VBA Programming: Part II

How to win friends using functions and influence people using methods (not really but it sounds catchier than learning how to use pre-created functions and methods in VBA)

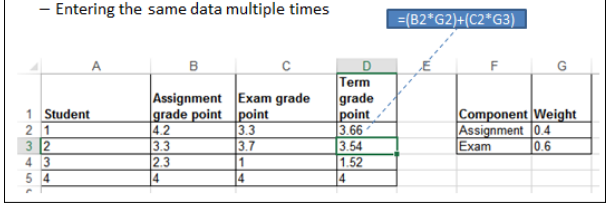
Online support: https://support.office.com/en-US/article/create-or-run-a-macro-c6b99036-905c-49a6-818a-dfb98b7c3c9c

Review: Lookup Tables (For Constants)

- Excel: Lookup tables are used to define values that do not typically change but are referred to in multiple parts of a spreadsheet.

Lookup Tables

- As the name implies it contains information that needs to be referred to ("looked up") in a part of the spreadsheet.
- Can be used to address some of the issues related to the previous example:
  - Clarity
  - Entering the same data multiple times



	A	B	C	D	E	F	G
1	Student	Assignment grade point	Exam grade point	Term grade point		Component	Weight
2	1	4.2	3.3	3.66		Assignment	0.4
3	2	3.3	3.7	3.54		Exam	0.6
4	3	2.3	1	1.52			
5	4	4	4	4			

Named Constants

- They are similar to variables: a memory location that's been given a name.
- Unlike variables constants *cannot* change.
  - VBA makes a memory location **unchanging** when the **`Const`** keyword is used
- The naming conventions for choosing variable names generally apply to constants but constants should be all UPPER CASE. (You can separate multiple words with an underscore).
  - This isn't a usual Visual Basic convention but since it's very common with most other languages, you will be required to follow it for this class.
- Example `const PI As Double = 3.14`
  - **PI = Named constant**, 3.14 = Unnamed constant
- They are capitalized so the reader of the program can quickly distinguish them from variables.

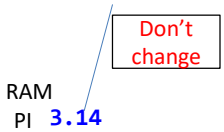
Declaring Named Constants

- **Learning objective:** how to declare constant.
- **Format**:

  `const <Name of constant> as <Data type> = <Expression>`[1]

  JT: it's preceded by the keyword '`const`' to indicate that it is a constant/unchanging.
- **Example**:

```
const PI As Double = 3.14
PI = 1.34
```



1 The expression can be any mathematical operation but can't be the result of a function call

Why Use **Named Constants**

- They can make your programs easier to read and understand
- Example:

```
Income = 315 * 80
```
No ☹

Vs.

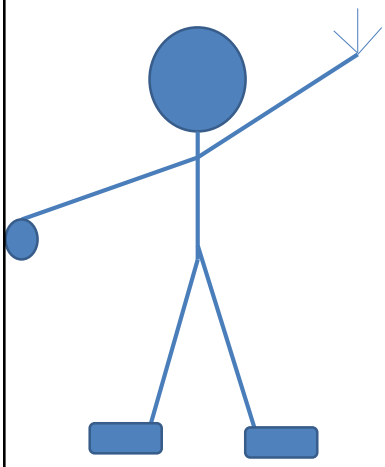
```
Income = WORKING_DAYS_PER_YEAR * DAILY_PAY
```
Yes ☺

Why Use **Named Constants**

- Updating the initial value for a named constant will update it throughout the program wherever the constant is referred to.
- **Learning objective**: Shows you how/why use named constants
- **Word document containing the macro**: `1declaringAConstant.docm`

```
Const TAX_RATE As Double = 0.2
Dim grossIncome As Long
Dim taxOwed As Long
Dim incomeAfterTax As Long
grossIncome = 100
taxOwed = grossIncome * TAX_RATE
incomeAfterTax = grossIncome - taxOwed
MsgBox (grossIncome & " " & taxOwed & " " & _
  incomeAfterTax
```

Example: A Person



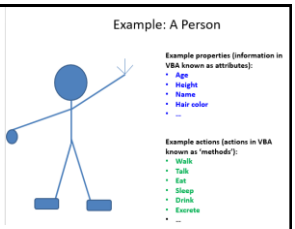
Example properties (information in VBA known as attributes):

