VBA (Visual Basic For Applications) Programming

Overview of concepts covered in this section:

- Finding and replacing things in a document
- Branching
- Looping
- Strings
- Linking MS-Office documents
- Printing documents

Collection

- An object that consists of other objects
- Example: The *`Documents`* collection will allow access to the documents that have been opened.
- Access 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:

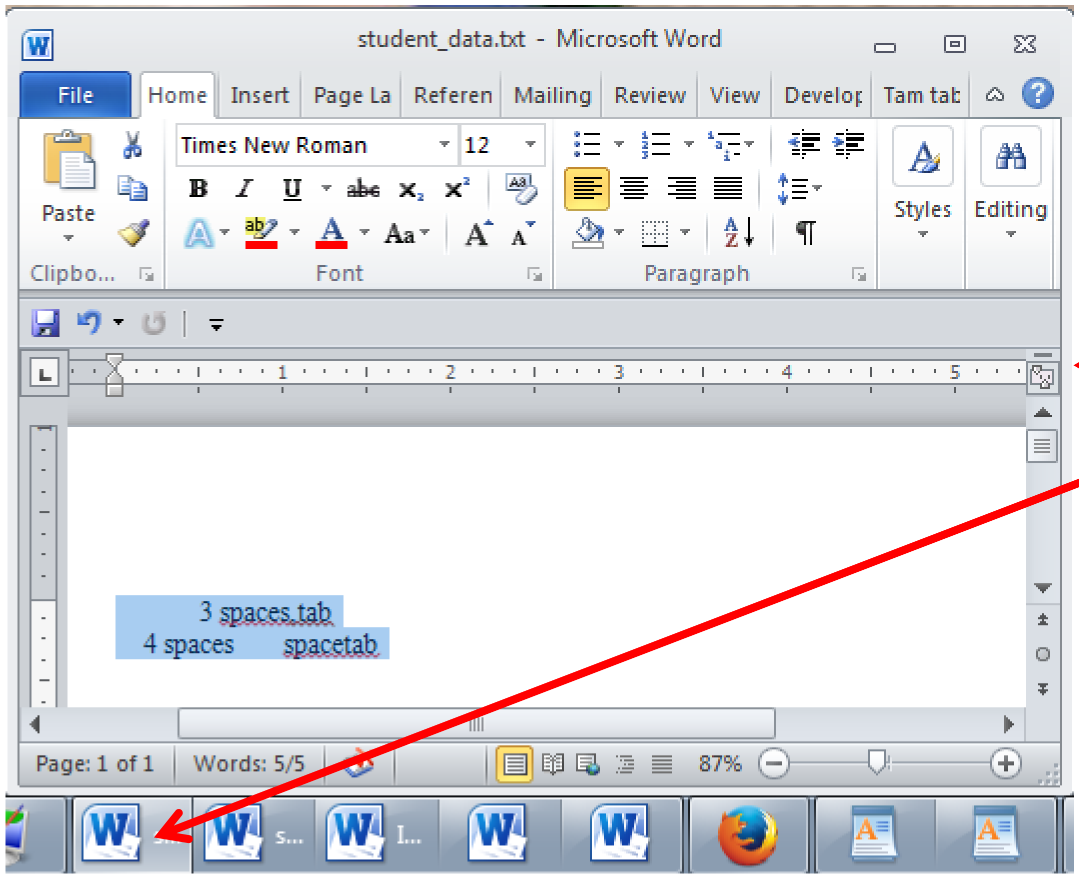
```
Documents.close
```

Types Of Collections

- Some attributes of a document that return a collection.
  - `Lists`: allows access to all lists in a document
  - `Shapes`: allows access to all shapes in a document
  - `Tables`: allows access to all tables in a document (detailed example coming up but a few brief examples below).
    - E.g., `ActiveDocument.Tables` – to access the tables in your document
    - `ActiveDocument.Tables(1)` – to access 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:



3 spaces tab
4 spaces spacetab

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.

  `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)

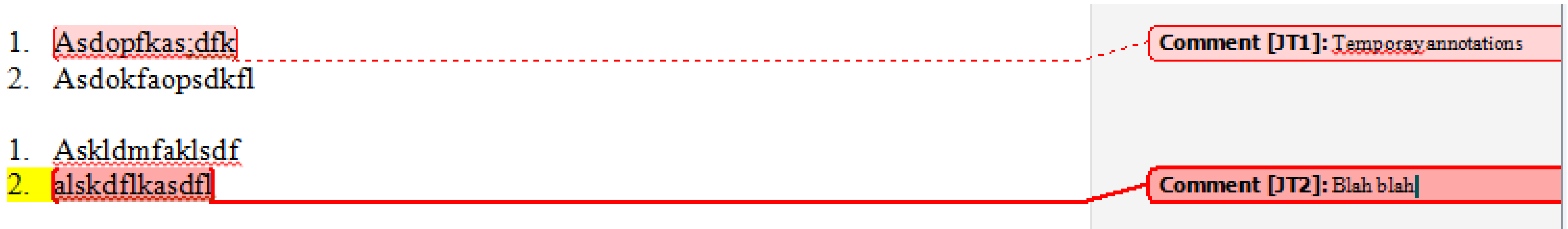
  Printout(): prints current active document on the default printer

  Save() : covered in the previous section

  SaveAs2() : covered in the previous section

  Select(): covered in the previous section

  SendMail(): see image (next slide)



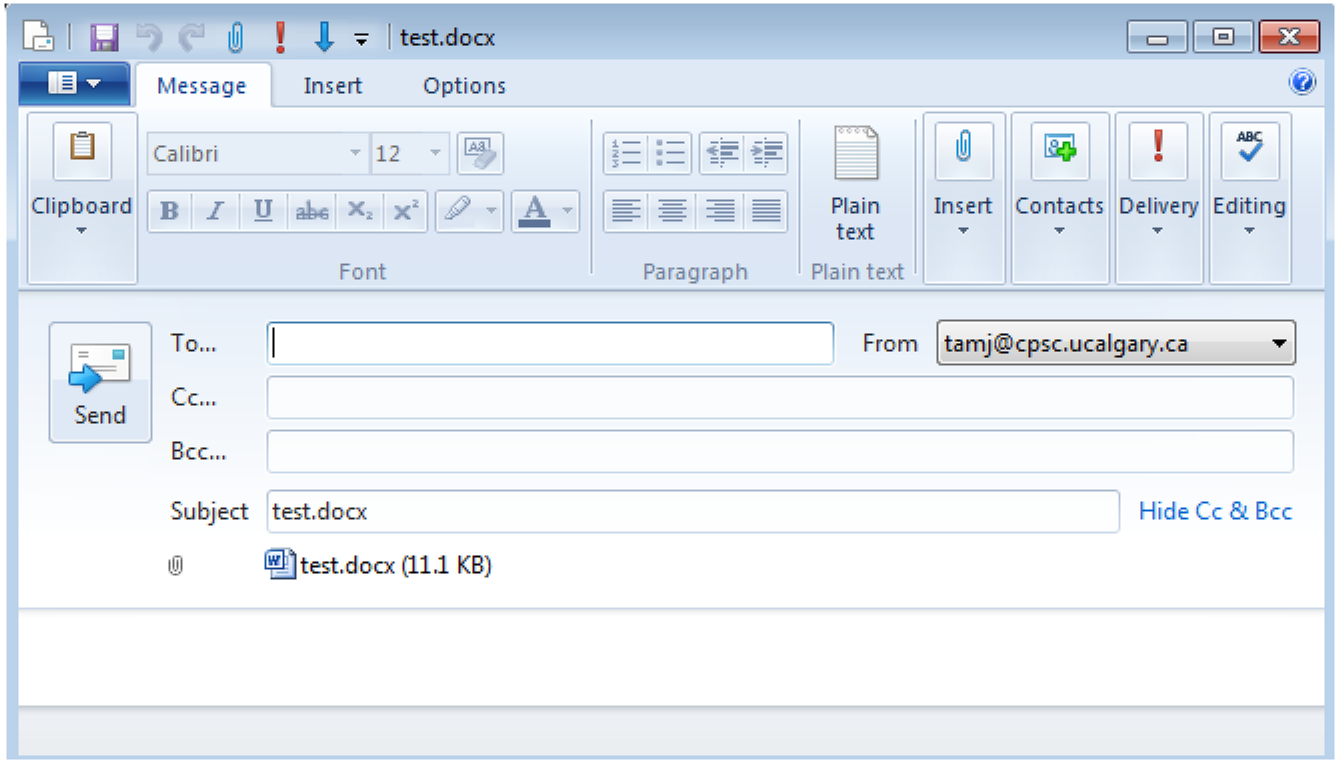
1. Asdopfkas;dfk
2. Asdokfaopsdkfl

1. Askldmfaklsdf
2. alskdflkasdfl

Comment [JT1]: Temporary annotations

Comment [JT2]: Blah blah

ActiveDocument.SendMail()



test.docx

From: tamj@cpsc.ucalgary.ca

Subject: test.docx

test.docx (11.1 KB)

- 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`’

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

Single Replacement

- **Word document containing the macro**: simpleFind.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:

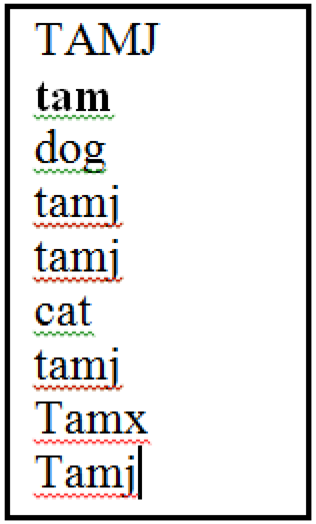
- My old email address (still works): tamj@cpsc.ucalgary.ca
- My new email address: tam@ucalgary.ca
- Incorrect variant: **tamj**@ucalgary.ca

More Complex Find And Replace

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

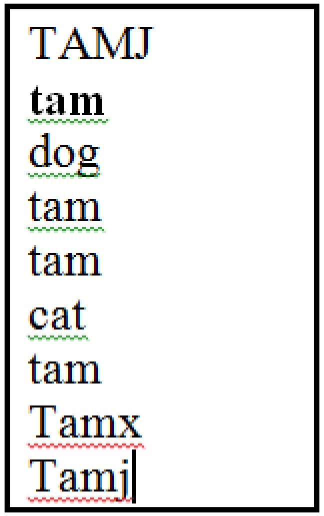
```
 Sub findReplaceAllCaseSensitive()
     ActiveDocument.Content.Find.Execute FindText:="tamj", _
       ReplaceWith:="tam", Replace:=wdReplaceAll, _
       MatchCase:=True
 End Sub
```

Before



TAMJ
tam
dog
tamj
tamj
cat
tamj
Tamx
Tamj

After



TAMJ
tam
dog
tam
tam
cat
tam
Tamx
Tamj

With, End With

`ActiveDocument.Content.Find.Execute`

- 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.

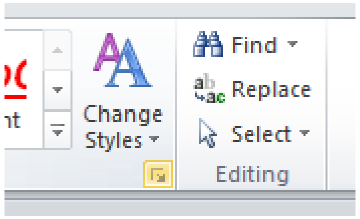
Finding And Replacing Bold Font

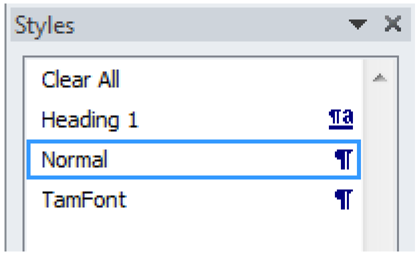
- **Word document containing the macro**: `findBold.docm`

