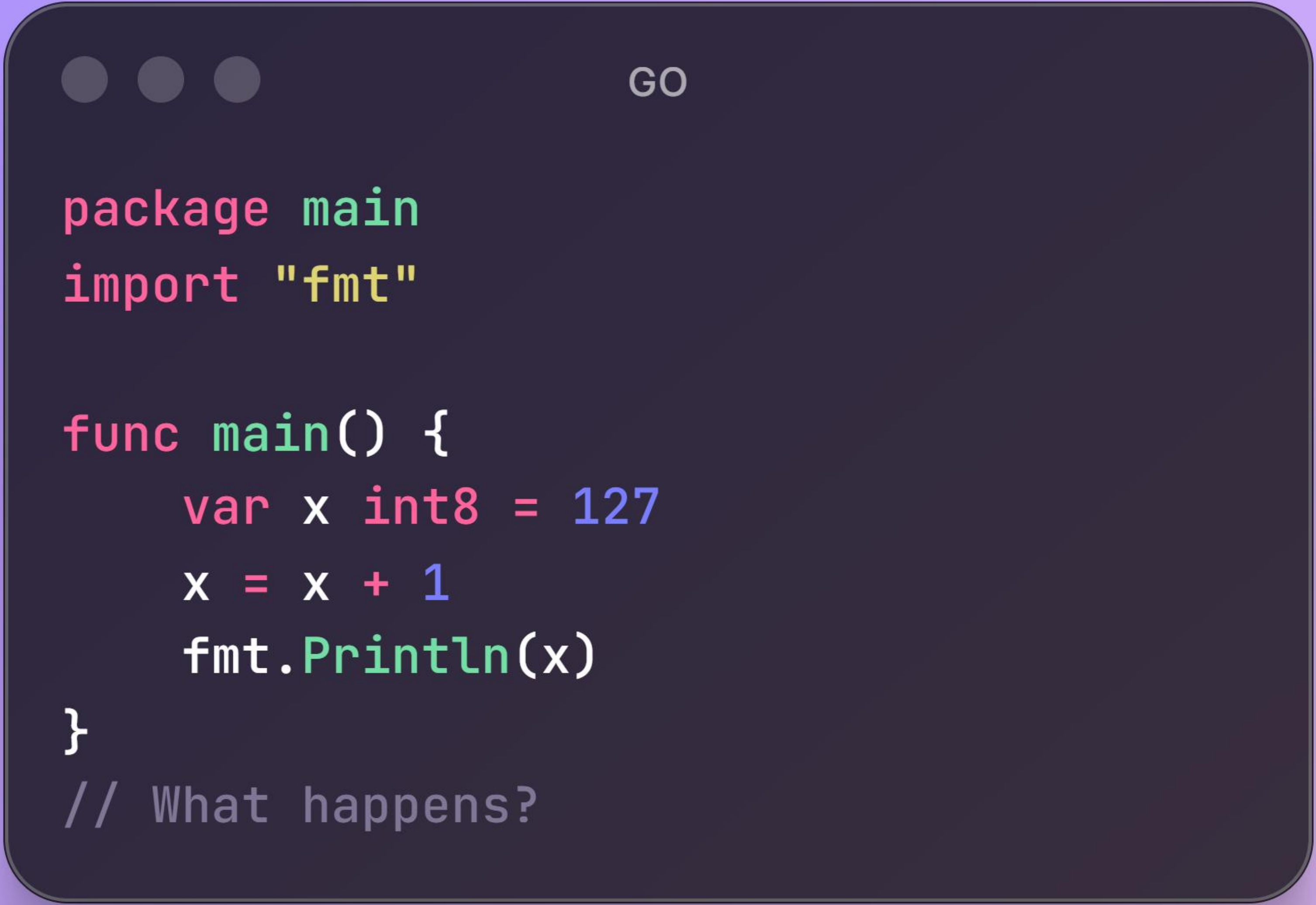
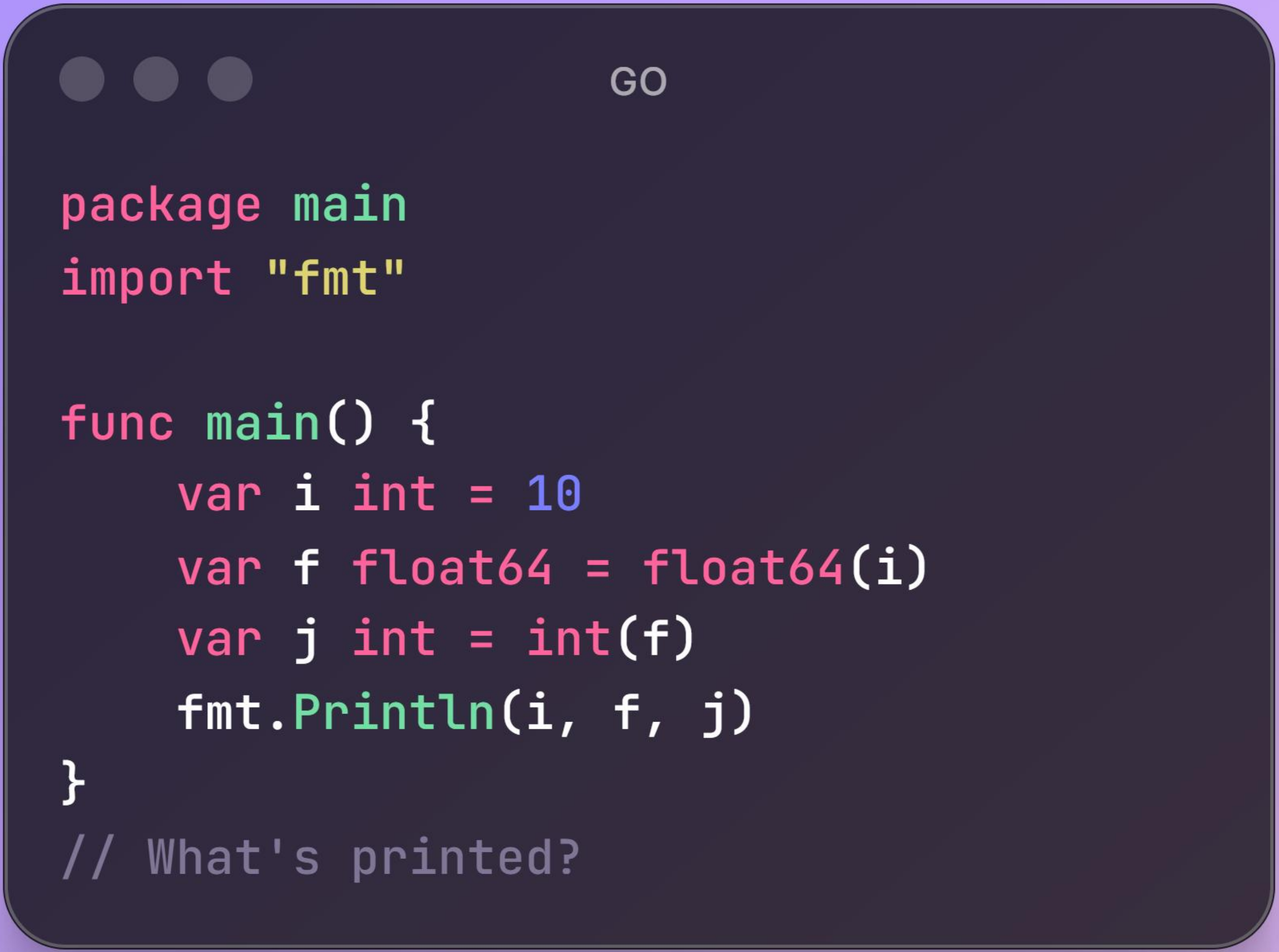

GO

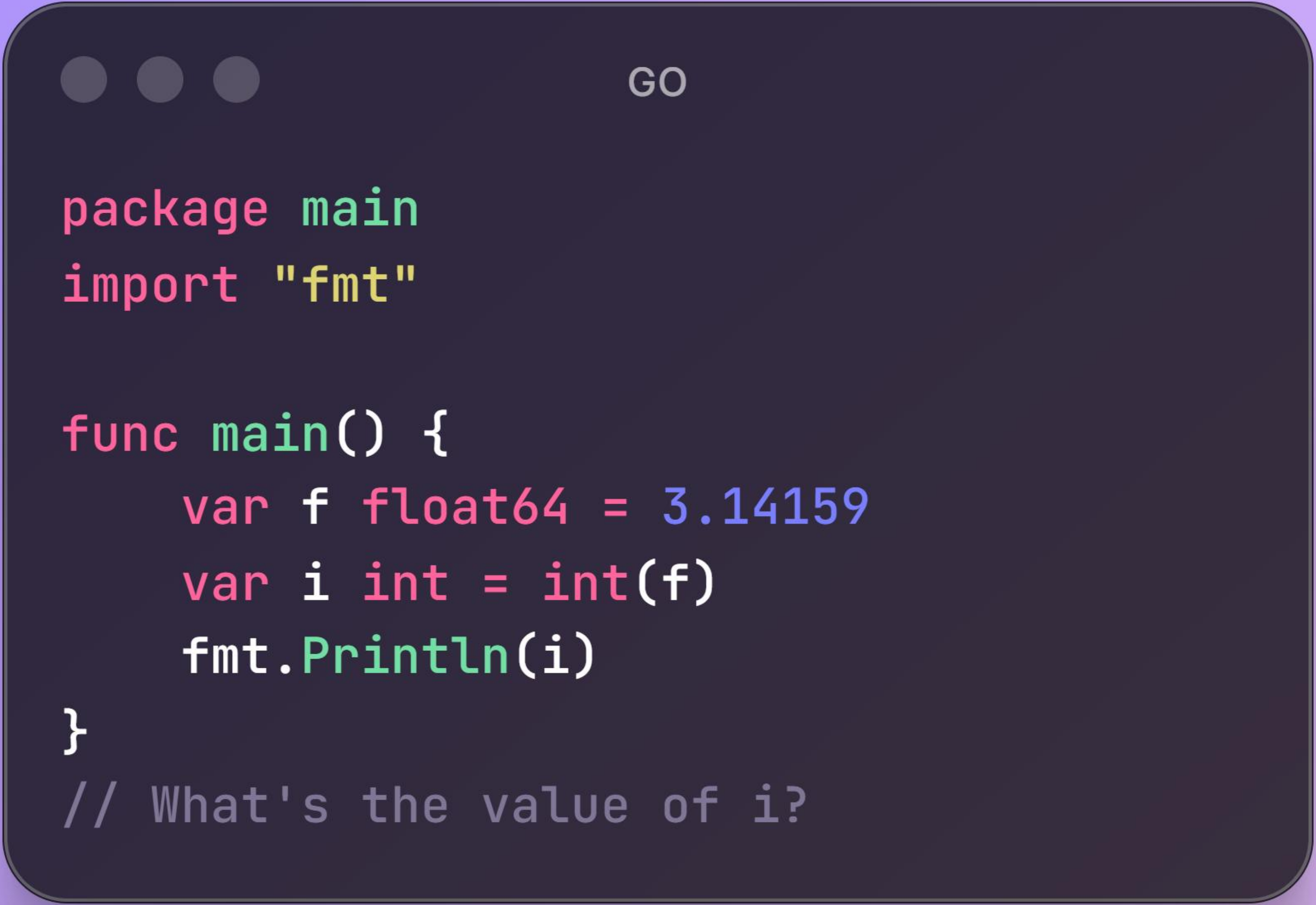
```go
package main
import "fmt"

func main() {
    var i int
    var f float64
    var b bool
    var s string
    fmt.Printf("%v %v %v %q\n", i, f, b, s)
}
// What gets printed?
```



