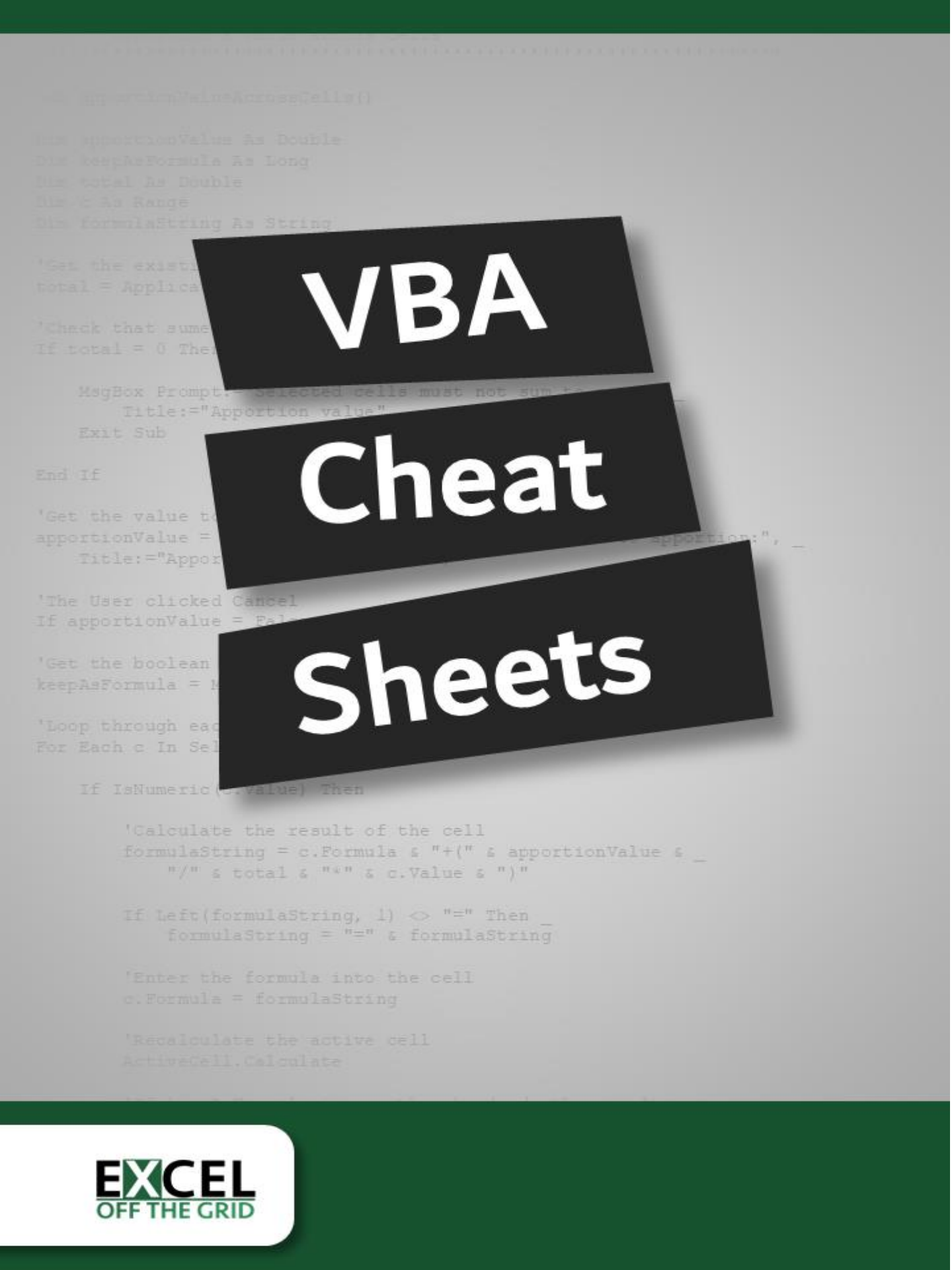


EXCEL
OFF THE GRID

# VBA Cheat Sheet – Workbooks

## About workbooks

- VBA assumes active workbook, unless explicitly stated
- Workbook must be open to use it

## Reference workbooks

**Create a workbook variable**

```
Dim wb As Workbook
```

**Assign a workbook to workbook variable**

```
'Reference a workbook by name
Set wb = Workbooks("WorkbookName.xlsx")

'Reference the workbook containing the macro
Set wb = ThisWorkbook

'Reference the active workbook
Set wb = ActiveWorkbook

'Reference a workbook by order opened
Set wb = Workbooks(1)

'Reference the last workbook opened
Set wb = Workbooks(Workbooks.Count)
```

## Create new workbooks

**Create new workbook**

```
Workbooks.Add
```

**Create new workbook and assign to variable**

```
Set wb = Workbooks.Add
```

## Opening workbooks

**Open a workbook**

```
Workbooks.Open ("C:\FilePath\Name.xlsx")
```

**Open a workbook and assign to variable**

```
Set wb = Workbooks.Open("C:\FilePath\Name.xlsx")
```

**Open a workbook and with Passwords (File Open & Modify)**

```
Set wb = Workbooks.Open(Filename:="C:\Path\Name.xlsx", _
    Password:="fileOpenPassword", _
    WriteResPassword:="modifyPassword")
```

**Open a workbook as read-only and update links on open**

```
Set wb = Workbooks.Open(Filename:="C:\Path\Name.xlsx", _
    readOnly:=True, _
    UpdateLinks:=True)
```

## Save workbooks

**Save a workbook**

```
wb.Save
```

**Save a workbook with a new name**

```
wb.SaveAs "C:\FilePath\NewWorkbookName.xlsx"
```

**Save a copy of the workbook**

```
wb.SaveCopyAs "C:\FilePath\NewWorkbookName.xlsx"
```

## Protect and unprotect workbooks

**Protect workbook without password**

```
wb.Protect
```

**Unprotect workbook without password**

```
wb.Unprotect
```

**Protect workbook with password**

```
wb.Protect "Password"
```

**Unprotect workbook with password**

```
wb.Unprotect "Password"
```

## Activate a workbook

```
wb.Activate
```

## Closing workbooks

**Close without saving changes**

```
wb.Close False
```

**Close and save changes**

```
wb.Close True
```

## Loop through workbooks

```
Dim wb As Workbook
For Each wb In Workbooks
    'Action to perform on each workbook
    Debug.Print wb.Name
Next wb
```

# VBA Cheat Sheet – Worksheets

## About worksheets

- VBA assumes active workbook, unless explicitly stated
- VBA assumes active worksheet, unless explicitly stated

## Reference worksheets

### Create a worksheet variable

```
Dim ws As Worksheet
```

### Assign a worksheet to a worksheet variable

```
'Reference a worksheet by name
Set ws = Worksheets("SheetName")

'Reference the active worksheet
Set ws = ActiveSheet

'Reference a worksheet by position
Set ws = Worksheets(1)

'Reference the last worksheet
Set ws = Worksheets(Worksheets.Count)
```

## Add worksheets

### Add a worksheet

```
'Add a worksheet
Worksheets.Add

'Add a worksheet and assign to variable
Set ws = Worksheets.Add
```

### Add a worksheet in a specific location

```
'Add a worksheet
Worksheets.Add

'Add 4 worksheets
Worksheets.Add Count:=4

'Add a worksheet before a worksheet
Worksheets.Add Before:=Worksheets(1)

'Add a worksheet after a worksheet
Worksheets.Add After:=Worksheets(1)

'Add a worksheet to the end
Worksheets.Add After:=Worksheets(Worksheets.Count)
```