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

'Removes bold facing effect on all text
```

Finding/Replacing Formatting Styles

- You may already have a set of pre-created formatting styles defined in MS-Word.





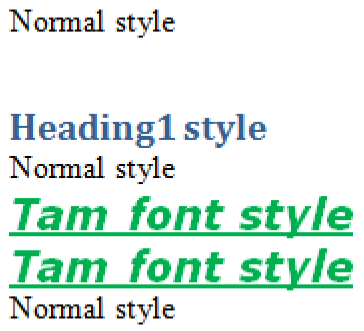
- You can redefine the characteristic of a style if you wish.
- Assume for this example that you wish to retain all existing styles and not change their characteristics.
- But you want to replace all *instances of one style* with another style e.g., all text that is ‘`normal`’ is to become ‘`TamFont`’
- ‘Find’ can be used to search (and replace) instances of a formatting style.

Finding/Replacing Formatting Styles (2)

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

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

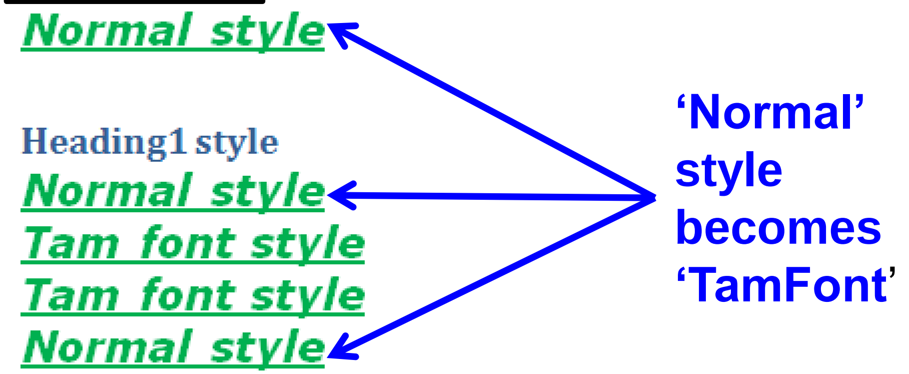
BEFORE



Normal style

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

AFTER



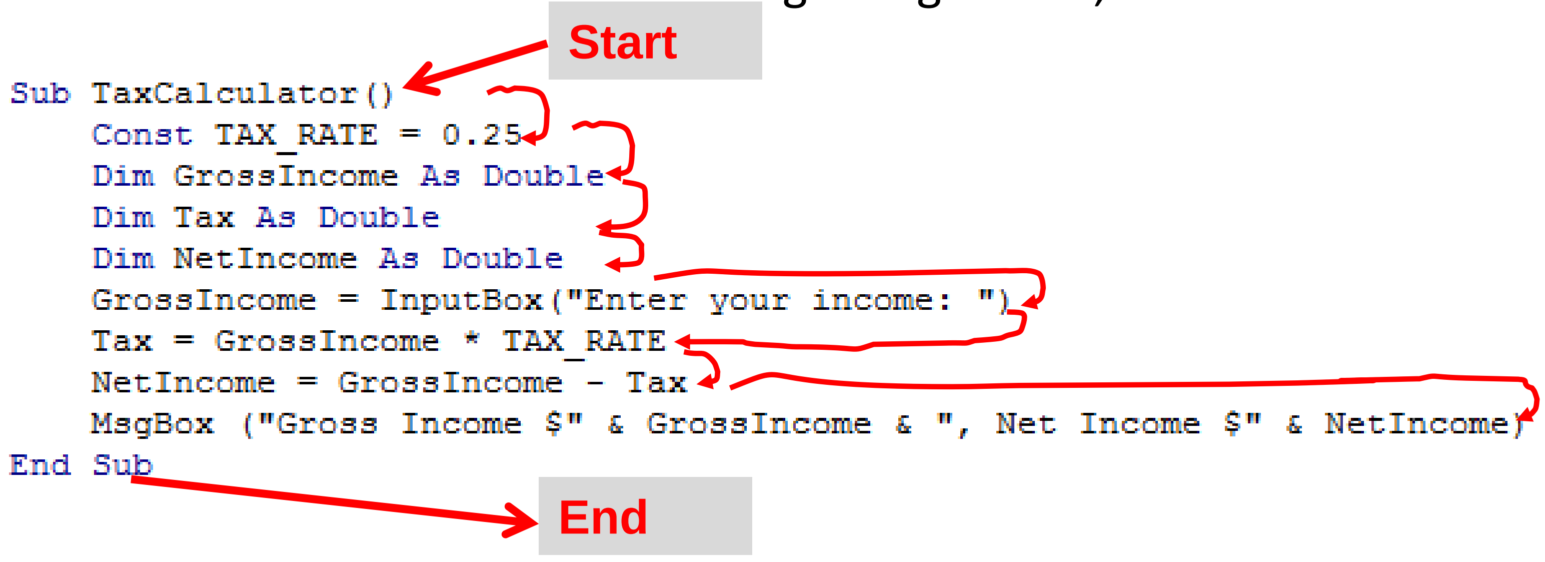
Normal style

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

'Normal' style becomes 'TamFont'

Recap: Programs You’ve Seen So Far

- How to write a program with a sequence of VBA instructions
  - Each instruction executes from beginning to end, one after the other

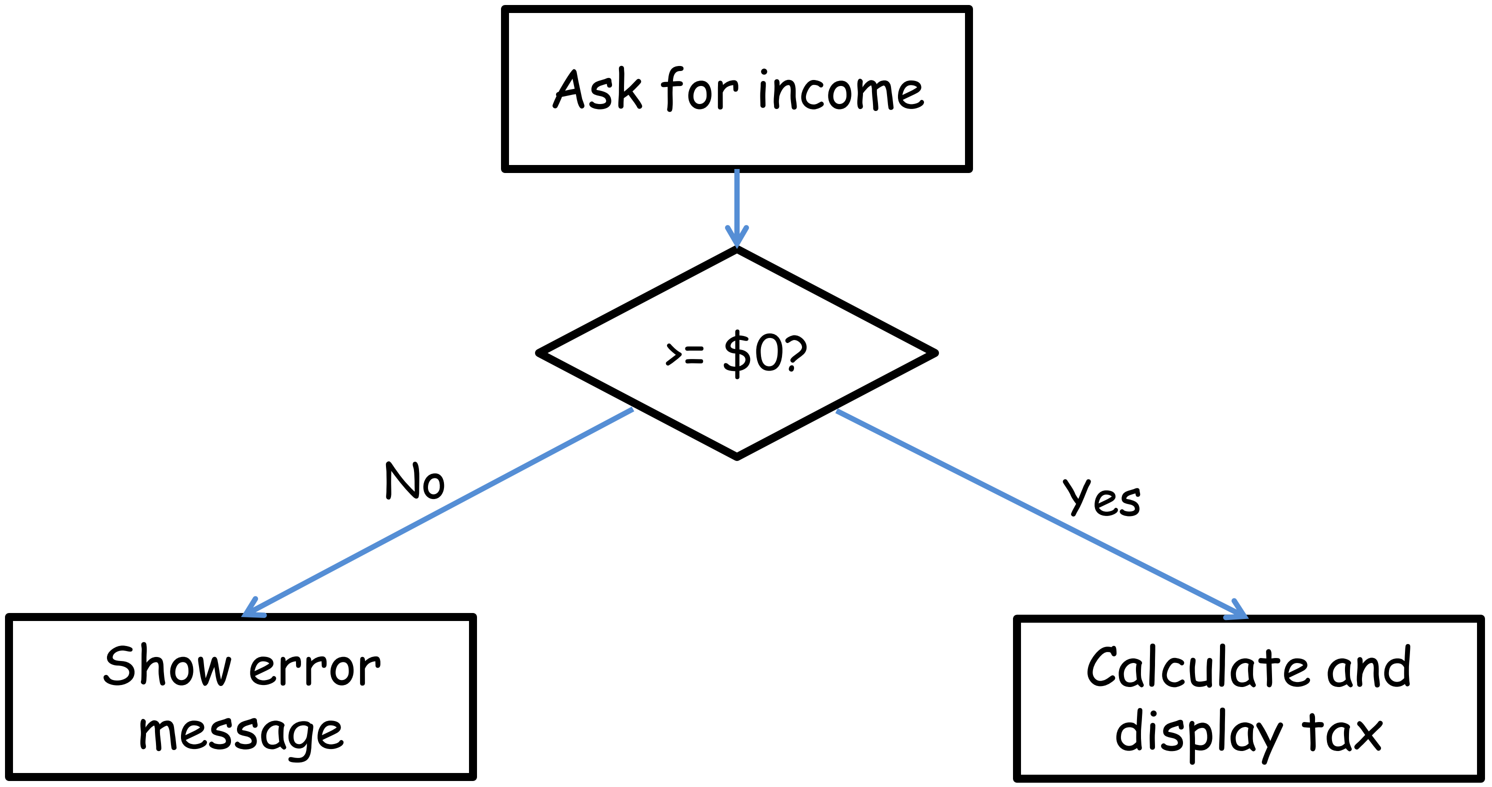


```
Sub TaxCalculator()
    Const TAX_RATE = 0.25
    Dim GrossIncome As Double
    Dim Tax As Double
    Dim NetIncome As Double
    GrossIncome = InputBox("Enter your income: ")
    Tax = GrossIncome * TAX_RATE
    NetIncome = GrossIncome - Tax
    MsgBox ("Gross Income $" & GrossIncome & ", Net Income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

- When the last instruction is reached then the program ends

What You Will Learn: Branching/Decisions

- What if alternatives may occur during execution (a branch in execution)
  - Each alternative may result in a different series of instructions being executed



How To Make Decisions In A Program

- Check if some condition has been met (e.g., password for the document correctly entered)
- Program may react one way if it's true that the condition has been met (e.g., password matches: `display confirmation message`)
- Program may also react another way if it's false that the condition has been met (e.g., password doesn't match: `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**

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

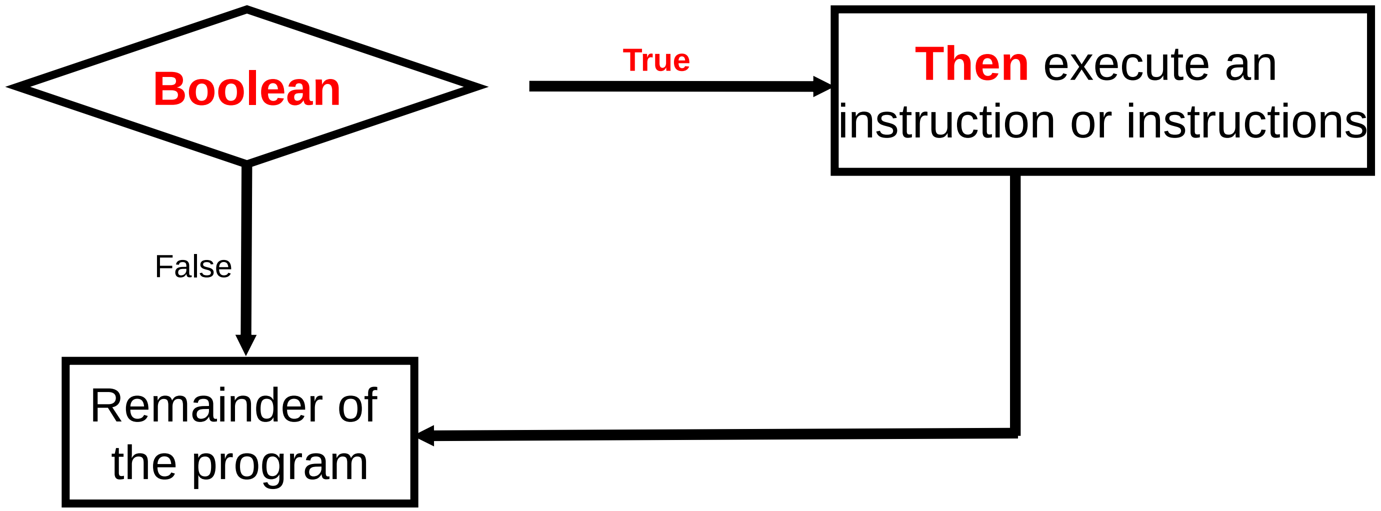
- **Body**: A block of program instructions that will execute under a specified condition.

```
Private 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

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**: `wordCount.docm`

```
' Try deleting all the words in the Word doc and run the
' macro again
Sub wordCount()
    Dim totalWords As Integer
    MIN_SIZE = 1000
    totalWords = ActiveDocument.Words.Count
    If (totalWords < MIN_SIZE) Then
        MsgBox ("Document too short, total words " &
          totalWords)
    End If
End Sub
```

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

Different Actions Required For The True Vs. False Cases

- While it is possible to explicitly state both cases using two if-then expressions…

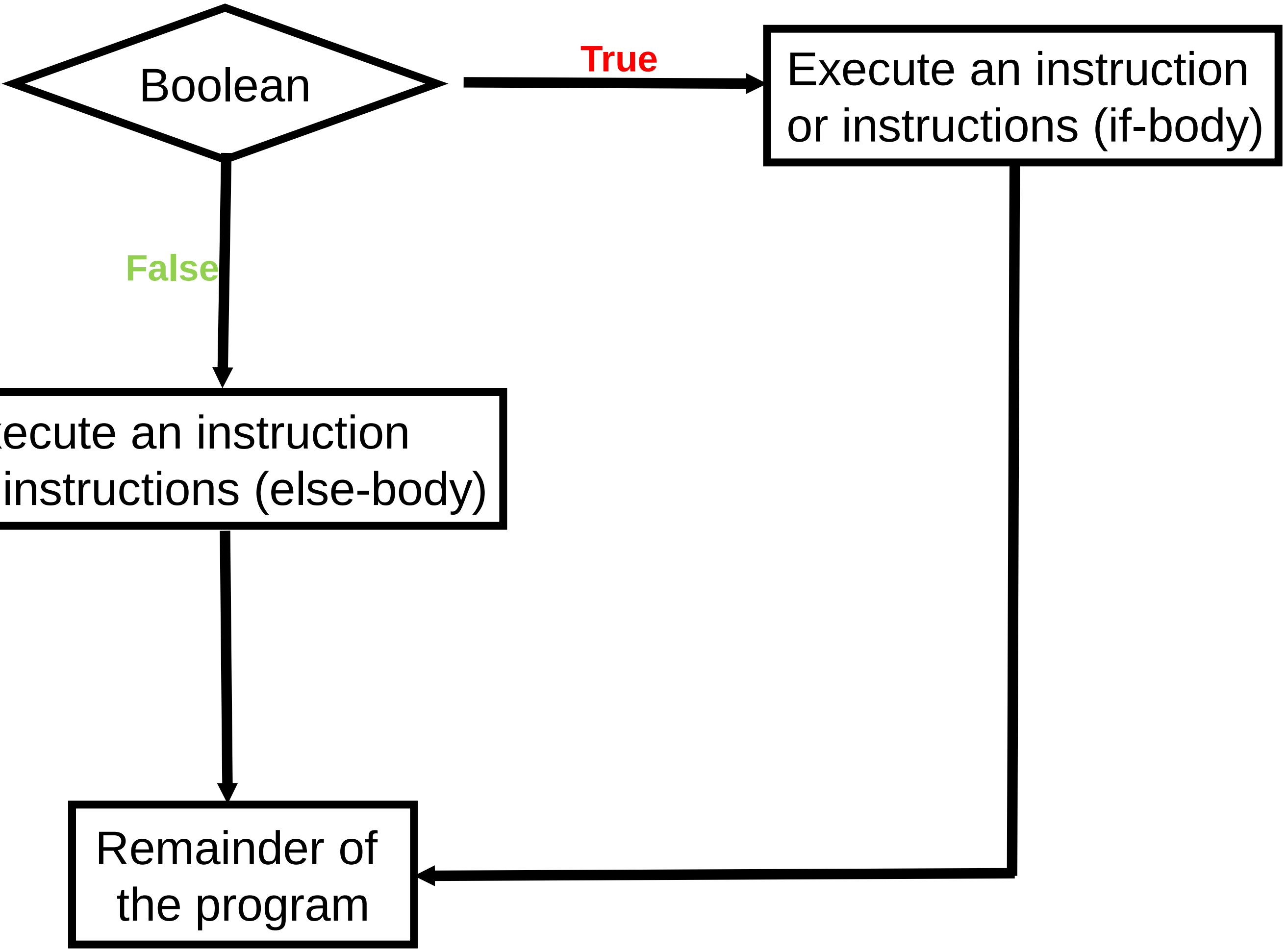
```
If (totalWords < MIN_SIZE) Then
    MsgBox ("Document too short, total words " &
      totalWords)
End If
If (totalWords >= MIN_SIZE) Then
    MsgBox ("Document meets min. length requirements")
End If
```

It's true that the document is too short

It's false that the document is too short

- The previous approach can be simplified
- Why? (What characteristics of the two if-then expressions may allow for an easy simplification)?

Decision Making With An ‘If, Else’


Boolean
True
Execute an instruction or instructions (if-body)
False
Execute an instruction or instructions (else-body)
Remainder of the program

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**: wordCount2.docm

