CS 200

Lecture 06
Excel Scripting
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
Please read and highlight the assignment and course notes before lab

Assignment 6 is due Monday June 22 at 11:59 pm
   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
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?
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)**

![Spreadsheet Image]

<table>
<thead>
<tr>
<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>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Format with a Macro**

|                      | B   | C   | D   | E   | F   | G   | H   | I   | J   | K   | L   | M   |
|----------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| Very Good to Excellent | 19  | 17  | 14  | 17  | 17  | 7   | 10  | 8   | 13  | 9   | 18  |
| Average              | 0   | 2   | 5   | 2   | 11  | 7   | 5   | 4   | 6   | 0   |
| Unsatisfactory       | 0   | 0   | 0   | 0   | 0   | 0   | 2   | 0   | 0   | 0   | 0   |
| Don't Know           | 0   | 0   | 0   | 0   | 0   | 1   | 0   | 4   | 2   | 3   | 1   |
| Total                | 19  | 19  | 19  | 19  | 19  | 19  | 17  | 19  | 18  | 19  |

**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
Comments or lab discussions on assignments are constructive and helpful
Ability to challenge you to learn
Presentations to the class are clear and effective
Lab attendance is regular and punctual
Prepared and organized for the labs
Approachable and helpful
Knowledgeable about the course and lab material

**Before**

**After**

<table>
<thead>
<tr>
<th></th>
<th>Sum=0</th>
<th>SCRL</th>
<th>CAPS</th>
<th>NUM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

CS 200 Spring 2020

02 – Styles
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

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>ID Number</td>
<td>Name</td>
<td>Mark</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>92016624</td>
<td>Augustine</td>
<td>72</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>94181906</td>
<td>Brenda</td>
<td>62</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>93007999</td>
<td>Daina</td>
<td>71</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>94087920</td>
<td>Daniel</td>
<td>68</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>93039714</td>
<td>Flora</td>
<td>77</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>91019942</td>
<td>Frieda</td>
<td>41</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>93033490</td>
<td>George</td>
<td>66</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>95141963</td>
<td>James</td>
<td>57</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>94128746</td>
<td>Joseph</td>
<td>60</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>94087558</td>
<td>Sudi</td>
<td>91</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>94013896</td>
<td>Taonga</td>
<td>94</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>94047282</td>
<td>Walter</td>
<td>83</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>Average</td>
<td>70.2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Sort By:

- By ID #
- By Name
- By Mark
- Add Student
Sort Marks — By Name

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
Extending Excel — F to C Conversion

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 6 / 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”)

---

CS 200 Spring 2020
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.
**“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
Referring to an object or property

EG: Application.Worksheets("Visa")
    .Range("Stmt").Font.Name = "Helvetica"

Here

application is an object
worksheets() is a method (that returns a Worksheet object)
range() is a method (that returns a Range object)
font is a property (whose value is a Font object)
name is a property (a string)

The () surround a method’s parameters — information the method needs to do its thing.

You can use the Properties Window to set an object’s properties manually.
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
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
Cell References in Excel Macros

Consider the Sort Marks example with these macros:

Suppose we added a student, or moved the list. Would the macros still work properly?

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
Using 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”*

```vbnet
Sub Sort_By_IDnumber2()
    Range("theData3").Select
    Selection.Sort _
        Key1 := Range("theData3").Range("A1"), _
        Order1 := xlAscending, _
        Header := xlGuess, _
        OrderCustom := 1, _
        MatchCase := False, _
        Orientation := xlTopToBottom
    Range("A1").Select
End Sub
```
Watching Your Macros Run

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
    ' 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 fTemp As Double
    ' To hold the temperature in Fahrenheit
    Dim cTemp As Double
    ' To hold the temperature in Celsius

    sVal = Range("Start2!").Value
    ' First "F" in the table
    eVal = Range("Stop2!").Value
    ' Last "F" in the table
    nVal = Range("Number2!").Value
    ' Number of entries in the table
   incr = (eVal - sVal) / (nVal - 1)
    ' Difference between successive entries

    row1 = 3
    ' Top row of the conversion table

    rowN = row1
    ' Initialize the loop control variables
    fTemp = sVal
    Do While rowN < row1 + nVal
        cTemp = FtoC(fTemp)
        ' Write a row in the table
        Cells(rowN, 1).Value = fTemp
        Cells(rowN, 3).Value = cTemp
        Cells(rowN, 2).NumberFormat = "0.00"
        ' Format the row just written
        Cells(rowN, 3).NumberFormat = "0.00"

        If fTemp < 32 Then
            ' Control the colour of this cell
            Cells(rowN, 3).Font.ColorIndex = 5
            ' Blue
        ElseIf fTemp > 60 Then
            Cells(rowN, 3).Font.ColorIndex = 3
            ' Red
        Else
            Cells(rowN, 3).Font.ColorIndex = 4
            ' Green
        End If

        fTemp = fTemp + incr
        ' Increment the loop control variables
        rowN = rowN + 1
        ' Row #s increase as you go DOWN the screen

        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

' Microsoft Excel Objects
' Sheet1 (1) Sheet1
' Sheet2 (TableCode)
' Sheet3 (TableCode2)
' Workbook
' NameBox
' x1Code
' TableCode
' MakeTable2
' VBAProject (Template.xsl)
' VBAProject (stowres.xsl)
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)
Attaching a Macro to a Spreadsheet Button

Draw a button after selecting Button on the Developer Ribbon

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

Select and OK the desired macro
Putting It Together: Tools > Macro > Macros...

