CS 200

Lecture 06
Excel Scripting
Miscellaneous Notes

Abbreviations

aka Also Known As

CWS Course Web Site (http://www.student.cs.uwaterloo.ca/~cs200)

VBE Visual Basic Editor

intra- a prefix meaning within — thus “intra-cellular” means “within the cell”

inter- a prefix meaning between — thus “inter-galactic” means “between galaxies”

For our purposes, a “macro” and a “script” are the same thing

Optional background reading

Writing Excel Macros, Chapters 3 and 4, (in library)

Excel Review (Learn)

Excel Review Exercises (Learn)

A good reference if you’re interested in learning more about Excel Scripting is

Writing Excel Macros (2/e), by Steven Roman
Administrativia

Please read and highlight the assignment and course notes before lab

   Staff suggests reading the assignment before lecture

There are hyper-text commented source files for most of the Excel macros used in this lecture

   see “Handouts > Commented Excel Macros” on the course website

Files used in this lecture are on Learn

   Week 6: Application Scripting (Excel): Files for Lecture

Today

   Scripting in Excel

   The Copy Pearl

   Remember that the CWS contains material on the pearls

Warning

   We are running Office 2011, in the Mac Labs

   —test Windows docs in the lab before submitting
Assumption

You have used a spreadsheet before

You understand a simple program/algorithm
Things to Think About

What are the data objects in a Spreadsheet program?

How does the interface differ from the other applications we have seen so far?

What are the efficiencies/deficiencies of the interface?

When would I use a spreadsheet?

How does a spreadsheet compare to a database?

How would macros be useful in other applications we have seen so far?
The Copy Pearl

Keep copies of your old stuff around
   in well-named files and folders
   works better if you comment what you do
Lurk on news groups and capture examples as they come by
Check out what comes with the app
Build a list of what examples can be found where
Application Scripting in Excel

What is “(intra-)application scripting”?
- a way to automate lengthy manual procedures
- a way to customize / extend an application
- a way to create a different interface
  “programming in the small”

Recording a script
- “record” a manual procedure (once)
  and “play it back” many times
  sort of like a player piano
- record a script to learn how to do something
  look up terms in the script as necessary
  “anything you can do manually, you can script” — & more

Edit a recorded script
- to make it more general
- to eliminate unnecessary steps
- to wrap a loop around it
- to do something similar
Why Scripting in CS 200?

Most major apps have some sort of scripting

   eg Word, Excel, Photoshop, FileMaker...

   eg system-level macro languages

      hooked to key presses or menu selection (eg iKey, UI Actions)

   or sometimes to user-defined palettes

      to move data between apps & tell those apps how to process the data (eg AppleScript, VBA)

   eg JavaScript in HTML pages, DreamWeaver, Acrobat...

   — in fact, some have more than one!

   — Photoshop supports three (or four, depending on how you count)

Scripting can save you a LOT of work
### Automating Spreadsheet Creation — Format with a Macro (1)

#### Sheet 1: Format with a Macro

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1B</td>
<td>1C</td>
<td>1D</td>
<td>1E</td>
<td>1F</td>
<td>1G</td>
<td>1H</td>
<td>1I</td>
<td>1J</td>
<td>1K</td>
<td>1L</td>
<td>1M</td>
<td>1N</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Very Good to Excellent</td>
<td>19</td>
<td>17</td>
<td>14</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>10</td>
<td>13</td>
<td>9</td>
<td>18</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Average</td>
<td>0</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>11</td>
<td>7</td>
<td>5</td>
<td>4</td>
<td>6</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Unsatisfactory</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Don't Know</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Total</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>17</td>
<td>19</td>
<td>18</td>
<td>19</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Very Good to Excellent</td>
<td>100%</td>
<td>89%</td>
<td>74%</td>
<td>89%</td>
<td>89%</td>
<td>37%</td>
<td>53%</td>
<td>47%</td>
<td>68%</td>
<td>50%</td>
<td>95%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Average</td>
<td>0%</td>
<td>11%</td>
<td>26%</td>
<td>11%</td>
<td>11%</td>
<td>58%</td>
<td>37%</td>
<td>29%</td>
<td>21%</td>
<td>33%</td>
<td>0%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Unsatisfactory</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>11%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Don't Know</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>5%</td>
<td>0%</td>
<td>24%</td>
<td>11%</td>
<td>17%</td>
<td>5%</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Total</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Notes
- Overall rating
- Marking is done with attention to fairness
- Assignments are returned in a timely manner
- Deals with problems or discrepancies in marking in an acceptable manner
- Ability to challenge you to learn
- Comments or lab discussions on assignments are constructive and helpful
- Lab attendance is regular and punctual
- Prepared and organized for the labs
- Knowledgeable about the course and lab material
- Approachable and helpful
Automating Spreadsheet Creation — Format with a Macro (2)

