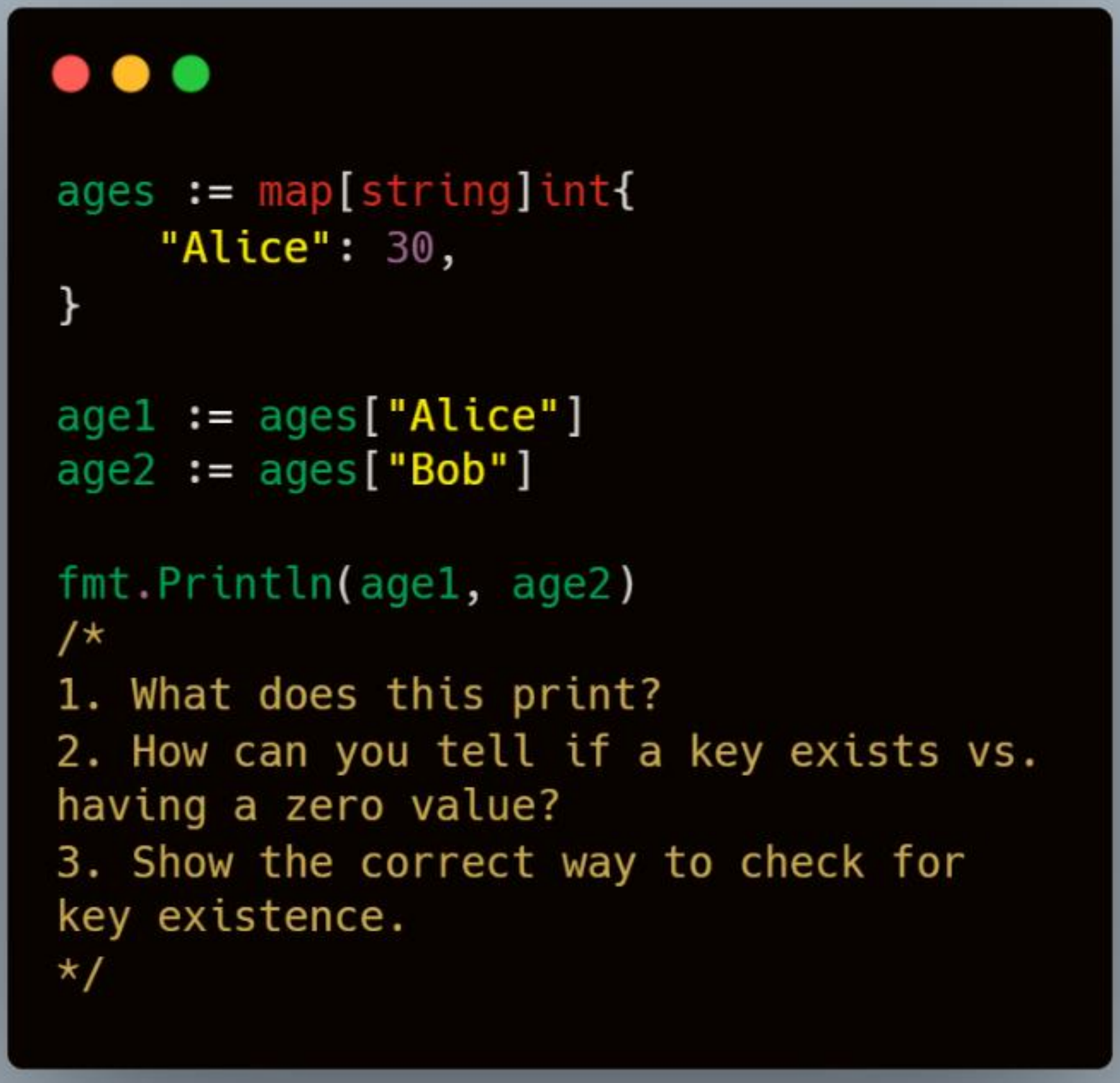


```go
ages := map[string]int{
    "Alice": 30,
}

age1 := ages["Alice"]
age2 := ages["Bob"]

fmt.Println(age1, age2)
/*
1. What does this print?
2. How can you tell if a key exists vs.
having a zero value?
3. Show the correct way to check for
key existence.
*/
```

```go
s := []int{1, 2, 3}
s = append(s, 4)
s = append(s, 5, 6, 7)

fmt.Println(s)

/*
1. What does this print?
2. Why do we need `s = append(s, 4)`
instead of just `append(s, 4)`?
*/
```

```go
type Person struct {
    Name string
    Age  int
}

// Which declarations are valid?
p1 := Person{"Alice", 30}          // A
p2 := Person{Name: "Bob", Age: 25} // B
p3 := Person{Age: 20}              // C
p4 := Person{}                     // D

// Which are valid? What values does
each have?
```

```go
// Three ways to create a slice
s1 := []int{1, 2, 3}
s2 := make([]int, 3)
s3 := make([]int, 3, 5)

/*
1. What's the length and capacity of
each slice?
2. What values are in `s2`?
3. What does the `5` mean in `s3`?
*/
```

```go
person := struct {
    Name string
    Age  int
}{
    Name: "Alice",
    Age:  30,
}

/*
1. Is this valid Go code?
2. When would you use an anonymous
struct?
3. Can you create a slice of these
anonymous structs?
*/
```

```go
ages := map[string]int{
    "Alice": 30,
    "Bob":   25,
}

// What happens with each operation?
fmt.Println(ages["Alice"])      // A
fmt.Println(ages["Charlie"])    // B
ages["David"] = 35              // C
delete(ages, "Bob")             // D

/*
What does each operation do? What does
B print?
