# **VBA 2 – Cleaning Up a File with VBA**

Reformatting, changing text to numbers, and adding formulas is a common Excel procedure when you have to import data from another source. This session shows efficient ways to automate this problem. Using the SELECT, RANGE, and OFFSET commands, you'll learn the difference between selecting a cell and referencing it. The session also covers converting dates, numbers, and text and then reformatting them as needed.

### **Review: Introduction to VBA**

- 1. The key to understanding macros and programming languages is action:
  - a. If you want to set a value based on a test, use an Excel function =IF(A1=4,27,0)
  - b. If you want to perform an action, such as formatting, deleting, changing cells, printing, opening/closing, etc. you need a macro.
- 2. Macros are created using a programming language. They are programs of a sort.
- 3. VBA is a programming language based on Visual Basic.
- 4. VB is a standalone programming language. You can create programs that run as EXE files.
- 5. VBA is tied to Microsoft Office. There is a version of VBA for Word, Excel, Access, PowerPoint, and Outlook. They use the application's "object libraries" which contain definitions of the application's components
- 6. VBA combines programming actions with the components of the selected application. This allows you to control the elements of the application using programming code.
  - a. You can automate tasks, create forms, and more using VBA.
  - b. VBA code is always tied to the application, either in an "application-wide" file (Personal Macro Workbook or NORMAL.DOTM) or the specific file.
- 7. Like most programming languages, VBA has the following:
  - a. Variables containers for specific values. Usually "typed" i.e., limited to a specific type of data (integer, string, etc.)
  - b. Statements—combinations of keywords (programming commands) and specifiers that perform some operation.
  - c. Flow of Control—versions of the IF THEN ELSE statement that lets the program execute different instructions based on conditions (variable = value, blank cell, etc.)
  - d. Loops—commands that let the program repeat an action or series of actions until some condition is reached.

```
WHILE ActiveCell <> ""
[code]
WEND
```

- e. Forms—dialog boxes that you can customize with text, buttons, dropdown lists, etc.
- 8. Modern "object-oriented" programming languages use objects.
  - a. A variable has a value only.
  - b. An object has properties and methods.
  - c. Properties include a value, formatting, etc.
  - d. Methods are actions that can affect the object (Close, Open, Print, Select, Delete, etc.)
  - e. VBA is object oriented. Don't get hung upon whether something is a property or a method. Just know that the thing does what you want it to.

#### **EXERCISE: Run Excel.**

1. Open VBA2 Exercises

## **Working with Cells and Ranges**

- 1. In Excel, the basic object you work with is the cell.
- 2. A cell can have a value plus formatting (properties).
- 3. You can also Clear, Delete, Copy, Move, etc. a cell (methods).
- 4. Typically, in a macro you want to be able to:
  - a. Select a cell
  - b. Read the value in the cell
  - c. Change the value in the cell
  - d. Format the contents of the cell

### **Selecting a Cell**

- 1. The VBA equivalent of clicking on a cell is Range. Select.
- 2. You need to specify the cell name (A1, BF27) or a range name (in quotes).
- 3. Selecting a cell makes it the "active" cell.
- 4. ActiveCell is a keyword that always refers to the selected cell, whether or not it is visible. ActiveSheet is the corresponding keyword for the current sheet.

## **Step Into**

- 1. The Step Into icon lets you run a macro step by step.
- 2. Useful for seeing exactly what your code is doing.
- 3. Click anywhere inside the macro.
- 4. Click Step Into.
- 5. The SUB MacroName () line is highlighted in yellow.
- 6. Each time you click Step Into, the current line is executed and the yellow line moves one line.
- 7. Use Run to run the rest of the macro without intervention.
- 8. Use Stop to exit out of the macro and stop running it.

