

CGN 2420

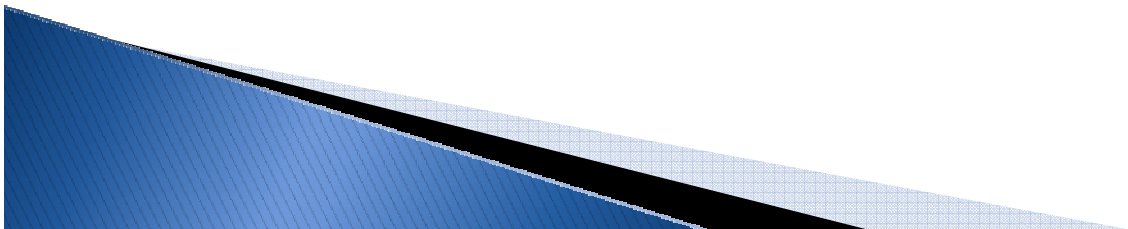
Macros and User–Written Functions for Excel

Instructor: Professor Cora Martinez, PhD
Department of Civil and Environmental Engineering
Florida International University



Objectives

- ▶ Know what an Excel macro is and how they can be created and used.
- ▶ Create recorded macros.
- ▶ Access Excel's Visual Basic programming Environment.
- ▶ Write your own functions.



Good to Know before Working with Macros

- ▶ Excel 2007 uses two different file extensions:
 - .xlsx–Excel file extension for macro–disabled workbooks (DEFAULT)
 - .xlsm–Excel file extension for macro–enabled workbooks

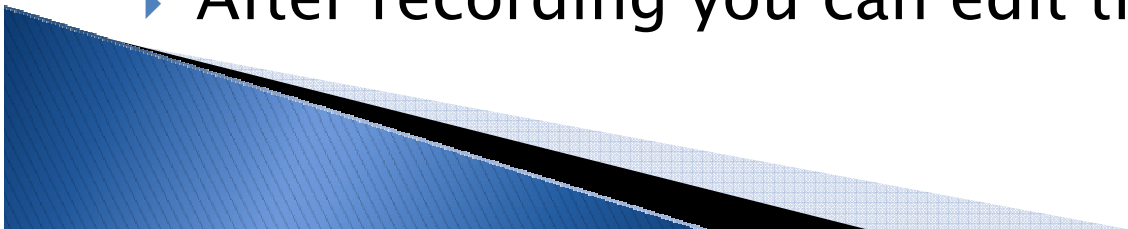
If you save the workbook as .xlsx you will loose all your macros!!!!

ALWAYS SAVE AS .xlsm



Recorded Macros

- ▶ *A macro* is a subprogram that does not receive any inputs (there are no arguments).
- ▶ Recording of keystrokes is the easiest way to create a macro.
- ▶ When you ask Excel to record a macro, it actually writes a program in VBA using Visual Basic statements that are equivalent to the commands you enter via the keyboard and/or mouse.
- ▶ After recording you can edit the program.



Recording a Macro

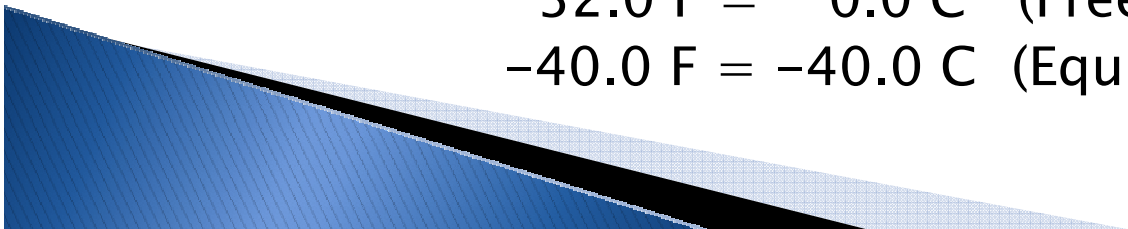
- ▶ Let's demonstrate the process of recording a macro with the following example:

Consider a macro that converts a temperature in degrees Fahrenheit to degrees Celsius:

$$T_C = \frac{T_F - 32}{1.8}$$

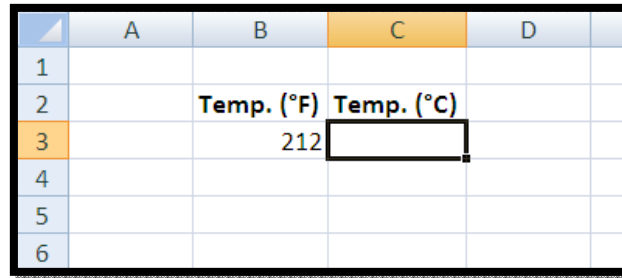
To make sure your macro is working properly use the following reference values:

212.0 F = 100.0 C (Boiling point of water)
98.6 F = 37.0 C (Human body temperature)
32.0 F = 0.0 C (Freezing point of water)
-40.0 F = -40.0 C (Equivalent temperature)



Recording a Macro (Cont.)

- ▶ Begin by entering a Fahrenheit temperature and selecting the cell that will contain the temperature in °C:



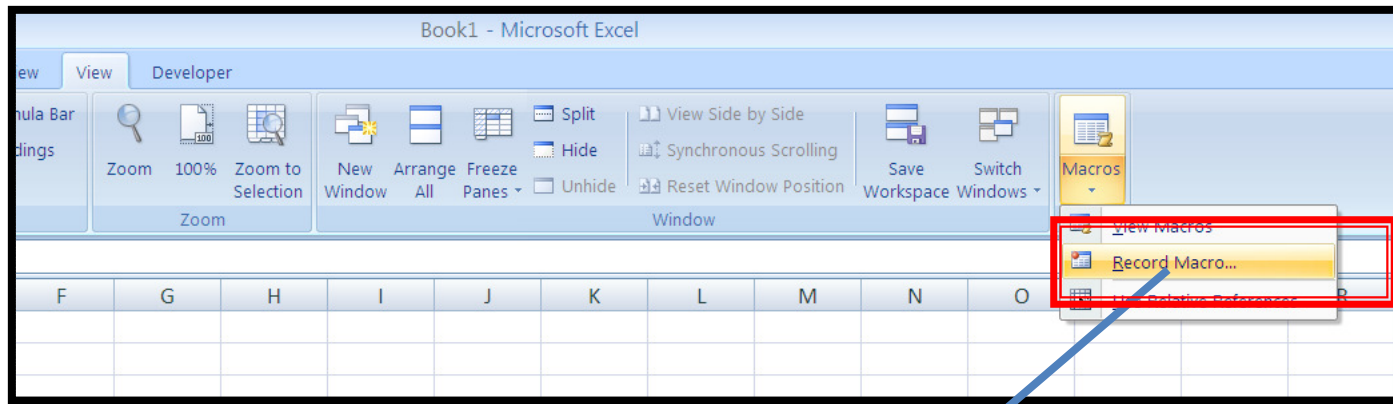
	A	B	C	D
1				
2		Temp. (°F)	Temp. (°C)	
3		212		
4				
5				
6				

- ▶ Tell Excel that you want to record a Macro using Ribbon options

View/Macros/Macros (menu)/Record macro

The macro dialog will be displayed

Recording a Macro (Cont.)