## Delete worksheets

### Delete a worksheet

```
ws.Delete
```

### Delete a worksheet without displaying a warning message

```
Application.DisplayAlerts = False
ws.Delete
Application.DisplayAlerts = True
```

## Worksheet visibility

There are 3 states of visibility

```
'Visible
Worksheets("SheetName").Visible = xlSheetVisible

'Hidden
Worksheets("SheetName").Visible = xlSheetHidden

'Very Hidden - not visible in hidden sheets list
Worksheets("SheetName").Visible = xlSheetVeryHidden
```

## Rename worksheet

```
ws.Name = "SheetName"
```

## Activate worksheet

```
Worksheets("SheetName").Activate
```

## Worksheet protection

### Protect and unprotect worksheets

```
'Protect a worksheet
Worksheets("SheetName").Protect

'Protect a worksheet with a password
Worksheets("SheetName").Protect "password"

'Unprotect a worksheet without a password
Worksheets("SheetName").Unprotect

'Unprotect a worksheet with a password
Worksheets("SheetName").Unprotect "password"
```

## Copy worksheets

```
'Copy worksheets
ws.Copy

'Move a worksheet in front of another worksheet
ws.Copy Before:=Worksheets("AnotherSheetName")

'Move a worksheet after another worksheet
ws.Copy After:=Worksheets("AnotherSheetName")

'Copy multiple worksheets
Worksheets(Array("SheetName", "SheetName2", _
    "SheetName3")).Copy
```

## Loop through workbooks

### Loop through each worksheet

```
Dim ws As Worksheet
For Each ws In ActiveWorkbook.Worksheets
    'Action to perform on each worksheet
Next ws
```

### Loop through selected worksheets

```
Dim ws As Worksheet
For Each ws In ActiveWindow.SelectedSheets
    'Action to perform on each worksheet
Next ws
```

### Apply single sheet actions to selected worksheets

```
Dim ws As Worksheet
Dim sheetArray As Variant
Set sheetArray = ActiveWindow.SelectedSheets
For Each ws In sheetArray
    ws.Select
    'Action to perform on each worksheet
Next ws
sheetArray.Select
```

# VBA Cheat Sheet – Ranges

## Declare a range

All ranges on this page use a Range variable called `rng`

```
Dim rng As Range
```

## Setting a range

```
'Set range on ActiveSheet (single cell) - Method 1
Set rng = Range("A1")

'Set range on ActiveSheet (single cell) - Method 2
Set rng = ActiveSheet.Range("A1")

'Set range based on another sheet
Dim ws As Worksheet
Dim rng As Range
Set ws = Sheet("SheetName")
Set rng = ws.Range("A1")

'Set a range (Multiple Cells)
Set rng = Range("A1:D4")

'Set a named range
Set rng = Range("NamedRange")

'Set rng to used range
Set rng = ws.UsedRange

'Set cell inside a range (Result B11)
Set rng = Range("A10:D14").Cells(2, 2)

'Set cell inside active worksheet (Result B2)
Set rng = Cells(2, 2)

'Set entire column
Set rng = Range("C1:D2").EntireColumn
Set rng = Range("C:D")
Set rng = Columns("C:D")

'Set entire row
Set rng = Range("C1:D2").EntireRow
Set rng = Range("3:4")
Set rng = Rows("3:4")
```

## Select & activate ranges

**Select v Activate:** Select = Multiple objects | Activate = one object

```
'Select a Range
rng.Select

'Active a range
rng.Activate
```

## Named ranges

```
'Create workbook named range
Dim wb As Workbook
Set wb = Workbooks("WorkbookName.xlsx")
wb.Names.Add Name:="WbRange1", _
    RefersTo:="=Sheet1!$A$1:$D$4"

'Create worksheet named range
Dim wb As Workbook
Dim ws As Worksheet
Set wb = Workbooks("WorkbookName.xlsx")
Set ws = wb.Sheets("Sheet1")
ws.Names.Add Name:="WsRange1", _
    RefersTo:="=Sheet1!$A$1:$D$4"
```

## Common methods & properties

### Changing values

```
'Change value of a range
rng.Value = "Text here"

'Set variable to cell value
Dim myVar As String
myVar = rng.Value
```

### Hiding / unhiding rows and columns

```
'Hide Rows / Columns
rng.EntireColumn.Hidden = True
rng.EntireRow.Hidden = True

'Unhide Rows / Columns
rng.EntireColumn.Hidden = False
rng.EntireRow.Hidden = False
```

### Counting cells, rows and columns

```
'Count Rows / Columns / Cells
Dim count As Long
count = rng.Rows.count
count = rng.Columns.count
count = rng.Cells.count
```

### Insert cells, rows and columns

```
'Insert rows or columns
rng.Insert

'Insert cells
rng.Insert Shift:=xlDown
rng.Insert Shift:=xlRight
```

### Delete cells, rows and column

```
'Delete rows or columns
rng.Delete

'Delete cells
rng.Delete Shift:=xlUp
rng.Insert Shift:=xlLeft
```

### Copy and paste ranges

```
'Copy and paste everything
rng.Copy
rng2.Paste

'Cut and paste everything
rng.Cut
rng2.Paste

'Copy and paste values only
rng.Copy
rng2.PasteSpecial Paste:=xlPasteValues

'Copy and paste formats only
rng.Copy
rng.PasteSpecial Paste:=xlPasteFormats

'Copy and paste with clipboard
rng.Copy Destination:=rng2

'Copy values only
rng.Value = rng2.Value
```

# VBA Cheat Sheet - Logic

## If statements

- **If** and **Then** are needed
- **EndIf** is needed when not a single line of code
- VBA stops at the first true condition found
- **If**s can be nested inside each other
- There can be any number of **ElseIf** statements

**If** (single line):

```
If [condition is true] Then [do thing if true]
```

**If** (multiple lines):

```
If [condition is true] Then
  [do thing if true]
EndIf
```

**If** with multiple conditions:

```
If [condition1 is true] And [condition2 is true] Then
  [do thing if true]
EndIf
```

**If Else**:

```
If [condition is true] Then
  [do thing if true]
Else
  [do something else]
EndIf
```

**If ElseIf Else**:

```
If [condition1 is true] Then
  [do thing if condition1 is true]
ElseIf [condition2 is true]
  [do thing if condition2 is true]
Else
  [do something else]
EndIf
```

**Nested If**:

```
If [condition1 is true] Then
  If [condition2 is true] Then
    [do thing if condition1 & 2 are true]
  Else
    [do thing if condition1 is true &
        condition2 is false]
  EndIf
Else
  [do thing if condition1 is false]
EndIf
```

## Comparison operators

| | |
|---|---|
| = | Equal to |
| <> | Not equal to |
| < | Less than |
| > | Greater than |
| <= | Less than or equal to |
| >= | Greater than or equal to |

## Logic operators

| | |
|---|---|
| And | Both conditions must be true.<br>`If a < 100 And b > 100 Then` |
| Or | Either condition can be true.<br>`If a < 100 Or b > 100 Then` |
| Xor | Only one condition can be true.<br>`If a < 100 Xor b > 100 Then` |
| Not | Reverses the True or False result.<br>`IF Not a < 100 Then` |

## Select case

- There can be any number of **Case** statements
- **Case Else** is optional, and is used to catch any item not meeting any other condition.

```
Select Case [variable]
  Case [Condition1]
    [do thing if Condition1 = variable]
  Case [Condition2]
    [do thing if Condition2 is variable]
  Case [ConditionX]
    [do thing if ConditionX is variable]
  Case Else
    [do thing if no other conditions met]
End Select
```

Comparison operators use **Case Is** instead of **Case**.

```
Case Is >= 100
```

Don't need to use Case Is with equals ( = )

