



Week 3:

- Working with Office applications
 - File opening with Office applications
- Working with Loops
 - While, Do/Until
- Looping over Excel cells
- Functions and Sub Programs



Wscript Object

- When a script file is launched, the WScript object is instantiated automatically.
- In *HundredthDigit.vbs* the *Arguments* property was used (it is a collection object):
`Set Args = WScript.Arguments`
- Other useful properties of WScript that can be accessed (are variables): `FullName`, `Name`, `Path`, `ScriptFullName` (*do not use "set" here*)



Wscript.CreateObject method

- The CreateObject method of WScript can be used to create useful objects
- See pp 134-48 of Lomax for examples
- The *NetWork* object has useful information about the computer and user (p. 136)
- This object created by a statement like:
`Set oNet = WScript.CreateObject("Wscript.Network")`

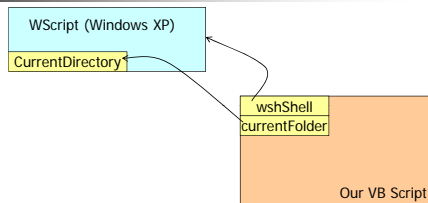


SetInitialFolder.vbs

- The WScript.Shell
 - Provides information on “shell” environment
 - E.g., current working folder
 - Reference: **A1** or text by Lomax
 - *Windows Script Host*
 - *References – Objects – WshShell Object*
 - *look under properties ... for “CurrentDirectory”*



Current Working folder (Directory)



```
Set wshShell = WScript.CreateObject("WScript.Shell")
currentFolder = wshShell.CurrentDirectory
```



Activity #5

- Work with partner

Open existing document

Two ways that use a similar file selection and open dialog box:

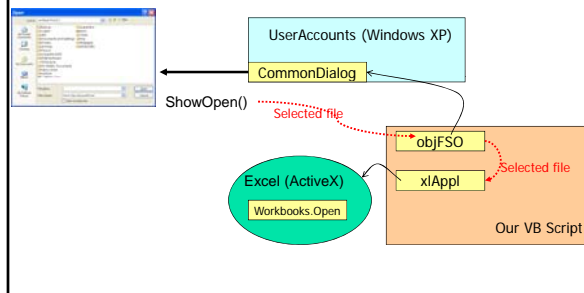
- "Common Dialog" or (OpenExcelWithCommon.vbs)

```
Dim objFSO
Set objFSO = CreateObject("UserAccounts.CommonDialog")
objFSO.Filter = ..
```

- File Dialog from within Excel (OpenExcelFromExcel.vbs)

```
Dim xlApp, dlgOpen
Set xlApp = CreateObject("Excel.Application")
Set dlgOpen = xlApp.Application.FileDialog(msoFileDialogOpen)
```

Using CommonDialog



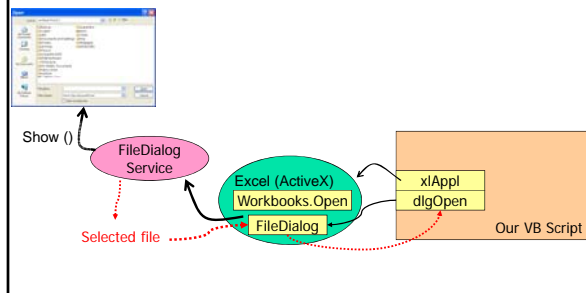
OpenExcelWithCommon.vbs

- "UserAccount.CommonDialog" ActiveX
 - Simple file selection dialog
 - Create the object, (`CreateObject()`)
 - Set Filter (what kinds of files)
 - Set InitialDir (where to start looking)
 - Call **ShowOpen()**

More about CommonDialog

- Filter = "desc1|*.ext1|desc2|*.ext2"
 - Ordered pairs of: Description | *.file extension
 - Notice the "|" is important
- FilterIndex = 1
- ShowOpen
 - Returns 0 if failed (*false*)
 - Returns -1 if ok (*true*)
- FileName is the file selected

Using FileDialog



OpenExcelFromExcel.vbs

- Start the Excel ActiveX services first!
- Access the FileDialog() object in Excel application
 - Notice we need to specify what kind of dialog we need
 - The *msoFileDialogOpen* is a constant defined in our script!
 - We will come back to this in a couple of slides.
- FileDialog is a little more elaborate
 - Title
 - Filters.Add "Desc", "File Extension", Index
 - AllowMultiSelect
- Otherwise, this application is very similar to the one using CommonDialog



What are ActiveX Services?

- “Standardized Interface” from one program to another
- A Set of functions you can call from one program to another
- How to find the special constants. We need the value of *msoFileDialogType* corresponding to dialog type open.



Excel Macro Trick: ... show me the code!

