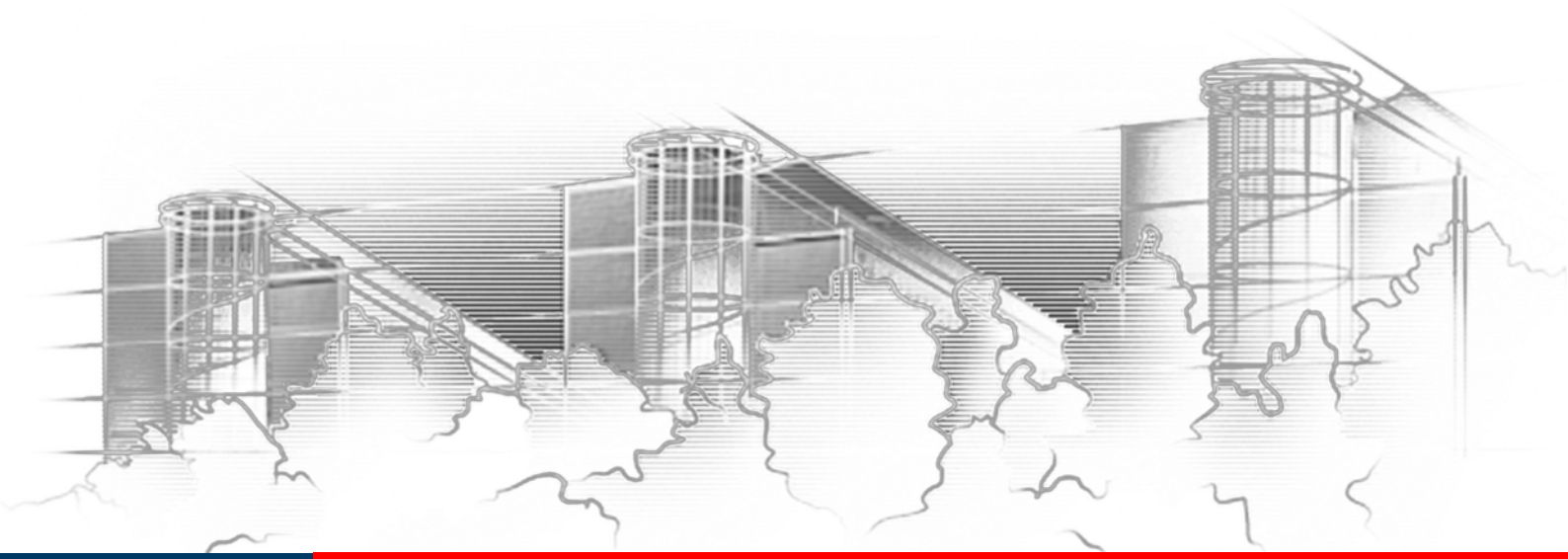
Freie
Hansestadt
Bremen



MAKROS – EINFÜHRUNG

EXCEL 2016

AFZ
wir bilden zukunft
Aus- und Fortbildungszentrum
für den bremischen öffentlichen Dienst

1. Einleitung und grundlegende Einstellungen

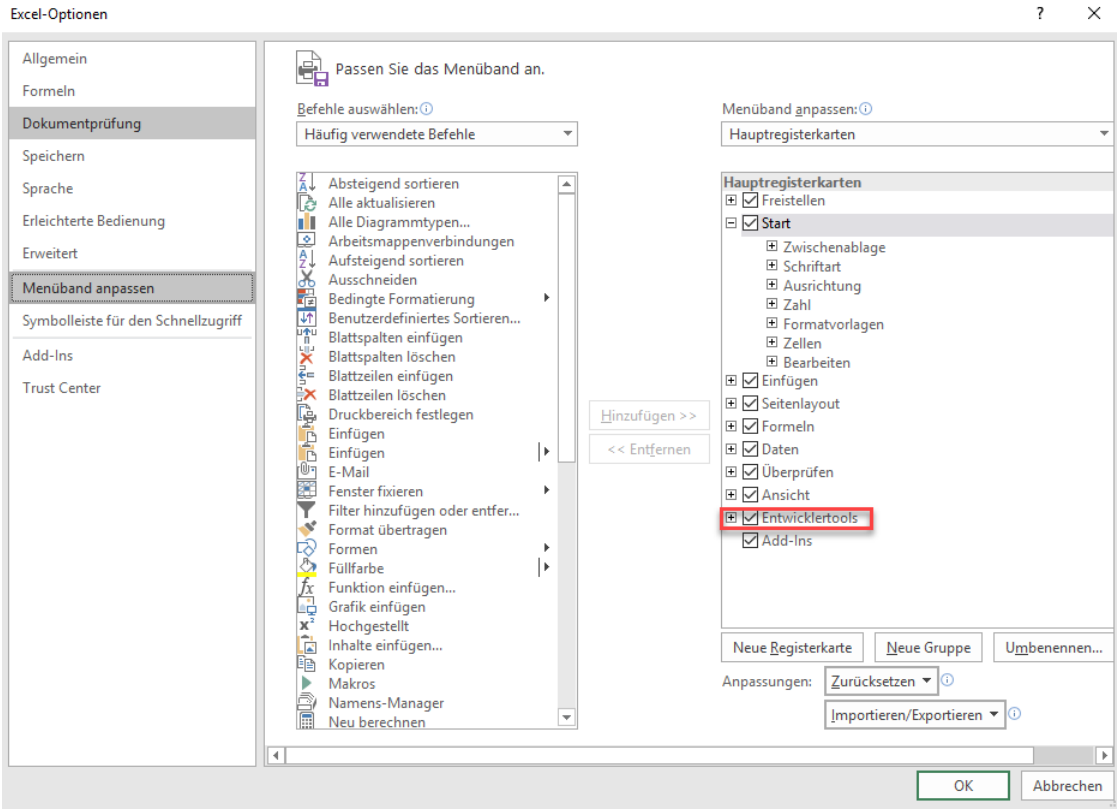
Ein Makro ist eine Zusammenfassung von Einzelanweisungen, um diese automatisiert gemeinsam auszuführen, ohne die Einzelanweisungen getrennt voneinander durchführen zu müssen.

Da Microsoft Office-Anwendungen bieten dabei grundsätzlich zwei Möglichkeiten, mit Makros zu arbeiten. Entweder man schreibt Makros selbst oder zeichnet diese auf. In beiden Fällen muss man wissen, wie man an die entsprechenden Werkzeuge herankommt.

1.1 Menüband Entwicklertools

Das Menüband Entwicklertools wird standardmäßig ausgeblendet. Um es sichtbar zu machen, gehen Sie wie folgt vor:

- Klicken Sie auf **Datei**, um in die Backstageansicht zu gelangen.
- Auf der linken Seite – im Navigationsmenü – wählen Sie den Eintrag **Optionen**.
- In der Dialogbox **Excel-Optionen** wählen Sie auf der linken Seite die Position **Menüband anpassen**.
- Im rechten Bereich der Dialogbox klicken Sie auf das Kontrollkästchen bei Entwicklertools.
- Klicken Sie abschließend auf die Schaltfläche OK, um die Dialogbox zu verlassen. Excel zeigt nun die Registerkarte **Entwicklertools** an.

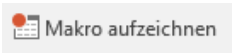


- Klicken Sie abschließend auf die Schaltfläche **OK**, um die Dialogbox zu verlassen.

Excel zeigt nun die Registerkarte **Entwicklertools** zusätzlich an.

- Die Registerkarte **Entwicklertools** beginnt mit der Gruppe **Code**.

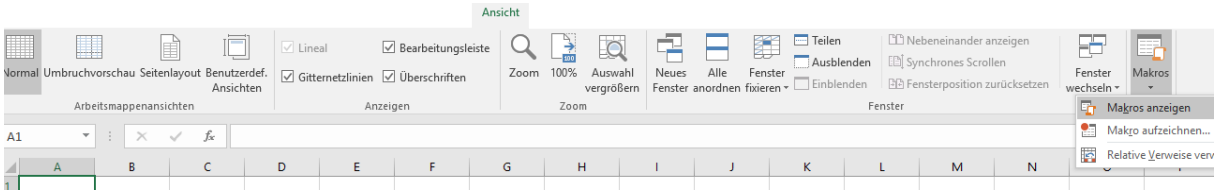
Die Schaltfläche **Visual Basic** öffnet den Visual Basic Editor. Im Editor können Sie Makros schreiben, ändern oder auch ganz löschen.



Die Schaltfläche **Makro aufzeichnen** startet die Aufzeichnung eines Makros.

1.2 Andere Wege der Makro-Aufzeichnung

Das Symbol **Makro aufzeichnen** finden Sie noch an weiteren Stellen, z. B. im Menüband **Ansicht**. Dort befindet sich ganz rechts die Gruppe **Makros**.



In der Statusleiste finden Sie ebenfalls das Symbol zum Aufzeichnen eines Makros.

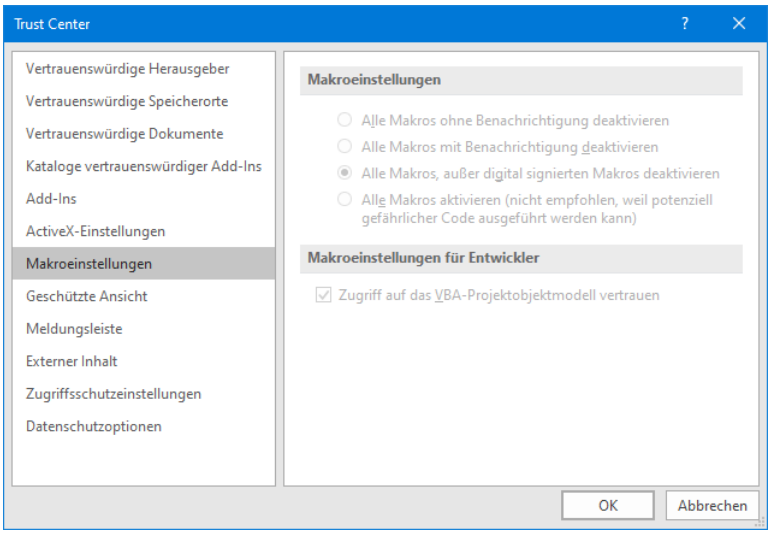
Fehlt das Symbol, können Sie es einblenden. Sie klicken mit der rechten Maustaste in die Statuszeile und wählen den Eintrag **Makroaufzeichnung**. Das Symbol bekommt einen Haken, das bedeutet, es ist jetzt eingeblendet.

1.3 Dateitypen und Makrosicherheit

Mit Office 2007 hat Microsoft neue Dateitypen für die Office-Anwendungen eingeführt. Eine Excel-Datei kann nun nicht mehr grundsätzlich Makros speichern. Dateien mit der Endung **xlsx** enthalten nie Makros im Gegensatz zu Dateien mit der Endung **xlsm**. Diese Dateien **können** Makros enthalten, müssen dies aber nicht.

Da Makros grundsätzlich auch ein Sicherheitsrisiko darstellen, gibt es ein extra Sicherheitscenter in den MS Office-Anwendungen. In diesem Sicherheitscenter können die Einstellungen eingesehen werden, die für die Makrosicherheit (von der Dataport) gesetzt worden sind. In der Regel können Sie die Sicherheitseinstellungen nicht ändern.

Um in das Sicherheitscenter für Makros zu gelangen, klicken Sie auf der Registerkarte **Entwicklertools** ganz links, in der Gruppe **Code**, auf den Befehl **Makrosicherh**. Das Dialogfeld **Trust Center** wird geöffnet.



Links in der Navigationsspalte ist die Option **Makrosteinstellung** bereits grau unterlegt. Im rechten Teil des Dialogfeldes sehen Sie die Voreinstellungen für die Makrosicherheit. Diese können Sie (zumeist) nicht ändern.
Die ausgewählte Einstellung bedeutet, dass beim Öffnen einer Excel-Datei mit Makros die Makros grundsätzlich deaktiviert werden. Es sei denn, dass das Makro mit einer gültigen Signatur versehen ist. Dies ist bei selbsterstellen Makros jedoch die absolute Ausnahme.