```
Sub wordCount2()
    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
' Try deleting words or changing the minimum size and observe
' the effect on the program.
```

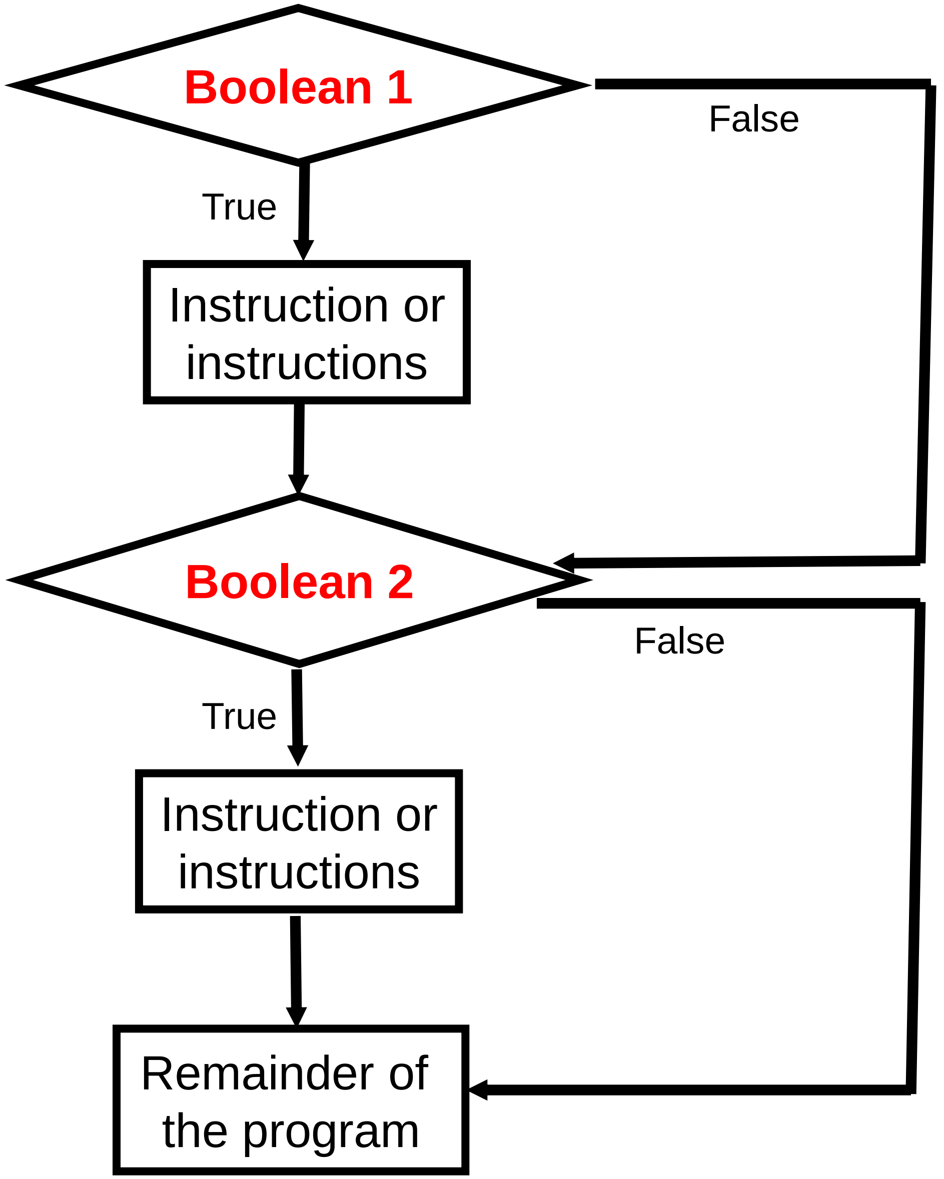
What To Do When Multiple Conditions Must Be Checked

- **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**: If the result of one condition affects other conditions (when one condition is true then the other conditions must be false)
  - `If-then`, `elseif`, `else` can 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`**



Q1: Are you an adult?
Q2: Are you a Canadian citizen?
Q3: Are you currently employed?

Multiple If-Then's

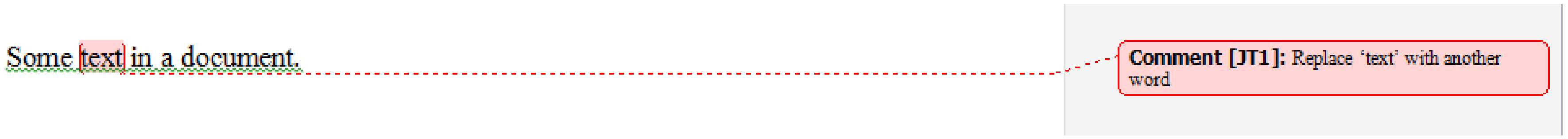
- Any, all or none of the conditions may be true
- Employ when a series of independent questions will be asked
- **Format:**

```
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:** `multipleIf.docm`

```
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
```



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`

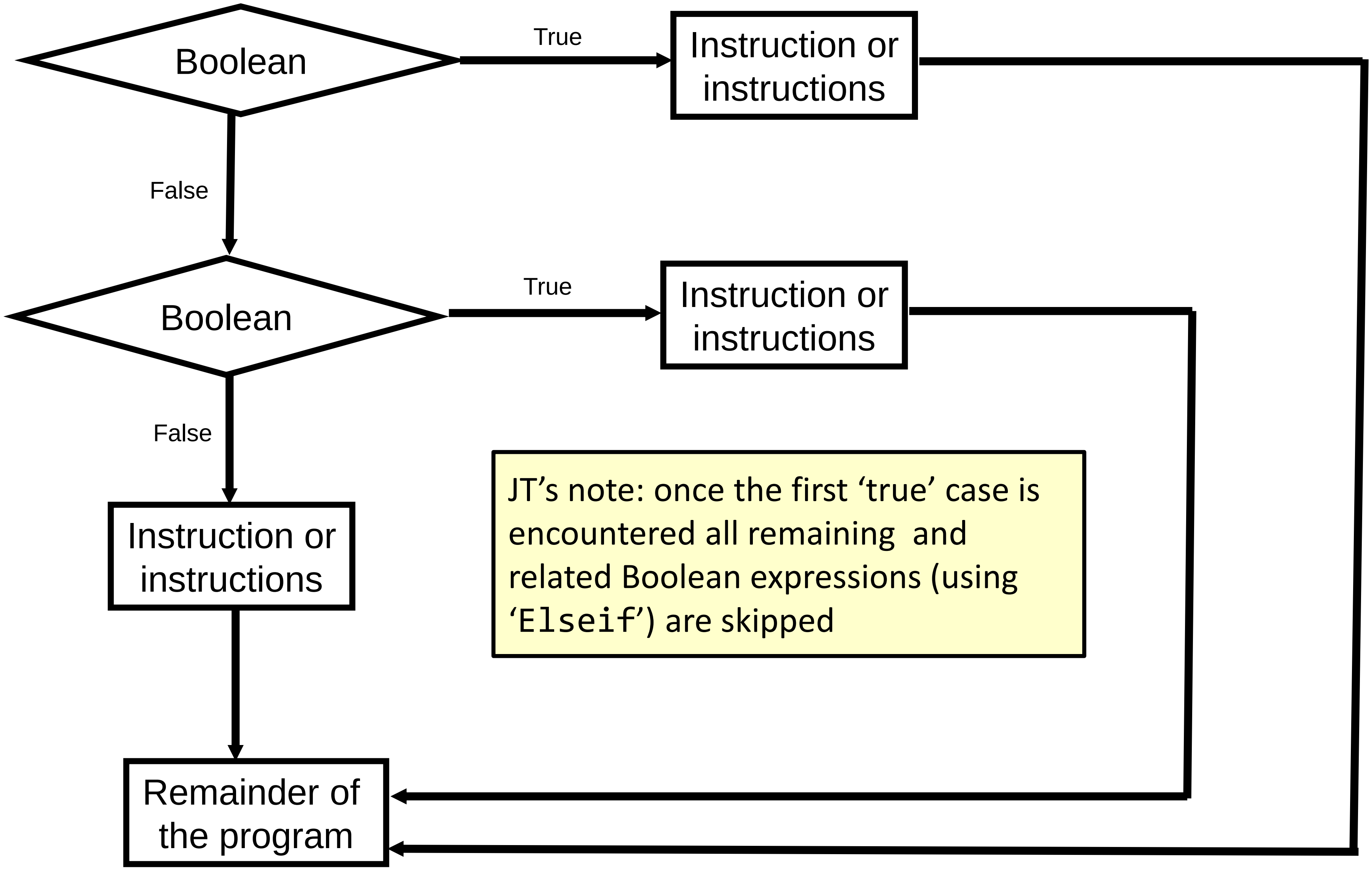
Inefficient combination!

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

```
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
    letter = "F"
End If
```

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


Boolean
True
Instruction or instructions
False
Boolean
True
Instruction or instructions
False
Instruction or instructions
Remainder of the program

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):
     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)**: “`gradesEfficient.py`”

```
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
```

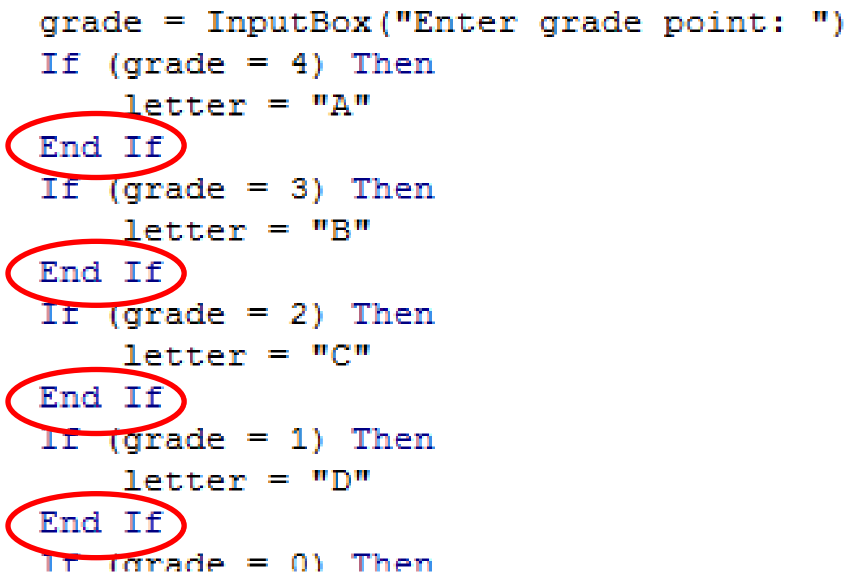
This approach is more efficient when at most only one condition can be true.

<u>Extra benefit:</u>

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

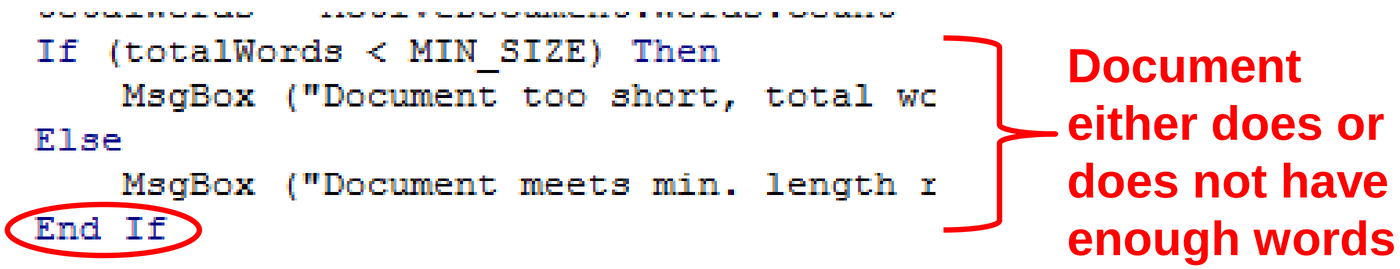
- Independent If-then’s:
  - Since each ‘if’ is independent each body must be followed by it’s own separate ‘end if’



```
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

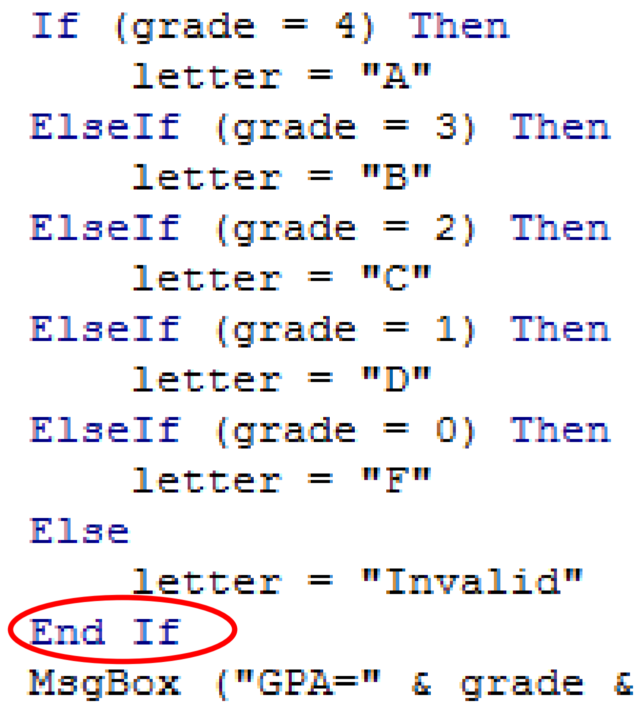
- If-then, Else:
  - Since the ‘if-then’ and the ‘else’ are dependent (either one body or the other must execute) the ‘end if’ must follow the body of the ‘else-body’ (last dependent “if-branch”)



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

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 &
```

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 an invalid value then the program can proceed.

