

## 第3章 单元格的基本操作

在本章中，将详细讲解如何操作 Excel 中的单元格对象。在 Excel 中，单元格对象是用户经常需要处理的对象。同时单元格对象也是用户操作 Excel 的最小单元。在本章中，将对涉及到单元格的常见操作进行介绍。

### 3.1 获取单元格的引用

在 Excel VBA 中，经常要使用到对单元格的引用，然后使用属性和方法对区域进行操作。根据单元格区域的使用属性不同，主要包括以下几种情况：

- 单个单元格；
- 多个连续或者不连续单元格组成的区域；
- 整行或整列。

用户如果希望使用 VBA 代码对单元格区域进行操作，就必须用将单元格区域赋值给某个变量，也就是首先获得某个单元格区域的引用。

#### 案例 17 使用 A1 样式引用单元格

##### 1. 功能说明

在使用 Excel 中，A1 样式是用户最熟悉的一种引用样式。A1 样式其实就是列名和行名的组合，确定对应的单元格。在 Excel VBA 中，当用户需要对单元格进行操作的时候，需要首先使用代码引用单元格。

##### 2. 语法说明

在 VBA 中，通过 Range 对象的 Range 属性返回 Range 对象。Range 属性返回一个单元格或单元格区域，对区域的引用如果使用 A1 样式，需将引用字符串包含在引号中。另外还可以使用以下方式引用单元格：

- [A3]：引用单元格“A3”；
- ActiveCell：当前单元格。

在本例中，因为引用的单元格为对象，因此需要创建对象变量。创建对象变量通常分两个步骤：

(1) 声明对象变量。与声明普通变量类似，用户可以使用 Dim 语句或其他声明语句之一来声明对象变量。引用对象的变量必须是 Variant、Object，或是一个对象的指定类型。下面的声明在 Excel 中都是有效的：

```
Dim MyRange           '声明 Variant 数据类型
Dim MyRange As Object '声明 Object 数据类型
Dim MyRange As Range  '声明 Range 类型
```

(2) 赋值对象变量给对象。在 Excel VBA 中，需要使用 Set 语句赋值对象给对象变量，可以赋值对象表达式或是 Nothing。下面的赋值语句在 Excel VBA 中是有效的：

```
Set MyRange = Range("B3") ' 赋值对象引用。
Set MyRange = Nothing     ' 中断关联。
```

设置对象变量为 Nothing，会中断此对象变量与其他对象的关联，可预防因意外改变变量而更改对象。在关闭关联对象后，对象变量总是设置为 Nothing，所以可以检测对象变量是否指到有效的对象。

### 3. 案例说明

本例中，用户需要首先引用单元格 D3，然后在对应的单元格中添加字符串“I love Excel VBA”。

### 4. 编写代码

实现 A1 样式引用的代码如下：

```
Sub ForCell()
    Dim RngCell As Range
    Set RngCell = Range("D3")

    With RngCell
        .Value = "I love Excel VBA"
        .Font.Name = "Tahoma"
        .Font.Italic = True
    End With

    Set RngCell = Nothing
End Sub
```

### 5. 运行结果

运行程序代码，得到的结果如图 3.1 所示。

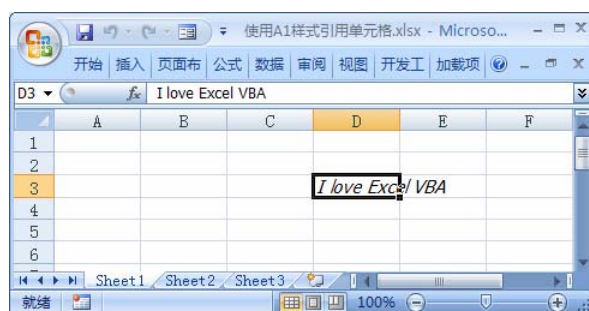


图 3.1 A1 样式引用结果

## 6. 程序分析

在上面的代码中，通过下面的代码段：

```
Dim RngCell As Range
Set RngCell = Range("D3")
```

定义了 Range 变量，然后通过 Range("D3")获取对单元格 D3 的引用。

## 案例 18 使用 R1C1 样式引用单元格

### 1. 功能说明

在 Excel 中，R1C1 的引用格式在开发过程中也是经常用到的。在很多情况下，使用单元格的行列序号来引用单元格，会给程序开发带来更大的便利。例如，当用户需要在单元格区域内进行循环设置的时候，使用 R1C1 样式就便利很多。因为，对于循环而言，数字会更加便利，而 A1 样式中的列名字母则不适合循环。

### 2. 语法说明

Excel 的工作表由行和列构成。通过使用行列索引号，可用 Cells 属性引用单个单元格。该属性返回代表单个单元格的 Range 对象。Cells(3,2)返回对工作表中单元格 B3 的引用。同时，R1C1 样式可以使用多种引用方式：绝对引用、相对引用和混合引用，得到的结果都是一样的。因此，用户在实际的编程中，可以根据情况选择合适的方法。

使用 Cells 属性引用单元格时，用户可以使用变量替代行列索引号，所以 Cells 属性适合在单元格区域中循环。另外，使用 Cells 属性还可按以下方式引用单元格区域：

**Cells(2, "B")：表示第 2 行 B 列**

如果使用 Cells 属性时，不指定行列索引号，程序将返回工作表上所有单元格的 Range 对象。

### 3. 案例说明

运行本例的效果如图 4-2 所示，单击“使用索引号引用单元格”按钮，将在单元格区域“A1:E10”中按顺序填充数字。

### 4. 编写代码

R1C1 样式的代码如下：

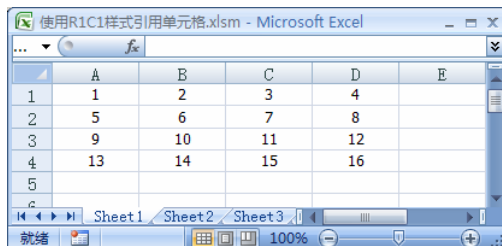
```
Sub R1C1_cells()
    Dim i As Integer
    Dim j As Integer

    For i = 1 To 4
        For j = 1 To 4
            Cells(i, j).Value = (i - 1) * 4 + j
        Next
    Next
End Sub
```

End Sub

## 5. 运行结果

运行程序代码，得到的结果如图 3.2 所示。



	A	B	C	D	E
1	1	2	3	4	
2	5	6	7	8	
3	9	10	11	12	
4	13	14	15	16	
5					

图 3.2 R1C1 样式引用单元格

## 6. 程序分析

从上面的例子中可以看出，当用户使用 R1C1 的样式引用单元格时，可以很方便的使用循环结构，进行各种复杂的运算。

# 案例 19 引用多个单元格区域

## 1. 功能说明

在使用 Excel 分析数据或者处理问题时，有时需要同时选择多个单元格，然后再进行处理。同时，这些单元格区域在范围上可能并不连续，因此需要使用特殊方法引用单元格区域。当单元格区域范围比较多，同时比较杂乱，没有规律的时候，单个单元格区域依次选择，会很耽误效率。因此，需要使用特定的语法对多个单元格区域进行引用。

## 2. 语法说明

在 Excel VBA 中，可以使用两种方法选择单元格区域。如果是连续的单元格区域，可以使用左上角和右下角的单元格来确定区域。例如，使用下面的代码：

```
Range("B2:D3")
```

表示引用的是单元格 B2:D3 的区域。如果是不连续的单元格区域，可以选择使用 Range 属性。其中，使用 Range 属性的方法是，用逗号将不同的单元格区域隔开，例如：

```
Range("B2:D3, E4:F8")
```

在上面的代码中，逗号必须在引号的内部。

## 3. 案例说明

在本例中，用户需要通过程序代码同时选中单元格中多个不连续的单元格区域，然后设置这些单元格区域的填充红色。

## 4. 编写代码

本例代码如下：

```
Sub select_range()  
    Dim rng As Range  
    Set rng = Range("A1:B4, D1:G6, B8:C9")  
  
    rng.Select  
    Selection.Interior.ColorIndex = 3  
  
    Set rng = Nothing  
End Sub
```

## 5. 运行结果

运行程序代码，得到的结果如图 3.3 所示。

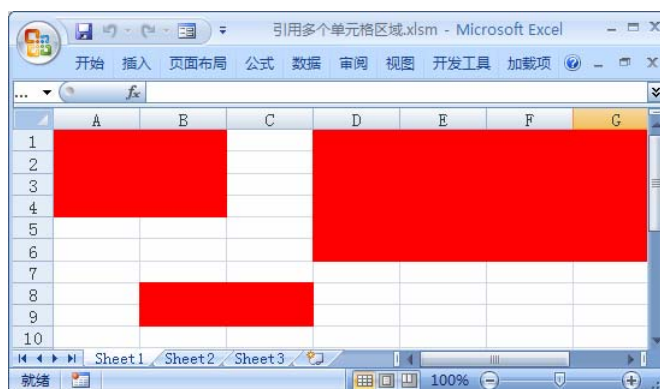


图 3.3 选择多个单元格区域

## 6. 程序分析

当用户使用代码 `Range("A1:B3","D4:E5")` 时，程序选择的单元格区域是 A1:E5，即以第一个区域左上角单元格为起点，第二个区域右下角单元格为终点，连接成一个新的连续区域。

# 案例 20 合并单元格区域

## 1. 功能说明

前面的案例说明了如何使用 `Range` 属性来引用多个单元格区域。在 Excel VBA 中，用户还可以使用 `Union` 方法来合并不同的单元格区域。在本小节中，将详细讲解如何使用 `Union` 方法来合并单元格的区域。

## 2. 语法说明

使用 `Application` 对象的 `Union` 方法，可将多个单元格区域组合到 `Range` 对象中。

Application.Union 方法的语法格式如下：

表达式.Union(Arg1, Arg2, Arg3, Arg4, Arg5, Arg6, Arg7, Arg8, Arg9, Arg10, Arg11, Arg12, Arg13, Arg14, Arg15, Arg16, Arg17, Arg18, Arg19, Arg20, Arg21, Arg22, Arg23, Arg24, Arg25, Arg26, Arg27, Arg28, Arg29, Arg30)

用户在使用 Union 方法时，至少需要两个单元格区域，最多可合并 30 个单元格区域。

例如：

```
Union(Range("A1:C3"), Range("E1:F5"))
```

### 3. 案例说明

在本例中，用户需要引用多个单元格区域，然后在引用区域的单元格中添加随机数。

### 4. 编写代码

合并单元格区域的具体代码如下：

```
Sub union_range()
    Dim rng1 As Range
    Dim rng2 As Range
    Dim rng3 As Range

    Set rng1 = Range("A1:B3")
    Set rng2 = Range("D4:F6")
    Set rng3 = Union(rng1, rng2)

    rng3.Formula = "=int(100*RAND()+20)"

End Sub
```

### 5. 运行结果

运行程序代码，得到的结果如图 3.4 所示。

	A	B	C	D	E	F
1	39	33				
2	20	119				
3	74	113				
4				33	49	87
5				84	94	68
6				36	65	34
7						

图 3.4 合并单元格区域的结果

### 6. 程序分析

在本例的代码中，首先使用两个对象变量获取单元格区域，然后使用 Union 方法将两个区域合并为一个区域。

## 案例 21 引用合并区域的子区域

### 1. 功能说明

当用户同时选中多个单元格区域时，某些操作不能在选定区域内同时执行，必须在选定区域内的单个子区域上循环，对每个单独的子区域分别执行该操作。因此，这个时候有必要引用合并区域的子区域。

### 2. 语法说明

在 Excel VBA 中，用户选定多个区域后，将生成 Areas 集合。Areas 集合的每个成员是 Range 对象。选定区域内每个离散连续单元格区域都有 Range 对象。如果选定区域内只有一个子区域，则 Areas 集合包含一个与该选定区域对应的 Range 对象。

使用 Areas(index) 可从集合中返回单个 Range 对象。该索引号对应选定这些区域的顺序。下例中，如果当前选定区域包含多个子区域，就清除该选定区域中的第一个子区域。

### 3. 案例说明

选择工作表中的多个单元格式区域，将其单元格区域进行合并，然后依次遍历合并后的单元格区域的各个子区域。

### 4. 编写代码

引用合并区域的子区域的具体代码如下：

```
Sub Get_Part_rubrange()  
    Dim rng1 As Range  
    Dim rng2 As Range  
  
    Dim i As Integer  
  
    Dim str As String  
  
    Set rng1 = Range("A1:C2, B3:D4, E4:F6")  
    rng1.Select  
  
    For i = 1 To rng1.Areas.Count  
        Set rng2 = rng1.Areas(i)  
        str = "子区域" & i & "的地址是：" & vbCrLf & vbCrLf  
        str = str & rng2.Address  
        MsgBox str  
    Next  
  
    Set rng1 = Nothing  
    Set rng2 = Nothing  
End Sub
```

## 5. 运行结果

运行程序代码，得到的结果如图 3.5 所示。

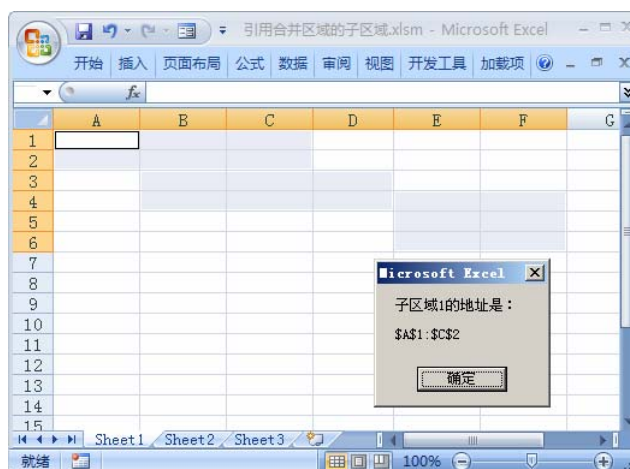


图 3.5 遍历第一个子区域的结果

单击对话框中的“确定”按钮，然后依次运行程序代码，得到的最后一个子区域的结果如图 3.6 所示。

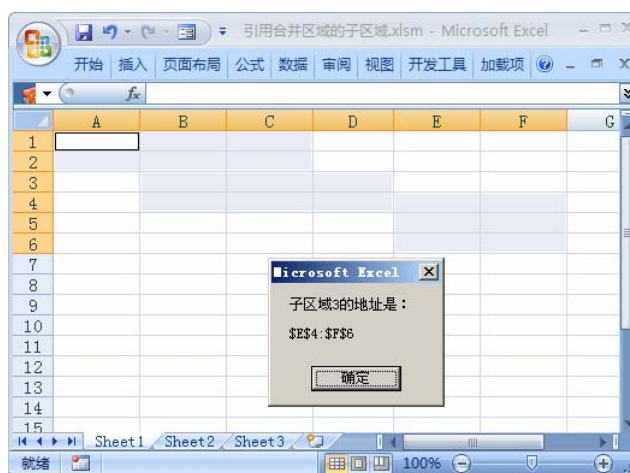


图 3.6 遍历最后一个子区域的结果

## 6. 程序分析

在本例的代码中，Areas 集合包括三个子区域，通过循环分别显示每个子区域的地址，然后显示在对话框中。



## 案例 22 当前单元格的前一单元格

### 1. 功能说明

Excel 表格是一个二维结构，所有的单元格组成了一个工作表。这些单元格在物理位置上是相互关联的：前、后、左和右等。这些都是相邻单元格的区域关系，在 Excel 中，用户通过 VBA 代码还可以处理非相邻的单元格物理关系。在本小节中，将首先讲解如何引用当前单元格的前一单元格。

### 2. 语法说明

使用 Range 对象的 Previous 属性，可获取对指定单元格的前一个单元格的引用。其语法格式如下：

#### 表达式.Previous

如果表达式为一个对象为区域，则 Previous 属性会模拟 Shift+Tab 键，但此属性只是返回上一单元格，并不选定它。

### 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区。本例中，用户在选中某单元格后，需要查看该单元格的前面一个单元格数据。原始数据如图 3.7 所示。



	A	B	C	D	E	F
1	编号	地区	上半年销量	下半年销量		
2	114	北部	1600	1300		
3	113	北部	1810	1330		
4	116	东部	1110	1300		
5	118	东部	1124	1375		
6	117	东部	3420	1390		
7	111	南部	1141	1375		
8	110	南部	1180	1300		
9	112	南部	1460	1300		
10	115	西部	1530	1270		
11	总计		14375	11940		
12						
13						

图 3.7 原始数据

### 4. 编写代码

获取当前单元格前一单元格的引用的代码如下：

```
Sub pre_cells()  
  
    Dim rng As Range  
  
    Set rng = ActiveCell.Previous
```

```
MsgBox "当前单元格的前一单元格的值为：" & rng1.Value
```

```
Set rng = Nothing
```

```
End Sub
```

## 5. 运行结果

选择单元格 D6，运行程序代码，得到的结果如图 3.8 所示。



图 3.8 运行结果

## 6. 程序分析

在上面的程序代码中，通过 Previous 属性获取了前一单元格的引用，然后显示该单元格的数值。

## 案例 23 当前单元格的后一单元格

### 1. 功能说明

前面小节的例子中，已经详细介绍了如何获取当前单元格的前一单元格的引用。类似的，Excel 中同样可以引用当前单元格的后一单元格。在本节中，将详细介绍如何获取当前单元格的后一单元格。

### 2. 语法说明

使用 Range 对象的 Next 属性，可获取对指定单元格的后一个单元格的引用。其语法格式如下：

**表达式.Next**

如果表达式为一个对象为区域，则 Next 属性会模拟 Tab 键，但此属性只是返回下一单元格，并不选定它。

### 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区。本例中，用户在选中某单元格后，需要查看该单元格的后面一个单元格数据。原始数据如图 3.9 所示。



The image shows an Excel spreadsheet with the following data:

	A	B	C	D	E	F
1	编号	地区	上半年销量	下半年销量		
2	114	北部	1600	1300		
3	113	北部	1810	1330		
4	116	东部	1110	1300		
5	118	东部	1124	1375		
6	117	东部	3420	1390		
7	111	南部	1141	1375		
8	110	南部	1180	1300		
9	112	南部	1460	1300		
10	115	西部	1530	1270		
11	总计		14375	11940		
12						

图 3.9 原始数据

### 4. 编写代码

取当前单元格的后一单元格的引用的代码具体如下：

```
Sub next_cells()  
    Dim rng As Range  
    Set rng = ActiveCell.Next  
    MsgBox "当前单元格的后一单元格的值为: " & rng.Value  
    Set rng = Nothing  
End Sub
```

### 5. 运行结果

选择单元格 C7，运行程序代码，得到的结果如图 3.10 所示。



The image shows the same Excel spreadsheet as in Figure 3.9, but with a message box overlay. The message box contains the text: "当前单元格的后一单元格的值为: 1375". The message box has a "确定" (OK) button. The spreadsheet data is the same as in Figure 3.9.

