Exercise : Star Counter

Friday, September 20, 2019 9:27 AM

Files

- StarCounter_Unsolved/star_counter.xlsm

Instructions

- Create a VBA Script that tallies the number of "Full Stars" per row and enters them into the Total column. Starter Code is provided, but feel free to start from scratch if you want an extra challenge :-)

Hints

- You will need to use a nested for loop.
- You will need to create a variable to hold the number of stars and continually reset this variable at the start of each row.

Review: Star Counter

Friday, September 20, 2019 10:13 AM

```
Sub StarCounter()

  ' Create a variable to hold the StarCounter. We will repeatedly use this.
  dim StarCounter as Integer

  ' Loop through each row
  for i = 2 to 51

    ' Initially set the StarCounter to be 0 for each row
    StarCounter = 0

    ' While in each row, loop through each star column
    for j = 4 to 8

      ' If a column contains the word "Full-Star"...
      if (Cells(i, j).value = "Full-Star") then

        ' Add 1 to the StarCounter
        StarCounter = StarCounter + 1

      end if

    Next j

    ' Once we've completed all rows, print the value in the total column
    Cells(i, 9).value = StarCounter

  Next i
```

- Creates loop that goes through rows 2 to 51

```
Sub StarCounter()

  ' Create a variable to hold the StarCounter. We will repeatedly use this.
  dim StarCounter as Integer

  ' Loop through each row
  for i = 2 to 51

    ' Initially set the StarCounter to be 0 for each row
    StarCounter = 0

    ' While in each row, loop through each star column
    for j = 4 to 8

      ' If a column contains the word "Full-Star"...
      if (Cells(i, j).value = "Full-Star") then

        ' Add 1 to the StarCounter
        StarCounter = StarCounter + 1

      end if

    Next j

    ' Once we've completed all rows, print the value in the total column
    Cells(i, 9).value = StarCounter

  Next i
```

- Creates loop that goes through columns 4 to 8 for each row

```
Sub StarCounter()

  ' Create a variable to hold the StarCounter. We will repeatedly use this.
  dim StarCounter as Integer

  ' Loop through each row
  for i = 2 to 51

    ' Initially set the StarCounter to be 0 for each row
    StarCounter = 0

    ' While in each row, loop through each star column
    for j = 4 to 8

      ' If a column contains the word "Full-Star"...
      if (Cells(i, j).value = "Full-Star") then

        ' Add 1 to the StarCounter
        StarCounter = StarCounter + 1

      end if

    Next j

    ' Once we've completed all rows, print the value in the total column
    Cells(i, 9).value = StarCounter

  Next i
```

- Create variable 'StarCounter' that will hold count of stars
- 'StarCounter' resets to zero for each row

```
Sub StarCounter()

 ' Create a variable to hold the StarCounter. We will repeatedly use this.
 dim StarCounter as Integer

 ' Loop through each row
 for i = 2 to 51

  ' Initially set the StarCounter to be 0 for each row
  StarCounter = 0

  ' While in each row, loop through each star column
  for j = 4 to 8

   ' If a column contains the word "Full-Star"...
   if (Cells(i, j).value = "Full-Star") then

    ' Add 1 to the StarCounter
    StarCounter = StarCounter + 1

   end if

  Next j

  ' Once we've completed all rows, print the value in the total column
  Cells(i, 9).value = StarCounter

 Next i
```

- Every time we find a "Full-Star" value within a column, we add one to our `StarCounter`

```
Sub StarCounter()

  ' Create a variable to hold the StarCounter. We will repeatedly use this.
  dim StarCounter as Integer

  ' Loop through each row
  for i = 2 to 51

    ' Initially set the StarCounter to be 0 for each row
    StarCounter = 0

    ' While in each row, loop through each star column
    for j = 4 to 8

      ' If a column contains the word "Full-Star"...
      if (Cells(i, j).value = "Full-Star") then

        ' Add 1 to the StarCounter
        StarCounter = StarCounter + 1

      end if

    Next j

    ' Once we've completed all rows, print the value in the total column
    Cells(i, 9).value = StarCounter

  Next i
```

