



## 3 VBA ALLGEMEIN

In diesem Abschnitt sind allgemeine Funktionen von VBA beschreiben. Wenn Sie bereits grundlegende Kenntnisse von VisualBasic oder VBA aus anderen Office Anwendungen haben könnten Sie diesen Abschnitt überspringen. Die speziellen Excel Objekte und Befehlen sind im nachfolgenden Abschnitt beschrieben.

### 3.1 With Anweisung



Wenn Sie einmal komplexere Tabellen programmieren, kann es leicht dazu kommen, dass sich zum Teil sehr lange Ketten von Objektname und Eigenschaften ergeben. Mit dem kleinen Beispiel werden Font-Eigenschaften verändert.

```
Sub Font_Test()
    ActiveCell.Font.Italic = True
    ActiveCell.Font.Bold = True
End Sub
```

Solche langen Codes können die Programmierung leicht unübersichtlich machen. Mit der Anweisung *With* und *End With* wird der Code etwas übersichtlicher, was das programmieren sehr erleichtert.

```
Sub Font_Test_With()
    With ActiveCell.Font
        .Italic = True
        .Bold = True
    End With
End Sub
```

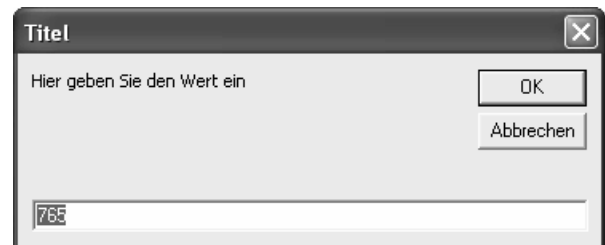
	A	B	
1			
2		<b>Visual Basic</b>	
3			
4			

Das Ergebnis der beiden Beispielcodes ist dasselbe.

### 3.2 InputBox



Dieses Fenster zeigt eine Eingabeaufforderung in einem Dialogfeld an und wartet auf die Eingabe eines Textes und auf den Klick auf eine Schaltfläche. Bei **OK** wird der eingetragene Text zur Weiterverarbeitung zurückgegeben.



Im Beispielcode wird der Wert der aktiven Zelle gelesen und in das Eingabefeld der Inputbox geschrieben.

```
Sub Eingabe_Box()
    Wert = ActiveCell.Value
    Wert = InputBox("Hier geben Sie den Wert ein", _
        "Titel", Wert, 500, 700)
    ActiveCell.Value = Wert
End Sub
```

#### Argumente

`InputBox("Text", "Titel", Wert, 500, 700)`

Die Meldung im Fenster

Beschriftung der Titelzeile

Wert der im Eingabefenster angezeigt wird

x-Position und y-Position der Inputbox vom linken und oberen Rand des Bildschirms in Twips.

**Übungen:**  
 Liter in Kg .....62  
 Adresse in Rechnung .....79



### 3.3 MsgBox

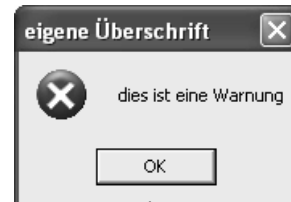
Mit der MsgBox erzeugt das Programm ein Meldungsfenster auf dem Bildschirm. Sie zeigt eine Meldung in einem Dialogfeld an und wartet darauf, dass der Benutzer auf eine Schaltfläche klickt. Ein Rückgabewert wird in dieser Variante nicht erzeugt.



```
Sub Meldung_einfach()
    MsgBox "dies ist eine Meldung"
End Sub
```

#### Darstellung und Titel

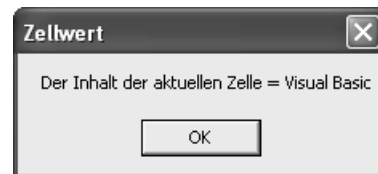
Bei der MsgBox können Sie durch zusätzliche Argumente die Darstellung verändern und den Text der Überschrift selbst bestimmen.



```
Sub Meldung_erweitert()
    MsgBox "dies ist eine Warnung", vbCritical, "eigene Überschrift"
End Sub
```

#### Variable Anzeigetexte

Mit der MsgBox können Sie nicht nur starre Dialoge ausgegeben, sondern auch variable Meldungen, wie etwa der Wert einer bestimmten Zelle anzeigen.



```
Sub Meldung_Parameter()
    Text = "Der Inhalt der aktuellen Zelle = " & ActiveCell.Value
    MsgBox Text, vbOKOnly, "Zellwert"
End Sub
```

