Golang Session

- Harsh Dusane

Topic : Golang Slices

What is slicing?

- A slice is a flexible and extensible data structure to implement and manage collections of data. Slices are made up of multiple elements, all of the same type. A slice is a segment of dynamic arrays that can grow and shrink as you see fit. Like arrays, slices are index-able and have a length. Slices have a capacity and length property.

How to create an empty Slice in Golang?

```
package main
import (
        "fmt"
        "reflect"
)
func main() {
        var intSlice []int
        var strSlice []string
        fmt.Println(reflect.ValueOf(intSlice).Kind())
        fmt.Println(reflect.ValueOf(strSlice).Kind())
}
```

How to create Slice using Make function in Golang?

```
package main
import (
        "fmt"
        "reflect"
)
func main() {
        var intSlice = make([]int, 10)        // when length and capacity is same
        var strSlice = make([]string, 10, 20) // when length and capacity is different
        fmt.Printf("intSlice \tLen: %v \tCap: %v\n", len(intSlice), cap(intSlice))
        fmt.Println(reflect.ValueOf(intSlice).Kind())
        fmt.Printf("strSlice \tLen: %v \tCap: %v\n", len(strSlice), cap(strSlice))
        fmt.Println(reflect.ValueOf(strSlice).Kind())
}
```

How to initialize the slice with values using a slice literal?

```
package main
import "fmt"
func main() {
        var intSlice = []int{10, 20, 30, 40}
        var strSlice = []string{"India", "Canada", "Japan"}
        fmt.Printf("intSlice \tLen: %v \tCap: %v\n", len(intSlice), cap(intSlice))
        fmt.Printf("strSlice \tLen: %v \tCap: %v\n", len(strSlice), cap(strSlice))
}
```

How to create Slice using new keyword in Golang?

```
package main
import (
        "fmt"
        "reflect"
)
func main() {
        var intSlice = new([50]int)[0:10]
        fmt.Println(reflect.ValueOf(intSlice).Kind())
        fmt.Printf("intSlice \tLen: %v \tCap: %v\n", len(intSlice), cap(intSlice))
        fmt.Println(intSlice)
}
```

How to add items to Slice using append function in Golang?

```
package main
import "fmt"
func main() {
        a := make([]int, 2, 5)
        a[0] = 10
        a[1] = 20
        fmt.Println("Slice A:", a)
        fmt.Printf("Length is %d Capacity is %d\n", len(a), cap(a))
        a = append(a, 30, 40, 50, 60, 70, 80, 90)
        fmt.Println("Slice A after appending data:", a)
        fmt.Printf("Length is %d Capacity is %d\n", len(a), cap(a))
}
```

How to access slice items in Golang?

```
package main
import "fmt"
func main() {
        var intSlice = []int{10, 20, 30, 40}
        fmt.Println(intSlice[0])
        fmt.Println(intSlice[1])
  fmt.Println(intSlice[0:4])
}
```

How to change slice item value in Golang?

```
package main
import "fmt"
func main() {
        var strSlice = []string{"India", "Canada", "Japan"}
  fmt.Println(strSlice)
  strSlice[2] = "Germany"
  fmt.Println(strSlice)
}
```

How to delete an element from a Slice in Golang?

```
package main
import "fmt"
func main() {
        var strSlice = []string{"India", "Canada", "Japan", "Germany", "Italy"}
        fmt.Println(strSlice)
        strSlice = RemoveIndex(strSlice, 3)
        fmt.Println(strSlice)
}
func RemoveIndex(s []string, index int) []string {
        return append(s[:index], s[index+1:]...)
}
```

How to copy one slice items into another slice in Golang?

```
package main
import "fmt"
func main() {
        a := []int{5, 6, 7} // Create a smaller slice
        fmt.Printf("[Slice:A] Length is %d Capacity is %d\n", len(a), cap(a))
        b := make([]int, 5, 10) // Create a bigger slice
        copy(b, a)              // Copy function
        fmt.Printf("[Slice:B] Length is %d Capacity is %d\n", len(b), cap(b))
        fmt.Println("Slice B after copying:", b)
        b[3] = 8
        b[4] = 9
        fmt.Println("Slice B after adding elements:", b)
}
```

How to slice elements in Golang?

```
package main
import "fmt"
func main() {
        var countries = []string{"india", "japan", "canada", "australia", "russia"}
        fmt.Printf("Countries: %v\n", countries)
        fmt.Printf(":2 %v\n", countries[:2])
        fmt.Printf("1:3 %v\n", countries[1:3])
        fmt.Printf("2: %v\n", countries[2:])
        fmt.Printf("2:5 %v\n", countries[2:5])
        fmt.Printf("0:3 %v\n", countries[0:3])
        fmt.Printf("Last element: %v\n", countries[4])
        fmt.Printf("Last element: %v\n", countries[len(countries)-1])
        fmt.Printf("Last element: %v\n", countries[4:])
        fmt.Printf("All elements: %v\n", countries[0:len(countries)])
        fmt.Printf("Last two elements: %v\n", countries[3:len(countries)])
        fmt.Printf("Last two elements: %v\n", countries[len(countries)-2:len(countries)])
        fmt.Println(countries[:])
        fmt.Println(countries[0:])
        fmt.Println(countries[0:len(countries)])
}
```

How to Iterate Over a Slice in Golang?

```
package main
import "fmt"
func main() {
        var strSlice = []string{"India", "Canada", "Japan", "Germany", "Italy"}
        fmt.Println("\n---------------Example 1 --------------------\n")
        for index, element := range strSlice {
                fmt.Println(index, "--", element)
        }
        fmt.Println("\n---------------Example 2 --------------------\n")
        for _, value := range strSlice {
                fmt.Println(value)
        }
        j := 0
        fmt.Println("\n---------------Example 3 --------------------\n")
        for range strSlice {
                fmt.Println(strSlice[j])
                j++
        }
}
```

How append a slice to an existing slice in Golang?

```
package main
import "fmt"
func main() {
        var slice1 = []string{"india", "japan", "canada"}
        var slice2 = []string{"australia", "russia"}
        slice2 = append(slice2, slice1...)
}
```

How to check if an item exists in Slice in Golang?

```
package main
import (
        "fmt"
        "reflect"
)
func main() {
        var strSlice = []string{"India", "Canada", "Japan", "Germany", "Italy"}
        fmt.Println(itemExists(strSlice, "Canada"))
        fmt.Println(itemExists(strSlice, "Africa"))
}
func itemExists(slice interface{}, item interface{}) bool {
        s := reflect.ValueOf(slice)
        if s.Kind() != reflect.Slice {
                panic("Invalid data-type")
        }
        for i := 0; i < s.Len(); i++ {
                if s.Index(i).Interface() == item {
                        return true
                }
        }
        return false
}
```