VBA (Visual Basic For Applications) Programming

Overview of concepts covered in this section:
- Collections
- Find and replace
- Branching
- Looping
- Linking MS-Office documents
- Practical application of VBA

Collection

- An object that consists of other objects
  - Real World example: a book consists of pages, a library consists of books
- Example: The *Documents* collection will allow access to the documents that have been opened.
- Access to a collection rather than the individual objects may be time-saving shortcut.
  - Instead of manually closing all open documents this can be done in one instruction:
  ```
  Documents.close
  ```

Types Of Collections

- Some attributes of a document that return a collection.
 - **Lists**: access to all lists in a document (example coming up).
 - **Shapes:** access to MS-Word shapes in a document (rectangles, circles etc. example coming up).
 - **InlineShapes:** access to images inserted into a Word document (example coming up).
 - **Tables**: access to all tables in a document (detailed example coming up but a few brief examples below).
   - E.g., `ActiveDocument.Tables` –accesses all the tables in your document
   - `ActiveDocument.Tables(1)` –access to the first table in a document.
 - **Windows**: briefly introduced in the last section

The `ActiveDocument` Object

- Quick recap: although you may have many documents open, the 'active document' is the document that you are currently working with:

The active document
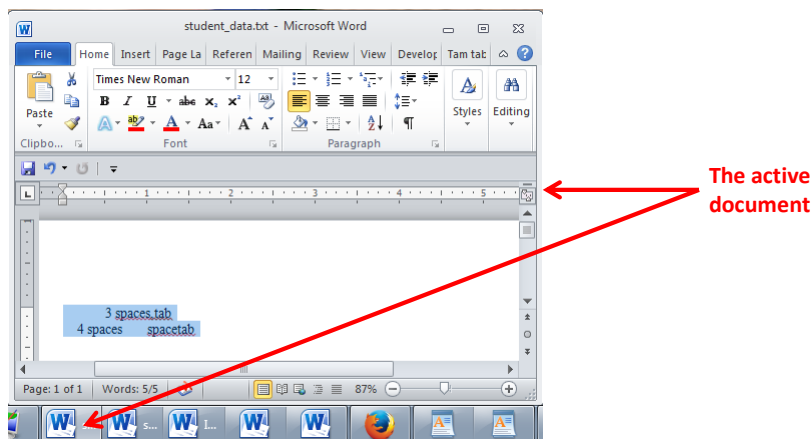
Attributes Of The `ActiveDocument` Object

- **Some of the basic attributes** of `ActiveDocument`.

  `Application:` the application/program associated with the document (useful if a VBA macro is linking several applications)

  `Name`: the name of the current document (useful for determining the active document if multiple documents are currently open).

  `Path`: the save location of the active document e.g. C:\Temp\

  `FullName`: the name and save location of the current document.

  `HasPassword`: true/false that document is password protected

  `SpellingChecked`: true/false that has been spell checked since document was last edited

  Note: Information for these attributes can be viewed by passing the information as a parameter to a message box e.g., `MsgBox` (`ActiveDocument.Name`)

Methods Of The `ActiveDocument` Object

- **Some useful methods** of `ActiveDocument`.

  Checkspelling(): exactly as it sounds!

  Close(): covered in the previous section

  CountNumberedItems(): see image (this slide)

  DeleteAllComments(): see image (this slide)

  Printout(): prints current active document on the default printer

  Save() : covered in the previous section

  SaveAs2() : rename document

  Select(): covered in the previous section

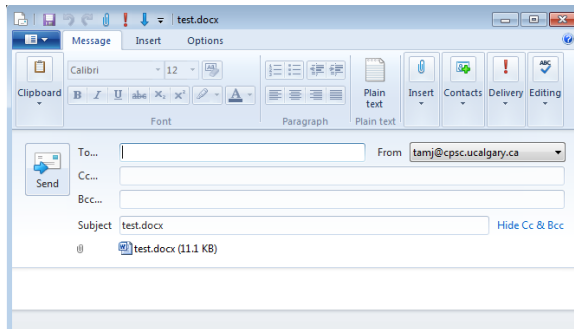 **SendMail()**: see image (next slide)

`ActiveDocument.SendMail()`

- Runs the default email program
- The active document automatically becomes an attachment
- Subject line = name of document
- (For anything more 'fancy' you should use VBA to create and access an MS-Outlook object)

"Finding" Things In A Document

- It can be done in different ways
- Example (common) '`Find`' is an object that is part of the '`Selection`' object in a document.
  - JT's note: although it may appear to be confusing at first it doesn't mean that the find (or find and replace) requires text to be selected.
  - Making '`Find`' a part of '`Selection`' was merely a design decision on the part of Microsoft.
- Example (alternative is JT's preferred approach) '`Find`' is an object that is part of the '`Content`' object of the '`ActiveDocument`'
  - `ActiveDocument.Content.Find`
  - More details coming up...

One source of information:
http://msdn.microsoft.com/en-us/library/office/aa211953(v=office.11).aspx

Single Replacement

- **Word document containing the macro**: `1simpleFind.docm`

```
sub simpleFind()
    ActiveDocument.Content.Find.Execute FindText:="tamj",ReplaceWith:="tam"
end Sub
```

```
'The instruction can be broken into two lines without causing
'An error by using an underscore as a connector
ActiveDocument.Content.Find.Execute FindText:="tamj", _
        ReplaceWith:="tam"
```

Background for example:
- My old email address (still works): tamj@cpsc.ucalgary.ca
- My new email address: tam@ucalgary.ca
- Incorrect variant: **tamj**@ucalgary.ca

More Complex Find And Replace

- **Word document containing the macro**: `findReplaceAllCaseSensitive.docm`

```
Sub findReplaceAllCaseSensitive()
    ActiveDocument.Content.Find.Execute FindText:="tamj", _
        ReplaceWith:="tam", Replace:=wdReplaceAll, _
        MatchCase:=True
End Sub
```

Before

```
TAMJ
tam
dog
tamj
tamj
cat
tamj
Tamx
Tamj
```

After

```
TAMJ
tam
dog
tam
tam
cat
tam
Tamx
Tamj
```

With, End With

<div style="background:#ffffcc">`ActiveDocument.Content.Find.Execute`</div>

- For 'deep' commands that require many levels of 'dots', the 'With', 'End With' can be a useful abbreviation.
- Example

  ```
  With ActiveDocument.Content.Find
       .Text = "tamj"
  ```
  Equivalent to (if between the 'with' and the 'end with':
  ```
       ActiveDocument.Content.Find.Text = "tamj"
  ```
- Previous example, the 'Find' employing 'With', 'End With':
- Also the search and replacement text are specified separately to shorten the 'execute' (the "ActiveDocument.Content.Find" listed once)

  ```
  With ActiveDocument.Content.Find
       .Text = "tamj"
       .Replacement.Text = "tam"
       .Execute MatchCase:=True, Replace:=wdReplaceAll
  End With
  ```

  'Find text' and 'replacement text' moved here to simplify the '.execute'

Find And Replace

- It's not just limited to looking up text.
- Font effects e.g., bold, italic etc. can also be 'found' and changed.

Finding And Replacing Bold Font

- **Word document containing the macro**: `2findBold.docm`

```
'Removes all bold text
Sub findBold()
   With ActiveDocument.Content.Find
      .Font.Bold = True
      With .Replacement
        .Font.Bold = False
      End With
      .Execute Replace:=wdReplaceAll
   End With
End Sub
```

Finding/Replacing Formatting Styles

- You may already have a set of pre-created formatting styles defined in MS-Word.
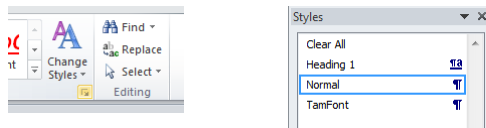
- You can redefine the characteristic of a style if you wish.
- Assume for this example that you wish to retain all existing styles and not change their characteristics.
- But you want to replace all _instances of one style_ with another style e.g., all text that is 'normal' is to become 'TamFont'
- 'Find' can be used to search (and replace) instances of a formatting style.

