VBA mit Excel

VBA steht für *Visual Basic for Applications.*
VBA ist eine vollwertige Programmiersprache, die hinter allen Programmen aus dem Microsoft Office Paket (und einigen anderen mehr, z. B. *CorelDraw* oder *AutoCad*) zur Verfügung steht. Mit VBA entwickelte Anwendungen können nicht zu alleine lauffähigen EXE-Programmen kompiliert werden – sie laufen immer nur in der zugeordneten Programmumgebung: Wenn Sie ein VBA-Programm also unter Excel entwickelt haben, dann benötigt ein User Excel, um es ausführen zu können (der kostenlose Excel-Viewer von Microsoft reicht dazu nicht aus).

Hinweis: *OpenOffice* ist ein für Privatanwender kostenloses Officepaket, das auch eine ähnliche Basic-Programmiersprache beinhaltet.

VBA startet man über *Extras/ Makro/ Visual Basic Editor* oder mit dem Shortcut: Alt+F11

In Direktfenster (*Ansicht/ Direktfenster* oder Strg+G) kann man einzelne Code-Zeilen eingeben, die beim Verlassen mit *Enter* ↵ ausgeführt werden.

Wichtige Befehle:

- **Debug.print** oder **?**
  ‚Druckt' das Folgende in das Direktfenster
  Beispiel:
  ? "Hallo Welt" zeigt *Hallo Welt* in der folgenden Zeile
  ? 5+2*8^2 (ergibt 133)

Hinweise
Rechenoperatoren sind:

^ Caret-Zeichen	Potenzierung:	5^2	entspricht 5^2 (also 5*5)	ergibt 25	} Potenz (^)
	Wurzel	25^(1/2)	entspricht $\sqrt[2]{25}$	ergibt 5	
* Asterix	Multiplikation	5*8	entspricht $5 \cdot 8$	ergibt 40	} Punkt (* und /)
/ Slash	Division	32/8	entspricht $\frac{32}{8}$ oder **32:8**	ergibt 4	
+ Plus	Addition	32+8		ergibt 40	} Strich (+ und -)
- Minus	Subtraktion	32-8		ergibt 24	

Die **Reihenfolge** der Abarbeitung ist *Potenz vor Punkt vor Strich* ^ * / + -
Will man diese Reihenfolge ändern, dann muss man **runde Klammern** setzen: ? ((5+2)*8)^2 (ergibt 3136)
Das **Dezimaltrennzeichen** ist der Punkt, nicht das Komma: 3.5 statt 3,5
& kaufmännisches UND Verknüpfung "Vorname" & " " & "Nachname" ergibt Vorname Nachname
(Achtung: Achten Sie darauf, dass vor und nach dem &-Zeichen eine Leerstelle ist)

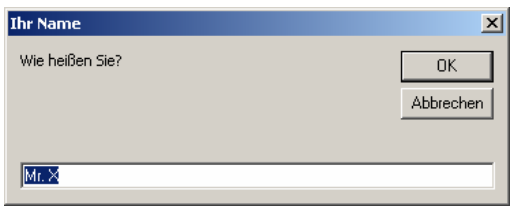
- **Msgbox** *MsgBox* Prompt, [Buttons], [Title] (Parameter in eckigen Klammern sind *nicht* verpflichtend)

Beispiel:
MsgBox "Hallo Welt" *(linkes Bild)*
MsgBox "Hallo Welt",vbCritical+vbYesNo,"Neuer Titel"
(rechtes Bild oben)

MsgBox 5+5,vbInformation+vbOKCancel,"Titel 2"
(rechtes Bild unten)

Hinweise
Die Groß-Kleinschreibung ist gleichgültig.
Subprozeduren und Funktionen werden mit nachgestellten *Parametern* gesteuert, deren Reihenfolge erheblich ist. Die MsgBox-Prozedur erwartet als ersten Parameter den *Prompt*, also den Meldungstext. Ein Komma zeigt als Listentrennzeichen an, dass danach der zweite Parameter folgt, nach dem zweiten Komma folgt der dritte Parameter …
Text (in VBA heißt der *String*) wird immer in Anführungsstriche gesetzt – Text ohne Anführungsstriche wird von VBA als eine Variable interpretiert (wenn diese nicht anders ‚gefüllt' wurde, dann hat sie den Anfangswert *NULL*, ist also leer)

- **Inputbox** *InputBox (*Prompt, [Title], [Default]*)*



Beachten Sie die runden Klammern um die Parameter der Inputbox-Funktion: sie zeigen an, dass die Inputbox-Funktion ein Ergebnis zurückgibt – nämlich die Eingabe des Users.
Default ist ein voreingestellter Standardwert, der dem User vorgeschlagen wird.