Makros in Office-Dateien können aber trotzdem erstellt und genutzt werden. Die Dataport (oder ihr IT-Dienstleister) hat dazu vertrauenswürdige Speicherorte definiert. Makros in Office-Dateien, die an diesen Speicherorten gespeichert sind, werden ohne weitere Rückfrage ausgeführt.

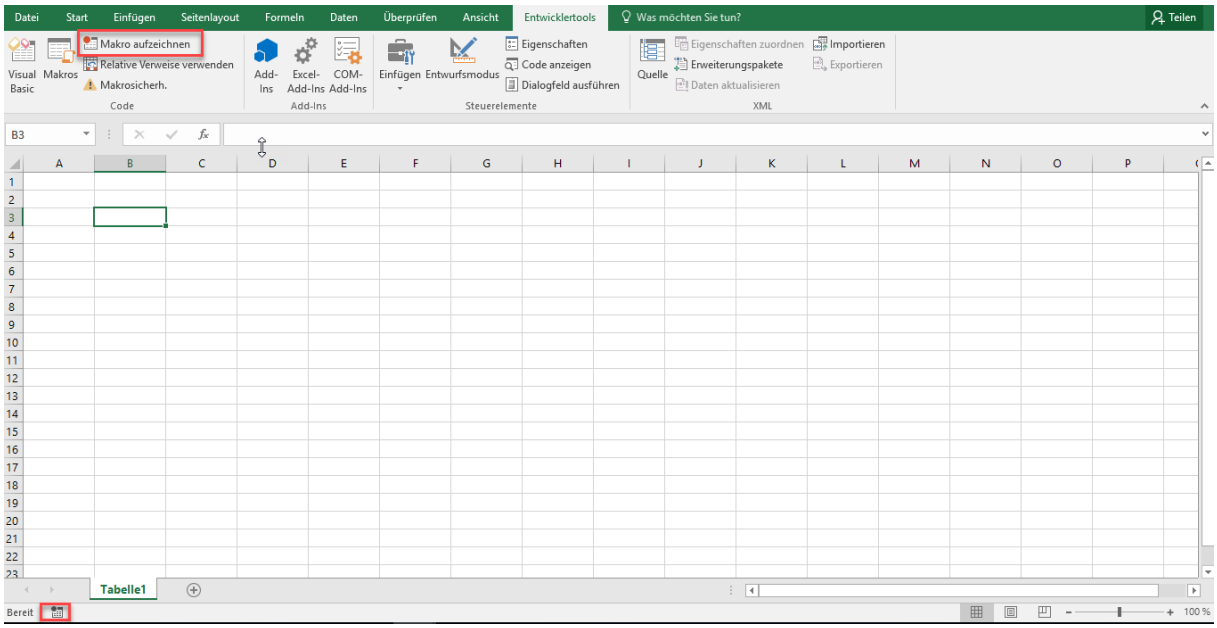
Aus Gründen der Sicherheit werden hier keine vertrauenswürdigen Speicherorte aufgelistet. Diese werden im zugehörigen Kurs ausführlich mit ihren Vor- und Nachteilen besprochen.

2. Makro aufzeichnen und speichern

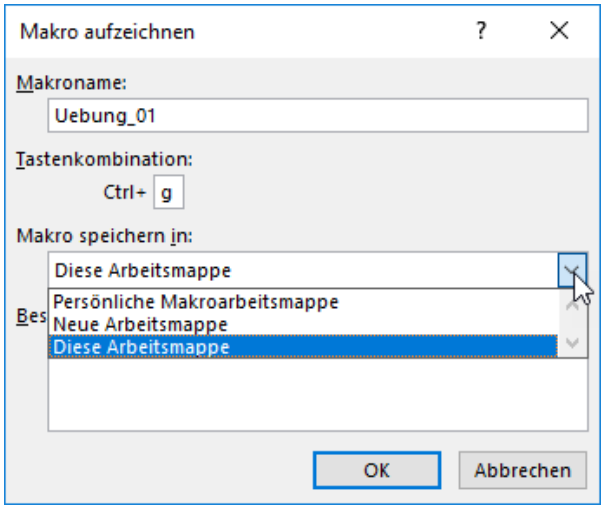
Wie einleitend bereits erwähnt können Makros auf zwei Wegen erstellt werden. Der deutlich einfachere Weg ist dabei gegenüber dem manuellen Schreiben das Aufzeichnen von Makros.

2.1 Makro aufzeichnen

Wenn Sie ein Makro aufzeichnen wollen, klicken Sie entweder in der Statuszeile auf das Symbol für Makro aufzeichnen oder Sie klicken auf die Registerkarte **Entwicklertools** und dort in der Gruppe **Code** auf die Schaltfläche **Makro aufzeichnen**.



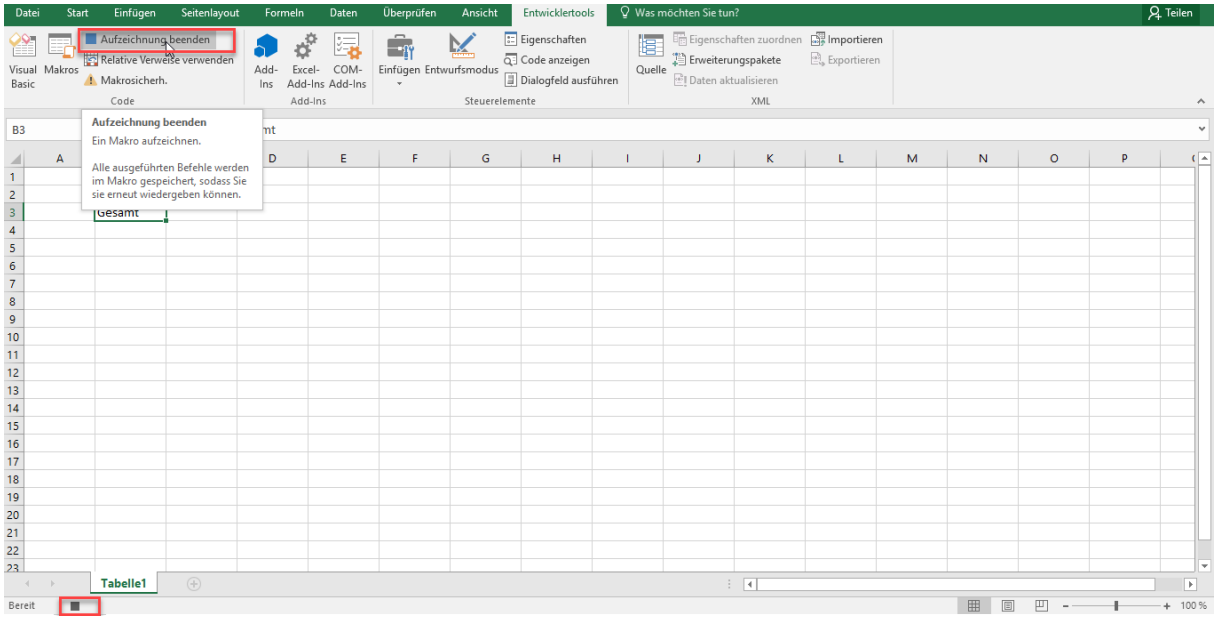
In diesem Beispiel soll das Wort Gesamt in die bereits ausgewählte Zelle B3 geschrieben werden, anschließend soll der Inhalt bestätigt werden ohne die Zelle B3 zu verlassen.



- Um die Aufzeichnung zu beginnen, klicken Sie auf den Befehl **Makro aufzeichnen**. Das Dialogfeld **Makro aufzeichnen** öffnet sich.
- Tragen Sie im Feld **Makroname** einen Namen für das Makro ein. Der Name darf keine Leerzeichen oder Sonderzeichen enthalten.
- Sie können im Feld **Tastenkombination** einen Tastenschlüssel festlegen, mit dem Sie das Makro aufrufen. Wenn Sie für dieses Makro den Buchstaben **g** eingeben, können Sie das Makro anschließend mit der Tastenkombination **Strg + g** ablaufen lassen. Bereits bestehende Tastenschlüssel werden dadurch allerdings überschrieben.
- Im Feld **Makro speichern in:** legen Sie fest, wo das Makro gespeichert werden soll. Dabei gilt:

  - **Persönliche Arbeitsmappe** ist eine Mappe, die automatisch beim Starten von Excel geöffnet wird. Somit stehen alle Makros, die hier abgespeichert sind, sofort zur Verfügung.
  - Speichern Sie die Makros in einer anderen Mappe (der gerade geöffneten oder einer neuen Arbeitsmappe) ab, stehen die Makros nur dann zur Verfügung, wenn Sie diese Mappe geöffnet haben.
- In das Feld **Beschreibung** kann optional eine kurze Beschreibung, was das Makro macht, eingegeben werden.
- Sobald Sie auf die Schaltfläche **OK** klicken, beginnt die Aufzeichnung des Makros. Grundsätzlich werden dabei alle Interaktionen mit der Tastatur oder der Maus vom Makrorekorder aufgezeichnet.
- Tippen Sie das Wort **Gesamt** in die bereits zuvor ausgewählte Zelle B3 ein und bestätigen Sie die Eingabe mit Kick auf das Symbol **Eingeben** in der Bearbeitungsleiste.

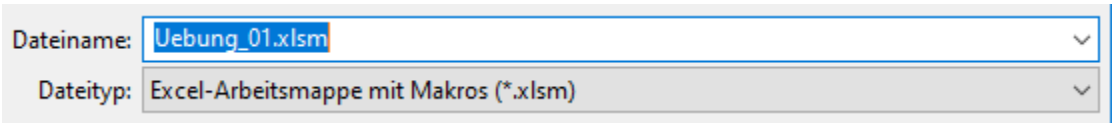
- Beenden Sie die Makroaufzeichnung. Klicken Sie auf den Befehl **Aufzeichnung beenden**.



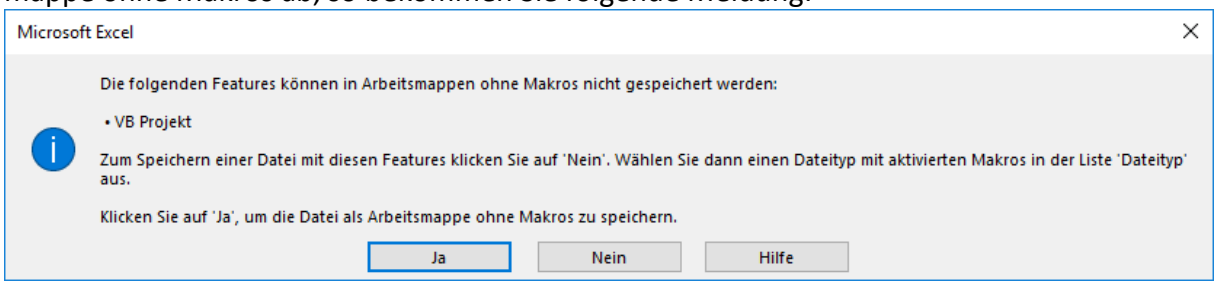
Sobald Sie auf das Symbol **Aufzeichnung beenden** geklickt haben, ist die Makroaufzeichnung abgeschlossen und Sie können das Makro mit dem Tastenschlüssel, in diesem Fall **Strg + g** ausführen lassen.

2.2 Speichern der Makroarbeitsmappe

Wenn Sie die Makros dauerhaft sichern wollen, müssen Sie die Mappe, die die Makros enthält, als Excel-Arbeitsmappe mit Makros abspeichern.



Enthält eine Excel-Arbeitsmappe Makros und Sie speichern diese als einfache Excel-Arbeitsmappe ohne Makros ab, so bekommen Sie folgende Meldung:



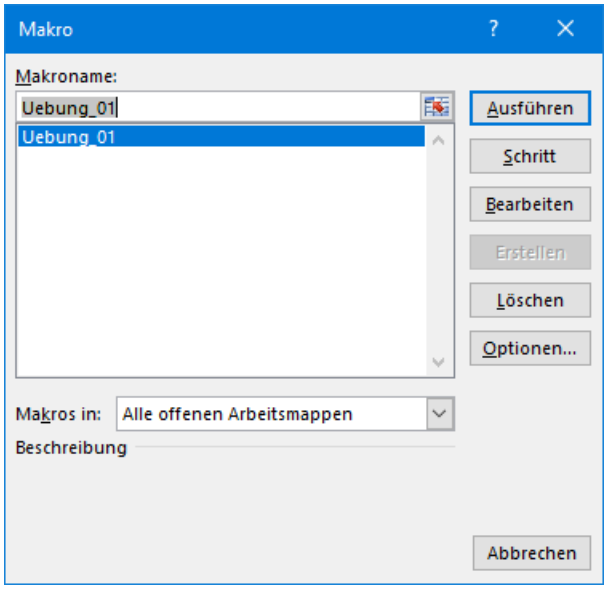
Klicken Sie auf die Schaltfläche **Nein**, wenn Sie die Makros speichern wollen. Damit gelangen Sie in das Dialogfeld **Speichern unter**. Hier können Sie nun die richtige Dateiendung auswählen.