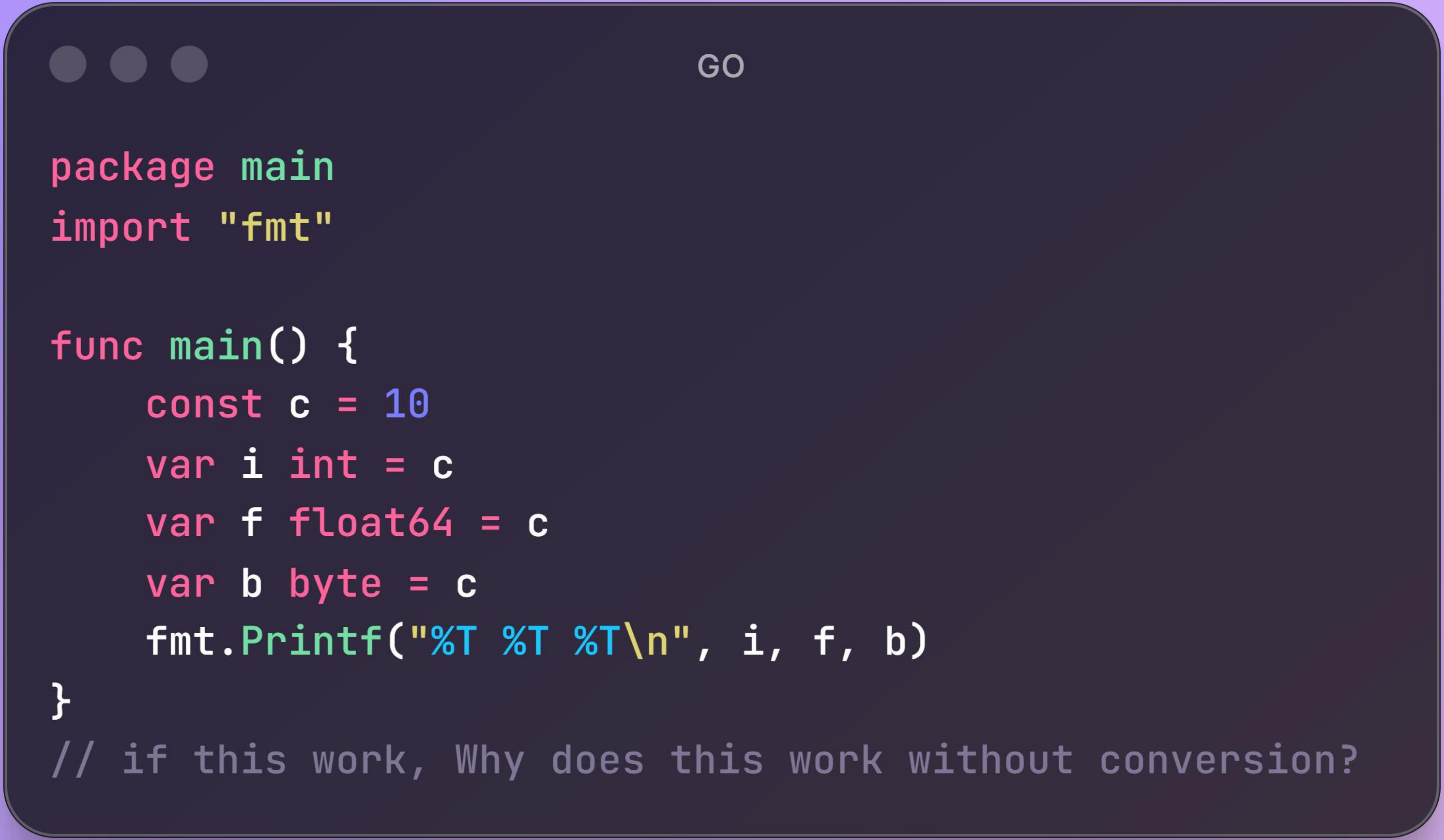
```go
package main
import "fmt"

func main() {
    var x int8 = 127
    x = x + 1
    fmt.Println(x)
}
// What happens?
```



```go
package main
import "fmt"

func main() {
    var i int = 10
    var f float64 = float64(i)
    var j int = int(f)
    fmt.Println(i, f, j)
}
// What's printed?
```

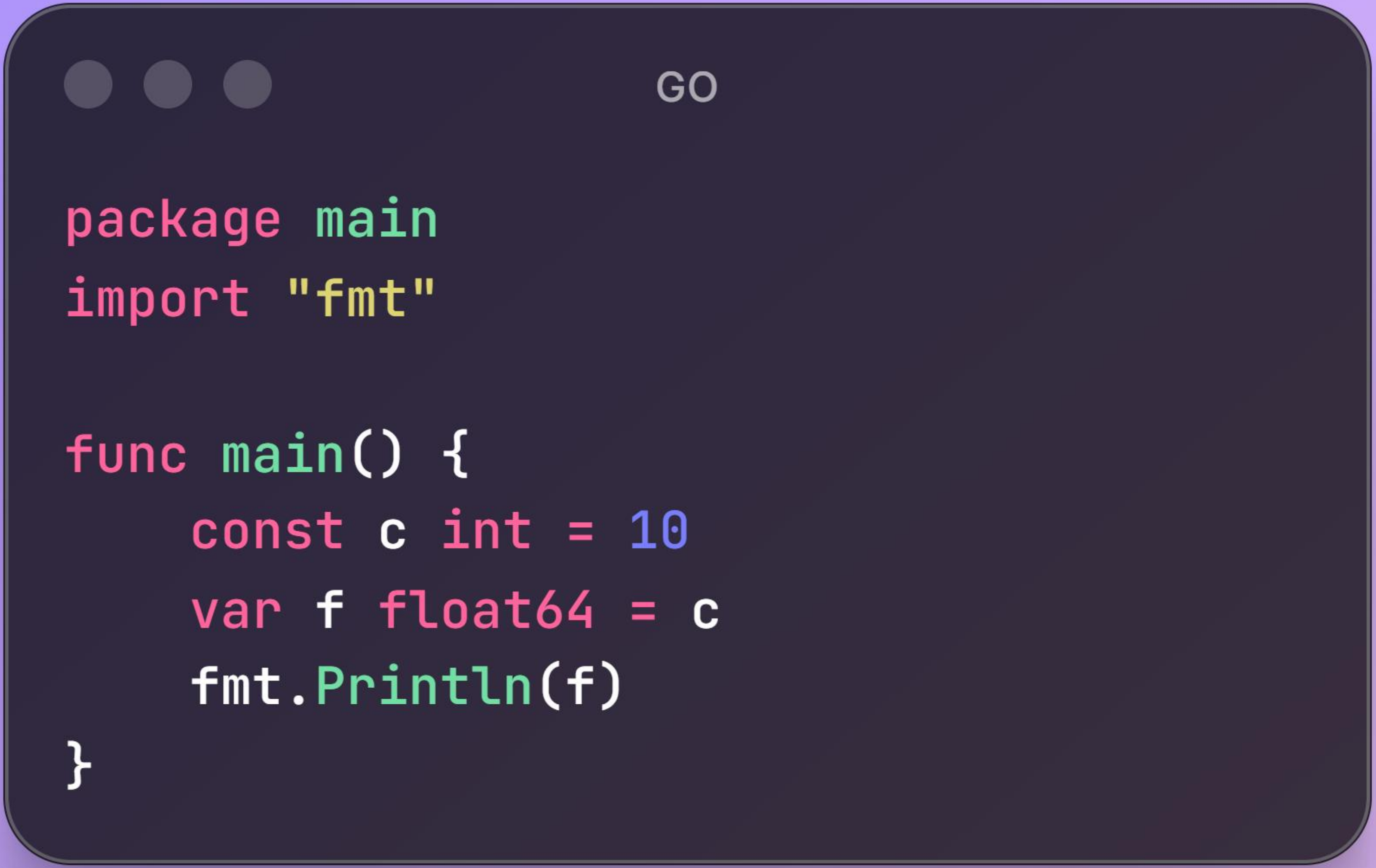

GO

```go
package main
import "fmt"

func main() {
    var f float64 = 3.14159
    var i int = int(f)
    fmt.Println(i)
}
// What's the value of i?
```

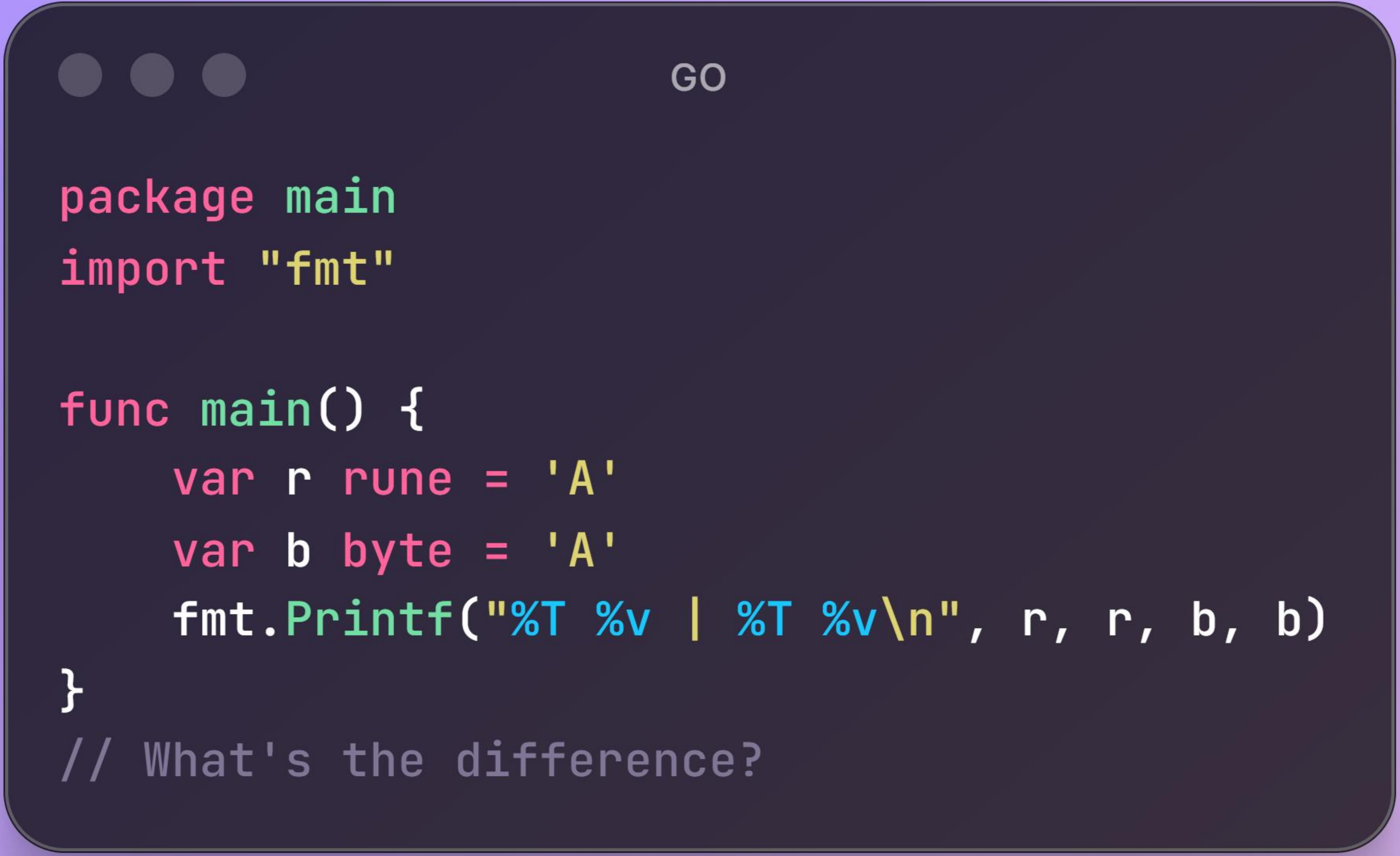


GO

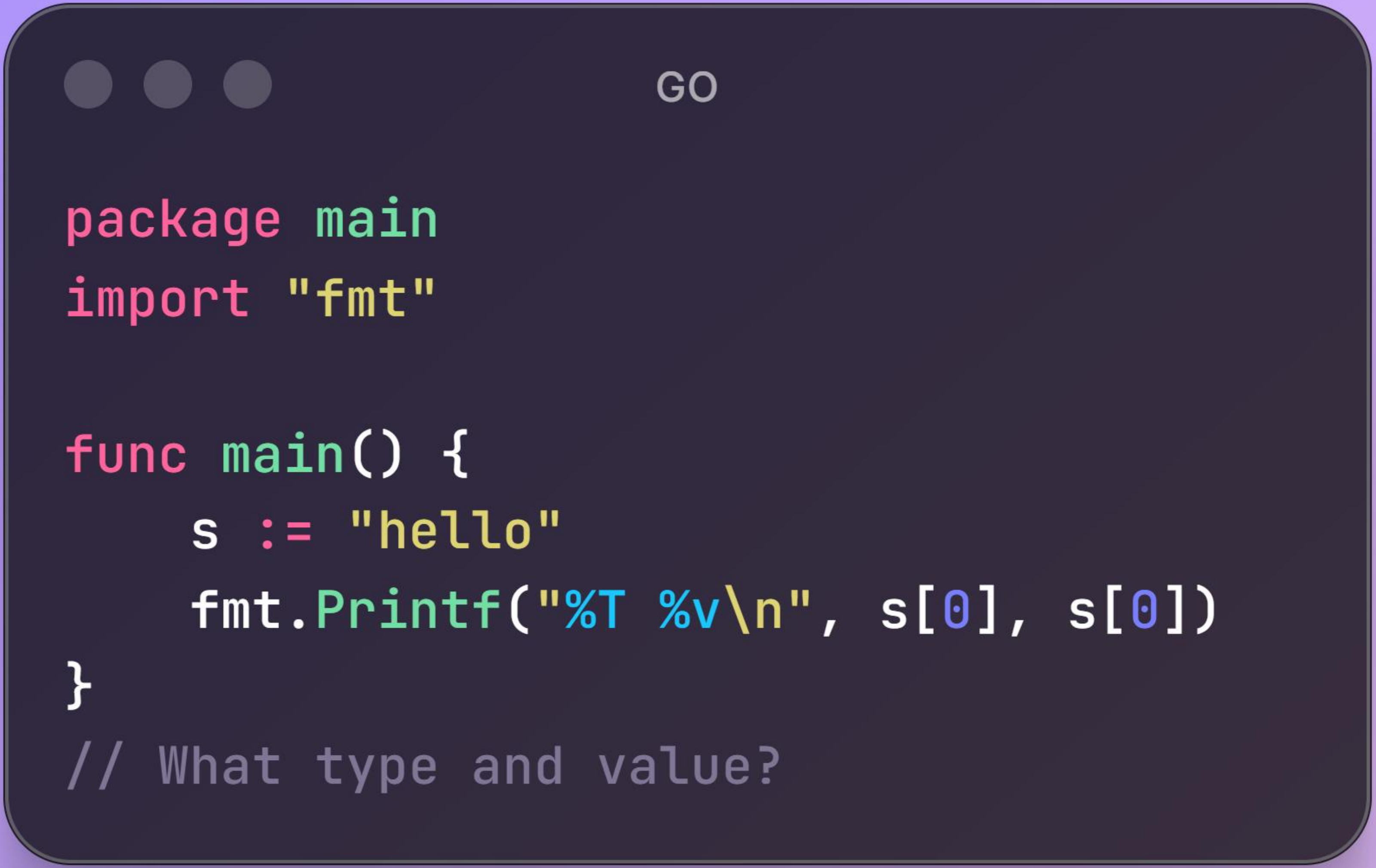
```go
package main
import "fmt"

func main() {
    const c = 10
    var i int = c
    var f float64 = c
    var b byte = c
    fmt.Printf("%T %T %T\n", i, f, b)
}
// if this work, Why does this work without conversion?
```



