Programming Fundamentals In VBA (Visual Basic For Applications)

You will learn about basic program writing tools such as input-output, variables, branching and looping mechanisms.

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

B.A.S.I.C.

- Beginner's All-Purpose Symbolic Instruction Code (BASIC)
  - From: www.acm.org (original full article: http://time.com/69316/basic/)
- A widely used programming language
- It was relatively simple to learn (statements were "English-like" e.g., "if-then")
- Widely popular and it was commonly packaged with new computers in the 1970's and 1980's
- (A then relatively unknown company: Microsoft got it's initial cash inflows and reputation producing several versions of the language)

Visual Basic

- A newer programming language developed by Microsoft
- It was designed to make it easy to add practical and useful features to computer programs e.g., programmers could add a graphic user interface, database storage of information etc.
- Also it can take advantage of the built in capabilities of the various versions of the Windows operating system
  - Why write a feature of a program yourself when it already "comes with the computer"
- For more information:
  - http://msdn.microsoft.com/en-us/library/2x7h1hfk.aspx

Visual Basic For Applications (VBA)

- Shares a common core with Visual Basic.
  - Statements 'look' similar
- Unlike Visual Basic, VBA programs aren't written in isolation (creating a program just for it's own sake).
  - Most programs are written to be standalone: a computer game can be run without (say) running a web browser or MS-Office.
- VB = Visual Basic, VB_A_ = Visual Basic for _Applications_
- Each VBA program 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).
  - The host application is enhanced or supplemented by the VBA program
  - "Why doesn't this stupid word processor have this feature??!!"
    - Now you can add that feature yourself using VBA

Visual Basic For Applications (VBA): 2

- **Important**! Because every VBA program must be run within the context of host application when you are learning to write your programs do not open up an important MS-Word document and run your programs.
  - The host program often needs an Word document in order to run certain capabilities.
  - VBA programs often change documents (formatting, style, text).
  - Therefore use only small 'test' MS-Word documents when running your VBA programs otherwise your information may be lost or corrupted.

Macros

- Macro: a sequence of keystrokes or mouse selections (instructions to the computer) that can be repeated over and over
  - MS-Office can be augmented by writing Macros (essentially computer programs) that will run either for multiple documents or only for a particular document.
  - In this class we will focus solely on MS-Word macro programming
- VBA (as guessed) is an example of a macro programming language e.g., you can write a program that includes a series of formatting and other commands that you frequently carry out in Word documents
- Write the commands once in the form of a program and just re-run this program instead of re-entering each command

Macros And The Web-Based Office

- According to Microsoft macros are not accessible via their online Office products:
  - https://support.office.com/en-us/article/Differences-between-using-a-workbook-in-the-browser-and-in-Excel-f0dc28ed-b85d-4e1d-be6d-5878005db3b6?CorrelationId=917b1609-97e9-4cc7-9eeb-d188939ad740&ui=en-US&rs=en-US&ad=US
- Result: use a computer with the desktop version of Office installed.
  - 203 lab
  - Other campus computers
    - Some 'labs' may have open access hours
  - It is CRUICIAL that you test your program on the computers in the 203 lab because your assignment must work on the lab machines in order to receive credit.

Writing Macros

- It is not assumed that you have any prior experience writing computer programs (macro language or something else).
- Consequently early examples and concepts will be quite rudimentary i.e., "we will go slow"
  - The effect is that you may find that the capabilities of the early examples will duplicate familiar capabilities already built into MS-Word
    - Why are we writing a macro program for this feature?
    - Makes it easier to understand (you know the expected result).
    - Keeps the example simpler.
- Later examples will eventually demonstrate the 'power' of macros
  - You can do things that would be impossible (or at least difficult) with the default capabilities built into MS-Word

Assignments: 203 vs. 217

- Program size (CPSC 217 non-majors programming class):
  - A text-based computer game "The hobbit": 677 line program

- Program size (this class):
  - VBA (changing the capabilities of MS-Word): 60 line program
  - JavaScript (running a program through a web page): 70 line program

Final Exam: Programming Questions

- Average grades (programming questions, final exam):
  - The values indicate that the typical student shows a reasonable grasp of the material (i.e., they did "get through it")
  - In order to do well you need to be coming to class and doing extra work:
    - The **absolute minimum** workload is to complete the assignment
    - Write and trace as many programs as you have time for
    - The more practice you get, the more skilled and knowledgeable you will become

Can You Complete The Following Tasks?

- Open a MS-Word document and replace every instance of one phrase e.g., tamj@ucalgary.ca with another tamj@cpsc.ucalgary.ca
- Open every document in a folder and perform the same search and replace operation:
  - 2 documents?
  - 10 documents?
  - 100 documents?
  - All the documents in a particular folder?
  - What if you just wanted to open the word documents with a particular word or phrase in the name e.g., "assignments_2014"?
- This is an example where writing a macro once is a more efficient approach
  - One answer to the question: "Why are we learning this???"

Advanced Use Of Macros

- Although it's beyond the scope of this class the following example is introduced now to make you aware of the power of VBA and macro languages.
  - It can actually be used to perform real tasks.
- You can use a macro to take advantage of the capabilities of each MS-Office application:
  - Establishing references to applications to 'link' them
  - Take the output from one application and making it the input of another.

Advanced Use Of Macros (2)

- Example: macros can automate the following task
  - Store data in MS-Access
  - Store the query results in MS-Excel and perform calculations on the data
  - Use the formatting capabilities of MS-Word to produce reports
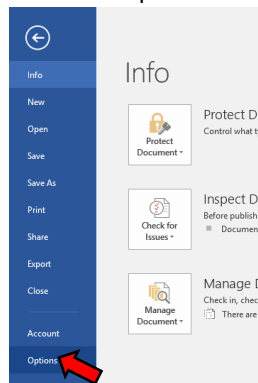 - MS-Outlook can email the final documents

Viewing The 'Developer' Ribbon (MS-Word)

- The macro programming capability comes built-in to the MS-Office suite.
  - You simply have to enable that functionality
- Steps
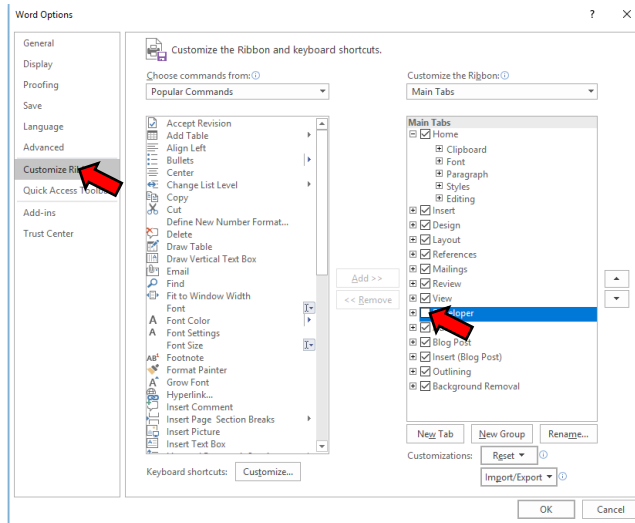
1. Select the 'File' ribbon
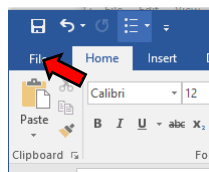
2. Select 'options'
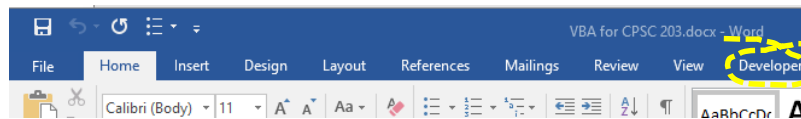
Viewing The 'Developer' Ribbon (MS-Word): 2

