Orbit Challange 001

2025-05-11 19:26

Status: #adult

Tags: programming Golang bgce

Code conundrum

1. Here's a code snippet:

```
func main() {
    s := make([]int, 1024)

    // ☠️ One more push
    s = append(s, 42)

    _ = s // 🔇 What is the new capacity now?
}
```

Answer:

New capacity is 1536. Because go doesn't necessarily increase 25% after 1024. it allocates extra cells in the memory by it's default byte allocation which is also known as memory block. (if you don't believe you can run the code in your terminal and ask Chatgpt for more explanation)

2. What will be printed?

```
func main() {
    a := []int{1, 2, 3, 4}
    b := a[:2]
    c := a[2:]

    b = append(b, 99)
    c[0] = 88

    fmt.Println("a:", a)
    fmt.Println("b:", b)
    fmt.Println("c:", c)
}
```

Answer:

```
a: [1 2 88 4]
b: [1 2 88]
c: [88 4]
```

3. How many lines will be printed?

```
func main() {
    var a []int
    if a == nil {
        fmt.Println("nil slice")
    }

    b := make([]int, 0)
    if b == nil {
        fmt.Println("also nil?")
    }
}
```

Answer:

```
nil slice
```

Only one line would be printed cause `b` is not a nil slice. if you wanna know the deep explanation on why b is not a nil slice. maybe you should rewatch the youtube videos or maybe join our discord session regularly

4. What is the value of a after trick?

```
func main() {
    a := []int{10, 20, 30}
    trick(a...) //[90 20 30]
    fmt.Println(a) //[90 20 30]
}

func trick(nums ...int) {
    nums[0] = 99 //[90 20 30]
    nums = append(nums, 77) //[90 20 30 77] and nums now represent a new underlying
array but trick didn't return it. so a will only be changed for nums[0] = 99
}
```

Answer:

```
[99 20 30]
```

I guess it's better try running it in a terminal

5. What will be the value of x and y?

```go
func main() {
    x := make([]int, 0, 2)
    x = append(x, 1)
    y := x
    x = append(x, 2)
    y = append(y, 3)
}
```

Answer:

```
[1 3]
[1 3]
```

Well Because both x and y is representing the same underlying array

6.

```go
func main() {
    slice := []int{1, 2, 3, 4}
    target := 4
    find := false

    for _, v := range slice {
        if v == target {
            find := true
            break
        }
    }

    if find {
        fmt.Println("Found")
    } else {
        fmt.Println("Not Found")
    }
}
```

Answer:
whats the output and why?
the code doesn't even compile, why? Because the find := true redeclares the find variable inside the if block, instead of modifying the existing one (find = true). After redeclaring it, the new find variable is never used.

7. What will be the output?

```go
package main
import "fmt"
func main() {
    a := 5
    b := &a
    *b = 10
    fmt.Println(a) } //what will be the output?
```

Answer:
10

References