Record Macro [?] [X]

Macro name:
Macro1

Shortcut key:
Ctrl+

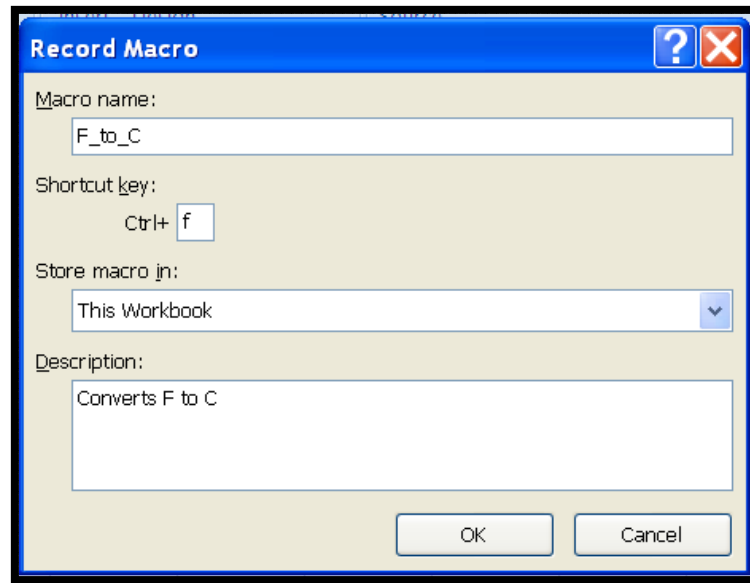
Store macro in:
This Workbook

Description:

OK Cancel

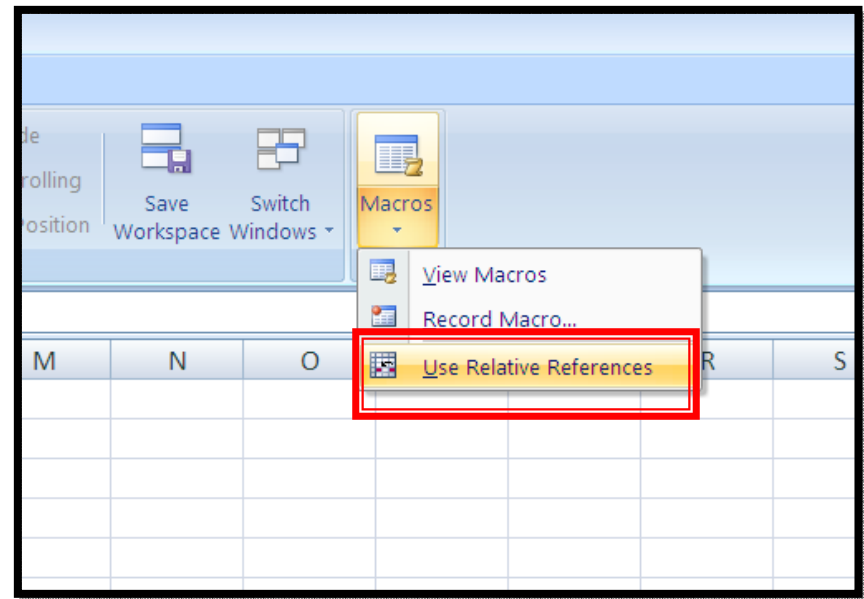
Recording a Macro (Cont.)

- ▶ Use the record dialog to set the macro name and shortcut key (optional). You could also include a brief description of the macro.



Recording a Macro (Cont.)

- ▶ Select relative references:
 - If you want the macro to use the value “in the cell to the left of the currently selected cell” use **relative** referencing.
 - If you want the macro to always use the value in cell B3, use **absolute** referencing.

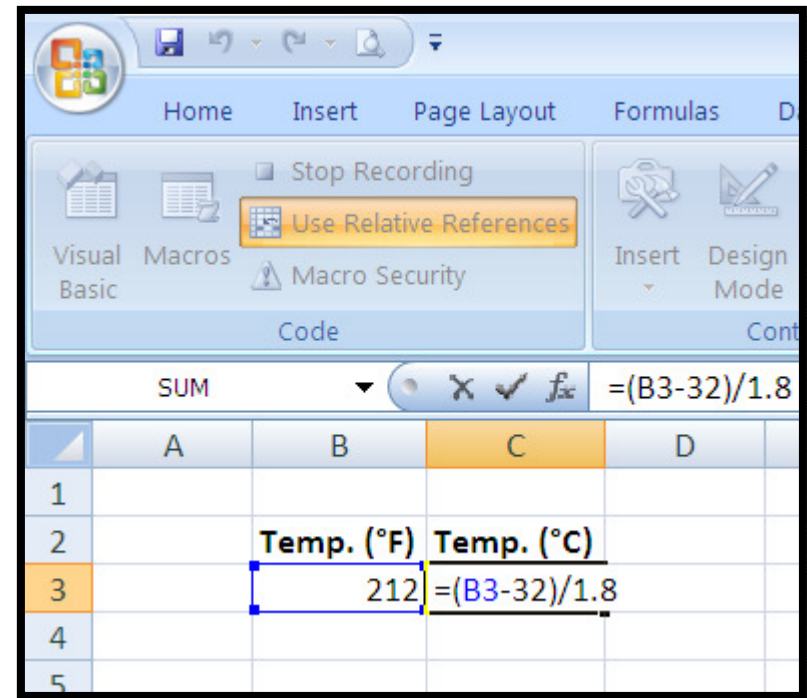


To select relative referencing:
View/Macros/Macros (menu),
toggle on [Use Relative References]

Recording a Macro (Cont.)

- ▶ Enter the conversion equation in cell C3 normally.

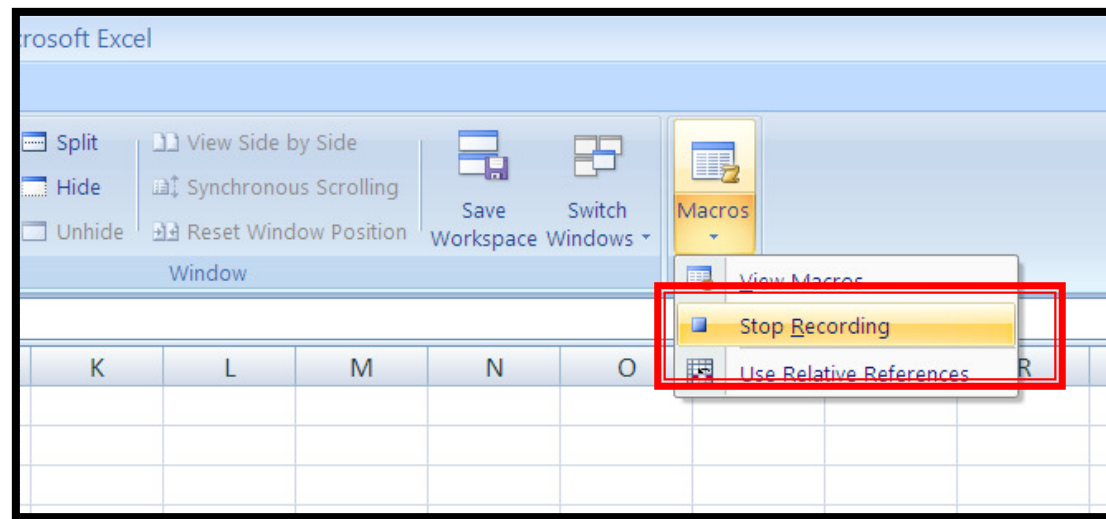
The macro recorder is running as the formula is entered.



Recording a Macro (Cont.)