- **Age**
- **Height**
- **Name**
- **Hair color**
- **…**

Example actions (actions in VBA known as 'methods'):

- **Walk**
- **Talk**
- **Eat**
- **Sleep**
- **Drink**
- **Excrete**
- …

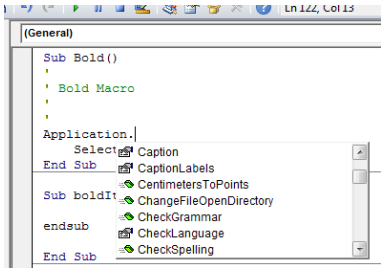
VBA Object



- Similar to everyday objects VBA-Objects have **attributes** and **actions**
  - **Attributes**: **information** that describe the object.
    - E.g., the **name** of a document, **size** of the document, **date modified**, **number of words** etc.
  - **Methods**: **actions** that can be performed (sometimes referred to as 'functions' or 'procedures' or 'subroutines' depending upon the language).
    - E.g., **save**, **print**, **spell check** etc.

Common VBA Objects

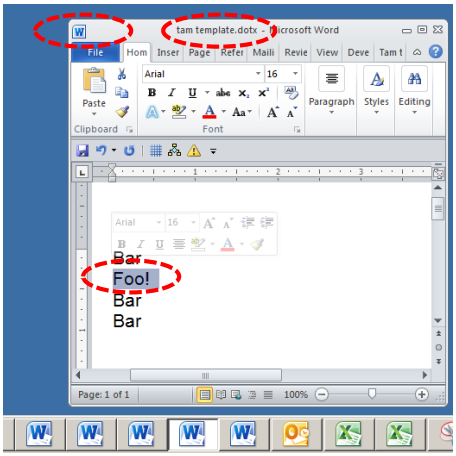
- **Application**: the MS-Office program running (for CPSC 203 it will always be MS-Word)
- **ActiveDocument**
- **Selection**
- When enter one of these keywords in the editor followed by the 'dot' you can see more information.



Take advantage of the benefits of VBA:

1. The list of properties and methods is a useful reminder if you can't remember the name
2. If you don't see the pull down then this is clue that you entered the wrong name for the object

Example: What Are The Three Objects

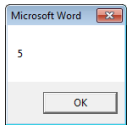


- Application:
  - **MS-Word**
- Active/current Document:
  - "**tam template**"
- Selection
  - "**Foo!**"

Using Pre-Built Capabilities/Properties Of Objects

- **Use the "dot-operator" to access the attribute or method**
- **Format**:
  <Object name>.<method or attribute name>
- **Simple illustrative example** (this one is not in a Word doc)

```
Sub ApplicationTest()
    MsgBox (Application.Windows.Count)
End Sub
```



`Application.Windows.Count` ← Property of Window:
• Number

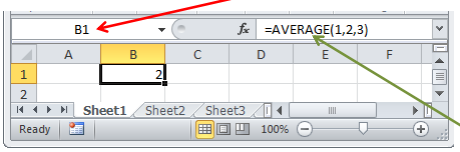
Object referred to: 'Application'

Accessing the Windows property of Word (the application)
• Info about the windows currently opened

Attributes Vs. Methods/Functions

- Recall
  - **Property**: information about an object
  - **Method**: capabilities of an object (possible actions)

Property: current cell

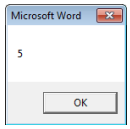


Using the 'average()' function

The Application Object

- As mentioned this object is the VBA application running e.g. MS-Word
- **Learning objective**: accessing a part (attribute) of an object
- **Program illustrating an example usage**: `2applicationObject.docm`

```
Sub ApplicationTest()
    MsgBox (Application.Windows.Count)
End Sub
```



`Application.Windows.Count`

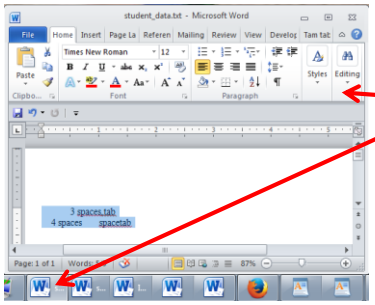
Property of Window:
- **Number**

Object referred to: 'Application'

Accessing the Windows property of Word (the application)
- **Info about the windows currently opened**

Introduction To 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

– Because it may be easy to confuse documents it's best to **only have a single Word document open** when writing a VBA program.

