

## 基础篇 第四回

### 追根溯源探究变量本色

### 强制声明避免张冠李戴

<http://study.163.com/course/courseLearn.htm?courseId=1003088001#/learn/video?lessonId=1003510007&courseId=1003088001>

[3088001#/learn/video?lessonId=1003510007&courseId=1003088001](http://study.163.com/course/courseLearn.htm?courseId=1003088001#/learn/video?lessonId=1003510007&courseId=1003088001)

[088001](http://study.163.com/course/courseLearn.htm?courseId=1003088001#/learn/video?lessonId=1003510007&courseId=1003088001)

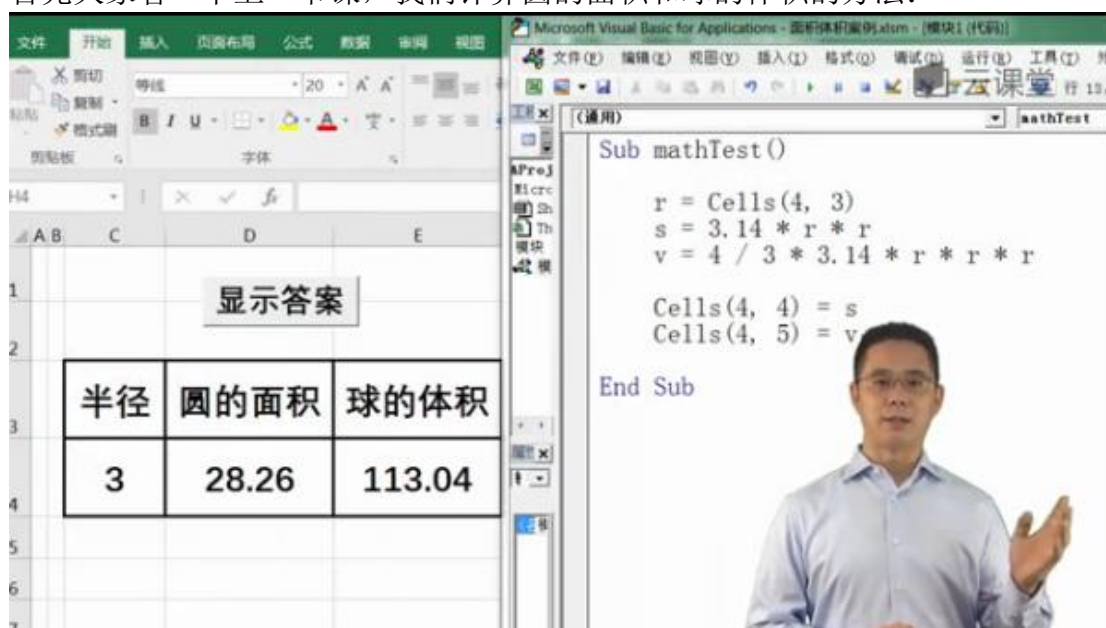
上次课我们讲了怎样使用我们的变量，使我们的程序更加灵活，能够根据需要处理任何一个数字和单元格，由于变量是我们计算机里面最基本和最重要的单元，所以这节课里，我们继续围绕它加深大家对他的理解。具体说来我们主要介绍三个知识点：

[1]变量的本质

[2]什么是强制声明

[3]怎样使用常量

首先大家看一下上一节课，我们计算圆的面积和球的体积的方法：



The screenshot shows a video lecture interface. On the left, a Microsoft Excel spreadsheet is visible with a table containing the following data:

半径	圆的面积	球的体积
3	28.26	113.04

On the right, the Microsoft Visual Basic for Applications (VBA) editor is open, showing a subroutine named `mathTest()` with the following code:

```
Sub mathTest()  
    r = Cells(4, 3)  
    s = 3.14 * r * r  
    v = 4 / 3 * 3.14 * r * r * r  
  
    Cells(4, 4) = s  
    Cells(4, 5) = v  
  
End Sub
```

A man in a light blue shirt is visible in the bottom right corner of the VBA window, gesturing with his hand.

首先从第四行三列取出 `r` 变量，然后计算出 `s` 和 `v` 的值，然后分别写入到单元格中。现在假设我们遇到一个问题，在程序中可能会遇到多个半径，计算多个圆的面积和球的体积，那么我们就需要将代码中的 `r` 改称 `r1` 以免和其他的半径冲突，那么怎么改呢？杨老师手动改了一下，

```

Sub mathTest()

    r1 = Cells(4, 3)
    s = 3.14 * r1 * r1
    v = 4 / 3 * 3.14 * r1 * r * r1

    Cells(4, 4) = s
    Cells(4, 5) = v

End Sub

```

看起来没什么问题，发现没有？球的体积都变成了 0.

