R|R|Z|N
Regionales Rechenzentrum für Niedersachsen

Leibniz Universität Hannover

Excel – VBA Grundlagen

Code-Gerüst für ein Makro selber schreiben

- Öffnen Sie mit <ALT>+<F11> den Visual Basic Editor.
- Klicken Sie auf *Einfügen* – *Modul*. Es wird ein neues Modul im Codefenster geöffnet.
- Klicken Sie mit der Maus auf die erste freie Zeile. Die Einfügemarke wird eingeblendet.
- Geben Sie `Sub AusgabeInZelle` ein. Bestätigen Sie die Eingabe mit <RETURN>.
- Die Zeile `End Sub` sowie die runden Klammern werden automatisch eingefügt.

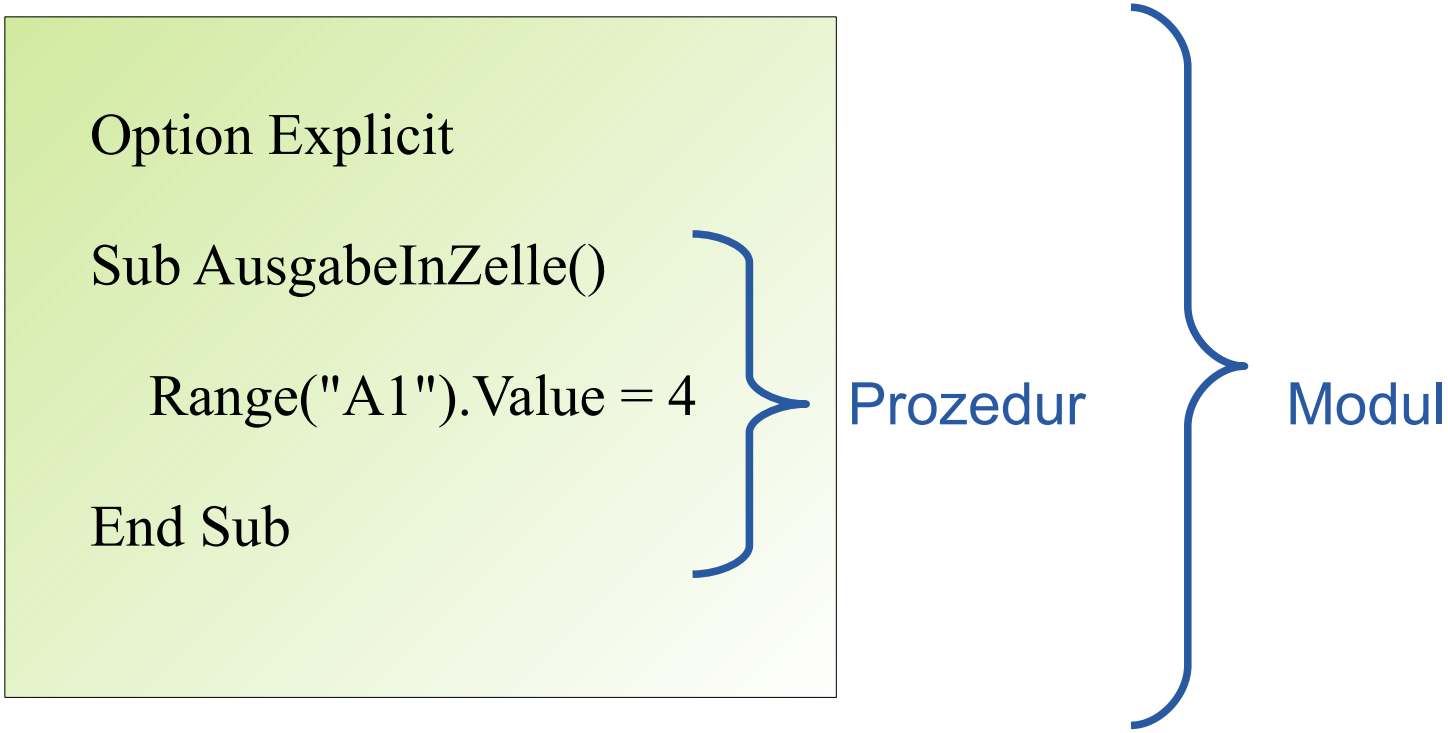
Aktion "Schreibe in Zelle" einfügen

- Setzen Sie die Einfügemarke zwischen die Anweisung `Sub ... End Sub`.
- Geben Sie die Anweisung `Range("A1").Value = 4` ein.
- Führen Sie das Makro mit Hilfe von <F5> aus.

Programmcode "Wert in eine Zelle schreiben"

```
Option Explicit

Sub AusgabeInZelle()

    Range("A1").Value = 4

End Sub
```

