Day 2

1pm-3pm

INTRODUCTION TO CONDITIONS

Introduction to Boolean Values

Boolean Values and Procedures

Boolean Built-In Functions

Logical Operators

Checking Whether a Condition is True/False

What Else When a Condition is True/False?

**Using Labels**

Sub linelabels()

Dim myinput As Integer

myinput = 14

If myinput = 1 Then GoTo ifone

If myinput <> 1 Then GoTo notone

ifone:

MsgBox "one"

Exit Sub

notone:

MsgBox "not one"

Exit Sub

End Sub

**Using the Range Object**

1. Active worksheet

Range(“A1:C5”) ‘cell range

Range(“K9”) ‘single cell address

Range(“PriceList”) ‘named range in excel

Range(“A:A”) ‘entire column

Range(“3:3”) ‘entire row

1. Other worksheet

Worksheets(“Sheet1”).Range(“A1:C5”)

1. Other workbook

Workbooks(“Budget.xlsx”).Worksheets(“Sheet1”).Range(“A1:C5”)

1. Multiple non-contiguous section

Range(“A1:B8,D9:G16”)

**The Cells property**

‘takes in two arguments, row and column. The intersection can be used to point to a cell

Worksheets(“Sheet2”).Cells(2, 3)

‘can also be used to highlight a cell range

Range(cells(1,1),cells(10,8))

‘cells property has an advantage over range when you use variables and logic to select cells.

**The Offset property**

‘two rows below A1 and three cells to the right of A1, this refers to D3

Range(“A1”).Offset(2,3)

‘can also accept negative values for the arguments, the arguments still reflect rows and columns(in order).

‘the following example refers to A1

Range(“C2”).Offset(-1,-3)

‘to use relative references, offset can be used along with the ActiveCell object

‘example: write the date on the right of the active cell

ActiveCell.Offset(0,1)=Date

**The Value property**

‘displays a message box that shows the value in cell A1 on Sheet1

MsgBox Worksheets(“Sheet1”).Range(“A1”).Value

‘without assigning to a variable, you can read the Value property only for a single-cell range object. The ‘following statement generates an error:

MsgBox Worksheets(“Sheet1”).Range(“A1:C3”).Value

‘change the Value property for a range of any size.

Worksheets(“Sheet1”).Range(“A1:C3”).Value = 123

‘Value is the default property for a Range object.

Range(“A1”).Value = 75

Range(“A1”) = 75

‘assign values in a multi-cell range to a variable

‘the following example displays the data in C1

Dim range1 as variant

Range1=Range(“A1:C3”).Value

Msgbox Range1(1,3)

**The Text property**

‘returns the formatted display value in a cell (not the raw logical value).

Worksheets(“Sheet1”).Range(“A1”).Text

**The Count property**

Dim a As Integer, b As Integer, c As Integer

'returns all cells

a = Range("A1:B3").Count

'returns all non-blank cells

b = Range("A1:B3").SpecialCells(2).Count

'1-numbers

'2-textvalues

'4-logical

'returns all non-blank cells

c = Application.WorksheetFunction.CountA(Range("a1:b3"))

**The Column and Row properties**

‘The Column property returns the column number of a single-cell range. (the ff example returns 6)

MsgBox Sheets(“Sheet1”).Range(“F3”).Column

‘The Row property, returns the row number of a single-cell range. (the ff example returns 3)

MsgBox Sheets(“Sheet1”).Range(“F3”).Row

**The Address property**

‘displays the cell address for a Range object as an absolute reference (a dollar sign before the column letter and before ‘the row number) (read-only property)

MsgBox Range(Cells(1, 1), Cells(5, 5)).Address ‘this returns $A$1:$E$5

**The HasFormula property**

‘this is a Boolean read-only property that returns true if the selected range has a formula and false if none

Dim FormulaTest As Boolean

FormulaTest = Range(“A1”).HasFormula

‘when using a range that has cells with formula and non-formula, the ff will generate an error

Dim FormulaTest As Boolean

FormulaTest = Range(“A1:A3”).HasFormula

‘use the following to resolve this

Sub CheckForFormulas()

Dim FormulaTest As Variant

FormulaTest = Range(“A1:A2”).HasFormula

If TypeName(FormulaTest) = “Null” Then

MsgBox “Mixed!”

