

```
Sub AddDataToCSVFile()
    Dim fso As scripting.FileSystemObject
    Dim ts As scripting.TextStream
    Dim R As Range
    Dim colCount As Integer
    Dim i As Integer
    Set fso = New scripting.FileSystemObject
    Set ts = fso.OpenTextFile(Envirion("Userprofile") &
        "Desktop\wise owlTest.csv",
        ForAppending, True)
    sheet1.Activate
    colCount = Range("A2", Range("A2").End(xlToRight)).Cells.Count
    For Each R In Range("A2", Range("A1").End(Down))
        For i = 1 To colCount
```

```
ts.write & offset(0,i-1).value  
if i < colcount Then ts.write ", "  
next i  
ts.WriteLine // Text.csv -> Excel/Notepad.  
next x  
ts.close  
set fso = Nothing  
end sub
```

Opening a File for Reading

```
sub ReadFromTextFile()  
Dim fso As scripting.FileSystemObject  
Dim ts As scripting.TextStream  
  
set fso = New scripting.FileSystemObject  
set ts = fso.OpenTextfile (-  
Environ("UserProfile") & "\Desktop\wise out"  
Text.txt" ForReading) → Default so no need
```

workbooks • Add

'reading lines' from a Text File

DO until fs.AEndOfStream, read/ReadAll

Activecell.Value = fs.ReadLine

Activecell.Offset(1, 0).Select

Loop

fs.Close

set fso=Nothing

end sub.

// But all data (column) into
single cells A1 to A20

// so, how to separate

into single cells

Parsing Text From a Text File

Sub ReadFromTextfile()

Set fso = New Scripting.FileSystemObject

Dim fs As Scripting.TextStream

Dim Textline As String

Dim TabPosition As Integer

Set fso = New Scripting.FileSystemObject

Set fso = New Scripting.FileSystemObject

```
set fs=fs0. OpenTextFile(Environ("UserProfile")&"\Desktop\wiseowl\Text.txt")
```

workbooks. Add

Do Until ts = A End Of Stream

`TextLine = ls::ReadLine` \Rightarrow first Tab character in the string

tab position = insts(Fearline, vbTab)

Do until Tabposition = 0

Activerecell::valve = left(TexHline, Tabposition - 1)

Activcell - Offset (0i) • select

Textline = ~~Right~~(Textline, Len(Textline) -

Tab position)

Tab position = insto(Textline, vbTab)

loop

Activecell.value = Textline

Activewcell, offset (0, 0), end (x) To left) - selected

woop

bs. close

```
set fso = Nothing
```

```
End Sub
```

Using Text To columns

```
Sub ReadFromTextFile Easier Method()
```

```
Dim fso As Scripting.FileSystemObject
```

```
Dim ts As Scripting.TextStream
```

```
Set fso = New Scripting.FileSystemObject
```

```
Set ts = fso.OpenTextFile(Environ("UserProfile")
```

```
& "\Desktop\wise owl\Test.txt")
```

```
Workbooks.Add
```

```
Do Until ts.AtEndofstream
```

```
ActiveCell.Value = ts.ReadLine
```

```
ActiveCell.Offset(1, 0).Select
```

```
Loop
```

```
Range("A:A").TextToColumns Tab:=True
```

to close
set $F_{S0} = \text{nothing}$
end sub

Opening a Text File as a Workbook :-

sub readFromTextFileEasiestMethod()

```
File name :=  
workbooks.OpenText Environment("UserProfile")&  
"\\Desktop\\wise owl\\Test.txt",
```

Tab := TRUE

Best

and sub-vertebrates (fish) have evolved

Converting Dates from (autoValue) to Date

sub readFromTextFile()

```
Dim Fso As scripting.FileSystemObject
```

Dim ts As scripting.TextStream

Dim Textline As string

Dim TabPosition As Integer

```
Dim ThisValue As Variant  
set fso = New Scripting.FileSystemObject  
set ts = fso.OpenTextFile(environ("Userprofile") &  
    "\Desktop\wise owl\test.txt")  
workbooks.Add  
DO Until ts.AtEndOfStream  
    Textline = ts.ReadLine  
    TabPosition = InStr(Textline, vbTab)  
    ThisValue = Left(Textline, TabPosition - 1)  
    If IsDate(ThisValue) Then  
        ThisValue = CDate(ThisValue)  
    End If  
    Activecell.Value = ThisValue  
    Activecell.Offset(0, 1).Select  
    Textline = Right(Textline, Len(Textline) - TabPosition)
```

TabPosition = Insts(TextLine, VBTAB)

loop

ActiveCell.Value = TextLine

ActiveCell.Offset(1, 0).End(X1TOLEFT).Select

loop

ls.Close

set fso = Nothing

end sub

'skipping lines :-'

sub ReadFromTextFile()

workbooks.Add

{'ls.skipline'}
'ls.skipline'
'ls.skipline'
'ls.skipline'
'ls.skipline'

} for loop in loop - if set miss
if we know how many rows we need to add
skip - previous line will be

Do until `l$` `= "Data starts here"`
(68) case sensitive.

