VBA: Tutorial Week 2

- Going over the requirements of Workbook exercise #4
- Displaying the number of spelling mistakes
- Changing font properties
- Writing text to a document
- Finding things in a document (text strings, font effects, formatting styles in Word)

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

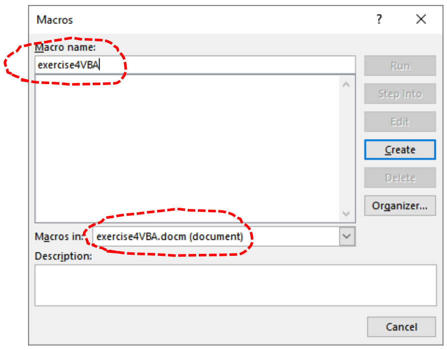
First Tutorial (Monday or Tuesday)

Workbook Exercise #4

- This is the first graded component where you must write a program from scratch.
- Don’t underestimate the time and level of challenge required by the previous assignments.
- Start as soon as you possible.
  - Even if not all the programming concepts have been covered yet, implement the parts that you have been taught.
  - Don’t wait until “everything has been fully covered” before starting or you may not have enough time.
- To ease the transition to implementing the assignment the solution for this workbook is smaller and less complex than the solution required for the assignment.

Starting The Workbook Exercise

- Write a VBA program called "`exercise4VBA`"
  - This will be the name of the Word document i.e. the full name is `exercise4VBA.docm`
  - This will be the name of the subroutine (i.e. the name that you give to your program)



Macro name: exercise4VBA

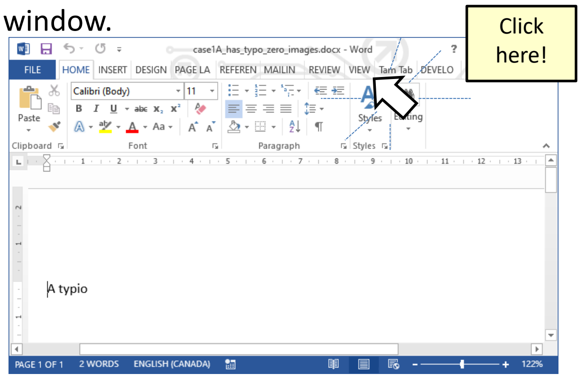
Macros in: exercise4VBA.docm (document)

Minimum Number Of Tests

- As a minimum use the test files provided in the description of the exercise to verify features of the program:
  - case1A_has_typo_zero_shapes (at least one spelling mistake and no inline shapes)
  - case1B_has_typo_3_shapes (at least one spelling mistake and multiple inline shapes)
  - case2_no_typos_1_shape (no spelling mistakes and a single inline image)
  - case3_no_typos_3_shapes (no spelling mistakes and multiple inline shapes)

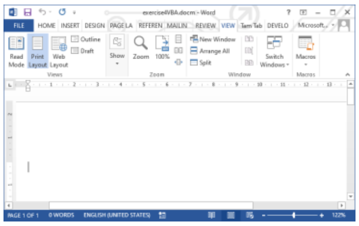
Running Your Program

- The Word document containing the test data (such as the 4 Word documents that you have been provided) must be the currently active Word document.
- You can make a Word document active by clicking on its window.



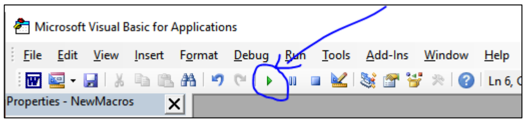
Running Your Program (2)

- When you run your solution to the exercise do not do so via the Word document containing the program.
  - NO! ☹


exercise4VBA.docm - Word
FILE HOME INSERT DESIGN PAGE LA REFEREN MAILIN REVIEW VIEW Tam Tab DEVELO
Read Mode Print Layout Web Layout Outline Draft Show Zoom 100% New Window Arrange All Split Switch Windows Macros
PAGE 1 OF 1 0 WORDS ENGLISH (UNITED STATES) 122%

If you click here to run the program then this Word document becomes the active one.

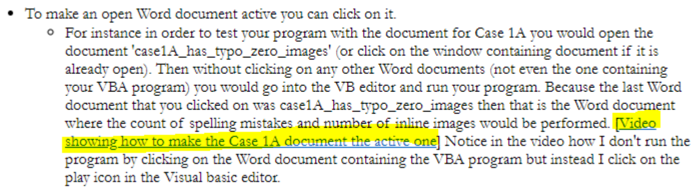
  - YES ☺


Microsoft Visual Basic for Applications
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
Ln 6, C
Properties - NewMacros

VBA tutorial notes by James Tam

Running Your Program (3)

- If are still unclear about how to make the proper Word document the active one when running your program then you can view the sequence of events in this sample run of a solution.
  - https://pages.cpsc.ucalgary.ca/~tamj/2022/203W/assignments/workbook_exercise4/case1A.mp4
- This link is also provided in the description of the exercise.

