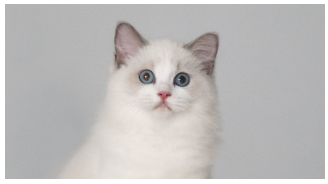
VBA Course 1 课件

吕梦珂 Mengke, Lyu



2020

b

Preface

本 VBA 课程分为六个课时，每个课时一个小时。第一个课时试听不收费，后面几个课时收费。购买三次课时以上的同学还可以得到课程期间无数次面对面课程答疑机会。

VBA 上手非常容易，写出好的代码却很难。原因在在于很多同学只懂得如何实现，却不会优化，加快代码运行效率。我会带领大家从新手写的代码入手，一步一步完成代码优化，讲一些非常好的代码习惯，带你写出优雅的代码。

这个课会以实践为主，一些很难遇到/很抽象的知识点就不讲了，尽量让没有任何编程基础的同学也能听懂，学会。

如果大家在学习 VBA，或者学习任何编程语言的时候遇到问题，我推荐在我的网站 actuarygarden.com 的数据科学板块提问，然后给我发消息进行回答。

课程中如果有任何建议和问题，非常欢迎随时戳我。

课程的目标是：在课程结束的时候，你能学会：

- 如何优雅地 Debug
- 如何操作 VBA 数组
- 如何写出效率高/可读的代码
- 如何做一个简单的 Excel Application

一个非常好的学习 VBA 的习惯是做代码笔记。一些常常用到的代码可以写一个笔记记下来，在写代码的时候直接复制粘贴。欢迎大家把代码笔记发布到论坛上。我会给你们点赞的:)

VBA 是一个很好玩的东西，我自己学习了一年半左右，已经完全爱上了它。希望你也如此。

话不多说，让我们开始我们的第一课叭。

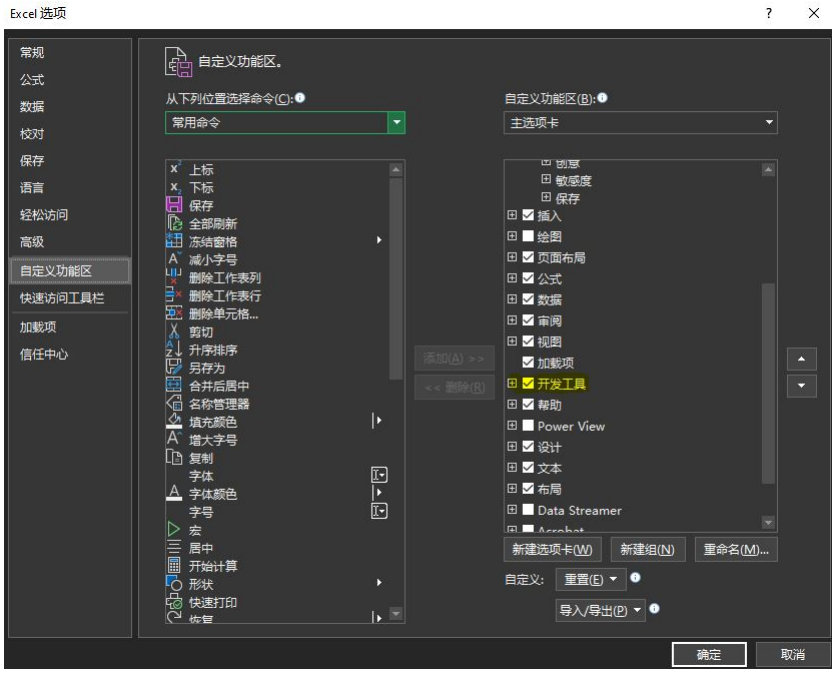
Contents

CHAPTER 1

VBA Introduction

1.1 如何打开 VBA Editor

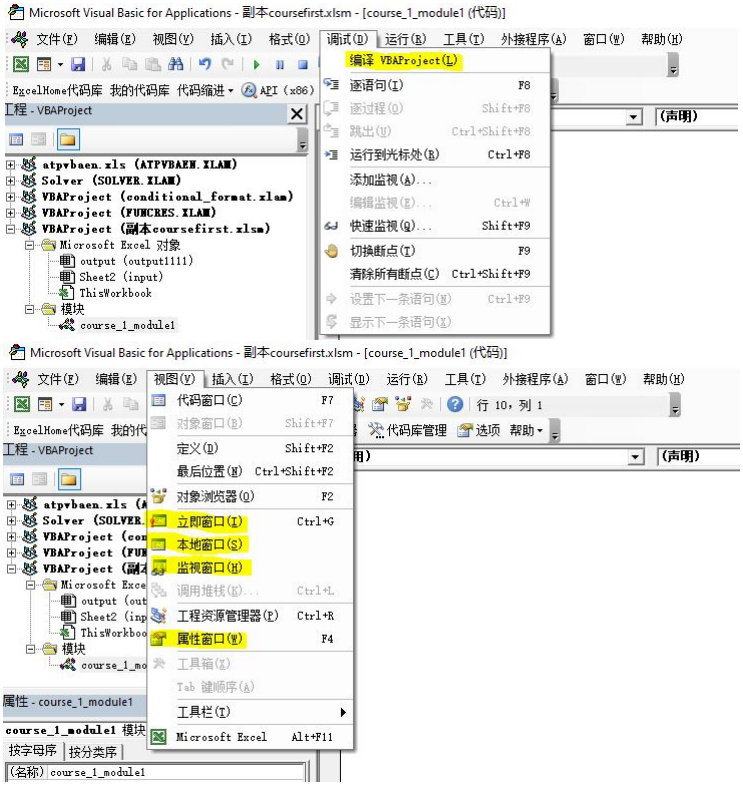
首先，打开自己 Excel 的选项，打开自定义功能区，看看右边主选项卡的开发工具是否勾选。如果没有勾选需要勾选一下。