- The value of `StarCounter` is placed within a total columns of the current row after the conclusion of the inner loop and then we move onto the next value in the outer loop

```
Sub StarCounter()

  ' Create a variable to hold the StarCounter. We will repeatedly use this.
  dim StarCounter as Integer

  ' Loop through each row
  for i = 2 to 51

    ' Initially set the StarCounter to be 0 for each row
    StarCounter = 0

    ' While in each row, loop through each star column
    for j = 4 to 8

      ' If a column contains the word "Full-Star"...
      if (Cells(i, j).value = "Full-Star") then

        ' Add 1 to the StarCounter
        StarCounter = StarCounter + 1

      end if

    Next j

    ' Once we've completed all rows, print the value in the total column
    Cells(i, 9).value = StarCounter

  Next i
```

VBA Formatting

Friday, September 20, 2019 10:53 AM

- Not only can we use VBA to change the values within cells, but we can also code in formatting fairly easily using a variety of functions.

```
Sub formatter()

  ' Set the Font color to Red
  Range("A1").Font.ColorIndex = 3

  ' Set the Cell Colors to Red
  Range("A2:A5").Interior.ColorIndex = 3
```

- How do I know which Color Index number corresponds to each color?
- Send to slack -> http://dmcritchie.mvps.org/excel/colors.htm

- Live Demo….How can we color columns B, C, D?

formatter

Exercise: VBA Gradebook

Friday, September 20, 2019 12:43 PM


grader

Files

- GradeBook_Unsolved/grader.xlsm

Instructions

- Using `grader.xlsm` as a starting point, create a grade calculator using **conditionals**. This calculator will convert a student's numeric grade into a letter grade, and style the resulting cell accordingly.
- Once complete your script should perform the following:
  - If the score is over 90, the student will receive an "A" in the letter grade cell, and the Pass/Warning/Fail cell will be filled green with the text "Pass."
  - If the score is between 80 and 89 (inclusive), the student will receive a "B" in the letter grade cell, and the Pass/Warning/Fail cell will be filled green with the text "Pass."
  - If the score is between 70 and 79 (inclusive), the student will receive a "C" in the letter grade cell, and the Pass/Warning/Fail cell will be filled yellow with the text "Warning."
  - Finally, if the score is below a 70, the student will receive an "F" in the letter grade cell, and the Pass/Warning/Fail cell will be filled red with the text "Fail."

Review: VBA Gradebook

Friday, September 20, 2019 2:48 PM

- Modifying the formatting/value of cells B2/C2 based upon the value stored within A2.
- When the value of A2 changes, so too does the formatting/value of cells B2/C2.
- General Format
- IF >= 90
    - Else If >=80
    - Else If >= 70
    - Else

```
Sub GradeBook()

  ' Check if the student's grade is greater than or equal to 90...
  If Cells(2, 2).Value >= 90 Then

      ' Establish that the grade is Passing
      Cells(2, 3).Value = "Pass"

      ' Color the Passing grade green
      Cells(2, 3).Interior.ColorIndex = 4

      ' Set the letter grade to "A"
      Cells(2, 4).Value = "A"

  ' Check if the student's grade is greater than or equal to 80...
  ElseIf Cells(2, 2).Value >= 80 Then

      ' Establish that the grade is Passing
      Cells(2, 3).Value = "Pass"

      ' Color the Passing grade green
      Cells(2, 3).Interior.ColorIndex = 4

      ' Set the letter grade to "B"
      Cells(2, 4).Value = "B"
```

Exercise: Checker Board

Friday, September 20, 2019 3:11 PM

checkerbo...

Instructions

- Using VBA scripts, create an 8x8 grid with alternating red and black squares.

Hints

- You will need to use nested for loops, conditionals, mods, and formatting to create the board.
- This is a tricky problem! Try to pseudocode a plan first.
  - Unlike previous activities, this activity can be solved in a multitude of different ways. While some methods may be more efficient than others, simply finding a solution to the problem is a great start!

BREAK TIME!!!!

