VBA Programming: Part I

This is a Basic introduction into creating VBA programs for Word

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

VBA (*V*isual *B*asic For *A*pplications)

- VBA is a "programming language"
  - A programming language allows a person to write a computer program
  - VBA is based on but not identical to Visual Basic (VB)
  - Visual Basic is seldom used
  - VBA programs must be associated with a 'host' *application* (usually it's Microsoft Office document such as MS-Word but other applications can also be augmented by VBA programs).
  - Because host applications such as a Office are popular, VBA is still used to create actual programs.
    - The host application is enhanced or supplemented by the VBA program
    - "Why do I have to keep going through these same steps over and over again for each document?"
      - VBA can be used to automate tedious tasks
    - "Why doesn't this stupid word processor have this feature??!!"
      - Now you can add that feature yourself using VBA

VBA And Macros

- A macro programming language: allows a series of commands to be recorded into a computer program.
  - Running the program will execute that sequence of commands.
  - Because VBA does have a macroing capability VBA is often referred to as a 'macroing' language.

!!!Important Warnings!!!

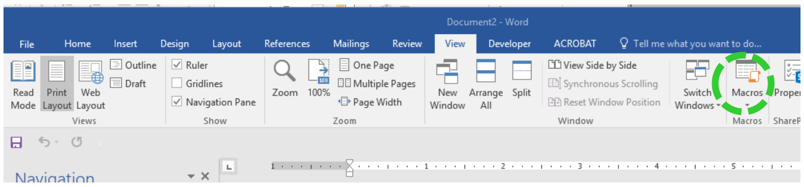
- Because the VBA programs modify Word documents, **close important Word documents** when you run your programs.
- Also, if a VBA program you are running will modify Word documents at a specified location (e.g. `C:\work\`) make sure that you **NEVER put any of your important documents at that location**.
- And it's always a good idea to **regularly backup your computer files** with the backup stored in the cloud or a storage device that isn't continuously connected.

Accessing The Program Writing Capability In Word

- **Method 1**: In the Word ribbon look under the 'View' ribbon
  - This ribbon is visible by default
  - But looking the View-ribbon doesn't show all the options for macros
    - Can't 'record' macros: this doesn't affect your working 203 assignment but it's something to keep in mind if you ever need the auto record capability in the future.
- **Method 2**: Alternatively you can enable the 'Developer' ribbon
  - This ribbon has more options than the View-ribbon

Method 1: View-Ribbon

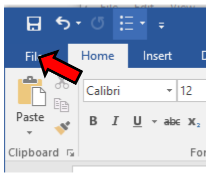
- `View->Macros`



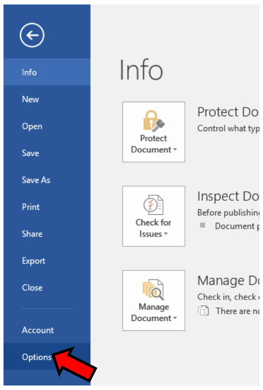
Method 2: Developer-Ribbon

- Steps for making a new ribbon tab (`Developer`) visible

1. Select the '`File`' tab in the ribbon

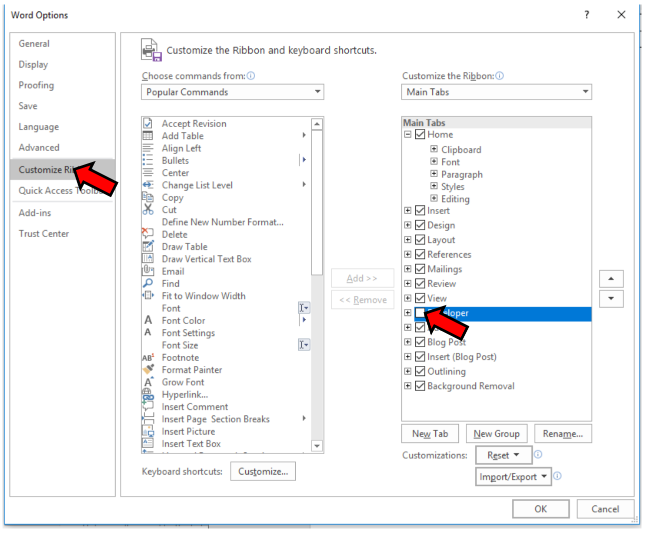


2. Select '`Options`'



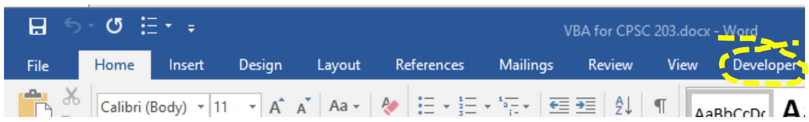
Viewing The 'Developer' Ribbon (MS-Word)

3A) Select "`Customize the ribbon`" 3B) Check the 'Developers' box



Viewing The ‘Developer’ Ribbon (MS-Word): 2

- This should add a new tab in the ribbon “Developer”

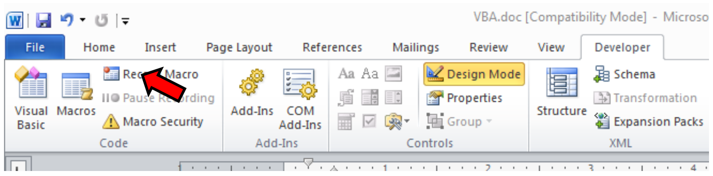


How To Create VBA Programs

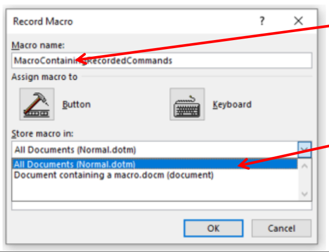
1. **Method 1**: Record the macro automatically: keystrokes and mouse selections will be stored as part of the program.
2. **Method 2**: Manually enter the program (type it in yourself into the VBA editor).

Method 1: Recording A Macro

- Developer ribbon
  - Click on "`Record Macro`"



  - Recording details



Macro name:
MacroContainingRecordedCommands

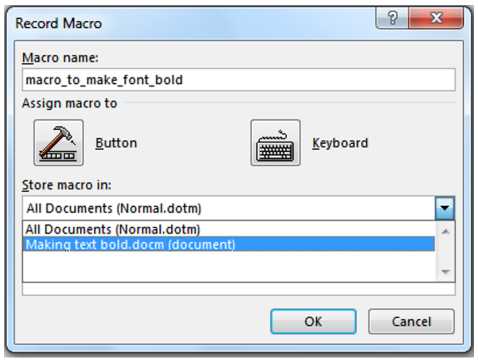
Store macro in:
All Documents (Normal.dotm)
All Documents (Normal.dotm)
Document containing a macro.docm (document)

What to name the macro (next slide for more info)

Document to store the macro (select the current Word document – in the example it's the 2nd option)

Recording A Macro (2)

- Give the macro a self explanatory name and press 'OK' (recording begins).