勾选之后，可以看到在工具栏出现了开发者工具的选项，点击里面的 Visual Basic，就可以进入宏啦

另一种方法是用 $Alt + F11$ 的快捷键进入宏编辑器。

1.2 常用编译器工具



1.3 运行宏的几种方式

运行一个 Procedure 代码块 (F5)

- 设置快捷键

- 指定到形状或图片上
- 按“播放”按钮

逐步运行 (F8)

]

CHAPTER 2

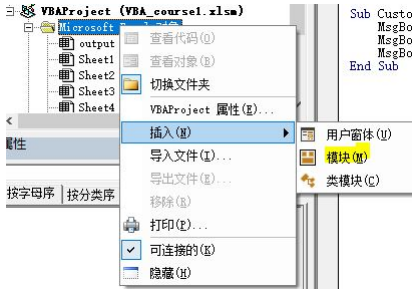
Let's Rock it!

这一节我们会从一个很小很小的例子入手，逐渐了解怎么写宏。这节课简单的内容能给你充实的信心继续学习。(我希望) :) 这是人生路上的一小步，但是是学习 VBA 过程中的一大步。

这个例子的任务目标是：假设我们有两个变量，a 和 b，我们想要求 a+b 的结果是什么。

2.1 最简单的写法

首先打开 VBA Editor，右键 Microsoft Excel 对象，选择插入模块



注意，细心的小朋友或者大朋友有可能会发现，Microsoft Excel 对象下面有所有工作表的名称，当你双击这些名称的时候，也会出现一个可以用来写代码的窗口。小朋友或者大朋友可能就会真诚发问了：为什么不能直接把代码写在这里而要重新创建一个模块呢。

原因是工作表里面的代码主要是用来写和工作表相关的事件 (Event) 的。(Event 在后面的课程里面会讲)。为什么一般不用来储存一般代码呢，参考 StackOverflow 里面的答案What's the difference between module and a module sheet? (如果你还不熟

悉 StackOverflow，那么可以抓住这个机会了解一下，任何 Bug 几乎都能在上面找到答案）

- 工作表里面的代码和工作表是相联系的，所以当你删掉工作表的时候，里面的代码也会被随之删除。
- 导出一张工作表的时候，它的代码也会被导出。
- 复制工作表的时候，工作表里面的代码也会被复制
- 在这个 Excel 外运行工作表里面的程序的时候较为麻烦。

总之，如果上面的内容太繁琐，只要记住一句话

> 目前的阶段，永远不要在工作表模块里面写代码

可以把模块名在属性窗口改一下，当然也可以不改。这里改名字主要是为了标识模块用途等，没有什么实际的用处。

双击打开新建的模块，刚刚的任务最简单的写法如下：

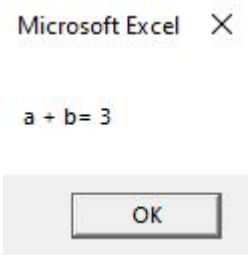
```
Sub Course_1()
    a = 40
    b = 31

    Msgbox "a + b= " & a + b
End Sub
```

.

用 *Sub* 和 *End Sub* 包裹的代码块是一个可以被执行的整体。

代码的原理是先把 40 和 31 赋值给了变量 a 和变量 b，然后通过 Msgbox 显示出结果。

Msgbox 是 VBA 最查用的方程之一，它的用处是显弹出一个如下所示的窗口



可别小看这个弹出窗口哦，它的 Documentation 在这里：Msgbox 的 Documentation

这里面有很多可以修改的参数,其中最常需要被修改的是 *prompt*, *title* 和 *buttons*

举个栗子：

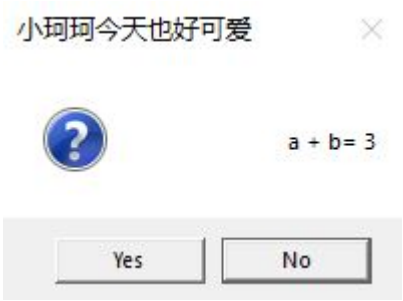
```
Sub Course_1()
    a = 40
    b = 31

    MsgBox "a + b= " & a + b, vbYesNo + vbQuestion + vbDefaultButton2 + vbMsgBoxRight,
          "小珂珂今天也好可爱"

End Sub
```

.

每个参数用逗号分隔，这里 *prompt* 参数的值为 "$a+b=$"&$a+b$，其中 & 是把两个字符串（或者字符串和值，其中值被转化为字符串）连起来的意思。*buttons* 参

数的值为 $vbYesNo + vbQuestion + vbDefaultButton2 + vbMsgBoxRight$。每个因子含义如下：

- $vbYesNo$: 弹出框的下方显示 yes 和 no 的选项
- $vbQuestion$: 显示问号标识
- $vbDefaultButton2$: 把第二个选项作为默认选项（即按 Enter 键会输入的选项）
- $vbMsgBoxRight$: 文本向右对齐



如果想要根据用户点击不同的按钮做出不同的反应，应该怎么做?

要先把 $Msgbox$ 得到的结果存入一个变量，假设这个变量是 ans，然后要写一个判断语句对它的值做判断。判断语句我们之后会仔细讲。

$vbYes$ 是点击 “Yes” 所代表的值，$vbNo$ 是点击”No” 代表的值。

这里注意两点。一，和别的编程语言不同，VBA 不用两个等于号做判断。二，这里 $Msgbox$ 里面的参数需要用引号引起来。

```
Sub Course_1()
    a = 40
    b = 31

    ans = MsgBox("a + b= " & a + b, vbYesNo + vbQuestion + vbDefaultButton2 +
        vbMsgBoxRight, "小珂珂今天也好可爱")

    If ans = vbYes Then
        Msgbox "Good"
    Else
        Msgbox "?小傻子"
    End if
End Sub
```

.