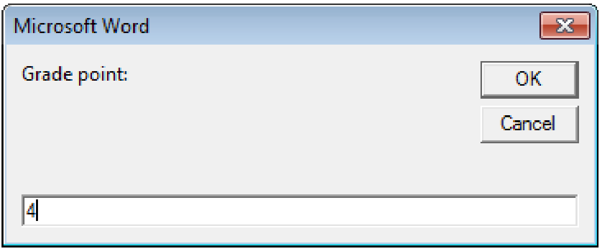
Logic: The "Or" Operator

- **Format:**

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

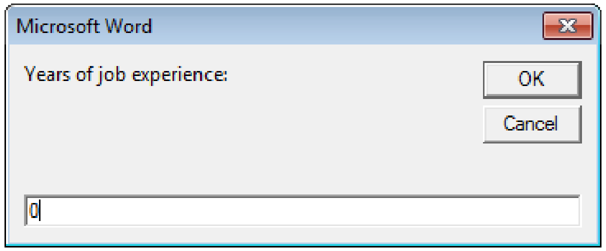
- **Word document containing the macro (empty document, see macro editor for the important details)**: "`if_or_hiring.docm`"

```
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"
```


Microsoft Word

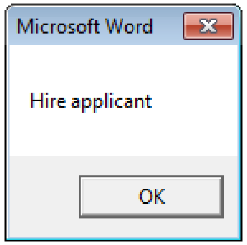
Grade point:

4


Microsoft Word

Years of job experience:

0


Microsoft Word

Hire applicant

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

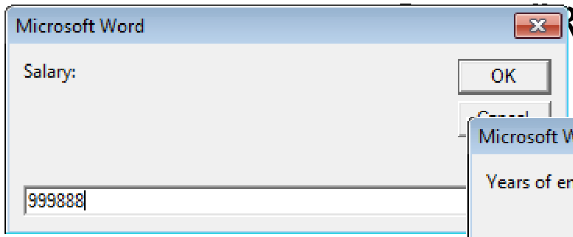
Logic: The “AND” Operator

- **Format:**

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

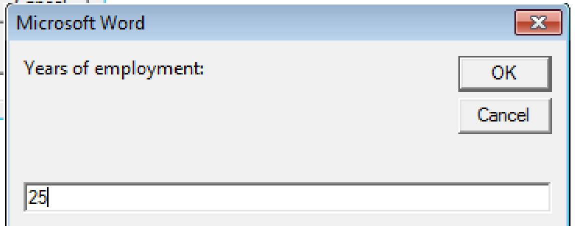
- **Word document containing the macro (empty document, see macro editor for the important details)**: `if_and_firing.py`

```
salary = InputBox("Salary: ")
years = InputBox("Years of employment: ")
If (salary >= 100000) And (years < 2) Then
    result = "Fired!"
Else
                    Retained"
```


Microsoft Word

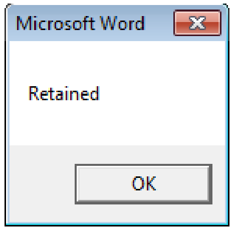
Salary:

999888


Microsoft Word

Years of employment:

25


Microsoft Word

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**:
  `checkSave.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
- Strings require the concatenation operator ‘&’

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

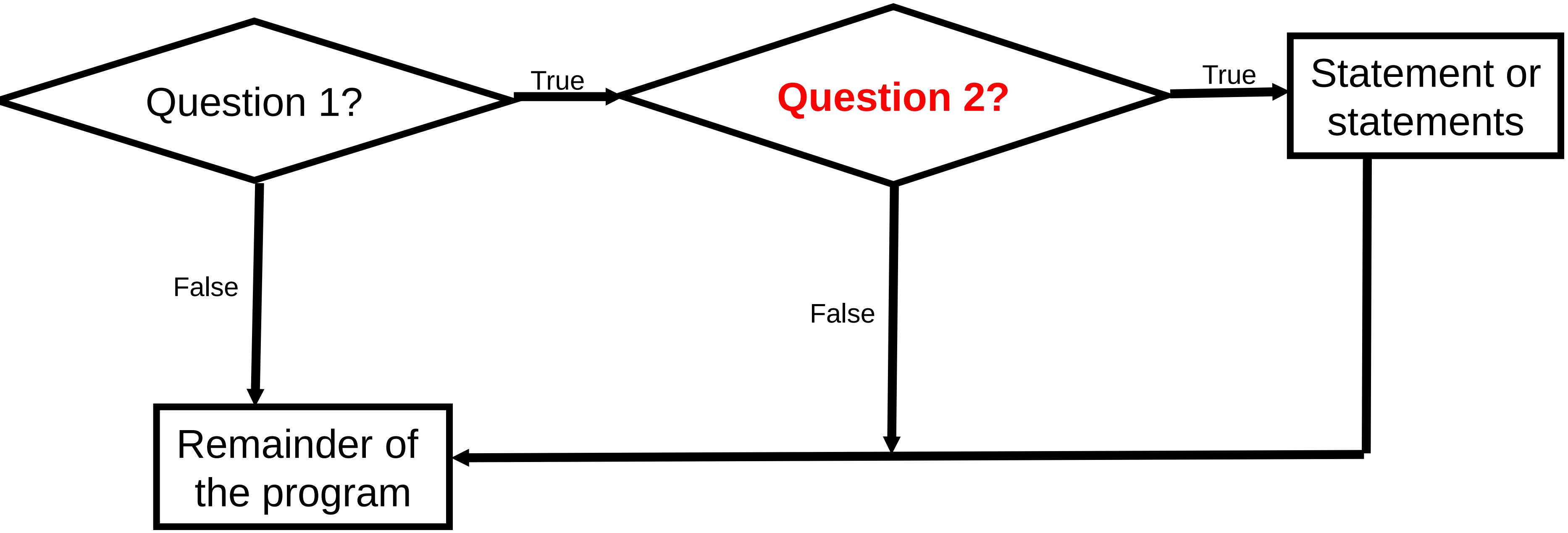
Line Continuation Character (2)

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

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

Nested Decision Making

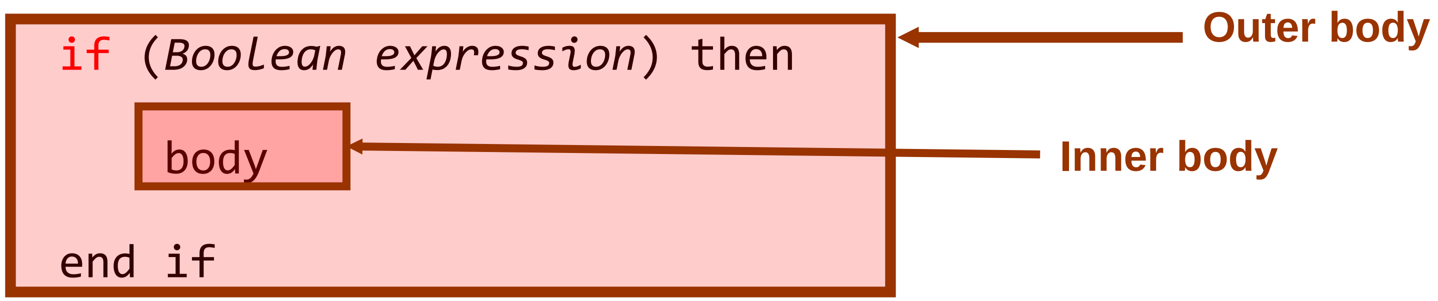
- Decision making is dependent.
  - One branch is 'nested' inside of another branch
- The first decision must evaluate to true ("gate keeper") before successive decisions are even considered for evaluation.



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
```



```
    if (Boolean expression) then
          body
    end if
```

```
end if
```

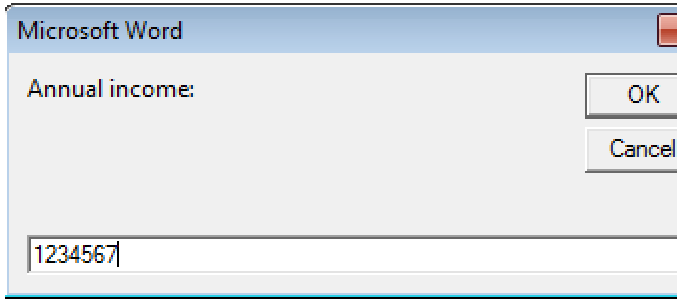
Example: Nested Branches

- **Word document containing the macro (empty document, see macro editor for the important details)**: “`nested.docm`”

```
Sub nested()
    Const TAX_RATE = 0.5
    Dim citizen As String
    Dim taxCredit As Long
```

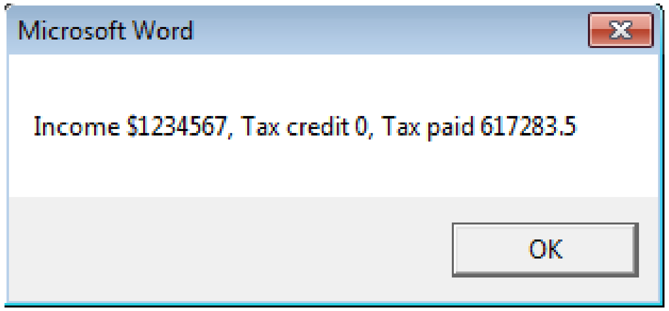
Example: Nested Branches (2)

```
    income = InputBox("Annual income: ")
    If (income < 10000) Then
        citizen = InputBox("Enter 'y' if citizen: ")
        If (citizen = "y") Then
            MsgBox ("This person can receive social
                     assistance")
            taxCredit = 100
        End If
    End If
    tax = (income * TAX_RATE) - taxCredit
    MsgBox ("Income $" & income & ", Tax credit " & taxCredit
& ", Tax paid " & tax)
End Sub
```

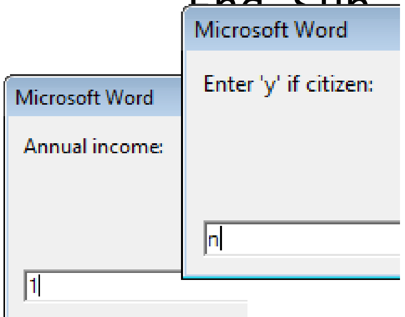

Microsoft Word

Annual income:

1234567


Microsoft Word

Income $1234567, Tax credit 0, Tax paid 617283.5


Microsoft Word

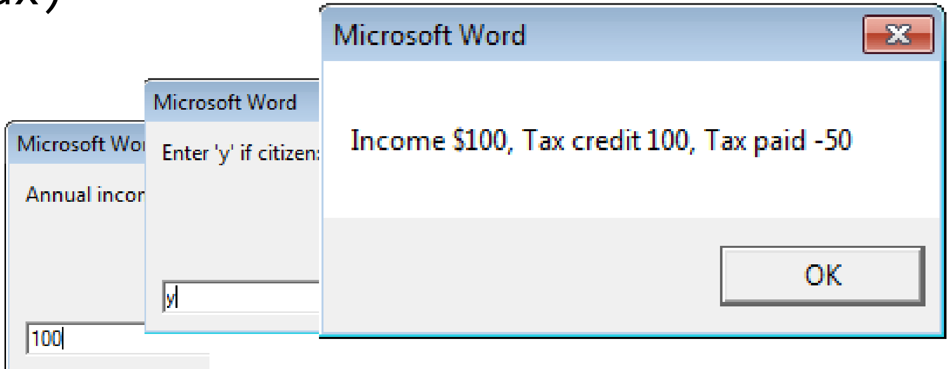
Annual income:

1

Microsoft Word

Enter 'y' if citizen:

n


Microsoft Wo

Annual incor

100

Microsoft Word

Enter 'y' if citizen:

y

Microsoft Word

Income $100, Tax credit 100, Tax paid -50

The Selection Object again

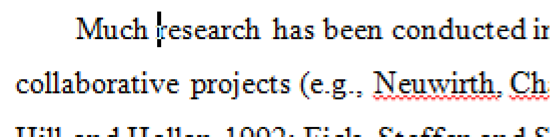
- With a previous example if no text was selected then the program would produce no visible effect.

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

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

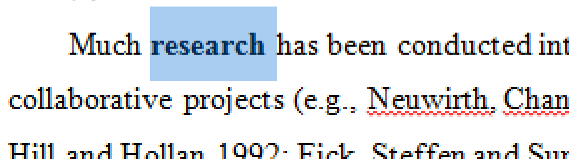
```
Sub AutoSelectedFontChange()
    Selection.Expand
    Selection.Font.Bold = wdToggle
End Sub
```

Before



Much research has been conducted ir
collaborative projects (e.g., Neuwirth, Ch
Hill and Hollan 1992; Fick, Steffen and S

After



Much **research** has been conducted int
collaborative projects (e.g., Neuwirth, Chan
Hill and Hollan 1992; Fick, Steffen and Sur

Constants For The Selection Object

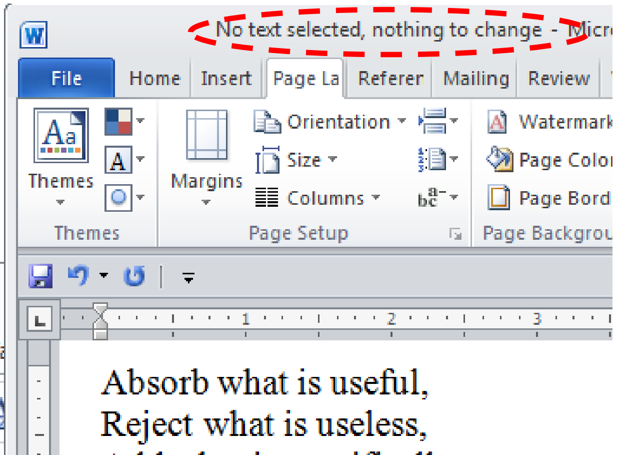
Name of constant	Meaning of constant
`wdSelectionIP`	No text selected
`wdSelectionNormal`	Text (e.g., word, sentence) has been selected
`wdSelectionShape`	A graphical shape (e.g., circle, text book) has been selected

The Selection Object again

- Application of branching: check if a selection has been made and only apply the selection if that is the case.
- **Word document containing the macro:** “selectionExample.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
```

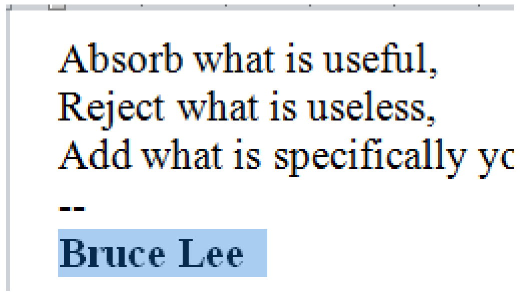
Run macro: No selection



No text selected, nothing to change - Micr

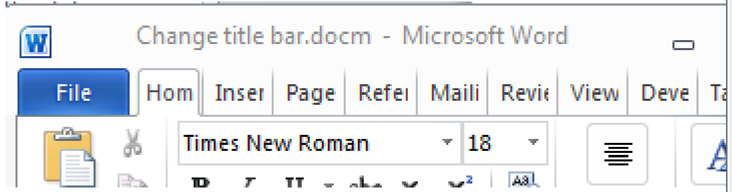
Absorb what is useful,
Reject what is useless,

Run macro: selected text bolded



Absorb what is useful,
Reject what is useless,
Add what is specifically yo
--
Bruce Lee

Default title bar



Change title bar.docm - Microsoft Word

Marking/Spelling Checking A Document

- Suppose you want to mark a document with a pass/fail grade based on the number of typographical errors (e.g., more than 30 is a fail, anything less is a pass).
- Assume that document names match student names
- For document to be marked you will create another document in the same folder.
- To make it easier to pair up marking with the student the 'marking document' will be named "Marking for: <document name>"
  - E.g., "`james tam.doc`" would produce a marking document called "`Marking for: James Tam.doc`"
  - Inside the marking document will be the text "`Marking for: <document name> <pass or fail>`"

“Marking_Program”