Attributes Of The `ActiveDocument` Object

- **`Application`**: the application/program associated with the document (useful if a VBA macro is linking several applications):details on next slide
- **`Content`**: the data (text) of the currently active document (needed if you want to perform a text search 'Find' in a VBA program):details later in these notes
- **`Name`**: the (file) name of the current document (useful for determining the active document if multiple documents are currently open): next slide
- **`Path`**: the save location of the active document e.g. C:\Temp\ :details on next slide
- **`FullName`**: the name and save location of the current document :details on next slide
- **`HasPassword`**: true/false that document is password protected: details on next slide
- **`SpellingChecked`**: true/false document has been spell checked since document was last edited: :next slide
- **`SpellingErrors.Count`**: the number of typographical errors

Note: Information for these attributes/properties can be viewed by passing the information as a parameter to a message box

Format: `MsgBox (ActiveDocument.<Attribute Name>)`

Example: `MsgBox (ActiveDocument.SpellingErrors.Count)`

Example Of Accessing **Attributes**

- **Learning objective**: accessing some common attributes of the `ActiveDocument` object (e.g. accessing document name, path).
- **Program illustrating an example usage**: `3activeDocumentAttributes.docm`

```
Sub activeDocumentAttributes()
    MsgBox (ActiveDocument.Application)
    MsgBox (ActiveDocument.Name)
    MsgBox (ActiveDocument.Path)
    MsgBox (ActiveDocument.FullName)
    MsgBox ("Spell checked? " & _
      ActiveDocument.SpellingChecked)
    MsgBox ("Password protected? " & _
      ActiveDocument.HasPassword)
    MsgBox ("# typos=" & ActiveDocument.SpellingErrors.Count)
End Sub
```

Some **Methods** Of The `ActiveDocument` Object

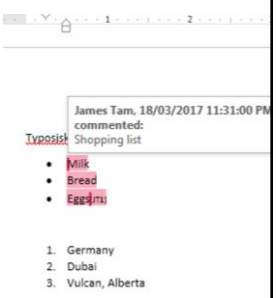
- `Checkspelling()`: exactly as it sounds: next slide
- `Close()`: closes the active document (different options available)
- `CountNumberedItems()`: number of bulleted and numbered elements: next slide
- `DeleteAllComments()`: removes comments from the current document: next slide
- `Printout()`: prints current active document on the default printer : next slide
- `Save()`: saves the current document under the same name: next slide
- `SaveAs2()`: saves the current document under a different name: : next slide
- `Select()`: select all text in the active document
- `SendMail()`: sends an email using MS-Outlook, the currently active document becomes a file attachment
- `ComputeStatistics(wdStatisticWords)`: counts the number of words in a document.

```
Const wdStatisticWords As Long = ?
```

Predefined Named constant (MS for Word)

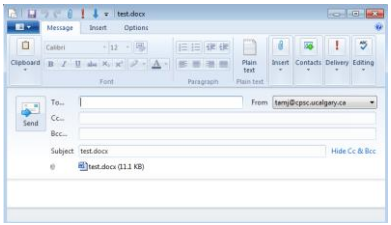
Example Of Using **Methods**

- **Learning objective:** accessing some common methods of the `ActiveDocument` object.
- **Program illustrating an example usage:** `4activeDocumentMethods.docm`




```
Sub activeDocumentAttributes()
    ActiveDocument.CheckSpelling
    MsgBox (ActiveDocument.CountNumberedItems)
    ActiveDocument.DeleteAllComments
    ActiveDocument.PrintOut
    ActiveDocument.Save
    ActiveDocument.SaveAs2 ("Copy")
    ActiveDocument.Range.ComputeStatistics(wdStatisticWords)
End Sub
```

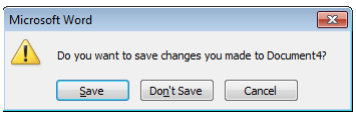
ActiveDocument.SendMail()



- 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)

Closing The Active Document

- Default action when closing a MS-Word document that has been modified (prompt)