Loop

End Sub

Creating a change log

⇒ Double click on sheet → code module

General : worksheet

Declarations : ~~selectionchange~~ change (event)

option explicit

private sub worksheet-change (Byval Target As Range)

dim fso As scripting.FileSystemObject

dim changelog As scripting.TextStream

Dim r As Range

set fso = New scripting.FileSystemObject

```
set changeLog = fso.OpenTextFile(  
    filename := Environ("UserProfile") & "\Desktop\"  
    & "wiseowl\changes.txt",  
    FileMode := ForAppending,  
    Create := True)
```

For each domain Target

changeLog.WriteLine & vbTab

changeLog.WriteLine Environ("UserName") & vbTab

changeLog.WriteLine & Address & vbTab

changeLog.WriteLine & Value & vbCrLf

Next &

changeLog.Close

Set fso = Nothing

End Sub

If Target.Cells.CountLarge > 1000 Then

changelog.write Now & vbTab

changelog.write Environ("UserName") & vbTab

changelog.write Target.Address & vbTab

changelog.write "Multiple values changed" &
vbnewline

else

For Each s In Target

changelog.write Now & vbTab

changelog.write Environ("UserName") & vbTab

changelog.write s.Address & vbTab

changelog.write s.value & vbnewline

Next s

End If

changelog.close

set fso = Nothing

End Sub

File Dialogs

Picking files and folders

- Displaying a File Open Dialog
- Executing the Default Action
- Testing which Button was clicked
- Setting File Filters
- Basic File Dialog properties
- Using the Save As Dialog
- Using the File and Folder pickers
- Looping over the Selected Items
- Combining file pickers with filesystemobjects

' what are File Dialogs ?

⇒ File Dialogs are Excel standard way allowing you to select files and folders.

' Displaying a File Open Dialog :-

Module → Using FileDialogs

option explicit

sub OpenAFiler()

Dim fd As FileDialog

set fd = Application.FileDialog(msFileDialogOpen)

fd.show

End sub.

{ msFileDialogFilePicker
msFileDialogFolderPicker
msFileDialogOpen
msFileDialogSaveAs }

' Executing the Default Action :-

sub OpenAFiler()

Dim fd As FileDialog

set fd = Application.FileDialog(msFileDialogOpen)

fd.show
fd.execute
end sub

Testing which button was clicked

option explicit

sub openAFile()

Dim fd As FileDialog

Dim filewaschosen As Boolean

Set fd = Application.FileDialog(msFileDialogOpen)

filewaschosen →
 fd.open → open
 fd.cancel → cancel
 filewaschosen = fd.show

If Not filewaschosen Then

msgbox "you didn't select a file"

Exit sub

End If

fd.execute

End sub

'setting File Filters :-'

sub openfile()

dim --

set fd =

fd.Filters.Clear

fd.Filters.Add "^{old} All excel files", "*.xls"

fd.Filters.Add "New excel files", "*.xlsx"

fd.Filters.Add "Macro Excel files", "*.xlm"

fd.Filters.Add "Any Excel files", "*.*"

fd.FilterIndex = 4

File was chosen = fd.show

end sub.

'selecting multiple files:- (Default)'

fd.FilterIndex = 4

fd.AllowMultiSelect = False

end sub

'setting the initial folder path:-'

```
fd.AllowMultiSelect = False  
fd.InitialFileName = Environ("Userprofile") &  
"\\Desktop\\"  
fd.Title = "Save As"  
End Sub
```

changing the Title and Button Name :-

```
--  
fd.InitialFileName = "C:\Temp\file.txt"  $\Rightarrow$  changes when we select  
fd.Title = "SSH"  $\Rightarrow$  changes the file name  
fd.ButtonName = "OK!"  $\Rightarrow$  changes the button name  
File was chosen = fd.show
```

End Sub

Displaying the Save As Dialog :-

option explicit

Sub SaveAFile()

Dim fd As FileDialog

Dim SaveButtonClicked As Boolean

```
Set fd = Application.FileDialog(msoFileDialogSaveAs)  
fd.Show
```

'setting the initial folder and filter etc
fd.InitialFileName = Environ("Userprofile") &
"Desktop" & ThisWorkbook.Name

If saveAs, we can't clear the existing filters.

fd.FilterIndex = 2 \Rightarrow 2nd filter from save as type is
Dialog Box and path

'Executing the Save As Dialog

SaveButtonClicked = fd.show

If Not SaveButtonClicked Then

msgBox "You didn't choose to save"

Exit Sub

End If

fd.Execute

End Sub

Using the File Picker Dialog

```

option Explicit
Sub PickAFile()
    Dim fd As FileDialog
    Dim ActionClicked As Boolean
    Dim fd As Application.FileDialog(msoFileDialogPicker)
    Set fd = Application.FileDialog(msoFileDialogPicker)
    fd.InitialFileName = Environ("Userprofile") &
    "\Desktop"
    fd.AllowMultiSelect = False
    ActionClicked = fd.Show
    If ActionClicked Then // view → immediate window
        ' debug.print fd.SelectedItems(1) ' index of selected items
        Call CreateCopyOfFile(fd.SelectedItems(1))
    End If
End Sub

```

Using Dialogs with FileSystem Objects

⇒ Create a Backup copy of the selected file in some kind of Archive folder in this machine.

option explicit

Tools → references → Microsoft Scripting Runtime

sub CreateCopyOfFile (filepath, ^{TOCOPY} string)

Dim fso As scripting.FileSystemObject

Dim fileToCopy As scripting.File

Dim ArchiveFolderPath AS string

set fso = New scripting.FileSystemObject

ArchiveFolderPath = Environ("UserProfile") &

"\Desktop\Archive"

if not fso.FolderExists(ArchiveFolderPath) Then

fso.CreateFolder ArchiveFolderPath

end if

set fileToCopy = fso.GetFile(filepath)

fileToCopy.Copy ArchiveFolderPath & "\\" &
fileToCopy.Name

set FSO = Nothing

end sub

looping over the selected items :-

option explicit

sub PickAFile()

dim fd as FileDialog

dim ActionClicked as Boolean

dim LoopCounter as Long

set fd = Application.FileDialog(msoFileDialogFilePicker)

fd.InitialFileName = Environ("Userprofile") & "\Desktop"

fd.AllowMultiSelect = True

ActionClicked = fd.Show

If ActionClicked Then

For LoopCounter = 1 To fd.SelectedItems.Count

Call CreateCopyOfFile(fd.SelectedItems(1))

Next LoopCounter

End If

call createcopyoffile (fd, selectedItems (LoopCounter))

next LoopCounter

End If

End Sub.