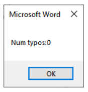


- To make an open Word document active you can click on it.
  - For instance in order to test your program with the document for Case 1A you would open the document 'case1A_has_typo_zero_images' (or click on the window containing document if it is already open). Then without clicking on any other Word documents (not even the one containing your VBA program) you would go into the VB editor and run your program. Because the last Word document that you clicked on was case1A_has_typo_zero_images then that is the Word document where the count of spelling mistakes and number of inline images would be performed. [Video showing how to make the Case 1A document the active one] Notice in the video how I don't run the program by clicking on the Word document containing the VBA program but instead I click on the play icon in the Visual basic editor.

VBA tutorial notes by James Tam

Features Of Your Program

- These popups will always appear regardless of the Word document that is the currently active one used as the test input file.
  - Count of the number of typographical errors in the document.
    - **Required format of the message[1]:**
      `<"Num typos: "> <`*`Actual # typographical mistakes`*`>`
    - **Example message:**



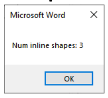
Microsoft Word

Num typos:0

OK

  - Count of the number of `InlineShapes` in the document.
    - **Required format of the message[1]:**
      <"Num inline shapes: "> <*# of inline shapes*>

Example message:



Microsoft Word

Num inline shapes: 3

OK

1 The Angled brackets don't appear in the actual popup.

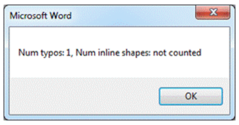
VBA tutorial notes by James Tam

Features Of Your Program (2)

- Based upon the number of typographical errors in the document and the number of inline shapes the program will display different popups.
  - **Case 1**: If there are <u>any typographical errors</u> then the number of inline shapes will not have an effect, the following message will appear:

```
Format:
<"Num typos: "> <Actual # typographical mistakes> <", Num inline
shapes: not counted">
```

`Actual message:`



Microsoft Word

Num typos: 1, Num inline shapes: not counted

OK

VBA tutorial notes by James Tam

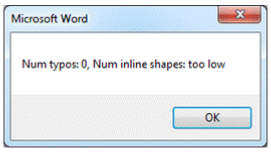
Features Of Your Program (3)

– **Case 2**: If there are no typographical errors and the number of inline shapes is fewer than 3 the following message will appear:

Format:

<"Num typos: "> *<Actual # typographical mistakes>* <", Num inline shapes: too low">

Actual message:



Num typos: 0, Num inline shapes: too low

VBA tutorial notes by James Tam

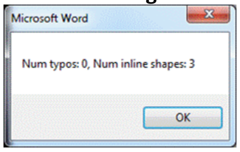
Features Of Your Program (4)

– **Case 3**: If there are no typographical errors and the number of inline shapes is 3 or greater following message will appear:

Format:

<"Num typos: "> *<# typographical mistakes>* <", Num inline shapes: "> *<# of inline shapes>*

Actual message:



Num typos: 0, Num inline shapes: 3

VBA tutorial notes by James Tam

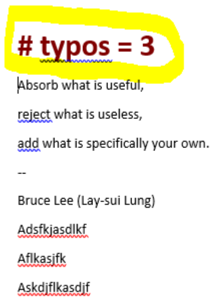
Second Tutorial (Wednesday or Thursday)

Activities In Tutorial

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

Example: Writing Text To A Document

- The following program will write the number of typographical mistakes at the top of the currently active Word document.
- The written text will have the following font characteristics:
  - Bolded text
  - Dark red in color
  - 24 point



typos = 3

Absorb what is useful,

reject what is useless,

add what is specifically your own.

--

Bruce Lee (Lay-sui Lung)

Adsfkjasdlkf

Aflkasjfk

Askdjflkasdjf

VBA tutorial notes by James Tam

Example 1

- **Example name:** `1TypoCountFormattingFonts`

```
Dim numTypos As Long
Dim message As String

numTypos = ActiveDocument.SpellingErrors.Count
message = "# typos = " & numTypos
MsgBox (message)
Selection.Font.Bold = True
Selection.Font.ColorIndex = wdDarkRed
Selection.Font.Size = 24
```

VBA tutorial notes by James Tam

Student Exercise #1

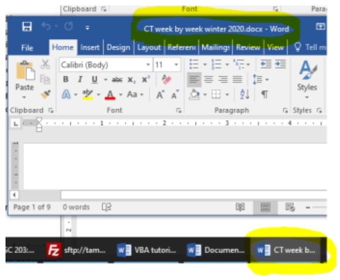
- **Starting document**: `exercise1_starting`
- **Document with solution**: `exercise1_solution`
- Open the starting document and add the following capabilities to the starting program:
  - Selected Text will have the following font effects applied:
    - Bold the text
    - Change the color of the text to violet
    - Change the font to Arial