Word document containing the macro:
`markingProgram.docm`

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

“Marking_Program” (2)

```
'Get Name of current document
currentDocument = ActiveDocument.Name

'Name of marking document based on current doc
markingDocument = "MARKS FOR " & currentDocument

fileLocation = ActiveDocument.Path
totalTypos = ActiveDocument.SpellingErrors.count

'Feedback is prefaced by student(document) name
feedback = currentDocument
```

“Marking_Program” (3)

```
'Creates a new word document based on the 'normal' template
'Create a variable ‘wordDocument’ to refer to the newly created
'document
Set wordDocument = Documents.Add("Normal.dot")
```

“Marking_Program” (4)

```
    'Recall: before this feedback just = document name
    If (totalTypos > MAX_TYPOS) Then
        feedback = feedback & ": Too many typographical
          errors: Fail"
        Selection.TypeText (feedback)
    Else
        feedback = feedback & ": Pass"
        Selection.TypeText (feedback)
    End If
    ‘Saving feedback doc in same location but under name of
    ‘marking (and not the student) document
    wordDocument.SaveAs2 (fileLocation & "\" &
        markingDocument)
End Sub
```

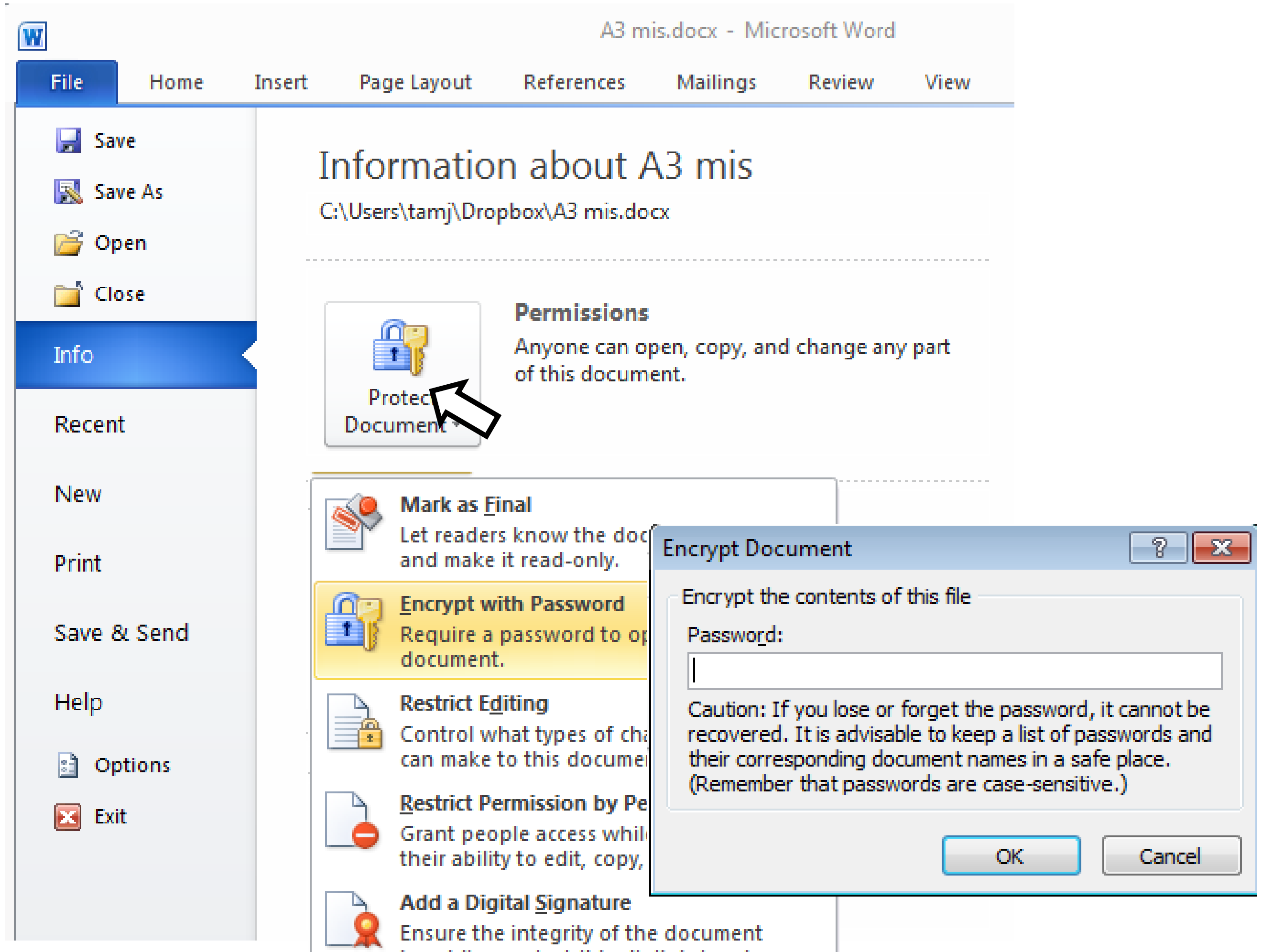
Example Run Of Marking Program

- Suppose that this macro was part of a word document "`marking program.docm`"
- Running the macro would then produce a file called "MARKS FOR `marking program.docm`"
  - (Assuming that the program had no spelling errors) this file would contain the following text:

```
Marking program.docm: Pass
```

Securing A Document: Using MS-Word

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


A3 mis.docx - Microsoft Word

File | Home | Insert | Page Layout | References | Mailings | Review | View

Save
Save As
Open
Close
Info
Recent
New
Print
Save & Send
Help
Options
Exit

Information about A3 mis

C:\Users\tamj\Dropbox\A3 mis.docx

Protect Document

Permissions

Anyone can open, copy, and change any part of this document.

Mark as Final
Let readers know the doc
and make it read-only.

Encrypt with Password
Require a password to op
document.

Restrict Editing
Control what types of cha
can make to this docume

Restrict Permission by Pe
Grant people access whil
their ability to edit, copy,

Add a Digital Signature
Ensure the integrity of the document

Encrypt Document

Encrypt the contents of this file

Password:

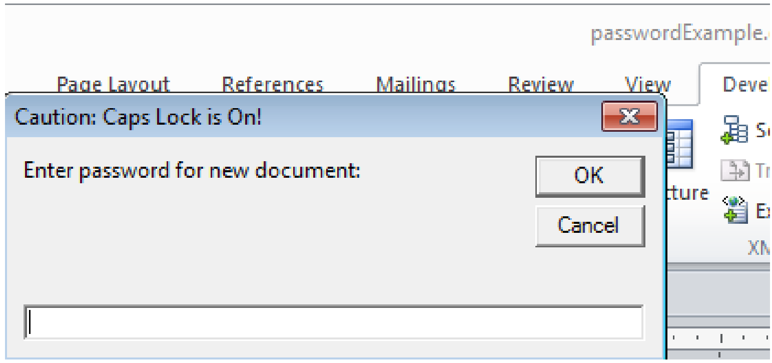
Caution: If you lose or forget the password, it cannot be recovered. It is advisable to keep a list of passwords and their corresponding document names in a safe place. (Remember that passwords are case-sensitive.)

OK | Cancel

Securing A Document: Simple VBA Example

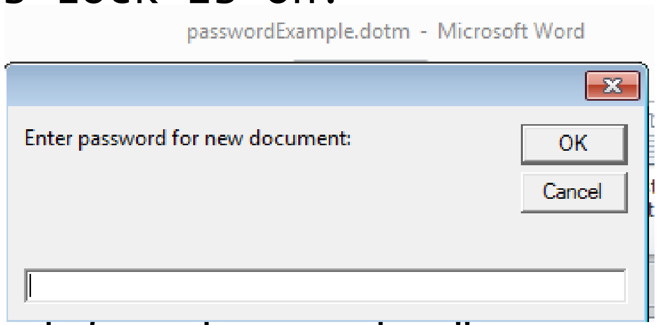
- **Word document containing the macro:**
  passwordBranchExample.docm

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

    If (Application.CapsLock = True) Then
        warningCaps = "Caution: Caps Lock is On!"
    Else
        warningCaps = ""
    End If

    yourPassword = InputBox("Password for document: ",
       warningCaps)
    ActiveDocument.Password = yourPassword
End Sub
```



Caution: Caps Lock is On!

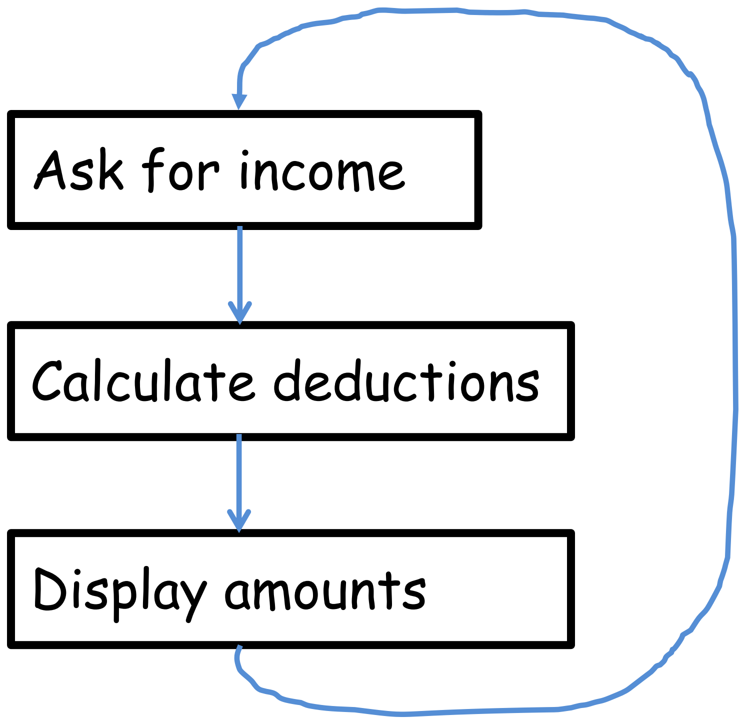
Enter password for new document:



Enter password for new document:

What You Will Learn: Repetition/Loops

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



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

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, while the user enters an erroneous value ask the user for input.
  - `Do-while loop`

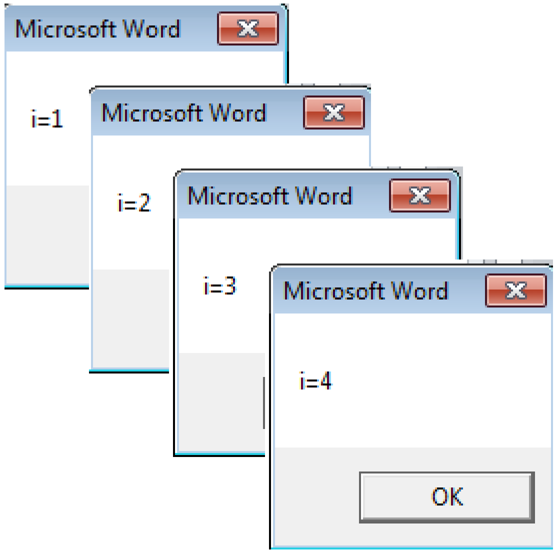
For-Next Loops

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

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

- **Example**: “`for1.docm`”

```
Dim i As Integer

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



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

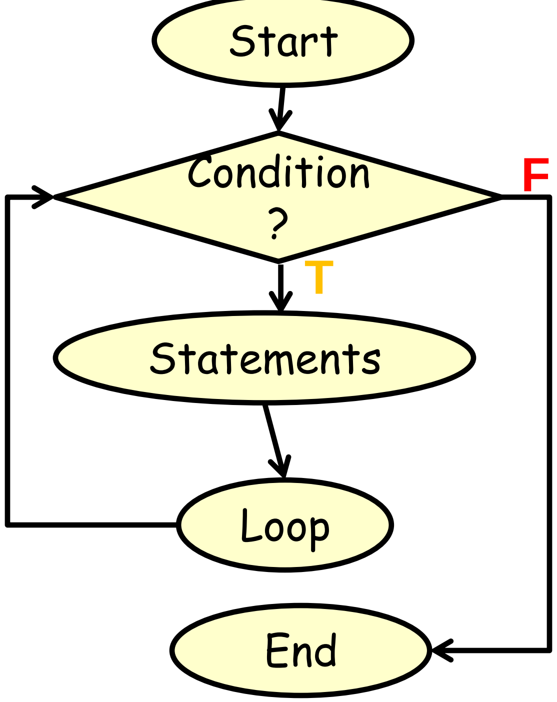
For-Next Loops (2)

- For-next loops can count down as well as up
- The Steps can be values other than one.
- **Example**: “`for2.docm`”

```
Dim i As Integer

For i = 12 To 0 Step -3
    MsgBox ("i=" & i)
Next i
```

```
12
9
6
3
0
```

Do-While Loop



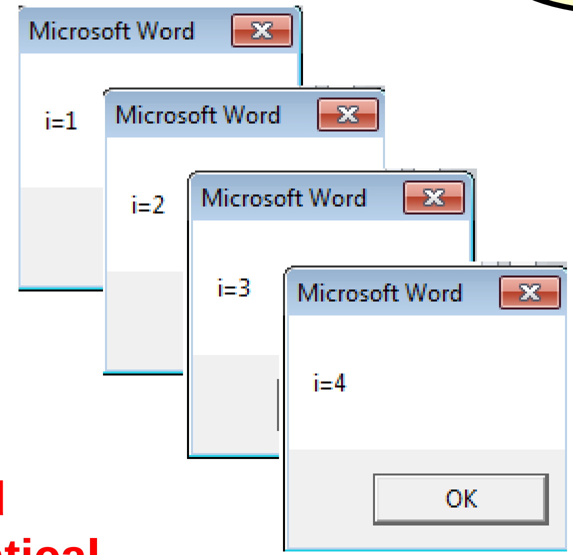
- **Format**:

```
Do While <Condition>
    <Statement(s)>
Loop
```

- **Example**: "while1.docm"

```
Dim i As Integer
i = 1
Do While i <= 4
    MsgBox ("i=" & i)
    i = i + 1
Loop
```