显示答案

半径	圆的面积	球的体积
3	28.26	0

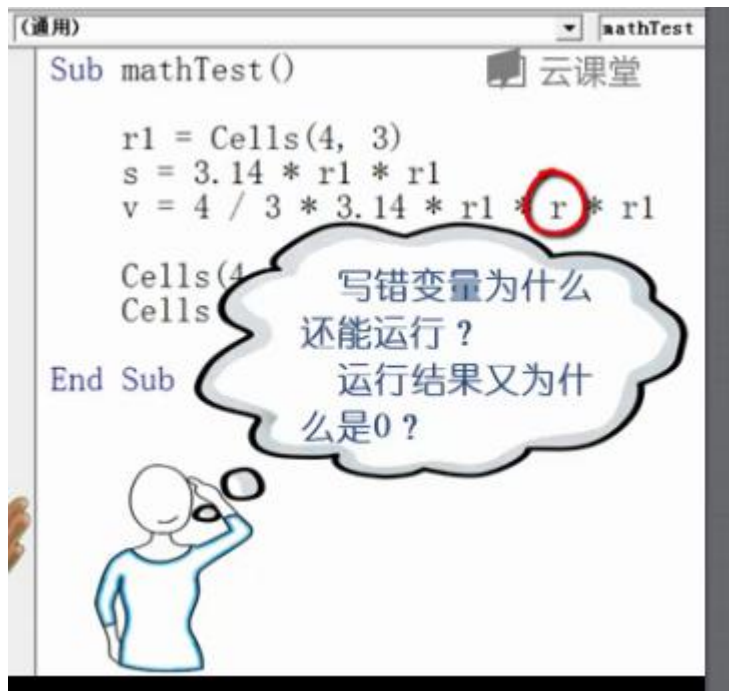
这是为什么呢？我们只是改了一个变量的名字，并没有修改计算逻辑，为什么会计算出错呢？我们查一下代码，一行一行看下来，原来是在

```

v = 4 / 3 * 3.14 * r1 * r * r1

```

这个地方少改了一个 r.不要问为什么，当时杨老师没有戴眼镜。我想有的同学会觉得奇怪，我们只是少改了一个变量的话，程序应该提醒我们出错呀？！



为什么没有提醒？还把结果算错了，还是一个零。这是为什么？要想理解这个问题，我们需要理解一下变量到底是什么？变量简单的说就是**计算机内部用于存放数值的小空间（储存位置）**。我们知道计算机在运行的时候，它的数值都是存放在它的内存当中的，也就是我们看到的计算机内存条上的一个个的小模块，



从程序的角度上想，我们可以把计算机的内存想象成空旷而又广阔的青青草原。当然上面没有喜洋洋和灰太狼，



当我们摁下这个按钮的时候，计算机就知道我们接下来要运行一个名字叫mathTest()的小程序。

The screenshot shows the Microsoft Visual Basic for Applications editor window. The macro code is as follows:

```
Sub mathTest()  
    r1 = Cells(4, 3)  
    s = 3.14 * r1 * r1  
    v = 4 / 3 * 3.14 * r1 * r1 * r1  
  
    Cells(4, 4) = s  
    Cells(4, 5) = v  
  
End Sub
```

In the background, an Excel worksheet is visible with a table containing the following data:

半径	圆的面积	球的体积
3	28.26	0

A red circle highlights the '显示答案' (Show Answer) button in the worksheet, and a red arrow points from it to the 'mathTest()' macro in the VBA editor.

于是计算机在这一瞬间决定在这一片草原上给我们自动的开辟出一块空间



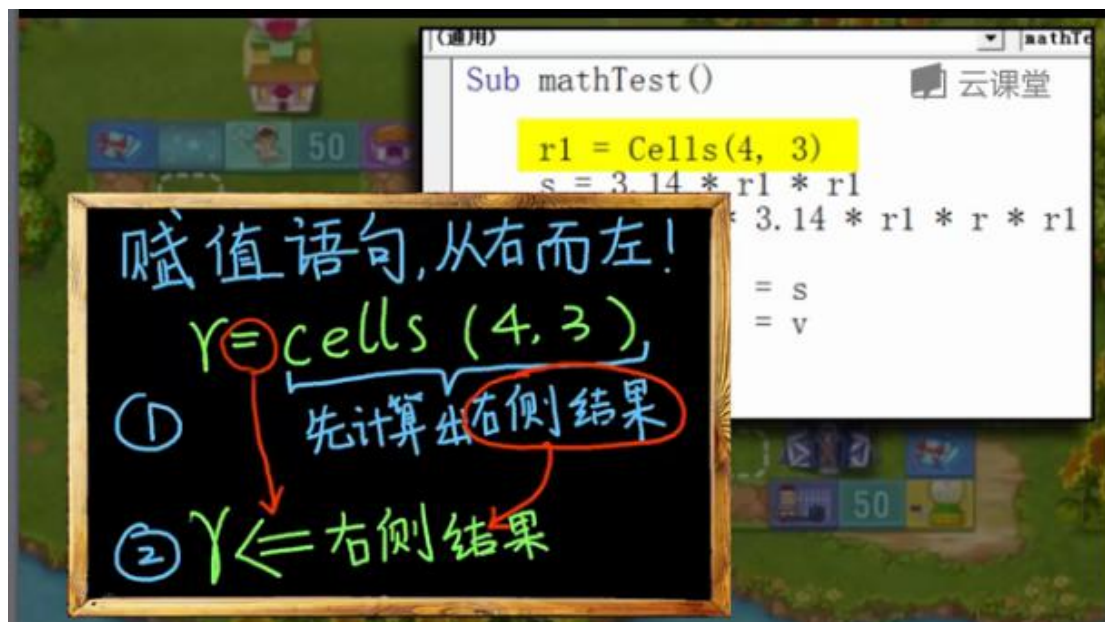
专门供 mathTest 这块内存使用，接下来我们执行第一行：

```
Sub mathTest()  
r1 = Cells(4, 3)  
s = 3.14 * r1 * r1  
v = 4 / 3 * 3.14 * r1 * r * r1  
  
Cells(4, 4) = s  
Cells(4, 5) = v  
End Sub
```