Friday, September 20, 2019 5:32 PM

Review: Checker Board

Friday, September 20, 2019 3:35 PM

- Do you know Tim???

```
' This solution uses a trick of adding the row and column numbers:
' You can Mod that sum and use it to determine the color
```

```
Sub CheckerboardAlternateSolution()
    Dim r, c As Integer
    For r = 1 To 8

        For c = 1 To 8

            If (r + c) Mod 2 = 0 Then
                Cells(r, c).Interior.ColorIndex = 1 ' Black
            Else
                Cells(r, c).Interior.ColorIndex = 3 ' Red

            End If
        Next c
    Next r
End Sub
```

```
Sub CheckerBoard()

  ' Setup a counter to track cell number
  Dim cellnumber as Integer
  Dim i, j As Integer
  cellnumber = 1

  ' Loop through each row of the board
  For i = 1 to 8

    ' Loop through each column of the board
    For j = 1 to 8

      ' If we are on a cell that is divisible by 2 then color it black
      If (cellnumber Mod 2 = 0) then

        Cells(i, j).Interior.ColorIndex = 1

      ' Otherwise color it red
      Else

        Cells(i, j).Interior.ColorIndex = 3

      End if

      ' Add one to our cell number each time
      cellnumber = cellnumber + 1

    Next j

    ' Whenever we start on a new row, we also add one to the cell number (to create the alternation)
    cellnumber = cellnumber + 1

  Next i

End Sub
```

- Creates loop for each row on board

```
Sub CheckerBoard()

  ' Setup a counter to track cell number
  Dim cellnumber as Integer
  Dim i, j As Integer
  cellnumber = 1

  ' Loop through each row of the board
  For i = 1 to 8

    ' Loop through each column of the board
    For j = 1 to 8

      ' If we are on a cell that is divisible by 2 then color it black
      If (cellnumber Mod 2 = 0) then

        Cells(i, j).Interior.ColorIndex = 1

      ' Otherwise color it red
      Else

        Cells(i, j).Interior.ColorIndex = 3

      End if

      ' Add one to our cell number each time
      cellnumber = cellnumber + 1

    Next j

    ' Whenever we start on a new row, we also add one to the cell number (to create the alternation)
    cellnumber = cellnumber + 1

  Next i

End Sub
```

- Creates loop for each column on the board

```
Sub CheckerBoard()

  ' Setup a counter to track cell number
  Dim cellnumber as Integer
  Dim i, j As Integer
  cellnumber = 1

  ' Loop through each row of the board
  For i = 1 to 8

    ' Loop through each column of the board
    For j = 1 to 8

      ' If we are on a cell that is divisible by 2 then color it black
      If (cellnumber Mod 2 = 0) then

        Cells(i, j).Interior.ColorIndex = 1

      ' Otherwise color it red
      Else

        Cells(i, j).Interior.ColorIndex = 3

      End if

      ' Add one to our cell number each time
      cellnumber = cellnumber + 1

    Next j

    ' Whenever we start on a new row, we also add one to the cell number (to create the alternation)
    cellnumber = cellnumber + 1

  Next i

End Sub
```

- Creates a counter to track cell number
  - Starts counter at 1
- Adds 1 to counter each time through the inner loop (columns)
- Adds 1 to counter each time through the outer loop (rows)

```
Sub CheckerBoard()

  ' Setup a counter to track cell number
  Dim cellnumber as Integer
  Dim i, j As Integer
  cellnumber = 1

  ' Loop through each row of the board
  For i = 1 to 8

    ' Loop through each column of the board
    For j = 1 to 8

      ' If we are on a cell that is divisible by 2 then color it black
      If (cellnumber Mod 2 = 0) then

        Cells(i, j).Interior.ColorIndex = 1

      ' Otherwise color it red
      Else

        Cells(i, j).Interior.ColorIndex = 3

      End if

      ' Add one to our cell number each time
      cellnumber = cellnumber + 1

    Next j

    ' Whenever we start on a new row, we also add one to the cell number (to create the alternation)
    cellnumber = cellnumber + 1

  Next i

End Sub
```