Else

MsgBox FormulaTest

End If

End Sub

**The Font property**

Range("A1:B3").Font.Bold = True

Range("A1:B3").Font.Color = vbRed

Range("A1:B3").Font.Italic = True

Range("A1:B3").Font.FontStyle = "arial"

Range("A1:B3").Font.Underline = True

**The Interior property**

Range(“A1:B3”).Interior.Color = rgb(0,200,0)

**The Formula property**

‘represents the formula in a cell

Range(“A13”).Formula = “=SUM(A1:A12)”

‘if you want to concatenate a string(with quotes) to the formula, the following will display an error:

=SUM(A1:A12)&” Stores”

‘to fix, replace all single double quotes with two double quotes

”=SUM(A1:A12)&”” Stores”””

**The NumberFormat property**

Columns(“A:A”).NumberFormat = “0.00%”

‘more number formats can easily be seen in format cells dialog box

**The Select method**

‘selects a range on the active worksheet:

Range(“A1:C12”).Select

‘if Sheet1 contains the range you want to select, use the following statements to select the range:

Sheets(“Sheet1”).Activate

Range(“A1:C12”).Select

‘This statement activates Sheet1 and then selects the range:

Application.Goto Sheets(“Sheet1”).Range(“A1:C12”)

**The Copy and Paste methods**

‘copies range A1:A12 and pastes it to the same worksheet, beginning at cell C1:

Sub CopyRange()

Range(“A1:A12”).Select

Selection.Copy

Range(“C1”).Select

ActiveSheet.Paste

End Sub

‘the ff can also be used

Range(“A1:A12”).Copy Range(“C1”)

**The Clear method**

‘removes formatting only

Columns(“D:D”).ClearFormats

‘remove contents but leaves formatting

Columns(“D:D”).ClearContents

‘removes contents and formatting

Columns(“D:D”).Clear

**The Delete method**

‘When you delete a range, Excel shifts the remaining cells around to fill up the range you deleted.

Rows(“6:6”).Delete

‘When you delete a range that’s not a complete row or column, Excel needs to know how to shift the cells.

‘The following statement deletes a range and then fills the resulting gap by shifting the other cells to the left:

Range(“C6:C10”).Delete xlToLeft

‘The Delete method uses an argument that indicates how Excel should shift the remaining cells. In this case, I use a built-‘in constant (xlToLeft) for the argument. I could also use xlUp, another named constant.