#### mehrzeiliger Text

Wollen Sie den anzuzeigenden Text auf mehrere Zeilen verteilen, so trennen Sie den gewünschten Text mit der Konstanten `vbCrLf` an den entsprechenden Stellen.



```
Text = "Wert = " & 10 & vbCrLf
Text = Text & "Wert zwei = " & 22 & vbCrLf
Text = Text & "Wert drei = " & 333
MsgBox Text, , "Anzeige eins"
```

#### Argumente

MsgBox Text, , "Anzeige eins"

Die Meldung im Fenster

Angezeigtes Symbol (siehe nachfolgende Abbildungen)

Beschriftung der Titelzeile

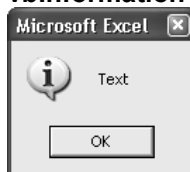
Wenn zwischen den Kommas kein Argument steht wird der Standardwert verwendet.

Mit dem zweiten Argument legen Sie die Anzahl und die Beschriftung der Schaltflächen fest. Dabei können folgende Parameter verwendet werden:

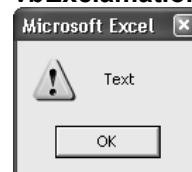
#### vbCritical



#### vbInformation



#### vbExclamation





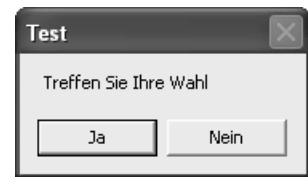
## MsgBox mit Rückgabewert

Bei der erweiterten Version der MsgBox wird ein Wert zurückgegeben, abhängig von der betätigten Schaltfläche. Hier werden, im Gegensatz zur vorhergehenden Version, die Argumente in Klammern geschrieben. Der Variablen vor dem "=" Zeichen wird der Wert der gedrückten Schaltfläche übergeben.



**Übung:**  
Rechnung in  
Liste.....78

```
Sub Meldung_Rückgabe ()
    'mit "Ja" und "Nein" Schaltflächen
    z = MsgBox("Treffen Sie Ihre Wahl", vbYesNo, "Test")
    If z = vbYes Then MsgBox "Sie haben Ja gewählt"
    Else MsgBox "Sie haben Nein gewählt"
    End If
End Sub
```



In diesem Beispielcode wird ein Meldungsfenster mit je einer **Ja** und einer **Nein** Schaltfläche erzeugt. An die Variable **z** wird ein Wert zurückgegeben, der die gedrückte Schaltfläche repräsentiert. Mit der If-Abfrage wird entschieden welche MsgBox angezeigt wird.



Argument	Auswirkung
<b>vbOKOnly</b>	Zeigt nur die Schaltfläche <b>OK</b> an.
<b>vbYesNo</b>	Zeigt die Schaltflächen <b>Ja</b> und <b>Nein</b> an.
<b>vbYesNoCancel</b>	Zeigt zusätzlich zu <b>Ja</b> und <b>Nein</b> die Schaltfläche <b>Abbrechen</b> an.
<b>vbDefaultButton3</b>	Markiert die 3. Schaltfläche.
<b>vbCritical</b>	Zeigt eine Meldung mit Stop-Symbol an.
<b>vbExclamation</b>	Zeigt ein Ausrufezeichen an.
<b>vbMsgBoxRight</b>	Richtet den Meldungstext rechtsbündig aus.

Sie können Konstanten auch durch addieren kombinieren. Sollen zwei Schaltflächen angezeigt werden, eine mit *Ja* und die andere mit *Nein*, so verwenden Sie die Konstante *vbYesNo*. Wollen Sie, dass beim Aufruf immer die zweite Schaltfläche markiert ist, geben Sie ein Pluszeichen ein und fügen *vbDefaultButton2* hinzu. Damit wird in dieser MsgBox immer die Nein-Schaltfläche markiert.

Antwort = MsgBox (Meldung, **vbYesNo + vbDefaultButton2**)



### 3.4 Abfragen

Mit Abfragen können Bedingungen abgefragt und davon abhängig der Programmablauf gesteuert werden.