Wichtig Bei Klick auf **Ja** gehen die Makros in der Arbeitsmappe **unwiederbringlich** verloren!

Wenn Sie bei der Aufzeichnung des Makros den Eintrag **Persönliche Makroarbeitsmappe** ausgewählt haben, wird das Makro in einer Datei namens **PERSONAL.XLSB** abgelegt. Diese Datei wird automatisch erzeugt, wenn Sie das erste Mal Makros in Ihrer persönlichen Makroarbeitsmappe speichern. Die Datei ist standardmäßig ausgeblendet und wird jedes Mal, wenn Sie Excel starten automatisch geöffnet. Makros, die hier gespeichert sind, stehen immer zur Verfügung. Makros, die in einer gesonderten Arbeitsmappe gespeichert sind, stehen nur dann zur Verfügung, wenn diese Mappe geöffnet ist.

3. Makro ablaufen lassen

Wie oben bereits erwähnt, können Sie die Makros über Tastenschlüssel, sofern Sie einen vergeben haben, aufrufen. Grundsätzlich können Makros aber immer über die Registerkarte **Entwicklertools** aufgerufen werden. Klicken Sie hierzu auf das Symbol **Makros** in der Gruppe **Code.** Es öffnet sich das Dialogfeld **Makro**. Das Dialogfeld können Sie auch mit der Tastenkombination **Alt + F8** aufrufen.



Wählen Sie das Makro aus und klicken Sie auf die Schaltfläche **Ausführen.**

Hinweis
Sofern die markierte Zelle immer noch B3 ist, werden Sie keine Änderung feststellen. Wählen Sie eine beliebige andere Zelle aus und führen Sie das Makro erneut aus.

Die Schaltfläche **Schritt** ruft den Visual Basic Editor (VBE) auf und führt die aufgezeichneten oder geschriebenen Befehl einzeln aus. Dies ist bei der Fehlersuche in Makros hilfreich.

Über die Schaltfläche **Bearbeiten** gelangen Sie in den Visual-Basic-Editor (VBE) und über die Schaltfläche **Löschen** können Sie das in der linken Liste markierte Makro löschen.

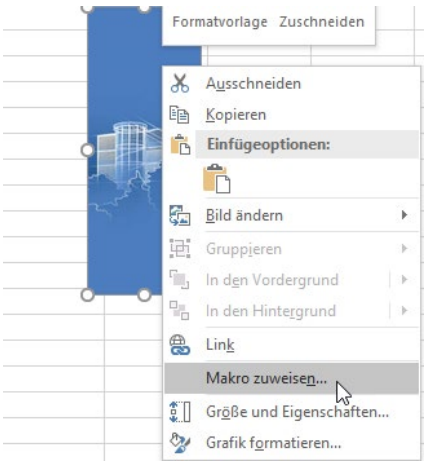
Falls Sie einem bereits vorhandenen Makro nachträglich einen Tastenschlüssel hinzufügen wollen, klicken Sie auf die Schaltfläche **Optionen** und ergänzen Sie den Tastenschlüssel.

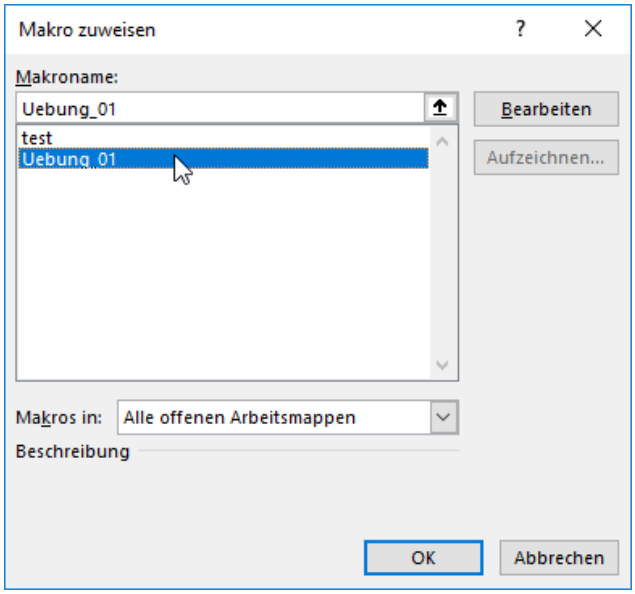
Sie haben aber noch weitere Möglichkeiten, Makros abzuspielen.

3.1 Makros Objekten zuweisen

Sie können Grafiken, Bildern oder Formularsteuerelementen Makros zuweisen. Dazu fügen Sie zunächst das Objekt ein. Dann klicken Sie mit der rechten Maustaste in das jeweilige Objekt.

Im Kontextmenü wählen Sie den Eintrag **Makro zuweisen**. Es öffnet sich das Dialogfeld **Makro zuweisen**.





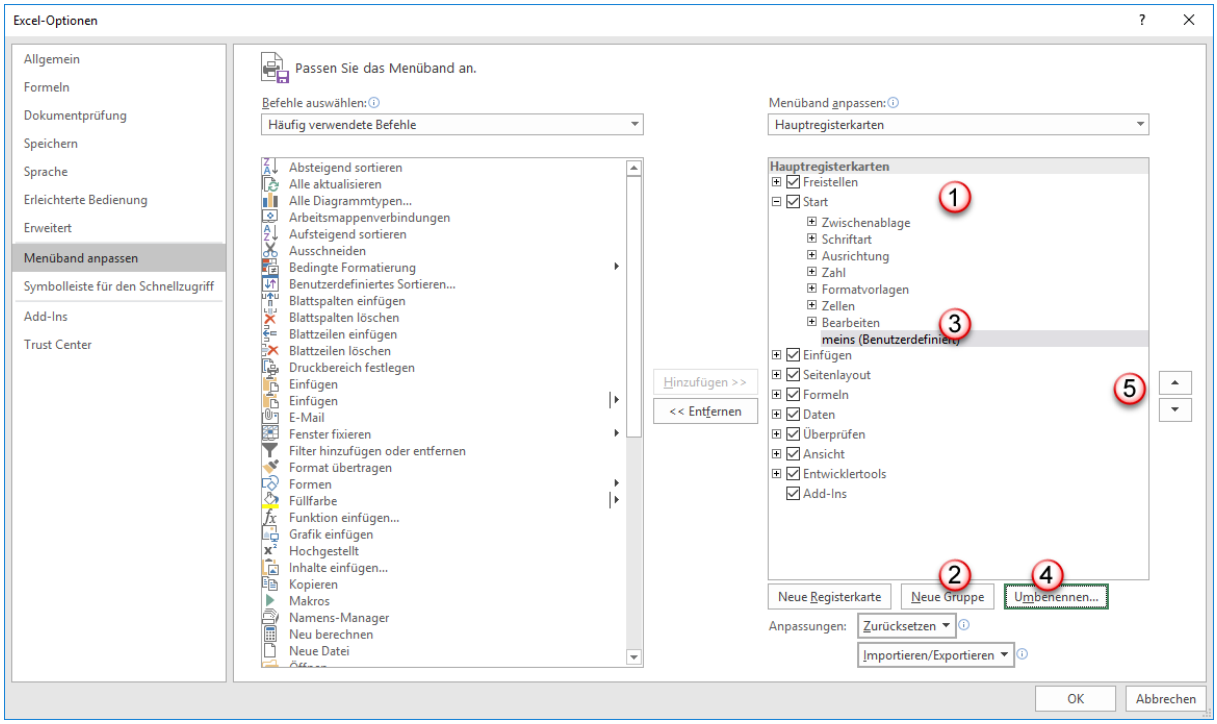
Markieren Sie das gewünschte Makro und klicken Sie auf die Schaltfläche **OK**.

Wenn Sie anschließend auf das Objekt klicken, wird das Makro ausgeführt.

Falls Sie das Makro selbst an dieser Stelle noch bearbeiten möchten, klicken Sie auf die Schaltfläche **Bearbeiten**.

3.2 Makros in Registerkarten einbinden

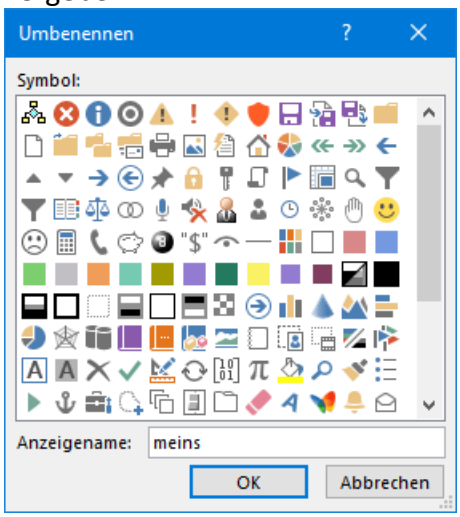
Eine weitere Möglichkeit, Makros zu starten, besteht darin, sie auf einer neuen oder bereits bestehenden Registerkarte zu platzieren. Wechseln Sie dazu wie unter 1.1 Menüband Entwicklertools beschrieben in das Dialogfeld **Excel-Optionen**. Rechts im Navigationsbaum klicken Sie auf den Eintrag **Menüband anpassen**. Alternativ können Sie aber auch mit der rechten Maustaste in ein vorhandenes Menüband hineinklicken und dort den Eintrag **Menüband anpassen** auswählen.



Das Dialogfeld **Excel-Optionen** öffnet sich. Hier gehen Sie wie folgt vor:

1. Wählen Sie die Registerkarte aus, auf der Sie eine neue Gruppe anlegen wollen.
2. Klicken Sie auf die Schaltfläche **Neue Gruppe**.
3. Die neue Gruppe wird am Ende der ausgewählten Registerkarte angelegt.

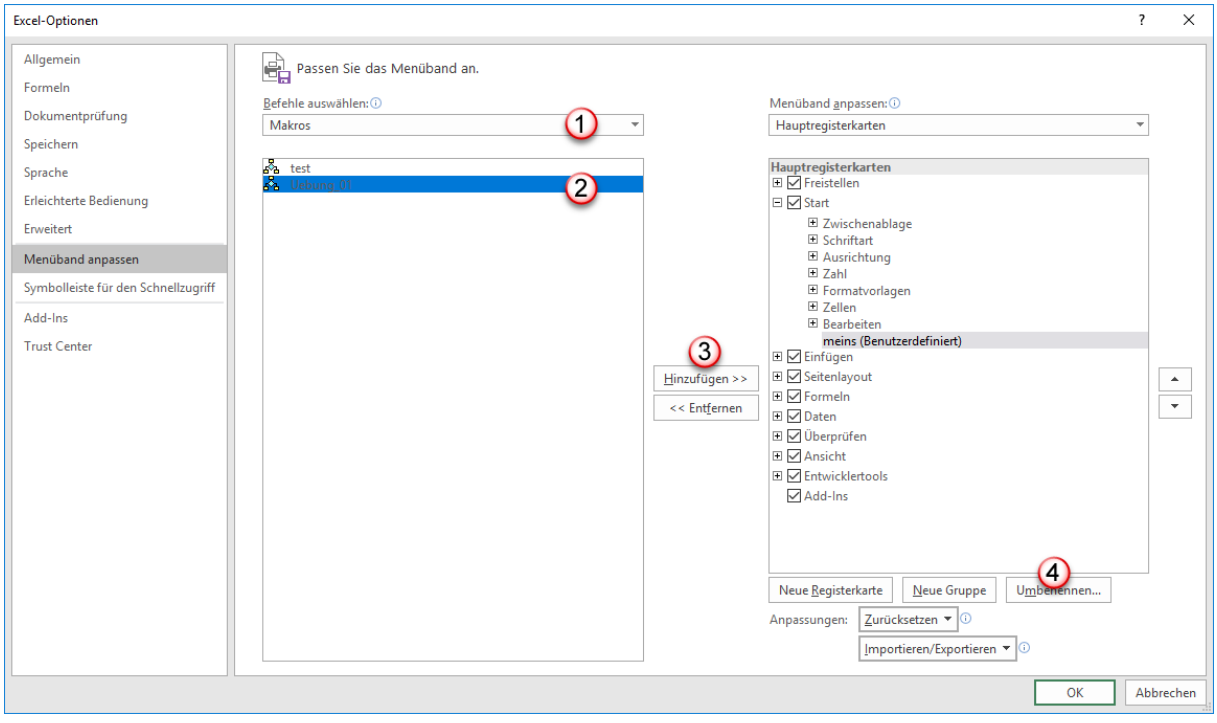
4. Klicken Sie nun auf die Schaltfläche **Umbenennen**. Das Dialogfeld **Umbenennen** öffnet sich. Ganz unten – unter den Symbolen – können Sie den neuen Namen, hier „meins“, vergeben.



