Layout and Formatting – excel project

I want to do my Excel project faster! This is the right place to learn how to do it. Our experts will share their secrets with you.

The layout and appearance of code really does matter and spending a few extra keystrokes ensuring it looks good is well worth the effort. Here's why.

Readability

You and others will read your code far more than you will write it. Even as you write code, you are constantly reading what has just been written or looking at other parts of the codebase. The neater and better formatted the code, the easier it is to read. This saves time, and eliminates frustration.

Comprehension

A key element of reading code is to have visual clues that reinforce the intended structure of the code. The layout of code can infer certain flow and logic, so getting the layout wrong can actually confuse the reader and make them think the code is doing one thing when in fact it is doing another. For example, in the following extract at a cursory glance, the indentation can lead one to assume that the gas is turned off when the boiling point is reached or exceeded, but in fact it's a mistake and the gas is turned off irrespective of the temperature.

```
' ... some code

If temperature < BOILING_POINT Then
    ContinueHeating
Else
    StopHeating
    End If
    TurnOffGas

' ...more code
```

So a key factor when formatting code is to ensure that the layout is closely aligned to the structure and flow of your program.

Consistency

Exactly how one formats code can be a highly contentious issue and people are very passionate about certain styles and conventions. Throughout this article (and the blog as a whole), I do not want to prescribe one style over another. I would however say that consistency is vital. Use indentation correctly to convey meaning and intent, but use the same indentation level throughout. Whether you use 4 spaces or 2 everywhere is far less of an issue than using 4 here, 2 there, and every so often not bothering with any.

If you are adding functionality to existing code that is well formatted, continue in that style, even if it doesn't match your own — consistency is key. If you are maintaining someone else's badly formatted code, apply the principles of the scout rule of leaving the campsite tidier than when you found it and tidy up code as you go.

Whatever style you pick and stick to, make sure it's one that is easy to maintain over time — pick something fairly standard and don't use some ornate layout that takes ages to modify every time you change the functionality slightly. (Complex comment blocks come to mind.)

Some Guidelines

As mentioned above, what follows is just one way of formatting code such that it looks reasonably readable and comprehensible. Additionally, the examples here are focussed on layout and readability, not optimum solutions to problems.

White Space

In computing terms whitespace normally refers to characters such as the space, tab, and new line, that together form space in and around other typed characters. Using whitespace effectively can dramatically improve the readability of code.

White space can be used within a single line to help separate characters such as this.

```
If temp>0 And temp<=100 Then ...
If temp > 0 And temp <= 100 Then ...
```

It can also be used to separate logical segments of code within a function or procedure. Compare the following extracts.

```
Public Sub HeatToThreshold(threshold As Double, isWinter as Boolean)

    Dim heatingTimeSecs As Long
```

```
    heatingTimeSecs = 5
    If isWinter Then
        heatingTimeSecs = 60
    Else


        heatingTimeSecs = 20

    End If
    Dim temperature As Double

    temperature = GetTemperature()
    Do While temperature < threshold

        HeatForDuration heatingTimeSecs


        temperature = GetTemperature()
    Loop

End Sub
```

and hopefully prefer this

```
Public Sub HeatToThreshold(threshold As Double, isWinter As Boolean)

    Dim heatingTimeSecs As Long
    If isWinter Then
        heatingTimeSecs = 60
    Else
        heatingTimeSecs = 20
    End If

    Dim temperature As Double
    temperature = GetTemperature()

    Do While temperature < threshold
        HeatForDuration heatingTimeSecs
        temperature = GetTemperature()
    Loop

End Sub
```

Hopefully, the second one is clearer, as the white space has been used to separate segments of code: the initialisation from the loop.

I also like to ensure there are two blank lines between functions or procedures to make it easier to spot as you scroll up and down, and that the body of a function or routine is indented.

```
Public Sub HeatToThreshold(threshold As Double, isWinter As Boolean)
    ' ... code here
End Sub


Public Sub HeatForDuration
    ' ... code here
End Sub
```

Multiple Statements on a Line

Generally I can't see many (any?) reasons to put multiple statements on a single line — it rarely in my opinion adds anything to readability. A common occurrence of this though is with variable declarations such as below, but I still don't like it and a common derivate of this is often error prone. (See Declaring Variables and the Variant).

```
' really really avoid this (it's not what you think)
Dim temperature, threshold as Integer

' avoid this too if you can
Dim temperature As Integer, threshold As Integer

'use this instead
Dim temperature As Integer
Dim threshold as Integer
```

Line Length and Splitting Lines

In days gone by, when screen sizes were fixed to 80 characters across, it made sense to follow rules to keep your code limited in length. Now most people have wide screens and small fonts and you can get a lot of code on one line. However, there are a few things to remember. Firstly, scrolling horizontally to read code is more awkward than scrolling vertically, so it makes sense to limit line length. Secondly, remember that just because you may be working on a 27 inch monitor, the next person to maintain your code may not. These days I think aiming for about 90-100 characters is not unreasonable.

I don’t think there should be a hard and fast rule. Common sense should prevail and there will be occasions where code is more readable if the line is slightly longer than splitting it over two lines such for the sake of adhering to a standard. Overall though, for a given project there should be line length you aim for and stick to.

If you do need to splt a line in VBA, you can do so by adding a space followed by and underscore _ character at the end of a line. Quite a common place where lines need to be split is with complicated boolean expressions. If this is the case try and find locgical places to split the line that also help indicate to the reader more is to follow.

```
If (temperature > 0 And hourOfDay > 0) Or (temperature < 100 And _
      hourOfDay < 12) Then
  ' Do something
```

is difficult to read when compared with something like the following

```
If (temperature > 0 And hourOfDay > 0) Or _
     (temperature < 100 And hourOfDay < 12) Then
     ' Do something
```

Having the hanging Or at the end of line indicates that there is more to follow since it requires another part of the expression, and hence a better place to split the line, as the two parts of the Or expression are on each line.