```
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 (Environ ("Userprofile") &
                "\Desktop\wise owl\Test.csv",
                ForAppending, True)
 sheet1.Activate

colcount = Range("A2", Range("A2").end(xlToRight)
                                .cells.count

For each r In Range("A2", Range("A1").End(xlDown

For i=1 To colcount
```

```
ts.write r.offset(0,i-1).value
If i<colcount Then ts.write ",,"
next i
ts.writeline                    // Text.csv -> Excel/Notepad.
next r
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 owl\
Text.txt",ForReading)
```

→ Default so no need

```
workbooks.Add
'Reading lines from a TextFile

DO until ts.AtEndOfStream, read/ReadAll
  Activecell.value = ts.readline
  Activecell.offset(1,0).select
                        // But all data (colums) into
Loop                      single cells A1 to A20....
ts.close                  //so, how to seperate
set fso=Nothing           into single cells.
End sub.

'parsing Text from a Text File:-
Sub ReadFromTextFile()
Dim fso AS scripting.FileSystemObject
Dim ts AS scripting.TextStream
Dim TextLine AS string
Dim TabPosition AS integer

set fso= New scripting.Filesystemobject
```

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

workbooks.Add

Do Until ts.AtEndOfStream

 TextLine = ts.ReadLine
TabPosition = Instr(TextLine, vbTab)   => first Tab character in the string.

 Do until TabPosition = 0

  Activecell.value = Left(TextLine, TabPosition-1)
  Activecell.offset(0,1).select
  TextLine = Right(TextLine, Len(TextLine) -
                   TabPosition)
  TabPosition = Instr(TextLine, vbTab)

 Loop

 Activecell.value = TextLine
Activecell.offset(0,1).End(xlToLeft).select
loop
ts.close
```

```
set fso = Nothing
End sub.
```

'Using Text To columns eg:-

```
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
```

ts.close
set fso=Nothing
end sub.

'opening a Text File as a Workbook:-

sub ReadFromTextFileEasiestMethod()
workbooks.OpenText Filename:=
Environ("Userprofile")&
"\Desktop\wise owl\Test.txt",
Tab:=True

Best

end sub.

'converting Dates:-

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)
  Do Until Tabposition =0
  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 = InStr(TextLine, vbTab)

loop
ActiveCell.Value = TextLine
ActiveCell.Offset(1,0).End(xlToLeft).Select

Loop
ts.Close
Set fso = Nothing
End Sub
```

'skipping lines :-

```
Sub ReadFromTextFile()
'...
Workbooks.Add
'ts.SkipLine
'ts.SkipLine
'ts.SkipLine
'ts.SkipLine
'ts.SkipLine
```

or keep in loop.
If we know how many rows we need to skip.

Do until ts.Readline (or) = "Data starts here"
case sensitive.

Loop

--

--

End sub.

'Creating a change log:-

=> Double click on sheet1 -> 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\
    wise owl\changes.txt", _
    iomode:= ForAppending, _
    create:= True)
For each r in Target
  changelog.write Now & vbTab
  changelog.write Environ("UserName") & vbTab
  changelog.write r.Address & vbTab
  changelog.write r.value & vbNewline
Next r
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 x In Target
 changelog.write Now & vbTab
changelog.write Environ("UserName") &vbTab
changelog.write x.Address &vbTab
changelog.write x.value & vbNewline

Next x
End If
changelog.close
```

set fso = Nothing
End sub.

File Dialogs :-

(97)

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 1 → Using FileDialogs

option Explicit

sub openAFile()

Dim fd as FileDialog

msFileDialogFilePicker
msFileDialogFolderPicker
msFileDialogOpen
msFileDialogSaveAs

set fd = Application.FileDialog(msFileDialogOpen)

fd.show // FS → Sun sub routine/program

End sub.

'Executing the Default Action :-

sub openAFile()

Dim fd As FileDialog

set fd = Application.FileDialog(msoFileDialogOpen)

