Application.InputBox

Using the Application.InputBox method

- Limitations of the Generic InputBox function
- Using the Application.InputBox
- setting the Return type
- Validating User input
- Returning a Formula
- Returning a Range
- Returning an Assay

Recap of the InputBox Functions-

```
sub GenericInputBox()
  Dim FilmName As string
  Dim FilmDate As Date
  Dim FilmLength As Integer
 FilmName = InputBox("Enter a Film Name")
Range("B2").end(xlDown).offset(1,0).value
                              = FilmName
End sub.
```

Limitations of the InputBox Function :-

```
Sub GenericInputbox()
Dim FilmName As string
Dim FilmDate As Date
Dim FilmLength As Integer
FilmName = InputBox("enter a film name")
FilmDate = InputBox ("Enter a Date") // give => runtime errors.
Range("B2").End(xlDown).offset(1,0).Value = FilmName
Range("B2").End(xlDown).offset(0,1).Value = FilmDate
End sub.
```

(1) => implicit conversion won't work.

(2) while InputBox is there, we cannot use the Excel sheet except InputBox.

'Using the Application.InputBox method :-

```
Sub ApplicationInputBox()
Dim FilmName As string
```

FilmName = Application.InputBox("Enter a Film name")

Don't have return type.

Range("B2").End(xlDown).Offset(1,0).Value = FilmName

End Sub.

'Specifying the Return Type.

View → Object Browser (F2).

global box → InputBox.

Application → InputBox. (1) Help.

Type (Specifies the return data type.

If the argument is omitted, the dialogue box returns text.

0 - A formula
1 - A number
2 - Text (a string)
4 - A logical value (True or False)
8 - A cell reference as a Range object
16 - An error value, such as #N/A
64 - An Array of values

```
sub Application=inputBox()
    Dim FilmName As string
    Dim Filmlength As integer.

'Returning Numbers from an InputBox

FilmName = Application.InputBox ("Enter a Film Name")
Filmlength = Application.InputBox ("Enter the length",,,,,,)
                                    Named parameter.
Filmlength=Application.InputBox(prompt:="Enter the length",
                                    Type:=1)

Range("B2").End(xlDown).offset(1,0).value=FilmName
Range("B2").end(xlDown).offset(0,2).value= Filmlength

End sub.
```

'Validating the Application.InputBox. (Automatic)

'selecting cells to populate an InputBox :-

=> Application.=InputBox can interact with the cells directly.

'Dealing with Dates :-

```
sub Application=InputBox()
Dim FilmDate As Date
FilmDate = Application.InputBox (prompt:= "Enter the length" a date dd/mm/yyyy",
                                 Type:=1)
Range ("C2").End (xlDown).Offset(1,0).value
                                 = FilmDate.
End sub.
```

'Returning a Formula :-

```
sub EnterMyFormula()
Dim myformula As string
myformula = Application.InputBox (prompt:="enter a
                formula", Type:=0)
'Range ("G2").value = myformula
Range ("G2").FormulaLocal = my formula  Better one.
End Sub / D3+D4.
```

Entering a function :-

```
Sub enterAFormula()
  Dim myformula As string
myformula = Application.InputBox(prompt:="enter a
        formula", Type:=0, Default:="=sum(")
Range("G2").FormulaLocal = myformula
End sub.
```

=sum(D3:D15)

Returning a Reference to a Range :-

```
Sub enterAFormula()
  Dim myformula As string
  Dim formulacell As Range
myformula = Application.InputBox(prompt:="Enter a formula",
        Type:=0, Default:="=sum(")
set formulacell=Application.InputBox(prompt:="choose
    formula cell", Type:=8)
```

Formulacell.formulaLocal = myformula

End sub.

'Returning a Reference to Multiple cells :-

sub copyData()

Dim copyRange As Range

Dim DestinationRange As Range

set copyRange = Application.InputBox(prompt: "choose cells to copy", Type:= 8)

⇒ select singlecell is also ok for this.

set DestinationRange = Application.InputBox(prompt:="choose destination cell", type:=8)

copyRange.copy destinationRange

End sub.

'Returning an Assay of Values :-

sub ReturnAssay() ; Dimension.

Dim FilmLengths() as variant => can hold any datatype.
=> Assay

Dim LoopCounter as long

Dim ResultRange As Range

FilmLengths = Application.InputBox(Prompt:="Choose lengths to convert", Type:=64)

view → locals window

'Looping over an Assay

=> 2 dimensions (No. of columns, No. of rows)

for loopcounter = lbound(FilmLengths, 2) to ubound(FilmLengths, 1)

2 rows

1 row

~~next~~ loopcounter => rows

single column in the cells.

