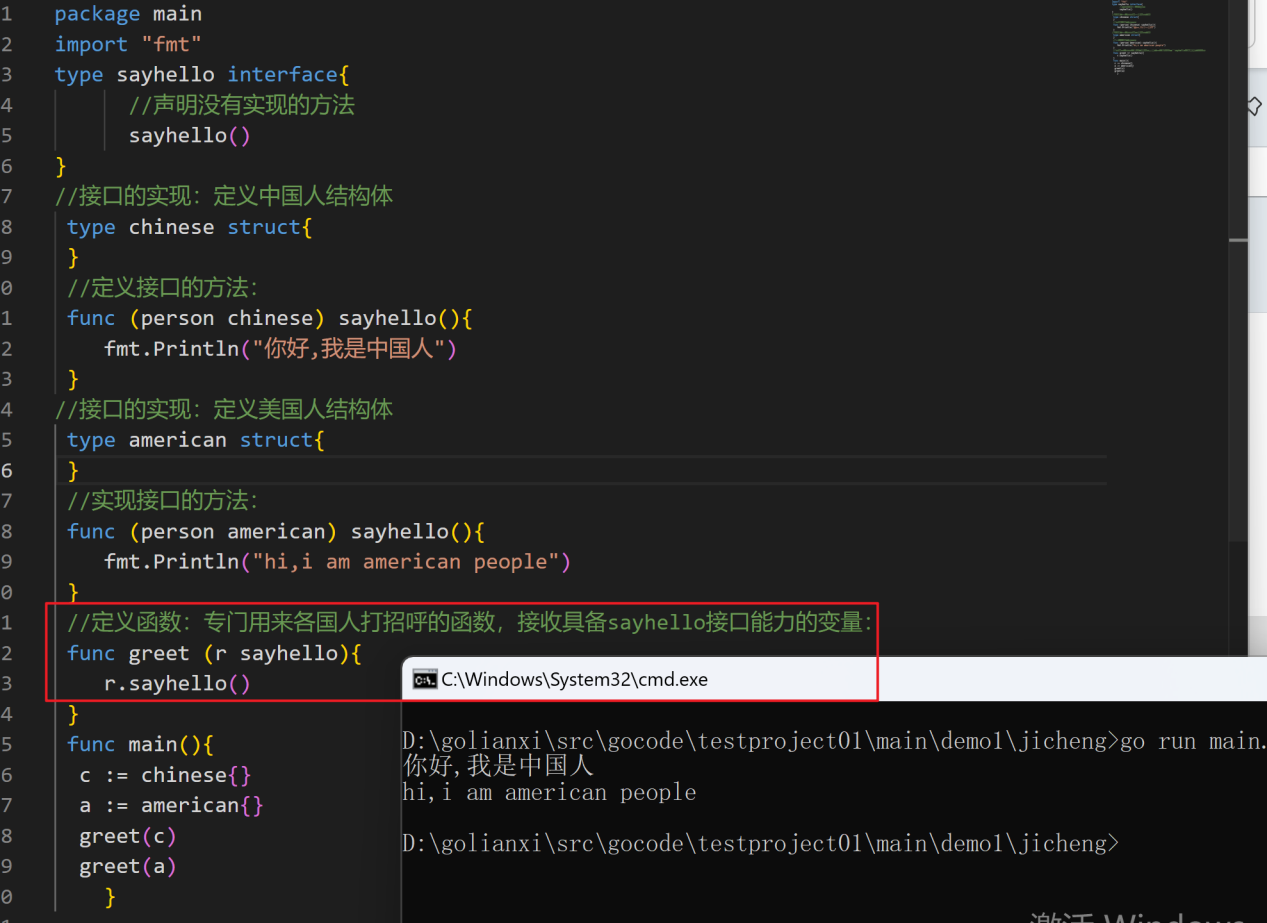
第四章

接口的实现方式：

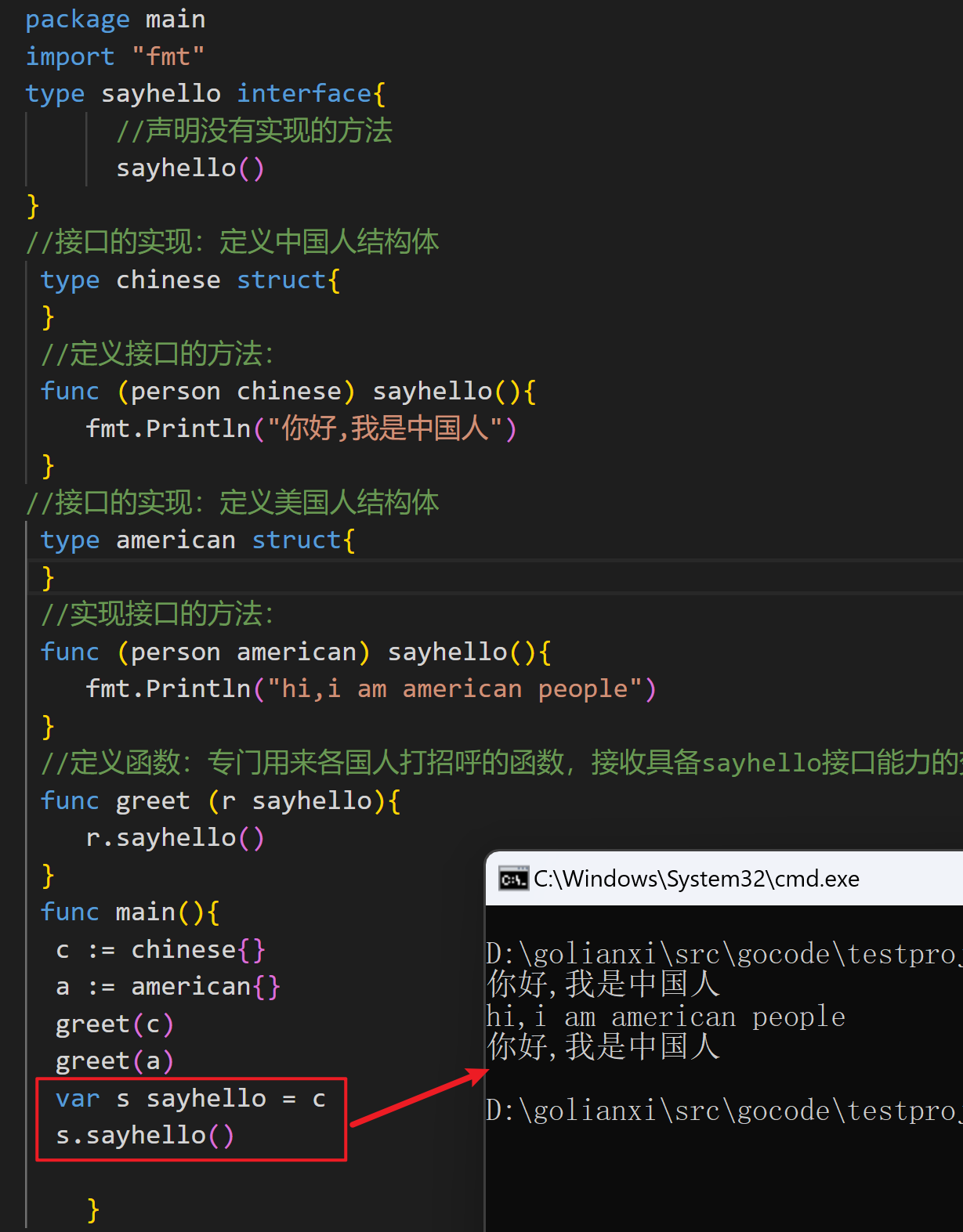


总结：

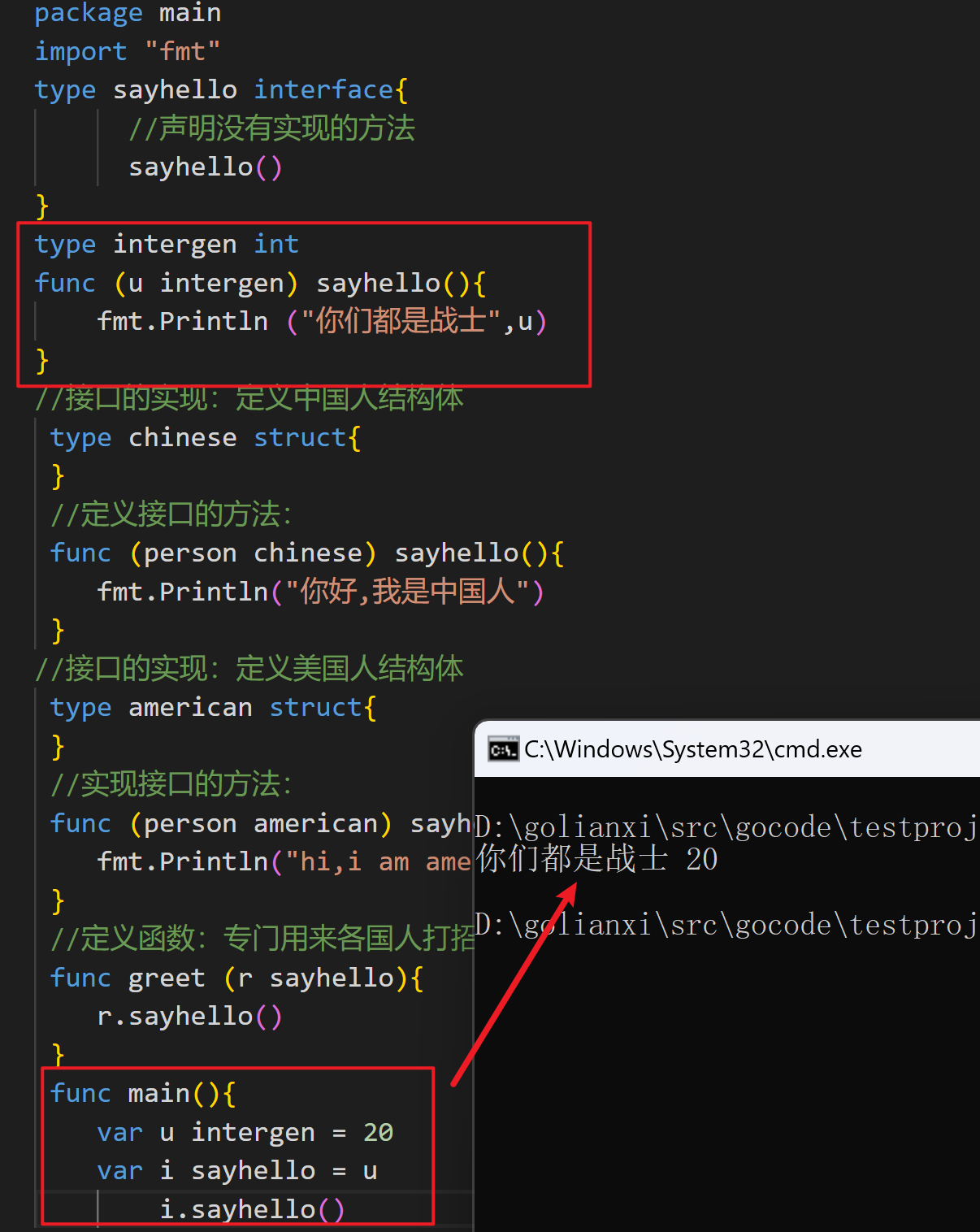
1. 接口中可以定义一组方法，但不需要实现，不需要方法体，并且接口中不能包含任何变量，当某个自定义类型要使用的时候（实现接口的时候），再根据具体情况把这些方法实现出来
2. 接口实现要实现所有的方法，才是实现
3. Golang中实现接口是基于方法，不是基于接口
4. 接口目的是为了定义规范，具体由别人来实现即可

