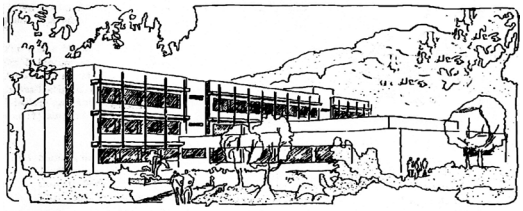
Hochschulrechenzentrum
Justus-Liebig-Universität Gießen



Excel für Microsoft 365

Funktionen (benutzerdefiniert)

HRZ
HOCHSCHULRECHENZENTRUM

JUSTUS-LIEBIG-UNIVERSITÄT GIESSEN

Inhaltsverzeichnis

Einleitung **2**
Erstellung einer benutzerdefinierten Funktion **2**
Berechnung des Ostersonntags **5**
Schaltjahr **6**
Notenspiegel **7**
Quersumme **9**
Letzten Wert einer Spalte bzw. Zeile ermitteln **9**
Zahl in Worten darstellen **10**
Internetadressen **11**
Literatur **12**

Abbildungsverzeichnis

Abb. 1: *Das Register **Entwicklertools** in den Excel-Optionen aktivieren* 3
Abb. 2: *Das Visual Basic-Fenster, Ausschnitt* 3
Abb. 3: *Beispiel für die benutzerdefinierte Funktion EuroDM* 5
Abb. 4: *Beispiel für die benutzerdefinierte Funktion Ostersonntag* 5
Abb. 5: *Die Berechnung des Schaltjahres mit herkömmlichen Excel-Funktionen* 6
Abb. 6: *Die benutzerdefinierten Funktionen **Schaltjahr1** und **Schaltjahr2** in Aktion* 6
Abb. 7: *Die benutzerdefinierte Funktion **NOTE** in Aktion* 8
Abb. 8: *Die abgewandelte Funktion **NOTE*** 9
Abb. 9: *Die benutzerdefinierte Funktion **`LetzterSpaltenwert`** in Aktion* 10
Abb. 10: *Die benutzerdefinierte Funktion **`ZahlInText`** in Aktion* 11

Einleitung

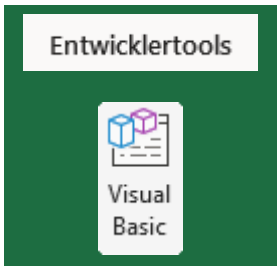
Excel bietet für verschiedenste Problemfälle eine Fülle von integrierten Funktionen (über 400; siehe Skript **Excel Microsoft 365 - Funktionen (Übersicht)**), um Berechnungen durchführen zu können. Mit diesen Funktionen können Sie bereits einen Großteil von Problemstellungen lösen (zumal die Anzahl der Berechnungsmöglichkeiten dadurch steigt, dass verschachtelte Funktionen eingesetzt werden können). Trotzdem gibt es immer wieder Situationen, in denen die vorhandenen Funktionen nicht ausreichen, um eine Aufgabe zu lösen (oder die Formel wird zu lang und damit unübersichtlich). In diesen Fällen ist es sinnvoll bzw. notwendig, eine benutzerdefinierte Funktion zu erstellen. Damit können Sie den Umfang der vorhandenen Excel-Funktionen beliebig erweitern. In **Excel für Microsoft 365** können Sie dazu die neue Funktion *LAMBDA* benutzen (siehe Skript **Excel für Microsoft 365 – Funktion LAMBDA**) oder Sie erstellen die benutzerdefinierten Funktionen in **VBA** (**V**isual **B**asic for **A**pplications). Das erfordert i. Allg. aber entsprechende Kenntnisse in der Programmiersprache *Visual Basic* und zusätzliche Kenntnisse der Excel-Befehle in VBA. An dieser Stelle soll keine Einführung in VBA erfolgen (das würde den Rahmen dieses Skripts bei weitem sprengen), sondern lediglich an ein paar kleinen Beispielen gezeigt werden, wie Sie eine benutzerdefinierte Funktion erstellen können. Das Skript gilt in erster Linie für **Excel für Microsoft 365**, sollte aber problemlos auch für ältere Excel-Versionen gelten.

Erstellung einer benutzerdefinierten Funktion

Im folgenden Beispiel wird eine kleine Umrechnungsfunktion gezeigt, mit der ein Geldbetrag von Euro nach DM umgerechnet werden kann (und umgekehrt)[1]. Hier nun die einzelnen Schritte:

1. Erstellen Sie eine neue Arbeitsmappe (beim Start von Excel oder über das Register **Datei**, Befehl **Neu**, Eintrag **Leere Arbeitsmappe**).

