INSTITUTE OF ACTUARIAL & QUANTITATIVE STUDIES

Class: BSc

Subject : Application of IT- Basics and Advance Excel

Chapter: Unit 3 Chapter 3

Chapter Name: Debugging

Debugging

- *When you write VBA, or any programming language, you are going to encounter errors in it, or should we call them unintended features? Basically you can't write any substantial amount of code without needing to fix errors and make sure that it works as it's supposed to.*

- *The debugging tools in VBA will not only let you fix problems, they will also allow you to gain a better understanding of code, so can be used to familiarize yourself with code written by someone else.*

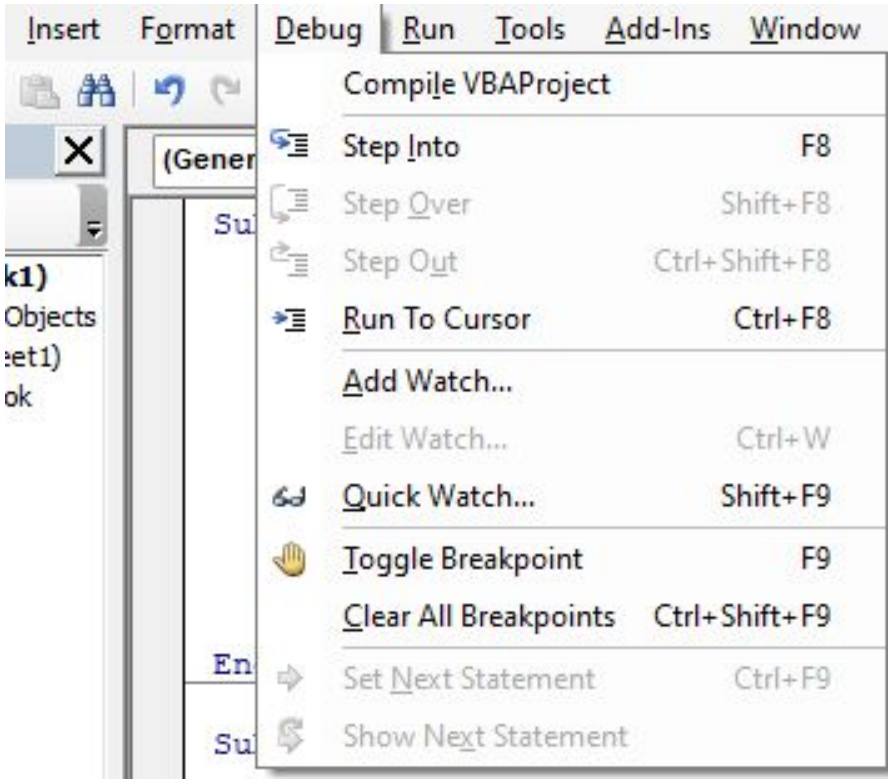
- *To explore the debugging tools provided by VBA, we will use this simple code.*

```
Sub ListNumbers()

    Dim counter As Integer
    Dim myRange As Range
    Set myRange = Range("B11")
    For counter = 0 To 10

        myRange.Offset(counter).Value = counter - 1

    Next counter
    ChangeFont
End Sub
```

```
Sub ChangeFont()

    ActiveCell.Font.Bold = True

End Sub
```

