VBA: Tutorial Week 5

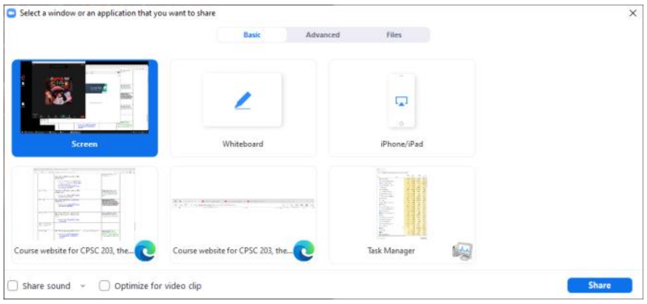
- Branching, looping and the `InlineShapes` collection
- Option Explicit
- Using the VBA debugger
- Requirements for Workbook Exercise #5

Official resource for MS-Office products: https://support.office.com

First Tutorial: Monday Or Tuesday

FYI For The Tutorial Instructor/TA

- Since you will be playing a video with a voice narration make sure that you have enabled the “Share sound” option if you are using Zoom.
  - You might want to use the pulldown (triangle) to ensure that audio is set to ‘stereo’ rather than ‘mono’.


Select a window or an application that you want to share
Basic
Advanced
Files
Screen
Whiteboard
iPhone/iPad
Course website for CPSC 203, the...
Course website for CPSC 203, the...
Task Manager
Share sound
Optimize for video clip
Share

VBA tutorial notes by James Tam

Microsoft Introduction/Overview Of VBA

- https://docs.microsoft.com/en-us/office/vba/library-reference/concepts/getting-started-with-vba-in-office

VBA tutorial notes by James Tam

Activities In Tutorial

- TA demos:
  - Used for more complex features (typically multiple steps are required).
  - The tutorial instructor will show on the projector/instructor computer each step for running the feature in Excel.
  - Unless otherwise specified the tutorial material will take the form of a TA demonstrating the use of features in Excel.
  - Slides titled "Lecture Review" are covered for the second time and dealing with less complex material.
    - For this reason they will only be covered briefly in tutorial.
- Student exercises:
  - Used instead of TA demos for simpler features.
  - You will have already been given a summary of how to invoke the feature and the purpose of the exercise is to give you a chance to try it out and get help if needed.

Return To Collections

- Recall with the collections you have seen: `Documents`, `InlineShapes`, `Shapes`, `Tables` you can access a particular element or item in the collection by that item's index.
  - Example (accesses the first `InlineShape`): `ActiveDocument.InlineShapes(1)`
- Also the number of items in the collection can be accessed through the collection's count attribute.
  - Example (the variable `numTables` will contain the current number of tables in the currently active Word document):

```
numTables =  ActiveDocument.Tables.count
```

Return To Collections (2)

- Now that you have learned how to use looping and branching structures, you can:
  - Access each item in a collection (using a loop).
  - Check if the number of items in the collection is the desired amount (using a branch).

VBA tutorial notes by James Tam

Collections: `Inline Shapes`

- **Example program**: `1_loop_branch_inline_shapes`
  - For documents containing 2 - 4 `InlineShapes` (images) the program will halve the size of odd numbered images.

```
Sub reduceOddInlineShapes()
    Const MIN_SHAPES As Long = 2
    Const MAX_SHAPES As Long = 4
    Dim count As Long
    Dim numShapes As Long
    Dim tempWidth As Long
    numShapes = ActiveDocument.InlineShapes.count
    If ((numShapes < MIN_SHAPES) Or (numShapes > MAX_SHAPES)) Then
        MsgBox ("Number of Inline Shapes not " & MIN_SHAPES & _
          "-" & MAX_SHAPES)
```

VBA tutorial notes by James Tam

Collections: `Inline Shapes` (2)

```
    Else
        count = 1
        Do While (count <= numShapes)
            If ((count Mod 2) = 0) Then
                tempWidth = ActiveDocument.InlineShapes(count).Width / 2
                ActiveDocument.InlineShapes(count).Width = tempWidth
            End If
            count = count + 1
        Loop
    End If
End Sub
```

