Solution Review: Compute Factorial of a Number

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

```
package main
import (
        "fmt"
)

func main() {
        for i := uint64(0); i < uint64(22); i++ {
                fmt.Printf("Factorial of %d is %d\n", i, Factorial(i)) // calculating factori
        }
}

// named return variables:
func Factorial(n uint64) (fac uint64) {
        if n<=1{        //base case
                return 1
        }
        fac = n * Factorial(n-1)        // recursive case
        return
}
```

Factorial of a Number

In `main`, we are calling function `Factorial` for numbers between **0** and **21** inclusively using `for` loop at **line** 7, and printing the `result` for every number. See the `Factorial` function. We always need a *base* and *recursive* case to implement the recursive function. We know that when the number is less than or equal to **1**, we stop the cycle and set the value to *1*. This is the base case. The following code implements the base case:

```
if n<=1{ //base case
  return 1
}
```

Next, we have a recursive case (for `n`>1). You may have noticed the pattern. The factorial of a number `n` is equal to :

```
Factorial(n) = n * Factorial(n-1)
```

This pattern is implemented at **line 17**. We start solving this problem from smaller instances, which in turn solve problems for bigger instances.

That’s it about the solution. In the next lesson, you’ll study *higher-order functions*.