Use **Case** with number ranges

```
Case 50 To 100
```

**Case** with multiple conditions:

```
Case [Condition1], [Condition2], [ConditionX]
```

## Case sensitive vs non-case sensitive

Case sensitive - at the top of the module use:

```
Option Compare Binary
```

Non-case sensitive - at the top of the module use:

```
Option Compare Text
```

# VBA Cheat Sheet - Loops

## For Loop

- Cycles through a sequence of numbers
- Use **Step** to increment by a number other than 1
- Use **Exit For** to leave a loop early
- If using **For Loop** to delete objects always use **Step -1**

**For Loop**

```
Dim i as Integer
For i = 1 To 100
  Range("A" & i) = i
Next i
```

**For Loop** with **Step**

```
Dim i as Integer
For i = 1 To 100 Step 5
  Range("A" & i) = i
Next i
```

**For Loop** in reverse (**Step -1**)

```
Dim i as Integer
For i = 100 To 1 Step -1
  Range("A" & i) = i
Next i
```

**For Loop** with **Exit For**

```
Dim i as Integer
For i = 1 To 100
  Range("A" & i) = i
  If i > 20 Then Exit For
Next i
```

Nested **For Loop**

```
Dim i as Integer
Dim j as Integer

For i = 1 To 10
  For j = 1 To 5
    Cells(i, j) = i + j
  Next j
Next i
```

## For Each Loop

- Cycles through each object in a collection
- Use **Exit For** to leave a loop early

**For Each Loop Structure**

```
Dim [Variable] as [Object]
For Each [Variable] In [Object Collection]
  [Thing to do for each object]
Next [Variable]
```

**For Each Loop** through cells in Range

```
Dim c As Range
For Each c In ActiveSheet.Range("A1:B4")
  c.Value = c.Address
Next c
```

**For Each Loop** through worksheets in workbook

```
Dim ws As Worksheet
For Each ws In ActiveWorkbook.Sheets
  MsgBox ws.Name
Next ws
```

**For Each Loop** with **Exit For**

```
Dim c As Range
For Each c In ActiveSheet.Range("A1:B4")
  c.Value = c.Row
  If c.Value >= 3 Then MsgBox "Exit here"
  Exit For
Next c
```

## Do While

- Loops while the condition remains true
- 2 forms: (1) check condition at start (2) check condition at end
- Exit loop by using **Exit Do**

**Check condition at start**

If condition is false at the start, code in loop never executes.

Structure:

```
Do While [condition1]
  [Thing to do]
Loop
```

Example:

```
Dim i As Integer
i = 1
Do While i < 10
  MsgBox i
  i = i + 1
Loop
```

**Check condition at end**

Code in the loop is executed at lease once

Structure:

```
Do
  [Thing to do]
Loop While [condition1]
```

Example:

```
Dim i As Integer
i = 1
Do
  MsgBox i
  i = i + 1
Loop While i < 10
```

## Do Until

- Loops until a condition is true
- 2 forms: (1) check condition at start (2) check condition at end
- Exit loop by using **Exit Do**

**Check condition at start**

If condition is true at the start, code in loop never executes.

Structure:

```
Do Until [condition1]
  [Thing to do]
Loop
```

Example:

```
Dim i As Integer
i = 1
Do Until i > 5
  MsgBox i
  i = i + 1
Loop
```

**Check condition at end**

Code in the loop is executed at least once

Structure:

```
Do
  [Thing to do]
Loop Until [condition1]
```

Example:

```
Dim i As Integer
i = 1
Do
  MsgBox i
  i = i + 1
Loop Until i > 5
```

## While

- Works the same as Do Until
- Condition is checked at the start

Structure:

```
While [condition1]
  [Thing to do]
Wend
```

Example:

```
Dim i As Integer
i = 10
While i < 10
  MsgBox i
  i = i + 1
Wend
```

# VBA Cheat Sheet – Tables (page 1)

## About Tables

- Tables are known as ListObjects in VBA

## Reference Tables

### Create a Table variable

```
Dim ws As Worksheet
Dim tbl As ListObject

Set ws = Sheets("Sheet1")
Set tbl = ws.ListObjects("myTable")
```

### Reference parts of a Table

```
'Entire Table range
tbl.Range.Select

'Table data range only
tbl.DataBodyRange.Select

'Single cell within a Table (row 2, column4)
tbl.DataBodyRange(2, 4).Select

'Table column including header by position
tbl.ListColumns(2).Range.Select

'Table column including header by name
tbl.ListColumn("ColumnName").Range.Select

'Table column, data only, by position
tbl.ListColumns(2).DataBodyRange.Select

'Table column, data only, by name
tbl.ListColumns("ColumnName").DataBodyRange.Select

'Entire header row
tbl.HeaderRowRange.Select

'Single cell within header row
tbl.HeaderRowRange(2).Select

'Entire total row
tbl.TotalsRowRange.Select

'Single cell within total row
tbl.TotalsRowRange(2).Select

'Single row of data
tbl.ListRows(2).Range.Select

'Reference parts of table using range object
ws.Range("myTable[ColumnName]").Select
```

## Converting to/from a Table

### Convert current region to Table

```
Dim tableName As String
Dim tableRange As String
tableName = "myTable"
tableRange = "A1:D10"
ws.ListObjects.Add(SourceType:=xlSrcRange, _
    Source:=Range(tableRange), _
    xlListObjectHasHeaders:=xlYes _
    ).Name = tableName
```

### Convert Table back to range

```
tbl.Unlist
```

## Changing Tables

### Change name of Table

```
tbl.Name = "myNewTable"
```

### Resizing a Table

```
tbl.Resize Range("$A$1:$D$10")
```

### Remove auto filter

```
tbl.ShowAutoFilterDropDown = False
```

### Set Table style

```
tbl.TableStyle = "TableStyleLight15"
```

### Remove row stripes

```
tbl.ShowTableStyleRowStripes = False
```

### Hide header row

```
tbl.ShowHeaders = False
```

## Columns & rows

### Add columns & rows

```
'Add column at the end
tbl.ListColumns.Add

'Add column in position 2
tbl.ListColumns.Add Position:=2

'Add row at bottom
tbl.ListRows.Add

'Add row at position 2
tbl.ListRows.Add Position:=2
```

### Delete columns & rows

```
'Delete column 2
tbl.ListColumns(2).Delete

'Delete a column by name
tbl.ListColumns("ColumnName").Delete

'Delete row 2
tbl.ListRows(2).Delete

'Delete multiple rows
tbl.Range.Rows("2:3").Delete
```

### Count columns & rows

```
'Count columns
tbl.ListColumns.Count

'Count rows
tbl.ListRows.Count
```

# VBA Cheat Sheet – Tables (page 2)

## Loop through Tables

**Loop through all tables on a worksheet**

```
Dim ws As Worksheet
Dim tbl As ListObject
Set ws = ActiveSheet
For Each tbl In ws.ListObjects
    'Action to perform for each table
Next tbl
```

**Loop through all tables in a workbook**

```
Dim wb As Workbook
Dim ws As Worksheet
Dim tbl As ListObject
Set wb = ActiveWorkbook
For Each ws In wb.Worksheets
    For Each tbl In ws.ListObjects
        'Action to perform for each table
    Next tbl
Next ws
```

## Table total row

**Display total row with value in last column**

```
tbl.ShowTotals = True
```

**Add calculation to other columns**

```
'Change calculation by column position
tbl.ListColumns(2).TotalsCalculation = _
    xlTotalsCalculationAverage

'Change calculation by column name
tbl.ListColumns("ColumnName").TotalsCalculation = _
    xlTotalsCalculationAverage
```