The macro

```
' ' Rule_Left_and_Bottom Macro
' Macro recorded 10/12/95 by John C. Beatty
'
Sub Rule_Left_and_Bottom()
    Selection.BorderAround Weight := xlThin, ColorIndex := xlAutomatic
    Selection.Borders(xlRight).LineStyle = xlNone
    Selection.Borders(xlTop).LineStyle = xlNone
End Sub
```

Note the comments, introduced by the character ' — anything from there to the end of the line is ignored
(add your own to remind yourself later of things you figure out)

This example illustrates speeding spreadsheet development

macros are easy to read & usually you can RECORD what you want to do, or something close to it, and just edit the recording

look up terms you don’t know with online help (in the VBE environment)

eg select a term like ColorIndex and press the help key
Automating Use Of A Spreadsheet — Sort Marks

This illustrates speeding the use of a spreadsheet
Sub Sort_By_IDnumber2()
    Range("B3:D14").Select
    Selection.Sort
        Key1 := Range("B3"),
        Order1 := xlAscending,
        Header := xlGuess,
        OrderCustom := 1,
        MatchCase := False,
        Orientation := xlTopToBottom
    Range("A1").Select
End Sub
Sub Sort_By_Name2()
    Range("B3:D14").Select
    Selection.Sort
        Key1 := Range("C3"),
        Order1 := xlAscending,
        Header := xlGuess,
        OrderCustom := 1,
        MatchCase := False,
        Orientation := xlTopToBottom
    Range("A1").Select
End Sub

“_” means “the statement continues on the next line”

It’s pretty easy to guess what each piece of the Selection.Sort statement does, right?
Sub Sort_By_Mark2()
    Range("B3:D14").Select
    Selection.Sort
        Key1 := Range("D3"),
        Order1 := xlDescending,
        Header := xlGuess,
        OrderCustom := 1,
        MatchCase := False,
        Orientation := xlTopToBottom
    Range("A1").Select
End Sub
The macro

```
Function FtoC( fTemp )
    FtoC = (fTemp - 32) * 5 / 9
End Function
```

illustrates extending an application by means of a macro

Note the use of “Function” instead of “Sub”

“functions” return a value (the value assigned to their name)

“subroutines” don’t — they just “do something”

FtoC can be used anywhere a built-in Excel function can be used

See also “Marks to Grades” in Week 7 / Files for Lecture: Excel Macros
Making a spreadsheet look like a hand-built app
Selecting Macros... opens the dialog shown above right

Note the “Record New Macro...” menu item
Editing a Macro

To edit a macro

- click the Edit button in the Macros dialog
- or select “Visual Basic Editor” from the menu shown on the previous slide

Note the new menu bar, & especially the View menu in it

All the active menu items are interesting
The VBE Environment (The Big Picture)
Excel vs the Visual Basic Editor

A bit like two programs wrapped as one
— one dock icon, but with different menu bars and different windows/palettes

in the VBE, click on a spreadsheet to “flip back to Excel”
(or select “Excel > Close and Return to Microsoft Excel” !)

in Excel, click on a Code window to “flip back to the VBE”
(or select “Tools > Macros > Visual Basic Editor”)

Excel Scripting
Excel’s Scripting Environment

The Project Explorer window

Recorded macros go into “Modules”

“Sheets” & “ThisWorkbook” can hold macros, too

Double-click any of the three to edit its macros

Suggestion: keep all your macros in modules
—otherwise you must refer to them as Sheet1.macroName, etc

The Properties window

Click on one of the entries in the Projects window

Its “properties” are displayed in the Properties window

You can change many of them; you won’t need to for the assignment.
Cell References in Excel Worksheets & Macros

Before row/column insertions

After row/column insertions

The macros

MyLiteralTotal()
MyNamedTotal()

after inserting a new column
at the left edge of the spreadsheet
— notice which total is correct!
 — and why!

So ... when you insert/delete rows/columns

Excel updates all cell references in a worksheet,
literal references (“E43”, “$R$13”), range definitions, etc,
but the text of macros is untouched

The moral ... you nearly always want to use named ranges in macros
Consider the Sort Marks example with these macros:

Sub Sort_By_Name()
    Range("B3:C14").Select
    Selection.Sort
        Key1 := Range("B3"), _
        Order1 := xlAscending, _
        Header := xlNo, _
        OrderCustom := 1, _
        MatchCase := False, _
        Orientation := xlTopToBottom
    Range("A1").Select
End Sub

Sub Sort_By_Mark()
    Range("B3:C14").Select
    Selection.Sort
        Key1 := Range("C3"), _
        Order1 := xlDescending, _
        Header := xlNo, _
        OrderCustom := 1, _
        MatchCase := False, _
        Orientation := xlTopToBottom
    Range("A1").Select
End Sub

Suppose we added a student, or moved the list. Would the macros still work properly?
Use a named range instead of explicit literal cell references

What if I want to add a new student?
   adding in the middle of a named range
   adding to the end of a named range
   use a final blank row?
   would the average be correct?

