

Microsoft Excel 2007 - Illustrated

Programming with Excel

UNIT
P

Excel 2007



Objectives

- View VBA code
- Analyze VBA code
- Write VBA code
- Add a conditional statement

Objectives

- Prompt the user for data
- Debug a macro
- Create a main procedure
- Run a main procedure

Unit Introduction

- Excel macros are written in a programming language called Visual Basic for Applications, or **VBA**
 - Create a macro with the Excel macro recorder
 - The recorder writes the VBA instructions for you
 - Enter VBA instructions manually
 - Sequence of VBA statements is called a **procedure**

Viewing VBA Code

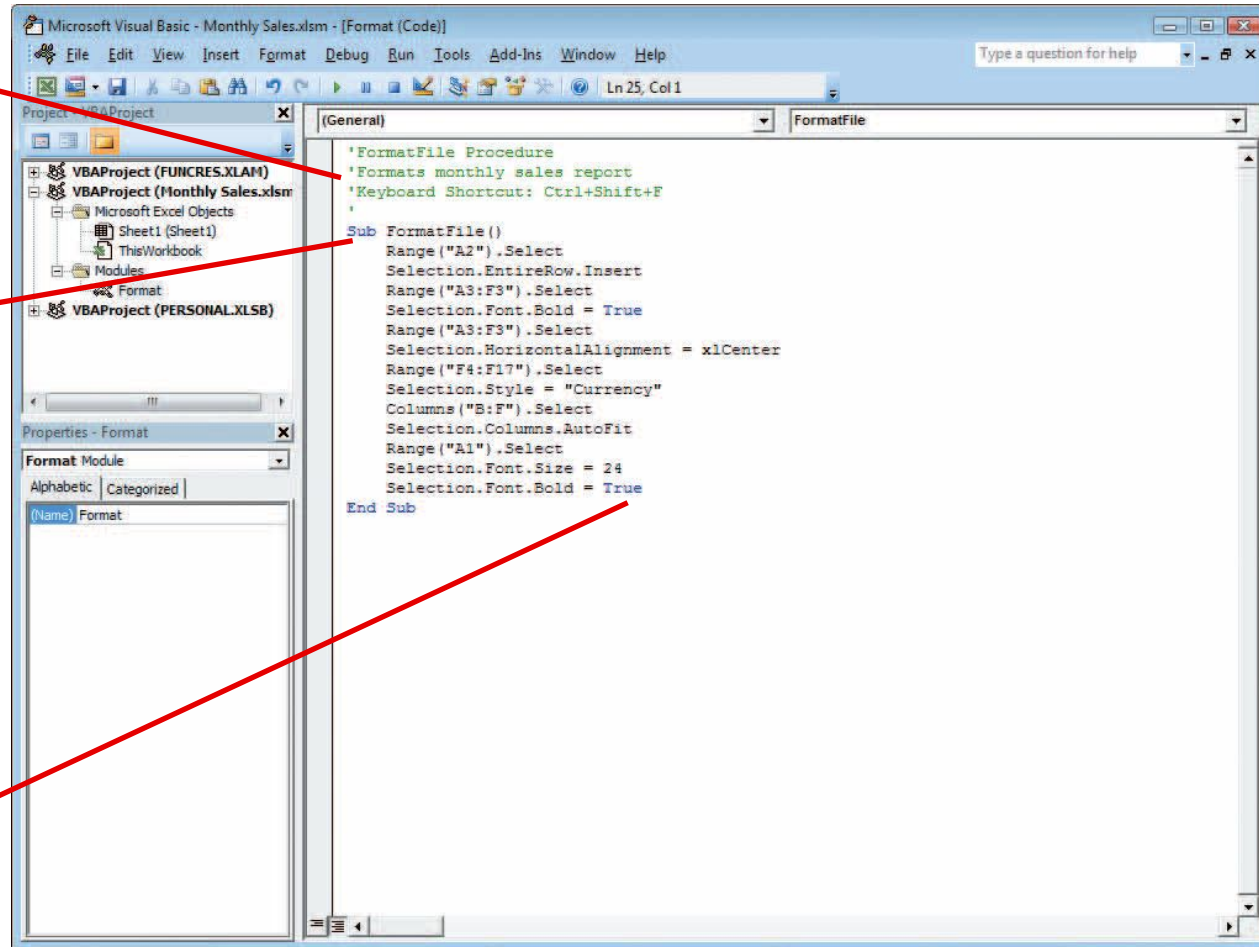
- View existing VBA code to learn the language
 - To view VBA code, open the Visual Basic Editor
 - Contains a Project Explorer window, a Properties window, and a Code window
 - VBA code appears in the Code window
 - The first line of a procedure is called the **procedure header**
 - Items displayed in blue are **keywords**
 - Green notes explaining the code are called **comments**

Viewing VBA Code (cont.)