接口的注意事项：

【1】接口本身不能创建实例，但是可以指向一个实现了该接口的自定义类型的变量

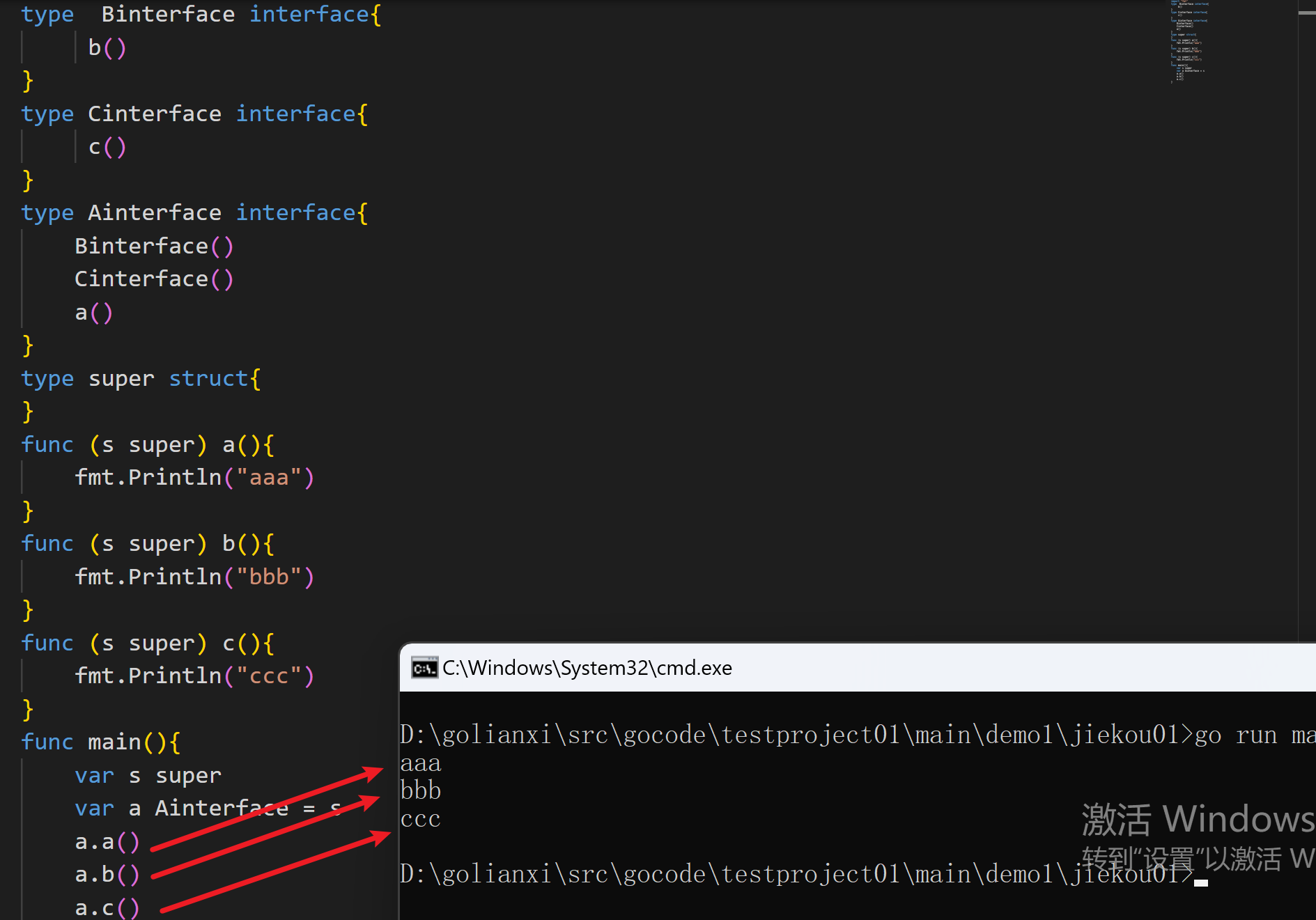


【2】只要是自定义数据类型，就可以实现接口，不仅仅是结构体类型



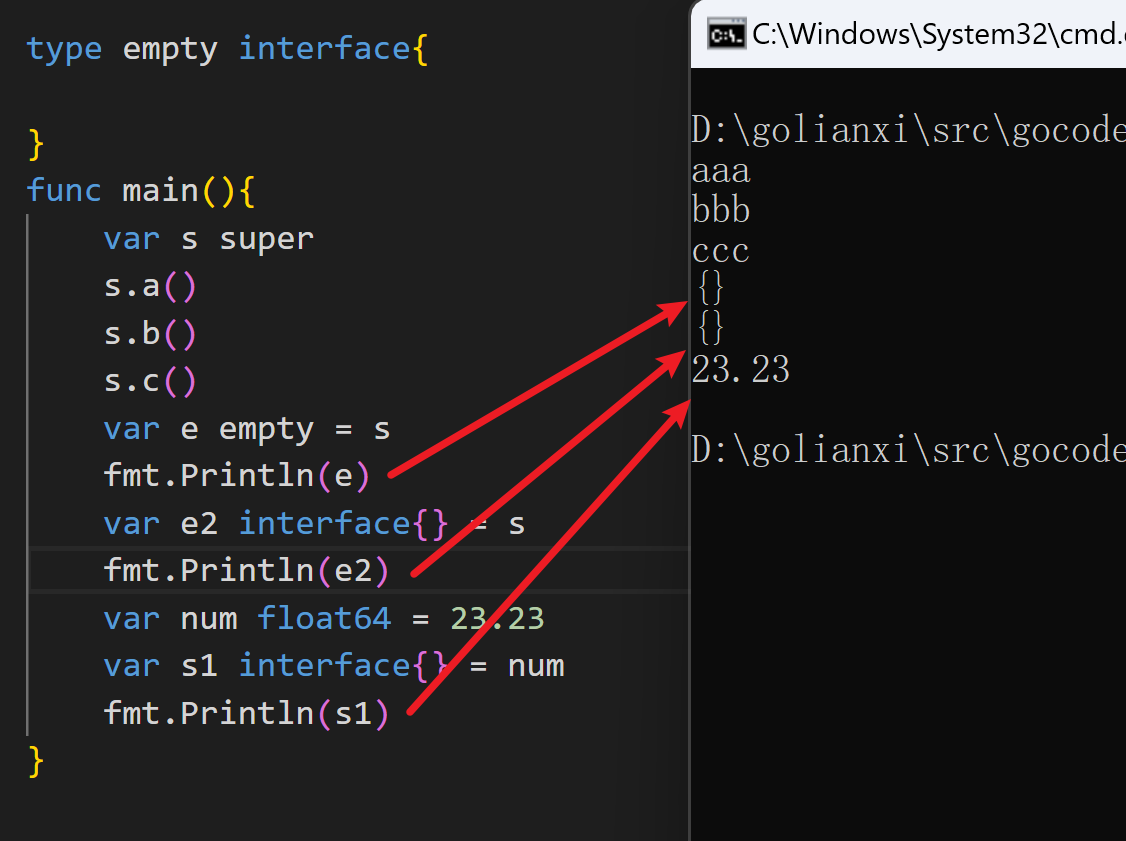
【3】一个自定义类型可以实现多个接口

【4】一个接口（比如A接口）可以继承多个别的接口（比如B，C接口），这时如果要实现A接口，也必须将B，C接口的方法全部实现