Student Exercise 1

- Write a program that will prompt the user for a positive integer value (1 or greater).
- The program will then double the size of the item # of the `Inline Shape` in the currently active document.
- **Name of the document containing the solution**: `1_enlarge_A_pic_solution`

Student Exercise 4

- Modify the solution to the previous exercise so that the program error checks the user's input.
- If the value enter by the user is less than 1 or it exceeds the current number of In line shapes in the document:
  - The program will display an error message specifying the correct range of values that can be entered (it needs to be based on the actual number of shapes in the currently active document).
  - The program will repeat the prompt until a value within the correct range has been entered.
- **Name of the document containing the solution**: `2_enlarge_A_pic_solutionV2`

Counting Occurrences Of A Word

- It's an application of the '`Find`' method of the `ActiveDocument` object combined with looping.
- Why count occurrences:
  - Evaluating resumes by matching skills sought vs. skills listed by the applicant.
  - Ranking the relevance of a paper vs. a search topic by the number of times that the topic is mentioned.
    - Word frequency may be one criteria employed when websites rank search results according to relevance

Checking Occurrences

- **Word document containing the macro** (actually it checks if word is or isn't found rather than doing an actual count but a small modification will allow a count to be performed):
- `2_determine_if_word_occurs`

```
Sub checkingOccurence()
    Dim occurs As Boolean
    Dim searchWord As String
    searchWord = InputBox("Word to search for")
    occurs = False
    With ActiveDocument.Content.Find
        Do While .Execute(FindText:=searchWord, Forward:=True, _
          MatchWholeWord:=True) = True
            occurs = True 'Word was found change state
        Loop
    End With
```

VBA tutorial notes by James Tam

- True: search for exact word
- False: partial match counted e.g. when looking for 'the' words like 'there' are counted

Checking Occurrences

Search not started, assume Word not in document

- **Word document containing the macro** (actually it checks if word is or isn't found rather than doing an actual count but a small modification will allow a count to be performed):
- `2_determine_if_word_occurs`

```
Sub checkingOccurence()
    Dim occurs As Boolean
    Dim searchWord As String
    searchWord = InputBox("Word to search for")
    occurs = False
    With ActiveDocument.Content.Find
        Do While .Execute(FindText:=searchWord, Forward:=True, _
          MatchWholeWord:=True) = True
            occurs = True
        Loop
    End With
```

Word to find in document

Body of Do-While entered when a match occurs (in this case can set variable to indicate that it's true that wo was found)

VBA tutorial notes by James Tam

Checking Occurrences

- Once the search is complete display the results of the search

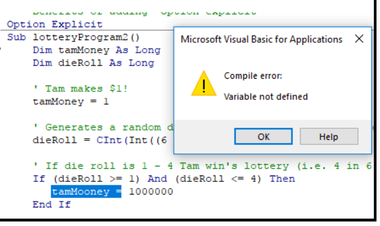
```
    If (occurs = True) Then
        MsgBox ("'" & searchWord & "'" & " was found")
    Else
        MsgBox ("'" & searchWord & "'" & " could not be found")
    End If
End Sub
```

Student Exercise 3

- Modify the previous program. Instead of determining if the search word was or was not found have your program count the number of occurrences.
  - A word should be counted if it's a partial match e.g. when search for 'the' the words 'the', 'their', 'they're' and 'there' should all be counted.
- After the search is complete the number of occurrences should be displayed in a popup
- **Name of the document containing the solution**: `3_count_occurences.docm`
- Example data used to test the correctness of your solution.
  - Search for 'the', count should be 2
  - Search for 'at', count should be 2
  - Search for 't', count should be 4

