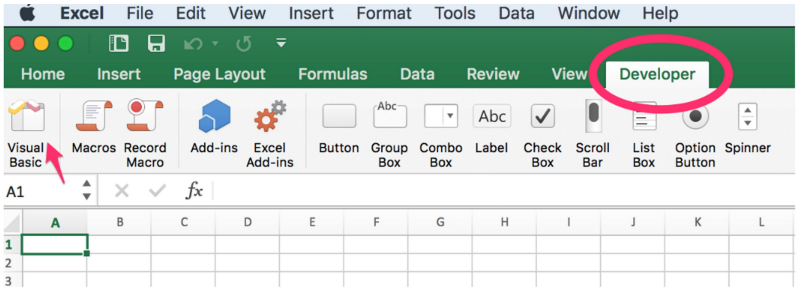
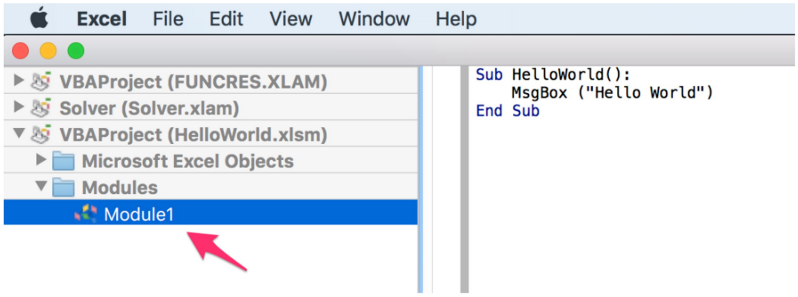
Hello World

Friday, September 13, 2019 6:57 PM

- Go to file hello_world.xlsm
- Open your VBA Editor and navigate to Module 1

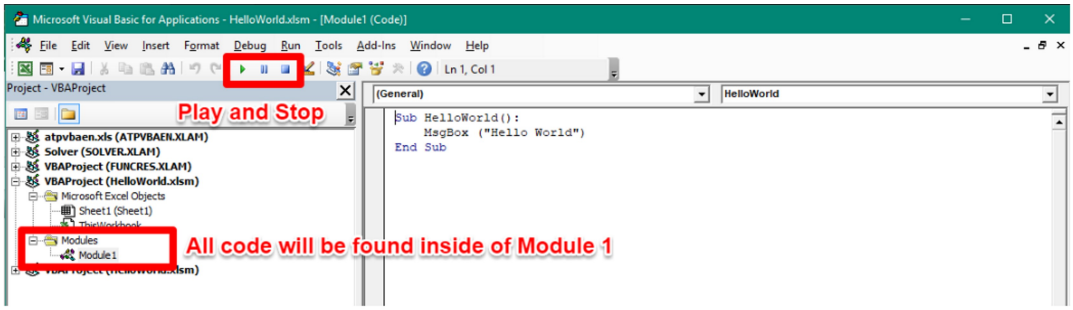


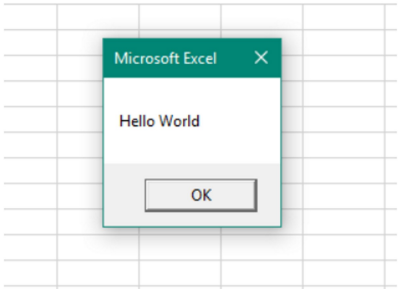


- hello_world (3)

- Modules
  - Modules are organizational units of VBA code that are usually attached to a workbook or worksheet.
  - They can be created by right clicking on a workbook or worksheet and then selecting "Insert Module"
  - Once inside a module, we can begin to write out VBA script.