图 3.10 运行结果

## 6. 程序分析

在上面的程序代码中，通过 `Next` 属性获取了后一单元格的引用，然后显示该单元格的数值。

## 案例 24 引用整行或者整列单元格

### 1. 功能说明

在使用 Excel VBA 进行数据处理的时候，经常会需要对某列或某行单元格进行设置的情况。这个使用，用户就需要首先选择整行或者整列单元格区域。

### 2. 语法说明

在 Excel VBA 中，可以使用多种方法选择整行或者整列。首先，可用 `Rows` 属性或 `Columns` 属性来处理整行或整列。这两个属性返回代表单元格区域的 `Range` 对象。在下面的代码中，将返回工作表的第二行，然后将区域字体加粗。

```
Worksheets("Sheet1").Rows(2).Font.Bold = True
```

下面列出了使用 `Rows` 和 `Columns` 属性的一些行和列的引用。

<code>Rows(2)</code>	第二行
<code>Rows</code>	工作表上所有的行
<code>Columns(3)</code>	第三列
<code>Columns("C")</code>	第三列
<code>Columns</code>	工作表上所有的列

同时，用户可以直接使用 `Range` 属性来返回整行或者整列单元格，如下：

<code>Range("D:D").Select</code>	选择 D 列
<code>Range("D:F").Select</code>	选择 D 列至 F 列
<code>Range("2:2").Select</code>	选择第二行
<code>Range("2:4").Select</code>	选择第 2 行至第 4 行
<code>Range("D:D").EntireColumn</code>	选择 D 列
<code>Range("E1").EntireColumn</code>	选择 E 列

### 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区。本例中，需要设置工作表的最后一行的格式，原始数据如图 3.11 所示。

	A	B	C	D	E
1	编号	地区	上半年销量	下半年销量	
2	114	北部	1600	1300	
3	113	北部	1810	1330	
4	116	东部	1110	1300	
5	118	东部	1124	1375	
6	117	东部	3420	1390	
7	111	南部	1141	1375	
8	110	南部	1180	1300	
9	112	南部	1460	1300	
10	115	西部	1530	1270	
11	总计		14375	11940	
12					

图 3.11 原始数据

#### 4. 编写代码

获取整行单元格引用的代码具体如下：

```
Sub All_RowCol()
```

```
    Range("11:11").Select
```

```
    With Selection.Font
```

```
        .Bold = True
```

```
        .Italic = True
```

```
    End With
```

```
End Sub
```

#### 5. 运行结果

运行程序代码，得到的结果如图 3.12 所示。

	A	B	C	D	E
1	编号	地区	上半年销量	下半年销量	
2	114	北部	1600	1300	
3	113	北部	1810	1330	
4	116	东部	1110	1300	
5	118	东部	1124	1375	
6	117	东部	3420	1390	
7	111	南部	1141	1375	
8	110	南部	1180	1300	
9	112	南部	1460	1300	
10	115	西部	1530	1270	
11	<b>总计</b>		<b>14375</b>	<b>11940</b>	
12					

图 3.12 运行结果

## 6. 程序分析

在上面的程序代码中，通过代码“Range("11:11").Select”引用了第 11 行的所有数据。

## 案例 25 引用相对其他单元格的单元格

### 1. 功能说明

前面小节已经讲解了多种不同的引用方法。使用不同的引用样式，用户可以应用不同的单元格区域。其中，以某单元格为基准单元格，然后根据偏移数引用其他单元格应用十分广泛。理论上分析，使用这种方法可以引用任何相对位置。

### 2. 语法说明

在 Excel VBA 中，用户可以使用 Range 对象的 Offset 属性返回 Range 对象，代表位于指定单元格区域的一定的偏移量位置上的区域。其使用格式为：

**Offset(RowOffset, ColumnOffset)**

两个参数的含如下：

- RowOffset 为行偏移量，区域偏移的行数可为正数、负数或 0（零）。正数表示向下偏移，负数表示向上偏移。默认值是 0；
- ColumnOffset 为列偏移量，区域偏移的列数可为正数、负数或 0（零）。正数表示向右偏移，负数表示向左偏移。默认值是 0。

例如下面的程序代码：

Range("E4").Offset(, -1)

表示单元格 D4

Range("B2").Offset(2, 2)

表示单元格 D4

### 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区。根据用户选择的单元格，设置该单元格右下方单元格的背景。原始数据如图 3.13 所示。

	A	B	C	D	E	F
	编号	地区	上半年销量	下半年销量		
2	114	北部	1600	1300		
3	113	北部	1810	1330		
4	116	东部	1110	1300		
5	118	东部	1124	1375		
6	117	东部	3420	1390		
7	111	南部	1141	1375		
8	110	南部	1180	1300		
9	112	南部	1460	1300		
10	115	西部	1530	1270		
11	总计		14375	11940		

图 3.13 原始数据

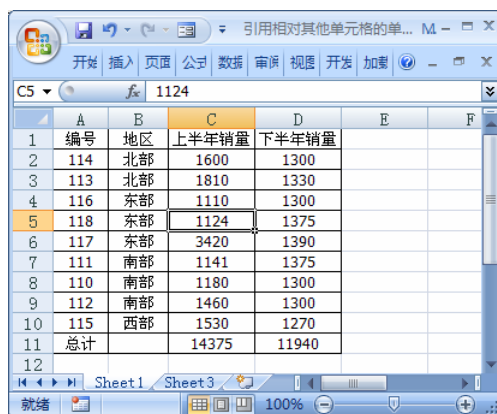
#### 4. 编写代码

引用相对其他单元格的单元格的具体代码如下：

```
Sub Offset_Cells()  
    Dim rng As Range  
    Set rng = ActiveCell.Offset(1, 1)  
    rng.Select  
    Selection.Interior.ColorIndex = 3  
    Set rng = Nothing  
End Sub
```

#### 5. 运行结果

选择工作表的单元格 C5，如图 3.14 所示。



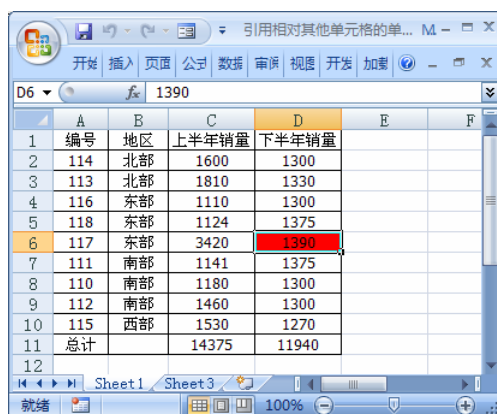
The screenshot shows an Excel spreadsheet with the following data:

	A	B	C	D	E	F
1	编号	地区	上半年销量	下半年销量		
2	114	北部	1600	1300		
3	113	北部	1810	1330		
4	116	东部	1110	1300		
5	118	东部	1124	1375		
6	117	东部	3420	1390		
7	111	南部	1141	1375		
8	110	南部	1180	1300		
9	112	南部	1460	1300		
10	115	西部	1530	1270		
11	总计		14375	11940		

Cell C5 is selected, and the formula bar shows the value 1124.

图 3.14 选中单元格

运行程序代码后，得到的程序结果如图 3.15 所示。



The screenshot shows the same Excel spreadsheet as Figure 3.14, but now cell D6 is selected, and its value (1390) is highlighted in red. This is the result of the VBA code being executed on cell C5.

	A	B	C	D	E	F
1	编号	地区	上半年销量	下半年销量		
2	114	北部	1600	1300		
3	113	北部	1810	1330		
4	116	东部	1110	1300		
5	118	东部	1124	1375		
6	117	东部	3420	1390		
7	111	南部	1141	1375		
8	110	南部	1180	1300		
9	112	南部	1460	1300		
10	115	西部	1530	1270		
11	总计		14375	11940		

Cell D6 is selected, and the formula bar shows the value 1390.

图 3.15 运行的程序代码结果

## 6. 程序分析

在上面的程序代码中，是以当前单元格为基准，所以每执行一次上面的子过程，当前单元格就向下移动三行、向右移动三列。

## 案例 26 引用当前区域

### 1. 功能说明

当前区域是 Excel 工作表中一个十分特殊的单元格区域，是指以空行与空列的组合为边界的区域。当前区域这个概念在数据处理领域使用的十分普遍，特别是当用户需要添加、编辑或者修改当前数据的时候。

在工作表中，选择“开始”|“编辑”|“查找和选择”|“定位条件”命令，在“定位条件”对话框中，选中“当前区域”选项按钮，也能获取关于当前区域的信息，如图 3.16 所示。

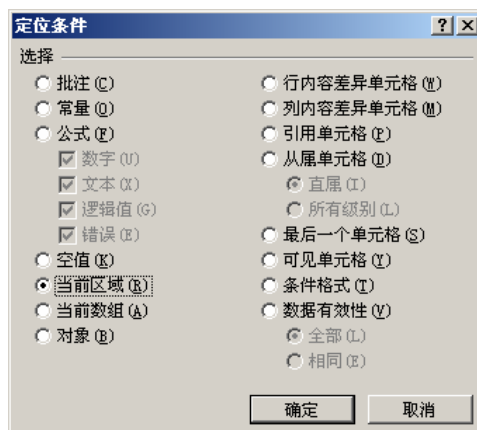


图 3.16 选择“当前区域”选项

### 2. 语法说明

在 Excel VBA 中，使用 Range 对象的 CurrentRegion 属性，可返回表示当前区域的 Range 对象。其语法格式如下：

**表达式.CurrentRegion**

该属性对于许多自动展开选择区域，以包括整个当前区域的操作很有用。

注意：该属性不能用于被保护的工作表。

### 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区。现在需要通过 VBA 代码获取当前区域的信息。原始数据如图 3.17 所示。





	A	B	C	D	E
1	编号	地区	上半年销量	下半年销量	
2	114	北部	1600	1300	
3	113	北部	1810	1330	
4	116	东部	1110	1300	
5	118	东部	1124	1375	
6	117	东部	3420	1390	
7	111	南部	1141	1375	
8	110	南部	1180	1300	
9	112	南部	1460	1300	
10	115	西部	1530	1270	
11	总计		14375	11940	
12					

图 3.17 原始数据

#### 4. 编写代码

引用当前区域的具体代码如下：

```
Sub Current_Region()  
    Dim rng As Range  
    Set rng = Range("A1")  
    Set rng = rng.CurrentRegion  
    rng.Select  
    Set rng = Nothing  
End Sub
```

#### 5. 运行结果

运行程序代码，得到的结果如图 3.18 所示。



	A	B	C	D	E
1	编号	地区	上半年销量	下半年销量	
2	114	北部	1600	1300	
3	113	北部	1810	1330	
4	116	东部	1110	1300	
5	118	东部	1124	1375	
6	117	东部	3420	1390	
7	111	南部	1141	1375	
8	110	南部	1180	1300	
9	112	南部	1460	1300	
10	115	西部	1530	1270	
11	总计		14375	11940	
12					

平均值: 1850.206897 计数: 43 求和: 53656

图 3.18 引用当前区域

## 6. 程序分析

在当前区域范围内，无论活动单元格是哪一个单元格，所在的当前区域均为同一区域。如上例中的 A1:D11 区域，活动单元格 B2 的当前区域为 A1:D11，当活动单元格为 D2 时，其当前区域仍为 A1:D11。

## 案例 27 引用已使用区域

### 1. 功能说明

已使用区域在 Excel 中是一个十分常用的概念，其主要功能是显示当前使用的单元格的范围区域。和当前区域不同，已使用区域表示的是所有工作表中所使用单元格的区域。

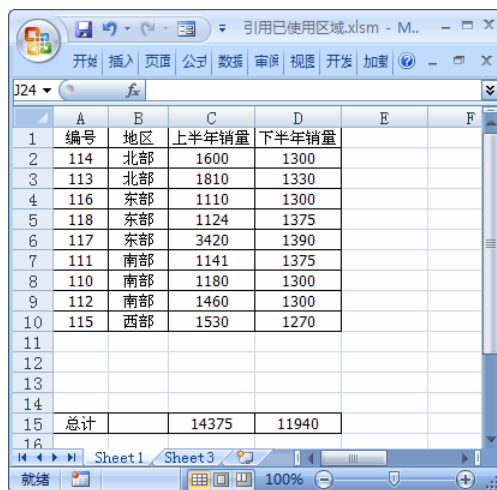
### 2. 语法说明

在 Excel VBA 中，UsedRange 属性返回指定工作表中已使用区域的 Range 对象，即返回工作表中已使用的单元格区域。因此，该属性也可以用于选取单元格区域。UsedRange 属性与上例中的 CurrentRegion 属性的区别：

- UsedRange 属性是 Worksheet 对象的一个属性，返回指定工作表中所有已使用单元格区域，无论各单元格之间是否有空行或空列隔开。
- CurrentRegion 属性是 Range 对象的一个属性，返回的是一个由空行空列围起来的区域，空行空列之外的单元格不被包含在内。

### 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区。现在需要通过 VBA 代码获取已使用区域的信息。原始数据如图 3.19 所示。



	A	B	C	D	E	F
1	编号	地区	上半年销量	下半年销量		
2	114	北部	1600	1300		
3	113	北部	1810	1330		
4	116	东部	1110	1300		
5	118	东部	1124	1375		
6	117	东部	3420	1390		
7	111	南部	1141	1375		
8	110	南部	1180	1300		
9	112	南部	1460	1300		
10	115	西部	1530	1270		
11						
12						
13						
14						
15	总计		14375	11940		
16						

图 3.19 原始数据

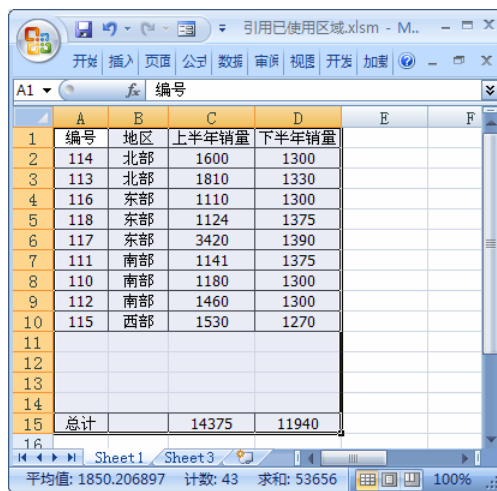
#### 4. 编写代码

选择已使用区域的程序代码如下：

```
Sub used_range()  
    Dim rng As Range  
    Set rng = ActiveSheet.UsedRange  
    rng.Select  
    Set rng = Nothing  
End Sub
```

#### 5. 运行结果

运行程序代码，得到的结果如图 3.20 所示。



	A	B	C	D	E	F
1	编号	地区	上半年销量	下半年销量		
2	114	北部	1600	1300		
3	113	北部	1810	1330		
4	116	东部	1110	1300		
5	118	东部	1124	1375		
6	117	东部	3420	1390		
7	111	南部	1141	1375		
8	110	南部	1180	1300		
9	112	南部	1460	1300		
10	115	西部	1530	1270		
11						
12						
13						
14						
15	总计		14375	11940		
16						

图 3.20 选择已使用区域

#### 6. 程序分析

当用户使用 `UsedRange` 属性来获取已使用的区域时，可以通过 `Address` 属性返回该区域的地址。例如，在上面的程序代码中添加下面的代码：

```
MsgBox rng.Address
```

运行程序代码后，会显示对应的地址，如图 3.21 所示。



图 3.21 显示区域的地址

## 案例 28 调整单元格区域的大小

### 1. 功能说明

在前面小节中，用户已经了解了如何使用 VBA 来引用单元格区域的方法。在实际应用中，用户可能在程序代码中需要调整引用的单元格区域。这个时候，用户就需要首先调整单元格区域的大小。在本小节中，将详细讲解如何调整单元格区域的大小。

### 2. 语法说明

在 Excel VBA 中，使用 Range 对象的 Resize 属性，可以调整指定单元格区域的大小，并返回一个 Range 对象，该对象代表调整后的区域。其语法格式如下：

表达式.Resize(RowSize, ColumnSize)

两个参数的含义如下：

- RowSize 为新区域中的行数。如果省略该参数，则该区域中的行数保持不变。
- ColumnSize 为新区域中的列数。如果省略该参数，则该区域中的列数保持不变。

### 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区。现在需要通过 VBA 代码引用除标题行之外的单元格。原始数据如图 3.22 所示。



	A	B	C	D	E
1	编号	地区	上半年销量	下半年销量	
2	114	北部	1600	1300	
3	113	北部	1810	1330	
4	116	东部	1110	1300	
5	118	东部	1124	1375	
6	117	东部	3420	1390	
7	111	南部	1141	1375	
8	110	南部	1180	1300	
9	112	南部	1460	1300	
10	115	西部	1530	1270	
11	总计		14375	11940	
12					