*/
```

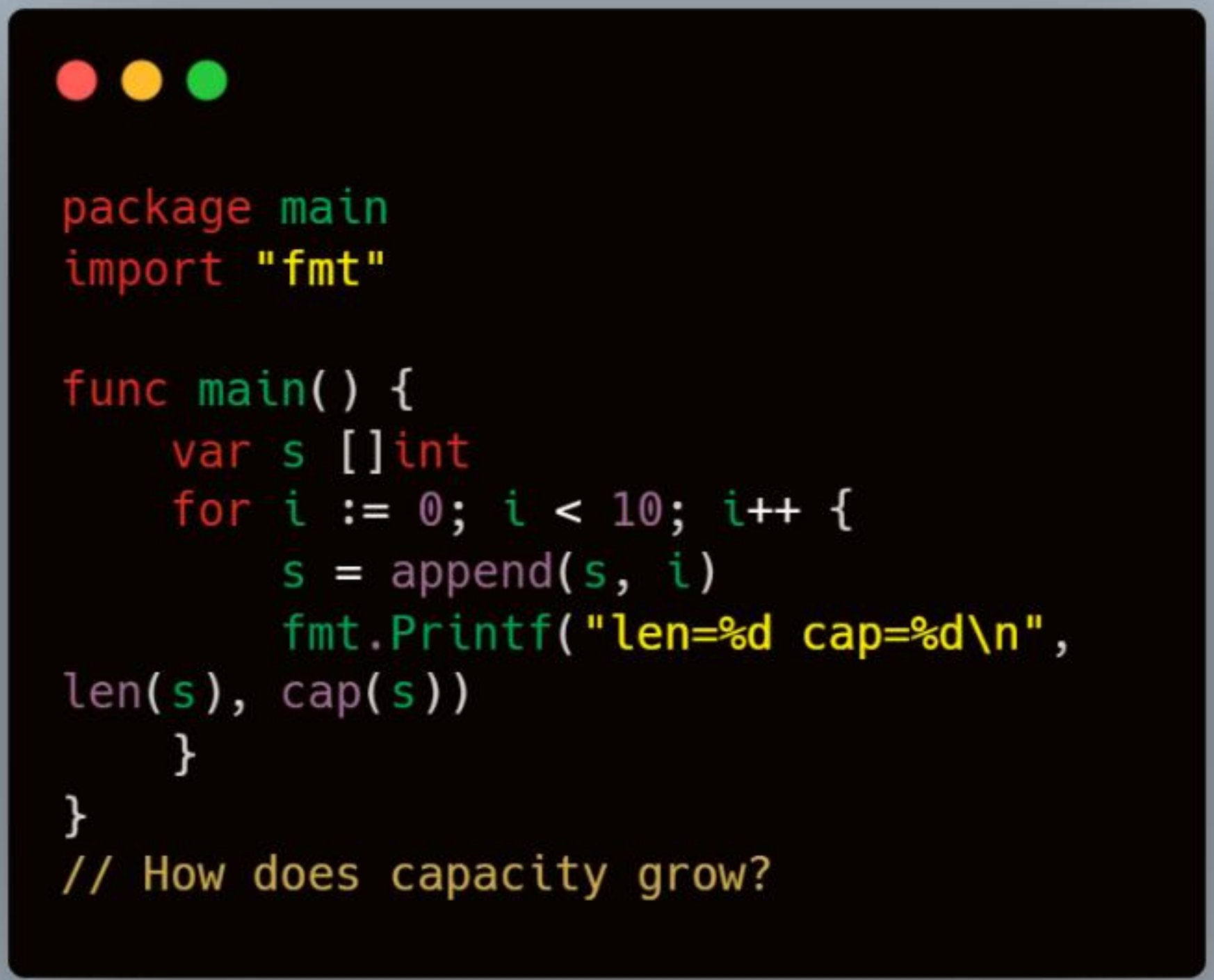
```go
arr := [3]int{1, 2, 3}
slice := []int{1, 2, 3}

fmt.Println(len(arr), len(slice))

/*
1. What's the fundamental difference
between `arr` and `slice`?
2. Can you append to `arr`? Why or why
not?
3. Can you append to `slice`? Show the
syntax.
*/
```

```
// Which of these array declarations
are valid? Which are invalid and why?

var a [5]int                          // A
var b = [5]int{1, 2, 3, 4, 5}  // B
var c = [...]int{1, 2, 3}          // C
var d [5]int = [3]int{1, 2, 3}  // D
```

Your Task: Identify which are
valid/invalid and explain why.

```
## 🎯 BONUS CHALLENGE: Array Monster Problem

package main
import "fmt"

func main() {
    s := make([]int, 0, 5)
    s = append(s, 1, 2, 3)
    s1 := s[0:2]
    s2 := s[1:3]

    s1 = append(s1, 100)
    s2[0] = 200

    fmt.Println(s)
    fmt.Println(s1)
    fmt.Println(s2)
}
// What are the final values of s, s1, and s2?