- Example
  - In our case, we've pre-created a script that will trigger Excel to deliver a pop-up message.
  - Click Play inside your VBA editor to trigger this message. (Make sure your cursor is inside the HelloWorld subroutine)





Hello World Breakdown

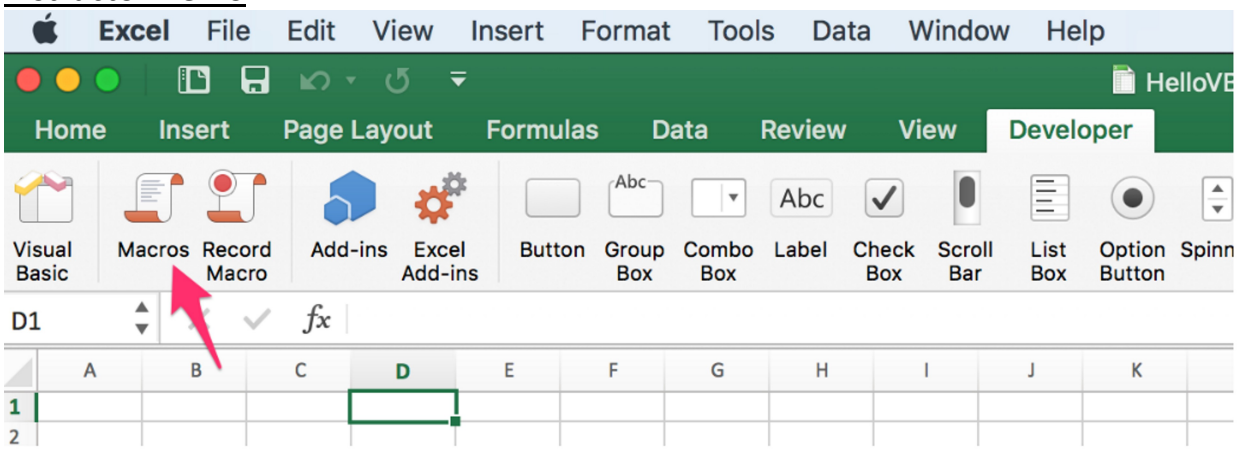
```
Sub HelloWorld():
  MsgBox("Hello World")
End Sub
```

- The code begins with the keyword **Sub**, which is short for subroutine.
- This line is followed by HelloWorld(), which marks the title of our subroutine.
- The empty brackets indicate that our subroutine takes in no arguments.
    - we'll be talking about functions, which do take in arguments at a later point
- This subroutine has a single aim -- to create a pop-up message box (MsgBox) with the phrase "Hello World".
- Once our subroutine has triggered its pop-up message, its work is done and the subroutine completes.
- This completion is denoted by the **End Sub** keywords.
- Every subroutine must begin with the keyword **Sub** and end with the words **End Sub**

Exercise: Hello

Sunday, September 15, 2019 12:40 PM

- Instructor Demo




hello_vba (1)