Comments

Procedure
header

Keyword



```

'FormatFile Procedure
'Formats monthly sales report
'Keyboard Shortcut: Ctrl+Shift+F
'
Sub FormatFile ()
Range ("A2").Select
Selection.EntireRow.Insert
Range ("A3:F3").Select
Selection.Font.Bold = True
Range ("A3:F3").Select
Selection.HorizontalAlignment = xlCenter
Range ("F4:F17").Select
Selection.Style = "Currency"
Columns ("B:F").Select
Selection.Columns.AutoFit
Range ("A1").Select
Selection.Font.Size = 24
Selection.Font.Bold = True
End Sub
    
```

Viewing VBA Code (cont.)

- Understanding the Visual Basic Editor
 - A **module** is the Visual Basic equivalent of a worksheet
 - Store macro procedures
 - A module is stored in a workbook, or **project**, along with worksheets
 - View and edit modules in the Visual Basic Editor

Analyzing VBA Code

- Analyzing VBA code
 - Every element of Excel, including a range, is considered an **object**
 - A **range object** represents a cell or a range of cells
 - A **property** is an attribute of an object that defines one of the object's characteristics, such as size
 - The last line in VBA code is the **procedure footer**

Analyzing VBA Code (cont.)

Selects
range object
cell A2

Applies bold
formatting to
range A3:F3

Sets width of
columns B-F
to AutoFit

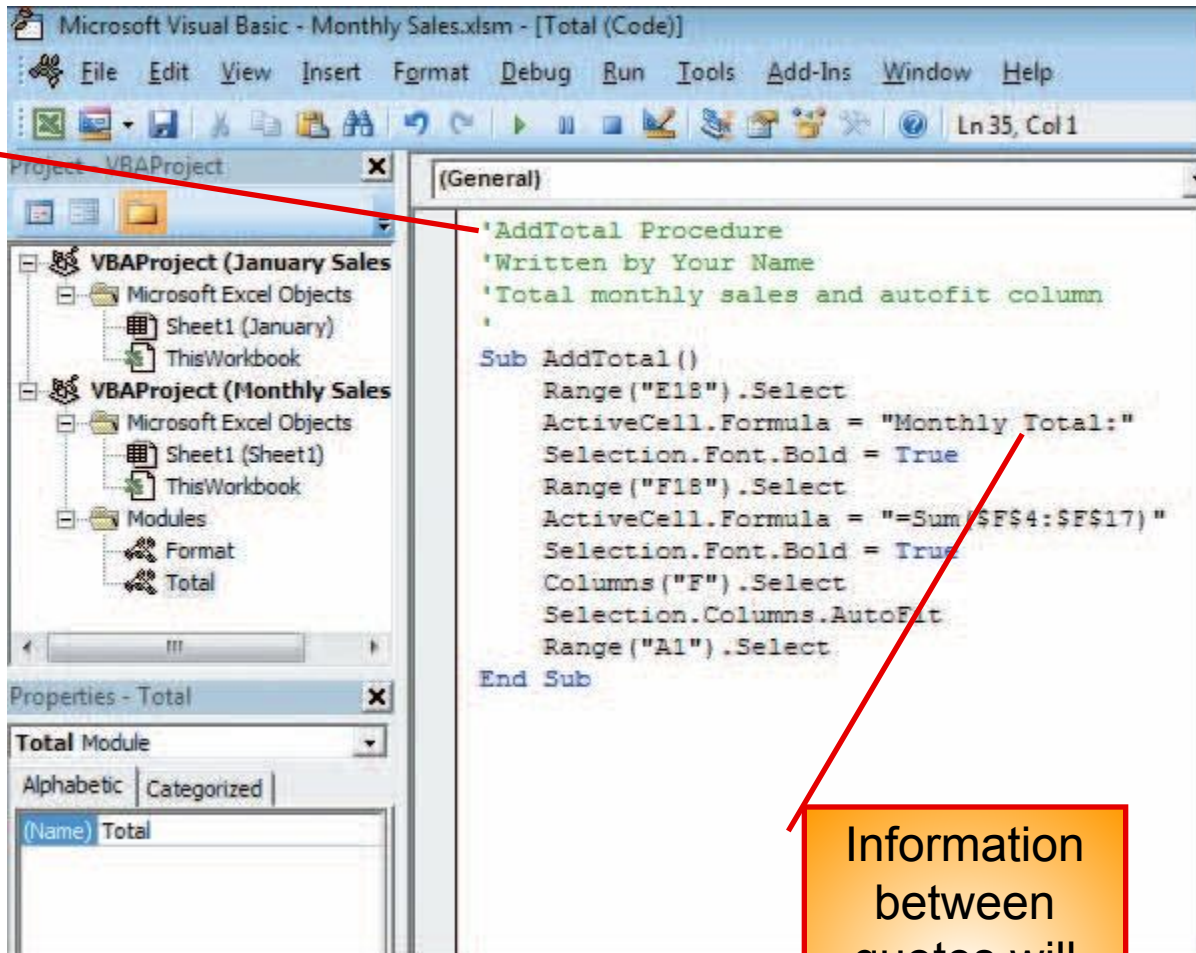
```
'FormatFile Procedure
'Formats monthly sales report
'Keyboard Shortcut: Ctrl+Shift+F
'
Sub FormatFile()
    Range("A2").Select
    Selection.EntireRow.Insert
    Range("A3:F3").Select
    Selection.Font.Bold = True
    Range("A3:F3").Select
    Selection.HorizontalAlignment = xlCenter
    Range("F4:F17").Select
    Selection.Style = "Currency"
    Columns("B:F").Select
    Selection.Columns.AutoFit
    Range("A1").Select
    Selection.Font.Size = 24
    Selection.Font.Bold = True
End Sub
```