#### If ... Then Abfragen

##### Einzeilige Abfragen

Die einfachste Form dieser Abfragen sind einzeilige. In diesen Abfragen werden die Bedingung und der weitere Verlauf in einer Zeile notiert.

```
If Wert = 1 Then T = "Wahr"
```

##### Mehrzeilige Abfragen

Die mehrzeiligen Abfragen sind eine weiterführende Form der einzeiligen Abfragen. Hierbei wird der weitere Programmverlauf innerhalb von *If...Then* und *End If* definiert.

```
If Wert = 1 Then
    ActiveCell.Value = "Wahr"
End If
```

##### Else-Abfragen

Bei Else-Abfragen (auch "Sonst-Abfragen" genannt) können Sie einen weiteren möglichen Programmverlauf definieren, in Abhängigkeit vom Wert einer Eingabe oder eines Ergebnisses aus einer Berechnung.

```
If Wert = 1 Then
    ActiveCell.Value = "Wahr"
Else
    ActiveCell.Value = "Falsch"
End If
```

##### Mehrstufige Abfragen

Solche Abfragen können auch ineinander verschachtelt werden, d.h. Sie können innerhalb einer Abfrage mehrere "Dann-Abfragen" definieren und somit die Auswahl noch verfeinern.

```
If Wert = 1 Then
    ActiveCell.Value = "eins"
ElseIf Wert = 2 Then
    ActiveCell.Value = "zwei"
ElseIf Wert = 3 Then
    ActiveCell.Value = "drei"
End If
```

#### IIF-Abfrage

Diese Abfrage entspricht der Wenn ... dann ... sonst-Struktur der *WENN Funktion*, die Sie von Excel her kennen.

```
IIF(Wert = 1, x = "Wahr", x = "Falsch")
```

#### Choose

Damit können Sie aus einer Liste von Argumenten, einen an der angegebenen Position auswählen. Das Argument Zahl gibt die Stelle der Position an.

```
Wochentag = Choose(Zahl, "So", "Mo", "Di", "Mi", "Do", "Fr", "Sa")
```

<b>Übungen:</b>	
Umfrage .....	65
Eingabe-	
maske .....	68
Rechnungen in	
Liste .....	78
Rechnungsnum-	
mer aus Liste	80
Ostertermin .....	87



## Select ... Case – Abfragen

Sie können alternativ zur *If...Then* Verzweigungsstruktur auch die *Select...Case* Abfragen benutzen. Der Vorteil von der *Select...Case* Verzweigungsstruktur ist, dass diese Variante bei der Formulierung von Verzweigungen in vielen Fällen übersichtlicher ist als die *If...Then* Abfrage.



### Einfache Bedingung

Im Anschluss an *Select Case* müssen Sie den zu analysierenden Ausdruck angeben. Dieser gilt dann für die gesamte Verzweigungsstruktur, was eine Einschränkung gegenüber der *If...Then* Abfrage darstellt. In den nachfolgenden *Case*-Zweigen müssen Sie dann Bedingungen formulieren, die den Ausdruck erfüllen. Dazu können Sie einzelne Werte aufzählen wie im folgenden Beispiel.

#### Übungen:

Kegelbe- rechnung .....	84
Zahl in Worten.....	85
Feiertage .....	87

```
Select Case Wert
    Case 1: Txt = "eins"
    Case 2: Txt = "zwei"
    Case 3: Txt = "drei"
    Case Else: Txt = "nichts"
End Select
```

### Mehrere Bedingungen

Des Weiteren können Sie mit dem Schlüsselwort *To* auch Bereiche angeben, in denen die Abfrage erfüllt sein soll. Außerdem können Sie einzelne Werte, durch ein Komma getrennt, angeben. Werden Vergleiche mit *>* oder *<* durchgeführt, so muss *Is* vorangestellt werden.

```
Select Case Wert
    Case 1 To 9
        Txt = "unter Zehn"
    Case 12, 14, 16
        Txt = "gerade"
    Case Is > 16
        Txt = "größer 16"
    Case Else
        Txt = "nicht gefunden"
End Select
```



### 3.5 Schleifen

#### For ... Next

Bei dieser Schleife wird schon zu Beginn festgelegt, wie oft die Schleife durchlaufen werden soll. Mit anderen Worten die *For...Next*-Schleifen sind abweisend, d.h. wenn der Startwert größer als der Endwert ist, so wird die Schleife erst gar nicht durchlaufen.