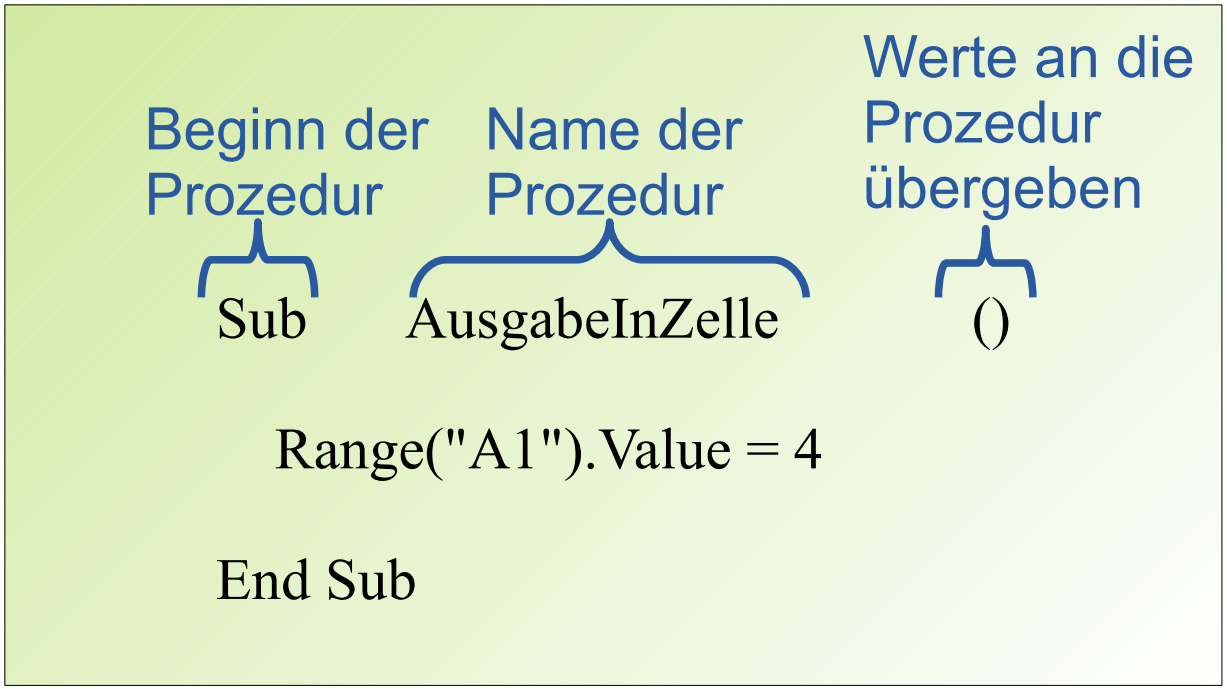
Bestandteile eines Programms



Module und Prozeduren

- Ein Modul
  - ... fasst Code zu einer Aufgabe zusammen.
  - ... ist ein Container für viele einzelne Codeschnipsel.
  - ... ist ein Buch aus Code.
  - Am Beginn eines Moduls stehen Anweisungen, die für alle Kapitel Gültigkeit haben.
- Eine Prozedur
  - ... fasst eine Gruppe von Anweisungen zusammen.
  - ... sind die verschiedenen Kapitel in einem Buch.
  - ... können Werte übergeben werden.
  - Jede Prozedur beginnt mit Sub und endet mit End Sub.

Aufbau einer Prozedur



Der Prozedurnamen

- ... ist frei wählbar.
- ... kann aus bis zu 255 Zeichen bestehen.
- ... besteht aus Zahlen, den Groß- und Kleinbuchstaben des Alphabets sowie den Unterstrich.
- ... beginnt immer mit einem Buchstaben.
- ... sollte die Aktion widerspiegeln.

Anweisungen in Prozeduren

```
Option Explicit

Sub AusgabeInZelle()

    Range("A1").Value = 4

End Sub
```

- ... sind Aktionen, die der Computer ausführen soll. In diesem Beispiel "Schreibe einen Wert in eine bestimmte Zelle".
- Pro Zeile steht eine Anweisung.

Zuweisungen

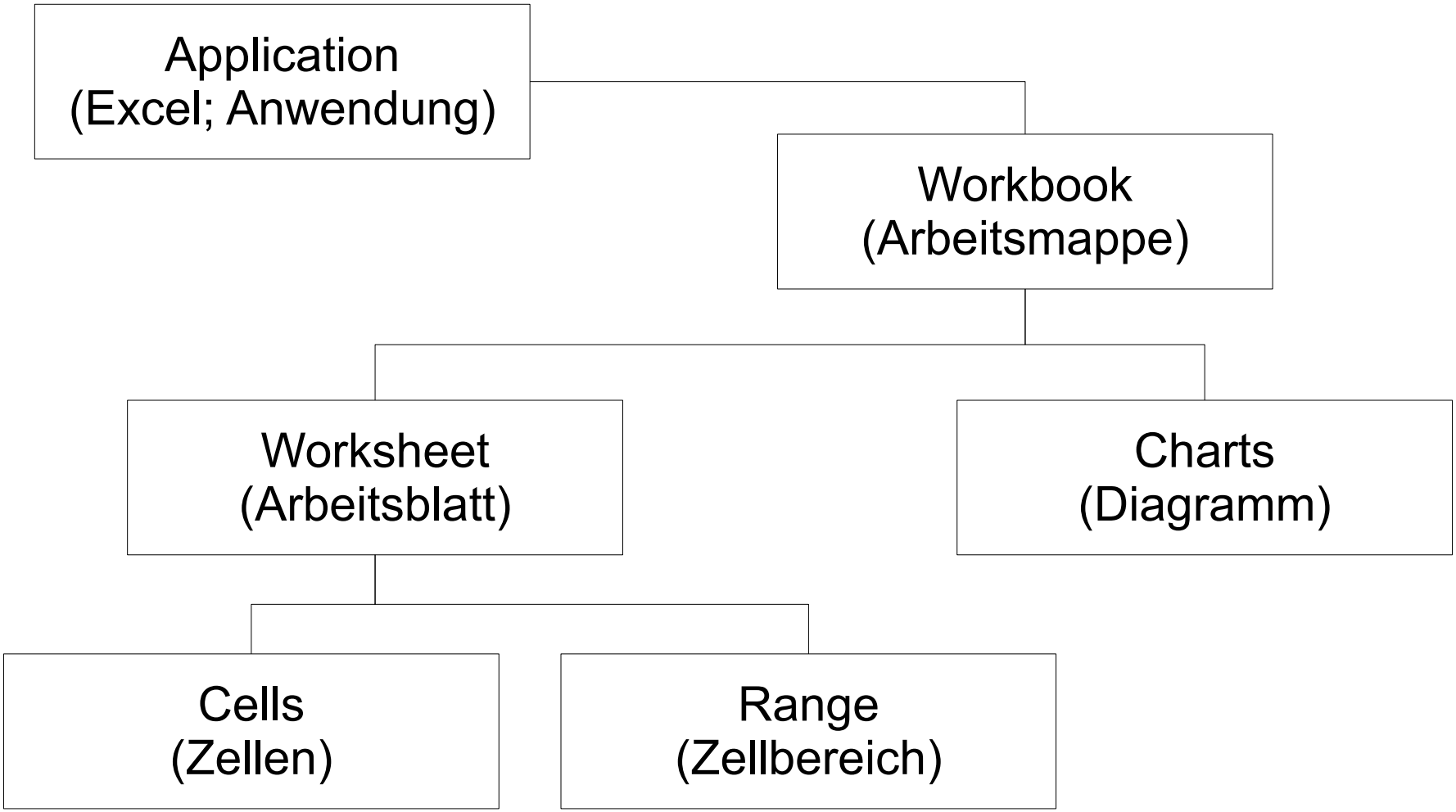
```
Range("A1").Value = 4
```

- Mit Hilfe des Gleichheitszeichen wird einem Element ein Wert zugewiesen.
- Dem Platzhalter links vom Gleichheitszeichen wird der Wert rechts vom Gleichheitszeichen zugewiesen. Der Wert rechts vom Gleichheitszeichen kann auch berechnet werden.