Beispiel:

? InputBox("Wie heißen Sie?", "Ihr Name", "Mr. X") erzeugt das nebenstehende Eingabe-Fenster. Das Ergebnis dieser Inputbox ist die Eingabe des Users – in diesem Beispiel würde also *Mr. X* im Direktfenster ausgegeben werden.

MsgBox InputBox("Ihr Name bitte!", "Namenseingabe") erzeugt das zweite Eingabe-Fenster und gibt das, was der User in es eingegeben hat mit einer *Messagebox* wieder aus.

MsgBox InputBox("Ihr Name bitte!", "Namenseingabe"),vbAbortRetryIgnore+vbQuestion,"Username"
würde dasselbe Eingabefenster wie eben erzeugen. Die Eingabe des Users würde aber in einem veränderten Meldungsfenster ausgegeben werden *(rechtes Bild)*.

Prozeduren:

Funktion

- **Funktionen** und Subprozeduren müssen in einem *Modul* stehen.
- *Subprozeduren* sind wie *Makros*, die der Reihe nach Anweisungen ausführen.
- *Funktionen* sind *Subprozeduren* mit einem *Rückgabewert* (=Ergebnis).

Beispiel (im Modul *Modul1*): *(Einfügen/ Modul Einfügen/ Prozedur:* Name: *fktPi Funktion Public)*

```
Public Function fktPi()
fktPi = 3.14159265358979
End Function
```

Diese Funktion kann an beliebiger Stelle in der aktuellen Mappe aufgerufen werden, z. B. in einer Zelle mit: *=fktPi()* oder im Direktfenster mit *? fktPi()* oder *MsgBox fktPi()*

fktPi( ) entspricht dem Ergebnis dieser Funktion, also dem Zahlenwert *3.14159265358979*.
Man kann mit *fktPi( )* ebenso rechnen wie mit der Zahl, also z. B.: ? 5 * *fktPi( )* + 2

Aufbau einer *Function*:
Im Funktionskopf stehen das Schlüsselwort *Public* und das Schlüsselwort *Function* mit einem Leerzeichen getrennt. Danach folgt nach einem weiteren Leerzeichen der Name der Funktion, gefolgt von einem Klammernpaar.
Im Körper der Funktion wird dem Namen der Funktion mit dem Gleichheitszeichen ein Ergebnis zugewiesen. Das ist der Rückgabewert der Funktion.
Den Fuß der Funktion bildet *End Function*.

Subprozedur

Beispiel: *(Einfügen/ Prozedur:* Name: *subPi Sub Public)*

```
Public Sub subPi()
MsgBox "Pi ist " & 3.14159265358979
End Sub
```

Diese Subprozedur kann an beliebiger Stelle in der aktuellen Mappe aufgerufen werden, z. B. in Excel mit *Extras/ Makro/ Makros* Makroname auswählen *Ausführen*-Knopf anklicken oder im Direktfenster mit: *subPi*

Man kann Funktionen und Subprozeduren **Parameter** übergeben:

Beispiel: *(Einfügen/ Prozedur:* Name: *fEcHo Function Public)*

```
Public Function fEcHo(varEcho)
fEcHo =varEcho
End Function
```

Diese Funktion gibt den Wert zurück, der ihr übergeben wurde: =fEcHo(2) in einer beliebigen Excel-Zelle ergibt also 2 *=fEcHo("hallo")* ergibt *hallo.*
Im Direktfenster gibt *? fecho(date()))* das aktuelle Systemdatum als Ergebnis aus.