2. Wählen Sie im Register **Entwicklertools** in der Gruppe ***Code*** den Befehl **Visual Basic** und Sie erhalten ein neues Fenster (das Visual Basic-Fenster; siehe Abbildung 2, Seite 3). Wird zum aller ersten Mal das Register **Entwicklertools** verwendet, ist es vermutlich gar nicht sichtbar. Sie müssen deshalb das Register **Datei** und den Befehl **Optionen** wählen. Im Dialogfeld **Excel-Optionen** aktivieren Sie in der Kategorie **Menüband anpassen** das Kontrollkästchen **Entwicklertools** (siehe Abbildung 1, Seite 3) und bestätigen das Dialogfeld.



3. Wählen Sie im Menü **Einfügen** den Befehl **Modul**.

4. Geben Sie nun die Visual Basic-Funktion ein. **Bitte achten Sie darauf, dass Sie die hier angegebenen Zeilennummern nicht mit eingeben. Sie dienen an dieser Stelle nur zur Orientierung für die nachfolgende Beschreibung der einzelnen Zeilen. Außerdem sollten Sie beachten, dass die letzte Zeile (Zeile 7) nicht eingetippt werden muss, da diese Zeile bereits von Visual Basic vorgegeben wird (nach dem Sie die erste Zeile eingegeben und mit der Eingabetaste (↵) bestätigt haben). Wie Sie an dem Beispiel sehen können, sind einige**

[1] Es handelt sich hierbei wirklich nur um ein kleines Beispiel. Es gibt in Excel für Microsoft 365 bereits einen Befehl für die Umrechnung von Euro in andere Währungseinheiten und umgekehrt. Dieses Beispiel eignet sich aber sehr gut, um einen Einstieg in VBA zu bekommen.

Zeilen von links nach rechts eingezogen. Das erhöht die Lesbarkeit des Programms, hat aber ansonsten keine Bedeutung für die Funktion selbst.