#### Einfache Schleife

Die Variable *Zahl* beginnt bei 1 und wird schrittweise bis 5 hochgezählt, dabei wird der Code zwischen *For* und *Next* sofort ausgeführt bis die Zahl hinter *To* erreicht ist.

Im Beispiel wird in die Zellen die jeweilige Zahl eingetragen.

```
Sub Einfach()
    For Zahl = 1 To 5
        ActiveCell.Cells(Zahl, 1).Value = Zahl
    Next Zahl
End Sub
```

	A
1	1
2	2
3	3
4	4
5	5

<b>Übungen:</b>	
Tabellenblätter	
sortieren .....	74
Liste der Tabellenblätter .....	75
Farbe	
summieren .....	86

#### Schrittweite festlegen

Sie können bei dieser Art von Schleifen auch eine Schrittweite angeben, mit der hoch gezählt werden soll.

Im Beispiel wird mit einer Schrittweite 2 gezählt, dies wird durch den Eintrag *Step 2* in der *For*-Zeile festgelegt

```
Sub Test()
    For Zahl = 1 To 5 Step 2
        ActiveCell.Cells(Zahl, 1).Value = Zahl
    Next Zahl
End Sub
```

	A
1	1
2	
3	3
4	
5	5

#### Rückwärts zählen

Es ist außerdem möglich rückwärts zu zählen. Dafür müssen Sie lediglich vor der Zahl die bei *Step* angegeben ist ein Minuszeichen setzen.

```
Sub Test()
    For Zahl = 15 To 10 Step -1
        ActiveCell.Cells(Zahl, 1).Value = Zahl
    Next Zahl
End Sub
```

	A
10	10
11	11
12	12
13	13
14	14
15	15

#### Verschachtelte Schleifen

Sie können auch jederzeit verschiedene Schleifen ineinander verschachteln. Im Beispiel startet die Schleife *Spalte*, danach wird die Schleife *Zeile* viermal durchlaufen. Danach zählt die Schleife *Spalte* um eins weiter, danach die Schleife *Zeile* wieder viermal, solange bis die äußere Schleife beendet ist.

	A	B	C	D
1	11	12	13	14
2	21	22	23	24
3	31	32	33	34
4	41	42	43	44

```
Sub Verschachtelt()
    For Spalte = 1 To 5
        For Zeile = 1 To 4
            Cells(Zeile, Spalte).Value = Zeile & Spalte
        Next Zeile
    Next Spalte
End Sub
```



## Do ... Loop – Schleife

Bei dieser Schleifenart steht am Anfang noch nicht fest, wie oft diese durchlaufen wird. Die Bedingung wird während der Laufzeit ermittelt. Dies bedeutet, dass die *Do...Loop*-Schleifen in ihrer einfachsten Form Endlosschleifen sind. Dies ist jedoch nicht alles, denn Sie können mit den Schlüsselwörtern *While* und *Until* sowohl hinter *Do* als auch hinter *Loop* noch Bedingungen formulieren. Dabei ist entscheidend, wo die jeweilige Bedingung eingefügt wird.

### While

Bei diesem Beispiel wird diese Schleife solange ausgeführt, wie die Zahl kleiner als 5 ist.

```
Do
    Zahl = Zahl + 1
    Cells(Zahl, 1).Value = Zahl
Loop While Zahl < 5
```

	A
1	1
2	2
3	3
4	4
5	5
6	

### Until

Im Gegensatz zu *While* wird bei der *Until* Schleife diese solange ausgeführt, bis die Zahl nicht größer als 4 ist. Das Ergebnis ist in diesem Falle das gleiche wie im Beispiel oben, da nach dem vierten Durchlauf die Schleife noch einmal durchlaufen wird, da die Bedingung noch nicht erfüllt ist.

```
Do
    Zahl = Zahl + 1
    Cells(Zahl, 1).Value = Zahl
Loop Until Zahl > 4
```

### Abfrage am Anfang

Im folgenden Beispiel ist die Bedingung von vorn herein unwahr. Darum wird diese Schleife nicht durchlaufen. Es wird auch kein Wert in die Tabelle geschrieben.

```
Zahl = 5
Do While Zahl < 5
    Zahl = Zahl + 1
    Cells(Zahl, 1).Value = Zahl
Loop
```