   Der neue Name wird in der Liste der Gruppen angezeigt. Der Eintrag (benutzerdefiniert) bleibt erhalten.

5. Abschließend können Sie bei Bedarf die benutzerdefinierte Gruppe mit Hilfe der Pfeile noch auf der Registerkarte positionieren.

Wenn die neue Gruppe auf der Registerkarte erzeugt worden ist, können Sie nun die Makros dort positionieren.

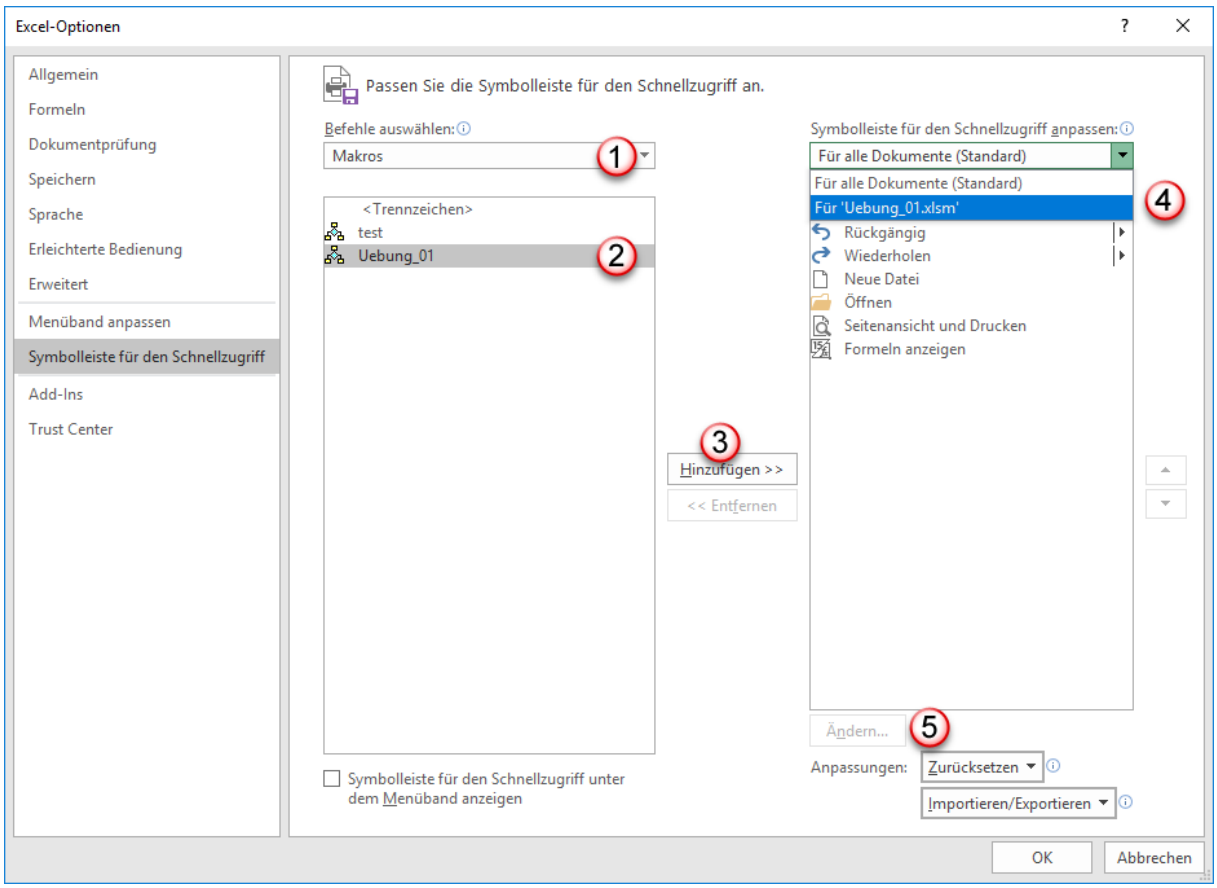


1. Klicken Sie zunächst in das Listenfeld **Befehle auswählen** und wählen den Eintrag **Makros**.
2. Alle vorhandenen Makros werden angezeigt. Wählen Sie das Makro aus, das Sie einfügen wollen.
3. Klicken Sie auf die Schaltfläche **Hinzufügen**. Das Makro wird in die, im rechten Fenster zuvor markierte Gruppe, hinzugefügt.

4. Abschließend können Sie noch ein anderes Symbol und eine andere Beschriftung für das Makro wählen. Klicken Sie hierzu auf die Schaltfläche **Umbenennen** und wählen Sie ein neues Symbol aus. Klicken Sie in das Feld **Anzeigename** und geben Sie einen Text ein, der die Schaltfläche für das Makro beschriftet.

3.3 Makros in die Symbolleiste für den Schnellzugriff einbinden

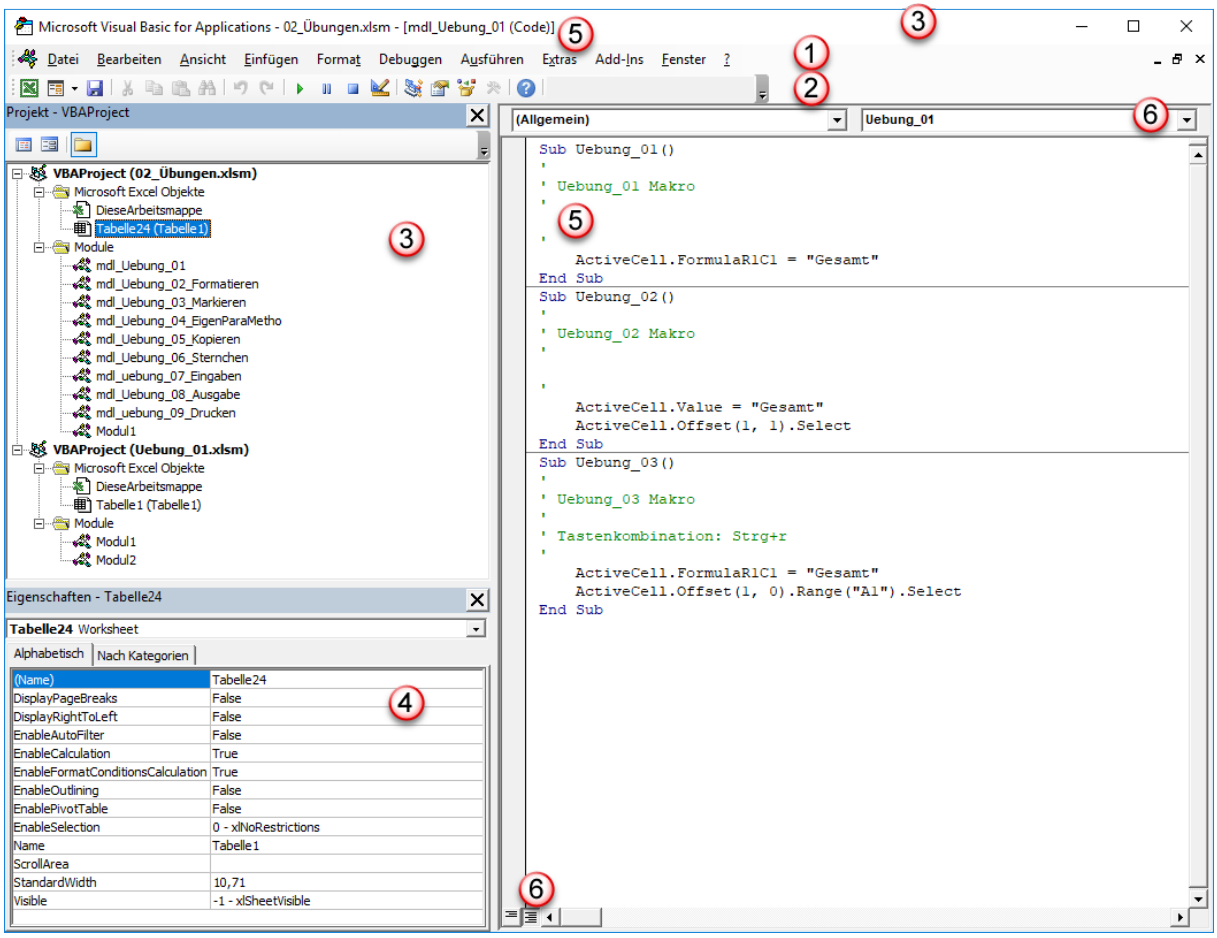
Sie können Makros auch in die Symbolleiste für den Schnellzugriff einbinden. Das Verfahren ist nahezu dasselbe wie unter 3.2 Makros in Registerkarten einbinden beschrieben. Außer, dass Sie im Dialogfeld **Excel-Optionen** den Eintrag **Symbolleiste für den Schnellzugriff auswählen**.



1. Wählen Sie aus dem Listenfeld **Befehle auswählen** den Eintrag **Makros** aus.
2. Aus der Liste der vorhandenen Makros wählen Sie das gewünschte Makro aus
3. Klicken Sie auf die Schaltfläche **Hinzufügen**, so dass das Makro in der rechten Fensterhälfte erscheint.
4. Im Listenfeld darüber können Sie auswählen, ob das Makro immer in der Symbolleiste angezeigt werden soll oder nur dann, wenn die Datei, die das Makro enthält, geöffnet ist.
5. Sobald das Makro in der Symbolleiste eingefügt ist, wird die Schaltfläche **Ändern** aktiv. Sie können nun dem Makro ein anderes Symbol und ein Quickinfo zuweisen.

4. Visual-Basic-Editor

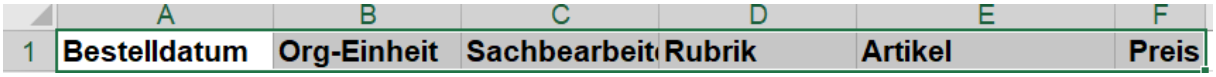
Um den aufgezeichneten Programmcode zu sehen und zu editieren, müssen Sie in den Visual Basic-Editor (VBE) wechseln. Dazu klicken Sie auf das Symbol für **Visual Basic** auf der Registerkarte **Entwicklertools** in der Gruppe **Code**. Sie können aber auch mit der Tastenkombination Alt + F11 zwischen Excel-Fenster und VBE-Fenster wechseln.



1. In der Menüleiste können Sie unter den einzelnen Menüpunkten Einstellungen für den Editor vornehmen.
2. Es gibt verschiedene Symbolleisten im VBE, die Sie sich ein- bzw. ausblenden lassen können. Menüpunkt **Ansicht > Symbolleisten**.
3. Der **Projekt-Explorer** zeigt alle geöffneten Dateien. Wenn Sie einen Doppelklick auf ein Modul machen, öffnet sich das zugehörige Codefenster mit den enthaltenen Makros.
4. Das **Eigenschaftenfenster** zeigt Eigenschaften des jeweils ausgewählten Objekts an. In diesem Fall ist im Projekt-Explorer das Objekt **Tabelle24(Tabelle1)** markiert.
5. Das **Code-Fenster** zeigt den aufgezeichneten oder geschriebenen Makro-Code an. Im Code-Fenster können Sie aufgezeichnete Makros editieren oder schreiben.
   In diesem Beispiel befinden Sie sich im Modul **mdl_Uebung_01**.
6. Im ausgewählten Modul gibt es mehrere Makros. Das, worin die Einfügemarke steht, ist aktiv. Wenn Sie nicht alle Makros auf einmal sehen wollen, können Sie ganz unten links auf das Symbol klicken und zwischen der Modulansicht mit allen Makros und der Prozeduransicht mit dem jeweils aktiven Makro wechseln. In der Prozeduransicht wechseln Sie über das rechte Listenfeld das aktive Makro (die Prozedur).