- ▶ Stop the macro recorder by pressing the Stop Recording button on the Ribbon:

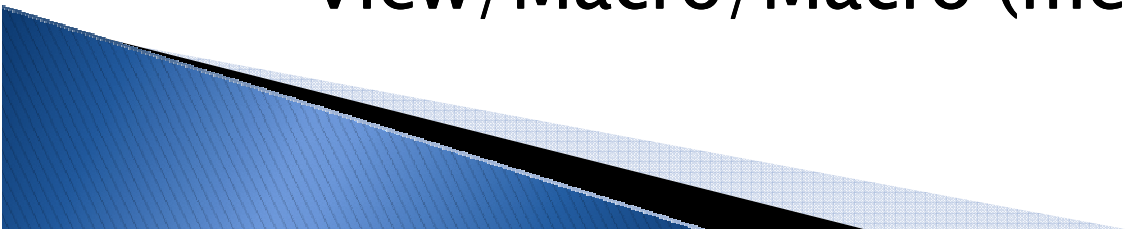
View/Macros/Macros (menu)/Stop Recording



Testing the Recorded Macro

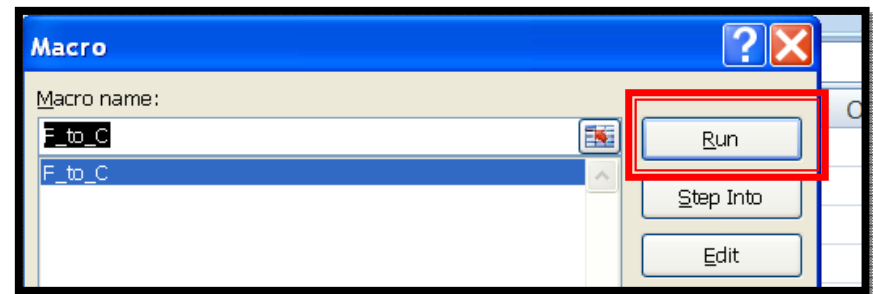
- ▶ Once the macro was recorded, let's use the benchmark temperatures given before to test the macro.
- ▶ Select the cell next to the temperature that you want to convert into °C.
- ▶ The list of currently available macros is displayed by using Ribbon options:

View/Macro/Macro (menu)/View Macros



Testing the Recorded Macro (Cont.)

- ▶ Run the F_to_C macro by selecting the macro name and pressing the [Run] button.



- ▶ You can also run the macro by using the assigned shortcut key [Ctrl-f].

	A	B	C	D	E
1					
2		Temp. (°F)	Temp. (°C)		
3		212	100		
4		98.6	37		
5		32	0		
6		-40			
7					
8					

Editing a Recorded Macro

- ▶ Recorded macros are stored as VBA (Visual Basic for Applications) subprograms or “Subs”. The F_to_C was stored like this:

```
Sub F_to_C()  
'  
' F_to_C Macro  
' Converts F to C  
'  
' Keyboard Shortcut: Ctrl+f  
'  
    ActiveCell.FormulaR1C1 = "=(RC[-1]-32)/1.8"  
    ActiveCell.Offset(1, 0).Range("C4").Select  
End Sub
```

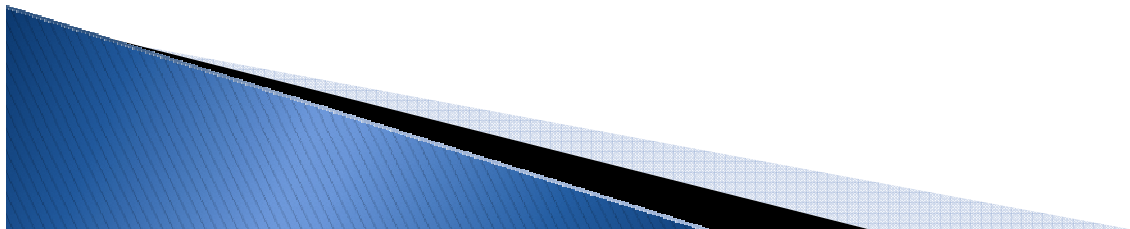
Editing a Recorded Macro (Cont.)

```
Sub F_to_C()  
'  
' F_to_C Macro  
' Converts F to C  
'  
' Keyboard Shortcut: Ctrl+f  
'
```

Comments are ignored when macro is run. They are included to provide information about the macro.

```
    ActiveCell.FormulaR1C1 = "=(RC[-1]-32)/1.8"  
    ActiveCell.Offset(1, 0).Range("C4").Select  
End Sub
```

Two operational lines

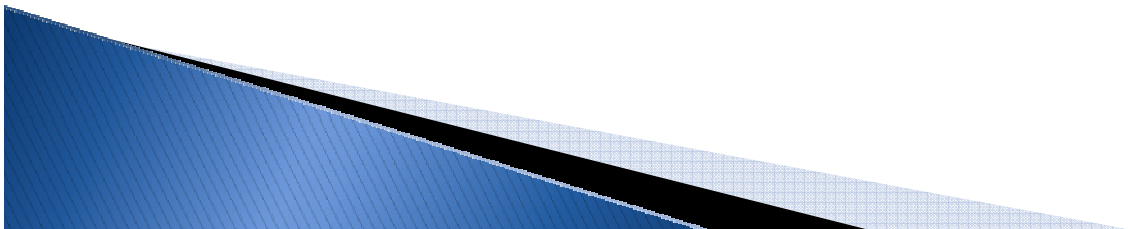


Editing a Recorded Macro (Cont.)

- ▶ ActiveCell.FormulaR1C1 = "(RC[-1]-32)/1.8"

This portion assigns (=) a formula in R1C1 notation (used for relative cell references) to the active cell.

The statement "RC[-1]" tells Excel to use the cell in the current row (R) and one cell to the left of the active cell (C[-1]).



Editing a Recorded Macro (Cont.)

- ▶ ActiveCell.Offset(1, 0).Range("C4").Select

This programming line moves the cursor down one row after entering the formula.

- *ActiveCell*

Starting from the current active cell location

- *Offset (1,0)*

Move one row down and zero columns right

- *Range ("C4")*

On the currently selected worksheet.

- *Select*

Makes the offset cell the selected active cell.



Editing a Recorded Macro (Cont.)

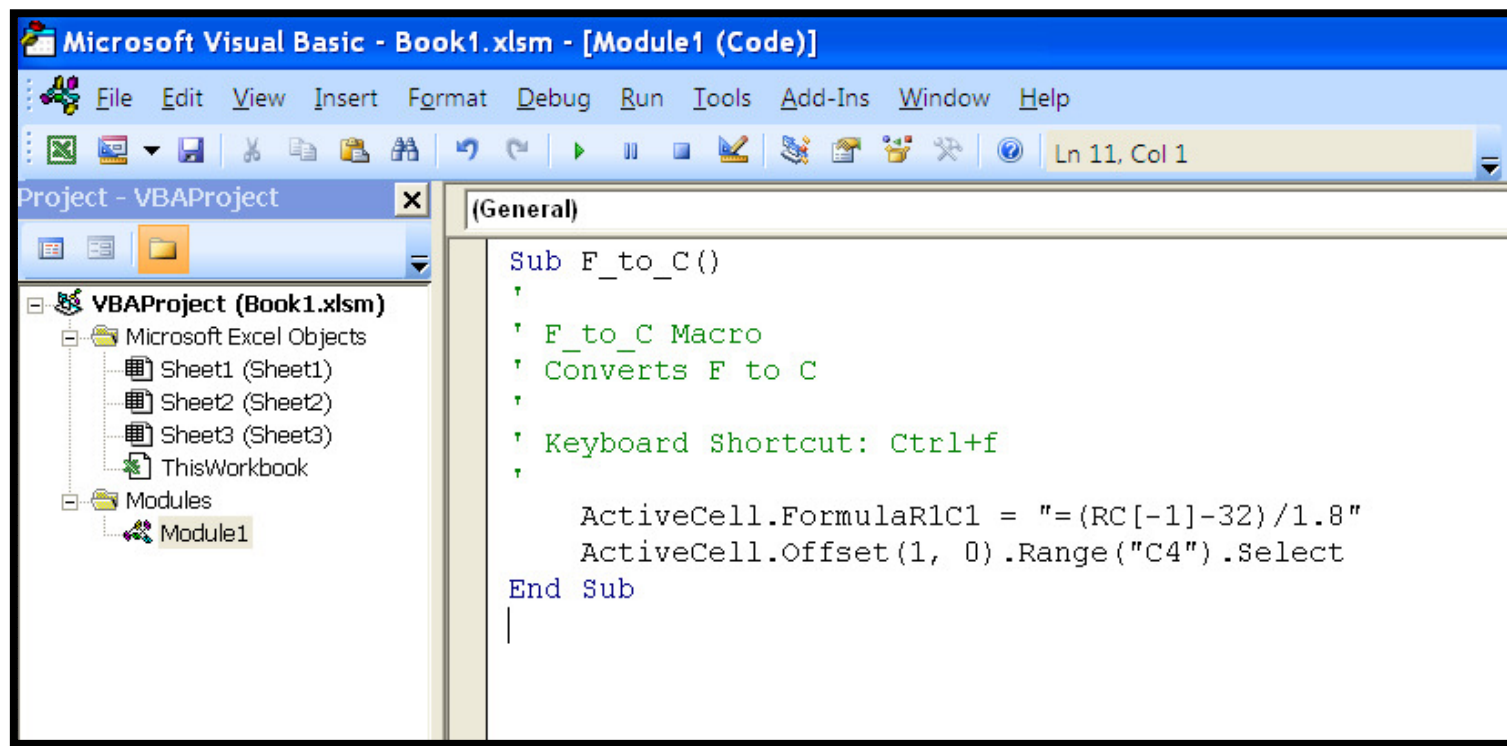
- ▶ The recorded macro can be edited as a VBA program by using the VBA editor.