Student Exercise #2

- **Starting document**: `exercise2_starting`
- **Document with solution**: `exercise2_solution`
- Open the starting document and add the following capabilities to the starting program:
  - Modify all the text in Word document in the following way:
    - Underline the text with a dotted line.
    - Change the color of the text to bright green.
    - Prompt the user for a font name and the selected text will be changed to this type of font.
    - Prompt the user for a name.
  - Write the name (that the user was prompted to enter in the starting program) after the last part of the text in the document "Ip Man 2"
  - After this is done automatically close and save the document without a user prompt.
  - JT: this exercise may appear fairly daunting for beginners but you can refer to the lecture notes: VBA Part II (focus on the `ActiveDocument` and the `Selection` objects)

Finding (Replacing) Text In A Document

- It can be done via the `ActiveDocument` object
- (This will perform the 'find' in the currently active Word document).
- If you have multiple Word documents open this may not always be the document containing your VBA program.
  – Close all Word documents except for your program to avoid confusion!



VBA tutorial notes by James Tam

Finding Things In A Document

- It's done through the '`Find`' method and the find is done in the `ActiveDocument` object.
  – i.e. some form of `With ActiveDocument.Content.Find`
  – Some types of things that a VBA program can find:
    - Text
    - Font effects
    - Styles (you have created)

VBA tutorial notes by James Tam

Finding Text: V1

- **Name of the document containing the example**: `2findReplaceOneCaseSensitive`
  - Features: replaces first instance, case sensitive find, find and replacement strings are constant strings (always the same)

```
Sub findReplaceOneCaseSensitive()
   With ActiveDocument.Content.Find
       .Text = "cool"
       .Replacement.Text = "kewl"
       .Execute MatchCase:=True, Replace:=wdReplaceOne
   End With
End Sub
```

Finding Text: V2

- **Name of the document containing the example**: `3findReplaceAllCaseInsensitive`
  - Features: replaces all instances, case insensitive find, find and replacement strings are variable (depend upon user input)

```
Sub findReplaceAllCaseInsensitive()
    Dim findWord As String
    Dim replacementWord
    findWord = InputBox("Word to find (try 'cOoL': ")
    replacementWord = InputBox("Replacement word (try 'aBx': ")
    With ActiveDocument.Content.Find
        .Text = findWord
        .Replacement.Text = replacementWord
        .Execute MatchCase:=False, Replace:=wdReplaceAll
    End With
End Sub
```

Writing To A Document

- It is accomplished using the `Selection` object and the `TypeText` method
- **To move the selection to the start of the document** (write text at the start): `Selection.HomeKey Unit:=wdStory`
- **To move the selection to the end of the document** (write text at the end): `Selection.EndKey Unit:=wdStory`
- `vbCr`
  - It is a Visual basic constant.
  - Stands for 'Visual Basic' carriage return.
  - Each each `vbCr` is equivalent to hitting enter.
  - Example: `"hi" & vbCr & "there"`
    - This puts 'hi' and 'there' on separate lines

VBA tutorial notes by James Tam

Writing A Text String To A Document

- **Name of the document containing the example**: `4writingToDocument`
- Features: writes the current location, at the start and end of the document. Spaces and carriage returns are concatenated into the text to be written.

```
 Sub writingToDocument()
    Dim textInsertAtEnd As String
    textInsertAtEnd = InputBox("Type in text to insert at end: ")
    Selection.TypeText ("Inserted where the cursor was located")
    Selection.HomeKey Unit:=wdStory
    Selection.TypeText ("New text inserted at the very top")
    Selection.EndKey Unit:=wdStory
    Selection.TypeText ("   " & textInsertAtEnd & vbCr & vbCr)
End Sub
```

VBA tutorial notes by James Tam

Visually Highlighting Written Text

- To make text stand out font effects (color, bolding, size, font type can be applied to the written text).
- This can be done via the attributes of the `Font` attribute of the `Selection` object.

Highlighting Written Text

- **Name of the document containing the example**: `5modifyingFontText`
- Features: highlights inserted text by increasing the font size, changing the color and changing the font to the type specified by the user.

```
Sub modifyingFontText()
   Dim insertionText As String
   Dim newFontSize As Long
   Dim newFontName As String
   insertionText = InputBox("Type in text to insert: ")
   newFontSize = InputBox("Size of the font: ")
   newFontName = InputBox("Name of font (e.g. Arial black): ")
```

Highlighting Written Text: 2

```
    insertionText = vbCr & insertionText & vbCr
    'Order is cruical! Right after moving the selection to the top is
    'when the statements to apply the font formatting effects should
    'occur. Inserting other VBA instructions between may change the
    'selection.
    Selection.HomeKey Unit:=wdStory
    Selection.Font.Bold = True
    Selection.Font.Size = newFontSize
    Selection.Font.Name = newFontName
    Selection.Font.ColorIndex = wdBrightGreen
    Selection.TypeText (insertionText)
End Sub
```

VBA tutorial notes by James Tam