- Instructions

Hello VBA

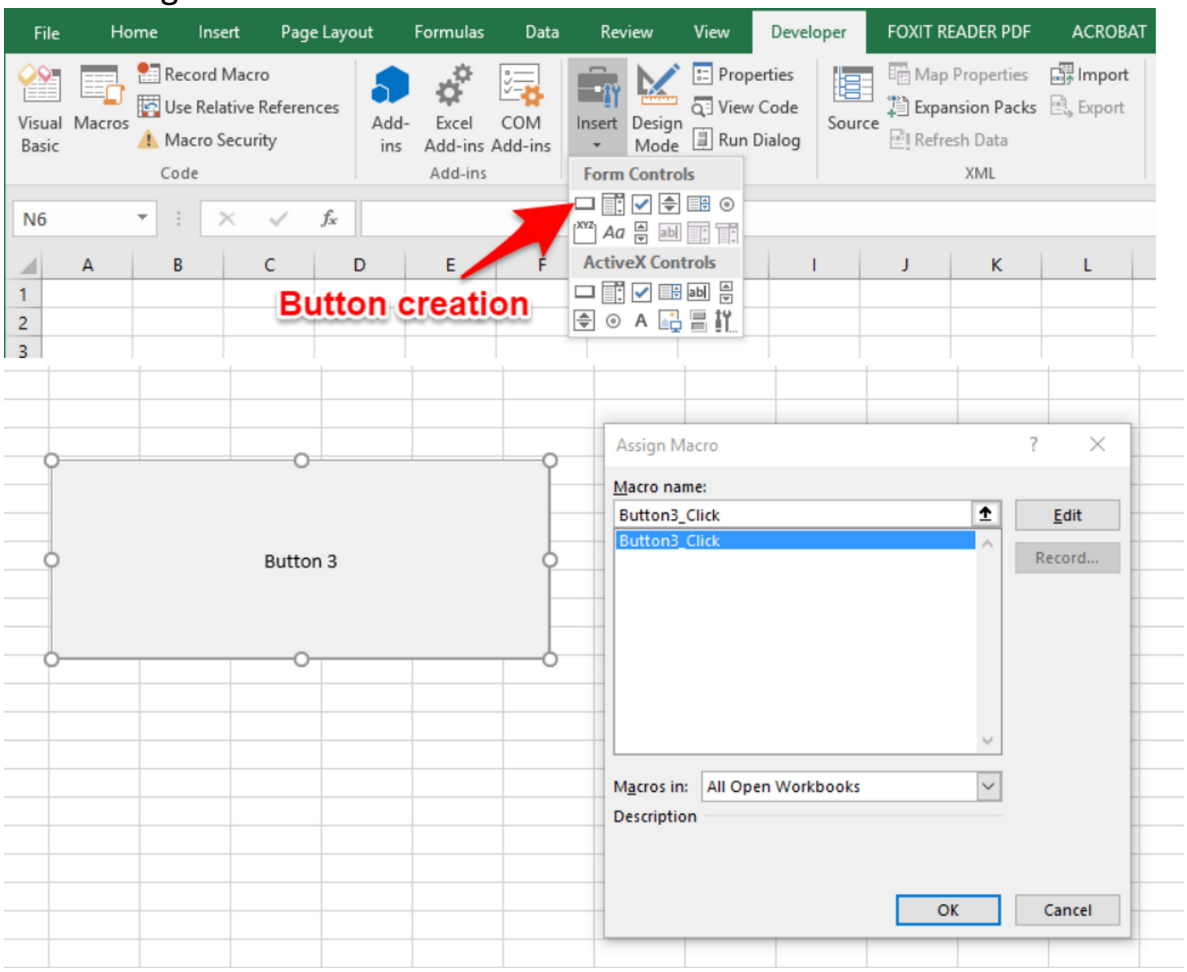
Instructions

- Create and execute a VBA script that generates three pop-up messages with text contained therein.
- If you finish early, ensure the people around you complete the task as well.

Button Clicks

Sunday, September 15, 2019 1:41 PM

- Adding Buttons



- Once the button is created, you will be asked to "Assign a Macro" to the button.
- You can choose to create a new macro or select a pre-existing one.
- *Note: If you accidentally close this window, you can always return to it by right-clicking your button and selecting "Assign Macro".*

Exercise: Choose Your Button

Sunday, September 15, 2019 2:15 PM

- Instructor Demo

button_clic...
(3)

- Instructions

Choose Your Button

Instructions

- Create an Excel file with two interactive buttons. These buttons should each be associated with a different VBA subroutine. When clicked, each button should trigger a different pop-up message.
- If you finish early, ensure the people around you complete the task as well.

Break (15 Minutes)

Sunday, September 15, 2019 4:23 PM

Cells and Ranges

Sunday, September 15, 2019 3:15 PM

cells_and_...

```
Sub CellsAndRanges():

    ' Inserting Data Via Cells
    Cells(2, 1).Value = "Cat"
    Cells(2, 2).Value = "In"
    Cells(2, 3).Value = "The"
    Cells(2, 4).Value = "Hat"

    ' Inserting Data Via Ranges
    Range("F1").Value = "I"
    Range("F2").Value = "Am"
    Range("F3").Value = "Sam"

    ' Inserting Data Across Ranges
    Range("F5:F7") = 5

End Sub
```

Single cells changed at a time.

