**Go slices**

Go slices are the abstraction over the go array.

Array will allows you to define several data items of same kind

But does not provide increase the size dynamically or to get sub array of its own

Slices overcome this limitation

var numbers []int /\* a slice of unspecified size \*/

/\* numbers == []int{0,0,0,0,0}\*/

numbers = make([]int,5,5) /\* a slice of length 5 and capacity 5\*/

**Defining the slice**

Declare the array without specifying the size will be slice

Alternatively can create the make function too

package main

import (

    "fmt"

)

func main() {

    number := []int{0, 1, 2, 3, 4}

    var number1 = make([]int, 3, 5) //3 is length and 5 is the capacity here

    fmt.Println(number)

    fmt.Println(len(number), cap(number))

    fmt.Println(number1)

    fmt.Println(len(number1), cap(number1))

    var number2 []int //3 is length and 5 is the capacity here

    if number2 == nil {

        fmt.Printf("SLice is nil \n")

        fmt.Println(number2)

    }

}

[0 1 2 3 4]

5 5

[0 0 0]

3 5

SLice is nil

[]

**Subslice**

Subslice allows to create new lice from current slice use upper bound and lower bound limits as per below

**[lower-bound:upper-bound]**

package main

import "fmt"

func main() {

    numbers := []int{0, 1, 2, 3, 4, 5, 6, 7, 8}

    fmt.Println(numbers)

    fmt.Println(numbers[2:3])

    fmt.Println(numbers[:3])

    fmt.Println(numbers[4:])

    numbers1 := make([]int, 0, 5)

    fmt.Println(numbers1)

    number2 := numbers[1:5]

    fmt.Println(number2)

    number3 := numbers[3:]

    fmt.Println(number3)

}

[0 1 2 3 4 5 6 7 8]

[2]

[0 1 2]

[4 5 6 7 8]

[]

[1 2 3 4]

[3 4 5 6 7 8]

**Slice append and copy**

package main

import "fmt"

func main() {

    var number []int

    fmt.Printf("Len = %d , Cap = %d , slice = %v \n", len(number), cap(number), number)

    number = append(number, 0)

    number = append(number, 1)

    number = append(number, 2)

    number = append(number, 3, 4, 5)

    fmt.Printf("Len = %d , Cap = %d , slice = %v \n", len(number), cap(number), number)

    numbers1 := make([]int, len(number), (cap(number))\*2)

    copy(numbers1, number)

    fmt.Printf("Len = %d, Cap = %d , slice = %v \n", len(numbers1), cap(numbers1), numbers1)

}

Len = 0 , Cap = 0 , slice = []

Len = 6 , Cap = 8 , slice = [0 1 2 3 4 5]

Len = 6, Cap = 16 , slice = [0 1 2 3 4 5]