Workshop 4x4 - Who's that GUI (Cheat Sheet)

This is where you can find out more about the functions that were used in this workshop's scripts. When creating new VBA scripts we often visit the following sites:

- Excel Easy
- Easy Excel VBA
- Code VBA
- Excel Macro Mastery
- And Google is always your friend

For more in-depth information head over to Microsoft VB Documentation or to Microsoft Excel VBA Documentation

Initializing Variables

Dim variableName

This method declares and allocates requeired space in memory. There are a few ways to declare a variables, you can simply declare a variable like this:

```
Dim objPan As New Object
```

Or maybe declare them in a batch, like so:

```
Dim objFlour, objEggs, objMilk, objOil As New Object
```

Another way is to set the initial value:

```
Dim intPancakes As Integer = 5
Dim strMessage As String = "Have some pancakes"
```

The most used types of variables in this workshop are the following: Boolean, String, Integer, Double, Object, Worksheet, Range and Array.

Array

An array can accept any type of variable. This declares a single-dimension array of Strings with undefined size.

```
Dim arrIngredients() As String
```

You can also declare the array with a specific size. Both lines will create the same array with space for 5 elements. Beware that element count starts at zero, hence it's zero-indexed.

```
Dim arrIngredients(4) As String
Dim arrIngredients(0 to 4) As String
```

Code Commenting

To comment your code, simply add a apostrophe (') at the beggining of the line.

Loops

If...Then...Else Statement

An If statement executes code when a condition is met (returns true).

```
If (condition1) Then
    'Do something
ElseIf (condition2) Then
    'Do something different
Else
    'Do something else
End If
```

Conditions can be specified by using any combination of the following keywords:

- True
- False
- And
- Or
- Not
- Is
- Nothing

For example, in the next snippet, the program only executes code when the variable rngDimension is not empty.

```
Dim rngDimension As Range

If Not rngDimension Is Nothing Then
    'Do something
End If
```

For Each...Next Statement

Repeats a group of statements for each element in a collection. In this snippet the code is executed for each item in folder.Files

```
For Each file In folder.Files
    'Do something
Next file
```

For...Next Statement

Repeats a group of statements a specified number of times. In the next snippet the code executes 100 times with a step of 1, meaning that the counter variable will be incremented by one unit at each iteration.

```
For i = 0 To 100 step 1
    'Do something
Next i
```

GUI Controls (Windows Forms Controls)

Button

The command button control is used to begin, interrupt, or end a process. When clicked, it invokes a command that has been written into its Click event procedure.

Combo Box

A combo box control combines the features of a text box and a list box. This control allows the user to select an item either by typing text into the combo box, or by selecting it from the list.

Generally, a combo box is appropriate when there is a list of suggested choices, and a list box is appropriate when you want to limit input to what is on the list. A combo box contains an edit field, so choices not on the list can be typed in this field.

Check Box

The check box control displays a check mark when it is selected. It is commonly used to present a Yes/No or True/False selection to the user. You can use check box controls in groups to display multiple choices from which the user can select one or more.

Excel specific

Range Object

Represents a cell, a row, a column, a selection of cells containing one or more contiguous blocks of cells, or a 3-D range.

Use Range (arg), where arg names the range, to return a Range object that represents a single cell or a range

of cells. The following example places the value of cell A1 in cell A5.

```
Worksheets("Sheet1").Range("A5").Value =
Worksheets("Sheet1").Range("A1").Value
```

Worksheet Object

Represents a worksheet. The Worksheet object is a member of the Worksheets collection. The Worksheets collection contains all the Worksheet objects in a workbook.

ThisWorkbook Property

Use this property to refer to the workbook that contains your macro code. **ThisWorkbook** is the only way to refer to an add-in workbook from inside the add-in itself. The **ActiveWorkbook** property doesn't return the add-in workbook; it returns the workbook that's calling the add-in.The **Workbooks** property may fail, as the workbook name probably changed when you created the add-in. **ThisWorkbook** always returns the workbook in which the code is running.

Application.ScreenUpdating Property

Turn screen updating off to speed up your macro code. You won't be able to see what the macro is doing, but it will run faster.

Remember to set the ScreenUpdating property back to True when your macro ends.

```
'Turn off application screen update
updateApplication.ScreenUpdating = False
```

Application.DisplayAlerts Property

The default value is True . Set this property to False to suppress prompts and alert messages while a macro is running; when a message requires a response, Microsoft Excel chooses the default response.

If you set this property to False , Microsoft Excel sets this property to True when the code is finished, unless you are running cross-process code.

```
'Turn off display alerts
Application.DisplayAlerts = False
```