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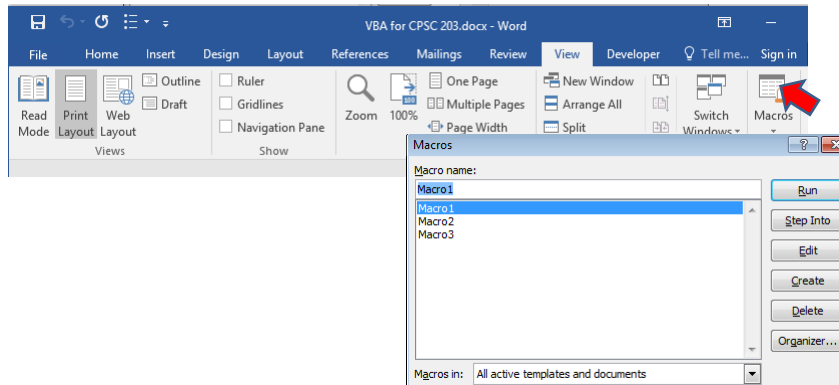

Viewing The 'Developer' Ribbon (MS-Word): 3

- This should add a new ribbon "Developer"

Alternate: View And Run Macros

- You may or may not be able to edit the MS-Word ribbon with some computer labs.
- You can see view macros via the 'view' tab on the ribbon (albeit with fewer options)

Macros And Computer Security

- Computer viruses are simply malicious computer programs.
- Macros can be a useful mechanism for reducing repetition or adding new capabilities to MS-Office.
- But as is the case when writing a computer program malicious code can also be written with a macro and the virus can be activated by just opening the MS-office document that contains the macro.
- Just because you are writing macro programs does not mean that you shouldn't take macro security seriously!

Examples Macros Viruses

- "Melissa": Information about an old but 'successful' Macro Virus
  - http://www.cnn.com/TECH/computing/9903/29/melissa.02.idg/index.html?_s=PM:TECH
  - http://www.symantec.com/press/1999/n990329.html
  - http://support.microsoft.com/kb/224567

- Macro viruses aren't just "ancient history", take the potential threat seriously!
  - http://www.symantec.com/avcenter/macro.html
  - http://www.microsoft.com/security/portal/threat/encyclopedia/search.aspx?query=Virus
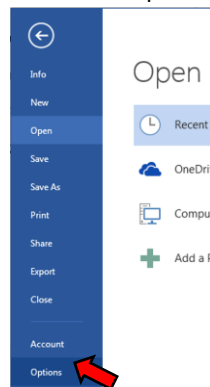 - http://ca.norton.com/search?site=nrtn_en_CA&client=norton&q=macro+virus

Enabling Macros To Run

- If you can' t run macros in MS-office (you see odd error messages) then examine the "Trust Center" settings in Word

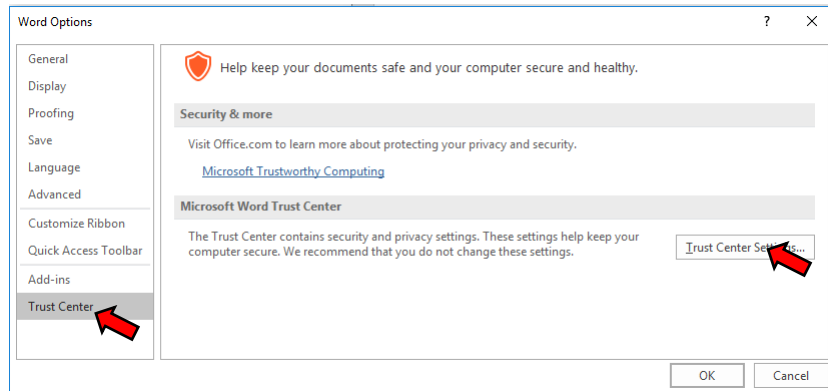1. Select the 'File' ribbon
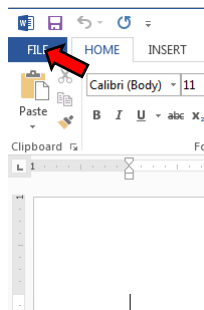
2. Select 'options'

Enabling Macros To Run (2)

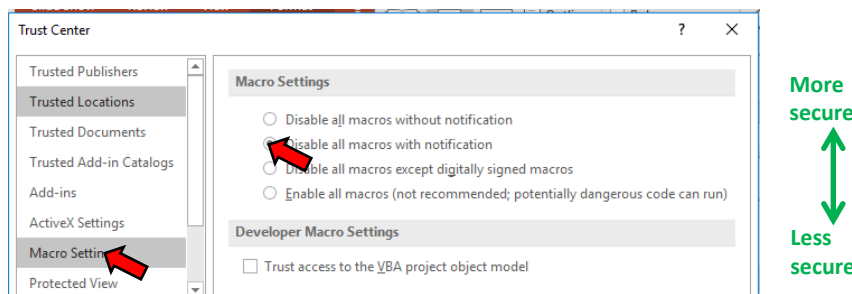3A) Select "Trust Center"   3B) Select "Trust Center Settings"

Enabling Macros To Run (3)

4A) Select "Macro Settings"   4B) Select "Disable all macros with notification"

5) Exit MS-Word (close ALL documents)

Enabling Macros To Run (4)

- The default setting for MS-Word should already be set to "disable macros with notification" but these steps will allow you to use machines that aren't set to default values.

Effect: Opening Word Documents

- Using the default setting will disable all macros by default (safer approach) but you can still enable the macros as the document is opened.
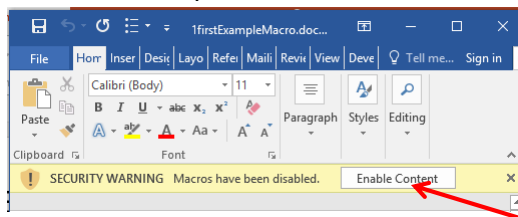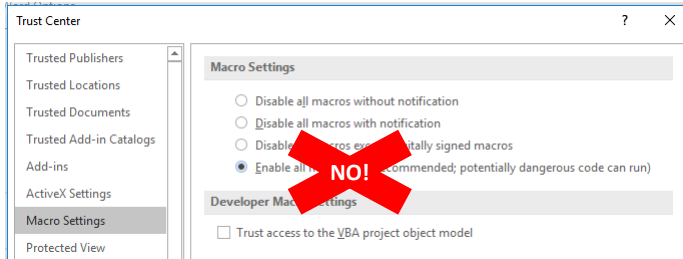

JT's caution
- You should NOT casually select this option for all MS-Word documents
- It's recommended that you ONLY enable macros you have created (or the lecture examples)

Macro Security

- DO NOT take the 'easy' way out

For more information:
http://www.office.microsoft.com/en-us/help/enable-or-disable-macros-in-office-documents-HA010031071.aspx

Creating Macros

1. Record the macro automatically: keystrokes and mouse selections will be stored as part of the macro (you will be briefly shown how to do this in tutorial)
2. **Manually enter the Macro** (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.

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
  - Click the "Visual Basic" icon under "Developer"
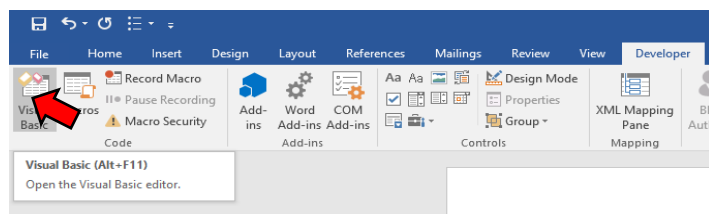
Overview Of The Important Parts Of The VBA Editor

Export:

Useful for transferring or backing up your work
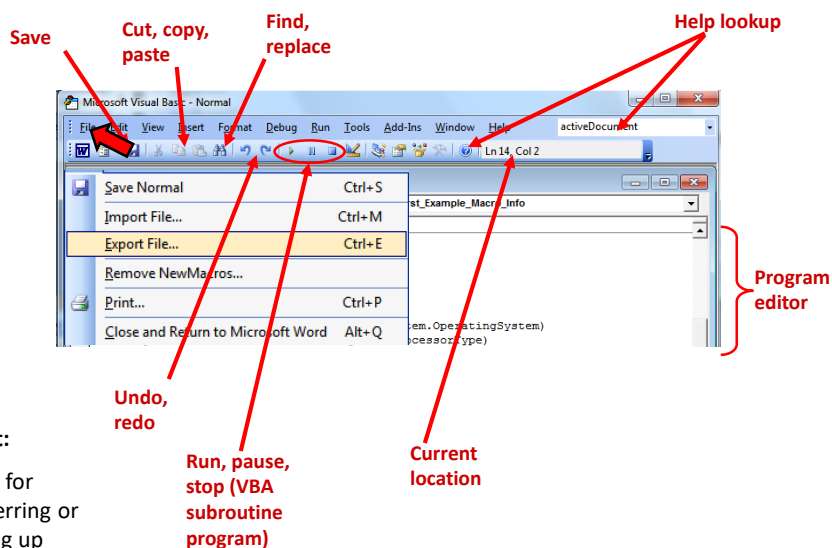
VBA Editor: Don't Mix It Up With The Word Editor
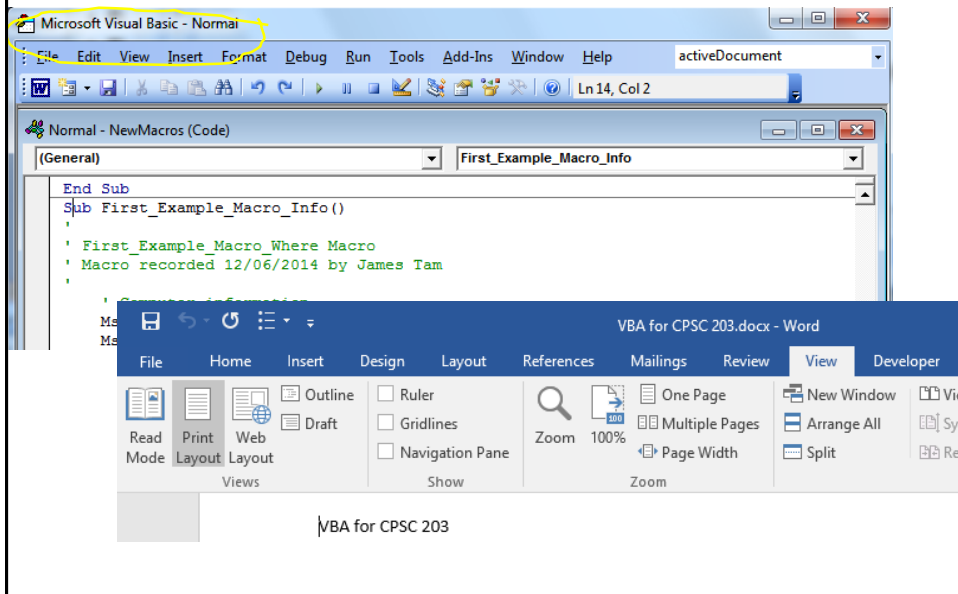
Writing A Program In The VBA Editor

- **Format**:

```
' Program documentation goes here (more on this later)
sub <sub-program name>()
    Instructions in the body of program (indent 4 spaces)
End Sub
```

- **Example**:

```
' Author, version, features etc.
Sub first_example_macro_info()
    MsgBox ("Congratulations! This your 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'.

The 'Sub' Keyword

- Sub stands for 'subroutine' or a portion of a VBA program
  Format:

  `Sub <subroutine name>()`

  `    <Instructions in the subroutine>`

  `End Sub`

- **Example**:

  `Sub First_Example_Macro_Info()`

  `End Sub`

- Unless otherwise told all VBA program statements must be inside the subroutine

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

Note: all lines in between are indented (4 spaces)

End of subroutine:
- Has 'End Sub'

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.
  - It will play a role in determining your assignment grade.

Choosing A Name: 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)

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_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'

First VBA Example

- Learning Objectives:
  - Creating/running a VBA program
  - Creating a Message Box "MsgBox"
- Reminder steps (since this is your first example)
  - Start up the application (MS-Word)
  - Invoke the VBA editor: `Developer->Visual Basic`
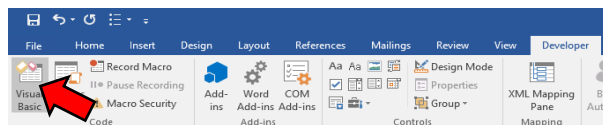
  - If successful you should see something similar to the image
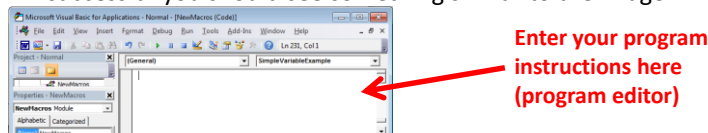
Enter your program instructions here (program editor)

First VBA Example (2)

- Type in or cut-and-paste the following example into the *VBA editor* (see last image for location of the editor, previous slide)
  - 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

OK
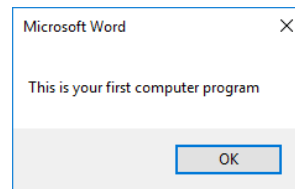
Reminder: Running Macros

- (You must first have the 'developer' tab visible).
- Developer->Macros
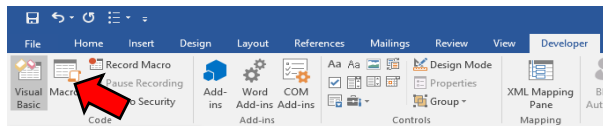
- The single macro should be highlighted, then click 'run'
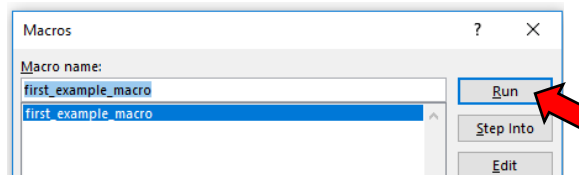
Running VBA Programs You Have Entered (2)

- Or you can run the program right after you have entered it (in the editor).
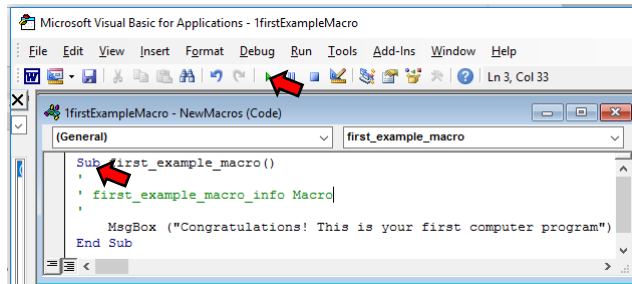
1. Ensure correct program "sub" is to be executed (click there)

2. Press the 'play/run' button or "F5"

Structure Of VBA Programs: Reminder Of Important Points

- As you just saw a program must begin with the "sub" keyword followed by the name of the "subroutine" (sub-part of the program).
- It also ends with end "end sub"
- Important style requirement: The part between the 'sub' and 'end sub' must be indented by 4 spaces (8 spaces if sub-indenting is used – next set of notes).

