# Do a quick search for a value

= .Find(What:="0" looks in the range where the value is 0, if no value is 0, then *Empty* is returned

Sub FindZeros()

With Range("B1:B16")

Set Rng = .Find(What:="0", LookAt:=xlWhole, LookIn:=xlValues) -- 0

If Not Rng Is Nothing Then

firstaddress = Rng.Address -- $B$3

Do

Rng.Offset(, 1).Value = "LOW"

Set Rng = .FindNext(Rng) -- .36751

Loop While Not Rng Is Nothing And Rng.Address <> firstaddress

End If

End With

End Sub

# Delete row if column 1 is blank

Sub DeleteBlankRows()

lastrow = Cells(Rows.Count, 1).End(xlUp).Row

Dim ent\_row As Range

For i = 1 To lastrow

If IsEmpty(Cells(i, 1)) Then

Set ent\_row = Cells(i, 1).EntireRow

Cells(i, 1).Resize(1, 4).Interior.ColorIndex = 2

ent\_row.Delete

End If

Next i

End Sub

# Delete row if entire row is blank

Sub DeleteBlankRows()

lastrow = Cells(Rows.Count, 1).End(xlUp).Row

Dim ent\_row As Range

For i = 1 To lastrow ' 1 looks like l, this is ONE not an L

Set ent\_row = Cells(i, 1).EntireRow

If Application.WorksheetFunction.CountA(ent\_row) = 0 Then

ent\_row.Delete

End If

Next i

End Sub

# Basic for loop

For i = 1 To 10

Cells(i, i).Value = i

Next i

# Heres a loop that colors columns 1 through 8 green, whenever the value in column 6 is greater than 0

FinalRow = Cells(Rows.Count, 1).End(xlUp).Row

For i = 2 To FinalRow

If Cells(i, 6).Value > 0 Then

Cells(i, 8).Value = "Service Revenue"

Cells(i, 1).Resize(1, 8).Interior.ColorIndex = 4

End If

Next i

# To disable right click

Private Sub Worksheet\_BeforeRightClick(ByVal Target As Range, Cancel As Boolean)

Cancel = True

End Sub

# To trap a calculate. Note if you are on sheet 9, and hit CTL-F9, then active sheet is sheet 3. Hit CTL-ALT-F9 then active sheet is sheet 9.

Private Sub Worksheet\_Calculate()

If ActiveSheet.CodeName = "Sheet9" Then

End If

End Sub

F9: Calculates all worksheets in all open workbooks. Shift+F9: Calculates the active worksheet. Ctrl+Alt+F9: Calculates all worksheets in all open workbooks, regardless of whether they have changed since the last calculation

# Filter by 2 values

Sub SelectTwoCustomers()

Worksheets("NewFilter").Select

Range("A1:A1").AutoFilter Field:=4, \_

Criteria1:="Amazing Shoe Company", Operator:=xlOr, Criteria2:="Mouthwatering Inc."

End Sub

# Filter 3 values

Sub SelectThreeCustomers()

Worksheets("NewFilter").Select

Range("A1:A1").AutoFilter Field:=4, \_

Criteria1:=Array("Tasty Yogurt Corporation", "Cool Saddle Traders", "Agile Aquarium Inc."), \_

Operator:=xlFilterValues

End Sub

# Filter the highest 3 values in column 6

Sub Top6Filter()

Worksheets("NewFilter").Select

Range("A1:A1").AutoFilter Field:=6, \_

Criteria1:="3", \_

Operator:=xlTop10Items

End Sub

# Filter by substring

Sub FilterSearchBox()

Worksheets("NewFilter").Select

Range("A1:A1").AutoFilter Field:=4, \_

Criteria1:="\*at\*"

End Sub

# Get unique values in column C, put in “Customers” worksheet

Sub UniqueCustInPlace()

FinalRow = Cells(Rows.Count, 4).End(xlUp).Row

Range("C1").Resize(FinalRow, 1).AdvancedFilter Action:=xlFilterInPlace, Unique:=True

Range("C1").Resize(FinalRow, 2).Copy Destination:=Worksheets("Customers").Range("A1")

ActiveSheet.ShowAllData

End Sub

# Read csv file in one big gulp

Sub LoadLinesFromCSV()

Dim sht As Worksheet

Dim strtxt As String

Dim textArray() As String

' Add new sheet for output

Set sht = Sheets.Add

' open the csv file

With CreateObject("Scripting.FileSystemObject") \_

.GetFile("c:\temp\sales.csv").OpenAsTextStream(1)

'read the contents into a variable

strtxt = .ReadAll

' close it!