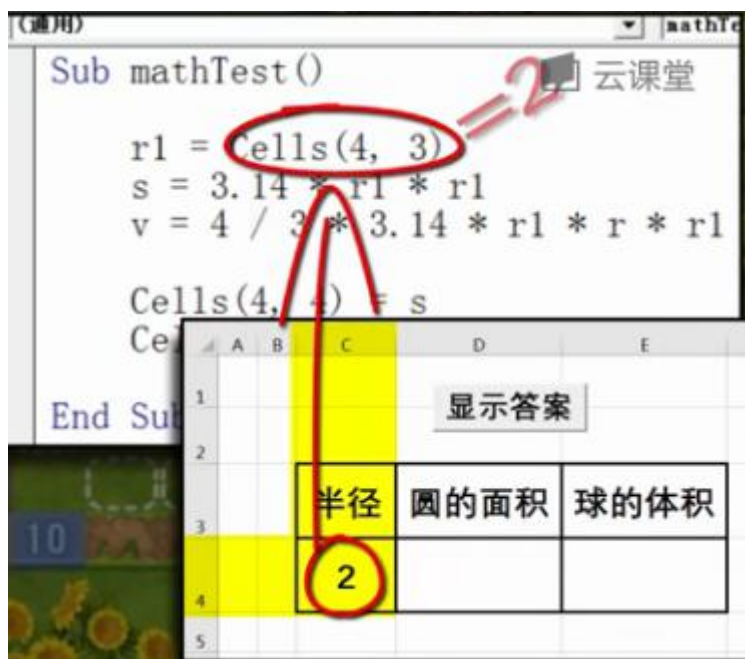
云课堂

我想大家看到这行语句，就知道这样的赋值语句都是从右往左执行的。计算机先计算右边，第四行第三列是什么，然后再把这个结果赋值给左边这个变量。

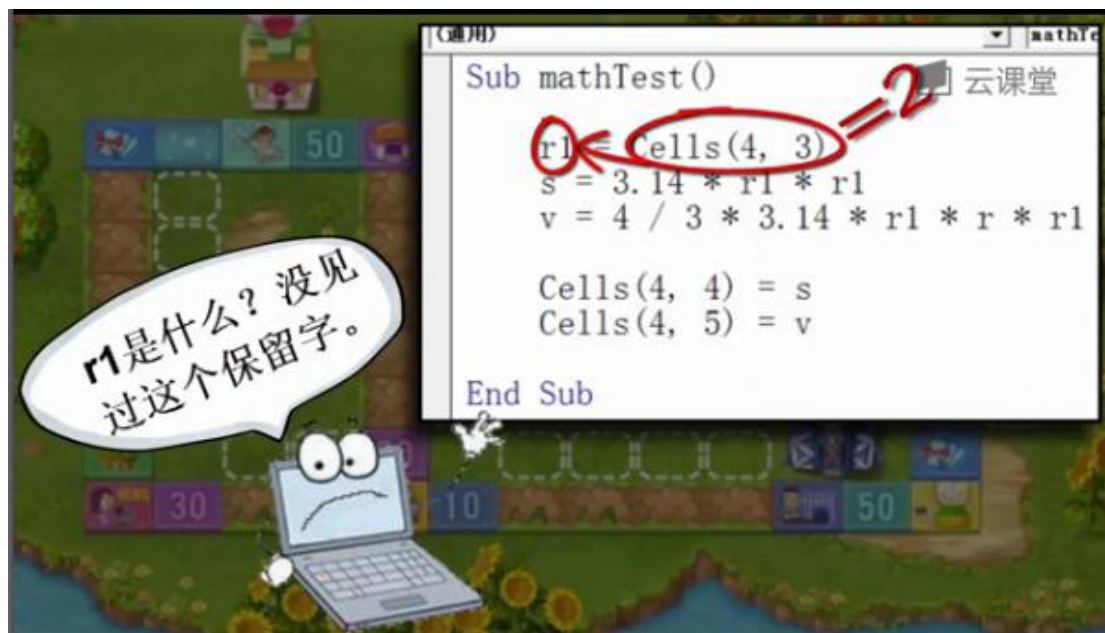




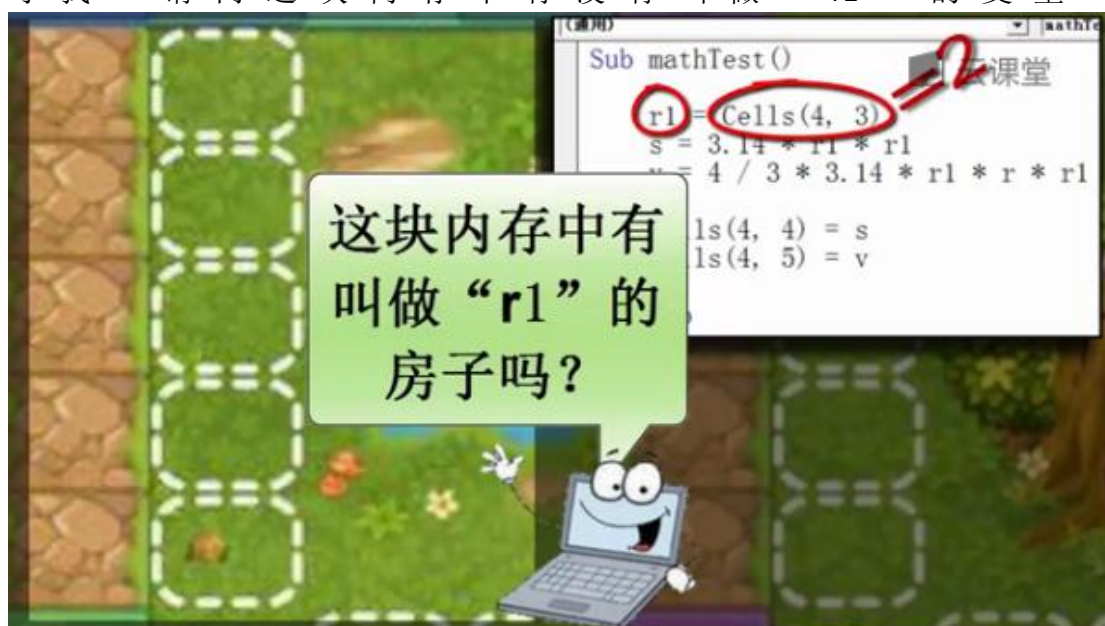
等号可以理解为箭头，求出右边之后，再赋值给左边，那么计算机先来看右边的 Cells(4,3), Cells 他是认识的，他是 VBA 里面它常用的函数，根据这个函数，它就到数据表里面的第四行第三列找到它的数值 2。