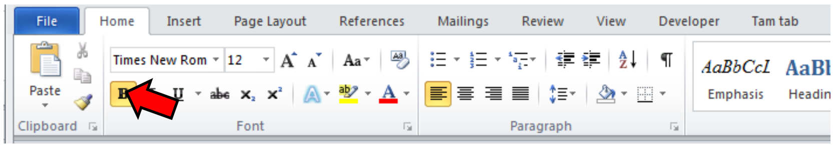
Macro name:
macro_to_make_font_bold

Store macro in:
All Documents (Normal.dotm)
All Documents (Normal.dotm)
Making text bold.docm (document)

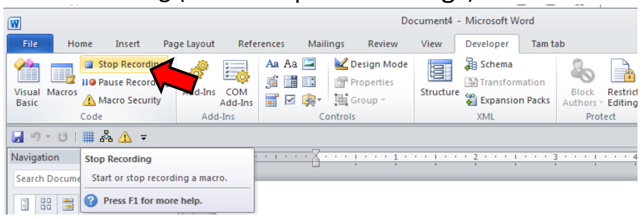
- **Important reminder: record the macro in the current document and not "All documents" (Important!)**

Recording A Macro (3)

- While recording you can run whatever Word features you want to add to the recording of the macro
  – In this case you would select bold font

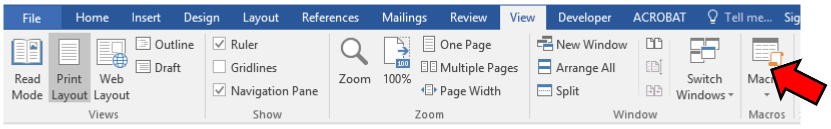


  – For this simple example all commands have been entered so you can stop the recording (click "`Stop recording`")

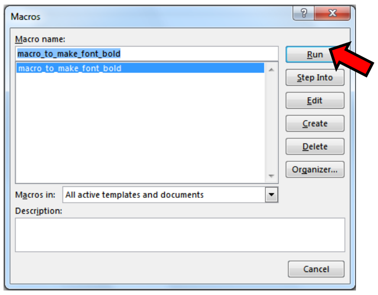


Running A Recorded Macro

- Under the 'View' ribbon select 'Macros'

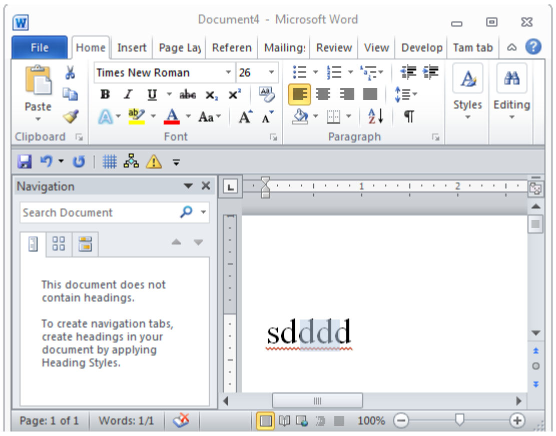


- Select the macro (Under "`Macro name`") and then click 'Run'



Running A Recorded Macro (2)

- In this case nothing happened?
  - This macro changes selected/highlighted text to bold
  - You need to select some text before running the macro



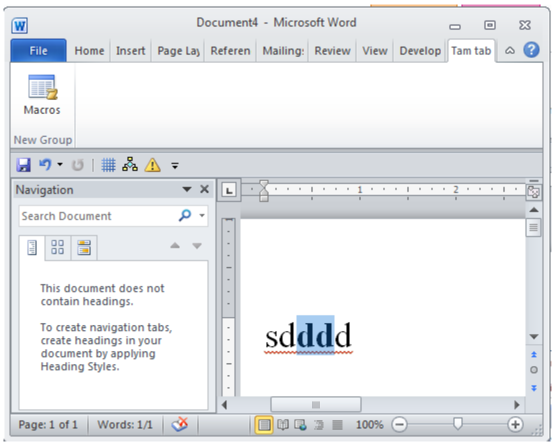
This document does not contain headings.

To create navigation tabs, create headings in your document by applying Heading Styles.

sdddd

Running A Recorded Macro (3)

- After selecting the text and running the macro again, whatever text was highlighted now becomes bold.



This document does not contain headings.

To create navigation tabs, create headings in your document by applying Heading Styles.

sdddd

Other Examples Of Useful (?) Recorded Macros

- Printing: selecting a printer and setting print options can be tedious if your print driver application does not save your previous selections.
  - Even if previous selections are saved recording a macro may be useful if you have multiple printing profiles e.g. I print low quality black and white double sided documents stapled on the top left for staff and high quality color single sided with multiple staples with glossy paper for clients.
- Entering hard to spell words (especially if they are new)
  - Example: "Gotterdammerung" is hard enough to spell but this word is supposed to have the "umlaut" for the 'o' (actually ö) and 'a' (actually ä).
  - JT: shifting to German keyboard in Word can be done via: `<Ctrl>-<shift>-<;>` (this key combination must be entered prior to typing *each* alternate character).

Method 1: Recording Macros (Comments)

- It's a good and simple way for anyone to automate a tedious series of commands in Word.
- This approach is quite limited.
  - E.g. how would you record a macro to automatically open, edit and print all the documents in a folder?
  - The main goal in introducing the recording approach is to make you aware of this capability for your future use.
- For the assignment this approach will not work.
- Also on the exam you must manually write macros (on paper) or trace a macro (figure out what happens when it runs).
  - "Recording" is not feasible for the exam

How To Create VBA Programs

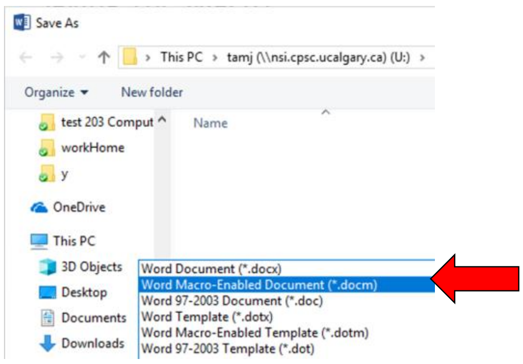
1. **Method 1**: Record the macro automatically: keystrokes and mouse selections will be stored as part of the program
   – An example was just covered.
2. **Method 2**: Manually enter the program (type it in yourself into the VBA editor)
   – This is how you are to complete your assignment and is how many VBA programs are created.
   – Consequently the remainder of the course notes will focus on Method 2

Steps For Method 2: Typing In Your VBA Macro Programs

1. Open Word
2. If you created a new blank document: change this document to one that contain macros:
   – Save As (document that can contain a macro which is a “Word macro enabled document” or ‘docm’)
3. View->Macro
4. Save the VBA macro program into current document if not selected:
   – Select the option that includes the name of the Word document that you just created.
   – Under macro name enter a good descriptive name (no spaces)
   – Click the create button
5. Type in your program (more later) e.g. `MsgBox("hi")`

Step 2: Saving The Macro Into A Document That Can Contain Macros

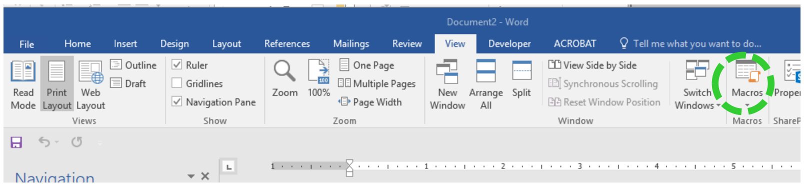
- Save using Word rather than in the VBA editor
- **Make sure you 'Save as' as a MACRO ENABLED Word document**



Word Document (*.docx)
Word Macro-Enabled Document (*.docm)
Word 97-2003 Document (*.doc)
Word Template (*.dotx)
Word Macro-Enabled Template (*.dotm)
Word 97-2003 Template (*.dot)

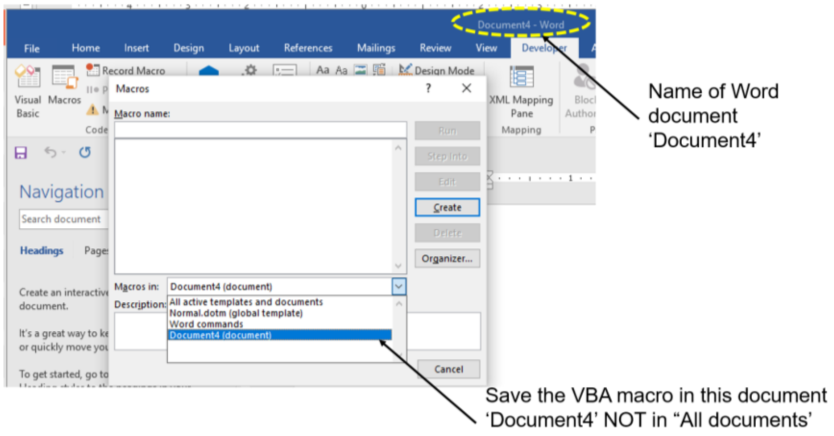
Step 3: Viewing The Ability To Write Programs In Word

- `View->Macros`



Step 4: Saving The Macro In Current (Not All) Word Documents

- Reminder: After the previous step (clicking on "Visual Basic") you will need to indicate that you want to save the VBA program in the **CURRENT DOCUMENT** not in all documents.