```

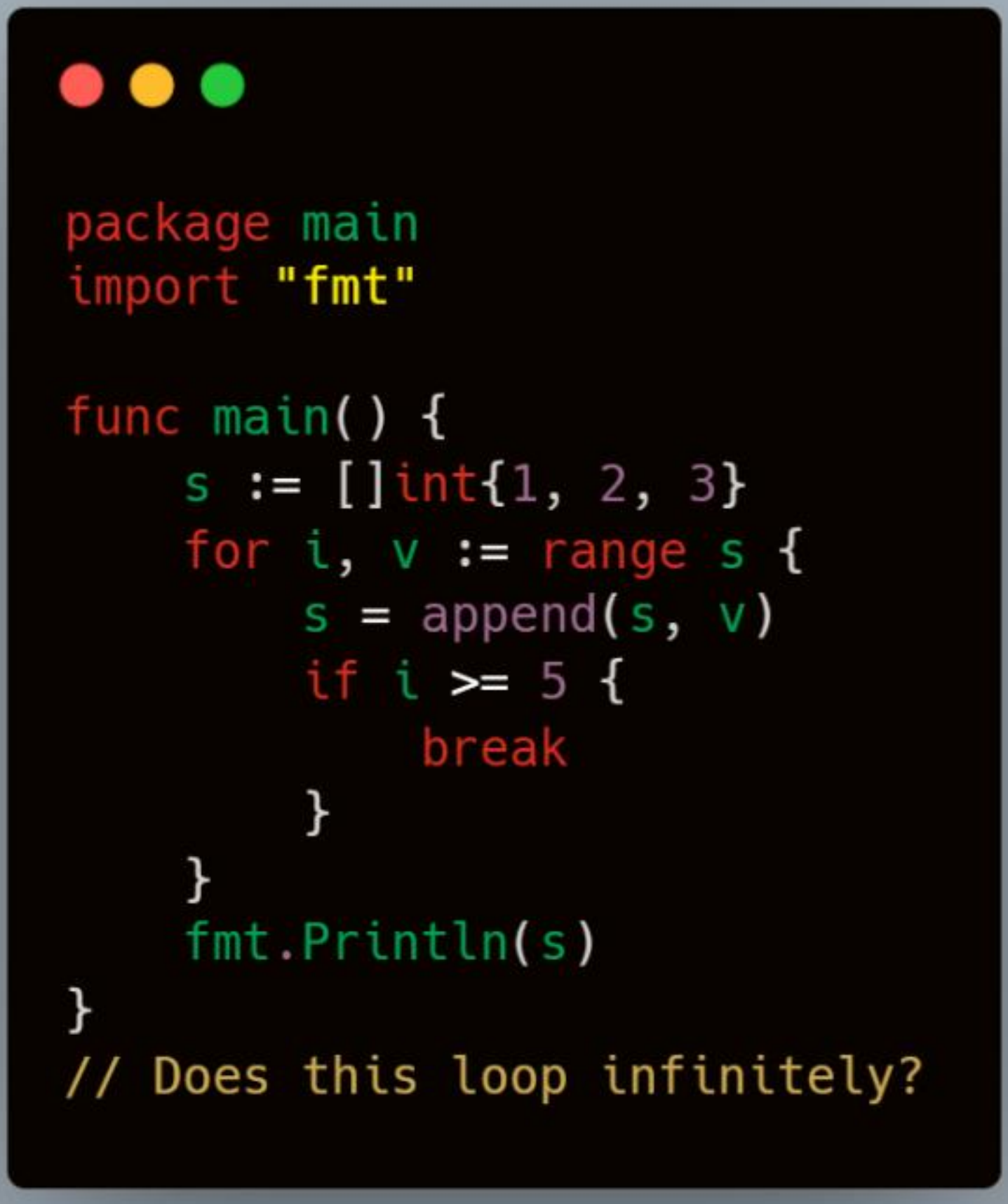
Quick Reference:
- Arrays: Fixed size, value type, comparable
- Slices: Dynamic, reference type, not comparable
- Capacity: Underlying array size, can exceed length
- append(): May reallocate, always assign result
- copy(): Creates independent copy
- Full slice `[low:high:max]`: Limits capacity
- Nil vs Empty: Different but both usable



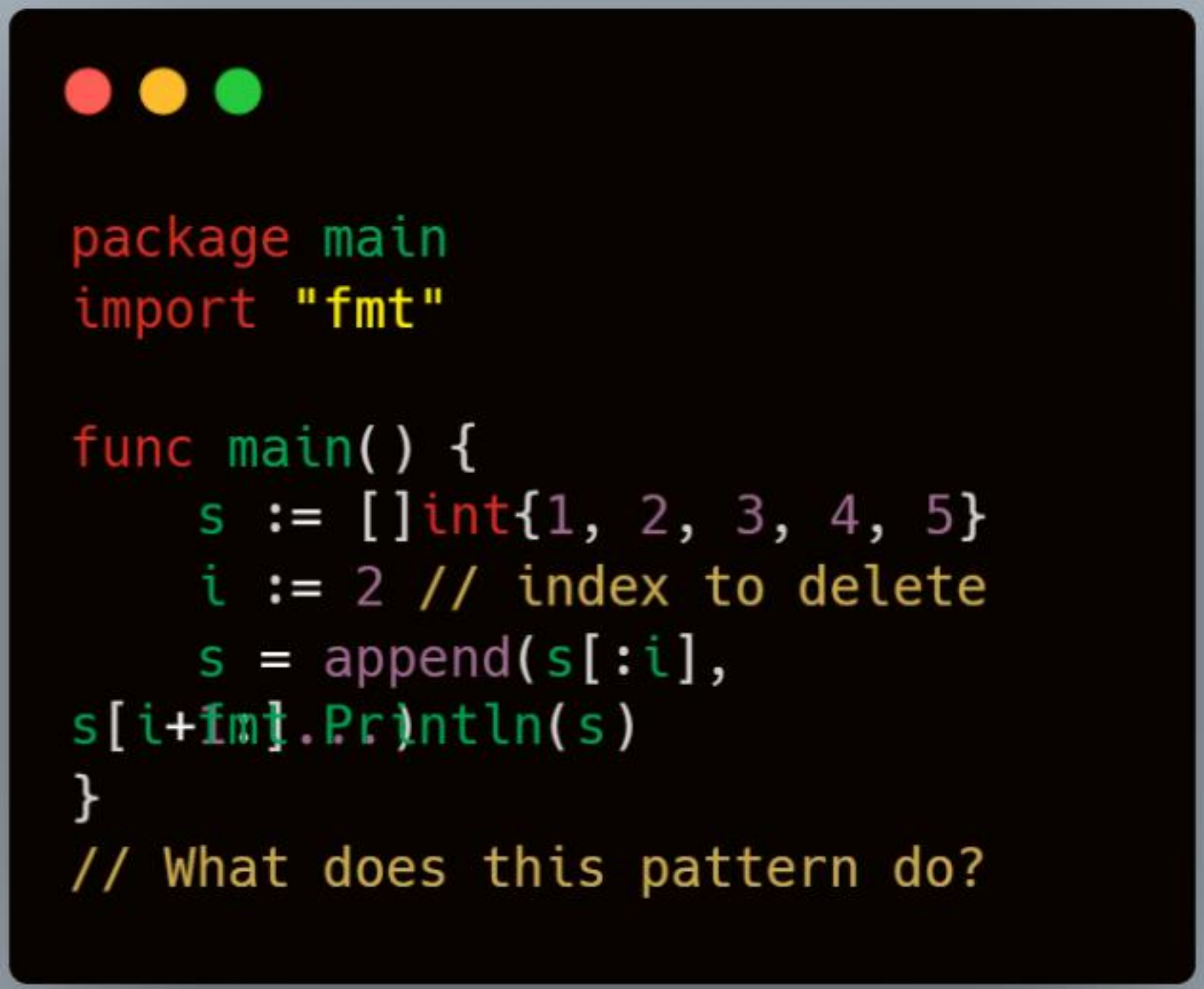
```go
package main
import "fmt"

func main() {
    var s []int
    for i := 0; i < 10; i++ {
        s = append(s, i)
        fmt.Printf("len=%d cap=%d\n",
len(s), cap(s))
    }
}
// How does capacity grow?
```

```go
package main
import "fmt"

func reverse(s []int) {
    for i, j := 0, len(s)-1; i < j; i,
j = i+1, j-1 {
        s[i], s[j] = s[j], s[i]
    }
}

func main() {
    s := []int{1, 2, 3, 4, 5}
    reverse(s)
    fmt.Println(s)
}
// What's the output?