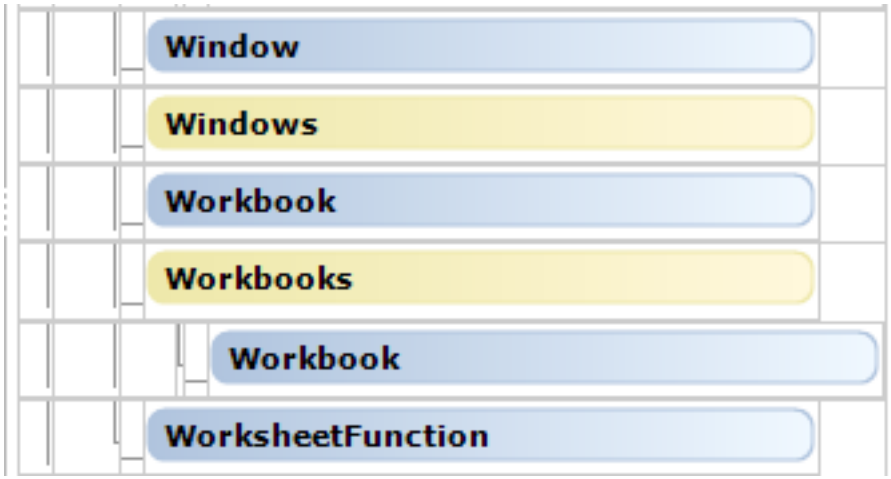
Objekte

- ... sind Bestandteile der Anwendung Excel oder anderer Office-Anwendungen.
- ... sind Substantive in einem Text.
- ... haben bestimmte Eigenschaften (Attribute) und Methoden (Prozeduren).
- ... berichten über ihren Zustand und können diesen über Methoden verändern.
- ... reagieren auf Aktionen mit Hilfe von Ereignissen.
- ... werden in einer Baumstruktur abgebildet. Die einzelnen Hierarchie-Ebenen werden durch Punkte miteinander verbunden. Beispiel: `Worksheet("Tabelle").Range("A1")`.

