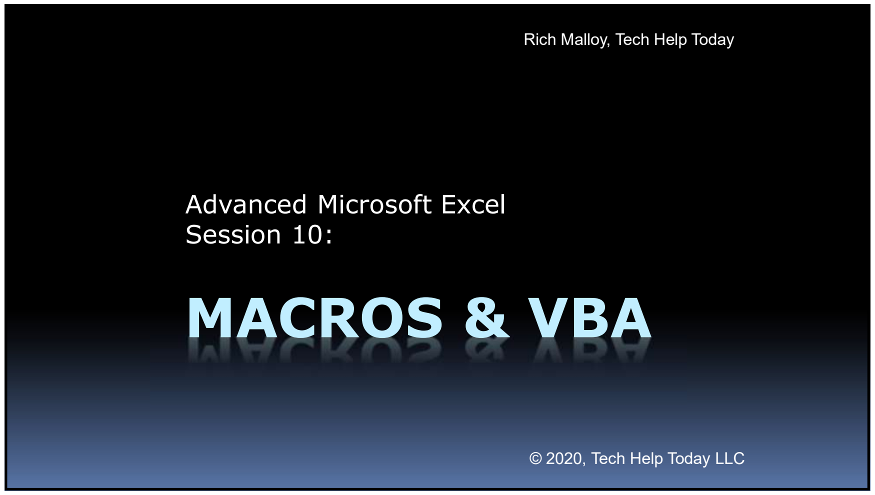


Rich Malloy, Tech Help Today

Advanced Microsoft Excel
Session 10:

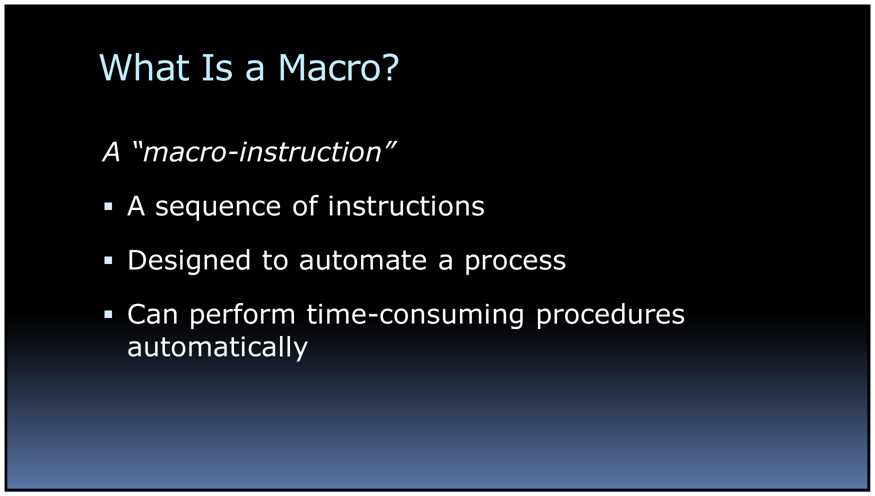
MACROS & VBA

© 2020, Tech Help Today LLC

1

MACROS

2

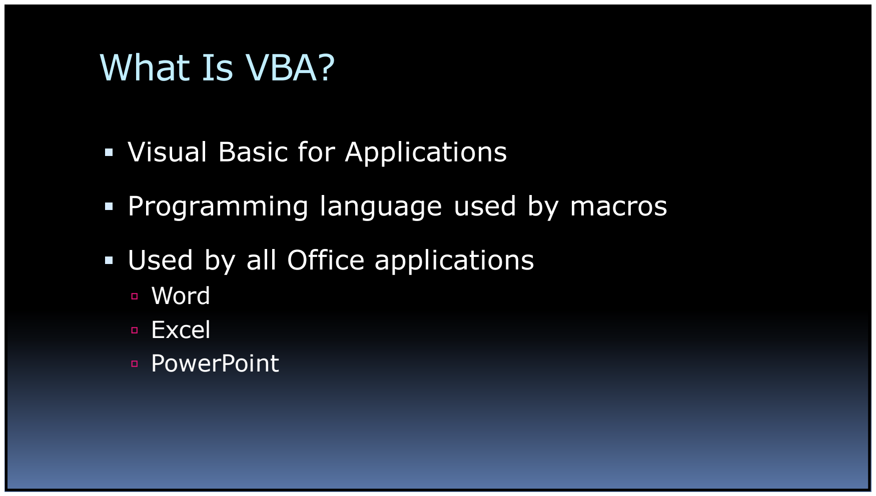


What Is a Macro?

A "macro-instruction"

- A sequence of instructions
- Designed to automate a process
- Can perform time-consuming procedures automatically

3



What Is VBA?

- Visual Basic for Applications
- Programming language used by macros
- Used by all Office applications
  - Word
  - Excel
  - PowerPoint

4

What Can a Macro Do?

- Insert boilerplate text
- Jump to another location/worksheet
- Automate repetitive operations
  - E.g., Same procedure has to be done in 10 different workbooks

5

What Can VBA Do?

Everything Macros can do, plus:

- Automate complex procedures
- Create custom functions
- Create custom commands
- Create a simple “front-end” for fool-proof data entry

6

How to Create a Macro

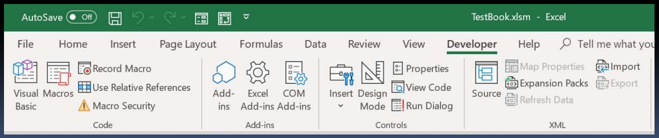
Two ways:

- Record it
  - With the **Developer Tab**
- Write it in the **VB Editor**

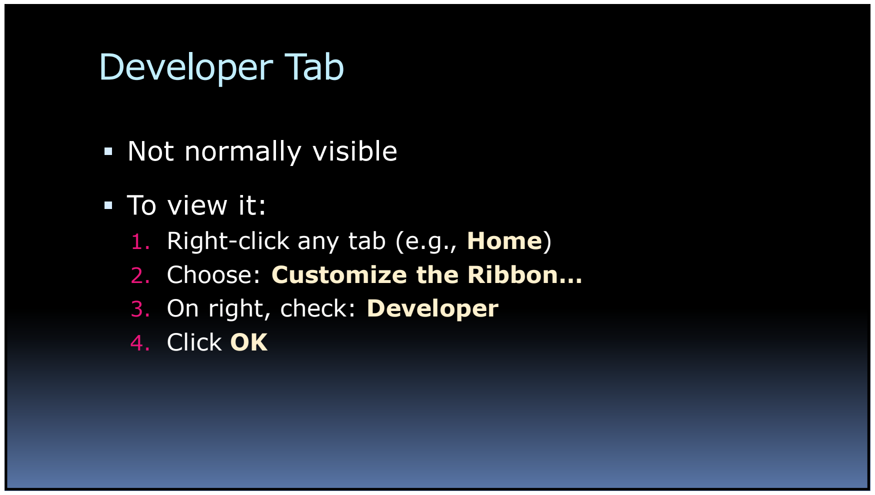
7

Developer Tab

- Used to create and edit macros
- Normally hidden from users



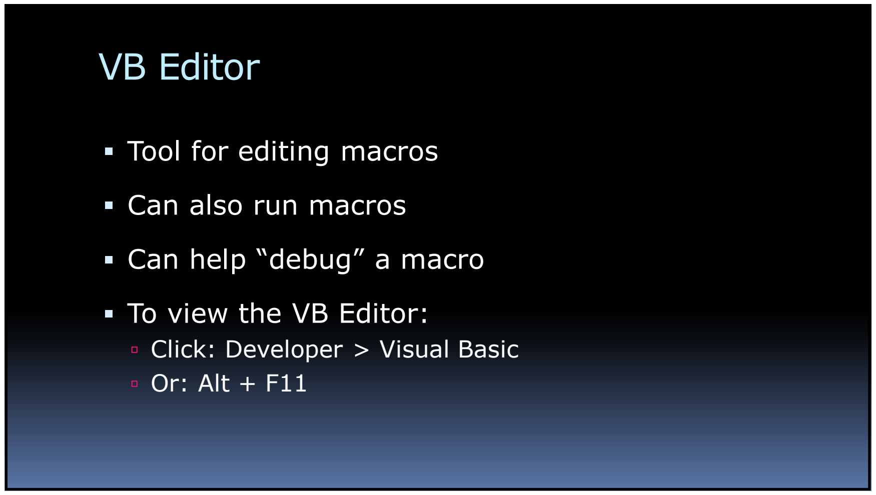
8



Developer Tab

- Not normally visible
- To view it:
  1. Right-click any tab (e.g., **Home**)
  2. Choose: **Customize the Ribbon…**
  3. On right, check: **Developer**
  4. Click **OK**

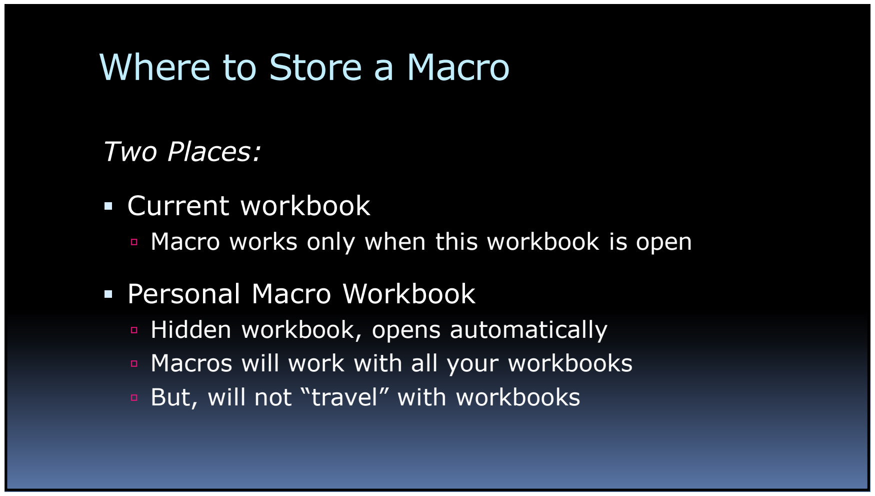
9



VB Editor

- Tool for editing macros
- Can also run macros
- Can help "debug" a macro
- To view the VB Editor:
  - Click: Developer > Visual Basic
  - Or: Alt + F11

