VBA – Cheat Sheet

Creating a new VBA module

- Either **record the macro** or
- Press **Alt F11** to display the VB Editor (or select **Developer** tab and click **Visual Basic** button on the left)
- Click **Microsoft Excel Objects**
- Choose **Insert** | **Module** from the menu (you can also right-click on **Microsoft Excel Objects**).
- Type **Sub your_module_name()**. You can't use spaces, but you can use underscore.
- **End Sub** appears automatically.

Stepping through or debugging VBA code

- **F8** - Step through the code, line by line
- **F9** - Set or remove a break point
- **Ctrl+Shift+F9** - Remove all break points
- Use the word '**Stop**' within the code to end the module right there
- Click the **blue square icon** on the VB tool bar to stop the macro
- Hover over a variable to see its current value. You can also right-click the variable and choose '**Add Watch**' to watch its value in a separate window.

Selecting Cells, Sheets and Workbooks

Select a cell

```
Range("A8").Select
```

Select multiple cells

```
Range("A8, C5").Select
```

Select a cell range

```
Range("A1:A8").Select
```

Select a named cell or cell range

```
Range("my_range").Select
```

VBA – Cheat Sheet

Select a row or rows

```
Range("2:6").Select or Rows("2:6").Select
```

Select a column or columns

```
Range("B:G").Select or Columns("B:G").Select
```

Select a cell by row (8) and column number (1)

```
Cells(8,1).Select
```

Select a sheet

```
Sheets("Sheet2").Activate
```

Select a cell on a sheet

```
Sheets("Sheet2").Range("A8").Select
```

Select a cell on a sheet on a workbook

```
Workbooks("Book2.xlsx").Sheets("Sheet2").Range("A8").Select
```

Setting Properties

Set a value

```
Range("A1").Value = 48
Range("A1").Value = "Sample text"
Sheets("Sheet2").Range("A1").Value = "Sample text"
Workbooks("MyWorkbook.xlsx").Sheets("Sheet2").Range("A1").Value = "Sample text"
```

Clear the contents or clear the formatting

```
Range("A:A").Clear
Range("A:A").ClearContents
Range("A:A").ClearFormats
```

Text Formatting

```
Range("A1:A5").Font.Name = "Arial"
```

```
Range("A1:A5").Font.Size = 18
Range("A1:A5").Font.Italic = True
Range("A1:A5").Font.Bold = True
Range("A1:A5").Font.Underline = True
Range("A1:A5").Font.Color = vbRed
```

Colour

```
Range("A1:A5").Font.Color = vbRed
Range("A1:A5").Font.Color = RGB(255,0,0)
Range("A1:A5").Font.ColorIndex = 3 (3 means Red, can use colors 1-56)
```

Borders

```
Range("A1:A5").Borders.Value = 1
Range("A1:A5").Borders.Weight = xlThick
Range("A1:A5").Borders.LineStyle = xlContinuous
Range("A1:A5").Borders.ColorIndex = xlAutomatic
```

Interior (Fill)

```
Range("A1:A5").Interior.Color = RGB(32, 64, 96)
Range("A1:A5").Interior.TintAndShade = RGB(64, 96, 128)
```

Tab colour

```
Sheets("Sheet1").Tab.Color = RGB(255, 0, 0)
```

Border options

Selection.Borders()	.Value	.Weight	.LineStyle
xlEdgeLeft	0 (invisible)	xlHairline	xlContinuous
xlEdgeRight	1 (visible)	xlThin	xlDash
xlEdgeTop		xlMedium	xlDashDot
xlEdgeBottom		xlThick	xlDashDotDot
xlInsideVertical		Numerical value e.g. 3	xlDot
xlInsideHorizontal			xlDouble
			xlSlantDashDot
			xlLineStyleNone

ColorIndex

Color Values can be 1 to 56.

Type this (as a bit of practice) to generate a color index.

```
Sub PrintColorIndexTable()
    For i = 1 To 56
        With Cells(Int((i - 1) / 5) + 1, (i - 1) Mod 5 + 1)
            .Interior.ColorIndex = i
            .Value = i
            .HorizontalAlignment = xlCenter
        End With
    Next i
End Sub
```

The With Construct

If there are a number of properties that need to be set for one object, you can place all the properties inside a **With … End With** construct.