```
sub first_example_macro()


    MsgBox("Congrats!")


end Sub
```

Saving Your Macro

You can save your macro into a Word document:

1. Create a new Word document
2. Save the document as a macro-enable document (Word document that has a macro computer program embedded within it).
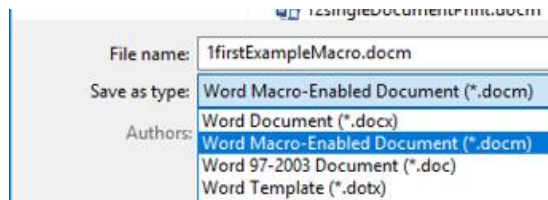
Saving Your Macro (2)

3. Next you have tell Word that you want to save your macro program inside this macro-enabled document.

 – By default when you save your macros Word will select "Normal.dotm" (All documents) as the location.

  • DO NOT save your macro in this document:
   – You will have trouble transferring your macro to other computers
   – Because "Normal.dotm" is the default template used to create some Word documents it may result in security warnings.
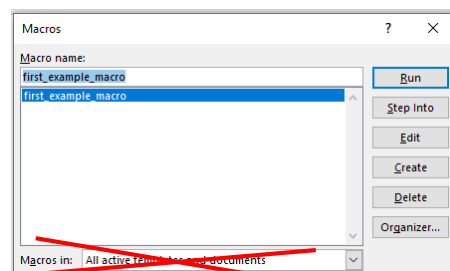
Saving Your Macro (3)

– Instead: Save your macro in your current document (in the example below it's "Assignment4.docm".

 • Transferring this document will allow the macros to be transferred as well.
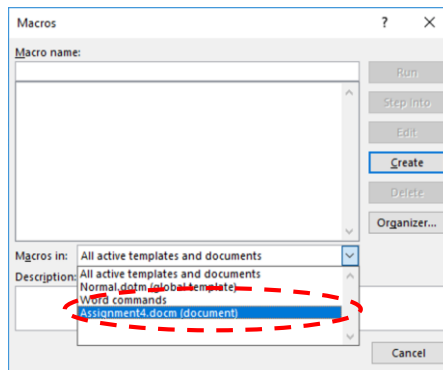
Viewing Macros

• All macros that you have created can be viewed in the VBA macro editor:

 – Macros manually entered in the editor (Message Box example)

 – Macros automatically recorded (not covered in lecture but in tutorial)

Which Document Contains A Macro?

a	11/1/2015 9:34 PM	Microsoft ...	0 KB
b	11/1/2015 9:35 PM	Microsoft ...	12 KB
c	11/1/2015 9:34 PM	Microsoft ...	0 KB
d	11/1/2015 9:36 PM	Microsoft ...	12 KB
e	11/1/2015 9:39 PM	Microsoft ...	12 KB

Question: What Is The Security Difference?

- Opening the following documents:
  - Document.docm
  - Document.docx
  - Document.doc

Types Of Documents That Can Contain Macros

- You store the macros that you write for this class this way
  - In a single document 'doc-m' document
- You can also store macros in these documents (not for this class but important to be aware in terms of computer security).
  - Normal 'dot-m' template
    - Default template used to produce all Word documents (formatting, layout etc.)
  - Custom 'dot-m' template
    - You can override the default by creating your own template documents

Viewing File Information: Learning What Type Of File Is That Word Document

- View details: select 'view' in a folder
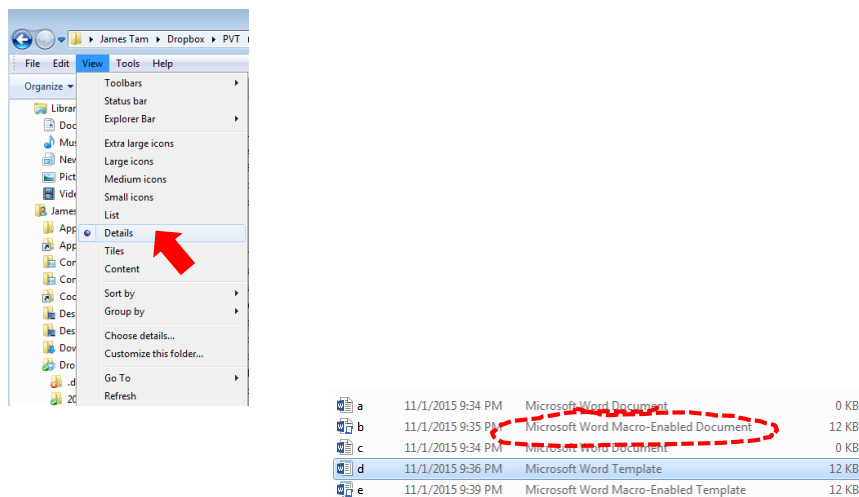
Viewing File Suffixes

- In a folder select: `Tools->Folder options`
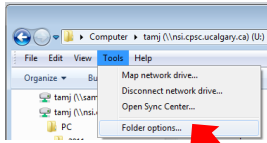
- Under the 'view' tab uncheck 'Hide extensions for known file types'
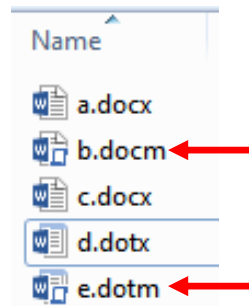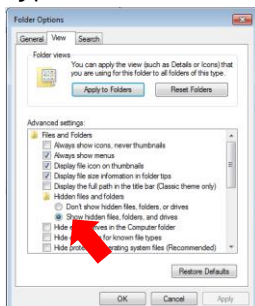
Reminder About Macro Security

- A macro is a computer program that is attached to another program's documents (e.g., MS-Word documents)
- It can supplement the program's features by automating repetitive tasks
- But like another computer program the instructions can either be useful or malicious
- Do not forget the security setting are specified in the MS-Office "Trust
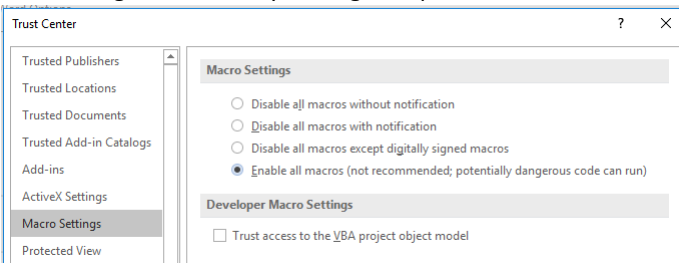
.DOCX (And .XLSX, .PPTX)

- As mentioned these types of files cannot have macros attached to them.
  - Reduced capabilities (no macros) but increased security (no macros)
- Question: Are these files with these extensions 100% safe?

Trust me - I'm safe.docx — File name extensions hidden

Trust me - I'm safe.docx.doc — Enabling the display of file name extensions

Transferring Your Macros (This Class)

- If you create a macro-enabled MS-Word document (file name suffix ".docm") then transfer the Word document itself.

File name:	Documents with local only Macros
Save as type:	Word Document (*.docx)
Authors:	Word Document (*.docx)
	Word Macro-Enabled Document (*.docm)
	Word 97-2003 Document (*.doc)
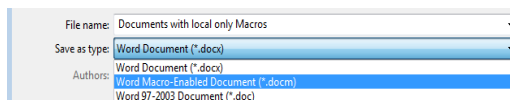
- Save the macro enabled word document to your portable USB flash drive
- OR
- Save the template document on your web disk drive (or any other 'cloud' storage system such as Dropbox.com)
  - To download files stored on web disk onto another computer: https://webdisk.ucalgary.ca/

Message Box

Microsoft Word ✕

Congratulations! This is your first computer program

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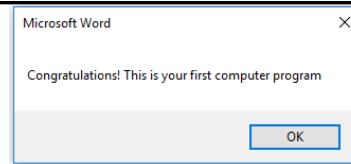
- (Details of the previous example)
- Creates a popup window to output information from your program
- Useful for testing
  - Is my program working?
  - Which part is running?
- Also useful for displaying status messages about the current state of the program

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.
  - Enter the function name and then a space

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

Function arguments

(Bold): mandatory arguments

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
    Age = InputBox("What is
    DogAge = Age * 7
    MsgBox (Name & " your
End Sub

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

Microsoft Visual Basic for Applications

⚠ Compile error:

Syntax error

[OK]   [Help]
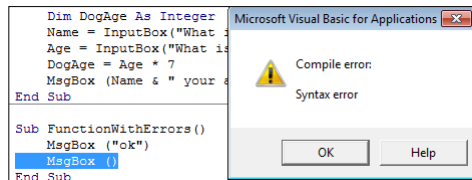
Required argument missing

Part of program that contains errors (yellow highlight)

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

Specific statement/instruction causing the error (red font)

Basic Mathematical Operators

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

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`

CPSC mail

TAM
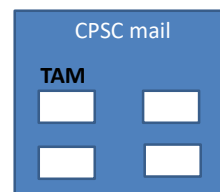
Common Types Of Variables

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

Examples Of Assigning Values To Variables

CPSC mail

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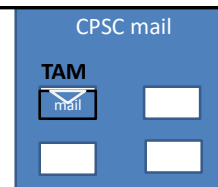
Note: some types of variables requires some mechanism to specify the type of information to be stored:

- 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"
```

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)
```

Variables: Metaphor To Use

- Think of VBA variables like a "mailslot in memory"
- Unlike an actual mailslot 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


```
num = 1
```

```
num = 17
```

num

17

3/3/2017

Common Mistake #2

- Assigning values from one variable does not 'link' them

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

Variables: Metaphor To Use (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!