- ▶ To open the editor, use Ribbon options:
 1. View/Macro/Macro (menu)
 2. Choose the macro you want to edit
 3. Click the [Edit] button to open the VBA editor

The program code for the selected macro will be displayed.



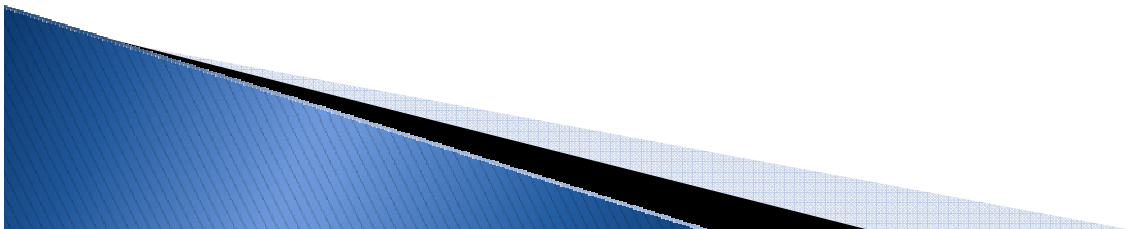
Editing a Recorded Macro (Cont.)



Editing a Recorded Macro (Cont.)

- ▶ You can modify the existing program or create entirely new macros. Let's illustrate this with the following example:

Create a new macro for converting temperatures in degrees Celsius to degrees Fahrenheit by copying the existing macro "F_to_C"



Editing a Recorded Macro (Cont.)

1. Select and copy the code of the recorded macro:

```
(General)
Sub C_to_F()
'
' C_to_F Macro
' Converts C to F
'
' Keyboard Shortcut: Ctrl+g
'
    ActiveCell.FormulaR1C1 = "=RC[-1]*1.8+32"
    ActiveCell.Offset(1, 0).Range("C4").Select
End Sub
Sub C_to_F()
'
' C_to_F Macro
' Converts C to F
'
' Keyboard Shortcut: Ctrl+g
'
    ActiveCell.FormulaR1C1 = "=RC[-1]*1.8+32"
    ActiveCell.Offset(1, 0).Range("C4").Select
End Sub
```

Editing a Recorded Macro (Cont.)

2. Modify the macro as needed:

```
(General)

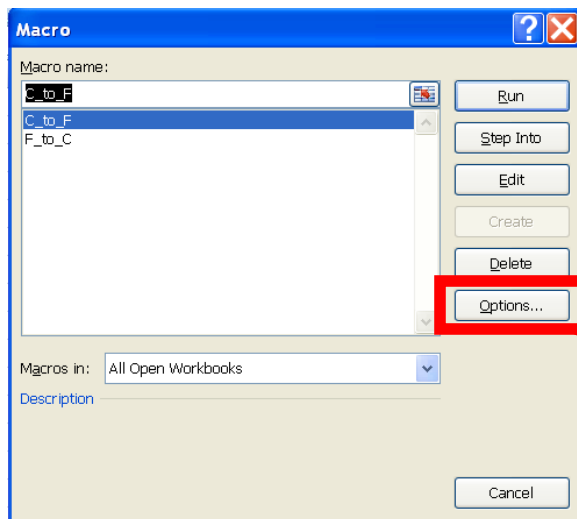
Sub F_to_C()
    '
    ' F_to_C Macro
    ' Converts F to C
    '
    ' Keyboard Shortcut: Ctrl+f
    '
    ActiveCell.FormulaR1C1 = "= (RC[-1]-32)/1.8"
    ActiveCell.Offset(1, 0).Range("C4").Select
End Sub

Sub C_to_F()
    '
    ' C_to_F Macro
    ' Converts C to F
    '
    ' Keyboard Shortcut: Ctrl+g
    '
    ActiveCell.FormulaR1C1 = "=RC[-1]*1.8+32"
    ActiveCell.Offset(1, 0).Range("C4").Select
End Sub
```

Editing a Recorded Macro (Cont.)

3. Save the changes.

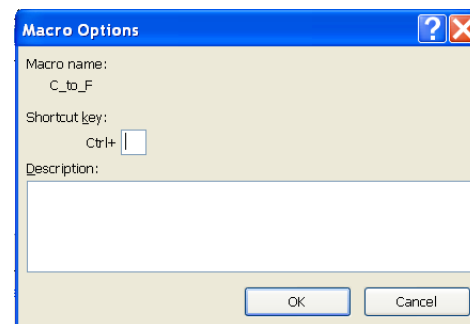
Creating a new sub in VBA makes the macro available to the worksheet.



Note:

The shortcut key and description are not assigned to the macro when it was created by using VBA.

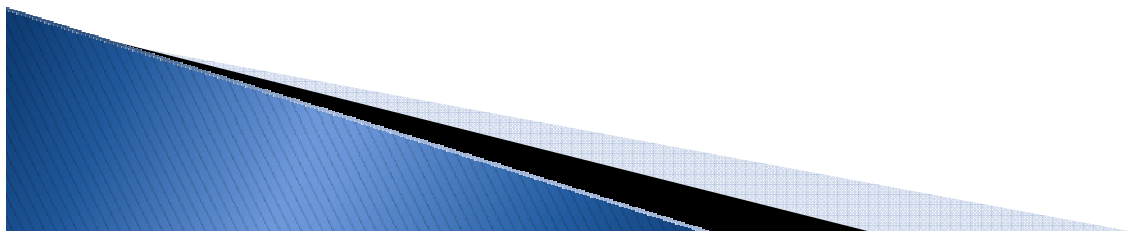
To assign the shortcut key and description, press the option button and entering the information.



Editing a Recorded Macro (Cont.)

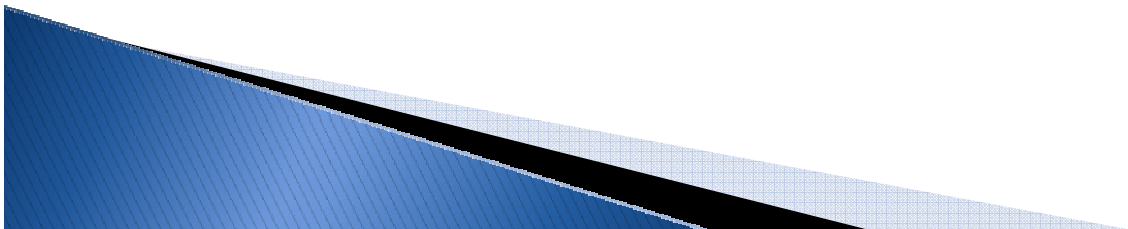
4. Finally, test the new macro in the worksheet

	A	B	C	D	E	F	G
1							
2		Using F_to_C			Using C_to_F		
3							
4		Temp. (°F)	Temp. (°C)		Temp. (°C)	Temp. (°F)	
5		212	100		100	212	
6		98.6	37		37	98.6	
7		32	0		0	32	
8		-40	-40		-40	-40	
9							
10							



User Written-Functions

- ▶ A Function is a simple piece of a program that receives input through arguments (or parameters), performs calculations, and returns a result.
- ▶ User-written functions can be written whenever needed. They can be tailored to meet very specific needs when built-in functions are not available.