Using the Folder Picker Dialog

Option Explicit

Sub PickAFolder()

Dim fd As FileDialog

Dim ActionClicked As Boolean

Dim LoopCounter As Long

Dim SelectedFolderPath As String

Set fd = Application.FileDialog(msoFileDialogFolderPicker)
fd.Title = "Pick the folder to copy files into"
fd.InitialFileName = Environ("Userprofile") & "\Desktop"

fd.AllowMultiSelect = False

ActionClicked = fd.Show

if ActionClicked Then
selectedFolderPath = fd.SelectedItems(1)
msgBox "you didn't pick a folder"
exit sub

end if

set fd=Application.FileDialog(msoFileDialogFilePicker)
fd.Title = "Select files to copy"
fd.InitialFileName = Environ("Userprofile") & "\Desktop"\
fd.AllowMultiSelect = True
ActionClicked = fd.Show

if ActionClicked Then
for loopCounter = 1 To fd.SelectedItems.Count
call CreateCopyOfFile(fd.SelectedItems(loopCounter),
selectedFolderPath)
Next LoopCounter
end if
end sub

```
sub CreateCopyOfFile(filepathToCopy as string,  
                     ArchiveFolderPath as string)  
  
    Dim FSO As scripting.FileSystemObject  
    Dim FileToCopy As scripting.File  
  
    set FSO = New scripting.FileSystemObject  
  
    If Not FSO.FolderExists(ArchiveFolderPath) Then  
        FSO.CreateFolder(ArchiveFolderPath)  
    End If  
  
    set FileToCopy = FSO.GetFile(filepathToCopy)  
    FileToCopy.Copy ArchiveFolderPath & FileToCopy.Name  
  
    Set FSO = Nothing  
End Sub
```

AssayS%

(28)

VBA Arrays :-

what is an Array?

- Declaring a Fixed-size Array
- Writing to and Reading from an Array
- Looping over an Array
- Erasing Arrays
- Multi-Dimension Arrays
- Dynamic Arrays

Declaring a Fixed-size Array :-

option explicit

sub FixedSizeArray()

Dim TopThreeFilms(2) as string

end sub

Using the Option Base Statement

option explicit → 0 is ok, other than 0 !!
→ causes syntax error.

option Base 1

sub FixedSizeArray()

Dim TopThreeFilms(3) As String
End Sub.

Declaring the lower and Upper Bounds,

Option Explicit

Sub FixedSizeArray()
Dim TopThreeFilms(1 To 3) As String
(13 To 23) //also possible
End Sub

Populating an Array:

TopThreeFilms(1) = Range("B3").Value

TopThreeFilms(2) = Range("B3").Value

TopThreeFilms(3) = Range("B4").Value

End Sub

Reading from an Array:

Worksheets.Add

Range("A1").Value = TopTenFirms(3)

Range("A2").Value = TopThreeFirms(2)

Range("A3").Value = TopThreeFirms(1)

④ Formatting an Assay →
Format → Local window
Format → Better
Format → Standard

End Sub

Looping over an Assay

Sub LoopOverAssay()

Dim TopTenFirms(1 To 10) As String

Dim Counter As Long

Sheet1.Activate

For Counter = 1 To 10

TopTenFirms(Counter) = Range("B2").Offset(Counter, 0).Value

Next Counter

worksheets. Add

For counter = 10 To 1 Step -1

ActiveCell.Value = TOPTENFilms(counter)

ActiveCell.Offset(1, 0).Select

Next counter

End Sub

The LBound and UBound Functions:

Sub LoopoverArray()

Dim TOPTENFilms(1 To 13) As String

Dim counter As Long

Sheet1.Activate

For counter = LBound(TOPTENFilms) To
UBound(TOPTENFilms)

TOPTENFilms(counter) = Range("B2").Offset(counter).Value

Next Counter

worksheets. Add
for counter = UBound(TOPTEMFIMS) To
(LBound(TOPTEMFIMS)) step -1

Activecell.value = TOPTEMFIMS(counter)

Activecell.Offset(1, 0).Select

Next counter

Erase TOPTEMFIMS

End Sub

'Declaring Multi-Dimension Arrays'

Sub MultiDimensionArray()
 ' 10 rows, 5 columns

Dim TOPFIMS(0 To 9, 0 To 4) As Variant

// In VBA we can declare an Array upto
// 60 dimensions.

'Populating Multi-Dimension Arrays'

TOPFIMS(0, 0) = Range("A3").Value

TOPFIMS(0, 1) = Range("B3").Value

$(0,2) = ("C3")$ // view → locals window
 $(0,3) = ("D3")$
 $(0,4) = ("E3")$

End Sub

Sub Print

'Looping over multi-Dimension Arrays

Sub multiDimensionArray()

Dim TOPFILMS(0 To 9, 0 To 4) As Variant

Dim Dimension1 As Long, Dimension2 As Long

For Dimension1 = 0 To 9

For Dimension2 = 0 To 4

 Show TOPFILMS(Dimension1, Dimension2)

 Range("A3").Offset(Dimension1, Dimension2)

 Next Dimension2

Next Dimension1

End Sub

LBound, UBound and multi-Dimensions :-

sub --> Dimension 1 \rightarrow First Dimension

For Dimension1 = Lbound(TOPFILMS, 1) To UBound(TOPFILMS, 1)

For Dimension2 = Lbound(TOPFILMS, 2) To UBound(TOPFILMS, 2)

End Sub

Reading from multi-Dimension Arrays

Sub --> Next dimension

worksheets.Add

For Dimension1 = Lbound(TOPFILMS, 1) To UBound(TOPFILMS, 1)

For Dimension2 = Lbound(TOPFILMS, 2) To UBound(TOPFILMS, 2)

Activecell.Offset(Dimension1, Dimension2).Value =
TopCells(Dimension1, Dimension2)

Next Dimension

Next Dimension / View → Load window
Exege TopCells.

