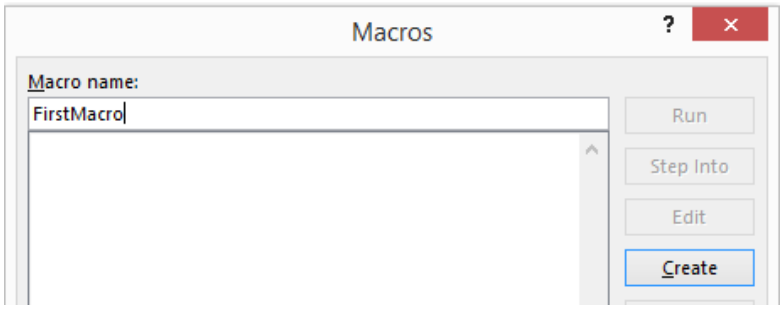
To do week of Nov 13 - 19

Go over A4

The generated method

```
Sub FirstMacro()
'
' FirstMacro Macro
'
'

End Sub
```



- A macro starts with “Sub”
- And it ends with “End Sub”
- The name of the macro is placed next to “Sub”
- (Note for TA: Anything that comes after a single quotation is considered as a comment but don’t talk about it for now).

Our first macro will be displaying a message in a popping dialog

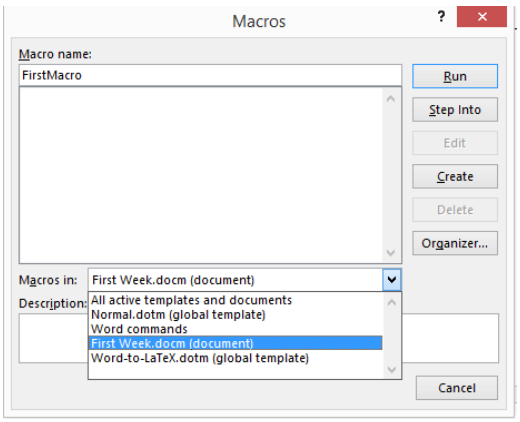
```
Option explicit
Sub HelloWorldMacro()
    MsgBox(“Hello World”)
End Sub
```

Option explicit

Saving the macros and documents

- Make sure to inform the students to save their documents as “docm”.
- The default extension is “docx”
- The default extension will work fine if they are working on their computer but if they copied the file or uploaded it to D2L, the macros will not be uploaded

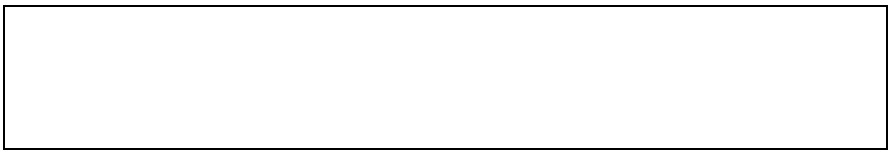
- Make sure when they create a macro to save it in the document itself



- The default is to save on their local computers only and this means that it will not be uploaded with the document on D2L

Students do

Have students type in a VBA program that will display their name with a message box e.g.,



(TA notes: give them time to do this! Although it's not hard to type and run if you know what you are doing the students are new to this and will likely make many mistakes e.g., typing the program into Word and not the VB editor, they will make many syntax errors, they won't remember details like how/where to save the macro, how to run it etc.)