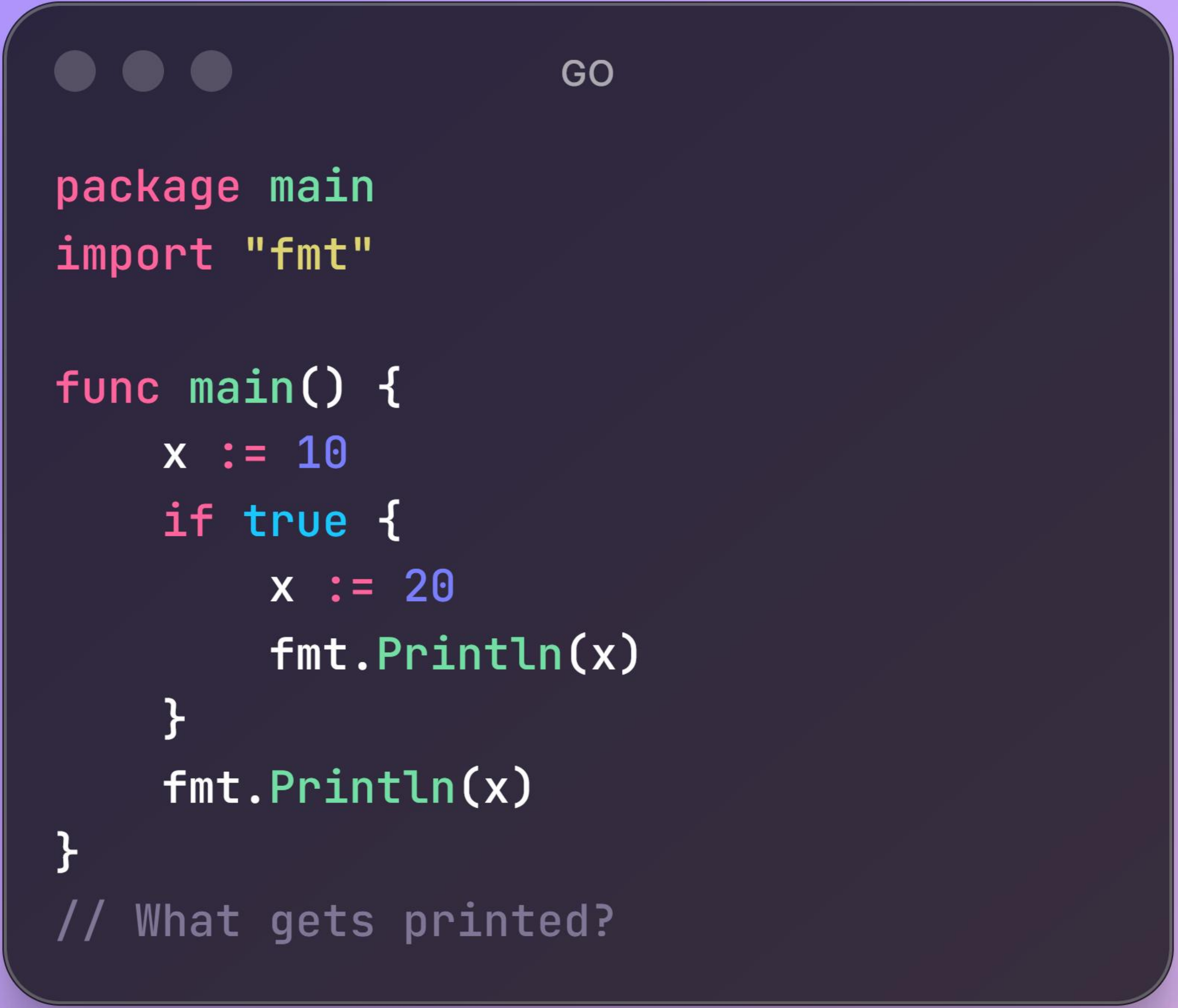
```go
package main
import "fmt"

func main() {
    const c int = 10
    var f float64 = c
    fmt.Println(f)
}
```


GO

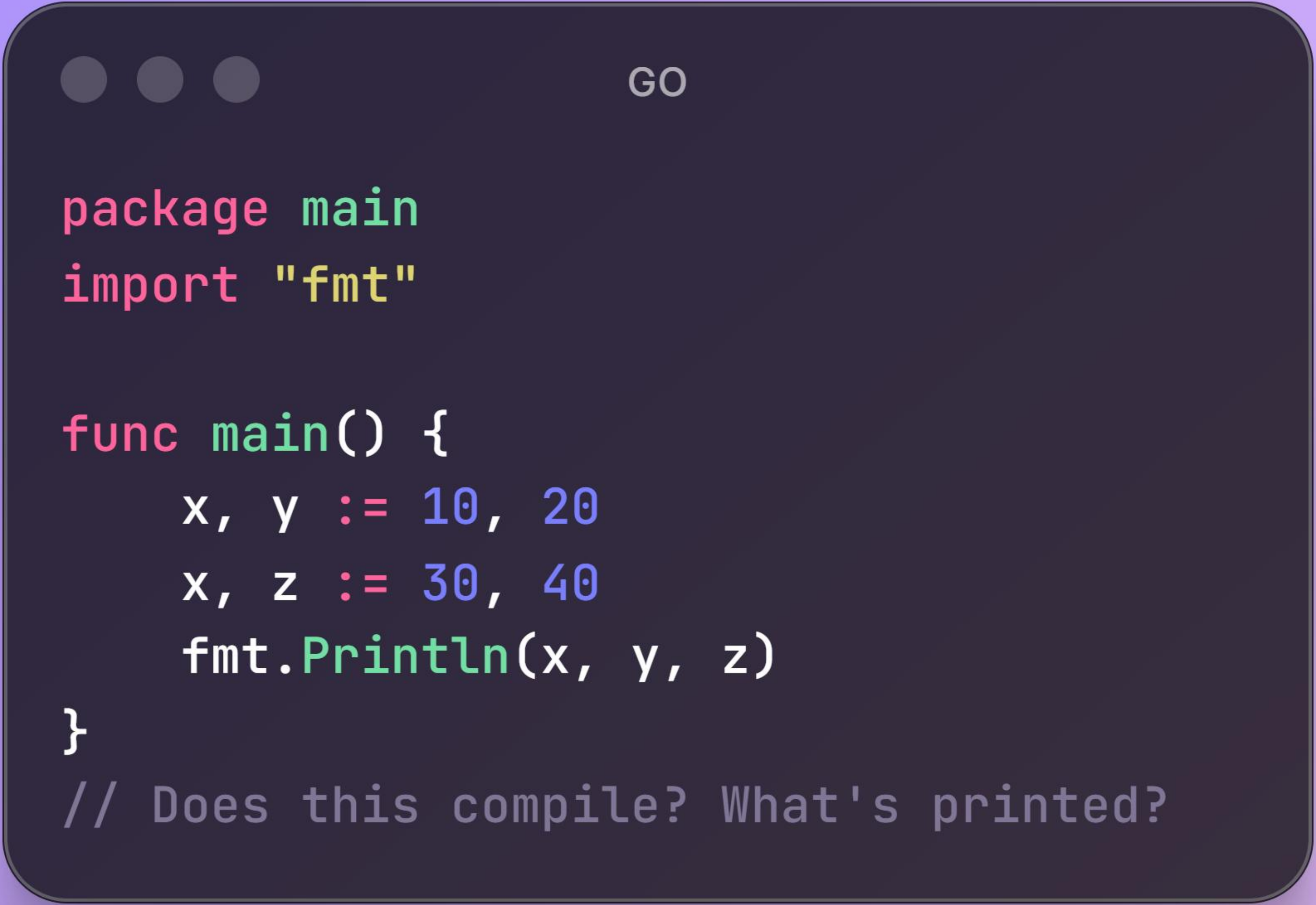
```go
package main
import "fmt"

func main() {
    var r rune = 'A'
    var b byte = 'A'
    fmt.Printf("%T %v | %T %v\n", r, r, b, b)
}
// What's the difference?
```



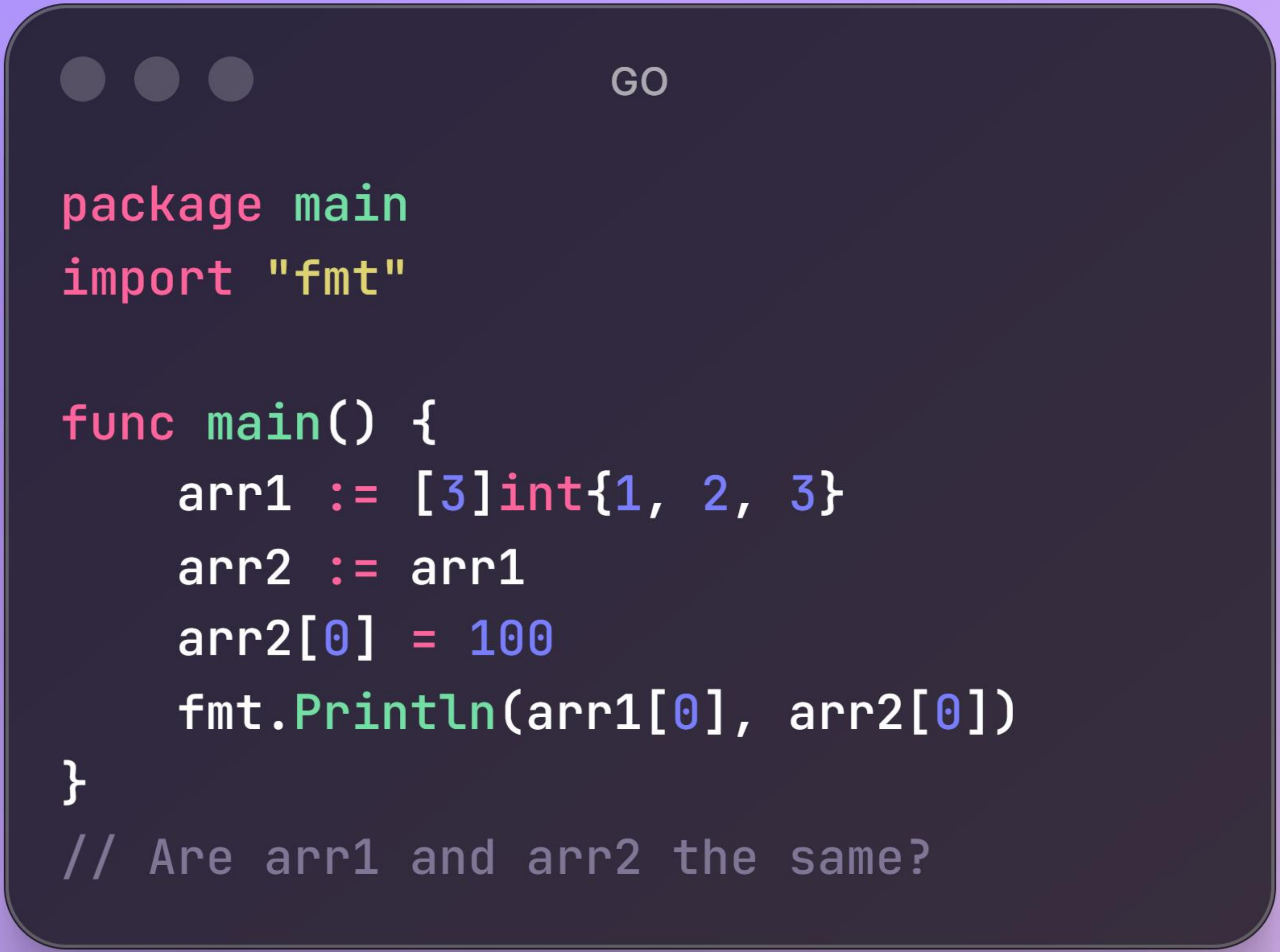
```go
package main
import "fmt"

func main() {
    s := "hello"
    fmt.Printf("%T %v\n", s[0], s[0])
}
// What type and value?
```