FilmLengths(loopcounter, 1) = int(FilmLengths(loopcounter, 1)/60) &

returns whole no. of hours.

"hours"

& (FilmLengths(loopcounter, 1) mod 60) & "minutes"

view → locals window.

'Transferring the Assay contents to cells.

Next Loopcounter

set ResultRange = Application.InputBox(prompt:="choose where to put results", type:=8)

'we have to select equal range for destination also.

set ResultRange = range(ResultRange, ~~Range~~

ResultRange.offset(UBound(filmlengths,1)-1,0))

ResultRange = filmlengths.

end sub.

-> D3; D14

-> F3

---

with Statements :-

(13)

Using with statements

- writing a simple with statement
- Creating more complex References
- Referencing other objects.

'simple with statements :-

```
sub FormatFilmReleaseDates()

'Range("C3:C15").Interior.Color=rgbAquamarine

  with  Range ("C3:C15")
      .Interior.color = rgbAquamarine
      .Font.color = rgbRed
      .Font.size = 12
      .NumberFormat = "dddd dd mmm yyyy"
  End with

End sub.
```

'Using complex References :-

```
sub FormatFilmReleaseDates()
with Range("C3", Range("C2").End(xlDown))
    .Interior.color = rgbAquamarine
    .Font.color = rgbRed
    .Font.size = 12
```

```
    .NumberFormat = "dddd dd mmm yyyy"
End With
End Sub
```

```
Sub FormatFilmReleaseDates()
With Worksheets("Sheet1").Range("C3", _
    Worksheets("Sheet1").Range("C2").End(xlDown))
    .Interior.Color = rgbAquamarine
    .Font.Color = rgbRed
    .Font.Size = 12
    .NumberFormat = "dddd dd mmm yyyy"
End With
End Sub
```

```
'Referencing other Objects
Sub FormatFilmReleaseDates()
With Worksheets("Sheet1").Range("C3", _
Worksheets("Sheet1").Range("C2").End(xlDown)
    .Interior.Color = rgbAquamarine
```

Worksheets("Sheet3").cells.Interior.Color =

.Interior.Color
=
copies color of sheet1 to sheet3.

.Font.Color = rgbRed

.Font.Size = 12

.NumberFormat = "dddd dd mmm yyyy"

End with

End Sub.

If statements

(14)

Testing Conditions in VBA

- Simple If statements
- Single-line ifs
- The elseif statement
- Nesting ifs
- Combining logical tests

simple If statement

```
sub TestingFilmlength()
  Dim FilmName As string
  Dim Filmlength As Integer
Dim FilmDescription As string
Range("B10").select
FilmName = Activecell.value
Filmlength = Activecell.offset(0,2).value
If Filmlength <100 then
    FilmDescription = "short"
End If
Else
    FilmDescription = "Long"

End If
MsgBox FilmName & "is" & FilmDescription.
End sub.
```

'single-line if statements

```
sub TestingFilmLength()
Dim FilmName As string
Dim FilmLength As integer
Dim FilmDescription As string
Range("B1").select
FilmName = Activecell.value
FilmLength = Activecell.offset(0,2).value
If Filmlength < 100 Then FilmDescription = "short" Else
    FilmDescription = "Long"
msgBox FilmName & " is " & FilmDescription
End sub.
```

'The ElseIf statement :-

```
If Filmlength < 100 Then
   FilmDescription = "short"
ElseIf Filmlength < 120 then
   FilmDescription = "medium"
ElseIf Filmlength < 150 then
   FilmDescription = "Long"
End If.
```

```
Else
    FilmDescription = "Epic"
End If
```

'Nested If statements :-

```
If FilmLength < 100 Then
    FilmDescription = "short"
Else
    If FilmLength < 120 Then
        FilmDescription = "medium"
    Else
        If FilmLength < 150 Then
            FilmDescription = "long"
        Else
            FilmDescription = "Epic"
        End If
    End If
End If
```

Tab for indentation

'Using the Or keyword :-

```
If FilmLength > 120 or len(FilmName) > 15 Then
    FilmDescription = "Epic"
Else
    FilmDescription = "Normal"
End If
```

'Using the And keyword

```
If FilmLength > 120 And Len(FilmName) > 15 Then
   FilmDescription = "Epic"
Else
   FilmDescription = "Normal"
End If
```


