Solution Review: Filtering with Higher-Order Functions

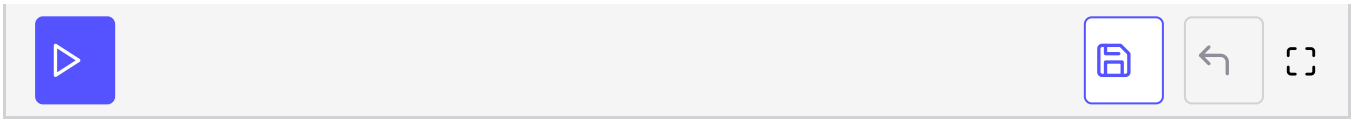
This lesson discusses the solution to the challenge given in the previous lesson.

```
package main
import (
        "fmt"
)

func main() {
        s := []int{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
        s = Filter(s, even)
        fmt.Println(s)
}

// Filter returns a new slice holding only
// the elements of s that satisfy fn()
func Filter(s []int, fn func(int) bool) []int {
        var p []int // == nil
        for _, i := range s {
                if fn(i) {
                        p = append(p, i)
                }
        }
        return p
}

func even(n int) bool {
        if n%2 == 0 {
                return true
        }
        return false
}
```



Filtering with Higher Order Functions

The program above has one basic function. The function `even` takes `n` as a parameter and returns a *boolean* value (see its header on **line 24**). If `n` is even, it will return *true*. Otherwise, it will return *false*. See the header of the `Filter` function on **line 14**. It takes a slice of *integers* as a first parameter, and a function `fn` of type `func` as a second parameter. The function `func` takes an `int` type variable as a parameter and returns a `bool` value.

The function `Filter` returns a slice of integers that are even. We declare a new slice of type int `p`, which will contain the filtered values (that are even) from `s` at **line 15**. Then, in the next line, we have a for loop that will iterate through all of the elements of `s` using `range`. For each element `i` of `s`, the function `fn` will be called. If `fn` returns *true*, it means `i` is even; otherwise it is *false*. If true is returned, we use the `append` function at **line 18** to append `i` to `p`. Once the loop is over, `p` is returned from the `Filter` function.

Let's see the `main` function now. At **line** 7, we declare a slice of integers named `s`. Then at **line 8**, we call the `filter` function with `s` as the first parameter and `even` as the second parameter and store the result back in `s`. Printing `s` at **line 9** verifies the result.

That's it about the solution. In the next lesson, you'll study how strings, arrays, and slices work together.