```
    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(msoFileDialogOpen)
                    -1  open
fileDes  ->         0   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 openAFile()
  Dim ---
  Set fd =
fd.Filters.Clear
fd.Filters.Add "All old Excel Files", "*.xls"
fd.Filters.Add "New Excel Files", "*.xlsx"
fd.Filters.Add "Macro Excel Files", "*.xlsm"
fd.Filters.Add "Any Excel Files", "*.xl*"
fd.FilterIndex = 4
Filewaschosen = fd.show
  ---
  ---
End sub.
```

=> index 1 to 4

'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\"

End Sub

'changing the Title and Button Name :-

fd.InitialFileName - - -
fd.Title = "SSH"
fd.ButtonName = "GO!" => changes when we select the file.
FileWasChosen = 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 Filters -

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

//In saveAs, we can't clear the Existing Filters.

fd.FilterIndex = 2 <= and filter from save as type in Dialog Box.

'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
set fd=Application.FileDialog(msoFileDialogFilePicker)
fd.InitialFileName = Environ("Userprofile") &
                     "\Desktop"
fd. Allowmultiselect =False
Actionclicked = fd. show
If Actionclicked Then          //view -> =immediate window.
'debug. print fd. selected Items(1)  '=Index of selected Items.
call CreatecopyOfFile(fd. selected Items(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 (FilePathToCopy As 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(FilepathToCopy)

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(i))
Next Loopcounter
End If
```

```
call createcopyofFile (fd.selectedItems (LoopCounter))
Next loopcounter
End If
End sub.
```

'Using the Folder Picker Dialog &-

```
option Explicit
  sub pickAFile()
  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 Action clicked Then
  selectedFolderpath = fd.selectedItems(1)

else
  msgBox "you didn't pick a folder"
  exit sub

End If
set fd=Application.FileDialog(msoFileDialogFilePicker)
fd.Title="seled 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(FilepathTo copy As string,
                     ArchiveFolderpath As string)
Dim fso AS scripting.Filesystem object
Dim FileTocopy As scripting.File
'set fso= New scripting.Filesystem Object

If Not fso.Folder Exists(Archive Folder path) Then
  fso.createFolder ArchiveFolderpath
End If
set FileTocopy=fso.GetFile(FilePathTO copy)
FileTOCopy.copy Archivefolder path & "\" &
                FileTO copy.Name
Set fso= Nothing
End sub.
```

Arrays :-

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.
```

(0, 1, 2)

```
'Using the Option Base statement
option Explicit
option Base 1
sub FixedSizeArray()
```

0 is ok, other than 0 or 1 - causes syntax error.

Dim TopThreeFilms(3) As string (1,2,3)

End sub.

'Declaring the lower and Upper Bounds:-

Option Explicit

sub FixedSizeArray() => To ignore option Base 1

Dim TopThreeFilms(1 TO 3) As string

(13 TO 23) ✓ //also possible

~~End sub~~

'populating an Assay:-

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

TopThreeFilms(2) = Range("B4").Value

TopThreeFilms(3) = Range("B5").value

~~End sub~~

'Reading from an Assay:-

Worksheets.Add

```
Range("A1").value = TopThreeFilms(3)
Range("A2").value = TopThreeFilms(2)
Range("A3").value = TopThreeFilms(1)
```

* 'Erasing an Assay :-

```
Erase TopThreeFilms
```

view → locals window.

```
end sub
```

'Looping over an Assay :-

```
sub LoopOverAssay()
  Dim TopTenFilms (1 To 10) As String
  Dim counter As long

   sheet1.Activate
For counter = 1 TO 10
   TopTenFilms(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
Erase TopTenFilms
End sub.
```

The LBound and UBound functions:-

```
sub LoopoverArray()
Dim TopTenfilms(1 To 3) As String
Dim counter As Long
Sheet1.Activate
For counter = LBound(TopTenFilms) To
                    UBound(TopTenFilms)
TopTenFilms(counter) = Range("B2").Offset(counter,0).
                                    value
Next counter
```

```
worksheets.Add
For counter = UBound(TopTenFilms) To
                LBound(TopTenFilms) Step -1
ActiveCell.Value = TopTenFilms(counter)
ActiveCell.Offset(1,0).Select
Next counter
Erase TopTenFilms
End Sub
```

'Declaring Multi-Dimension Assays:-

```
Sub MultiDimensionArray()
Dim TopFilms(0 To 9, 0 To 4) As Variant
    // In VBA, we can declare an Assay upto
                                60 Dimensions.
'Populating multi-Dimension Assays.
TopFilms(0,0) = Range("A3").Value
TopFilms(0,1) = Range("B3").Value
```

→ 10 rows → 5 columns

(0,2) = ("C3")
(0,3) = ("D3")
(0,4) = ("E3")

// view -> Locals window.

End Sub

~~sub Arul~~

'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
      TopFilms(Dimension1, Dimension2) =
         Range("A3").offset(Dimension1, Dimension2).value
   Next Dimension2
Next Dimension1
```

// view -> Locals window.