Select case statements

The select case statement

- Using a simple select case statement
- phrasing a Logical Test
- Nesting select case statements

```
Sub TestingFilmLength()
 Dim FilmName As String
Dim FilmLength As Integer
Dim FilmDescription As String

 Range("B11").select
FilmName = Activecell.value
FilmLength = Activecell.offset(0,2).value
```

```
'select case statement goes here
 select case FilmLength
    case IS < 100
       FilmDescription = "short"
    case IS < 120
       FilmDescription = "medium"
    case IS < 150
       FilmDescription = "Long"
    case ELSE
       FilmDescription = "epic"
 end select
MsgBox FilmName & " is " & FilmDescription
end sub
```

```
'Testing for a Range of Numbers :-
 select case FilmLength
    case 0 TO 100
       FilmDescription = "short"
    case 100 TO 120
       FilmDescription = "medium"
```

```
case 120 To 150
    FilmDescription = "Long"
case else
    FilmDescription="Epic"
End select
```

Testing for a list of values :-

```
sub FilmName
sub TestingFilmLength()
  Dim FilmName As string
  Dim FilmLength As integer
  Dim FilmDescription As string
  Dim FilmSeason As string
  Dim FilmMonth As Integer
    Range("B11").select
FilmName = Activecell.Value
FilmLength = Activecell.offset(0,2).Value
'select case  function in VBA to calculate month.
FilmMonth = Month(Activecell.offset(0,1).Value)
```

```
select case FilmMonth  -> 1 or 2 or 12
    case 1,2,12
        Filmseason = "winter"
    case 3,4,5
        Filmseason = "spring"
    case 6,7,8
        Filmseason = "summer"
    case 9,10,11
        Filmseason = "Autumn"
End select.
End sub.
```

'Nesting select case statements :-

```
select Case FilmMonth
    case 1, 2, 12
        select case Filmlength
            case 0 TO 100
                FilmDescription = "short winter"
            case 100 TO 180
                FilmDescription = "medium winter"
```

```
        case 120 To 150
            FilmDescription = "Long winter"
        case Else
            FilmDescription = "Epic winter"
    End select
Case 3,4,5
    select case Filmlength
        case 0 To 100
            FilmDescription = "short spring"
        case 100 To 120
            FilmDescription = "medium spring"
        case 120 To 150
            FilmDescription = "long spring"
        case Else
            FilmDescription = "Epic"
    End select
case 6,7,8
    - - -
case 9,10,11
    - - -
End select.
```

Do until and Do while loops :-

Conditional Loops in VBA

- The Basic DO Loop statement
- conditionally Ending a loop
- DO until vs. Loop until
- Do while vs. Loop while
- Exiting from a Loop
- Some practical Excamples.

The Do loop statement :-

```
sub simpleDoLoop()
  Range("A3").select
  DO
    Activecell.offset(1,0).select
  loop                    //no end.
end sub
```

'Adding an until condition to a DO loop.

```
sub simpleDOloop()
  Range("A3").select
  Do until Activecell.value = ""
     Activecell.offset(1,0).select
  loop
end sub.
```

'Using the loop until statement :-

```
Do
   Activecell.offset(1,0).select
Loop until Activecell.value = "1"
```

one time it will execute even though condition satisfies.

~~End sub~~

'The Do while & loop while statements :-

```
Do while Activecell.value <> ""
     Activecell.offset(1,0).select
  loop
```

```
Do
   Activecell.offset(1,0).select
loop while Activecell.value <> ""
```

'Exciting from a DO loop :-

```
DO
  If Activecell.value = "" Then Exit DO
  Activecell.offset(1,0).select
loop
```

=> we can place anywhere in the loop.

'A simple practical example:-

```
option Explicit
  sub simpleDoloop()
     Dim FilmLength As Integer
     Dim FilmRating As string
     Range("A3").select
  Do until Activecell.value=""
   FilmLength =Activecell.offset(0,3).value
 If FilmLength < 100 Then
     FilmRating ="short"
Else If FilmLength < 150 Then
     FilmRating= "medium"
 Else
     FilmRating ="Long"
 End If
```

```
Activecell.offset(0,4).value = FilmRating
Activecell.offset(1,0).select

Loop
End sub
```

'Separating a list with a Do loop:-

```
sub simpleDoloop()
  Dim FilmLength As Integer
  Dim FilmRating As string
  Worksheets("sheet1").Activate
Range("A3").select

  Do until Activecell.value=""
  FilmLength = Activecell.offset(0,3).value
If FilmLength <100 Then
    FilmRating = "short"
Else-If FilmLength <150 Then
    FilmRating = "medium"
Else
    FilmRating= "Long"
End If
```

```
Range(Activecell, Activecell.end(xlToRight)).copy
worksheets(FilmRating).Activate
Activecell.pastespecial
Activecell.offset(1,0).select

worksheets("sheet1").Activate
Activecell.offset(1,0).select
Loop
End sub.
```