如果用户点击了 Yes，就会出现如下的界面

如果用户点击了 No，就会出现如下的界面

下面是一点进阶的知识。

如果我觉得按钮只有 Yes 和 No 的选项太枯燥了，有没有什么别的办法能够自己自定义按钮名称呢?

有!

参考如何自定义按钮?

第一步：把这段代码复制粘贴到一个新建的模块中。(不需要明白这段代码的意思，不过如果你愿意研究，欢迎和我讨论:))

```
' This module includes Private declarations for GetCurrentThreadId, SetWindowsHookEx,
    SetDlgItemText, CallNextHookEx, UnhookWindowsHookEx
' plus code for Public Sub MsgBoxCustom, Public Sub MsgBoxCustom_Set, Public Sub
    MsgBoxCustom_Reset
' plus code for Private Sub MsgBoxCustom_Init, Private Function MsgBoxCustom_Proc
' DEVELOPER: J. Woolley (for wellsr.com)
#If VBA7 Then
    Private Declare PtrSafe Function GetCurrentThreadId Lib "kernel32" _
        () As Long
    Private Declare PtrSafe Function SetWindowsHookEx Lib "user32" Alias "
        SetWindowsHookExA" _
        (ByVal idHook As Long, ByVal lpfn As LongPtr, ByVal hmod As LongPtr, ByVal
            dwThreadId As Long) As LongPtr
    Private Declare PtrSafe Function SetDlgItemText Lib "user32" Alias "
        SetDlgItemTextA" _
```

```
        (ByVal hDlg As LongPtr, ByVal nIDDlgItem As Long, ByVal lpString As String) As
            Long
    Private Declare PtrSafe Function CallNextHookEx Lib "user32" _
        (ByVal hHook As LongPtr, ByVal ncode As Long, ByVal wParam As LongPtr, lParam
            As Any) As LongPtr
    Private Declare PtrSafe Function UnhookWindowsHookEx Lib "user32" _
        (ByVal hHook As LongPtr) As Long
    Private hHook As LongPtr        ' handle to the Hook procedure (global variable)
#Else
    Private Declare Function GetCurrentThreadId Lib "kernel32" _
        () As Long
    Private Declare Function SetWindowsHookEx Lib "user32" Alias "SetWindowsHookExA" _
        (ByVal idHook As Long, ByVal lpfn As Long, ByVal hmod As Long, ByVal
            dwThreadId As Long) As Long
    Private Declare Function SetDlgItemText Lib "user32" Alias "SetDlgItemTextA" _
        (ByVal hDlg As Long, ByVal nIDDlgItem As Long, ByVal lpString As String) As
            Long
    Private Declare Function CallNextHookEx Lib "user32" _
        (ByVal hHook As Long, ByVal ncode As Long, ByVal wParam As Long, lParam As Any
            ) As Long
    Private Declare Function UnhookWindowsHookEx Lib "user32" _
        (ByVal hHook As Long) As Long
    Private hHook As Long           ' handle to the Hook procedure (global variable)
#End If
' Hook flags (Computer Based Training)
Private Const WH_CBT = 5            ' hook type
Private Const HCBT_ACTIVATE = 5     ' activate window
' MsgBox constants (these are enumerated by VBA)
' vbOK = 1, vbCancel = 2, vbAbort = 3, vbRetry = 4, vbIgnore = 5, vbYes = 6, vbNo = 7
    (these are button IDs)
' for 1 button, use vbOKOnly = 0 (OK button with ID vbOK returned)
' for 2 buttons, use vbOKCancel = 1 (vbOK, vbCancel) or vbYesNo = 4 (vbYes, vbNo) or
    vbRetryCancel = 5 (vbRetry, vbCancel)
' for 3 buttons, use vbAbortRetryIgnore = 2 (vbAbort, vbRetry, vbIgnore) or
    vbYesNoCancel = 3 (vbYes, vbNo, vbCancel)
' Module level global variables
Private sMsgBoxDefaultLabel(1 To 7) As String
Private sMsgBoxCustomLabel(1 To 7) As String
Private bMsgBoxCustomInit As Boolean

Private Sub MsgBoxCustom_Init()
' Initialize default button labels for Public Sub MsgBoxCustom
    Dim nID As Integer
    Dim vA As Variant               ' base 0 array populated by Array function (must
        be Variant)
    vA = VBA.Array(vbNullString, "OK", "Cancel", "Abort", "Retry", "Ignore", "Yes", "
        No")
    For nID = 1 To 7
        sMsgBoxDefaultLabel(nID) = vA(nID)
        sMsgBoxCustomLabel(nID) = sMsgBoxDefaultLabel(nID)
```

```
51      Next nID
52      bMsgBoxCustomInit = True
53  End Sub
54
55  Public Sub MsgBoxCustom_Set(ByVal nID As Integer, Optional ByVal vLabel As Variant)
56  ' Set button nID label to CStr(vLabel) for Public Sub MsgBoxCustom
57  ' vbOK = 1, vbCancel = 2, vbAbort = 3, vbRetry = 4, vbIgnore = 5, vbYes = 6, vbNo = 7
58  ' If nID is zero, all button labels will be set to default
59  ' If vLabel is missing, button nID label will be set to default
60  ' vLabel should not have more than 10 characters (approximately)
61      If nID = 0 Then Call MsgBoxCustom_Init
62      If nID < 1 Or nID > 7 Then Exit Sub
63      If Not bMsgBoxCustomInit Then Call MsgBoxCustom_Init
64      If IsMissing(vLabel) Then
65          sMsgBoxCustomLabel(nID) = sMsgBoxDefaultLabel(nID)
66      Else
67          sMsgBoxCustomLabel(nID) = CStr(vLabel)
68      End If
69  End Sub
70
71  Public Sub MsgBoxCustom_Reset(ByVal nID As Integer)
72  ' Reset button nID to default label for Public Sub MsgBoxCustom
73  ' vbOK = 1, vbCancel = 2, vbAbort = 3, vbRetry = 4, vbIgnore = 5, vbYes = 6, vbNo = 7
74  ' If nID is zero, all button labels will be set to default
75      Call MsgBoxCustom_Set(nID)
76  End Sub
77
78  #If VBA7 Then
79      Private Function MsgBoxCustom_Proc(ByVal lMsg As Long, ByVal wParam As LongPtr,
            ByVal lParam As LongPtr) As LongPtr
80  #Else
81      Private Function MsgBoxCustom_Proc(ByVal lMsg As Long, ByVal wParam As Long, ByVal
            lParam As Long) As Long
82  #End If
83  ' Hook callback function for Public Function MsgBoxCustom
84      Dim nID As Integer
85      If lMsg = HCBT_ACTIVATE And bMsgBoxCustomInit Then
86          For nID = 1 To 7
87              SetDlgItemText wParam, nID, sMsgBoxCustomLabel(nID)
88          Next nID
89      End If
90      MsgBoxCustom_Proc = CallNextHookEx(hHook, lMsg, wParam, lParam)
91  End Function
92
93  Public Sub MsgBoxCustom( _
94    ByRef vID As Variant, _
95    ByVal sPrompt As String, _
96    Optional ByVal vButtons As Variant = 0, _
97    Optional ByVal vTitle As Variant, _
98    Optional ByVal vHelpfile As Variant, _
```

```
    Optional ByVal vContext As Variant = 0)
' Display standard VBA MsgBox with custom button labels
' Return vID as result from MsgBox corresponding to clicked button (ByRef...Variant is
    compatible with any type)
' vbOK = 1, vbCancel = 2, vbAbort = 3, vbRetry = 4, vbIgnore = 5, vbYes = 6, vbNo = 7
' Arguments sPrompt, vButtons, vTitle, vHelpfile, and vContext match arguments of
    standard VBA MsgBox function
' This is Public Sub instead of Public Function so it will not be listed as a user-
    defined function (UDF)
    hHook = SetWindowsHookEx(WH_CBT, AddressOf MsgBoxCustom_Proc, 0,
        GetCurrentThreadId)
    If IsMissing(vHelpfile) And IsMissing(vTitle) Then
        vID = MsgBox(sPrompt, vButtons)
    ElseIf IsMissing(vHelpfile) Then
        vID = MsgBox(sPrompt, vButtons, vTitle)
    ElseIf IsMissing(vTitle) Then
        vID = MsgBox(sPrompt, vButtons, , vHelpfile, vContext)
    Else
        vID = MsgBox(sPrompt, vButtons, vTitle, vHelpfile, vContext)
    End If
    If hHook <> 0 Then UnhookWindowsHookEx hHook
End Sub
```