接下来他把这个值赋值给左边 r1, r1 是什么，不认识，VBA 没有这个保留字，



说明 r1 一定是一个变量，说明 r1 是一个变量，这是后 VBA 解释器就要把内存中寻找“请问这块内存中有没有叫做”r1“的变量？



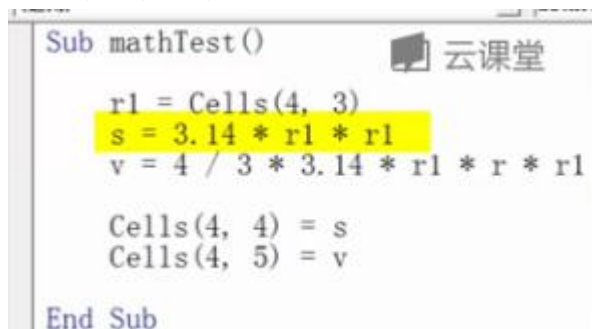
没有！没有怎么办？它就会自动的在这块空间里给我们盖一块小房子，



房子的门牌号写上 r1,里面是空的，不错马上就不空了，因为我们有一个数值 2，马上就要赋值进来，所以大家看，我们的 mathTest 这个空间里就有了个小房子，名字叫 r1,值是 2



接下来再往下执行：

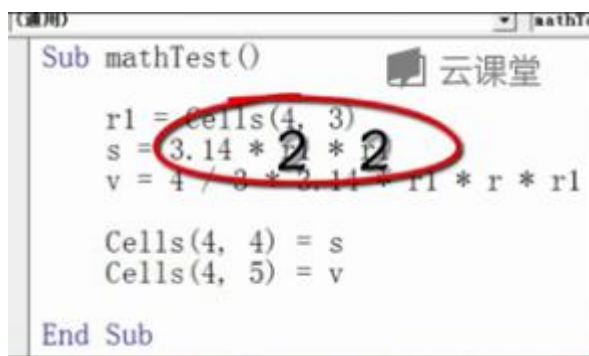


还是从右边往左执行，请问有没有一个叫 r1 的变量。





有，找到了一个门牌号叫 **r1** 的小房子，于是把里面的内容 **2** 取了出来，替换掉 **r1**，我们的右边就编程了。



将  $3.14 * 2 * 2$  得到结果 12.56，然后再把结果赋值给左边，左边是 **s**。那么再到内存里去找：请问有没有 **s** 这个小房子？那么就给他建一个小房子，门牌号是 **s**，然后把 12.56 放进去，



这就是变量的工作原理，同样的  $v$  也是一样的原理，只不过在计算  $v$  的时候，先计算右边，

```
Sub mathTest()  
    r1 = Cells(4, 3)  
    s = 3.14 * r1 * r1  
    v = 4 / 3 * 3.14 * r1 * r * r1  
    Cells(4, 4) = s  
    Cells(4, 5) = v  
End Sub
```