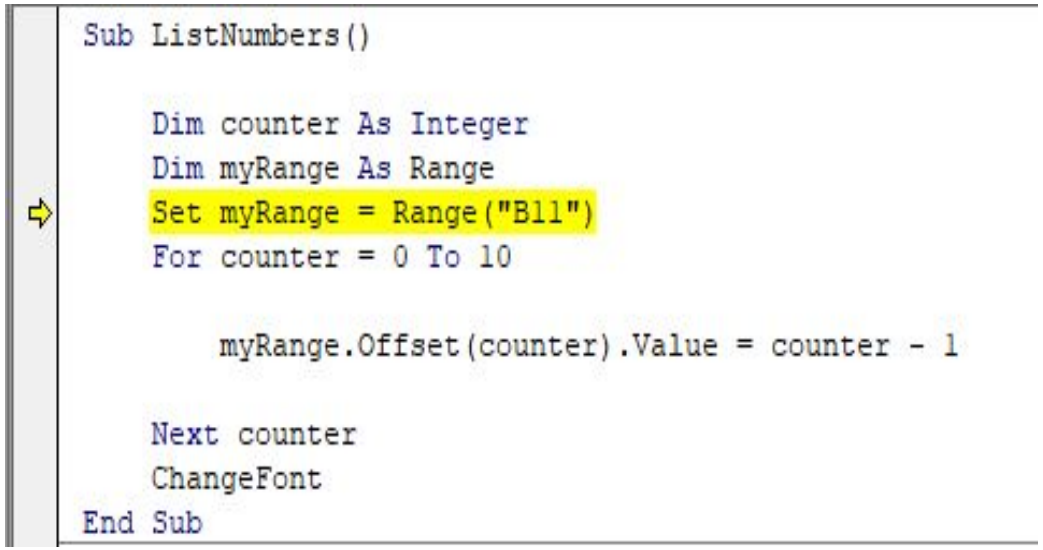
Debugging Tools

- *There are ways the VBA editor allows us to interact and run the code.*
- *If you look at the Debug menu in the VBA editor this is what you will see*
- *Almost all the tools (or commands) have a shortcut key and you should learn these as it's a much quicker way to debug, rather than going to the menu all the time.*



Running Code : F5 & F8

- *When you place the cursor into a sub, either with a mouse or using the keyboard, press F5 to run that sub. It is the easiest way to find out if your code throws an error, however it is not useful for logical errors*
- *With the cursor in the sub, press F8 to execute one line of code at a time.*
- *The next line to be executed will be highlighted in yellow, with a yellow arrow pointing to it.*
- *Each time you press F8, the line highlighted in yellow is executed, and the next line to be run is highlighted.*
- *This is a good way to understand where your code is wrong especially when you have loops in your code.*



```
Sub ListNumbers()

    Dim counter As Integer
    Dim myRange As Range
    Set myRange = Range("B11")
    For counter = 0 To 10

        myRange.Offset(counter).Value = counter - 1

    Next counter
    ChangeFont
End Sub
```

Stepping Over Code : SHIFT + F8

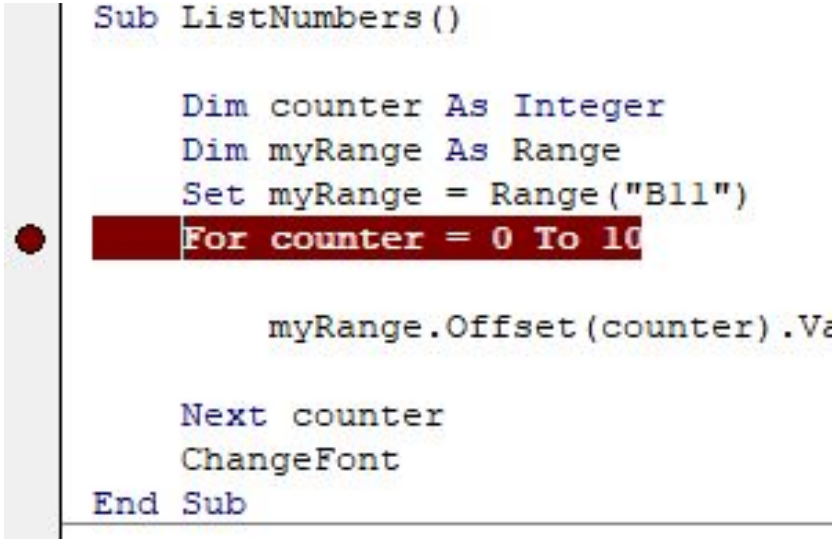
- *If your code calls another sub, you may not want to step through each line of code in that 2nd sub. In this case you can 'step over' the 2nd sub and immediately continue executing the code in your 1st sub.*
- *When code execution gets to the call for ChangeFont, I can press SHIFT + F8, the ChangeFont sub is executed, but we don't have to wait for this, and the code pauses again at the next line which is the End Sub statement.*

```
            myRange.Offset

        Next counter
        ChangeFont
End Sub
```

```
        ChangeFont
End Sub

Sub ChangeFont()
```

