Solution Review: Bubble Sort the Slice

This lesson discusses the solution to the challenge given in the previous lesson.

```go
package main
import (
        "fmt"
)

func main() {
        sla := []int{2, 6, 4, -10, 8, 89, 12, 68, -45, 37}
        fmt.Println("before sort: ",sla)
        // sla is passed via call by value, but since sla is a reference type
        // the underlying array is changed (sorted)
        bubbleSort(sla)
        fmt.Println("after sort:  ",sla)
}

func bubbleSort(sl []int) {
        // passes through the slice:
        for pass:=1; pass < len(sl); pass++ {
                // one pass:
                for i:=0; i < len(sl) - pass; i++ {             // the bigger value 'bubbles
                        if sl[i] > sl[i+1] {
                                sl[i], sl[i+1] = sl[i+1], sl[i]
                        }
                }
        }
}
```

Bubble Sort

In the program above, in `main` at **line 7**, we make a slice `sla` of integers and assign random integers to it in an unsorted manner. Now, at **line 11**, we call the function `bubbleSort` and pass `sla` to it. Look at the header for `bubbleSort` at **line 15**, it accepts a slice `sl`.

Now, according to the bubble sort algorithm, we have to pass through the slice until it is sorted. In every pass, we take the element at index `i` and compare it with the element at index `i+1`. If the element at index `i` is greater than the element at index `i+1`, we swap them. Every pass makes sure that the greatest value, not in its correct place, is placed at its correct position. That's why loop in every pass managing the iterator `i` should run `len(sl)-pass` times. Doing this means that we need `len(sl)-1` number of passes in maximum to sort the slice. The last pass will sort two values after swap.

Look at **line 17**, where we write a for loop controlling the passes required to sort the slice. We need `len(sl)-1` maximum passes. For each pass, we'll iterate the slice until `len(sl)-pass` times (see **line 19**). During the iteration, if the element at index `i` is greater than the element at index `i+1`, we swap them according to condition at **line 20**. Results are verified through **line 8** and **line 12** in `main` by printing `sla` before sorting and after sorting respectively.

That's it about the solution. In the next lesson, you'll attempt another challenge.