5. Aufgezeichnete Makros nachbearbeiten

Wenn Sie wissen wollen, wie sich die aufgenommenen Sequenzen im VBE darstellen, ordnen Sie das VBE-Fenster und das Excel zur Vereinfachung nebeneinander an. So können Sie die Aufzeichnung verfolgen. Es kommt außerdem darauf an, wie Sie aufzeichnen.
Im folgenden Beispiel soll eine Überschrift fett gestaltet werden. Dazu müssen Sie den Bereich A1:F1 zunächst markieren:



	A	B	C	D	E	F
1	**Bestelldatum**	**Org-Einheit**	**Sachbearbeit**	**Rubrik**	**Artikel**	**Preis**

Wenn Sie das Makro aufzeichnen und mit der Maus markieren, nimmt der Makrorecorder folgende Sequenz auf: **Range("A1:F1").Select**. Wenn Sie das Makro ausführen, wird immer der Bereich A1:F1 markiert. Wenn Sie einen unabhängigeren Code generieren wollen, verwenden Sie den Tastenschlüssel **Strg + Shift + Pfeiltaste rechts**. Damit zeichnen Sie folgenden Programmcode auf: **Range(Selection, Selection.End(xlToRight)).Select.** Dieser Programmcode ist nicht auf einen Zellbereich fixiert, sondern markiert eine Zeile ab der Cursorposition bis zum Ende des ausgefüllten Bereichs.

Aufzeichnung	Makrocode
Bereich mit **Maus** markiert	Range("A1:F1").Select
Strg + Shift + Pfeiltaste **rechts**	Range(Selection, Selection.End(xlTo**Right**)).Select
Strg + Shift + Pfeiltaste **links**	Range(Selection, Selection.End(xlTo**Left**)).Select
Strg + Shift + Pfeiltaste **oben**	Range(Selection, Selection.End(xl**Up**)).Select
Strg + Shift + Pfeiltaste **unten**	Range(Selection, Selection.End(xl**Down**)).Select
Strg + A (**Alles** markieren)	Range("A1:F6").Select hier wird trotz Tastenschlüssel ein fixer Bereich markiert
Schaltfläche **Aktuellen Bereich markieren**  , nicht über Tastenschlüssel aufrufbar und z.B. manuell der Symbolleiste für den Schnellzugriff hinzugefügt	Selection.**CurrentRegion**.Select wählt nur den aktuellen Bereich mit Daten aus

Makros, die aufgezeichnet werden, haben meistens mehr Programmzeilen als nötig.

Ein Beispiel: Es soll ein Bereich fett, in Arial 14 und vertikal zentriert werden. Alle Elemente der betroffenen Gruppen (Schriftart und Ausrichtung) werden in das Makro mit aufgenommen, obwohl sie nicht verändert (benötigt) werden. Das Makro hat dadurch mehr Programmzeilen und eventuell überschreiben diese Programmzeilen bereits vorgenommene Einstellungen bei erneuter Ausführung. Der Programmcode sollte daher immer um die nicht benötigten Code-Zeilen reduziert werden.

Im Beispiel unten sehen Sie die Aufzeichnung des Makros. Das Makro beginnt mit dem Schlüsselwort Sub gefolgt von dem Namen des Makros und einer sich öffnenden und schließenden Klammer **Sub Gestalten()** und endet mit dem Schlüsselwort **End Sub**.

Die grünen Texte in Hochkommata dazwischen sind Kommentare, die bei der Ausführung des Makros nicht berücksichtigt werden. Kommentare können die Arbeitsweise des Makros näher beschreiben. Bei umfangreicher Programmierung sind Kommentare hilfreich. Wenn Tastenkombinationen definiert wurden, werden diese hier ebenfalls durch den Makrorecorder als Kommentar ergänzt.

Die rot umrandeten Programmzeilen bewirken, dass der zuvor nach rechts erweiterte Bereich fett, vertikal zentriert und die Schrift Arial 14 ist. Die restlichen Einträge in diesen Abschnitten sind unnötig und sollten entfernt werden.

```
Sub Gestalten()
'
'weist einem markierten Bereich die Schriftart Arial, Schrifgrad 14,
'in fett und vertikal zentriert
'anschließend werden die Spalten in die optimale Breite gebracht.
'

'
    Range(Selection, Selection.End(xlToRight)).Select
    Selection.Font.Bold = True
    With Selection
        .VerticalAlignment = xlCenter
        .WrapText = False
        .Orientation = 0
        .AddIndent = False
        .IndentLevel = 0
        .ShrinkToFit = False
        .ReadingOrder = xlContext
        .MergeCells = False
    End With
    With Selection.Font
        .Name = "Arial"
        .Size = 14
        .Strikethrough = False
        .Superscript = False
        .Subscript = False
        .OutlineFont = False
        .Shadow = False
        .Underline = xlUnderlineStyleNone
        .ColorIndex = xlAutomatic
        .TintAndShade = 0
        .ThemeFont = xlThemeFontNone
    End With
    Cells.Select
    Cells.EntireColumn.AutoFit

End Sub
```

Sie können den Programmtext wie folgt kürzen:

```
Sub Gestalten()
'
'weist einem markierten Bereich die Schriftart Arial, Schrifgrad 14,
'in fett und vertikal zentriert
'anschließend werden die Spalten in die optimale Breite gebracht.
'

'
    Range(Selection, Selection.End(xlToRight)).Select

    Selection.VerticalAlignment = xlCenter

    With Selection.Font
        .Name = "Arial"
        .Size = 14
        .Bold = True

    End With
    Cells.EntireColumn.AutoFit
End Sub
```

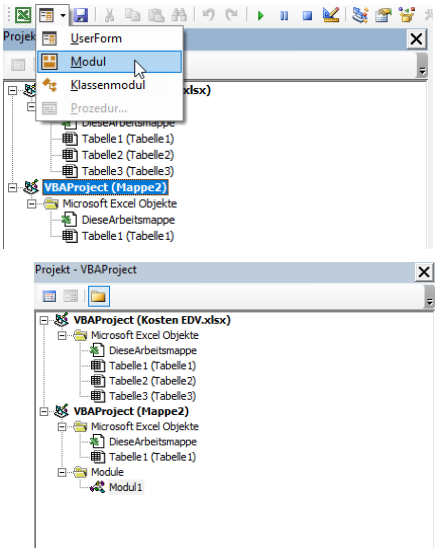
Im obigen Beispiel können Sie zudem erkennen, dass der Makrorekorder mit der **With**-Anweisung arbeitet. Eine With-Anweisung ermöglicht das Ausführen mehrerer Befehle hintereinander, die sich immer wieder auf dasselbe Element (Objekt) beziehen.

Die Anweisung **With Selection.Font** fasst alle ausgewählten Elemente (Selection), die zur Schrift (Font) gehören zu einem Objekt zusammen. Folgezeilen, die mit einem Punkt beginnen, beziehen sich immer auf dieses definierte Objekt. Selection.Font muss daher nicht mehrfach wiederholt werden. Der Code-Abschnitt endet zwingend mit der auflösenden Anweisung **End With**.

Alle Anweisungen, die zu Font gehören, können Sie zudem zusammenfassen und diejenigen, die Sie nicht benötigen, löschen.

6. Makros schreiben

Alternativ zur Aufzeichnung können Sie Makros auch im VBE schreiben. Dazu wechseln Sie in den VBE.



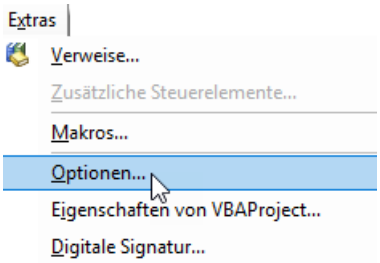
- Markieren Sie zunächst die **Arbeitsmappe** im Projekt-Explorer, in der das Makro gespeichert werden soll.
- Klicken Sie anschließend auf den Listenpfeil rechts neben dem Excel-Logo in der Symbolleiste.
- Wählen Sie den Eintrag **Modul**.
- Ein Ordner Module und das Modul selber, in dem Sie den Programmtext schreiben können, werden eingefügt.
- Sobald Sie ein Modul angelegt haben, können Sie damit beginnen, den Programmcode einzutippen.
- Zunächst beginnen Sie mit dem Schlüsselwort **SUB** (es leitet den Programmcode ein), dann der Name des Makros und die Klammern. Wenn Sie die Return-Taste drücken, erscheint automatisch das Schüsselwort **END SUB**.

Zwischen den Schlüsselworten **SUB** und **END SUB** steht der Makrocode.

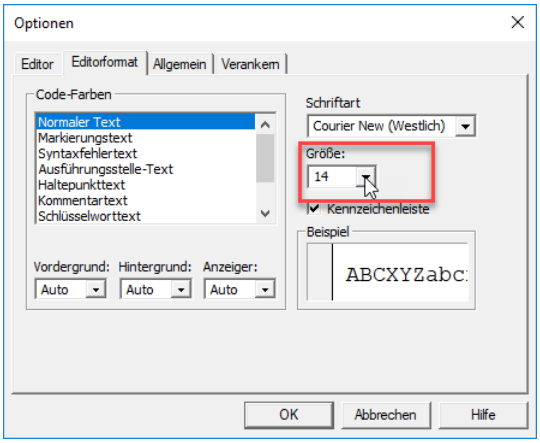
Um die einzelnen Programmteile, die zusammengehören, als solche zu kennzeichnen, empfiehlt es sich, diese einzurücken. Drücken Sie **Tabulator (Tab)-Taste**, um eine Einrückung vorzunehmen.

Im Editor ist standardmäßig die Schriftart Courier New als Schriftart mit fester Laufweite eingestellt. Behalten Sie dies bei, da proportionale Schriftarten die Lesbarkeit von Programmcode erschweren. Sie können aber die Schriftgröße verändern. Dabei gehen Sie wie folgt vor.

- Klicken Sie in der Menüsymbolleiste auf den Menüpunkt **Extras** und wählen dort den Eintrag **Optionen.**



- Es öffnet sich das Dialogfeld **Optionen**. Wählen Sie dort auf der Registerkarte **Editorformat** aus dem Listenfeld **Größe** die für Sie angenehme Schriftgröße aus.
- Bestätigen Sie Ihre Eingaben mit Klick auf die Schaltfläche **OK**.



Auf der Registerkarte Editor können Sie zudem die Schrittweite für eine Einrückung durch die Tab-Taste ändern, wenn Ihnen der Standard von 4 Zeichen zu groß ist.

6.1 Automatismen des Visual Basic-Editors

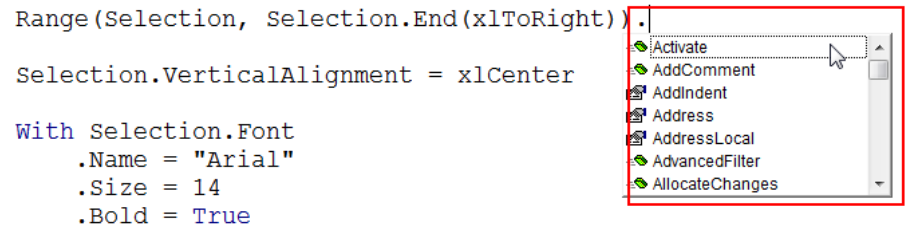
Wenn Sie die Anweisungen alle in kleinen Buchstaben schreiben, z. B. activeworkbook und dann weiterschreiben oder bestätigen, werden manche Buchstaben automatisch in Großbuchstaben umgesetzt: **A**ctive**W**orkbook. Das ist eine Hilfe, damit Sie erkennen können, ob die Anweisung richtig geschrieben wurde. Wenn keine Umsetzung in Großschrift erfolgt, könnte die Anweisung falsch geschrieben sein oder gar nicht existieren. Sie können auch nur die Anfangsbuchstaben schreiben, z. B. activewo. Mit der Tastenkombination **Strg + Leertaste** vervollständigt der VBE das Wort zu **A**ctive**W**orkbook. Das gilt jedoch nur, sofern der Befehl eindeutig ergänzt werden kann. Lassen Sie den letzten Buchstaben weg - hier das o von activewo - würde nicht nur das Wort **A**ctive**W**orkbook, sondern auch alle anderen Befehle in einer Liste aufgeführt werden, die mit dem Suchbegriff beginnen, das intelliSense. Wählen Sie einen Befehl aus der Liste aus, um diesen zu vervollständigen.