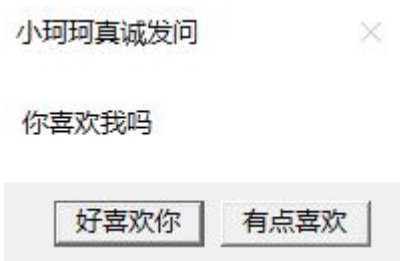
.

第二步：在想要调用有自定义按钮的窗口的地方按照如下语法调用

```

Sub Custom_MsgBox_Demo()
    MsgBoxCustom_Set vbYes, "好喜欢你"
    MsgBoxCustom_Set vbNo, "有点喜欢"
    'ans是被赋予结果的变量
    MsgBoxCustom ans, "你喜欢我吗", vbYesNo, "小珂珂真诚发问"
    Debug.Print (ans)
End Sub
```

.

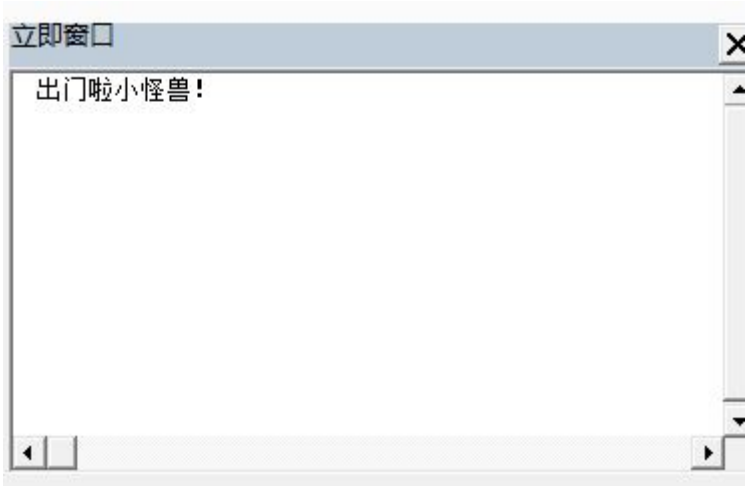
结果如下



2.2 不同输出结果的选择

2.2.1 Debug.Print

码如其名：是一个用于 Debug 的利器，需要结合即时窗口 (immediate window) 来使用。



2.2.2 输出到单元格

在输出到单元格前，需要解决的第一个问题是：如何引用单元格?

各位小朋友和大朋友开动脑筋想一想，怎么才能确定一个单元格的位置呢?