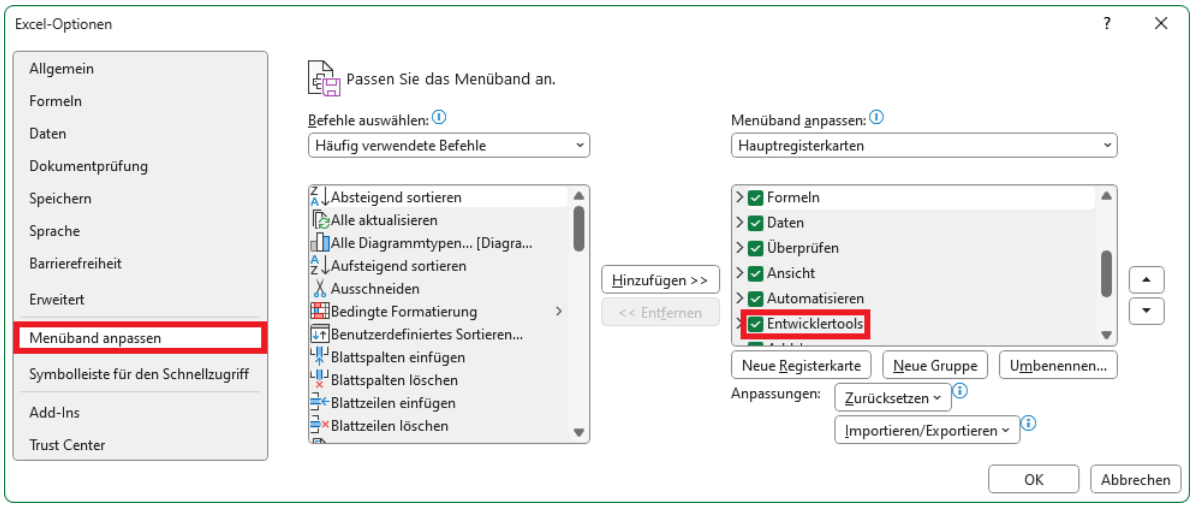


Abb. 1: *Das Register **Entwicklertools** in den Excel-Optionen aktivieren*

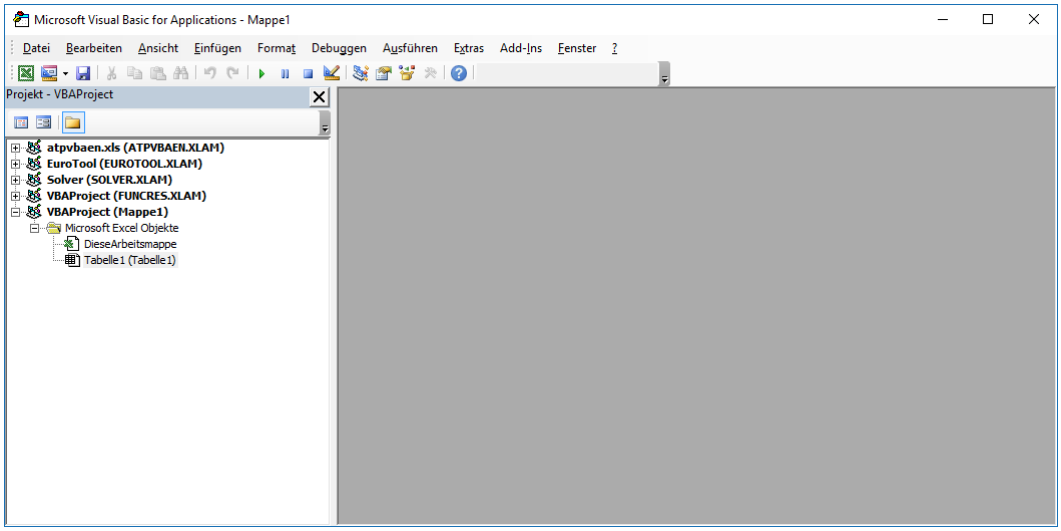


Abb. 2: *Das Visual Basic-Fenster, Ausschnitt*

```
1  Function EuroDM(Währung As String, Betrag As Single) As Single
2       If Währung = "DM" Then
3            EuroDM = Betrag / 1.95583
4       Else
5            EuroDM = Betrag * 1.95583
6       End If
7  End Function
```

Diese Zeilennummern nicht mit eingeben.

Beschreibung der einzelnen Zeilen:

1	Mit dem Schlüsselwort **Function** teilen Sie Visual Basic mit, dass Sie eine Funktion erstellen wollen. Der Name ***EuroDM*** ist selbst gewählt und kann auch anders lauten (Groß-/Kleinschreibweise ist ohne Bedeutung). Sie dürfen aber keinen Namen einer integrierten Excel-Funktion auswählen (z.B. darf Ihre benutzerdefinierte Funktion nicht SUMME heißen). Die Funktion benötigt in diesem Beispiel zwei Funktionsargumente: Für jedes Argument dürfen Sie sich wieder frei für einen Namen entscheiden. Das erste Argument wird hier ***Währung*** genannt. Mit dem Zusatz **As String** wird festgelegt, dass es ein Argument vom Typ *Text* ist. Das zweite Argument wird hier ***Betrag*** genannt. Mit dem Zusatz **As Single** wird festgelegt, dass es sich um eine *Dezimalzahl* handelt. Die Funktion liefert auch eine *Dezimalzahl* als Ergebnis zurück, daher wird hinter der schließenden, runden Klammer dies mit **As Single** auch gekennzeichnet.
2	Mit dem Schlüsselwort **If** wird eine Wenn-Abfrage gestartet. Hier wird einfach abgefragt, ob der Inhalt des Parameters ***Währung*** dem Textkürzel **DM** entspricht. Nach der Bedingung folgt noch das Schlüsselwort **Then**.
3	Ist die Bedingung aus Schritt 2 wahr, wird der eigentliche umzurechnende Wert (ist in dem Argument ***Betrag*** enthalten) durch den konstanten Wert **1.95583** dividiert. Das Ergebnis wird direkt an den Funktionsnamen EuroDM übertragen. **Achten Sie bitte darauf, dass bei dem konstanten Wert das Dezimaltrennzeichen ein Punkt ist und kein Komma.**
4	Falls die Bedingung aus Schritt 2 falsch sein sollte, soll von Euro nach DM umgerechnet werden. Dies wird mit dem Schlüsselwort **Else** eingeleitet.
5	Hier erfolgt die eigentliche Berechnung. Im Prinzip identisch mit Schritt 3, aber hier wird multipliziert und nicht dividiert.
6	Mit den beiden Schlüsselwörtern **End If** wird die Überprüfung aus Schritt 2 abgeschlossen.
7	Die beiden Schlüsselwörter **End Function** zeigen an, dass die Funktion beendet ist.

5. Schließen Sie den Visual Basic-Editor über das Menü **Datei** und den Befehl **Schließen und zurück zu Microsoft Excel** (alternativ: Alt + Q).