**Other calculation types**

```
xlTotalsCalculationNone
xlTotalsCalculationAverage
xlTotalsCalculationCount
xlTotalsCalculationCountNums
xlTotalsCalculationMax
xlTotalsCalculationMin
xlTotalsCalculationSum
xlTotalsCalculationStdDev
xlTotalsCalculationVar
```

## Miscellaneous

**Enter data into table from array**

```
Dim myArray As Variant
myArray = Range("A2:D2")
tbl.ListRows(2).Range.Value = myArray
```

**Simulate ActiveTable object**

If active cell is not in a Table, then ActiveTable Is Nothing = True

```
Dim ActiveTable As ListObject
On Error Resume Next
Set ActiveTable = ActiveCell.ListObject
On Error GoTo 0

If ActiveTable Is Nothing Then
    'ActiveCell not in Table
Else
    'ActiveCell is in Table
End If
```

**Show Table data entry form**

Only works if Table starts at A1

```
Dim ws As Worksheet
Set ws = ActiveSheet
ws.ShowDataForm
```

# VBA Cheat Sheet – Names

## About Names

- Names can refer to ranges, formulas, constants and arrays
- Can have workbook or worksheet scope
- Print_Area is the name used for the print range

## Reference names

**Create Name variable**

```
Dim wb As Workbook
Dim nm As Name
Set wb = ActiveWorkbook
Set nm = wb.Names("myNamedConstant")
```

**Reference parts of a Name**

```
'Name of a Name
MsgBox nm.Name

'RefersTo part of a Name
MsgBox nm.RefersTo
```

## Add Names

**Create Name with workbook scope (use wb.Names)**

```
wb.Names.Add Name:="myNamedRange", _
    RefersTo:="=Sheet1!$C$1:$D$5"
```

**Create Name with worksheet scope (use ws.Names)**

```
ws.Names.Add Name:="myNamedRange", _
    RefersTo:="=Sheet1!$C$1:$D$5"
```

**Create Name with a formula**

Formulas must be created using the R1C1 reference method

```
wb.Names.Add Name:="myNamedFormula", _
    RefersTo:="=SUM(Sheet1!R2C2:R4C4)"
```

**Create Name with a constant**

```
wb.Names.Add Name:="myNamedConstant", _
    RefersTo:="=20%"
```

## Delete a Name

**Delete Name**

If name macthes will delete workbook and worksheet Names

```
wb.Names("myNamedRange").Delete
```

**Delete worksheet scoped Name**

Only deletes the worksheet scoped Name

```
wb.Names("Sheet1!myNamedRange").Delete
```

## Change a Name

**Change name and Refers to of a Name**

```
nm.Name = "changedMyNamedConstant"
nm.RefersTo = "=50%"
```

## Name Visibility

Names can be hidden from the Name Manager.

```
Names("myNamedRange").Visible = False
```

## Loop Through Names

**For Each through all Names**

```
Dim nm As Name
For Each nm In wb.Names
    'Action to perform for each name
Next
```

# VBA Cheat Sheet – Filters

## About Filters

- For many AutoFilter actions need to check if AutoFilterMode applied already, else causes error

## Applying AutoFilter

**Check if AutoFilter already exists**

```
If ws.AutoFilterMode = True Then
    'Do something
End If
```

**Apply AutoFilter to current region**

```
ws.Range("A1").AutoFilter
```

**Remove AutoFilter**

```
ws.AutoFilterMode = False
```

**Hide a filter on a field**

```
ws.Range("A1").AutoFilter Field:=1,_
    Visibledropdown:=False
```

**Get Range of the AutoFilter**

```
MsgBox ws.AutoFilter.Range.Address
```

## Filtering

**Filter with value criteria**

```
ws.Range("A1").AutoFilter _
    Field:=1, Criteria1:="=Apple", _
    Operator:=xlOr, Criteria2:="=Banana"
```

Search operators:

| | |
|---|---|
| *Equals:* | `Criteria1:="=Apple"` |
| *Does Not Equal:* | `Criteria1:="<>Apple"` |
| *Begins with:* | `Criteria1:="=Apple*"` |
| *Ends with:* | `Criteria1:="=*Apple"` |
| *Contains:* | `Criteria1:="=*Apple*"` |
| *Does Not Contain:* | `Criteria1:="<>*Apple*"` |

Search logic:

| | |
|---|---|
| *Or:* | `Operator:=xlOr` |
| *And:* | `Operator:=xlAnd` |

**Show all data**

```
ws.ShowAllData
```

**Filter by color (RGB)**

```
ws.Range("$A$1").AutoFilter _
    Field:=1, Criteria1:=RGB(255, 255, 0), _
    Operator:=xlFilterCellColor
```

**Filter by no color**

```
ws.Range("$A$1").AutoFilter Field:=1,_
    Operator:=xlFilterNoFill
```

## Applying Sort

**Applying column sorting**

```
ws.AutoFilter.Sort.SortFields.Add _
    Order:=xlAscending, _
    SortOn:=xlSortOnValues, _
    Key:=Range("A1:A7")
ws.AutoFilter.Sort.Apply
```

**Clear sorting**

```
ws.AutoFilter.Sort.SortFields.Clear
```

# VBA Cheat Sheet – PivotTables

## Reference a PivotTable

**Reference a PivotTable**

```
Dim ws As Worksheet
Dim pvt As PivotTable

Set ws = ActiveWorkbook.Sheets("Sheet1")
Set pvt = ws.PivotTables("myPivotTable")
```

## Create a PivotTable

**Create a PivotTable**

```
Dim wb As Workbook
Dim ws As Worksheet
Dim pvtCache As PivotCache
Dim pvt As PivotTable

Set wb = ActiveWorkbook
Set ws = wb.Sheets("Sheet1")

Set pvtCache = wb.PivotCaches.Create _
    (SourceType:=xlDatabase, _
    SourceData:="Sheet1!R2C2:R10C4")

Set pvt = pvtCache.CreatePivotTable _
   (TableDestination:="Sheet1!R2C6", _
    tableName:="myPivotTable")
```

## Delete a PivotTable

**Delete a PivotTable**

```
pvt.TableRange2.Clear
```

## Change PivotTable source

**Create a PivotTable**

```
Dim wb As Workbook
Dim ws As Worksheet
Dim pvtCache As PivotCache
Dim pvt As PivotTable

Set wb = ActiveWorkbook
Set ws = wb.Sheets("Sheet1")
Set pvt = ws.PivotTables("myPivotTable")

Set pvtCache = wb.PivotCaches.Create _
    (SourceType:=xlDatabase, _
    SourceData:="Sheet1!R2C2:R12C4")
pvt.ChangePivotCache pvtCache
```

## Turn off AutoFit columns

**Remove auto format**

```
pvt.HasAutoFormat = False
```

## Add and remove fields

**Add fields to the rows, columns and filters**

```
pvt.PivotFields("fieldName").Orientation = xlRowField
pvt.PivotFields("fieldName").Orientation = xlColumnField
pvt.PivotFields("fieldName").Orientation = xlPageField
```

**Change position of field**

```
pvt.PivotFields("fieldName").Position = 1
```

**Add field to values section**

```
pvt.AddDataField pvt.PivotFields("fieldName"), _
    "calcName", xlSum
```

**11 Calculation options:**

```
xlAverage       xlMin           xlSum
xlCount         xlProduct       xlVar
xlCountNums     xlStDev         xlVarP
xlMax           xlStDevP
```

**Remove field from PivotTable**

```
pvt.PivotFields("fieldName").Orientation = xlHidden
```

**Clear all fields**

```
pvt.ClearTable
```

## PivotTable Filters

**Clear existing filters**

```
pvt.PivotFields("fieldName").ClearAllFilters
```

**Add field filter**

```
pvt.PivotFields("fieldName").PivotFilters. _
    Add2 Type:=xlCaptionEquals, _
    Value1:="Apple"
```