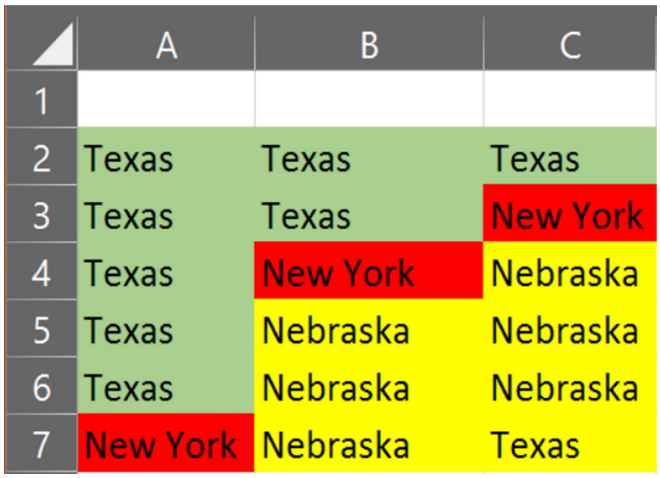
- If counter is even, the format to black……else format to red

```
Sub CheckerBoard()

  ' Setup a counter to track cell number
  Dim cellnumber as Integer
  Dim i, j As Integer
  cellnumber = 1

  ' Loop through each row of the board
  For i = 1 to 8

    ' Loop through each column of the board
    For j = 1 to 8

      ' If we are on a cell that is divisible by 2 then color it black
      If (cellnumber Mod 2 = 0) then

        Cells(i, j).Interior.ColorIndex = 1

      ' Otherwise color it red
      Else

        Cells(i, j).Interior.ColorIndex = 3

      End if

      ' Add one to our cell number each time
      cellnumber = cellnumber + 1

    Next j

    ' Whenever we start on a new row, we also add one to the cell number (to create the alternation)
    cellnumber = cellnumber + 1

  Next i

End Sub
```

Looking to the next cell

Friday, September 20, 2019 4:22 PM

- When looping through rows and/or columns, it sometimes becomes necessary to check for changes and then run some alternative code based upon those changes
- This code is looping through the rows in the first column and printing a message to the screen whenever the value changes from one row to the next



	A	B	C
1			
2	Texas	Texas	Texas
3	Texas	Texas	New York
4	Texas	New York	Nebraska
5	Texas	Nebraska	Nebraska
6	Texas	Nebraska	Nebraska
7	New York	Nebraska	Texas

```
Sub NextCells()

  ' Set a variable for specifying the column of interest
  Dim column As Integer
  column = 1

  ' Loop through rows in the column
  For i = 2 To 6

    ' Searches for when the value of the current cell is different than that of the next cell
    If Cells(i, column).Value <> Cells(i + 1, column).Value Then

      ' Message Box the value of the current cell and value of the next cell
      MsgBox (Cells(i, column).Value & " and then " & Cells(i + 1, column).Value)

    End If

  Next i

End Sub
```

Exercise: Card Checker

Friday, September 20, 2019 5:01 PM

Files

- 06-Stu_CreditCardChecker/credit_charges.xlsm

Instructions

- Create a VBA script that will process the credit card purchases, identifying each of the unique brands listed.
- For the *Basic* assignment, create a single pop-up message for each of the Credit Card brands listed by looping through the list.

Review: Credit Card Checker

Friday, September 20, 2019 5:04 PM

```
Sub credit_card()

  ' Set an initial variable for holding the brand name
  Dim Brand_Name As String

  ' Loop through all credit card purchases
  For i = 2 To 101

    ' Check if we are still within the same credit card brand, if we are not...
    If Cells(i + 1, 1).Value <> Cells(i, 1).Value Then

      ' Message Box the unique Bank Name
      MsgBox(Cells(i, 1).Value)

    End If

  Next i

End Sub
```

- Create a loops that goes through all of the rows where credit card names exist (column A).

```
Sub credit_card()

  ' Set an initial variable for holding the brand name
  Dim Brand_Name As String

  ' Loop through all credit card purchases
  For i = 2 To 101

    ' Check if we are still within the same credit card brand, if we are not...
    If Cells(i + 1, 1).Value <> Cells(i, 1).Value Then

      ' Message Box the unique Bank Name
      MsgBox(Cells(i, 1).Value)

    End If

  Next i

End Sub
```