Example 1

```
With Selection.Font
   .Size = 18
   .Bold = True
End With
```

Example 2

```
With Selection.Border
   .Weight    = xlThick
   .LineStyle = xlDash
End With
```

Example 3

```
With ActiveCell
   .Borders.Weight = 3
   .Font.Bold      = True
   .Font.Size      = 18
End With
```

Example 4

```
With ActiveCell
   With .Borders
      .Weight    = 3
      .LineStyle = xlContinuous
   End With
   With .Font
      .Bold      = True
      .Size      = 18
   End With
End With
```

Variables

Declaration

```
Dim last_name As String, first_name As String, age As Integer
```

Example of use

```
last_name  = Range("A2")
first_name = Range("B2")
age        = Range("C2")
MsgBox first_name & " " & last_name & ", " & age & " years old"
```

Arrays

Arrays are arranged like pigeon holes. Each hole is referenced by row, then column, then depth

```
Dim array1(4)       As String     ' 1 dimensional array
Dim array2(4, 3)    As String     ' 2 dimensional array
Dim array3(4, 3, 2) As String     ' 3 dimensional array
```

NB. The first cell in an array is numbered zero. Therefore an array defined as (4) actually has 5 pigeon holes numbered 0, 1, 2, 3 and 4. Arrays can be used to store fixed values, values from cells, or results of calculations.

VBA – Cheat Sheet

Constants

A constant is defined like a variable, but as its name suggests, it doesn't change.

Constants are declared with **Const** rather than **Dim**.

```
Const commission_rate As Double = 4.5
```

Local variable vs Global variables

Variables declared within a **Sub … End Sub** are local.

Global variables are declared with **Global** rather than **Dim**.

```
Global myGlobalVar As Integer
```

Conditions

If … ElseIf … ElseIf … ElseIf … Else … End If

Basic structure

```
If [Condition 1] Then
   *** Do this ***
ElseIf [Condition 2] Then
   *** Do this instead ***
Else
   *** Otherwise do this ***
End If
```

Checking that the value in A1 is numeric

```
If IsNumeric(Range("A1")) Then
   *** Do your stuff ***
Else
   MsgBox "Your entry" & Range("A1") & " is not valid !"
   Range("A1").ClearContents
End If
```

Other variations to check: **IsDate**, **IsEmpty**, **IsMissing**

VBA – Cheat Sheet

Within an IF condition you can use:

The standard comparison operators (=, <, <=, >, >=, <>).

```
If my_number >= 5 And my_number <= 10 Then …
```

Logical operations such as AND, OR and NOT.

```
If Range("A1") ="QLD" AND Range("B1") = "Admin" …
If Not IsNumeroc(Range("A1")) Then …
```

Wildcard comparisons using LIKE with *, #, ? or [].

```
If my_variable LIKE "*st*"  Then …
If my_variable LIKE "123##" Then …
If my_variable LIKE "P???T" Then …
If my_variable LIKE "[abc]" Then …
If my_variable LIKE "[a-g]" Then …
If my_variable LIKE "[234]" Then …
If my_variable LIKE "[1-9]" Then …
If my_variable LIKE "[!abc]" Then …
```

Case

```
Select Case my_variable
   Case Is = 1 *** Do This ***
   Case Is = 2 *** Do This ***
   Case Is = 3 *** Do This ***
   Case Is = 4 *** Do This ***
   Case Is = 5 *** Do This ***
   Case Else   *** Do This ***
End Select
```

Other variations

```
Case Is >= 6
Case Is 6, 7, 8
Case <> 10, 11
Case 6-10
```

Iteration (Loops)

While ... Wend

```
x=1
While x <= 10
   *** Do something using x ***
   x = x + 1
Wend
```

Do While ... Loop

```
Do While [condition is true]
   *** Do your stuff ***
   If [condition is true]    ' Optional
      Exit Do
   End If
Loop
```

Do Until ... Loop

```
Do Until [condition is true]
   *** Do your stuff ***
Loop
```