图 3.22 原始数据

### 4. 编写代码

调整单元格区域大小的代码如下：

```
Sub Change_Cells()  
Dim rng As Range  
  
Set rng = ActiveCell.CurrentRegion  
  
rng.Offset(1, 0).Resize(rng.Rows.Count - 1, _
```

```
rng.Columns.Count).Select
```

```
Set rng = Nothing
```

```
End Sub
```

## 5. 运行结果

运行程序代码，得到的结果如图 3.23 所示。



	A	B	C	D	E
1	编号	地区	上半年销量	下半年销量	
2	114	北部	1600	1300	
3	113	北部	1810	1330	
4	116	东部	1110	1300	
5	118	东部	1124	1375	
6	117	东部	3420	1390	
7	111	南部	1141	1375	
8	110	南部	1180	1300	
9	112	南部	1460	1300	
10	115	西部	1530	1270	
11	总计		14375	11940	
12					

图 3.23 引用结果

## 6. 程序分析

在上面的代码中，先用 `Offset` 将当前区域向下偏移一个单元格，然后使用 `Resize` 属性，将区域的行数减少 1。这个方法有广泛的使用范围，可以选择多种数据区域。

# 案例 29 引用命名的单元格区域

## 1. 功能说明

在 Excel 的基本操作中，对单元格区域进行命名是常见的操作。对一些特定的单元格区域进行命名，可以方便用户使用函数和公式，也可以更加便利其他读者阅读。在 Excel VBA 中，用户同样可以通过代码引用命名的单元格区域。

## 2. 语法说明

在本小节中，主要涉及到 `Range` 属性的用法。关于 `Range` 属性的具体用法，前面小节已经讲解过。本例中讲解的是，使用 `Range` 引用命名区域的方法。其具体的表达是：

```
Range (" name" )
```

其中的参数 `name` 表示的是命名区域的名称。该表达式返回的是 `Range` 类型的变量。

### 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区，并对该原始数据进行命名。现在需要通过 Excel VBA 引用该命名的区域，原始数据如图 3.24 所示。



	A	B	C	D	E
1	编号	地区	上半年销量	下半年销量	
2	114	北部	1600	1300	
3	113	北部	1810	1330	
4	116	东部	1110	1300	
5	118	东部	1124	1375	
6	117	东部	3420	1390	
7	111	南部	1141	1375	
8	110	南部	1180	1300	
9	112	南部	1460	1300	
10	115	西部	1530	1270	
11	总计		14375	11940	
12					

图 3.24 原始数据

### 4. 编写代码

为了演示本案例，用户需要首先设定原始数据的名称，然后再通过 VBA 代码对该名称进行引用。具体步骤如下：

(1) 选择定义名称。选择单元格区域 A1~D11，然后选择“公式”|“定义的名称”|“定义名称”|“定义名称”选项，如图 3.25 所示。

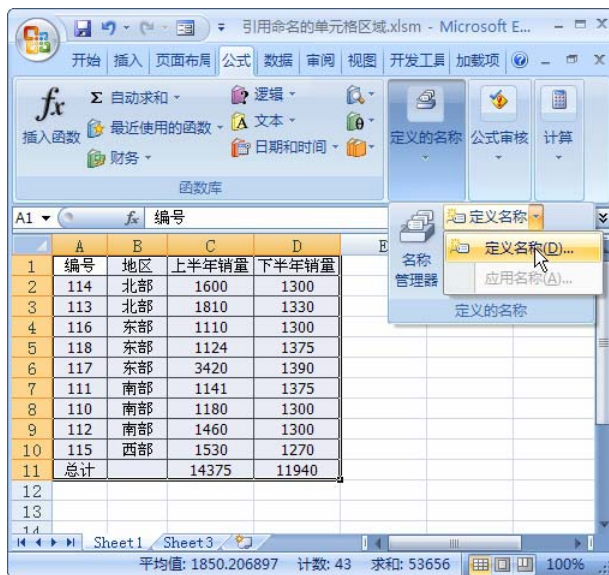


图 3.25 选择定义名称

(2) 设置名称。在打开的“新建名称”对话框中，在“名称”选框中输入“Source”，引用位置保持默认范围，如图 3.26 所示。



图 3.26 输入定义的名称

(3) 查看结果。单击上面对话框中的“确定”按钮，查看名称设定的结果，如图 3.27 所示。



图 3.27 查看定义名称的结果

(4) 引用名称单元格区域的代码如下：

```
Sub UseDataName()  
  
Dim rng As Range  
  
Set rng = Range("Source")  
  
rng.Select  
Set rng = Nothing  
End Sub
```

## 5. 运行结果

运行程序代码，得到的结果如图 3.28 所示。

	A	B	C	D	E	F
1	编号	地区	上半年销量	下半年销量		
2	114	北部	1600	1300		
3	113	北部	1810	1330		
4	116	东部	1110	1300		
5	118	东部	1124	1375		
6	117	东部	3420	1390		
7	111	南部	1141	1375		
8	110	南部	1180	1300		
9	112	南部	1460	1300		
10	115	西部	1530	1270		
11	总计		14375	11940		

图 3.28 引用结果

## 6. 程序分析

在 Excel VBA 中, 在很多情况下, 使用命名区域的方法选择单元格会比引用单元格区域便利, 用户可以尝试使用命令单元格区域的方法解决引用问题。

## 案例 30 引用交叉区域

### 1. 功能说明

在数据分析中, 很多情况下需要引用交叉领域的单元格区域。这种引用区域和前面小节例子中讲解的区域引用在性质上会有不同。在前面的案例中, 用户引用的大多是单个区域, 而交叉区域则主要是多个区域的交集运算。在本小节中, 将详细讲解如何引用交叉区域的内容。

### 2. 语法说明

在 Excel VBA 中, 使用 `Application` 对象的 `Intersect` 方法, 可返回 `Range` 对象, 该对象包含将两个或多个区域重叠的矩形区域。其语法格式如下:

```
表达式.Intersect(Arg1, Arg2, Arg3, Arg4, Arg5, Arg6, Arg7, Arg8, Arg9, Arg10, Arg11, Arg12, Arg13, Arg14, Arg15, Arg16, Arg17, Arg18, Arg19, Arg20, Arg21, Arg22, Arg23, Arg24, Arg25, Arg26, Arg27, Arg28, Arg29, Arg30)
```

使用 `Intersect` 方法时, 至少需要 2 个 `Range` 区域, 最多 30 个 `Range` 区域进行计算。

### 3. 案例说明

某公司统计了部分员工上、下两半年销量, 同时提供了员工所处的地区数据。在本例中, 用户希望通过交叉区域来引用单元格, 并设置单元格的底纹, 原始数据如图 3.29 所示。





	A	B	C	D	E
1	编号	地区	上半年销量	下半年销量	
2	114	北部	1600	1300	
3	113	北部	1810	1330	
4	116	东部	1110	1300	
5	118	东部	1124	1375	
6	117	东部	3420	1390	
7	111	南部	1141	1375	
8	110	南部	1180	1300	
9	112	南部	1460	1300	
10	115	西部	1530	1270	
11	总计		14375	11940	

图 3.29 原始数据

#### 4. 编写代码

引用交叉区域的代码如下。

```
Sub intersect_range()
```

```
Dim rng As Range
```

```
Set rng = Application.Intersect(Range("3:3"), Range("D:D"))
```

```
rng.Select
```

```
Selection.Interior.ColorIndex = 3
```

```
Set rng = Nothing
```

```
End Sub
```

#### 5. 运行结果

运行程序代码，得到的结果如图 3.30 所示。



	A	B	C	D	E
1	编号	地区	上半年销量	下半年销量	
2	114	北部	1600	1300	
3	113	北部	1810	1330	
4	116	东部	1110	1300	
5	118	东部	1124	1375	
6	117	东部	3420	1390	
7	111	南部	1141	1375	
8	110	南部	1180	1300	
9	112	南部	1460	1300	
10	115	西部	1530	1270	
11	总计		14375	11940	

图 3.30 运行程序代码结果

## 6. 程序分析

在上面的例子中，为了方便讲解，用的直接选择单元格区域，然后选择交叉的区域。在实际应用中，用户也可以使用命名区域来完成交叉引用。

## 案例 31 引用区域内的单元格

### 1. 功能说明

在使用 Excel 处理数据的时候，很多情况下处理都是某些区域范围内的单元格。为此，当用户需要处理区域内的数据时，则需要首先引用区域内的单元格。在本小节中，将详细讲解如何引用区域内的单元格。

### 2. 语法说明

在 Excel VBA 中，获取选择区域中的某个单元格值，可以使用 Range 对象的 Item 属性来访问。Item 属性返回 Range 对象，代表对指定区域某一偏移量处的区域。其语法格式如下：

**表达式.Item(RowIndex, ColumnIndex)**

其中参数 RowIndex 是必选的，表示要访问的单元格的索引号，顺序为从左到右，然后往下。例如：

**Range.Item(1)**

返回区域左上角单元格：

**Range.Item(2)**

返回紧靠左上角单元格右侧的单元格。

参数 ColumnIndex 可省略，用来指明要访问的单元格所在列的列号的数字或字符串，用数字 1 或字符“A”表示区域中的第一列。例如：

**Range("A1:E5").Item(2,2)**

表示单元格 B2，这个单元格处于以指定区域中左上角单元格 A1（为起点的第 2 行第 2 列。因为 Item 属性为默认属性，因此也可以简写为：

**Range("A1:E10")(2,2)**

当省略 ColumnIndex 参数时，计数方式为从左向右，即在区域中的第一行开始从左向右计数，第一行结束后，然后从第二行开始从左到右接着计数，依此类推。

**技巧：**Item 属性的值即可为正数，也可为负数。其序号值指定的单元格不一定包含在引用的区域内。

### 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区数据。现在需要引用数值单元格区域中的某个单元格，原始数据如图 3.31 所示。



	A	B	C	D	E
1	编号	地区	上半年销量	下半年销量	
2	114	北部	1600	1300	
3	113	北部	1810	1330	
4	116	东部	1110	1300	
5	118	东部	1124	1375	
6	117	东部	3420	1390	
7	111	南部	1141	1375	
8	110	南部	1180	1300	
9	112	南部	1460	1300	
10	115	西部	1530	1270	
11	总计		14375	11940	
12					

图 3.31 原始数据

#### 4. 编写代码

引用区域内单元格的具体代码如下：

```
Sub Choose_range()  
    Dim rng1 As Range  
    Dim rng2 As Range  
    Dim str As String  
  
    Set rng1 = Range("A1").CurrentRegion  
    Set rng2 = rng1.Offset(1, 1).Resize(rng1.Rows.Count - 1, rng1.Columns.Count - 1)  
  
    rng2.Select  
  
    str = "所选区域: " & Selection.Address & vbCrLf  
    str = str & "区域第 5 行第 3 列单元格的数值是: " & Selection.Item(5, 3)  
    MsgBox str  
  
End Sub
```

#### 5. 运行结果

运行程序代码，得到的结果如图 3.32 所示。

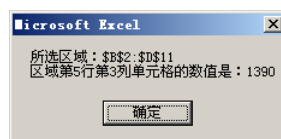


图 3.32 运行程序的结果

## 6. 程序分析

在上面的代码中，使用 Range 对象的 Address 属性显示引用区域的地址。

## 案例 32 引用最后一行单元格

### 1. 功能说明

在 Excel 的数据表中，最后一行单元格是比较特殊的单元格区域。当用户需要更新数据的时候，都需要从区域的最后一行单元格开始。因此，用户需要特别处理区域最后一行单元格的情况。在本小节中，将详细讲解如何引用最后一行单元格。

### 2. 语法说明

在 Excel VBA 中，用户可以通过使用 Range 对象的 End 属性来实现类似的操作。使用 Range 对象的 End 属性将返回 Range 对象，该对象代表包含源区域的区域尾端的单元格。其语法格式如下：

表达式.End(Direction)

参数 Direction 指定移动的方向，具体的参数选择如下：

- xlDown: 向下；
- xlToLeft: 向左；
- xlToRight: 向右；
- xlUp: 向上。

在本例中，由于要确定的是末行单元格，因此参数选择应该是 xlDown。通过程序代码确定工作表的末行，这是十分常用的方法。在 Excel 的基础操作中，按组合键“Ctrl+光标键”，可定位动作表的数据区域边界。例如，按“Ctrl+向下光标”，用户就可以定位工作表的末行，如图 3.33 所示。

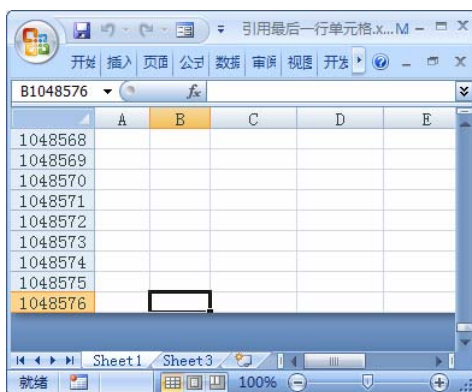


图 3.33 引用最后一行单元格

### 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区数据。由于数据不完整，现在需要在没有完成的数据后面添加新数据，原始数据如图 3.34 所示。



	A	B	C	D	E	F
1	编号	地区	上半年销量	下半年销量		
2	114	北部	1600	1300		
3	113	北部	1810	1330		
4	116	东部	1110	1300		
5	118	东部	1124	1375		
6	117	东部	3420	1390		
7	111	南部	1141			
8	110	南部	1180			
9	112	南部	1460			
10	115	西部	1530			
11						
12						

图 3.34 原始数据

### 4. 编写代码

添加新数据的具体代码如下：

```
Sub Add_Data()
    Dim rng As Range
    Dim Data As Integer

    Set rng = ActiveSheet.Range("D1").End(xlDown)
    Data = InputBox("输入新数据", "输入销量")

    rng.Offset(1, 0).Select
    Selection.Value = Data
End Sub
```

### 5. 运行结果

运行程序代码，提示输入新的数据，如图 3.35 所示。



图 3.35 提示输入新的数据

单击对话框中的“确定”按钮，查看添加的结果，如图 3.36 所示。

	A	B	C	D	E	F
1	编号	地区	上半年销量	下半年销量		
2	114	北部	1600	1300		
3	113	北部	1810	1330		
4	116	东部	1110	1300		
5	118	东部	1124	1375		
6	117	东部	3420	1390		
7	111	南部	1141	2580		
8	110	南部	1180			
9	112	南部	1460			
10	115	西部	1530			
11						
12						

图 3.36 添加的结果

## 6. 程序分析

上面的代码本身并不复杂，但是应用十分广泛。例如，用户可以重复运行上面的代码，然后依次添加新的数据，或者继续扩展或者细化代码，这里就不详细介绍。

## 案例 33 引用列末单元格

### 1. 功能说明

在 Excel 的数据表中，用户需要有引用行末单元格的需要之外，还需要引用列末单元格。作为二维数据表的 Excel，列和行是两个经常用来处理的对象。因此，在前面例子讲解了如何引用最后一行单元格后，本小节将讲解如何引用列末单元格。

### 2. 语法说明

在 Excel VBA 中，使用 Range 对象的 End 属性将返回一个 Range 对象，该对象代表包含源区域的区域尾端的单元格。其语法格式如下：

**表达式.End(Direction)**

参数 Direction 指定移动的方向，可为以下常量之一：

- xlDown: 向下
- xlToLeft: 向左
- xlToRight: 向右
- xlUp: 向上

在本例中，由于要确定的是末行单元格，因此参数选择应该是 xlToRight。通过程序代码确定工作表的末列，这是十分常用的方法。在 Excel 的基础操作中，按“Ctrl+向右光标”，用户就可以定位工作表的末行，如图 3.37 所示。

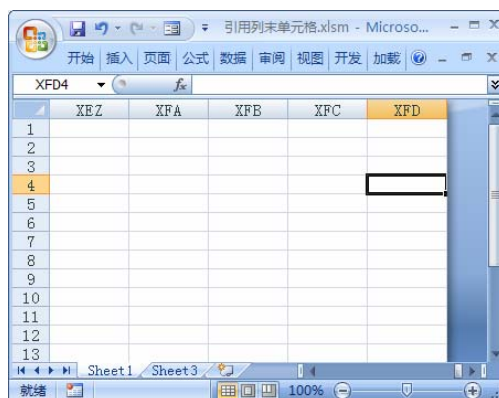


图 3.37 引用列末单元格

### 3. 案例说明

在本例中，将演示如何使用 Excel VBA 代码来引用列末的单元格。

### 4. 编写代码

引用列末单元格的具体代码如下：

```
Sub selectendcells()  
    Dim rng As Range  
    Set rng = ActiveCell.End(xlToRight)  
    rng.Select  
    Set rng = Nothing  
End Sub
```

### 5. 运行结果

运行程序代码，提示输入新的数据，如图 3.38 所示。

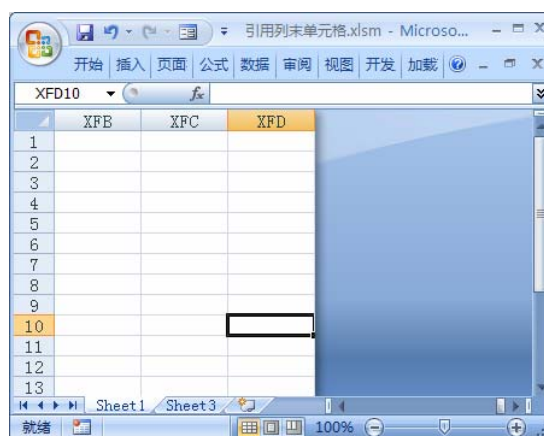


图 3.38 运行程序代码结果

## 6. 程序分析

在实际的数据处理中，用户需要确定所需处理单元格的列向范围，这个时候用户就需要选择列末的单元格，或者单元格的地址。

## 案例 33 引用整行数据

### 1. 功能说明

引用整行已经有数据的单元格是数据处理中经常遇到的操作。特别是当某行单元格的数据恰好是某类数据的时候，引用该行数据就代表着可以对整类数据进行处理分析。在本小节中，将详细讲解如何引用整行数据。