同样的道理，在找是否有  $r$  的房子的时候，计算机不会像杨老师一样，把  $r$  看成  $r1$ ，同样在内存中新建一个小房子，门牌号是  $r$ 。





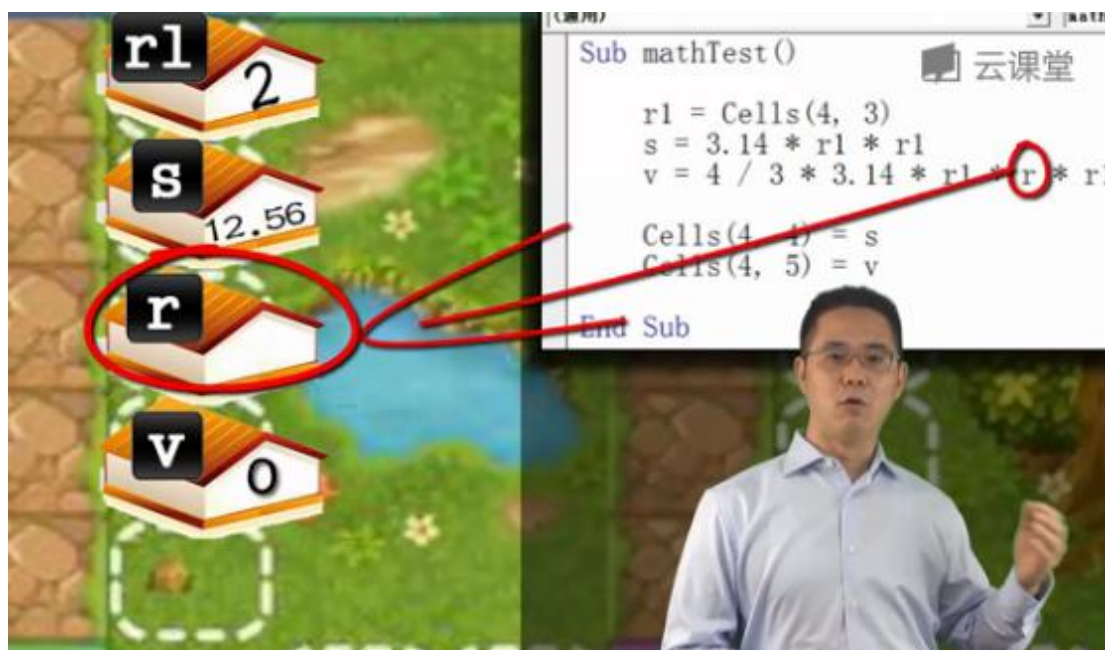
然后里面的东西是空的，接下来把它取出来，空的东西取出来什么呢？当然是取出 0 来，于是右边我们就变成了

```
s = 3.14 * r1 * r1  
v = 4 / 3 * 3.14 * 2 * 0 * 2
```

右边的计算结果为 0，再把 0 赋值给左边，新建一个小房子 v, v 就是 0.



讲到这里，我想大家就会明白了，我们的程序错在哪里了吧！程序还是能够正常的执行，只是我们多盖了一个小房子，我们意想不到的房子，名字叫 r。



好的，这就是关于变量的本质。

刚才的那个解释，可能有点啰嗦，但是对大家如何理解变量，变量和内存的关系等等非常有用，后面大家就会体会到了。那么要说明的规则是很简单的，VBA 中当遇到一个新的变量名时 VBA 会自动的给这个变量分配一个空间，无需事先声明。