Warum die Auswahl eines Befehls mit Pfeiltasten und Punkt (.) eine große Hilfe ist, erfahren Sie im folgenden Abschnitt.

6.2 Ergänzungen von Befehlen mit intelliSense

Mit Strg + Leertaste haben Sie bereits gelernt, dass intelliSense Befehle vervollständigen kann. IntelliSense bietet aber darüber hinaus die vollständige Ergänzung von Befehlen mit zugehörigen Unterobjekten, Methoden und Eigenschaften. Mehr zu Objekten, Methoden und Eigenschaften erfahren Sie im nächsten Abschnitt 7 Strukturen in VBA.

Sobald Sie einen Punkt gesetzt haben wird nicht nur ein Befehl vervollständigt. Es wird zudem ein Listenfeld aufgeklappt, welches zu dem zuvor eingegebenen Befehl alle zugehörigen Unterobjekte, Methoden und Eigenschaften enthält. Sie können in diesem Listenfeld scrollen, um den gesuchten Eintrag zu finden. Wenn Sie einfach weitertippen, z. B. die ersten Buchstaben für selection, werden ihnen alle Einträge, die mit den eingetippten Buchstaben beginnen, angezeigt. Mit der **Tab**-Taste können Sie einen Eintrag übernehmen. Die Übernahme des Eintrags mit einem Punkt (.) bietet zusätzlich, sofern vorhanden, eine Liste der zugehörigen Unterobjekte, Methoden und Eigenschaften an.



7. Strukturen in VBA

Visual Basic for Applications (VBA) ist eine objektbezogene Programmiersprache. Es gibt vielfältige Excel Objekte, die durch Aktionen verändert werden können. Objekte können auch Parameter oder Eigenschaften haben, deren Wert verändert werden kann. Dabei folgt VBA einer einfachen Grammatik, wie bei einer Fremdsprache gelernt werden muss.

Stellen Sie sich vor, jemand sagt: Trink den Kaffee. Der Satz gewinnt Sinn durch das Substantiv und das Verb. Diese Anweisung würde in VBA heißen: Kaffee.Trink, also Objekt.**Methode**.

Wenn Sie in Excel ein neues Tabellenblatt hinzufügen wollen, programmieren Sie nicht: Füge ein neues Tabellenblatt ein, sondern: Tabellenblatt.Einfügen **Worksheets.Add**

Das Objekt Worksheets ist ein Auflistungsobjekt, da es mehrere Tabellenblätter in einer Mappe geben kann. Wenn Sie also ein Tabellenblatt per VBA einfügen wollen, schreiben Sie folgende Zeilen:

```
Sub BlattEinfügen()
    Worksheets.Add
End Sub
```

Standardmäßig wird nun vor dem jeweiligen aktiven Tabellenblatt ein neues Tabellenblatt eingefügt. In diesem Fall heißt das aktive Tabellenblatt **Grunddaten**. Wäre das aktive Tabellenblatt Tabelle3, würde vor **Tabelle3** das neue Tabellenblatt eingefügt werden.

Möchten Sie, dass jedes neue Blatt am Anfang eingefügt wird, dann drücken Sie nach der Methode Add die Leertaste. Es werden **Parameter** aufgelistet, die bei dieser Methode möglich sind. Die Zuweisung erfolgt durch die Kombination **:=**

```
Sub BlattEinfügen()
    Worksheets.Add before:=Worksheets(1)
End Sub
```

Sie können die Tabellenblätter im Objekt Worksheets über den Index ansprechen. Wenn Sie statt der 1 eine 3 eingeben, wird vor dem dritten Tabellenblatt ein neues Blatt eingefügt. Wie die Tabellenblätter in der Excel Mappe heißen ist egal. Tabelle1 ist in diesem Fall an vierter Stelle, also Worksheets(4).

Folgende Programmzeilen löschen das vierte Tabellenblatt, das in diesem Beispiel Tabelle1 heißt.

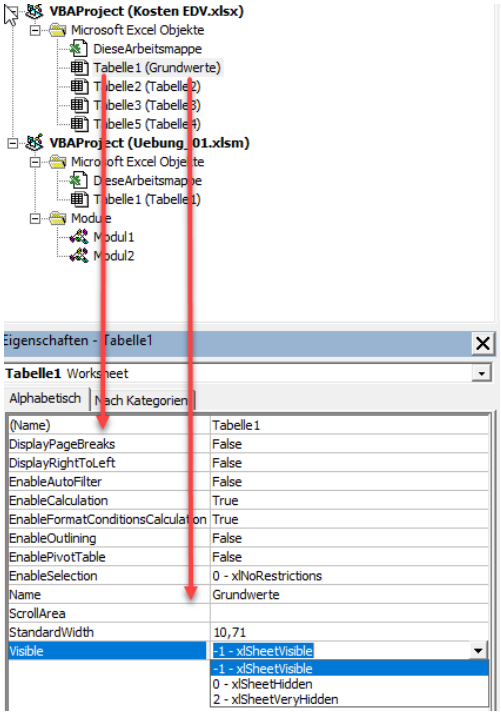
```
Sub BlattLoeschen()
    Worksheets(4).Delete
End Sub
```

Sie können die Tabellenblätter auch über deren Namen ansprechen. Der Name muss dabei in Anführungszeichen angegeben werden. Dann würden die Programmzeilen folgendermaßen lauten:

```
Sub BlattLoeschen()
    Worksheets("Tabelle1").Delete
End Sub
```

Bislang wurden **Methoden** und **Parameter** beschrieben. Kommen wir nun zu den **Eigenschaften**. Das Objekt Worksheets kann einen Namen haben. Dies ist eine Eigenschaft, der man einen Wert zuweisen kann. Die Zuweisung zu einer Eigenschaft erfolgt durch das Gleichheitszeichen (=).

```
Sub TabellenBlattUmbenennen()
    Worksheets(1).Name = "Grundwerte"
End Sub
```



Wie Sie sehen, wird unterschieden zwischen **(Name)** und **Name.**

Der erste Eintrag ist der sogenannte Codename. Diesen Namen können Sie ändern, in dem Sie im Eigenschaftenfenster in den ersten Eintrag klicken und diesen überschreiben.

Der zweite Eintrag ist derjenige, der für den Anwender sichtbar ist. Wenn Sie beim Programmieren diesen Namen in Ihrem Programmcode verwenden (siehe Makro BlattLoeschen), könnte es sein, dass der Blattname verändert wurde, deshalb das Blatt nicht gefunden wird. Das Makro gibt dann einen Fehler aus.

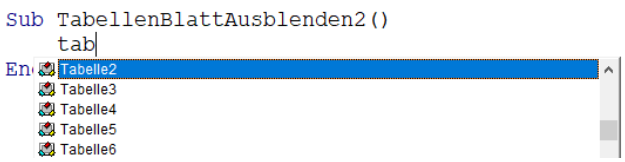
Beim Objekt Tabellenblatt haben Sie zudem die Möglichkeit im Eigenschaftenfenster einige Eigenschaften direkt zu verändern. Z. B. sehen Sie als letzten Eintrag die Eigenschaft **Visible**.

Wenn Sie rechts in das Feld klicken, können Sie die Auflistung sehen, die Ihnen zur Verfügung steht. Sie können aber auch wie beim Blattnamen die Eigenschaft durch ein kleines Programm ändern.

```
Sub TabellenBlattAusblenden()
    Worksheets(1).Visible = False
End Sub
```

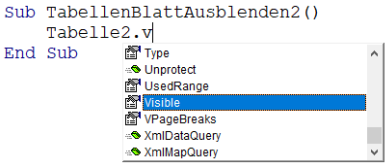
Wenn Sie die Tabellenblätter nicht - wie oben gezeigt - über den Blattnamen, so wie ihn der Anwender sieht, sondern den Namen nehmen wollen, wie er im VBE sichtbar ist, gehen Sie wie folgt vor:

Geben Sie zunächst die ersten Buchstaben des Tabellennamens ein. Drücken Sie dann die Tastenkombination Strg + Leertaste. Die vorhandenen Tabellen werden Ihnen aufgelistet. Wählen Sie dann mit den Pfeiltasten und der TAB-Taste die entsprechende Tabelle aus.

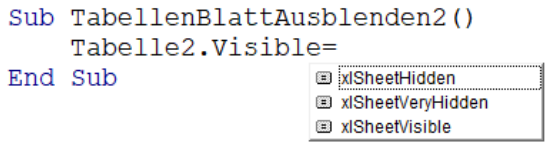


Sobald Sie einen Punkt gesetzt haben, folgen die möglichen Methoden und Eigenschaften zu diesem Objekt. Methoden sind die grünen sich bewegenden Würfel, Eigenschaften die Hand, die auf einen Eintrag in einer Liste deutet.

Wenn Sie nun ein V eintippen springt der Cursor sofort zur ersten Eintragung mit dem Buchstaben V.



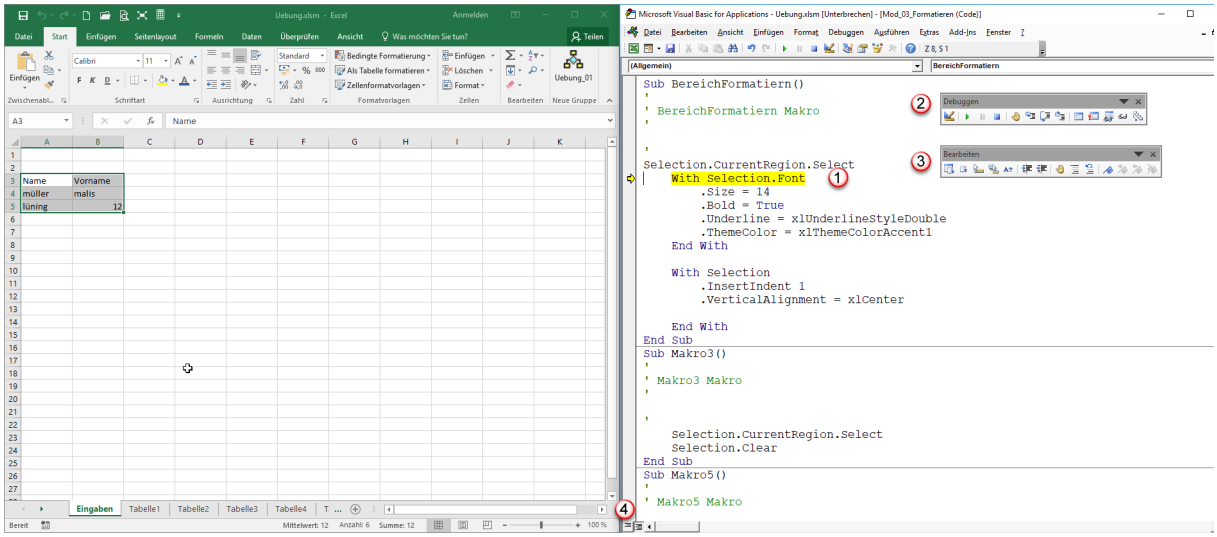
Übernehmen Sie den Eintrag mit der TAB-Taste. Sobald Sie das Gleichheitszeichen getippt haben, werden Ihnen die möglichen Eintragungen aufgelistet.



Neben den Einträgen **Sichtbar** und **Unsichtbar**, erscheint auch noch der Eintrag "besonders Unsichtbar". Das heißt, das Tabellenblatt kann vom Anwender nicht mehr eingeblendet werden, wenn mit der rechten Maustaste in das Blattregister geklickt wird.