Another named constant

- VBA code to close a document in this fashion:

```
ActiveDocument.Close (wdPromptToSaveChanges)
```

One closing option:
Allow the user to save

Reinforcing Example: Using **Pre-Defined Constants** (Closing Documents)

- **Learning objectives**: writing a VBA instruction to close the currently active Word document, reinforcing the value of named constants (in this case the constants have been predefined by Microsoft).
- **Word document containing the macro**: "`5closingActions.docm`"

```
Sub ClosingActions()
    ActiveDocument.Close (<Selected option for closing action>)
```

1

'Choose one action
wdPromptToSaveChanges 1
wdDoNotSaveChanges 2
wdSaveChanges 3

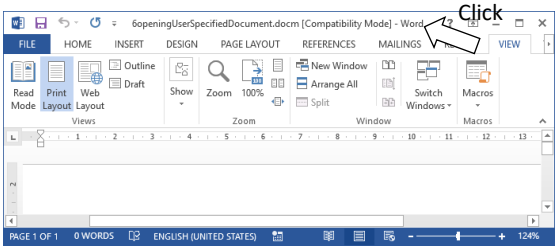
Prompt to save

Close and not save

Automatically save and close

Opening A Document

- Given the name of a document a VBA program can be used to open that document in Word.
- **Learning objective**: opening a user specified Word document in the same folder as the currently active document resides.
  - To avoid confusion make sure that the currently active document is the one containing this VBA program.
  - Click on the window for this Word document prior to running this program.



Opening A Document (2)

- **Word document containing the macro**: `6openingUserSpecifiedDocument`

```
Sub findHighLight()
    Dim documentName As String
    Dim path As String

    path = ActiveDocument.path
    documentName = InputBox("Name of Word document: ")
    Documents.Open (path & "\" & documentName & ".docx")
End Sub
```

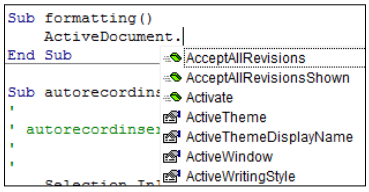
Suffix for regular Word 2007 document

C:\203\VBA\part2

6openingUserSpecifiedDocument

Formatting A Document

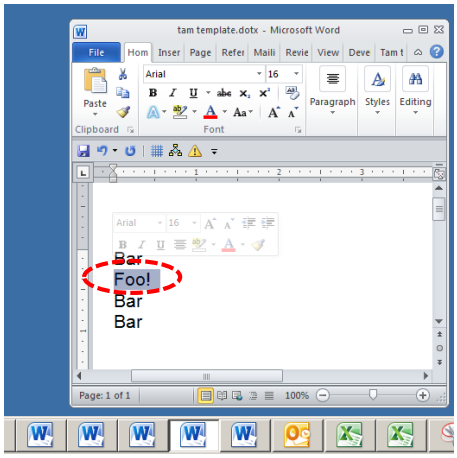
- Selected text:
  - Only format the currently selected text via the '`Selection`' object).
- Entire document:
  - You first need to specify the document or part of a document to be formatted
  - One way is through the '`ActiveDocument`' object



  - Then choose the '`Select`' method of that document.
    - Review: it's a method and not a property because it applies an action: select = selecting the text of the entire document

Introduction To The `Selection` Object

- This is the currently selected text in a document.
  - It may be empty (nothing selected)



Basic **Attributes** Of The `Selection` Object

- **`Font.Name`**: specifies the type (name) of font
- **`Font.Size`**: specifies the font size
- **`Font.ColorIndex`**: specifies the color of the font
- **`Font.UnderLine`**: specifies the type of underlining to be applied (or to remove underlining)
- **`Font.Bold`**: allows bolding to change (toggle or set)

Similar to how the Attributes/Properties of `ActiveDocument` Object affect only the currently active document these Attributes/Properties only take effect on the currently selected text (if there's any).