End sub

' LBound, UBound and multi-Dimensions :-

```
sub ---
---
For Dimension1 = lbound(TopFilms, 1) To      → First Dimension
                 UBound(TopFilms, 1)
  For Dimension2 = lbound(TopFilms, 2) To
                   Ubound(TopFilms, 2)
  ---
  ---
  ---
End sub
```

' Reading from multi-Dimension Arrays :-

```
sub ---
Next Dimension1
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 =
TopFilms(Dimension1, Dimension2)
next Dimension2
Next Dimension1
Erase TopFilms
End sub.

// view -> Load window.

Dynamic Arrays and ReDim :-

Sub DynamicMultiDimensionArray()
Dim TopFilms() As Variant
Dim Dimension1 As Long, Dimension2 As Long

Sheet1.Activate

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

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

```
ReDim TopFilms(0 To Dimension1, 0 To Dimension2)
For Dimension1 = LBound - - -
==
==
Erase TopFilms
End sub:
```

'writing a Range to a Dynamic Assay:-

```
Sub QuickDynamicMultiDimensionArray()
  Dim TopFilms() As Variant
  Sheet1.Activate
```

=> Always starts from 1 but not 0..

```
TopFilms = Range("A3", Range("A2").End(xlDown).
                       End(xlToRight))
Erase TopFilms
End sub.
```

'Erasing Dynamic Arrays:-

=> whole thing will be De-allocated (free up space)

'writing a Dynamic Array to a Range:-

```
sub QuickDynamicMultiDimensionArray()
Dim TopFilms() As Variant
sheet1.Activate
TopFilms = Range("A3", Range("A2").End(xlDown).
                                End(xlToRight))
worksheets.Add
Range(Activecell, Activecell.offset(Ubound(TopFilms,1)+1
=Rowoffset
, Ubound(TopFilms,2)-1)).value = TopFilms
=columnoffset
Erase TopFilms
End sub
```

performing Calculations in Assays :-

```
option Explicit
sub CalculatewithAssays()

  Dim FilmlengthsC() AS Variant

  Dim AnswersC() AS Variant
  Dim Dimension1 As Long, Counter AS Long
Sheet1.Activate          //View -> Localswindow
```

```
Filmlengths = Range("D3", Range("D2").End(xlDown))
Dimension1 = UBound(Filmlengths, 1)        // no of rows
```

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

For Counter = 1 To Dimension1
    Answers(Counter, 1) = Int(Filmlengths(Counter, 1)/60)   // = hours
    Answers(Counter, 2) = Filmlengths((Counter, 1) mod 60   // minutes
Next Counter

Range("F3", Range("F3").offset(Dimension1 - 1, 1)).
                                    value = Answers
```

```
Erase FilmLengths
Erase Answers

End Sub
```

' Resizing Arrays Dynamically :-

```
Option Explicit
Sub ResizeDynamicArray()
  Dim ActionFilms() As Variant
  Dim r As Range
  Dim ActionCounter As Long, LoopCounter As Long

  Sheet1.Activate
  For Each r in Range("A3", Range("A2").
                      End(xlDown))
'   If r.Offset(0,4).Value = "Action" Then
                  or                  (case Sensitive
  If LCase(r.Offset(0,4).Value = "action" Then
    ActionCounter = ActionCounter (0 to 1) + 1
```

```
ReDim ActionFilms(1 To ActionCounter, 1 To 5)
  For LoopCounter = 1 To 5
    ActionFilms(ActionCounter, LoopCounter) =
        x.offset(0, LoopCounter - 1).Value
  Next LoopCounter      //view -> localswindow.
End If                  // problem is, ~~wefe~~ when the
Next x                  and Dimension is created ~~Auto~~Dynamically,
                        1st Dimension also looses values.
```

'processing the contents of Assays :-

```
ReDim preserver ActionFilms(1 To ActionCounter, 1 To 5)
  !!
Runtime "9", subscript out of range.
```