Option Explicit Used

- Including ‘Option Explicit’ requires that variables must be created via ‘Dim’ variable declaration
  - E.g. `Dim tamMoney As Long`
- After creating/declaring the variable the memory location can be used by assigning a value into that location.
  - E.g. `tamMoney = 1`
- Advantage: helps catch bugs
  - If you type in the wrong variable name if you use Option Explicit then VBA may tell you exactly where the error lies.



```
Option Explicit
Sub lotteryProgram2()
    Dim tamMoney As Long
    Dim dieRoll As Long

    ' Tam makes $1!
    tamMoney = 1

    ' Generates a random d
    dieRoll = CInt(Int((6

    ' If die roll is 1 - 4 Tam win's lottery (i.e. 4 in 6
    If (dieRoll >= 1) And (dieRoll <= 4) Then
       tamMooney = 1000000
    End If
```

Microsoft Visual Basic for Applications

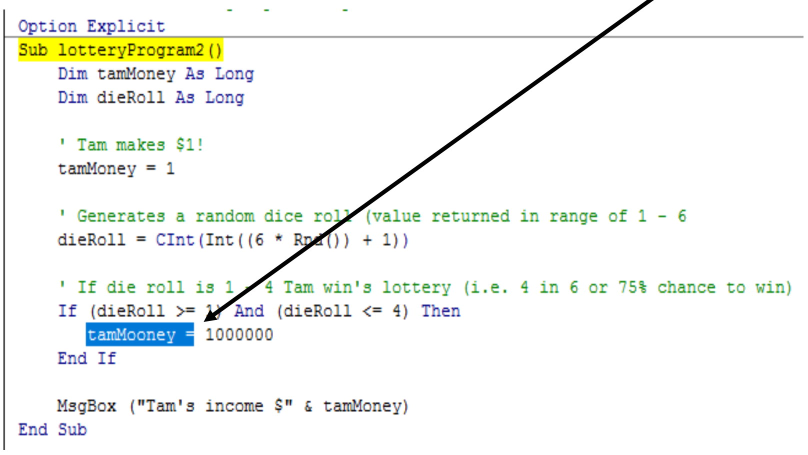
Compile error:

Variable not defined

Example: Option Explicit Used

VBA will automatically catch the error and point out the location

- **Example:** `3A_optionExplicitUsed.docm`



```
Option Explicit
Sub lotteryProgram2()
    Dim tamMoney As Long
    Dim dieRoll As Long

    ' Tam makes $1!
    tamMoney = 1

    ' Generates a random dice roll (value returned in range of 1 - 6
    dieRoll = CInt(Int((6 * Rnd()) + 1))

    ' If die roll is 1 - 4 Tam win's lottery (i.e. 4 in 6 or 75% chance to win)
    If (dieRoll >= 1) And (dieRoll <= 4) Then
       tamMooney = 1000000
    End If

    MsgBox ("Tam's income $" & tamMoney)
End Sub
```

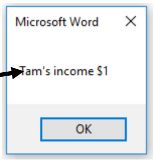
Example: Option Explicit Not Used

- **Example:** `3B_optionExplicitNotUsed.docm`

```
Sub lotteryProgram2()
    Dim tamMoney As Long
    Dim dieRoll As Long
    tamMoney = 1
   dieRoll = CInt(Int((6 * Rnd()) + 1))
   If (dieRoll >= 1) And (dieRoll <= 4) Then
       tamMooney = 1000000
    End If
    MsgBox ("Tam's income $" & tamMoney)
End Sub
```

The program erroneously set the wrong variable!