'making the code more efficient:-

```
sub simpleDoLoop()
    Dim Filmlength As Integer
    Dim FilmRating As String
Application.screenUpdating = False  =>To make more fast
worksheets("sheet1").Activate
    .
    .
    .
 Loop
```

```
Application.CutCopyMode = False
Application.ScreenUpdating = True

End Sub.
```


Find and FindNext :-

The Find and FindNext methods
- Applying the Find method to a Range
- Case-sensitive searches
- Finding whole and partial words
- Dealing with Failed Finds
- Creating a Basic search system
- Using FindNext to Find All Instances of a value

Applying the Find method to a Range :-

```
Option Explicit
Sub FindAFilm()

  Range("B3:B15").Find("Ted").Select

End Sub.
```

=> returns a reference a range object

" " Not case sensitive text in anywhere

'making the Find method Case-Sensitive:-

```
sub FindAFilm()
'Range("B3:B15").Find("Ted",,,,,True).select
Range("B3:B15").Find(what:="Ted", matchcase:=True).
                                              select
End sub.
```

'Finding whole words or parts of words:-

```
sub FindAFilm()
Range("B3:B15").Find(what:="Ted", matchcase:=True,
                     LookAt:=xlWhole).select
                              xlPart
End sub.
```

'Dealing with Finding Nothing:-

```
sub FindAFilm()          //If it didnot finds, gum time
Dim FilmCell AS Range                          error.
Set FilmCell = Range("B3:B15").Find(what:="Ted 3", matchcase:True,
                                    LookAt:=xlWhole
```

```
If Filmcell is nothing Then
    msgBox "No film was found"
Else
    Filmcell.select
End If
End sub.
```

View → locals window.

Creating a Basic Search System :-

```
sub FindAFilm()
  Dim searchRange As Range
  Dim Filmcell As Range
  Dim FilmName As string → FilmName = InputBox("TYPE in a Film Name")
set searchRange = Range("B3", Range("B2").End(xlDown))
set Filmcell = searchRange.Find(what:=FilmName,
        matchCase:=False, LookAt:=xlpart)
  If Filmcell Is Nothing Then
    msgBox "No film was found"
  Else
  msgBox Filmcell.value & "was released on"
        & Filmcell.offset(0,1).value
  End If
```

End Sub

'Using the FindNext method :-

```
sub FindAFilm()
  Dim searchRange As Range
  Dim Filmcell As Range
  Dim FilmName As string
  Dim FirstFilmcell As string
FilmName = InputBox("Type in a film name")
set SearchRange = Range("B3", Range("B2").End(xlDown))
set Filmcell = SearchRange.Find(what:= FilmName,
  match case:= False, LookAt:=xlpart)
If Filmcell Is Nothing Then
  msgBox "no film was found"
Else
  FirstFilmcell = Filmcell.Address
DO
msgBox Filmcell.value & "was released on" &
                Filmcell.offset(0,1).value
```

```
    set Firstcell = searchRange.FindNext(Firstcell)
Loop while Firstcell.Address <> firstFirstcell
End If
End sub
```

For Next loops :-

The For Next statement

- Using loop counters
- The Step keyword
- Exciting from a For Next loop
- Looping Through items in a collection

The syntax of a For Next Loop :-

```
sub simpleForNextLoop()
  Dim loopcounter AS integer
  For loopcounter = 1 TO 10
      Debug.print "loopcounter =" & loopcounter
  Next loopcounter
End sub
```

view → Immediate window.

'The step keyword :-

```
sub simpleForNextLoop()
  Dim i As Integer
For i=1 To 10 step2
  Debug.Print "i=" & i
Next i
End sub
```

o/p :- 1, 3, 5, 7, 9

'Reversing a For Next loop :-

```
sub simpleForNextLoop()
  Dim i As Integer
For i=10 To 1 step-1
  Debug.Print "i=" & i
Next i
End sub
```

o/p :- 10, 9, 8, 7, 6, 5, 4, 3, 2, 1

'Exiting from a For Next loop :-

```
sub simpleForNextLoop()
  Dim i As Integer
  Dim RandomNumber As Double

For i = 10 To 1 step -1
  RandomNumber = Math.Rnd
If RandomNumber > 0.7 Then Exit For
  Debug.Print "i=" & i
Next i
End Sub.
```

```
' Looping over Worksheets:-
option Explicit
sub protectAllWorksheets()
        LoopCounter
  Dim i As Integer
  Dim NumberofWorksheets As Integer
NumberofWorksheets = Worksheets.Count
```

```
for loopcounter = 1 TO NumberofWorksheets
'worksheets(1).protect
worksheets(loopcounter).protect
Next loopcounter
End sub.
```

worksheets(loopcounter).Unprotect ⇒ To unprotect worksheets.