- View->Macros->Record Macro
 - Name: Test1
 - Store Macro in: "This Workbook"
 - This corresponds to: xl.Application
 - Now do something fun e.g.
 - Select a range of cells
 - Change color of rows, etc. (Home -> Cells -> Format)
- View->Macros->Stop Recording
- Go to: Visual Basic Editor Window
 - View -> Macros ->View Macros
 - Select Test1
 - Hit the "Edit" button
 - We can see the VBScript code!!
 - Since we are *not* inside the MS Excel application, we need to append xlApp.Application for every command in our VBScript programs.



Now: figuring out VBScript Code

- Open MS Excel, make sure *no* document is opened
- Figure out how to create a new worksheet
 - **View -> Macros -> Record Macro**
 - Name: test
 - Store macro in: Person Macro Worksheet
 - So we can examine how to create a new worksheet
 - Officebutton->New->Blank workbook
 - Click on Cell(B3), type a number
 - Click on Cell(B4), type a second number
 - Click on Cell (D7) type "=sum(B3:B4)
 - View ->Macros->Stop Record Macro
- Window->unhide->PERSONAL.XLS
 - This is a dummy front for storing the "test" macro
- View->Macros->view macros
 - Select "test" and hit edit
- Examine the VBScript code!



Row/Column or Range

- Range("B3")
 - The quote in "B3" is important!
 - B is column (vertical), 3 is row (horizontal)
- R1C1 Notation:
 - Absolute: R3C2 (3rd row, 2nd col) or "B3"
 - Relative from B3: R[-2]C[1] is "C1"
- How to find out more?
 - Look into the manual at **D**



More about Cells and Ranges

- **D** -> Microsoft Excel Object Model
 - Look for "**Range**" object
- Click on "Application Object"
 - we do this because we see xlAppI.Cells and xlAppI.Range used in *OpenExcel.vbs*
- In Application Object page
 - Lists of "See Also" "Properties" "Methods", etc.
 - Under Properties->Cells
 - Cells property returns a range object that is entire set of cells on spreadsheet.
 - .Cells (rn,cn) returns *only* the particular cell in row-rn and column-cn
Worksheets("Sheet1").Cells(5, 3).Font.Size = 14



More about Cells and Ranges

- In Application Object page
 - Under Properties->Range Property
 - .Range (cell1)
 - Worksheets("Sheet1").Range("A1").Value = 3.14159
 - Worksheets("Sheet1").Range("A1").Formula = "=10*RAND()"
 - .Range (cell1;cell2)
 - For Each c in Worksheets("Sheet1").Range("A1:D10")
 - .Range (cell1,cell2)
 - Worksheets("Sheet1").Range(Cells(1, 1), Cells(5, 3))._Font.Italic = True



Where are we?

- We have seen two examples of ActiveX services:
 - UserAccount.CommonDialog
 - Applications.Excel
- We have Worked with the services provided by ActiveX controls
 - FileDialog (from UserAccounts.CommonDialog)
 - Cells/Range, etc. (from Excel)
- Seen two ways of opening an Excel File
- Seen how to set current directory for file search
- Seen how to write and read in particular cells of Excel File.

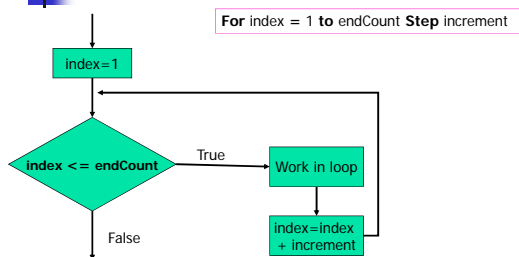


Loops

- Known number of iterations
- Use For-Loop (*Lomax, p. 315*)
 - Looping forward
 - For index = 1 To endCount Step increment
 - Do your work
 - Next
 - Looping backward
 - For index = endCount To 1 Step decrement
 - Do your work
 - Next



Flowchart – For Loop





Unknown number of iterations

- Loop **until** something becomes true ...
- Control variable
 - Figure out what it is! You need to set it up.
- Use positive logic
 - Use variable with names that reflect "positive" results
 - Idea: when end of loop condition is reached, we might "assign" a "True" value to the control variable



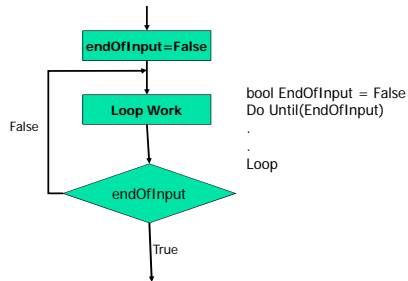
Do until

- Examples:
 - E.g. when looping to read until end of input use:


```
EndOfInput
Bool EndOfInput = False (we are not starting at the end)
Do Until (EndOfInput)
  Read from input
  If (end of input is reached) Then
    EndOfInput = True
  EndIf
Loop
```



