VBA Programming & Data Visualization: Part 1

- Integrating VBA Word programs with other Office applications
- Implementing VBA in other Office applications (Excel, PowerPoint)

Applying Many Of The Previous Concepts In A Practical Example & Linking Documents

- As you are aware different programs serve different purposes:
  - Database: storing and retrieving information
  - Spreadsheet: performing calculations, displaying graphical views of results
  - Word processor: creating text documents with many features for formatting and laying out text
- VBA allows the output of one program to become the input of another program.
  - Although this can be done 'manually' (reading the documents and typing in changes) if the dataset is large this can be a tedious and error-prone process
    - Copy-pasting may alleviate some of these issues but it isn't always an option.
  - VBA can be used to automate the process

Accessing Other Office Applications With A Word VBA Program

Example Problem

- Financial statements (monetary data) about many companies can be stored in a spreadsheet where an analysis can be performed e.g. does the company have enough $$$ on hand to meet its financial commitments.
- This information can be read into a VBA program which can further evaluate the data.
- The results can be presented in Word using the numerous text formatting features to highlight pertinent financial information.
- **Names of the documents used in this example**:
  - **Data file:** `FNCE.xlsx` (contains the financial data: program input)
  - **Word document:** `1spreadSheetAnalyzer.docm` (contains the VBA program as well as the presentation of results: program output)

Spread Sheet Analyzer

```
Sub spreadsheetAnalyzer()
    Const MIN_INCOME As Long = 250
    Const MIN_RATIO As Long = 25
    Const PERCENT As Double = 100
    Const SPREADSHEET_NAME As
      String = "1FNCE.xlsx"
    Const PERCENT = 100
    Dim company1 As String
    Dim income1 As Long
    Dim ratio1 As Long
    Dim company2 As String
    Dim income2 As Long
    Dim ratio2 As Long
    Dim company3 As String
    Dim income3 As Long
    Dim ratio3 As Long
    Dim comment1 As String
    Dim comment2 As String
    Dim comment3 As String
```

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

TAMCO: 33%

HAL: Net income $250

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

Spread Sheet Analyzer (2)

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

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

Dim workbook As Variant
Dim location As String
location = ActiveDocument.Path()
Set workbook = excel.workbooks.Open(location & "\" &
  SPREADSHEET_NAME)
```

Spread Sheet Analyzer (2)

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

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

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

Spread Sheet Analyzer (3)

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

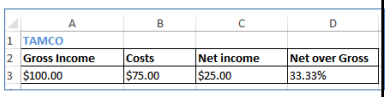
```
' Get company names
company1 = excel.Range("A1").Value
company2 = excel.Range("A5").Value
company3 = excel.Range("A9").Value

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

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

TAMCO: 33%

Spread Sheet Analyzer (4): First Company



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

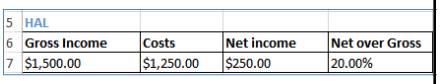
```
comment1 = company1 & ": "
If (income1 >= MIN_INCOME) Then
   comment1 = comment1 & "Net income $" & income1
   Selection.Font.Color = wdColorRed
   Selection.TypeText (comment1)
   If (ratio1 >= MIN_RATIO) Then
       comment1 = ", " & ratio1 & "% <== BUY THIS!"
       Selection.Font.Color = wdColorBlue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

250

25

HAL: Net income $250

Spread Sheet Analyzer (5): Second Company



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

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

250

25

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

Spread Sheet Analyzer (6): Third Company

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