```
Dim num1 as Long
Dim num2 as Long
num1 = 1
num2 = num1
num1 = 2
```

- What is the result?

MsgBox: Dislaying Mixes Of Strings And Variables

- **Format**:

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

- Name of the online example: `2variablesMixedOutput.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		Student 1: 4.3
3.3		Student 2: 3.3
3.7		Student 3: 3.7
2.0	Vs.	Student 4: 2.0
4.0		Student 5: 4.0
4.0		Student 6: 4.0
3.9		Student 7: 3.9
1.0		Student 8: 1.0
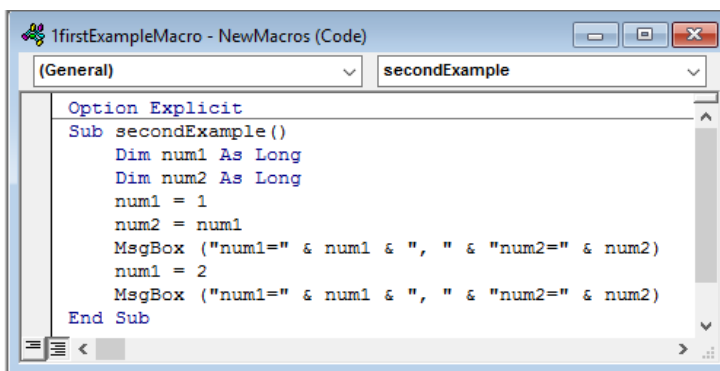
Student Exercise 1: Mixed Output

- What is the output of the following MsgBox:

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

Option Explicit

- It's not mandatory to include for your program "to work".
- But including it at the very start of your program before the 'sub' and subroutine name will save you many headaches!
- More details will be provided in tutorial.

```
1firstExampleMacro - NewMacros (Code)
(General)                    secondExample
Option Explicit
Sub secondExample()
    Dim num1 As Long
    Dim num2 As Long
    num1 = 1
    num2 = num1
    MsgBox ("num1=" & num1 & ", " & "num2=" & num2)
    num1 = 2
    MsgBox ("num1=" & num1 & ", " & "num2=" & num2)
End Sub
```

Third VBA Example

- Learning Objectives:
  - Using variables
  - Using mathematical operators

Third VBA Example (2)

Word document containing the macro: 3types.docm

```
Sub thirdExample()
    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
```

Microsoft Word
0.333333333333333

Microsoft Word
0
OK

Microsoft Word
1
OK
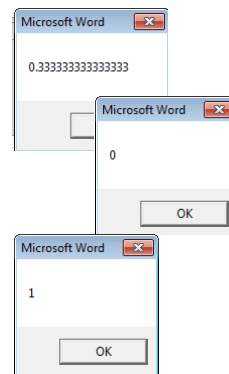
JT's note: Anything over 0.5 is rounded up

Student Exercise #2

- What would appear as output (MsgBox) when the following VBA macro program was executed

```
Sub exercise2()
    Dim num1 As Long
    num1 = 1
    MsgBox (num1)
    num1 = num1 + 1
    MsgBox (num1)
End Sub
```

Variable Naming Conventions

- Language requirements:
    - 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).

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

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.)

#4
firstName, last_name
(yes to either approach)

5. Avoid using keywords as names (next slide)

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.

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")
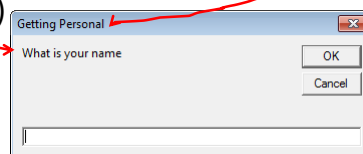
Example: `InputBox`

- Learning: getting user input with an `InputBox`
- **Word document containing the macro:** `4inputBox.docm`

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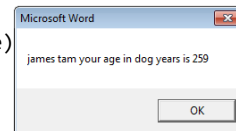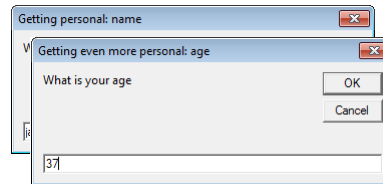
Note: there are two input boxes, one that prompts for the name and the other for the age. Each is given a *self-descriptive name* to distinguish them (an example of good programming style – more on this shortly)

The VBA Debugger

- 'Bug':
  - An error in the logic of your program.
  - The program "doesn't do what it is supposed to do"
  - Example: an erroneous formula for calculating an area of a rectangle
  - `area = length + width`
  - Bugs will seldom be this obvious
- Debuggers can be used to help find errors in your program
- More information on using the VBA debugger will be provided in tutorial

September 9, 1947
First Instance of Actual Computer Bug Being Found

At 3:45 p.m., Grace Murray Hopper records the first computer bug in her log book as she worked on the Harvard Mark II. The problem was traced to a moth stuck between a relay in the machine, which Hopper duly taped into the Mark II's log book with the explanation: "First actual case of bug being found."

Log entry describing "first computer bug"

Screenshot: www.computerhistory.org

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
  - List the program features (from the assignment description) and clearly indicate if the feature was completed or not completed.
- 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 "James Tam"

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 translated into machine language/binary
- 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
  - Before 'Option explicit' if included


```
' 
' documentationExample Macro
' Author:   James Tam
' ID: 123456
' Tut: 888
Option Explicit

Sub documentationExample()
```

Documentation: marked in red

Program Documentation (4)

- Program features (this will be worth many marks)
- Example assignment description

1. Word search and replace: The user is prompted for the word to search for and the word to replace it with. The prompts will occur via the program will then search for and replace all instances. If the search word does not exist in the document then nothing will happen. (+0.2

2. Style search and replace: The program will search and replace for all instances of the style "Heading 1" with "Normal". (**+0.2 GPA**)

3. Count the number of spelling mistakes in the document. Display the number with a MsgBox. (**+0.2 GPA**)

4. Count the number of occurrences of a word in the document. The word will be entered by the user as the program runs via an InputBox awarded if the program counts the number of occurrences of the same word each time that it runs. (**+0.2 GPA**)

5. Evaluate the quality of the document. (**Max of 1.5 GPA**). In order to get credit for this feature, Features #3 & #4 must be correctly imple the document status (REJECTED, NEUTRAL, SUPERIOR) must appear at the top of the document or you will be awarded no credit for

   - (a) If there are any spelling mistakes in document then the document is automatically rejected (text = "REJECTED") (*+0.3 GPA*).

- Program documentation
  - ' Author: James Tam   ID: 123456
  - ' Version: Nov 2, 2015
  - ' Tutorial: 99
  - ' PROGRAM FEATURES
  - ' Word search and replace: completed
  - ' Style search and replace: not completed
  - etc

Recap: Programs You've Seen So Far Produces Sequential Execution

- Each instruction executes from beginning to end, one after the other

```
Sub TaxCalculator()                                    ← Start
    Const TAX_RATE = 0.25
    Dim GrossIncome As Double
    Dim Tax As Double
    Dim NetIncome As Double
    GrossIncome = InputBox("Enter your income: ")
    Tax = GrossIncome * TAX_RATE
    NetIncome = GrossIncome - Tax
    MsgBox ("Gross Income $" & GrossIncome & ", Net Income $" & NetIncome)
End Sub                                                 ← End
```
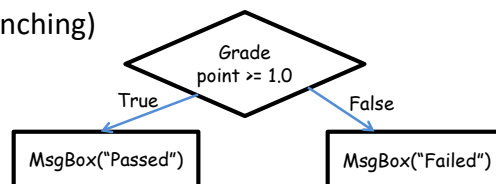
- When the last instruction is reached then the program ends

New Program Writing Concepts (Non-Sequential)

- Making decisions (branching)

- Looping (repetition)
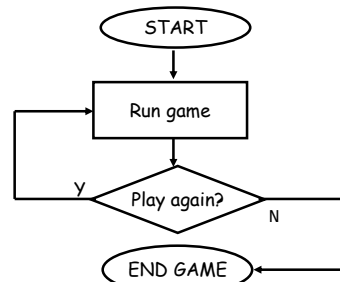
New Terminology

- **What you know; Boolean expression**: An expression that must work out (evaluate to) to either a true or false value.
  - e.g., it is over 45 Celsius today
  - e.g., the user correctly entered the password
- **New term, body**: A block of program instructions that will execute under a specified condition (for branches the body executes when a Boolean is true)