想一想哦

想一想哦

想一想哦

想好了吗？揭晓答案的时候到啦

- 第一种，先知道在哪张工作簿，再知道工作表，然后知道在哪行哪列
- 第二种，，先知道在哪张工作簿，再知道工作表，然后知道它的单元格引用坐标（类似”A1”）
- 第三种，是这个单元格被命名了，所以可以直接引用名字
- 第四种，通过相对别的单元格的位置找这个单元格

2.3 优化

2.3.1 定义变量

数据类型	变量类型	注释
整数	Integer	范围很小
整数	Long	绝对值很大的整数
小数	Double/Single	Double 精度更高
布尔型变量	Boolean	True / False
数组	变量名 (dimension1, dimension2)	
可变类型	Variant	不定义变量时候的默认类型

2.3.2 引用工作表的优化

如何避免因为错误修改工作表名字导致工作表不能使用?

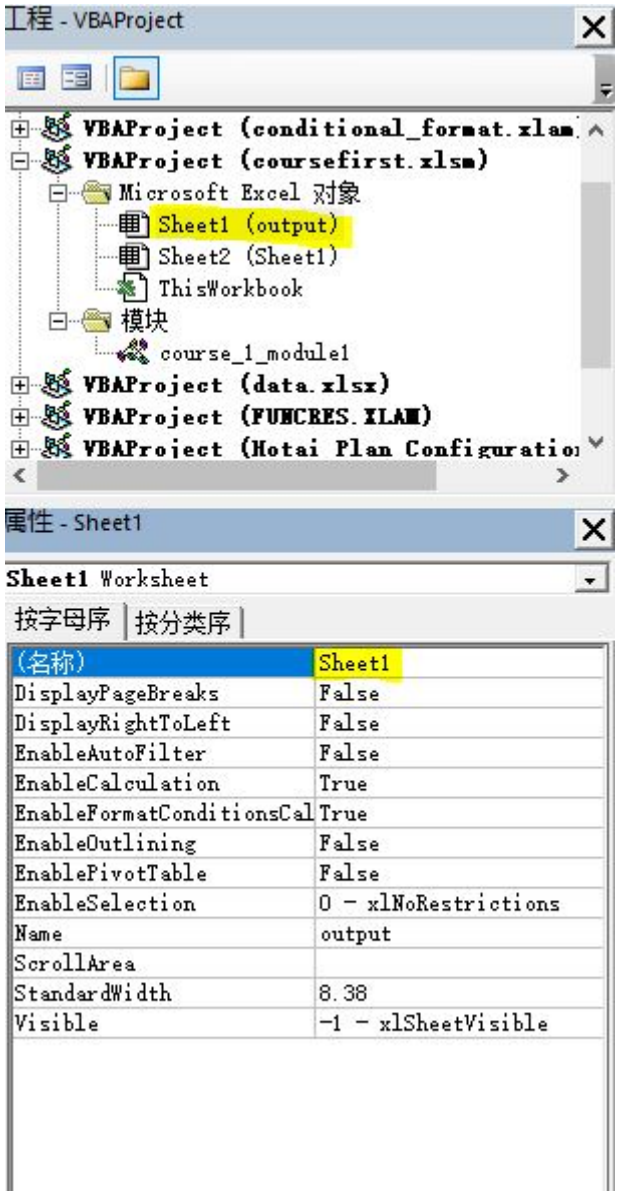
在我一开始学习写 VBA 的时候，是这样引用工作表的

```
1 Worksheets("工作表的名字").Activate
```

.

但是这个引用工作表的办法有一个问题，就是它依赖于工作表的名字。当我们在公司工作的时候，常常需要分发 Excel 给同事跑程序，这个时候毫不知情的同事可能会无意中修改工作表名，导致代码无法运行。

有两个解决这个问题的办法。第一个，也是我比较推荐的办法，是在 VBA Editor 窗口，点击对应工作表的名字，找到下方的属性窗口（找不到属性窗口？点击 F4 就会出现啦），在下图高亮的位置改变工作表的代码名称。



注意！表的代码名称是无法在 VBA 外面修改的，也就是使用者没有办法修改它。这种引用工作表的方式减小了运行错误的可能性，引用起来也更加方便，具体如下：

```
工作表的代码名.Activate
```

.

2.3.3 起别名

```
Dim output as Worksheet
Set output = Worksheets("output")
```

.

问题：用 Set 和直接赋值有什么区别?

答案：Set 只是取了一个别名。并不是赋值哦。具体来说的话，在上面的代码中，如果修改 output 的某个属性，Worksheets(”output”) 的属性也会被修改。赋值的话就没有这种效果啦。假如把 c 的值赋成 a 的值，也就是 c = a，那么改变 c 的值，a 的值并不会随之改变。

2.3.4 With 语句的运用

常常用于一个 Excel 对象在一段代码内被引用多次的情况

```
With Worksheets("output")
    .Range("A1") = "a + b"
End With
```

.

一个经典的例子是改变单元格的内部格式

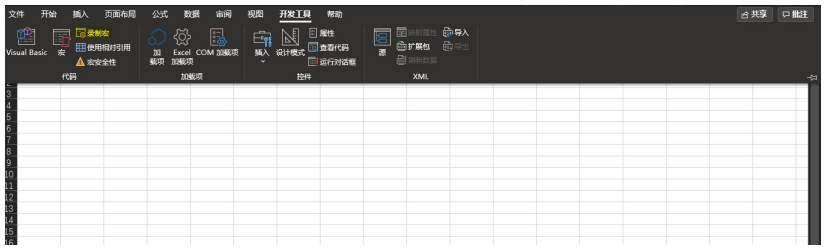
```
With Selection.Interior
        .Pattern = xlSolid
        .PatternColorIndex = xlAutomatic
        .ThemeColor = xlThemeColorAccent2
        .TintAndShade = 0
        .PatternTintAndShade = 0
End With
```

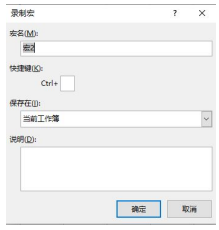
.