Multiple cells changed at once. (Only for Ranges)

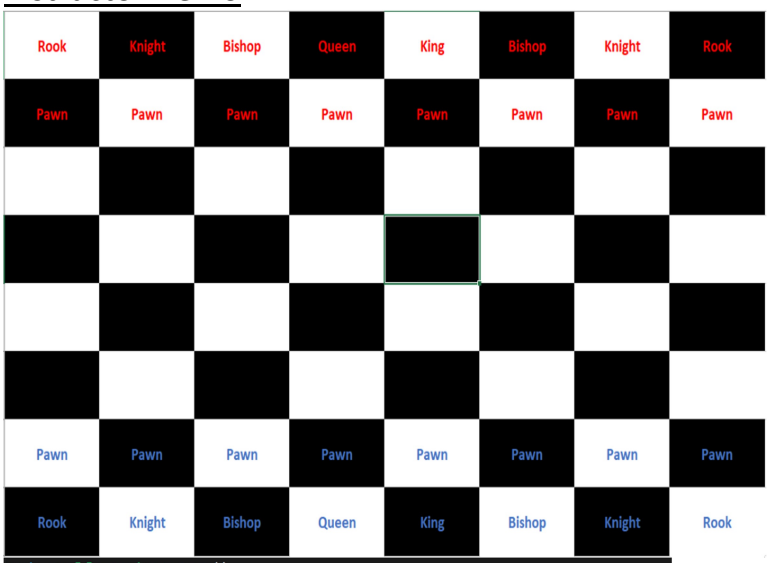
  - `Cells(X, X)` isn't just capturing the contents of the cell, but rather the entire "Cell Object" -- and with it, the formatting, style, and other aspects of the cell beyond the contents itself.

- **Cells** only allow a developer to capture a single cell at a time, while **Ranges** allow us to capture multiple cells at once.
- **Ranges** are used more often, but it can be especially useful to refer to **Cells** in "loop-based" programs because we can iterate the coordinates and manipulate the cells that are referenced.
  - *We will get time to work on this concept as the week progresses.

Exercise: Chess Board

Sunday, September 15, 2019 3:40 PM

Instructor Demo




```
Sub CellsAndRanges():

    ' Inserting Data Via Cells
    Cells(2, 1).Value = "Cat"
    Cells(2, 2).Value = "In"
    Cells(2, 3).Value = "The"
    Cells(2, 4).Value = "Hat"

    ' Inserting Data Via Ranges
    Range("F1").Value = "I"
    Range("F2").Value = "Am"
    Range("F3").Value = "Sam"

    ' Inserting Data Across Ranges
    Range("F5:F7") = 5

End Sub
```

Single cells changed at a time.

Multiple cells changed at once. (Only for Ranges)

ChessBoard (2)

Instructions:

- Populate the Chess Board provided with text-based chess pieces. For the top-half of the chess board use Ranges, for the bottom-half of the chess board use Cells.

Hint:

- Remember that with `Ranges`, it is possible to modify multiple cells at once.

Variables

Sunday, September 15, 2019 4:24 PM

- Variables are named items in programming.
- They can store strings (text), numerics (integers and doubles for decimals), booleans (true/false), and more.

```
' Basic String Variable
' ----------------------------------------
Dim name as String
name = "Gandalf"

msgbox(name)

' Basic String Concatenation (Combination)
' ----------------------------------------
Dim title as String
title = "The Great"

Dim fullname as String
fullname = name + " " + title

msgbox(fullname)
```

- VBA uses the single quote (') to denote a comment: everything following it is just for humans

```
' Basic String Variable
' ----------------------------------------
Dim name as String
name = "Gandalf"

msgbox(name)

' Basic String Concatenation (Combination)
' ----------------------------------------
Dim title as String
title = "The Great"

Dim fullname as String
fullname = name + " " + title

msgbox(fullname)
```