Writing VBA Code

- To write your own code, open the Visual Basic Editor and add a module to the workbook
 - You must follow the formatting rules, or **syntax**, of the VBA programming language exactly
 - A misspelled keyword or variable name will cause a procedure to fail

Writing VBA Code (cont.)

Comments
begin with
apostrophes



```
Microsoft Visual Basic - Monthly Sales.xlsm - [Total (Code)]
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
Ln 35, Col 1

Project - VBAProject
  VBAProject (January Sales)
    Microsoft Excel Objects
      Sheet1 (January)
        ThisWorkbook
  VBAProject (Monthly Sales)
    Microsoft Excel Objects
      Sheet1 (Sheet1)
        ThisWorkbook
    Modules
      Format
      Total

Properties - Total
Total Module
Alphabetic | Categorized
(Name) Total

'AddTotal Procedure
'Written by Your Name
'Total monthly sales and autofit column
'
Sub AddTotal ()
  Range ("E18") .Select
  ActiveCell.Formula = "Monthly Total:"
  Selection.Font.Bold = True
  Range ("F18") .Select
  ActiveCell.Formula = "=Sum ($F$4:$F$17) "
  Selection.Font.Bold = True
  Columns ("F") .Select
  Selection.Columns.AutoFit
  Range ("A1") .Select
End Sub
```

Information
between
quotes will
be inserted
in the active
cell

Writing VBA Code (cont.)

- Entering code using AutoComplete
 - To assist you in entering the VBA code, the Editor often displays a list of words that can be used in the macro statement
 - Typically the list appears after you press period [.]