End Sub

Dynamic Arrays and Redim

Sub Dynamic(multiDimensionArray)

Dim TopCells() As Variant

Dim Dimension1 As Long, Dimension2 As Long

Sheet1.Activate

Dimension1 = Range("A3").Range("A2").End(Down)
Cells.Count - 1

Dimension2 = Range("A2").Range("A2").End(ToLeft))
Cells.Count - 1

Redim TOPFilms(0 To Dimension1, 0 To Dimension2)

For Dimension1 = LBound To UBound

For Dimension2 = LBound To UBound

Erase TOPFilms

End Sub

Writing a Range to a Dynamic Array:-

Sub QuickDynamicMultiDimensionArray()

Dim TOPFilms() As Variant

Sheet1.Activate

⇒ Always starts from 1 but not 0.

TOPFilms = Range("A3", Range("A2").End(Down).End(ToRight))

Erase TOPFilms

End Sub

Erasing Dynamic Assays

⇒ whole thing will be Re-allocated (free up space)

Writing a Dynamic Assay to a Range :-

sub QuickDynamicMultiDimensionAssay()

Dim TOPFilms As Variant

Sheet1.Activate

TOPFilms = Range("A3", Range("A2").End(xlDown))

End (xlToRight))

Worksheets

worksheets.Add

Range(Activcell, Activcell.Offset(Ubound(TOPFilms, 1) -
= ColumnOffset
Ubound(TOPFilms, 2) - 1)).Value = TOPFilms

Erase TOPFilms

End sub.

I performing Calculations in Arrays :-

option Explicit

```
sub calculatewitharrays()
```

```
dim FilmLengths() as Variant
```

```
dim Answers() as Variant
```

```
dim Dimension1 as long, Counter as long
```

```
sheet1.activate // New → Local windows
```

```
FilmLengths = Range("D3", Range("D21").End(ExcelDown))
```

```
Dimension1 = UBound(FilmLengths, 1) // No of rows
```

```
ReDim Answers(1 To Dimension1, 1 To 2)
```

```
For Counter = 1 To Dimension1
```

hours

```
Answers(Counter, 1) = Int(FilmLengths(Counter, 1) / 60)
```

```
Answers(Counter, 2) = FilmLengths(Counter, 1) mod 60
```

Next Counter

```
Range("F3", Range("F3").Offset(Dimension1 - 1, 1)).Value = Answers
```

Erase FilmLengths

Erase Answers

End Sub

Resizing Assays Dynamically

option Explicit

Sub ResizeDynamicAssay()

Dim ActionFilms() As Variant

Dim S As Range

Dim ActionCounter As Long, LoopCounter As Long

Sheet1.Activate

For Each S In Range("A3", Range("A2").End(xlDown))

End (xlDown))

If S.Offset(0, 1).Value = "Action" Then

or

Case Sensitive

If LCase(S.Offset(0, 1).Value) = "action" Then

ActionCounter = ActionCounter + 1

Reading ActionFilms (1 To ActionCounter, 1 To 5)

for LoopCounter = 1 To 5

ActionFilms(ActionCounter, LoopCounter) =

& offset(0, LoopCounter - 1).Value

next LoopCounter // view → local window.

End If // problem is, ~~where~~ when the

next & // 2nd dimension is created ~~Auto~~ Dynamically,
1st dimension also loses values.

Preserving the contents of Arrays

Regim preserves ActionFilms (1 To ActionCounter, 1 To 5)

!! Runtime "0", subscript out of range.

when we add preserve keyword to the
~~ReDim~~ statement, we can only Re-Dimension
the Last Dimension of the Array.

So, flip around the way we considered
the Rows and Columns.

PCDIM preserve ActionFilms(1 TO 5, the number
of action films) = 1 TO 1 (ActionCounter)

For LoopCounter = 1 TO 5 and i = 0

ActionFilms(LoopCounter, ActionCounter)

= offset(0, LoopCounter - 1) * value

Next LoopCounter

End If

Next &

Exit Sub

Transposing an Array in a horizontal way

Workbooks.Add

Range(ActiveCell, ActiveCell.Offset(0, 1).Offset(-1, 0).Value)

= ActionFilms

End Sub

Worksheets. Add
Range(Activcell).Activate().Offset(UBound(ActionFilms, 2),
0).Value = ActionFilms
Application.Transpose(ActionFilms)

That takes the array & flips the columns to
rows & and rows to column.

End sub.

Constants and Enumerations :-

89

'Using Fixed Values in VBA :-'

- Constants vs. Variables
- Using Built-in Constants
- Declaring and Using constants
- Using Enumerations
- Declaring Custom Enumerations

Using Enumerations in Variables and Parameters

Calculating Enumeration Members

Constants in VBA → values which doesn't changes.

option explicit

sub BuiltInconstants()

= constant → symbol

Activcell.Interior.Color = VbBlue

→ v

// press F8 and point → VbBlue = 16711680 (long integer)

number
(integer)
(0x)

View → Object Browser (F2) → Displays Constants.

Globals → we can see values also.

end sub.