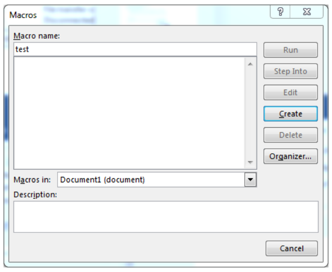
Macros in: Document4 (document)

Description:

All active templates and documents
Normal.dotm (global template)
Word commands
Document4 (document)

(Prior To Step 5)

- "`Macro name`": usually you should enter a good self descriptive name, if you can't think of anything else for your test program then call it 'test'.
  - Then select 'create'



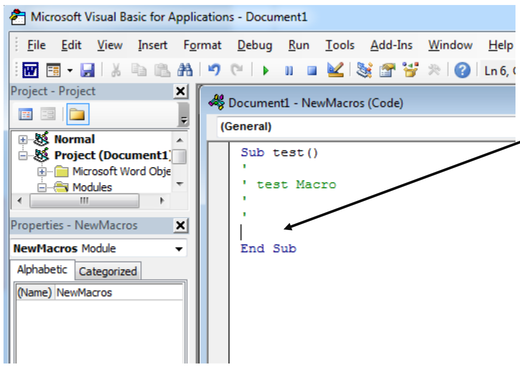
Macro name: test

Macros in: Document1 (document)

Description:

Step 5:Entering The Program Into the VBA Editor

- (Selecting ‘create’ in the previous step then causes the VBA editor to appear)



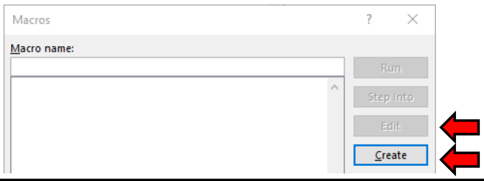
```
Sub test()
'
' test Macro
'
'

End Sub
```

Enter your program here (indent each level by at least one tab), until you see the examples in the last section of notes there will only be one level of indenting.

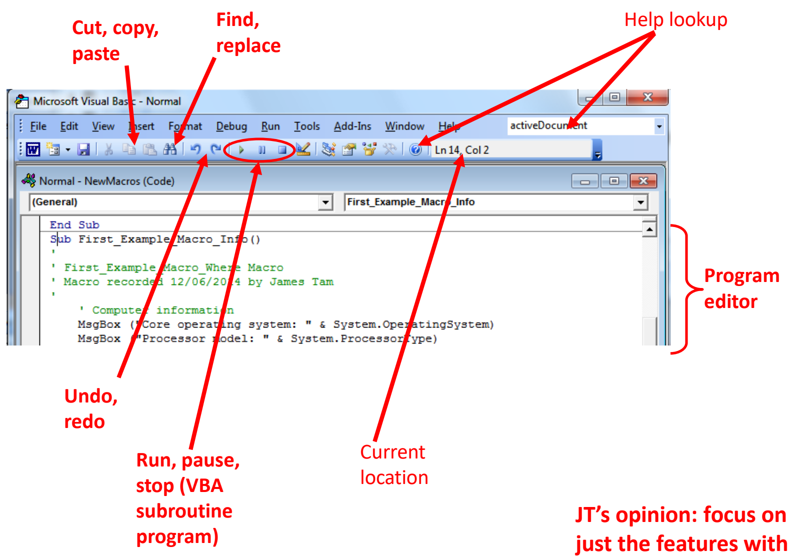
- Much of the rest of the VBA lectures will describe the types of instructions that you can type between the ‘sub’ and ‘end sub’

The Visual Basic Editor

- You don’t need to familiarize yourself with every detail of the editor in order to create VBA programs.
- Just a few key features should be sufficient
- Starting the editor:
  - Because VBA programs are associated with an office application open the editor from MS-Word
  - ‘View->Macros->’
    - Making a new macro then select: ‘Create’
    - Accessing a macro you already created: ‘Edit’ (greyed out in the image because no macros have been created yet).



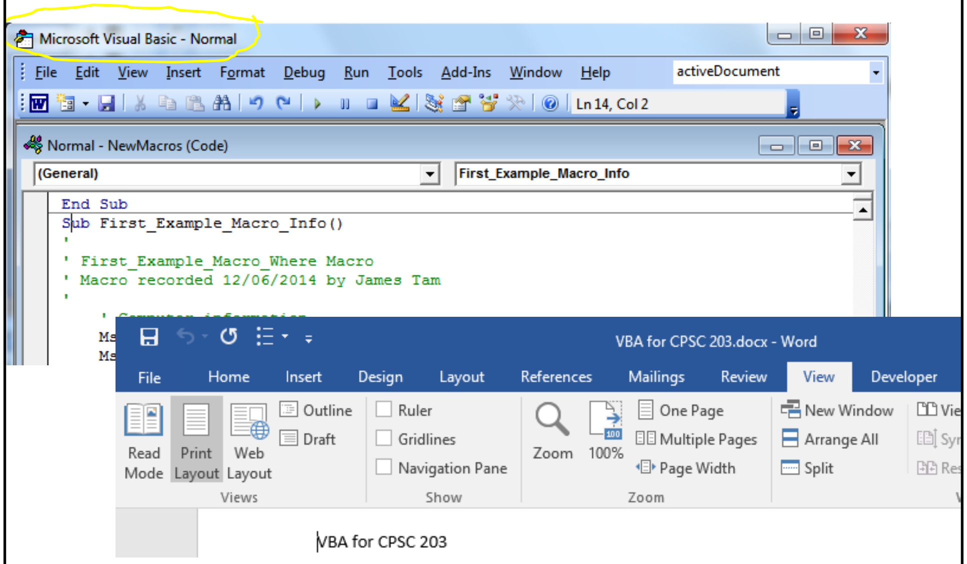
Overview Of The Important Parts Of The VBA Editor


Cut, copy, paste
Find, replace
Help lookup
Program editor
Undo, redo
Run, pause, stop (VBA subroutine program)
Current location
Microsoft Visual Basic - Normal
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
activeDocument
Ln 14, Col 2
Normal - NewMacros (Code)
(General)
First_Example_Macro_Info

```
End Sub
Sub First_Example_Macro_Info()
'
' First_Example_Macro_Where Macro
' Macro recorded 12/06/2014 by James Tam
'
    ' Computer information
    MsgBox ("Core operating system: " & System.OperatingSystem)
    MsgBox ("Processor model: " & System.ProcessorType)
```

JT's opinion: focus on just the features with bolded text

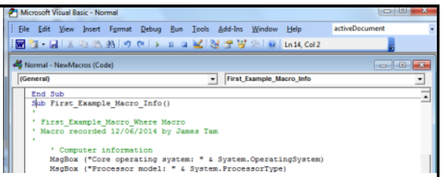
VBA Editor: Don't Mix It Up With The Word Editor


Microsoft Visual Basic - Normal
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
activeDocument
Ln 14, Col 2
Normal - NewMacros (Code)
(General)
First_Example_Macro_Info
VBA for CPSC 203.docx - Word
File Home Insert Design Layout References Mailings Review View Developer
Read Mode
Print Layout
Web Layout
Outline
Draft
Views
Ruler
Gridlines
Navigation Pane
Show
Zoom
100%
One Page
Multiple Pages
Page Width
Zoom
New Window
Arrange All
Split

```
End Sub
Sub First_Example_Macro_Info()
'
' First_Example_Macro_Where Macro
' Macro recorded 12/06/2014 by James Tam
'
```