并且给它的默认值为。我们可以看一下 VBA 是一个多么仁慈的开发商啊。如果我们的开发商也能给我们每个人按需分配一个小房子该多好啊，



只不过这种方式会给我们带来一个很小的麻烦，就是我们会因为一个很小笔误造成产生莫名其妙的结果，而且很难排查出来，

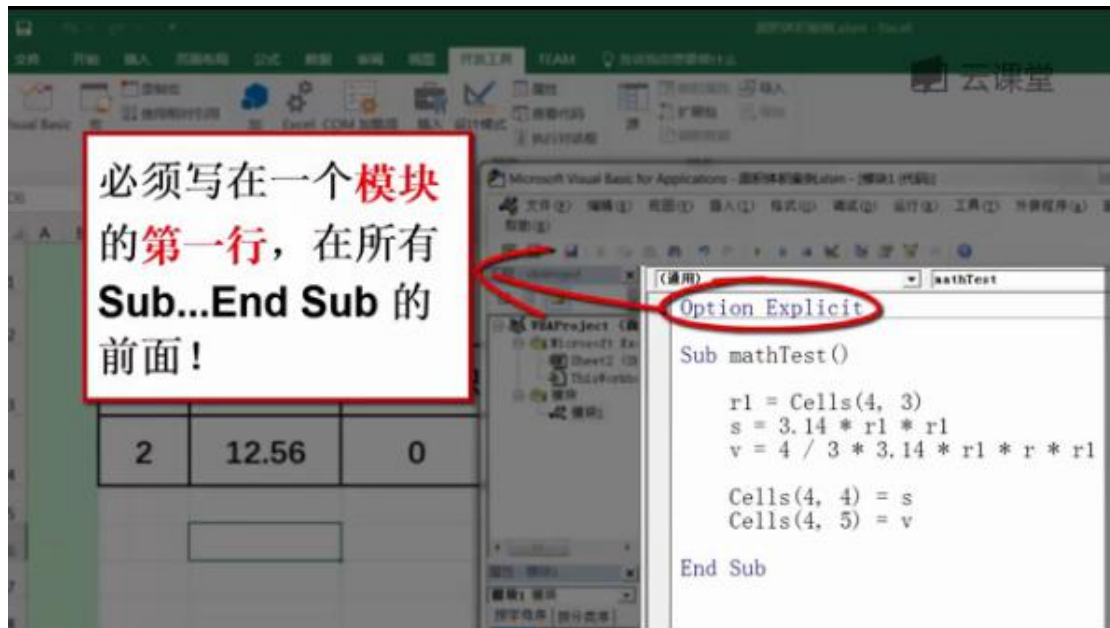




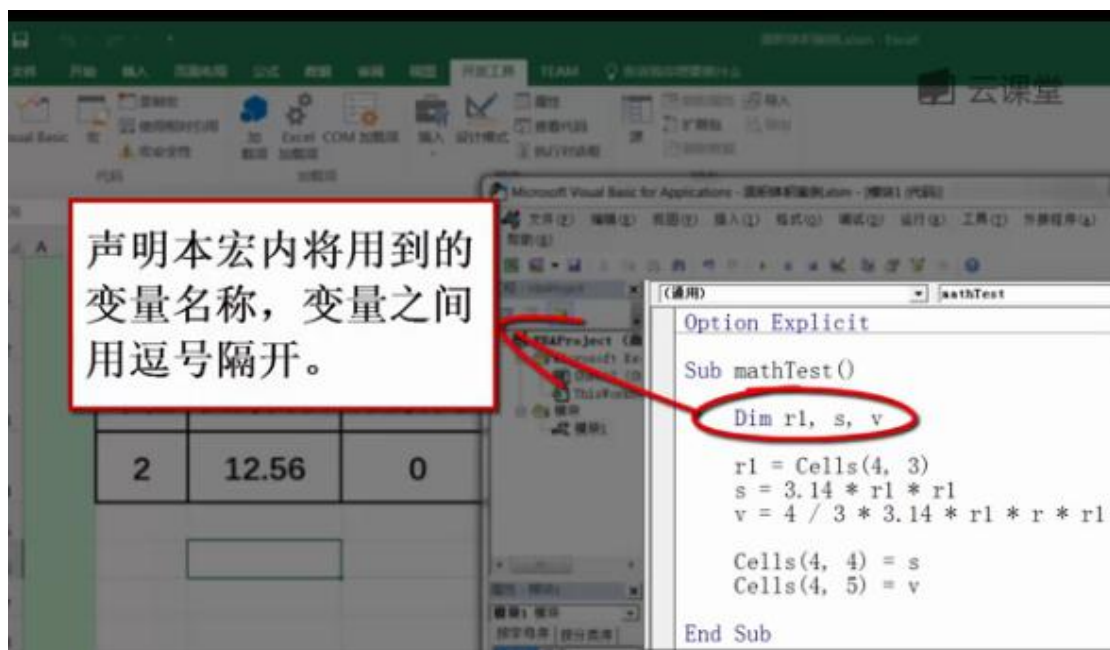
怎么样解决呢.?我们希望他既仁慈有严格的管理,引进一种**登记注册的制度**.什么是登记注册,很简单,就是由我们编程的人明确的告诉 VBA,



在这个程序里,我们只是用一下四个变量名,如果你遇到了其他没有注册过的变量名,请不要给他们分配空间,怎么操作呢?首先我在模块里的第一行写下  
`option Explicit`



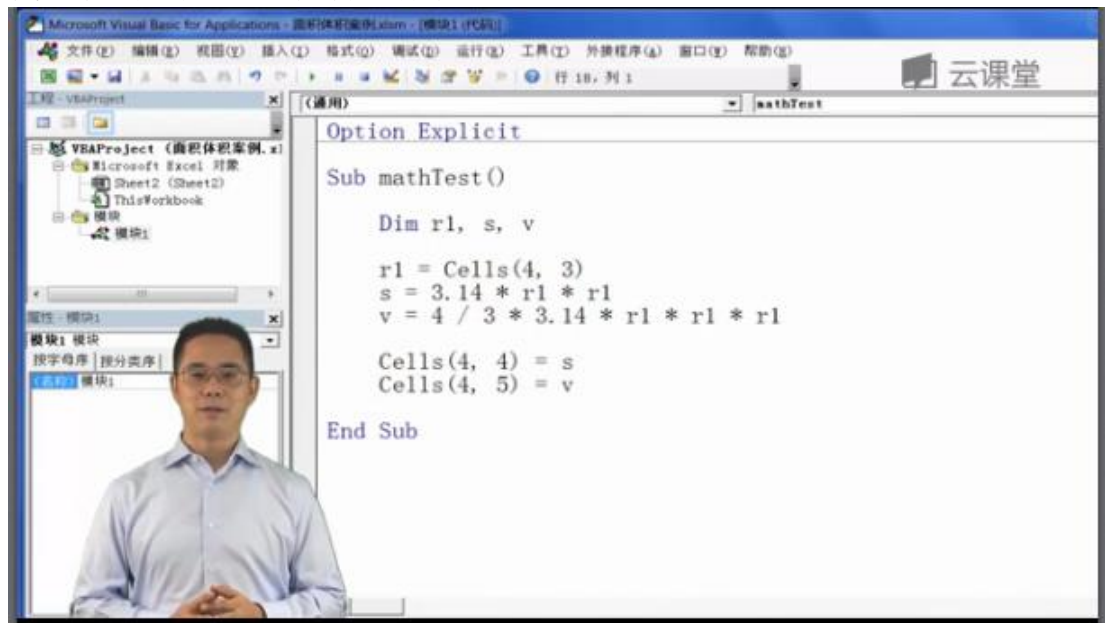
大家要记住我们一定要把 Option Explicit 写在模块的第一行。所有的 sub 和 end sub 前面，他的意思就是告诉 VBA,在这个模块里，请使用强制声明登记注册这种制度。接下来再 sub 里面我们声明三个变量 r,s,v