Finding/Replacing Formatting Styles (2)

- **Word document containing the macro**:

3findReplaceStyle.docm

```
Sub findReplaceStyle()
    With ActiveDocument.Content.Find
        .Style = "Normal"
        With .Replacement
            .Style = "TamFont"
        End With
        .Execute Replace:=wdReplaceAll
    End With
```

BEFORE

Normal style

Heading1 style
Normal style
Tam font style
Tam font style
Normal style

AFTER

Normal style

Heading1 style
Normal style
Tam font style
Tam font style
Normal style

'Normal' style becomes 'TamFont'

Recap: Programs You've Seen So Far Is Sequential Execution

- Each instruction executes from beginning to end, one after the other

Start

```
Sub TaxCalculator()
    Const TAX_RATE = 0.25
    Dim GrossIncome As Double
    Dim Tax As Double
    Dim NetIncome As Double
    GrossIncome = InputBox("Enter your income: ")
    Tax = GrossIncome * TAX_RATE
    NetIncome = GrossIncome - Tax
    MsgBox ("Gross Income $" & GrossIncome & ", Net Income $" & NetIncome)
End Sub
```
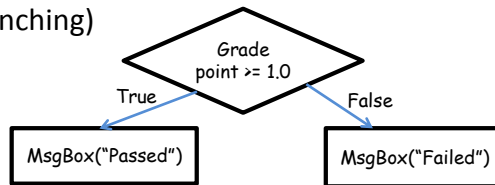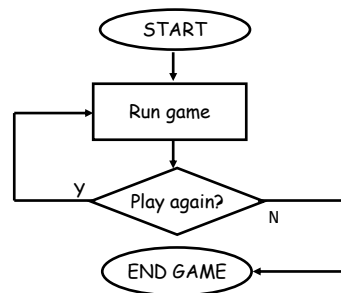
End

- When the last instruction is reached then the program ends

New Program Writing Concepts (Non-Sequential)

- Making decisions (branching)

- Looping (repetition)

New Terminology

- **Boolean expression**: An expression that must work out (evaluate to) to either a true or false value.
  - e.g., it is over 45 Celsius today
  - e.g., the user correctly entered the password

- **Body**: A block of program instructions that will execute under a specified condition.

```
Sub Document_Open()
    MsgBox ("Fake virus!")
End Sub
```

This/these instruction/instructions run when you tell VBA to run the macro, the 'body' of the macro program

  - Style requirement
    - The 'body' is indented (4 spaces)
    - A "sub-body" is indented by an additional 4 spaces (8 or more spaces)

Branching: Making Decisions In A Program

- Similar to the Excel (IF-Function): Check if some condition has been met (e.g., password for the document correctly entered): Boolean expression
- But the IF-Construct employed with programming languages is not just a function that returns a value for the true or false cases.
  - For each condition: a statement or a collection of statements can be executed (this is referred to as "the body" of the if or else case.
  - Example: entering a password
    - Boolean expression true, password matches:
      - True case body: display confirmation message and run program
    - Boolean expression false, password doesn't match:
      - False case body: display error message

## Branching/Decision Making Mechanisms

- If-Then
- If-Then, Else   **Similar to Excel if-then**
- If-Then, ElseIf, Else **Similar to Excel nested if's**

Allowable **Operators** For Boolean Expressions

```
if (value operator value) then
```

VBA operator	Mathematical equivalent	Meaning	Example
<	<	Less than	5 < 3
>	>	Greater than	5 > 3
=	=	Equal to	5 = 3
<=	≤	Less than or equal to	5 <= 5
>=	≥	Greater than or equal to	5 >= 4
<>	≠	Not equal to	x <> 5

Decision Making With '**If-Then**'
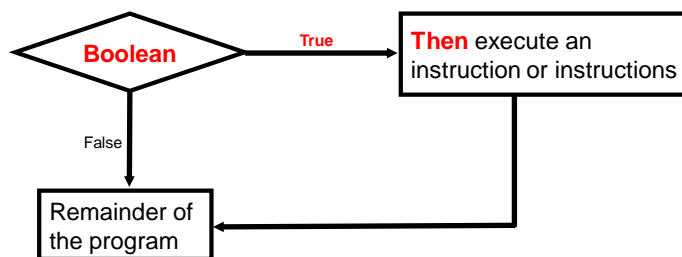
If-Then

- **Format**:
  ```
  If (Boolean expression) Then
      If-Body
  End if
  ```

- **Example**:
  ```
  If (totalWords < MIN_SIZE) Then
      MsgBox ("Document too short, total words " &
        totalWords)
  End If
  ```

If-Then: Complete Example

- **Word document containing the macro**: 4wordCount.docm
  ```
  ' Try deleting all the words in the Word doc and run the
  ' macro again
  Sub wordCount()
      Dim totalWords As Integer
      MIN_SIZE = 100
      totalWords = ActiveDocument.Words.Count
      If (totalWords < MIN_SIZE) Then
          MsgBox ("Document too short, total words " &
            totalWords)
      End If
  End Sub
  ```

Decision Making With An 'If, Else'

- Used when different Actions (separate bodies) are required for the true result (IF-case) vs. the false result (ELSE-case)

Decision Making With An 'If, Else'
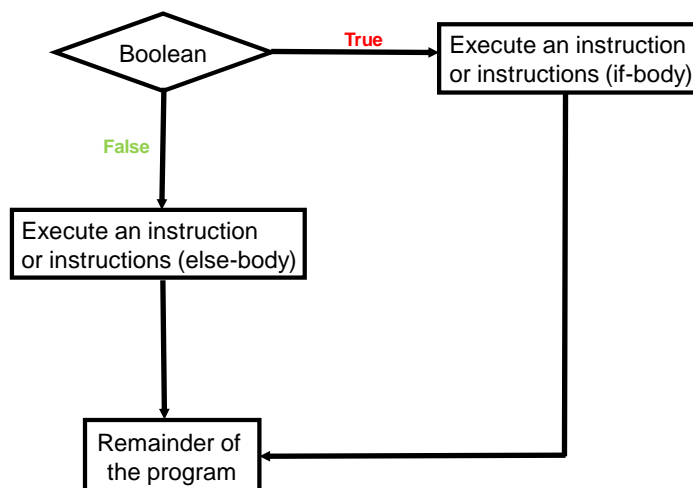
If-Then **(True)**, Else **(False)**

- **Format**:

```
If (Boolean expression) Then
    If-Body
Else
    Else-Body
End if
```

- **Example**:

```
If (totalWords < MIN_SIZE) Then
    MsgBox ("Document too short, total words " & totalWords)
Else
    MsgBox ("Document meets min. length requirements")
End If
```

If-Then, Else: Complete Example

- **Word document containing the macro**: 5wordCountV2.docm

```
' Try deleting words or changing the minimum size and observe
' the effect on the program.
Sub wordCountV2()
    Dim totalWords As Integer
    MIN_SIZE = 1000
    totalWords = ActiveDocument.Words.Count
    If (totalWords < MIN_SIZE) Then
        MsgBox ("Document too short, total words " &
            totalWords)
    Else
        MsgBox ("Document meets min. length requirements")
    End If
End Sub
```

## Logic Can Be Used In Conjunction With Branching

- Typically the logical operators AND, OR are used with multiple conditions/Boolean expressions:
  - If multiple conditions *must all be met* before the body will execute. (And)
  - If *at least one condition* must be met before the body will execute. (Or)
- The logical NOT operator can be used to check if something has 'not' occurred yet
  - E.g., If it's true that the user *did not* enter the correct password then the program will end.