10

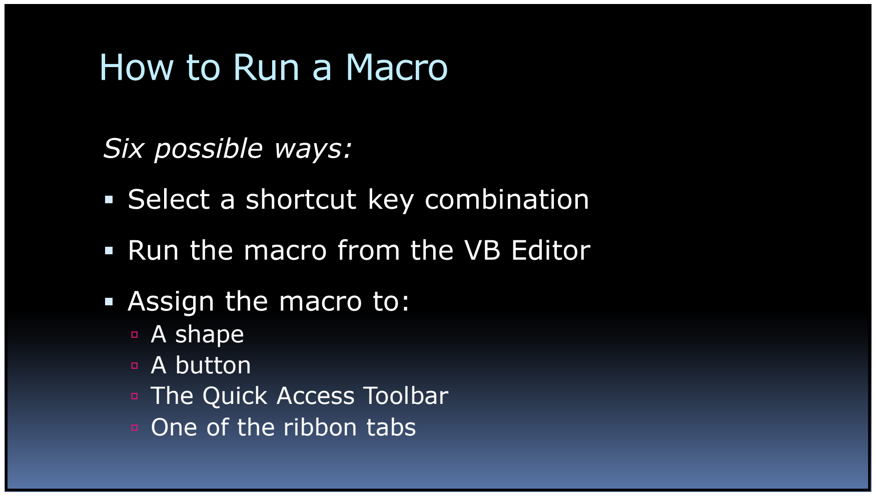


Where to Store a Macro

Two Places:

- Current workbook
  - Macro works only when this workbook is open
- Personal Macro Workbook
  - Hidden workbook, opens automatically
  - Macros will work with all your workbooks
  - But, will not "travel" with workbooks

11

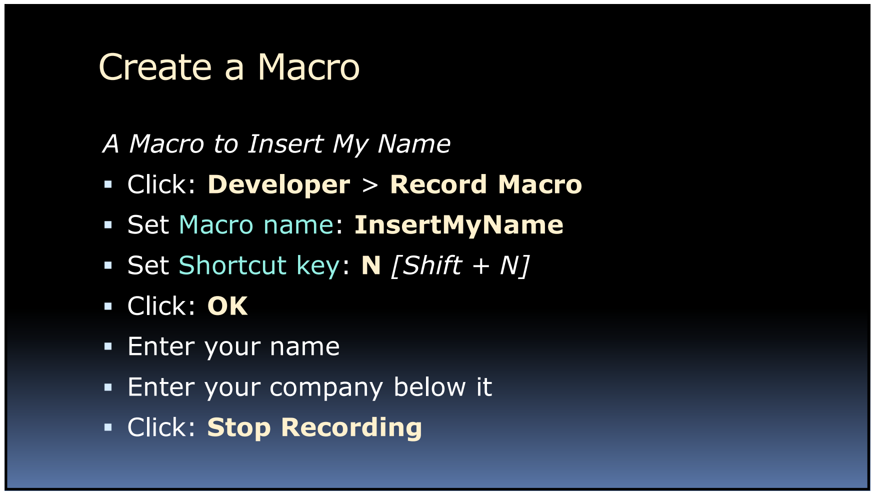


How to Run a Macro

Six possible ways:

- Select a shortcut key combination
- Run the macro from the VB Editor
- Assign the macro to:
  - A shape
  - A button
  - The Quick Access Toolbar
  - One of the ribbon tabs

12

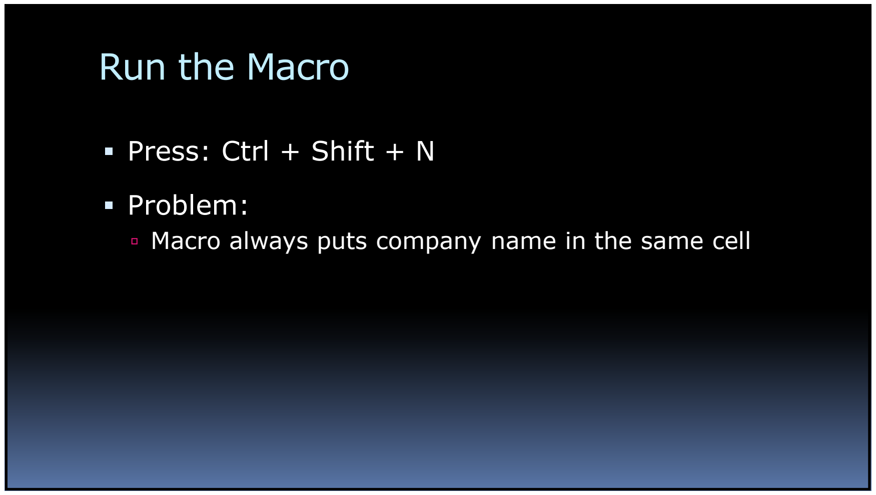


Create a Macro

A Macro to Insert My Name

- Click: **Developer** > **Record Macro**
- Set Macro name: **InsertMyName**
- Set Shortcut key: **N** *[Shift + N]*
- Click: **OK**
- Enter your name
- Enter your company below it
- Click: **Stop Recording**

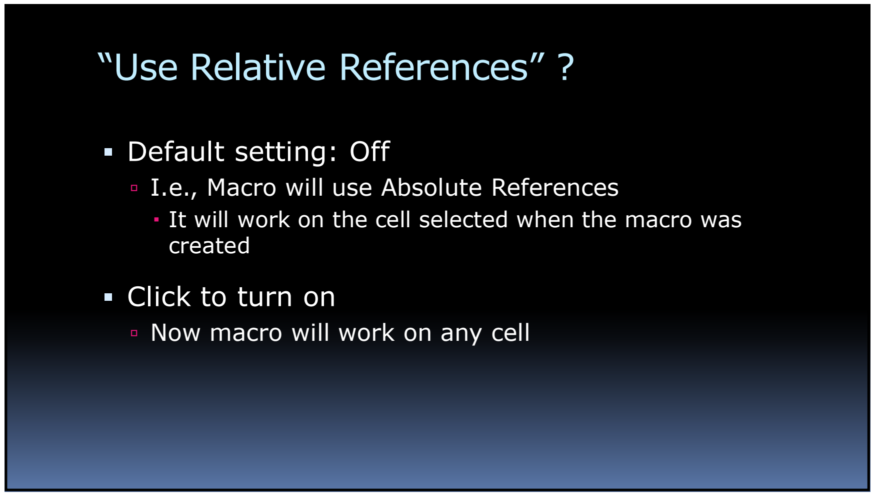
13



Run the Macro

- Press: Ctrl + Shift + N
- Problem:
  - Macro always puts company name in the same cell

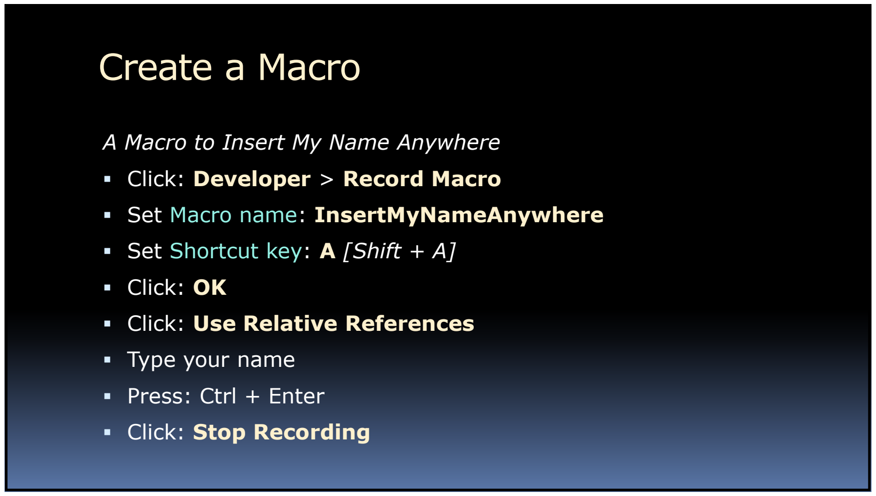
14



"Use Relative References" ?

- Default setting: Off
  - I.e., Macro will use Absolute References
    - It will work on the cell selected when the macro was created
- Click to turn on
  - Now macro will work on any cell

15



Create a Macro

A Macro to Insert My Name Anywhere

- Click: **Developer** > **Record Macro**
- Set Macro name: **InsertMyNameAnywhere**
- Set Shortcut key: **A** *[Shift + A]*
- Click: **OK**
- Click: **Use Relative References**
- Type your name
- Press: Ctrl + Enter
- Click: **Stop Recording**

16

How to Edit a Macro

Two Choices:

- Easy Way: Remove and re-record it
- Hard Way: Edit the VBA code

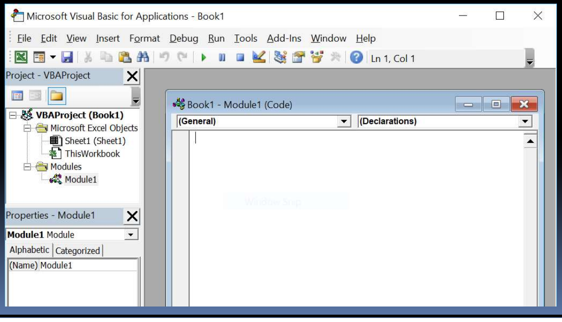
17

THE VISUAL BASIC EDITOR

18

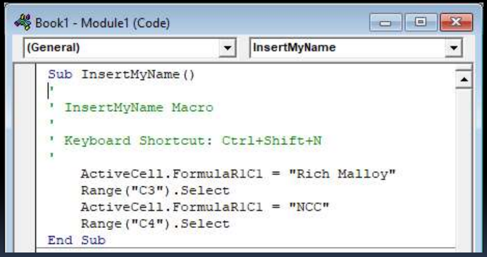
The VB Editor

- For editing macros and procedures



19

Code for a Simple Macro



```
Sub InsertMyName()
'
' InsertMyName Macro
'
' Keyboard Shortcut: Ctrl+Shift+N
'
    ActiveCell.FormulaR1C1 = "Rich Malloy"
    Range("C3").Select
    ActiveCell.FormulaR1C1 = "NCC"
    Range("C4").Select
End Sub
```