6. Speichern Sie die Arbeitsmappe (Register **Datei**, Befehl **Speichern unter**). Tragen Sie in das Textfeld **Dateiname** einen gewünschten Namen ein (z.B. könnte die Datei *Euro_DM* lauten). **Achten Sie aber darauf, dass Sie als Dateityp Excel-Add-In (*.xlam) wählen.**

7. Schließen Sie Excel.

Wenn Sie beim nächsten Excel-Start die benutzerdefinierte Funktion verwenden wollen, müssen Sie Excel das neue Add-In erst kenntlich machen (**Achtung: diese Aktion muss nur einmal durchgeführt werden und nicht bei jedem neuen Excel-Start**). Wählen Sie das Register **Datei** und den Befehl **Optionen**. Im Dialogfeld **Excel-Optionen** klicken Sie in der Kategorie **Add-Ins** auf die Schaltfläche Los... . Im Dialogfeld **Add-Ins** sollte nun der Dateiname des Add-Ins zu sehen sein, den Sie in Schritt 6 (siehe oben) beim Speichern eingegeben haben. Sollte der Dateiname nicht in der Liste enthalten sein, können Sie den Dateinamen auch über die Schaltfläche Durchsuchen... wählen (Sie sollten natürlich noch wissen, auf welchem Laufwerk und in welchem Ordner Sie die Datei gespeichert haben). Aktivieren Sie im Dialogfeld **Add-Ins** dann das Kontrollkästchen vor dem Namen und bestätigen die Änderung.

Jetzt können Sie die Funktion wie jede andere Excel-Funktion einsetzen. In Abbildung 3 wird das beispielhaft dargestellt.

	A	B	C	D	E	F
1	135,76 DM	69,41 €		Formel in B1: =EuroDM("DM";A1)		

Abb. 3: *Beispiel für die benutzerdefinierte Funktion EuroDM*

Anmerkung: Für die Tabellenzelle **B1** wurde das Zahlenformat **#.##0,00 €** gewählt. Die Funktion ***EuroDM*** liefert zunächst nur eine Dezimalzahl ohne jegliche Formatierung.

Bedenken Sie bitte, dass es sich hier um ein sehr einfaches Beispiel handelt. In der Realität sind benutzerdefinierte Funktionen wesentlich umfangreicher. Dabei können solche Funktionen durchaus hunderte oder tausende von Programmzeilen besitzen und sehr komplex aufgebaut sein.

Berechnung des Ostersonntags

Mit einer sehr interessanten Funktion können Sie das Datum des Ostersonntags bestimmen. Die Funktion benötigt lediglich die Jahreszahl. Die dieser Funktion zugrundeliegende Methode stammt von *Carl Friedrich Gauß*. Bei der Erstellung des Add-Ins benutzen Sie im Grunde dieselben Schritte wie bei dem vorherigen Beispiel. An dieser Stelle wird nur die eigentliche Funktion in der VBA-Schreibweise wiedergegeben (ohne Zeilennummern; die werden ja eh nicht mit eingegeben).

```
Function Ostersonntag(Jahr As Integer) As Date
    Dim Datum As Integer
    Datum = (((255 - 11 * (Jahr Mod 19)) -21) Mod 30) + 21
    Ostersonntag = DateSerial(Jahr, 3, 1) + Datum + (Datum > 48) + 6 - _
                   ((Jahr + Jahr\4 + Datum + (Datum > 48) + 1) Mod 7)
End Function
```

Wenn Sie das Makro als Add-In gespeichert (beispielsweise unter dem Namen *Ostersonntag*) und dann über das Dialogfeld **Add-Ins** installiert haben (siehe Vorgehensweise beim vorherigen Beispiel *EuroDM*), können Sie die Funktion entsprechend einsetzen (siehe Abbildung 4).

	A	B	C	D
1	Jahr	Ostersonntag		
2	2022	17.04.2022		Formel in B2: =Ostersonntag(A2)
3	2023	09.04.2023		Formel in B3: =Ostersonntag(A3)
4	2024	31.03.2024		Formel in B4: =Ostersonntag(A4)
5	2025	20.04.2025		Formel in B5: =Ostersonntag(A5)
6	2026	05.04.2026		Formel in B6: =Ostersonntag(A6)
7	2027	28.03.2027		Formel in B7: =Ostersonntag(A7)
8	2028	16.04.2028		Formel in B8: =Ostersonntag(A8)
9	2029	01.04.2029		Formel in B9: =Ostersonntag(A9)

Abb. 4: *Beispiel für die benutzerdefinierte Funktion Ostersonntag*

Anmerkung: In Abbildung 4 wurde für die Tabellenzelle **B2** (und natürlich auch für die Tabellenzellen **B3** bis **B9**) das Zahlenformat **T. MMMM JJJJ** eingestellt. Die Funktion liefert zunächst nur eine serielle Zahl (z.B. in Tabellenzelle **B3**: **42099**). Dabei handelt es sich nur um die Anzahl der Tage seit dem 1. Januar 1900.