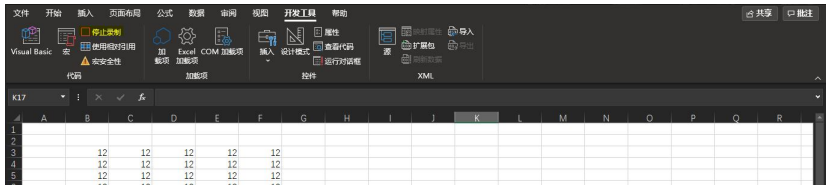
CHAPTER 3

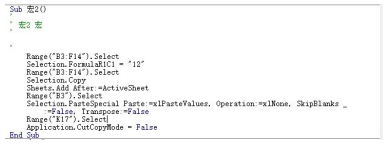
宏的录制

3.1 如何录制宏









```
Sub 宏2()
'
' 宏2 宏
'

'
    Range("B3:F14").Select
    Selection.FormulaR1C1 = "12"
    Range("B3:F14").Select
    Selection.Copy
    Sheets.Add After:=ActiveSheet
    Range("B3").Select
    Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
        :=False, Transpose:=False
    Range("K17").Select
    Application.CutCopyMode = False
End Sub
```

```
Sub 宏2()
'
' 宏2 宏

Application.Calculation = xlManual
Application.ScreenUpdating = False

    Range("B3:F14").Value = "12"
    Range("B3:F14").Copy
    Sheets.Add After:=ActiveSheet
    Range("B3").PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
        :=False, Transpose:=False


Application.ScreenUpdating = True
Application.Calculation = xlAutomatic
End Sub
```

3.2 录制宏究竟有什么用

可以快速了解一个功能要怎么写。

但是录制的宏一般不能直接用于生产。

CHAPTER 4

基本语法

4.1 Private 和 Public 宏以及 Function 的区别

在 VBA Editor 中可以运行四种代码块，

- Public Sub (简写为 Sub)
- Private Sub
- Public Function (简写为 Function)
- Private Function

```
Public Sub demo()

End Sub

Private Sub demo1()

End Sub

Public Function demo2()

End Function

Private Function demo3()

End Function
```

.

这里需要了解两个知识点：

首先是 Sub 和 Function 的区别：Sub 会不会返回值，Function 会返回值。只需要对 Function 的 < 名字 > 进行赋值，这个名字就会成为返回值。

Public 和 Private 的区别在于在哪里可以调用宏。Private 宏只能在该模块调用，不能在工作表和别的模块中调用。

4.2 条件

```
If Sheet1.Range("A1").Value > 5 Then
    Debug.Print "Value is greater than five."
ElseIf Sheet1.Range("A1").Value < 5 Then
    Debug.Print "value is less than five."
Else
    Debug.Print "value is equal to five."
End If
```

.

4.3 循环

```
Dim i As Integer, j As Integer
For i = 1 To 6
    For j = 1 To 2
        Cells(i, j).Value = 100
    Next j
Next i
```

.

```
Dim i As Integer, j As Integer
For i = 1 To 6
    For j = 1 To 2
        Cells(i, j).Value = 100
    Next j
Next i
```

.

```
Dim i As Integer, j As Integer
For i = 1 To 6
    For j = 1 To 2
        Cells(i, j).Value = 100
    Next j
Next i
```

.

For Each 循环一般用于不知道要循环的对象有多少子对象的时候

```
Dim x as Collection, y as Collection, tmpX as Variant, tmpY as Variant
'...
For Each tmpX In x
  For Each tmpY In y
    res = tmpX Mod tmpY
  Next tmpY
Next tmpX
```

.

```
Dim i As Integer
i = 1

Do While i < 6
    Cells(i, 1).Value = 20
    i = i + 1
Loop
```

.

> 为什么不建议用 Do while 循环?
>
> 因为 Do while 语句在调试过程中容易出现 While 后面跟的条件永远满足的情况，这样 Excel 就会卡死……
>
> 所以如果必须用 Do while，最好设定一个最大循环的次数。

```
Dim i As Integer
Dim max_rounds as Integer
i = 1
max_rounds = 100

Do While True
    i = i + 1
    If i > max_rounds Then
        Exit Do
    End If
    Cells(i, 1).Value = 20
    i = i + 1
Loop
```

.

退出代码块运行

```
Exit Sub
```

.

退出循环

```
Exit For / Do
```

.