Looping over Workbooks :-

```
option explicit
sub closeAllButThisWorkbook()
Dim loopcounter As Integer
Dim NumberofBooks As Integer

NumberofBooks = Workbooks.count
for loopcounter= NumberofBooks To 2 step -1
    workbooks(loopcounter).close
Next loopcounter
End sub.
```

```
For LoopCounter = 2 To NumberOfBooks
    Workbooks(LoopCounter).Close
Next LoopCounter
```

// => causes Runtime Error - problem is re-indexes the collections & deletes wrong work book so, loop Back to front.

Looping over cells :-

```
Option Explicit
Sub RateAllFilms()
    Dim LoopCounter As Long
    Dim NumberOfCells As Long
    Dim FilmLength As Integer
    Dim FilmRating As String
Worksheets("sheet1").Activate
NumberOfCells = Range("A3", Range("A2").End(xlDown)).Count

For LoopCounter = 1 To NumberOfCells
    FilmLength = Range("A2").Offset(LoopCounter, 3).Value
If FilmLength < 100 Then
    FilmRating = "short"
```

```
elseif filmlength < 150 Then
  filmRating = "Medium"
Else
  filmRating = "Long"
end If
Range("A8").offset(loopCounter, 0).value =
                                    filmRating
Next loopCounter
end sub.
```


For Each Loops:-

Looping over Collections

- The For Each Next statement
- Using object variables
- Looping over the Worksheets Collection
- Looping over Workbooks
- Looping over chartobjects
- looping over cells
- Nested For Each loops

'The For Each Next Statement :-

```
object explicit
sub listworksheetNames()
   Dim singlesheet As worksheet
   For Each singlesheet in worksheets
     Debug.print singlesheet.name
   Next singlesheet          //view -> immediate window.
End sub
```

'A practical Example :-

```
sub protectAllsheets()
   Dim singlesheet As worksheet
for Each singlesheet in worksheets
     singlesheet.protect
  Next singlesheet )
End sub.                    singlesheet.unprotect.
```

'Looping over workbooks :-

```
sub listworkbookNames()
  Dim singleBook AS workbook
For each singleBook In workbooks
   Debug.print singleBook.Name
Next singleBook      // view -> Immediate window.
End sub
```

'Excluding Objects from a Loop:-

```
sub closeAllBooksExceptThisone()
Dim singleBook AS workbook
For each singleBook In workbooks
 ' singleBook.close
```

This will close this workbook first and as a result code will stop. so, we have to exclude this workbook.

```
 If singleBook.Name <> "Top Movies 2018.xlsm" Then
   singleBook.close
 End If
Next singleBook
End sub
```

what happens if I change this name i.e; Rename.

ThisWorkbook.Name

'Qualifying Collection Names:-

object Explicit

sub protectsheetsInAnotherWorkbook()

Dim singlesheet As Worksheet

For Each singlesheet in Workbooks("Book2").worksheets

    singlesheet.protect

Next singlesheet

End sub.

singlesheet.unprotect

'Looping over chartobjects:-

=>In this case it is not compulsory to use Workbooks("Book2").worksheets.