Option explicit

Tell them to include this in all their programs before the subroutine

```
‘ PROGRAM: USED AND PRODUCES SYTAX ERROR
Option Explicit

Sub explicitUsed()
    Dim agePerson1 As Long
    Dim agePerson2 As Long
    agePerson1 = -1
    agePerson2 = -1
    agePerson = InputBox("Person 1 age")  'Typo: creates a new variable, agePerson1
not set
    agePerson2 = InputBox("Person 2 age")
    MsgBox ("Person 1 age " & agePerson1) 'agePerson1: still at default value
    MsgBox ("Person 2 age " & agePerson2)
End Sub


‘ PROGRAM: NOT USED AND PRODUCES A LOGIC ERROR

Sub explicitNotUsed()
    Dim agePerson1 As Long
    Dim agePerson2 As Long
    agePerson1 = -1
    agePerson2 = -1
    agePerson = InputBox("Person 1 age")  'Typo: creates a new variable, agePerson1
not set
    agePerson2 = InputBox("Person 2 age")
    MsgBox ("Person 1 age " & agePerson1) 'agePerson1: still at default value
    MsgBox ("Person 2 age " & agePerson2)


End Sub
```

Now let's make it a bit more personal

```
Sub PersonalMacro()
    Dim MyName As String
    MyName = "your name"

    Dim BirthDate As Date
    BirthDate = #5/23/1991#

    Dim Age As Integer
    Age = 23

    MsgBox ("My name is " & MyName & " and I was born in " & BirthDate & " so my age is " & Age)
    ActiveDocument.ActiveWindow.Caption = "My name is " & MyName &
      " and I was born in " & BirthDate & " so my age is " & Age
End Sub
```

[Half typed; just the name; and half displayed]

- Defining a variable requires defining its type
    - First, set the type by starting with "Dim"
    - Second, define the name of the variable
    - Third, define the type such as "As String"
- Note that the value of a string is store between two double quotations
- Note that the value of a date is stored between two hashes
- Note that the value of an integer or decimal is stored without any surrounding quotations or hashes
- Note the difference between displaying the output in a message box and in the title of the current active window

The goal of this macro is to learn how to apply arithmetic operations.

```
Sub ArithmetricsMacro()

    Dim FirstNumber As Integer
    Dim SecondNumber As Integer
    FirstNumber = 4
    SecondNumber = 5

    Dim Summation As Integer
    Summation = FirstNumber + SecondNumber

    MsgBox ("Summation: " & Summation)

End Sub
```

[To be typed] they can quickly show them the subtraction and multiplication [-, *]

- The first numeric type we get to investigate is "Integer"
- Notice how simple it is to apply a simple mathematic operation

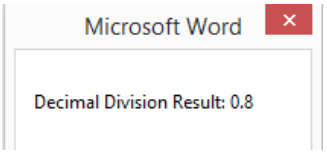
Students to do

- Ask the students to change this code to include division
- Most probably they will use integer because they have never seen double before
- Their results will be rounded; not exact value
- Use the code below to show them the difference between the results

Now let's take it a step further and investigate the second numeric type "Double"

[To be typed]

The output will look like:

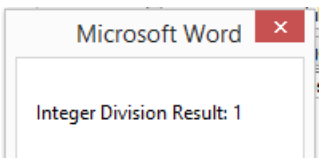


- Note that the decimal division result is the actual value: 0.8
- But the integer division result is the rounded value of the actual value (0.8)
    - Anything above 0.5 is round up else it is rounded down

```
Sub InputMacro()

    Dim Age As Integer
    Dim DogeAge As Integer

    Age = InputBox("What is your age?", "Getting Age")
    DogAge = Age * 7

    MsgBox ("You age in dog years is " & DogAge)

End Sub
```

[To be typed]

- InputBox is a method that can be used to retrieve an input from the user



    - The first argument is the prompt message
    - The second argument is the title of the dialog
- It is a bad practice to put a constant number directly in the code as shown in this code (* 7)

The optimal way to do it is as follow:

```
Sub InputMacro()

    Dim Age As Integer
    Dim DogeAge As Integer

    Const DOG_HUMAN_RATIO = 7

    Age = InputBox("What is your age?", "Getting Age")
    DogAge = Age * DOG_HUMAN_RATIO
    MsgBox ("You age in dog years is " & DogAge)

End Sub
```

[To be typed]

- Note that the DogHumanRatio starts with Const
    - Const indicates that the variable is constant, the value cannot be changed programmatically, it is hard coded