For ... Next

```
For I = 1 to 10
   *** Do your stuff using i ***
Next
```

For Each ... Next

```
For Each _____ In _____
   *** Do your stuff ***
Next
```

Example - Red and Black Checkerboard

```
Sub Red_and_Black()
   Const TotalColumns As Integer = 7, TotalRows As Integer = 5
   Dim offset_row As Integer, offset_col As Integer

   offset_row = ActiveCell.Row - 1
   offset_col = ActiveCell.column - 1

'Colour the cells red or black
   For r = 1 To TotalRows          ' Row number
      For c = 1 To TotalColumns    ' Column number
         If (r + c) Mod 2 = 0 Then
            Cells(r + offset_row, c + offset_col).Interior.Color = RGB(255, 0, 0)
         Else
            Cells(r + offset_row, c + offset_col).Interior.Color = RGB(0, 0, 0)
         End If
      Next
   Next

'Resize column widths and row heights to makecells square
   For c = 1 To TotalColumns
      Columns(offset_col + c).ColumnWidth = 3
   Next

   For r = 1 To TotalRows
      Rows(offset_row + r).RowHeight = 21
   Next
End Sub
```

Procedures, Sub procedures Arguments & Functions

Public vs Private

First some terminology. '**Sub**' means the same as '**Procedure**' or '**Module**'.

Procedures are **Public** by default which means they are accessible from any module.

Private modules are only accessible from within the current module

To call another module simply write the module name.

```
Private Sub Warning()
   MsgBox "Caution !!!"
End Sub

Sub macro1()
   If Range("A1") = "" Then
      Warning
   End If
End Sub
```

Arguments

Arguments allow you to pass data from one module to another and back again.

```
Private Sub Warning(var_text As String)
   MsgBox "Caution : " & var_text & " !"
End Sub

Sub macro2()
   If Range("A1") = "" Then
      Warning "Empty cell"
   ElseIf Not IsNumeric(Range("A1")) Then
      Warning "Value is not numeric"
   End If
End Sub
```

Variables may be prefixed with 'Optional'.

VBA – Cheat Sheet

Functions

```
Function square(var_number)
   square = var_number ^ 2
End Function

Sub macro3()
   Dim myNum As Double, result As Double
   myNum = Range("A1").Value
   result = square(myNum)
   MsgBox "The square of " & myNum & " is " & result
End Sub
```

Dialog boxes

MsgBox

```
MsgBox "*** Your message ***"
```

```
MsgBox ( [Text], [Buttons], [Title] )

MsgBox("Are you sure?", vbYesNo, "Confirm")

Example:
Sub delete_cell()
   If MsgBox("Are you sure?", vbYesNo, "Confirm") = vbYes Then
      Activecell.ClearContents
      MsgBox "The contents have been deleted !"
   End If
End Sub
```

VBA – Cheat Sheet

Button option	Description	Numerical value
vbOKOnly	It displays a single **OK** button.	0
vbOKCancel	It displays two buttons **OK** and **Cancel**.	1
vbAbortRetryIgnore	It displays three buttons **Abort**, **Retry**, and **Ignore**.	2
vbYesNoCancel	It displays three buttons **Yes**, **No**, and **Cancel**.	3
vbYesNo	It displays two buttons **Yes** and **No**.	4
vbRetryCancel	It displays two buttons **Retry** and **Cancel**.	5
vbCritical	It displays a **Critical Message** icon.	16
vbQuestion	It displays a **Query** icon.	32
vbExclamation	It displays a **Warning Message** icon.	48
vbInformation	It displays an **Information Message** icon.	64
vbDefaultButton1	First button is treated as default.	0
vbDefaultButton2	Second button is treated as default.	256
vbDefaultButton3	Third button is treated as default.	512
vbApplicationModal	(Forces user to answer before continuing to use Excel).	0
vbSystemModal	(Forces user to answer before continuing to use any program on the computer - dialog box in foreground).	4096
vbMsgBoxHelpButton	Adds a Help button to the message box.	
vbMsgBoxSetForeground	Message box appears in foreground.	
vbMsgBoxRight	Right-aligns text.	
vbMsgBoxRtlReading	Text appears right-to-left.	