we can use:-

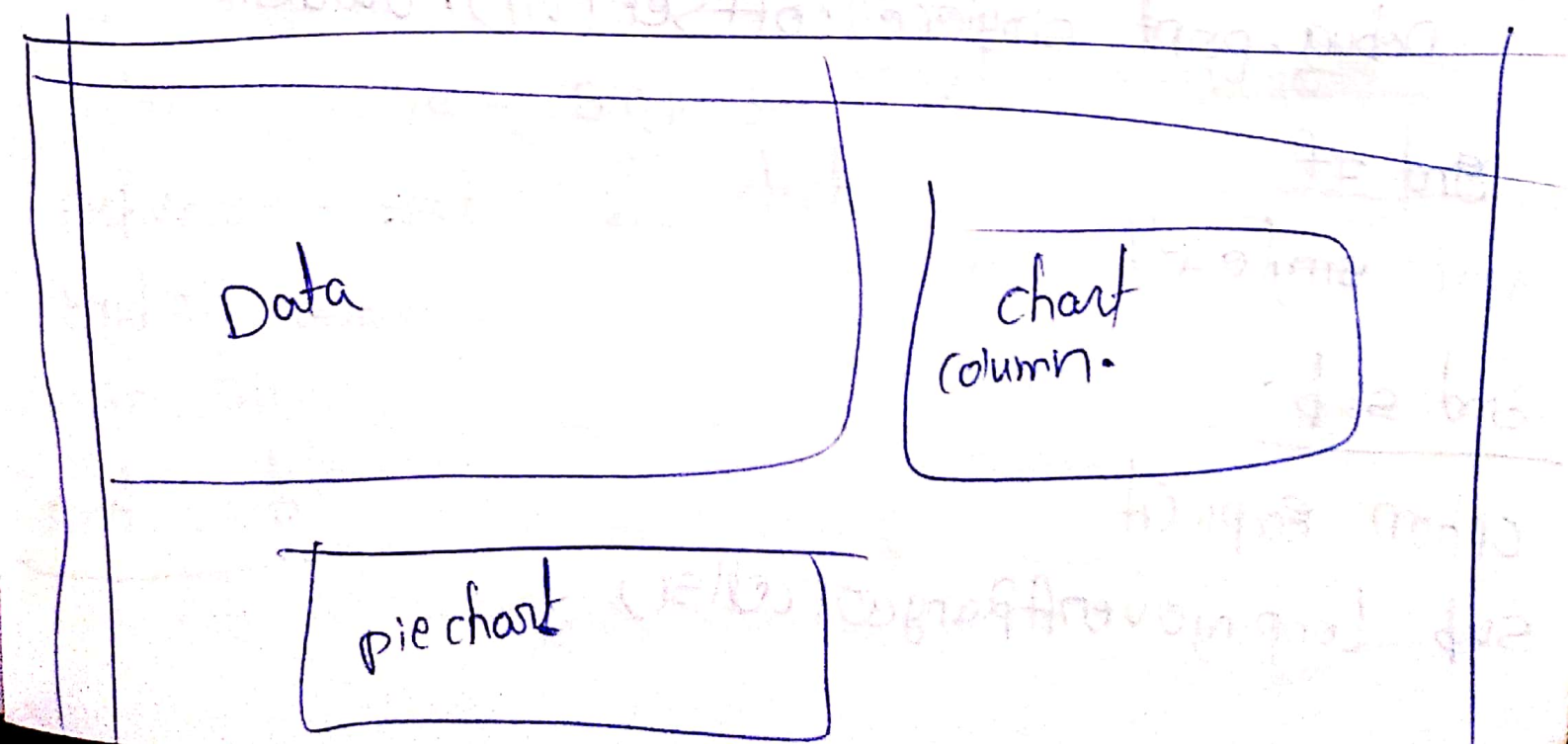
```
Workbooks("Book2").Activate
For Each singlesheet In worksheets.
```

But in some cases like charts, it is compulsory.

'Looping over chartobjects:-

insert -> column -> chart
insert -> pie -> chart etc... } drag inside work sheet.

object Explicit
sub setdataforcharts()
=> we don't use chart because, the charts are in worksheet.
Dim singlechartobject as chartobject
qualifying the collection name with the containing object.
For each singlechartobject IN (sheet1.chartobjects)
Singlechartobject.chart.setSourceData Range("B2:B15, D2:D15")
(The chartobject is the container of chart)
Next singlechartobject
End sub



'Looping over cells :-

```
Option Explicit
Sub LoopingOverARangeOfCells()
  Dim singlecell As Range
'For Each singlecell In Range("A3:A15")
'For Each singlecell In Range("A3", Range("A2").End(xlDown))
  Dim listofcells As Range
Set listofcells = Range("A3", Range("A2").End(xlDown))
For Each singlecell In listofcells
  If singlecell.Offset(0,3).Value > 120 Then
    Debug.Print singlecell.Offset(0,1).Value
  End If
Next singlecell
End Sub
```

```
Option Explicit
Sub LoopingOverARangeOfCells()
```

```
Dim singlecell As Range
Dim listofcells As Range

Thisworkbook.Activate
sheet1.Activate
set listofcells = Range("A3", Range("A2").
                                End(xlDown))

workbooks.Add
Range("A1").value = "List of long files"
Range("A2").select

For Each singlecell In listofcells
  If singlecell.offset(0,3).value > 100 Then
  Activecell.value = singlecell.offset(0,1).value
  Activecell.offset(1,0).select
  End If
Next singlecell
End sub.
```

Nested For Each Loops :-

```
option Explicit
sub NestedForEachLoops()

  Dim singleBook As workbook
  Dim singlesheet As worksheet

For each singleBook in workbooks
   for each singlesheet in singleBook.worksheets
      singlesheet.protect
   next singlesheet
Next singleBook
End sub
```

singlesheet.unprotect

Creating Functions :-

Returning values from procedures :-

- writing a simple function
- Calling Functions
- creating parameters
- Optional parameters
- Rewriting code to use Functions

'writing a simple function :-

```
option Explicit
Function CustomDate() As String
CustomDate = Format(Date, "dddd dd mmmm yyy")
End Function.
```

what this function will return.