Using The **Selection Object Attributes**

- **Learning objective:** changing the properties of fonts in Word
- **Word document containing the macro**: `7selectionAttributes.docm`

```
Sub selectionObjectAttributes()
    ' Selection.Font.Underline = <Selection for underlining>
    ' wdUnderlineNone, wdUnderlineSingle
    ' e.g. Selection.Font.Underline = wdUnderlineSingle
    Selection.Font.Name = "Wingdings"  'Must be in quotes
    Selection.Font.Size = 36
    Selection.Font.ColorIndex = wdBlue
    ' Bolding options
    Selection.Font.Bold = wdToggle ' On/off
    Selection.Font.Bold = True     ' Turn on (false = off)
End Sub
```

Seeing Color (And Under Line Options)

- Use the 'auto complete' feature of VBA to view the options

Some **Methods Of The `Selection` Object**

- **`ClearFormatting`**: removes all formatting effects (e.g. bold, italics)
- **`TypeText`**: insert the text specified in the VBA program
- **`Delete`**: deletes any selected text
- **`EndKey`**: move the cursor to the end of the document (covered in a later and in a large example)
- **`HomeKey`**: move the cursor to the start of the document (covered in a later and in a large example)
- **`InsertFile`**: replace selection with text from the specified file (covered in a later example)

Similar to how the method of `ActiveDocument` Object affect only the currently active document these Attributes/Properties only take effect on the currently selected text (if there's any).

Using Simple **Methods Of The `Selection` Object**

- **Learning objective:** writing text into the active Word document.
- **Word document containing the macro**: `8selectionMethods.docm`
- Try running it with and without some text selected

```
Sub selectionObjectMethod()
   Selection.ClearFormatting
   Selection.TypeText ("My new replacement text")
End Sub
```

Formatting Text (Entire Active Document): An Example

- Objective:
  - Suppose you want to format a document in the following way
  - Entire document
    - Font = Calibri

Formatting: Entire Document

- This how an entire document can be selected.

```
ActiveDocument.Select
```

- Now for the 'selected text' (in this case it's the whole document) access the 'Font' property and the 'Name' property of that font and give it the desired name.

```
Selection.Font.Name = "Calibri"
```

- **Learning objective:** on the previous page.
- **Word document containing the macro**: 9formattingEntireDocument.docm

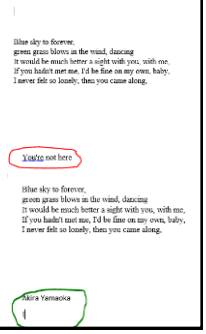
```
Sub formattingEntireDocument()
    ActiveDocument.Select
    Selection.Font.Name = "Wingdings"
    ' The previously covered ways of formatting selected text
    ' can then be run.
```

Writing Text To **Start**/**End**

- **Learning objective:** moving the selection (cursor) to the start or end of the currently active document.
- **Word document containing the macro**: `10selectionHomeEndKey.docm`
  - HomeKey docs: https://msdn.microsoft.com/en-us/library/office/ff192384.aspx
  - EndKey docs: https://msdn.microsoft.com/en-us/library/office/ff195593.aspx

```
Sub selectionHomeEndKey()
    Const SONG_TITLE = "You're not here"
    Const SONG_LYRICIST = "Akira Yamaoka"
    Selection.HomeKey Unit:=wdStory
    Selection.TypeText (SONG_TITLE)
    Selection.EndKey Unit:=wdStory
    Selection.TypeText (SONG_LYRICIST & vbCr & "!")
End Sub
```

vbCr (line break) = hitting enter

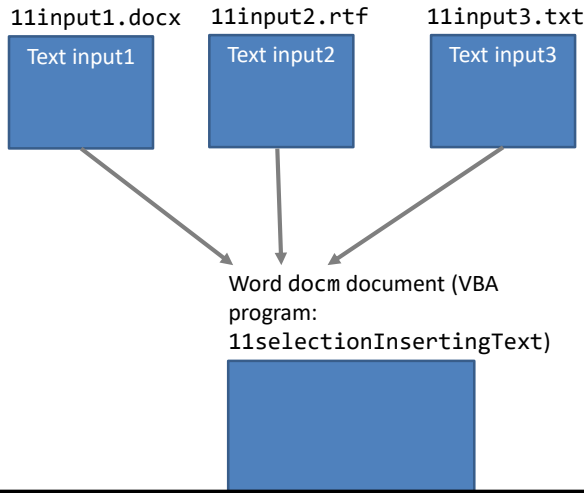


Automatically **Inserting Text** Into A Word Document

- **Learning objective:** inserting the text containing in different types of files (Word document, rich text file, plain text file) into the currently active Word document.
- **Word document containing the macro**: `11selectionInsertingText.docm`

Inserting Text Into One Document From Other Documents Via `InsertFile`

- Example files (must all be in the same folder)



Types of files used as input for the program in this example:

- `11input1.docx` = Word 2007 document
- `11input2.rtf` = Rich text file
- `11input3.txt` – Text document (no formatting).

