Public, Private, Global and Dim - Excel Homework Help

Is doing Excel homework too stressful for you? Let our Excel homework help experts solve your problems.

I wanted to follow on from the last post with some information on how variables behave that are declared at the top of a module. If you are not sure what a module is, I'll explain that too. I'll cover Forms and Classes at the end, but most of the concepts will translate across.

Modules

Modules are created in VBA from the Insert menu in the VBA Editor and then appear in the VBA Project, below where you see the worksheet objects and workbook. The purpose behind a module is to create a place where you can put logically related code. My default, modules are given the name Module1, Module2, but you really should rename them to something more meaningful.

So, at the top of a module, you can write the following statements. For now, I am going to ignore naming conventions.

```
' This is moduleTest

' Declared at the top of testModule
Dim variableDim As String
Private variablePrivate As String
Public variablePublic As String
Global variableGlobal As String

Sub TestSubOne()
    Dim localVariable As String
    localVariable = "I'm local"

    ' inside this sub, it has access to localVariable, and all of the
    ' variables defined at the top

    variablePrivate = "I'm private"

End Sub

Sub TestSubTwo()
    Dim localVariable As String
```

```
    Debug.Print localVariable
    Debug.Print variablePrivate

End Sub
```

If we ran TestSubOne followed by TestSubTwo, you would set localVariable to “I’m local”, and privateVariable to “I’m private” in the first routine. When the second one is called, the new localVariable is just an empty string - the previous one is out of scope and can’t be seen; it effectively dies at the end of the TestSubOne. Local variables are only visible in the subroutine/function in which they were declared.
However, variablePrivate can be seen and modified in any sub/function in the same module. So now, you need to be very careful, as modifying a variable defined at the top of the module in one place, will have an impact right across the module. This creates a source of possible bugs, or just complexity as it gets harder to track down what is modifying a variable. This is one reason I feel variables defined at the top of a module should be named differently (with a prefix such as m_). Hence if you have a subroutine like the one below, it’s obvious to the reader that m_spend is in use elsewhere in the module (for read or modificaiton).

```
Sub TestSubThree()
    m_spend = 123
    cost = 456
End Sub
```

The same thing applies to all the variables declared at the top of the module. However, the difference comes when these variables are accessed from another module.

Variables declared as Private can only be seen inside the module they are declared in. They are private to the module. Variables declared with just Dim at the top are also private. Therefore I would always use Private as it is much clearer to the reader what the intention is.

You can also declare a variable as Public at the top of a module - this too is accessible throughout the module, but is also accesible from outside the module. So as an example let’s suppose a module is called Spending and at the top of it we have Public m_totalSpend As String. This is available to all sub/functions in the module, but can also be accessed from everywhere else in the application by using the module name followed by the variable name separated by a dot e.g. Spending.m_totalSpend. In general, the use of public definitions should be minimised, as it increase the chance of modification in

multiple places that become difficult to track down. In a future post I will talk more about the benefits and techniques of information hiding.

It's also possible to declare variables Global which for all intents and purposes is the same as Public, except Global doesn't work in Classes. My recommendation is don't use Global - I'm guessing it's been left in VBA for backward compatibility from VB6 days.

Functions and Subs

You should also apply the same rules to functions and subs, so declare them as Public if they need to be called from outside the module and Private if they are only used within the module.

```
Private Sub RoutineLocalToModule()
    ' some code that only needs to be called from within the module
End Sub

Public Sub RoutineAccessibleOutsideModule()
    ' some code that needs to be called from outside the module
End Sub
```

Classes and Forms

All of the rules above for Modules also apply to Forms and Classes. The use of Private and Public are typically found much more in Classes, as in many other programming languages you have to define variables and methods as public or private.

Summary

Use Dim for local variables (those within a function or sub). Anything declared at the top of a module/form/class should be declared with Public or Private and not Dim. All functions and subs should likewise be declared with Public or Private.