Workshop 1: VBA-Programmierung mit MS Excel

1 Der Makro-Recorder

Der Makrorecorder von Excel bietet die Möglichkeit, automatisch Programmcode aufzeichnen zu lassen. Im Folgenden wird gezeigt, wie der Makrorecorder hilfreich eingesetzt werden kann. Im Rahmen des dazugehörigen Excel-Workshops erfahren Sie

- was Makros sind,
- wie Makros aufgezeichnet und ausgeführt werden,
- wie und wo Makros gespeichert werden,
- wie Makros schneller gestartet werden können.

1.1 Grundlagen

Bei der Erstellung von Makro werden die einzelnen Arbeitsschritte, die Sie ausführen, nacheinander aufgezeichnet. Solche Mitschreibmakros können äußerst hilfreich sein. Beispielsweise kann man sich ansehen, wie die ausgeführten Befehle in der Programmiersprache *Visual Basic* for Applications (VBA) aussehen. Auch wenn man schon programmieren kann, ist diese Vorgehensweise oft nützlich, wenn man einen Befehl vergessen hat. Man kann dann schnell den Befehl mitschreiben lassen, anstatt in der Online-Hilfe nach den entsprechenden Befehlen zu suchen.

1.2 Grenzen

Viele Funktionalitäten lassen sich als Mitschreibmakro automatisch erzeugen. Trotzdem ist es beispielsweise für

- Schleifen,
- Bedingungen und
- benutzerdefinierte Dialogfelder

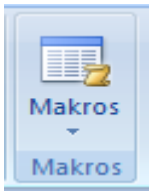
unerlässlich, eigenen VBA-Code zu schreiben. Zunächst soll aber erst einmal ein erstes (Mitschreib-) Makro erzeugt werden.

Workshop 1: VBA-Programmierung mit MS Excel

2 Beispielanwendung

2.1.Makroaufzeichnung starten

Klicken Sie im Register ANSICHT in der Gruppe MAKROS auf den Pfeil des Symbols MAKROS und wählen Sie in der geöffneten Liste den Eintrag MAKRO AUFZEICHNEN:



Das Dialogfenster MAKRO AUFZEICHNEN erfordert bis zu vier Eingaben: Diese werden im Folgenden beschrieben.

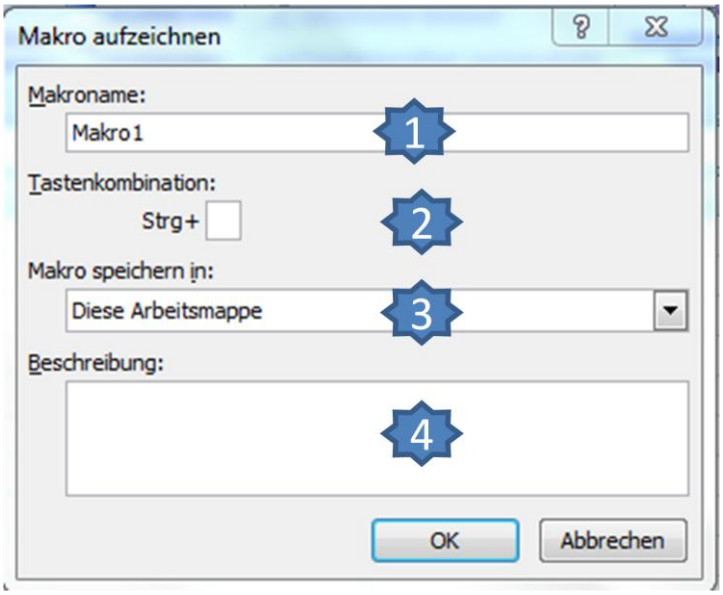


Abb. 1: Makroeigenschaften definieren

❶	Belassen Sie den voreingestellten Namen des Makros.
❷	Möchten Sie das Makro später über eine Tastenkombination ausführen, geben Sie einen Buchstaben in dieses Feld ein. Die für ein Makro festgelegte Tastenkombination überschreibt eine möglicherweise bereits existierende Tastenkombination, solange die Arbeitsmappe geöffnet ist, die das Makro enthält. Wählen Sie eine Tastenkombination, die in Excel noch nicht verwendet wird. In der Hilfe finden Sie unter dem Suchbegriff *Tastenkombinationen und Funktionstasten in Excel* die standardmäßig in Excel verwendeten Tastenkombinationen.
❸	Belassen Sie den Eintrag DIESE ARBEITSMAPPE, wenn Sie das Makro in der momentan geöffneten Arbeitsmappe anlegen möchten. Wenn Sie hingegen ein Makro aufzeichnen, welches immer für jede Arbeitsmappe verfügbar sein soll, wählen Sie aus dem Dropdown den Eintrag PERSÖNLICHE MAKROARBEITSMAPPE aus, ansonsten behalten Sie die standardmäßig eingestellte Option bei. In diesem Fall können Sie das Makro nur nutzen, wenn Sie die entsprechende Arbeitsmappe geöffnet haben.
❹	Geben Sie in dieses optionale Feld eine Beschreibung ein, die beispielsweise den Ablauf und/oder Verwendungszweck des Makros erläutert.

Nachdem Sie die Makroaufzeichnung mit einem Klick auf OK gestartet haben, befindet sich Excel im Aufzeichnungsmodus. Das erkennen Sie daran, dass ein Symbol in der Statusleiste am unteren Bildschirmrand erscheint.