Declaring Constants :-

option explicit

sub DeclaringConstants()

const w0Blue as Long = 13877184

// view Immediate window

?RUB(0,152,202)

13277184

?RUB(205,51,1)

78797

const worked AS long = 78797

'Using Constants :-

Range("C1").Value = "wise owl"

Range("C1").Interior.Color = w0Blue

Range("C1").Font.Color = worked

End Sub

'The scope of Constants :-

option Explicit

const w0blue AS long = 13277184 } module level
const worked AS long = 78797 } (in class)

public const woblue AS long = 13277184 } Project
public const wored AS long = 78797 } and
modul e
public const
& Variables

changing the values of constants

Compile error

Assignment to constant not permitted.

Using constants in calculations :-

option explicit

function circleCircumference (circleDiameter AS Double) AS Double

$$\text{CircleCircumference} = \text{circleDiameter} * 3.14159265359$$

End Function

function CircleArea(CircleRadius AS Double) AS Double

$$\text{circleArea} = \frac{3.14159265359}{\pi} (\text{CircleRadius} * \text{CircleRadius})$$

End Function.

option Explicit

const pi AS Double = 3.14159265359

sub CircleProperties()

Dim CircRad AS Double

Dim CircDiam AS Double

CircRad = InputBox("What's the radius?")

CircDiam = CircRad * 2

MsgBox "The circumference is." & CircleCircumference(CircDiam)

(CircleCircumference(CircDiam))

MsgBox "The Area is " & strArea (strRad)

End Sub.

Using Enumerations →

- | - xlYellowColor
- | - symbol
- | - Enumeration
- | - XlRgbColors
- | - Contains 8gb.
- | - (collection of constants)

option explicit

Sub BuiltInenumerations()

~~Action~~ ActiveCell.Interior.Color = XlRgbColor.xlBlue
View → ObjectBrowser → <globals> (1,8) xlBlue

click on XlRgbColor

End Sub.

Declaring an Enumeration

Module → Enumerations

option explicit

public enum wocolors

woblue = 13277184

wored = 78797

worange = 2000112

end wBrown = 2566755
view → Object Browser → VBAProject → wColor

const wBlue = 0 \Rightarrow value: 13277184

const wRed = 1 \Rightarrow 78797

const wOrange = 2 \Rightarrow 2000112

const wBrown = 3 \Rightarrow 2566755

view → immediate window

? RGB(240, 132, 30)

2000112

? RGB(99, 142, 39)

2566755

Enumeration Data Types

small restriction:

in constants we can use Double, String,
Integer, Date, Long, Boolean

But in Enumeration, however we are restricted to use Long Datatype.

As a result of

- 2.147 Billions → we can use whole numbers for constants in Enumeration.

Using a Declared Enumeration

option Explicit

```
Sub UseEnum()
```

```
Range("D1").Value = "wise"
```

```
Range("D1").Interior.Color = WoodOrange
```