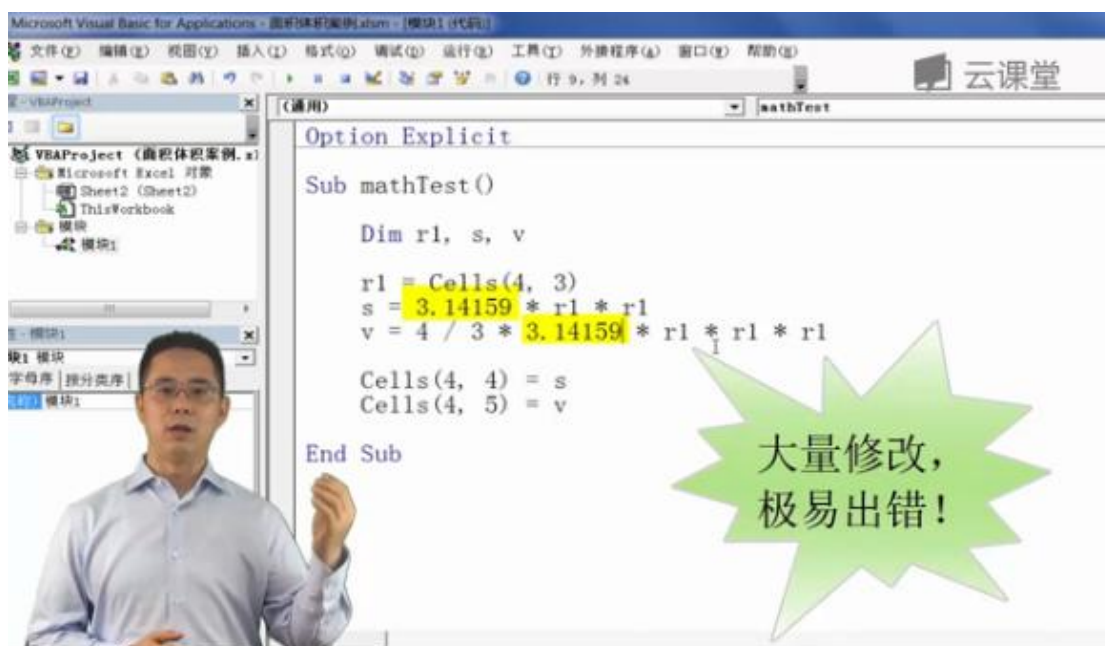
这就是说在这个 sub 里我们只是用以下几个变量名，凡是没有用 dim 声明的变量都是非法的。这次我们在运行一下，直接提示出错，我们就能排查出错误出在哪里。这个机制就叫强制声明。

在默认的情况下 VBA 是不允许强制声明的，很多同学也不愿意强制声明，感觉多写几行代码太麻烦了，不过可以说这样做，是后患无穷。比如他会导致我们的财务统计结果出错，竟然没有被发现，然后层层上报，最后影响了国家宏观经济统计结果，引发全球经济危机和第三次世界大战。总之强制声明非常重要。这也是为什么我们为什么把变量的概念放在后面来讲，因为在一开始就讲强制声明的概念，即使希望大家养成一个良好的习惯，避免这些错误的风险，我们回到这个

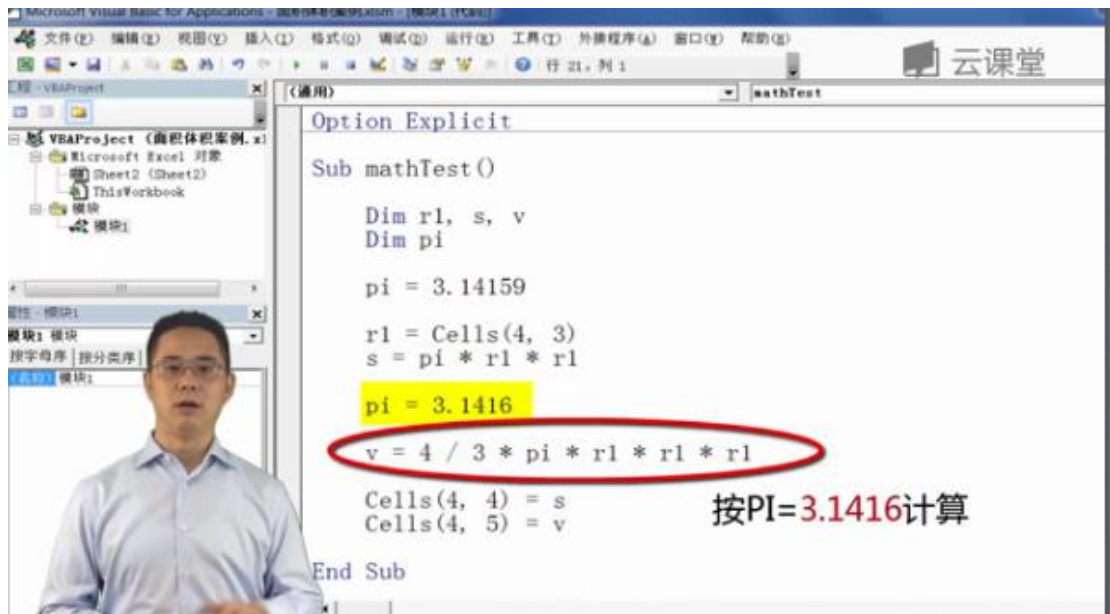
代码来，大家是不是觉得还有个小问题呢？



如果说我们的数值  $\pi$  想一个更精确的数值，3.141596，那么我们的程序该怎么改？假如我们需要学 100 个  $\pi$  的话，我们就需要 100 次修改，万一我们修改许仙了一点笔误，修改错了。我们的计算结果就会不匹配。



怎么解决了？我想大家都想到了，用一个变量来代表  $\pi$ 。好的，我们现在来定义一个变量：dim pi=3.1415926,然后把后面的 3.14 都修改为 pi,就可以了，确实一劳永逸，只不过还有一个隐患，大家发现没有，如果我们在中间某行代码写下了：pi=3.1416,