【5】interface类型默认是一个指针（引用类型），如果没有对interface进行初始化就使用，会输出nil

【6】空接口没有任何方法，所以可以理解为所有类型都实现了空接口，也可以理解为我们可以把任何一个变量赋给空接口。

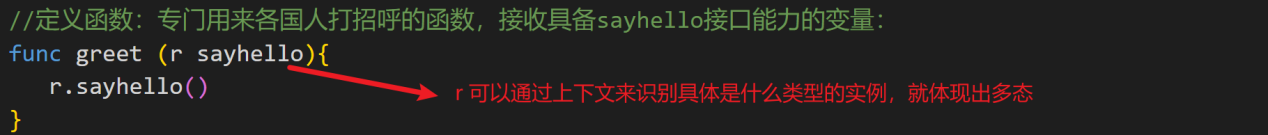


多态

【1】基本介绍

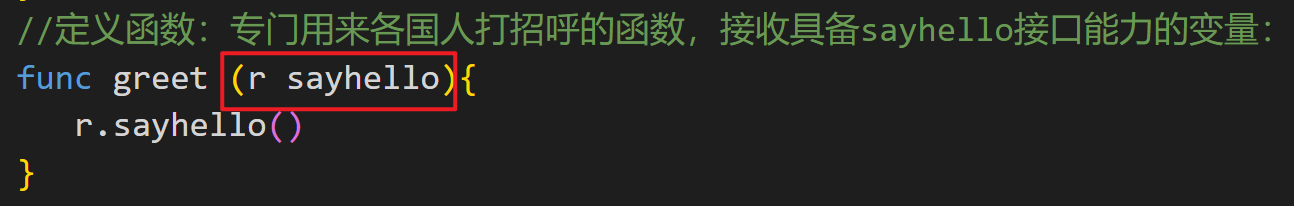
变量（实例）具有多种形态，面向对象的第三大特征，在go语言中，多态特征是通过接口实现的。可以按照统一的接口来调用不同的实现。这时接口变量就呈现不同的形态