8. Makros testen

Wenn Sie zwei Bildschirme haben, können Sie auf dem einen Bildschirm sich den VBE anzeigen lassen und auf dem anderen die Excel Anwendung. Bei einem Bildschirm können Sie diesen über die Tastenkombination Windows-Taste + Pfeiltaste links oder rechts teilen.



Zum Testen eines Makros empfiehlt es sich, dieses in Einzelschritten auszuführen.

1. Setzen Sie den Cursor in das Makro, das Sie testen wollen. Wenn Sie die **F8** drücken, wird das Makro im Einzelschritt ausgeführt. Der Schritt, der als nächstes ausgeführt wird, ist gelb hinterlegt. Sobald Sie die Taste **F8** nochmals drücken, wird die gelb markierte Zeile ausgeführt. Sofern der ausgeführte Befehl eine sichtbare Änderung im Excel-Fenster auslöst, können Sie den Schritt im Excel-Anwendungsfenster direkt nachvollziehen.
2. Wenn Sie die Ausführung des Makros abbrechen wollen, klicken Sie auf die Schaltfläche **Zurücksetzen** in der Symbolleiste **Debuggen**.
3. In der Symbolleiste **Debuggen** und der Symbolleiste **Bearbeiten** finden Sie ein **Hand**-Symbol. Mit diesem oder der Taste **F9** können im Programmtext bei einer Codezeile einen **Haltepunkt** setzen oder löschen. D. h. das Makro wird nur bis zu diesem Punkt ausgeführt und kann dann z.B. im Einzelschrittmodus (F8) weiter durchgetestet werden.
4. In einem Modul können sich mehrere Makros befinden, die durch waagerechte Striche voneinander getrennt sind. Zwischen der Ansicht aller Makros und nur einem ausgewählten Makro im Modul kann umgeschaltet werden. Wenn Sie nur ein Makro zurzeit sehen wollen, klicken Sie im Editor-Fenster unten links auf die linke Schaltfläche **Prozeduransicht**. Wenn Sie auf die rechte Schaltfläche der **vollständigen Modulansicht** klicken, werden wieder alle im Modul enthaltenen Makros angezeigt.

9. Eingaben in Zellen machen und wieder am Bildschirm ausgeben

Für die Ein- und Ausgabe von Werten stellt VBA verschiedene Wege bereit. In den folgenden Abschnitten wird die Eingabe von festen Werten über Programmcode sowie die variable Ein- und Ausgabe von Werten über zwei Funktionen beschrieben.

9.1 Feste Eingaben über Code

Sie können per Makro in eine Zelle etwas eingeben. Wenn Sie ein Makro dazu aufzeichnen, sieht er wie folgt aus:

```
Sub Eingaben()
    Range("C1").Select
    ActiveCell.FormulaR1C1 = "33"
    Range("C2").Select
    ActiveCell.FormulaR1C1 = "Hallo"
    Range("C3").Select
    ActiveCell.FormulaR1C1 = "11/20/2015"
    Range("C4").Select
    Range("N2").Select
    Range("I9").Select
End Sub
```

Dieses Makro wählt Zelle C1 aus und schreibt dort die Zahl 33 hinein. Dann wählt es die Zelle C2 aus und schreibt dort das Wort Hallo hinein usw. Wenn Sie das Makro ablaufen lassen, schreibt es immer wieder denselben Inhalt in dieselben Zellen.

Wenn Sie statt der konkreten Zelle, die jeweils aktive Zelle mit Inhalt versehen, sieht das Makro wie folgt aus:

```
Sub Eingaben()
    ActiveCell.Value = "33"
    ActiveCell.Value = "Hallo"
    ActiveCell.Value = "11/20/2015"
End Sub
```

Wenn Sie dieses Makro im Einzelschritt Modus ablaufen lassen, wird in die jeweils aktuelle ausgewählte Zelle nacheinander der Inhalt eingefügt. Der vorherige Inhalt wird also überschrieben, da die aktive Zelle sich nicht ändert. Wenn Sie wollen, dass die erste Zelle befüllt wird, dann die nächste müssen Sie mit der Eigenschaft **Offset** arbeiten. Der erste Wert in der Klammer gibt dabei den Versatz in Zeilen nach unten an, der zweite den Versatz in Spalten nach rechts.

```
Sub Eingaben()
    ActiveCell.Value = "33"
    ActiveCell.Offset(1, 0).Value = "Hallo"
    ActiveCell.Offset(2, 0).Value = "11/20/2015"
End Sub
```

Sie haben vielleicht bemerkt, dass der Zellzeiger in der ersten aktiven Zelle verbleibt. Wenn Sie wollen, dass er mitwandert, müssen Sie die Programmzeilen wie folgt ändern:

```
Sub Eingaben()
    ActiveCell.Value = "33"
    ActiveCell.Offset(1, 0).Select
    ActiveCell.Value = "Hallo"
    ActiveCell.Offset(1, 0).Select
    ActiveCell.Value = "11/20/2015"
End Sub
```

Hinweis Die Eingabe von Werten in Zellen ist ohne Select-Anweisung möglich. Select-Anweisungen sollten auch vermieden werden, da Sie das Ausführen von Makro-Code unnötig langsam machen. Springen Sie ggf. erst ganz zum Schluss Ihres Makros in die Zelle, die dann aktiv sein soll.

Zwar ist jetzt der Eingabebereich variabler geworden, aber es werden immer wieder dieselben Inhalte eingefüllt.

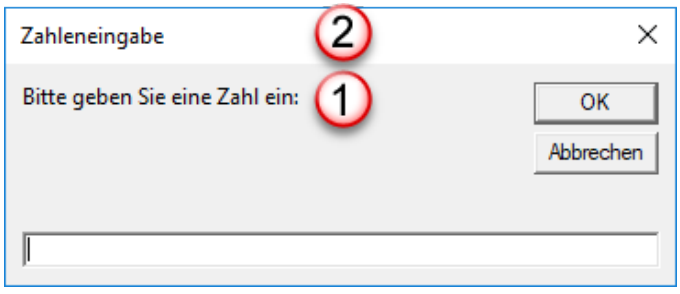
9.2 Variable Eingaben über die Funktion InputBox

Um variable Inhalte in Zellen einzugeben, können Sie eine Inputbox benutzen. Der Code kann dazu so aussehen:

```
Sub Eingaben()
    ActiveCell.Value = InputBox("Bitte geben Sie eine Zahl ein: ", "Zahleneingabe")
    ActiveCell.Offset(1, 0).Select
    ActiveCell.Value = InputBox("Bitte geben Sie einen Text ein ", "Texteingabe")
    ActiveCell.Offset(1, 0).Select
    ActiveCell.Value = InputBox("Bitte geben Sie ein Datum ein", "Datumseingabe")
End Sub
```

```
InputBox(
InputBox(Prompt, [Title], [Default], [XPos], [YPos], [HelpFile], [Context]) As String
```

1 2



Das Argument, das gerade bearbeitet wird, wird fett dargestellt. Dies ist das Argument **Prompt**. Das Argument **Prompt** ist ohne eckige Klammern angegeben, d. h. es ist ein Pflichtargument und muss eingegeben werden. **Prompt** ist der Text in der Box.

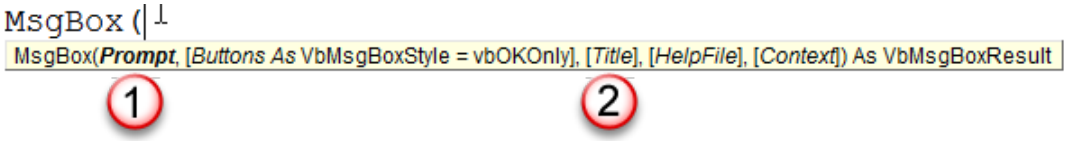
Alle übrigen Argumente sind in eckigen Klammern **[ ]** angegeben. Sie sind somit optional und können weggelassen werden.

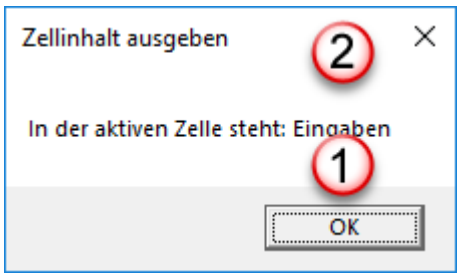
Das Argument **Title** fügt die Beschriftung der InputBox ein. Fehlt das Argument steht dort Microsoft Excel.

Das Argument **Default** gibt einen Standardwert an, der beim Aufrufen im Eingabefeld angezeigt wird. Bei der Eingabe des Datums könnte als Default daher „TT.MM.JJJJ“ angegeben werden, um aufzuzeigen, in welchem Format das Datum eingegeben werden soll.

9.3 Ausgaben am Bildschirm über die Funktion MsgBox

Mit der **InputBox** können Sie einen Dialog mit dem Anwender führen. Dabei geben Sie Daten ein. Die **MessageBox** zeigt Ihnen am Bildschirm Daten, die z. B. in den Zellen stehen, an.





Ähnlich wie bei der **InputBox** muss das erste Argument, der Prompt ausgefüllt werden. In diesem Fall der Text "In der aktiven Zelle steht: " gefolgt von dem Wert der aktiven Zelle.

Das Argument für den Parameter **Buttons** der anzuzeigenden Schaltflächen der MessageBox wird leer gelassen. Die MessageBox zeigt dann den Standardwert für den optionalen Parameter an. Im Falle des Parameters Buttons ist das nur die Schaltfläche OK.

Der **Titel** ist ebenfalls in eckigen Klammern dargestellt und muss als optionaler Parameter nicht ausgefüllt werden. Wenn Sie als Titel nicht angeben, steht dort "Microsoft Excel".

```
Sub ZellInhalt()
    MsgBox "In der aktiven Zelle steht: " & ActiveCell.Value, , "Zellinhalt ausgeben"
End Sub
```

Möchten Sie zusätzlich die Adresse der aktuellen Zelle, lauten die Programmzeilen wie folgt:

```
Sub ZellInhalt2()
    MsgBox "In der aktiven Zelle steht: " & ActiveCell.Value _
    & vbLf & "Die aktive Zelle ist " & ActiveCell.Address(False, False), , "Zellinhalt ausgeben"
End Sub
```

Das **&** ist ein Verkettungszeichen und verknüpft die einzelnen Teile mit einander. Das **vbLF** ist ein Zeilenvorschub und bewirkt, dass in der nächsten Zeile weitergeschrieben wird.

Falls eine Programmzeile zu lang und unübersichtlich wird, können Sie eine Leertaste und den Unterstrich _ eingeben. Anschließend drücken Sie die Returntaste, so dass Sie in der nächsten Zeile den Code weiterschreiben können. Der Unterstrich kennzeichnet, dass der Code in der nächsten Zeile fortgesetzt wird.

10. Variablen definieren

Sie haben zwar geschrieben: "Geben Sie eine Zahl ein", trotzdem können Sie alle möglichen Daten eingeben. Um tatsächlich nur einen Text, eine Zahl oder ein Datum eingeben zu können, müssen Sie Variablen definieren. Variablen sind Speicherplätze, die unterschiedliche Werte annehmen können und einen unterschiedlichen Platzbedarf haben.

Für das obige Beispiel brauchen Sie Variablen vom Typ **Zahl**, vom Typ **Text** und vom Typ **Datum.** Sie sollten zur Übersichtlichkeit diese Variablen deklarieren, bevor der Programmtext kommt. Deklarieren bedeutet, dass die Variable dem Programmcode bekannt gemacht wird. Dies geschieht mit der Anweisung **Dim**.

```
Sub EingabenMitVariablen()
Dim intZahl As Integer
Dim strText As String
Dim dtmDatum As Date
```

Es gibt Regeln für Variablennamen, z. B. keine Leerzeichen oder Sonderzeichen usw. Diese Konventionen **müssen** eingehalten werden, sonst gibt es Fehlermeldungen. Die Präfixe davor **können** eingehalten werden. Sie sollen später beim Programmtext darauf hinweisen, um welchen Variablentyp es sich handelt, z. B. **intZahl**. Das Wort **Zahl** ist beliebig, das Präfix **int** bedeutet, dass es sich um eine Variable vom Typ **Integer** (ganze Zahl) handelt.