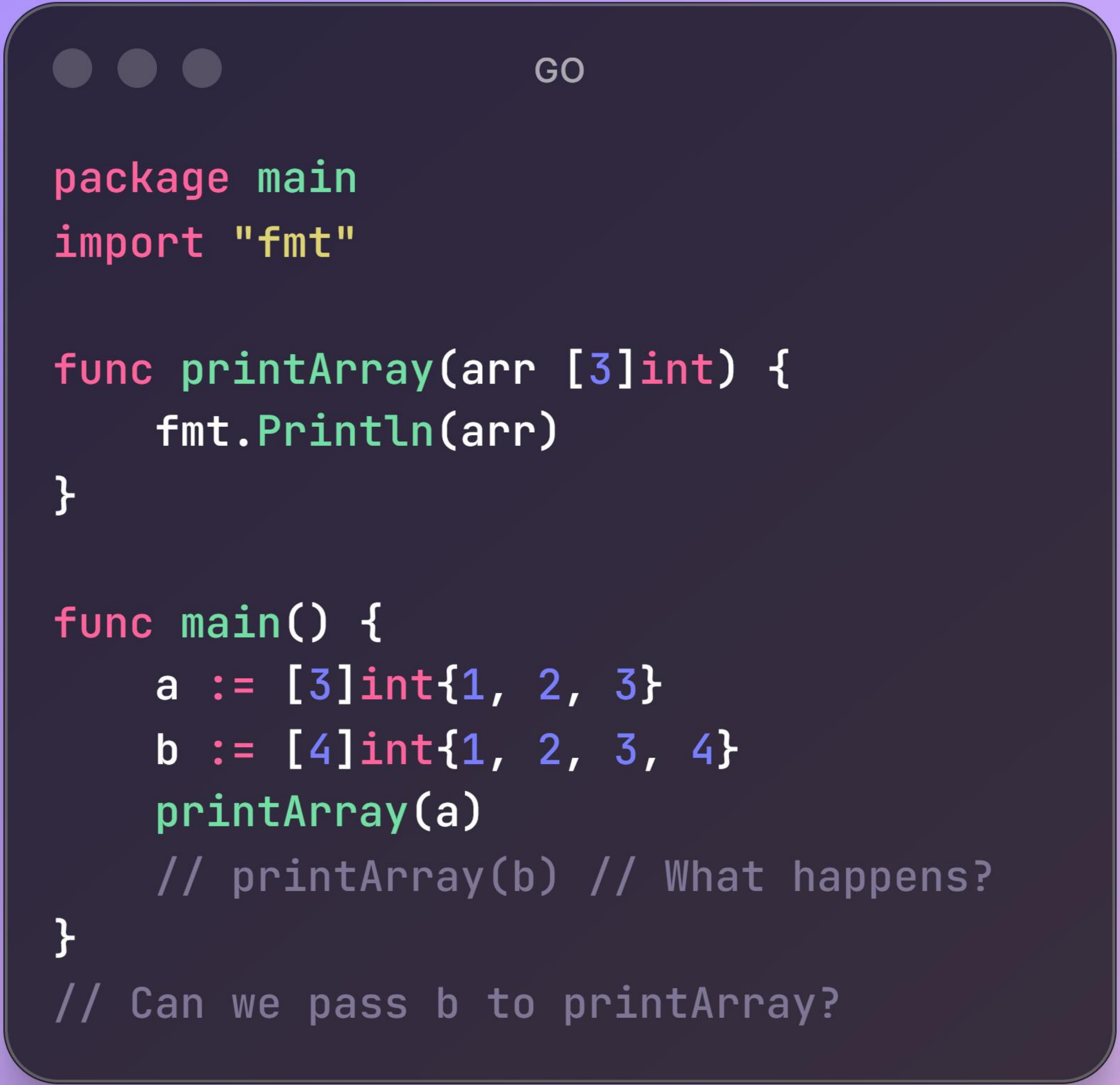
```go
package main
import "fmt"

func main() {
    x := 10
    if true {
        x := 20
        fmt.Println(x)
    }
    fmt.Println(x)
}
// What gets printed?
```



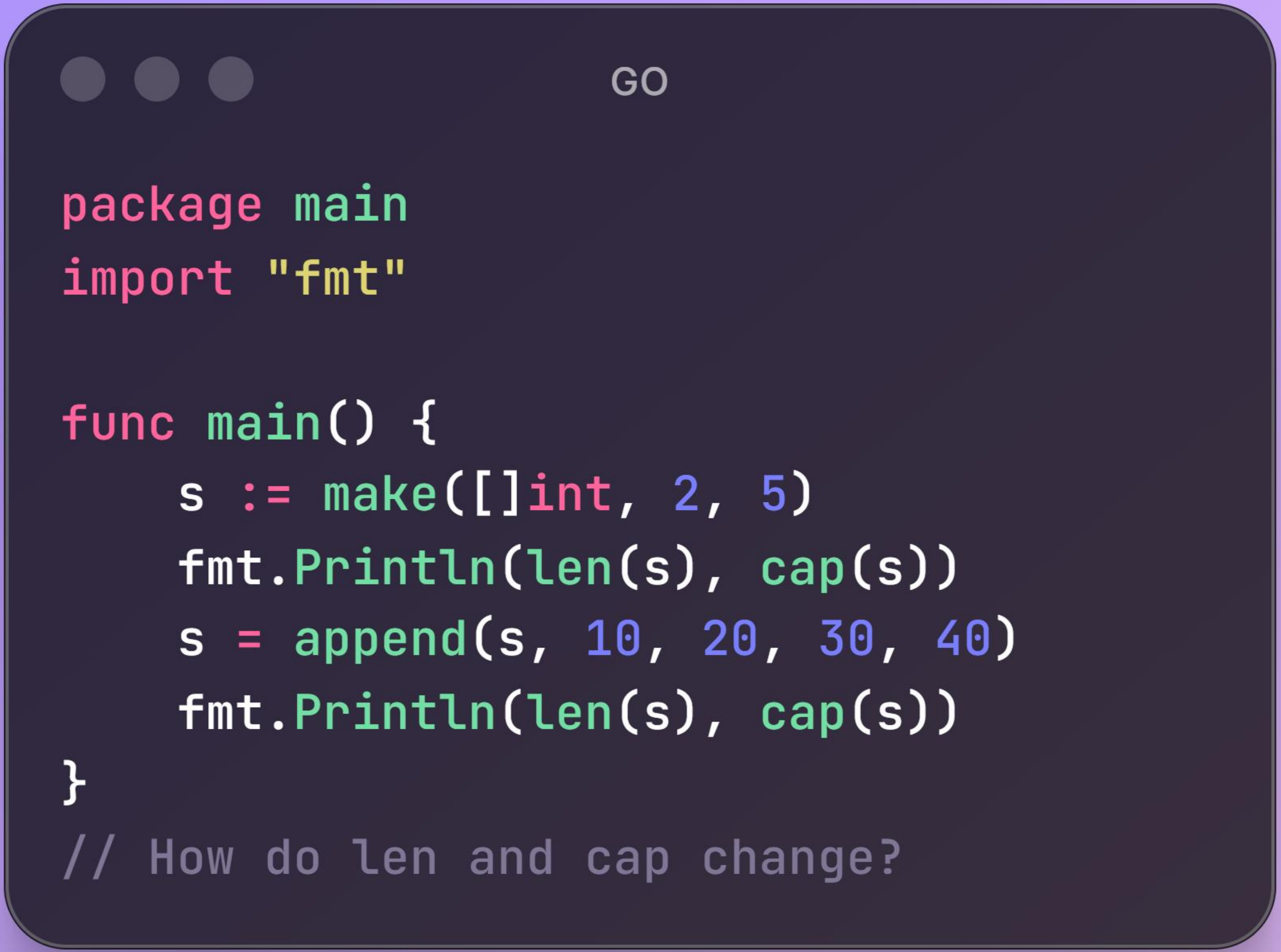
```go
package main
import "fmt"

func main() {
    x, y := 10, 20
    x, z := 30, 40
    fmt.Println(x, y, z)
}
// Does this compile? What's printed?
```



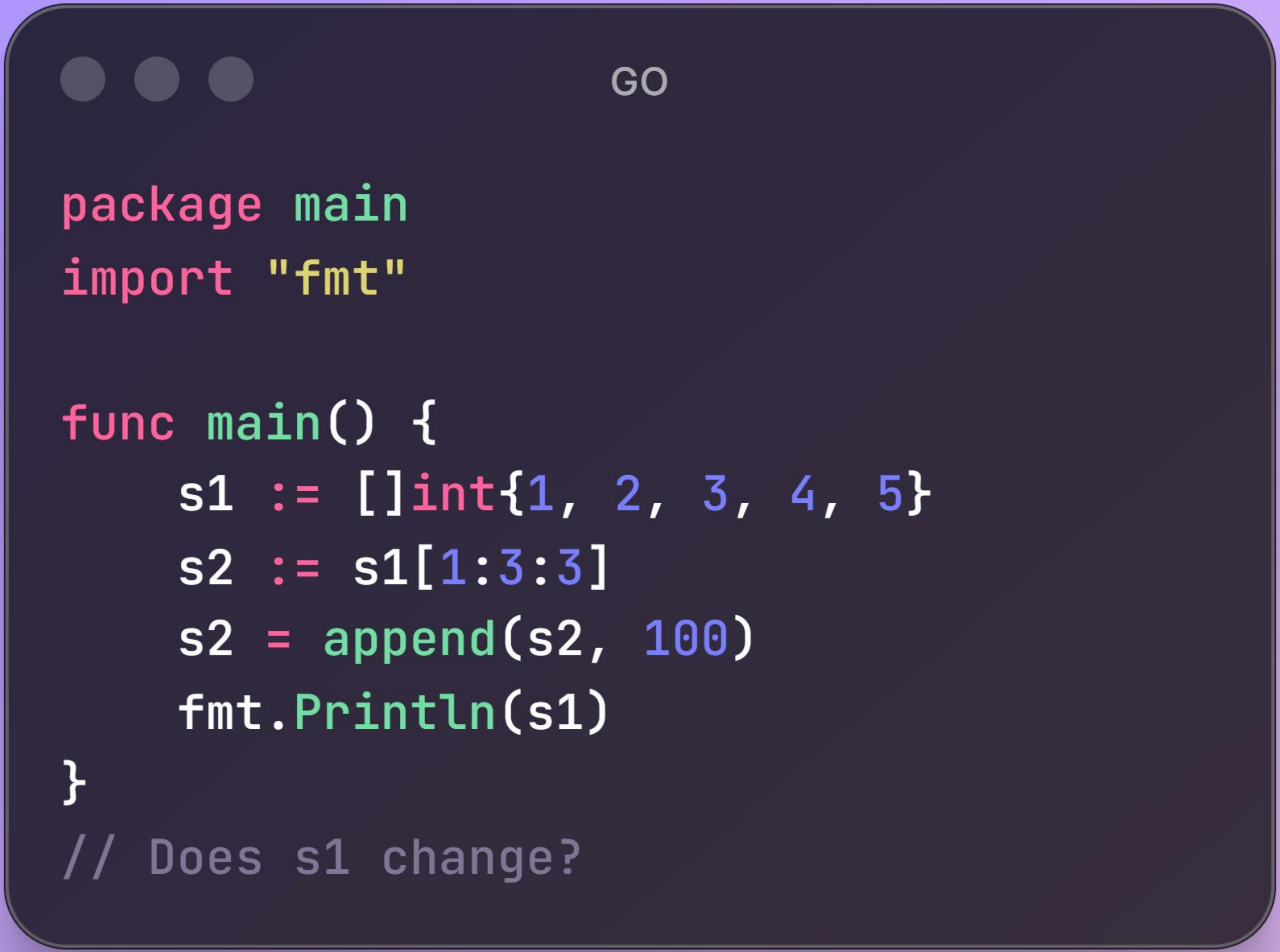
```go
package main
import "fmt"

func main() {
    arr1 := [3]int{1, 2, 3}
    arr2 := arr1
    arr2[0] = 100
    fmt.Println(arr1[0], arr2[0])
}
// Are arr1 and arr2 the same?
```