Logic: The "**OR**" Operator

- **Format:**

```
If (Boolean expression) OR (Boolean expression) then
    body
End if
```

- **Word document containing the macro (empty document, see macro editor for the important details)**: 6if_or_hiring.docm

```
gpa = InputBox("Grade point: ")
experience = InputBox("Years of job experience: ")
If (gpa > 3.7) Or (experience > 5) Then
    result = "Hire applicant"
Else
    result = "Insufficient qualifications"
```
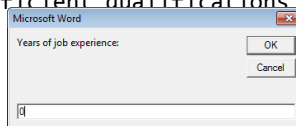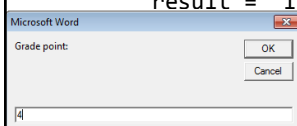
Hiring Example: Example Inputs & Results

```
If (gpa > 3.7) Or (experience > 5) then
```

GPA	Years job experience	Result
2	*0*	*Insufficient qualifications*
1	10	Hire
4	1	Hire
4	7	Hire

Logic: The "**AND**" Operator

- **Format:**
  ```
  If (Boolean expression) And (Boolean expression) then
      body
  End if
  ```
- **Word document containing the macro (empty document, see macro editor for the important details)**: `7if_and_firing.docm`
  ```
  salary = InputBox("Salary: ")
  years = InputBox("Years of employment: ")
  If (salary >= 100000) And (years < 2) Then
      result = "Fired!"
  Else
      result = "Retained"
  ```
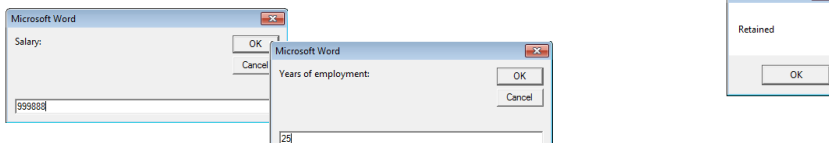
Firing Example: Example Inputs & Results

```
If (salary >= 100000) And (years < 2) Then
```

Salary	Years on job	Result
1	100	Retained
50000	1	Retained
123456	20	Retained
1000000	*0*	*Fired!*

Logic: The "**NOT**" Operator

- **Format**:
  ```
  If Not (Boolean Expression) then
        body
  End if
  ```

- **Word document containing the macro example**:
  ```
  8checkSave.docm
    If Not (ActiveDocument.Saved) Then
        MsgBox ("You haven't saved " & ActiveDocument.Name
          & " yet")
    End If
  ```

Line Continuation Character

- To increase readability long statements can be split over multiple lines.

```
If (income  > 99999) And _
   (experience <=  2) And _
   (numRepramands > 0) Then
      MsgBox ("You're fired!")
End If
```

- To split the line the line continuation character (underscore) must be preceded by a space.

- Keywords cannot be split between lines e.g.

```
Msg _
Box
```

For more details see: http://support.microsoft.com/kb/141513

Line Continuation Character (2)

- Strings split over multiple lines require a combination of the proper use of the **line continuation character** '**_**' and the **concatenation operator** '**&**':

```
MsgBox ("Your " _
      & "name")
```

What To Do When Multiple Conditions Must Be Checked

- **Case 1**: If each condition is independent of other questions
  - Multiple `if-then` expressions can be used
  - Example:
  - Q1: Are you an adult?
  - Q2: Are you a Canadian citizen?
  - Q3: Are you currently employed?

What To Do When Multiple Conditions Must Be Checked (2)

- **Case 2 (mutually exclusive)**: If the result of one condition affects other conditions (when one condition is true then the other conditions cannot be true)
  - `If-then`, `elseif`, `else` should be used
  - Which of the following is your place of birth? (Answering true to one option makes the options false)
    a) Calgary
    b) Edmonton
    c) Lethbridge
    d) Red Deer
    e) None of the above

Decision Making With **Multiple If-Then's**

Each question is independent (previous answers have no effect on later questions because all questions will be asked).
- Q1: Are you an adult?
- Q2: Are you a Canadian citizen?
- Q3: Are you currently employed?
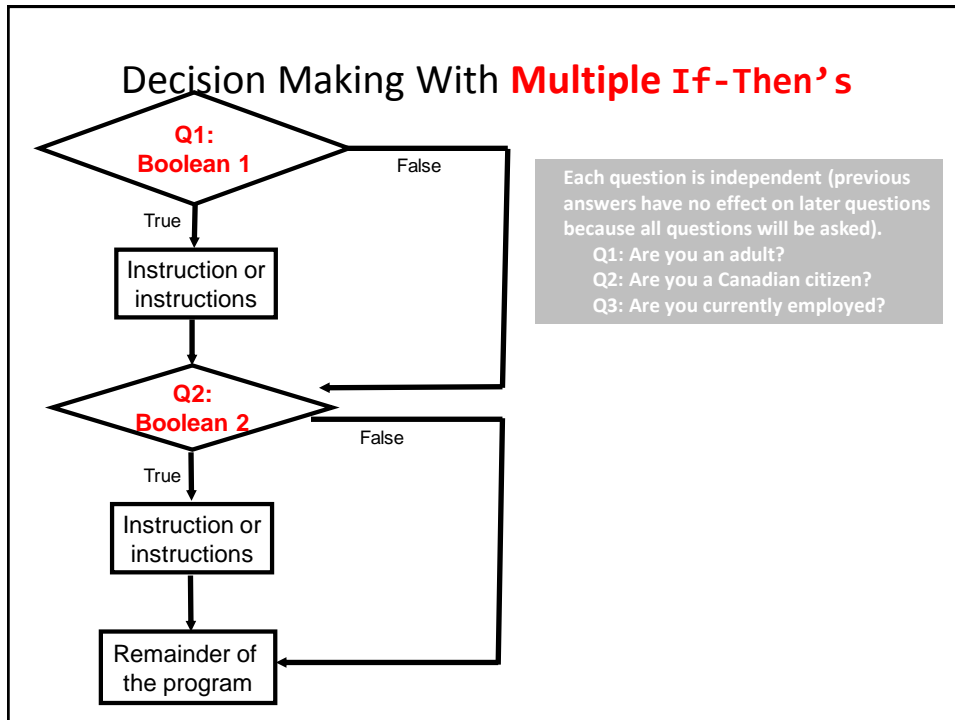
Multiple If-Then's

- Any, all or none of the conditions may be true
- Employ when a series of independent questions will be asked
- **Format:**

```
if (Boolean expression 1) then
     body 1
end if
if (Boolean expression 2) then
     body 2
end if
  ...
statements after the conditions
```

Multiple `If-Then's` (2)

- **Word document containing the macro:** `9multipleIfs.docm`

```vba
Sub multipleIf()
  ' Check if there were any 'comments' added to the document.
    If (ActiveDocument.Comments.Count > 0) Then
        MsgBox ("Annotations were made in this document")
    End If
  ' A numbered item includes numbered and bulleted lists.
    If (ActiveDocument.CountNumberedItems() > 0) Then
        MsgBox ("Bullet points or numbered lists used")
    End If
End Sub
```

- Bull1
- Bull2

Some text in a document.

Comment [JT1]: Replace 'text' with another word

Multiple `If's`: Mutually Exclusive Conditions

- At most *only one* of many conditions can be true
- Can be implemented through multiple `if`'s

Inefficient combination!

- **Word document containing the macro (empty document, see macro editor for the important details)**: "10gradesInefficient.docm"

```vba
If (grade = 4) Then          If (grade = 1) Then
    letter = "A"                 letter = "D"
End If                       End If
If (grade = 3) Then          If (grade = 0) Then
    letter = "B"                 letter = "F"
End If                       End If
If (grade = 2) Then
    letter = "C"
End If
```

Decision Making With `If-Then, Elseif, Else`

JT's note: once the first 'true' case is encountered all remaining and related Boolean expressions (using 'Elseif') are skipped

---

Multiple **If-Elif-Else**: Use With Mutually Exclusive Conditions

- **Format:**

```
if (Boolean expression 1) then
     body 1
elseif (Boolean expression 2) then
     body 2
       ...
else
     body n
' Only one 'end-if' at very end
end if
statements after the conditions
```

Mutually exclusive
- One condition evaluating to true excludes other conditions from being true
- Example: having your current location as 'Calgary' excludes the possibility of the current location as 'Edmonton', 'Toronto', 'Medicine Hat'

If-Elseif-Else: Mutually Exclusive Conditions (Example)

- **Word document containing the macro (empty document, see macro editor for the important details)**: "11gradesEfficient.docm"

```
If (grade = 4) Then
    letter = "A"
ElseIf (grade = 3) Then
    letter = "B"
ElseIf (grade = 2) Then
    letter = "C"
ElseIf (grade = 1) Then
    letter = "D"
ElseIf (grade = 0) Then
    letter = "F"
Else
    letter = "Invalid"
End If
```

This approach is more efficient when at most only one condition can be true.