Wenn Sie das Makro ablaufen lassen, weisen Sie den Variablen per **InputBox** einen Wert zu. Die Werte werden nacheinander in die Variablen eingelesenen. Die eingelesenen Werte geben Sie dann mit den nächsten Programmzeilen in die Zellen aus.:

```
Sub EingabenMitVariablen()
Dim intZahl As Integer
Dim strText As String
Dim dtmDatum As Date
    intZahl = InputBox("Bitte geben Sie eine Zahl ein: ", "Zahleneingabe")
    strText = InputBox("Bitte geben Sie einen Text ein ", "Texteingabe")
    dtmDatum = InputBox("Bitte geben Sie ein Datum ein", "Datumseingabe")

    ActiveCell.Value = intZahl
    ActiveCell.Offset(1, 0) = strText
    ActiveCell.Offset(2, 0) = dtmDatum
End Sub
```

11. Fehlerbehandlung mit der On Error-Anweisung

Jetzt haben Sie zwar festgelegt, dass bei Zahlen nur Zahlen und bei Text nur Text eingeben werden darf. Was aber, wenn der Anwender in die Zahlenbox einen Text eingibt? Das Makro produziert dann einen Laufzeitfehler, das Makro bricht ab und es erscheint eine unschöne Fehlermeldung. Mit der Anweisung **On Error** kann dies abgefangen werden.

D. h. Sie setzen bevor die Programmzeilen anfangen die Anweisung: falls ein Fehler auftritt, gehe zu …**On Error Goto Fehler** (das Wort **Fehler** ist nur der Name der Sprungmarke. Sie können genauso gut **SpringHierHin** schreiben)

Falls ein Fehler produziert wird, springt das Programm zur Sprungmarke. Diese wird nach den Programmzeilen wie folgt platziert:

Fehler: oder eben **SpringHierHin:** wichtig ist der Doppelpunkt.

Da der Programmcode für die Fehlerbehandlung zumeist nach dem allgemeinen Programmcode steht, muss dann verhindert werden, dass dieser zusätzlich und ungewollt ausgeführt wird. Deshalb muss am Ende des allgemeinen Programmcodes ausgewiesen werden, dass dieser zu Ende ist und das Makro an sich beendet werden kann. Dies erfolgt mit der Anweisung **Exit Sub**.

Nun läuft das Makro korrekt, wenn alle Angaben richtig sind, wenn nicht wird es ohne Eingaben beendet. Es wäre nun wünschenswert, dass der Anwender weiß, was falsch war. Dazu kann eine MsgBox ausgegeben werden. In dieser teilen Sie mit, was der Fehler war und was zu tun ist.

```
Sub EingabenMitVariablen()
Dim intZahl As Integer
Dim strText As String
Dim dtmDatum As Date

On Error GoTo fehler

    intZahl = InputBox("Bitte geben Sie eine Zahl ein: ", "Zahleneingabe")
    strText = InputBox("Bitte geben Sie einen Text ein ", "Texteingabe")
    dtmDatum = InputBox("Bitte geben Sie ein Datum ein", "Datumseingabe")
    ActiveCell.Value = intZahl
    ActiveCell.Offset(1, 0) = strText
    ActiveCell.Offset(2, 0) = dtmDatum
    Exit Sub
fehler:
    MsgBox "Sie haben einen falschen Datentyp eingegeben" _
    & vbLf & "Fangen Sie von vorn an"

End Sub
```

12. Schleifen

Im obigen Beispiel muss der Anwender immer wieder das Makro aufrufen, um es ablaufen zu lassen. Besser wäre, das Makro läuft solange bis der Anwender das Makro beendet. Dies können Sie mit einer Schleife programmieren. Es gibt **bedingungsgesteuerte** Schleifen, die werden so oft durchlaufen bis die Bedingung erfüllt ist. Bezogen auf das Beispiel mit der falschen Dateneingabe bedeutet das, das Makro läuft solange bis alle Datentypen richtig eingegeben werden.

Zudem gibt es **zählergesteuerte** Schleifen, die wiederholen die Programmzeilen so oft, wie es vorgegeben wird. Z. B. sollen die Programmschleifen 10-mal durchlaufen werden.

An dieser Stelle wird zunächst die zählergesteuerte **For … Next-Schleife** beschrieben.

12.1 Die zählergesteuerte For … Next-Schleife

Der folgende Programmcode füllt die aktuelle Zelle mit dem Wort Hallo und setzt den Cursor eine Zelle darunter.

```
ActiveCell.Value = "Hallo"
ActiveCell.Offset(1, 0).Select
```

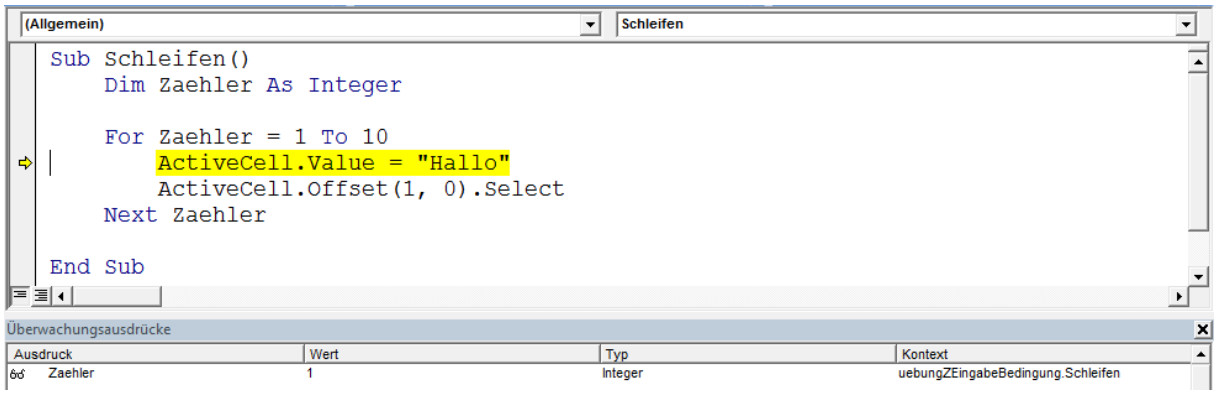
Wenn Sie den ganzen Programmtext 10-mal durchlaufen lassen wollen, programmieren Sie eine **For … Next-Schleife**. Sie definieren die Variable Zaehler und setzen den Startwert auf 1. Die Programmzeile lautet also: von 1 bis 10 schreibe das Wort Hallo. Die Programmzeile next Zaehler erhöht den Wert der Variablen um 1.

```
Sub Schleifen()
    Dim Zaehler As Integer

    For Zaehler = 1 To 10
        ActiveCell.Value = "Hallo"
        ActiveCell.Offset(1, 0).Select
    Next Zaehler

End Sub
```

Wenn Sie die Abarbeitung der einzelnen Schritte verfolgen und den jeweiligen Wert der Variable sehen wollen, klicken Sie im VBE auf den Menüpunkt **Ansicht** und dort auf den Eintrag **Überwachungsfenster**. Markieren Sie die Variable **Zaehler** und ziehen Sie die Variable in das Überwachungsfenster. Wenn Sie nun das Makro im Einzelschrittmodus ablaufen lassen, wird Ihnen der jeweilige Wert, der in der Variablen gespeichert ist, angezeigt.



12.2 Die bedingungsgesteuerte Do Until-Schleife

Wenn Sie das Fortführen des vorstehenden Beispiels der For…Next-Schleife an eine Bedingung knüpfen wollen, können Sie das beispielsweise mit einer **DO Until-Schleife** tun.

Zunächst wird eine **Variable** mit dem Namen Bedingung definiert. Die Eingabe soll dann solange fortgesetzt werden, bis die Variable einen bestimmten Wert angenommen hat. In unserem Falle bis die Variable den Wert 7 (steht für Nein) angenommen hat.

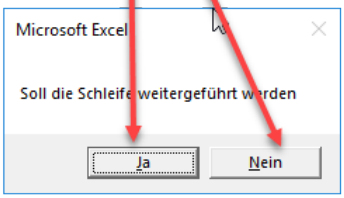
Die Werte für die Variable mit dem Namen Bedingung werden durch eine MsgBox erzeugt. MsgBoxen können nicht nur Nachrichten ausgeben. Sie liefern auch zurück, was für eine Schaltfläche durch den oder die Benutzenden angeklickt worden ist. An dieser Stelle spielt der eingangs bei der MessageBox beschriebene Parameter **Buttons** eine Rolle. Denn über diesen wird gesteuert, welche Schaltflächen in der MessageBox angezeigt werden.

Zur Erinnerung:

Bedingung = MsgBox("....." , vbYesNo, [Title])

Das entscheidende ist die Visual Basic-Konstante **vbYesNo**. Sie gibt an, dass in der MessageBox die Schaltfläche **Ja** und **Nein** angezeigt werden.

```
Sub SchleifeMitBed()
Dim Bedingung As Integer
Do Until Bedingung = 7
    ActiveCell.Value = "Hallo"
    ActiveCell.Offset(1, 0).Select
    Bedingung = MsgBox("Soll die Schleife weitergeführt werden" & vbLf _
    , vbYesNo)
Loop
End Sub
```



Klickt nun der Anwender auf das Ja, dann wird durch die MessageBox die Zahl 6 zurückgegeben. Damit steht in der Variablen Bedingung die Zahl 6. Das Programm springt durch die Anweisung Loop wieder auf den Schleifenanfang, um zu prüfen, ob die Zahl eine 7 ist. Wenn nicht wird Schleife erneut durchlaufen. Klickt der Anwender auf Nein, dann wird durch die MessageBox die Zahl 7 zurückgegeben. Die Bedingung am Schleifenanfang ist somit erfüllt und die Schleife wird nicht erneut durchlaufen.

Den jeweiligen Wert der Variablen können Sie durch das Überwachungsfenster nachverfolgen.

Lernmaterial, Beratung und Kontakt

Auf der Internetseite

http://www.afz.bremen.de/lernen

stellt das AFZ Ihnen Kursunterlagen zu den IT-Kursen in elektronischer Form zur Verfügung. Diese werden regelmäßig aktualisiert und an neue Programmversionen angepasst. Das bietet Ihnen die Möglichkeit, jederzeit Kursthemen zu wiederholen und Ihre Kenntnisse zu aktualisieren.

Bei unseren Kursunterlagen handelt es sich um PDF-Dokumente, die Sie am Bildschirm lesen können. Die Dateien sind barrierefrei und können nach Stichworten durchsucht (strg + F) werden. Das Inhaltsverzeichnis und Links sind dynamisch verwendbar. Sie können die Dateien auf Ihrem Rechner speichern und bei Bedarf ausdrucken.

Auskünfte und Beratung

Sollten Sie als Beschäftigte der Freien Hansestadt Bremen bei Ihrer Arbeit auf Probleme stoßen, die beim Einsatz Ihrer Softwareausstattung auftreten (Probleme mit Word-Dokumenten, Excel-Tabellen etc.), können Sie sich mit Ihren Fragen, Problemstellungen oder Fehlermeldungen telefonisch oder per E-Mail an uns wenden:

it-fortbildung@afz.bremen.de **Tel. 361-16 999**

Beschreiben Sie Ihre Frage bzw. die Fehlersituation und Ihre bisherige Vorgehensweise und fügen Sie die Dateien im Original-Dateiformat als Anlage bei. Wir beantworten Ihre Fragen so schnell wie möglich, in jedem Fall melden wir uns innerhalb weniger Tage bei Ihnen.

Kontakt

Wir sind sehr an Ihren Anregungen und Verbesserungsvorschlägen zu unseren Kursangeboten, zu den Lernmaterialien und Ihrer Meinung zu unseren E-Learning-Kursen interessiert. Bitte nutzen Sie das

Kontaktformular

auf unserer Internetseite oder senden Sie eine Nachricht an **it-fortbildung@afz.bremen.de**.

Version vom November 2021

Impressum

Redaktion und Koordination

Referat 20 – Informationstechnologie – Qualifizierung und Beratung
Aus- und Fortbildungszentrum
Doventorscontrescarpe 172C

28195 Bremen

Telefon: +49 421 361-16999
E-Mail: it-fortbildung@afz.bremen.de

Herausgeber

Aus- und Fortbildungszentrum
für den bremischen öffentlichen Dienst
Doventorscontrescarpe 172C

28195 Bremen