Any valid mathematical expression here



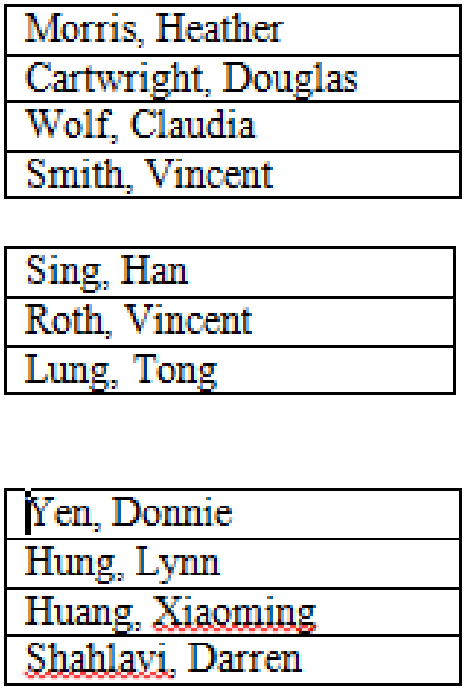
```
i=1
i=2
i=3
i=4
```

Simple Example: Sorting Three Tables

- Instructions needed for sorting 3 tables

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

Before

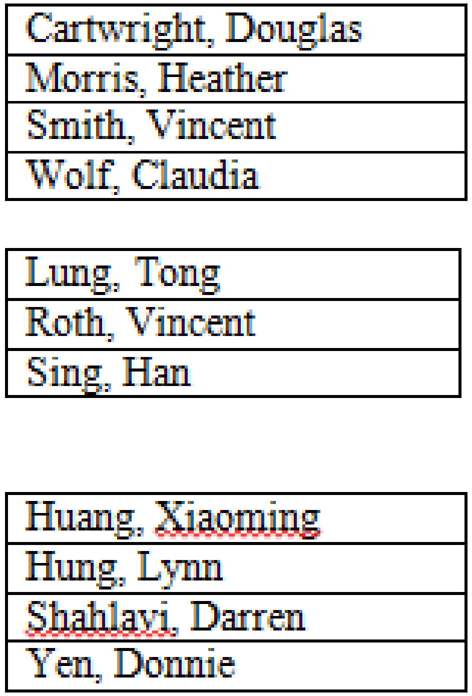


Morris, Heather
Cartwright, Douglas
Wolf, Claudia
Smith, Vincent

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 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
```

- Sorting can be applied reduce the duplicated statements

Revised Example: Sorting Tables With A Loop

Word document containing the macro:
“`sortingTables.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

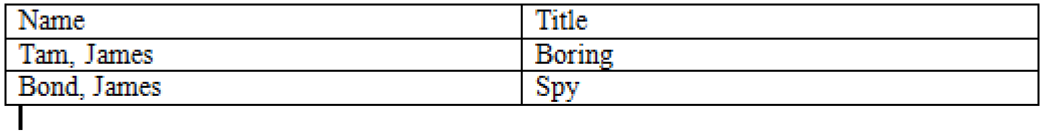
Adama, Bill	Commander
Adama, Lee	CAG
Morris Heather	Heroine

More On Sort

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

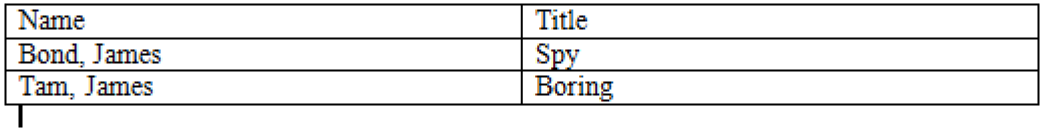
  `Sort (<Boolean to Exclude header – True or False>)`

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



Name	Title
Tam, James	Boring
Bond, James	Spy

  - After



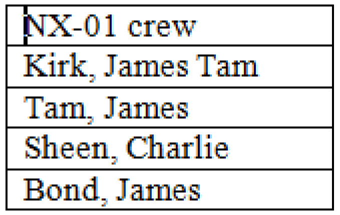
Name	Title
Bond, James	Spy
Tam, James	Boring

Second Sorting Example: Exclude Headers

- **Document containing the macro**:
  “sortingTablesExcludeHeader.docm”

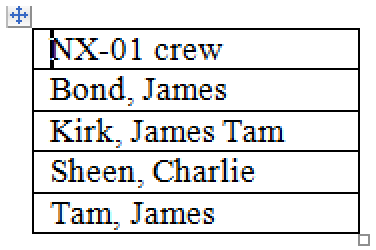
```
 Sub Sort()
      Dim CurrentTable As Integer
      Dim NumTables As Integer
      NumTables = ActiveDocument.Tables.Count
      If NumTables = 0 Then
           ' Don't bother sorting
           MsgBox ("No tables to sort")
      Else
           For CurrentTable = 1 To NumTables Step 1
        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

The Need For String Operations

- Sometimes you only want a part of a string (“substring”)
- Example a string containing location information
  - Address = “`ABCalgary`”
- If there is a standard format in the data e.g., the first two characters will always be the province then you can apply a string operation to remove the desired sub-string from the original string
  - “AB”
  - `Left(address,2)` <= start counting from the left extract the first two characters

More On Strings

- A string consists of a series of characters.
- Each character in a string has a position (referred to as an 'index').
  - The first character is at position zero
- Examples
  - "`Hello`"

0	1	2	3	4
'H'	'e'	'l'	'l'	'o'

  - "`u r`"

0	1	2
'u'	<SPACE>	'r'

Some Useful String Operators

- Assume we have the following strings created for the examples

```
Dim str1 as String
Dim str2 as String
Dim num as Integer
str1 = "hello world"
str2 = "hello"
```

Desired operation	Function	Example usage	Result
Retrieve the first 'n' characters (count from left)	`Left(<string>, n)`	`str2 = left(str1,5)`	Str2 contains the string "hello"
Retrieve the last 'n' characters (count from right)	`Right(<string>,n)`	`Str2 = right(str1,4)`	Str2 contains the string "orld"
Determine a string length	`Len(<string>)`	`num = Len(str1)`	Num is 11
Comparing strings	`StrComp(<string1>, <string2>)`	`Num = strComp(str1,str2)`	Num is zero if identical, non-zero if different

String Compare Example

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

```
Sub stringCompare ()
    Dim str1 As String
    Dim str2 As String
    Dim num As Integer
    str1 = InputBox("enter a string")
    str2 = InputBox("enter a string")
    num = StrComp(str1, str2)
    MsgBox (num)
```

```
Str1= “ab”
Str2 = “ab”
Num = 0
```

```
Str1= “ab”
Str2 = “ba”
Num = -1
```

```
Str1= “ab”
Str2 = “aa”
Num = 1
```

JT: “Why are we learning this stuff (string compare function)???”

Linking Office Documents

- One document contains a link to another document (typically this is done with two different type of MS-Office applications to take advantages of the strengths of each application).
- Pro
  - There are two separate documents (saves on file size, changes in the original document automatically show in document containing the link)
- Con:
  - It's location specific (moving documents or sharing documents results in 'breaking' the link)

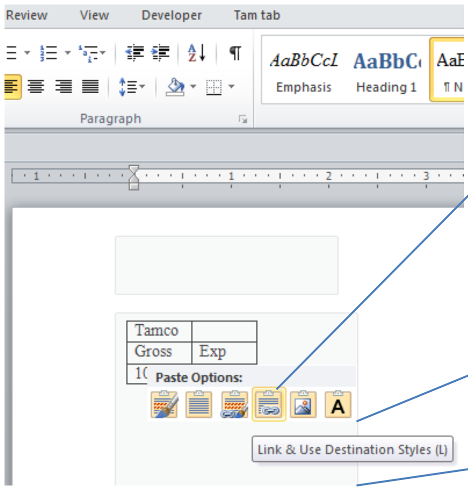
For more information: http://support.microsoft.com/kb/76993

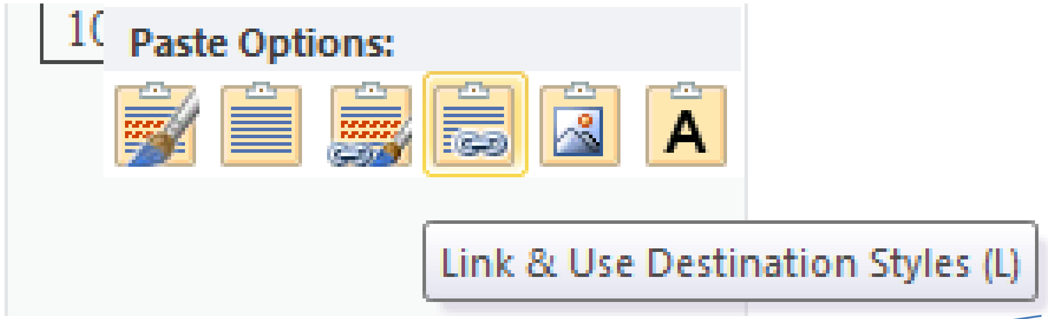
How To Link Documents (Word Linked To Excel)

- Suppose you have an extensive amount of financial information entered and calculated in a spreadsheet

TAMCO			
Gross income	Costs	Net income	Net:Gross Income
100	75	25	25.00%
HAL			
Gross income	Costs	Net income	Net:Gross Income
1500	1250	250	16.67%
Pear computer			
Gross income	Costs	Net income	Net:Gross Income
9999	999	9000	90.01%

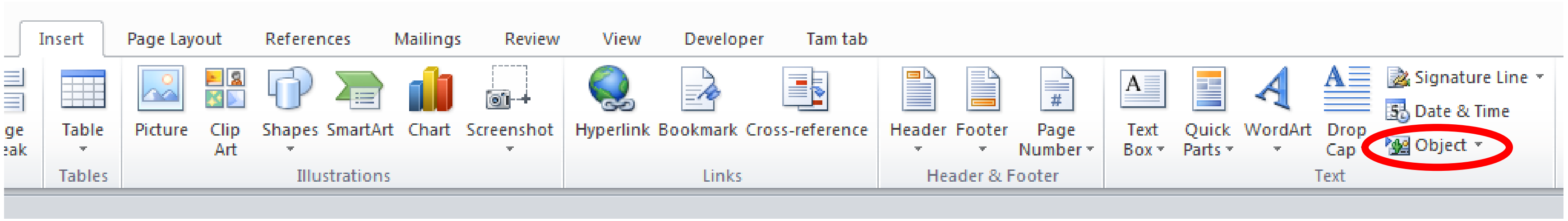
- The information is imported via 'linking' into a Word document so it can be formatted



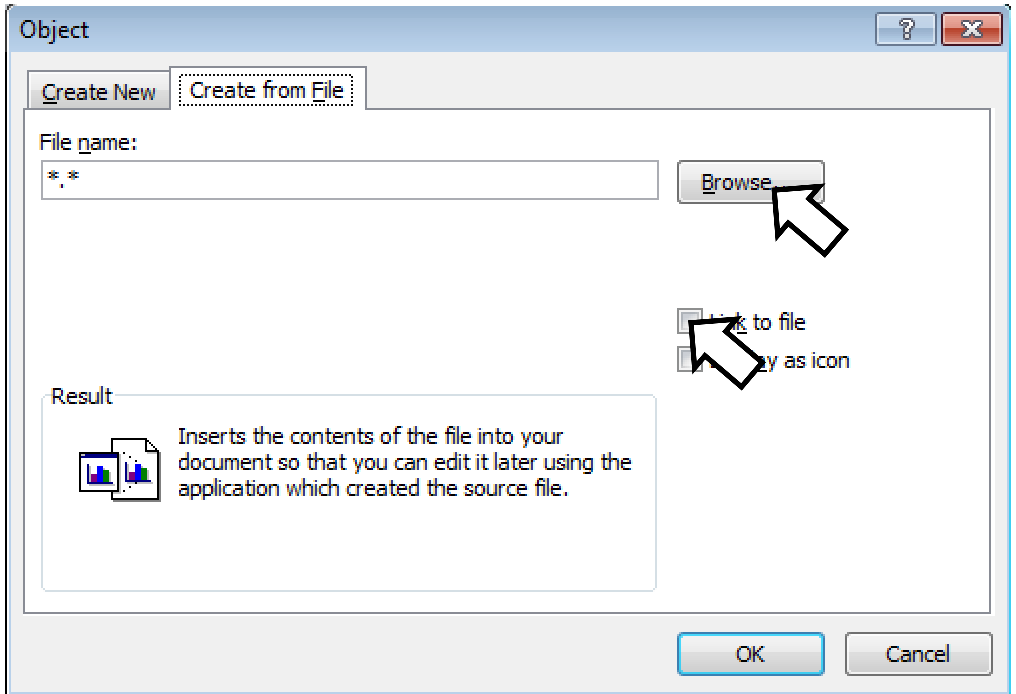


Alternate Approach For Linking Documents

- Insert->Object



- Create from file->Link to file->Browse



File name: *.*

Link to file

Display as icon

Result

Inserts the contents of the file into your document so that you can edit it later using the application which created the source file.

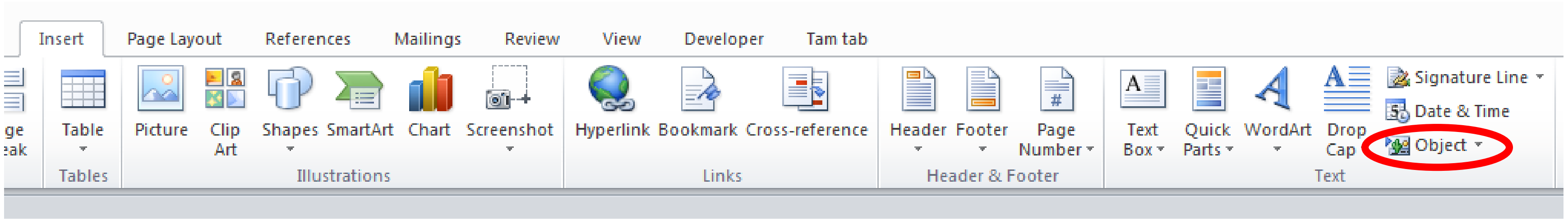
Embedding Office Documents

- Copy all of the information from one document to another document (e.g., embed a copy of an Excel spreadsheet inside of Word document).
- The capabilities of another application such as Excel can be used inside of Word (formulas, updated calculations etc.)
- Pro
  - The document with another document embedded is complete. That file can copied, shared etc.
- Con:
  - The embedded document is copied and if the file is large a great deal of space can be duplicated
  - If the original document is changed (spreadsheet updated), the changes are not reflected in the document that contains the embedded document (word document containing the spreadsheet)

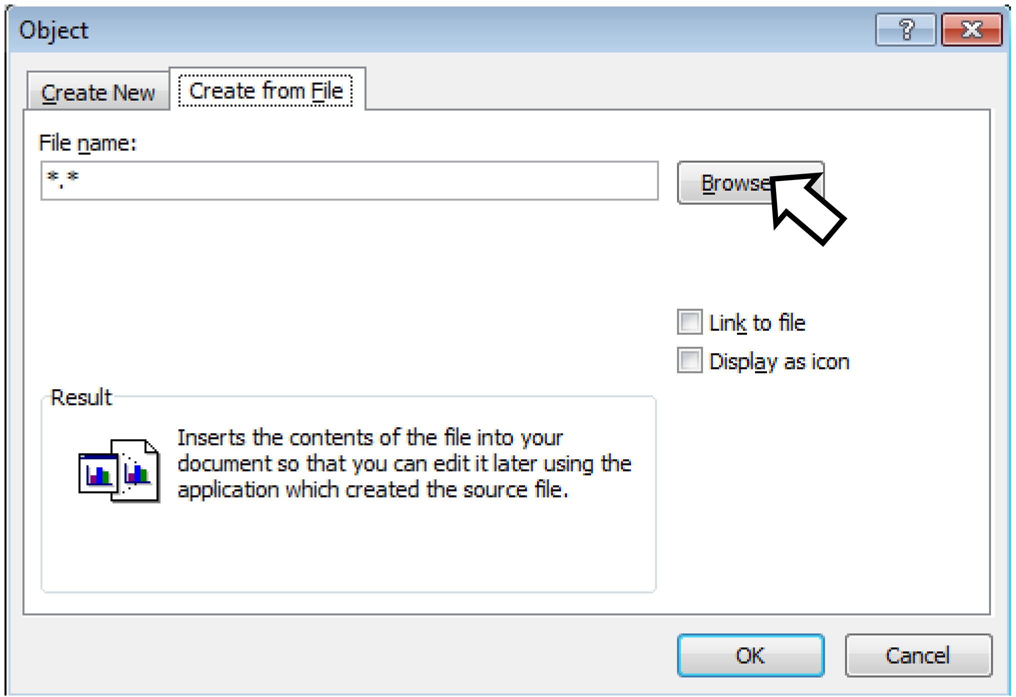
For more information: http://support.microsoft.com/kb/76993

How To Embed One Document In Another

- Insert->Object



- Create from file->Browse



File name: *.*

Result

Inserts the contents of the file into your document so that you can edit it later using the application which created the source file.