- Checks for times where the contents of a cell in column A do not match the contents of the cell in the next row down.

```
Sub credit_card()

  ' Set an initial variable for holding the brand name
  Dim Brand_Name As String

  ' Loop through all credit card purchases
  For i = 2 To 101

    ' Check if we are still within the same credit card brand, if we are not...
    If Cells(i + 1, 1).Value <> Cells(i, 1).Value Then

      ' Message Box the unique Bank Name
      MsgBox(Cells(i, 1).Value)

    End If

  Next i

End Sub
```

- Whenever the contents of the two cells do not match, we print the contents of the first cell to the screen before continuing

on.

```
Sub credit_card()

  ' Set an initial variable for holding the brand name
  Dim Brand_Name As String

  ' Loop through all credit card purchases
  For i = 2 To 101

    ' Check if we are still within the same credit card brand, if we are not...
    If Cells(i + 1, 1).Value <> Cells(i, 1).Value Then

      ' Message Box the unique Bank Name
      MsgBox(Cells(i, 1).Value)

    End If

  Next i

End Sub
```

Wells Fargo Setup

Friday, September 20, 2019 5:33 PM

- For the rest of class, you will get into small groups to create a VBA script that takes an unmodified Excel workbook with several sheets of Wells Fargo data, formats each sheet so that they are more readable, and then combine all of the data into a single table.
- One member of the group in charge of writing the code out so that they everyone is working on the same workbook.
- This activity will be split into individual parts and the class will be coming back together to review once each part has been completed.

Exercise: Wells Fargo (Part 1)

Friday, September 20, 2019 6:24 PM

Files

- Activities/07-Stu_WellsFargo_Pt1/combined_wells_fargo.xlsm

Instructions

i. **Extract words before the phrase "_Wells_Fargo" to figure out which State**.
   i. Get the Worksheet Name (SomeVariable = ws.Name)
   ii. Use Split to get the State Name

ii. **Add the State to the first column of each spreadsheet.**
   i. ws.Range("A1").EntireColumn.Insert
   ii. ws.Range("A2:A" & LastRow)

iii. **Convert the headers of each row to simply say the year**.
   i. Use Split Method

iv. **Convert the numbers to currency values for all cells**
   i. ws.Cells(1, 1).Style = "Currency"

** Other Tips

- Determine the last row of data in a Worksheet
  - SomeVariable = ws.Cells(Rows.Count, 1).End(xlUp).Row
- Determine the last column of data in a Worksheet
  - SomeVariable = ws.Cells(1, Columns.Count).End(xlToLeft).Column

- To solve this, we will make use of a For Each loop: in Excel, we can use this to loop over each worksheet in the workbook, no matter how many there are. The syntax to loop over all worksheets is this:

```
For Each ws In Worksheets
  ' do stuff with the worksheet (ws) '
Next ws
```

- First work on getting the correct formatting on one sheet before moving onto creating a loop that formats each sheet within your workbook.

Review: Wells Fargo (Part 1)

Friday, September 20, 2019 6:28 PM

- To loop through all worksheets using VBA we use a `For Each` loop that loops through the built-in array of `Worksheets`

```
Sub WellsFargo_PtI()

    ' ' ----------------------------------------
    ' LOOP THROUGH ALL SHEETS
    ' ----------------------------------------
    For Each ws in Worksheets
```

- To collect the state's name, the code looks at the name of the current worksheet - `ws.Name` - and then splits on underscores ('_').
- It can then place the name alone by referencing `State(0)` since the state name is stored at index 0 in the `State` array.

```
' Created a Variable to Hold File Name, Last Row, Last Column, and Year
Dim WorksheetName As String

' Determine the Last Row
LastRow = ws.Cells(Rows.Count, 1).End(xlUp).Row

' Grabbed the WorksheetName
WorksheetName = ws.Name
' MsgBox WorksheetName

' Split the WorksheetName
State = Split(WorksheetName, "_")
' MsgBox State(0)

' Add the State to the Column
ws.Range("A1").EntireColumn.Insert

' Add the word State to the First Column Header
ws.Cells(1, 1).Value = "State"

' Add the State to all rows
ws.Range("A2:A" & LastRow) = State(0)
```

-