```
Sub Document_Open()
    MsgBox ("Fake virus!")
End Sub
```

This/these instruction/instructions run when you tell VBA to run the macro, the 'body' of the macro program

  - Style requirement
    - The 'body' is indented (4 spaces)
    - A "sub-body" (IF-branch) is indented by an additional 4 spaces (8 or more spaces)

Branching: Making Decisions In A Program

- Similar to the Excel (IF-Function): Check if some condition has been met (e.g., password for the document correctly entered): Boolean expression
- But the IF-Construct employed with programming languages is not just a function that returns a value for the true or false cases.
  - For each IF: a statement or a collection of statements can be executed (this is referred to as "the body" of the if or else case.
  - Example: entering a password
    - Boolean expression true, password matches:
      - True case body: display confirmation message and run program
    - Boolean expression false, password doesn't match:
      - False case body: display error message

Branching/Decision Making Mechanisms

- `If-Then`
- `If-Then, Else` **Similar to Excel if-then**
- `If-Then, ElseIf, Else` **Similar to Excel nested ifs**

Allowable **Operators** For Boolean Expressions

`if (value `**`operator`**` value) then`

VBA operator	Mathematical equivalent	Meaning	Example
<	<	Less than	5 < 3
>	>	Greater than	5 > 3
=	=	Equal to	5 = 3
<=	≤	Less than or equal to	5 <= 5
>=	≥	Greater than or equal to	5 >= 4
<>	≠	Not equal to	x <> 5

Decision Making With '**If-Then**'
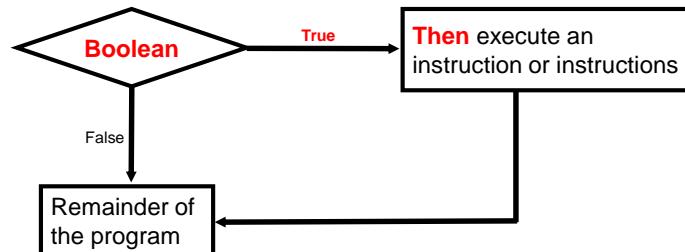
If-Then

- **Format**:
  ```
  If (Boolean expression) Then
      If-Body
  End if
  ```

- **Example**:
  ```
  If (totalWords < MIN_SIZE) Then
      MsgBox ("Document too short, total words " & totalWords)
  End If
  ```

If-Then: Complete Example

- **Word document containing the macro**: `5wordCount.docm`

```
' Try deleting all the words in the Word doc and run the
' macro again
Sub wordCount()
    Dim totalWords As Integer
    Const MIN_SIZE = 100
    totalWords = ActiveDocument.Words.Count
    If (totalWords < MIN_SIZE) Then
        MsgBox ("Document too short, total words " &
            totalWords)
    End If
End Sub
```

81% 7:34 PM 3/1/2017

Decision Making With An 'If, Else'

- Used when different Actions (separate bodies) are required for the true result (IF-case) vs. the false result (ELSE-case)

Decision Making With An 'If, Else'

If-Then (True), Else (False)

- **Format**:

```
If (Boolean expression) Then
    If-Body
Else
    Else-Body
End if
```

- **Example**:

```
If (totalWords < MIN_SIZE) Then
    MsgBox ("Document too short, total words " & totalWords)
Else
    MsgBox ("Document meets min. length requirements")
End If
```
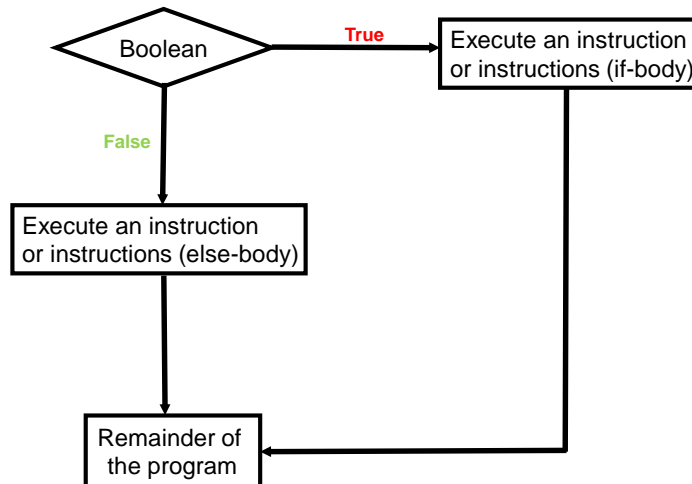
`If-Then, Else`: Complete Example

- **Word document containing the macro**: `6wordCountV2.docm`

```
' Try deleting words or changing the minimum size and observe
' the effect on the program.
Sub wordCountV2()
    Dim totalWords As Integer
    MIN_SIZE = 1000
    totalWords = ActiveDocument.Words.Count
    If (totalWords < MIN_SIZE) Then
        MsgBox ("Document too short, total words " &
            totalWords)
    Else
        MsgBox ("Document meets min. length requirements")
    End If
End Sub
```

Logic Can Be Used In Conjunction With Branching

- Typically the logical operators AND, OR are used with multiple conditions/Boolean expressions:
  - If multiple conditions *must all be met* before the body will execute. (And)
  - If *at least one condition* must be met before the body will execute. (Or)
- The logical NOT operator can be used to check if something has 'not' occurred yet
  - E.g., If it's true that the user *did not* enter the correct password then the program will end.

Logic: The "**OR**" Operator

- **Format:**

  ```
  If (Boolean expression) OR (Boolean expression) then
      body
  End if
  ```

- **Word document containing the macro (empty document, see macro editor for the important details)**: `7if_or_hiring.docm`

  ```
  gpa = InputBox("Grade point: ")
  experience = InputBox("Years of job experience: ")
  If (gpa > 3.7) Or (experience > 5) Then
      result = "Hire applicant"
  Else
      result = "Insufficient qualifications"
  ```

Hiring Example: Example Inputs & Results

```
If (gpa > 3.7) Or (experience > 5) then
```

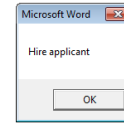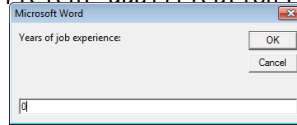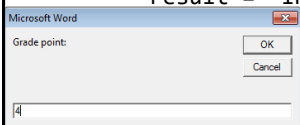| GPA	Years job experience	Result
2	0	Insufficient qualifications
1	10	Hire
4	1	Hire
4	7	Hire

Logic: The "**AND**" Operator

- **Format:**

  ```
  If (Boolean expression) And (Boolean expression) then
       body
  End if
  ```

- **Word document containing the macro (empty document, see macro editor for the important details)**: `8if_and_firing.docm`

  ```
  salary = InputBox("Salary: ")
  years = InputBox("Years of employment: ")
  If (salary >= 100000) And (years < 2) Then
       result = "Fired!"
  Else
       result = "Retained"
  ```

Firing Example: Example Inputs & Results

```
If (salary >= 100000) And (years < 2) Then
```

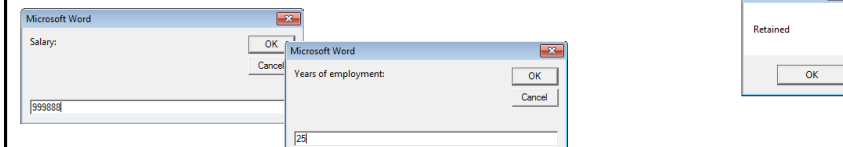| Salary	Years on job	Result
1	100	Retained
50000	1	Retained
123456	20	Retained
1000000	*0*	*Fired!*

Logic: The "NOT" Operator

- **Format**:

```
If Not (Boolean Expression) then
    body
End if
```

- **Word document containing the macro example**:

9checkSave.docm

```
If Not (ActiveDocument.Saved) Then
    MsgBox ("You haven't saved " & ActiveDocument.Name
      & " yet")
End If
```

Line Continuation Character

- To increase readability long statements can be split over multiple lines.