'Calling a function :-

(view -> Immidiate window)

```
? customdate
monday 17 February 2014
```

```
Sub CreateNewSheet()
  Worksheets.Add
Range("A1").Value = "created on" & CustomDate
End Sub.
```

CustomDate ⇓ function.

'Adding parameters to a functions-

~~Object~~ & option Explicit
Function CustomDate (DateToFormat As Date) As String

CustomDate = Format(DateToFormat, "dddd dd mmmm yyyy")

End Function

'calling a Function with parameters &-

sub CreateNewsheets()
 Worksheets.Add
Range("A1").value = "Created on" &
 CustomDate(Date)
 ⇓
 #19/02/2016#)
End sub

'Adding multiple parameters &-

option Explicit
Function CustomDate(DateToFormat As Date,
 IncludeTime As Boolean) As String
If IncludeTime Then → IncludeTime = True
CustomDate = Format(DateToFormat, "dddd dd mmmm
 yyy hh:mm:ss")
Else

```
customDate = Format(DateToFormat, "dddd dd mmmm
                                   yyyy")
End If
End Function
```

```
Sub CreateNewSheet()
Worksheets.Add
Range("A1").Value = "Created on" & customDate
                                   (Date, True)
End Sub
```

Now => correct result.

```
'Optional Parameters :-
Option Explicit
```

// All the optional parameters must come after compulsory ones.

```
Function customDate(DateToFormat As Date,
        Optional IncludeTime As Boolean = False)
                                   As String
If IncludeTime Then
    customDate = Format(DateToFormat,
                 "dddd dd mmmm yyy hh:mm:ss")
Else
```

default

```
customDate = Format(DateToFormat, "dddd dd mmmm yyyy")
End If
End Function
```

```
Sub createNewSheet()
  worksheets.Add
Range("A1").value = "created on" & customDate(Now)
End sub
```

(Now) ⇒ (Now, True)

Using Functions in worksheets :-

In Excel:

In same workbook => In excel, it will re-calculate but from macro fixed.

=customDate(now(), true)

⇓

Monday 17 February 2014 08:13:53

=> click on Fx in formula bar → user defined → => In category

customDate → ok

now()
false

→ ok.

Rewriting code to use Functions:-

```
option Explicit
Function FilmLength(RunTime As Integer) As String
  If RunTime < 100 Then
     FilmLength = "short"
  ElseIf RunTime < 150 Then
     FilmLength = "medium"
  ElseIf RunTime < 200 Then
     FilmLength = "Long"
  Else
     FilmLength = "Epic"
  End If
End Function
```

```
sub RateFilmsByLength()
 Sheet1.Activate
 Range("A3").Select
 Do until ActiveCell.Value = ""
```

```
Activecell.offset(0,4).value= Filelength (Activecell.
                                offset(0,3).value)

Activecell.offset(1,0).select

Loop
End sub
```

Error Handling (On Error, Resume, GoTo) (21)

"Dealing with Run Time Errors

- What are Run Time Errors?
- The On Error statement
- Ignoring Errors
- Creating an Error Handler
- Resuming After an Error
- Testing for the Type of Error
- Raising custom Errors.

---

i Run Time Errors in VBA

-> A RunTime Error in VBA is literally an Error which occurs while your program is Running.

-> It is imp to distinguish RunTime Errors from other type of Errors in VBA.

-> syntax Error, punctuation or grammer mistakes.

=> syntax Error

worksheets (sheet 12") . select

Red color

=> compile Error

worksheet ("sheet 12"). select

S -> is missing

Debug -> compile VBA project.

=> RunTime Error

worksheets ("sheet 12") . select

runtime error '9':
subscript out of range
(worksheet not their error).

Ignoring Run Time Errors :-

```
option explicit
sub errorsInVBA()
on error Resume Next
 worksheets("sheet42").select

End sub.
```

=> Bad Reason.

// It will ignores the line causing the Run-Time error.

