Solution Review: Finding Fibonacci Numbers with Slices

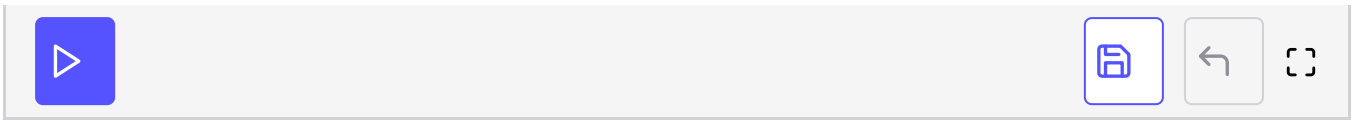
This lesson discusses the solution to the challenge given in the previous lesson.

```
package main
import "fmt"

func main() {
        result := fibarray(5)
        for ix, fib := range result {
                fmt.Printf("The %d-th Fibonacci number is: %d\n", ix,  fib)
        }
}

func fibarray(term int) []int { // calculating Fibonacci for first term numbers
        farr := make([]int, term)
        farr[0], farr[1] = 1, 1         // base case

        for i:= 2; i < term; i++ {
                farr[i] = farr[i-1] + farr[i-2] // calculating sequence for i index
        }
        return farr
}
```



Finding Fibonacci Numbers with Slices

In the code above, look at the header of the function `fibarray` at **line 11**, which takes `term` as the input and returns the Fibonacci sequence until `term` in an array of type **int**. We make a slice `farr` of size `term` at **line 12**. Since we know that the first and second Fibonacci numbers are both **1**, we set the first two indexes of the array `farr` to **1**. Now, we have a *for* loop at **line 15**, which starts from **2** and ends at the index **term-1**. At **line 16**, we calculate the Fibonacci value for any value at index `i` as: `farr[i] = farr[i-1] + farr[i-2]`, and return `farr` while exiting from the function.

In `main` at **line 5**, we called `fibarray` to store the result in a separate slice `result`. Then, at the end, we have another for loop at **line 6**, which prints all

the Fibonacci values from the `result`.

That's it about the solution. In the next lesson, you'll study a concept of multidimensionality.