20

Code for a Simple Macro

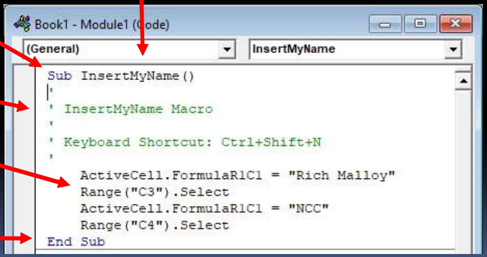
Sub: Subprocedure

Name of Macro

Comments

Instruction code

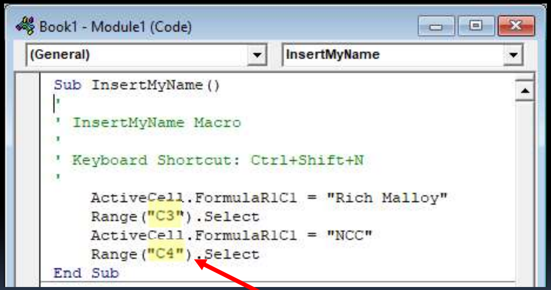
End of procedure



```
Sub InsertMyName()
'
' InsertMyName Macro
'
' Keyboard Shortcut: Ctrl+Shift+N
'
    ActiveCell.FormulaR1C1 = "Rich Malloy"
    Range("C3").Select
    ActiveCell.FormulaR1C1 = "NCC"
    Range("C4").Select
End Sub
```

21

Code for the First Macro

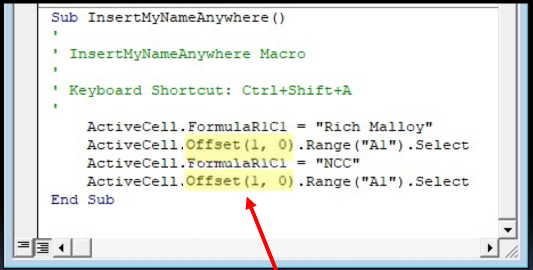


```
Sub InsertMyName()
'
' InsertMyName Macro
'
' Keyboard Shortcut: Ctrl+Shift+N
'
    ActiveCell.FormulaR1C1 = "Rich Malloy"
    Range("C3").Select
    ActiveCell.FormulaR1C1 = "NCC"
    Range("C4").Select
End Sub
```

Absolute References

22

Code for the Second Macro



```
Sub InsertMyNameAnywhere()
'
' InsertMyNameAnywhere Macro
'
' Keyboard Shortcut: Ctrl+Shift+A
'
    ActiveCell.FormulaR1C1 = "Rich Malloy"
    ActiveCell.Offset(1, 0).Range("A1").Select
    ActiveCell.FormulaR1C1 = "NCC"
    ActiveCell.Offset(1, 0).Range("A1").Select
End Sub
```

A Relative Reference

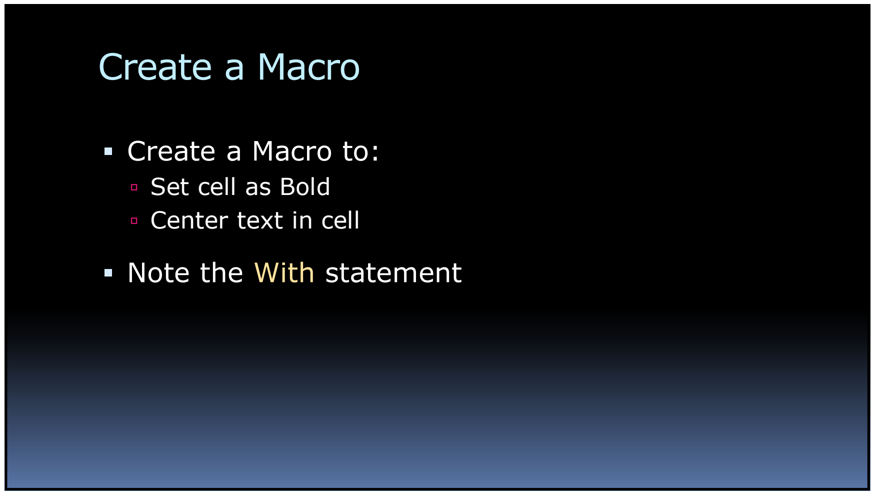
Move Down 1 Row

23

Watch Code Being Created

- Open both Excel and VB Editor
  - Click: Developer > Visual Basic
- Put them side by side
- In Excel, record a macro
- Watch code appear in VB Editor

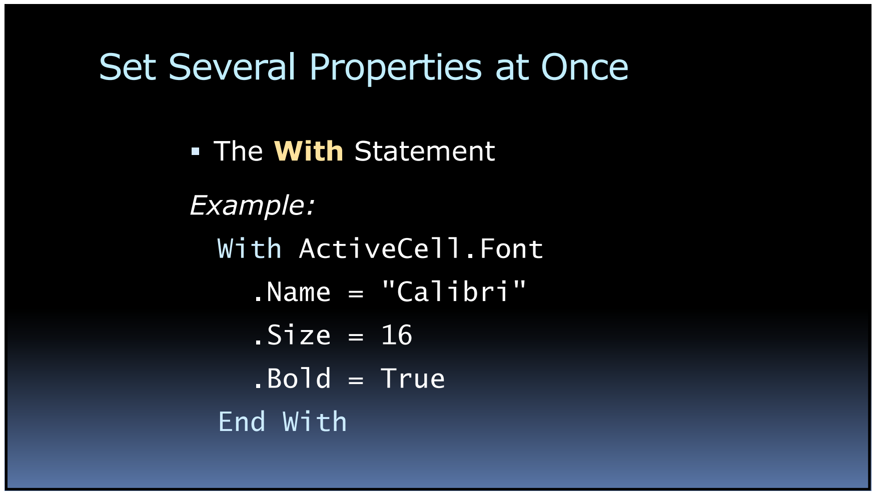
24



Create a Macro

- Create a Macro to:
  - Set cell as Bold
  - Center text in cell
- Note the With statement

25



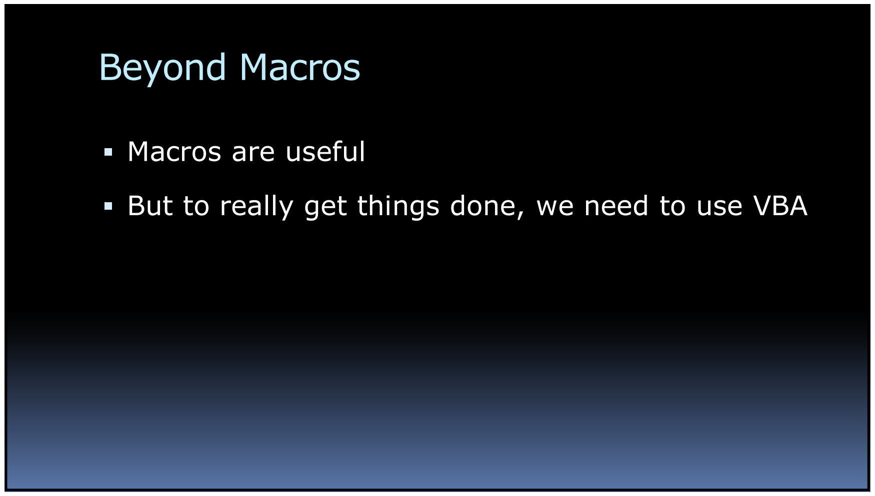
Set Several Properties at Once

- The **With** Statement

Example:

```
With ActiveCell.Font
  .Name = "Calibri"
  .Size = 16
  .Bold = True
End With
```

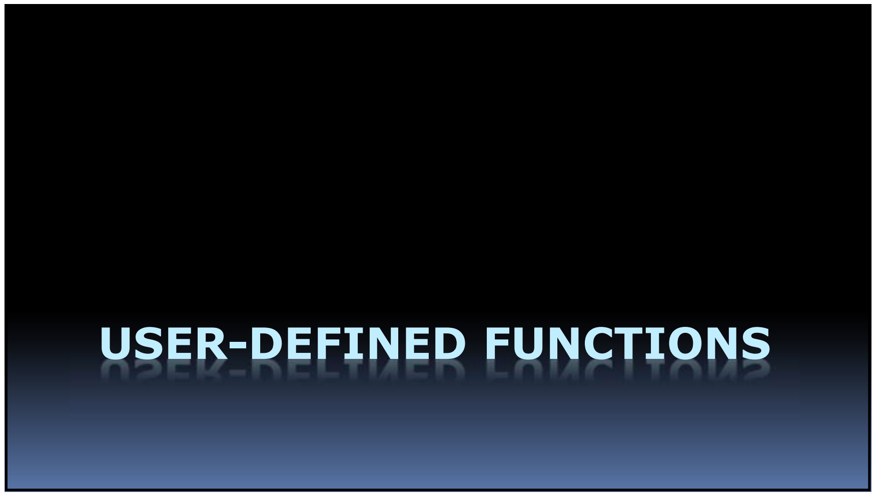
26



Beyond Macros

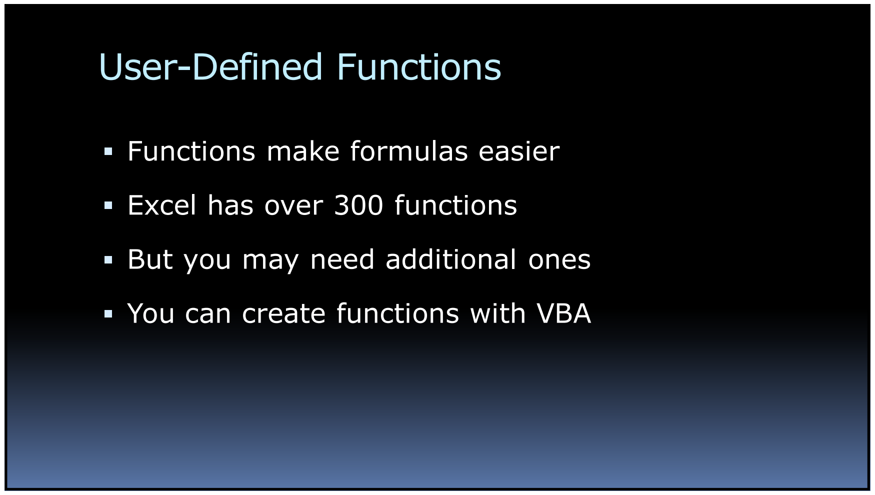
- Macros are useful
- But to really get things done, we need to use VBA