- We create (or declare) variables by using the **`Dim`** keyword followed by the name ("title" for this example) of the variable and the type **`As String`**.

```
' Basic String Variable
'
Dim name as String
name = "Gandalf"

msgbox(name)

' Basic String Concatenation (Combination)
' ----------------------------------------
Dim title as String
title = "The Great"

Dim fullname as String
fullname = name + " " + title

msgbox(fullname)
```

- We can then utilize these variables using their names

```
' Basic String Variable
' ----------------------------------------
Dim name as String
name = "Gandalf"

msgbox(name)

' Basic String Concatenation (Combination)
' ----------------------------------------
Dim title as String
title = "The Great"

Dim fullname as String
fullname = name + " " + title

msgbox(fullname)
```

- We can "concatenate" strings by combining them together (can also perform mathematical functions by combining numeric variables with operators)

```
' Basic String Variable
' ----------------------------------------
Dim name as String
name = "Gandalf"

msgbox(name)
```

```
' Basic String Concatenation (Combination)
' ----------------------------------------
Dim title as String
title = "The Great"

Dim fullname as String
fullname = name + " " + title

msgbox(fullname)
```

Numeric Examples Demo

Variables

Exercise: Total Calculator

Sunday, September 15, 2019 4:25 PM

Files:

- TotalCalculator_Unsolved.vbs
- TotalCalculator_Unsolved.xlsm

Instructions:

- Using the Spreadsheet and Unsolved VBS code as a starter, complete the script such that `Price`, `Tax`, `Quantity`, and `Total` are stored in variables.
- These variables should be then assigned the value of the cell they are associated with in the spreadsheet.
- When finished, your code should set the `Total` value in the spreadsheet and print a message box with the total in the form of: "Your total is $45.00"

Bonus:

- Try to complete the exercise, *without* looking at the starter code.

Arrays

Sunday, September 15, 2019 5:35 PM

- Arrays use zero-based numbering (0-indexed), meaning that the first element is 0.
- Zero-based numbering is a common paradigm across programming languages (Python, JavaScript, etc.)

```
' Basic Array Example
' ----------------------------------------
' Create the Ingredients Array
Dim Ingredients(5) as String

' Add Ingredients to the Array
Ingredients(0) = "Chocolate Bar"
Ingredients(1) = "Peanut Butter"
Ingredients(2) = "Jelly"
Ingredients(3) = "Macaroni"
Ingredients(4) = "Potato Salad"
Ingredients(5) = "Dragonfruit"

' Retrieve specific elements of the array
MsgBox(Ingredients(4))
MsgBox(Ingredients(0))
```

- We created an array called **`Ingredients`** to hold six strings but because of zero indexing five is passed in. For example (0,1,2,3,4,5) is six spots in the array

```
' Basic Array Example
' ----------------------------------------
' Create the Ingredients Array
Dim Ingredients(5) as String

' Add Ingredients to the Array
Ingredients(0) = "Chocolate Bar"
Ingredients(1) = "Peanut Butter"
Ingredients(2) = "Jelly"
Ingredients(3) = "Macaroni"
Ingredients(4) = "Potato Salad"
Ingredients(5) = "Dragonfruit"

' Retrieve specific elements of the array
MsgBox(Ingredients(4))
MsgBox(Ingredients(0))
```

- We then added elements to this array using the `Ingredients(X)` syntax.

```
' Basic Array Example
' ----------------------------------------
' Create the Ingredients Array
Dim Ingredients(5) as String

' Add Ingredients to the Array
Ingredients(0) = "Chocolate Bar"
Ingredients(1) = "Peanut Butter"
Ingredients(2) = "Jelly"
Ingredients(3) = "Macaroni"
Ingredients(4) = "Potato Salad"
Ingredients(5) = "Dragonfruit"

' Retrieve specific elements of the array
MsgBox(Ingredients(4))
MsgBox(Ingredients(0))
```