This is another instance of “indirection”
Relevant Programming Concepts

Variables and declarations

Assignment statements

Sequential execution

Loops
  - initialization
  - termination test
  - changing the loop control variable(s)

If–then–else statements

Subroutines and functions
  - parameters (aka arguments)

Debugging
  - interactive source-level debugging
“Objects” in Excel — how to name objects on a spreadsheet

Excel’s debugger

Conveniently triggering macro execution

Cell references
  in worksheets
  in macros
The Assignment For This Week

Given the raw data, duplicate the functionality of this spreadsheet
This Week’s Assignment is “Layered”

Trivial formulas
  eg for Actual Balance

Simple formulas
  eg for Statement Balance or Next Transaction Number

Not quite so simple formulas
  eg for the CD Charge cell for US$ purchases

Simple macros
  eg for scrolling, sorting, or filtering

Not quite so simple macros
  eg for making new entries

Note that you can create all of these macros by recording them, you do not need to use VBA
Strategy For The Assignment

Start with the simple stuff

and proceed in stages towards the more difficult

Test as you go

Make a copy of your spreadsheet at each successful stage

eg, when you have the simple formulas working,

ie, squirrel a copy of that away
in case you totally muck up the next step
and want to start over on it
(remember “backups”?)
“Objects” in Excel Macros

Just about everything you see in a workbook is an “object”

There is a natural hierarchy to objects

You can have multiple instances of everything except the Application

These are Excel’s “application data objects” (remember the Model pearl?)

A “module” is for holding macros (aka “scripts”)

think of it as a document containing multiple macro definitions

in the VBE: Insert > Module and type

or ... a module is created automatically when you record a macro

See Chapter 5 of The Microsoft Excel Visual Basic User’s Guide on “Working with Objects in Visual Basic” (pp 65–84), which is reprinted in the course notes, for more on objects in Excel.
Objects

Objects have “properties” like “Color,” “Value,” “Font” that you can read or set in macros.

- sometimes a property is just a value (eg 2)
- sometimes a property is another object

Objects have “methods” like “ClearContents,” “Sort,” “Worksheets” that cause something to be done to or with the object.

- a “method” is just another kind of built-in function or subroutine that does something to or with the object it’s attached to.

Methods often have “parameters” (aka “arguments”)

- information they need
- just like built-in Excel functions

eg C4, ForEx, and 2 in VLOOKUP( C4, ForEx, 2 )

“Member” is programmer-speak for both properties and methods.

All objects of a particular kind are a “class” of objects.
Especially Useful Properties

Application.ActiveWorkbook
  you can just write ActiveWorkbook
  for the frontmost workbook

Application.ActiveSheet
  you can just write ActiveSheet
  for the worksheet whose tab was last clicked

Application.ActiveCell
  you can just write ActiveCell
  for the currently selected cell

And usually you can omit Application.ActiveSheet
  eg Range... instead of Application.ActiveSheet.Range...
  eg Selection... instead of ...

How do you find out about objects, properties and methods?
  record a macro, then highlight a method or property name & press the help key
  ie if you see something and wonder about it
  use the "object browser"
  if you want to go looking to see, for example, if worksheets have a particular property

Sub Sort_By_Mark2()
  Range("B3:D14").Select
  Selection.Sort                   _
      Key1        :=        ("D3"), _
      Order1      :=      xlDescending, _
      Header      :=      xlGuess, _
      OrderCustom :=       1, _
      MatchCase   :=      False, _
      Orientation :=  xlTopToBottom
  Range("A1").Select
End Sub
### Watching Your Macros Run