27



USER-DEFINED FUNCTIONS

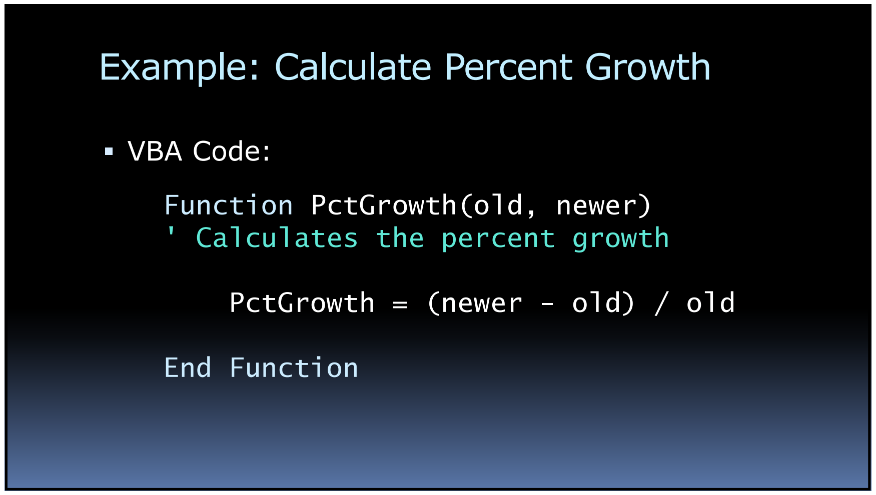
28



User-Defined Functions

- Functions make formulas easier
- Excel has over 300 functions
- But you may need additional ones
- You can create functions with VBA

29

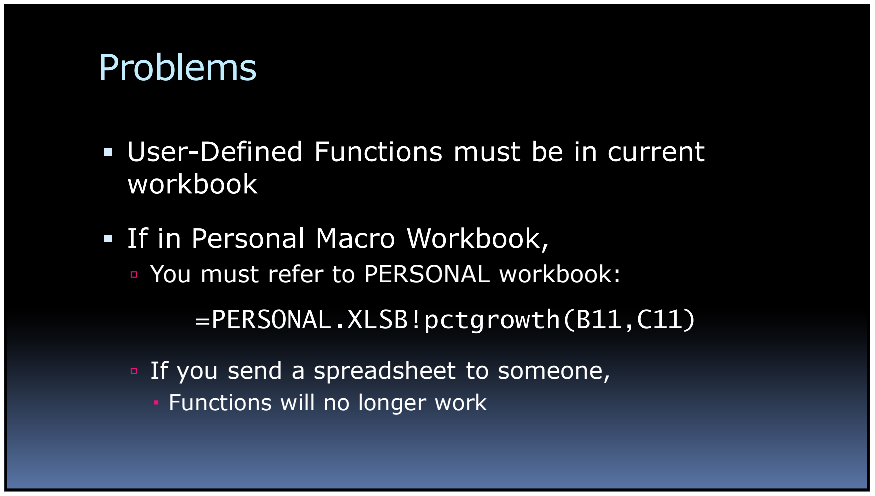


Example: Calculate Percent Growth

- VBA Code:

```
Function PctGrowth(old, newer)
' Calculates the percent growth

    PctGrowth = (newer - old) / old

End Function
```

30

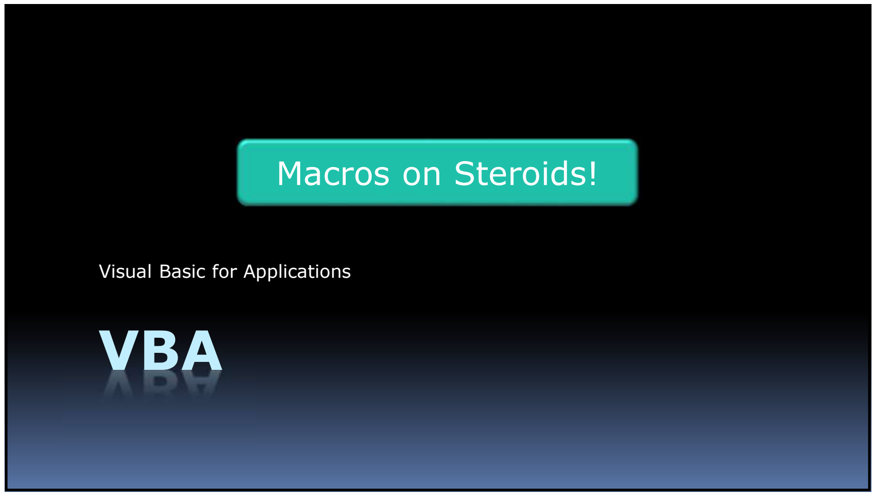


Problems

- User-Defined Functions must be in current workbook
- If in Personal Macro Workbook,
  - You must refer to PERSONAL workbook:

    `=PERSONAL.XLSB!pctgrowth(B11,C11)`
  - If you send a spreadsheet to someone,
    - Functions will no longer work

31

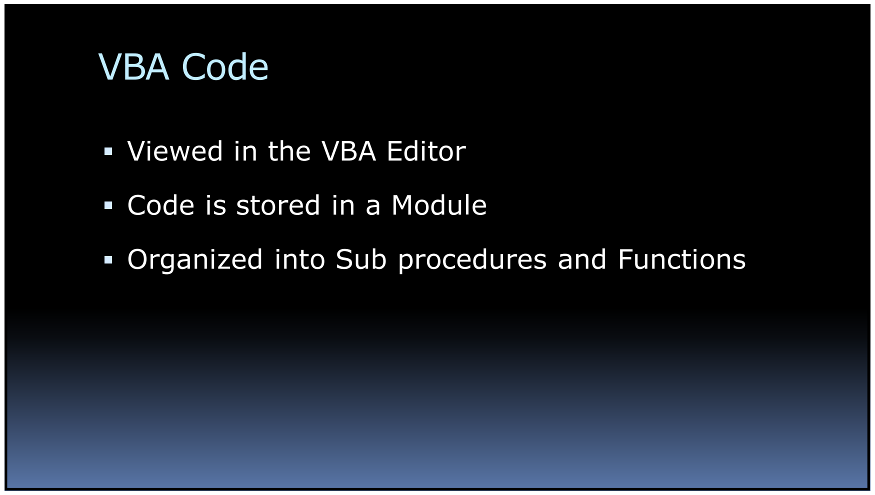


Macros on Steroids!

Visual Basic for Applications

VBA

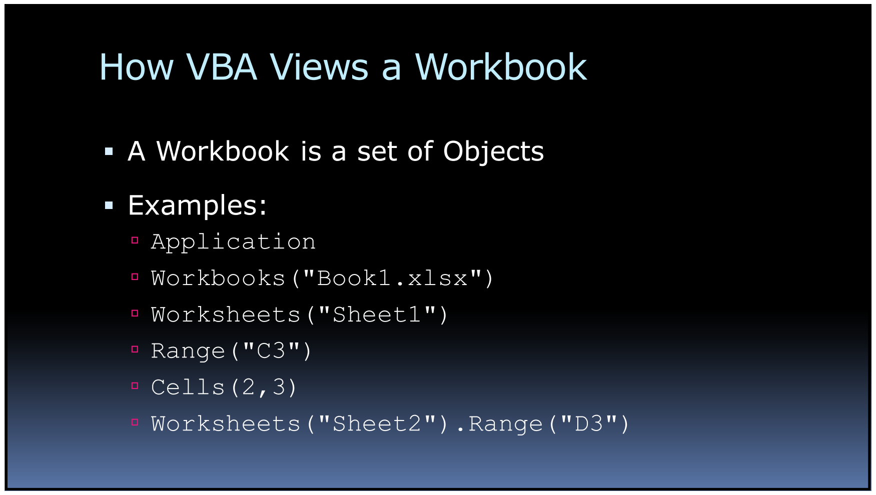
32



VBA Code

- Viewed in the VBA Editor
- Code is stored in a Module
- Organized into Sub procedures and Functions

33

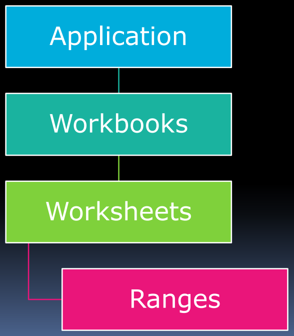


How VBA Views a Workbook

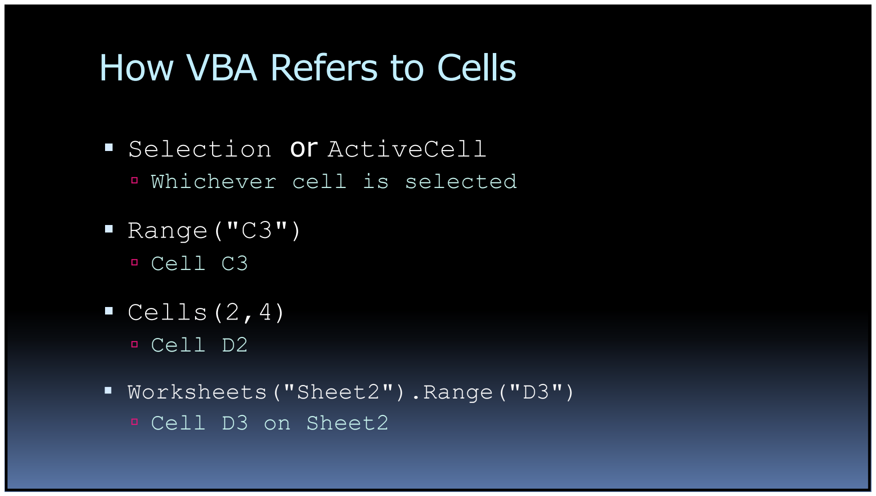
- A Workbook is a set of Objects
- Examples:
  - `Application`
  - `Workbooks("Book1.xlsx")`
  - `Worksheets("Sheet1")`
  - `Range("C3")`
  - `Cells(2,3)`
  - `Worksheets("Sheet2").Range("D3")`

