VBA and Macro creation (using Excel)

DSC340
Mike Pangburn

Agenda for Today

Object-Oriented Programming
Creating Macros with VBA

OBJECT ORIENTED PROGRAMMING: VBA

What is O-O programming?

- A programming style that uses “objects” to comprise programs.
- Objects:
  - In a pure O-O language, every thing in the program is an object.
  - An object is a data structure consisting of attributes (data fields) and methods (actions).
  - The attributes of the object hold its current information
  - The methods of the object determine what the object can do

Different objects types: classes

- Each programming environment comes with many different kinds of predefined objects.
- An object type is called a class. A class is thus some type of object.
- All objects within a class hold the same kind of information (identical attributes) and can perform the same actions (identical methods).
- A class is the “cookie cutter”/template you use to create new object instances.

Hollywood analogy: in Battlestar Galactica there are 12 “classes” of human robots. One of these 12 model types was the Sharon model.

Creating and using objects

- You 1st need a class.

  class Sharon
  Attributes: height, 3Dbodyshape, faceimage, currentlocation, currentlove, pastexperienceslog
  Methods: imitateHuman(), fallInLove(person), doSecretMission(password)

- You then “declare” a new object and can subsequently assign its attribute value and invoke its methods.

- Example: declaring a new Sharon object, assigning a value, and calling a method.

  Sharon sharonNo55
  sharonNo55.currentlocation = “KA.493.X7.1034”
  sharonNo55.fallInLove(Heike)
Object Oriented Programming Fundamentals

Appropriate object classes (their attributes and methods) heavily depend on the problem you are working on.

Example: A system for a bank:
- Objects: Customers, Accounts, etc.
- Attributes: First Name, Last Name, SSN, Address, etc.
- Methods: Withdraw, Open an account, Deposit, Cash check, etc.

Coding with VBA

There are two approaches to write a code in VBA:
- Standard Coding
- Macro Recording

The Macro approach “records” a series of mouse-clicks and keyboard strokes.
- Advantage: Corresponding VBA code is created automatically and recorded for you.
- Disadvantage: It’s limited.

To tackle this limitation, we use a mixed approach:
- First, record a simple macro.
- Then, tweak it to achieve more complex tasks.

Problem Definition

We have data spanning 3 columns and 13 rows (C6:F18).
- Unfortunately, there are some errors in the data entry process, so some of the rows are shifted to the right by one column.

Our job is to correct all these mistakes:
- First, record a simple macro that corrects a specific row (say Row 8).
- Then, tweak it to make corrections in any given row.
- Finally, let the code check any mistake and correct it.

While doing this, we learn how to:
- Declare and name variables.
- Create Subroutines and Functions. (We’ll learn what they are shortly).
- Use conditional statements.
- Create repetitions.

VBA: Visual Basic for Applications

VBA is a specialized version of Microsoft’s well-known Visual Basic language.

VBA is an Object-Oriented language within Microsoft Office suite:
- Excel, Word, Access, and PowerPoint
- We will focus on Excel.
- Shortcut to access the VBA interface in Excel: Alt+F11.

Excel treats everything in a spreadsheet as objects, so VBA lets us play with all these objects.
- Object that we are interested in: Workbooks, workbooks, cells, rows, ranges.
- There are over 200 different class (types of objects) in Excel.

Our First Macro

Download and open VBA_Example_class.xlsm
- In Excel 2010, we should save the file as a Macro Enabled Workbook (i.e., in the .xlsm format).
Our First Macro

Find the Macros option under the Excel 2010 View tab to record a VBA macro. Let’s call our Macro “Shifter.” We want our macro to do the following:
- Select the range D8:F8.
- Cut the selected cells.
- Select cell C8.
- Paste the cut cells.

To see the VBA code generated by your actions, go into VBA (Alt+F11) and then near the upper left of the window expand the Modules folder and double-click on Module1.

Execution of Subroutines

Each line of a subroutine is executed sequentially. Let’s see how one macro is executed:
(The first three green lines following each command are simply comments/documentation.)
- The first line, Range(“D8:F8”).Select, selects the cells we want to select.
- The second line, Selection.Cut, cuts the selected cells.
- The third line, Range(“C8”).Select, selects the cell C8.
- Finally, the last line, ActiveSheet.Paste, pastes the cut values to the selected cell of the worksheet.