```

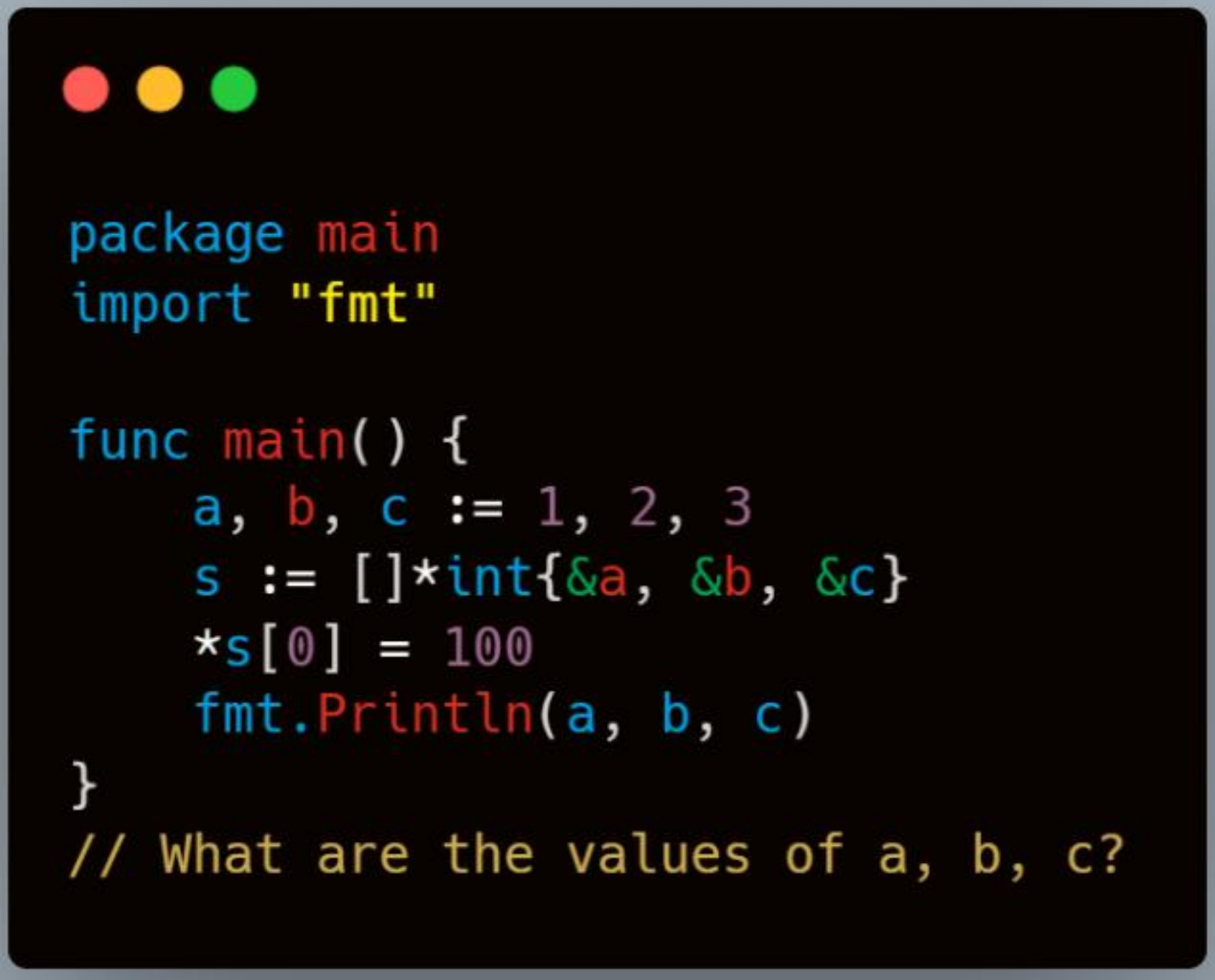
```go
package main
import "fmt"

func main() {
    s := []int{1, 2, 3, 4, 5, 6}
    result := s[:0]
    for _, v := range s {
        if v%2 == 0 {
            result = append(result, v)
        }
    }
    fmt.Println(result)
}
// What gets printed?
```

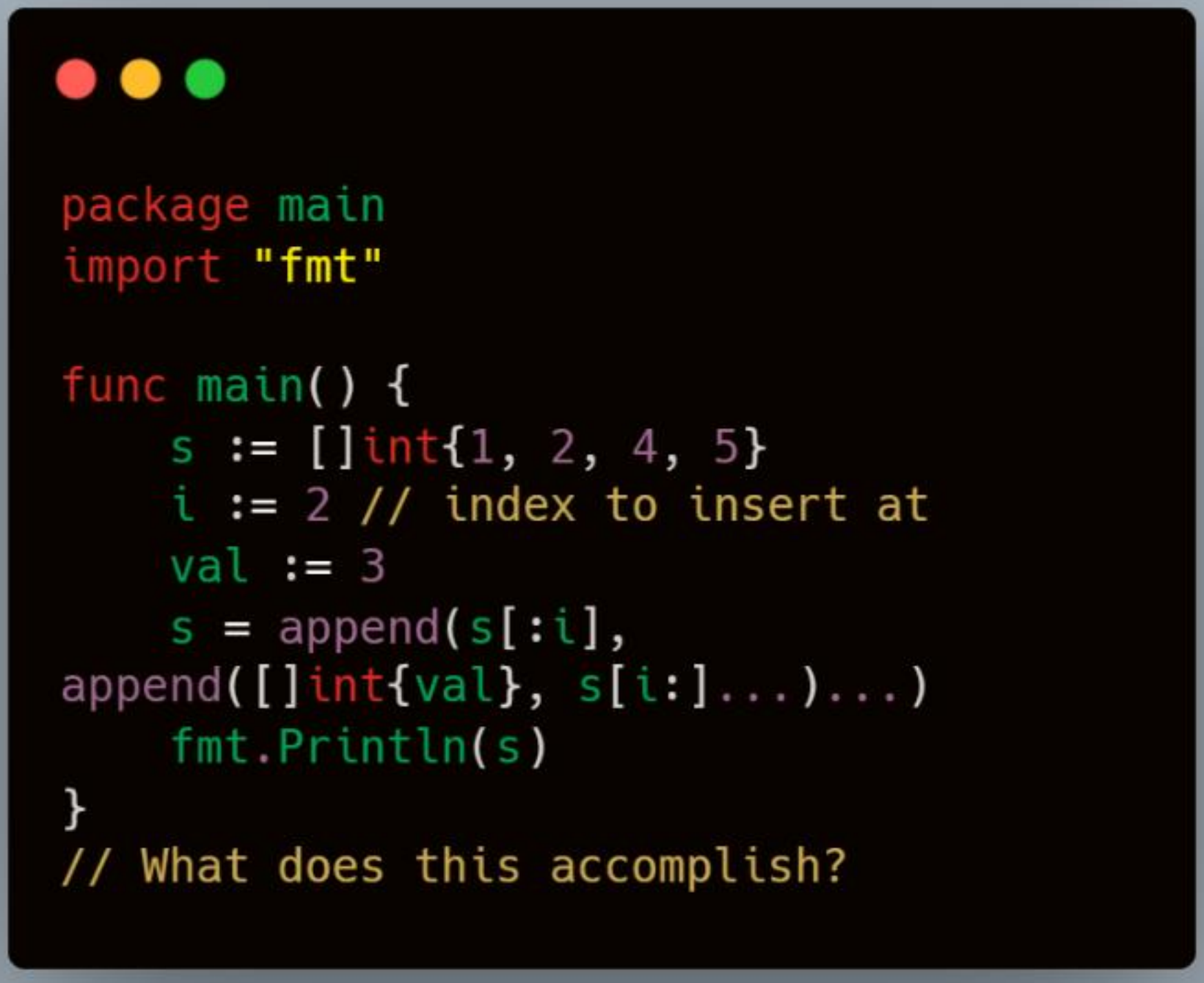


```go
package main
import "fmt"

func main() {
    s := []int{1, 2, 3}
    for i, v := range s {
        s = append(s, v)
        if i >= 5 {
            break
        }
    }
    fmt.Println(s)
}
// Does this loop infinitely?