【2】案例：

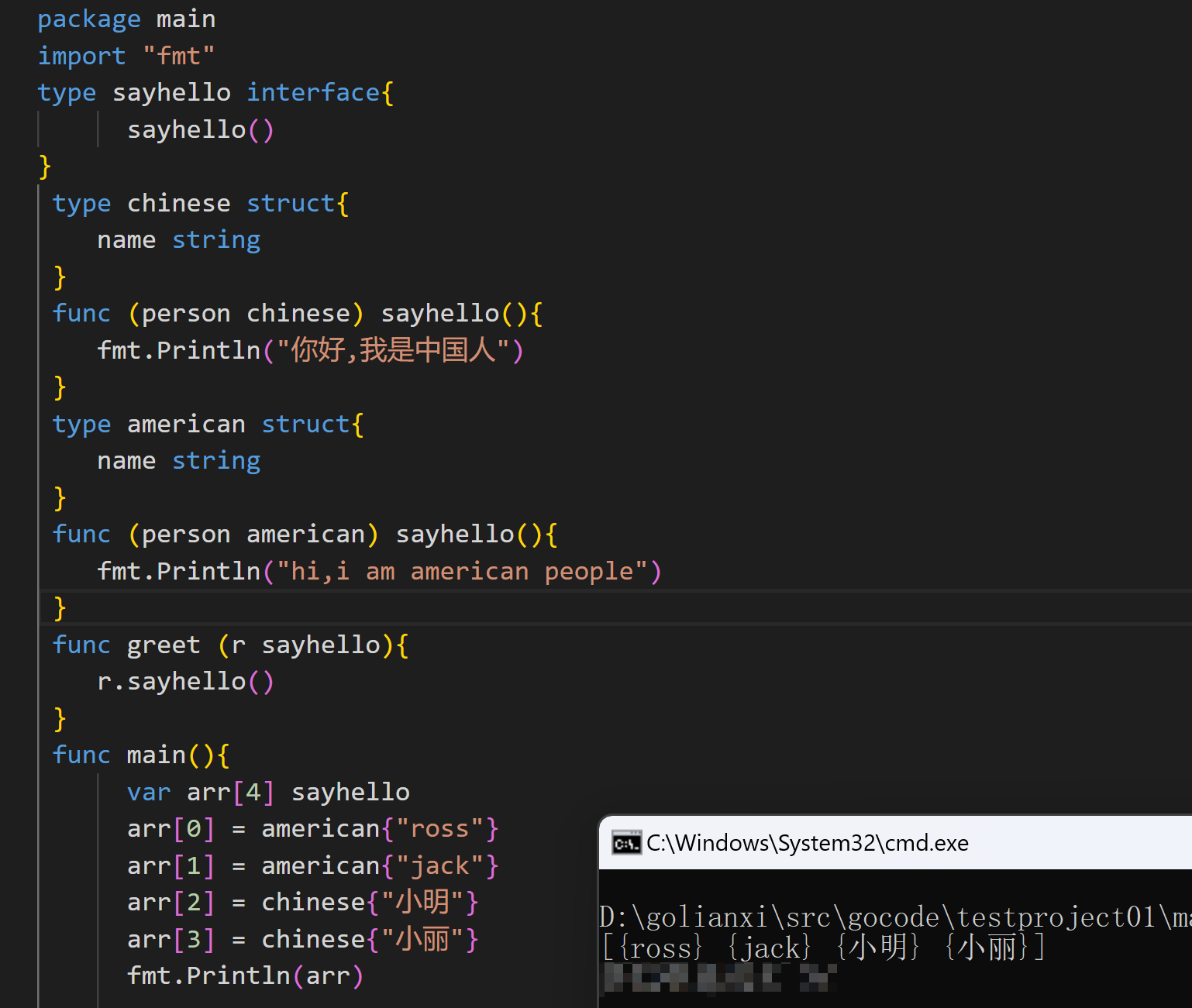


【3】接口体现多态特征

1. 多态参数：



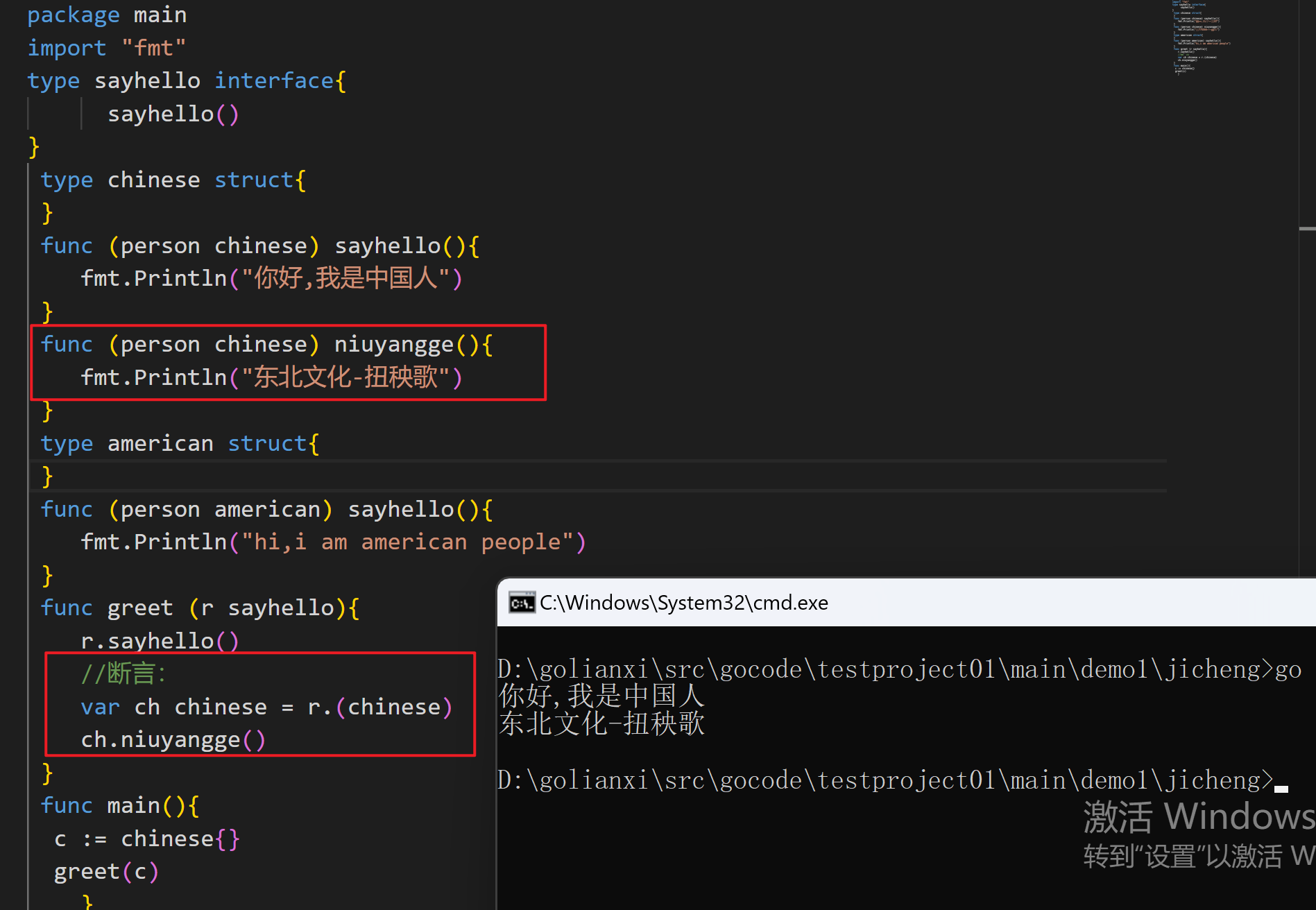
1. 多态数组：



【1】什么是断言？

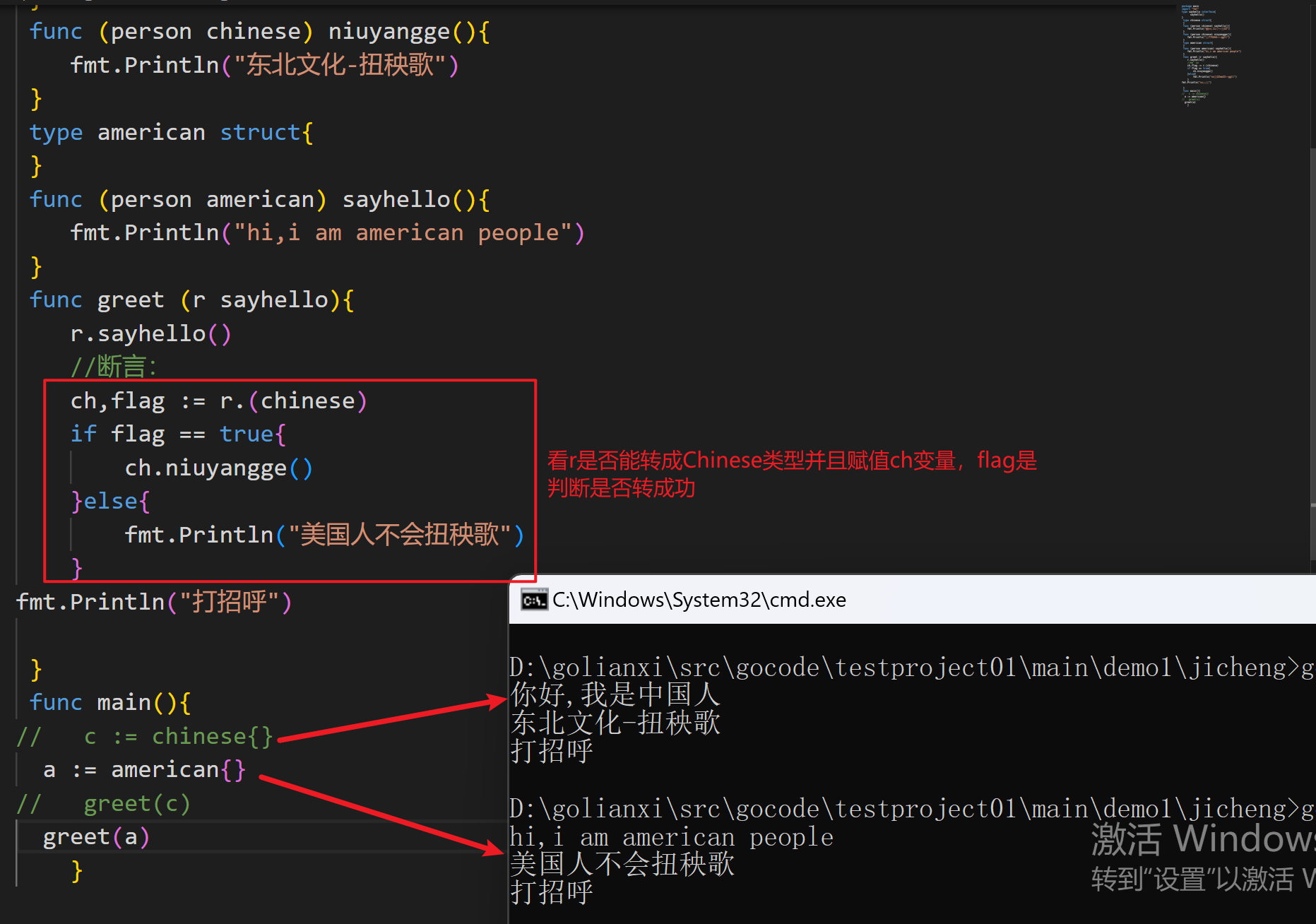