#### **EXERCISE: Selecting a Cell**

- 1. Record the macro SelectCell as an Absolute Addressing Macro.
- 2. Select B4.
- 3. Stop Recording
- 4. Open the VBA Editor.
- 5. Examine the code.
- 6. Add the following lines of code:

```
Range("E7").Select
Range("D11").Select
```

- 7. Run the code. Can't see each selection, only the last D11.
- 8. Arrange Windows with Excel on left, VBA Editor on right.
- Select SelectCell.
- 10. Use the Step Into icon to step through each line of code.

# Referencing a Cell

- 1. In Excel, you select a cell, then view or modify it.
- 2. In VBA, you can view or modify a cell without selecting it.
- 3. This means that you don't have to move from the ActiveCell in order to work with another cell.
- 4. The contents of a cell are a property, usally called either .VALUE or .FORMULAR1C1. (There are others, but these are the most common.)
- 5. Typically, you assign this to a variable to read the value of a cell or assign the cell the value of an existing variable.

```
VariableName = Range(CellRef).Value
Range(CellRef).Value = 23
```

### **EXERCISE: Using the ActiveCell**

1. Add the following lines to your SelectCell macro:

```
Range("E7").Select
ActiveCell.Value = "Added to cell E7"
Range("D11").Select
ActiveCell.Value = 22 + 11
```

2. Notice the guotes around the text and no guotes and no guotes around the formula.

### **Variables**

- 1. A variable is really a memory location that holds a value the way a cell holds a value.
- 2. In Excel, a variable is an object, which makes it easy to work with. There are commands for each of the operations you want to do to a variable.
- 3. Variables have a type, which is the kind of data the variable can hold. For example:
  - a. An Integer can hold any whole number from -32,365 to + 32,364.
  - b. A Double (Double Precision Floating Point) can hold a decimal value from -4 billion to +4 billion.
  - c. A String can hold 255 characters.
  - d. A Date can hold a date from 1/1/100 to 12/31/9999.

- e. If you do not define a specific variable, VBA uses the type Variant, which can hold any type of value.
- f. There are additional variables for longer integers, currency, media, Boolean (Yes/No) values, and shorter decimal numbers.
- 4. Dim VariableName As Type is used to define a variable in a subroutine.
- Option Specific at the top of a module makes variable declaration mandatory for that module
- 6. Use Tools | Options | Editor | Require Variable Declaration to make declaration mandatory. Recommended.
- 7. You can do math or text processing on a variable.

```
TotalCost = SalesPrice + SalesPrice*.025
FullName = FirstName & " " & LastName
```

8. Use assignment to set and read the value of a variable. Note that strings need to be in quotes.

```
Range("B2).Value = "Sale"
SalesPrice = Range("B3).Value
Range("B4").Value = SalesPrice* .0625
```

#### **EXERCISE: Using Variables**

- 1. Open VBA 2 Exercises if it is not open.
- 2. Select the Variables sheet.
- 3. Put a number in B2. (\$1,234.56)
- 4. Create a macro that will read B2, add sales tax to B3, add Total Sale to B4.
- 5. Record a second macro that formats B2:B4 as Currency and add this formatting to your macro.
- 6. Test your macro by formatting B2:B4 as General, deleting B3 and B4 and running again.

### **Variable Names**

- 1. It is best to have a convention for naming variables.
- 2. Remember that MixedCase is more readable than lowercase or UPPERCASE.
- 3. There is a convention for naming variables that uses initial letters to identify the type of data the variable can hold. Note that this does not define the variable as of a certain type. It only helps the programmer understand what type of data the author intended.
- You can use your own names or use single letter variables, but it is very helpful to use the standard naming conventions.
- 5. The full set of variable types and suggested names is listed below:

Data Type	Description	Prefix
Integer	Whole numbers from -32768 to 32767	int
Long	Whole numbers from -2,147,483,648 to 2,147,483,647	Ing
String	Set of characters with a maximum of 65,536 characters.	str
Single	Single Precision number. (Think decimal value.)	sng
Double	Double Precision number. (Think decimal value.)	dbl
Currency	Single Precision number formatted as currency to four decimal places. It accepts \$ and , as well as numbers.	cur
Date	Valid date from Jan 1, 0001 thru Dec 31, 9999	dte
Boolean	Variable holds either TRUE or FALSE, represented as 0 (TRUE) or 1 (FALSE)	bol
Object	Holds a non-alphanumeric object such as a picture, a sound file, a video file, a file path, etc.	obj
Variant	Large enough to hold any data type. Default if DIMAS not used.	var