```

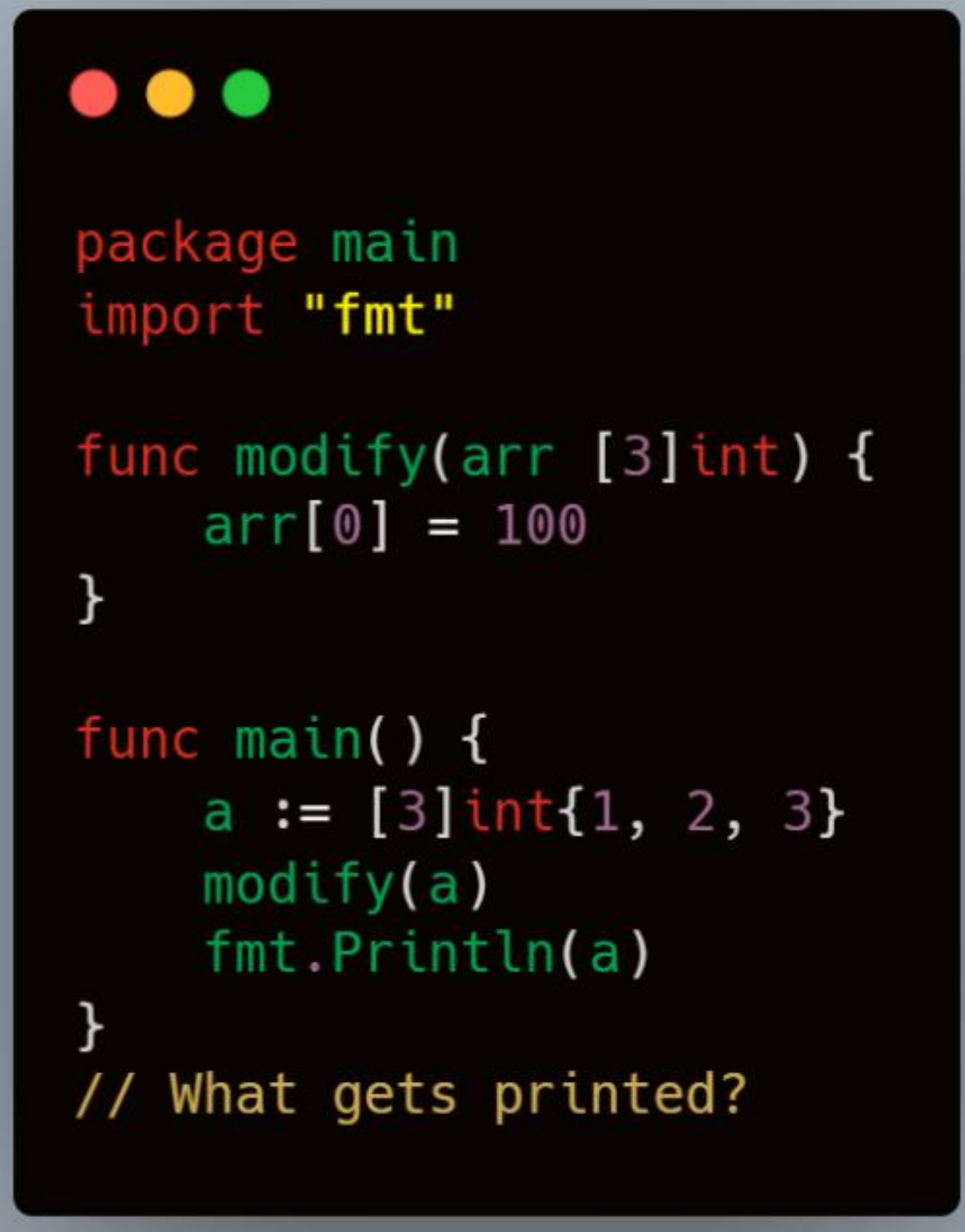


```go
package main
import "fmt"

func main() {
    s := []int{1, 2, 3, 4, 5}
    i := 2 // index to delete
    s = append(s[:i],
s[i+1:]...)
fmt.Println(s)
}
// What does this pattern do?