Extra benefit:

The body of the else executes only when all the Boolean expressions are false. (Useful for error checking/handling).

Location Of The "End If": Multiple If's

- Independent If-then's:
  - Since each 'if' is independent each body must be followed by it's own separate 'end if'

```
grade = InputBox("Enter grade point: ")
If (grade = 4) Then
    letter = "A"
End If
If (grade = 3) Then
    letter = "B"
End If
If (grade = 2) Then
    letter = "C"
End If
If (grade = 1) Then
    letter = "D"
End If
If (grade = 0) Then
```

Location Of The "End If": If-then, Else

- If-then, Else:
  - Since the 'if-then' and the 'else' are dependent (either one body or the other must execute) the 'end if' must follow the body of the 'else-body' (last dependent "if-branch")

```
If (totalWords < MIN_SIZE) Then
    MsgBox ("Document too short, total wc
Else
    MsgBox ("Document meets min. length r
End If
```

Document either does or does not have enough words

Location Of The "End If": If-Then, ElseIf

- Dependent If-then, Else-If:
  - Since the results of earlier Boolean expressions determine whether later ones can be true (reminder: because at most only one can be true) all of the if-then and Elseif expressions are dependent (one related block).
  - The "end if" belongs at the very end of the block

```
If (grade = 4) Then
    letter = "A"
ElseIf (grade = 3) Then
    letter = "B"
ElseIf (grade = 2) Then
    letter = "C"
ElseIf (grade = 1) Then
    letter = "D"
ElseIf (grade = 0) Then
    letter = "F"
Else
    letter = "Invalid"
End If
MsgBox ("GPA=" & grade &
```

VBA: `If,` `Else-If` And **Excel**: `Nested-Ifs`

- These two concepts are comparable:

VBA:
```
If (grade = 4) Then
    letter = "A"
ElseIf (grade = 3) Then
    letter = "B"
ElseIf (grade = 2) Then
    letter = "C"
ElseIf (grade = 1) Then
    letter = "D"
ElseIf (grade = 0) Then
    letter = "F"
Else
    letter = "Invalid"
End If
```

Excel (display different messages for different grade points e.g. Display "Perfect" if grade point is 4.0 or greater):
```
=IF(D2>=4,"Perfect",
   IF(D2>=3,"Excellent",
      IF(D2>=2,"Adequate",
         IF(D2>=1,"Pass",
            "Fail"))))
```

---

## VBA (Programming Language): **Nested-IFs**

- Similar to the `IF, ELSE-IF`: Decision making is dependent.
  - One branch is 'nested' inside of another branch
  - The first decision must evaluate to true ("gate keeper") before successive decisions are even considered for evaluation.
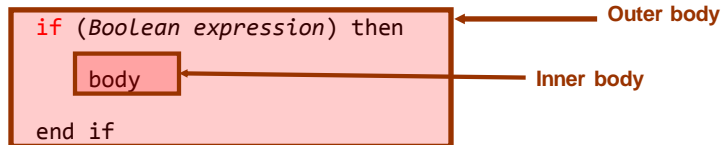- Unlike the `IF, ELSE-IF` more than one case can be true:
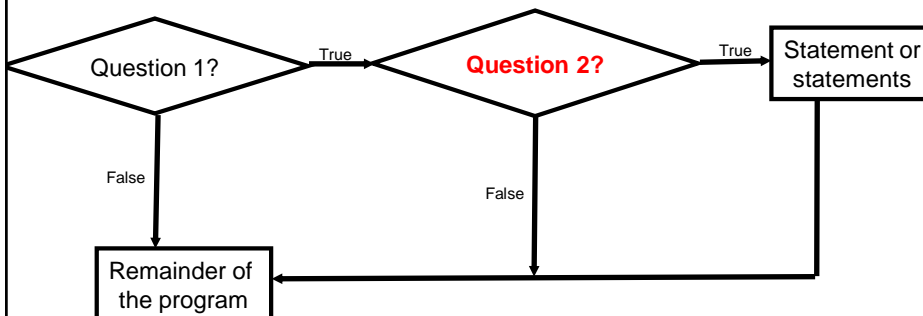
## Nested Decision Making

- One decision is made inside another.
- Outer decisions must evaluate to true before inner decisions are even considered for evaluation.
- **Format:**

```
if (Boolean expression) then
```


- Both (or all if 2+ IFs) must evaluate to true before the inner-most body will execute.

Nested IFs: Simple 'Toy' Example

- **Word document containing the macro (empty document, see macro editor for the important details)**:
"12simpleNesting.docm"

```
' Suggested inputs
' num1=1, num2=1
' num1=1, num2=0
' num1=0, num2=1


Sub simpleNestingExample()
    Dim num1 As Long
    Dim num2 As Long
    Dim result As String
```

Nested `IF`s: Simple 'Toy' Example (2)

```
    num1 = InputBox("Enter a whole number")
    num2 = InputBox("Enter a whole number")
    If (num1 > 0) Then
        result = "Num1 positive - "
        If (num2 > 0) Then
            result = result & "Num2 positive"
        End If
    Else
        result = "Num1 not positive, didn't bother checking
                  num2"
    End If
    MsgBox (result)
End Sub
```

Nesting: `End-If`

```
If (num1 > 0) Then
      result = "Num1 positive - "
      If (num2 > 0) Then
          result = result & "Num2 positive"
      End If
Else
    result = "Num1 not positive, didn't bother checking
              num2"
End If
```

Multiple `IFs`: Distinguishes All Cases

- **Word document containing the macro (although you've seen multiple-Ifs this example is include to contrast vs. nesting)**: "13nonNestedExample.docm"

```
' Suggested inputs
' num1=0, num2=1
' num1=1, num2=0
' num1=1, num2=1
' num1=0, num2-0

Sub nonNestedExample()
    Dim num1 As Long
    Dim num2 As Long
    Dim result As String
```

Multiple `IFs`: Distinguishes All Cases (2)

```
    num2 = InputBox("Enter a whole number")
    If (num1 > 0) Then
        result = "Num1 positive - "
    End If
    If (num2 > 0) Then
        result = result & "Num2 positive"
    End If
    MsgBox (result)

End Sub
```

From The Last Section: What If there Is No Selection?

- The VBA program attempts to format the currently selected text but there is 'no' selection.

  ```
  Selection.Font.Bold = wdToggle
  ```

  – What modifications can be made to the program to handle this case?