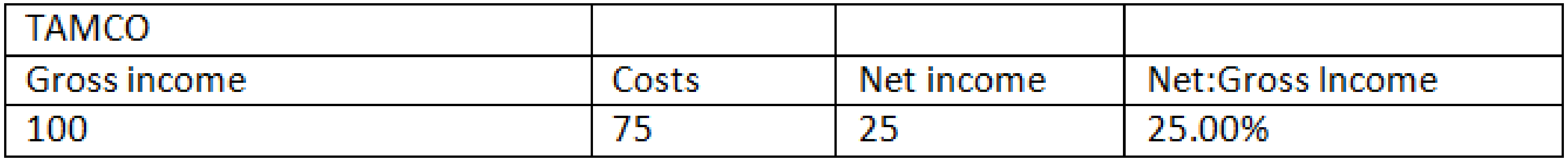
Example: Using Branches, Loops, Strings

- Suppose that this data is not only extensive (many tables), it is also dynamic (changes over time).
- You need to analyze the data and highlight the important information
  - Which companies may be a good investment?
  - Which criteria make it a good investment?
  - With a real example many companies are listed on the stock exchange
  - For each company there can be a great deal of background information
    - “Minimum” current stock price, dollar value of change
    - Other information could include detailed financial statements (e.g., how much money is that company making, what’s the ratio of debt vs. cash etc.)

Example: Using Branches, Loops, Strings (2)

- (Note: the problem of having to sort through large sets of data is not unique to finance and investing)
  - E.g., Suppose you want to work at companies that are hiring based on certain qualifications (“MS-Word VBA programming”) or provide certain benefits (“Unlimited vacation time”)

Example: Background Knowledge



TAMCO			
Gross income	Costs	Net income	Net:Gross Income
100	75	25	25.00%

- Gross income: total income earned (total sales dollars)
- Costs: expenses of running the business
  - Cost to purchase items sold
  - Salaries
  - Rent
  - Utilities
  - Taxes etc.
- Net income: Gross income minus costs
- Ratio of net to gross income
  - Ratio = (Net income) / (Gross income) * 100

Example Requirements

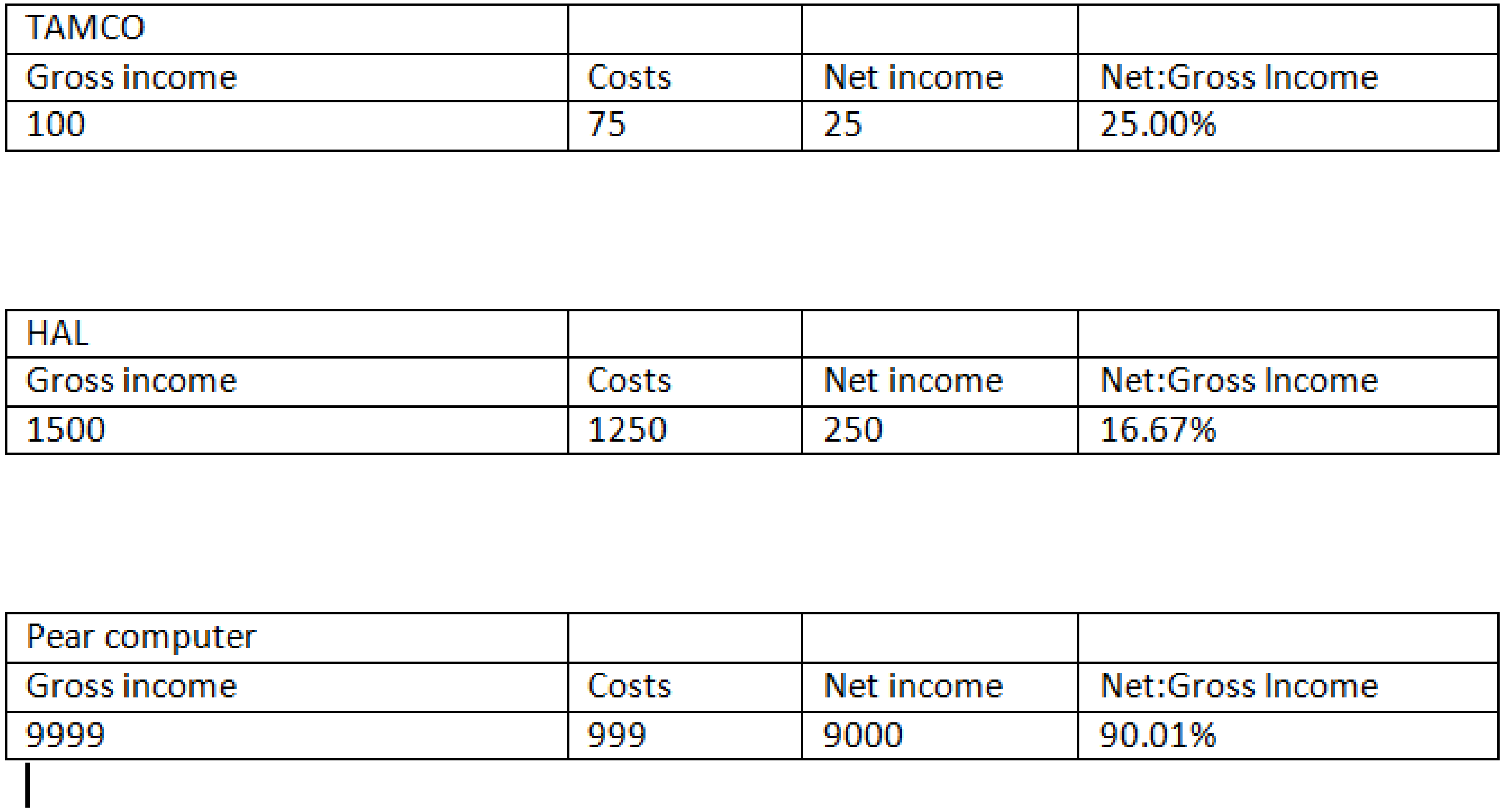
- Highlight companies with a net income that is $250 or greater (red)
- Highlight companies whose ratio of net to gross income is 25% or greater (blue)
- If a company meets both requirements draw extra attention (bold, larger font, extra comments - "BUY THIS!!!"

TAMCO			
Gross income	Costs	Net income	Net:Gross Income
100	75	25	**25.00%**

HAL			
Gross income	Costs	Net income	Net:Gross Income
1500	1250	**250**	16.67%

Pear computer <== BUY THIS!!!			
Gross income	Costs	Net income	Net:Gross Income
9999	999	**9000**	**90.01%**

Example File: Before

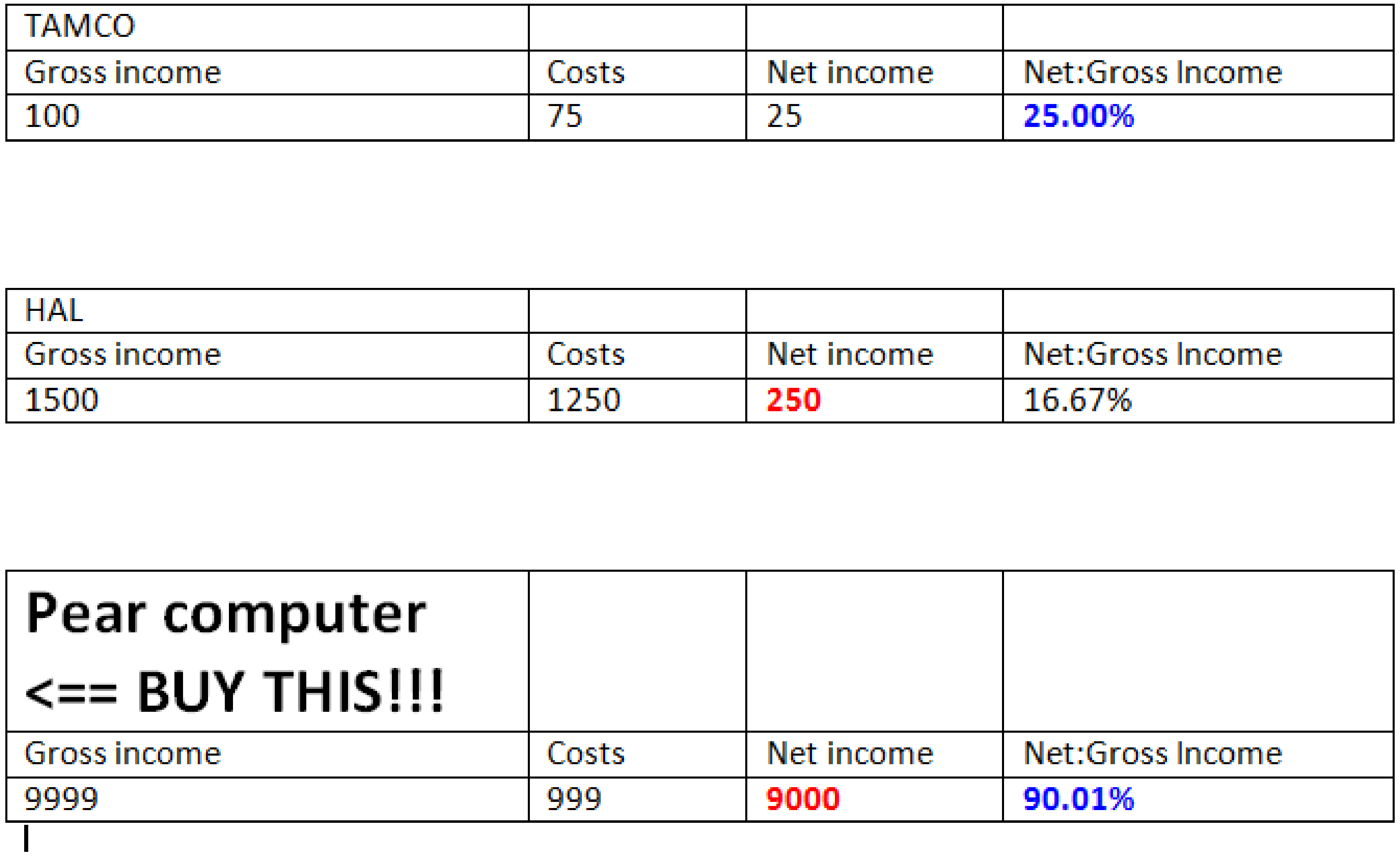


TAMCO			
Gross income	Costs	Net income	Net:Gross Income
100	75	25	25.00%

HAL			
Gross income	Costs	Net income	Net:Gross Income
1500	1250	250	16.67%

Pear computer			
Gross income	Costs	Net income	Net:Gross Income
9999	999	9000	90.01%

Example File: After



TAMCO			
Gross income	Costs	Net income	Net:Gross Income
100	75	25	**25.00%**

HAL			
Gross income	Costs	Net income	Net:Gross Income
1500	1250	**250**	16.67%

Pear computer <== BUY THIS!!!			
Gross income	Costs	Net income	Net:Gross Income
9999	999	**9000**	**90.01%**

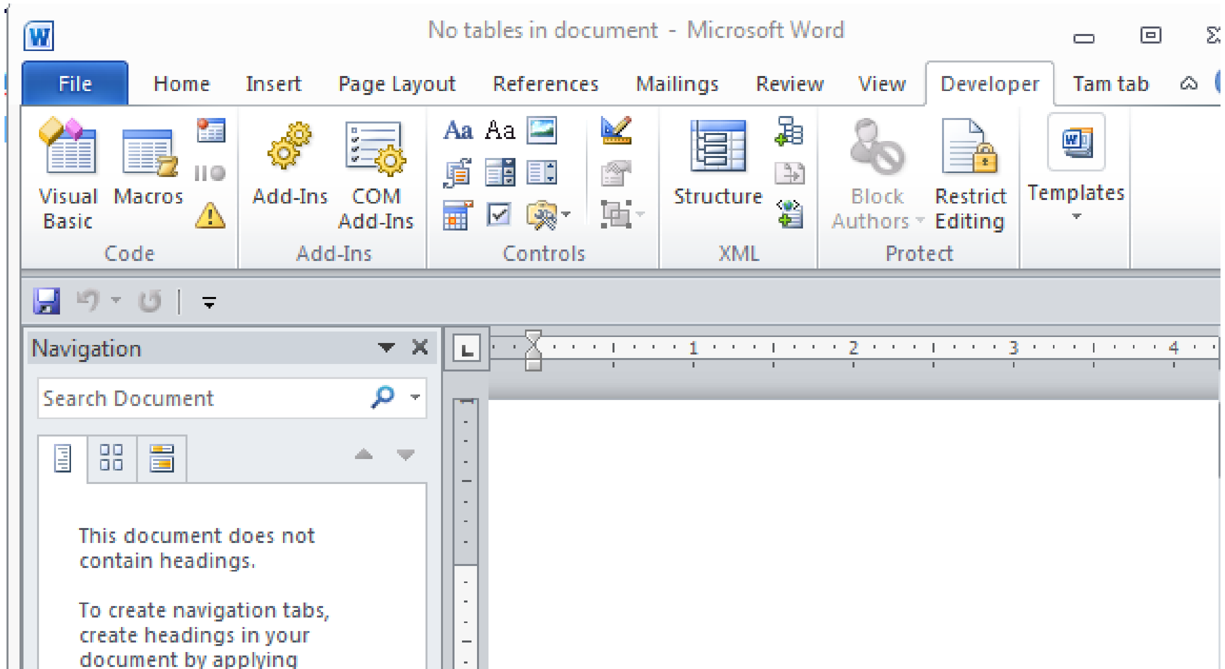
Highlighting Important Table Data: VBA Solution

- **Word document containing the macro:** “`tableHighLight.docm`”

```
Sub tableHighlight()
    Const MIN_INCOME = 250
    Const MIN_RATIO = 25
    Const MATCH = 0
    Dim CurrentTable As Integer
    Dim NumTables As Integer
    Dim NetString As String
    Dim NetNumber As Integer
    Dim RatioString As String
    Dim RatioNumber As Integer
    Dim CompanyName As String
    Dim TempString As String
    Dim StringLength As Integer
    Dim i As Integer
```

Highlighting Important Table Data: VBA Solution (2)

```
' No tables to analyze, end the program
NumTables = ActiveDocument.Tables.Count
If NumTables = 0 Then
     ActiveDocument.ActiveWindow.Caption = "Error: No _
       tables in document!"
     Exit Sub
End if
```

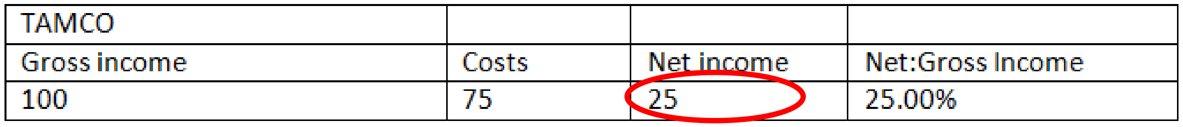


This document does not contain headings.