CHAPTER 5

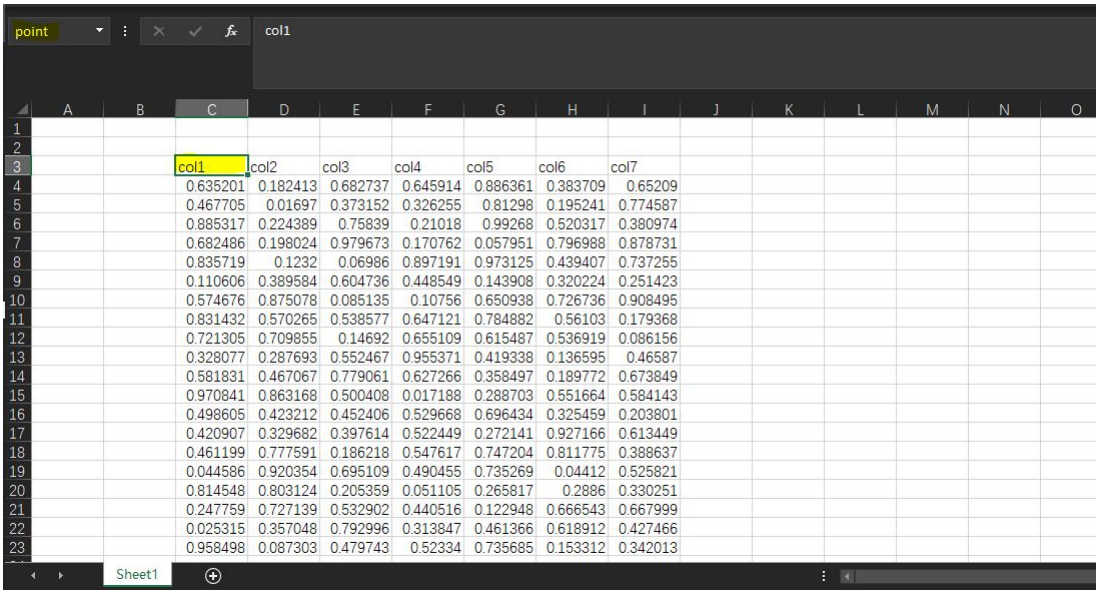
从 Range 对象讲起

5.1 最常用的引用方法

	col1	col2	col3	col4	col5	col6	col7
4	0.887524	0.07033	0.902378	0.868354	0.855584	0.315589	0.911367
5	0.082109	0.909203	0.243951	0.797649	0.42362	0.111108	0.887254
6	0.630729	0.361834	0.951398	0.938417	0.85106	0.380138	0.524428
7	0.968737	0.08578	0.281712	0.410195	0.79659	0.380126	0.378017
8	0.874044	0.365838	0.922706	0.143936	0.110351	0.008413	0.169267
9	0.351739	0.037688	0.566962	0.991888	0.235303	0.753689	0.685352
10	0.30589	0.326203	0.258166	0.12202	0.204322	0.845806	0.446969
11	0.5271	0.574098	0.514477	0.230877	0.900291	0.196282	0.482544
12	0.205693	0.280281	0.652612	0.131302	0.923167	0.320484	0.807284
13	0.596937	0.484951	0.698762	0.980684	0.307527	0.265102	0.112576
14	0.416815	0.894979	0.358392	0.333385	0.093555	0.971573	0.450691
15	0.020867	0.425052	0.812083	0.275322	0.749607	0.75553	0.455454
16	0.227694	0.193452	0.354242	0.916805	0.368635	0.839781	0.890552
17	0.759698	0.195219	0.066718	0.152979	0.205461	0.342679	0.851003
18	0.219254	0.400581	0.890491	0.557091	0.798668	0.950075	0.165232
19	0.910653	0.003926	0.182848	0.411211	0.287021	0.885255	0.748854
20	0.784518	0.560917	0.631241	0.880705	0.994426	0.980496	0.55593
21	0.237528	0.596013	0.00666	0.174107	0.864931	0.377958	0.038908
22	0.410414	0.459778	0.607738	0.104254	0.259771	0.836359	0.152718
23	0.922233	0.33386	0.853762	0.441709	0.040666	0.875909	0.02101

假设我需要引用非空区域，我应该怎么做呢？

我最常用的方法是找到这个区域左上的第一个单元格，给它取一个名字



	col1	col2	col3	col4	col5	col6	col7
4	0.635201	0.182413	0.682737	0.645914	0.886361	0.383709	0.65209
5	0.467705	0.01697	0.373152	0.326255	0.81298	0.195241	0.774587
6	0.885317	0.224389	0.75839	0.21018	0.99268	0.520317	0.380974
7	0.682486	0.198024	0.979673	0.170762	0.057951	0.796988	0.878731
8	0.835719	0.1232	0.06986	0.897191	0.973125	0.439407	0.737255
9	0.110606	0.389584	0.604736	0.448549	0.143908	0.320224	0.251423
10	0.574676	0.875078	0.085135	0.10756	0.650938	0.726736	0.908495
11	0.831432	0.570265	0.538577	0.647121	0.784882	0.56103	0.179368
12	0.721305	0.709855	0.14692	0.655109	0.615487	0.536919	0.086156
13	0.328077	0.287693	0.552467	0.955371	0.419338	0.136595	0.46587
14	0.581831	0.467067	0.779061	0.627266	0.358497	0.189772	0.673849
15	0.970841	0.863168	0.500408	0.017188	0.288703	0.551664	0.584143
16	0.498605	0.423212	0.452406	0.529668	0.696434	0.325459	0.203801
17	0.420907	0.329682	0.397614	0.522449	0.272141	0.927166	0.613449
18	0.461199	0.777591	0.186218	0.547617	0.747204	0.811775	0.388637
19	0.044586	0.920354	0.695109	0.490455	0.735269	0.04412	0.525821
20	0.814548	0.803124	0.205359	0.051105	0.265817	0.2886	0.330251
21	0.247759	0.727139	0.532902	0.440516	0.122948	0.666543	0.667999
22	0.025315	0.357048	0.792996	0.313847	0.461366	0.618912	0.427466
23	0.958498	0.087303	0.479743	0.52334	0.735685	0.153312	0.342013

取名字的方法是选中要起名字的单元格，在它的左上角的区域（如图所示，标黄的地方）写入你想给它取的名字，就叫它“point”吧。

然后打开 VBA Editor，写入代码：

```
Dim rg as range
'记得我们讲过Set代表的是取别名
‘CurrentRegion选中的是用点击以"point"命名的单元格以后，按住Ctrl + A选择的区域
Set rg = Range("point").CurrentRegion
rg.Select
```

.

如果我们想选的不是全部的区域，而是已知长宽的区域，那么这样引用这个区域

```
Dim rg as range
'选中从"point"这个单元格开始三行四列的区域
Set rg = Range("point").Resize(3,4)
rg.Select
```

.