=> Good Reason.

```
worksheets("sheet42").Delete (Deletes first time & ignores from 2nd run onwards).
```

Going To an Error Handler :-

```
option Explicit
sub ErrorsInVBA()
 on Error Resume next GOTO createsheet42

 worksheets("sheet42").select
 Range("A1").Value = Now
 Exit sub
createsheet42: //After this error must go to top where it stops.
 Worksheets.Add.Name = "sheet42"
 Resume Next
End sub
```

```
Exit Sub
'xxxxxxxxx
'Error Handling section
'xxxxxxxxx
createsheet2:
    Worksheets.Add.Name = "sheet2"
    Resume Next
End Sub.
```

Deactivating an Error Handler :-

→ In top Example, any other error also Navigates to createsheet2 only.

```
option Explicit
Sub ERRORSINVBA()
On Error GoTo CreateSheet2
Worksheets("sheet2").Select
On Error GoTo 0
Run-time error "11"
Division by zero
Range("A1").Value = Now
Range("A2").Value = 1/0
```

Deactivating an error handler and switch back to Normal.

```
Exit sub
'xxxxxxxx
'ERROR HANDLING SECTION
'xxxxxxxx
CreateSheet2:
  Worksheets.Add.Name = "sheet2"
  Resume Next
End sub.
```

```
'Resuming at Any point in a procedure-
  option Explicit
  sub FindAFilm()
    Dim FilmName As string
    Dim FilmDate As Date
      =>label
GuessAgain:
  FilmName = InputBox("Type in a Film name")
  sheet1.Activate

On Error GOTO FilmNotFound
Range("B3:B15").Find(FilmName).Select
On Error GOTO 0
```

```
FilmName = Activecell.value
FilmDate = Activecell.Offset(0,1).value
msgBox FilmName & "was released on" & FilmDate
Exit sub
FilmNotFound:
  msgBox FilmName & "was not found"
  Resume GuessAgain
End sub.
```

// unsobisticated Technique -

```
option Explict
  sub findAFilm()
    Dim FilmName As string
    Dim FilmDate As Date
    Dim Buttonclicked As vbmsgBoxResult

GuessAgain:
  FilmName = InputBox("Type in a film name")
  sheet1.Activate
```

```
On Error GoTo FilmNotFound
Range("B3:B15").Find(FilmName).Select
On Error GoTo 0

FilmName = ActiveCell.Value
FilmDate = ActiveCell.Offset(0,1).Value

MsgBox FilmName & "was released on "
                & FilmDate
Exit Sub
FilmNotFound:
  ButtonClicked = MsgBox(FilmName & "was not
                found", vbRetry Cancel)
If ButtonClicked = vbRetry Then Resume GuessAga
End Sub
```

'Using multiple Error Handlers :-

```
option Explicit
sub FindAFilm()
  Dim FilmName As String
  Dim FilmDate As Date
```

```
Dim ButtonClicked As VbMsgBoxResult

GuessAgain:
FilmName = InputBox("Type in a film name")
Sheet1.Activate
On Error GoTo FilmNotFound
Range("B3:B15").Find(FilmName).Select

On Error GoTo 0

FilmName = ActiveCell.Value

On Error GoTo InvalidDate
FilmDate = ActiveCell.Offset(0,1).Value
On Error GoTo 0
MsgBox FilmName & " was released on
                  " & FilmDate
Exit Sub

FilmNotFound:
ButtonClicked = MsgBox(FilmName & "was not
                found", vbRetryCancel)
```

```
If ButtonClicked = vbRetry Then Resume GuessAgain
Exit Sub
InvalidDate:
MsgBox "FilmName & "was found but its date
                is in wrong format"
End sub.
```

'The Err object :-

```
option Explicit
sub openAworkbook()
  ChDrive "E:" //Runtime error "68"
                Device unavailable.
  ChDir "E:\my files" // Runtime error "76"
                path not found.
  workbooks.open "Book1.xlsx" // Runtime error
                "1004" Book1.xlsx couldn't found.
End sub.
```

```
option Explicit
sub openAworkbook ()
 On Error GOTO problem
 chDrive "E:"
 chDir "E:\my files"
 workbooks.open "Book1.xlsx"

 Exit sub
problem:
If Err.Number = 68 Then
 MsgBox "Drive E: isn't available - have you
                              plugged it in?"
ElseIf Err.Number = 76 Then
MsgBox "The my Files Folder isn't on the drive"
Else
 msgBox "Book1 wasn't found - Are you sure
                              it's there?"
End If
End sub.
```

'Raising Custom Errors :-

```
Option Explicit
Sub RaisingCustomErrors()
  Worksheets("sheet12").select // Run time error '9' subscript out of range.
End sub
```

```
sub RaisingCustomErrors()

On Error GoTo customError
Worksheets("sheet12").select
  Exit sub
CustomError:
  Err.Raise vbObjectError + 1, "RaisingCustomErrors",
      "Worksheet12 doesnt exist"
End sub.
```

=> Generates unique error number which doesn't exists.

Run-time error: 2147... -
Worksheet12 doesn't exists.