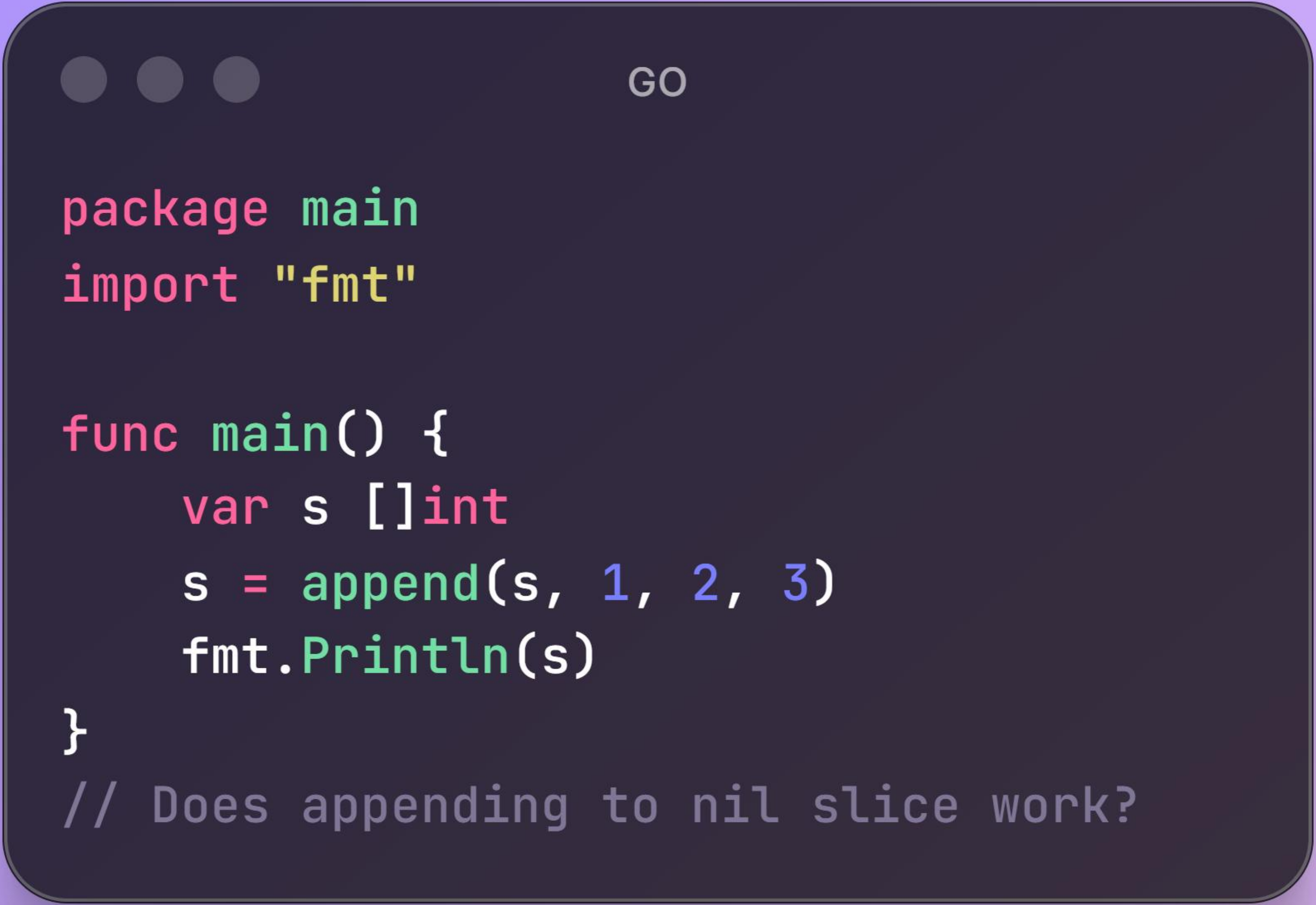
```go
package main
import "fmt"

func printArray(arr [3]int) {
    fmt.Println(arr)
}

func main() {
    a := [3]int{1, 2, 3}
    b := [4]int{1, 2, 3, 4}
    printArray(a)
    // printArray(b) // What happens?
}
// Can we pass b to printArray?
```



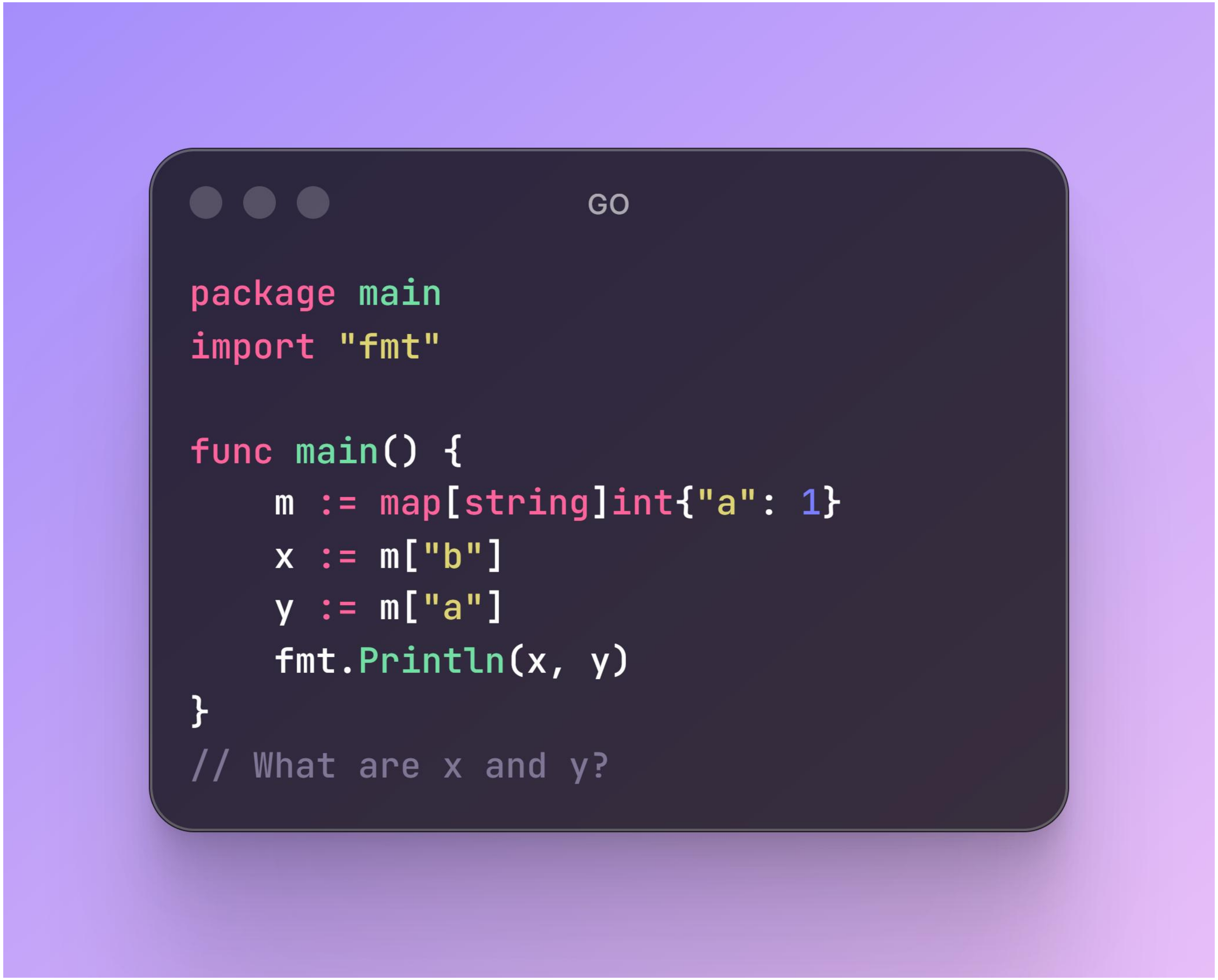
```go
package main
import "fmt"

func main() {
    s := make([]int, 2, 5)
    fmt.Println(len(s), cap(s))
    s = append(s, 10, 20, 30, 40)
    fmt.Println(len(s), cap(s))
}
// How do len and cap change?
```



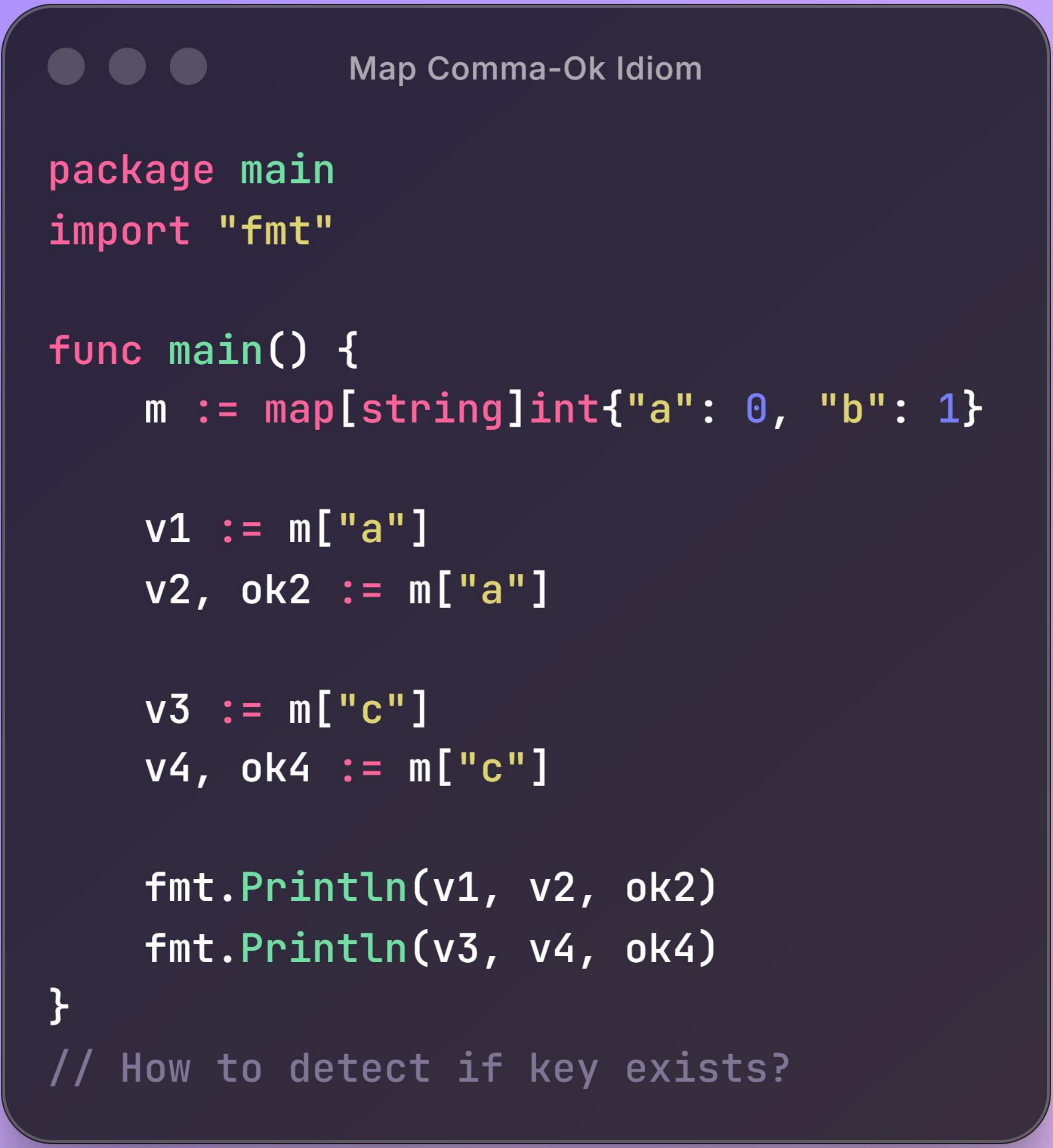
```go
package main
import "fmt"

func main() {
    s1 := []int{1, 2, 3, 4, 5}
    s2 := s1[1:3:3]
    s2 = append(s2, 100)
    fmt.Println(s1)
}
// Does s1 change?
```



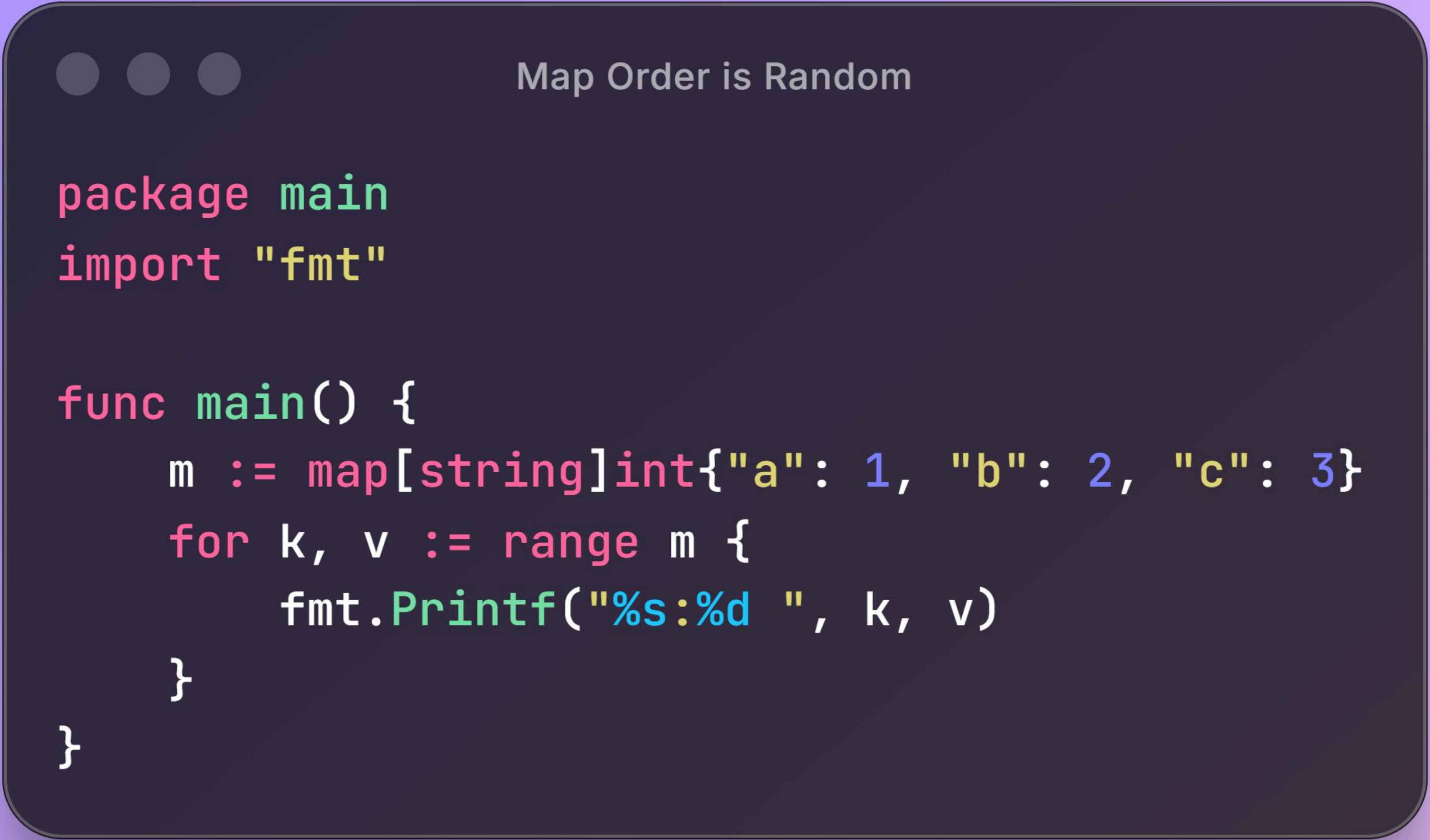
```go
package main
import "fmt"

func main() {
    var s []int
    s = append(s, 1, 2, 3)
    fmt.Println(s)
}
// Does appending to nil slice work?
```



```go
package main
import "fmt"

func main() {
    m := map[string]int{"a": 1}
    x := m["b"]
    y := m["a"]
    fmt.Println(x, y)
}
// What are x and y?
```