The `Selection Object` again

- With a previous example if no text was selected then the program would produce no visible effect.

  ```
  Sub SelectedFontChange()
      Selection.Font.Bold = wdToggle
  End
  ```

- Another example automatically selected text for you "expanded" the selection.

  ```
  Sub AutoSelectedFontChange()
      Selection.Expand
      Selection.Font.Bold = wdToggle
  End Sub
  ```

Before

Much research has been conducted in
collaborative projects (e.g., Neuwirth, Ch

After

Much research has been conducted int
collaborative projects (e.g., Neuwirth, Chan

Constants For The Selection Object

Name of constant	Meaning of constant
wdSelectionIP	No text selected
wdSelectionNormal	Text (e.g., word, sentence) has been selected
wdSelectionShape	A graphical shape (e.g., circle, text book) has been selected

The Selection Object And A Practical Application Of Branching

- An example application of branching: check if a selection has been made and only apply the selection if that is the case.
  - Checking if a condition is true
- **Word document containing the macro:**
  "14selectionExample.docm"

```
Sub checkSelection()
    If Selection.Type = wdSelectionIP Then
        MsgBox ("No text selected, nothing to change")
    Else
        Selection.Font.Bold = wdToggle  'wdToggle, constant
    End If
End Sub
```

Application Branching: Marking Program

- **Word document containing the macro: "**15Marking program.docm**"**

- Synopsis:
  - The program spells checks the document
    - Assume each document includes the name of the person in the file name
  - If the number of errors meets a cut-off value then it's a 'fail'
  - Otherwise it's a pass
  - The feedback is 'written' to the beginning of the document using a specific font and several font effects in order to stand out
    - The message is customized with the person's name at the beginning of the feedback

Marking Program

```
Sub MarkingForSpelling()
    Dim totalTypos As Integer
    Const MAX_TYPOS = 30
    Dim currentDocument As String
    Dim feedback As String

    'Get Name of current document
    currentDocument = ActiveDocument.Name

    'Tally the number of typos
    totalTypos = ActiveDocument.SpellingErrors.Count

    'Feedback is prefaced by student(document) name
    feedback = currentDocument & " marking feedback..."
```

Marking Program (2)

```
' HomeKey move to the home position (start of document)
Selection.HomeKey Unit:=wdStory

'Recall: before this feedback just = document name and
'an indication that feedback is coming
If (totalTypos > MAX_TYPOS) Then
    feedback = feedback & ": Too many typographical errors:
                             Fail"
Else
    feedback = feedback & ": Pass"

End If

' Chr(11) adds a newline (enter) to the end of feedback
feedback = feedback & Chr(11) & Chr(11)

' Alternative use the constant vbCr (VB cursor return)
```
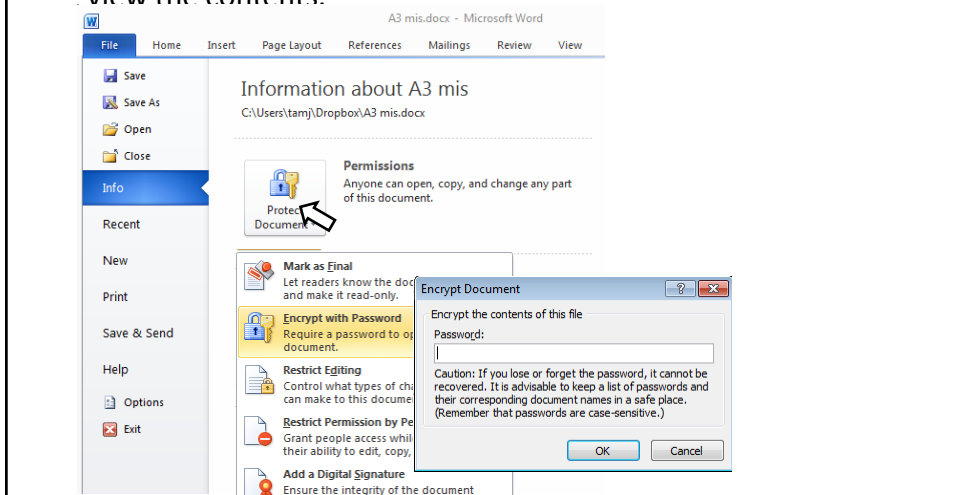
Marking Program (3)

```
' Font effects to make the feedback stand out
Selection.Font.ColorIndex = wdRed
Selection.Font.Size = 16
Selection.Font.Name = "Times New Roman"

' Write feedback into the document
Selection.TypeText (feedback)

End Sub
```

Securing A Document: Using MS-Word

- Documents can be configured so a password is required to view the contents.

Securing A Document: Simple VBA Example

- **Word document containing the macro:**
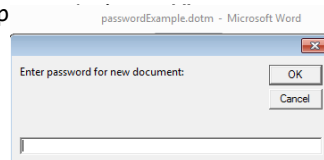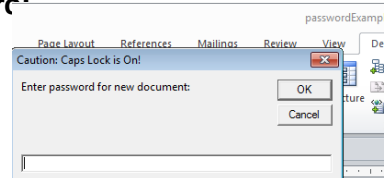 16passwordExample.docm

```vba
Sub passWordExample()
    Dim yourPassword As String
    Dim warningCaps As String

    If (Application.CapsLock = True) Then
        warningCaps = "Caution: Cap
    Else
        warningCaps = ""
    End If

    yourPassword = InputBox("Password for document: " _
        & warningCaps)
    ActiveDocument.Password = yourPassword
End Sub
```
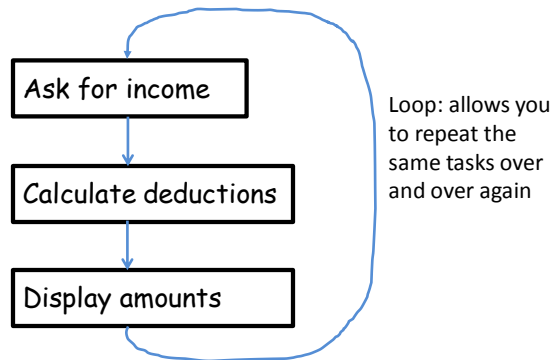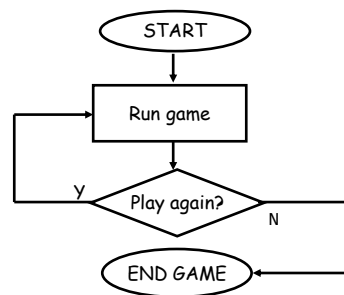
Looping/Repetition

- How to get the program or portions of the program to automatically re-run
  - Without duplicating the instructions
  - Example: you need to calculate tax for multiple people

Ask for income

Calculate deductions

Display amounts

Loop: allows you to repeat the same tasks over and over again

Looping/Repetition (2)

- The entire program repeats

Play game again?

www.colourbox.com

START

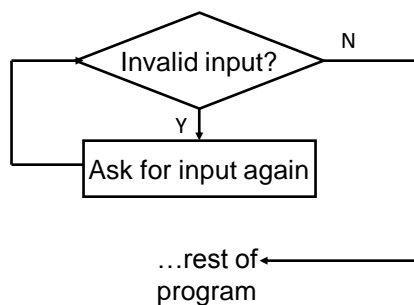Run game

Play again?

y

N

END GAME

Looping/Repetition (3)

- Only a specific part of the program repeats

```
Enter your age (must be non-negative):  -1
Enter your age (must be non-negative):  27
Enter your gender (m/f): ▮
```

Re-running specific parts of the program

Flowchart

Types Of Loops

- Fixed repetition loops: runs some integer 'n' times e.g., generates taxes for 10 clients
  - For-next
- Variable repetition loops: runs as long as some condition holds true
  - e.g., while the user doesn't quit the program re-run the program
  - e.g., while the user enters an erroneous value ask the user for input.
  - Do-while loop

For-Next Loops

- A 'counting' loop: counts out a sequence of numbers
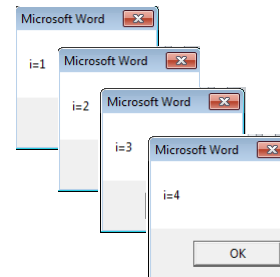- **Format**:

```
For <counter> = <start> To <end> Step <step size>[1]
    <Statement(s)>
Next <counter>
```

- **Complete Example**: "17forUpOne.docm"

```
' Synopsis of code illustrating new concepts
Dim i As Integer

For i = 1 To 4 Step 1
      MsgBox ("i=" & i)
Next i
```

Microsoft Word

i=1   Microsoft Word

i=2   Microsoft Word

i=3   Microsoft Word

i=4

OK

[1] Step size can be a positive or negative integer e.g., 1, -1, 5, -10 etc.