Schaltjahr

In einem weiteren Beispiel geht es darum, für ein vorgegebenes Datum bzw. Jahr herauszufinden, ob es sich bei der Jahresangabe um ein Schaltjahr handelt oder nicht. Prinzipiell kann diese Berechnung mit den vorhandenen Excel-Funktionen durchgeführt werden (siehe Abbildung 5). Allerdings ist diese Lösung auf Dauer nicht sinnvoll, da sie bei jeder Verwendung immer wieder neu eingegeben werden muss. Es ist also besser, eine benutzerdefinierte Funktion mit VBA zu erstellen. Nachfolgend gibt es zwei Lösungen. Bei der ersten Variante wird davon ausgegangen, dass der Funktion (sie wird hier **Schaltjahr1** genannt) nur die Jahreszahl übergeben wird, d.h. in einer Tabellenzelle steht lediglich die Jahreszahl, auf die die Funktion zugreift. Bei der zweiten Variante (**Schaltjahr2**) steht in einer Tabellenzelle ein komplettes Datum. Erst in der benutzerdefinierten Funktion wird aus dem Datum die Jahreszahl extrahiert. Abbildung 6 zeigt beide Varianten in Aktion.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Jahr												
2	2015	Kein Schaltjahr		Formel in B2: =WENN(ODER(REST(A2;400)=0;UND(REST(A2;4)=0;REST(A2;100)<>0));"Schaltjahr";"Kein Schaltjahr")									
3	2016	Schaltjahr		Formel in B3: =WENN(ODER(REST(A3;400)=0;UND(REST(A3;4)=0;REST(A3;100)<>0));"Schaltjahr";"Kein Schaltjahr")									
4	2017	Kein Schaltjahr		Formel in B4: =WENN(ODER(REST(A4;400)=0;UND(REST(A4;4)=0;REST(A4;100)<>0));"Schaltjahr";"Kein Schaltjahr")									
5	2018	Kein Schaltjahr		Formel in B5: =WENN(ODER(REST(A5;400)=0;UND(REST(A5;4)=0;REST(A5;100)<>0));"Schaltjahr";"Kein Schaltjahr")									
6	2019	Kein Schaltjahr		...									
7	2020	Schaltjahr											
8	2021	Kein Schaltjahr											
9	2022	Kein Schaltjahr											
10	2023	Kein Schaltjahr											
11	2024	Schaltjahr											
12	2025	Kein Schaltjahr											
13	2026	Kein Schaltjahr											
14	2027	Kein Schaltjahr											
15	2028	Schaltjahr											

Abb. 5: *Die Berechnung des Schaltjahres mit herkömmlichen Excel-Funktionen*

	A	B	C	D
1	Jahr			
2	2023	Kein Schaltjahr		Formel in B2: =Schaltjahr1(A2)
3	2024	Schaltjahr		Formel in B3: =Schaltjahr1(A3)
4				
5	Datum			
6	23.04.2023	Kein Schaltjahr		Formel in B6: =Schaltjahr2(A6)
7	12.07.2024	Schaltjahr		Formel in B7: =Schaltjahr2(A7)

Abb. 6: *Die benutzerdefinierten Funktionen **Schaltjahr1** und **Schaltjahr2** in Aktion*

Hier nun die beiden Varianten als VBA-Funktion:

Variante 1:

```
Function Schaltjahr1(Jahr As Integer) As String
   If (Jahr Mod 4) = 0 And (Jahr Mod 100) <> 0 Or (Jahr Mod 400) = 0 Then
       Schaltjahr1 = "Schaltjahr"
   Else
       Schaltjahr1 = "Kein Schaltjahr"
   End If
End Function
```

Variante 2:

```
Function Schaltjahr2(Datum As Date) As String
   Dim Jahr As Integer
   Jahr = Year(Datum)
   If (Jahr Mod 4) = 0 And (Jahr Mod 100) <> 0 Or (Jahr Mod 400) = 0 Then
       Schaltjahr2 = "Schaltjahr"
   Else
       Schaltjahr2 = "Kein Schaltjahr"
   End If
End Function
```

