Reslicing

This lesson discusses an important concept called reslicing.

WE'LL COVER THE FOLLOWING

Introduction

We saw that a slice is often made smaller than the underlying array initially, like this:

```
slice1 := make([]type, start_length, capacity)
```

Here, `start_length` is the length of slice and `capacity` is the length of the underlying array. This is useful because now our slice can grow until `capacity`. Changing the length of the slice is called **reslicing**, it is done like:

```
slice1 = slice1[0:end]
```

where `end` is another end-index (length) than before.

Resizing a slice by **1** can be done as follows:

```
sl = sl[0:len(sl)+1] // extend length by 1
```

A slice can be resized until it occupies the whole underlying array.

Explanation

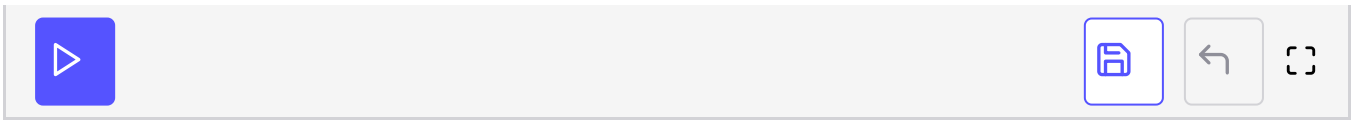
The following program is a detailed implementation of reslicing a slice:

```
package main
```

```
import "fmt"

func main() {
  slice1 := make([]int, 0, 10)
  // load the slice, cap(slice1) is 10:
  for i := 0; i < cap(slice1); i++ {
    slice1 = slice1[0:i+1] // reslice
    slice1[i] = i

    fmt.Printf("The length of slice is %d\n", len(slice1))
  }
  // print the slice:
  for i := 0; i < len(slice1); i++ {
    fmt.Printf("Slice at %d is %d\n", i, slice1[i])
  }
}
```



Reslicing

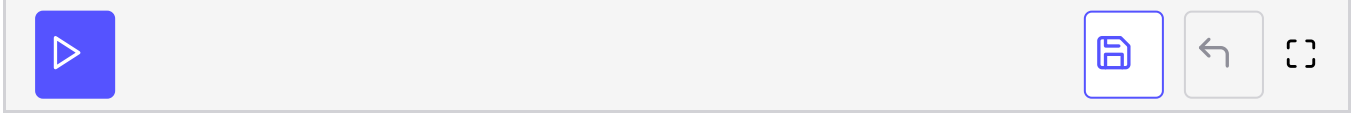
In the code above, at **line 5** in `main`, we make a slice `slice1` via the `make` function. Its length is **0** and its capacity is **10**. At **line 7** is the for loop that is iterating *capacity* times. At **line 8**, we are reslicing `slice1` as: `slice1 = slice1[0:i+1]`. The length of `slice1` is initially **0**. However, reslicing makes the length of `slice1` `i+1`, in every iteration. So in the first iteration, the length of `slice1` is **1**. In the second iteration, the length of `slice1` is **2**, and so on. At **line 9**, we are initializing each element at index `i` with the value `i`. The length of `slice1` is printed in each iteration at **line 11**. Finally, the loop at **line 14** prints each element of `slice1`.

Another example is:

```
package main
import "fmt"

func main(){
    var ar = [10]int{0,1,2,3,4,5,6,7,8,9}
    var a = ar[5:7]
    a = a[0:4] // ref of subarray {5,6,7,8}

    fmt.Println(a)
}
```



Creating a slice of the array `ar` as `ar[5:7]` makes `a` equal to **{5,6}**. Here, the

length of `a` is **2** and the capacity of `a` is **5**. In the last line, we are reslicing `a` as `a=a[0:4]`. Now, `a` equals to {5,6,7,8} with a length of **4** and a capacity of **5**.

To get a piece of information from a slice and expand or contract a slice, reslicing works perfectly. Golang also provides the functionality to copy a slice or append a slice in another slice. Move on to the next lesson to see how this works.