```
comment3 = company3 & ": "
If (income3 >= MIN_INCOME) Then
   comment3 = comment3 & "Net income $" & income3
   Selection.Font.Color = wdColorRed
   Selection.TypeText (comment3)
   If (ratio3 >= MIN_RATIO) Then
       comment3 = ", " & ratio3 & "% <== BUY THIS!"
       Selection.Font.Color = wdColorBlue
       Selection.TypeText (comment3)
   End If
   Selection.TypeText (vbCr)
Else
   If (ratio3 >= MIN_RATIO) Then
       comment3 = comment3 & ratio3 & "%" & vbCr
       Selection.Font.Color = wdColorBlue
       Selection.TypeText (comment3)
   End If
End If
```

250

25

Writing A VBA Program For Excel And PowerPoint

Writing VBA Programs For Other Office Applications

- The following is only a brief introduction.
  - (More comprehensive practical examples could take up most or all of the curriculum of an entire course).
- Examples in this section will include VBA programs for:
  - MS-Excel
  - MS-PowerPoint
  - Unlike the previous example which was developed within Word and accessed an Excel document these examples are written specifically within and for Excel and PowerPoint
  - Programs other than MS-Office applications (e.g. AutoCAD) can have a VBA application written to augment them as long as they purchased a license to do so from Microsoft:
    - https://news.microsoft.com/1999/09/30/microsoft-announces-corporate-licensing-for-visual-basic-for-applications/
    - (Online tutorials can be found).

Accessing The VBA Programming Feature

- Similar to Word you can write programs for Excel or PowerPoint via the '`View`' tab in the Ribbon.
- Alternatively if you've customized the Ribbon to add the '`Developer`' tab then you access the basic and other programming capabilities here as well.

Accessing/Modifying Cell Data

- You can use either of the following objects: `Cells` or `Range`.
- JT's rules of thumb regarding **when to use each one**:
  - Cells:
    - Modifying or accessing a single cell
    - Accessing cells via an integer loop control
  - Range: allows access/modifications to a range of cells (although a single cell option is possible)

Accessing Cell Data

- Using the `Cells` attribute[1]:
  - **Format**:

    ```
    Cells(<row index>, <column index>)
    ```

  - **Example**:

    ```
    Cells(row, column) 'row, column are integer variables 'long'
    ```

  - Because 'Cells' provides access via integer values if you are accessing an Excel spreadsheet using a loop then this method is likely the easiest approach to use.
- Using the `Range` object[2]:
  - **Format**:

    ```
    Range(<Cell>)
    Range(<Start cell>:<End cell>)
    ```

  - **Examples**:

    ```
    Range("A3")
    Range("B10:J15")
    Range(sCell & ":" & eCell) 'sCell, eCell are string variables
    ```

1 For more information: https://docs.microsoft.com/en-us/office/vba/api/excel.worksheet.cells
2 For more information: https://docs.microsoft.com/en-us/office/vba/api/Excel.Range(object)

Modifying Cell Data

- Using the `Cells` method:
  - **Format**:

```
Cells(<row index>, <column index>) = <expression>
```

  - **Example**:

```
Cells(1, 2) = $1000000 '649 Lottery winner!
```

- Using the `Range` method:
  - **Format**:

```
Range(<Cell>) = <expression>
'Multi-range assignment: all cells are assigned the same value
'(what the expression evaluates to).
Range(<Start cell>:<End cell>) = <expression>
```

  - **Examples**:

```
'COURSE_NAME_NUMBER a predefined named constant
Range(COURSE_NAME_NUMBER) = "CPSC 203"
'startCell, endCell, newData: predefined string variables
Range(Range(startCell & ":" & endCell) = newData)
```

First Excel VBA Example: Accessing/Modifying Cells

- **Name of the spreadsheet that contains the VBA example**: `2Excel1_accessing_modifying_cell_data.xlsm` (Regular spreadsheet = `.xlsx`)
  - **Learning objective**: simple example getting/setting cell values (range of cells using the `Range` method).

```
Sub accessingModifyingCellsV1()
    MsgBox (Range("A1"))
    MsgBox (Cells(1, 1))
    Range("A1:B1") = "change1"
    MsgBox (Range("A1"))
    Cells(1, 2) = "change2"
    MsgBox (Cells(1, 2))
End Sub
```

Second Excel VBA Example: Accessing Cells Based On The Contents Of Variables

- **Name of the spreadsheet that contains the VBA example**: `3Excel2_accessing_modifying_cell_data_via_variables_named_constants`
  - **Learning objective**: getting/setting cell values based on the contents of variables, user input. Applying good style conventions by using named constants to access cells.
  - (See above spreadsheet for the full example)

```
Const ID_COLUMN As Long = 1
Const START_ROW As Long = 4
Const END_ROW As Long = 9
i = START_ROW
Do While (i <= END_ROW)
    Cells(i, ID_COLUMN) = i
    i = i + 1
Loop
```

Second Excel VBA Example: Accessing Cells Based On The Contents Of Variables (2)

```
row = InputBox("Row to modify (e.g. 1,2,3...): ")
column = InputBox("Column to modify (e.g. 1,2,3...): ")
newData = InputBox("Contents for Cell (row/column): (" & _
  row & "/" & row & ")")
Cells(row, column) = newData

startCell = InputBox("Start to modify (e.g. A1): ")
endCell = InputBox("End to modify (e.g. E3): ")
Range(startCell & ":" & endCell) = newData
```

Formatting Spreadsheet Cells In VBA

- It can be done by via the `Font` object (attribute of the `Cells`, `Range` object).
  - **Format** :

    `Cells(<row>, <column>).Font.<attribute> = <value>`

    `Range(<Cell range>).Font.<attribute> = <value>`
  - **Examples** :