**VBA functions**

Here are some of popular VBA built-in commands

|  |  |
| --- | --- |
| Function | What It Does |
| Abs | Returns a number’s absolute value. |
| Array | Returns a variant containing an array. |
| Choose | Returns a value from a list of items. |
| Chr | Converts an ANSI value to a string. |
| CurDir | Returns the current path. |
| Date | Returns the current system date. |
| DateAdd | Returns a date to which a specified time interval has been added — for example, one month from a particular date. |
| DateDiff | Returns an integer showing the number of specified time intervals between two dates — for example, the number of months between now and your birthday. |
| DatePart | Returns an integer containing the specified part of a given date —for example, a date’s day of the year. |
| DateSerial | Converts a date to a serial number. |
| DateValue | Converts a string to a date. |
| Day | Returns the day of the month from a date value. |
| Dir | Returns the name of a file or directory that matches a pattern. |
| Err | Returns the error number of an error condition. |
| Error | Returns the error message that corresponds to an error number. |
| Exp | Returns the base of the natural logarithm (e) raised to a power. |
| FileLen | Returns the number of bytes in a file. |
| Fix | Returns a number’s integer portion. |
| Format | Displays an expression in a particular format. |
| GetSetting | Returns a value from the Windows registry. |
| Hour | Returns the hours portion of a time. |
| InputBox | Displays a box to prompt a user for input. |
| InStr | Returns the position of a string within another string. |
| InStrRev | Returns the position of a string within another, from the end of a string. |
| Int | Returns the integer portion of a number. |
| IsArray | Returns True if a variable is an array. |
| IsDate | Returns True if an expression is a date. |
| IsEmpty | Returns True if a variable has not been initialized. |
| IsError | Returns True if an expression is an error value. |
| IsMissing | Returns True if an optional argument was not passed to a procedure. |
| IsNull | Returns True if an expression contains no valid data. |
| IsNumeric | Returns True if an expression can be evaluated as a number. |
| LBound | Returns the smallest subscript for a dimension of an array. |
| LCase | Returns a string converted to lowercase. |
| Left | Returns a specified number of characters from the left of a string. |
| Len | Returns the number of characters in a string. |
| Mid | Returns a specified number of characters from a string. |
| Minute | Returns the minutes portion of a time value. |
| Month | Returns the month from a date value. |
| MsgBox | Displays a message box and (optionally) returns a value. |
| Now | Returns the current system date and time. |
| Replace | Replaces a substring in a string with another substring. |
| RGB | Returns a numeric RGB value representing a color. |
| Right | Returns a specified number of characters from the right of a string. |
| Rnd | Returns a random number between 0 and 1. |
| Second | Returns the seconds portion of a time value. |
| Shell | Runs an executable program. |
| Space | Returns a string with a specified number of spaces. |
| Split | Splits a string into parts, using a delimiting character. |
| Sqr | Returns a number’s square root. |
| String | Returns a repeating character or string. |
| Time | Returns the current system time. |
| Timer | Returns the number of seconds since midnight. |
| TimeSerial | Returns the time for a specified hour, minute, and second. |
| TimeValue | Converts a string to a time serial number. |
| Trim | Returns a string without leading or trailing spaces. |
| TypeName | Returns a string that describes a variable’s data type. |
| UBound | Returns the largest available subscript for an array’s dimension. |
| UCase | Converts a string to uppercase. |
| Val | Returns the numbers contained in a string. |
| Weekday | Returns a number representing a day of the week. |
| Year | Returns the year from a date value. |

Examples:

Msgbox “today is “ & Date

Msgbox “string length of data in A1: ” & Len(Range(“A1”))

Msgbox “the current month is ” & MonthName(Month(Date))

Ucase(Range(“A1”))

**The GoTo Statement**

‘a programming construct that lets you jump to a line label (usually for ‘on error’ events but try to avoid as much as you can as it can result in a messy code)

Example:

Sub CheckUser()

UserName = InputBox(“Enter Your Name: “)

If UserName <> “John Doe” Then GoTo WrongName

MsgBox (“Welcome Steve...”)

Exit Sub

WrongName:

MsgBox “Sorry. Only John Doe can run this.”

End Sub

**The If-Then structure**

‘a programming construct that handles and delivers results for conditional statements

‘if-then-else sample

Dim checkgrade as double

checkgrade = Inputbox(“enter grade to check”)

if checkgrade >= 75 then

msgbox “passed”

else

msgbox “failed”

end if

‘if-then-elseif-else

Dim Msg As String

If Time < 0.5 Then

Msg = “Morning”

ElseIf Time >= 0.5 And Time < 0.75 Then

Msg = “Afternoon”

Else

Msg = “Evening”

End If

MsgBox “Good “ & Msg

**The Select Case structure**

‘an alternative to the if-then-else structure

Dim Quantity As Integer

Dim Discount As Double

Quantity = InputBox(“Enter Quantity: “)

Select Case Quantity

Case 0 To 24

Discount = 0.1

Case 25 To 49

Discount = 0.15

Case 50 To 74

Discount = 0.2

Case Is >= 75

Discount = 0.25

End Select

MsgBox “Discount: “ & Discount

**For-Next loops**

1. For next loop with a single step increment

Example:

Sub loopovercells()

Dim cnt As Integer

For cnt = 1 To 10

Cells(cnt, 1).Value = "loop " & cnt

Next cnt

End Sub

1. For next loop with step increment

Example:

Sub loopovercells()

Dim cnt As Integer

For cnt = 1 To 10 step 2

Cells(cnt, 1).Value = "loop " & cnt

Next cnt

End Sub

**Do-While loop**

‘for some cases that loops may handle more than just number brackets, you may want to try the ff:

‘this loop evaluates the condition first before executing the loop

‘example 1:

Dim i As Integer

i = 1

Do While i <= 10

Cells(i, 1).Value = "test"

i = i + 1

Loop

‘example 2 (create a row with numbers in your spreadsheet for this demo):

Do While ActiveCell.Value <> Empty

ActiveCell.Value = ActiveCell.Value \* 2

ActiveCell.Offset(0, 1).Select

**Do-Until loop**

‘same as the do-while loop, this loop can be used to iterate over a non-numeric condition

‘example 1:

Dim i As Integer

i = 1

Do until i = 10

Cells(i, 1).Value = "test"

i = i + 1

Loop

‘example 2 (create a row with numbers in your spreadsheet for this demo):

Do Until IsEmpty(ActiveCell.Value)

ActiveCell.Value = ActiveCell.Value \* 2

ActiveCell.Offset(1, 0).Select

Loop

**Looping through a Collection**

‘VBA supports yet another type of looping — looping through each object in a collection of objects. Recall that a ‘collection consists of a number of objects of the same type.

‘For example, Excel has a collection of all open workbooks (the Workbooks collection), and each workbook has a ‘collection of worksheets (the Worksheets collection). When you need to loop through each object in a collection, use ‘the For Each-Next structure.

‘Example loops through each worksheet in the active workbook and deletes the worksheet if it’s empty:

Sub DeleteEmptySheets()

Dim WkSht As Worksheet

Application.DisplayAlerts = False

For Each WkSht In ActiveWorkbook.Worksheets

If WorksheetFunction.CountA(WkSht.Cells) = 0 Then

WkSht.Delete

End If

Next WkSht

Application.DisplayAlerts = True

End Sub

‘example 2 (loop to hide all worksheets in the active workbook, except the active sheet).

Sub HideSheets()

Dim Sht As Worksheet

For Each Sht In ActiveWorkbook.Worksheets

If Sht.Name <> ActiveSheet.Name Then

Sht.Visible = xlSheetHidden

End If

Next Sht

End Sub

‘example 3 (unhides all worksheets)

Sub UnhideSheets()

Dim Sht As Worksheet

For Each Sht In ActiveWorkbook.Worksheets

Sht.Visible = xlSheetVisible

Next Sht

End Sub

3pm-5pm

INTRODUCTION TO FORMS

Forms Fundamentals

Using a Form

The Characteristics of a Form

INTRODUCTION TO CONTROLS

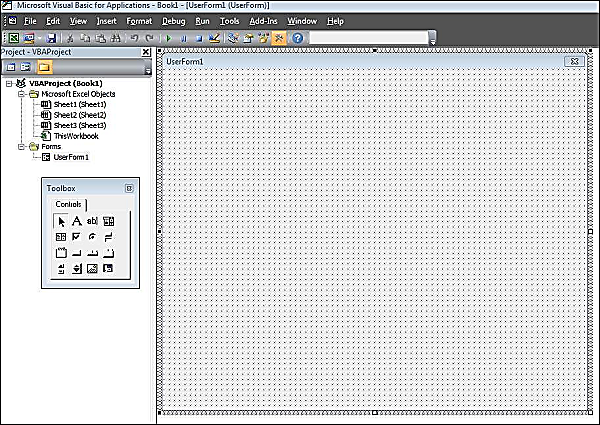
Controls Fundamentals

Adding a Control to a Container

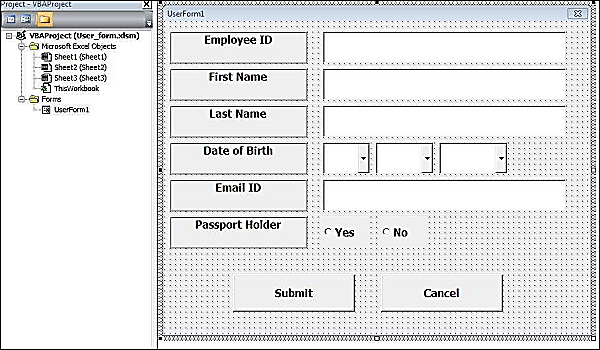
Control Selection

A User Form is a custom-built dialog box that makes a user data entry more controllable and easier to use for the user. In this chapter, you will learn to design a simple form and add data into excel.