#### Option Explicit
- Require declaration of variables

#### Private LastRow As Integer
- For use both by MakeTable and by ClearTable

#### Sub MakeTable()
- Start, Stop & Number are defined in TableSheet

| Dim sVal As Integer | To hold the first °F in the table |
| Dim eVal As Integer | To hold the last °F in the table |
| Dim nVal As Integer | To hold the number of entries in the table |
| Dim row1 As Integer | To hold the number of the top row in the table |
| Dim rowN As Integer | To hold the current row number |
| Dim incr As Double | To hold the change in °F from one row to the next |
| Dim fTmp As Double | To hold the temperature in Farenheit |
| Dim cTmp As Double | To hold the temperature in Centigrade |

```
sVal = Range("Start").Value  ' Fetch the value of sVal from the worksheet
eVal = Range("Stop").Value   ' Fetch the value of eVal from the worksheet
nVal = Range("Number").Value ' Fetch the length of the table from the worksheet
incr = (eVal - sVal) / (nVal - 1) ' Compute the difference between successive entries
```

```
row1 = 3 ' Top row of the conversion table
rowN = row1  ' Initialize the loop control variables
fTmp = sVal
```

```
Do While rowN < row1 + nVal
    cTmp = FtoC(fTmp)  ' Write a row in the table
    Cells(rowN, 2).Value = fTmp
    Cells(rowN, 3).Value = cTmp
    Cells(rowN, 2).NumberFormat = "0.00"  ' Format the row just written
    Cells(rowN, 3).NumberFormat = "0.00"
    fTmp = fTmp + incr  ' Increment the loop control variables
    rowN = rowN + 1  ' Row #'s increase as you go DOWN the screen
```

---

This dot marks a “breakpoint” 

---

This dot marks a “breakpoint”
Debugging (The Big Picture)

Private LastRow2 As Integer
    ' For use both by MakeTable and by ClearTable

Sub MakeTable2()
    ' Start, Stop & Number are defined in MainSheet
    Dim sVal As Integer
    Dim eVal As Integer
    Dim nVal As Integer
    Dim row1 As Integer
    Dim rowN As Integer
    Dim incr As Double
    Dim fTmp As Double
    Dim cTemp As Double

    sVal = Range("Start2").Value
    eVal = Range("Stop2").Value
    nVal = Range("Number2").Value
    row1 = 3
    rowN = row1
    incr = (eVal - sVal) / (nVal - 1)

    Do While rown < row1 + nVal
        rown = rown + 1
        fTmp = sVal
        cTemp = FtoC(fTmp)
        Cells(rown, 1).Value = rown
        Cells(rown, 2).Value = fTmp
        Cells(rown, 3).Value = cTemp

        If cTemp < 32 Then
            Cells(rown, 2).Font.ColorIndex = 5 ' Blue
        ElseIf cTemp > 60 Then
            Cells(rown, 2).Font.ColorIndex = 3 ' Red
        Else
            Cells(rown, 2).Font.ColorIndex = 4 ' Green
        End If

        fTmp = fTmp + incr
        rown = rown + 1
    Loop

    LastRow2 = rown
    ' Save for use by ClearTable2()
End Sub

Sub ClearTable2()
    Range(Cells(3, 2), Cells(LastRow2, 3)).Value = ""
    Range(Cells(3, 2), Cells(LastRow2, 3)).Font.ColorIndex = 1 ' Black
End Sub

Make

Use the Make Table button to create a Fahrenheit-to-Celsius conversion table in columns B and C, beginning in row 3 and extending as far as is necessary. Use the Clear Table button to erase a previously-created table. Specify the first and last Fahrenheit temperatures as cells F2 and F3 above; specify the number of entries in the table via cell F4.
Using the Debugger

Set a breakpoint

by clicking in the left margin
to halt a macro when it gets to that statement

“clear” the breakpoint by clicking again in the left margin

Use

? variableName <return> in the Immediate Window to display the value of the variable

Use

Debug > Step Into (does dive into functions or subroutines)
Debug > Step Over (doesn’t dive into functions or subroutines)
to execute one statement and stop again

Use

Run > Continue
to turn the macro loose

The Visual Basic Editor’s Debug toolbar has buttons for all of these
(View > Toolbars... > Debug)
To run a macro, delete a macro, or edit a macro, you can attach a macro to a key (or key combination) on the keyboard. And via the Options... button, you can add a new macro or change an existing one's description.
Attaching a Macro to a Spreadsheet Button

Draw a button after selecting on the Forms toolbar

Ctrl-click on the button to open the Assign Macro dialog

Select and OK the desired macro
Strategies For Working With Macros

Record

• to learn how to do something
• to build a macro faster

Edit recorded macros to customize / elaborate them

Use on-line help and the object browser
to find object names, properties, & methods that you need

Experiment

• with toy macros & documents
• eg the workbooks used in lecture (see Learn Week 6: Application Scripting (Excel) / Files For Excel Scripting)

Use the debugger

• to understand what your macros are doing!

Save everything you do, and include comments (they start with the single quote character ' )
The Assignment For This Week

Given the raw data, duplicate the functionality of this spreadsheet.