"Finding" Things In A Document

- Example: 'find' can also be performed by using the `Selection` object but I prefer this approach.
- Find by using the `ActiveDocument` object
  - '`Find`' is an object that is part of the '`Content`' object of the '`ActiveDocument`'
  - `ActiveDocument.Content.Find`

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

Find: Single Replacement

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

- **Learning objective:** finding & replacing an instance of text, splitting long instructions onto multiple lines.
- **Word document containing the macro**: `12simpleFind.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"
```

More Complex Find And Replace: Case Sensitive

- **Learning objective**: a case sensitive find-replace.
- **Word document containing the macro**: `13findReplaceAllCaseSensitive.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

```
Complete original command
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.

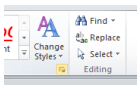
Finding And Replacing **Bold Font**

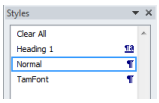
- **Learning Objective:** find/replace font effects (e.g. bolding).
- **Word document containing the macro**: `14findBold.docm`

```
'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 already have a set of pre-created formatting styles defined in MS-Word.



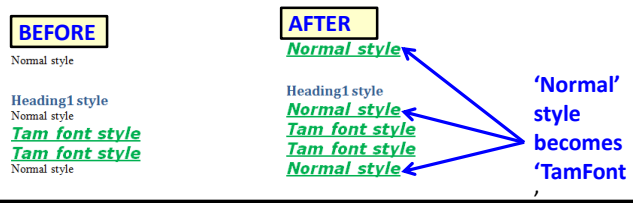


- You can redefine the characteristic of a style if you wish.
- Assume for this example that you wish to retain all existing styles and not change their characteristics.
- But you want to replace all *instances of one style* with another style e.g., all text that is '`normal`' is to become '`TamFont`'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

Finding/Replacing Formatting Styles (2)

- **Learning objective:** finding & replacing instances of a text style.
- **Word document containing the macro**:
  15findReplaceStyle.docm

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



Highlighting 'Found' Text

- **Learning objective:** finding text and visually highlighting the text to make it stand out.
- **Word document containing the macro**:
  16highlightingFoundText.docm

```
Sub findHighLight()
    Dim searchWord As String
    Dim fontSize As Long
    searchWord = InputBox("Enter word to emphasize: ")
```

Highlighting 'Found' Text (2)

```
With ActiveDocument.Content.Find
        .Text = searchWord
        .Replacement.Font.Bold = True
        .Replacement.Font.ColorIndex = wdBlue
        fontSize = Selection.Font.Size
        .Replacement.Font.Size = fontSize + 4
        .Forward = True
        .MatchCase = False
        .MatchWholeWord = True
        .Execute Replace:=wdReplaceAll
    End With
End Sub
```

Collections

- 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 in Word.
- 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
```

- You have actually seen an example using the Documents collection 'open' method:
  - `Documents.open<"Document name>")`

Types Of **Collections**

- Some Attributes/Properties of a document that return a collection.
  - **`Documents`**: access to all the currently open documents
  - **`Shapes:`** access to MS-Word shapes in a document (rectangles, circles etc.): *If there is time*
  - **`InlineShapes:`** access to images inserted into a Word document
  - **`Tables`**: access to all tables in a document: *If there is time*
    - 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 at the start of this section of notes (as part of the `Applications` object)

`Documents` Collection For **Printing: Multiple Documents**

- 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!
  - Also the program requires that you have at least 3 Word documents open when you run it.
    - Otherwise it will crash.
  - **Learning objective:** printing open documents (first three).
  - **Word document containing the macro example: "`17printThreeDocuments.docm`"**

```
Sub PrintDocumentsCollection()
    Documents.Item(1).PrintOut
    Documents.Item(2).PrintOut
    Documents.Item(3).PrintOut
End Sub
```