To create navigation tabs, create headings in your document by applying

Highlighting Important Table Data: VBA Solution (2)

```
For CurrentTable = 1 To NumTables Step 1
    NetString =
       ActiveDocument.Tables(CurrentTable). _
       Rows(3).Cells(3).Range.Text
    StringLength = Len(NetString)
    ' column labels        0 1 2 3
    ' data in each  column 1 2 ? ?
    ' left("12??", (4-1 = 2)) so yields "12"
    TempString = Left(NetString, (StringLength - 1))
```



TAMCO			
Gross income	Costs	Net income	Net:Gross Income
100	75	25	25.00%

Highlighting Important Table Data: VBA Solution (3), Net Income

```
If IsNumeric(TempString) Then
    NetNumber = CLng(TempString)
Else
    MsgBox ("Error non-numeric net income informatio
    NetNumber = 0
End If
                 Const MIN_INCOME = 250
If (NetNumber >= MIN_INCOME) Then
    ActiveDocument.Tables(CurrentTable). _
     Rows(3).Cells(3).Range.Select
    With Selection
        .Font.Bold = True
        .Font.Color = wdColorRed
    End With
 End If
```

Net income
25

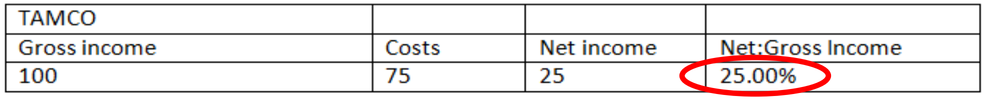
Net income
250

Net income
9000

TAMCO			
Gross income	Costs	Net income	Net:Gross Income
100	75	25	25.00%

Highlighting Important Table Data: VBA Solution (4), Ratio (Net:Gross)

```
RatioString = ActiveDocument.Tables(CurrentTable). -
  Rows(3).Cells(4).Range.Text
StringLength = Len(RatioStri
```

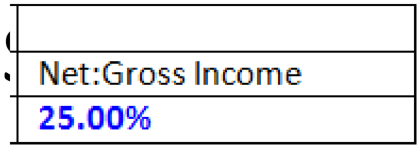


TAMCO			
Gross income	Costs	Net income	Net:Gross Income
100	75	25	25.00%

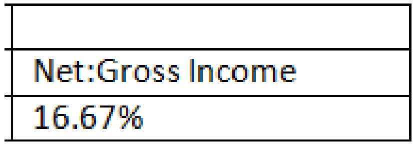
```
TempString = Left(RatioString, (StringLength - 3))
If IsNumeric(TempString) Then
    RatioNumber = CLng(TempString)
Else
  MsgBox ("Error non-numeric information in ratio of net _
    income:gross")
RatioNumber = 0
End If
```

Highlighting Important Table Data: VBA (5), Ratio

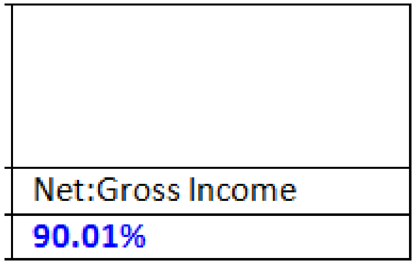
```
  Const MIN_RATIO = 25
If (RatioNumber >= MIN_RATIO) Then
    ActiveDocument.Tables(CurrentTable). _
      Rows(3).Cells(4).Range.Select
             With Selection
                .Font.Bold = True
                 .Font.Color = wdColorBlue
            End With
        End If
```



Net:Gross Income
25.00%



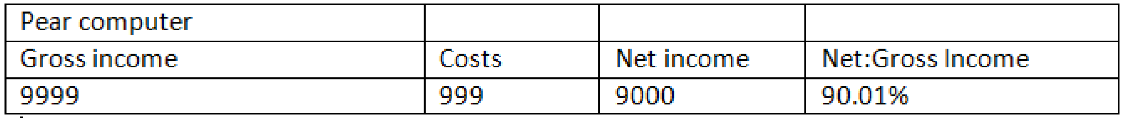
Net:Gross Income
16.67%



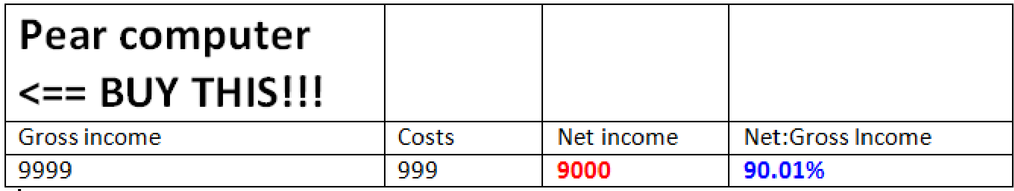
Net:Gross Income
90.01%

Highlighting Important Table Data: VBA Solution (6)

```
  If (RatioNumber >= MIN_RATIO) And (NetNumber >= MIN_INCOME) _
  Then
      CompanyName =
        ActiveDocument.Tables(CurrentTable). _
          Rows(1).Cells(1).Range.Text _
          CompanyName = CompanyName & "<== BUY THIS!!!“
        ActiveDocument.Tables(CurrentTable). _
          Rows(1).Cells(1).Range.Text = CompanyName

      ActiveDocument.Tables(CurrentTable). _
        Rows(1).Cells(1).Range.Select
        With Selection
            .Font.Size = 20
            .Font.Bold = True
        End With
   End If
Next  ' Examine the next table
```



Pear computer			
Gross income	Costs	Net income	Net:Gross Income
9999	999	9000	90.01%



Pear computer <== BUY THIS!!!			
Gross income	Costs	Net income	Net:Gross Income
9999	999	**9000**	**90.01%**

Printing: Single

- Printing a single document (currently opened, active MS-Word document)
- **Word document containing the macro example:** “`singleDocumentPrint.docm`”

```
Sub PrintSingleDocument()
    ActiveDocument.PrintOut
End Subs
```

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:** “`multiDocumentPrint.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

The ‘`Dir`’ Function

- A directory = Folder
- The Dir function allows access to the files in a directory
- Examples:
  - Check if a file exists in a particular location
  - Loop through all the files in a directory and process each file

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

- **Word document containing the macro example:** `openExistingDocument.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 =
           "Path/filename cannot be empty"
```

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

```
    ' 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
```

Example: Using `Dir` To Access Each File In A Directory

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

```
Sub DirectoryLoop()
    Dim directoryPath As String
    Dim currentFile As String

    directoryPath = InputBox
      ("Enter full path of search folder")
    currentFile = Dir(directoryPath & "*.*")
    Do While currentFile <> ""
        MsgBox (currentFile)
        currentFile = Dir
     Loop
End Sub
```

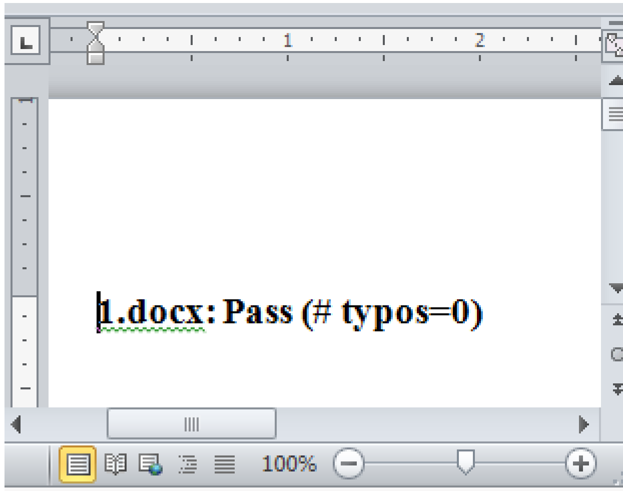
Revision Of An Earlier Example

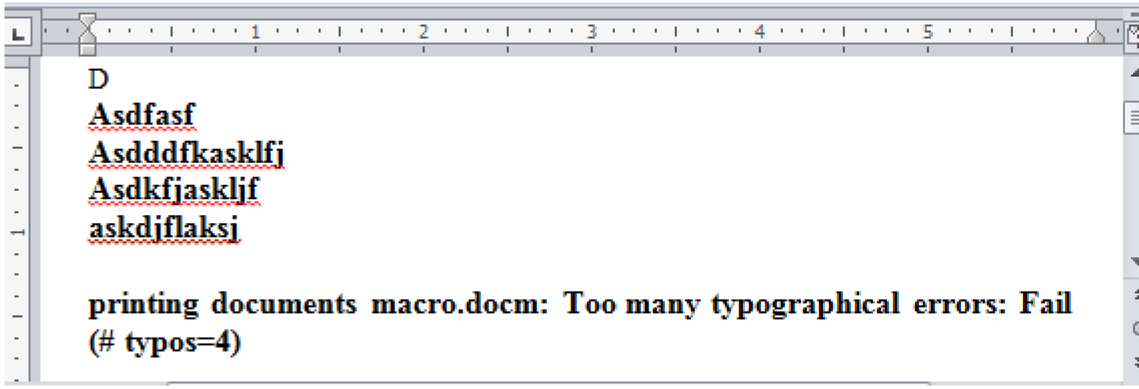
- The original version created a single document and creating an accompanying marking document.
- This new version will automatically mark all the documents in a user-specified folder and insert the marking information at the bottom of each document.
- Details:
  - Open each document in the folder
  - 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 end 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**: “`markAllFolderDocuments.docm`”

```
Sub markAllFolderDocuments()
   Const MAX_TYPOS = 3
   Const LARGER_FONT = 14
   Dim directoryPath As String
   Dim currentFile As String
   Dim totalTypos As Integer
   Dim feedback As String
```

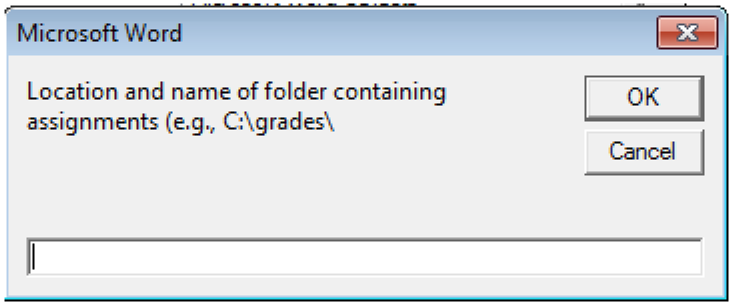

1.docx: Pass (# typos=0)


D
Asdfasf
Asdddfkasklfj
Asdkfjaskljf
askdjflaksj

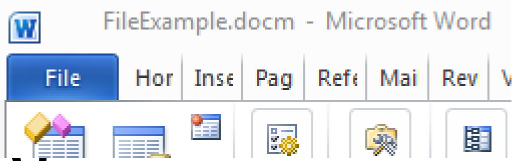
printing documents macro.docm: Too many typographical errors: Fail (# typos=4)

Revised Marking Program (2)

```
directoryPath = InputBox("Location and name of folder
  containing assignments (e.g., C:\grades\")
If (Len(directoryPath) = 0) Then
    MsgBox ("No path specified, looking in default
      location C:\temp\")
    directoryPath = "C:\temp\"
End If
```


Microsoft Word

Location and name of folder containing assignments (e.g., C:\grades\



Revised Marking Program (3)

currentFile

FileExample.docm

Feedback

<Enter>

Feedback

<Enter>
FileExample

```
currentFile = Dir(directoryPath & "*.doc*")
Do While currentFile <> ""
    feedback = vbCr   'Comments on a separate line
    feedback = feedback & currentFile
    currentFile = directoryPath & currentFile
    Documents.Open (currentFile)
    totalTypos = ActiveDocument.SpellingErrors.Count

    'Marking is based solely on typos
    If (totalTypos > MAX_TYPOS) Then
        feedback = feedback & ": Too many typographical
         errors: Fail (# typos=" & totalTypos & ")"
    Else
        feedback = feedback & ": Pass (# typos=" &
          totalTypos & ")"
    End If
```

Feedback

<Enter>
FileExample
Pass (# typos=1)

Revised Marking Program (4)

```
        'Comments appear at end of document
        Selection.EndOf Unit:=wdStory
        Selection.Text = feedback

        'Visually highlight the feedback text
        Selection.Font.Bold = True
        Selection.Font.Size = LARGER_FONT
        ActiveDocument.Close (wdSaveChanges)
        currentFile = Dir  'Access next document
        Loop  'Each loop: open and mark a document each
End Sub
```

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)

- Strings
  - What is a string
  - How to access the individual elements of a string
  - How common and useful string functions work
- The advantages of linking vs. embedding MS-Office documents
- How to print documents from VBA programs
- How to use the ‘`Dir`’ function to access a folder/directory