For-Next Loops (2)

- For-next loops can count down as well as up
- The Steps can be values other than one.
- **Complete example**: "18forDownThree.docm"

```
' Synopsis of code illustrating new concepts
Dim i As Integer
For i = 12 To 0 Step -3
    MsgBox ("i=" & i)
Next i
```

```
12
9
6
3
0
```

Do-While Loop

```
Start
Condition?  F
T
Statements
Loop
End
```

- **Format**:
  ```
  Do While <Condition>
      <Statement(s)>
  Loop
  ```

- **Example**: "19whileUpOne.docm"
  ```
  Dim i As Integer
  i = 1
  Do While (i <= 4)
      MsgBox ("i=" & i)
      i = i + 1
  Loop
  ```
  Any valid mathematical expression here

```
Microsoft Word
i=1
Microsoft Word
i=2
Microsoft Word
i=3
Microsoft Word
i=4
OK
```

Programming Style: Variable Names

- In general variable names should be self-descriptive e.g., 'age', 'height' etc.
- Loop control variables are an exception e.g., 'i' is an acceptable variable name
  - It's sometimes difficult to come up with a decent loop control name
  - Loop control variables are given shorter names so the line length of a loop isn't excessive
  ```
  Dim loopControl As Integer
  loopControl = 1
  Do While (loopControl <= 4)
      ...
  ```

Loops That Never Execute

- **Word document containing the complete program**:
  20nonExecutingLoops.docm

```
Dim i As Integer

i = 5
Do While (i <= 4)
    MsgBox ("i=" & i)
    i = i + 1
Loop


For i = 4 To 1 Step 1
    MsgBox ("i=" & i)
Next i
```

Exercise #1: Loops

- The following program that prompts for and displays the user's age.

```
Sub errorChecking()
    Dim age As Long
    age = InputBox("Age (greater than or equal to zero)")
End Sub
```

- Modifications:
  - As long as the user enters a negative age the program will continue prompting for age.
  - After a valid age has been entered then stop the prompts and display the age.
  - Hint: Use a do-while loop (not a for-loop)

Accessing Tables

- The tables in the currently active Word document can be made through the `ActiveDocument` object:
  - `ActiveDocument.Tables:` accesses the 'tables' collection (all the tables in the document).
  - `ActiveDocument.Tables(<integer 'i'>):` accesses table *# i* in the document
  - `ActiveDocument.Tables(1).Sort:` sorts the first table in the document (default is ascending order)

Simple Example: Sorting Three Tables

- Instructions needed for sorting 3 tables

```
ActiveDocument.Tables(1).Sort
ActiveDocument.Tables(2).Sort
ActiveDocument.Tables(3).Sort
```

Before

Morris, Heather
Cartwright, Douglas
Wolf, Claudia
Smith, Vincent

Sing, Han
Roth, Vincent
Lung, Tong

Yen, Donnie
Hung, Lynn
Huang, Xiaoming
Shahlavi, Darren

After

Cartwright, Douglas
Morris, Heather
Smith, Vincent
Wolf, Claudia

Lung, Tong
Roth, Vincent
Sing, Han

Huang, Xiaoming
Hung, Lynn
Shahlavi, Darren
Yen, Donnie

Previous Example

- Critique of the previous approach: the program 'worked' for the one document with3 tables but:
  - What if there were more tables (cut and paste of the sort instruction is wasteful)?
  - What if the number of tables can change (i.e., user edits the document)
- Notice: The process of sorting just repeats the same action but on a different table.
  ```
  ActiveDocument.Tables(1).Sort
  ActiveDocument.Tables(2).Sort
  ActiveDocument.Tables(3).Sort
  ```
- Looping/repetition can be applied reduce the duplicated statements

---

Revised Example: Sorting Tables With A Loop

Word document containing the complete macro:
"21sortingTables.docm"

```
Sub Sort()
    Dim CurrentTable As Integer
    Dim NumTables As Integer
    NumTables = ActiveDocument.Tables.Count
    If NumTables = 0 Then
        MsgBox ("No tables to sort")
    Else
        For CurrentTable = 1 To NumTables Step 1
            MsgBox ("Sorting Table # " & CurrentTable)
            ActiveDocument.Tables(CurrentTable).Sort
        Next
    End If
End Sub
```

Result: Sorting Tables

- **Before**

A
B
c

Z
B
a

Morris Heather	Heroine
Adama, Lee	CAG
Adama, Bill	Commander

- **After**

A
B
c

a
B
Z

Adama, Bill	Commander
Adama, Lee	CAG
Morris Heather	Heroine

More On Sort

- A **handy parameter** that can be used to configure how it runs.
- **Format**

  Sort (*<Boolean to Exclude header – True or False>*)

- **Example**
  - ActiveDocument.Tables(CurrentTable).Sort(**True**)

  - Before

Name	Title
Tam, James	Boring
Bond, James	Spy

  - After

Name	Title
Bond, James	Spy
Tam, James	Boring

Second Sorting Example: **Exclude Headers**

- **Document containing the macro**:
  "22sortingTablesExcludeHeader.docm"

```
Sub Sort()
    Dim CurrentTable As Integer
    Dim NumTables As Integer
    NumTables = ActiveDocument.Tables.Count
    If NumTables = 0 Then
        ' Don't bother sorting
        MsgBox ("No tables to sort")
    Else
        For CurrentTable = 1 To NumTables Step 1
            MsgBox ("Sorting Table # " & CurrentTable)
            ActiveDocument.Tables(CurrentTable).Sort (True)
        Next
    End If
End Sub
```

Before

NX-01 crew
Kirk, James Tam
Tam, James
Sheen, Charlie
Bond, James

After

NX-01 crew
Bond, James
Kirk, James Tam
Sheen, Charlie
Tam, James

Accessing Shapes And Images

- Reminders (in VBA)
  - Shapes (basic shapes that are drawn by Word) ■ ★ ▲
  - InlineShapes (images that are created externally and inserted into Word)
- Both collections accessed via the ActiveDocument object:
  - ActiveDocument.Shapes: access to all the shapes in the currently active Word document
    - ActiveDocument.Shapes(<*index*>): access to shape #*i* in the document
  - ActiveDocument.InlineShapes: access to all the images in the currently active Word document
    - ActiveDocument.InlineShapes(<*index*>): access to image #*i* in the document

Example: Accessing Shapes And Images

Word document containing the complete macro:
"23accessingImagesFigures.docm"

```
Dim numImages As Integer
Dim numShapes As Integer

numImages = ActiveDocument.InlineShapes.Count
numShapes = ActiveDocument.Shapes.Count

MsgBox ("Images=" & numImages)
MsgBox ("Simple shapes=" & numShapes)
```

Example: Accessing Shapes And Images (2)

```
' Checks expected # images and alters first & third
If (numImages = 4) Then
    ActiveDocument.InlineShapes(1).Height = _
      ActiveDocument.InlineShapes(1).Height * 2
    ActiveDocument.InlineShapes(3).Height = _
      ActiveDocument.InlineShapes(3).Height * 2
End If

' Checks expected # shapes, alters 2nd & 6th
' Deletes the first shape
If (numShapes = 6) Then
    ActiveDocument.Shapes(2).Width = _
      ActiveDocument.Shapes(2).Width * 4
    ActiveDocument.Shapes(6).Fill.ForeColor = vbRed
    ActiveDocument.Shapes(1).Delete
End If
```

Printing: Multiple

- Printing all the documents currently open in MS-Word.
  - Take care that you don't run this macro if you have many documents open and/or they are very large!
  - **Word document containing the macro example:** "24multiDocumentPrint.docm"

```
Sub PrintDocumentsCollection()
    Dim numDocuments As Integer
    Dim count As Integer
    numDocuments = Documents.count
    count = 1
    Do While (count <= numDocuments)
        Documents.Item(count).PrintOut
        count = count + 1
    Loop
End Sub
```

> Learning: another practical application of looping e.g., automatically open multiple documents, make changes, print and save them without user action needed

Nesting

- Nesting: one construct is contained within another
  - What you have seen: nested branches:

```
If (Boolean) then
    If (Boolean) then
    End If
End if
```

- Branches and loops can be nested within each other

```
Do while (Boolean)
    if (Boolean) then
    End if
Loop
```

Example: Nesting

1. Write a program that will count out all the numbers from one to six.
2. For each of the numbers in this sequence the program will determine if the current count (1 – 6) is odd or even.
   a) The program display the value of the current count as well an indication whether it is odd or even.