Accessing Shapes And Images (For Fun And Profit)

- (VBA specific)
  - `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

Some Attributes/Properties InlineShapes & Shapes

- Common to Both:
  - `Height`
  - `Width`
  - Example usage: `ActiveDocument.InlineShapes(1).Height`
  - `Count`
  - Example usage: `ActiveDocument.Shapes.Count`
- Shapes
  - `.Fill.ForeColor`
  - Example usage: `ActiveDocument.Shapes(6).Fill.ForeColor = vbRed`

Example: Accessing Shapes And Images

Learning objective: accessing and modifying the images (`InlineShapes`) and simple geometric shapes built into Word (`Shapes`).

Word document containing the complete macro: "`18accessingImagesFigures.docm`"

```
Sub accessImagesShapes()
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

```
' Double the height of the first and third image
   ActiveDocument.InlineShapes(1).Height = _
     ActiveDocument.InlineShapes(1).Height * 2
   ActiveDocument.InlineShapes(3).Height = _
     ActiveDocument.InlineShapes(3).Height * 2


   ' Modify the MS-Word 'Shapes
   ' Second shape increases in size by a factor of 4
   ActiveDocument.Shapes(2).Width = _
     ActiveDocument.Shapes(2).Width * 4

   ' Sixth shape colored red
   ActiveDocument.Shapes(6).Fill.ForeColor = vbRed
End Sub
```

Accessing Tables (If There Is Time)

- 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
    - i = 1 accesses the first table
    - i = 2 accesses the second table
  - `ActiveDocument.Tables(1).Sort`: sorts the first table in the document (default is ascending order)
- Some attributes & methods of the `Table` collection
  - `Count` (attribute) : the current number of tables in the collection
  - `Sort` (method) : will arrange the tables in order (default is ascending order)

Simple Example: Sorting Three Tables (If There Is Time)

- Instructions needed for sorting 3 tables

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

Before

Kirk, James Tam
Tam, James
Sheen, Charlie
Bond, James

Kirk, James Tam
Tam, James
Sheen, Charlie
Bond, James

Kirk, James Tam
Tam, James
Sheen, Charlie
Bond, James

Kirk, James Tam
Tam, James
Sheen, Charlie
Bond, James

After

Bond, James
Kirk, James Tam
Sheen, Charlie
Tam, James

Bond, James
Kirk, James Tam
Sheen, Charlie
Tam, James

Bond, James
Kirk, James Tam
Sheen, Charlie
Tam, James

Kirk, James Tam
Tam, James
Sheen, Charlie
Bond, James

Full Example: Table (If There Is Time)

- **Learning objective:** sorting tables using the `Tables` collection.
- **Word document containing the complete macro:** "`19sortingTables.docm`"

```
Dim numTables As Long
numTables = ActiveDocument.Tables.Count
MsgBox ("# tables to sort " & numTables)
MsgBox ("Sorting Table #1")
ActiveDocument.Tables(1).Sort
MsgBox ("Sorting Table #2")
ActiveDocument.Tables(2).Sort
MsgBox ("Sorting Table #3")
ActiveDocument.Tables(3).Sort
```

Result: Sorting Tables (If There Is Time)

- **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

After This Section You Should Now Know

- How to copy and run the pre-created lecture examples
- How the VB editor identifies programming errors
- How to create and execute simple VBA macros
  - You should know that macros can be automatically recorded but specifics will be covered in tutorial
  - Manually entering programs into the VB editor yourself
- How to create/use a Message Box "MsgBox"
- How to use basic mathematical operators in VB expressions
- How to create and use variables
- Naming conventions for variables

After This Section You Should Now Know

- Objects
  - Properties/attributes vs. methods
- Using common properties/attributes and methods of the following objects
  - `Application`
  - `ActiveDocument`
  - `Selection`
- Collections
  - What are they
  - What is the advantage in using them
  - Common examples found in Word documents

After This Section You Should Now Know (2)

- Using common collections in VBA
  - `Documents`
  - `Shapes`
  - `InLineShapes`
  - `Tables`
  - `Windows`

Images

- "Unless otherwise indicated, all images were produced by James Tam

slide 60