(Per, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500, 1600, 1700, 1800, 1900, 2000, 2100, 2200, 2300, 2400, 2500, 2600, 2700, 2800, 2900, 3000, 3100, 3200, 3300, 3400, 3500, 3600, 3700, 3800, 3900, 4000, 4100, 4200, 4300, 4400, 4500, 4600, 4700, 4800, 4900, 5000, 5100, 5200, 5300, 5400, 5500, 5600, 5700, 5800, 5900, 6000, 6100, 6200, 6300, 6400, 6500, 6600, 6700, 6800, 6900, 7000, 7100, 7200, 7300, 7400, 7500, 7600, 7700, 7800, 7900, 8000, 8100, 8200, 8300, 8400, 8500, 8600, 8700, 8800, 8900, 9000, 9100, 9200, 9300, 9400, 9500, 9600, 9700, 9800, 9900, 10000, 10100, 10200, 10300, 10400, 10500, 10600, 10700, 10800, 10900, 11000, 11100, 11200, 11300, 11400, 11500, 11600, 11700, 11800, 11900, 12000, 12100, 12200, 12300, 12400, 12500, 12600, 12700, 12800, 12900, 13000, 13100, 13200, 13300, 13400, 13500, 13600, 13700, 13800, 13900, 14000, 14100, 14200, 14300, 14400, 14500, 14600, 14700, 14800, 14900, 15000, 15100, 15200, 15300, 15400, 15500, 15600, 15700, 15800, 15900, 16000, 16100, 16200, 16300, 16400, 16500, 16600, 16700, 16800, 16900, 17000, 17100, 17200, 17300, 17400, 17500, 17600, 17700, 17800, 17900, 18000, 18100, 18200, 18300, 18400, 18500, 18600, 18700, 18800, 18900, 19000, 19100, 19200, 19300, 19400, 19500, 19600, 19700, 19800, 19900, 20000, 20100, 20200, 20300, 20400, 20500, 20600, 20700, 20800, 20900, 21000, 21100, 21200, 21300, 21400, 21500, 21600, 21700, 21800, 21900, 22000, 22100, 22200, 22300, 22400, 22500, 22600, 22700, 22800, 22900, 23000, 23100, 23200, 23300, 23400, 23500, 23600, 23700, 23800, 23900, 24000, 24100, 24200, 24300, 24400, 24500, 24600, 24700, 24800, 24900, 25000, 25100, 25200, 25300, 25400, 25500, 25600, 25700, 25800, 25900, 26000, 26100, 26200, 26300, 26400, 26500, 26600, 26700, 26800, 26900, 27000, 27100, 27200, 27300, 27400, 27500, 27600, 27700, 27800, 27900, 28000, 28100, 28200, 28300, 28400, 28500, 28600, 28700, 28800, 28900, 29000, 29100, 29200, 29300, 29400, 29500, 29600, 29700, 29800, 29900, 30000, 30100, 30200, 30300, 30400, 30500, 30600, 30700, 30800, 30900, 31000, 31100, 31200, 31300, 31400, 31500, 31600, 31700, 31800, 31900, 32000, 32100, 32200, 32300, 32400, 32500, 32600, 32700, 32800, 32900, 33000, 33100, 33200, 33300, 33400, 33500, 33600, 33700, 33800, 33900, 34000, 34100, 34200, 34300, 34400, 34500, 34600, 34700, 34800, 34900, 35000, 35100, 35200, 35300, 35400, 35500, 35600, 35700, 35800, 35900, 36000, 36100, 36200, 36300, 36400, 36500, 36600, 36700, 36800, 36900, 37000, 37100, 37200, 37300, 37400, 37500, 37600, 37700, 37800, 37900, 38000, 38100, 38200, 38300, 38400, 38500, 38600, 38700, 38800, 38900, 39000, 39100, 39200, 39300, 39400, 39500, 39600, 39700, 39800, 39900, 40000, 40100, 40200, 40300, 40400, 40500, 40600, 40700, 40800, 40900, 41000, 41100, 41200, 41300, 41400, 41500, 41600, 41700, 41800, 41900, 42000, 42100, 42200, 42300, 42400, 42500, 42600, 42700, 42800, 42900, 43000, 43100, 43200, 43300, 43400, 43500, 43600, 43700, 43800, 43900, 44000, 44100, 44200, 44300, 44400, 44500, 44600, 44700, 44800, 44900, 45000, 45100, 45200, 45300, 45400, 45500, 45600, 45700, 45800, 45900, 46000, 46100, 46200, 46300, 46400, 46500, 46600, 46700, 46800, 46900, 47000, 47100, 47200, 47300, 47400, 47500, 47600, 47700, 47800, 47900, 48000, 48100, 48200, 48300, 48400, 48500, 48600, 48700, 48800, 48900, 49000, 49100, 49200, 49300, 49400, 49500, 49600, 49700, 49800, 49900, 50000, 50100, 50200, 50300, 50400, 50500, 50600, 50700, 50800, 50900, 51000, 51100, 51200, 51300, 51400, 51500, 51600, 51700, 51800, 51900, 52000, 52100, 52200, 52300, 52400, 52500, 52600, 52700, 52800, 52900, 53000, 53100, 53200, 53300, 53400, 53500, 53600, 53700, 53800, 53900, 54000, 54100, 54200, 54300, 54400, 54500, 54600, 54700, 54800, 54900, 55000, 55100, 55200, 55300, 55400, 55500, 55600, 55700, 55800, 55900, 56000, 56100, 56200, 56300, 56400, 56500, 56600, 56700, 56800, 56900, 57000, 57100, 57200, 57300, 57400, 57500, 57600, 57700, 57800, 57900, 58000, 58100, 58200, 58300, 58400, 58500, 58600, 58700, 58800, 58900, 59000, 59100, 59200, 59300, 59400, 59500, 59600, 59700, 59800, 59900, 60000, 60100, 60200, 60300, 60400, 60500, 60600, 60700, 60800, 60900, 61000, 61100, 61200, 61300, 61400, 61500, 61600, 61700, 61800, 61900, 62000, 62100, 62200, 62300, 62400, 62500, 62600, 62700, 62800, 62900, 63000, 63100, 63200, 63300, 63400, 63500, 63600, 63700, 63800, 63900, 64000, 64100, 64200, 64300, 64400, 64500, 64600, 64700, 64800, 64900, 65000, 65100, 65200, 65300, 65400, 65500, 65600, 65700, 65800, 65900, 66000, 66100, 66200, 66300, 66400, 66500, 66600, 66700, 66800, 66900, 67000, 67100, 67200, 67300, 67400, 67500, 67600, 67700, 67800, 67900, 68000, 68100, 68200, 68300, 68400, 68500, 68600, 68700, 68800, 68900, 69000, 69100, 69200, 69300, 69400, 69500, 69600, 69700, 69800, 69900, 70000, 70100, 70200, 70300, 70400, 70500, 70600, 70700, 70800, 70900, 71000, 71100, 71200, 71300, 71400, 71500, 71600, 71700, 71800, 71900, 72000, 72100, 72200, 72300, 72400, 72500, 72600, 72700, 72800, 72900, 73000, 73100, 73200, 73300, 73400, 73500, 73600, 73700, 73800, 73900, 74000, 74100, 74200, 74300, 74400, 74500, 74600, 74700, 74800, 74900, 75000, 75100, 75200, 75300, 75400, 75500, 75600, 75700, 75800, 75900, 76000, 76100, 76200, 76300, 76400, 76500, 76600, 76700, 76800, 76900, 77000, 77100, 77200, 77300, 77400, 77500, 77600, 77700, 77800, 77900, 78000, 78100, 78200, 78300, 78400, 78500, 78600, 78700, 78800, 78900, 