VBA for CPSC 203

Writing A Program In The VBA Editor


Microsoft Visual Basic - Normal
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
activeDocument
Ln 14, Col 2
Normal - NewMacros (Code)
(General)
First_Example_Macro_Info

```
End Sub
Sub First_Example_Macro_Info()
'
' First_Example_Macro_Where Macro
' Macro recorded 12/06/2014 by James Tam
'
    ' Computer information
    MsgBox ("Core operating system: " & System.OperatingSystem)
    MsgBox ("Processor model: " & System.ProcessorType)
```

- **Format**:

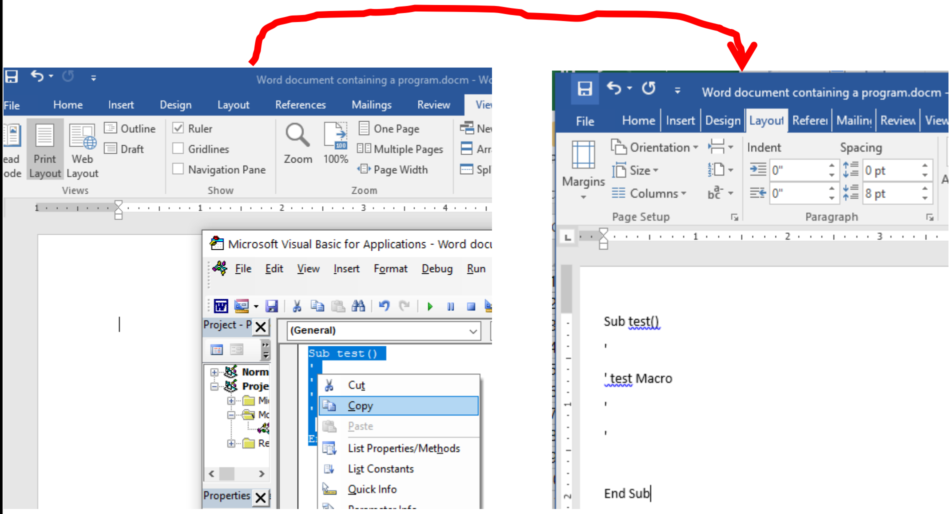
```
' Program documentation goes here (more on this later)
sub <sub-program name>()
    Instructions in the body of program (indent  by 1 tab)
End Sub
```

- **Example**:

```
' Author, version, features etc.
Sub first_example_macro_info()
    MsgBox ("First computer program")
End Sub
```

- Note: large VBA programs have multiple (sub) parts but **for this class you only need to define a single 'sub'**.

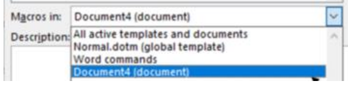
Copy-Paste The Program (VBA Editor->Word)



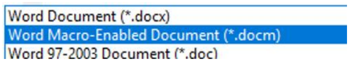
```
Sub test()
'
' test Macro
'
'

End Sub
```

Transferring Your Work

- If you work on your own computer you need to test your program periodically.
- If you work only on campus computers then you need to save on a flash drive or via the cloud.
- Important summary:
  - Save the VBA **macro in the current Word document**:



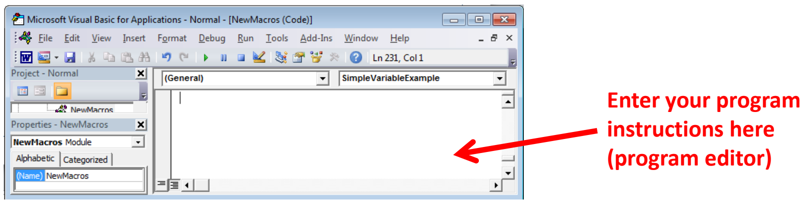
  - And you **saved that document as a Word Macro-Enabled document** (`*.docm`) then you simply have to transfer that Word document.


Word Document (*.docx)
Word Macro-Enabled Document (*.docm)
Word 97-2003 Document (*.doc)

  - **Copy-paste the VBA program into the body of regular Word document** (at least copy it in the document containing your program) just to be safe.

First VBA Example

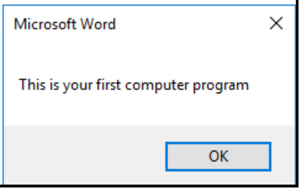
- Learning Objectives:
  - Creating/running a VBA program
  - Creating a popup message using a "`MsgBox`"
- Reminder steps (since this is your first example)
  - Start up the application (MS-Word)
  - Complete Steps 2 – 4 of creating a VBA program by typing
  - If successful you should see something similar to the image


Microsoft Visual Basic for Applications - Normal - [NewMacros (Code)]
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
Ln 231, Col 1
Project - Normal
NewMacros
Properties - NewMacros
NewMacros Module
Alphabetic Categorized
(Name) NewMacros
(General)
SimpleVariableExample
Enter your program instructions here (program editor)

First VBA Example (2)

- Type in or cut-and-paste the following example into the *VBA editor* (see the image previous slide for how the editor appears)
  - This is **NOT** the same as pasting it directly into MS-Word.
  - **Word document containing the macro (empty document see the macro editor for the important details)**: `1firstExampleMacro.docm`

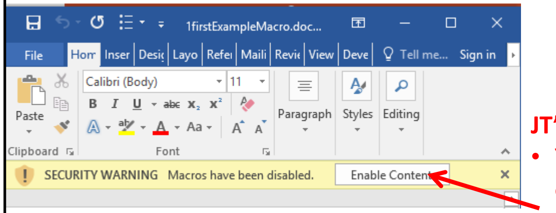
```
Sub first_example_macro_info()
    MsgBox ("This is your first computer program")
End Sub
```


Microsoft Word

This is your first computer program

Opening Word Documents Contain Macros

- If you don't change the default setting in Word you should get a warning when you open the Word document (containing macros).
- You can "enable the content" of your own programs or the examples that I wrote.



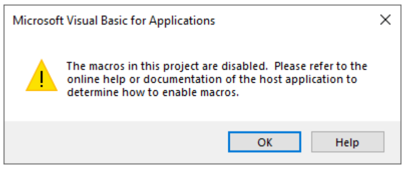
SECURITY WARNING Macros have been disabled.

JT's caution

- You should NOT casually select this option for all MS-Word documents
- It's recommended that you ONLY enable macros you have created (or the lecture & tutorial examples)

New: Opening Word Documents Contain Macros

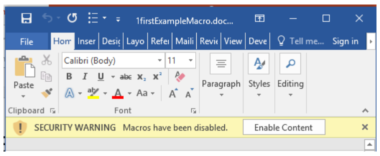
- Recently (noticed as of **Oct 2020**) there was a change in how Word handles Macro security.
- The popup shown in the previous slide may not appear yet and macros will not run even if you change the security settings in Word to the least restrictive security settings! :(
  - The popup error message is not helpful. ¯_(ツ)_/¯


Microsoft Visual Basic for Applications

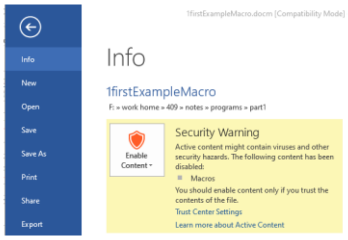
The macros in this project are disabled. Please refer to the online help or documentation of the host application to determine how to enable macros.

New: Opening Word Documents Contain Macros (2)

- What's the issue?
  - Microsoft's Word developers in it's collective wisdom decided to move the option to enable macros from a prominent place in the Home tab.


SECURITY WARNING Macros have been disabled.

  - You now have to look in the `File` tab under Info. (Much better right?)


1firstExampleMacro.docm [Compatibility Mode]

Info

1firstExampleMacro

F: » work home » 409 » notes » programs » part1

Security Warning

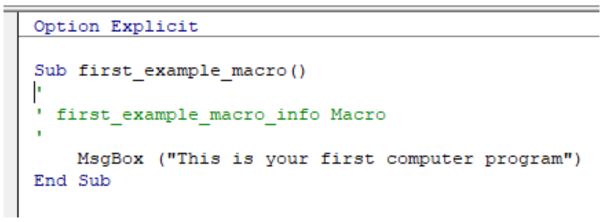
Active content might contain viruses and other security hazards. The following content has been disabled:

- Macros

You should enable content only if you trust the contents of the file.

