作业

1、求n的阶乘。至少使用递归完成一次。

```go
package main

import (
    "fmt"
)

// 循环版
func Factorialv1(n int) int {
    if n < 1 {
        panic("不能小于1")
    }
    p := 1
    for i := 1; i <= n; i++ {
        p *= i
    }
    return p
}

// 递归1 公式
func Factorialv2(n int) int {
    if n < 1 {
        panic("不能小于1")
    } else if n == 1 {
        return 1
    }
    return n * Factorialv2(n-1)
}

// 递归2 循环改递归
// p是初始值1
func Factorialv3(n int, p int) int {
    if n < 1 {
        panic("不能小于1")
    } else if n == 1 {
        return p
    }
    return Factorialv3(n-1, p*n)
}

func main() {
    fmt.Println(Factorialv1(6))
    fmt.Println(Factorialv2(6))
    fmt.Println(Factorialv3(6, 1))
}
```


2、编写一个函数，接受一个参数n，n为正整数。要求数字必须对齐

```
上三角
                         1
                       2 1
                     3 2 1
                   4 3 2 1
                 5 4 3 2 1
               6 5 4 3 2 1
             7 6 5 4 3 2 1
           8 7 6 5 4 3 2 1
         9 8 7 6 5 4 3 2 1
       10 9 8 7 6 5 4 3 2 1
    11 10 9 8 7 6 5 4 3 2 1
12 11 10 9 8 7 6 5 4 3 2 1
```

上三角所需要的所有数字，实现如下

```go
package main

import "fmt"

func main() {
    for i := 1; i < 13; i++ {
        for j := i; j >= 1; j-- {
            fmt.Print(j, " ")
        }
        fmt.Println()
    }
}
```

上面代码至少接近了，只需要实现右对齐，似乎就可以了。但是右对齐最大的问题是一行的宽度，这个问题这么解决呢?

大概估计一下最后一行的宽度就行，给一个较大的宽度就可以了。或者计算出最后一行的宽度。

```go
package main

import (
    "fmt"
)

func main() {
    width := 40
    var line string
    for i := 1; i < 13; i++ {
        line = ""
        for j := i; j >= 1; j-- {
            line += fmt.Sprint(j) + " "
        }
        fmt.Printf("%[2]*[1]s\n", line, width)
        // fmt.Printf("%[1]*s\n", width, line)
    }
}
```

如果能知道最后一行宽度，更加完美。那怎么做呢？

```go
package main

import (
    "fmt"
)

func PrintTriangle(n int) {
    last := ""
    for i := n; i >= 1; i-- {
        last += fmt.Sprintf("%d ", i)
    }

    width := len(last)
    var line string
    for i := 1; i < n; i++ {
        line = ""
        for j := i; j >= 1; j-- {
            line += fmt.Sprintf("%d ", j)
        }
        fmt.Printf("%*s\n", width, line)
    }
    fmt.Println(last)
}

func main() {
    PrintTriangle(12)
}
```


两层循环太多了，根据上面的代码是否有所启发？由此，有另一种思路

```go
package main

import (
    "fmt"
)

func PrintTriangle(n int) {
    last := ""
    for i := n; i >= 1; i-- {
        if i == 1 {
            last += fmt.Sprint(i) // 去掉最后的空格
        } else {
            last += fmt.Sprintf("%d ", i)
        }
    }
    width := len(last)
    // fmt.Println(width, last)
    for i := width - 1; i >= 0; i-- {
        if last[i] == 32 {
            fmt.Printf("%*s\n", width, last[i:])
        }
```

```
    }
    fmt.Println(last)
}

func main() {
    PrintTriangle(12)
}
```