### 2. 语法说明

本小节依然使用 **Range** 对象的 **End** 属性获取指定数据行最右端的单元格。为了确定区域范围，除了需要选中单元格区域之外，还需要获列数信息。需要用到 **Column** 属性，**Range.Column** 属性返回指定区域中第一块中的第一列的列号。语法表达式如下：

表达式.Column

表达式是代表 **Range** 对象的变量。在结果中，A 列返回数值 1，B 列返回数值 2，依此类推。

### 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区数据。现在需要通过 VBA 代码选择整行，原始数据如图 3.39 所示。



	A	B	C	D	E
1	编号	地区	上半年销量	下半年销量	
2	114	北部	1600	1300	
3	113	北部	1810	1330	
4	116	东部	1110	1300	
5	118	东部	1124	1375	
6	117	东部	3420	1390	
7	111	南部	1141	2580	
8	110	南部	1180	1385	
9	112	南部	1460	2345	
10	115	西部	1530	1950	
11					
12					
13					
14					

图 3.39 原始数据

### 4. 编写代码

选择整行单元格的具体代码如下：



```

Sub SelectRow()
    Dim rng As Range
    Dim IndexNumber As Integer
    Dim Intcol As Integer

    IndexNumber = Application.InputBox(prompt:="输入引用的行数: ", Type:=1)
    Intcol = Range("A" & IndexNumber).End(xlToRight).Column

    Set rng = Range(Cells(IndexNumber, 1), Cells(IndexNumber, Intcol))
    rng.Select

    Set rng = Nothing
End Sub

```

### 5. 运行结果

运行程序代码，提示输入需要引用的数据行，如图 3.40 所示。



图 3.40 提示输入的行数

单击对话框中的“确定”按钮，查看引用的数据结果，如图 3.41 所示。

	A	B	C	D	E
1	编号	地区	上半年销量	下半年销量	
2	114	北部	1600	1300	
3	113	北部	1810	1330	
4	116	东部	1110	1300	
5	118	东部	1124	1375	
6	117	东部	3420	1390	
7	111	南部	1141	2580	
8	110	南部	1180	1385	
9	112	南部	1460	2345	
10	115	西部	1530	1950	
11					

图 3.41 引用单元格的结果

### 6. 程序分析

在上面的代码中，设置 Application 对象的 InputBox 方法的 Type 参数为 1，表示接收用户输入的数据为数值型。代码 Range("A" & i).End(xlToRight).Column 获取最右侧的列数。

## 案例 34 引用不同长度的非连续列

### 1. 功能说明

在 Excel 处理数据的时候，很多时候用户需要选择不规则的单元格区域。例如，当两个数据列代表不同类型的数据，用户需要同时选中两个数据列的数据。同时，这两个数据列的行数未必相同。因此，需要引用不同长度的非连续列。本小节将详细讲解如何引用不同长度的非连续列。

### 2. 语法说明

在 Excel VBA 中，如果用户希望引用不同长度的非连续列，则需要使用 Range 对象的 End 属性和 Union 方法。具体的步骤如下：

(1) 使用 End 属性获取多个数据列的行数，确定引用的单元格区域范围；

(2) 通过 Range 方法引用多个区域，然后使用 Union 方法将两个 Range 对象区域组合在一起。

### 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区数据。现在需要同时引用“地区”和“下半年销量”数据列，原始数据如图 3.42 所示。



	A	B	C	D	E
1	编号	地区	上半年销量	下半年销量	
2	114	北部	1600	1300	
3	113	北部	1810	1330	
4	116	东部	1110	1300	
5	118	东部	1124	1375	
6	117	东部	3420	1390	
7	111	南部	1141	2580	
8	110	南部	1180	1385	
9	112	南部	1460		
10	115	西部	1530		
11					

图 3.42 原始数据

### 4. 编写代码

引用两个数据列的具体代码如下：

```
Sub SelectMltiRange()

    Dim StartCell As String
    Dim EndCell As String
    Dim FirstRange As Range
    Dim LastRange As Range
```

```
StartCell = "B1"  
EndCell = "D1"  
  
Set FirstRange = Range(StartCell, Range(StartCell).End(xlDown))  
Set LastRange = Range(EndCell, Range(EndCell).End(xlDown))  
  
Union(FirstRange, LastRange).Select  
  
End Sub
```

## 5. 运行结果

运行程序代码，查看引用单元格的范围，如图 3.43 所示。



	A	B	C	D	E
1	编号	地区	上半年销量	下半年销量	
2	114	北部	1600	1300	
3	113	北部	1810	1330	
4	116	东部	1110	1300	
5	118	东部	1124	1375	
6	117	东部	3420	1390	
7	111	南部	1141	2580	
8	110	南部	1180	1385	
9	112	南部	1460		
10	115	西部	1530		

图 3.43 引用不同数据列的结果

## 6. 程序分析

在上面的代码中，选择了两列不连续的数据列，如果数据本身繁多，可以使用类似的代码，引用不同的数据列。

# 案例 35 引用条件格式单元格

## 1. 功能说明

条件格式是在 Excel 处理数据中十分常见的数据分析功能，对于数据列设置条件格式，可以显示原始数据列的数据特点。而且，当用户设置了条件格式后，该单元格将不是普通的单元格。在 Excel VBA 中，需要使用特别的样式来引用条件格式单元格。

## 2. 语法说明

在 Excel VBA 中，使用 Range 对象的 SpecialCells 方法可返回 Range 对象，该对象代表与指定类型和值匹配的所有单元格。该方法的语法格式如下：

**表达式.SpecialCells(Type, Value)**

参数 Type 为一个常量，指定要包含的单元格，常用的常量如下：

- xlCellTypeAllFormatConditions: 含有条件格式单元格；
- xlCellTypeAllValidation: 含有验证条件的单元格；
- xlCellTypeBlanks: 空单元格；
- xlCellTypeComments: 含有注释的单元格；
- xlCellTypeConstants: 含有常量的单元格；
- xlCellTypeFormulas: 含有公式的单元格；
- xlCellTypeLastCell: 已用区域中的最后一个单元格；
- xlCellTypeSameFormatConditions: 含有相同格式的单元格；
- xlCellTypeSameValidation: 含有相同验证条件的单元格；
- xlCellTypeVisible: 所有可见单元格。

参数 Value 可省略。如果参数 Type 为 xlCellTypeConstants 或 xlCellTypeFormulas，则该参数可用于确定结果中应包含哪几类单元格。将这些值相加可使此方法返回多种类型的单元格。默认情况下，将选择所有常量或公式，无论类型如何。该参数可为以下常量值：

- xlErrors: 错误值；
- xlLogical: 逻辑值；
- xlNumbers: 数值；
- xlTextValues: 文本。

### 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区数据。现在需要对该数据设置条件格式，并使用代码引用条件格式的数据列，原始数据如图 3.44 所示。



	A	B	C	D	E	F
1	编号	地区	上半年销量	下半年销量		
2	114	北部	1600	1300		
3	113	北部	1810	1330		
4	116	东部	1110	1300		
5	118	东部	1124	1375		
6	117	东部	3420	1390		
7	111	南部	1141	2580		
8	110	南部	1180	1385		
9	112	南部	1460	2345		
10	115	西部	1530	1950		

图 3.44 原始数据

### 4. 编写代码

在本例中，首先需要为原始数据添加条件格式，然后是代码引用单元格。下面详细讲

解对应的操作步骤。

(1) 添加条件格式。选择单元格 D 列的数据，然后选择“开始”|“样式”|“条件格式”|“色阶”选项，添加数据列的条件格式，如图 3.45 所示。

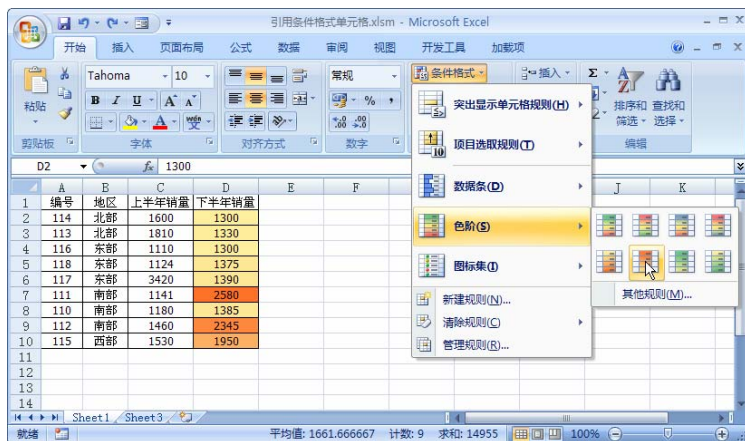


图 3.45 条件格式

(2) 选择条件格式的具体代码如下：

```
Sub Select_Specialcells()
    Dim rng As Range
    On Error GoTo err

    Set rng = Sheet1.Cells.SpecialCells(xlCellTypeAllFormatConditions)
    rng.Select
    Exit Sub

err:
    MsgBox "没有条件格式！"
    Set rng = Nothing

End Sub
```

## 5. 运行结果

运行程序代码，查看引用单元格的范围，如图 3.46 所示。



	A	B	C	D	E	F
1	编号	地区	上半年销量	下半年销量		
2	114	北部	1600	1300		
3	113	北部	1810	1330		
4	116	东部	1110	1300		
5	118	东部	1124	1375		
6	117	东部	3420	1390		
7	111	南部	1141	2580		
8	110	南部	1180	1385		
9	112	南部	1460	2345		
10	115	西部	1530	1950		
11						
12						

图 3.46 选择条件格式单元格

## 6. 程序分析

在 Excel VBA 中, 使用 `SpecialCells` 方法可以返回许多特殊形式的单元格。熟练掌握该方法, 会给用户分析数据带来很大的便利。

## 案例 36 引用所有单元格

### 1. 功能说明

在 Excel 处理数据的时候, 引用工作表的所有单元格是一个常见操作。当用户需要对工作表的单元格进行设置和编辑的时候, 首先需要引用所有的单元格。在本小节中, 将详细讲解如何引用所有单元格。

### 2. 语法说明

在 Excel VBA 中, 可以使用 `Worksheet` 的 `Cells` 属性引用工作表的单元格。该属性返回 `Range` 对象, 表示工作表中的所有单元格。其语法表达式是:

`表达式.Cells`

其中表达式是代表 `Worksheet` 对象的变量。

### 3. 案例说明

某公司统计了部分员工上、下两半年销量, 同时提供了员工所处的地区数据。现在需要同时引用“地区”和“下半年销量”数据列, 原始数据如图 3.42 所示。

	A	B	C	D	E	F
1	编号	地区	上半年销量	下半年销量		
2	114	北部	1600	1300		
3	113	北部	1810	1330		
4	116	东部	1110	1300		
5	118	东部	1124	1375		
6	117	东部	3420	1390		
7	111	南部	1141	2580		
8	110	南部	1180	1385		
9	112	南部	1460	2345		
10	115	西部	1530	1950		
11						

图 3.42 原始数据

#### 4. 编写代码

引用所有单元格的具体代码如下：

```
Sub GetAllCells()  
  
With Worksheets("Sheet1").Cells.Font  
    .Name = "Verdana"  
    .Size = 10  
    .ColorIndex = 26  
End With  
  
End Sub
```

#### 5. 运行结果

运行程序代码，查看设置的结果，如图 3.43 所示。

	A	B	C	D	E	F
1	编号	地区	上半年销量	下半年销量		
2	114	北部	1600	1300		
3	113	北部	1810	1330		
4	116	东部	1110	1300		
5	118	东部	1124	1375		
6	117	东部	3420	1390		
7	111	南部	1141	2580		
8	110	南部	1180	1385		
9	112	南部	1460	2345		
10	115	西部	1530	1950		
11						

图 3.43 查看引用单元格的结果

## 6. 程序分析

上面的程序代码比较简单,其中选择所有单元格区域的代码功能和常见的 Ctrl+A 的操作结果是一样的。

## 案例 37 跨表引用

### 1. 功能说明

在 Excel 处理数据的时候,有时候需要在工作表之间进行跨表操作。这种情况下,用户需要首先进行跨表引用。在本小节中,将详细讲解如何进行跨表引用。

### 2. 语法说明

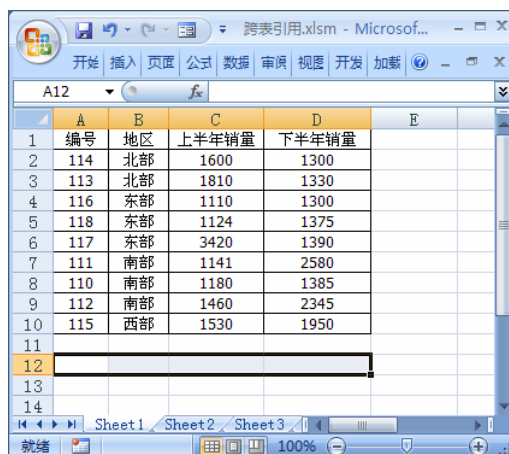
在 Excel VBA 中,使用 `Sheets(array)` 可指定多个工作表,其中 `Array` 函数用来选定两张或多张工作表。如:

```
Sheets(Array("Sheet2", "Sheet3", "Sheet4")).Select
```

多张工作表选定后的操作,与单张工作表操作类似。

### 3. 案例说明

在本例中,需要给三个工作表中的单元格区域设置格式,添加单元格区域的边框线。其中,原始的数据如图 3.44 所示。



	A	B	C	D	E
1	编号	地区	上半年销量	下半年销量	
2	114	北部	1600	1300	
3	113	北部	1810	1330	
4	116	东部	1110	1300	
5	118	东部	1124	1375	
6	117	东部	3420	1390	
7	111	南部	1141	2580	
8	110	南部	1180	1385	
9	112	南部	1460	2345	
10	115	西部	1530	1950	
11					
12					
13					
14					

图 3.44 原始数据

### 4. 编写代码

跨表引用的具体代码如下:

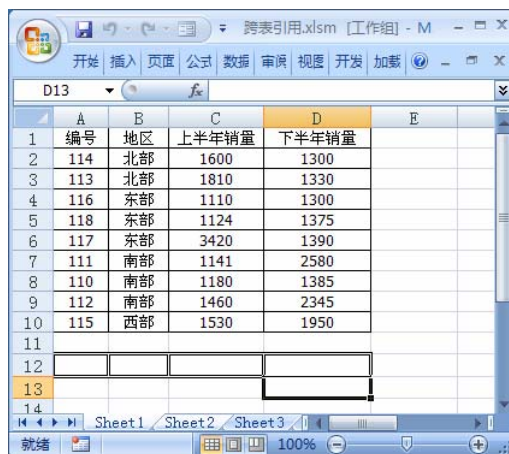
```
Sub AddLine()  
    Dim rng As Range
```



```
Sheets(Array("Sheet1", "Sheet2", "Sheet3")).Select  
Set rng = Range("A12:D12")  
  
rng.Borders.LineStyle = xlDouble  
Set rng = Nothing  
End Sub
```

### 5. 运行结果

运行程序代码，查看设置的结果，如图 3.45 所示。



	A	B	C	D	E
1	编号	地区	上半年销量	下半年销量	
2	114	北部	1600	1300	
3	113	北部	1810	1330	
4	116	东部	1110	1300	
5	118	东部	1124	1375	
6	117	东部	3420	1390	
7	111	南部	1141	2580	
8	110	南部	1180	1385	
9	112	南部	1460	2345	
10	115	西部	1530	1950	
11					
12					
13					
14					

图 3.45 添加边框的结果

### 6. 程序分析

从上面程序代码的结果中可以看出，程序代码直接选中了多个工作表。在 Excel 的标题栏中显示的是“工作组”。

## 3.2 获取单元格信息

在前面的案例中，用户已经了解到如何在 Excel VBA 中引用单元格。引用单元格，是用户对单元格进行操作的基础。在小节中，将详细讲解如何获取单元格的信息。单元格的信息包括行数、列数以及地址等。这些信息是用户在后续数据处理和分析中需要经常用到的，希望用户熟练掌握。

## 案例 38 获取标题行的信息

### 1. 功能说明

在 Excel VBA 中，标题行和数据行是不同的处理对象。标题行的主要功能是显示数据类型，而数据行则是分别显示对应的数据信息。因此，在进行数据处理的时候，需要首先获取标题行的信息。本小节详细讲解如何获取标题行的信息。

### 2. 语法说明

在 Excel VBA 中，使用 Range 对象的 ListHeaderRows 属性，可获取指定区域中标题行的数目。在使用该属性前，先使用 CurrentRegion 属性查找区域的边界。

### 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区数据。现在需要同时引用“地区”和“下半年销量”数据列，原始数据如图 3.46 所示。

	A	B	C	D	E
1	编号	地区	上半年销量	下半年销量	
2	114	北部	1600	1300	
3	113	北部	1810	1330	
4	116	东部	1110	1300	
5	118	东部	1124	1375	
6	117	东部	3420	1390	
7	111	南部	1141	2580	
8	110	南部	1180	1385	
9	112	南部	1460	2345	
10	115	西部	1530	1950	
11					
12					