Do until flow chart



Unknown number of iterations

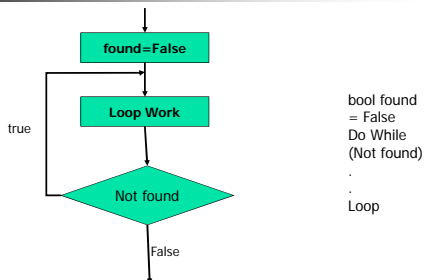
- Loop **While** something is true ...
- Control variable
 - Figure out what it is! You need to set it up.
- Use positive logic
 - Use variable with names that reflect "positive" results
 - Idea: when end of loop condition is reached, we might "assign" a "False" value to the control variable
 - We could test on "Not Found" if we are searching for an object in a list.

Loop While

- Examples:
 - E.g. when looping to find something, control variable could be named: found

```
bool Found = False (we have not found it yet!)
Do While (Not Found)
  Look for something
  If (find the thing) Then
    Found = True
  EndIf
WARNING: infinite loop if never found!!
Loop
```

Loop while flow chart





1. Loop.vbs

- Simple Do/While Loop
 - Single entry/exit point
 - Exit While statement: should **NEVER** use this statement!
 - Best: to have one control variable
 - Name variable for *positive* thinking
 - E.g. *shouldQuit* or *shouldContinue* are both GOOD
 - ShouldQuit – initialize to false and loop till shouldQuit is true
 - ShouldContinue – initialize to true and loop till shouldContinue is false
 - E.g. *ShouldNotQuit* or *ShouldNotContinue* are confusing!
 - shouldNotQuit – initialize to true, and loop till shouldNotQuit is false (very confusing!) involves double negative!!



1. Loop.vbs

- How to determine what is even?
- Loop is good for
 - Accumulate:
 - Count
 - Applying the same operation on different data items
 - Examine them item-by-item
 - The *mod* operator (remainder)
- Assigning boolean results
 - *shouldQuit* = (*userInput* = "q")



How to debug Loop.vbs

- **ALWAYS:** ask yourself when will my loop end!
- If infinite loop, script never ends!
 - Kill from task manager!!
- MsgBox
 - at *strategic* places in the code
 - Print out *useful* information
- Where to put MsgBox?
 - "binary search"
 - Or, simple put in a bunch of MsgBox statements
- What to print out?
 - Must print distinct message!
- Remember to **comment out** debug messages before submitting your program to me!



2. LoopOverExcelCells.vbs

- Excel: seeing for the first time
 - `Set activeSheet = xlApp.Worksheets("Sheet1")`
 - `Worksheets()` function
 - Use of `activeSheet`
 - `Cell(row,column)`
 - Column 1 is "A", 2 is "B", etc.
- Do/Loop-Until
 - Again, one loop control variable, assign to true when ready to quit
 - Use when do something at least once
- Example of For Loop
 - Know how many rows to loop over



LoopOverCollection.vbs

- `FileSystemObject` used to obtain a collection object of the files in the working directory
- For Each Next
 - `ForEach` loop used to loop over all items within the collection object.



Working with Excel Activity 6

- For the given Scores.xls
- How can we write a script to find average of all mp scores for all students? Do design at this point only.
- After discussing the designs, examine my design and implementation.



ExampleSubAndFunc.vbs

- Modularize solution design
 - Solution reads like English statements!
- Code Re-use
- Notice, if your script has Sub-Program or Function, they are simply ignored!
 - VB Script looks for first non-Sub-Program/Function statement and starts execution from there!
- ALWAYS: sub-program and function grouped together!
 - Your "main" program **must** be together
 - Do not scatter your program code in between sub-program and functions
 - Main program completely before or after sub-programs/functions



Sub-Program

- Does not return anything
- Invoke **with** the "Call" Keyword, use ()
 - Call MySubProgram(a, b) – OK
 - Call MySubProgram a, b – Syntax error
- Invoke **without** "Call", don't use()
 - MySubProgram a, b – OK
 - MySubProgram(a, b) – Syntax error
- Use one way, and be consistent e.g.
 - I usually use the "Call" keyword
 - Call MySubProgram (a, b) – This is what I do



Function:

- Function always returns a value
- Syntax: function name returns the value


```
Function ComputeSomething(a, b)
    Dim r
    r = a + b
    ComputeSomething = r
End Function
```

 - Notice we use "**ComputeSomething**" to receive the computed value!
- Invoke with ():
 - Result = ComputeSomething(a, b)
 - () is must!



Comment Block

- Each sub-program/function should start with a comment block (beginning with a "line")
 - E.g. *****
- A Comment block must contain:
 - Function Name
 - Input: Parameters passed in
 - Output: returned parameters
 - Error: Potentially what may cause error, what errors are checked
 - Description: of what the sub-program/function does