.Close

End With

'split the text into an array using carriage return and line feed

'separator

textArray = VBA.Split(strtxt, vbCrLf)

sht.Range("A1").Resize(UBound(textArray) + 1).Value = \_

Application.Transpose(textArray)

End Sub

# Open text file, fixed width (easier to code this by recording a macro)

The second value in each Array is the XlColumnDataType, 1=xlGeneralFormat, 2=xlTextFormat

Sub OpenText()

ChDir ThisWorkbook.Path

Workbooks.OpenText Filename:="sales.prn", Origin:=437, StartRow \_

:=1, DataType:=xlFixedWidth, FieldInfo:=Array(Array(0, 2), Array(8, 1), Array( \_

17, 3), Array(25, 1), Array(36, 1), Array(46, 1), Array(56, 9), Array(61, 9))

End Sub

# Read text file process line by line

Sub ImportAll()

ChDir ThisWorkbook.Path

ThisFile = "inventory.txt"

' Put a small inventory file out there

WriteInventory 100

Worksheets("Data").Cells.Clear

Open ThisFile For Input As #1

Ctr = 0

Do

Line Input #1, Data

Ctr = Ctr + 1

Worksheets("Data").Cells(Ctr, 1).Value = Data

Loop While EOF(1) = False

Close #1

Worksheets("Data").Select

End Sub

# Combine all xls files in a directory into one xls file

Sub CombineWorkbooks()

Dim CurFile As String, DirLoc As String

Dim DestWB As Workbook

Dim ws As Object 'allows for different sheet types

DirLoc = ThisWorkbook.Path & "\tst\" 'location of files to combine

CurFile = Dir(DirLoc & "\*.xls\*")

Application.ScreenUpdating = False

Application.EnableEvents = False

Set DestWB = Workbooks.Add(xlWorksheet)

Do While CurFile <> vbNullString

Dim OrigWB As Workbook

Set OrigWB = Workbooks.Open(Filename:=DirLoc & CurFile, ReadOnly:=True)

'Limits to valid sheet names and removes ".xls\*"

CurFile = Left(Left(CurFile, Len(CurFile) - 5), 29)

For Each ws In OrigWB.Sheets

ws.Copy After:=DestWB.Sheets(DestWB.Sheets.Count)

If OrigWB.Sheets.Count > 1 Then

DestWB.Sheets(DestWB.Sheets.Count).Name = CurFile & ws.Index

Else

DestWB.Sheets(DestWB.Sheets.Count).Name = CurFile

End If

Next

OrigWB.Close SaveChanges:=False

CurFile = Dir

Loop

Application.DisplayAlerts = False

DestWB.Sheets(1).Delete

Application.DisplayAlerts = True

Application.ScreenUpdating = True

Application.EnableEvents = True

Set DestWB = Nothing

End Sub

# Function to return user name

Private Declare PtrSafe Function WNetGetUser Lib "mpr.dll" Alias "WNetGetUserA" \_

(ByVal lpName As String, ByVal lpUserName As String, \_

lpnLength As Long) As Long

Private Const NO\_ERROR = 0

Private Const ERROR\_NOT\_CONNECTED = 2250&

Private Const ERROR\_MORE\_DATA = 234

Private Const ERROR\_NO\_NETWORK = 1222&

Private Const ERROR\_EXTENDED\_ERROR = 1208&

Private Const ERROR\_NO\_NET\_OR\_BAD\_PATH = 1203&

Function WinUsername() As String

'variables

Dim strBuf As String, lngUser As Long, strUn As String

'clear buffer for user name from api func

strBuf = Space$(255)

lngUser = WNetGetUser("", strBuf, 255)

'if no error from function call

If lngUser = NO\_ERROR Then

'clear out blank space in strBuf and assign val to function

strUn = Left(strBuf, InStr(strBuf, vbNullChar) - 1)

WinUsername = strUn

Else

WinUsername = "Error :" & lngUser

End If

End Function

# Check if user has admin rights

Private Declare PtrSafe Function GetUserName Lib "advapi32.dll" \_

Alias "GetUserNameA" (ByVal lpBuffer As String, nSize As Long) \_

As LongPtr

Private Function UserName() As String

Dim sName As String \* 256

Dim cChars As Long

cChars = 256

If GetUserName(sName, cChars) Then

UserName = Left$(sName, cChars - 1)

End If

End Function

Sub ProgramRights()

Dim NameofUser As String

NameofUser = UserName

Select Case NameofUser

Case Is = "Administrator"

MsgBox "You have full rights to this computer"

Case Else

MsgBox "UserName: " & NameofUser & Chr(10) & "You have limited rights to this computer"

End Select

End Sub

# Check if user has admin rights 2

Private Declare PtrSafe Function WNetGetUser Lib "mpr.dll" Alias "WNetGetUserA" \_