图 3.46 原始数据

### 4. 编写代码

查看标题栏的具体代码如下：

```
Sub FindHeadLines()  
  
    Dim rngTbl As Range  
    Dim intHdrRows As Integer  
  
    Set rngTbl = ActiveCell.CurrentRegion  
    intHdrRows = rngTbl.ListHeaderRows  
  
    If intHdrRows > 0 Then  
        Set rngTbl = rngTbl.Resize(intHdrRows)  
    End If  
End Sub
```

```
rngTbl.Select  
End If
```

```
End Sub
```

## 5. 运行结果

运行程序代码，查看设置的结果，如图 3.47 所示。



	A	B	C	D	E
1	编号	地区	上半年销量	下半年销量	
2	114	北部	1600	1300	
3	113	北部	1810	1330	
4	116	东部	1110	1300	
5	118	东部	1124	1375	
6	117	东部	3420	1390	
7	111	南部	1141	2580	
8	110	南部	1180	1385	
9	112	南部	1460	2345	
10	115	西部	1530	1950	
11					

图 3.47 选择结果

为了检测程序代码，用户可以在原来的数据上添加新的标题栏，如图 3.48 所示。



	A	B	C	D	E
1			销售表		
2	编号	地区	上半年销量	下半年销量	
3	114	北部	1600	1300	
4	113	北部	1810	1330	
5	116	东部	1110	1300	
6	118	东部	1124	1375	
7	117	东部	3420	1390	
8	111	南部	1141	2580	
9	110	南部	1180	1385	
10	112	南部	1460	2345	
11	115	西部	1530	1950	
12					

图 3.48 添加新的标题

在添加新的标题后，程序运行后的结果如图 3.49 所示。



	A	B	C	D	E
1					
2	编号	地区	上半年销量	下半年销量	
3	114	北部	1600	1300	
4	113	北部	1810	1330	
5	116	东部	1110	1300	
6	118	东部	1124	1375	
7	117	东部	3420	1390	
8	111	南部	1141	2580	
9	110	南部	1180	1385	
10	112	南部	1460	2345	
11	115	西部	1530	1950	
12					

图 3.49 运行结果

## 6. 程序分析

在上面中的程序代码中，首先使用语句 `ActiveCell.CurrentRegion` 获取当前单元格所在的区域，然后使用 `ListHeaderRows` 属性获取当前区域的标题行数，再根据标题行数计算出选择区域的大小，使用 `Resize` 属性选择出需要的区域。

## 案例 39 获取当前区域的信息

### 1. 功能说明

当前区域在 Excel 数据处理方面是一个很重要的概念，用户在分析当前数据的时候，很多时候是基于当前区域的信息进行处理的。因此，获取和分析当前区域信息是进行数据处理的重要基础内容。

### 2. 语法说明

在 Excel VBA 中，可以使用 `Range` 对象的多个属性值来获取信息。具体信息如下：

- `ActiveCell` 对象：代表活动窗口（当前工作表）的活动单元格。
- `Rows` 属性：代表指定单元格区域中的行，通过其 `Count` 属性可取得行的数量。
- `Columns` 属性：代表指定单元格区域中的列，通过其 `Count` 属性可取得列的数量。
- `ListHeaderRows` 属性：代表指定区域的标题行数。
- `Cells` 属性：代表指定单元格区域中的单元格，通过其 `Count` 属性取得单元格的数目。

### 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区数据。现在需要统计当前区域的信息，原始数据如图 3.50 所示。



	A	B	C	D	E
1					
2		编号	地区	上半年销量	下半年销量
3	114	北部	1600	1300	
4	113	北部	1810	1330	
5	116	东部	1110	1300	
6	118	东部	1124	1375	
7	117	东部	3420	1390	
8	111	南部	1141	2580	
9	110	南部	1180	1385	
10	112	南部	1460	2345	
11	115	西部	1530	1950	
12					

图 3.50 原始数据

#### 4. 编写代码

获取当前区域的信息的具体代码如下：

```
Sub CountCurrentRegion()  
  
    Dim rng As Range  
    Dim DataSource As Worksheet  
  
    Set DataSource = ActiveWorkbook.Worksheets("sheet1")  
    Set rng = DataSource.Range("A1").CurrentRegion  
  
    DataSource.Range("F2") = "当前区域标题行数"  
    DataSource.Range("G2").Value = rng.ListHeaderRows  
    DataSource.Range("F3") = "当前区域的行数"  
    DataSource.Range("G3").Value = rng.Rows.Count  
    DataSource.Range("F4") = "当前区域的列数"  
    DataSource.Range("G4").Value = rng.Columns.Count  
    DataSource.Range("F5").Value = "当前区域的单元格数"  
  
    DataSource.Range("G5").Value = rng.Cells.Count  
    DataSource.Columns("F:F").EntireColumn.AutoFit  
  
    rng.Resize(rng.Rows.Count - rng.ListHeaderRows, rng.Columns.Count).Offset(2, 0).Select  
  
End Sub
```

#### 5. 运行结果

运行程序代码，查看设置的结果，如图 3.51 所示。



图 3.51 程序运行结果

## 6. 程序分析

在本例中，根据原始的数据，统计了关于当前区域的各种信息，在实际开发过程中，用户可能需要分析当前所有数据。

## 案例 40 获取单元格的地址信息

### 1. 功能说明

单元格的地址信息是用户操作和了解单元格区域的一个重要因素。当用户能够获取单元格区域的地址，就可以对地址进行操作。在本小节中，将详细讲解如何获取单元格区域的地址信息。

### 2. 关键技术

在 Excel VBA 中，使用 Range 对象的 Address 属性可返回代表宏语言的区域引用。其语法格式如下：

**表达式.Address(RowAbsolute, ColumnAbsolute, ReferenceStyle, External, RelativeTo)**

其中各参数都可省略，各参数的含义如下：

- **RowAbsolute**：如果为 True，则以绝对引用返回引用的行部分。默认值为 True。
  - **ColumnAbsolute**：如果为 True，则以绝对引用返回引用的列部分。默认值为 True。
  - **ReferenceStyle**：设置为常量 xlA1 返回 A1 样式的引用，常量 xlR1C1 返回 R1C1 样式的引用。
  - **External**：如果为 True，则返回外部引用。如果为 False，则返回本地引用。默认值为 False。
  - **RelativeTo**：如果 RowAbsolute 和 ColumnAbsolute 为 False，并且 ReferenceStyle 为 xlR1C1，则必须包括相对引用的起始点。此参数是定义起始点的 Range 对象。
- 如果引用包含多个单元格，RowAbsolute 和 ColumnAbsolute 将应用于所有的行和列。

### 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区数据。现在需要统计当前区域的地址信息，原始数据如图 3.52 所示。

	A	B	C	D	E	F
1	编号	地区	上半年销量	下半年销量		
2	114	北部	1600	1300		
3	113	北部	1810	1330		
4	116	东部	1110	1300		
5	118	东部	1124	1375		
6	117	东部	3420	1390		
7	111	南部	1141	2580		
8	110	南部	1180	1385		
9	112	南部	1460	2345		
10	115	西部	1530	1950		
11						
12						

图 3.52 原始数据

### 4. 编写代码

获取地址信息的程序代码如下：

```
Sub GetCellAddress()  
  
    Dim rng As Range  
    Dim str As String, strTitle As String  
    Set rng = ActiveCell  
  
    strTitle = "单元格的地址信息"  
    str = "绝对地址: " & rng.Address & vbCrLf  
    str = str & "以 R1C1 形式显示: " & rng.Address(ReferenceStyle:=xlR1C1) & vbCrLf  
    str = str & "相对地址: " & rng.Address(False, False)  
  
    MsgBox prompt:=str, Title:=strTitle  
  
End Sub
```

### 5. 运行结果

运行程序代码，查看设置的结果，如图 3.53 所示。





图 3.53 获取地址信息

## 6. 程序分析

本例代码只是简单演示了绝对地址和相对地址的概念，读者也可以自行设置行或者列的相对地址，这里就不详细展开说明。

## 案例 41 获取单元格的公式信息

### 1. 功能说明

在 Excel 数据处理中，单元格的公式信息是一个特殊的信息。因此，在获取单元格信息的时候，需要对单元格的公式信息进行特定分析。在本小节中，将详细讲解如何获取单元格的公式信息。

### 2. 语法说明

在 Excel 中定义公式时，使用了两个概念：引用单元格和从属单元格。

- 引用单元格：是被其他单元格中的公式引用的单元格。例如，如果单元格 B9 包含公式“=A4”，那么单元格 A4 就是单元格 B9 的引用单元格。
- 从属单元格：中包含引用其他单元格的公式。例如，如果单元格 B9 包含公式“=A4”，那么单元格 B9 就是单元格 A4 的从属单元格。

读者可以使用“追踪引用单元格”和“追踪从属单元格”命令以图形方式显示或追踪这些单元格与包含追踪箭头的公式之间的关系。在 VBA 中，可以使用 Range 对象的以下几个属性来追踪这些单元格：

- DirectPrecedents 属性：该属性返回一个 Range 对象，该对象表示包含一个单元格的所有直接引用单元格的区域。如果有多个引用单元格，这可能有多个选择（多个 Range 对象）。
- Precedents 属性：该属性返回一个 Range 对象，该对象表示单元格的所有引用单元格。如果有多个引用单元格，则可以是一个多重选择（Range 对象的并集）。



- ShowPrecedents 方法：该方法绘制从指定区域指向直接引用单元格的追踪箭头。
- ShowDependents 方法：该方法绘制从指定区域指向直接从属单元格的追踪箭头。

### 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区数据。现在需要获取单元格的公式信息，原始数据如图 3.54 所示。



	A	B	C	D	E	F
1	编号	地区	上半年销量	下半年销量	总和	
2	114	北部	1600	1300	2900	
3	113	北部	1810	1330	3140	
4	116	东部	1110	1300	2410	
5	118	东部	1124	1375	2499	
6	117	东部	3420	1390	4810	
7	111	南部	1141	2580	3721	
8	110	南部	1180	1385	2565	
9	112	南部	1460	2345	3805	
10	115	西部	1530	1950	3480	
11						

图 3.54 原始数据

### 4. 编写代码

(1) 在模块声明部分声明两个逻辑变量。

```
Dim BoolWay1 As Boolean
```

```
Dim BoolWay2 As Boolean
```

(2) 追踪或者取消引用单元格箭头的代码如下。

```
Sub TrackPrecedents()
    BoolWay1 = Not BoolWay1
    If BoolWay1 Then
        ActiveCell.ShowPrecedents
    Else
        ActiveCell.ShowPrecedents Remove:=True
    End If
End Sub
```

(3) 显示或者取消追踪从属单元格箭头的代码如下。

```
Sub TrackDependent()
    BoolWay2 = Not BoolWay2
    If BoolWay2 Then
        ActiveCell.ShowDependents
    Else
        ActiveCell.ShowDependents Remove:=True
    End If
End Sub
```

## 5. 运行结果

选择单元格 E5，选择运行 TrackPrecedents 的过程，结果如图 3.55 所示。



	A	B	C	D	E	F
1	编号	地区	上半年销量	下半年销量	总和	
2	114	北部	1600	1300	2900	
3	113	北部	1810	1330	3140	
4	116	东部	1110	1300	2410	
5	118	东部	1124	1375	2499	
6	117	东部	3420	1390	4810	
7	111	南部	1141	2580	3721	
8	110	南部	1180	1385	2565	
9	112	南部	1460	2345	3805	
10	115	西部	1530	1950	3480	

图 3.55 查看引用单元格的信息

选择单元格 C5，选择运行 TrackDependent 过程，结果如图 3.56 所示。



	A	B	C	D	E	F
1	编号	地区	上半年销量	下半年销量	总和	
2	114	北部	1600	1300	2900	
3	113	北部	1810	1330	3140	
4	116	东部	1110	1300	2410	
5	118	东部	1124	1375	2499	
6	117	东部	3420	1390	4810	
7	111	南部	1141	2580	3721	
8	110	南部	1180	1385	2565	
9	112	南部	1460	2345	3805	
10	115	西部	1530	1950	3480	

图 3.56 查看隶属单元格的信息

## 6. 程序分析

从上面程序分析的结果可以看出，当用户选择对应的代码后，Excel 中会显示出对应的公式追踪和隶属关系的结果。

## 3.3 设置单元格格式

在用户进行 Excel 操作的时候，经常需要设置单元格的各种格式，例如单元格的背景

和表格底纹等。在 Excel VBA 中，用户同样可以代码来设置单元格的格式。在本小节中，将详细讲解如果设置单元格格式的情况。

## 案例 42 设置单元格的属性

### 1. 功能说明

在 Excel 进行数据处理的时候，经常需要对单元格的数据格式进行设置。例如，对于会计数据而言，一般都有明确的数据格式。因此，设置单元格的数据属性是经常需要进行的操作。在本小节中，将详细讲解如何使用 VBA 设置单元格的属性。

### 2. 语法说明

在 Excel VBA 中，设置单元格的属性最常用的方法是使用 Range 对象的 SpecialCells 方法。为了设置单元格中的数字格式，则需要使用 Range 对象的 NumberFormat 属性。在 Excel VBA 中，Range 的 NumberFormat 属性返回 Variant 类型的数值，它代表对象的格式代码。其语法表达式是：

表达式.NumberFormat

其中表达式是 Range 对象的变量。如果指定区域中的单元格中，包含不同的数字格式，则此属性返回 Null。例如，下面代码可以设置单元格的数值格式：

```
Range("B3").NumberFormat = "General"  
Rows(3).NumberFormat = "hh:mm:ss"
```

### 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区数据。现在需要对原始数据中的数值设置对应的格式，原始数据如图 3.57 所示。



	A	B	C	D	E
1	编号	地区	上半年销量	下半年销量	
2	A114	北部	1600	1300	
3	A115	北部	1810	1330	
4	A116	东部	1110	1300	
5	A117	东部	1124	1375	
6	A118	东部	3420	1390	
7	A119	南部	1141	1375	
8	A120	南部	1180	1300	
9	A121	南部	1460	1300	
10	A122	西部	1530	1270	
11					

图 3.57 原始数据

#### 4. 编写代码

设置数据格式属性的代码如下：

```
Sub Cellsformat()  
    Dim rng1 As Range, rng2 As Range  
    Set rng1 = Sheet1.Range("A1").CurrentRegion  
  
    Set rng2 = rng1.SpecialCells(Type:=xlCellTypeConstants, Value:=xlNumbers)  
    With rng2  
        .Font.Italic = True  
        .Font.Bold = True  
        .Font.Name = "Verdana"  
        .HorizontalAlignment = xlCenter  
        .VerticalAlignment = xlCenter  
    End With  
  
    Set rng2 = Nothing  
    Set rng1 = Nothing  
End Sub
```

#### 5. 运行结果

运行上面的程序代码，得到的结果如图 3.58 所示。



	A	B	C	D	E
1	编号	地区	上半年销量	下半年销量	
2	A114	北部	<b><i>1600</i></b>	<b><i>1300</i></b>	
3	A115	北部	<b><i>1810</i></b>	<b><i>1330</i></b>	
4	A116	东部	<b><i>1110</i></b>	<b><i>1300</i></b>	
5	A117	东部	<b><i>1124</i></b>	<b><i>1375</i></b>	
6	A118	东部	<b><i>3420</i></b>	<b><i>1390</i></b>	
7	A119	南部	<b><i>1141</i></b>	<b><i>1375</i></b>	
8	A120	南部	<b><i>1180</i></b>	<b><i>1300</i></b>	
9	A121	南部	<b><i>1460</i></b>	<b><i>1300</i></b>	
10	A122	西部	<b><i>1530</i></b>	<b><i>1270</i></b>	
11					

图 3.58 程序运行结果

#### 6. 程序分析

在上面的代码中，首先使用 `SpecialCells(Type:=xlCellTypeConstants, Value:=xlNumbers)` 代码获取对当前数据区域的数字单元格的引用，然后对其设置相关的属性。

## 案例 43 设置自动套用格式

### 1. 功能说明

在 Excel 中, 为了能够显示数据类型和数据关系, 经常需要套用格式。Excel 提供了多种自动套用的格式。用户同样可以通过 VBA 代码来设置自动套用格式。在本小节中, 将详细讲解如何使用 VBA 设置自动套用格式。

### 2. 语法说明

在 Excel 2003 之前的版本中, Range 对象提供了 AutoFormat 的方法, 使用该方法可对选中区域自动套用格式。在 Excel 2007 版本中该方法被隐藏了, 但仍然可以使用。该方法的语法格式如下:

表达式.AutoFormat(Format, Number, Font, Alignment, Border, Pattern, Width)

该方法各参数都可省略, 各参数的含义如下:

- **Format:** 指定的自动套用格式, 可为设置为 xlRangeAutoFormat 常量之一, 默认常量为 xlRangeAutoFormatClassic1 (古典 1 样式), 该类常量非常多, 使用时可查看帮助中的信息。
- **Number:** 如果该值为 True, 则在自动套用格式中包括数字格式。默认值为 True。
- **Font:** 如果该值为 True, 则在自动套用格式中包括字体格式。默认值为 True。
- **Alignment:** 如果该值为 True, 则在自动套用格式中包括对齐方式。默认值为 True。
- **Border:** 如果该值为 True, 则在自动套用格式中包括边框格式。默认值为 True。
- **Pattern:** 如果该值为 True, 则在自动套用格式中包括图案格式。默认值为 True。
- **Width:** 如果该值为 True, 则在自动套用格式中包括列宽和行高。默认值为 True。