	A
1	
2	
3	
4	
5	

### Abfrage am Ende

Obwohl die Bedingung von vorn herein unwahr ist, wird diese Schleife zumindest einmal durchlaufen, da die Bedingung erst am Ende der Schleife abgefragt wird.

```
Zahl = 5
Do
    Zahl = Zahl + 1
    Cells(Zahl, 1).Value = Zahl
Loop While Zahl < 5
```

	A
1	
2	
3	
4	
5	
6	6
7	

### Verlassen mit Exit Do

Sie können die Bedingung zum Verlassen der Schleife auch innerhalb der Schleife einfügen.

```
Do
    Zahl = Zahl + 1
    Cells(Zahl, 1).Value = Zahl
    If Zahl > 4 Then Exit Do
Loop
```

	A
1	1
2	2
3	3
4	4
5	5
6	



## 3.6 Datums und Zeitfunktionen

### Day, Month, Year

Diese Funktionen geben den Tag, das Monat oder das Jahr eines Datums aus.

```
Tag = Day("21.10.2003")
Monat = Month("21.10.2003")
Jahr = Year("21.10.2003")
```



Übung:  
Feiertage.....87

### Dateserial

Erzeugt aus einzelnen Angaben für Jahr, Monat und Tag ein gültiges Datum.

```
Datum = DateSerial(Jahr, Monat, Tag)
```

### Datevalue

Erzeugt aus einem Datum, das in einer gültigen Form, als Zeichenkette angegeben wurde, ein Datum mit dem gerechnet werden kann.

```
D = Datevalue("21.Oktober 2006")
```

### Hour, Minute, Second

Diese Funktionen geben die Stunden, die Minuten und die Sekunden einer angegebenen Uhrzeit aus.

```
Stunden = Hour("10:25:15")
Minuten = Minute("10:25:15")
Sekunden = Second("10:25:15")
```

### Systemdatum und Uhrzeit

Die Funktionen *Date*, *Time* und *Now* geben Werte aus der Systemzeit des Computers aus. *Now* gibt das Datum mit der Uhrzeit aus.

```
Uhrzeit = Time
Datum = Date
Datum_Uhrzeit = Now
```


Mit den entsprechenden Anweisungen wird die Systemzeit des Computers geändert.

```
Time = "8:00"
Date = "21.10.2006"
Now = "21.10.2006 8:00"
```

Diese Anweisungen sollten Sie aber mit Vorsicht verwenden!

#### Codebeispiele testen

Die hier gezeigten Beispiele enthalten nur die für die entsprechenden Funktionen entscheidenden Befehle. Zum Testen müssen Sie diese in eine Prozedur oder Funktion einfügen.

Schneller geht das Testen im Direktbereich. Hier geben Sie die ein ?-Zeichen ein, dahinter die Funktion und drücken die  Taste, dann wird Ihnen in der nächsten Zeile das Ergebnis angezeigt. Siehe auch **Befehle Testen im Direktbereich** auf Seite 56.

```
?Time 
15:40:29
?Format(500, "0.00 €") 
500,00 €
```



### 3.7 Textfunktionen



Zum Bearbeiten von Zeichenketten bietet VBA eine Reihe von Funktionen. Diese entsprechen teilweise den Textfunktionen, die in Excel zur Verfügung stehen. In VBA ist die englische Schreibweise zwingend.

#### Left, Right, Mid

Mit den Funktionen *Left* bzw. *Right* wird eine bestimmte Anzahl von Zeichen aus einem Text gelesen. Als erstes Argument wird die Zeichenkette angegeben, als zweites Argument die Anzahl der Stellen, die gelesen werden sollen.

```
Ort = "93462 Lam"
PLZ = Left(Ort, 5)
Ortsname = Right(Ort, 3)      ' Right(Zeichenkette, Zeichenanzahl)
```

In diesem Codebeispiel wird aus einer Ortsangabe mit *Left* die Postleitzahl und im zweiten Schritt mit *Right* der Ortsname gelesen. Letztere Funktion würde bei Ortsnamen mit unterschiedlichen Längen nur dann funktionieren, wenn vorher die Anzahl der Zeichen ermittelt würde. Eine einfache Lösung bietet die Verwendung von *Mid*. Diese Funktion liest die Zeichen ab der im zweiten Argument angegebenen Position.

```
Ortsname = Mid(Ort, 7)      ' Mid(Zeichenkette, Startposition)
```

Ein drittes Argument für die Anzahl der Zeichen, die gelesen werden sollen, brauchen Sie in diesem Fall nicht angeben.