Adding a Conditional Statement

- Sometimes you may want a procedure to take an action based on a certain condition or set of conditions
 - One way to add this type of statement is by using an **If...Then...Else** statement
 - The syntax for this statement is: *If condition then statements Else [else statements]*

Adding a Conditional Statement (cont.)

```
'SalesStatus Procedure
'Written by Your Name
'Tests whether total sales meets the monthly quota
'
Sub SalesStatus()
  Range("E20").Select
  ActiveCell.Formula = "Sales Status:"
  Selection.Font.Bold = True
  'If the total is less than 900000 then
  'insert "Missed Quota" in cell F20
  If Range("F18") <= 900000 Then
    Range("F20").Select
    ActiveCell.Formula = "Missed Quota"
  'otherwise, insert "Met Quota" in cell F20
  Else
    Range("F20").Select
    ActiveCell.Formula = "Met Quota"
  End If
  Range("A1").Select
End Sub
```

Elements of
the
If...then...Else
statement
appear in blue

Prompting the User for Data

- When automating routine tasks, sometimes you need to pause a macro for user input
 - Use the VBA InputBox function to display a dialog box that prompts the user for information
 - A **function** is a predefined procedure that returns a value

Prompting the User for Data (cont.)

```
'FooterInput Procedure
'Written by Your Name
'Customize worksheet footer
'
Sub FooterInput ()
    'Declares the LeftFooterText string variable
    Dim LeftFooterText As String
    'Prompts user for left footer text and stores
    'response in LeftFooterText variable
    LeftFooterText = InputBox("Enter name:")
    'Inserts contents of LeftFooterText into left footer
    '*****THERE IS AN ERROR IN THE FOLLOWING LINE *****
    Worksheets("January").PageSetup.LeftFooter = LeftFooterText
    'Inserts the date in right footer
    Worksheets("January").PageSetup.RightFooter = "&D"
End Sub
```

This text will appear in a dialog box

Comment points out error in next line of the procedure

Debugging a Macro

- When a macro procedure does not run properly, it can be due to an error, called a **bug**, in the code
 - To help you find bugs in a procedure, the Visual Basic Editor steps through the procedure's code one line at a time
 - When you locate an error, you can **debug**, or correct it

Debugging a Macro (cont.)

```
'FooterInput Procedure
'Written by Your Name
'Customize worksheet footer
'
Sub FooterInput()
'Declares the LeftFooterText string variable
Dim LeftFooterText As String
'Prompts user for left footer text and stores
'response in LeftFooter Text variable
LeftFooterText = InputBox("Enter name:")
'Inserts contents of LeftFooterText into left footer
'***** THERE IS AN ERROR IN THE FOLLOWING LINE *****
Worksheets("January").PageSetup.LeftFooter = LeftFooter
'Inserts the date in right footer
Worksheets("January").PageSetup.RightFooter = "&D"
End Sub
```

Indicates
that the
LeftFooter
variable is
empty

Creating a Main Procedure

- Combine several macros that you routinely run together into a procedure
 - This is a **main procedure**
 - To create a main procedure, type a Call statement for each procedure you want to run

Creating a Main Procedure (cont.)

MainProcedure
calls each
procedure in the
order shown

```
'MainProcedure Procedure  
'Written by Your Name  
'Calls sub procedures in sequence  
'  
Sub MainProcedure()  
    Call FormatFile  
    Call AddTotal  
    Call SalesStatus  
    Call FooterInput  
End Sub
```

Running a Main Procedure

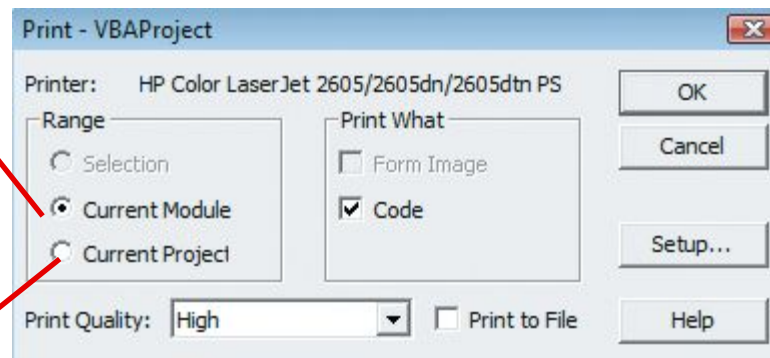
- Running a main procedure allows you to run several macros in sequence
 - Run a main procedure as you would any other macro

Running a Main Procedure (cont.)

Printing Macro Procedures

Current
Module
button

Current
Project
button



Summary

- Learn by viewing and analyzing VBA code
- Write VBA code using the Visual Basic Editor
- Use If..Then..Else statements for conditional actions
- Prompt user for data to automate input tasks
- Use the “Step Into” feature of the Visual Basic Editor to debug macros
- Use Main procedures to combine several macros