```

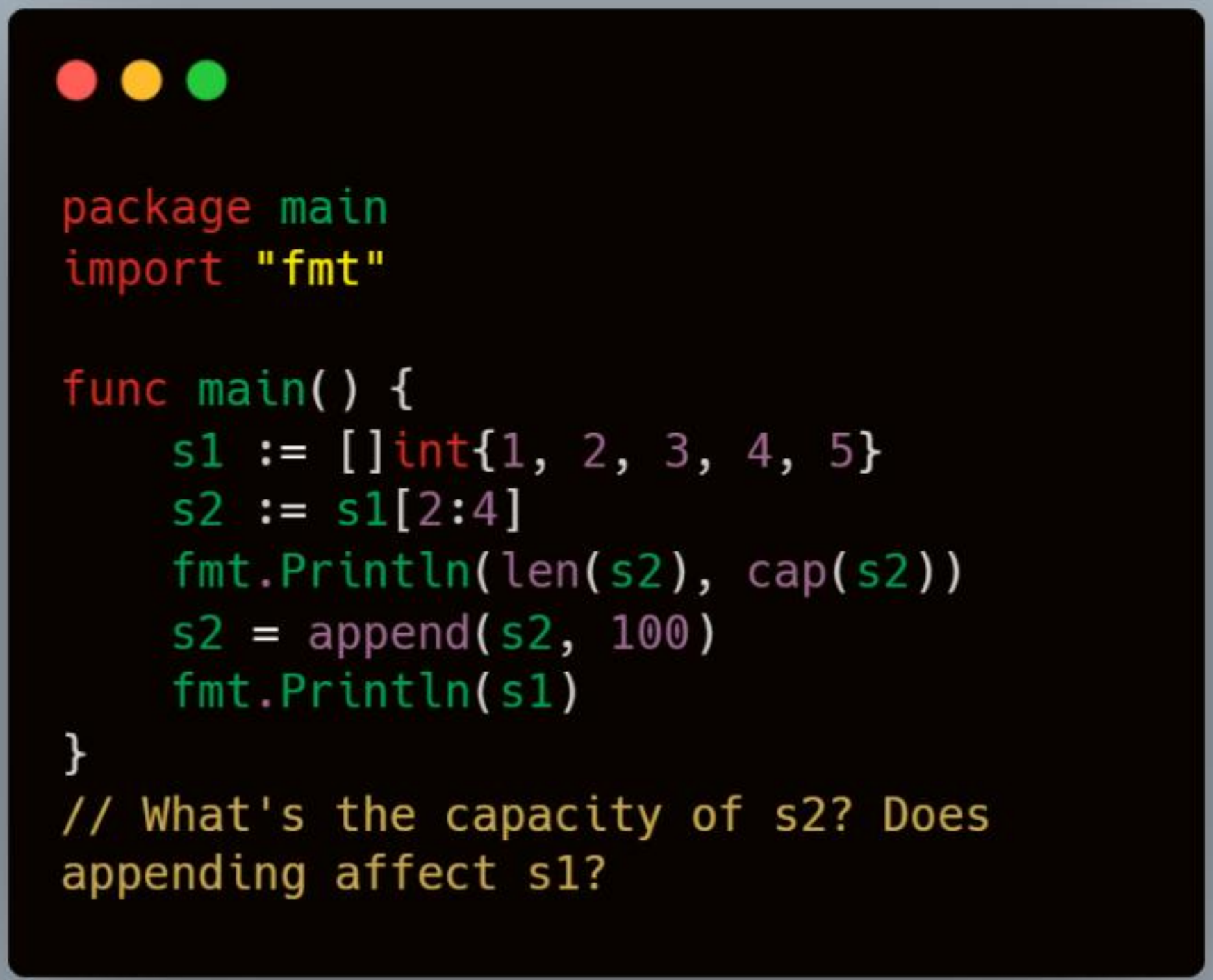

package main
import "fmt"
func main() {
a, b, c := 1, 2, 3
s := []*int{&a, &b, &c}
*s[0] = 100
fmt.Println(a, b, c)
}
// What are the values of a, b, c?



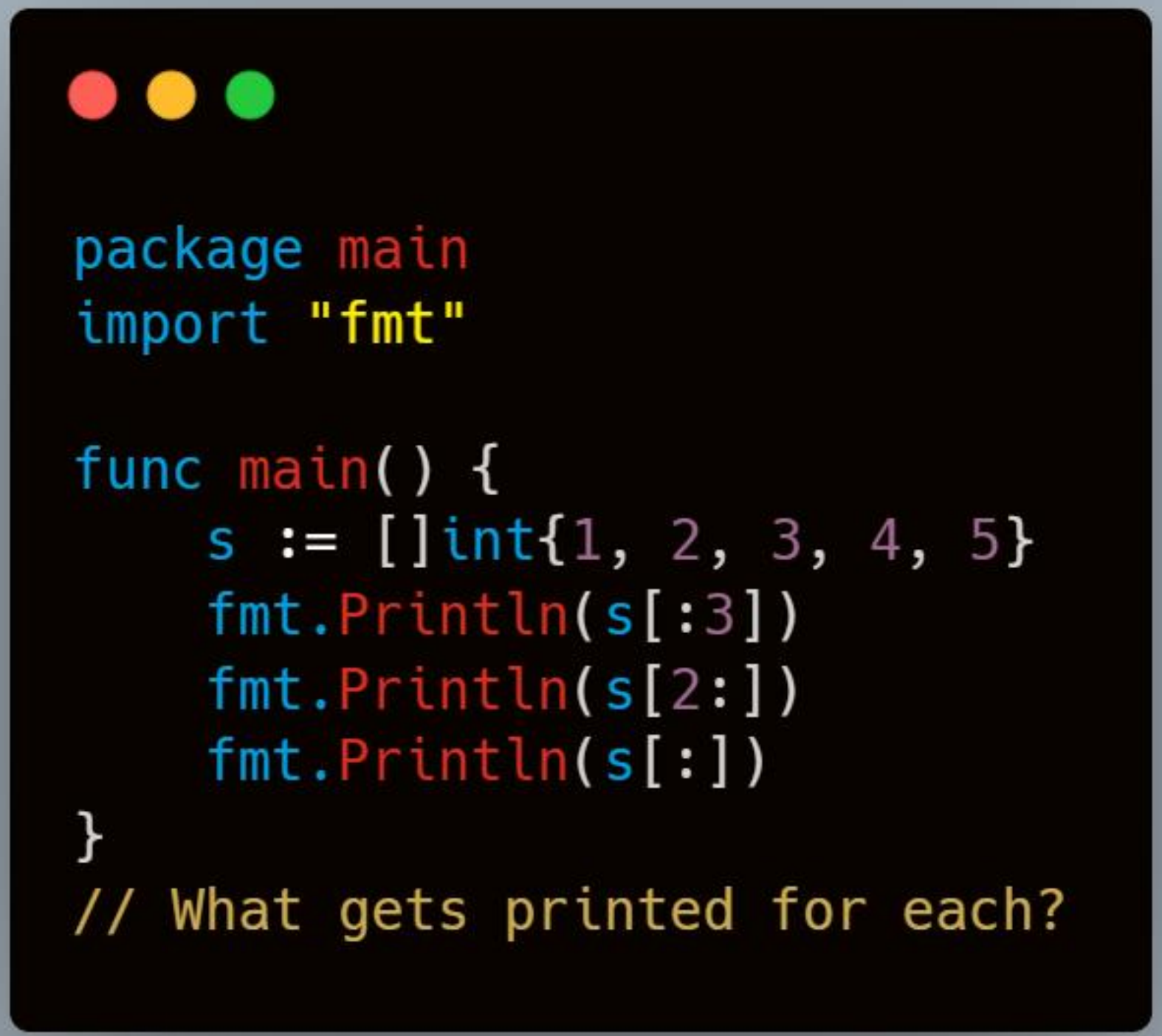
```go
package main
import "fmt"

func main() {
    s := []int{1, 2, 4, 5}
    i := 2 // index to insert at
    val := 3
    s = append(s[:i],
append([]int{val}, s[i:]...)...)