Tam didn't get the "big bucks" ☹ Errors like this can be hard to catch/fix in all but smallest programs



Tam's income $1

A More Complex But Practical Example

- **Example:** `4_set_fill_color_for_all_documents_in_a_folder.docm` (The file name describes exact what this program does).

```
Sub setFillAllFolderDocuments()
    Const ERROR_MESSAGE As String = "No shapes in document to
      fill"
    Dim location As String
    Dim currentDocumentName As String
    Dim fullname As String
    Dim currentShape As Long
    Dim numShapes As Long
    location = ""
    currentDocumentName = ""
```

A More Complex But Practical Example (2)

```
'Check if user didn't enter any location
Do While (location = "")
    location = InputBox("Enter path to Word documents " &
      _ "(e.g. C:\temp): ")
    If (location = "") Then
        MsgBox ("You entered '" & location & _
            "', don't enter an empty location")
    End If
Loop
```

A More Complex But Practical Example (3)

```
'Separator between last containing folder and filename
location = location & "\"

'Only consider and open Word (2003 or 2007+) documents
currentDocumentName = Dir(location & "*.doc*")

'Check if program is unable to find Word documents in location
If (currentDocumentName = "") Then
    MsgBox ("Unable to retreive any docs in the " & _
      "specified location")
```

A More Complex But Practical Example (4)

```
'Folder contains at least one Word document
 Else
    'Loop executes so long as there is another Word document
    'that hasn't already been accessed
    Do While (currentDocumentName <> "")
        fullname = location & currentDocumentName
        Documents.Open (fullname)
        numShapes = ActiveDocument.Shapes.Count

        'No shapes write error message
        If (numShapes = 0) Then
            Selection.Font.Bold = True
            Selection.Font.ColorIndex = wdRed
            Selection.Font.Size = 24
            Selection.TypeText (ERROR_MESSAGE)
```

VBA tutorial notes by James Tam

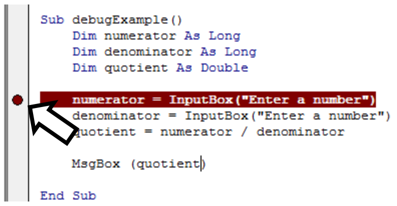
A More Complex But Practical Example (5)

```
    'Document has shapes
    Else
        'Starting with first shape so long as there's another
        'shape in document repeat loop
        currentShape = 1
        Do While (currentShape <= numShapes)
            ActiveDocument.Shapes(currentShape).Fill.ForeColor =
               vbRed
                'Move onto next shape
                currentShape = currentShape + 1
            Loop 'Goes through each shape in current doc
        End If 'Checks if any shapes in current doc
        'Automatically save and close document, move onto next
        ActiveDocument.Close (wdSaveChanges)
        currentDocumentName = Dir
    Loop 'Goes through each Word doc
End If 'Checks if any Word docs in folder
```

VBA tutorial notes by James Tam

The VBA Debugger

- Debuggers can be used to help find errors in your program
- Setting up breakpoints
  - Points in the program that will 'pause' until you proceed to the next step
  - Useful in different situations
    - The program 'crashes' but you don't know where it is occurring
      - Pause before the crash
    - An incorrect result is produced but where is the calculation wrong
- Set up breakpoints
  - Click in the left margin

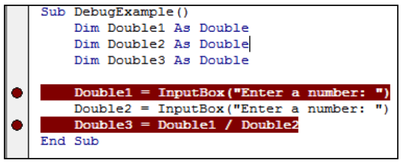


```
Sub debugExample()
    Dim numerator As Long
    Dim denominator As Long
    Dim quotient As Double

●   numerator = InputBox("Enter a number")
    denominator = InputBox("Enter a number")
    quotient = numerator / denominator

    MsgBox (quotient)

End Sub
```

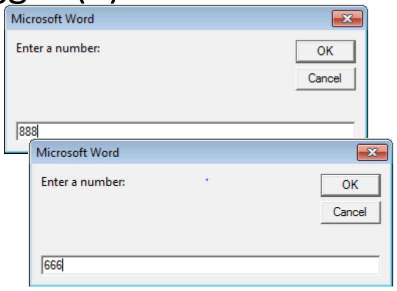
VBA tutorial notes by James Tam

The VBA Debugger (2)