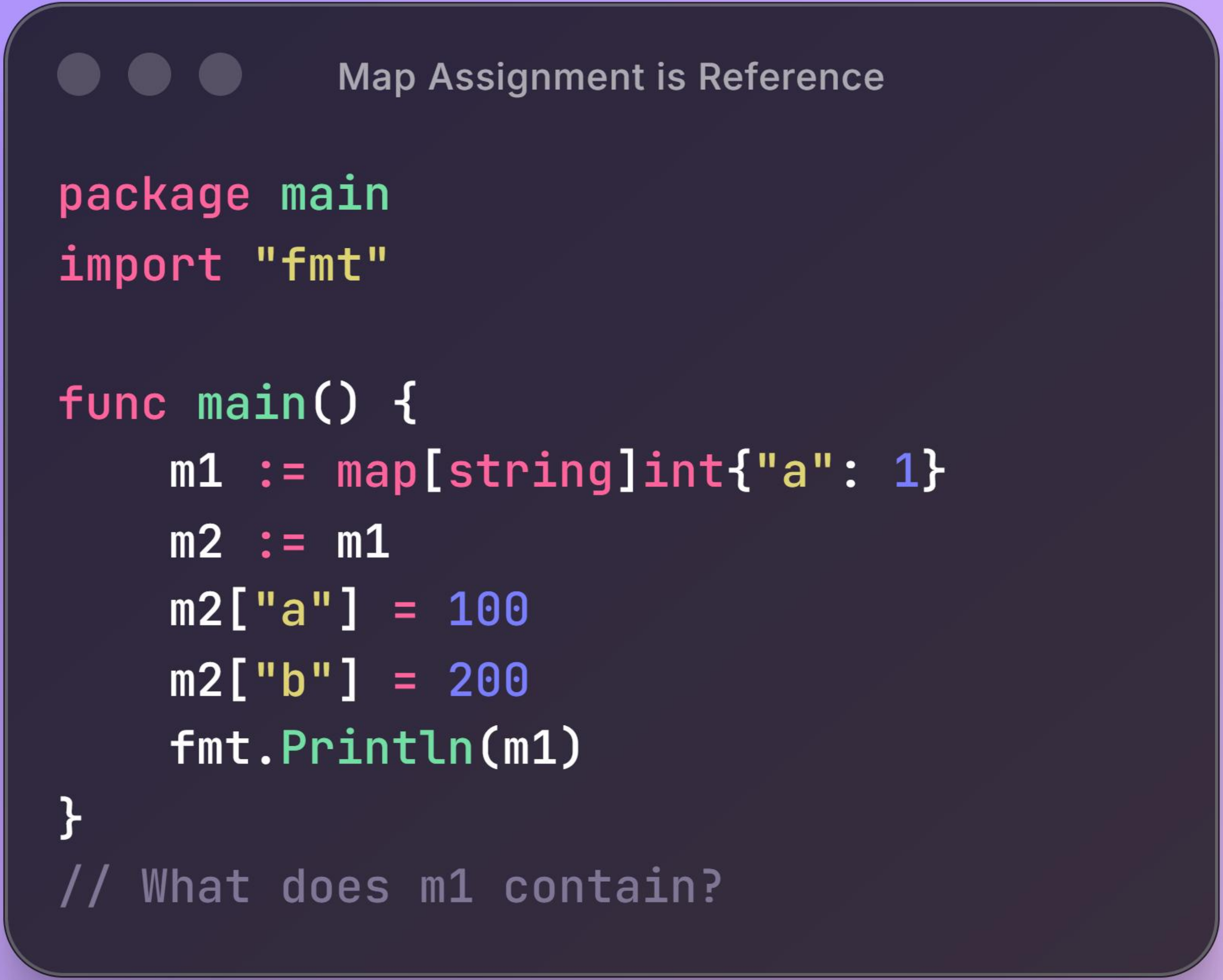


```go
package main
import "fmt"

func main() {
    m := map[string]int{"a": 0, "b": 1}

    v1 := m["a"]
    v2, ok2 := m["a"]

    v3 := m["c"]
    v4, ok4 := m["c"]

    fmt.Println(v1, v2, ok2)
    fmt.Println(v3, v4, ok4)
}
// How to detect if key exists?
```



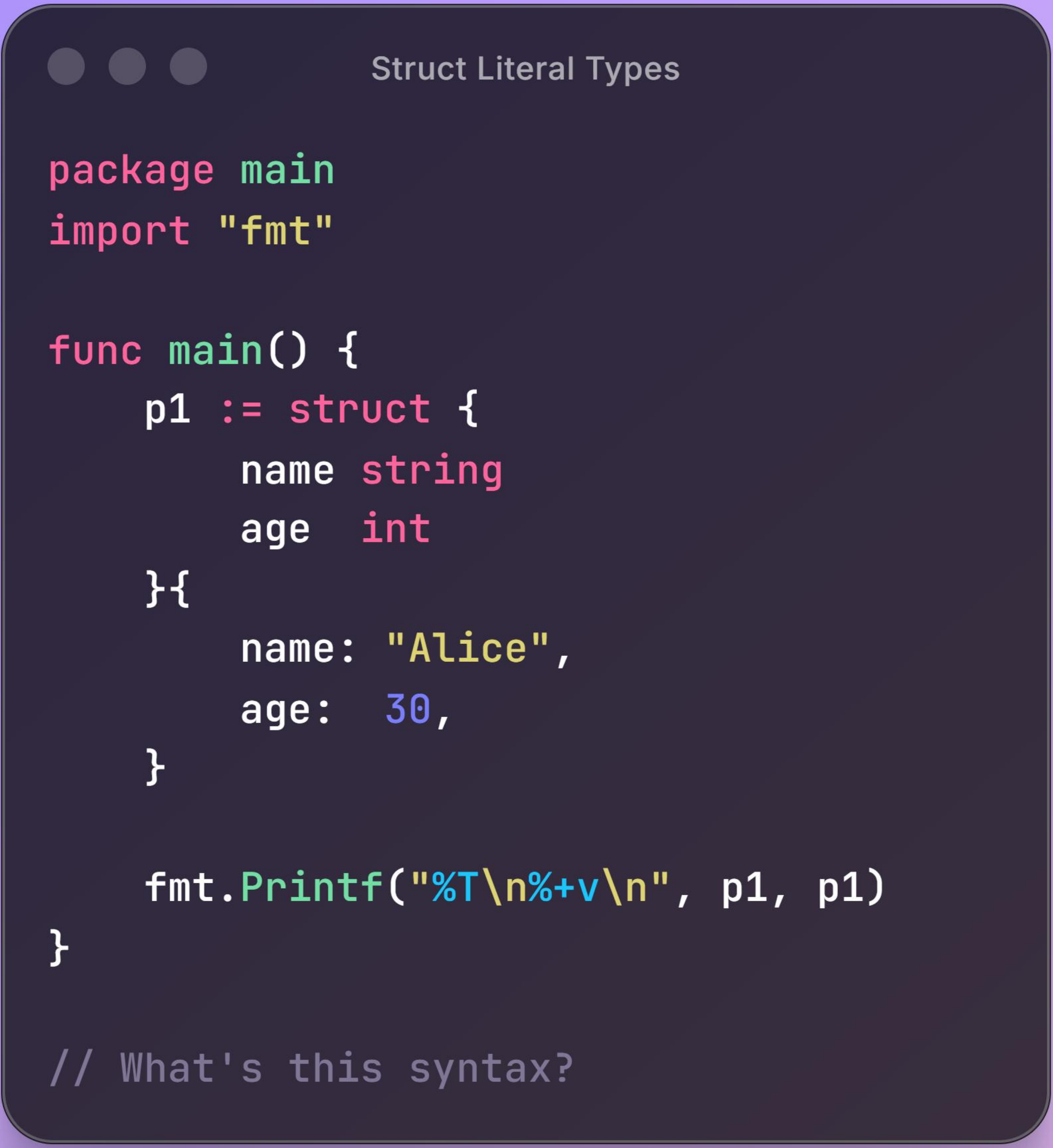
Map Order is Random

```go
package main
import "fmt"

func main() {
    m := map[string]int{"a": 1, "b": 2, "c": 3}
    for k, v := range m {
        fmt.Printf("%s:%d ", k, v)
    }
}
```



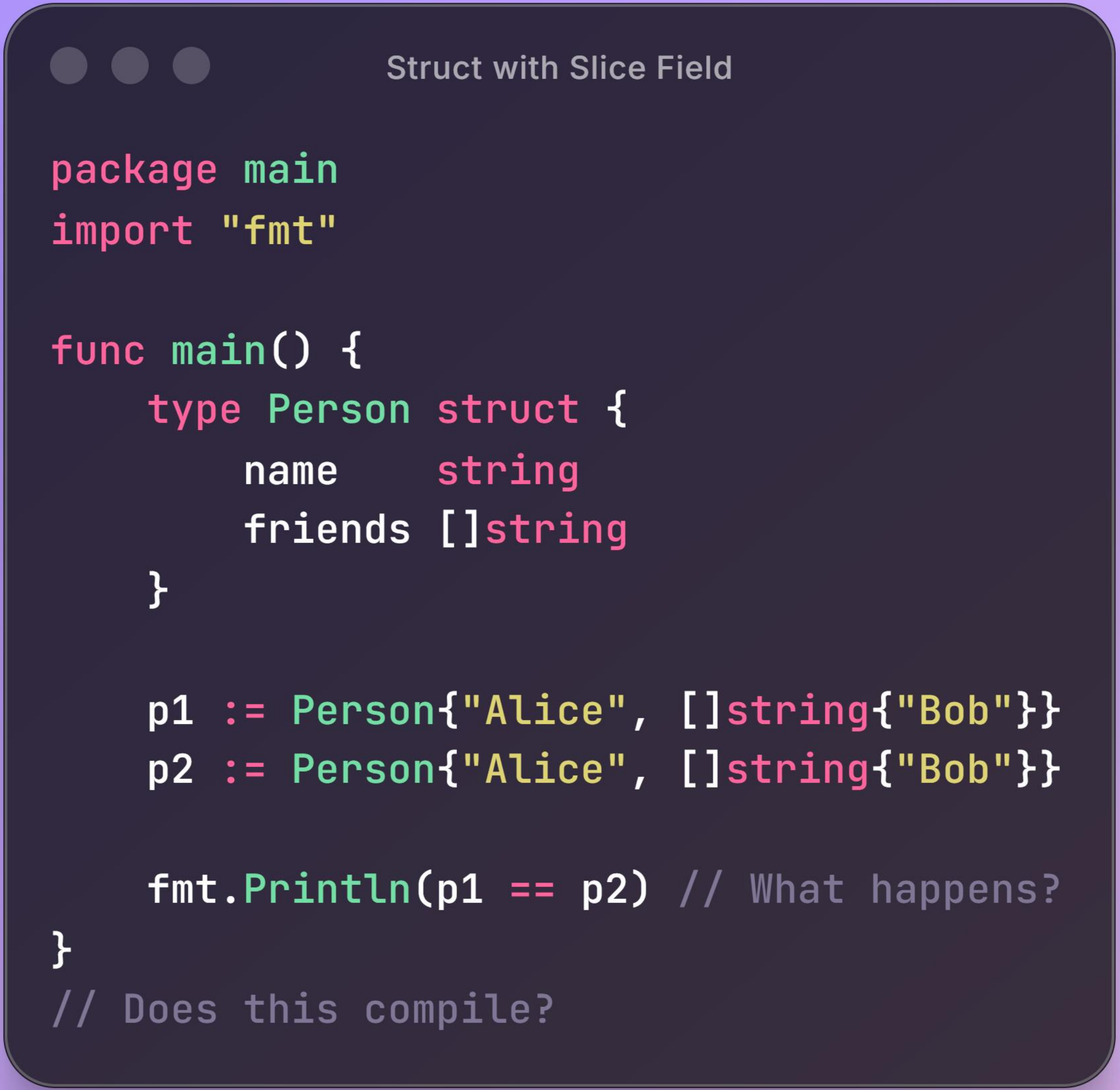
Map Assignment is Reference

```go
package main
import "fmt"

func main() {
    m1 := map[string]int{"a": 1}
    m2 := m1
    m2["a"] = 100
    m2["b"] = 200
    fmt.Println(m1)
}
// What does m1 contain?
```