## Add calculated field

**Add calculated field**

```
pvt.CalculatedFields.Add _
    "calcFieldName", "=fieldName2+FieldName3"
```

# VBA Cheat Sheet – Charts (page 1)

## About Charts

- Charts can be Chart Objects (face of the worksheet) or Chart Sheets (separate sheets). This page is for Chart Objects.
- Chart Objects are a collection of objects
- Each Chart Object contains a Chart

## Reference charts

**Reference a chart object**

```
Dim ws As Worksheet
Dim cht As Chart
Set ws = ActiveWorkbook.Sheets("Sheet1")
Set cht = ws.ChartObjects("Chart 1").Chart
```

**Reference the active chart**

```
Dim cht As Chart
Set cht = ActiveChart
```

## Loop through chart objects

```
Dim ws As Worksheet
Dim chtObj As ChartObject
Set ws = Worksheets("Sheet1")
For Each chtObj In ws.ChartObjects
    'Code to be applied to each ChartObjects
    'refer to the Chart using chtObj.Chart
Next chtObj
```

## Basic chart settings

```
'Change chart type (examples only, others exist)
cht.ChartType = xlArea
cht.ChartType = xlLine
cht.ChartType = xlPie
cht.ChartType = xlColumnClustered
cht.ChartType = xlColumnStacked
cht.ChartType = xlColumnStacked100
cht.ChartType = xlArea
cht.ChartType = xlAreaStacked
cht.ChartType = xlBarClustered
cht.ChartType = xlBarStacked
cht.ChartType = xlBarStacked100

'Create an empty chart embedded on a worksheet
Set cht = ws.Shapes.AddChart2.Chart

'Select source for a chart
Dim rng As Range
Set rng = ws.Range("A1:B4")
cht.SetSourceData Source:=rng

'Delete a ChartObject
cht.Parent.Delete   'Method 1
chtObj.Delete       'Method 2

'Set the size/position of a ChartObject - method 1
cht.Parent.Height = 200
cht.Parent.Width = 300
cht.Parent.Left = 20
cht.Parent.Top = 20

'Set the size/position of a ChartObject - method 2
chtObj.Height = 200
chtObj.Width = 300
chtObj.Left = 20
chtObj.Top = 20
```

## Basic chart settings (continued)

```
'Change the setting to show only visible cells
cht.PlotVisibleOnly = True

'Change the gap space between bars
cht.ChartGroups(1).GapWidth = 50

'Change the overlap setting of bars
cht.ChartGroups(1).Overlap = 75
```

## Chart axis

There are four chart axis:

- xlValue
- xlCategory
- xlValue, xlSecondary
- xlCategory, xlSecondary

These are used interchangeably in the examples.

```
'Set chart axis min and max
cht.Axes(xlValue).MaximumScale = 25
cht.Axes(xlValue).MinimumScale = 10
cht.Axes(xlValue).MaximumScaleIsAuto = True
cht.Axes(xlValue).MinimumScaleIsAuto = True

'Display axis
cht.HasAxis(xlCategory) = True

'Hide axis
cht.HasAxis(xlValue, xlSecondary) = False

'Display axis title
cht.Axes(xlCategory, xlSecondary).HasTitle = True

'Hide axis title
cht.Axes(xlValue).HasTitle = False

'Change axis title text
cht.Axes(xlCategory).AxisTitle.Text = "My Title"

'Reverse the order of a category axis
cht.Axes(xlCategory).ReversePlotOrder = True
```

## Gridlines

```
'Add gridlines
cht.SetElement (msoElementPrimaryValueGridLinesMajor)
cht.SetElement (msoElementPrimaryCategoryGridLinesMajor)
cht.SetElement (msoElementPrimaryValueGridLinesMinorMajor)
cht.SetElement (msoElementPrimaryCategoryGridLinesMinorMajor)

'Delete gridlines
cht.Axes(xlValue).MajorGridlines.Delete
cht.Axes(xlValue).MinorGridlines.Delete
cht.Axes(xlCategory).MajorGridlines.Delete
cht.Axes(xlCategory).MinorGridlines.Delete

'Change colour of gridlines
cht.Axes(xlValue).MajorGridlines.Format.Line. _
    ForeColor.RGB = RGB(255, 0, 0)

'Change transparency of gridlines
cht.Axes(xlValue).MajorGridlines.Format.Line. _
    Transparency = 0.5
```

# VBA Cheat Sheet – Charts (page 2)

## Chart title

```
'Display chart title
cht.HasTitle = True

'Hide chart title
cht.HasTitle = False

'Change chart title text
cht.ChartTitle.Text = "My Chart Title"

'Position the chart title
cht.ChartTitle.Left = 10
cht.ChartTitle.Top = 10

'Format the chart title
cht.ChartTitle.TextFrame2.TextRange.Font.Name = "Calibri"
cht.ChartTitle.TextFrame2.TextRange.Font.Size = 16
cht.ChartTitle.TextFrame2.TextRange.Font.Bold = msoTrue
cht.ChartTitle.TextFrame2.TextRange.Font.Bold = msoFalse
cht.ChartTitle.TextFrame2.TextRange.Font.Italic = msoTrue
cht.ChartTitle.TextFrame2.TextRange.Font.Italic = msoFalse
```

## Chart legend

```
'Display the legend
cht.HasLegend = True

'Hide the legend
cht.HasLegend = False

'Position the legend
cht.Legend.Position = xlLegendPositionTop
cht.Legend.Position = xlLegendPositionRight
cht.Legend.Position = xlLegendPositionLeft
cht.Legend.Position = xlLegendPositionCorner
cht.Legend.Position = xlLegendPositionBottom

'Allow legend to overlap the chart
'False = allow overlap, True = due not overlap
cht.Legend.IncludeInLayout = False
cht.Legend.IncludeInLayout = True

'Move legend to a specific point
cht.Legend.Left = 20
cht.Legend.Top = 200
cht.Legend.Width = 100
cht.Legend.Height = 25
```

## Plot area

```
'Set the size and position of the PlotArea
cht.PlotArea.Left = 20
cht.PlotArea.Top = 20
cht.PlotArea.Width = 200
cht.PlotArea.Height = 150
```

## Chart series

### Reference chart series

```
'Method 1
Dim srs As Series
Set srs = cht.SeriesCollection(1)

'Method 2
Dim srs As Series
Set srs = cht.SeriesCollection("Series Name")
```

## Chart series (continued)

### Loop through each chart series

```
For Each srs In cht.SeriesCollection
    'Do something to each series
    'Refer to each with srs.
Next srs
```

### Change series data

```
'Change series source data and name
srs.Values = "=Sheet1!$C$2:$C$6"
srs.Name = "=""Change Series Name"""

'Add a new chart series
Set srs = cht.SeriesCollection.NewSeries
srs.Values = "=Sheet1!$C$2:$C$6"
srs.Name = "=""New Series"""

'Set values for the X axis when using XY Scatter
srs.XValues = "=Sheet1!$D$2:$D$6"
```

### Display data labels

```
'Display data labels on all points in the series
srs.HasDataLabels = True

'Hide data labels on all points in the series
srs.HasDataLabels = False

'Position data labels
'Must be a valid option for the chart type.
srs.DataLabels.Position = xlLabelPositionAbove
srs.DataLabels.Position = xlLabelPositionBelow
srs.DataLabels.Position = xlLabelPositionLeft
srs.DataLabels.Position = xlLabelPositionRight
srs.DataLabels.Position = xlLabelPositionCenter
srs.DataLabels.Position = xlLabelPositionInsideEnd
srs.DataLabels.Position = xlLabelPositionInsideBase
srs.DataLabels.Position = xlLabelPositionOutsideEnd
```