Subprozeduren und Funktionen können mit den Parametern auch *rechnen*:

Beispiel 2: (*Einfügen/ Prozedur:* Name: *fPotenz Function Public)*

```
Public Function fPotenz(varBasis, varExponent)
fPotenz = varBasis ^ varExponent
End Function
```

Diese Funktion kann an beliebiger Stelle in der aktuellen Mappe aufgerufen werden, z.B. in einer Zelle mit: *= fPotenz (5;2)* **mit Semikolon ;** oder im Direktfenster mit *? fPotenz (5,2)* **mit Komma ,**

Beispiel 3: (*Einfügen/ Prozedur:* Name: *sPotenz Sub Public)*

```
Public Sub sPotenz(Romeo, Julia)
Range("A1") = Romeo ^ Julia
End Sub
```

Diese Subprozedur kann aufgerufen werden, indem Sie im Direktfenster einfach ihren Namen gefolgt von einer Leerstelle eingeben. Danach müssen Sie den Wert des Parameters *Romeo* und nach einem Komma den Wert des Parameters *Julia* eingeben. Wenn Sie anschließend die Enter-Taste betätigen, wird das Ergebnis von Romeo hoch Julia in die Zelle A1 der aktiven Tabelle eingetragen. *sPotenz 5,2* in A1 wird 25 eingetragen

Mit einem Parameter kann einer Subprozedur ein beliebiger Wert übergeben werden. Dieser Parameter kann (fast) beliebig[1] genannt werden. Er muss in dem Klammernpaar hinter dem Namen der Funktion im Funktionskopf stehen. Die Reihenfolge der Parameter ist wichtig: Steht der Parameter *Romeo* an erster Stelle und *Julia* mit einem Komma getrennt an der zweiten, dann wird immer der zuerst übergebene Parameter der Variablen *Romeo* zugewiesen und der zweite der Variablen *Julia*. Der Aufruf *sPotenz 5,2* im Direktfenster bewirkt, dass die 5 der Variablen *Romeo* und dass die 2 der Variablen *Julia* zugewiesen wird. In der Subprozedur *sPotenz* wird gerechnet, indem die Variable Romeo (also der ihr zugewiesene Wert 5) mit Julia (also mit dem ihr zugewiesen Wert 2) potenziert wird. Das Ergebnis dieser Berechnung ($5^2 = 25$) wird anschließend in die Zelle A1 eingetragen.

Beispiel 4: (*Einfügen/ Prozedur:* Name: *kreisumfang Function Public)*

```
Public Function kreisumfang(radius)
kreisumfang = 2 * radius * 3.14159265358979
End Function
```

Diese Funktion wird in einer Excel-Zelle aufgerufen durch: *=kreisumfang(10)* Als Ergebnis erhält man den Umfang eines Kreises mit einem Radius von 10
Geben Sie in B1 ein: *=kreisumfang(A1) Als Ergebnis erhalten Sie den Umfang eines Kreises mit einem Radius von der Größe, die in A1 steht.*

Bezüge:

Sie können mit *Range(*"Bereich"*)* eine bestimmte Zelle adressieren : *Range("A1:C5")=123* bewirkt, dass in den 15 Zellen des Bereichs A1 bis C5 immer 123 steht. Mit *Range("A1:C5")=""* leeren Sie die angegebenen Zellen wieder (*""* sind einfach zwei Gänsefüßchen direkt hintereinander). Mit *Range("Tabelle2!A1")="huhu"* schreiben Sie *huhu* in die Zelle A1 von Tabelle2. *ActiveCell=1* füllt die gerade aktive Zelle in Excel mit einer 1.
Mit der Offseteigenschaft können Sie den Eingabefokus relativ zur geraden aktiven Zelle verschieben: *ActiveCell.Offset(1, 0).Activate* bewirkt, dass die um eine Zeile unter der geraden aktiven Zelle liegende Zelle aktiviert wird. *ActiveCell.Offset(0, -5).Activate* lässt den Eingabefokus um 5 Spalten nach links springen.
Mit *Range("A1:D5").Select* markieren Sie die 20 besagten Zellen.
Zählen Sie diese mit *? selection.count*

[1] Vermeiden Sie alle Sonderzeichen, wie deutsche Umlaute sowie Leerzeichen, und verzichten Sie auf Zahlen am Anfang

Eigenschaften:

Excel besteht aus Objekten wie Zellen, Tabellen, Mappen. Jedes Objekt hat eine bestimmte Anzahl von Eigenschaften.
ActiveCell.Interior.Color = vbRed färbt die aktive Zelle rot ein.
Range("A1").Font.Color = vbGreen färbt die Schrift in Zelle A1 grün.
Erstellen Sie eigene Farben mit der RGB-Funktion: *R* steht für *Rot*, *G* für *Grün*, und *B* für *Blau*:
ActiveCell.Interior.Color =rgb(255,0,0) Da der höchste Wert für jede Farbe 255 ist und der niedrigste 0 würde obige Anweisung den Zellhintergrund der gerade aktiven Zelle rot einfärben.
Range("A1:C3").Font.Color=RGB(255,255,255) stellt die Schriftfarbe in den 9 Zellen des angegebenen Bereichs auf weiß.
Sie können auch die Rahmen der aktuellen Auswahl einstellen:
Selection.Borders(xlEdgeTop).LineStyle = xlContinuous bewirkt, dass ganz oben über der Auswahl eine Rahmenlinie erstellt wird. Ersetzen Sie *xlEdgeTop*, um andernorts Linien zu erzeugen*: xlEdgeLeft, xlEdgeBottom, xlEdgeRight, xlInsideVertical, xlInsideHorizontal* sowie *xlDiagonalDown* und *xlDiagonalUp*
Statt können Sie auch *xlContinuous, xlDash, xlDashDot, xlDashDotDot, xlDot, xlDouble, xlSlantDashDot* oder *xlLineStyleNone* einsetzen.
Mit *Selection.Borders(xlDiagonalUp).LineStyle = xlContinuous* streichen Sie die ausgewählten Zellen durch. *Selection.Borders(xlDiagonalUp).Color=RGB(0,0,255)* färbt die Diagonalen blau.
Mit einem Doppelpunkt können Sie im Direktfenster auch zwei Anweisungen gleichzeitig ausführen lassen:
Selection.Borders(xlDiagonalUp).LineStyle = xlContinuous:Selection.Borders(xlDiagonalUp).Color=RGB(0,0,255)

Formulare:

Erzeugen Sie ein eigenes Formular: *Einfügen/ Userform* in den Eigenschaften stellen Sie *ShowModal* auf *False* (sonst ist die Tabelle nicht mit der Maus erreichbar, solange das Formular geöffnet ist). Die *Caption*-Eigenschaft verändert den Formulartitel: *Dollarumrechner*
Plazieren Sie aus der Toolbox ein *Textfeld* und eine *Befehlsschaltfläche* auf dem Formular: *CommandButton1*: Caption auf *Umrechnen* -> ein Doppelklick auf den Button erstellt den Rumpf der Ereignisprozedur *Private Sub CommandButton1_Click()* , die immer ausgeführt wird, wenn auf den Commandbutton1 geklickt wird. Dort geben Sie als body ein:
*TextBox1 = TextBox1 * 1.1985* Diese Anweisung multipliziert den Wert in der Textbox mit dem Umrechnungskurs des Dollars (http://dollar-kurs.de). Wenn also vorher ein EUR-Wert in dem Textfeld gestanden war, so steht danach ein Dollarwert drin. Damit der Wert aus einer angeklickten Zelle in dem Textfeld *TextBox1* erscheint, erstellen Sie dafür eine Ereignisprozedur: Klicken Sie im linken Projekt-Explorer-Fenster doppelt auf *Tabelle1* und wählen Sie dort aus dem linken *Objekt*-Kombinationsfeld *Worksheet* und aus dem rechten *Prozedur*-Kombinationsfeld *SelectionChange* aus. VBA erstellt automatisch den Rumpf der Ereignisprozedur *Private Sub Worksheet_SelectionChange(ByVal Target As Range)*, die immer dann automatisch ausgeführt wird, wenn eine andere Zelle in *Tabelle1* ausgewählt wird. Damit das Formular *UserForm1* angezeigt wird, geben Sie als erste Anweisung ein: *UserForm1.Show* In der zweiten Zeile weisen Sie der *TextBox1* den Wert der gerade aktiven Zelle zu:
U*serForm1.TextBox1 = ActiveCell*
Immer, wenn Sie jetzt auf eine neue Zelle in Tabelle 1 klicken, öffnet sich unser Formular mit dem Wert der angeklickten Zelle im Textfeld. Mit einem Klick auf den Knopf wird der Wert in Dollar umgerechnet. Sie können sich den Button auch ersparen und die Worksheet-Ereignisprozedur wie folgt ändern:

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
UserForm1.Show
UserForm1.TextBox1 = ActiveCell * 1.1985
End Sub
```