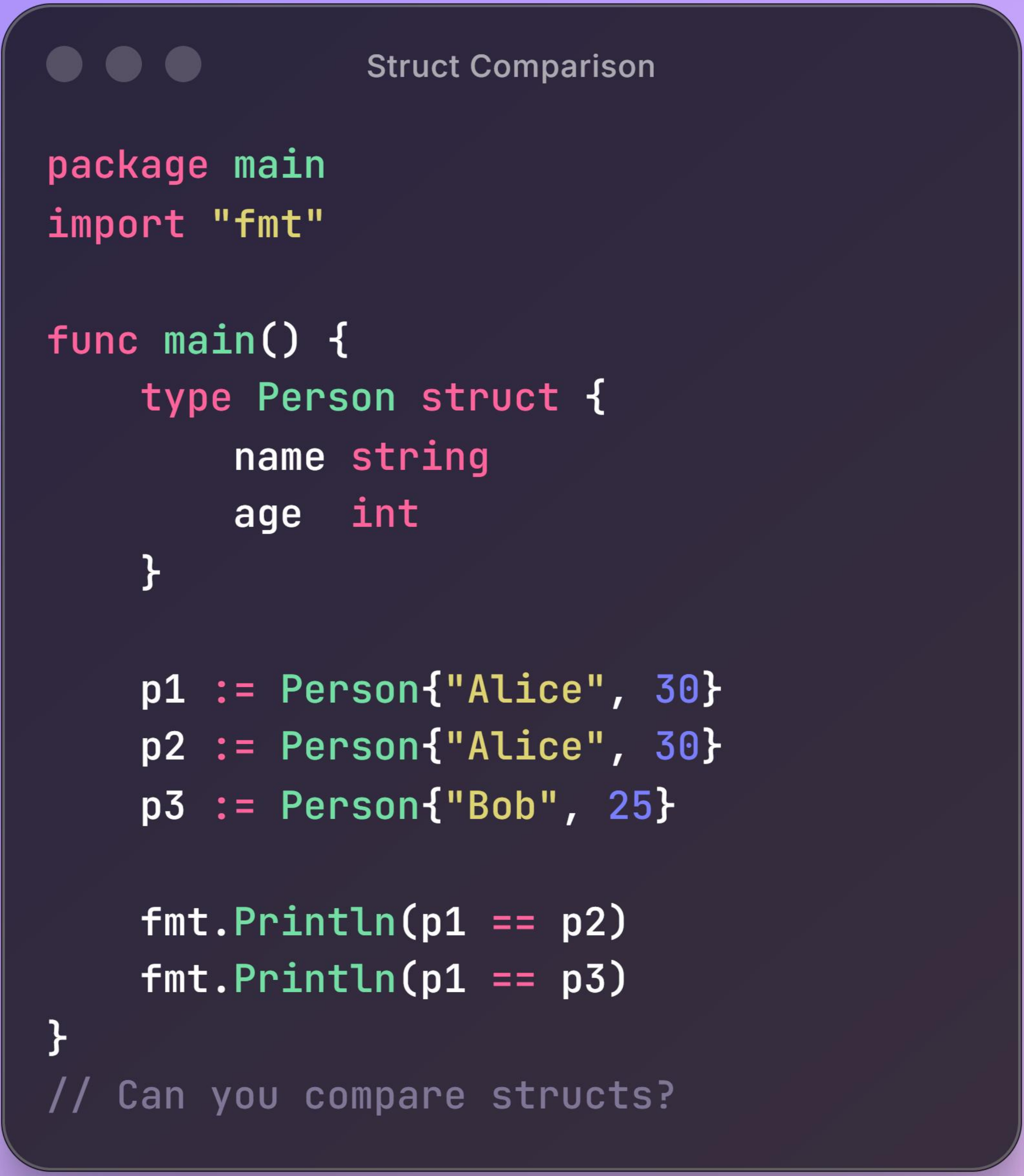


Struct Literal Types

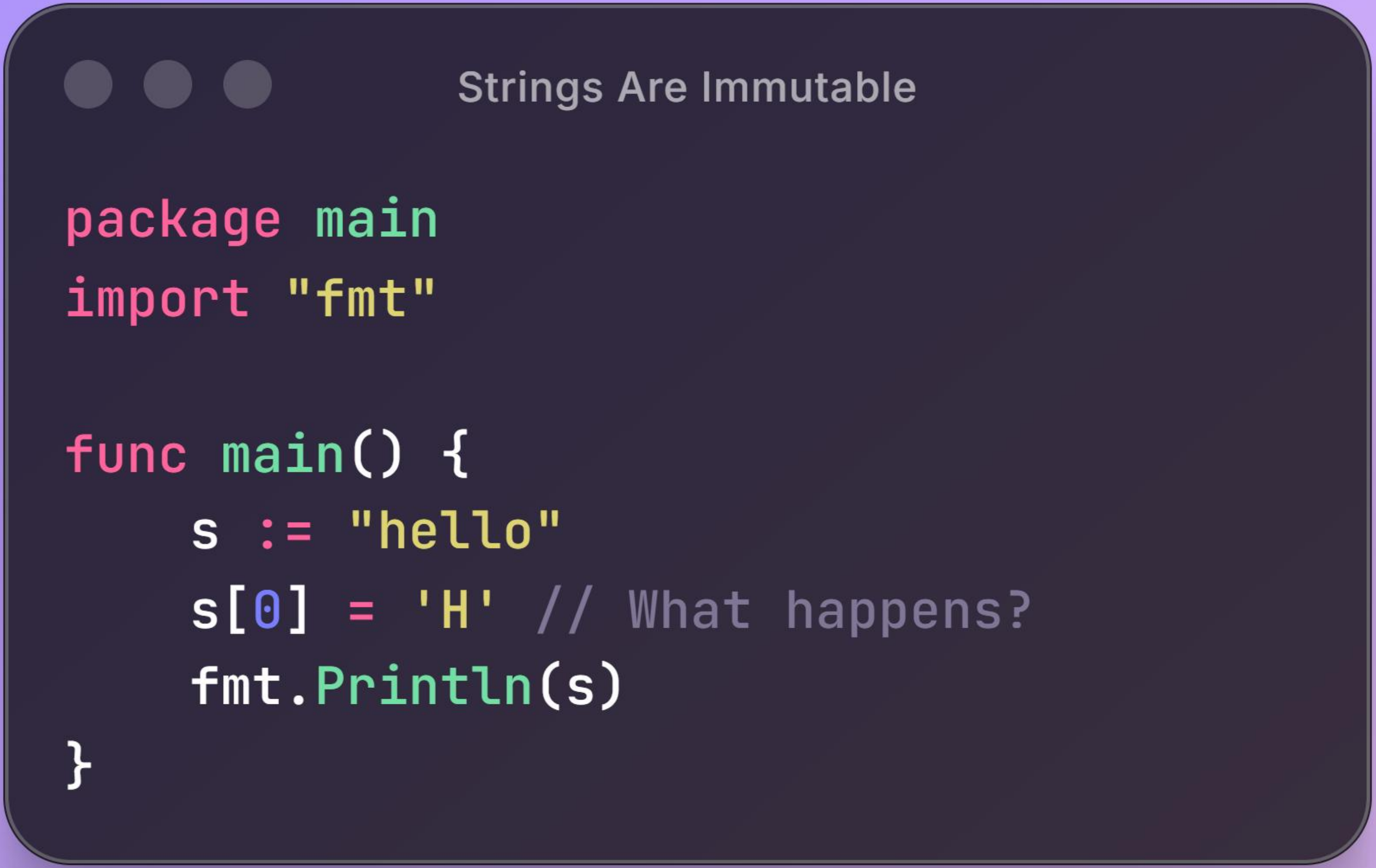
```go
package main
import "fmt"

func main() {
    p1 := struct {
        name string
        age  int
    }{
        name: "Alice",
        age:  30,
    }

    fmt.Printf("%T\n%+v\n", p1, p1)
}

// What's this syntax?
```


Struct with Slice Field

```go
package main
import "fmt"

func main() {
    type Person struct {
        name    string
        friends []string
    }

    p1 := Person{"Alice", []string{"Bob"}}
    p2 := Person{"Alice", []string{"Bob"}}

    fmt.Println(p1 == p2) // What happens?
}
// Does this compile?
```

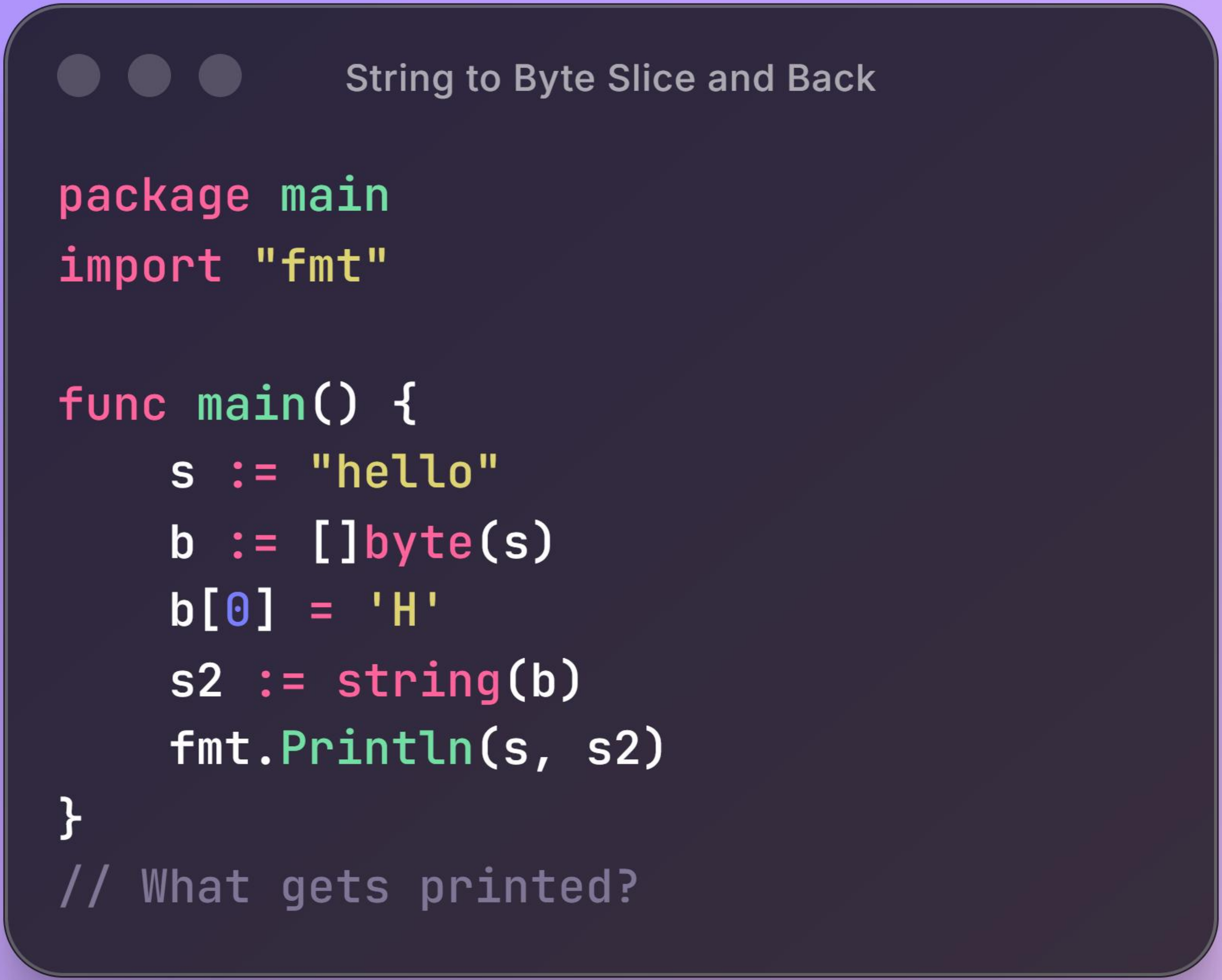


```go
package main
import "fmt"

func main() {
    type Person struct {
        name string
        age  int
    }

    p1 := Person{"Alice", 30}
    p2 := Person{"Alice", 30}
    p3 := Person{"Bob", 25}

    fmt.Println(p1 == p2)
    fmt.Println(p1 == p3)
}
// Can you compare structs?
```