79000, 79100, 79200, 79300, 79400, 79500, 79600, 79700, 79800, 79900, 80000, 80100, 80200, 80300, 80400, 80500, 80600, 80700, 80800, 80900, 81000, 81100, 81200, 81300, 81400, 81500, 81600, 81700, 81800, 81900, 82000, 82100, 82200, 82300, 82400, 82500, 82600, 82700, 82800, 82900, 83000, 83100, 83200, 83300, 83400, 83500, 83600, 83700, 83800, 83900, 84000, 84100, 84200, 84300, 84400, 84500, 84600, 84700, 84800, 84900, 85000, 85100, 85200, 85300, 85400, 85500, 85600, 85700, 85800, 85900, 86000, 86100, 86200, 86300, 86400, 86500, 86600, 86700, 86800, 86900, 87000, 87100, 87200, 87300, 87400, 87500, 87600, 87700, 87800, 87900, 88000, 88100, 88200, 88300, 88400, 88500, 88600, 88700, 88800, 88900, 89000, 89100, 89200, 89300, 89400, 89500, 89600, 89700, 89800, 89900, 90000, 90100, 90200, 90300, 90400, 90500, 90600, 90700, 90800, 90900, 91000, 91100, 91200, 91300, 91400, 91500, 91600, 91700, 91800, 91900, 92000, 92100, 92200, 92300, 92400, 92500, 92600, 92700, 92800, 92900, 93000, 93100, 93200, 93300, 93400, 93500, 93600, 93700, 93800, 93900, 94000, 94100, 94200, 94300, 94400, 94500, 94600, 94700, 94800, 94900, 95000, 95100, 95200, 95300, 95400, 95500, 95600, 95700, 95800, 95900, 96000, 96100, 96200, 96300, 96400, 96500, 96600, 96700, 96800, 96900, 97000, 97100, 97200, 97300, 97400, 97500, 97600, 97700, 97800, 97900, 98000, 98100, 98200, 98300, 98400, 98500, 98600, 98700, 98800, 98900, 99000, 99100, 99200, 99300, 99400, 99500, 99600, 99700, 99800, 99900, 100000, 100100, 100200, 100300, 100400, 100500, 100600, 100700, 100800, 100900, 101000, 101100, 101200, 101300, 101400, 101500, 101600, 101700, 101800, 101900, 102000, 102100, 102200, 102300, 102400, 102500, 102600, 102700, 102800, 102900, 103000, 103100, 103200, 103300, 103400, 103500, 103600, 103700, 103800, 103900, 104000, 104100, 104200, 104300, 104400, 104500, 104600, 104700, 104800, 104900, 105000, 105100, 105200, 105300, 105400, 105500, 105600, 105700, 105800, 105900, 106000, 106100, 106200, 106300, 106400, 106500, 106600, 106700, 106800, 106900, 107000, 107100, 107200, 107300, 107400, 107500, 107600, 107700, 107800, 107900, 108000, 108100, 108200, 108300, 108400, 108500, 108600, 108700, 108800, 108900, 109000, 109100, 109200, 109300, 109400, 109500, 109600, 109700, 109800, 109900, 110000, 110100, 110200, 110300, 110400, 110500, 110600, 110700, 110800, 110900, 111000, 111100, 111200, 111300, 111400, 111500, 111600, 111700, 111800, 111900, 112000, 112100, 112200, 112300, 112400, 112500, 112600, 112700, 112800, 112900, 113000, 113100, 113200, 113300, 113400, 113500, 113600, 113700, 113800, 113900, 114000, 114100, 114200, 114300, 114400, 114500, 114600, 114700, 114800, 114900, 115000, 115100, 115200, 115300, 115400, 115500, 115600, 115700, 115800, 115900, 116000, 116100, 116200, 116300, 116400, 116500, 116600, 116700, 116800, 116900, 117000, 117100, 117200, 117300, 117400, 117500, 117600, 117700, 117800, 117900, 118000, 118100, 118200, 118300, 118400, 118500, 118600, 118700, 118800, 118900, 119000, 119100, 119200, 119300, 119400, 119500, 119600, 119700, 119800, 119900, 120000, 120100, 120200, 120300, 120400, 120500, 120600, 120700, 120800, 120900, 121000, 121100, 121200, 121300, 121400, 121500, 121600, 121700, 121800, 121900, 122000, 122100, 122200, 122300, 122400, 122500, 122600, 122700, 122800, 122900, 123000, 123100, 123200, 123300, 123400, 123500, 123600, 123700, 123800, 123900, 124000, 124100, 124200, 124300, 124400, 124500, 124600, 124700, 124800, 124900, 125000, 125100, 125200, 125300, 125400, 125500, 125600, 125700, 125800, 125900, 126000, 126100, 126200, 126300, 126400, 126500, 126600, 126700, 126800, 126900, 127000, 127100, 127200, 127300, 127400, 127500, 127600, 127700, 127800, 127900, 128000, 128100, 128200, 128300, 128400, 128500, 128600, 128700, 128800, 128900, 129000, 129100, 129200, 129300, 129400, 129500, 129600, 129700, 129800, 129900, 130000, 130100, 130200, 130300, 130400, 130500, 130600, 130700, 130800, 130900, 131000, 131100, 131200, 131300, 131400, 131500, 131600, 131700, 131800, 131900, 132000, 132100, 132200, 132300, 132400, 132500, 132600, 132700, 132800, 132900, 133000, 133100, 133200, 133300, 133400, 133500, 133600, 133700, 133800, 133900, 134000, 134100, 134200, 134300, 134400, 134500, 134600, 134700, 134800, 134900, 135000, 135100, 135200, 135300, 135400, 135500, 135600, 135700, 135800, 135900, 136000, 136100, 136200, 136300, 136400, 136500, 136600, 136700, 136800, 136900, 137000, 137100, 137200, 137300, 137400, 137500, 137600, 137700, 137800, 137900, 138000, 138100, 138200, 138300, 138400, 138500, 138600, 138700, 138800, 138900, 139000, 139100, 139200, 139300, 139400, 139500, 139600, 139700, 139800, 139900, 140000, 140100, 140200, 140300, 140400, 140500, 140600, 140700, 140800, 140900, 141000, 141100, 141200, 141300, 141400, 141500, 141600, 141700, 141800, 141900, 142000, 142100, 142200, 142300, 142400, 142500, 142600, 142700, 142800, 142900, 143000, 143100, 143200, 143300, 143400, 143500, 143600, 143700, 143800, 143900, 144000, 144100, 144200, 144300, 144400, 144500, 144600, 144700, 144800, 144900, 145000, 145100, 145200, 145300, 145400, 145500, 145600, 145700, 145800, 145900, 146000, 146100, 146200, 146300, 146400, 146500, 146600, 146700, 146800, 146900, 147000, 147100, 147200, 147300, 147400, 147500, 147600, 147700, 147800, 147900, 148000, 148100, 148200, 148300, 148400, 148500, 148600, 148700, 148800, 148900, 149000, 1491