# **Changing Formatting**

- 1. In addition to setting a cell's value, you can change its formatting.
- 2. You can change any of the values on the Format Cells dialog by using the appropriate property.
- 3. Record a macro that changes formatting as a shortcut to learn which properties you can change and their names.
  - a. Colors are set with a long integer value and it is best to record seeting the color to see the value.
  - b. Font names and Standard Number Formatting are set by name, in quotes.
  - c. The With End With statements let you leave off the beginning parts of a property.

### **EXERCISE: Change Formatting**

- 1. Change your Invoice to General Formatting, Times New Roman text.
- 2. Record a macro that changes the font, font size, color, number formatting, and alignment of the Invoice.
- 3. Your instructor will review the macro with you.

## **Converting Numbers and Dates**

- 1. Common Excel formatting issues include numbers not coming through formatted correctly or coming through as text and dates needing formatting.
- 2. If a number is imported or downloaded and is left-aligned, it is often because it was imported as text.
  - a. Test by formatting as Dollars. If the number does not format with the \$ and 2 decimal points, it is text. Using it in a formula will usually convert it.
  - b. Use the Smart Tag to convert a number stored as text into a number.
  - c. To convert a number of cells:
    - i. Enter 1 into a blank cell.
    - ii. Copy it.
    - iii. Select the range to convert.
    - iv. Select Paste Special | Multiply (dropdown from Paste icon).
    - v. Excel will multiple each date by 1, converting it into a number.
    - vi. Delete the unneeded 1.
  - d. Often, you can simply read a number or a date stored as text into a double or a date variable and VBA will do the conversion:

```
dblTotalAmount = ActiveCell.Value
dteToday = ActiveCell.Value
```

#### **EXERCISE: Converting Numbers and Dates Using VBA**

- 1. Select the Sheet FormatExercise.
- 2. Write a macro, ConvertDate, that puts the value in B2 into C2 as a real date.
- 3. Write a macro, ConvertNumber, that puts the value in the B5 into C5 as a number.

### RANGE.OFFSET

- 1. Using **Select** to move to a cell, then changing its **Value** is fine, but you often have to move back to the original cell to continue processing.
- 2. Referring to a cell without a Select will let you change it: Range ("D7") . Value = 27
- 3. You have seen the .Offset method in your recorded Relative macros.
- 4. Using the .Offset method lets you refer to a cell by a row and column number. This allows you to calculate a cell reference in your code which is more efficient as well as more easily understood..
- 5. Range ("D4") .Offset (4, 3) .Value = 34 will put 34 in G8, which is 4 rows down and 3 columns right from D4.
- 6. Use a negative number for up a row or left a column:
  - Range ("D4") .Offset(-2, -3) .Value = 56 will put 56 in A2, which is 2 rows up and 3 columns left from D4.
- 7. The ActiveCell keyword is very useful with .Offset. You can select a cell, which becomes the ActiveCell, then offset so many rows and so many columns from the ActiveCell.

### **HOMEWORK EXERCISE: Range.Offset**

- 1. Select the OffsetExercise sheet.
- 2. Use three variables: dblCost, intAmount, and dblTotal.
- 3. Move to A2.
- 4. Set the values of dblCost and intAmount for row 2.
- 5. Calculate the Total for Soap and put it in D2. Do not move to D2.
- 6. Set the values of dblCost and intAmount for row 3.
- 7. Calculate the Total for Soup and put it in D3. Do not move to D3.
- 8. Set the values of dblCost and intAmount for row 4.
- 9. Calculate the Total for Sap and put it in D4. Do not move to D4.
- 10. Calculate the Total for all three items and put it in D5. Do not move to D5