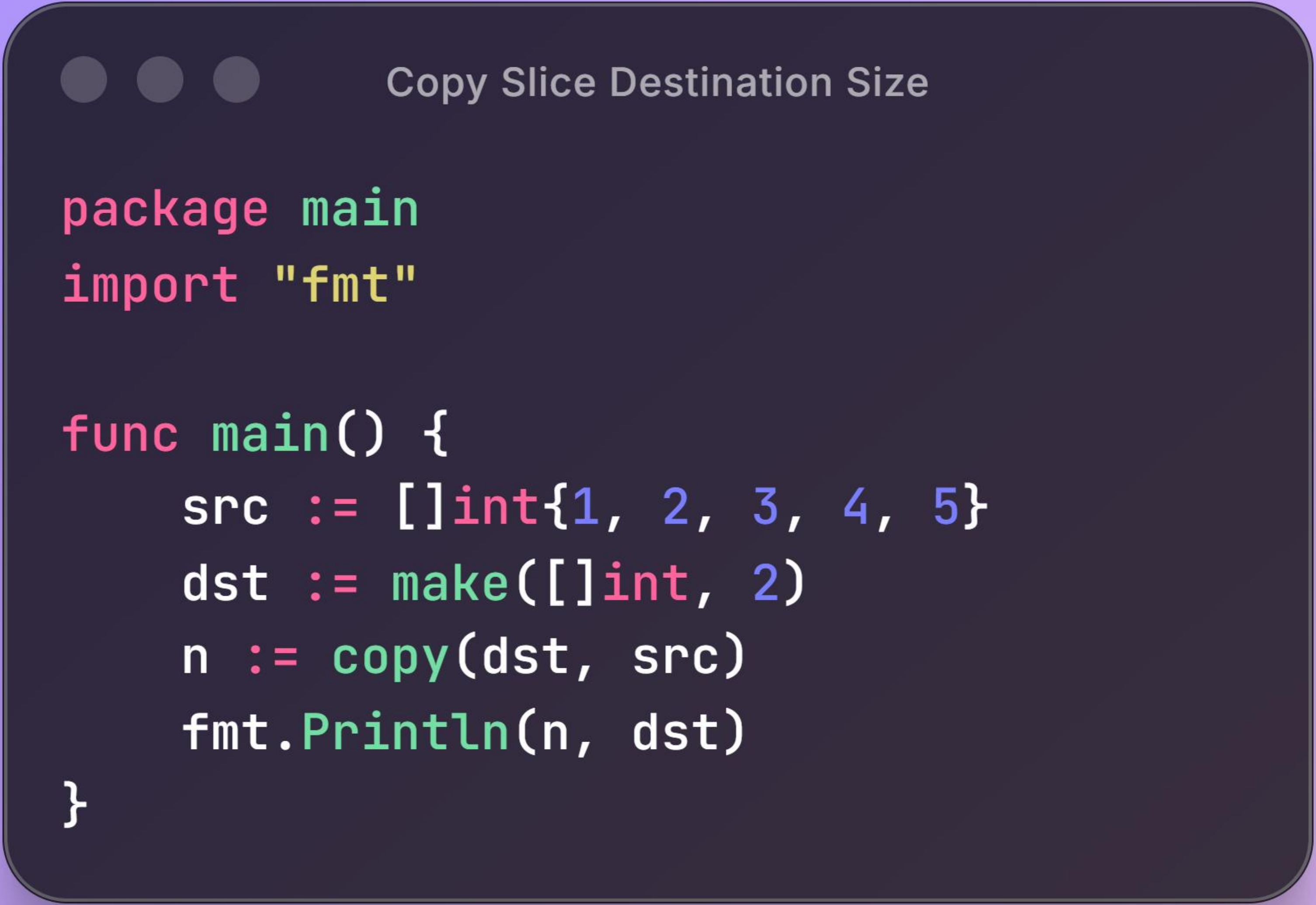
Strings Are Immutable

```go
package main
import "fmt"

func main() {
    s := "hello"
    s[0] = 'H' // What happens?
    fmt.Println(s)
}
```



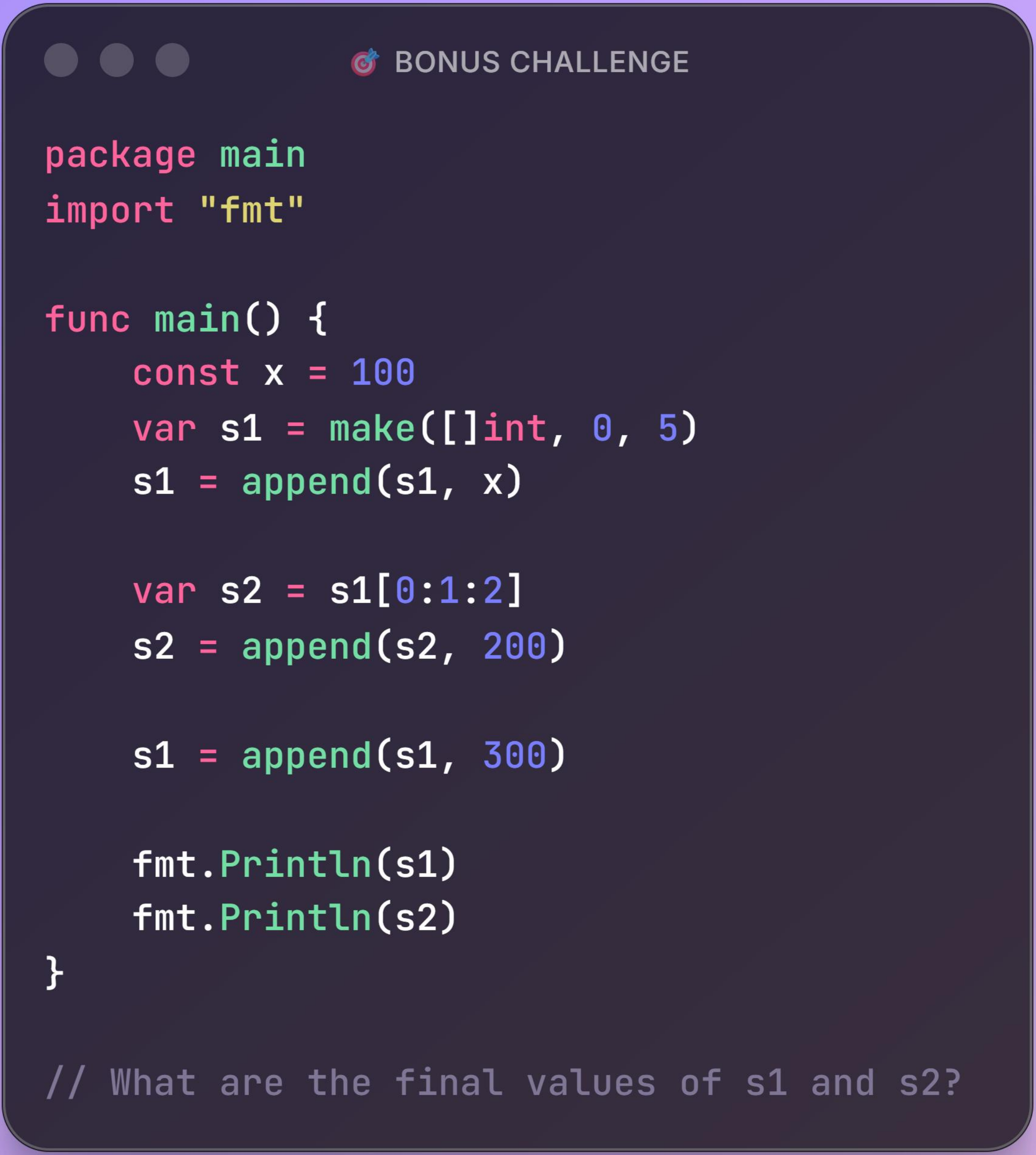
String to Byte Slice and Back

```go
package main
import "fmt"

func main() {
    s := "hello"
    b := []byte(s)
    b[0] = 'H'
    s2 := string(b)
    fmt.Println(s, s2)
}
// What gets printed?
```

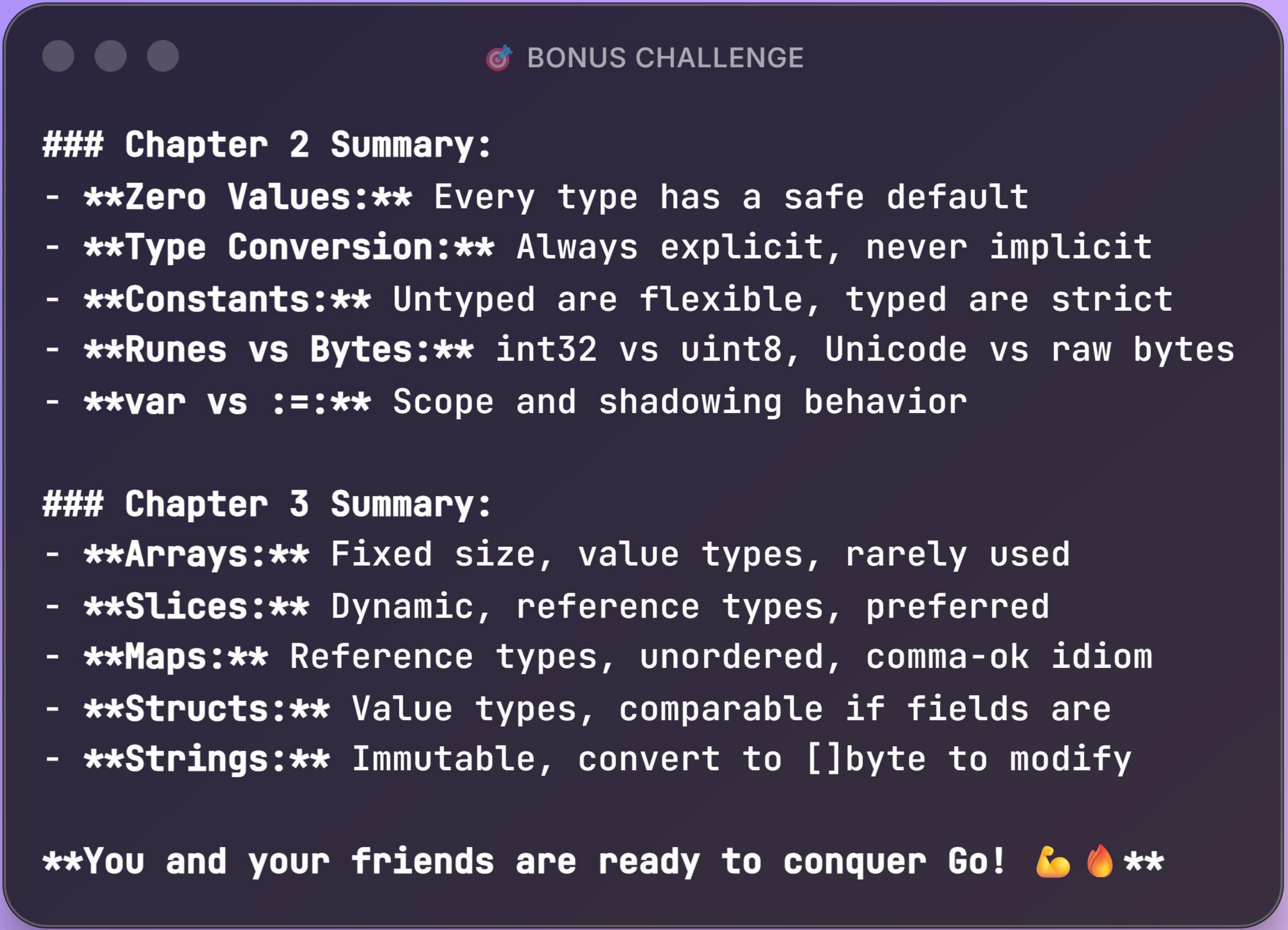


Copy Slice Destination Size

```go
package main
import "fmt"

func main() {
    src := []int{1, 2, 3, 4, 5}
    dst := make([]int, 2)
    n := copy(dst, src)
    fmt.Println(n, dst)
}
```



```go
package main
import "fmt"

func main() {
    const x = 100
    var s1 = make([]int, 0, 5)
    s1 = append(s1, x)

    var s2 = s1[0:1:2]
    s2 = append(s2, 200)

    s1 = append(s1, 300)

    fmt.Println(s1)
    fmt.Println(s2)
}

// What are the final values of s1 and s2?
```



```
### Chapter 2 Summary:
- **Zero Values:** Every type has a safe default
- **Type Conversion:** Always explicit, never implicit
- **Constants:** Untyped are flexible, typed are strict
- **Runes vs Bytes:** int32 vs uint8, Unicode vs raw bytes
- **var vs :=:** Scope and shadowing behavior

### Chapter 3 Summary:
- **Arrays:** Fixed size, value types, rarely used
- **Slices:** Dynamic, reference types, preferred
- **Maps:** Reference types, unordered, comma-ok idiom
- **Structs:** Value types, comparable if fields are
- **Strings:** Immutable, convert to []byte to modify

**You and your friends are ready to conquer Go! 💪🔥**
```