```
If (income  > 99999) And _
   (experience <=  2) And _
   (numRepramands > 0) Then
      MsgBox ("You're fired!")
End If
```

- To split the line the line continuation character (underscore) must be preceded by a space.

- Keywords cannot be split between lines e.g.

```
Msg _
Box
```

For more details see: http://support.microsoft.com/kb/141513

Line Continuation Character (2)

- Strings split over multiple lines require a combination of the proper use of the **line continuation character** '**_**' and the **concatenation operator** '**&**':

```
MsgBox ("Your " _
        & "name")
```

What To Do When Multiple Conditions Must Be Checked

- **Case 1**: If each condition is independent of other questions
  - Multiple if-then expressions can be used
  - Example:
  - Q1: Are you an adult?
  - Q2: Are you a Canadian citizen?
  - Q3: Are you currently employed?

What To Do When Multiple Conditions Must Be Checked (2)

- **Case 2 (mutually exclusive)**: If the result of one condition affects other conditions (when one condition is true then the other conditions cannot be true)
  - `If-then`, `elseif`, `else` should be used
  - Which of the following is your place of birth? (Answering true to one option makes the options false)
    - a) Calgary
    - b) Edmonton
    - c) Lethbridge
    - d) Red Deer
    - e) None of the above

Decision Making With **Multiple If-Then's**
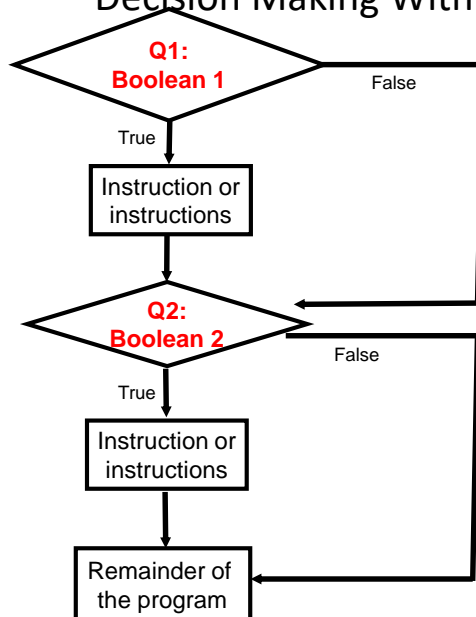
Each question is independent (previous answers have no effect on later questions because all questions will be asked).
Q1: Are you an adult?
Q2: Are you a Canadian citizen?
Q3: Are you currently employed?

Multiple If-Then's

- Any, all or none of the conditions may be true
- Employ when a series of independent questions will be asked
- **Format:**

```
if (Boolean expression 1) then
      body 1
end if
if (Boolean expression 2) then
      body 2
end if
   ...
statements after the conditions
```

Multiple If-Then's (2)

- **Word document containing the macro:**
  10multipleIfs.docm

```
Sub multipleIf()
 ' Check if there were any 'comments' added to the document.
    If (ActiveDocument.Comments.Count > 0) Then
        MsgBox ("Annotations were made in this document")
    End If
 ' A numbered item includes numbered and bulleted lists.
    If (ActiveDocument.CountNumberedItems() > 0) Then
        MsgBox ("Bullet points or numbered lists used")
    End If
```

- Bull1
- Bull2

Some text in a document.

Comment [JT1]: Replace 'text' with another word

Multiple If's: Mutually Exclusive Conditions

- At most *only one* of many conditions can be true
- Can be implemented through multiple if's

Inefficient combination!

- **Word document containing the macro**: "11gradesInefficient.docm"

```
If (letter = "A") Then
    grade = 4
End If
If (letter = "B") Then
    grade = 3
End If
If (letter = "C") Then
    grade = 2
End If
```

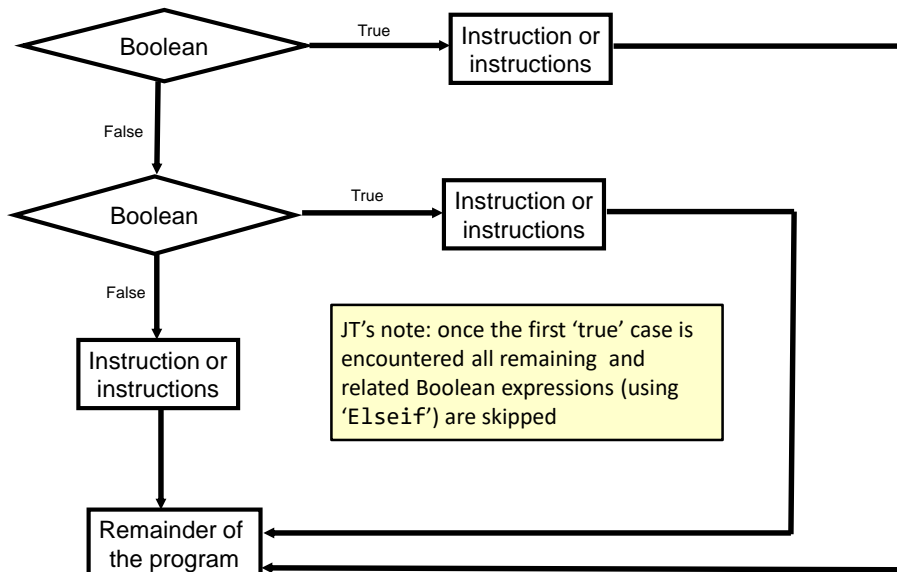```
If (letter = "D") Then
    grade = 1
End If
If (letter = "F") Then
    grade = 0
End If
```

---

# Decision Making With If-Then, Elseif, Else



JT's note: once the first 'true' case is encountered all remaining and related Boolean expressions (using 'Elseif') are skipped

## Multiple **If-Elif-Else**: Use With Mutually Exclusive Conditions

- **Format:**

```
if (Boolean expression 1) then:
    body 1
elseif (Boolean expression 2) then
    body 2
      ...
else
    body n
' Only one 'end-if' at very end
end if
statements after the conditions
```

Mutually exclusive
- One condition evaluating to true excludes other conditions from being true
- Example: having your current location as 'Calgary' excludes the possibility of the current location as 'Edmonton', 'Toronto', 'Medicine Hat'

If-Elseif-Else: Mutually Exclusive Conditions (Example)

- **Word document containing the macro (empty document, see macro editor for the important details)**: "12gradesEfficient.docm"

```
If (letter = "A") Then
    grade = 4
ElseIf (letter = "B") Then
    grade = 3
ElseIf (letter = "C") Then
    grade = 2
ElseIf (letter = "D") Then
    grade = 1
ElseIf (letter = "F") Then
    grade = 0
Else
    grade = -1 'A signal that letter was invalid
End If
```

This approach is more efficient when at most only one condition can be true.