Option Explicit

- It's a good idea to include this in all your programs.
- It should be located before the 'sub' keyword.

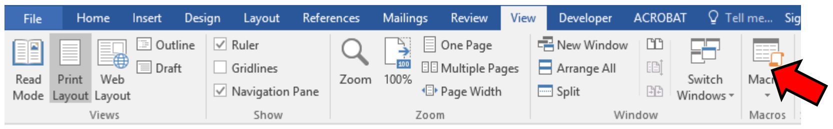


```
Option Explicit

Sub first_example_macro()
'
' first_example_macro_info Macro
'
    MsgBox ("This is your first computer program")
End Sub
```

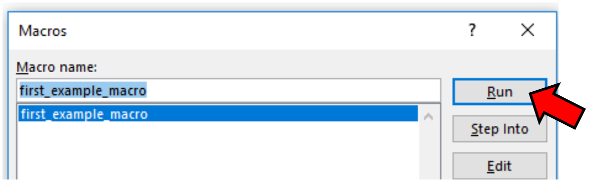
- Explanations about what it does will come in subsequent lectures (in conjunction with more advanced material).
- For now "just add it in" because it may result in fewer problems for you later.

Reminder: Running Macros

- Click `View->Macros`

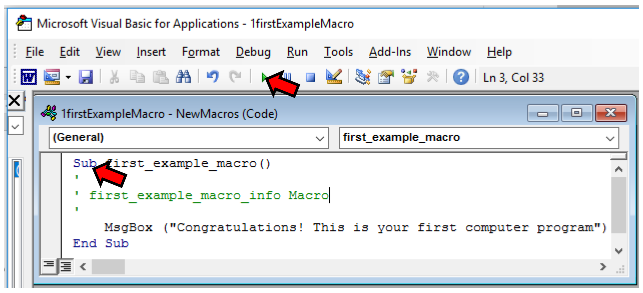


- The single macro should be highlighted, then click 'run'



Running VBA Programs You Have Entered (2)

- Alternatively: you can run the program right after you have entered it (in the editor).


Microsoft Visual Basic for Applications - 1firstExampleMacro
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
Ln 3, Col 33
1firstExampleMacro - NewMacros (Code)
(General)
first_example_macro

```
Sub first_example_macro()
'
' first_example_macro_info Macro
'
    MsgBox ("Congratulations! This is your first computer program")
End Sub
```

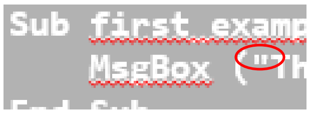
1. Ensure correct program "sub" is to be executed (click there)
2. Press the 'play/run' button or "F5"

Writing Programs On Your Own

- After this point you will be required to type in your programs from scratch rather than copy-paste a pre-created program into the editor.
- This means that additional details are needed.

VBA Programming: How To Study

- At the very least: try typing the programs into the VBA editor or cutting and pasting them yourself (watch for altered characters such as quotes)



```
Sub first_examp
    MsgBox ("Th
End Sub
```

```
Sub first_example
    MsgBox(“This
```

- For the more complex programs (end of this section as well as the next section) try re-creating the programs on your own:
  - Think about what tasks are accomplished by my solution program
  - Without looking at my solution try entering into the program into the VBA editor to accomplish these same tasks
- With programming you learn by “doing yourself” rather than by watching someone else ‘do’

Program Structure And The ‘Sub’ Keyword

- Sub stands for ‘subroutine’ or a portion of a VBA program

Format:

```
Sub <subroutine name>()
    <Instructions in the subroutine>
End Sub
```

Header, start of subroutine:
1. Has word ‘Sub’
2. Name of subroutine
3. Set of brackets

Note: all lines in between are indented (1 tab, 2+ tabs if sub-indenting is used – later sets of notes)

- **Example**:

```
Sub First_Example_Macro_Info()

End Sub
```

End of subroutine:
- Has ‘End Sub’

- Unless otherwise told all VBA program statements must be inside the subroutine (“Option Explicit” excluded)

The ‘Sub’ Keyword: 2

- Real world VBA programs will be broken down into multiple ‘subs’ (subroutines or program parts)
- Again: Because this is only brief introduction into writing VBA programs **you will only have to define one subroutine for your assignment**.

Naming The Subroutine

- This is what follows the ‘sub’ keyword.
- Example

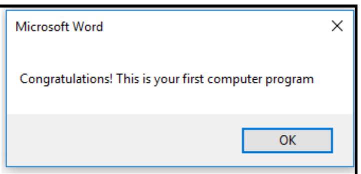
```
Sub formattingResume
End Sub
```

- Naming standards:
  - The name chosen should summarize what the program is supposed to do.
  - The choice of the name will play a role in determining your assignment grade.
  - If you can’t come up with a good name for the assignment then at least make it a practical one for the marker e.g. “a3”

Choosing A Name: VBA Technical Requirements

- Must start with an alphabetic letter, after than any combination of letters and numbers may be used
  - OK: “`assigment1`”, “`a2939`”   Not OK: “`1assignment`”, “`*assignment`”
- Maximum length of 80 characters
- It cannot contain spaces, punctuation or special characters such as # or !
  - ‘`resume headings`’ (Not Allowed: space character)
  - ‘`macros!`’ (Not Allowed: special character)
- Can contain underscores (separate long names)

Message Box


Microsoft Word

Congratulations! This is your first computer program

- (Details of the previous example)
- Creates a popup window to output information from your program
- Useful for testing
  - Is my program working?
  - Which part is running?
- Also useful for displaying status messages about the current state of the program

Creating A **Message Box**

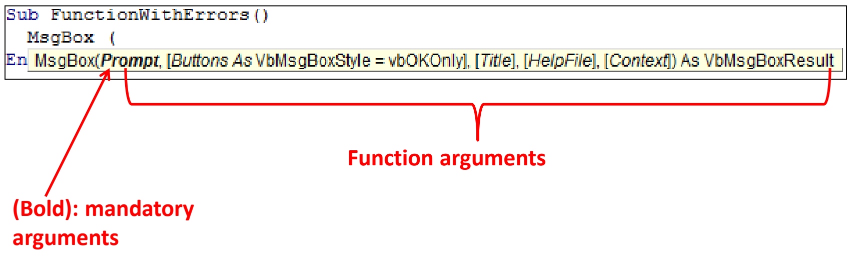
- **Format**:
  `MsgBox ("`*`<Message to appear>`*`")`

- **Example**:
  `MsgBox ("This your first computer program")`

Notes on 'Format':
- Italicized: you have a choice for this part
- Non-italicized: mandatory (enter it as-is)
- Don't type in the angled brackets (used to help you visually group)

VBA Visual Aids: **Function Arguments**

- As you type in the name of VB functions you will see visual hints about the arguments/inputs for the function.