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

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

```
Redim Preserve ActionFilms(1 To 5,
                           1 To ActionCounter)

For LoopCounter = 1 To 5
ActionFilms(LoopCounter, ActionCounter) =
    Range.Offset(0, LoopCounter - 1).Value

Next LoopCounter

End If
Next R
End Sub
```

Transposing an Array:- Horizontal way

```
Worksheets.Add
Range(ActiveCell, ActiveCell.Offset(4, UBound(
                  ActionFilms, 2) - 1)).Value
                  = ActionFilms

End Sub
```

Worksheets.Add

Range(ActiveCell, ActiveCell.Offset(UBound(ActionFilms, 2)-1, 0)).Value = ~~ActionFilms~~ _

Application.Transpose(ActionFilms)

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

End sub.

Constants and Enumerations :- (29)

Using Fixed Values ~~of~~ 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 → values which doesn't changes.

'Constants in VBA :-

option Explicit

sub Built-InConstants()

ActiveCell.Interior.color = VbBlue =constant ▯ symbol

//press F8 and point → VbBlue !! VbBlue = 16711680 (Long Integer) (or) number (Interior)

View → Object Browser (F2) → Displays Constants.

{globals → we can see values also.

End sub.

'Declaring Constants :-

option Explicit

sub DeclaringConstants()

const woBlue As Long = 13277184

//view Immediate window

? RGB (0,152,202)
1327784

? RGB (205,51,1)
#8797

const woRed As long = 78797

'Using Constants :-

Range ("C1") . value = "Wise Owl"

Range ("C1") . Interior . color = woBlue

Range ("C1") . Font . color = woRed

End sub

'The scope of Constants :-

option Explicit

Const woBlue As long = 1327784 } module level

const woRed As long = 78797 }

public const woBlue As long = 13277184
public const woRed As long = 78797

} Project level Module public Constants & 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

circleCircumference = circleDiameter * 3.14159265359
                                       Pi
End function
```

```
function circleArea(circleRadius As Double) As Double
circleArea = 3.1415926535 * (circleRadius * circleRadius)
                pi
End Function.
```

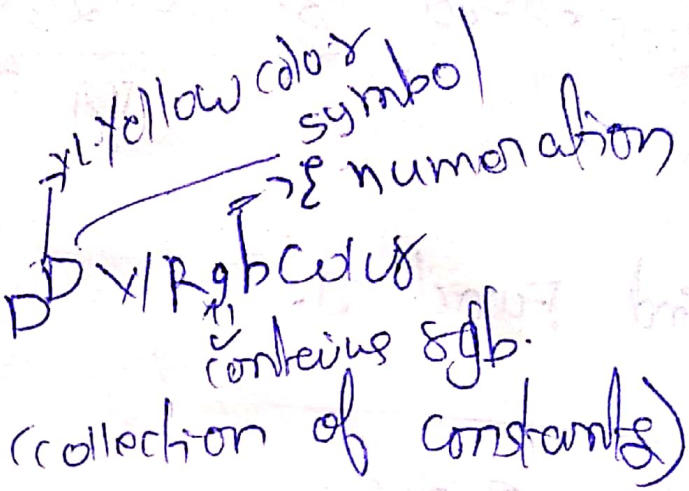
```
option Explicit
const pi As Double = 3.1415926535
```

```
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)
```

MsgBox "The Area is " & circleArea(circRad)

End sub.

' Using Enumerations :-



option Explicit

Sub BuiltInEnumeration()

Action ActiveCell.Interior.Color = XlRgbColor.rgbBlue

View -> Object Browser -> <globals> (or) rgbBlue.

Click on XlRgbColor

End sub.

' Declaring an Enumeration :-

module -> Enumerations

option Explicit

public Enum WdColor

WdBlue = 13277184

WdRed = 78797

'WdOrange = 3000112

End woBrown = 2566755
End Enum
view → objectBrowser → VBAProject → woColor

Const wvBlue = 0 => values
Const wvRed = 1 → 7879 7
Const wvOrange = 2 → 2000112
Const wvBrown = 3 → 2566755

view → Immediate window

? RGB(240, 132, 30)
3000112
? RGB(99, 42, 39)
2566755

'Enumeration Data Types :-

Small Restriction :-
=> in constants we can use Double, String, Integer, Date, Long, ---

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

As a result :-

0 - 2.147 Billions → we can use numbers for constants in Enumeration.

Using a Declared Enumeration :-