CurrentRegion 和 *Resize* 是最常用最好用的两个方法，一定要学会！

5.2 引用 Range 的办法

参考资料 https://powerspreadsheets.com/excel-vba-range-object/

单元格正规的应用方式是先告诉 VBA 这个单元格在哪个工作簿，在哪个工作表，然后再告诉 VBA 这个范围的坐标。

例如

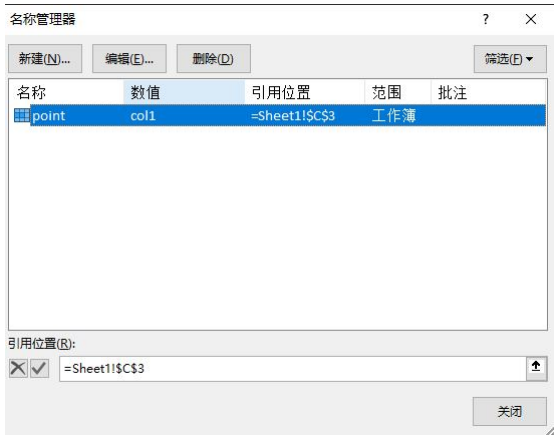
```
1    ThisWorkbook.Worksheets("Sheet1").Range("A1").Select
```

.

如果不写哪个工作簿，默认工作簿是 VBA 所在的工作簿。如果不写哪个工作表，默认工作表为运行代码时打开的工作表。

一般情况下，如果只对一个 Excel 进行操作，只需写好工作表名字即可，还记得我们之前讲过用工作表的代码名称吗?

在上面的例子里面我只用了 $Range(\text{"}point\text{"})$ 是因为对单元格区域起名是作用于工作表的，简单地讲就是一个工作簿只能起一个这样的名字。



注意这个作用范围是“工作簿”。如果有两个或三个单元格都起了同样的名字就要注意了，一般只有一个是作用于工作簿的，剩下的是作用于单个工作表的。

对于作用于工作表的单元格，我们还是需要告诉 VBA 它在的工作表是哪个。

5.3 Range 和 Cell

https://powerspreadsheets.com/excel-vba-range-object/有非常详细的各种区域引用方法介绍。

引用单元格一般用的两个函数一个是 *Range* 一个是 *Cell*

Range() 里面究竟可以填什么?

- Range 地址例如 Range(”A1:H8”) Range(”A1”) Range(”1:1”) 在上面的例子中，如果使用这种方法，可能会造成用户插入或删除无关单元格后程序无法使用的情况。
- 被命名的范围（如何命名？公式-> 名称管理器）
- 两个 Range（或者 Cell）对象。不推荐使用，代码过长

如果想要引用一个范围里面某一个单元格，应该怎么做呢？假如我想引用的是范围里面的第二行第三列的单元格，可以使用 *range.cell*

```
1    Range("A1:C23").Cells(2,3).Select
```

.

选中的就是 C2 这一单元格。

如果想要引用一个范围里面某一个整行或者整列，应该怎么做呢?

假如选择第二行

```
Range("A1:C23").Rows(2).Select
```

.

假如选择第二列

```
Range("A1:C23").Columns(2).Select
```

.

5.4 总结

单元格，该怎么做? 用如何引用想要的范围

- *CurrentRegion* 等于按住 Ctrl + A 选中的范围

```
Range("A1").CurrentRegion.Select
```

.

- *End* 等于按住 Ctrl + 上下左右键选中的范围，End 里面可以填写 *xlup xldown xltoleft xltoright*

```
Range("A1").End(xltoright).Select
```

.

- *Resize* 改变 Range 的大小

```
Range("A1").Resize(5,5).Select
```

.

- *UsedRange* 一个工作表里面用了的 Range 范围
- *Union* Range 的并集

```
Union(Range("B2:C7"), Range("C6:F8")).Select
'它和下面这种引用方式是相等的
Range("B2:C7, C6:F8").Select
```

.

- *Intersect* Range 的交集

```
Intersect(Range("B2:C7"), Range("C6:F8")).Select
'它和下面这种引用方式是相等的
Range("B2:C7 C6:F8").Select
```

.

5.5 如何循环过一个 Range

```
For each cell in rg
    Debug.Print cell.value
Next cell
```

.

5.6 Range 和数组的交互

操作数组比操作范围快至少 10 倍。因此基本操作是将 Range 读入数组，对数组进行操作，把结果写到 Range 里面。

5.6.1 如何从 Excel Range 读出数组

```
Dim arr as Variant
Arr = range( “A1” ).value
```

.

5.6.2 如何把数组写入 Excel Range

```
Dim arr(100, 1)
range( “A1” ).resize(UBond(arr, 1), UBond(arr, 2)).Value = arr
```

.

5.7 Case Study

如何把选中区域根据行变成下三角

```
Sub triangle()
Dim rg As Range
Dim countrows As Integer

’ Selection是选中的区域
countrows = Selection.Rows.Count

Dim i As Integer

For i = 1 To countrows
    If i = 1 Then
        Set rg = Selection.Cells(1, 1)
    Else
        Set rg = Union(rg, Selection.Cells(i, 1).Resize(1, i))
    End If
Next i

rg.Select
End Sub
```

.

CHAPTER 6

精算案例讲解

如何用 VBA 模拟马尔科夫链状态变化

1. 思路

了解 Input 是什么，想要的 Output 是什么，是什么格式。

一个通用思路：先把 Input 读入数组，用 VBA 进行一系列数组操作-> 数组格式 Output -> 写入单元格

什么样的 VBA Project 是好的 Project

- 运行迅速/有效率 -> 尽量使用数组操作
- 可读性强 -> 写好 Comments，尽量不要一个 Module 写完，记住封装功能 (Function /Sub)
- 更新起来很方便 -> 尽量少用 VBA 写死的东西，留下在 Excel 中更新的空间
- Robustness -> 有效引用范围可以降低出错的概率/多写一些 Validation/可以锁住一些不希望用户修改的范围