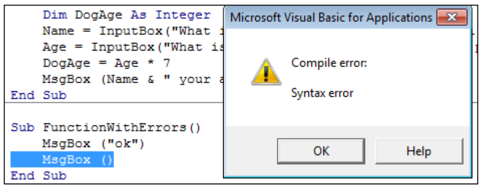


```
Sub FunctionWithErrors()
  MsgBox (
En MsgBox(Prompt, [Buttons As VbMsgBoxStyle = vbOKOnly], [Title], [HelpFile], [Context]) As VbMsgBoxResult
```

  - JT: You won't need to worry about all functional arguments for this class.
  - Enter the function name and then a space

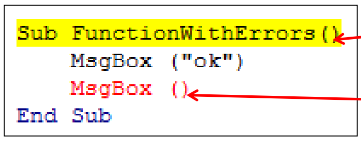
VBA Visual Aids: **Error Information**

- The requirements for forming VBA programming instructions are referred to as the 'syntax' (grammar/rules) of the language.
- Syntax violations are visually highlighted in VBA:



```
    Dim DogAge As Integer
    Name = InputBox("What i
    Age = InputBox("What is
    DogAge = Age * 7
    MsgBox (Name & " your a
End Sub

Sub FunctionWithErrors()
    MsgBox ("ok")
    MsgBox ()
End Sub
```

Required argument missing



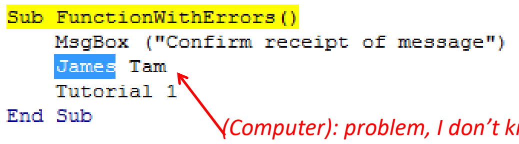
```
Sub FunctionWithErrors()
    MsgBox ("ok")
    MsgBox ()
End Sub
```

Part of program that contains errors (yellow highlight)

Specific statement/instruction causing the error (red font)

Program Documentation

- Your VBA assignment submission must include identification about you and information the features of your program
  - Full name
  - Student identification number
  - Tutorial number (if applicable)
  - List the program features (from the assignment description) and clearly indicate if the feature was completed or not completed.
  - Program version
- DON'T just enter this information into your program instructions



```
Sub FunctionWithErrors()
    MsgBox ("Confirm receipt of message")
    James Tam
    Tutorial 1
End Sub
```

Instructions for the computer

(Computer): problem, I don't know how to run the "James Tam" instruction

Program Documentation (2)

- You must '**mark**' this information so it doesn't cause an error
  - The marking will indicate to the VBA translation mechanism that the line is for the reader of the program and not to be translated and executed
  - The marking is done with the single quote '
- **Format**:

```
'<Documentation>
```

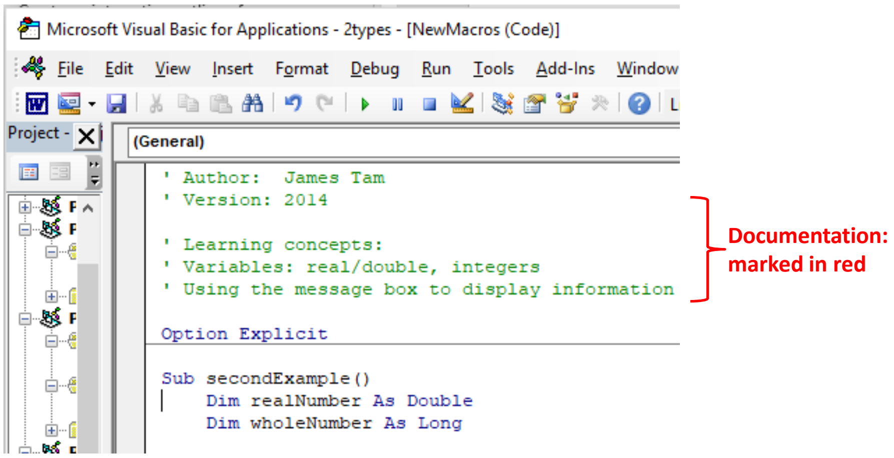
- **Example**:

```
'Author: James Tam
```

- No error: Everything after the quote until the end of the line will not be executed
- That means documentation doesn't have to be a valid and executable instruction

Program Documentation (3)

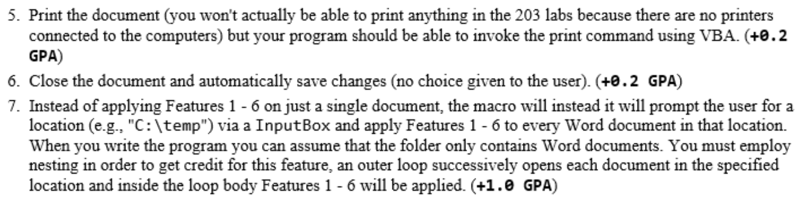
- Contact information should be located before your program
- Before the 'sub' keyword and before the "Option Explicit"


Microsoft Visual Basic for Applications - 2types - [NewMacros (Code)]
File Edit View Insert Format Debug Run Tools Add-Ins Window
Project -
(General)
Documentation: marked in red

```
' Author:  James Tam
' Version: 2014

' Learning concepts:
' Variables: real/double, integers
' Using the message box to display information

Option Explicit

Sub secondExample()
    Dim realNumber As Double
    Dim wholeNumber As Long
```

Program Documentation (4)

- Program features (this will be worth a fair number of marks)
- Example assignment description



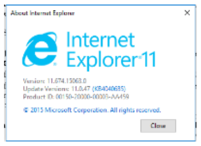
5. Print the document (you won't actually be able to print anything in the 203 labs because there are no printers connected to the computers) but your program should be able to invoke the print command using VBA. (**+0.2 GPA**)
6. Close the document and automatically save changes (no choice given to the user). (**+0.2 GPA**)
7. Instead of applying Features 1 - 6 on just a single document, the macro will instead it will prompt the user for a location (e.g., `"C:\temp"`) via a `InputBox` and apply Features 1 - 6 to every Word document in that location. When you write the program you can assume that the folder only contains Word documents. You must employ nesting in order to get credit for this feature, an outer loop successively opens each document in the specified location and inside the loop body Features 1 - 6 will be applied. (**+1.0 GPA**)

- Program documentation
  - ' Author: James Tam   ID: 123456
  - ' Version: Nov 2, 2015
  - ' Tutorial: 99
  - ' PROGRAM FEATURES
  - ' #5: Printing: completed
  - ' #6: Close and save: not completed
  - etc

} These are easy marks to earn!

Program Documentation (5)

- Versioning
  - It's a commonly used industry practice:



About Internet Explorer

Internet Explorer 11

Version: 11.674.15063.0
Update Versions: 11.0.47 (KB4040685)
Product ID: 00150-20000-00003-AA459

© 2015 Microsoft Corporation. All rights reserved.

  - For assignments you can either use version numbers or dates (recommended)

Program Versioning And Back Ups

- As significant program features have been completed (tested and the errors removed/debugged) a new version should be saved in a separate file.

A3.docm

```
' Version: Oct 2, 2012
' Program features:
' (2) Count, display typos

' Version: Sept 30, 2012
' Program features:
' (5) Print
' (6) Save & close
```

Make new backup file

A3.Oct2

```
' Version: Oct 2, 2012
' Program features:
' (2) Count, display typos

' Version: Sept 20, 2012
' Program features:
' (5) Print
' (6) Save & close
```

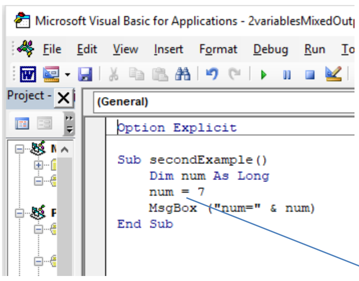
A3.Sept30

```
' Version: Sept 20, 2012
' Program features:
' (5) Print
' (6) Save & close
```

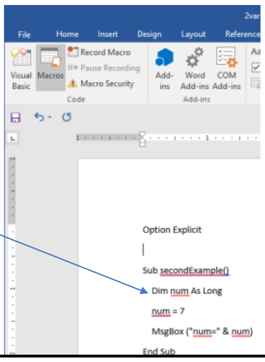
Don't touch the backups (date in the file name) and only work on A3.docm

Additional Backup Tip: Copy-Paste

- Periodically copy-paste the VBA program into a regular Word document


Microsoft Visual Basic for Applications - 2variablesMixedOut
File Edit View Insert Format Debug Run To
Project
(General)

```
Option Explicit

Sub secondExample()
    Dim num As Long
    num = 7
    MsgBox ("num=" & num)
End Sub
```



```
Option Explicit

Sub secondExample()
Dim num As Long
num = 7
MsgBox ("num=" & num)
End Sub
```

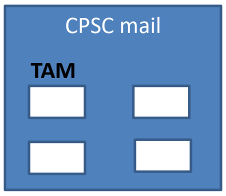
Variables

- Used to temporarily store information at location in memory
- Variables must be declared (created) before they can be used.
- **Format for declaration**:

  `Dim <Variable name> as <Type of variable>`

- **Example declaration**:

  `Dim BirthYear as Long`



Common Types Of Variables

Type of information stored	VBA Name	Example variable declaration	Default Value
Whole numbers	Long	`Dim LuckyNumber as Long`	0
Real numbers	Double	`Dim MyWeight As Double`	0
Chararacters[1]	String[2]	`Dim Name As String`	Empty string
Date[3]	Date	`Dim BirthDate As Date`	00:00:00

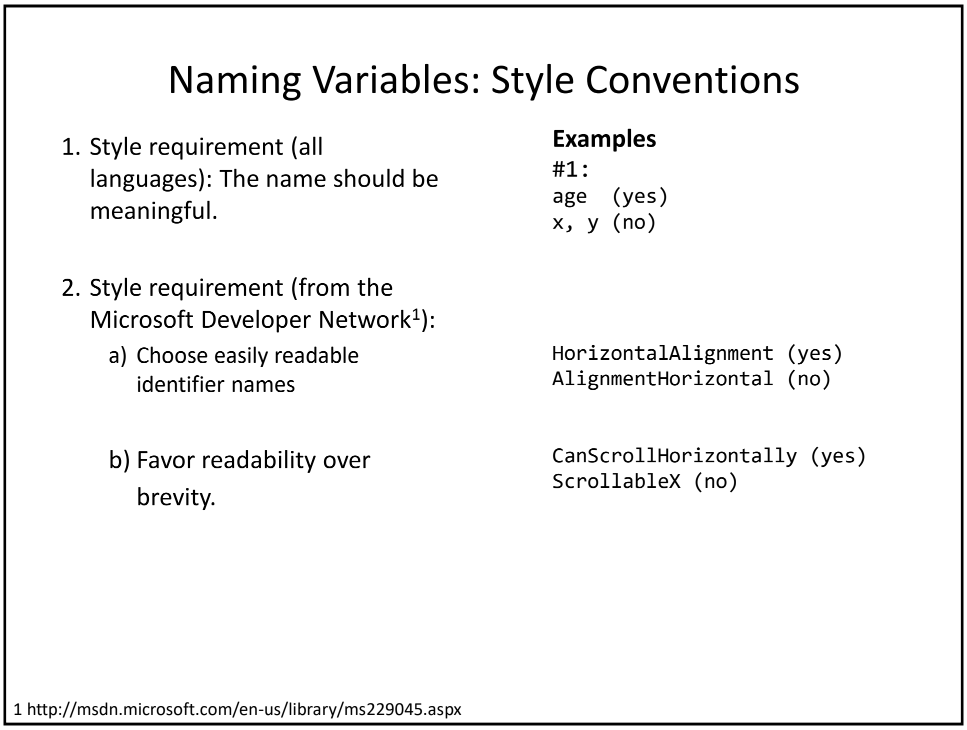
1) Any visible character you can type and more e.g., 'Enter' key
2) Each string can contain up to a maximum of 2 billion characters
3) Format: Day/month/year

Variable Naming Conventions

- Language requirements (syntax):
  - Rules built into the Visual Basic (recall VBA is essentially Visual Basic tied to an MS-Office Application) language.
  - Somewhat analogous to the grammar of a 'human' language.
  - If the rules are violated then the typical outcome is the program cannot execute.
- Style requirements:
  - Approaches for producing a well written program.
  - (The real life analogy is that something written in a human language may follow the grammar but still be poorly written).
  - If style requirements are not followed then the program can execute but there may be other problems (e.g., it is difficult to understand because it's overly long and complex - more on this during the term).

Naming Variables: VBA Language Requirements

- Names **must** begin with an alphabetic character
  - OK: `name1` Not OK: `1name`
- Names **cannot** contain a space
  - OK: `firstName` Not OK: `first name`
- Names **cannot** use special characters anywhere in the name
  - Punctuation: `! ? .`
  - Mathematical operators: `+ - * / ^`
  - Comparison operators: `< <= > >= <> =`



Naming Variables: Style Conventions

1. Style requirement (all languages): The name should be meaningful.
2. Style requirement (from the Microsoft Developer Network[1]):
   a) Choose easily readable identifier names
   b) Favor readability over brevity.

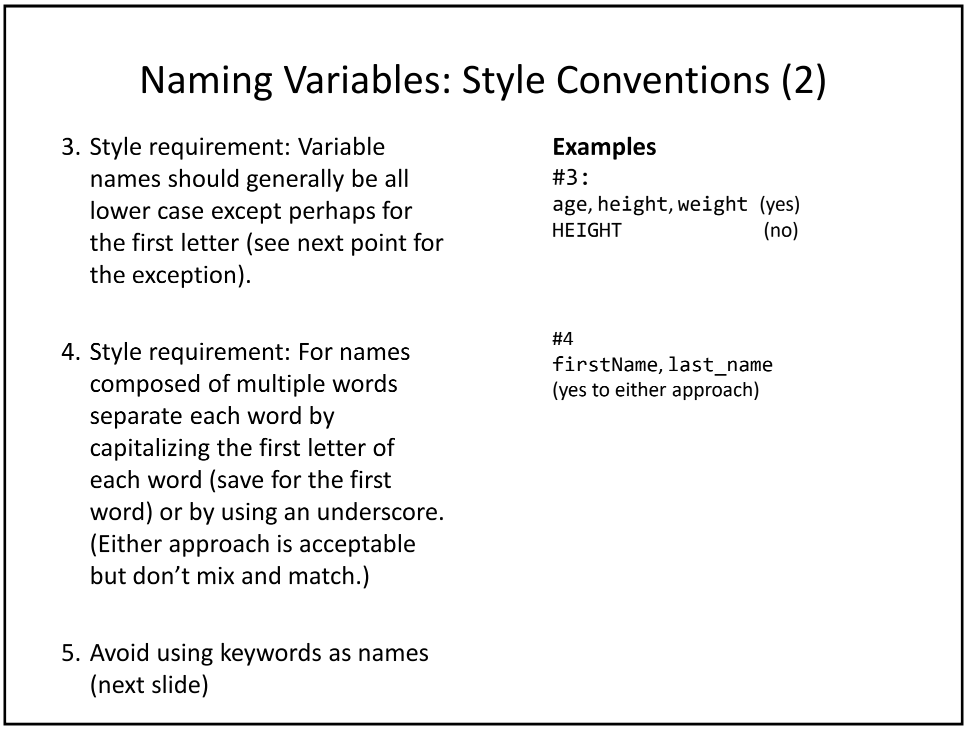
Examples

```
#1:
age  (yes)
x, y (no)
```

```
HorizontalAlignment (yes)
AlignmentHorizontal (no)
```

```
CanScrollHorizontally (yes)
ScrollableX (no)
```

1 http://msdn.microsoft.com/en-us/library/ms229045.aspx



Naming Variables: Style Conventions (2)

3. Style requirement: Variable names should generally be all lower case except perhaps for the first letter (see next point for the exception).
4. Style requirement: For names composed of multiple words separate each word by capitalizing the first letter of each word (save for the first word) or by using an underscore. (Either approach is acceptable but don't mix and match.)
5. Avoid using keywords as names (next slide)

Examples

```
#3:
age, height, weight  (yes)
HEIGHT               (no)
```

```
#4
firstName, last_name
```
(yes to either approach)

Some Common Visual Basic Keywords[1]

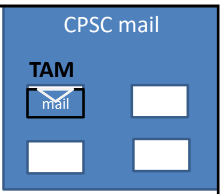
And	Boolean	Call	Case	Catch	Continue
Date	Decimal	Default	Dim	Do	Double
Each	Else	End	Erase	Error	Event
Exit	False	Finally	For	Friend	Function
Get	Global	Handles	If	In	Inherits
Integer	Interface	Is	Let	Lib	Like
Long	Loop	Me	Mod	Module	Next
Not	Nothing	Of	On	Operator	Option
Optional	Or	Out	Overrides	Partial	Private
Property	Protected	Public	Resume	Return	Select
Set	Shadows	Short	Single	Static	Step
Stop	String	Sub	Then	Throw	To
True	Try	Using	Variant	When	While
Widening	With				

1 The full list can be found on the MSDN http://msdn.microsoft.com/en-us/library/dd409611.aspx

Variable Naming Conventions: Bottom Line

- Both the language and style requirements should be followed when declaring your variables.

