《《Golang常见内置函数》》

【new，make】：均用于分配内存，不过new用于值类型和用户定义类型，如自定义的结构，make用于内置引用类型，包括slice，map和channel。他们的用法就像是函数，但是将类型作为参数：new(type)、make(type)。new(T)分配类型T的零值病返回其地址，也就是指向类型T的指针，它也可以用于基本类型：v := new(int)。make(T)返回类型T的初始化之后的值，因此它比new做更多的工作。new()是一个函数，不要忘记它的括号。

【close】：用于管理通信

【len】：用于返回某个类型的长度或数量（字符串，数组，切片、map和channel）

【cap】：容量的意思，用于返回某个类型的最大数量，只能用于切片和map

【copy、append】：用于赋值和连接切片

【panic、recover】：用于错误处理机制

【print、println】底层打印函数，部署环境建议使用fmt包

【complex、real、image】：用于创建和操作复数

《《Golang函数参数传递》》

Go语言的参数传递可以分为“按值传递”和“按引用传递”。Go语言默认按值传递，传递的是参数的副本，函数接收参数副本后，使用变量的过程中可能对副本 的值进行修改，但不会影响原来的变量。换句话说，调用函数时修改参数的值，不会影响原来的实参的值，因为数值变化只作用的副本上。

所以如果要让函数直接修改参数的值，而不是对参数的副本进行修改，就需要将参数的地址（变量名前加&符号，比如&variable）传递给函数，这就是“按引用传递，此时传递给函数的是一个指针。

如果把指针传递给函数，指针的值（一个地址）就会被复制，但指针的值指向的地址上的那个值不会被复制（被复制的是指针，但是两个指针指向同一个实际的值）。这样一来，修改这个指针的值，实际上意味着这个值所指向的地址上的值被修改了（别忘了指针也是变量类型，有自己的地址和值，通常指针的值指向一个变量的地址，所以说到底“按引用传递”其实也是值传递）。

传指针的三点好处：1. 传指针使得多个函数能够操作同一个对象；2. 传指针比较轻量级；3.传递指针给函数不仅可以节省内存，而且赋予了函数直接修改外部变量的能力，所以被修改的变量不再需要return返回。

不过需要注意的是，golang中的slice，map，channel这三种类型的实现机制类似指针，所以可以直接传递，而不用取地址后传递指针。不过若函数需改变slice的长度，则仍需要取地址传递指针。

《defer、return、返回值执行逻辑》

第一步：Return最先给返回值赋值，若为有名返回值则直接赋值，若为匿名返回值则先声明再赋值

第二步：调用RET返回指令并传入返回值，RET会检查defer是否存在，如果存在就先逆序插播执行defer语句

第三步：RET携带返回值退出函数

可以看出，return并不是一个原子操作，函数返回值与return返回值并不一定一致。

需要特别提醒的是，defer声明时会先计算确定参数的值，defer推迟执行的仅是其函数体，因此defer语句位置并非随意，defer的初始化还是受到外部影响的