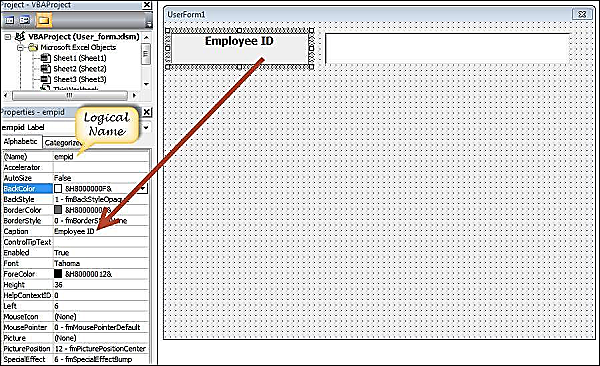
Step 1 − Navigate to VBA Window by pressing Alt+F11 and Navigate to "Insert" Menu and select "User Form". Upon selecting, the user form is displayed as shown in the following screenshot.



Step 2 − Design the forms using the given controls.



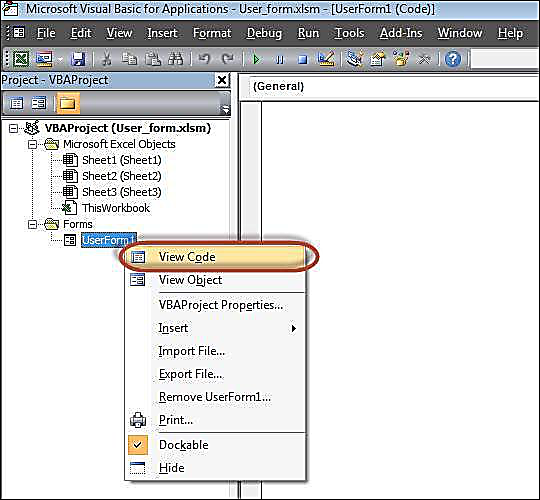
Step 3 − After adding each control, the controls have to be named. Caption corresponds to what appears on the form and name corresponds to the logical name that will be appearing when you write VBA code for that element.



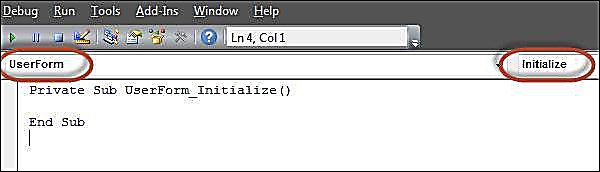
Step 4 − Following are the names against each one of the added controls.



Step 5 − Add the code for the form load event by performing a right-click on the form and selecting 'View Code'.



Step 6 − Select ‘Userform’ from the objects drop-down and select 'Initialize' method as shown in the following screenshot.



Step 7 − Upon Loading the form, ensure that the text boxes are cleared, drop-down boxes are filled and Radio buttons are reset.

Private Sub UserForm\_Initialize()

'Empty Emp ID Text box and Set the Cursor

txtempid.Value = ""

txtempid.SetFocus

'Empty all other text box fields

txtfirstname.Value = ""

txtlastname.Value = ""

txtemailid.Value = ""

'Clear All Date of Birth Related Fields

cmbdate.Clear

cmbmonth.Clear

cmbyear.Clear

'Fill Date Drop Down box - Takes 1 to 31

With cmbdate

.AddItem "1"

.AddItem "2"

.AddItem "3"

.AddItem "4"

.AddItem "5"

.AddItem "6"

.AddItem "7"

.AddItem "8"

.AddItem "9"

