1. 数组

数组 c := […]int{6,7,8,9}

切片 c := []int{6,7,8,9}

1. 切片

切片是对数组的引用,处理数组的函数可以传入arr[:]表示数组的全切片

数组本身的长度是固定的，那么切片如何具有动态长度。以及内部发生了什么，当新的元素被添加到切片时，会创建一个新的数组。现有数组的元素被复制到这个新数组中，并返回这个新数组的新切片引用。现在新切片的容量是旧切片的两倍

1. i := make([]int,5,5)
2. i = append(i,99)
3. fmt.Println(i,cap(i))
4. var names []string
5. names = append(names,"ee","ss")
6. fmt.Println(names)



3.方法

相同的名字的方法可以定义在不同的类型上，而相同名字的函数是不被允许的

func (r Rectangle) Area() int {

return r.length \* r.width

}

func (c Circle) Area() float64 {

return math.Pi \* c.radius \* c.radius

}

func main() {

r := Rectangle{

length: 10,

width: 5,

}

fmt.Printf("Area of rectangle %d\n", r.Area())

c := Circle{

radius: 12,

}

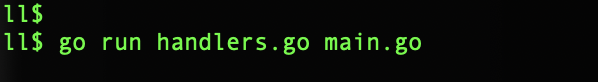
fmt.Printf("Area of circle %f", c.Area())

}

4.同包不同文件引入

https://blog.csdn.net/coolboyzero/article/details/77946653

同包不同文件的函数可以直接用,但需要通过go install编译后再用,用go run的话go run后面需要跟上用到的所有文件最后跟main.go



5.结构体

https://blog.csdn.net/benben\_2015/article/details/79807792

给结构体加标签使序列号的时候更方便

type Err struct {

    Error string 'json:"error"'

    ErrorCode string 'json:"error\_code"'

}

6.原子性操作  
栈内存：线程是私有的，也就是说局部变量和方法是不可共享的。

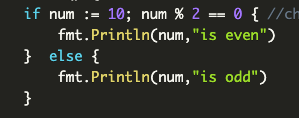
堆内存：对象和数组是在堆内存中创建的，所有线程都可以访问，包括成员变量、静态变量和数组元素是可共享的；

原子性操作：一个或某几个操作只能在一个线程执行完之后，另一个线程才能开始执行该操作，也就是说这些操作是不可分割的，线程不能在这些操作上交替执行。

<https://blog.csdn.net/dyhdengyahui/article/details/80014076>

7.if

可以在条件前加个声名语句,声名语句里的变量只能在if里用



8.函数返回值  
 两个返回值,只想用其中一个  
 q,\_ := func\_x(a,b)

1. go属于按值传递

函数需要改变外部变量的值需要传递该指针进去