```
Cells(10, 200).Font.Bold = True
Range("B1").Name = "Arial"
```

1 For more information about the Font attributes and methods: https://docs.microsoft.com/en-us/office/vba/api/excel.font(object)

Third Excel VBA Example: Changing Fonts & Formatting

- **Name of the spreadsheet that contains the VBA example**: `4Excel3_formatting_cells`
  - **Learning objective**: Changing the font and font properties of Excel spreadsheet text.
  - (See above spreadsheet for the full example)

```
Range("A1").Font.Bold = True
Range("A1:B5").Font.Color = vbRed
Cells(1, 3).Font.Bold = True
Cells(1, 3).Font.Name = "Wing dings"
```

Accessing A Specific Worksheet (Using An Index)

- Similar to writing VBA programs for Word, if a specific worksheet is not specified then the currently active worksheet will have the instructions of program applied to it.
  - Previous programs.
- Access to individual sheets: use the `Worksheets` collection.
  - **Format (via index)**:

```
Worksheets(<index>)
Worksheets(<index>)
```

Example (via index):

```
Worksheets(1)
Worksheets(2)
```

Accessing A Specific Worksheet (Using The Name)

- **Format (via worksheet name)**:

```
Worksheets(<"Worksheet name">)
Worksheets("Sheet1")
```

- **Example (via worksheet name)**:

```
Worksheets("CPSC 203 grades")
Worksheets("Sheet1")
```

Accessing Specific Spreadsheets (Workbooks)

- You can access individual workbooks (spreadsheets) via the `Workbooks` collection (similar to the `Documents` collection used in VBA for Word)
  - Example, number of open spreadsheets:
    `Application.Workbooks.Count`

Fourth Excel VBA Example: Accessing Specific Worksheets

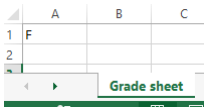
- **Name of the spreadsheet that contains the VBA example**: `5Excel4_specifying_a_worksheet`
  - **Learning objective**: accessing specific worksheets
  - (See above spreadsheet for the full example)

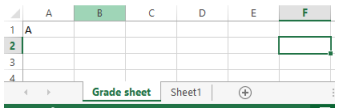
```
Worksheets("Grade sheet").Cells(1, 1) = "A"
Worksheets("Sheet1").Cells(2, 2).Font.Bold = True
Worksheets("Sheet1").Range("C5") = "adsdadfads"

'Grade sheet2 = Worksheet(2)
Worksheets(2).Cells(2, 6) = "worksheet 1 via Cells"
Worksheets(1).Range("B2") = "worksheet 2 via Range"
```

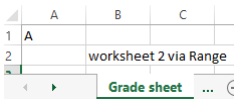
(To Allow You To Review Afterward)

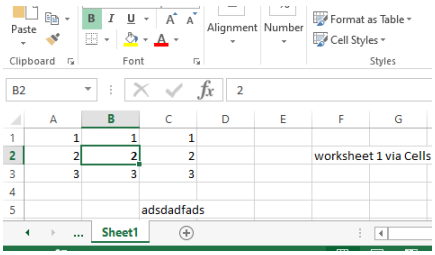
- Before





- After





VBA Example For PowerPoint

- **Name of the PowerPoint document that contains the VBA example**: `6PowerPoint1_inserting_slides.pptm` (Regular PowerPoint = `.pptx`)

- Functionality:
  - Prompts the user for the number of slides to insert
  - The new slides will be inserted immediately after the current content (a single slide that is a 'title' slide).
    - Even numbered slides will have a 'title' field (`ppLayoutCustom`)
    - Odd numbered slides will be completely blank (`ppLayoutBlank`)

VBA Example For PowerPoint (2)

```
Sub powerPointExample()
    Dim numSlides As Long
    Dim i As Long
    numSlides = InputBox("Number of slides to insert: ")
```

VBA Example For PowerPoint (3)

```
    i = 1
    Do While (i <= numSlides)
        If ((i Mod 2) = 0) Then  'Even
            ActivePresentation.Slides.Add _
              Index:=ActivePresentation.Slides.Count + 1, _
              Layout:=ppLayoutCustom
        Else 'Odd
            ActivePresentation.Slides.Add _
              Index:=ActivePresentation.Slides.Count + 1,  _
              Layout:=ppLayoutBlank
        End If
        i = i + 1
    Loop
```

After This Section You Should Now Know How To:

- Access other types of MS-Office programs with an VBA program written for Word.
- Access & modify cell contents via the Range and Cells objects.
- Change fonts and font effects in a spreadsheet.
- Access specific worksheets through the name and through the index
- Create a simple MS-PowerPoint VBA Program.

Images

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