Ausschnitt aus dem Objektmodell



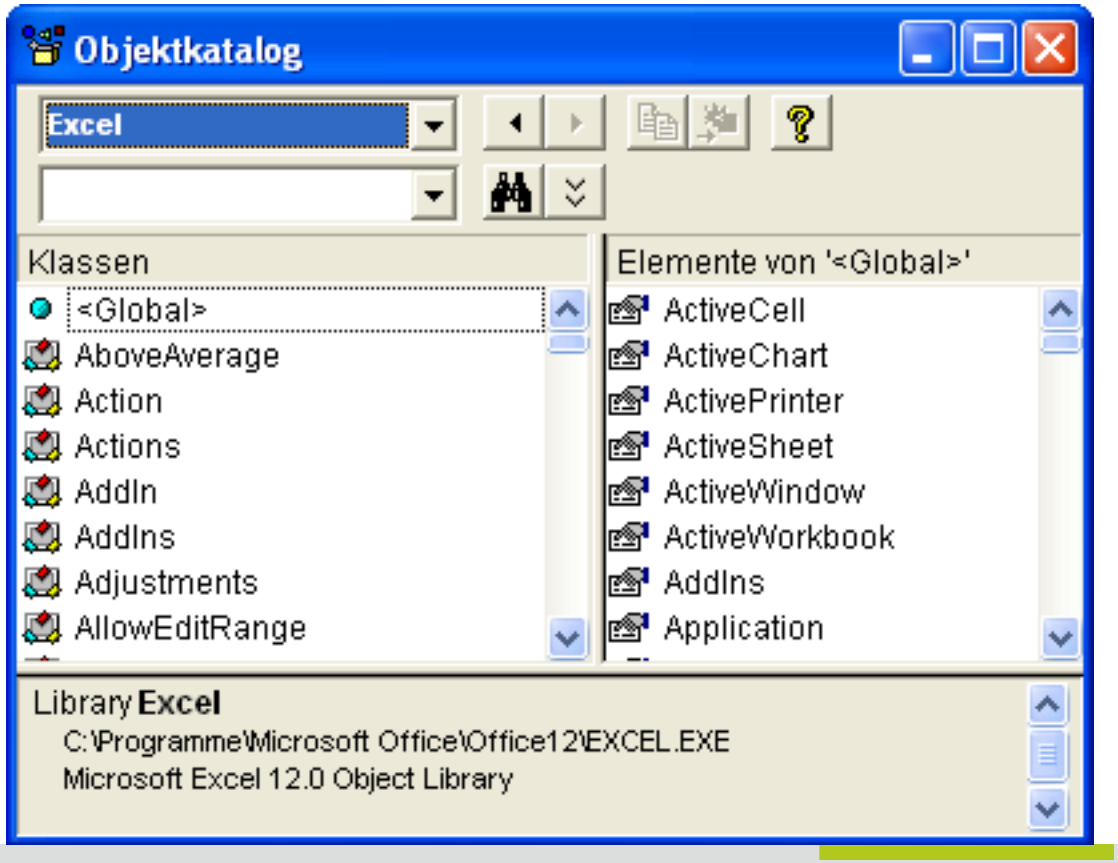
Informationen zum Objektmodell



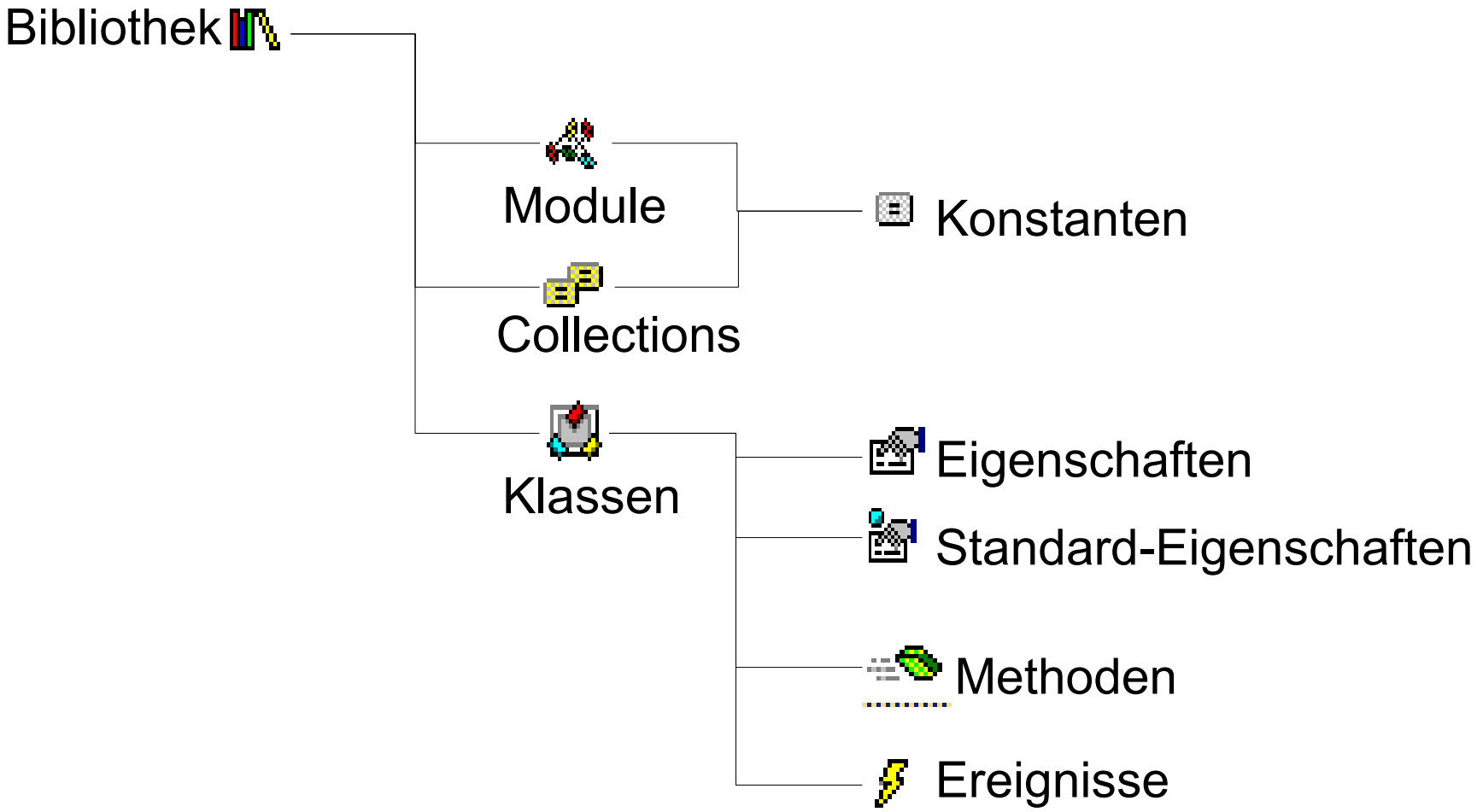
- http://msdn.microsoft.com/en-us/library/bb332345.aspx
- Objektkatalog, integriert im VBA-Editor.

Objektkatalog

- Öffnen Sie mit <ALT>+<F11> den Visual Basic Editor.
- Klicken Sie auf den Befehl *Ansicht - Objektkatalog*.



Symbole im Katalog



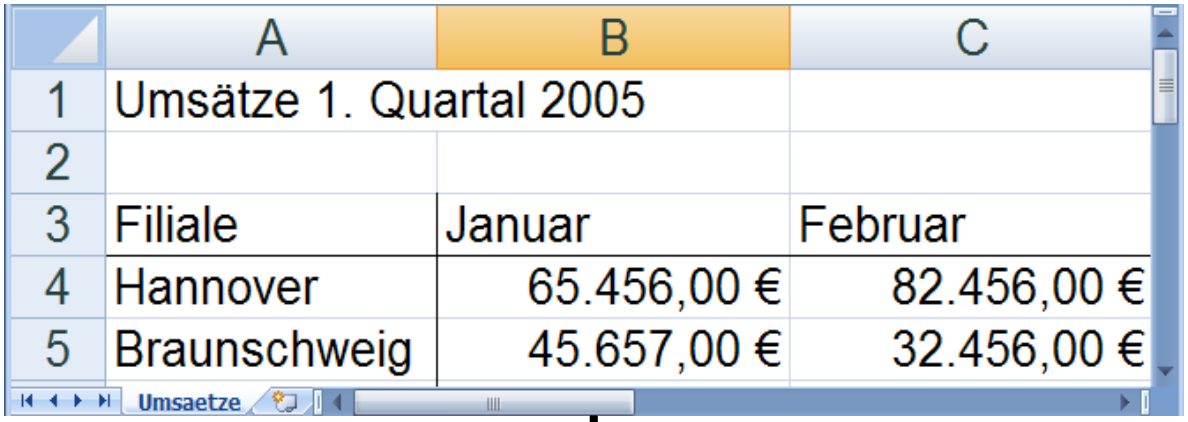
Bibliotheken und Objekte

- Objekte werden in Bibliotheken thematisch zusammengefasst. Wählen Sie aus dem oberen Kombinationsfeld eine Bibliothek (zum Beispiel [Excel]) aus.
- In der Liste Klassen werden alle Objekte von Excel angezeigt. Wählen Sie zum Beispiel das Objekt [Range] aus.
- In der Liste Elemente werden alle Eigenschaften und Methoden des ausgewählten Objekts angezeigt.