其中, Format 参数的常见数值列表如表 3.1 所示。

表 3.1 Format参数的含义

名称	值	描述
xlRangeAutoFormat3DEffects1	13	三维效果
xlRangeAutoFormat3DEffects2	14	三维效果
xlRangeAutoFormatAccounting1	4	会计1
xlRangeAutoFormatAccounting2	5	会计2
xlRangeAutoFormatAccounting3	6	会计3
xlRangeAutoFormatAccounting4	17	会计4
xlRangeAutoFormatClassic1	1	古典1
xlRangeAutoFormatClassic2	2	古典2
xlRangeAutoFormatClassic3	3	古典3
xlRangeAutoFormatClassicPivotTable	31	传统数据透视表
xlRangeAutoFormatColor1	7	彩色1
xlRangeAutoFormatColor2	8	彩色2
xlRangeAutoFormatColor3	9	彩色3
xlRangeAutoFormatList1	10	列表1
xlRangeAutoFormatList2	11	列表2

xlRangeAutoFormatList3	12	列表3
xlRangeAutoFormatLocalFormat1	15	本地格式1
xlRangeAutoFormatLocalFormat2	16	本地格式2
xlRangeAutoFormatLocalFormat3	19	本地格式3
xlRangeAutoFormatLocalFormat4	20	本地格式4
xlRangeAutoFormatNone	-4142	无指定格式
xlRangeAutoFormatPTNone	42	无指定数据透视表格式
xlRangeAutoFormatReport1	21	报表
xlRangeAutoFormatReport10	30	报表
xlRangeAutoFormatReport2	22	报表
xlRangeAutoFormatReport3	23	报表
xlRangeAutoFormatReport4	24	报表
xlRangeAutoFormatReport5	25	报表
xlRangeAutoFormatReport6	26	报表
xlRangeAutoFormatReport7	27	报表
xlRangeAutoFormatReport8	28	报表
xlRangeAutoFormatReport9	29	报表
xlRangeAutoFormatSimple	-4154	简单

### 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区数据。现在需要对原始数据中的数值设置对应的格式，原始数据如图 3.59 所示。



	A	B	C	D	E
1	编号	地区	上半年销量	下半年销量	
2	A114	北部	1600	1300	
3	A115	北部	1810	1330	
4	A116	东部	1110	1300	
5	A117	东部	1124	1375	
6	A118	东部	3420	1390	
7	A119	南部	1141	1375	
8	A120	南部	1180	1300	
9	A121	南部	1460	1300	
10	A122	西部	1530	1270	
11					
12					
13					

图 3.59 原始数据

### 4. 编写代码

设置自动套用格式的代码如下：

```
Sub autoCellformat()
    Dim rng As Range

    Set rng = Sheet1.Range("A1").CurrentRegion
    rng.autoformat
```

```
Set rng = Nothing
```

```
End Sub
```

## 5. 运行结果

运行上面的程序代码，得到的结果如图 3.60 所示。



	A	B	C	D	E
1	编号	地区	上半年销量	下半年销量	
2	A114	北部	1600	1300	
3	A115	北部	1810	1330	
4	A116	东部	1110	1300	
5	A117	东部	1124	1375	
6	A118	东部	3420	1390	
7	A119	南部	1141	1375	
8	A120	南部	1180	1300	
9	A121	南部	1460	1300	
10	A122	西部	1530	1270	
11					
12					

图 3.60 样式运行结果

## 6. 程序分析

在前面的代码中，所有的参数都使用默认设置数值。如果用户希望了解该方法的具体用处，可以修改参数的数值，来查看格式的变化。

# 案例 44 突显单元格

## 1. 功能说明

当用户在使用 Excel 处理数据的时候，为了能了解所选择的单元格，可以选择突显单元格。突显单元格的基本功能是，当用户选择该单元格的时候，会将该单元格所在的行和列都高亮显示。

## 2. 语法说明

要突出显示工作表当前位置，可以在 Excel 中使用“条件格式”设置相应的公式来完成。本例使用 VBA 代码来完成该功能，该例子中使用了以下几个知识点：SelectionChange 事件和单元格的 Interior 属性。

当用户选定区域发生改变时，将发生 SelectionChange 事件，在该事件过程中编写代码，当发生该事件时就执行该事件过程中的代码。该事件过程的结构如下：

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
```

## 事件执行代码

End Sub

该事件过程中的参数 **Target** 为新获得焦点的单元格引用。本例中，将显示当前单元格所在行和列的底色代码编写在该事件过程中。

通过 **Range** 对象的 **Interior** 属性返回一个 **Interior** 对象，该对象代表指定对象的内部。用该对象的属性 **ColorIndex** 来设置对象内部颜色。颜色可指定为当前调色板中颜色的索引值，也可设置为以下两个常量之一：

- **xlColorIndexAutomatic**：自动配色。
- **xlColorIndexNone**：无色。

## 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区数据。现在需在用户选择单元格区域后，突显单元格的顏色，原始数据如图 3.61 所示。

	A	B	C	D	E
1	编号	地区	上半年销量	下半年销量	
2	A114	北部	1600	1300	
3	A115	北部	1810	1330	
4	A116	东部	1110	1300	
5	A117	东部	1124	1375	
6	A118	东部	3420	1390	
7	A119	南部	1141	1375	
8	A120	南部	1180	1300	
9	A121	南部	1460	1300	
10	A122	西部	1530	1270	

图 3.61 原始数据

## 4. 编写代码

((1) 在 VBE 中的“工程”子窗口中，双击“Sheet1”，打开 Sheet 的代码窗口。然后在对象列表中选择“Worksheet”，如图 3.62 所示。

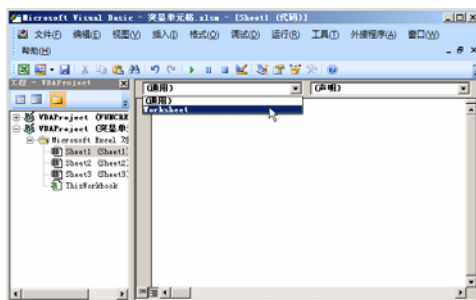


图 3.62 选择对象



(2) 在代码窗口顶端的事件列表中选择事件“SelectionChange”，如图 3.63 所示。

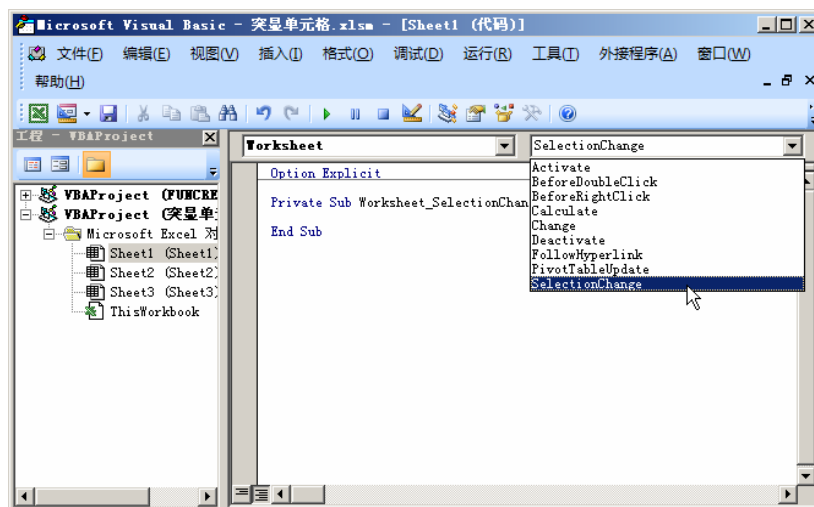


图 3.63 选择事件

(3) 查看自动产生的事件代码结构，如图 3.64 所示。

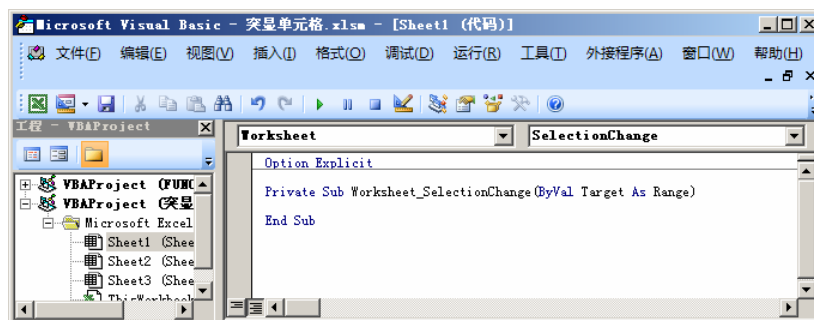


图 3.64 程序代码的结构

(4) 添加的程序代码如下：

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
```

```
    Dim iColor As Integer  
    On Error Resume Next
```

```
    Cells.Interior.ColorIndex = xlColorIndexNone  
    iColor = Int(50 * Rnd() + 3)
```

```
    Target.EntireRow.Interior.ColorIndex = iColor
```

```
Target.EntireColumn.Interior.ColorIndex = iColor
```

```
End Sub
```

## 5. 运行结果

运行上面的程序代码，得到的结果如图 3.65 所示。



	A	B	C	D	E
1	编号	地区	上半年销量	下半年销量	
2	A114	北部	1600	1300	
3	A115	北部	1810	1330	
4	A116	东部	1110	1300	
5	A117	东部	1124	1375	
6	A118	东部	3420	1390	
7	A119	南部	1141	1375	
8	A120	南部	1180	1300	
9	A121	南部	1460	1300	
10	A122	西部	1530	1270	
11					
12					

图 3.65 选择单元格

重新选择一个新单元格，查看程序运行的结果，如图 3.66 所示。



	A	B	C	D	E
1	编号	地区	上半年销量	下半年销量	
2	A114	北部	1600	1300	
3	A115	北部	1810	1330	
4	A116	东部	1110	1300	
5	A117	东部	1124	1375	
6	A118	东部	3420	1390	
7	A119	南部	1141	1375	
8	A120	南部	1180	1300	
9	A121	南部	1460	1300	
10	A122	西部	1530	1270	
11					
12					

图 3.66 选择新的单元格

## 6. 程序分析

在上面的例子中，填充的颜色是通过随机数实现的。其中“ $50 * \text{Rnd}() + 3$ ”表达式中，50 表示的是随机的间隔。如果间隔数太大，会让颜色的随机变化频率减少。

## 案例 45 标记特殊单元格

### 1. 功能说明

在 Excel 处理数据中，对于一些特殊数值的单元格，也许需要进行特殊标记。通过直观的标记，用户可以便利的查看各种数据情况。在本小节中，将详细讲解如何使用 Excel VBA 来标记特殊单元格。

### 2. 语法说明

在本例中，最关键的技术是程序设计的分支结构。在 Excel VBA 中，提供了多种分支结构来处理现实问题。在本例中，需要用到 If ... Then ... Else 这种二路分支结构也可完成，但需要复杂的嵌套结构才能解决该问题。其实 VBA 中提供了一种 If ...Then ... ElseIf 的多分支结构，其语法格式如下：

```
If 逻辑表达式 1 Then  
    语句序列 1  
ElseIf 逻辑表达式 2 Then  
    语句序列 2.  
ElseIf 逻辑表达式 3 Then  
    语句序列 3  
    ... ..  
Else  
    语句序列 n  
End If
```

在以上结构中，可以包括任意数量的 ElseIf 子句和条件，ElseIf 子句总是出现在 Else 子句之前。VBA 首先判断“逻辑表达式 1”的值。如果它为 False，再判断“逻辑表达式 2”的值，以此类推，当找到一个为 True 的条件，就会执行相应的语句块，然后执行 End If 后面的代码。如要所有“逻辑表达式”都为 False，且包含 Else 语句块，则执行 Else 语句块。其流程图如图 3.67 所示。

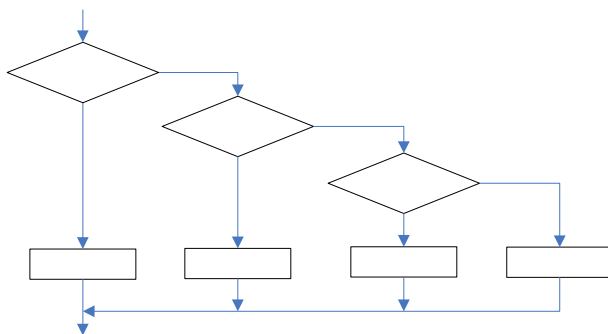


图 3.67 分析结构图

### 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区数据。现在需要标记一些特殊单元格，原始数据如图 3.68 所示。



	A	B	C	D	E	F
1	编号	地区	上半年销量	下半年销量	评级	
2	A114	北部	1600	1300	优秀	
3	A115	北部	1810	1330	优秀	
4	A116	东部	1110	1300	优秀	
5	A117	东部	1124	1375	优秀	
6	A118	东部	3420	1390	优秀	
7	A119	南部	1141	1375	优秀	
8	A120	南部	1180	1300	优秀	
9	A121	南部	1460	1300	优秀	
10	A122	西部	1530	1270	优秀	
11						

图 3.68 原始数据

### 4. 编写代码

标记特殊单元格的代码如下：

```
Sub ColorCells()  
    Dim r As Long  
  
    For r = ActiveSheet.UsedRange.Rows.Count To 1 Step -1  
        If Range("E" & r) = "优秀" Then  
            Range("A:E").Rows(r).Interior.ColorIndex = 7  
        ElseIf Range("E" & r) = "" Then  
            Range("A:E").Rows(r).Interior.ColorIndex = 4  
        End If  
    Next r  
End Sub
```

### 5. 运行结果

运行程序代码，得到的结果如图 3.69 所示。



图 3.69 标记特殊单元格

## 6. 程序分析

在上面的例子中，使用了循环结构，来搜索整个单元格区域。然后根据条件，还设置不同单元格的背景。关于循环结构的知识内容，可以参考前面小节的内容。

## 案例 46 设置单元格的边框

### 1. 功能说明

在 Excel 处理数据的过程中，单元格的边框是一个常用的设置。用户可以使用不同的边框属性来设置单元格属性。同样，用户可以使用 Excel VBA 来设置单元格的边框线。

### 2. 语法说明

设置所选区域的边框可通过 **Borders** 集合来进行设置。通过 **Range** 对象的 **Borders** 属性可返回一个 **Borders** 集合，该集合代表样式或单元格区域（包括定义为条件格式一部分的区域）的边框。

**Borders** 集合由四个 **Border** 对象组成，它们分别代表 **Range** 对象的四个边框：左边框、右边框、顶部边框和底部边框。

使用以下形式可返回单个 **Border** 对象：

**表达式.Borders(index)**

其中参数 **index** 用来指定边框，可分别用以下常量表示不同的边框：

- **xlDiagonalDown**：从区域中每个单元格的左上角至右下角的边框。
- **xlDiagonalUp**：从区域中每个单元格的左下角至右上角的边框。
- **xlEdgeBottom**：区域底部的边框。
- **xlEdgeLeft**：区域左边的边框。
- **xlEdgeRight**：区域右边的边框。
- **xlEdgeTop**：区域顶部的边框。

- **xlInsideHorizontal**: 区域中所有单元格的水平边框（区域以外的边框除外）。
- **xlInsideVertical**: 区域中所有单元格的垂直边框（区域以外的边框除外）。

在 **Borders** 对象的各种属性中，**LineStyle** 是比较常用的属性。如果用户设置 **LineStyle** 为 **XlLineStyle** 数值的话，可以设置边框的线型。其中，**XlLineStyle** 可以选择多个数值，其选项如表 3.2 所示。

表 3.2 XlLineStyle的数值选项

名称	值	描述
<b>xlContinuous</b>	1	实线
<b>xlDash</b>	-4115	虚线
<b>xlDashDot</b>	4	点划相间线
<b>xlDashDotDot</b>	5	划线后跟两个点
<b>xlDot</b>	-4118	点式线
<b>xlDouble</b>	-4119	双线
<b>xlLineStyleNone</b>	-4142	无线条
<b>xlSlantDashDot</b>	13	倾斜的划线

### 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区数据。现在需要对所有的数据添加边框，原始数据如图 3.70 所示。



	A	B	C	D	E
1	编号	地区	上半年销量	下半年销量	
2	A114	北部	1600	1300	
3	A115	北部	1810	1330	
4	A116	东部	1110	1300	
5	A117	东部	1124	1375	
6	A118	东部	3420	1390	
7	A119	南部	1141	1375	
8	A120	南部	1180	1300	
9	A121	南部	1460	1300	
10	A122	西部	1530	1270	

图 3.70 原始数据

### 4. 编写代码

设置边框的程序代码如下：

```
Sub SetBorders()
    Dim rng As Range
    Set rng = Sheet1.Range("A1").CurrentRegion

    rng.Borders.LineStyle = xlContinuous
    rng.Borders.Weight = xlMedium
```

```
Set rng = Nothing
End Sub
```

## 5. 运行结果

运行程序代码，得到的结果如图 3.71 所示。

	A	B	C	D	E
1	编号	地区	上半年销量	下半年销量	
2	A114	北部	1600	1300	
3	A115	北部	1810	1330	
4	A116	东部	1110	1300	
5	A117	东部	1124	1375	
6	A118	东部	3420	1390	
7	A119	南部	1141	1375	
8	A120	南部	1180	1300	
9	A121	南部	1460	1300	
10	A122	西部	1530	1270	
11					

图 3.71 设置单元格边框

## 6. 程序分析

在实际应用中，用户设置单元格的各种边框属性，只需要选用对应的属性名称就可以。这里就不重复详细介绍。

# 案例 47 设置文本的对齐方式

## 1. 功能说明

在 Excel 中，单元格的文本对齐方式也是用户经常需要用到的操作方法。对于不同的对象，使用不同的对齐方式，可以让用户很便利的区隔不同的数据类型。本小节中，将详细讲解如何设置对齐方式。

## 2. 语法说明

使用 VBA 设置单元格区域的对齐方式时，可使用 Range 对象的 HorizontalAlignment 属性和 VerticalAlignment 属性来进行设置。HorizontalAlignment 属性用来设置指定单元格区域的水平对齐方式。此属性的值可设为以下常量之一：

- xlCenter: 居中;
- xlDistributed: 分散对齐;
- xlJustify: 两端对齐;
- xlLeft: 左;