Go语言里面有一个语法，可以直接判断是否是该类型的变量

案例



断言的几种书写格式：

1.If else的用法



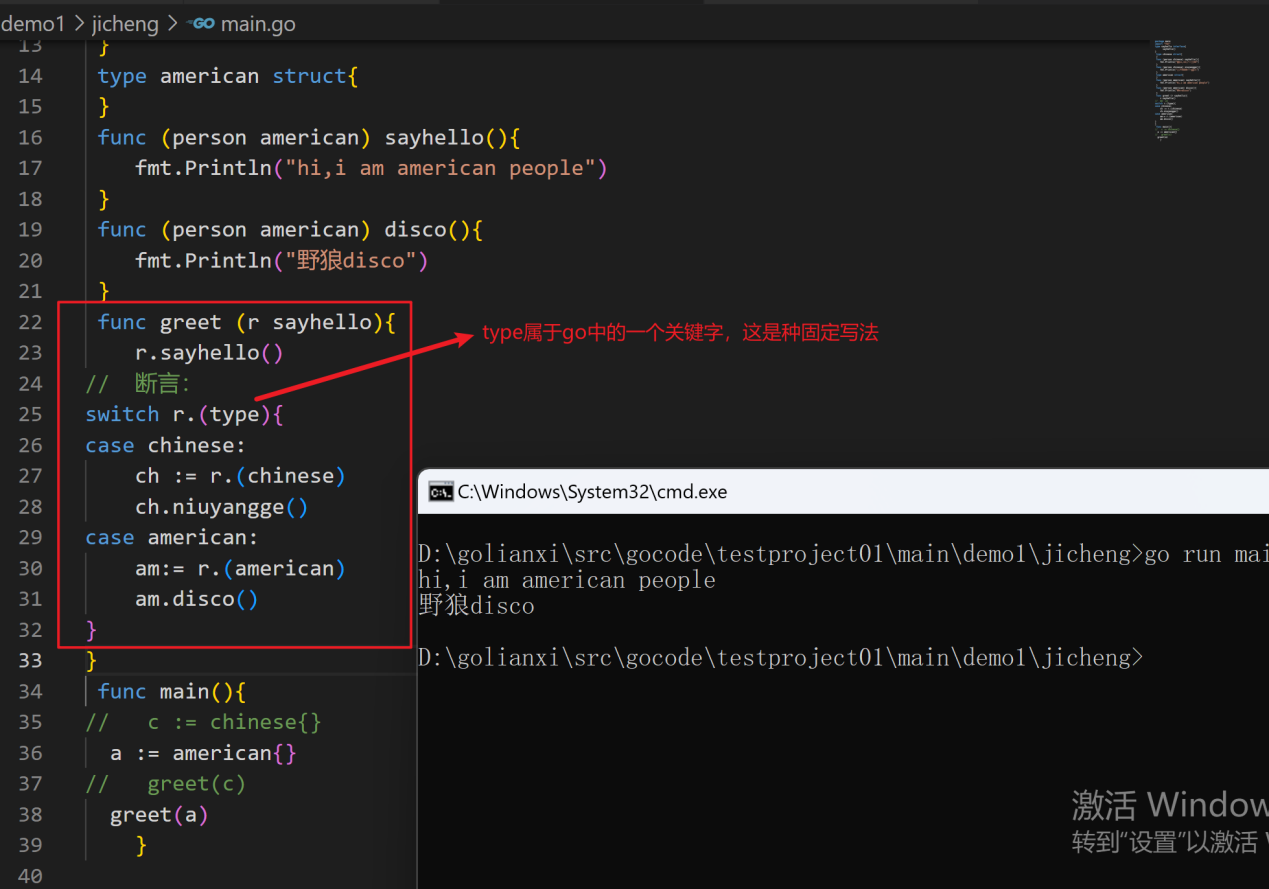
解决第二个返回值问题

2.还有类似简化写法，效果也是一样的