34

Hierarchy of Excel Objects



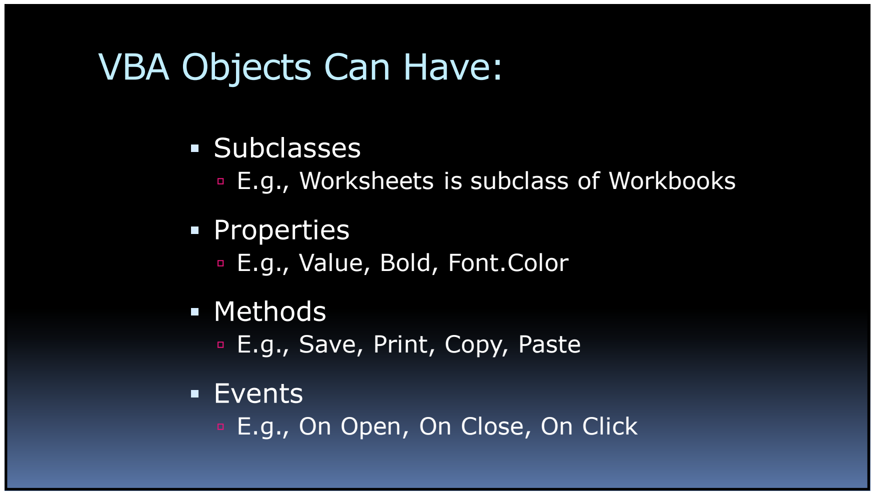
35



How VBA Refers to Cells

- `Selection` or `ActiveCell`
  - `Whichever cell is selected`
- `Range("C3")`
  - `Cell C3`
- `Cells(2,4)`
  - `Cell D2`
- `Worksheets("Sheet2").Range("D3")`
  - `Cell D3 on Sheet2`

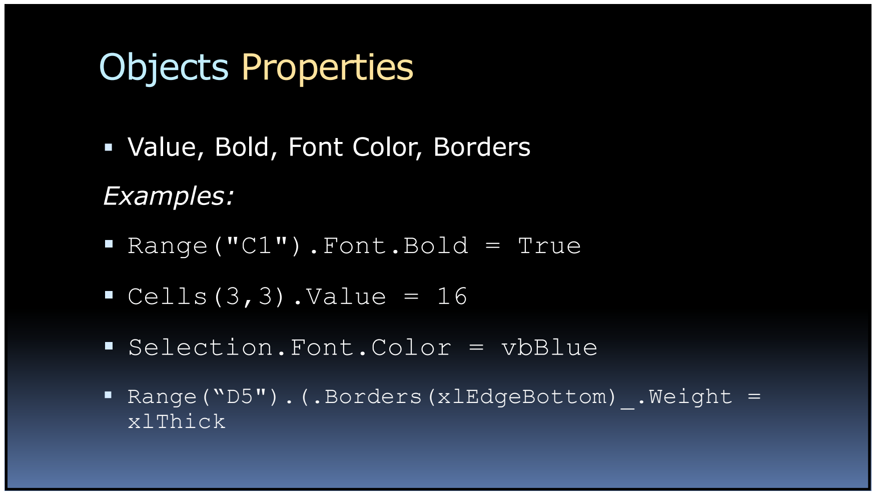
36



VBA Objects Can Have:

- Subclasses
  - E.g., Worksheets is subclass of Workbooks
- Properties
  - E.g., Value, Bold, Font.Color
- Methods
  - E.g., Save, Print, Copy, Paste
- Events
  - E.g., On Open, On Close, On Click

37



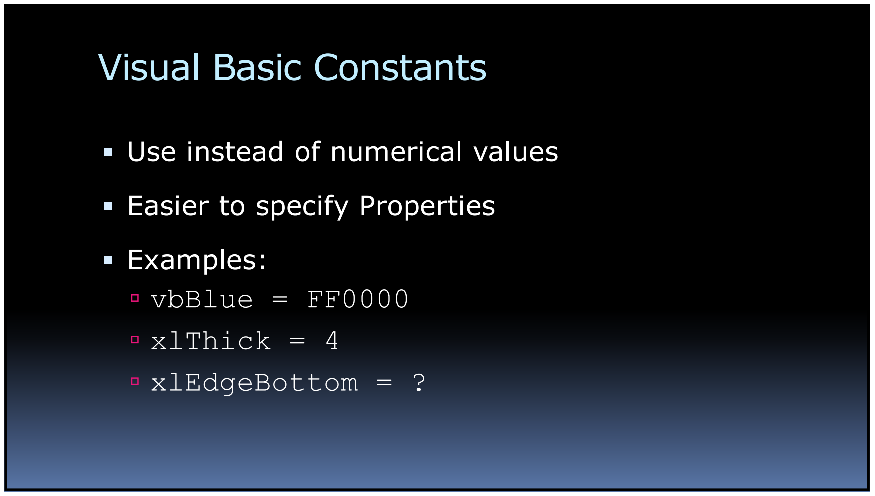
Objects Properties

- Value, Bold, Font Color, Borders

Examples:

- `Range("C1").Font.Bold = True`
- `Cells(3,3).Value = 16`
- `Selection.Font.Color = vbBlue`
- `Range(“D5").(.Borders(xlEdgeBottom)_.Weight = xlThick`

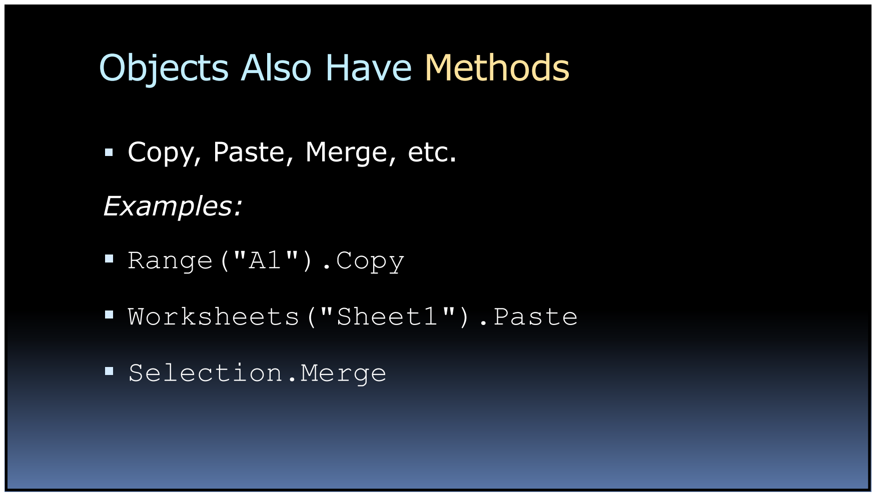
38



Visual Basic Constants

- Use instead of numerical values
- Easier to specify Properties
- Examples:
  - `vbBlue = FF0000`
  - `xlThick = 4`
  - `xlEdgeBottom = ?`

39



Objects Also Have Methods

- Copy, Paste, Merge, etc.

Examples:

- `Range("A1").Copy`
- `Worksheets("Sheet1").Paste`
- `Selection.Merge`

40

Object-Oriented Programming

- Old Style:
  - Tell the program to do something
  - `Copy Range("D3")`
- Object-Oriented Approach
  - Tell the object to do something
  - `Range("D3").Copy`

41

Warning

- Never work on a live workbook
- Always work on a copy
- **VBA has no Undo button**

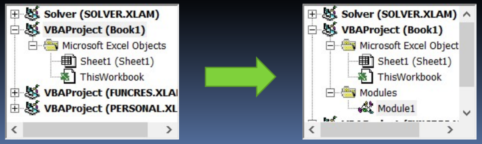
42

YOUR FIRST VBA PROGRAM

43

Your First VBA Program

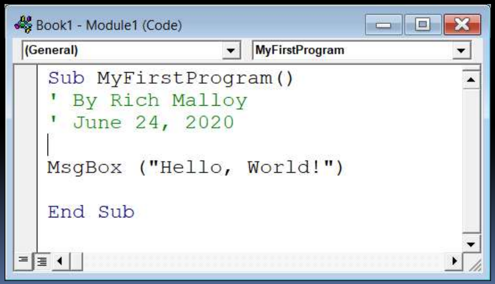
- Open VB Editor
- In Project Explorer, select current workbook (e.g., "Book 1")
- Click: Insert > Module



44

Your First VBA Program

- In the Module1 Code Window, type:



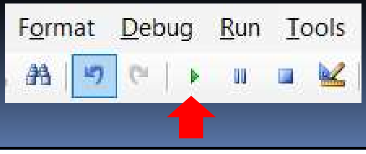
```
Sub MyFirstProgram()
' By Rich Malloy
' June 24, 2020

MsgBox ("Hello, World!")

End Sub
```

45

Run the Code

Do one of the following:

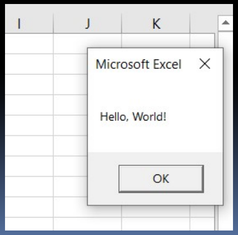
- Click: Run > Run Sub/Userform
- Press F5 key
- Click the Run button



46

Congratulations!

- It works!
- Click: OK
- And have a beer



Hello, World!