- Which Step (#1 or #2) should be completed first?

Step #1 Completed: Now What?

- For each number in the sequence determine if it is odd or even.
- This can be done with the modulo (remainder) operator: MOD
  – An even number modulo 2 equals zero (2, 4, 6 etc. even divide into 2 and yield a remainder or modulo of zero).
  – `If (counter MOD 2 = 0) then 'Even`
  – An odd number modulo 2 does not equal zero (1, 3, 5, etc.)
- Pseudo code visualization of the problem
  Loop to count from 1 to 6
       Determine if number is odd/even and display message
  End Loop
  – Determining whether a number is odd/even is a part of counting through the sequence from 1 – 6, checking odd/even is nested within the loop

The DIR Function

- It can be used to go through all the documents in a folder (this course will illustrate in advanced examples)
- It can be used to go through the entire contents of a folder including sub-folders and sub-sub folders (very advanced use: well beyond the scope of the this course)
- Basic use: this function takes a location (e.g., C:\temp\) and a filename as an argument and it determines if the file exists at the specified location.
  - If the file is found at this location then the function returns the name of the file.
  - If the file is not found at this location then the function returns an empty string (zero length)

Simple Use Of The DIR Function

- **Word document containing the macro example:**
  25DIRFunctionSimple.docm

```
Dim location As String
Dim filename As String
Dim result As String
location = "C:\temp\" 'Always look here

filename = "Doc1.docx"   'C:\temp\Doc1.dox
result = Dir(location & filename)
MsgBox (result)

result = Dir(location & "*.docx") 'Any .docx in C:\temp\
MsgBox (result)

filename = InputBox("File name in C:\temp")
result = Dir(location & filename)
MsgBox (result)
```

Example: Using Dir To Check If File Exists (2)

- **Word document containing the macro example:**
  26DIRFunctionIntermediate.docm

```
Sub openExistingDocument()
    Dim filename As String
    Dim checkIfExists As String
    Dim last As Integer

    filename = InputBox ("Enter the path and name of file to
        open e.g., 'C:\temp\tam.docx'")
    ' Error case: nothing to open, user entered no info
    If (filename = "") Then
        ActiveDocument.ActiveWindow.Caption =
            "No such file in this location"
```

Example: Using Dir To Check If File Exists (3)

```
    ' No error: non-empty info entered
    Else
        checkIfExists = Dir(filename)
        If (Len(checkIfExists) = 0) Then
            MsgBox ("File doesn't exist can't open")
        Else
            MsgBox ("File exists opening")
            Documents.Open (filename)
        End If
    End If
End Sub
```

Practical Use Of `Dir`: Access Each File In A Directory

- **Word document containing the macro example:**
  `27loopFolder.docm`

```
Sub loopFolder ()
    Dim directoryPath As String
    Dim currentFile As String
    directoryPath = InputBox("Enter full path of search
      folder e.g. C:\Temp\")
    currentFile = Dir(directoryPath)
    If (currentFile = "") Then
        MsgBox ("No documents in the specified folder")
    End If
    Do While (currentFile <> "")
        MsgBox (currentFile) ' Display file name in popup
        currentFile = Dir
    Loop
End Sub
```

Alternate Version: Access Only **Word Documents**

- **Word document containing the macro example:**
  `28loopWordFolder.docm`

```
Sub loopWordFolder()
    Dim directoryPath As String
    Dim currentFile As String
    directoryPath = InputBox("Enter full path of search
     folder")
    currentFile = Dir(directoryPath & "*.doc*")
    If (currentFile = "") Then
        MsgBox ("No documents in the specified folder")
    End If
    Do While (currentFile <> "")
        MsgBox (currentFile) ' Display file name in popup
        currentFile = Dir  ' Move onto next document in folder
     Loop
End Sub
```
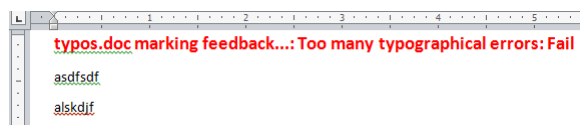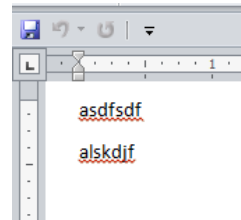
Revision Of An Earlier Example

- The original version only marked (and annotated at the top) a single document.
- This new version will automatically mark all the documents in a user-specified folder and insert the marking information at the top of each document.
- Details:
  - Open each document in the folder
  - Run a spell check of the document
  - Based on the number of spelling mistakes the document will be marked as either a pass or fail
  - The comments will be inserted at the top of the document
  - The marked document is then automatically closed and the program moves onto the next document until there are no more documents in that folder.

Revised Marking Program

- **Word document containing the macro**: "29markAllFolderDocuments.docm"

```
Sub markAllFolderDocuments()
    Const MAX_TYPOS = 1
    Const LARGER_FONT = 14
    Dim directoryPath As String
    Dim currentFile As String
    Dim totalTypos As Integer
    Dim feedback As String
```


asdfsdf
alskdjf

typos.doc marking feedback...: Too many typographical errors: Fail

asdfsdf
alskdjf

Revised Marking Program (2)

```
directoryPath = InputBox("Location and name of folder
  containing assignments (e.g., C:\grades\")
currentFile = Dir(directoryPath & "*.doc*")

If (directoryPath = "") Then
    MsgBox ("No Word documents in specified folder,
            looking in default location C:\Temp\")
    directoryPath = "C:\Temp\"
End If
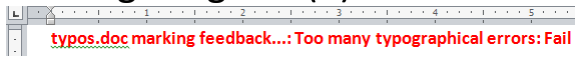```

## Revised Marking Program (3)

```
Do While (currentFile <> "")
    Documents.Open (directoryPath & currentFile)
    currentDocument = ActiveDocument.Name
    totalTypos = ActiveDocument.SpellingErrors.Count
    feedback = currentDocument & " marking feedback..."
    Selection.HomeKey Unit:=wdStory
    If (totalTypos > MAX_TYPOS) Then
        feedback = feedback & ": Too many typographical
                                errors: Fail"
    Else
        feedback = feedback & ": Pass"
    End If
    feedback = feedback & vbCr
    Selection.Text = feedback
    ' Loop body continued on next page
```

e.g. Feedback for "Typos.docx" = "Typos marking feedback…"

e.g. Feedback for "Typos.docx" = "typos.doc marking feedback…: Too many typographical errors: Fail"

Revised Marking Program (4)

typos.doc marking feedback...: Too many typographical errors: Fail

```
    ' Loop body continued from previous page
    With Selection.Font
        .Bold = True
        .Size = LARGER_FONT
        .Color = wdColorRed
    End With
    ActiveDocument.Close (wdSaveChanges)
    currentFile = Dir
Loop
End Sub
```

Counting The Number Of Occurrences Of A Word

- Example applications:
  - 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.
- Complete Word document containing the macro: `30counting occurences.docm`

Example: Counting Occurrences

```
Sub countingOccurences()
    Dim count As Long
    Dim searchWord As String
    count = 0
    searchWord = InputBox("Word to search for")