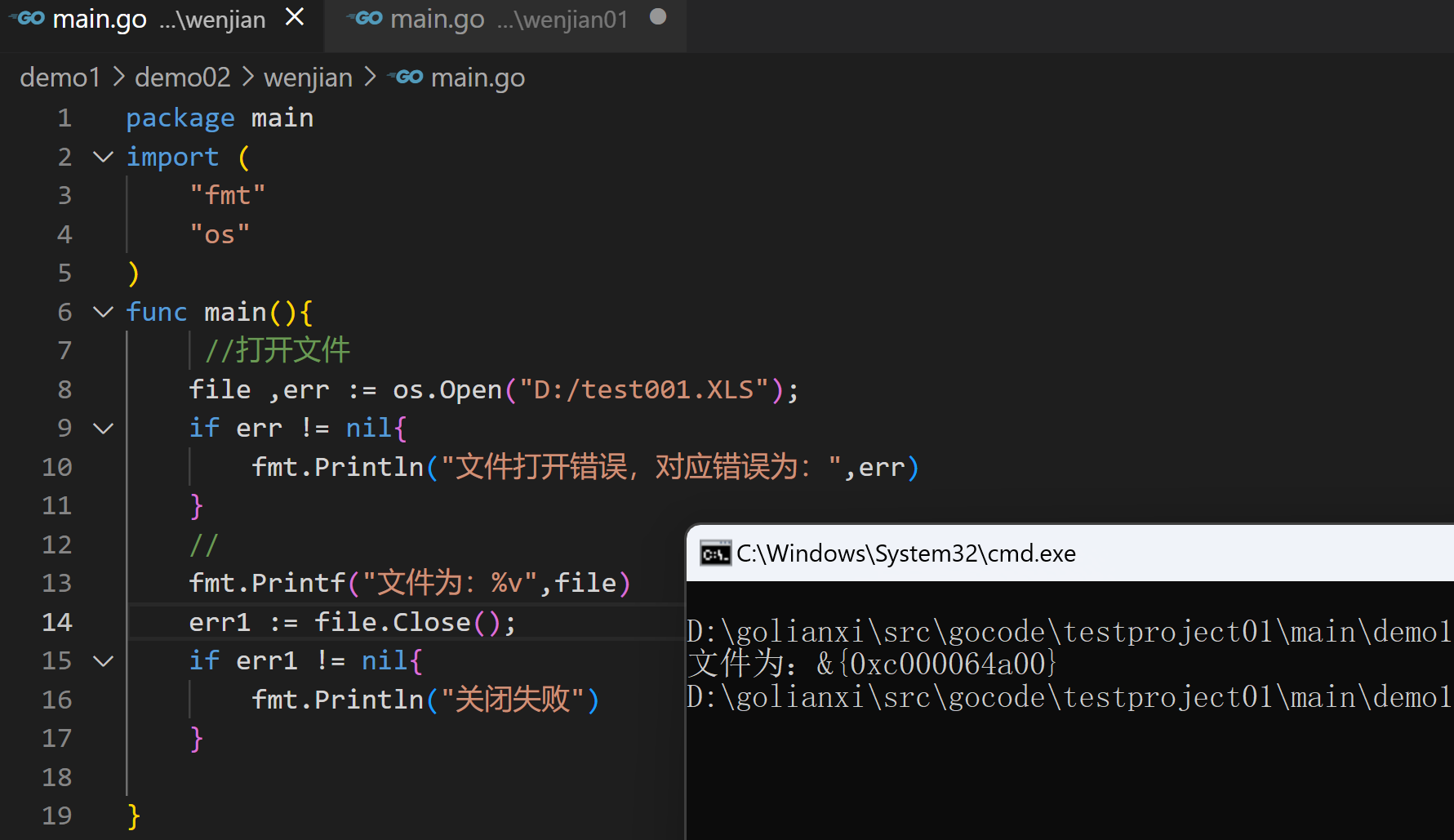


1. type switch的基本用法：

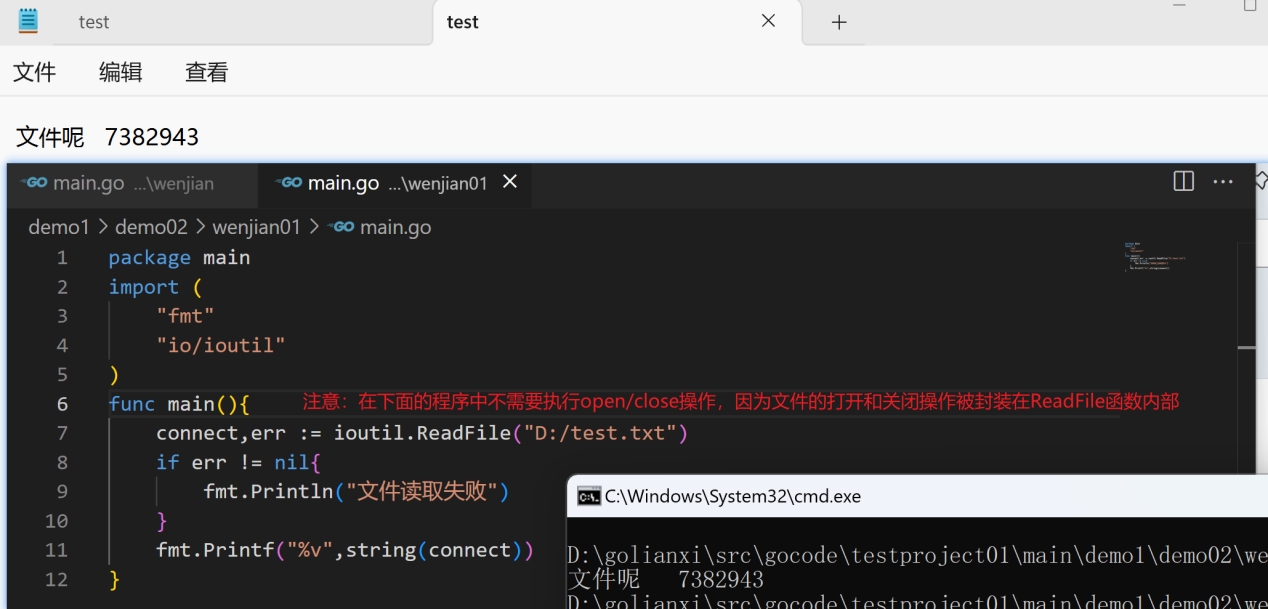
Type switch是go语言中一种特殊的switch语句，它比较的是类型而不是具体的值，它判断某个接口变量的类型，然后根据具体类型再做相应处理