47

CREATE A *REAL* VBA PROGRAM

48

Beyond Macros: A VBA Program Can:

- Alter its actions depending on conditions
- Do things over and over again
- Can send you messages about its status

49

VBA Programming Principles

- The Macro Recorder is your friend
  - Let the Recorder create code for you
  - Google is another good friend
- Practice Evolutionary Development
  - Start simple and gradually improve
  - Save & run at every step
  - Fix errors immediately
- Use comments and white space

50

Scenario: Angry Boss

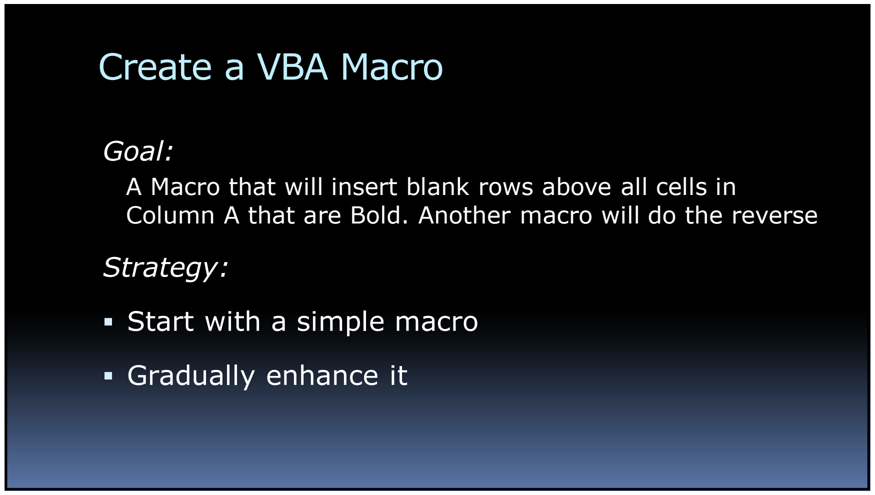
- Wants blank lines in table to separate groups of data
- But you want no blank lines
  - Easier to sort & create Pivot Tables

51

Setup

- Download the file: AX10C-Workbook - VBA.xlsx

52



Create a VBA Macro

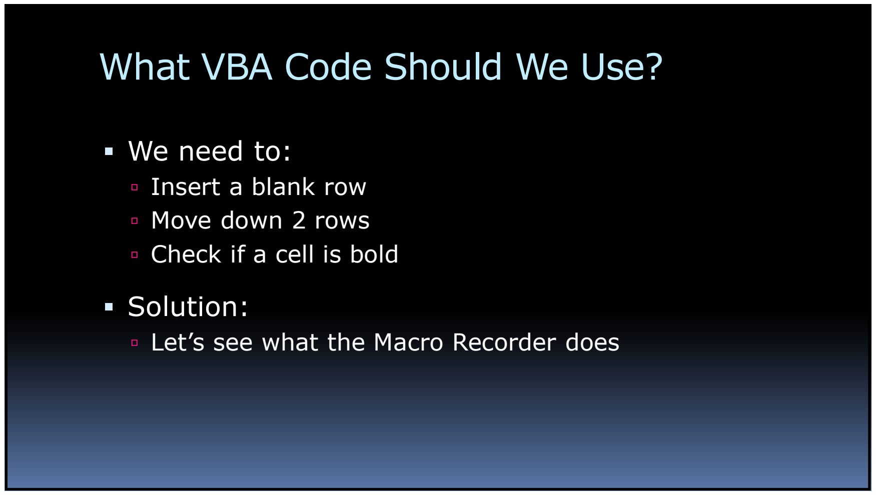
Goal:

A Macro that will insert blank rows above all cells in Column A that are Bold. Another macro will do the reverse

Strategy:

- Start with a simple macro
- Gradually enhance it

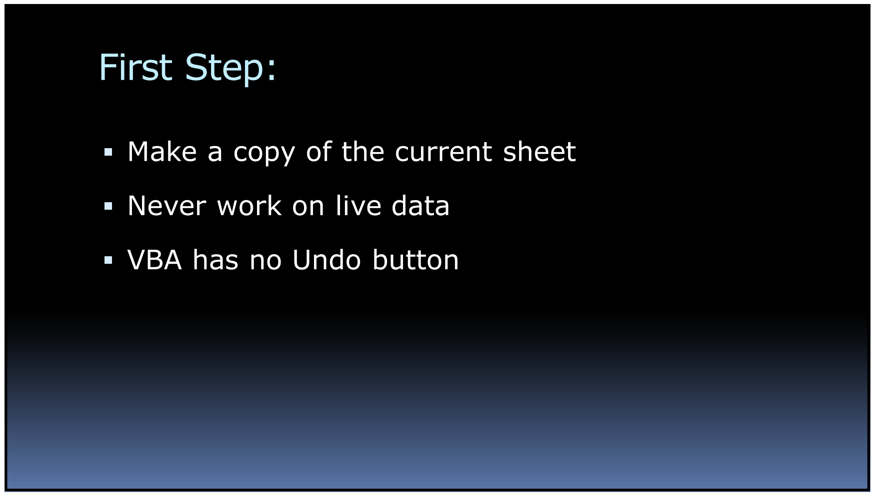
53



What VBA Code Should We Use?

- We need to:
  - Insert a blank row
  - Move down 2 rows
  - Check if a cell is bold
- Solution:
  - Let’s see what the Macro Recorder does

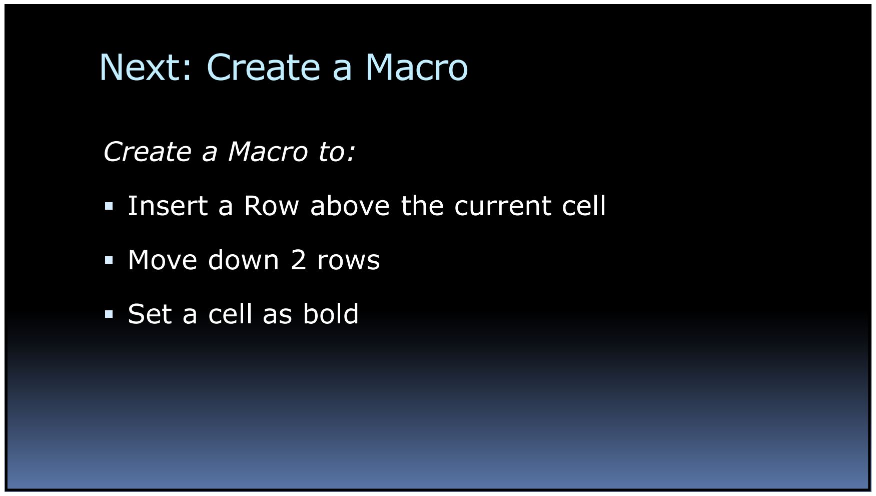
54



First Step:

- Make a copy of the current sheet
- Never work on live data
- VBA has no Undo button

55

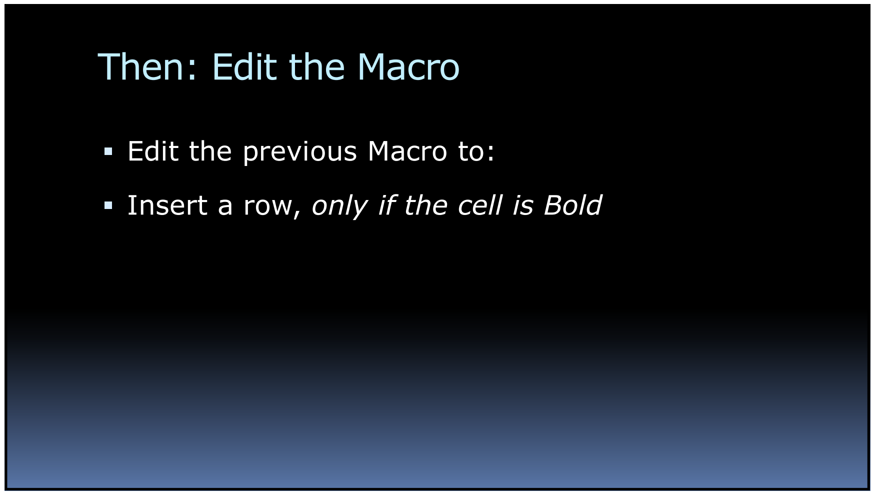


Next: Create a Macro

Create a Macro to:

- Insert a Row above the current cell
- Move down 2 rows
- Set a cell as bold

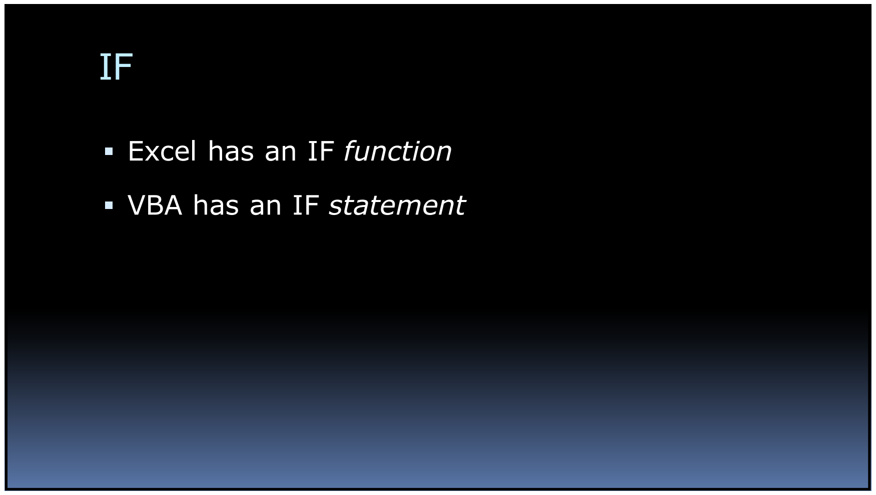
56



Then: Edit the Macro

- Edit the previous Macro to:
- Insert a row, *only if the cell is Bold*

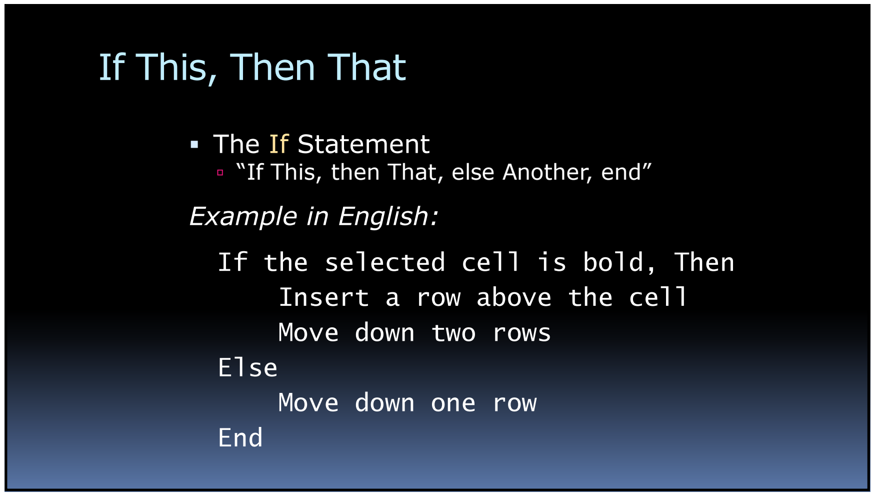
57



IF

- Excel has an IF *function*
- VBA has an IF *statement*

58



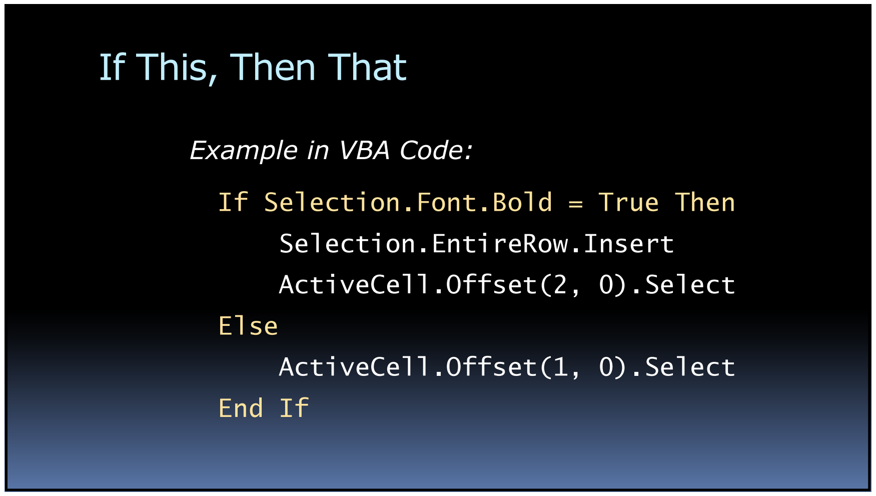
If This, Then That

- The If Statement
  - "If This, then That, else Another, end"

Example in English:

```
If the selected cell is bold, Then
    Insert a row above the cell
    Move down two rows
Else
    Move down one row
End
```

59

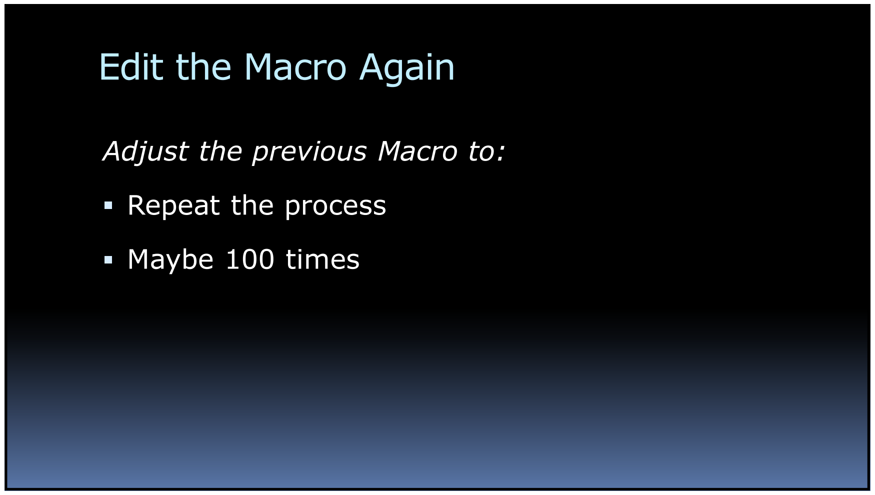


If This, Then That

Example in VBA Code:

```
If Selection.Font.Bold = True Then
    Selection.EntireRow.Insert
    ActiveCell.Offset(2, 0).Select
Else
    ActiveCell.Offset(1, 0).Select
End If
```

60

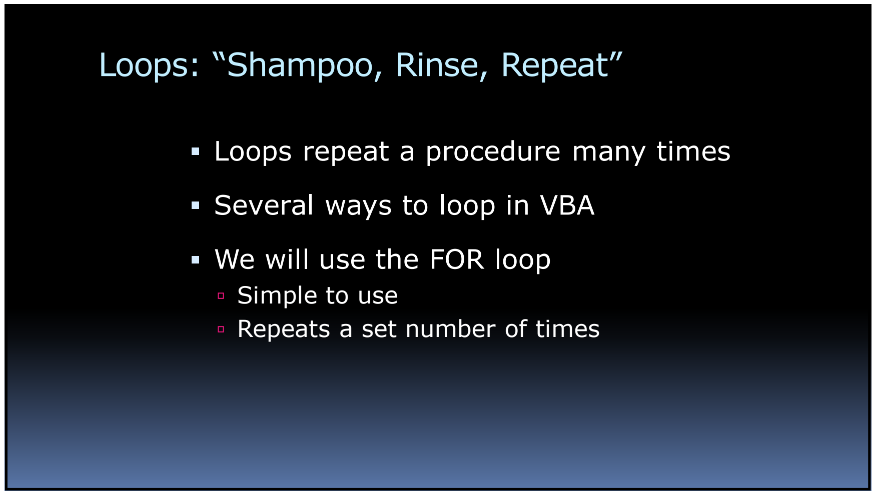


Edit the Macro Again

Adjust the previous Macro to:

- Repeat the process
- Maybe 100 times

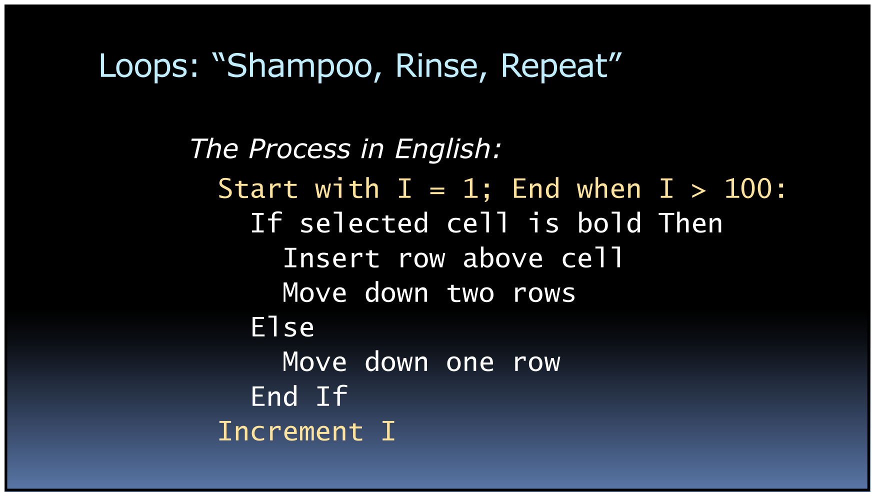
61



Loops: "Shampoo, Rinse, Repeat"

- Loops repeat a procedure many times
- Several ways to loop in VBA
- We will use the FOR loop
  - Simple to use
  - Repeats a set number of times

62

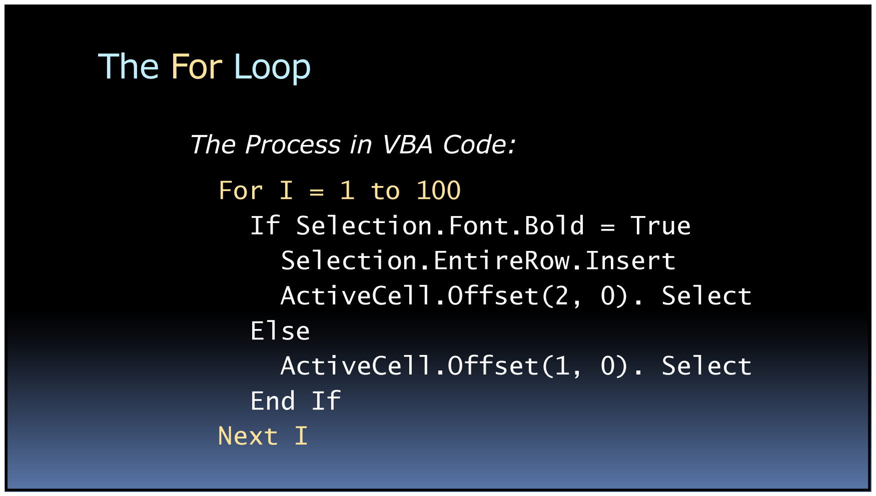


Loops: "Shampoo, Rinse, Repeat"

The Process in English:

```
Start with I = 1; End when I > 100:
  If selected cell is bold Then
    Insert row above cell
    Move down two rows
  Else
    Move down one row
  End If
Increment I
```

63

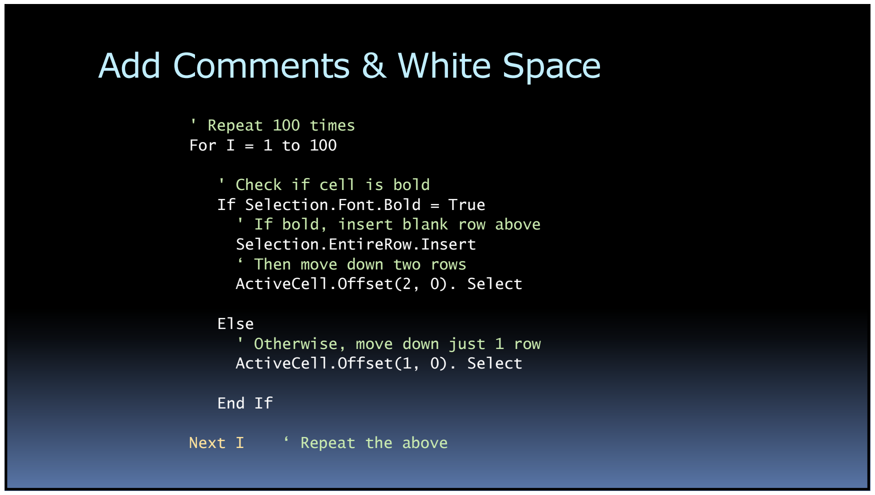


The For Loop

The Process in VBA Code:

```
For I = 1 to 100
  If Selection.Font.Bold = True
    Selection.EntireRow.Insert
    ActiveCell.Offset(2, 0). Select
  Else
    ActiveCell.Offset(1, 0). Select
  End If
Next I
```