- We then retrieved these values by referencing our array with the index number

```
' Basic Array Example
' ----------------------------------------
' Create the Ingredients Array
Dim Ingredients(5) as String

' Add Ingredients to the Array
Ingredients(0) = "Chocolate Bar"
Ingredients(1) = "Peanut Butter"
Ingredients(2) = "Jelly"
Ingredients(3) = "Macaroni"
Ingredients(4) = "Potato Salad"
Ingredients(5) = "Dragonfruit"

' Retrieve specific elements of the array
MsgBox(Ingredients(4))
MsgBox(Ingredients(0))
```

Splitting Strings

Sunday, September 15, 2019 5:53 PM

- **Split** method, which breaks apart strings based on a provided delimiter. These broken down strings then become elements of a larger array.

```
' String Splitting Example
' ----------------------------------------
dim Words() as String
dim Shakespeare as String
Shakespeare = "To be or not to be. That is the question"

' Break apart the Shakespeare quote into individual words
Words = Split(Shakespeare, " ")

' Print individual word
MsgBox(Words(5))
```

- A **Words** array is created with an undefined number of string elements.

```
' String Splitting Example
' ----------------------------------------
dim Words() as String
dim Shakespeare as String
Shakespeare = "To be or not to be. That is the question"

' Break apart the Shakespeare quote into individual words
Words = Split(Shakespeare, " ")

' Print individual word
MsgBox(Words(5))
```

- A variable **Shakespeare** is used to hold a line of text

```
' String Splitting Example
' -------------------------------------
dim Words() as String
dim Shakespeare as String
Shakespeare = "To be or not to be. That is the question"

' Break apart the Shakespeare quote into individual words
Words = Split(Shakespeare, " ")

' Print individual word
MsgBox(Words(5))
```

- We then use the **`Split`** method to break apart the **`Shakespeare`** variable on spaces (" ")
- Creating an array that looks like: `["To", "be", "or", "not", "to", "be", ...]`

```
' String Splitting Example
' -------------------------------------
dim Words() as String
dim Shakespeare as String
Shakespeare = "To be or not to be. That is the question"

' Break apart the Shakespeare quote into individual words
Words = Split(Shakespeare, " ")

' Print individual word
MsgBox(Words(5))
```

- We can then select individual words from this resulting arrays by referencing the word's index in the array

```
' String Splitting Example
' ----------------------------------------
dim Words() as String
dim Shakespeare as String
Shakespeare = "To be or not to be. That is the question"

' Break apart the Shakespeare quote into individual words
Words = Split(Shakespeare, " ")

' Print individual word
MsgBox(Words(5))
```

Exercise: Sentence Breaker

Sunday, September 15, 2019 6:17 PM

Instructor Demo

sentence_...

Files

- SentenceBreaker.vbs
- SentenceBreaker.xlsm

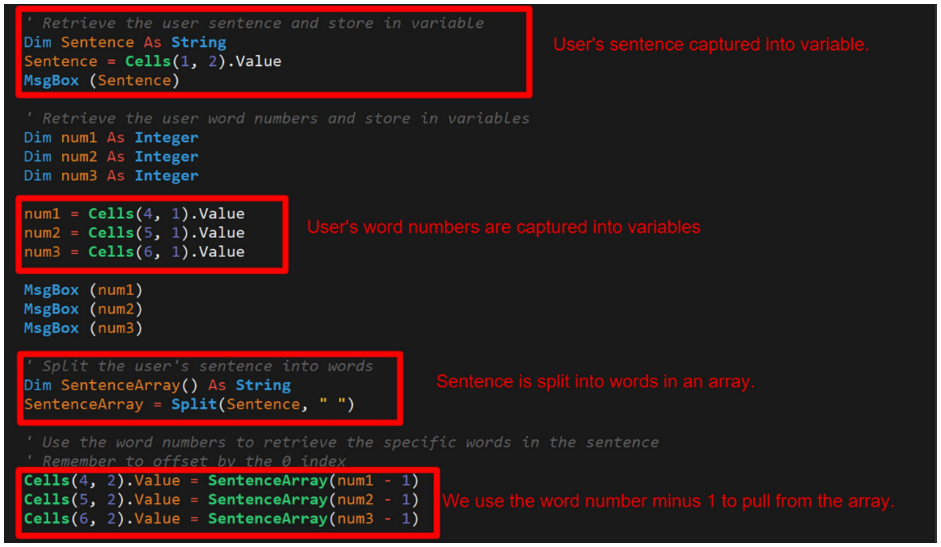
Instructions:

- Using the files provided as a starting point, create a VBA script such that it reads in a User Sentence then prints the correct words based on word numbers provided.