- Multiple breakpoints



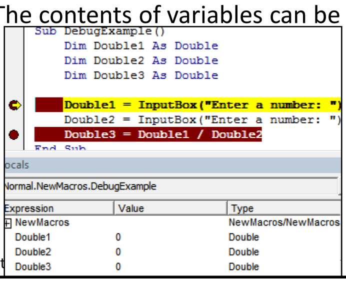
```
Sub DebugExample()
    Dim Double1 As Double
    Dim Double2 As Double
    Dim Double3 As Double

●   Double1 = InputBox("Enter a number: ")
    Double2 = InputBox("Enter a number: ")
●   Double3 = Double1 / Double2
End Sub
```



Microsoft Word
Enter a number:
888

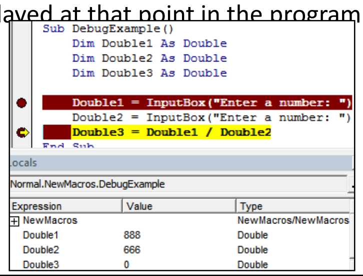
Microsoft Word
Enter a number:
666

- Program pauses when breakpoints are reached
  - The contents of variables can be displayed at that point in the program



```
Sub DebugExample()
    Dim Double1 As Double
    Dim Double2 As Double
    Dim Double3 As Double

    Double1 = InputBox("Enter a number: ")
    Double2 = InputBox("Enter a number: ")
●   Double3 = Double1 / Double2
End Sub
```

```
Locals
Normal.NewMacros.DebugExample
Expression      Value     Type
+ NewMacros               NewMacros/NewMacros
  Double1       0         Double
  Double2       0         Double
  Double3       0         Double
```



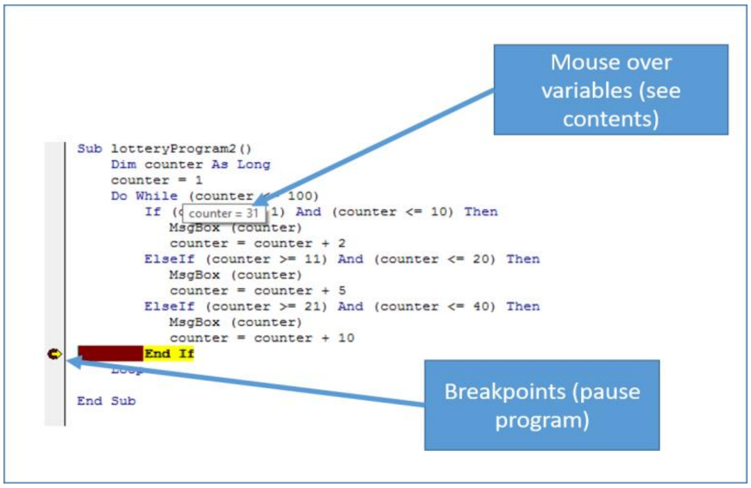
```
Sub DebugExample()
    Dim Double1 As Double
    Dim Double2 As Double
    Dim Double3 As Double

●   Double1 = InputBox("Enter a number: ")
    Double2 = InputBox("Enter a number: ")
    Double3 = Double1 / Double2
End Sub
```

```
Locals
Normal.NewMacros.DebugExample
Expression      Value     Type
+ NewMacros               NewMacros/NewMacros
  Double1       888       Double
  Double2       666       Double
  Double3       0         Double
```

VBA tutorial notes by James Tam

The VBA Debugger (3)

- Combining breakpoints and viewing variables.