```
option Explicit
Sub UseEnum()
    Range("D1").value = "wise owl"
    Range("D1").Interior.Colos = wdOrange
                                 (or)
                                 wdColor.wdOrange

Range("D1").Font.Color = wdColor.wdBlue

End sub
```

Variables and Enumerations :-

```
sub UseEnum()
  Dim primaryColor As woColors
  Dim secondaryColor As woColors

primaryColor = woBlue
secondaryColor = woOrange

Range("D1").Value = "WBS owl"
Range("D1").Interior.Color = primaryColor
Range("D1").Font.Color = secondaryColor
End Sub
```

'enumerations as parameters :-

```
option Explicit
public Enum woWorkHours
    woDay = 8
    woWeek = 40 => 5 x 8
    woMonth = 160 => 20 x 8
End Enum
```

```
Function wisecowlPay (HourlyRate As currency,
                      UnitofTime As wowoskHours)
                                  As Currency

wisecowlpay = HourlyRate * UnitofTime

End Function
```

```
Sub calculatemypay()
  Dim payperHour As Currency

payperHour = InputBox ("what's your hourly rate?")

MsgBox "your daily pay is " &
        wisecowlpay(payperHour, woDay)

MsgBox "your weekly pay is" &
        wisecowlpay (payperHour, cowweek)
```

```
msgBox "your monthly pay is " &
        wiseOwlpay(payperHour, woMonth)

End sub.
```

Calculating Enumeration Members :-

```
woDay = 8
woWeek = 40 = 5 x 8
woMonth = 160 = 20 x 8
```

If I change which is low

```
woDay = 7
woWeek = 35
woMonth = 140
```

previous.

So,

```
public Enum woWorkHours
    woDay = 7
    woWeek = woDay * 5
    woMonth = woDay * 20
End Enum.
```

```
option Explicit
public const woDay = 7
public Enum woworkHours
    woweek = woDay * 5
    womonth = woDay * 20
End Enum
```

```
public woDay As long
woDay =>
```

Compile Error
Constant Expression Requised

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:-

module1 → creatingwordDocument

option explicit

sub createBasicwordReport()

'Referencing the word object library

Tools → References → ☑ Microsoft word 15.0 object library
in 2013
14.0 = 2010

View → ObjectBrowser → word

'Referencing Applications with Variables :-

=> one for excel & one for word.

```
'Dim wdApp As Application =>  % Application
                              %, Application
```

Tools -> References -> Try to priority order.

we can't able to move word above excel, because we are programming ~~abo~~ in excel. Excel has to take precidence 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)
(or)
ctrl + shift + Esc
-> processes -> WINWORD.EXE

'making word Visible :-

```
sub createBasicWordReport()
  Dim wdApp As word.Application
  set wdApp = New word.Application   (Best)
  wdApp.Visible = True
  wdApp.Activate
End sub.
```

'Auto-Instancing Variables :-

```
sub createBasicWordReportAutoInstancing()
  Dim wdApp As New word.Application  => FS won't pass these.
  wdApp.Visible = True
  wdApp.Activate
End sub.
```

& new instance will be created when wdApp Asst encounters

Downsides :-

1) => 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  ==> This never happens because
---:        wdApp encounters &
---:        creats instance of.
End sub
```

'The CreateObject Function :-

```
Sub CreateBasicWordReportUsingCreateObject()
  Dim wdApp As word.Application  => It will work in different pcs with diff version of office
  Set wdApp = CreateObject("word.Application")
  wdApp.Visible = True
  wdApp.Activate
  End Sub.
```

```
sub CreateBasicWordReportUsingCreateObject()
' 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 an 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. => No intelligence list because variable object is Generic object.

P Early Binding (Best) has intelligence list

Creating a New Document :-

```
option Explicit
sub CreateBasicWordReport()
  Dim wdApp As word.Application
set wdApp = New word.Application  => Early Binding
  with wdApp
      .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 .Selection //Needed 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
          .BoldRun => ON Bold
```

```
    .Font.Size = 14
    .TypeText "Top Movies of 2018"
    .BoldRun => off Bold
End with
end with
End sub.
```

```
    .BoldRun.TypeText vbNewLine
    .TypeParagraph (oo)
    .Font.Size = 11
    .ParagraphFormat.Alignment =
                wdAlignParagraphLeft
    .TypeParagraph
End with
End with
End sub.
```