Notenspiegel

Vorgegeben ist eine Liste mit erreichten Punkten einer Klausur. In diesem Beispiel wird davon ausgegangen, dass die maximal erreichbare Punktzahl **100** beträgt. Nur die erreichten Punkte werden in einer Spalte auf einem Excel-Arbeitsblatt eingetragen. In der Spalte rechts davon sollen die Noten **1** bis **6** in Abhängigkeit der erreichten Punktzahl eingetragen werden. Die folgende Liste zeigt für jede Note den zugehörigen zu erreichenden Punktebereich (Sie können diese Angaben natürlich auch beliebig verändern; entsprechend muss die zugehörige VBA-Funktion angepasst werden):

Punkte	Note
90 oder mehr	1
75 bis 89	2
60 bis 74	3
45 bis 59	4
30 bis 44	5
0 bis 29	6

Wenn Sie nun versuchen, diese Aufgabe mit Hilfe der vorhandenen Excel-Funktionen zu lösen, benötigen Sie die **WENN**-Funktion. Allerdings müssen Sie diese Funktion mehrfach ineinander verschachteln. Das sieht dann in der Praxis so aus (es wird davon ausgegangen, dass die Punktzahl in der Tabellenzelle **A1** steht und die Formel in der Tabellenzelle **B1**):

```
=WENN(A1>=90;1;WENN(A1>=75;2;WENN(A1>=60;3;WENN(A1>=45;4;WENN(A1>=30;5;6)))))
```

Wie Sie an der Formel sehen können, ist die häufigere Benutzung dieser Formel nicht wirklich sinnvoll, da sie in einer neuen Arbeitsmappe auch immer wieder neu eingegeben werden muss. Auf Dauer eher lästig. Hierfür bietet sich eine benutzerdefinierte Funktion an, die Sie (wie im Kapitel **Erstellung einer benutzerdefinierten Funktion** beschrieben) als Add-In speichern und dann in allen Arbeitsmappen einsetzen können. Der VBA-Programmcode sieht folgendermaßen aus:

```
Function Note(Punkte As Integer) As Integer
    Select Case Punkte
        Case Is >= 90
            Note = 1
        Case 75 To 89
            Note = 2
        Case 60 To 74
            Note = 3
        Case 45 To 59
            Note = 4
        Case 30 To 44
            Note = 5
        Case Else
            Note = 6
    End Select
End Function
```

In Abbildung 7 können Sie sehen, wie die benutzerdefinierte Funktion *Note* eingesetzt wird.

	A	B	C	D	E	F
1	**Punkte**	**Note**				
2	**36**	**5**		**Formel in B2: =Note(A2)**		
3	**78**	**2**		**Formel in B3: =Note(A3)**		
4	**29**	**6**		**Formel in B4: =Note(A4)**		
5	**93**	**1**		**Formel in B5: =Note(A5)**		
6	**17**	**6**		**Formel in B6: =Note(A6)**		
7	**57**	**4**		**Formel in B7: =Note(A7)**		

Abb. 7: *Die benutzerdefinierte Funktion **NOTE** in Aktion*

Das Beispiel soll nun ein wenig abgewandelt werden. Anstelle der Noten in Form von Zahlen sollen nun die Bezeichnungen *sehr gut* (Note 1), *gut* (Note 2) usw. bis *ungenügend* (Note 6) als Ergebnis der Funktion angezeigt werden. Der VBA-Programmcode sieht folgendermaßen aus (siehe auch Abbildung 8, Seite 9; Die Ausrichtung *Zentriert* wurde manuell durchgeführt):

```
Function Note(Punkte As Integer) As String
    Select Case Punkte
        Case Is >= 90
            Note = "sehr gut"
        Case 75 To 89
            Note = "gut"
        Case 60 To 74
            Note = "befriedigend"
        Case 45 To 59
            Note = "ausreichend"
        Case 30 To 44
            Note = "mangelhaft"
        Case Else
            Note = "ungenügend"
    End Select
End Function
```

	A	B	C	D	E	F
1	**Punkte**	**Note**				
2	**36**	**mangelhaft**		**Formel in B2: =Note(A2)**		
3	**78**	**gut**		**Formel in B3: =Note(A3)**		
4	**29**	**ungenügend**		**Formel in B4: =Note(A4)**		
5	**93**	**sehr gut**		**Formel in B5: =Note(A5)**		
6	**17**	**ungenügend**		**Formel in B6: =Note(A6)**		
7	**57**	**ausreichend**		**Formel in B7: =Note(A7)**		