Mouse over variables (see contents)
Breakpoints (pause program)

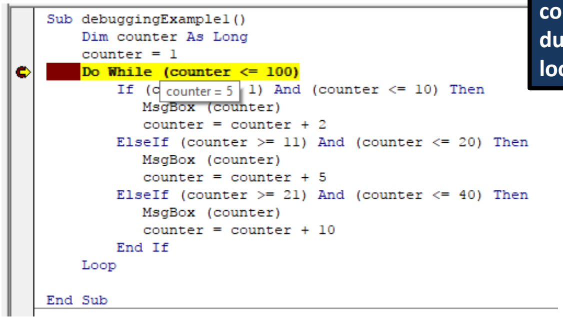
```
Sub lotteryProgram2()
    Dim counter As Long
    counter = 1
    Do While (counter <= 100)
        If (c counter = 31 1) And (counter <= 10) Then
            MsgBox (counter)
            counter = counter + 2
        ElseIf (counter >= 11) And (counter <= 20) Then
            MsgBox (counter)
            counter = counter + 5
        ElseIf (counter >= 21) And (counter <= 40) Then
            MsgBox (counter)
            counter = counter + 10
        End If
    Loop

End Sub
```

VBA tutorial notes by James Tam

An Example To Run With The Debugger

- **Example:** `5_debuggerExample.docm`
  - Sub: `DebuggingExample1`

Set up a breakpoint and trace through the program step-by-step while viewing the contents of the loop control during each iteration of the loop

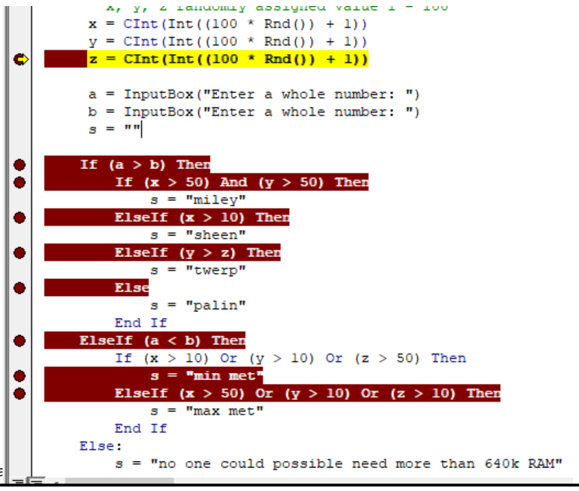


```
Sub debuggingExample1()
    Dim counter As Long
    counter = 1
    Do While (counter <= 100)
        If (c counter = 5  1) And (counter <= 10) Then
            MsgBox (counter)
            counter = counter + 2
        ElseIf (counter >= 11) And (counter <= 20) Then
            MsgBox (counter)
            counter = counter + 5
        ElseIf (counter >= 21) And (counter <= 40) Then
            MsgBox (counter)
            counter = counter + 10
        End If
    Loop

End Sub
```

VBA tutorial notes by James Tam

An Example To Run With The Debugger (2)

- **Example (cont'):** `5_debuggerExample.docm`
  - Sub: `DebuggingExample2`



```
    x, y, z randomly assigned value 1 - 100
    x = CInt(Int((100 * Rnd()) + 1))
    y = CInt(Int((100 * Rnd()) + 1))
    z = CInt(Int((100 * Rnd()) + 1))

    a = InputBox("Enter a whole number: ")
    b = InputBox("Enter a whole number: ")
    s = ""

    If (a > b) Then
        If (x > 50) And (y > 50) Then
            s = "miley"
        ElseIf (x > 10) Then
            s = "sheen"
        ElseIf (y > z) Then
            s = "twerp"
        Else
            s = "palin"
        End If
    ElseIf (a < b) Then
        If (x > 10) Or (y > 10) Or (z > 50) Then
            s = "min met"
        ElseIf (x > 50) Or (y > 10) Or (z > 10) Then
            s = "max met"
        End If
    Else:
        s = "no one could possible need more than 640k RAM"
```

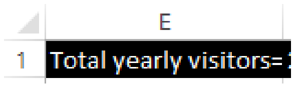
Program randomly assigns value into x, y, z (1 – 100)
Set up multiple breakpoint and mouse over variables at the breakpoints to view their contents.

- **This time x = 71, y = 54, z = 1**
- **Which branches execute**
- **What values will be assigned to the string 's**

VBA tutoria

Workbook Exercise #5: First Feature

- Writes the hard-coded text "Total yearly visitors=" to cell E1 using the Cells or Range object.