.AddItem "10"

.AddItem "11"

.AddItem "12"

.AddItem "13"

.AddItem "14"

.AddItem "15"

.AddItem "16"

.AddItem "17"

.AddItem "18"

.AddItem "19"

.AddItem "20"

.AddItem "21"

.AddItem "22"

.AddItem "23"

.AddItem "24"

.AddItem "25"

.AddItem "26"

.AddItem "27"

.AddItem "28"

.AddItem "29"

.AddItem "30"

.AddItem "31"

End With

'Fill Month Drop Down box - Takes Jan to Dec

With cmbmonth

.AddItem "JAN"

.AddItem "FEB"

.AddItem "MAR"

.AddItem "APR"

.AddItem "MAY"

.AddItem "JUN"

.AddItem "JUL"

.AddItem "AUG"

.AddItem "SEP"

.AddItem "OCT"

.AddItem "NOV"

.AddItem "DEC"

End With

'Fill Year Drop Down box - Takes 1980 to 2014

With cmbyear

.AddItem "1980"

.AddItem "1981"

.AddItem "1982"

.AddItem "1983"

.AddItem "1984"

.AddItem "1985"

.AddItem "1986"

.AddItem "1987"

.AddItem "1988"

.AddItem "1989"

.AddItem "1990"

.AddItem "1991"

.AddItem "1992"

.AddItem "1993"

.AddItem "1994"

.AddItem "1995"

.AddItem "1996"

.AddItem "1997"

.AddItem "1998"

.AddItem "1999"

.AddItem "2000"

.AddItem "2001"

.AddItem "2002"

.AddItem "2003"

.AddItem "2004"

.AddItem "2005"

.AddItem "2006"

.AddItem "2007"

.AddItem "2008"

.AddItem "2009"

.AddItem "2010"

.AddItem "2011"

.AddItem "2012"

.AddItem "2013"

.AddItem "2014"

End With

'Reset Radio Button. Set it to False when form loads.

radioyes.Value = False

radiono.Value = False

End Sub

Step 8 − Now add the code to the Submit button. Upon clicking the submit button, the user should be able to add the values into the worksheet.

Private Sub btnsubmit\_Click()

Dim emptyRow As Long

'Make Sheet1 active

Sheet1.Activate

'Determine emptyRow

emptyRow = WorksheetFunction.CountA(Range("A:A")) + 1

'Transfer information

Cells(emptyRow, 1).Value = txtempid.Value

Cells(emptyRow, 2).Value = txtfirstname.Value

Cells(emptyRow, 3).Value = txtlastname.Value

Cells(emptyRow, 4).Value = cmbdate.Value & "/" & cmbmonth.Value & "/" & cmbyear.Value

Cells(emptyRow, 5).Value = txtemailid.Value

If radioyes.Value = True Then

Cells(emptyRow, 6).Value = "Yes"

Else

Cells(emptyRow, 6).Value = "No"

End If

End Sub

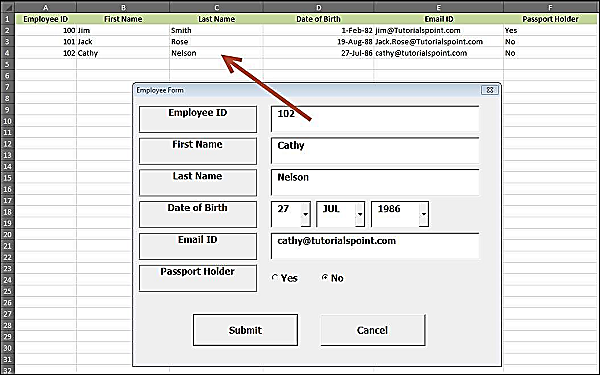
Step 9 − Add a method to close the form when the user clicks the Cancel button.

Private Sub btncancel\_Click()

Unload Me

End Sub

Step 10 − Execute the form by clicking the "Run" button. Enter the values into the form and click the 'Submit' button. Automatically the values will flow into the worksheet as shown in the following screenshot.



to call a userform from a macro