Students do

- Have students modify their previous example but make it dynamic. The program asks for the user's name, this value is then displayed in a message box e.g.,

```
Sub HelloWorldMacro()
    Dim Name As String
    Name = InputBox("What is your name?")
    MsgBox("Hello " & Name)
End Sub
```

The goal of this macro is to assure consistency in our word document. We would like to make sure that all the text in our document have the same font and same spacing before and after.

```
Sub DocumentStyleMacro()
    ActiveDocument.Select
    Selection.Font.Name = "Calibiri"
    Selection.Paragraphs.SpaceAfter = 5
    Selection.Paragraphs.SpaceBefore = 5
End Sub
```

[To be typed]

Once we are done, it is time to investigate other text formatting options. Let's create a macro that convert a convert a usual text to a title.

```
Sub SelectionStyleMacro()
    Selection.Font.Size = 14
    Selection.Font.Bold = True
    Selection.Font.Underline = True
    Selection.ParagraphFormat.Alignment = wdAlignParagraphCenter
End Sub
```

[To be typed]

What else can be done with the selected text?

cat The in the hat

```
Sub FixMacro()
    Selection.Expand
    Selection.Cut
    Selection.MoveLeft
    Selection.PasteAndFormat
(wdFormatPlainText)
End Sub
```

[To be displayed]

Place the cursor next to “The” then run this macro

- Our goal in this macro is to fix the sentence
- The first line of code (Expan) will select all the word next to the cursor
- The second line of code (Cut) will cut the word
- The third line of code (MoveLeft) will move the cursor one word backward
- The last line of code (PastAndFormat) will paste the word
- Note that “wdFormatPlainText” will allow the text to be pasted with the current cursor’s font style. This is a predefined constant by VBA

kostasfx@yahoo.gr
ibrahim.karakira@ucalgary.ca
omaddam@Gmail.com

We have the current list of emails and we decided to change all the yahoo accounts to yahoo

```
Sub SimpleFindAndReplaceMacro()

    ActiveDocument.Content.Find.Execute FindText:="gmail", _
        ReplaceWith:="yahoo", Replace:=wdReplaceAll, _
        MatchCase:=True

End Sub
```

- Notice that this single command is written on multiple lines
    - To do this, break it down to multiple lines
    - At the end of each line; except the last one; add “_”
- MatchCase := True, we can turn our macro to be case sensitive by adding this option

The code can be structured better; for easier readability; using With End as follow:

```
Sub ExtendedFindAndReplaceMacro()

    With ActiveDocument.Content.Find
        .Text = "gmail"
        .Replacement.Text = "yahoo"
        .Execute MatchCase:=True, Replace:=wdReplaceAll
    End With

End Sub
```

The “Find” method is not limited to text, it can also be used for any feature:

```
Sub AdvanceFindAndReplaceMacro()

    With ActiveDocument.Content.Find
        . Style = " Heading 1"
        With .Replacement
            .Style = "Normal"
            .Font.Bold = True
            .Font.Size = 20
        End With
        .Execute Replace:=wdReplaceAll
    End With

End Sub
```

Finally, it is time to save the document and close it

```
Sub SaveAndCloseMacro()

    ActiveDocument.Save

ActiveDocument.Close

End Sub
```

Or close and save it in one statement

```
Sub SaveWhileClosingMacro()

    ActiveDocument.Close (wdSaveChanges)

End Sub
```

Documentation

- Anything that comes after {‘}; till the end of the line; is considered a comment

- They have seen it many times; every time they create a macro
- Documentation requirements
    - Document every macro they create
    - Include the sources that they used; if any
    - Contact information: Name, student ID and tutorial section
    - A list of features (#1 - 8 above) as well as a clear indication if the feature was or was not fully completed and (if applicable)
    - Use of a versioning system that ties into a list of features completed e.g., "Version (Nov 12) has Feature 1 completed", "Version (Nov 13) Has Features 1, 2, 8 completed"