    fmt.Println(s)
}
// What does this accomplish?
```



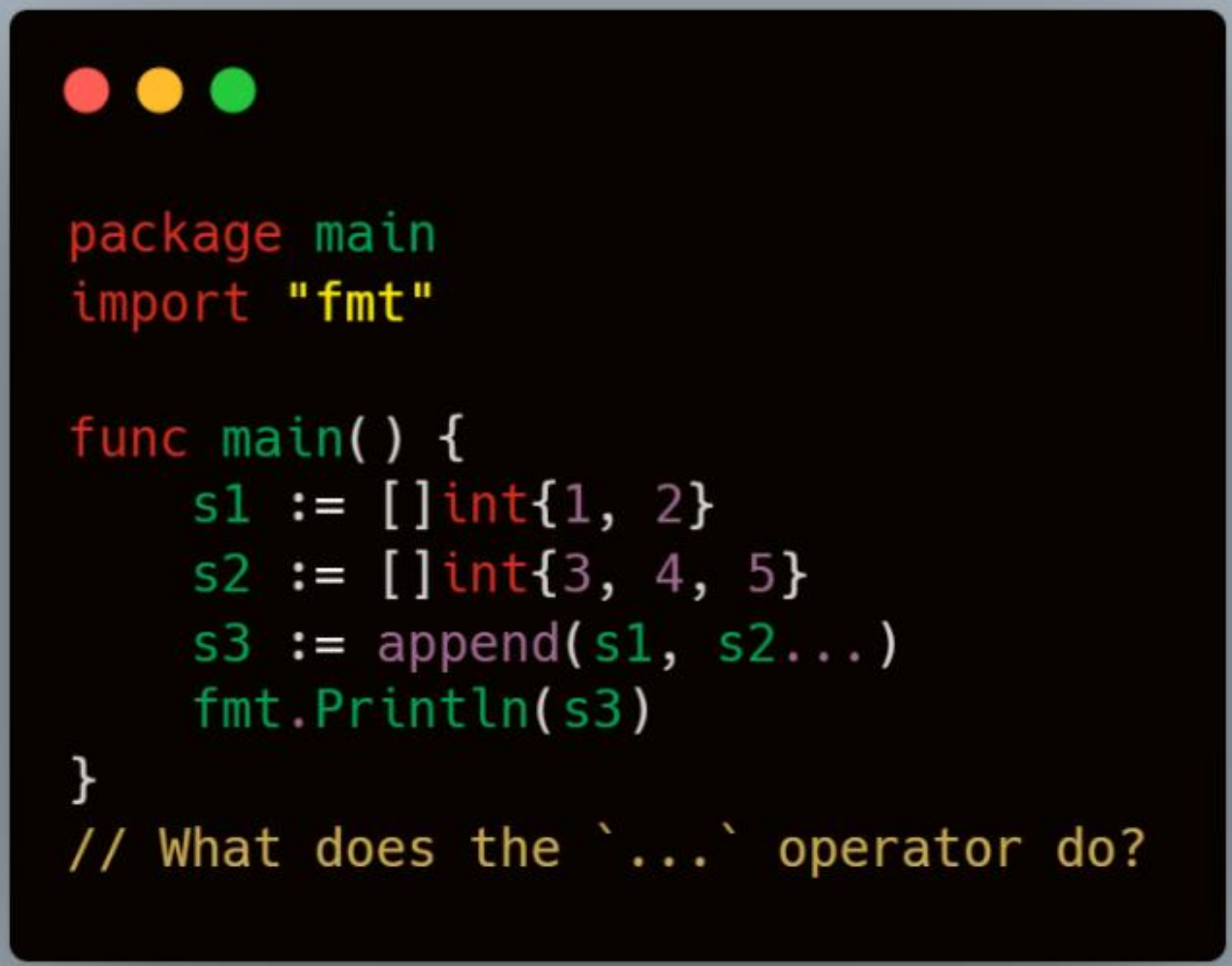
```go
package main
import "fmt"

func modify(arr [3]int) {
    arr[0] = 100
}

func main() {
    a := [3]int{1, 2, 3}
    modify(a)
    fmt.Println(a)
}
// What gets printed?
```



```go
package main
import "fmt"

func main() {
    s1 := []int{1, 2, 3, 4, 5}
    s2 := s1[2:4]
    fmt.Println(len(s2), cap(s2))
    s2 = append(s2, 100)
    fmt.Println(s1)
}
// What's the capacity of s2? Does
appending affect s1?