- **xlRight**: 右。

**VerticalAlignment** 属性可以用来设置指定单元格区域的垂直对齐方式。此属性的值可设为以下常量之一：

- **xlVAlignBottom**: 靠下；
- **xlVAlignCenter**: 居中对齐；
- **xlVAlignDistributed**: 分散对齐；
- **xlVAlignJustify**: 两端对齐；
- **xlVAlignTop**: 靠上。

### 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区数据。现在需要设置数据的对齐方式，原始数据如图 3.72 所示。



	A	B	C	D	E
1	编号	地区	上半年销量	下半年销量	
2	A114	北部	1600	1300	
3	A115	北部	1810	1330	
4	A116	东部	1110	1300	
5	A117	东部	1124	1375	
6	A118	东部	3420	1390	
7	A119	南部	1141	1375	
8	A120	南部	1180	1300	
9	A121	南部	1460	1300	
10	A122	西部	1530	1270	
11					
12					

图 3.72 原始数据

### 4. 编写代码

设置对应格式的程序代码如下：

```
Sub SetCellCenter()
    Dim rng As Range
    Set rng = ActiveCell.CurrentRegion

    rng.HorizontalAlignment = xlHAlignCenter

    MsgBox "单元格的文本已经水平居中！"

End Sub
```

### 5. 运行结果

运行程序代码，得到的结果如图 3.73 所示。





图 3.73 运行程序代码的结果

## 6. 程序分析

上面的代码中，只是演示了如何设置单元格的水平格式，用户同样可以设置单元格的垂直对齐方式，这里就不详细介绍。

## 案例 48 设置文本的方向

### 1. 功能说明

在默认情况下，Excel 中的文本都是水平方向的。但是，在用户进行数据分析和处理的时候，经常需要设置文本的方向。例如，在一个数据表中，所有的数据都是正当的水平方向，为了凸现其他文本的内容，需要设置这些文本的方向。

在 Excel VBA 中，用户可以通过 **Orientation** 属性来设置文本的对齐方向，在本小节中，将详细介绍如何通过程序代码设置文本的方向。

### 2. 语法说明

在 Excel VBA 中，使用 **Range** 对象的 **Orientation** 属性，可获取或设置单元格区域文字的方向。其值可从 -90 度到 90 度的一个整数值或为以下常量之一：

- **xlDownward**：文字向下排列；
- **xlHorizontal**：文字水平排列；
- **xlUpward**：文字向上排列；
- **xlVertical**：文字在单元格中向下居中排列。

### 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区数据。现在需要设置员工编号的文本方向，原始数据如图 3.74 所示。

图 3.74 原始数据

#### 4. 编写代码

设置文本方向的 VBA 代码如下：

```
Sub SetOrientation()  
    Dim IntOrient As Integer  
  
    IntOrient = Application.InputBox(prompt:="输入角度（-90~90）：", Type:=1)  
  
    Selection.Orientation = IntOrient  
  
End Sub
```

#### 5. 运行结果

选择单元格 A2~A10，运行程序代码，输入文本的角度，如图 3.75 所示。

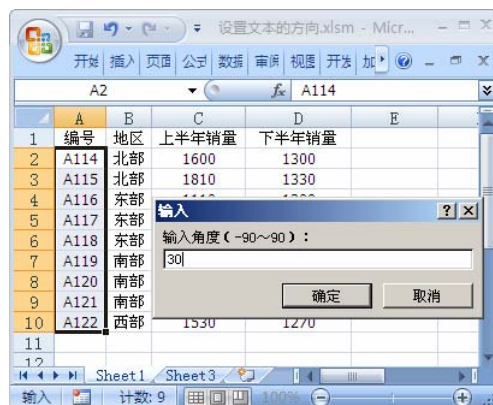


图 3.75 输入文本的角度

单击对话框中的“确定”按钮，查看修改后的结果，如图 3.76 所示。

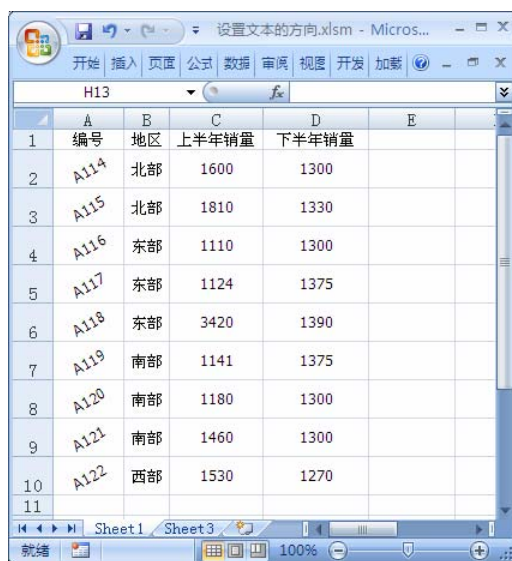


图 3.76 设置文本的角度结果

## 6. 程序分析

用户可以重新运行程序代码，设置不同的方向，查看最后的结果。

## 案例 49 设置条件格式

### 1. 功能说明

条件格式是一种特殊的单元格格式，通过条件格式，用户可以对数据进行快速设置。在 Excel 的基本操作中，用户可以对单元格范围设置各种不同的条件格式。在 Excel VBA 中，用户同样可以自行设置特殊的条件格式。

### 2. 语法说明

在 Excel VBA 中，FormatConditions 对象代表一个区域内所有条件格式的集合。FormatConditions 集合可以包含多个条件格式。每个格式由一个 FormatCondition 对象代表。使用 FormatConditions 对象的 Add 方法可向集合中添加新的条件格式。其语法格式如下：

`表达式.Add(Type, Operator, Formula1, Formula2)`

各参数的含义如下：

- Type: 指定条件格式是基于单元格值还是基于表达式，可使用的常量如表 3.3 所示。
- Operator: 条件格式运算符。如果 Type 为 xlExpression，则忽略 Operator 参数。可使用的常量如表 3.4 所示。
- Formula1: 与条件格式相关联的值或表达式。可为常量值、字符串值、单元格引用或公式。

- **Formula2:** 当参数 Operator 为 xlBetween 或 xlNotBetween 时，它是与条件格式第二部分相关联的值或表达式（否则忽略该参数）。可为常量值、字符串值、单元格引用或公式。

表 3.3 Type可用常量

xlAboveAverageCondition	高于平均值条件	xlIconSet	图标集
xlBlanksCondition	空值条件	xlNoBlanksCondition	无空值条件
xlCellValue	单元格值	xlNoErrorsCondition	无错误条件
xlColorScale	色阶	xlTextString	文本字符串
xlCompareColumns	比较列	xlTimePeriod	时间段
xlDatabar	数据条	xlTop10	前10个值
xlErrorsCondition	错误条件	xlUniqueValues	惟一值
xlExpression	表达式		

表 3.4 Operator可用常量

xlBetween	介于	xlNotBetween	不介于
xlEqual	等于	xlNotEqual	不等于
xlGreater	大于	xlGreaterEqual	大于等于
xlLess	小于	xlLessEqual	小于等于

向单元格区域添加条件格式的代码如下：

```
With 表达式.FormatConditions.Add(参数)
    设置格式代码
End With
```

使用 FormatConditions 集合对象的 Modify 方法可修改现有的条件格式，使用 Delete 方法可在添加新条件格式前删除现有的格式。

注意：对单个区域定义的条件格式不能超过三个。

### 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区数据。现在针对销量设置条件格式：

- 销量大于等于 1500 的，填充粉红色的背景色；
- 销量小于 1200 的，填充蓝色的背景色。

其中，原始数据如图 3.77 所示。



	A	B	C	D
	编号	地区	上半年销量	下半年销量
1	A114	北部	1600	1300
2	A115	北部	1810	1330
3	A116	东部	1110	1300
4	A117	东部	1124	1375
5	A118	东部	3420	1390
6	A119	南部	1141	1375
7	A120	南部	1180	1300
8	A121	南部	1460	1300
9	A122	西部	1530	1270
10				
11				
12				

图 3.77 原始数据

#### 4. 编写代码

设置条件格式的 VBA 代码如下：

```
Sub SetConditions()  
    Dim rng1 As Range  
    Set rng1 = Sheet1.Range("C2:D10")  
  
    With rng1.FormatConditions.Add(Type:=xlCellValue, _  
        Operator:=xlGreaterEqual, Formula1:=1500)  
        With .Interior  
            .ColorIndex = 7  
        End With  
    End With  
  
    With rng1.FormatConditions.Add(Type:=xlCellValue, _  
        Operator:=xlLess, Formula1:=1200)  
        With .Interior  
            .ColorIndex = 4  
        End With  
    End With  
End Sub
```

#### 5. 运行结果

运行程序代码，得到的结果如图 3.78 所示。

	A	B	C	D	E
	编号	地区	上半年销量	下半年销量	
2	A114	北部	1600	1300	
3	A115	北部	1810	1330	
4	A116	东部	1110	1300	
5	A117	东部	1124	1375	
6	A118	东部	3420	1390	
7	A119	南部	1141	1375	
8	A120	南部	1180	1300	
9	A121	南部	1460	1300	
10	A122	西部	1530	1270	

图 3.78 设置条件格式

## 6. 程序分析

在 Excel 中，条件格式是一种特殊的格式。用户不能对已经设置结果的条件格式进行修改。同样，用户可以通过 Excel 操作来查看 Excel VBA 得到的结果。选择“开始”|“样式”|“条件格式”|“管理规则”选项，如图 3.79 所示。

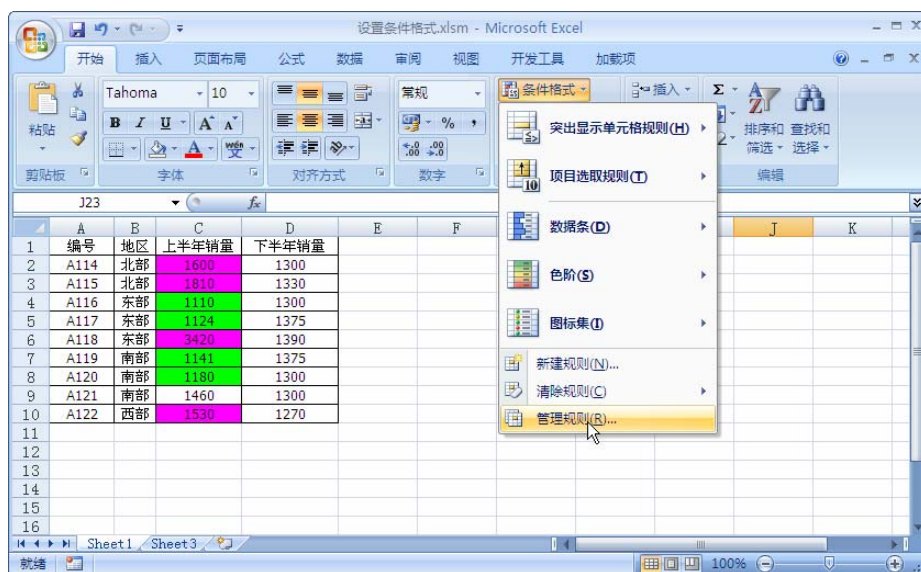


图 3.79 选择管理条件格式的规则

在选择该选项后，打开“条件格式规则管理器”对话框，在选项中选择“当前工作表”选项，如图 3.80 所示。

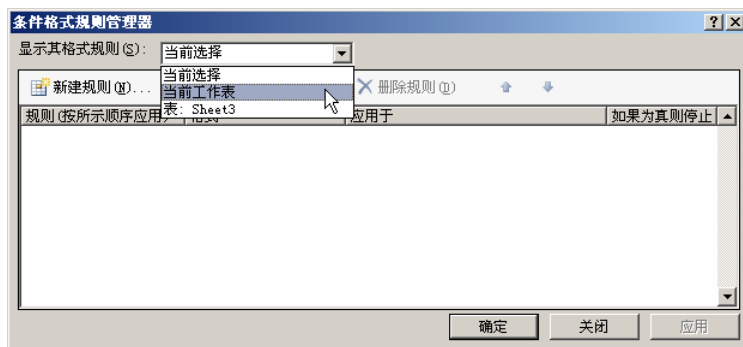


图 3.80 选择当前工作表对象

在对话框中用户可以通过 Excel VBA 设置的条件格式规则，如图 3.81 所示。



图 3.81 查看条件格式的规则

从实际操作的角度来看，用户可以使用 Excel VBA 来设置各种不同的条件格式规则。同时，通过条件格式设置的单元格格式，不能通过普通方式删除。

## 案例 50 合并单元格

### 1. 功能说明

合并单元格在用户对单元格进行操作的时候，经常需要遇到将不同范围的单元格进行合并。用户同样可以通过 Excel VBA 代码，对单元格进行合并。

### 2. 语法说明

在 Excel 中，经常会使用到单元格的合并操作。在 VBA 代码中可使用 Range 对象的方法进行合并（或取消合并）操作。在 Excel VBA 中，Merge 方法合并指定单元格区域，其语法规则如下：

**表达式.Merge(Across)**

参数 Across 若为 True，则将指定区域中每一行的单元格合并为一个单独的合并单元格。默认值是 False，将选中的多行区域合并为一个单元格。合并区域的值在该区域左上角的单元格中指定。

在 Excel VBA 中，用户可以使用 UnMerge 方法将合并区域分解为独立的单元格。其语法表达式如下：

**表达式.UnMerge**

其中表达式是 Range 对象的变量。同时，用户可以使用 MergeCells 属性判断指定区域是否为合并单元格。若指定区域包含合并单元格，该属性返回 True。

### 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区数据。现在需要对该数据表添加标题，原始数据如图 3.82 所示。



	A	B	C	D	E	F
1	销售表					
2	编号	地区	上半年销量	下半年销量		
3	A114	北部	1600	1300		
4	A115	北部	1810	1330		
5	A116	东部	1110	1300		
6	A117	东部	1124	1375		
7	A118	东部	3420	1390		
8	A119	南部	1141	1375		
9	A120	南部	1180	1300		
10	A121	南部	1460	1300		
11	A122	西部	1530	1270		
12						

图 3.82 原始数据表

### 4. 编写代码

合并单元格的 VBA 代码如下：

```
Sub CellsMerge()  
    Dim rng As Range  
  
    Set rng = ActiveSheet.Range("A1:D1")  
  
    With rng  
        .Merge  
        .HorizontalAlignment = xlCenter  
    End With  
End Sub
```

### 5. 运行结果

运行程序代码，得到的结果如图 3.83 所示。





	A	B	C	D	E	F
1	销售表					
2	编号	地区	上半年销量	下半年销量		
3	A114	北部	1600	1300		
4	A115	北部	1810	1330		
5	A116	东部	1110	1300		
6	A117	东部	1124	1375		
7	A118	东部	3420	1390		
8	A119	南部	1141	1375		
9	A120	南部	1180	1300		
10	A121	南部	1460	1300		
11	A122	西部	1530	1270		
12						

图 3.83 合并单元格

## 6. 程序分析

在 Excel VBA 中，如果需要取消合并的单元格，可以使用下面的代码：

```
With Selection
    If .MergeCells Then
        .UnMerge
    end With
```

## 3.4 操作单元格

在前面小节中，已经讲解了多种单元格引用和设置的内容。在 Excel 中，用户除了可以引用和设置单元格之外，还可以对单元格进行各种操作。对此，Excel VBA 提供了各种常见的方法和属性，用户可以使用这些方法来操作单元格。

### 案例 51 复制单元格

#### 1. 功能说明

复制和粘贴是非常常见的操作，当用户需要对数据进行分析的时候，复制单元格区域是基础的操作。在 Excel VBA 中，用户可以使用 Copy 方法来复制单元格，在本小节中，将详细讲解如何使用 VBA 代码复制单元格。

#### 2. 语法说明

对单元格区域中的数据进行复制和粘贴，需分别使用 Copy 和 PasteSpecial 两个方法。使用 Range 对象的 Copy 方法，将单元格区域复制到指定的区域或剪贴板中。其语法格式如下：

**表达式.Copy(Destination)**

参数 Destination 可省略，用来指定要将源区域复制到的目标区域。如果省略此参数，Excel 会将源区域复制到剪贴板中。使用 Range 对象的 PasteSpecial 方法，可将剪贴板中的内容粘贴到指定的区域中。其语法格式如下：

**表达式.PasteSpecial(Paste, Operation, SkipBlanks, Transpose)**

参数 Paste 用来设置要粘贴的区域部分，可设置为以下常量之一：

- xlPasteAll：粘贴全部内容。
- xlPasteAllExceptBorders：粘贴除边框外的全部内容。
- xlPasteAllUsingSourceTheme：使用源主题粘贴全部内容。
- xlPasteColumnWidths：粘贴复制的列宽。
- xlPasteComments：粘贴批注。
- xlPasteFormats：粘贴复制的源格式。
- xlPasteFormulas：粘贴公式。
- xlPasteFormulasAndNumberFormats：粘贴公式和数字格式。
- xlPasteValidation：粘贴有效性。
- xlPasteValues：粘贴值。
- xlPasteValuesAndNumberFormats：粘贴值和数字格式。

参数 Operation 设置具体的粘贴操作，可设置为以下常量之一：

- xlPasteSpecialOperationAdd：复制的数据与目标单元格中的值相加。
- xlPasteSpecialOperationDivide：复制的数据除以目标单元格中的值。
- xlPasteSpecialOperationMultiply：复制的数据乘以目标单元格中的值。
- xlPasteSpecialOperationNone：粘贴操作中不执行任何计算。
- xlPasteSpecialOperationSubtract：复制的数据减去目标单元格中的值。

参数 SkipBlanks 若设置为 True，则不将剪贴板上区域中的空白单元格粘贴到目标区域中。默认值为 False。参数 Transpose 若设置为 True，则在粘贴区域时转置行和列。默认值为 False。