### Series formatting

```
'Change fill colour
srs.Format.Fill.ForeColor.RGB = RGB(255, 0, 0)

'Change line colour
srs.Format.Line.ForeColor.RGB = RGB(255, 0, 0)

'Change visibility of line
srs.Format.Line.Visible = msoTrue

'Change line weight
srs.Format.Line.Weight = 10

'Change line style
srs.Format.Line.DashStyle = msoLineDash
srs.Format.Line.DashStyle = msoLineSolid
srs.Format.Line.DashStyle = msoLineSysDot
srs.Format.Line.DashStyle = msoLineSysDash
srs.Format.Line.DashStyle = msoLineDashDot
srs.Format.Line.DashStyle = msoLineLongDash
srs.Format.Line.DashStyle = msoLineLongDashDot
srs.Format.Line.DashStyle = msoLineLongDashDotDot
```

# VBA Cheat Sheet – Charts (page 3)

## Chart series (continued)

```
'Change marker type
srs.MarkerStyle = xlMarkerStyleAutomatic
srs.MarkerStyle = xlMarkerStyleCircle
srs.MarkerStyle = xlMarkerStyleDash
srs.MarkerStyle = xlMarkerStyleDiamond
srs.MarkerStyle = xlMarkerStyleDot
srs.MarkerStyle = xlMarkerStyleNone

'Change marker border color
srs.MarkerForegroundColor = RGB(255, 0, 0)

'Change marker fill color
srs.MarkerBackgroundColor = RGB(255, 0, 0)

'Change marker size
srs.MarkerSize = 8
```

## Error bars

```
'Turn error bars on/off
srs.HasErrorBars = True
srs.HasErrorBars = False

'Change end style of error bar
srs.ErrorBars.EndStyle = xlNoCap
srs.ErrorBars.EndStyle = xlCap

'Change color of error bars
srs.ErrorBars.Format.Line.ForeColor.RGB = RGB(255, 0, 0)

'Change thickness of error bars
srs.ErrorBars.Format.Line.Weight = 5

'Error bar settings
srs.ErrorBar Direction:=xlY, _
    Include:=xlPlusValues, _
    Type:=xlFixedValue, _
    Amount:=100

'Alternatives options for the error bar settings
Direction:=xlX
Include:=xlMinusValues
Include:=xlPlusValues
Include:=xlBoth
Type:=xlFixedValue
Type:=xlPercent
Type:=xlStDev
Type:=xlStError
Type:=xlCustom
```

## Points

### Reference points

```
Dim srs As Series
Dim pnt As Point

Set srs = cht.SeriesCollection(1)
Set pnt = srs.Points(1)
```

### Loop through points

```
Dim srs As Series
Dim pnt As Point
Set srs = cht.SeriesCollection(1)
For Each pnt In srs.Points
    'Do something to each point, using "pnt."
Next pnt
```

### Point data labels

```
'Turn on data label
pnt.HasDataLabel = True

'Set the position of a data label
pnt.DataLabel.Position = xlLabelPositionCenter
```

# VBA Cheat Sheet – Images

## Reference image

### Create image variable

```
Dim ws As Worksheet
Dim img As Shape
Set ws = ActiveWorkbook.Sheets("Sheet1")
Set img = ws.Shapes("myImage")
```

## Insert image

### Insert image and assign to variable

```
Dim img As Shape
Dim imagePath As String
Dim rng As Range
imagePath = "C:\Users\Documents\myImage.png"
Set rng = ActiveCell
Set img = ws.Shapes.AddPicture( _
    Filename:=imagePath, _
    LinkToFile:=msoFalse, _
    SaveWithDocument:=msoTrue, _
    Left:=rng.Left, _
    Top:=rng.Top, _
    Width:=-1, _
    Height:=-1)
```

### Insert shape around active cells

```
Dim ws As Worksheet
Dim img As Shape
Dim rng As Range
Set ws = ActiveSheet
Set rng = Selection
Set img = ws.Shapes.AddShape( _
    msoShapeRectangle, _
    Left:=rng.Left, _
    Top:=rng.Top, _
    Width:=rng.Width, _
    Height:=rng.Height)

img.Fill.Visible = msoFalse
img.Line.ForeColor.RGB = RGB(255, 0, 0)
img.Line.Weight = 2.25
```

## Image properties

### Set common image properties

```
'Set image aspect ratio lock
img.LockAspectRatio = msoTrue

'Change top, left, width and height of image
img.Top = 100
img.Left = 100
img.Width = 100
img.Height = 100

'Change Z-order.  Can bring forward / send backward
img.ZOrder msoBringToFront
img.ZOrder msoSendToBackward

'Change image name
img.Name = "imgName"

'Flip image horizontal or vertical
img.Flip msoFlipHorizontal
img.Flip msoFlipVertical

'Change image visibility
img.Visible = msoFalse
img.Visible = msoTrue
```

## Image properties (continued)

```
'Change image placement options
img.Placement = xlFreeFloating
img.Placement = xlMoveAndSize
img.Placement = xlMove

'Lock image (prevent editing when sheet protected)
img.Locked = True     'Opposite value = False

'Rotate to specific position
img.Rotation = 90
```

### Get image size and position properties

```
MsgBox img.Top
MsgBox img.Left
MsgBox img.Height
MsgBox img.Width
MsgBox img.ZOrderPosition
MsgBox img.TopLeftCell.Address
```

## Place image in relation to cells

### Centre image over a cell

```
'Center image in cell
Set cellLocation = ws.Range("B4")

img.Top = cellLocation.Top + _
    (cellLocation.Height / 2) - _
    (img.Height / 2)
img.Left = cellLocation.Left + _
    (cellLocation.Width / 2) - _
    (img.Width / 2)
```

### Stretch image over cells

```
Set rng = ws.Range("A2:D10")

img.LockAspectRatio = msoFalse

img.Left = rng.Left
img.Top = rng.Top
img.Width = rng.Width
img.Height = rng.Height
```

### Check if selection is an image

```
If TypeName(Selection) = "Picture" Then
    'Object is a picture
End If
```

## Export image from Excel

### Copy picture to chart background, then export empty chart

```
Dim img As Shape
Dim ws As Worksheet
Dim tempChartObj As ChartObject
Dim savePath As String

Set ws = ActiveWorkbook.Sheets("Sheet1")
Set img = ws.Shapes("Picture 1")
Set tempChartObj = ActiveSheet.ChartObjects.Add _
    (0, 0, myPic.Width, myPic.Height)
savePath = "C:\Users\Documents\mySavedPic.jpg"

img.Copy
tempChartObj.Chart.ChartArea.Select
tempChartObj.Chart.Paste
tempChartObj.Chart.Export savePath
tempChartObj.Delete
```

# VBA Cheat Sheet – PDFs

## Declaring a PDF file name

All the Macros on this page use a the following file path

```
Dim pdfFile As String
pdfFile = "C:\Users\marks\Documents\PDFName.pdf"
```

## Create a PDFs from Excel

### Save declared worksheet as PDF

```
Dim ws As Worksheet
Set ws = Worksheets("SheetName")
ws.ExportAsFixedFormat Type:=xlTypePDF, _
    Filename:=pdfFile
```

### Save active worksheet as PDF

```
ActiveWorksheet.ExportAsFixedFormat
Type:=xlTypePDF, _
    Filename:=pdfFile
```

### Save multiple worksheets in single PDF

```
Dim sheetArray As Variant
sheetArray = Array("Sheet1", "Sheet2")
Sheets(sheetArray).Select
ActiveSheet.ExportAsFixedFormat Type:=xlTypePDF, _
    Filename:=pdfFile
```