Examples Of Assigning Values To Variables



Note: some types of variables requires some mechanism to specify the type of information to be stored:

- Strings: the start and end of the string must be marked with double quotes "
- Date: the start and end of the string must be marked with the number sign #

```
Dim LuckyNumber As Long
LuckyNumber = 888

Dim BirthDay As Date
BirthDay = #11/01/1977#

Dim MyName As String
MyName = "James"
```

Basic Mathematical **Operators**

Operation	Symbol used in VBA	Example
Addition	+	2 + 2
Subtraction	-	3 – 2
Multiplication	*	10 * 10
Division	/	81 / 9
Exponent	^	2 ^ 3

MsgBox And Variables

- Typing the name of a variable in a message box (no quotes) will display the current contents of that variable.
- Example (from a program coming up)

```
Dim realNumber As Double
realNumber = 1 / 3
MsgBox (realNumber)
```

0.333... appears

```
MsgBox ("realNumber")
```

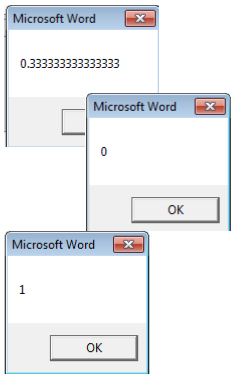
realNumber appears

Second VBA Example (2)

Learning objective: Display the contents of variables

Word document containing the macro: `2types.docm`

```
Sub secondExample()
    Dim realNumber As Double
    Dim wholeNumber As Long

    realNumber = 1 / 3
    MsgBox (realNumber)
    wholeNumber = 5 / 10
    MsgBox (wholeNumber)
    wholeNumber = 6 / 10
    MsgBox (wholeNumber)
End Sub
```



JT's note: Anything over 0.5 is rounded up

Common Mistake #1

- A variable is just that – it can change as a program runs.
- Approach #1: variable not used (lacks flexibility)

```
MsgBox ("My age is...")
MsgBox ("...37")
```

- Approach #2: variable employed (age can be changed with any mathematical expression)

```
Dim age As Long
age = 37
MsgBox ("My age is...")
MsgBox (age)
age = 38
MsgBox ("My age is...")
MsgBox (age)
```

MsgBox: Displaying Mixes Of Strings And Variables

- **Format**:

```
MsgBox ("<Message1>" & <variable name>)
```

- **Learning objective: Mixed output in a MsgBox (variables and constant text strings)**
- **Name of the online example**: `3variablesMixedOutput.docm`

```
Dim num as Long
num = 7
MsgBox ("num=" & num)
```

```
"num="      : A literal string
num         : contents of a variable (slot in memory)
```

Why Mix The Display Of Strings & Variables

- Labeling variables as they appear makes your program easier to understand

```
4.3
3.3
3.7
2.0
4.0
4.0
3.9
1.0
```

Vs.

```
Student 1:  4.3
Student 2:  3.3
Student 3:  3.7
Student 4:  2.0
Student 5:  4.0
Student 6:  4.0
Student 7:  3.9
Student 8:  1.0
```

Student Exercise #1: Mixed Output

- What is the output of the following `MsgBox`:

```
Sub exercise1()
    Dim num As Long
    Dim aStr As String
    num = 12
    aStr = "num="
    MsgBox (aStr & num)
End Sub
```

- **Solution**: `Exercise1.docms`

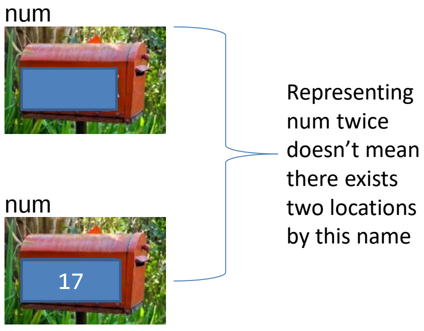
Variables: Metaphor To Use

- Think of VBA variables like a “mail slot in memory”
- Unlike an actual mail slot **computer variables can only hold one piece of information**
  - Adding new information results in the old information being replaced by the new information

```
Dim num as Long
```

```
num = 1
```

```
num = 17
```


num
num
17
Representing num twice doesn’t mean there exists two locations by this name

Common Mistake #2

- Assigning values from one variable does not ‘link’ them

```
num1 = 1
num2 = num1
num1 = 2
```

Student Exercise #2

- Also each computer variable is separate location in memory.
- Each location can hold information independently of other locations.
  - Note: This works differently than mathematical variables!

```
Sub exercise2()
    Dim num1 as Long
    Dim num2 as Long
    num1 = 1
    num2 = num1
    num1 = 2
End Sub
```

  - What is the result?
  - **Solution**: `Exercise2.docms`

Getting **User Input**

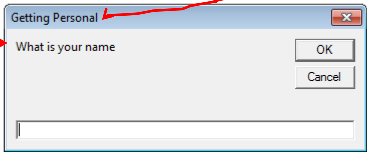
- A simple approach is to use an **Input Box**
- Format:

```
<Variable name> = InputBox(<"Prompt">, <"Title bar">)
```

- Example:

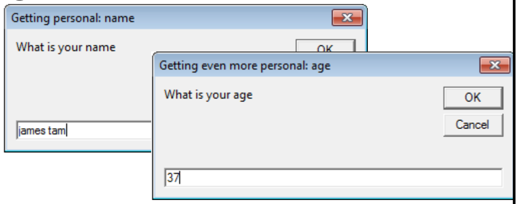
```
Name = InputBox("What is your name", "Getting Personal")
```

- Note: only the string for the prompt (first) is mandatory.
- If the title bar information is omitted then the default is the application name ("Microsoft Word")



Example: InputBox

- **Learning objective**: getting information from the user.
- **Word document containing the macro:** 4inputBox.docm



Getting personal: name

What is your name

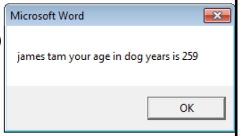
james tam

Getting even more personal: age

What is your age

37

```
Sub InputExample()
    Dim age As Long
    Dim name As String
    Dim dogAge As Long
    name = InputBox("What is your name", "Getting personal: name")
    age = InputBox("What is your age", "Getting even more personal: age")
    dogAge = age * 7
    MsgBox (Name & " your age in dog years is " & dogAge)
End Sub
```



Microsoft Word

james tam your age in dog years is 259

Note: there are two input boxes, one that prompts for the name and the other for the age. Each is given a *self-descriptive name* to distinguish them (an example of good programming style – more on this shortly)

Student Exercise #3: Getting Input, Display Output Based On The Input

- Write a program that will ask the user for their age and name.
- The program will then display the name and age of the user in the following format:
  - **Format of what program displays**:
    - `<Name> : <Age> is a good age`
  - **Example output**: user enters 'james' and 1
    - `James: 1 is a good age`
- (You can use Example 4 as a starting point for your program).
- **Solution**: Exercise3.docms

The Line Continuation Character

- Learning objective: how to split a long VBA instruction between multiple lines.
- Misnomer: The “line continuation character” is actually two characters.
  - Space (i.e. press the space bar)
  - This is immediately followed by the under score

Line Continuation Character Example

- Example VBA document containing the program: **5lineContinuationCharacter**

```
Dim name As String
Dim address As String
Dim phone As String
Dim age As Long
name = InputBox("What is your name")
address = InputBox("Address: ")
phone = InputBox("Phone (format: (xxx)xxx-xxxx): ")
age = InputBox("What is your age")
MsgBox (name & " lives at " & address _
        & " " & phone & " " & age)
```

The last line is equivalent to:

```
MsgBox (name & " lives at " & address & " " & phone & " " & age)
```

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 (2)

- What are commonly used variable ‘types’ in VB
- How to get user input with an Input Box “`InputBox`”
- How to create program documentation (as well contact information that should be included in documentation)

Images

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