Objekte suchen

- Wählen Sie aus dem oberen Kombinationsfeld eine Bibliothek (zum Beispiel [Alle Bibliotheken) aus.
- Geben Sie in das freie Textfeld darunter zum Beispiel <Workbook> ein.
- Klicken Sie auf das Symbol *Fernglas*.
- Anschließend werden eine Liste mit allen Fundstellen eingeblendet. Es wird die Bibliothek, das Objekt sowie das Element angegeben.
- In Abhängigkeit des ausgewählten Elements werden alle dazugehörigen Objekte und Eigenschaften, Methoden und so weiter angezeigt.

Eigenschaften, Methoden und Ereignisse eines ...



	A	B	C
1	Umsätze 1. Quartal 2005		
2			
3	Filiale	Januar	Februar
4	Hannover	65.456,00 €	82.456,00 €
5	Braunschweig	45.657,00 €	32.456,00 €

Umsaetze

Eigenschaften:
- Cells (Zellen)
- Columns (Spalten)
- Rows (Zeilen)
- Name

Methoden:
- Activate
- Delete
- PrintOut
- SaveAs

Ereignisse:
- Activate
- Calculate
- Change
- SelectionChange

Eigenschaften (Attribute)

- ... beschreiben das Aussehen eines Objekts.
- ... beschreiben einen Gegenstand.
- ... sind statische Werte, die ein Objekt kennzeichnen.
- ... beschreiben den aktuellen Zustand eines Objekts.
- Eigenschaftswerte können immer gelesen werden. Einige Eigenschaftswerte können mit Hilfe von VBA verändert werden.
- ... werden mit Hilfe des Punktes mit einem bestimmten Objekt verbunden.

Methoden (Member, Elementfunktion)

- ... beschreiben das Verhalten eines Objekts.
- ... verändern oder lesen Attribute des dazugehörigen Objekts.
- ... sind Prozeduren, die an ein Objekt gebunden sind.
- ... ermöglichen die Kommunikation mit anderen Objekten.
- ... werden mit Hilfe des Punktes mit einem bestimmten Objekt verbunden.

Ereignis (Event)

- .. tritt immer in Verbindung mit einem Objekt auf.
- ... wird vom Nutzer des Objekts ausgelöst.
- ... ist eine Reaktion auf eine Benutzeraktion.
- Beispiele für Ereignisse:
  - Ein Arbeitsblatt wird aktiviert.
  - Es werden Zellen berechnet.
  - Die Auswahl der Zelle verändert sich.
  - Ein neues Arbeitsblatt wird angelegt.

Arbeitsmappen nutzen

```
Sub ArbeitsmappeAktuell()
    Dim arbeitsmappeName As String

    arbeitsmappeName = ThisWorkbook.Name
    arbeitsmappeName = ActiveWorkbook.Name

    arbeitsmappeName = Workbooks(arbeitsmappeName).Name
    arbeitsmappeName = Workbooks("umsatz.xls").Name
    arbeitsmappeName = Workbooks(1).Name

End Sub
```

ThisWorkbook und ActiveWorkbook

ThisWorkbook

- ... beschreibt die Arbeitsmappe, in der der Code aktuell ausgeführt wird.

ActiveWorkbook

- ... beschreibt die aktuell aktive Arbeitsmappe.
- ... ist die vom Benutzer ausgewählte Arbeitsmappe.

- Beide Objekte können auf unterschiedliche Arbeitsmappen verweisen.

Variablen

- ... sind Platzhalter für Werte.
- ... speichern einen bestimmten Typ von Wert
- Die Werte sind veränderbar.
- ... können Werte zugewiesen bekommen.
- Definition: `Dim arbeitsmappeName As String`
  - `Dim`. Wer kann auf die Variable zugreifen? Momentan kann nur innerhalb der Prozedur auf die Variable zugegriffen werden.
  - Der Variablenname ist frei wählbar und einmalig.
  - `As` leitet die Typisierung der Variablen ein. In diesem Beispiel kann die Variable nur Text (`String`) aufnehmen.

Collections

- ... ist eine Auflistung von Elementen eines bestimmten Typs.
- ... ist eine Auflistung von Objekten mit bestimmten Eigenschaften und Methoden.
- Zum Beispiel werden alle geöffneten Arbeitsmappen in einer Collection gespeichert.
- ... sind immer an der Mehrzahlschreibweise (s am Ende der Bezeichnung) zu erkennen.
- Das erste Element in einer Auflistung hat den Index 0 und so weiter. Der Index verändert sich in Abhängigkeit der darin enthaltenen Elemente.
- ... bestehen aus übereinander gestapelten Schachteln. Der Index nummeriert die vorhandenen Schachteln durch. Der Inhalt der Schachteln spielt dabei keine Rolle.

Eigenschaften und Methoden einer Auflistung

- Die Eigenschaft Count gibt die Anzahl der Elemente innerhalb der Auflistung zurück.
- Die Methode Add fügt ein neues Element am Listenende ein.
- Die Methode Remove löscht ein Element aus der Liste. Der Index aller nachfolgenden Elemente wird angepasst.

Workbooks

- ... listet alle geöffneten Arbeitsmappen auf.
- ... ist eine Zusammenfassung von gleichen Objekten.
- ... ist ein Stapel von allen geöffneten Arbeitsmappen.
- Workbooks.Count ermittelt die Anzahl der geöffneten Arbeitsmappen. Falls sich Makros oder Code in der persönlichen Makroarbeitsmappe befindet, wird diese immer mitgezählt.

Elemente in der Liste

Workbooks(1).Name

- Durch Angabe einer Ganzzahl in den runden Klammern wird ein bestimmtes Objekt in der Liste angesprochen.
- In diesem Beispiel wird das erste Element in der Liste genutzt.

Workbooks(arbeitsmappeName).Name oder
Workbooks("umsatz.xls").Name

- Durch Angabe des Namens wird eine bestimmte Arbeitsmappe aus der Liste ausgewählt.
- Text wird immer durch Anführungszeichen am Anfang und Ende begrenzt.

