Day 23 - Go (Golang) Basics - Slices

Day 23 – Slices in Go

A slice is a flexible and lightweight data structure that represents a segment of an array. Unlike arrays, slices are dynamic in size and more versatile.

- Slices can grow or shrink.
- They provide access to a numbered sequence of elements.
- They are more flexible than arrays.

A slice consists of three components:

1. Pointer → Points to the first accessible element in the underlying array.
2. Length → Number of elements in the slice.
3. Capacity → Number of elements from the start index to the end of the array.

Declare and Initialize a Slice

You don't specify size while declaring slices.

```
package main

import "fmt"

func main() {
    slice := []int{10, 20, 30}
    fmt.Println(slice)
}
```

Create Slice from Array

You can slice an array using syntax: `array[start:end]`

- Start index is included
- End index is excluded

```
package main

import "fmt"

func main() {
    arr := [5]int{10, 20, 30, 40, 50}
    slice := arr[1:4]
    fmt.Println(slice) // Output: [20 30 40]
}
```

Sub-slicing:

```
package main

import "fmt"

func main() {
    arr := [10]int{10, 20, 30, 40, 50, 60, 70, 80, 90, 100}
    slice := arr[1:8]
```

```go
    subSlice := slice[0:3]
    fmt.Println(slice)
    fmt.Println(subSlice)
}
```

Using `make` to Create Slices

Syntax: `make([]T, length, capacity)`

```go
package main

import "fmt"

func main() {
    slice := make([]int, 5, 8)
    fmt.Println(slice)           // [0 0 0 0 0]
    fmt.Println(len(slice))      // 5
    fmt.Println(cap(slice))      // 8
}
```

Slice Shares Underlying Array

Changing a slice changes the underlying array too.

```go
package main

import "fmt"

func main() {
    arr := [6]int{1, 2, 3, 4, 5, 6}
    slice := arr[:3]
    slice[1] = 900
    fmt.Println("After change:")
    fmt.Println(arr)    // [1 900 3 4 5 6]
    fmt.Println(slice)  // [1 900 3]
}
```

Appending to Slice

Use `append(slice, elements...)`

```go
package main

import "fmt"

func main() {
    slice := []int{10, 20}
    slice = append(slice, 30, 40)
    fmt.Println(slice) // [10 20 30 40]
}
```

Appending One Slice to Another:

```go
package main

import "fmt"

func main() {
    a := []int{1, 2}
    b := []int{3, 4}
    combined := append(a, b...)
    fmt.Println(combined) // [1 2 3 4]
}
```

Deleting Elements from Slice

To delete element at index `i`:
`slice = append(slice[:i], slice[i+1:]...)`

```go
package main

import "fmt"

func main() {
    slice := []int{10, 20, 30, 40, 50}
    // Delete element at index 2 (30)
    slice = append(slice[:2], slice[3:]...)
    fmt.Println(slice) // [10 20 40 50]
}
```

Copying Elements from One Slice to Another

Use the built-in `copy()` function.

```go
package main

import "fmt"

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