User–Written Functions (Cont.)

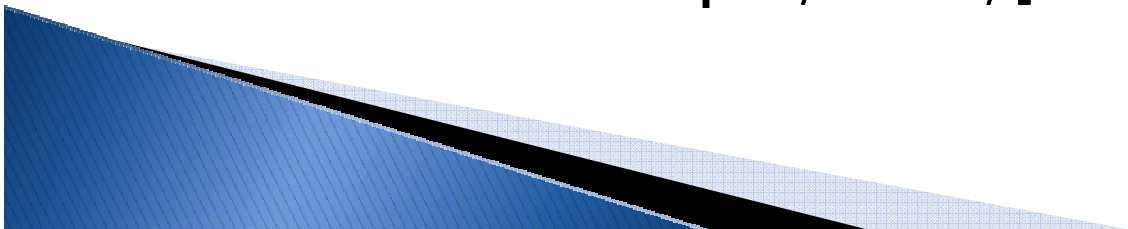
User–written functions in Excel are written in VBA (Visual Basic for Applications).

This a complete programming language and programming environment built–in into the Microsoft Office programs.

- ▶ **Starting VBA:**

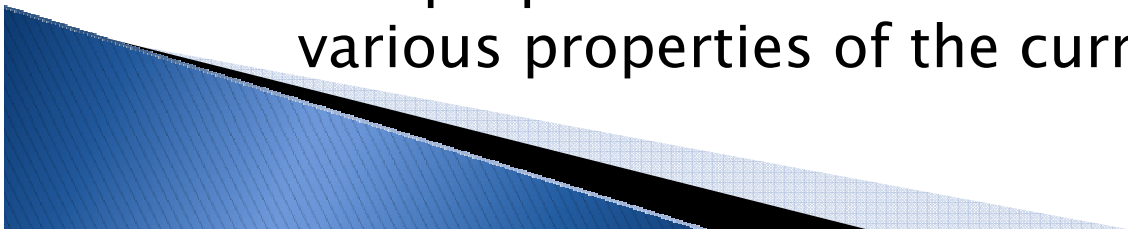
- The developer tab’s Code Group provides a [Visual Basic] button to access the Visual Basic editor.

Developer/Code/[Visual Basic]

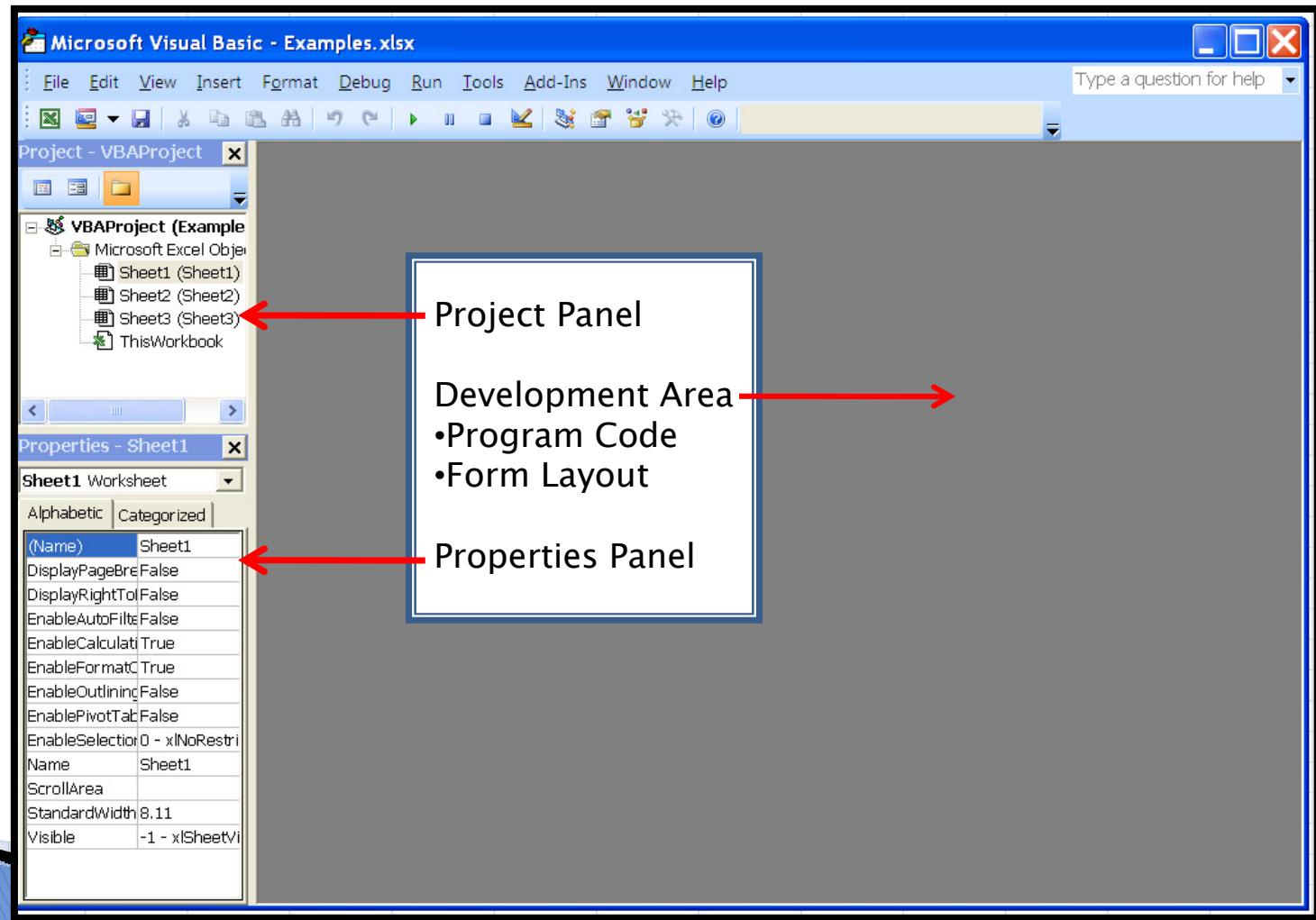


User Written-Functions (Cont.)

- ▶ The VBA Editor is a multipanel window:
 - The main area is called the *development area*. This area is used for writing program codes.
 - The *project panel* lists all of the items in the project.
 - A *project* contains all of the functions, forms and subprograms needed to make a program work. It also contains the worksheets in the workbook.
 - The properties Panel is used to access and modify the various properties of the currently presented object.