Name und Speicherort der Arbeitsmappe

Workbooks(1).Fullname

- ... gibt den Speicherort und den Namen der Arbeitsmappe zurück.
- ... gibt den vollständigen Pfad zurück.

Workbooks(arbeitsmappeName).Path

- ... gibt den Speicherort der Arbeitsmappe zurück.

Workbooks(arbeitsmappeName).Name

- ... gibt den Namen der Arbeitsmappe zurück.

Arbeitsmappe speichern

```
Sub ArbeitsmappeSpeichern()

    ' Datei – Speichern [aktive Arbeitsmappe]
    ActiveWorkbook.Save

    ' Datei – Speichern unter [Arbeitsmappe, die zu dem Code gehört]
    ThisWorkbook.SaveAs ActiveWorkbook.Path & "\umsatz2008.xlsm"

    '.Saved. Ist die Arbeitsmappe gespeichert?
    MsgBox "Gespeichert? " & ThisWorkbook.Saved

End Sub
```

Informationen für den Benutzer

MsgBox "Gespeichert? " & ThisWorkbook.Saved

- ... öffnet ein Dialogfeld.
- In dem Dialogfeld wird der Text "Gespeichert? " & ThisWorkbook.Saved angezeigt.
- Text wird immer durch Anführungszeichen am Anfang und Ende begrenzt. Mit Hilfe des kaufmännischen Unds kann Text verknüpft werden.

Arbeitsblätter auswählen

```
Sub TabellenblattAktuell()
    Dim tabellenblattName As String

    tabellenblattName = ActiveSheet.Name

    tabellenblattName = Worksheets(1).Name
    tabellenblattName = Worksheets("Umsaetze").Name

End Sub
```

Möglichkeiten

ActiveSheet.Name

- Das aktuell aktive Arbeitsblatt. Das aktive Arbeitsblatt befindet sich im Vordergrund der Arbeitsmappe.

Worksheets(1).Name

- Durch Angabe einer Ganzzahl in den runden Klammern wird ein bestimmtes Objekt in der Liste angesprochen.

Workbooks("Umsaetze).Name

- Durch Angabe des Namens wird ein bestimmtes Arbeitsblatt aus der Liste ausgewählt.

Arbeitsblätter einfügen

```
Sub TabellenblattNew()
    ThisWorkbook.Activate

    Worksheets.Add
    ActiveSheet.Name = "DiagrammUmsaetze"

    Worksheets.Add After:=Worksheets("DiagrammUmsaetze"), Count:=2
    Worksheets.Add Before:=Worksheets("DiagrammUmsaetze")

End Sub
```

Neues Arbeitsblatt

Worksheets.Add

- ... fügt ein Arbeitsblatt vor dem aktuell aktiven Arbeitsblatt ein.
- Das neue Arbeitsblatt hat die Bezeichnung "Tabelle" plus eine Ziffer. Mit Hilfe der Eigenschaft Name kann diese Bezeichnung verändert werden.

... an einer bestimmten Position einfügen

```
Worksheets.Add After:=Worksheets("DiagrammUmsaetze"), Count:=2
Worksheets.Add Before:=Worksheets("DiagrammUmsaetze")
```

- Mit Hilfe der Parameter After und Before wird das neue Arbeitsblatt vor oder nach einem bestimmten Blatt eingefügt.
- Arbeitsblätter können immer nur vor oder nach einem Blatt eingefügt werden. Arbeitsblätter können nicht zwischen zwei Blättern eingefügt werden.
- Count gibt die Anzahl der einzufügenden Blätter an.

Benannte Parameter

```
Worksheets.Add After:=Worksheets("DiagrammUmsaetze"), Count:=2
 Worksheets.Add Before:=Worksheets("DiagrammUmsaetze")
```

- An eine Prozedur können Parameter übergeben werden. Die Parameter werden innerhalb der Prozedur verarbeitet. Add ist eine vordefinierte Prozedur, der eine bestimmte Anzahl von Parametern übergeben wird.
- Die Parameter werden in einer Liste zusammengefasst. Die Elemente der Liste werden durch Kommata getrennt.
- [Parameter] := [Wert]. Der Name des Parameters wird wie eine Variable genutzt, der ein Wert übergeben wird. Für die Übergabe wird der Operator := genutzt.
- Parameter können optional sein. Das heißt, es muss nicht immer allen Parametern in der Liste ein Wert übergeben werden.

Arbeitsblätter löschen

```
Sub TabellenblattDelete()

    Worksheets("DiagrammUmsaetze").Delete

End Sub
```

In einer Arbeitsmappe befindet sich mindestens ein Arbeitsblatt. Falls Arbeitsblätter mit Daten gelöscht werden, wird eine Warnmeldung ausgegeben.

Arbeitsblätter aktivieren

```
Sub TabelleCell()
    Dim zeile As Integer

    ThisWorkbook.Activate


    Worksheets.Add
    ActiveSheet.Name = "ZeilenNr"
    Worksheets("ZeilenNr").Activate

End Sub
```

Die, zum Code gehörende Arbeitsmappe wird aktiviert.

Es wird ein neues Arbeitsblatt erzeugt. Der Name wird angepasst. Das neu erstellte Arbeitsblatt wird aktiviert.

Zellbereich auswählen

```
Sub TabelleRange()

    Range("A3").Select              ' Eine Zelle auswählen

    Range("A4:D4").Select           ' Ein Zellbereich auswählen

    Range("A4, C4, D4").Select  ' Kombination aus beiden
    Range("A1, A4:D4").Select
End Sub
```

Mit Hilfe der Methode Select wird der angegebene Zellbereich ausgewählt.

Zellen auswählen

```
Sub TabelleCell()
    Dim zeile As Integer

    ThisWorkbook.Activate
    Worksheets.Add
    ActiveSheet.Name = "ZeilenNr"
    Worksheets("ZeilenNr").Activate

    zeile = 1
    ActiveSheet.Cells(zeile, 1).Value = zeile

    zeile = zeile + 1
    ActiveSheet.Cells(zeile, 1).Value = zeile
End Sub
```

Mit Hilfe von Cells wird auf eine bestimmte Zelle im aktiven Arbeitsblatt verwiesen. Die Zelle kann mit Hilfe von Variablen bestimmt werden.