### Save declared workbook as PDF

```
Dim wb As Workbook
Set wb = Workbooks("WorkbookName.xlsx")
wb.ExportAsFixedFormat Type:=xlTypePDF,
    _ Filename:=pdfFile
```

### Save active workbook as PDF

```
ActiveWorkbook.ExportAsFixedFormat
    _Type:=xlTypePDF,
    _ Filename:=pdfFile
```

### Save declared range as PDF

```
Dim rng As Range
Set rng = Sheets("Sheet1").Range("A1:H20")
rng.ExportAsFixedFormat Type:=xlTypePDF, _
    Filename:=pdfFile
```

### Save selection as PDF

```
Selection.ExportAsFixedFormat Type:=xlTypePDF, _
    Filename:=pdfFile
```

### Save declared chart as PDF

```
Dim ws As Worksheet
Dim cht As Chart
Set ws = Worksheets("SheetName")
Set cht = ws.ChartObjects("Chart 1").Chart
cht.ExportAsFixedFormat Type:=xlTypePDF, _
    Filename:=pdfFile
```

### Save active chart as PDF

```
ActiveChart. ExportAsFixedFormat Type:=xlTypePDF, _
    Filename:=pdfFile
```

## PDF Options

### PDF Print options

```
'Open the document after save
OpenAfterPublish:=False

'Include Excel document properties in
PDFIncludeDocProperties:=True

'Adhere to the existing Print Areas
IgnorePrintAreas:=False

'Set the output quality of the created document
'xlQualityMinimum / xlQualityStandard
Quality:=xlQualityStandard

'The page to start printing from.
From:=1

'The page to print to.
To:=2
```

### Example using all options

```
ActiveSheet.ExportAsFixedFormat _
    Type:=xlTypePDF, _
   pFilename:=pdfFile, _
    OpenAfterPublish:=False, _
    IncludeDocProperties:=True, _
    IgnorePrintAreas:=False, _
    Quality:=xlQualityStandard, _
    From:=1, To:=2
```

# VBA Cheat Sheet - Arrays (page 1)

## About arrays

- **Arrays**: stores data of same variable type (type cannot change)
- **Size / length**: the number of elements in the array
- **Element**: one item with an array
- **Index**: the position of the element within the array
- **Static arrays**: the size is fixed at creation (cannot be resized)
- **Dynamic arrays**: the size can be adjusted as the macro runs
- Array index starts counting at 0, can change to count from 1 by using `Option Base 1` at the start of the code module
- Arrays created from ranges always have 2 dimensions

## Declaring arrays

| | |
|---|---|
| Static (1 dimension) | `Dim arr(0 To 10) As Integer` |
| Static (1 dimension) | `Dim arr(10) As Integer` |
| Static (2 dimension) | `Dim arr(0 To 5, 0 To 5) As Integer` |
| Dynamic | `Dim arr() As Integer` |
| Dynamic | `Dim arr As Variant` |

## Resizing dynamic arrays

- Resizing with **ReDim** will clear the values in the array.
- Resizing with **ReDim Preserve** keeps in the values in the array, but only last dimension can be resized

| | |
|---|---|
| Resize | `ReDim arr(0 To 10)` |
| Resize + keep data | `ReDim Preserve arr(0 To 10)` |

## Assign values to an array

### Assign values individually

```
Dim arr(0 To 2) As String
arr(1) = "Alpha"
arr(2) = "Bravo"
arr(3) = "Charlie"
```

### Assign values individually from list

```
Dim arr As Variant
arr = Array("Alpha", "Bravo", "Charlie")
```

### Assign values individually from list (multi dimension)

```
Dim arr As Variant
arr = Array(Array("Alpha", "Bravo"), Array(1,2))
```

### Assign values by splitting string with a separator

```
Dim arr As Variant
arr = Split("Alpha, Bravo, Charlie", ", ")
```

### Assign values from range

```
Dim arr As Variant
arr = ActiveSheet.Range("A1:C3").Value2
```

## Convert array to string or range

### Convert array to string with separator

```
Dim arr As Variant
arr = Array("Alpha", "Bravo", "Charlie")
MsgBox Join(arr, " ;")
```

### Convert array to range

```
Dim arr As Variant
arr = Array("Alpha", "Bravo", "Charlie")
ActiveSheet.Range("A1:C1") = arr
```

## Looping through array

| | |
|---|---|
| Index of first element | `LBound(arr)` |
| Index of last element | `UBound(arr)` |

### Loop through 1-dimension array

```
Dim i As Long
For i = LBound(arr) To UBound(arr)
  'Thing to do for element
Next i
```

### Loop through 2-dimension array

```
Dim i As Long
Dim j As Long
For i = LBound(arr, 1) To UBound(arr, 1)
  For j = LBound(arr, 2) To UBound(arr, 2)
    'Thing to do for element
  Next j
Next i
```

### Loop through array with For Each

```
Dim element As Variant
For Each element In arr
    'Thing to do for element
Next element
```

## Array functions

### Check if value is in array

```
Function IsValueInArray(arr As Variant, _
    find As Variant) As Boolean
Dim element As Variant
For Each element In arr
    If element = find Then
        IsValueInArray = True
        Exit Function
    End If
Next element
IsValueInArray = False
End Function
```

### Return index position of a matched value in an array

```
Function PositionInArray(arr As Variant, _
    find As Variant) As Variant
Dim i As Long
For i = LBound(arr) To UBound(arr)
    If arr(i) = find Then
        PositionInArray = i
        Exit Function
    End If
Next i
PositionInArray = False
End Function
```

# VBA Cheat Sheet – Arrays (page 2)

## Array functions (continued)

### Reverse the order of an array

```
Function ReverseArray(arr As Variant)
Dim temp As Variant
Dim i As Long
Dim arrSize As Long
Dim arrMid As Long
arrSize = UBound(arr)
arrMid = (UBound(arr) - LBound(arr)) \ 2 + _
  LBound(arr)
For i = LBound(arr) To arrMid
    temp = arr(arrSize)
    arr(arrSize) = arr(i)
    arr(i) = temp
    arrSize = arrSize - 1
Next i
ReverseArray = arr
End Function
```

### Sort an array

```
Function SortingArrayBubbleSort(arr As Variant)
Dim i As Long
Dim j As Long
Dim temp As Variant
For i = LBound(arr) To UBound(arr) - 1
  For j = i + 1 To UBound(arr)
    If arr(i) > arr(j) Then
      temp = arr(j)
      arr(j) = arr(i)
      arr(i) = temp
    End If
  Next j
Next i
SortingArrayBubbleSort = arr
End Function
```

### Filter an array

```
Function FilterArray(arr As Variant, _
  filterValue As Variant)
FilterArray = Filter(arr, filterValue)
End Function
```

# VBA Cheat Sheet – Files & Folders (page 1)

## About files an folders

- Folder paths always need to end with a slash ( \ )

## File functions

**Check if a file exists (True = exists, False = does not exist)**

```
Function DoesFileExist(filePath) As Boolean
  DoesFileExist = Dir(filePath) <> ""
End Function
```

**Check if a folder exists (True = exists, False = does not exist)**

```
Function DoesFolderExist(folderPath As String) _
  As Boolean
DoesFolderExist = Dir(folderPath, _
  vbDirectory) <> ""
End Function
```

**Check if a file is already open (True = open, False = not open, other = error occured**

```
Function IsFileOpen(fileName As String)
Dim fileNum As Integer
Dim errNum As Integer
On Error Resume Next
fileNum = FreeFile()
Open fileName For Input Lock Read As #fileNum
Close fileNum
errNum = Err
On Error GoTo 0
Select Case errNum
  Case 0 'file closed
    IsFileOpen = False
  Case 70 'file already open
    IsFileOpen = True
  Case Else 'Something else went wrong
    IsFileOpen = errNum
End Select
End Function
```