    ' Exact match (assignment)
    With ActiveDocument.Content.Find
        Do While .Execute(FindText:=searchWord, Forward:=True, _
          MatchWholeWord:=True) = True
            count = count + 1
        Loop
    End With
    MsgBox ("Exact matches " & count)
End Sub
```

Applying Many Of The Previous Concepts In A Practical Example & Linking Documents And (If There's Time)

- As you are aware different programs serve different purposes:
  - Database: storing and retrieving information
  - Spreadsheet: performing calculations, displaying graphical views of results
  - Word processor: creating text documents with many features for formatting and laying out text
- VBA allows the output of one program to become the input of another program.
  - Although this can be done 'manually' (reading the documents and typing in changes) if the dataset is large this can be a tedious and error-prone process
  - VBA can be used to automate the process

Example Problem

- Financial statements (monetary data) about many companies can be stored in a spreadsheet where an analysis can be performed e.g. does the company have enough $$$ on hand to meet its financial commitments.
- This information can be read into a VBA program which can further evaluate the data.
- The results can be presented in Word using the numerous text formatting features to highlight pertinent financial information.
- Names of the documents used in this example:
  - FNCE.xlsx (contains the financial data: program input)
  - 31spreadSheetAnalyzer.docm (contains the VBA program as well as the presentation of results: program output)

Spread Sheet Analyzer

```
Sub spreadsheetAnalyzer()
    Const MIN_INCOME = 250
    Const MIN_RATIO = 25

    Const PERCENT = 100
    Dim company1 As String
    Dim income1 As Long
    Dim ratio1 As Long
    Dim company2 As String
    Dim income2 As Long
    Dim ratio2 As Long
    Dim company3 As String
    Dim income3 As Long
    Dim ratio3 As Long
    Dim comment1 As String
    Dim comment2 As String
    Dim comment3 As String
```

	A	B	C	D
1	TAMCO			
2	Gross Income	Costs	Net income	Net over Gross
3	$100.00	$75.00	$25.00	33.33%
4				
5	HAL			
6	Gross Income	Costs	Net income	Net over Gross
7	$1,500.00	$1,250.00	$250.00	20.00%
8				
9	PEAR COMPUTER			
10	Gross Income	Costs	Net income	Net over Gross
11	$9,999.00	$999.00	$9,000.00	900.90%

TAMCO: 33%

HAL: Net income $250

PEAR COMPUTER: Net income $9000, 901% <== BUY THIS!

Spread Sheet Analyzer (2)

> Object =
> Type for any MS-Office variable
> https://msdn.microsoft.com/

```
Dim excel As Object
Set excel = CreateObject("excel.application")
excel.Visible = True

Dim workbook
Dim location As String
location = InputBox("Path and name of spreadsheet e.g.
                    C:\Temp\FNCE.xlsx")
Set workbook = excel.workbooks.Open(location)
```

Spread Sheet Analyzer (2)

> Object =
> Type for any MS-Office variable
> https://msdn.microsoft.com/

```
Dim excel As Object
Set excel = CreateObject("excel.application")
excel.Visible = True

Dim workbook
Dim location As String
location = InputBox("Path and name of spreadsheet e.g.
                    C:\Temp\FNCE.xlsx")
Set workbook = excel.workbooks.Open(location)
```

Spread Sheet Analyzer (3)

	A	B	C	D
1	TAMCO			
2	Gross Income	Costs	Net income	Net over Gross
3	$100.00	$75.00	$25.00	33.33%
4				
5	HAL			
6	Gross Income	Costs	Net income	Net over Gross
7	$1,500.00	$1,250.00	$250.00	20.00%
8				
9	PEAR COMPUTER			
10	Gross Income	Costs	Net income	Net over Gross
11	$9,999.00	$999.00	$9,000.00	900.90%

```
' Get company names
company1 = excel.Range("A1").Value
company2 = excel.Range("A5").Value
company3 = excel.Range("A9").Value

' Get net income and ratio
income1 = excel.Range("C3").Value
ratio1 = excel.Range("D3").Value * PERCENT
income2 = excel.Range("C7").Value
ratio2 = excel.Range("D7").Value * PERCENT
income3 = excel.Range("C11").Value
ratio3 = excel.Range("D11").Value * PERCENT

' Move the selection to the top of the Word document
Selection.HomeKey Unit:=wdStory
```

TAMCO: 33%

Spread Sheet Analyzer (4): First Company

	A	B	C	D
1	TAMCO			
2	Gross Income	Costs	Net income	Net over Gross
3	$100.00	$75.00	$25.00	33.33%

```
comment1 = company1 & ": "
If (income1 >= MIN_INCOME) Then
    comment1 = comment1 & "Net income $" & income1
    Selection.Font.Color = wdColorRed
    Selection.TypeText (comment1)
    If (ratio1 >= MIN_RATIO) Then
        comment1 = ", " & ratio1 & "% <== BUY THIS!"
        Selection.Font.Color = wdColorBlue
        Selection.TypeText (comment1)
    End If
    Selection.TypeText (vbCr)
Else
    If (ratio1 >= MIN_RATIO) Then
        comment1 = comment1 & ratio1 & "%" & vbCr
        Selection.Font.Color = wdColorBlue
        Selection.TypeText (comment1)
    End If
End If
```

Spread Sheet Analyzer (5): Second Company

HAL: Net income $250

5	HAL			
6	**Gross Income**	**Costs**	**Net income**	**Net over Gross**
7	$1,500.00	$1,250.00	$250.00	20.00%

```
comment2 = company2 & ": "
If (income2 >= MIN_INCOME) Then
   comment2 = comment2 & "Net income $" & income2
   Selection.Font.Color = wdColorRed
   Selection.TypeText (comment2)
   If (ratio2 >= MIN_RATIO) Then
       comment2 = ", " & ratio2 & "% <== BUY THIS!"
       Selection.Font.Color = wdColorBlue
       Selection.TypeText (comment2)
   End If
   Selection.TypeText (vbCr)
Else
   If (ratio2 >= MIN_RATIO) Then
       comment2 = comment2 & ratio2 & "%" & vbCr
       Selection.Font.Color = wdColorBlue
       Selection.TypeText (comment2)
   End If
End If
```

Spread Sheet Analyzer (6): Third Company

PEAR COMPUTER: Net income $9000, 901% <== BUY THIS!

	A	B	C	D
9	**PEAR COMPUTER**			
10	**Gross Income**	**Costs**	**Net income**	**Net over Gross**
11	$9,999.00	$999.00	$9,000.00	900.90%

```
comment2 = company2 & ": "
If (income2 >= MIN_INCOME) Then
   comment2 = comment2 & "Net income $" & income2
   Selection.Font.Color = wdColorRed
   Selection.TypeText (comment2)
   If (ratio2 >= MIN_RATIO) Then
       comment2 = ", " & ratio2 & "% <== BUY THIS!"
       Selection.Font.Color = wdColorBlue
       Selection.TypeText (comment2)
   End If
   Selection.TypeText (vbCr)
Else
   If (ratio2 >= MIN_RATIO) Then
       comment2 = comment2 & ratio2 & "%" & vbCr
       Selection.Font.Color = wdColorBlue
       Selection.TypeText (comment2)
   End If
End If
```

After This Section You Should Now Know

- Collections
  - What are they
  - What is the advantage in using them
  - Common examples found in Word documents
- The Active document
  - What are some of the commonly accessed attributes
  - What are some useful methods
- Finding things using macros
  - How to find and replace: text, font effects or font styles
- Using the `end-with`

After This Section You Should Now Know (2)

- How to use branches to make decisions in VBA
  - `If`
  - `If-else`
  - `Multiple If's`
  - `If, else-if, else`
  - Nested branches
  - Using logic (`AND, OR, NOT`) in branches
- How to use the line continuation character to break up long instructions
- How to get a program to repeat one or more instructions using loops
  - `For-next`
  - `Do-while`

After This Section You Should Now Know (3)

- How to print multiple documents
- How to use the 'Dir' function to access a folder
  - Using this function to step through all the documents or specific types of documents in a folder
- Accessing other types of MS-Office programs with an VBA program written for Word

Copyright Notice

- Unless otherwise specified, all images were produced by the author (James Tam).