Range und Cells nutzen

- Cells([zeile], [spalte]) verweist auf eine Zelle. Die Zelle wird durch Angabe der Zeile und Spalte ausgewählt.
- Range("[zellbereich]") verweist auf einen Zellbereich. Für die Spalte wird ein Buchstabe und für die Zeile eine Nummer genutzt. Beispiele:

Range("A3")	Eine Zelle.
Range("A3:E4")	Zusammenhängender Zellbereich.
Range("A1, B2, C2")	Nicht zusammenhängender Zellbereich.
Range("A3:E4, E2")	Kombination aus den vorhergehenden Möglichkeiten.
Range("C:E")	Ganze Spalten nutzen.
Range("3:5")	Ganze Zeilen nutzen.

Werte eintragen

```
Sub TabelleWert()

    ActiveSheet.Cells(1, 1).Value = 1                  'Ganzzahl

    ActiveSheet.Cells(2, 1).Value = 1.564              'Dezimalzahl

    ActiveSheet.Cells(3, 1).Value = #5:04:23 PM#       ' Uhrzeit

    ActiveSheet.Cells(4, 1).Value = #12/31/2009#       ' Datum: Monat/Tag

    ActiveSheet.Cells(5, 1).Value = "Einkauf"          ' Text

End Sub
```

Werte formatieren

```
With ActiveSheet.Cells(1, 1)
      .NumberFormat = "000"                    ' 003
      .NumberFormat  = "€ #,##0.00"            ' € 1,50; € 1.000,50
      .NumberFormat = "hh:nn"                  ' 13:30
      .NumberFormat = "dd. mmmm. yyyy"         ' 01. Mai. 2007
       .NumberFormat = "@"                     ' Einkauf
End With
```

Hinweise

- Für jedes Zeichen kann ein Formatierungszeichen als Platzhalter genutzt werden.
- Die Formatierungszeichen werden in der Hilfe von VBA unter dem Stichwort
  - Benutzerdefinierte Datums- und Zeitformate
  - Benutzerdefinierte numerische Formate
  - Benutzerdefinierte Zeichenfolgeformat
- Des Weiteren gibt es benannte Formatierungen wie General.

Formeln in der Landessprache nutzen

```
Sub TabelleFormel()

    ThisWorkbook.Activate
    Worksheets("Rechnung").Activate

    ActiveSheet.Range("C1").FormulaLocal = "=Summe(A1:B1)"

End Sub
```

Es wird eine berechnende Zelle erstellt. Die vordefinierten Funktionen werden in der Systemsteuerung gewählten Sprache geschrieben.

Hinweise

- Es können alle vordefinierten Funktionen, die aus Excel bekannt sind, genutzt werden.
- Die Formel aus der Bearbeitungsleiste wird automatisch mit Hilfe von VBA in die Zellen geschrieben.
- Eine Formel wird als Text übergeben. Die Formel wird durch Anführungsstriche am Anfang und Ende begrenzt.
- Text kann mit Hilfe des kaufmännischen Unds verknüpft werden. Zum Beispiel: "Summe: " & Cells(1, 1).Value.
- Anführungsstriche innerhalb einer Formel müssen durch einen zweiten Anführungsstrich maskiert werden. Zum Beispiel: "=""Summe: "" & SUMME(A1:B1)".

International arbeiten

```
Sub TabelleFormel()

    ThisWorkbook.Activate
    Worksheets("Rechnung").Activate

    ActiveSheet.Range("C1").Formula = "=Sum(A1:B1)"

End Sub
```

Es wird eine berechnende Zelle erstellt. Die Funktion wird in der englischsprachigen Schreibweise geschrieben. Diese Schreibweise nutzt Excel auch intern.

RC-Format für Zellbezüge nutzen

```
Sub TabelleFormel()

    ThisWorkbook.Activate
    Worksheets("Rechnung").Activate

    ActiveSheet.Range("C3").Select
    ActiveSheet.Range("C3").FormulaR1C1 = "=Sum(RC[-2]:RC[-1])"

End Sub
```

In Abhängigkeit der aktuellen Zellen werden die zu addierenden Zellen ermittelt. R bezeichnet Rows (Zeilen) und C Columns (Spalte).

Standardformatierung für alle Zellen

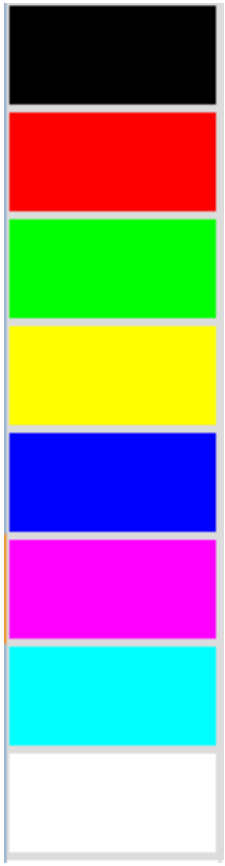
```
Sub TabellenblattFormat()
    ThisWorkbook.Activate

    Worksheets("Umsaetze").Activate

    ActiveSheet.Cells.Font.Name = "Courier New"
    ActiveSheet.Cells.Font.Size = 10
    ActiveSheet.Cells.Font.Color = vbRed

End Sub
```

Die Schrift, die Schriftgröße sowie die Schriftfarbe wird für alle Zellen eingestellt.

Einstellungsmöglichkeiten für die Schrift

- .Font bietet Attribute für die Schriftformatierung der Zelle.
- .Font.Name legt die Schriftart fest.
- .Font.Size legt die Schriftgröße fest.
- .Font.Color legt die Schriftfarbe fest.
- .Font.Bold = True nutzt eine Fettschrift. Der Wert False setzt die Schrift wieder auf Normalschrift.
- .Font.Italic = True nutzt eine Kursivschrift.
- .Font.Underline = False unterstreicht Text.

