VBA4-Using Cell Formulas in VBA

First, you'll use the WHILE-WEND and FOR...NEXT loop commands to repeat your processing for all the rows in your data file. Then, as your formulas grow more complex, it is easier to reuse a formula than to retype it again and again. This session will show you how to create a relative and absolute formula and add it to your spreadsheet. You'll learn the R1C1 style of addressing to precisely control which cells your formula references. The session will show you how to add Excel's built in functions to a formula using the <code>WorksheetFunction</code> statement. You'll also learn how to use the <code>Cells</code> statement to add formulas with calculated cell references so that your code can work with different files.

Looping – Performing the Same Operation Repetitively

- 1. Once you can process a row using Range. Offset, you will usually want to perform this operation for a number of rows.
- 2. The programming operation for this process is called looping.
- 3. You can loop in a number of different ways:
 - a. Loop a set number of times (FOR...NEXT)
 - b. Loop until a condition is met (WHILE...WEND). This only performs the loop if the condition is True.
 - c. Loop and test at the bottom of the loop (DO...UNTIL). This performs the loop at least once.
 - d. Loop through a collection of objects (FOREACH...NEXT).
- 4. We will talk about the first two since they are the most common.
- 5. IMPORTANT: When you loop through a list, remember to change rows after processing or you will loop endlessly.
 - a. CTRL+Break or CTRL+C will often break into a loop.
 - b. REMEMBER TO SAVE YOUR WORK BEFORE TESTING A LOOP, just in case.

Looping Through a List Using For...Next

- 1. The FOR...NEXT loop repeats one or more actions a set number of times.
- 2. You need to use a loop control variable to count the number of actions.
- 3. Using a lower case i, j, k, l etc. for the loop control variable is common practice and does not use the three letter identifiers.
- 4. Begin with the statement: FOR i = [lower] to [higher]
- 5. Add whatever statements you need.
- 6. End with the statement: Next i
- 7. Sample loop:

- ' Repeats the task for 10 rows. Stops on the 11th row.
- 8. A *kludge* is to loop for 500 rows and delete the rows that don't matter. (A kludge, pronounced "klooge" is a solution that is clumsy, inelegant, or inefficient, but gets the job done.)
- 9. You can add **Step** # to the first line to increment the loop control variable by a value other than 1:

```
For k=2 to 20 Step 2 will loop 10 times and increase the loop control variable by 2 each time. (i.e., 2, 4, 6, 8, etc. through 20. It will stop when the loop control variable is 22.)

For j = 10 to 1 Step -1 will loop 10 times. The loop control variable will start at 10 and be decreased by 1 each time. On the last iteration of the loop, the value will be 1
```

- 10. You can add Exit For or Exit Do to break out of a loop (usually using an IF statement to test for a condition.)
- 11. If you are kludging the For...Next loop, you can add an If statement such as:

 If Activecell.Offset(0,-1) = "" Then "" to avoid placing any text in a blank row.

EXERCISE: Loop for a set number of times

- 1. Open VBA4 Exercises.xlsm.
- 2. Select the ForNextLoop sheet.
- 3. Start in C2
- 4. Create the formula to test the Collateral Amount in B minus the Loan Amount in A.
- 5. If the result is a positive number, add the value "Good Loan" to the cell. If it is a negative number, add the value "Bad Loan" to the cell. Do this in VBA, not as an Excel formula to get practice in processing in VBA.
- 6. Move down one row.
- 7. Loop through the list as many times as necessary.

Looping Through a List Using While

- 1. FOR...NEXT is useful when you know the number of times you want to repeat the loop.
- 2. Often, you want to process a list until you reach the end of it.
 - a. The process would be "Process the row, move down a row, stop when you hit a blank row."
- 3. The first statement is **While** [some test]
- 4. Add processing steps.
- 5. The last statement is **Wend**
- 6. VBA will test the While value. If it is true, it will execute the loop. If it is not true, it will skip to the next statement after **Wend**. Therefore the **While** loop is executed only if the initial test is True.
- 7. As always, remember to move your cell pointer if you are processing a list.
 - a. SAVE YOUR WORK BEFORE TESTING A LOOP.

8. While ActiveCell.Value <> "" is the way to loop as long as there is a value in the current cell. You also might use an offset: While ActiveCell.Offset(0,-1).Value <> "" will test the column to the left of the current column.

EXERCISE: Loop while a condition is true

- 1. Select the WhileLoop Sheet.
- 2. Create a second loop, using a While loop to process the list of loans.
- 3. Which loop is more efficient for this task?

Using Shift+End to Find the Last Cell

The End arrow keys (End, $\uparrow \lor \leftarrow \rightarrow$) will move the cell pointer in the direction of the arrow. Press End, then press the arrow key.

- If your cursor starts in a cell with text, it will stop at the last cell with text in it.
- If your cursor starts in a blank cell, it will stop at the last cell that is blank.
- Using the combination Shift+End, arrow while holding down the Shift key will highlight as well
 move.

You can use the End, arrow key combination to quickly find the last row in a list. If the column contains blank cells, it may be problematic to use End, arrow. You can select a column that you know has data in each row. Or, you can start at the bottom of the spreadsheet and use End, ♠. This will stop on the last row of the list that has data. You need to be aware that .XLS sheets end at 65,536 and XLSX sheets end at 1,048,576. If you think your macros will be used on .XLS sheets, make sure to use 65,536.

The GoTo key, F5, is also used in this process. Press GoTo (or select it from Home | Editing | Find and Select icon on the Ribbon) and type the cell reference or range name you want to go to.