文件的打开及关闭：



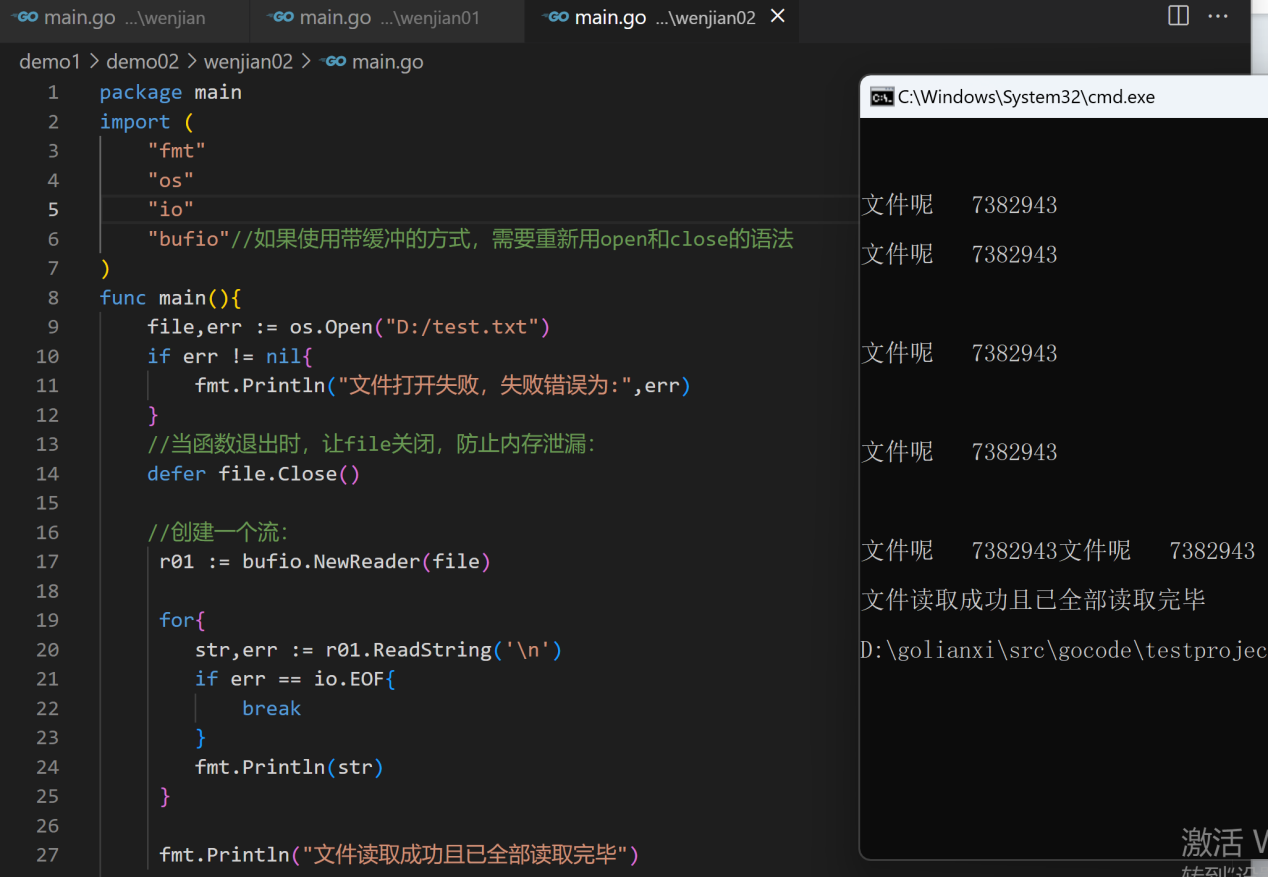
文件的读取操作：



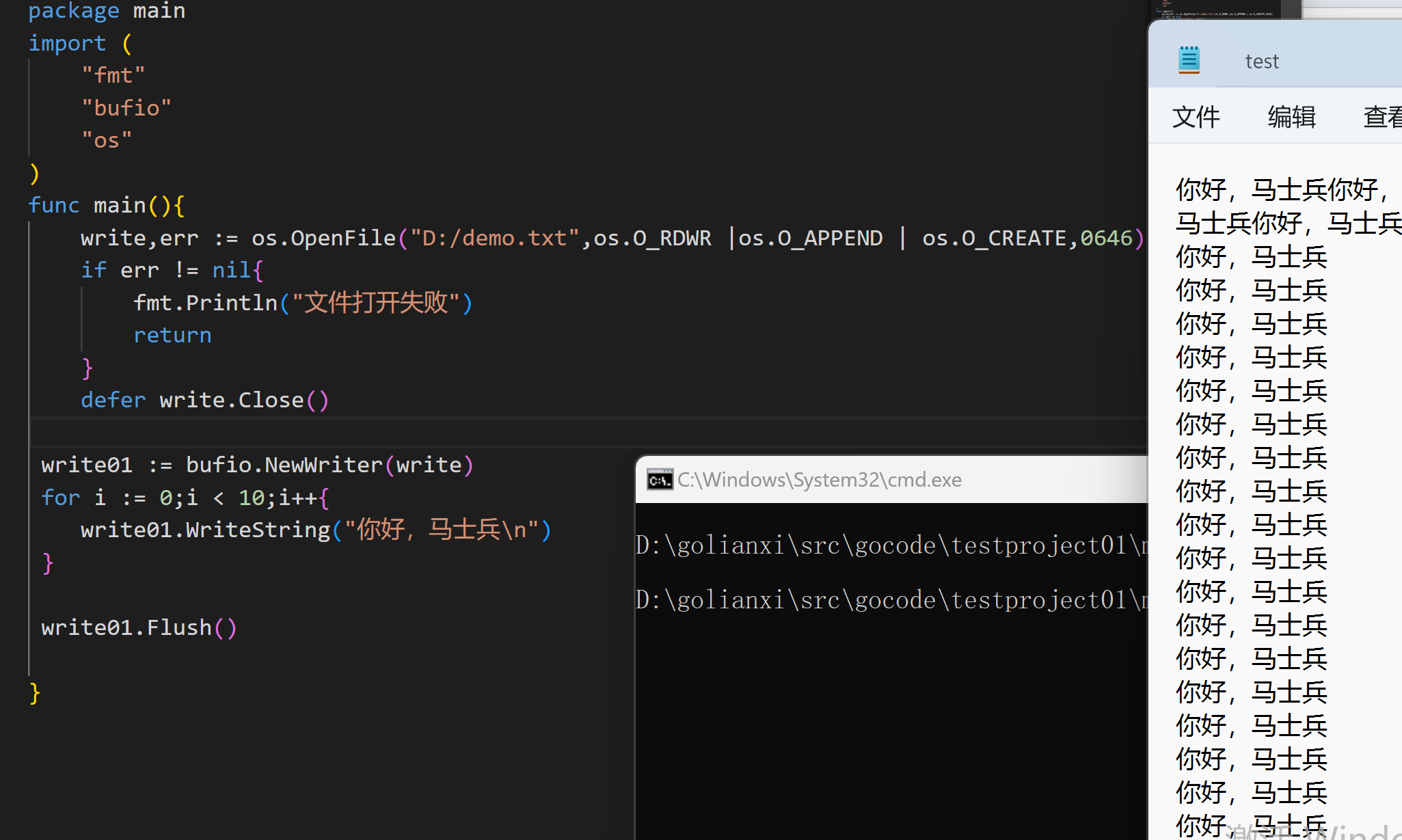
注意：这种只能适应于小文件的格式，如果要读取比较大的文件，需要使用有缓冲的方式

带缓冲的方式如下：

【1】读取文件的内容并显示在终端（带缓冲区的方式4096），适合读取比较大的文件，使用os.Open,file.Close,bufio.NewReader(),reader.ReadString函数和方法。



【2】写入文件



【3】文件复制