### 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区数据。现在需要复制单元格到其他工作表中，原始数据如图 3.84 所示。



The screenshot shows an Excel window titled '复制单元格.xlsm - Micros...'. The active sheet is 'Sheet1'. The data is as follows:

	A	B	C	D	E
1	编号	地区	上半年销量	下半年销量	
2	A114	北部	1600	1300	
3	A115	北部	1810	1330	
4	A116	东部	1110	1300	
5	A117	东部	1124	1375	
6	A118	东部	3420	1390	
7	A119	南部	1141	1375	
8	A120	南部	1180	1300	
9	A121	南部	1460	1300	
10	A122	西部	1530	1270	

图 3.84 原始数据

#### 4. 编写代码

复制单元格的代码如下：

```
Sub CopyCellNumbers()  
    Dim rng As Range  
  
    Set rng = Sheets(1).Range("A1").CurrentRegion  
    rng.Copy  
  
    Sheets(2).Activate  
    Range("A1").PasteSpecial operation:=xlPasteSpecialOperationNone  
  
    Set rng = Nothing  
End Sub
```

#### 5. 运行结果

运行程序代码，得到的结果如图 3.85 所示。



The screenshot shows the same Excel window, but now the data from the first sheet has been copied to the second sheet (Sheet2). The active sheet is 'Sheet2'. The data is as follows:

	A	B	C	D	E
1	编号	地区	上半年销量	下半年销量	
2	A114	北部	1600	1300	
3	A115	北部	1810	1330	
4	A116	东部	1110	1300	
5	A117	东部	1124	1375	
6	A118	东部	3420	1390	
7	A119	南部	1141	1375	
8	A120	南部	1180	1300	
9	A121	南部	1460	1300	
10	A122	西部	1530	1270	

The status bar at the bottom shows: 平均值: 1461.944444 计数: 40 求和: 26315

图 3.85 复制单元格

## 6. 程序分析

从本例的结果中用户可以发现，Copy 方法将对象的内容复制到剪贴板中。如果用户希望复制内容到目标单元格，则需要使用 PasteSpecial 方法。

## 案例 52 添加公式

### 1. 功能说明

在一般情况下，用户可以在 Excel 表格中直接添加公式和用户自定义的函数。但是，在某些情况下，用户需要在编写代码的过程中，为单元格添加公式。在本小节中，将结合具体的例子来说明如何添加公式。

### 2. 语法说明

在 Excel VBA 中，如果需要添加公式，用户需要使用的 VBA 语法包括：Formula 属性和 FormulaR1C1 属性，以及自动填充 AutoFill 方法。在 Excel VBA 中，可使用 Range 对象的 Formula 和 FormulaR1C1 属性来设置公式。

- Formula 属性：返回或设置单元格的公式，该公式使用 A1 样式表示的公式。
- FormulaR1C1 属性：返回或设置指定对象的公式，该公式使用宏语言 R1C1 格式符号表示。

在很多情况下，填充公式后，需要将公式自动填充到其他单元格区域。因此需要使用 AutoFill 方法。该方法对指定区域中的单元格执行自动填充。其语法表达式是：

表达式.AutoFill(Destination, Type)

表达式是 Range 对象的变量。参数 Destination 是必选参数，表示 Range 要填充的单元格。同时，目标区域必须包括源区域。参数 Type 是可选参数，表示填充类型，其数据类型是 XlAutoFillType。在 Excel 中，XlAutoFillType 包括了常见的几种类型，如表 3.5 所示。

表 3.5 填充类型

名称	值	说明
xlFillCopy	1	将源区域的值和格式复制到目标区域。
xlFillDays	5	将星期中每天的名称从源区域扩展到目标区域中。格式从源区域复制到目标区域。
xlFillDefault	0	默认的填充目标区域的值和格式。
xlFillFormats	3	只将源区域的格式复制到目标区域。
xlFillMonths	7	将月名称从源区域扩展到目标区域中。格式从源区域复制到目标区域。
xlFillSeries	2	将源区域中的值扩展到目标区域中，形式为系列。
xlFillValues	4	只将源区域的值复制到目标区域。
xlFillWeekdays	6	将工作周每天的名称从源区域扩展到目标区域中。
xlFillYears	8	将年从源区域扩展到目标区域中。
xlGrowthTrend	10	将数值从源区域扩展到目标区域中，假定数字之间是乘法关系。
xlLinearTrend	9	将数值从源区域扩展到目标区域中，假定数字之间是加法

		关系。
--	--	-----

### 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区数据。现在需要计算员工全年的销量，原始数据如图 3.86 所示。



	A	B	C	D	E
	编号	地区	上半年销量	下半年销量	总销量
2	A114	北部	1600	1300	
3	A115	北部	1810	1330	
4	A116	东部	1110	1300	
5	A117	东部	1124	1375	
6	A118	东部	3420	1390	
7	A119	南部	1141	1375	
8	A120	南部	1180	1300	
9	A121	南部	1460	1300	
10	A122	西部	1530	1270	

图 3.86 原始数据

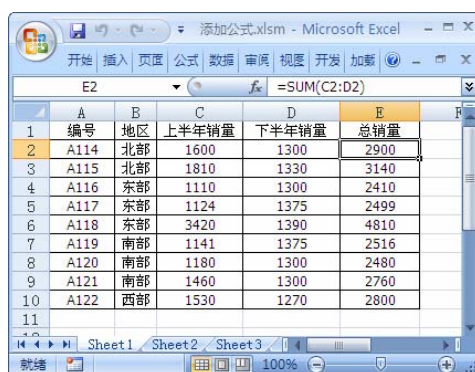
### 4. 编写代码

添加公式的代码如下：

```
Sub AddForlma()
    Range("E2").Select
    ActiveCell.FormulaR1C1 = "=SUM(RC[-2]:RC[-1])"
    Selection.AutoFill Destination:=Range("E2:E10"), Type:=xlFillDefault
End Sub
```

### 5. 运行结果

运行程序代码，得到的结果如图 3.87 所示。



	A	B	C	D	E
	编号	地区	上半年销量	下半年销量	总销量
2	A114	北部	1600	1300	2900
3	A115	北部	1810	1330	3140
4	A116	东部	1110	1300	2410
5	A117	东部	1124	1375	2499
6	A118	东部	3420	1390	4810
7	A119	南部	1141	1375	2516
8	A120	南部	1180	1300	2480
9	A121	南部	1460	1300	2760
10	A122	西部	1530	1270	2800

图 3.87 填充公式

## 6. 程序分析

本例中的程序代码演示的操作过程，实质上就是在单元格中添加公式，然后利用自动填充功能，将公式填充到对应的单元格中。

## 案例 53 设置单元格范围

### 1. 功能说明

在默认情况下，用户可以选择查看整个工作表中的单元格范围。但是，在一些特定情况下，需要显示用户操作单元格的范围。例如，对于普通用户，也许只能阅读一定范围的单元格。在本例中，将介绍如何使用 VBA 代码设置单元格的范围。

### 2. 语法说明

使用 Worksheet 对象的 ScrollArea 属性，可设置允许用户滚动的区域。其参数为一个以 A1 样式的区域引用形式表示的区域。用户不能选定滚动区域之外的单元格。将该属性设置为空字符串 ("")，则允许对整张工作表内所有单元格的选定（相当于取消对滚动区域的限制）。也可通过 ScrollArea 属性获取当前滚动区域的设置值。

### 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区数据。现在需要限制用户查看工作表的单元格范围，原始数据如图 3.88 所示。



	A	B	C	D	E	F
	编号	地区	上半年销量	下半年销量	总销量	
1	A114	北部	1600	1300	2900	
2	A115	北部	1810	1330	3140	
3	A116	东部	1110	1300	2410	
4	A117	东部	1124	1375	2499	
5	A118	东部	3420	1390	4810	
6	A119	南部	1141	1375	2516	
7	A120	南部	1180	1300	2480	
8	A121	南部	1460	1300	2760	
9	A122	西部	1530	1270	2800	
10						
11						

图 3.88 原始数据

### 4. 编写代码

设置单元格范围的具体代码如下：

```
Sub Set_ScrollArea_Cells()
    Dim Rng As Range
    Dim Str As String
```

```

Set Rng = ActiveCell.CurrentRegion
Str = Rng.Address

ActiveSheet.ScrollArea = Str
End Sub

```

### 5. 运行结果

运行程序代码，得到的结果如图 3.89 所示。

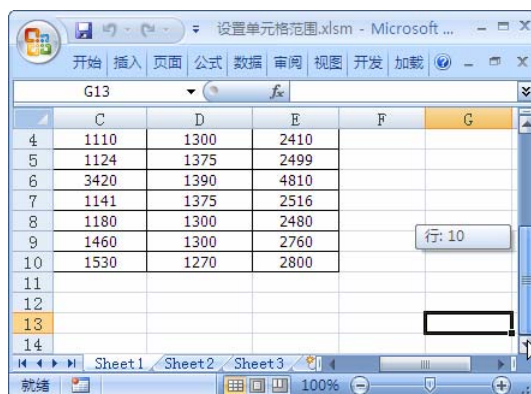


图 3.89 限制单元格的移动范围

### 6. 程序分析

如果用户需要清除单元格的移动范围，可以使用下面的代码：

```

Sub dis_ScrollArea()
    ActiveSheet.ScrollArea = ""
End Sub

```

## 案例 54 添加批注

### 1. 功能说明

在 Office 体系中，标注是一种十分常用的辅助工具。用户可以为所有单元格添加相应的说明和注释。在 Excel 中，用户可以为单元格添加说明性文字的批注。同时，为批注进行对应的编辑工作。

### 2. 语法说明

在 Excel 中，可以通过插入批注来对单元格添加注释。通过 Excel 2007 功能区的相关命令按钮可插入批注，在 VBA 中使用 Range 对象的 AddComment 方法也可以插入批注，其语法格式如下：

表达式.AddComment(Text)

参数 Text 为添加的批注文字。

### 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区数据。现在需要为其总和单元格添加批注，原始数据如图 3.90 所示。



	A	B	C	D	E	F
1	编号	地区	上半年销量	下半年销量	总销量	
2	A114	北部	1600	1300	2900	
3	A115	北部	1810	1330	3140	
4	A116	东部	1110	1300	2410	
5	A117	东部	1124	1375	2499	
6	A118	东部	3420	1390	4810	
7	A119	南部	1141	1375	2516	
8	A120	南部	1180	1300	2480	
9	A121	南部	1460	1300	2760	
10	A122	西部	1530	1270	2800	

图 3.90 原始数据

### 4. 编写代码

添加批注的具体代码如下：

```
Sub AddCellComment()

    Dim rng As Range
    Dim str As String
    Dim strP As String
    Dim strT As String

    strP = "插入批注信息： "
    strT = "插入批注"
    str = Application.InputBox(prompt:=strP, Title:=strT, Type:=2)

    If str <> "" Then
        Set rng = ActiveCell
        rng.AddComment str

        Set rng = Nothing
    End If

End Sub
```



## 5. 运行结果

选择单元格 E2，然后运行程序代码，在对话框中输入批注，如图 3.91 所示。

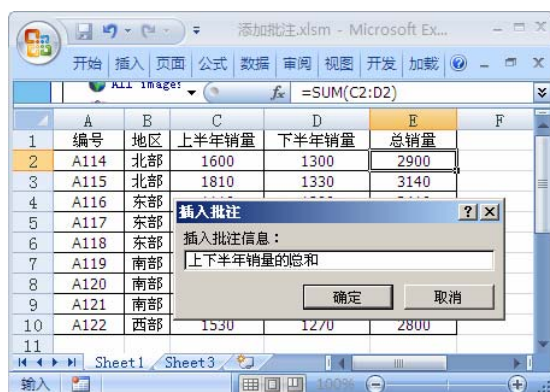


图 3.91 输入批注信息

单击对话框中的“确定”按钮，查看添加的结果，如图 3.92 所示。

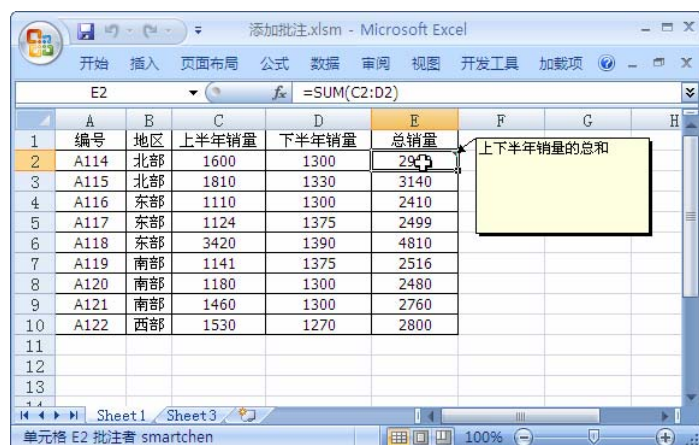


图 3.92 查看添加的批注

## 6. 程序分析

从上面的结果中可以看出，用户可以很便利的通过 Excel VBA 添加文字注释。同样，用户可以通过 VBA 代码便利的编辑文字注释。

## 案例 55 清除单元格的格式

### 1. 功能说明

在 Excel 中，用户经常需要设置单元格的样式。同时，当用户需要修改或者编辑单元格样式的时候，则需要清除单元格的格式。在本小节中，将介绍如何使用 VBA 代码来清除单元格的格式。

### 2. 语法说明

在 Excel VBA 中，用户可以使用 **Range** 对象的多种方法清除单元格，各方法的功能分别如下：

- **ClearContents** 方法：清除单元格区域中的内容公式。
- **ClearFormats** 方法：清除单元格区域中的格式设置。
- **ClearComments** 方法：清除单元格区域中的所有单元格批注。
- **Clear** 方法：清除单元格区域的公式和格式设置。

### 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区数据。现在需要清除该单元格的格式，原始数据如图 3.93 所示。



	A	B	C	D	E
1	编号	地区	上半年销量	下半年销量	总销量
2	A114	北部	1600	1300	2900
3	A115	北部	1810	1330	3140
4	A116	东部	1110	1300	2410
5	A117	东部	1124	1375	2499
6	A118	东部	3420	1390	4810
7	A119	南部	1141	1375	2516
8	A120	南部	1180	1300	2480
9	A121	南部	1460	1300	2760
10	A122	西部	1530	1270	2800
11					

图 3.93 原始数据

### 4. 编写代码

清除格式的 VBA 代码如下：

```
Sub DeleteFormats()  
Range("A1").CurrentRegion.ClearFormats  
End Sub
```

### 5. 运行结果

运行程序代码，查看程序的结果，如图 3.94 所示。



	A	B	C	D	E	F
1	编号	地区	上半年销量	下半年销量	总销量	
2	A114	北部	1600	1300	2900	
3	A115	北部	1810	1330	3140	
4	A116	东部	1110	1300	2410	
5	A117	东部	1124	1375	2499	
6	A118	东部	3420	1390	4810	
7	A119	南部	1141	1375	2516	
8	A120	南部	1180	1300	2480	
9	A121	南部	1460	1300	2760	
10	A122	西部	1530	1270	2800	
11						

图 3.94 查看程序的结果

## 6. 程序分析

从上面的结果可以看出，当用户清除设置的表格格式后，表格中的数据将以默认的样子显示。

## 案例 56 删除单元格

### 1. 功能说明

在 Excel 中，删除单元格是十分常见的操作。用户在进行数据处理的时候，经常需要删除一些错误或者无关的单元格。在 VBA 代码中，用户同样可以实现类似的功能。在本小节中，将详细讲解如何删除单元格。

### 2. 语法说明

在 Excel 环境中，在功能区“开始”选项卡的“单元格”组中，单击“删除单元格”命令按钮，将打开如图 3.95 所示对话框，根据需要选择替换删除单元格的方向。

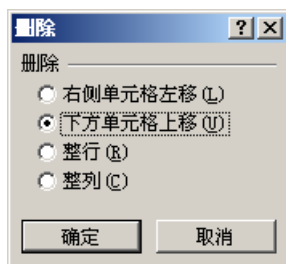


图 3.95 删除单元格

在 Excel VBA 中，使用 Range 对象的 Delete 方法也可完成删除选中单元格区域的功能，

其语法格式如下：

**表达式.Delete(Shift)**

参数 Shift 可省略，用来设置如何调整单元格以替换删除的单元格。如果省略此参数，Excel 将根据区域的形状确定调整方式。该参数可设置为以下常量之一：

- xlShiftToLeft：单元格向左移动替换被删除的单元格。
- xlShiftUp：单元格向上移动替换被删除的单元格。

### 3. 案例说明

某公司统计了部分员工上、下两半年销量，同时提供了员工所处的地区数据。现在需要删除 E 列的数据，原始数据如图 3.96 所示。



	A	B	C	D	E	F
1	编号	地区	上半年销量	下半年销量	总销量	
2	A114	北部	1600	1300	2900	
3	A115	北部	1810	1330	3140	
4	A116	东部	1110	1300	2410	
5	A117	东部	1124	1375	2499	
6	A118	东部	3420	1390	4810	
7	A119	南部	1141	1375	2516	
8	A120	南部	1180	1300	2480	
9	A121	南部	1460	1300	2760	
10	A122	西部	1530	1270	2800	

图 3.96 原始数据

### 4. 编写代码

删除单元格的 VBA 代码如下：

```
Sub DeleteCells()
    Dim rng1 As Range

    Set rng = Range("E1:E10")
    rng.Delete (xlShiftToLeft)

    Set rng = Nothing
End Sub
```

### 5. 运行结果

运行程序代码，查看程序的结果，如图 3.97 所示。



图 3.97 删除单元格

## 6. 程序分析

从上面的结果可以看出，当用户使用 VBA 代码删除单元格时，可以直接选择删除的样式。