In diesem Beispiel ist die Angabe des dritten Arguments erforderlich. Aus einer Nummer soll die Buchstabenkombination an der 6. und 7. Stelle gelesen werden. Dazu verwenden Sie die *Mid-Funktion* mit 3 Argumenten. Der zweite Parameter ist die *Startposition* der 3. die *Anzahl der Zeichen*.

```
Nummer = "0256-BK-5689"
Kennz = Mid(Nummer, 6, 2)
'      Mid(Zeichenkette, Startposition, Zeichenanzahl)
```

Wenn Sie diesen Code ausführen enthält die Variable *Kennz* den Wert *BK*.

#### Mid-Anweisung

*Mid* ist auch eine Anweisung mit der Zeichen in einem Text an der angegebenen Position getauscht werden können. In dieser Codezeile werden in der zuvor verwendeten Variablen *Nummer* an den Positionen 6 und 7 die beiden Buchstaben *XV* eingetragen und damit die Nummer aktualisiert.

```
Mid(Nummer, 6, 2) = "XV"
```

#### Len, Trim, LTrim und RTrim

Die Funktion *Len* ermittelt die Länge einer Zeichenkette, einschließlich evtl. vorhandener Leerzeichen am Anfang und am Ende. Die Funktion *Trim* entfernt diese Leerzeichen. *LTrim* entfernt nur die Leerzeichen links und *RTrim* nur die Leerzeichen rechts.

```
Vorgabe = "   Beispieltext   "
Zahl1 = Len(Vorgabe)
Getrimmt = Trim(Vorgabe)
Zahl2 = Len(Getrimmt)
```



## Groß und Kleinschreibung

Die Funktionen mit denen Zeichenkette in Klein- bzw. Großbuchstaben umgewandelt werden sind in VBA *LCase* für Kleinbuchstaben und *UCase* für Großbuchstaben.

### UCase("Text")

Die Funktion *UCase* wandelt alle Kleinbuchstaben in Großbuchstaben um. Es wird ein Wert vom Typ Variant (String) zurückgegeben, der die Zeichenfolge, in Großbuchstaben umgewandelte Zeichenfolge enthält.

### LCase("Text")

Die Funktion *LCase* gibt einen Wert vom Typ String zurück, in dem alle Buchstaben in Kleinbuchstaben umgewandelt worden sind.

```
klein = LCase("kleinGROSS")
GROSS = UCase("kleinGROSS")
```



<b>Übung:</b> Tabellen sortieren .....74
------------------------------------------------

### Dezimaltrennzeichen Punkt.

Beachten Sie, das Dezimaltrennzeichen in VBA ist der Punkt. Enthält eine Zahlenreihe ein Komma, wird mit *Val* nur der Teil vor dem Komma als Zahl verwendet.

## Text in Zahl und umgekehrt

Manchmal ist es erforderlich eine Zeichenkette in eine Zahl oder eine Zahl in eine Zeichenkette umzuwandeln. *Str* erzeugt einen Text und *Val* eine Zahl

```
Text = Str(9999)
Zahl = Val("12345.34 €")
```

## Stelle eines Zeichens

Mit der Funktion *InStr* wird die Stelle ermittelt, an der ein gesuchtes Zeichen im Text vorkommt. Es kann der Anfang angegeben werden ab welcher Stelle der Text durchsucht werden soll. In diesem Beispiel wird gesucht an welcher Stelle im Text der Punkt ist. Als Ergebnis wird 3 ausgegeben.

```
Text = "12.34 €"
Stelle = InStr(1, Text, ".")
' InStr(Anfang, Text, Zeichen)
```

## Mit dem ASCII-Code arbeiten

Jedes Zeichen, das am PC dargestellt wird hat eine Nummer. Mit der Funktion *Asc* ermitteln Sie die Nummer eines Zeichens, während die Funktion *Chr* das Zeichen mit der angegebenen Nummer ausgibt.

```
Zeichen = Chr(216)
Nummer = Asc("€")
```

Wenn Sie die Befehle ausprobieren. Die Nummer des € Zeichens ist 128 und das Zeichen mit der Nummer 216 ist das Ø-Zeichen.

## Format

wandelt eine Zahl in eine formatierte Zeichenkette um, ähnlich wie beim benutzerdefinierten Format.

```
Betrag = Format(500, "0.00 €")
Datum = Format(39000, "dd/mm/yy")
```