(ByVal lpName As String, ByVal lpUserName As String, \_

lpnLength As Long) As Long

Private Const NO\_ERROR = 0

Private Const ERROR\_NOT\_CONNECTED = 2250&

Private Const ERROR\_MORE\_DATA = 234

Private Const ERROR\_NO\_NETWORK = 1222&

Private Const ERROR\_EXTENDED\_ERROR = 1208&

Private Const ERROR\_NO\_NET\_OR\_BAD\_PATH = 1203&

Function WinUsername() As String

'variables

Dim strBuf As String, lngUser As Long, strUn As String

strBuf = Space$(255) 'clear buffer for user name from api func

lngUser = WNetGetUser("", strBuf, 255)

If lngUser = NO\_ERROR Then

strUn = Left(strBuf, InStr(strBuf, vbNullChar) - 1) 'clear spaces and assign val to function

WinUsername = strUn

Else

'error,give up

WinUsername = "Error :" & lngUser

End If

End Function

Sub CheckUserRights()

'see Module2 for the API calls and the WinUserName function

Dim UserName As String

UserName = WinUsername

Select Case UserName

Case "Administrator"

MsgBox "Full Rights"

Case "Guest"

MsgBox "You cannot make changes"

Case Else

MsgBox "Limited Rights"

End Select

End Sub

# Export data to word

Sub Export\_Data\_Word\_Table()

Dim wdApp As Word.Application

Dim wdDoc As Word.Document

Dim wdCell As Word.Cell

Dim wbBook As Workbook

Dim wsSheet As Worksheet

Dim rnData As Range

Dim vaData As Variant

Set wbBook = ThisWorkbook

Set wsSheet = wbBook.Worksheets("ExportWord")

With wsSheet

Set rnData = .Range("A1:A10")

End With

vaData = rnData.Value 'Add the values in the range to a one-dimensional variant-array.

'Here we instantiate the new object.

Set wdApp = New Word.Application

'Here the target document reside in the same folder as the workbook.

Set wdDoc = wdApp.Documents.Open(ThisWorkbook.Path & "\Test.docx")

'Import data to the first table and in the first column.

For Each wdCell In wdDoc.Tables(1).Columns(1).Cells

i = i + 1

wdCell.Range.Text = vaData(i, 1)

Next wdCell

'Save and close the document.

With wdDoc

.Save

.Close

End With

wdApp.Quit 'Close the hidden instance of MS Word.

Set wdDoc = Nothing 'Release the external variables from the memory

Set wdApp = Nothing

MsgBox "The data has been transfered to Test.docx.", vbInformation

End Sub

# Trap Function Keys

Private Sub Workbook\_Activate()

Application.OnKey "{F9}", "f9handler"

Application.OnKey "{F5}", "printAddIns"

Application.OnKey "^a", "Procedure" ‘ CTL-a

Application.OnKey "+^{RIGHT}", "Procedure" ‘ SHIFT+CTRL+RIGHT

Application.OnKey "^{+}", "Procedure" ‘ CTL-Plus

Application.OnKey "%{A}", "Procedure" ‘ ALT-A

End Sub

# F9 Handler

Private Sub f9handler()

If Range("AutoCalc") = True Then

Application.Calculation = xlAutomatic

Else

Application.Calculation = xlManual

End If

Range("UserName").Value = VBA.Environ("USERNAME")

Application.Calculate

End Sub

# Print Registered dlls

Private Function listXLLs() As Collection

Dim list As New Collection

Dim i As Integer

i = 3

On Error Resume Next 'ignore errors

Dim funcInfo As Variant

For Each funcInfo In ApplicationRegisteredFunctions

If Right(funcInfo, 4) Like ".xll" Then list.Add funcInfo, funcInfo

Cells(i, 8).Value = funcInfo

i = i + 1

Next

Set listXLLs = list

End Function

Private Sub printAddIns()

Dim xllPath As Variant

Dim i As Integer

i = 3

Cells(4, 2).Value = "Running !"

For Each xllPath In listXLLs()

Cells(i, 5).Value = funcInfo

MsgBox (i)

i = i + 1

Next

End Sub

Registering a dll is done through regsvr32 but you can do it in VBA with

ShellExecute 0, "RunAs", "cmd", "/c regsvr32 /u " & """" & FilePath & """", "C:", 0

# File name from Path

Private Function baseName(ByVal fileSpec As String) As String

Const seperator As String = "\"

Dim iPos As Integer

iPos = InStr(fileSpec, seperator)

Do While iPos <> 0

fileSpec = Right(fileSpec, Len(fileSpec) - iPos)

iPos = InStr(fileSpec, seperator)

Loop

baseName = fileSpec

End Function