Subroutines

Notice that the code starts with Sub command and ends with End Sub command. Sub refers to the Subroutines:
- Subroutine: A portion of code within a larger program that performs a specific task and is relatively independent of the remaining code.
- We can use a subroutine in a different subroutine by “calling” it.
  - e.g., Call Shifter()

Subroutines with Input Arguments

The limitation of Shifter is that it only corrects Row 8. We can solve this by creating a subroutine which will take a row number as an input parameter.

Shifting Any Given Column

How do we tell VBA to shift the row according to our input, RowNum?
- Currently, we select cells D8:F8 by writing “D8:F8.”
- We will construct that D8:F8 string for our row i (i.e., RowNum)
  (The argument & operator simply combines (“concatenates”) text together)

Shifting Repeatedly

We now have a subroutine that can correct any given row. We want to apply this subroutine to any rows between 6 and 18. We use a loop (e.g., a FOR-NEXT Loop) for this task:
Shifting Repeatedly

So, we add a line before the loop where we declare our variable RowNum.

A Function to Check the First Column

We could write our conditional statement to check the first column within the subroutine, but we will define a separate "Function" for the checking task.

Functions are very similar to the subroutines. Then, why a function?
- Unlike a subroutine, a Function returns a value.
- Useful for reporting the result of a calculation or other results.
- Our function will return:
  1 if the first column is empty
  0 otherwise.

Variable Declaration

The subroutine ShiftOneColumn requires an integer input.

We decided to introduce something called RowNum which stores a row number.
- We want the data type for RowNum to be Integer. The VBA interpreter will attempt to guess the data type if you forget to declare your variables.

To avoid that problem, we need to define the variables we use properly:

Some rules about variable declaration:
- Variables names must start with a letter, and have no spaces.
- VBA is not case sensitive (RowNum and rownum are the same).
- Names that are more than one word are usually written with the first letter of words capitalized, it’s not required but it’s fairly standard practice.

Checking If First Column is Empty

We need to check whether the first column of a row is empty or not.

VBA’s conditional statement IF-THEN-ELSE allows us to achieve this goal.

More on Functions vs. Subroutines

Functions are not always the best choice:
- A function cannot directly write data back to the spreadsheet.
  - We can write data to the spreadsheet via subroutines. Easiest way:

```vba
Sub ShiftOneColumn(RowNum As Integer)
  Dim RowNum As Integer
  For RowNum To 18 Step 1
    Call ShiftOneColumn(RowNum)
  Next RowNum
End Sub
```
- e.g. Cells(1,3).Value = "Ducks Rock" writes Ducks Rock to the cell C1.
  - In this example, our function doesn’t need to change the value of cell, but our function does return a value.
A Function to Check the First Column

Let’s go back to our task: Creating a function to check the first column.
We name our function as CheckColOne.
How do we check if the first column is empty?
- We can use Cells(RowNum,3).Value. Then check whether it is empty.
- We know the row number! It’s our input argument, RowNum.
- The column number is 3 since the first column of data is in Column C.
- So, our test condition is if Cells(RowNum,3).Value = "".

Let’s go back to our task: Creating a function to check the first column.
We name our function as CheckColOne.
What is our input argument? Row Number.

Function to check the first column
Function CheckColOne(RowNum as Integer)
If Cells(RowNum,3).Value = "" Then
CheckColOne = 1
Else
CheckColOne = 0
End If
End Function

How do we check if the first column is empty?
- We can use Cells(RowNum,3).Value. Then check whether it is empty.
- We know the row number! It’s our input argument, RowNum.
- The column number is 3 since the first column of data is in Column C.
- So, our test condition is if Cells(RowNum,3).Value = "".

Using the Check Function in the Main Subroutine

Sub ShifterLoop()
    Dim RowNum As Integer
    Dim RowFlag As Integer
    For RowNum = 6 To 18 Step 1
        If CheckColOne(RowNum) Then
            Call ShiftOneColumn(RowNum)
        End If
    Next RowNum
End Sub

Function CheckColOne(RowNum As Integer)
    If Cells(RowNum,3).Value = "" Then
        CheckColOne = 1
    Else
        CheckColOne = 0
    End If
End Function

Practice: Extending our program

- How would we extend our program if we wanted to have the program highlight each moved row with a yellow background?
- Remember the approach most business people using VBA take:
  - Perform “Macro Recording” to start
  - Use/modify the resulting code