```

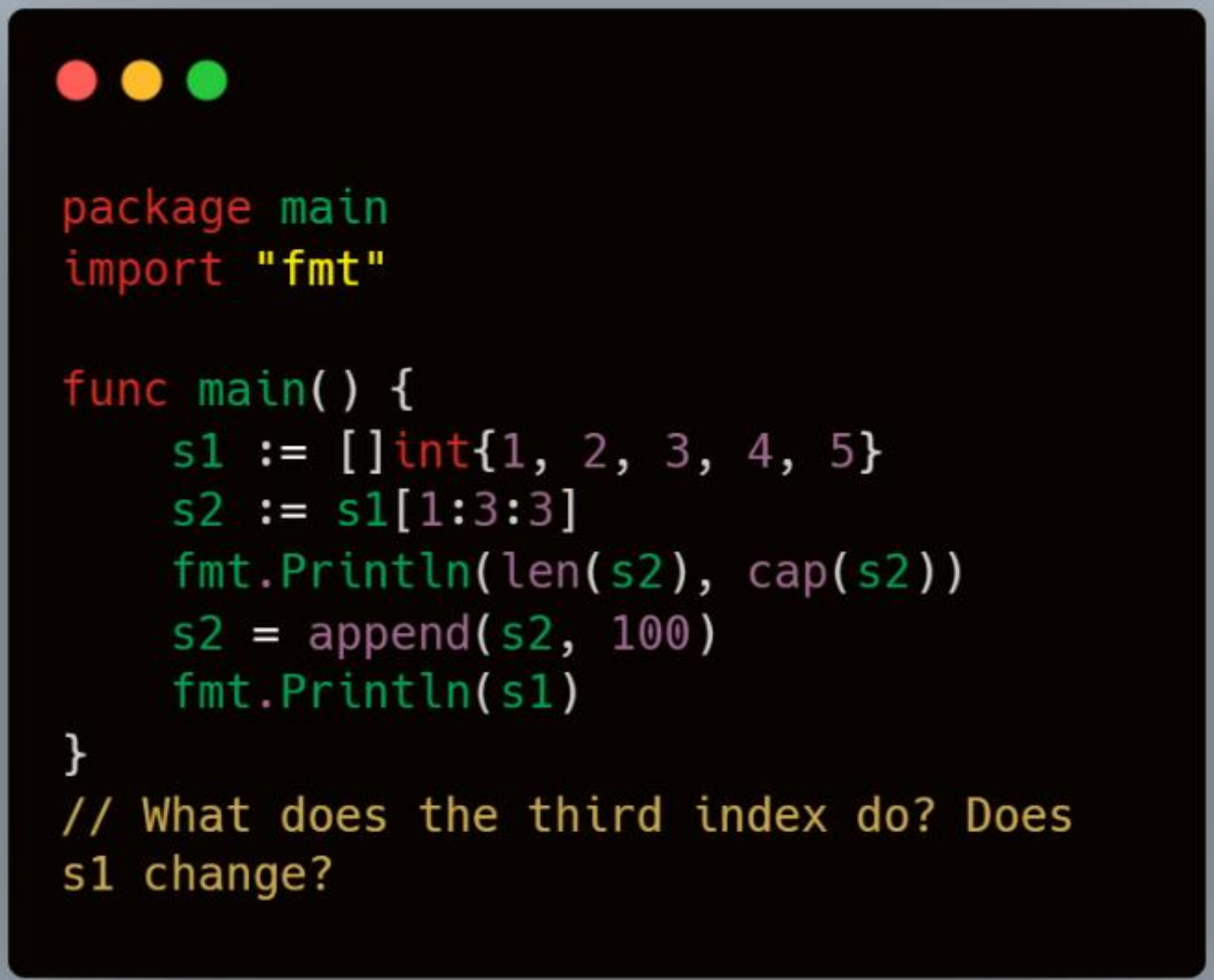


```go
package main
import "fmt"

func main() {
    s := []int{1, 2, 3, 4, 5}
    fmt.Println(s[:3])
    fmt.Println(s[2:])
    fmt.Println(s[:])
}
// What gets printed for each?
```


package main
import "fmt"
func main() {
s1 := []int{1, 2}
s2 := []int{3, 4, 5}
s3 := append(s1, s2...)
fmt.Println(s3)
}
// What does the `...` operator do?

```go
package main
import "fmt"

func main() {
    matrix := [][]int{
        {1, 2, 3},
        {4, 5},
        {6, 7, 8, 9},
    }
    fmt.Println(len(matrix))

}fmt.Println(len(matrix[1]))
// What are the lengths?
```

```go
package main
import "fmt"

func main() {
    s1 := []int{1, 2, 3}
    s2 := s1
    s1 = append(s1, 4)
    s1[0] = 100
    fmt.Println(s1)
    fmt.Println(s2)
}
// What do s1 and s2 contain after
modifications?
```



```go
package main
import "fmt"

func main() {
    s1 := []int{1, 2, 3, 4, 5}
    s2 := s1[1:3:3]
    fmt.Println(len(s2), cap(s2))
    s2 = append(s2, 100)
    fmt.Println(s1)
}
// What does the third index do? Does
s1 change?
```

```go
package main
import "fmt"

func main() {
    a1 := [3]int{1, 2, 3}
    a2 := [3]int{1, 2, 3}
    a3 := [3]int{1, 2, 4}

    fmt.Println(a1 == a2)
    fmt.Println(a1 == a3)
}
// Can you compare arrays? What gets
printed?
```

```go
package main
import "fmt"

func main() {
    s := []int{1, 2, 3, 4, 5}
    s2 := s[1:4]
    s3 := s2[:cap(s2)]
    fmt.Println(s2)
    fmt.Println(s3)
}
// What do s2 and s3 contain?
```

```go
package main
import "fmt"

func main() {
    s1 := []int{1, 2, 3}
    s2 := s1
    s3 := make([]int, len(s1))
    copy(s3, s1)

    s1[0] = 100
    fmt.Println(s1, s2, s3)
}
// What do s1, s2, and s3 contain?
```

```go
package main
import "fmt"

func main() {
    var s1 []int
    s2 := []int{}
    s3 := make([]int, 0)

    fmt.Println(s1 == nil, len(s1),
cap(s1))
    fmt.Println(s2 == nil, len(s2),
cap(s2))
    fmt.Println(s3 == nil, len(s3),
cap(s3))
}
// What's the difference between these
three slices?
```

```go
package main
import "fmt"

func modify(s []int) {
    s[0] = 100
}

func main() {
    arr := [3]int{1, 2, 3}
    s := arr[:]
    modify(s)
    fmt.Println(arr)
}
//What gets printed?
```

```go
package main
import "fmt"

func main() {
    s := make([]int, 3, 5)
    fmt.Println(len(s), cap(s))
    s = append(s, 1, 2, 3)
    fmt.Println(len(s), cap(s))
}
//What are the lengths and capacities
printed?
```