Breakpoints : F9

- *A breakpoint is a line in your code where you tell VBA to pause and wait.*
- *To create a breakpoint you can either position the cursor on the desired line and press F9, or just click in the margin beside that line. A dark red/brown dot will appear beside the line in question to indicate a breakpoint has been set, and that line of code is highlighted in the same color.*
- *Clicking on the dot again will remove the breakpoint, as will pressing F9*
- *You can use breakpoints to interrupt the code in a function and check that it is working.*



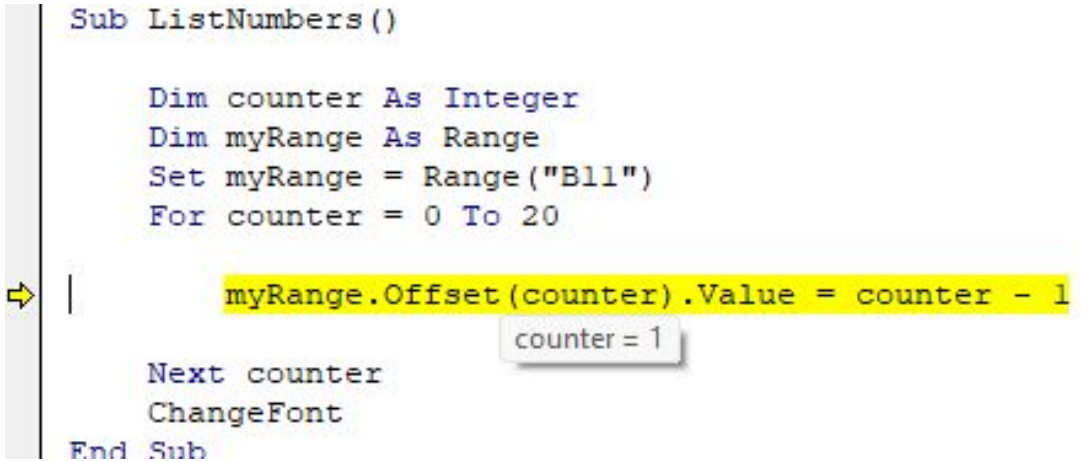
Changing the Next Line to Execute

- *As you are stepping through code, you'll notice that the next line to execute is highlighted in yellow, and has a yellow arrow pointing to it in the margin.*
- *You can use your mouse to drag this arrow to whatever line you want in the same sub (provided it's an executable line), and execution will then continue from there.*
- *This is extremely useful if you want to alter the value of something or fix a bug and then re-execute the code to see if your changes have the desired effect.*

```
Sub ListNumbers()

    Dim counter As Integer
    Dim myRange As Range
    Set myRange = Range("B11")
    For counter = 0 To 20

        myRange.Offset(counter).Value = counter - 1

    Next counter
    ChangeFont
End Sub
```

Checking the Value of Variables

There are a number of ways to do this, the easiest is to just hover your mouse over the variable while your are debugging:

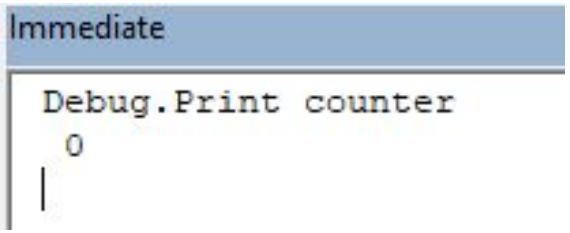


Checking the Value of Variables

Debug.Print

- *Or you can use the Debug.Print statement. This will print the vale of a variable to the Immediate Window. CTRL + G to show the Immediate Window.*
- *Insert the Debug.Print statement directly into your code like so:*
- *You can also type the statement directly into the Immediate window (and hit enter) and get the variable's value*

```
    Dim counter As Integer
    Dim myRange As Range
    Set myRange = Range("B11")
    For counter = 0 To 20

        myRange.Offset(counter).V
        Debug.Print counter
    Next counter
```



Watches

- *You can Watch a variable and its value is shown in the Watch window. If your Watch window isn't showing, turn it on from the View menu in the VBA editor.*
- *To watch a variable, place the cursor in the variable, right click and then click on Add Watch.*
- *If we are just interested in seeing the value as it changes then choose Watch Expression, otherwise choose the appropriate Break option.*