To

run a macro
delete a macro
edit a macro

And via the Options... button
you can attach a macro to a key (or key combination) on the keyboard.
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
The Assignment For This Week

Given the raw data, duplicate the functionality of this spreadsheet

![Spreadsheet Image]

---

**Accounting Solution.xls**

- **S** Last Statement Received: 252
- **Sort**
- **Sort**
- **Bottom**

<table>
<thead>
<tr>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>09/01/20</td>
<td>Office Depot $7.83</td>
</tr>
<tr>
<td>09/10/20</td>
<td>Buy in the mail on three pairs of pants $5.95</td>
</tr>
<tr>
<td>09/11/20</td>
<td>Chapters, Waterloo $28.75</td>
</tr>
<tr>
<td>09/12/20</td>
<td>Sebys $4.79</td>
</tr>
<tr>
<td>09/13/20</td>
<td>Lunch with Debbie &amp; Lori (was 36.97) $39.99</td>
</tr>
<tr>
<td>09/14/20</td>
<td>iTunes Audio Books $29.69</td>
</tr>
<tr>
<td>09/15/20</td>
<td>Sebys $1.99</td>
</tr>
<tr>
<td>09/16/20</td>
<td>Dataart $15.79</td>
</tr>
<tr>
<td>09/17/20</td>
<td>Two Tilby mesh pop short sleeve shirts $20.75</td>
</tr>
<tr>
<td>09/18/20</td>
<td>K. W. Surplus $26.75</td>
</tr>
<tr>
<td>09/19/20</td>
<td>ES: Take Control $15.75</td>
</tr>
<tr>
<td>09/20/20</td>
<td>Upgrad to Script Debugger v4 $25.97</td>
</tr>
<tr>
<td>09/21/20</td>
<td>Late Night Software $28.75</td>
</tr>
<tr>
<td>09/22/20</td>
<td>Shopper's Drug Mart $43.53</td>
</tr>
<tr>
<td>09/23/20</td>
<td>Sebys $5.95</td>
</tr>
<tr>
<td>09/24/20</td>
<td>Paymet $2,172.39</td>
</tr>
<tr>
<td>09/25/20</td>
<td>Chapters Online $26.75</td>
</tr>
<tr>
<td>09/26/20</td>
<td>Take in the mail on three pairs of pants $27.95</td>
</tr>
<tr>
<td>09/27/20</td>
<td>CDN Canada $71.60</td>
</tr>
<tr>
<td>09/28/20</td>
<td>MMM Food Shop $28.95</td>
</tr>
<tr>
<td>09/29/20</td>
<td>Walley wateroo theatrs $2.99</td>
</tr>
<tr>
<td>09/30/20</td>
<td>Canadian Tire $38.48</td>
</tr>
<tr>
<td>10/03/20</td>
<td>Active State $36.47</td>
</tr>
<tr>
<td>10/04/20</td>
<td>CostcoTech $34.95</td>
</tr>
<tr>
<td>10/05/20</td>
<td>Sebys $5.95</td>
</tr>
<tr>
<td>10/06/20</td>
<td>PathFinder $14.76</td>
</tr>
<tr>
<td>10/07/20</td>
<td>KAGI / Default Folder $17.21</td>
</tr>
<tr>
<td>10/08/20</td>
<td>KAGI / Skyscraper Master $24.62</td>
</tr>
<tr>
<td>10/09/20</td>
<td>Sebys $24.35</td>
</tr>
<tr>
<td>10/10/20</td>
<td>Sebys $23.95</td>
</tr>
<tr>
<td>10/11/20</td>
<td>Sebys $28.75</td>
</tr>
<tr>
<td>10/12/20</td>
<td>OpenDoor $59.00</td>
</tr>
<tr>
<td>10/13/20</td>
<td>MOS Laboratories $30.00</td>
</tr>
<tr>
<td>10/14/20</td>
<td>Airways Transit $69.00</td>
</tr>
<tr>
<td>10/15/20</td>
<td>Airways Transit for the annual trip to Oregon $1,840.00</td>
</tr>
<tr>
<td>10/16/20</td>
<td>TomLee Music $97.06</td>
</tr>
<tr>
<td>10/17/20</td>
<td>Sebys $42.47</td>
</tr>
<tr>
<td>10/18/20</td>
<td>2nd Binder $30.00</td>
</tr>
<tr>
<td>10/19/20</td>
<td>10 Months $51.94</td>
</tr>
<tr>
<td>10/20/20</td>
<td>10 Months $51.94</td>
</tr>
<tr>
<td>10/21/20</td>
<td>Next Data Entry Sequence Number $150</td>
</tr>
</tbody>
</table>

---

**CSV Format**

- **Date**
- **Description**
- **Amount**

---

**Excel Functionalities**

- **Sort**
- **Filter**

---

**Additional Information**

- Upgrade from v2 -> v3
- Take Control of Fonts & Take Control of Font Problems...
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”?)
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 ' )