Abb. 8: *Die abgewandelte Funktion* ***NOTE***

Quersumme

Im nächsten Beispiel soll die Quersumme einer vorgegebenen Ganzzahl ermittelt werden (z.B. wird aus 3245 = 3+2+4+5 = 14). Hier das VBA-Programm[2]:

```
Function Quersumme(c As Range) As Long
    Dim i As Integer
    If IsNumeric(c) Then
        For i = 1 To Len(c)
            Quersumme = Quersumme + Mid(c, i, 1)
        Next i
    End If
End Function
```

Letzten Wert einer Spalte bzw. Zeile ermitteln

Gelegentlich brauchen Sie den letzten Wert einer Spalte oder Zeile. Beispielsweise ist der letzte Wert die Summe der Spalte oder der Zeile. Zwar könnten Sie auch einfach einen Bezug auf die entsprechende Tabellenzelle machen, aber bei sehr großen Tabellen mit tausenden von Daten ist es nicht ganz so einfach den letzten Spalten- bzw. Zeilenwert zu ermitteln[3]. Mit den beiden folgenden VBA-Funktionen geht das ganz schnell (auch dann, wenn weitere Spalten bzw. Zeilen hinzugefügt oder gelöscht werden)[4]:

```
Function LetzterSpaltenwert(Zelle)
    Application.Volatile
    LetzterSpaltenwert = Cells(Rows.Count, Zelle.Column).End(xlUp)
End Function
```

[2] Microsoft Excel 2010 Programmierung; M. Can-Weber/T. Wendel; Microsoft Press; ISBN-13: 978-3866454606; Seite 403

[3] Excel VBA (Excel 97 bis 2007); B. Held; Markt&Technik; ISBN-13: 978-3827241177; Seite 458 u. 459

[4] Die Zeile **`Application.Volatile`** bewirkt, dass bei einer Neuberechnung einer beliebigen Tabellenzelle des Arbeitsblatts auch das Ergebnis der benutzerdefinierten Funktion aktualisiert wird. Ohne diese Zeile werden nur integrierte Excel-Funktionen automatisch neu berechnet.

```
Function LetzterZeilenwert(Zelle)
    Application.Volatile
    LetzterSpaltenwert = Cells(Zelle, Columns.Count).End(xlToLeft)
End Function
```

	A	B	C	D	E	F	G
1	5101	-3044		Letzter Wert in Spalte A:	7938		Formel in E1: =LetzterSpaltenwert(A1)
2	7536	-1235		Letzter Wert in Spalte B:	-3024		Formel in E2: =LetzterSpaltenwert(B1)
3	7233	-4094					
4	6890	3423					
5	6000	6575					
6	6513	-3660					
7	5372	2226					
8	7938	-3024					

Abb. 9: *Die benutzerdefinierte Funktion* ***LetzterSpaltenwert*** *in Aktion*

Zahl in Worten darstellen

Beim letzten Beispiel geht es darum, eine Zahl (z.B. **387,52**) in Worten darzustellen (für das vorgegebene Beispiel: **Drei Acht Sieben Komma Fünf Zwei**). Mit dieser benutzerdefinierten Funktion können Zahlen im Bereich von -9.999.999.999,99 bis +9.999.999.999,99 in Worten dargestellt werden[5]. Dabei wird davon ausgegangen, dass die Zahl 2 Nachkommastellen besitzt. Hat sie nur eine Stelle nach dem Komma, wird eine zusätzliche Null angezeigt. Bei mehr als 2 Stellen nach dem Komma, werden nur die ersten beiden Dezimalstellen berücksichtigt). Die Zahlenwerte in Spalte **A** in Abbildung 10, Seite 11, sind mit dem Zahlenformat **?.???,????** formatiert worden.

```
Function ZahlInText(x As Variant) As String
    Dim i, Letzteszeichen As Long
    Dim Resultat, Zeichen As String
    Dim Ziffer(9) As String

    Ziffer(0) = "Null": Ziffer(1) = "Eins": Ziffer(2) = "Zwei"
    Ziffer(3) = "Drei": Ziffer(4) = "Vier": Ziffer(5) = "Fünf"
    Ziffer(6) = "Sechs": Ziffer(7) = "Sieben": Ziffer(8) = "Acht"
    Ziffer(9) = "Neun"

    Application.Volatile

    If IsEmpty(x) Then
        ZahlnInText = ""
        Exit Function
    End If
    If x >= 10000000000# Or x <= -10000000000# Then
        ZahlInText = "Zahl ist zu groß oder zu klein"
        Exit Function
    End If
```