Grundfarben als Farbkonstanten



vbBlack

vbRed

vbGreen

vbYellow

vbBlue

vbMagenta

vbCyan

vbWhite

Ausrichtung von Text in Zellen

```
Sub FormatText()
  ActiveSheet.Range("A1").Value = 0.26
    ActiveSheet.Range("A1").HorizontalAlignment = xlRight
    ActiveSheet.Range("A1").VerticalAlignment = xlCenter

    ActiveSheet.Range("C1").Value = "Langer Text"
    ActiveSheet.Range("C1").WrapText = True

    ActiveSheet.Range("E1").Value = "H. C. Andersen"
    ActiveSheet.Range("E1:E4").MergeCells = True
    ActiveSheet.Range("E1:E4").Orientation = 45

End Sub
```

Einstellungsmöglichkeiten

- Die Einstellungen entsprechen dem Inhalt der Registerkarte Ausrichtung im Dialogfenster Zellen formatieren.
- Text wird mit Hilfe von HorizontalAlignment und VerticalAlignment in der Zelle ausgerichtet.
- Mit Hilfe von WrapText wird ein automatischer Zeilenumbruch durchgeführt.
- MergeCells verbindet die angegebenen Zellen.
- Falls Orientation
  - ... den Wert xlVertical hat, werden die Buchstaben untereinander geschrieben.
  - ... einen Wert zwischen -90 und +90 Grad hat, wird der Text in einem schrägen Winkel geschrieben.

Rahmen-Formatierungen

```
Sub ZellRahmen()

    ThisWorkbook.Activate
    Worksheets("Text").Activate

    ActiveSheet.Range("A8").Borders.LineStyle = xlDot
    ActiveSheet.Range("A8").Borders.Weight = xlThin
    ActiveSheet.Range("A8").Borders.Color = vbRed

    ActiveSheet.Range("A9:B9").Borders(xlEdgeRight).Color = vbGreen
    ActiveSheet.Range("A9:B9").Borders.Weight = xlThick

End Sub
```

Hinweise

- Die Einstellungen entsprechen dem Inhalt der Registerkarte Rahmen im Dialogfenster Zellen formatieren.
- ActiveSheet.Range("A8").Borders bezieht sich immer auf den gesamten Rahmen.
- ActiveSheet.Range("A8").Borders(xlEdgeRight). In den runden Klammern wird der Name einer Kante an den Rahmen übergeben. In diesem Beispiel wird die rechte Kante formatiert. Weitere Möglichkeiten:
  - xlEdgeLeft. Linke Rahmenkante.
  - xlEdgeTop. Obere Rahmenkante.
  - xlEdgeBottom. Untere Rahmenkante.
  - xlInsideHorizontal. Innere horizontale Zwischenlinie.
  - xlInsideVertical. Innere vertikale Zwischenlinie.

Einstellungsmöglichkeiten

- LineStyle legt die Linienart fest. Zum Beispiel:
  - xlContinuous für eine durchgehende Linie.
  - xlDot für eine gepunkte Linie.
- Weight legt die Dicke der Linie fest. Zum Beispiel
  - xlThin für eine dünne Linie.
  - xlMedium für eine mitteldicke Linie.
  - xlThick für eine dicke Linie.
- Color legt die Rahmenfarbe fest.

Inhalt einer Zelle kopieren und einfügen

```
Sub ZellCopy()

    Worksheets("Umsaetze").Activate

    ActiveSheet.Range("A3:D3").Copy
    ActiveSheet.Paste Destination:=Worksheets("U2009").Range("A3:D3")


    ActiveSheet.Range("A4:A8").Copy _
                    Destination:=Worksheets("U2009").Range("A4:A8")

End Sub
```

Inhalt einer Zelle kopieren

```
ActiveSheet.Range("A3:D3").Copy
```

- Der Inhalt der angegebenen Zellen wird in die Zwischenablage kopiert.
- Mit Hilfe des Parameters Destination kann ein Ablageort für die Kopie angegeben werden.

Inhalt in eine Zelle einfügen

```
ActiveSheet.Paste Destination:=Worksheets("U2009").Range("A4:A8")
```

- Der Inhalt der Zwischenablage wird den angegebenen Zellbereich kopiert.
- Der Inhalt der Zwischenablage kann auf einem beliebigen Arbeitsblatt in einem beliebigen Zellbereich eingefügt werden.

Zellen löschen

```
Sub ZellCopy()
    Worksheets("Umsaetze").Activate

    ActiveSheet.Range("A3").Clear              ' Inhalt und Formatierung

    ActiveSheet.Range("A3").ClearContents      ' Inhalt

    ActiveSheet.Range("A3").ClearFormats       ' Formatierung

End Sub
```

Programm: Nachricht an Benutzer

```
Option Explicit

Sub AusgabeAnBenutzer()

    MsgBox "Hello...", vbOKCancel, "Willkommen"

End Sub
```

Es wird eine Dialogbox mit dem Text "Hello" angezeigt. In der Titelleiste der Dialogbox steht "Willkommen" als Hinweis auf die Nutzung. Es werden die Schaltfläche OK und Abbrechen angezeigt.
Text wird immer in Anführungszeichen gesetzt.

Programm: Ausgabe in das Direktfenster

```
Option Explicit

Sub AusgabeDirekt()

    Debug.Print Range("A1").Value

End Sub
```

Es wird der Inhalt der Zelle A1 im Direktfenster ausgegeben. Das Direktfenster wird häufig für die Überprüfung von Werten genutzt.

Direktfenster

- Öffnen Sie das Direktfenster mit Hilfe des Menüs *Ansicht* – *Direktfenster*.
- In diesem Fenster
  - ... kann der Ablauf eines Programms in einem gewissen Umfang protokolliert werden.
  - ... können Prozeduren oder Anweisungen getestet werden.
- Mit Hilfe von <STRG>+<A> wird der gesamte Inhalt des Direktfensters markiert.
- Mit Hilfe der Taste <ENTF> kann der markierte Inhalt im Direktfenster gelöscht werden.

Eingaben in das Direktfenster

- Geben Sie die Anweisung `?Range("A1").Value` in eine freie Zeile im Direktfenster ein. Schließen Sie die Anweisung mit <RETURN> ab. In der nächsten Zeile wird der Wert der Zelle A1 angezeigt.
- Die Anweisung `Range("A1").Value = 5` verändert den Wert in der Zelle A1.