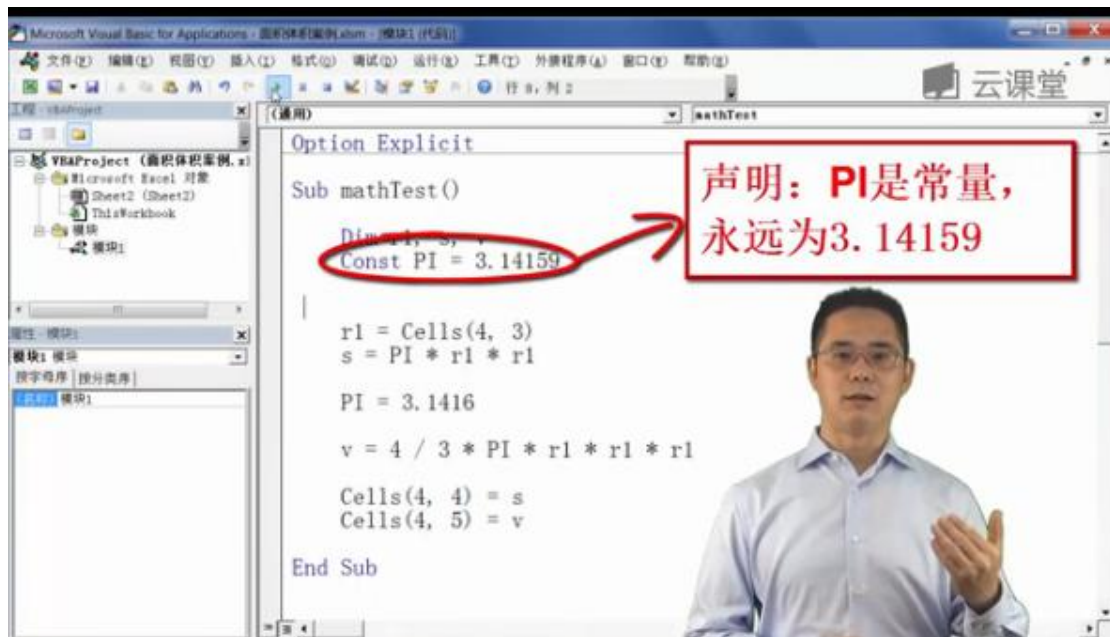


那么后面就按照 3.1416 来执行，前面仍然按照 3.1415926 执行，大家会说我们会犯这种低级错误吗？请大家一定不要低估这种风险，程序一长，就难免会发生这些不小心的修改，那么怎么解决这个问题呢？这里给大家介绍一个和变量相对的概念：**常量**。常量的含义是他的数值永恒不变，一旦确定就不可能被修改，那么他的关键字叫做 **const**

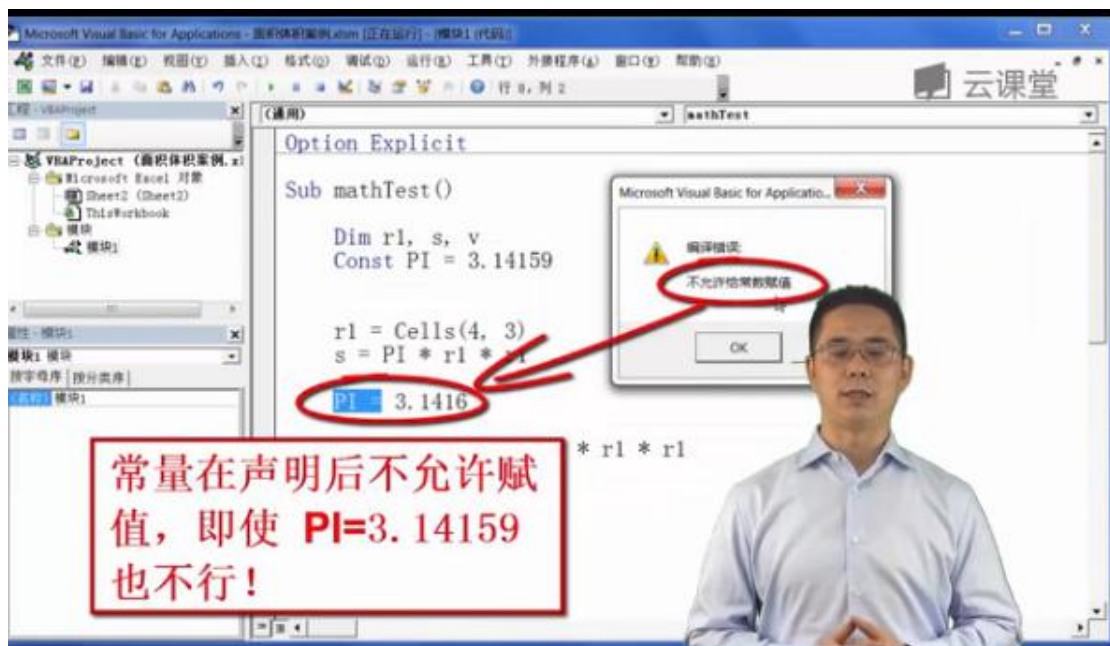


我们把 pi 定义为常量





运行一下，他会弹出一个错误，因为他是一个常量，不能再中间修改它。



好的，现在用了 const，我们就可以把经常用的数值定义为常量，比如圆周率，比如财务里面的税率，就不会出现计算不一致的情况，修改的时候，只需要修改一步，后面去全部都修改了。好了，今天的内容就是给大家介绍变量的本质，怎么样强制声明变量，以及常量的使用，如果大家能够不很好的遵循这些规则，我们的程序就会变得清晰易懂，减少错误的风险，那么今天回去的任务就是我们将前面的练习都改成强制声明变量的形式。感受一下。