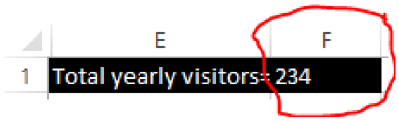


	E
1	Total yearly visitors=

VBA tutorial notes by James Tam

Workbook Exercise #5: Second Feature

- Correctly writes the total number yearly visits to Cell F1
  - Requires Feature 3A to be correct for credit to be awarded because a loop is required step through the rows in order to perform the tally.



	E	F
1	Total yearly visitors:	234

VBA tutorial notes by James Tam

Workbook Exercise #5: Second Feature

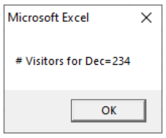
- Correctly writes the total number yearly visits to Cell F1
  - Requires Feature 3A to be correct for credit to be awarded because a loop is required step through the rows in order to perform the tally.

	E	F
1	Total yearly visitors:	234

VBA tutorial notes by James Tam

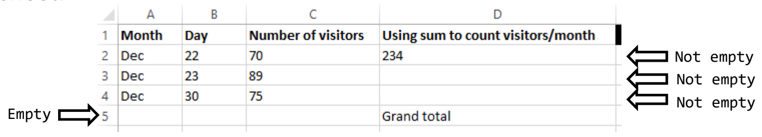
Workbook Exercise #5:Third Feature

- Correctly counts the total number of visitors for each month and displays the count using a MsgBox.
  - Example with the '2018' worksheet active



```
# Visitors for Dec=234
```

  - **Feature 3A**: Uses a loop to step through each non-empty row in the worksheet.



	A	B	C	D
1	Month	Day	Number of visitors	Using sum to count visitors/month
2	Dec	22	70	234
3	Dec	23	89	
4	Dec	30	75	
5				Grand total

VBA tutorial notes by James Tam

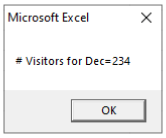
Workbook Exercise #5:Third Feature (2)

- **Feature 3B**: Uses a loop nested inside of the one written for Feature 3A to step through the visits for a month.
  - This loop repeats the process (or reruns from start to end for each month where visits have occurred).
  - In the starting spreadsheet the nested loop for Feature 3B will run 10 times from start to finish for the 10 months for the 2019 year and once for the 2018 year.
  - The number of times the nested loop runs for each of those 10 months varies depending upon the number of entries for that month e.g. Jan. 2019 it runs 4 times, Feb. 2019 it runs twice.
  - Obviously this feature requires Feature 3A (the code for the outer loop) to be complete and correct.
  - An alternative (which can also be awarded full credit) to nesting a loop within a loop for 3B is to nest a branch within loop (there is a lecture example that shows branch nested in a loop).

VBA tutorial notes by James Tam

Workbook Exercise #5:Third Feature (3)

- **Feature 3C**: The nested loop from 3B (or nested branch) is used to count and display the visits for each month via popup `MsgBox`.
- Obviously requires Feature 3B to be complete and correct.
  - Example with the 2018 worksheet active.



```
Microsoft Excel
# Visitors for Dec=234
```

- The same information to be shown in the popups is the same as results from the Excel SUM function shown in Column 'D'.
  - The information is shown in that column to help you check the results of your program.

VBA tutorial notes by James Tam

Workbook Exercise #5: Video

- Although this is a fairly simple exercise here is a run of my solution to the exercise in case you need it.
  - https://pages.cpsc.ucalgary.ca/~tamj/2022/203W/assignments/workbook_exercise5/WB_EX5.mp4

VBA tutorial notes by James Tam

Second Tutorial (Wednesday or Thursday)

Open Tutorial

- No new teaching will occur but the TA will be available for help. During this "Open Tutorial"
- Any CPSC 203 student can ask for help and not just the students who are registered in a particular tutorial.
- The purpose is to provide extra help.