If you record the process, it looks like this:

```
Sub FindLastRow()
' FindLastRow Macro
          Application.Goto Reference:="R1048576C1"
          Selection.End(xlUp).Select
End Sub
```

This macro moves to the last row, first column. By adding the following line, you can save the last row of the list into a variable:

```
intLastRow = ActiveCell.Row
```

This code makes use of R1C1 addressing which we will cover in this class.

Adding a Formula to a Cell

- 1. In addition to changing the value and formatting of a cell, sometimes you want to put a formula in it.
- 2. You can add a formula that is absolute using

```
Range([cellref]).Formula = "=[formula]"
```

3. Notice the leading = and the quotes.

```
ActiveCell.Formula = "=B7*C6"
```

- 4. If your formula has quotes in it, you need to double the quotes, plus add additional quotes around the whole formula:
- 5. ActiveCell.Formula = "=""Monday""&"" Morning""" results in ="Monday"&" Morning" being placed in the cell and "Monday Morning" being displayed.
- 6. This is helpful if you know the cells referenced by the formula and the cell references will not change.
- 7. If you need to calculate the cell references, you'll need to use the R1C1 style of addressing.

Using the R1C1 Style of Addressing

- 1. Formulas are not stored as cell references such as =B4/B7.
- 2. Instead, they are stored as references to the row and column.

Assume your cursor is in A1.

```
=B4 is really stored as =R[3]C[1] 'Three rows down from current row, one column right 
=$B$4 is really stored as =R4C2 'The cell that is Row 4 and Column 2
```

- 3. Turn R1C1 notation on and off using File | Options | Formulas | R1C1 Reference Style.
- 4. Add Relative and Absolute formulas, then turn on R1C1 Reference Style. This is the easiest way to see what the R1C1 version of your formula is.
- 5. Rules for formulas stored using Row and Column numbers:

R# is Absolute. Always refers to a specific row.

C# is Absolute. Always refers to a specific column.

R[#] is Relative. It is so many rows from the current cell.

C[#] is Relative. It is so many columns from the current cell.

Positive numbers go down the rows. Negative numbers go up the rows.

Positive numbers go so many columns to the right. Negative numbers go so many columns to

Excel shows an error if the references are off the worksheet. (i.e., if your cursor is in B1 and the formula refers to =RC [-3].

6. Turning R1C1 notation on and off displays the formulas as specified. You can change back and forth as needed. Formulas still refer to the same location(s).

EXERCISE: Record using R1C1

- 1. In A1, enter 27.
- 2. In A2, enter 31.
- 3. In B1, enter 3.
- 4. Move to D4.
- 5. Record a macro called AddAFormula.
- 6. Use Absolute Addressing.
- 7. Add a formula in D4: =(\$A\$1+\$A\$2)/\$B\$1
- 8. Stop recording.
- 9. Move to D5.
- 10. Record a macro called AddRelativeFormula
- 11. Use Relative Addressing.
- 12. Add the formula in D5: =(A1+A2)/B1
- 13. View the macros in VBA Editor.

CELLS

- 1. You can't use row and column references in a Range ([cellref]) statement.
- In order to use row and column references, you need to use the Cells (Row, Column) equivalent.
- 3. Cells (Row, Column) takes numerical values for the row and column which means you can substitute variables for the numbers:

- 4. Cells has a similar list of methods and attributes as Range and ActiveCell so you can use it to Select, change Value or Formula, set formats, etc.
- 5. You may feel this is backwards to the Range statement: Range ("B4") is Cells (4,2).

Range.FormulaR1C1

- 1. If you are adding a line of code using an R1C1 style formula, you need to use the attribute .FormulaR1C1. You can use this with Range, ActiveCell, and Cells.
- 2. You can create the formula you need normally, then display it in R1C1 notation to copy and add to your VBA code.

EXERCISE: Using Cells

- 1. Add a Relative formula to cell A4 to total A1, A2, and A3.
- 2. Display it as R1C1.
- 3. Copy it.
- 4. Add it to a .FormulaR1C1 line of VBA code. Make sure to put it inside of quotes.
- 5. Delete the contents of A4.
- 6. Run the code.
- 7. Switch back and forth between R1C1 and Normal view.

Calculating an R1C1 Formula

- 1. If you want to calculate the row and column in your code, you will need to add the variables to the code in a slightly different manner.
- 2. Use VBA to calculate the row and column values and store in variables.
- 3. Create the formula line, but concatenate the variable names into the code. If you add them directly, VBA will place the names into the cell, resulting in an error.

- 4. Note the _ in the last line of the formulas. It allows a VBA line to wrap to the next line without an error. The convention is to indent the wrapped line(s) an additional tab stop or two.
- 5. If you want to add relative references, you need to encase the variables in [] and offset from the cell in which the formula is placed. The same formula as a relative reference would look like this:

Your formula would now read "Sum the cells in the same column, 11 rows up through the same column, one row up."

This would appear in the cell as:

```
=SUM(R[-11]C:R[-1]C) \text{ or } =SUM(E10:E20)
```

EXERCISE: Calculating an R1C1 Formula

- 1. Calculate the rows and columns needed to total the balances in columns A, B, and C of the ForNextLoop sheet.
- 2. Change the formula to a relative reference SUM function using the R1C1 values.
- 3. What is one way to find the last row of the list without having to count to it?

WorksheetFunctions

- 1. VBA has a good set of built-in functions. You should always try to use a VBA function first. If you get an error, then you need to try another solution.
- 2. VBA for Excel makes most of the built-in functions available in VBA using the **WorksheetFunctions** statement.
- 3. Type WorksheetFunctions. and Intellisense will display all the functions you can use.
- 4. Click on **WorksheetFunctions** and press F1 to see a Help screen of the available functions plus an explanation of their use.
- 5. This is different from creating a formula that uses a function and inserting it into a cell. **WorksheetFunctions** lets you use the function within VBA without going to Excel.