sub (use enum)

Dim primaryColor AS woColor

Dim secondaryColor AS woColor

primaryColor = woBlue

secondaryColor = woOrange

Range("D1").Value = "wise owl"

Range("D1").Interior.Color = primaryColor

Range("D1").Font.Color = secondaryColor

end sub

enumerations as parameters

option explicit

public enum woWorkHours

woDay = 80

woWeek = 160

woMonth = 160

end enum

Function wiseowlPay (HourlyRate As currency,
UnitofTime As wordofhours)

 As currency

$$\text{wiseowlPay} = \text{HourlyRate} * \text{UnitofTime}$$

End Function

Sub calculateMyPay()

 Dim payperhour As Currency

 payperhour = InputBox("What's your hourly rate?")

 MsgBox "Your daily pay is" &

 wiseowlPay(payperhour, wOfDay)

 MsgBox "Your weekly pay is" &

 wiseowlPay(payperhour, wOfWeek)

msgBox "Your monthly pay is " & wiseowlpay(PayperHour, wMonth)

end sub.

I calculating Enumeration Members :-

$$woDay = 8$$

$$woWeek = 10 = 5 \times 8$$

$$woMonth = 100 = 50 \times 8$$

If \neq change which is low
woDay = 7
woWeek = 35
woMonth = 100

so,

public enum workHours

$$woDay = 7$$

$$woWeek = woDay * 5$$

$$woMonth = woDay * 20$$

End enum.

(08)

option Explicit

public const woDay = 7

public enum woWorkflows

woWeek = woDay * 5

woMonth = woDay * 30

End enum

public woDay As Long

woDay = 7

Compile Error

Constant Expression Required

Creating Word Documents

30

Controlling Microsoft Word Using Excel VBA

• Referencing the Word Object Library

- Creating a new instance of word
- making code version-independent
- writing and formatting Text
- copying Data into word
- Using Version-specific methods
- Using Templates and Bookmarks
- Creating word Document

module → CreatingWordDocument

option explicit
 sub CreateBasicWordReport()

| Referencing the word object library

Tools → preferences → Microsoft Word 15.0 Object Library

View → ObjectBrowser → word

Referencing Applications with Variables :-

Dim wdApp As Application
⇒ one for word & one for excel.

Tools → References → Try to priority order.

We can't able to move word above excel, because we are programming about excel. Excel has to take precedence over any other object library.

Early Binding

Dim wdAPP As Word.Application

Creating a New Instance of word

Set wdAPP = New Word.Application

F8 → Takes couple of seconds.

End Sub.

⇒ We are not able to see in tabs.

>> Ctrl + Alt + Delete (Task manager)

→ Processes → WINWORD.EXE

Ctrl + Shift + Esc

'making word visible :-

sub CreateBasicWordReport()

Dim wdApp As Word.Application

set wdAPP = New Word.Application Best

wdAPP.Visible = True

wdAPP.Activate

end sub

Auto - Instantiating Variables

sub CreateBasicWordReportAutoInstancing()

Dim wdAPP As New Word.Application

wdAPP.Visible = True

wdAPP.Activate

end sub

Downsides :-

- 1) \Rightarrow It checks each time it encounters wdAPP. Time taking process.

2)

```
sub ---  
dim - wdAPP As New word.Application  
If wdAPP Is Nothing Then  
    MsgBox "Word doesn't exist yet!"  
end if  
---  
end sub
```

msgBox "Word doesn't exist yet!"

end if } => This never happens because
--- } wdAPP encountering creates instances.

The CreateObject Function :-

```
Sub CreateBasicWordReportUsingCreateObject()  
    Dim wdAPP As word.Application  
    Set wdAPP = CreateObject("word.Application")
```

wdAPP.Visible = True

wdAPP.Activate

End Sub

```
sub CreateBasicWordReport Using CreateObject()  
    ' Late Binding  
    Dim wdAPP As Object  
    set wdAPP = CreateObject("Word.Application")  
    wdAPP.Visible = True  
    wdAPP.Activate // works without  
end sub
```

Microsoft Word Object Library

Early Binding :- When we set a reference to our object library before we start coding.

Late Binding :- We don't set a reference to an object library before we start coding while we coding. The reference is essentially set while the code is runs.

Downside of Late Binding is,
wdApp. \Rightarrow No intelligence list
because variable object is
Generic Object

P Early ^{Best} Binding Has Intelligence List

Creating a New Document

option Explicit

sub CreateBasicWordReport()

Dim wdApp As Word.Application \Rightarrow Early Binding

set wdApp = New Word.Application

with wdApp \Rightarrow word object is created.

- Visible = True

- Activate

- Documents.Add

end with

end sub.

Begin 'Typing' and 'Formatting' Text :-

sub :-

• Documents.Add

1. Selection.TypeText "TOP movies of 2012"

with ~~Set~~ ~~nest~~ with

with

end sub

sub CreateBasicWordReport()

Dim wdApp As Word.Application Early Binding -

set wdAPP = New Word.Application

with wdAPP

• Visible = True

• Activate

• Documents.Add

with • Selection

• ParagraphFormat.Alignment =

wdAlignParagraphCenter

• Bold Run => on Bold

- Font.size = 14
- TypeText "TOP Movies (of 2012)"
- BoldRun = off Bold

End with

End with

End sub'

-
- BoldRun = TypeText vbnewline
 - TypeParagraph

• Font.size = 11

• ParagraphFormat.Alignment =

wdAlignParagraphLeft

• TypeParagraph

End with

End with

End sub'