**Get file system attributes**

```
Function CheckFileAttribute(filePath As String, _
    fileAttribute As Long)
CheckFileAttribute = (GetAttr(filePath) _
    And fileAttribute) <> 0
End Function
```

The fileAttribute argument can be:

| *Attribute* | *Enumerator* | *Description* |
|---|---|---|
| *vbNormal* | *0* | *Normal* |
| *vbReadOnly* | *1* | *Read-only* |
| *vbHidden* | *2* | *Hidden* |
| *vbSystem* | *4* | *System* |
| *vbVolume* | *8* | *Volume* |
| *vbDirectory* | *16* | *Directories* |

## Copy, move and delete files

**Rename or move a file or folder**

```
Name "C:\Users\marks\Documents\FileName.xlsx" _
    As "C:\Users\marks\Documents\NewFileName.xlsx"
```

**Copy a file**

```
FileCopy "C:\Users\marks\Documents\File.xlsx", _
  "C:\Users\marks\Documents\Copied File.xlsx"
```

**Delete a file**

```
Kill "C:\Users\marks\Documents\DeleteMe.xlsx"
```

**Delete files using wildcards (? = 1 character, * = 2+ characters)**

```
Kill "C:\Users\marks\Documents\*.xlsx"
```

**Delete all files in a folder**

```
Kill "C:\Users\marks\Documents\*.*"
```

**Delete a folder (need to ensure all files deleted first)**

```
Kill "C:\Users\marks\Documents\Delete Folder\" _
  & "*.*"
RmDir folderPath
```

**Create a new folder**

```
MkDir "C:\Users\marks\Documents\New folder"
```

**Create all folders along a file path**

```
Sub MakeAllFolders(folderPath As String)

Dim individualFolders() As String
Dim tempFolderPath As String
Dim arrayElement As Variant

individualFolders = Split(folderPath, "\")

For Each arrayElement In individualFolders
  tempFolderPath = tempFolderPath & _
    arrayElement & "\"
  If Dir(tempFolderPath, vbDirectory) = "" Then
       MkDir tempFolderPath
  End If
Next arrayElement

End Sub
```

## Loop files in a folder

**Loop through all files a folder**

```
Dim folderName As Variant
folderName = Dir("C:\Users\marks\Documents\")
While folderName <> ""
  'Thing to do for each file
  folderName = Dir
Wend
```

# VBA Cheat Sheet – Files & Folders (page 2)

## Selecting files with the dialog box

### Select a file with the dialog box

```
Function SelectFile(dialogTitle As String)
Dim dialogBox As FileDialog
Set dialogBox = _
  Application.FileDialog(msoFileDialogOpen)
dialogBox.AllowMultiSelect = False
dialogBox.Title = dialogTitle
If dialogBox.Show = -1 Then
   SelectFile = dialogBox.SelectedItems(1)
Else
    SelectFile = ""
End If
End Function
```

### Select a folder with the dialog box

```
Function selectFolder(dialogTitle As String)
Dim dialogBox As FileDialog
Set dialogBox = _
  Application.FileDialog(msoFileDialogFolderPicker)
dialogBox.Title = dialogTitle
If dialogBox.Show = -1 Then
   selectFolder = dialogBox.SelectedItems(1)
Else
    selectFolder = ""
End If
End Function
```

## Zipping and unzipping folders

### Zip a folder

```
Sub CreateZipFile(folderToZipPath As Variant, _
  zippedFileFullName As Variant)
Dim ShellApp As Object
Open zippedFileFullName For Output As #1
Print #1, Chr$(80) & Chr$(75) & Chr$(5) & _
  Chr$(6) & String(18, 0)
Close #1
Set ShellApp = CreateObject("Shell.Application")
ShellApp.Namespace(zippedFileFullName).CopyHere _
  ShellApp.Namespace(folderToZipPath).items
On Error Resume Next
Do Until ShellApp.Namespace(zippedFileFullName). _
  items.Count = _
  ShellApp.Namespace(folderToZipPath).items.Count
  Application.Wait (Now + TimeValue("0:00:01"))On
Error GoTo 0
End Sub
```

### Unzip a folder

```
Sub UnzipAFile(zippedFileFullName As Variant,_
  unzipToPath As Variant)
Dim ShellApp As Object
Set ShellApp = CreateObject("Shell.Application")
ShellApp.Namespace(unzipToPath).CopyHere
ShellApp.Namespace(zippedFileFullName).items
End Sub
```

# Visual Basic Editor – Shortcut Keys (Page 1)

| | On its own | Shift | Ctrl | Alt | Ctrl + Shift |
|---|---|---|---|---|---|
| F1 | Help | | | | |
| F2 | Object browser | Procedure definition | Activate object box | | Previous position |
| F3 | Find next for last searched word | Find previous | | | |
| F4 | Properties window | Find next | Close window | Close VBE | |
| F5 | Runs selected procedure | | | Run error handler | |
| F6 | Toggle split windows | | | | |
| F7 | Displays code window of object | | | | |
| F8 | Step into | Step over line by line | Run to cursor | Step error handler | Step out of code |
| F9 | Toggle breakpoint | Quick watch | | | Clear all breakpoints |
| F10 | Activate menu bar | Show right-click menu | Activate menu bar | | |
| F11 | | | | Toggle between VBE & application | |

# Visual Basic Editor – Shortcut Keys (Page 2)

| | On its own | Shift | Ctrl | Alt | Ctrl + Shift |
|---|---|---|---|---|---|
| Insert | Toggle insert mode | Paste | Copy | | |
| Delete | Delete | | Delete to end of word | | |
| Home | Beginning of line | Select to start of line | Move to top of module | | |
| End | End of line | Select to end of line | Move to end of module | | |
| Page Up | Page up | Select to top of module | Top of current procedure | | |
| Page Down | Page down | Select to end of module | End of current procedure | | |
| Left Arrow | Move cursor left | Extend selection 1 character left | Moves cursor left 1 word | | |
| Right Arrow | Move cursor right | Extend selection 1 character right | Moves cursor right 1 word | | |
| Up Arrow | Move cursor up | Extend selection up | Previous procedure | | |
| Down Arrow | Move cursor down | Extend selection down | Next procedure | | |
| Space | | | Complete word drop down | Activate system menu | |
| Tab | Indent / complete word | Un-indent | Cycle through windows | Cycle applications | |
| Enter | New line | | | | |
| Backspace | Delete previous character | | Delete to start of word | Undo | |

# Visual Basic Editor – Shortcut Keys (Page 3)

| | On its own | Shift | Ctrl | Alt | Ctrl + Shift |
|---|---|---|---|---|---|
| A | | | Select all | Add-ins menu | |
| C | | | Copy | | |
| D | | | | Debug menu | |
| E | | | Export module | Edit menu | |
| F | | | Find | File menu | |
| G | | | Immediate window | | |
| H | | | Replace | Help menu | |
| I | | | Quick info | | Turn on list parameter info |
| J | | | List available properties | Insert menu | Drop down of constants |
| L | | | Call stack | | |
| M | | | Import file | | |
| N | | | New line (no indentation) | | |
| O | | | | Format menu | |
| P | | | Print | | |
| Q | | | | Close VBE | |
| R | | | Project explorer | Run menu | |
| S | | | Save | | |
| T | | | Components dialog box | Tools menu | |
| V | | | Paste | View Menu | |
| W | | | | Window Menu | |
| X | | | Cut | | |
| Y | | | Cut entire line | | |
| Z | | | Undo | | |