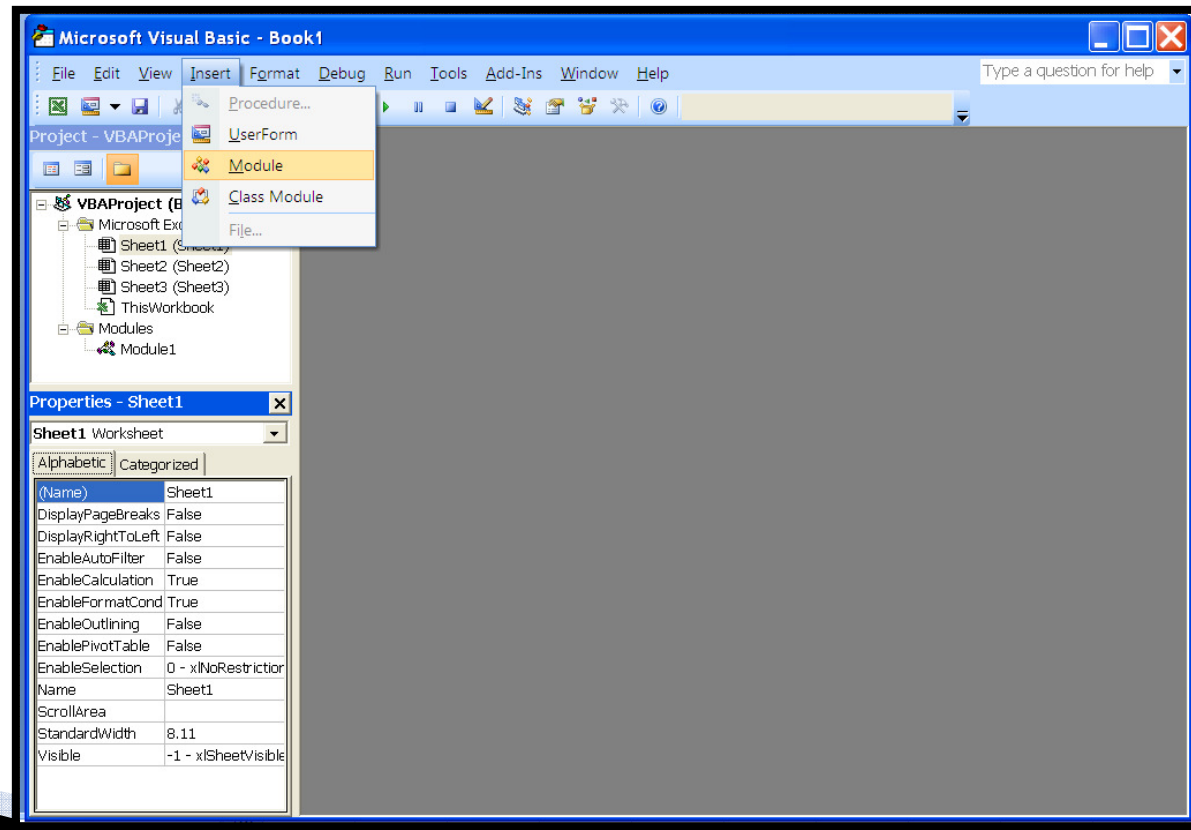
User Written-Functions (Cont.)



Writing a Function in VBA

Insert a module:

- ▶ To insert a module, select Insert/Module from the VBA menu.



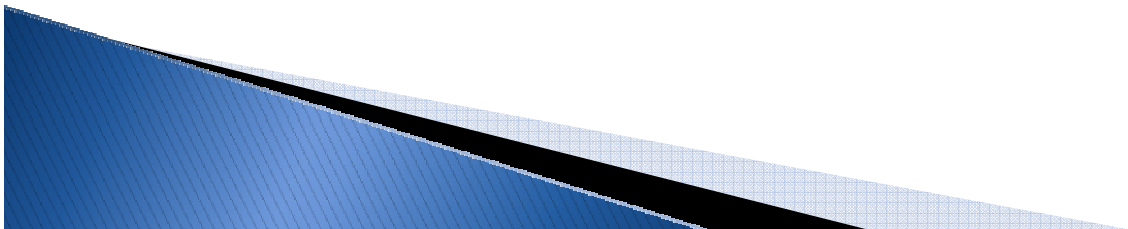
Writing a Function In VBA (Cont.)

Insert the Function Procedure into the module:

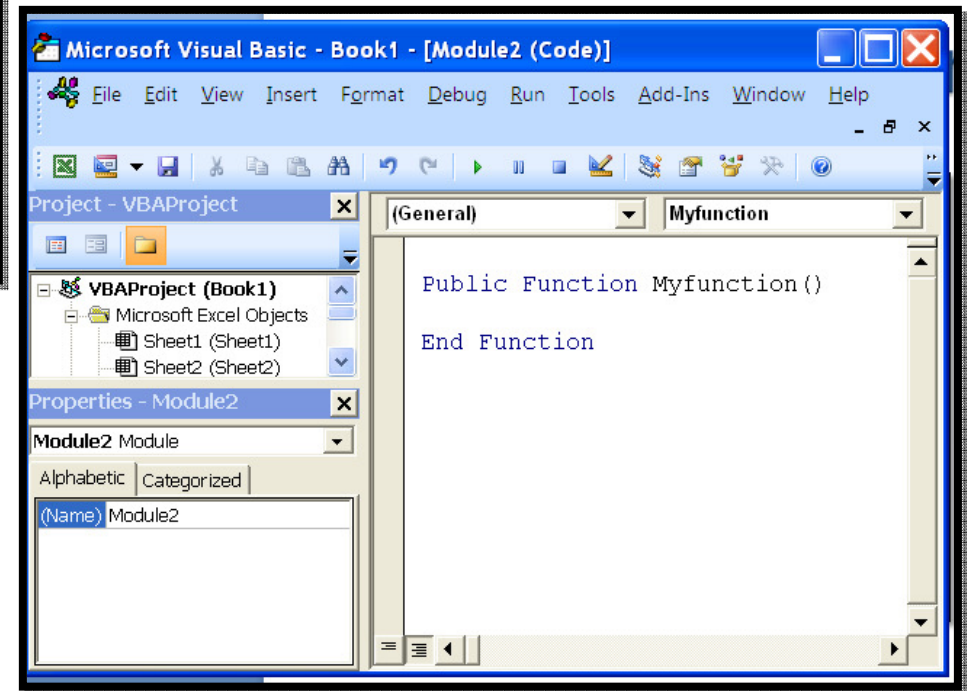
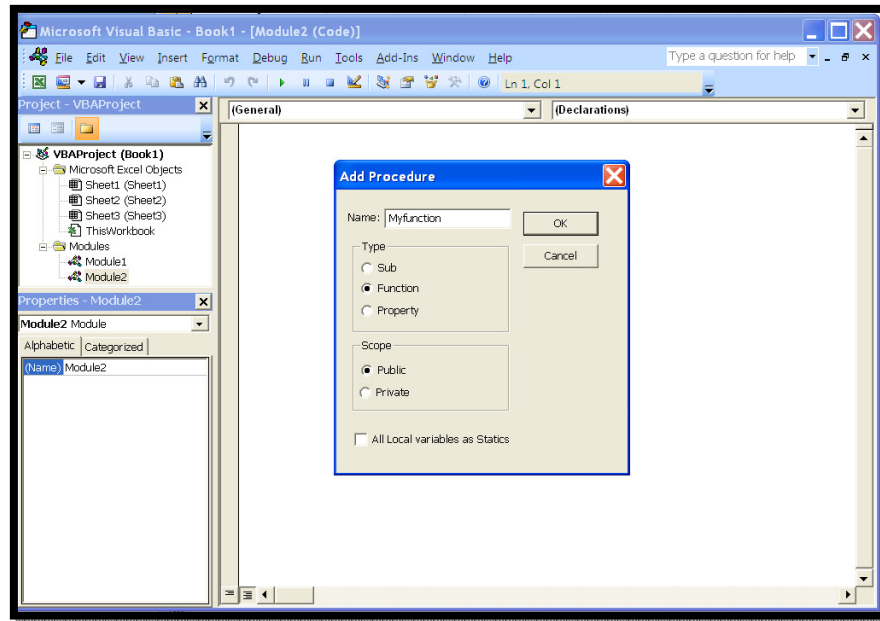
To have VBA create the first and last line of your function:

Select Insert/Procedure

- ▶ Every procedure must have a unique name.
- ▶ For user-written functions, the Type *Function* is used.



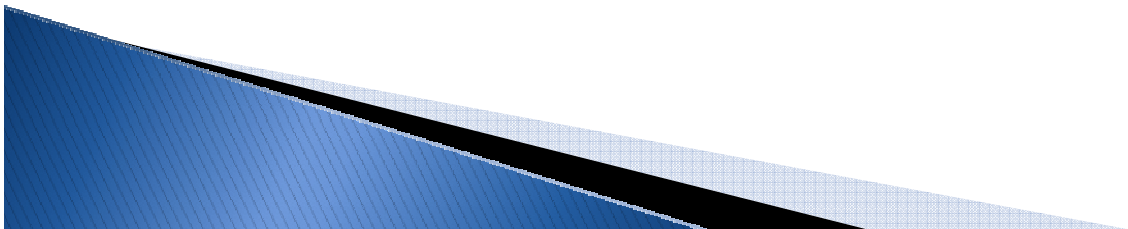
Writing a Function In VBA (Cont.)