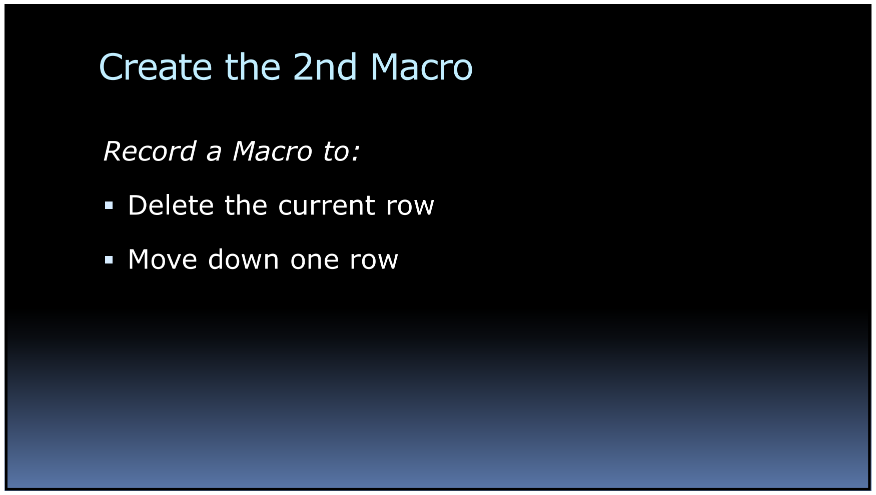
64



Add Comments & White Space

```
' Repeat 100 times
For I = 1 to 100

   ' Check if cell is bold
   If Selection.Font.Bold = True
     ' If bold, insert blank row above
     Selection.EntireRow.Insert
     ' Then move down two rows
     ActiveCell.Offset(2, 0). Select

   Else
     ' Otherwise, move down just 1 row
     ActiveCell.Offset(1, 0). Select

   End If

Next I     ' Repeat the above
```

65

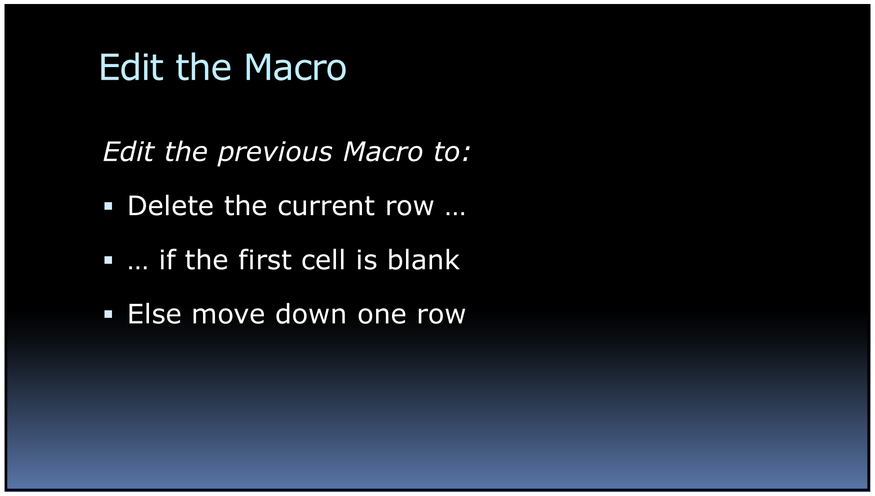


Create the 2nd Macro

Record a Macro to:

- Delete the current row
- Move down one row

66

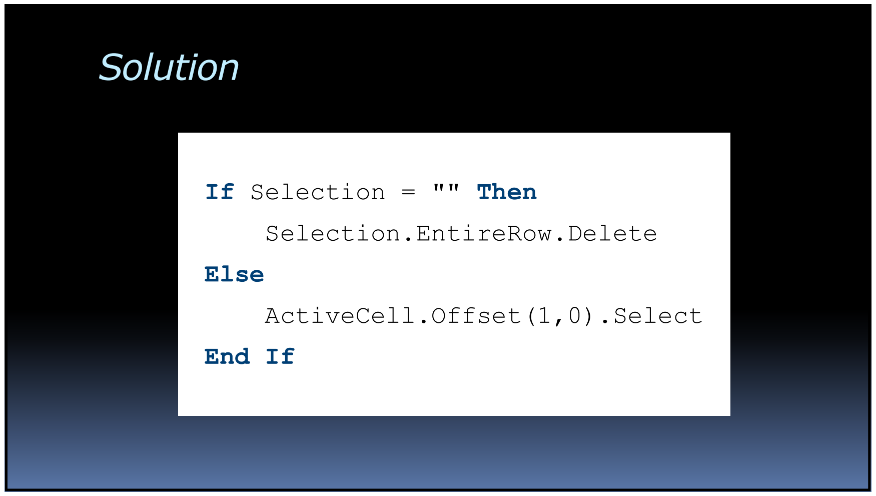


Edit the Macro

Edit the previous Macro to:

- Delete the current row …
- … if the first cell is blank
- Else move down one row

67



Solution

```
If Selection = "" Then
    Selection.EntireRow.Delete
Else
    ActiveCell.Offset(1,0).Select
End If
```

68

Edit the Macro Again

*Edit the previous Macro to delete **all** blank rows:*

1. Delete the current row …
2. … if the first cell is blank
3. Move down one row
4. Repeat maybe 100 times

69

Solution

```
For i = 1 To 100
   If Selection = "" Then
      Selection.EntireRow.Delete
   Else
      ActiveCell.Offset(1, 0).Select
   End If
Next i
```

70

Run the Macros from Buttons

- Add two buttons to run the macros
- Click: Developer > Insert
- Use Form Controls
- Active X Controls are more complex

71

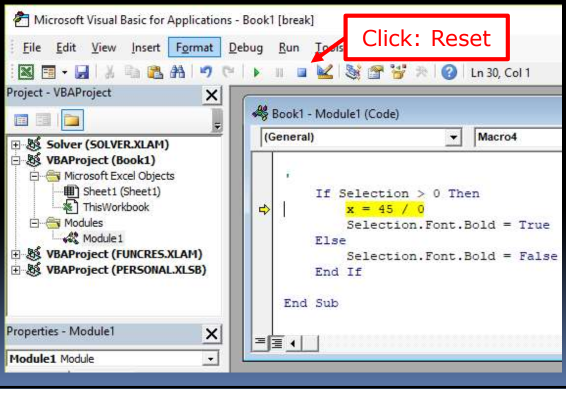
Form Controls

Various tools on the Developer Tab:

- Buttons
- Checkboxes
- Slidebars
- Text boxes
- List boxes
- Combo boxes

72

Bugs Happen – The Debug Mode



```
'
    If Selection > 0 Then
        x = 45 / 0
        Selection.Font.Bold = True
    Else
        Selection.Font.Bold = False
    End If

End Sub
```

73

Quote

"There are two ways to write error-free programs; only the third one works."

— Alan J. Perlis

74

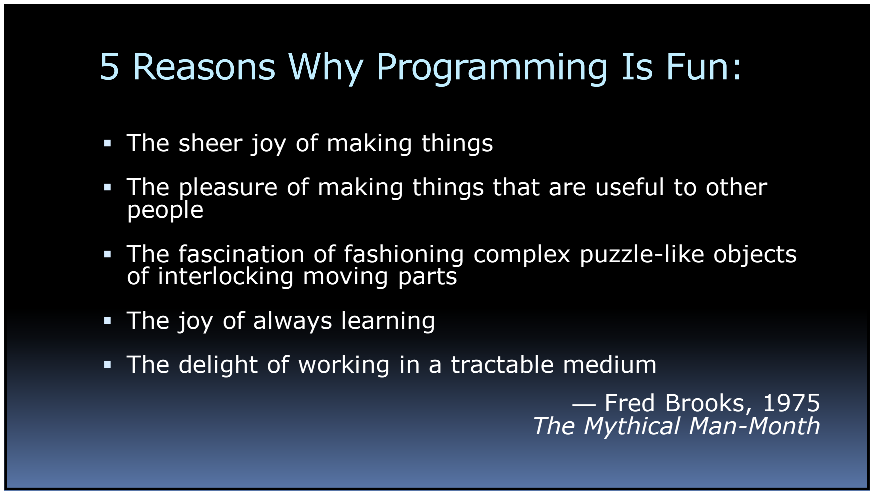
Further Learning on YouTube:

- Leila Gharani:
  - Excel VBA & Macros Tutorials (1 of 22)
  - The Visual Basic Editor (2 of 22)
- Learnit Training:
  - Excel VBA Beginner Tutorial (2 hrs)
- ExcelVBAHelp:
  - 14-Hour VBA Course

75

THE JOY OF CODE

76

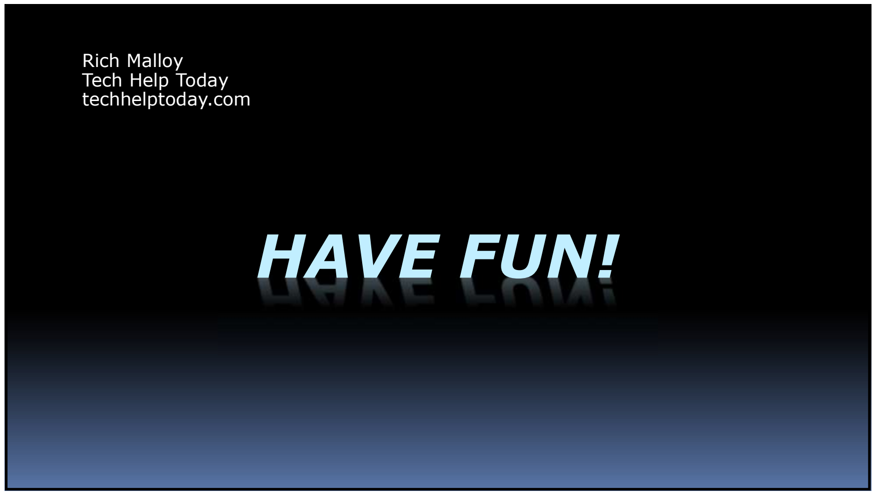


5 Reasons Why Programming Is Fun:

- The sheer joy of making things
- The pleasure of making things that are useful to other people
- The fascination of fashioning complex puzzle-like objects of interlocking moving parts
- The joy of always learning
- The delight of working in a tractable medium

— Fred Brooks, 1975
The Mythical Man-Month

77



Rich Malloy
Tech Help Today
techhelptoday.com

HAVE FUN!

78

79