You can combine different elements, e.g.

```
MsgBox("Are you sure?", vbYesNo + vbQuestion + vbDefaultButton3, "Confirm")
```

You can add line returns using character 10, e.g.

```
MsgBox("Are" & Chr(10) & "you" & Chr(10) & "sure?", vbYesNo, "Confirm")
```

VBA – Cheat Sheet

Input Box

Example 1

```
InputBox("BodyText", "TitleText", "DefaultValue")   ' Default value optional
```

Example 2

```
result = InputBox("BodyText", "TitleText")
```

Example 3

```
If InputBox("BodyText", "TitleText") = "" Then …
```

Events

Running macro code as soon as the workbook opens

1. In the sidebar of the VB Editor, double-left-click on **ThisWorkbook** (under Microsoft Excel Objects)
2. Click on the drop-down box that says **(General)** and change to **Workbook**.
3. Click on the second drop-down box that says **(declarations)** and change to **Open**.
4. Write your code between **Private Sub Workbook_Open** and **End Sub**.

Running macro code immediately before the workbook closes, prints or saves

As above but substitute **BeforeClose**, **BeforePrint** or **BeforeSave** in step 3. Each of these events has Cancel or Success variables which can be used if required.

Running macro code immediately before a double-click, right-click or sheet change

As above but substitute **SheetBeforeDoubleClick**, **SheetBeforeRightClick** or **SheetChange** in step 3. Each of these events has Cancel or Success variables which can be used if required.

There are many events each with different behaviours and options. Explore.

Forms

Add a new form

- In the VB editor, right-click **Microsoft Excel Objects** then choose **Insert** | **UserForm**.
- A blank user form and **Toolbox** will appear.
- You also need to make sure the **Properties** panel is displayed (select **View** | **Properties**). There are properties for everything from name, visibility and colour to height, width, positioning and special effects.
- Try changing the **Caption** from **UserForm1** to **myUserForm**.

Add Events

- Double-click the user form to display the Code window.
- The first drop-down box will show 'UserForm'. The second drop-down box shows the selected Event. Events are triggered when something specific happens such as click, double-click, right-click and when something changes.
- The **Initialize** event sets the initial properties when the form is first activated.

```
Private Sub UserForm_Initialize()
   my_userform.Height = 100
   my_userform.Width  = 100
End Sub
```

- Within the form, this type of code can be optimized to

```
Private Sub UserForm_Initialize()
Me.Height = 100
Me.Width = 100
End Sub
```

Add Form Controls

Commonly used Form controls include Label, Text Box, Command Button, Combo Box, Check Box and Option Buttons.

- Click the control within the toolbox then click on the form to insert it.
- Drag to reposition it or resize it.
- Double-click the control to display the Code window.
- Events and properties differ for each type of control.

VBA – Cheat Sheet

Code Examples – Textbox & Command Button

```
Private Sub CommandButton_validate_Click()    ' Validate on command button click
   Range("A1") = Textbox_number.Value         ' Textbox_Number is the name of the
                                              ' textbox control
   Unload Me                                  ' Close form
End Sub
```

```
Private Sub Textbox_number_Change()
   If IsNumeric(Textbox_number.Value) Then
      Label_error.Visible = False
      Me.Width = 150
   Else
      Label_error.Visible = True
      Me.Width = 250
   End If
End Sub
```

Code Example – 3 Checkboxes and 3 corresponding cells

```
Private Sub CheckBox1_Click() 'Number 1
   If CheckBox1.Value = True Then Range("A2") = "Checked"
   Else Range("A2") = "Unchecked"
   End If
End Sub

Private Sub CheckBox2_Click() 'Number 2
   If CheckBox2.Value = True Then Range("B2") = "Checked"
   Else Range("B2") = "Unchecked"
   End If
End Sub

Private Sub CheckBox3_Click() 'Number 3
   If CheckBox3.Value = True Then Range("C2") = "Checked"
   Else Range("C2") = "Unchecked"
   End If
End Sub
```

And in reverse ...

```
Private Sub UserForm_Initialize() 'Check box if "Checked"
   If Range("A2") = "Checked" Then CheckBox1.Value = True
   End If

   If Range("B2") = "Checked" Then CheckBox2.Value = True
   End If

   If Range("C2") = "Checked" Then CheckBox3.Value = True
   End If
End Sub
```

Option Buttons

A user may only select one option button in a group. First create a frame (from the Toolbox) then place option buttons inside the form, and rename if required.