Writing a Function in VBA (Cont.)

Write Your Own Function:

- ▶ Enter the required lines of program code to accomplish the desired task.
- ▶ Return to the worksheet and use the function in the same way build-in functions are used..



Writing a Function in VBA (Cont.)

Let's do an example (textbook pg. 524):

13.6.2 A Linear Interpolation Function

A linear-interpolation function is useful for interpolating in data tables. The known X values will include a low, middle, and high value. The low and high Y values will be known, and we will solve for the middle Y value by linear interpolation:

X Values (Knowns)	Y Values (Low and High Values are Known)
x_{LOW}	y_{LOW}
x_{MID}	$[y_{\text{MID}}]$ <i>unknown</i>
x_{HIGH}	y_{HIGH}

A quick way to write a linear-interpolation equation uses the ratios of differences of X values and Y values:

$$\frac{x_{\text{MID}} - x_{\text{LOW}}}{x_{\text{HIGH}} - x_{\text{LOW}}} = \frac{[y_{\text{MID}}] - y_{\text{LOW}}}{y_{\text{HIGH}} - y_{\text{LOW}}}. \quad (13.1)$$

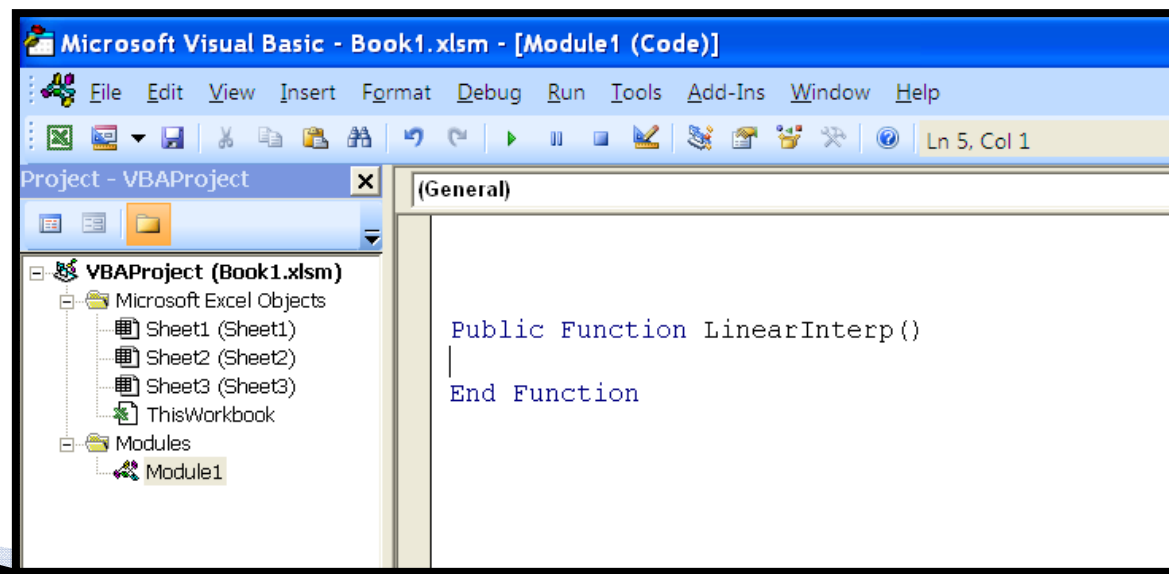
Solving for the unknown, y_{MID} , gives

$$y_{\text{MID}} = y_{\text{LOW}} + (y_{\text{HIGH}} - y_{\text{LOW}}) \times \left[\frac{x_{\text{MID}} - x_{\text{LOW}}}{x_{\text{HIGH}} - x_{\text{LOW}}} \right]. \quad (13.2)$$

Writing a Function in VBA (Cont.)

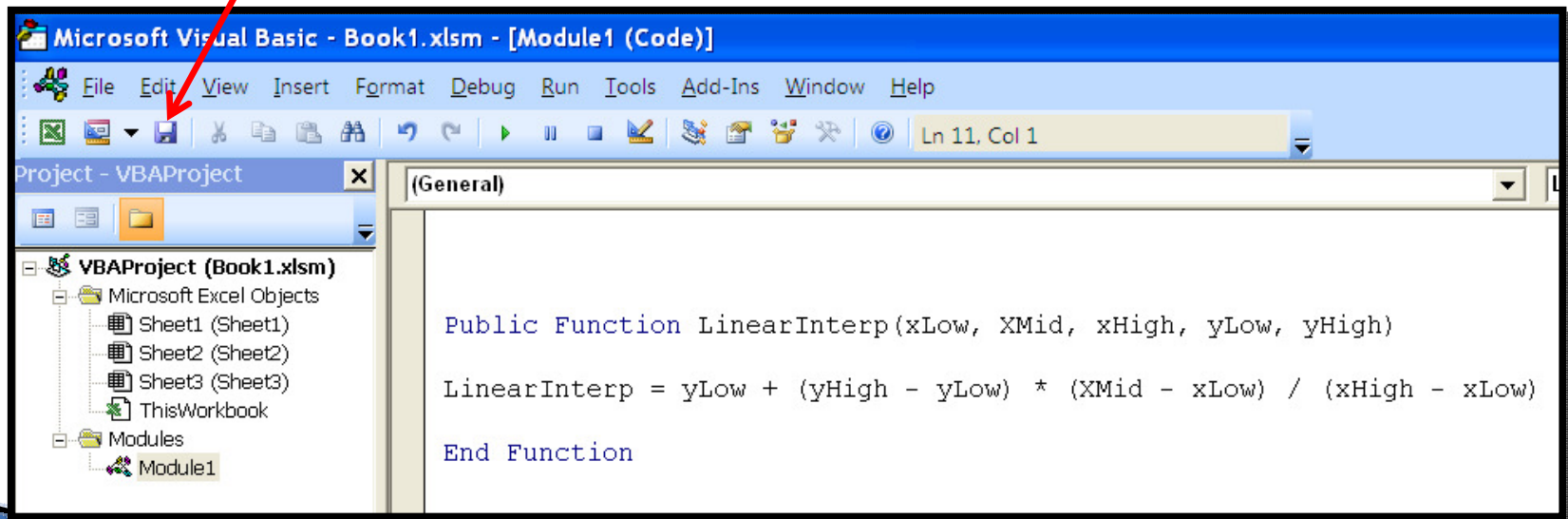
► Procedure:

1. Open the VBA editor (Alt+ F11)
2. Insert a module.
3. Insert the function Procedure.
Name the function “LinearInterp”



Writing a Function in VBA (Cont.)

4. Insert variables (parameters).
xLow, xMid, xHigh, yLow, yHigh
5. Insert Function (see eq. 13.2 problem statement).
6. Save the function.



Testing your Function

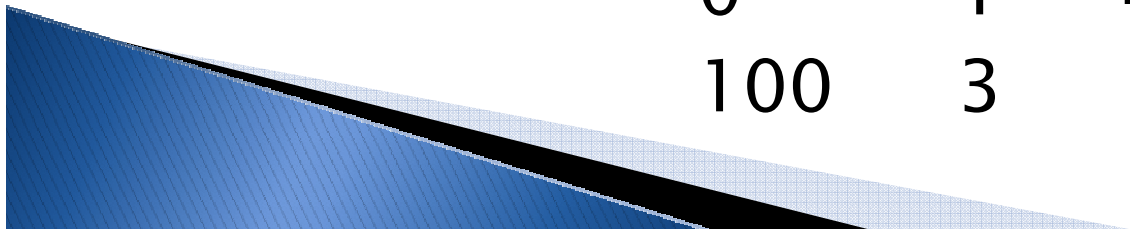
Now that you have created a new function called “LinearInterp” you can use it as any other Excel built-in function.