[5] Excel 2007 programmieren; M. Kofler/R. Nebelo; Addison-Wesley; ISBN-13: 978-3827325679; Seite 312f

```
    If x < 0 Then
        Resultat = "Minus "
        x = -x
    End If

    x = Format$(x, "0.00")
    x = Space(13 - Len(x)) + x

    If Right(x, 3) = ",00" Then
        Letzteszeichen = 10
    Else
        Letzteszeichen = 13
    End If

    For i = 1 To Letzteszeichen
        Zeichen = Mid(x, i, 1)
        If Zeichen >= "0" And Zeichen <= "9" Then
            Resultat = Resultat + Ziffer(Val(Zeichen)) + " "
        ElseIf Zeichen = "," Then
            Resultat = Resultat + "Komma "
        End If
    Next i

    ZahlInText = Trim(Resultat)
End Function
```

	A	B	C	D	E	F
1	73.629,18	Sieben Drei Sechs Zwei Neun Komma Eins Acht		Formel in B1: =ZahlInText(A1)		
2	4.536,67	Vier Fünf Drei Sechs Komma Sechs Sieben		Formel in B2: =ZahlInText(A2)		
3	195,7	Eins Neun Fünf Komma Sieben Null		Formel in B3: =ZahlInText(A3)		
4	18.465,3	Eins Acht Vier Sechs Fünf Komma Drei Null		Formel in B4: =ZahlInText(A4)		
5	132,	Eins Drei Zwei		Formel in B5: =ZahlInText(A5)		
6	12,5646	Eins Zwei Komma Fünf Sechs		Formel in B6: =ZahlInText(A6)		

Abb. 10: *Die benutzerdefinierte Funktion* ***ZahlInText*** *in Aktion*

Internetadressen

Eine Fülle an benutzerdefinierten Funktionen für Excel finden Sie auch im Internet. Am besten geben Sie in einer Suchmaschine (z.B. Google, Yahoo, Bing, Web.de) mal den Suchbegriff *VBA Excel* ein. Sie erhalten mehrere Millionen Links zu diesem Suchbegriff (darunter auch Links nur zum Thema VBA bzw. nur zum Thema Excel). Hier zwei interessante Links, die u.a. auch Beispiele an VBA-Programmen beinhalten:

- **www.office-loesung.de**
- **www.online-excel.de**

Literatur

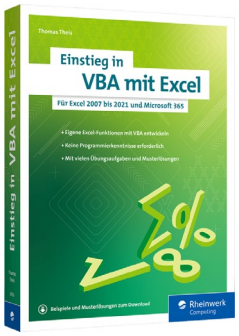
Zum Thema Visual Basic for Applications (VBA; hier speziell für Excel) gibt es eine ganze Reihe an Büchern. Dabei spielt die Excel-Version (ab Excel 2007) nur eine untergeordnete Rolle. Normalerweise sollten VBA-Programme, die für die Excel-Version 2010 oder 2013 entwickelt und programmiert worden sind, ohne jegliche Probleme auch in der Version Excel für Microsoft 365 einsetzbar sein (Ausnahmen kann es aber immer mal geben). Umgekehrt (von Excel 2019 nach Excel 2016, 2013 oder 2010) gibt es keine hundertprozentige Funktionsgarantie. Nachfolgend nur eine kleine Auflistung an Büchern.

VBA mit Excel: das umfassende Handbuch

Bernd Held

Rheinwerk Computing (5. Auflage, 2022); 1.032 Seiten

ISBN-13: 978-3-8362-8690-9

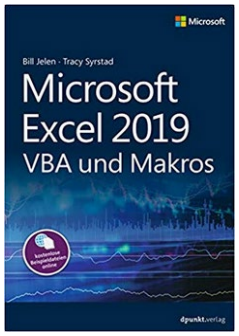


Einstieg in VBA mit Excel

Thomas Theis

Rheinwerk Computing (6. Auflage; 2022); 500 Seiten

ISDBN-13: 978-3-8362-9059-3



Microsoft Excel 2019 VBA und Makros

Bill Jelen, Tracy Syrstad

dpunkt.verlag GmbH (1. Auflage; 07/2019); 706 Seiten

ISBN-13: 978-3864906930

VBA mit Excel - Der leichte Einstieg: Vom ersten Makro zur eigenen Eingabemaske - Für Excel 2010 bis 2019

Inge Baumeister, Dieter Klein

BILDNER Verlag (1. Auflage; 06/2018)

ISBN-13: 978-3832803032