Notes:

- This is a more challenging assignment. So take your time on it. Try to bite it off bit by bit.

****
****
****
****


' Retrieve the user sentence and store in variable
Dim Sentence As String
Sentence = Cells(1, 2).Value
MsgBox (Sentence)
User's sentence captured into variable.
' Retrieve the user word numbers and store in variables
Dim num1 As Integer
Dim num2 As Integer
Dim num3 As Integer
num1 = Cells(4, 1).Value
num2 = Cells(5, 1).Value
num3 = Cells(6, 1).Value
User's word numbers are captured into variables
MsgBox (num1)
MsgBox (num2)
MsgBox (num3)
' Split the user's sentence into words
Dim SentenceArray() As String
SentenceArray = Split(Sentence, " ")
Sentence is split into words in an array.
' Use the word numbers to retrieve the specific words in the sentence
' Remember to offset by the 0 index
Cells(4, 2).Value = SentenceArray(num1 - 1)
Cells(5, 2).Value = SentenceArray(num2 - 1)
Cells(6, 2).Value = SentenceArray(num3 - 1)
We use the word number minus 1 to pull from the array.

Conditionals

Sunday, September 15, 2019 6:59 PM

VBA Conditionals

- VBA conditionals introduce a real benefit over traditional Excel formulas.
    - cleaner syntax
    - more nuanced conditionals.

```
' Simple Conditional Example
' ----------------------------------------
If Range("A2").Value > Range("B2").Value Then
    MsgBox ("Num 1 is greater than Num 2")
End If

' Simple Conditional with If, Else, and Elseif
' ----------------------------------------
If Range("A5").Value > Range("B5").Value Then
    MsgBox ("Num 3 is greater than Num 4")

ElseIf Range("A5").Value < Range("B5").Value Then
    MsgBox("Num 4 is greater than Num 3")

Else
    MsgBox("Num 3 and Num 4 are equal")

End If
```

- In VBA, the syntax for conditionals involves: **`If Then`** and **`End If`**. Additional keywords include **`ElseIf`** and **`Else`**.
- In VBA, we can combine conditions using the keywords **`And`** and **`Or`**.

Exercise: Choose your story

Sunday, September 15, 2019 7:14 PM

Instructor Demo

choose_yo...

Choose Your Story

Instructions

- Create a simple Excel workbook and VBA macro in which a user is provided a single button to click. Based on the number they provide in a text box above, a different message box will appear.
    - If the user enters a value of 1, display: "You choose to enter the wooded forest of doom!"
    - If the user enters a value of 2, display: "You choose to enter the fiery volcano of doom!"
    - If the user enters a value of 3, display: "You choose to enter the terrifying jungle of doom!"
    - If the user enters a value of 4, display a similar custom message.
    - If the user enters anything else, display: "Try following directions"

****
****
****

```
' Use conditionals to change message box based on user input
If (Range("B1").Value = 1) Then
    MsgBox("You choose to enter the wooded forest of doom!")

Elseif (Range("B1").Value = 2) Then
    MsgBox("You choose to enter the fiery volcano of doom!")

Elseif (Range("B1").Value = 3) Then
    MsgBox("You choose to enter the terrifying jungle of doom!")

Elseif (Range("B1").Value = 4) Then
    MsgBox("You choose to enter the bathroom")

Else
    MsgBox("Try following directions")

End If
```

Homework (Due 9/28)

Sunday, September 15, 2019 7:34 PM

In essence, the homework assignment tasks them with creating a VBA script to loop through stock market records to identify various stocks based on provided conditions. You will be able to complete the homework by the end of Saturday's class.