Extra benefit:

The body of the else executes only when all the Boolean expressions are false. (Useful for error checking/handling)

Location Of The "End If": Multiple If's

- Independent `If-then`'s:
  - Since each 'if' is independent each body must be followed by it's own separate 'end if'

```
letter = InputBox("Enter letter grade: ")
If (letter = "A") Then
    grade = 4
End If
If (letter = "B") Then
    grade = 3
End If
If (letter = "C") Then
    grade = 2
End If
If (letter = "D") Then
    grade = 1
End If
If (letter = "F") Then
    grade = 0
End If
```

Location Of The "End If": If-then, Else

- `If-then, Else`:
  - Since the 'if-then' and the 'else' are dependent (either one body or the other must execute) the 'end if' must follow the body of the 'else-body' (last dependent "if-branch")

```
If (totalWords < MIN_SIZE) Then
    MsgBox ("Document too short, total wo
Else
    MsgBox ("Document meets min. length r
End If
```

Document either does or does not have enough words (one option IF or the other option ELSE must be applied)

Location Of The "End If": If-Then, ElseIf

- Dependent `If-then`, `Else-If`:
  - Since the results of earlier Boolean expressions determine whether later ones can be true (reminder: because at most only one can be true) all of the `if-then` and `Elseif` expressions are dependent (one related block).
  - The "`end if`" belongs at the very end of the block

```
If (letter = "A") Then
    grade = 4
ElseIf (letter = "B") Then
    grade = 3
ElseIf (letter = "C") Then
    grade = 2
ElseIf (letter = "D") Then
    grade = 1
ElseIf (letter = "F") Then
    grade = 0
Else
    grade = -1 'A signal that letter was invalid
End If
MsgBox ("Letter=" & letter & " " & "GPA=" & grade)
```

VBA: If, Else-If And Excel: Nested-Ifs

- These two concepts are comparable:

VBA:
```
If (letter = "A") Then
    grade = 4
ElseIf (letter = "B") Then
    grade = 3
ElseIf (letter = "C") Then
    grade = 2
ElseIf (letter = "D") Then
    grade = 1
ElseIf (letter = "F") Then
    grade = 0
Else
    grade = -1
End If
```

Excel (display different messages for different grade points e.g. Display "Perfect" if grade point is 4.0 or greater):
```
=IF(D2="A",4,
    IF(D2="B",3,
        IF(D2="C",2,
            IF(D2="D",1,
                IF(D2="F",0,
                    -1)))))
```
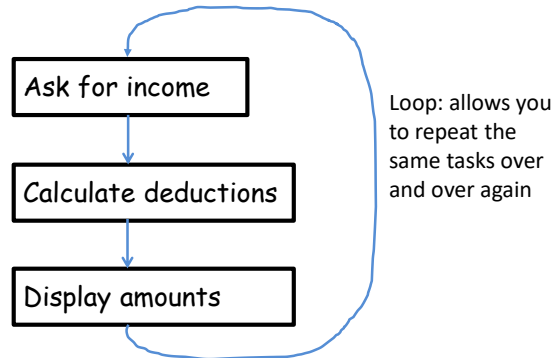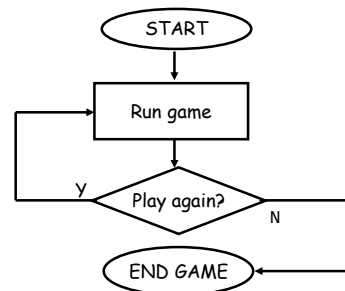
Looping/Repetition

- How to get the program or portions of the program to automatically re-run
  - Without duplicating the instructions
  - Example: you need to calculate tax for multiple people

Ask for income

Calculate deductions

Display amounts

Loop: allows you to repeat the same tasks over and over again

Looping/Repetition (2)

- The entire program repeats

Play game again?

www.colourbox.com
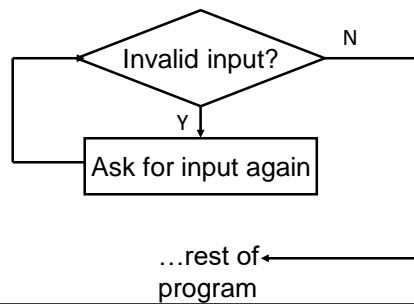
START

Run game

Y   Play again?   N

END GAME

Looping/Repetition (3)

- Only a specific part of the program repeats

```
Enter your age (must be non-negative):  -1
Enter your age (must be non-negative):  27
Enter your gender (m/f): █
```
Re-running specific parts of the program

Flowchart

Characteristics Of Do-While Loops

- Described as variable repetition loops: runs as long as some condition holds true (number of times that the loop repeats is variable)
  - e.g., while the user doesn't quit the program re-run the program
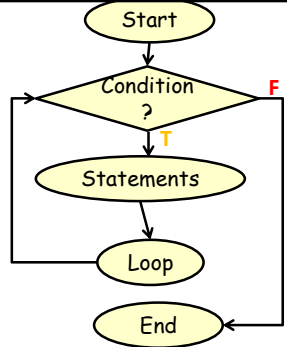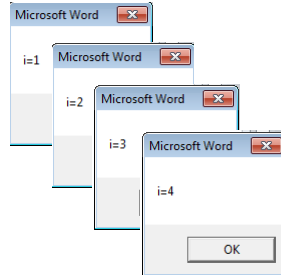 - e.g., while the user enters an erroneous value ask the user for input.

Do-While Loop

- **Format**:
```
Do While <Condition>
    <Statement(s)>
Loop
```

- **Example**: "13whileUpOne.docm"
```
Dim i As Integer
i = 1
Do While (i <= 4)
    MsgBox ("i=" & i)
    i = i + 1
Loop
```

Any valid mathematical expression here

Programming Style: Variable Names

- In general variable names should be self-descriptive e.g., 'age', 'height' etc.
- Loop control variables are an exception e.g., 'i' is an acceptable variable name
  - It's sometimes difficult to come up with a decent loop control name
  - Loop control variables are given shorter names so the line length of a loop isn't excessive
    ```
    Dim loopControl As Integer
    loopControl = 1
    Do While (loopControl <= 4)
        ...
    ```

Loops That Never Execute

- **Word document containing the complete program**:
  14nonExecutingLoops.docm

```
    Dim i As Integer

    i = 5
    Do While (i <= 4)
        MsgBox ("i=" & i)
        i = i + 1
    Loop

    i = 1
    j = -1
    Do While (i < 0) and (j < 0)
        MsgBox ("i=" & i & " " & "j=" & j)
        i = i + 1
    Loop
```

Exercise #3: Loops

- The following program that prompts for and displays the user's age.

```
Sub errorChecking()
   Dim age As Long
   age = InputBox("Age (greater than or equal to zero)")
End Sub
```

- Modifications:
  - As long as the user enters a negative age the program will continue prompting for age.
  - After a valid age has been entered then stop the prompts and display the age.
  - Hint: first determine which parts of the program should be repeated and which parts should not.

Common Mistake #3

- Mixing up branches (IF and variations) vs. loops (do-while)
- Related (both employ a Boolean expression) but they are not identical
- Branches
  - General principle: If the Boolean evaluates to true then execute a statement or statements (**once**)
  - Example: display a popup message if the number of typographical errors exceeds a cutoff.
- Loops
  - General principle: As long as (or while) the Boolean evaluates to true then execute a statement or statements (**multiple times**)
  - Example: While there are documents in a folder that the program hasn't printed then continue to open another document and print it.

Common Mistake #3 (2)

- Contrast
- **Word document containing the complete program**: 15branchVsLoop.docm

```
age = InputBox("Age (positive only)")
If (age <= 0) then
    age = InputBox("Age (positive only)")
End if
MsgBox(age)
```

                Vs.

```
age = InputBox("Age (positive only)")
Do While (age <= 0)
    Age = InputBox("Age (positive only)")
Loop
MsgBox(age)
```

After This Section You Should Know

- The history and background behind VBA
- How to copy and run the pre-created lecture examples
- 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 the VB editor identifies programming errors
- How to use basic mathematical operators in VB expressions
- How to create and use variables
- Naming conventions for variables
- What are commonly used variable 'types' in VB

After This Section You Should Know (2)

- 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)
- The security settings in the MS-Office "Trust Center"
- Different levels of security that exist for different types of MS-Word documents
- How to use branches to make decisions in VBA
  - If
  - If-else
  - Multiple If's
  - If, else-if, else
  - Using logic (AND, OR, NOT) in branches

After This Section You Should Now Know (3)

- How to use the line continuation character to break up long instructions
- How to get a program to repeat one or more instructions using `Do-while` loops

Copyright Notice

- Unless otherwise specfied, all images were produced by the author (James Tam).