函数返回值

我们通过 `return` 关键字返回一组值。事实上，任何一个有返回值（单个或多个）的函数都必须以 `return` 或 `panic` 结尾。

在函数块里面，`return` 之后的语句都不会执行。如果一个函数需要返回值，那么这个函数里面的每一个代码分支（code-path）都要有 `return` 语句。

有些函数只是完成一个任务，并没有返回值。我们仅仅是利用了这种函数的副作用，就像输出文本到终端，发送一个邮件或者是记录一个错误等。

但是绝大部分的函数还是带有返回值的。

函数可以有0或多个返回值，返回值需要指定数据类型，返回值通过`return`关键字来指定。

1. `return`可以有参数，也可以没有参数，这些返回值可以有名称，也可以没有名称。Go中的函数可以有多个返回值。
2. `return`关键字中指定了参数时，返回值可以不用名称。如果`return`省略参数，则返回值部分必须带名称。
3. 当返回值有名称时，必须使用括号包围，逗号分隔，即使只有一个返回值。
4. 但即使返回值命名了，`return`中也可以强制指定其它返回值的名称，也就是说`return`的优先级更高。
5. 命名的返回值是预先声明好的，在函数内部可以直接使用，无需再次声明，命名返回值的名称不能和函数参数名称相同，否则报错提示变量重复定义。就像在函数体开头声明的变量那样使用。返回值的名称应当具有一定的意义，可以作为文档使用。
6. 命名返回参数可看做与形参类似的局部变量，最后由 return 隐式返回。
7. 命名返回参数可被同名局部变量遮蔽，此时需要显式返回。
8. 命名返回参数允许 defer 延迟调用通过闭包读取和修改。
9. `return`中可以有表达式，但不能有赋值表达式，例如`return a+b`这是正确的，但`return c=a+b`是错误的。
10. `"_"`标识符，用来忽略函数的某个返回值。
11. 返回值可以被命名，并且就像在函数体开头声明的变量那样使用。
12. 没有参数的 return 语句返回各个返回变量的当前值。这种用法被称作“裸”返回。
13. 多返回值可直接作为其他函数调用实参。

示例：

```go
package main

import (
    "fmt"
)

// 没有返回值
func test(msg string) {
    fmt.Printf("msg: %v\n", msg)
}

// 有一个返回值
func add(a, b int) (c int) {
    c = a + b
```

```
    // return “裸”返回隐式返回
    return c
}

// 有多个返回值
func calc(a, b int) (sum int, avg int) {
    sum = a + b
    avg = (a + b) / 2

    return sum, avg //也可以直接 return裸返回 隐式返回
}

func main() {
    var a, b int = 1, 2
    c := add(a, b)
    sum, avg := calc(a, b)
    fmt.Printf("c: %v\n", c)
    fmt.Printf("sum: %v\n", sum)
    fmt.Printf("avg: %v\n", avg)

    //只想要一个值，那么另一个可以用_忽略
    x, _ := calc(a, b)
    fmt.Printf("x: %v\n", x)
    _, y := calc(a, b)
    fmt.Printf("y: %v\n", y)

    //多返回值可直接作为其他函数调用实参
    s := add(calc(a, b))
    fmt.Printf("s: %v\n", s)
}
#结果
c: 3
sum: 3
avg: 1
x: 3
y: 1
s: 4
```