Let’s try our new function:

1. Set an spreadsheet with the following values of X and Y:

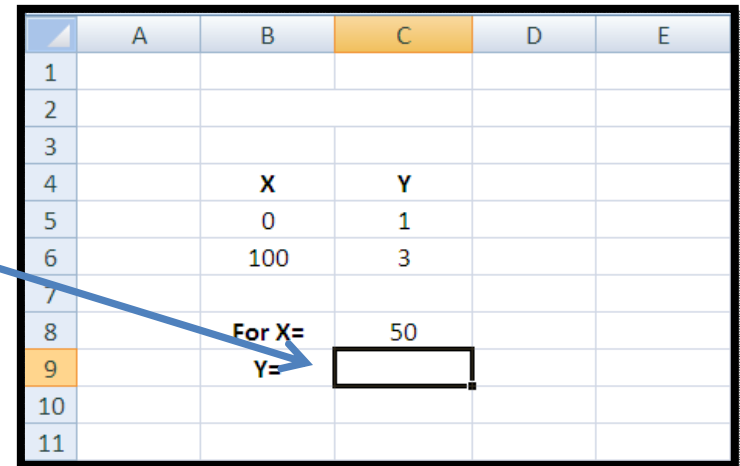
<u>X</u>	<u>Y</u>
0	1
100	3

We want to know the value of Y for a value of X=50



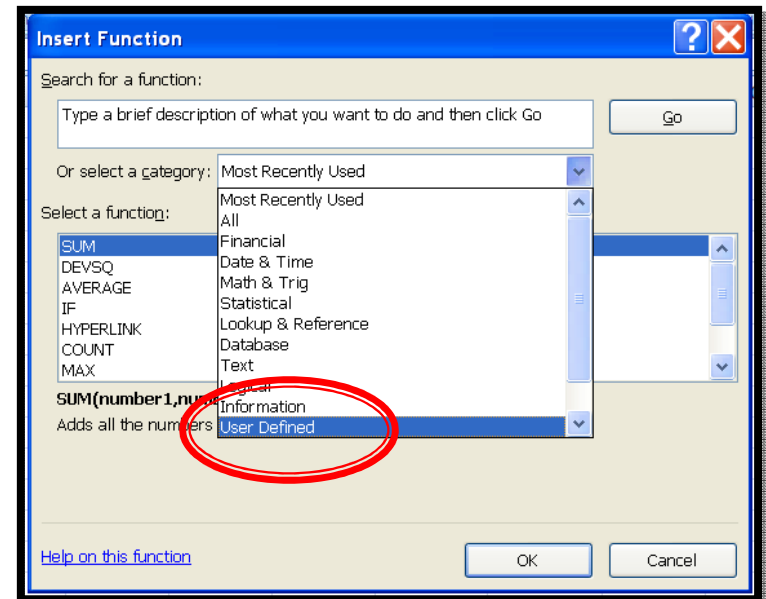
Testing your Function (Cont.)

2. Insert the function in the selected cell:
Use the Ribbon option
Formulas/Insert Function



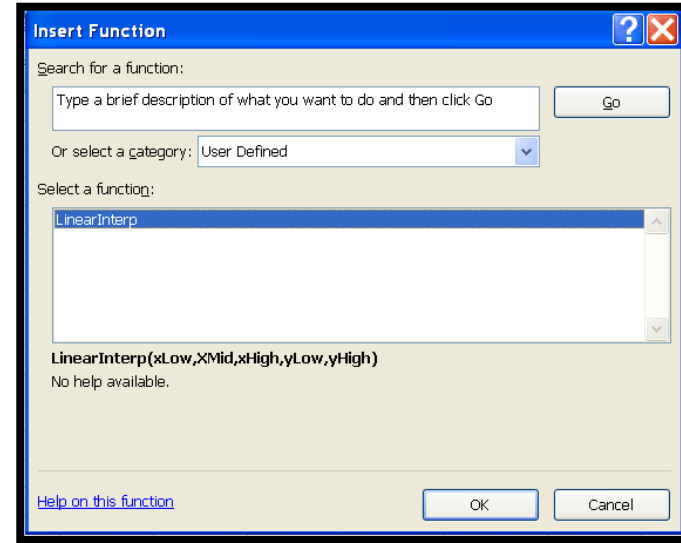
	A	B	C	D	E
1					
2					
3					
4		X	Y		
5		0	1		
6		100	3		
7					
8		For X=	50		
9		Y=			
10					
11					

The insert function dialog will appear. In the box “Select a Category”, select User Defined , click OK

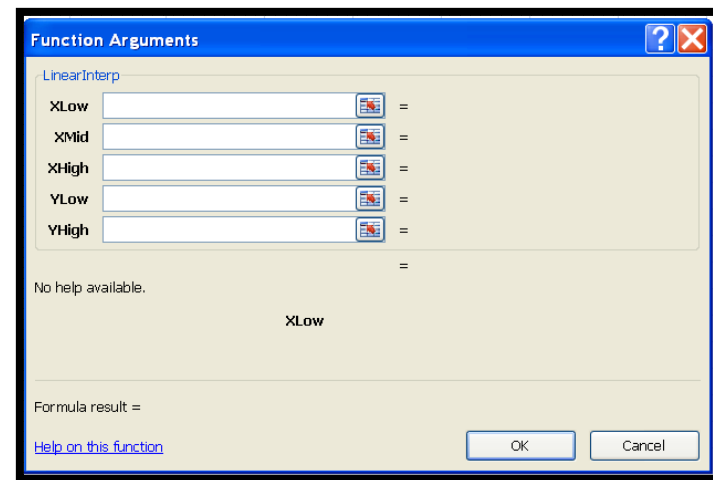


Testing your Function (Cont.)

3. Select your function name and click OK.



3. The function argument dialog will be displayed



Testing your Function (Cont.)

5. Input the function arguments and click OK:

The screenshot shows an Excel spreadsheet with a data table and a dialog box for the LinearInterp function. The data table is as follows:

X	Y
0	1
100	3

For X= 50
Y= 16,C5,C6)

The dialog box, titled "Function Arguments", shows the following arguments for the LinearInterp function:

Argument	Value	Result
XLow	B5	= 0
XMid	C8	= 50
XHigh	B6	= 100
YLow	C5	= 1
YHigh	C6	= 3

The dialog box also displays "No help available.", "YHigh", and "Formula result = 2".

Testing your Function (Cont.)

The result is displayed in the selected cell.

	A	B	C	D
1				
2				
3				
4		X	Y	
5		0	1	
6		100	3	
7				
8		For X=	50	
9		Y=	2	
10				

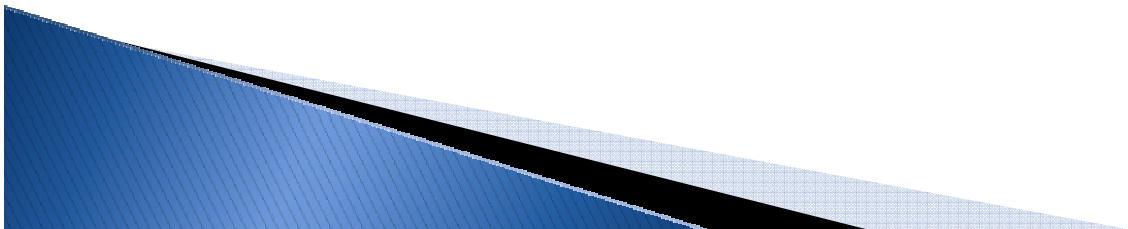
Conditional Execution

- ▶ The classic conditional execution statement is the *If* statement.
- ▶ An *If* statement is used to select from two options by means of the result of a logical condition.

Ex.

RV = "No Ice"

If Temp < 32 *Then* RV = "Ice"



Loops

- ▶ Loop structures are used to perform calculations over and over again.
- ▶ The classic execution is:

```
suma = 0.0  
For i = 1 To 10  
    suma = suma + i  
next i
```