Führen Sie jetzt die Arbeitsschritte nacheinander durch, die das Makro später automatisch in der vorgegebenen Reihenfolge ausführen soll.

2.2 Arbeitsschritte aufzeichnen

Als erste Aufgabe schreiben Sie in Zelle A1 des Tabellenblatts TABELLE1 das heutige Datum und ziehen das Ausfüllkästchen (links unten in Zelle A1) nach rechts bis zur Zelle D1.

Excel bietet keine Möglichkeit, die Aufzeichnung eines Makros zu unterbrechen (i. S. einer Pausentaste). Die vorgesehene Aufgabe des Makros kann aber erneut aufgenommen werden. Später kann dann im VBA-Editor der erzeugte Quellcode der Makros bearbeitet und aneinander gefügt werden.

Welche Aufzeichnungsarten unterstützt Excel?

Allgemein unterscheidet Excel zwischen Bezügen, die sich auf Zellen mit fester Adresse beziehen (absolute Bezüge) und solchen, die sich aus der Position der Zellen ergeben (relative Bezüge). Dementsprechend können Makros absolut oder relativ aufgezeichnet werden:

Absolute Aufzeichnung	Standardmäßig zeichnet Excel genau die Adressen der Zellen auf, die während der Makroaufzeichnung bearbeitet werden (absolute Bezüge)
Relative Aufzeichnung	Wenn ein Makro auf unterschiedliche Zellbereiche angewandt werden soll oder Zellen zu bearbeiten sind, die relativ zur markierten Zelle positioniert sind, kann ein Makro auch unabhängig von der tatsächlichen Zellposition aufgezeichnet werden (relative Bezüge)

Die Aufzeichnungsart wird wie folgt eingestellt:

Absolute Aufzeichnung	• Klicken Sie vor Aufzeichnung des Makros im Register ANSICHT in der Gruppe MAKROS auf den Pfeil des Symbols MAKROS • Deaktivieren Sie Eintrag RELATIVE AUFZEICHUNG.
Relative Aufzeichnung	• Markieren Sie die Zelle, von der die relativen Befehle des Makros ausgehen sollen. • Aktivieren Sie den Eintrag RELATIVE AUFZEICHNNG.

2.3 Makroaufzeichnung beenden

Klicken Sie auf das Symbol AUFZEICHNUNG BEENDEN.

2.4 Aufzeichnung des Makrorekorders betrachten

Markieren Sie das gerade aufgezeichnete Makro im Listenfeld.
Drücken Sie auf die Schaltfläche BEARBEITEN

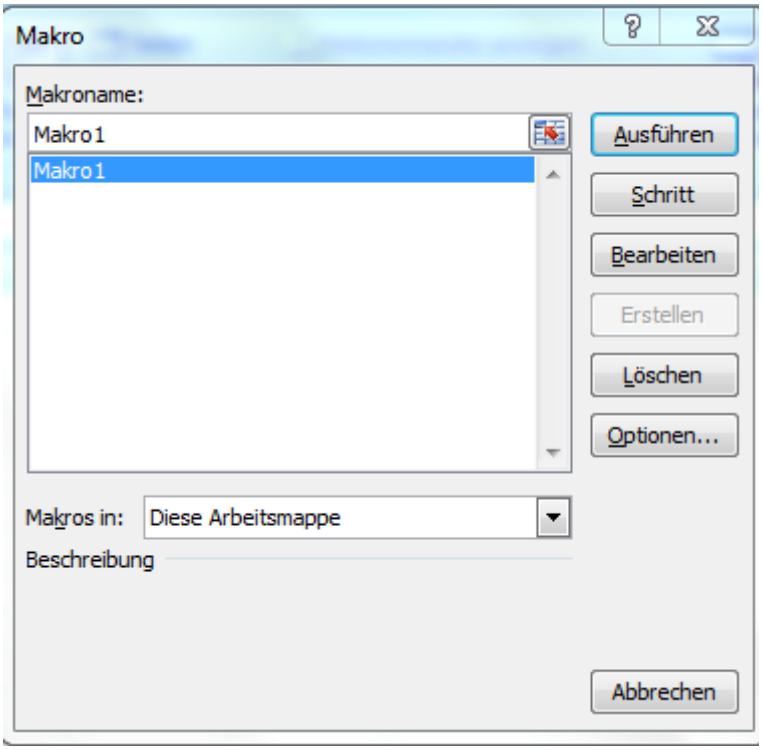


Abb. 2: Aufgezeichnetes Makro auswählen

Das aufgezeichnete Makro enthält folgenden Quellcode:

```
Sub Makro1()
'
' Makro1 Makro
    Range("A1").Select
    ActiveCell.FormulaR1C1 = "2/12/2011"
    Selection.AutoFill Destination:=Range("A1:D1"),
Type:=xlFillDefault
    Range("A1:D1").Select
End Sub
```

Aus dem aufgezeichneten Quellcode ist ersichtlich, dass die Programmiersprache VBA Englisch ist.

Zeilen im Code, die mit einem Hochkomma beginnen, sind Kommentarzeilen. Der Makrorecorder liefert wertvolle Hinweise über die Syntax der aufgezeichneten Befehle, leider verschluckt er gelegentlich einige davon oder zeichnet Befehle auf, die gar nicht benötigt werden. Aus diesem Grund stellt ein Mitschreibmakro oft nur den ersten Schritt dar, um ein Makro zu erstellen. Dann ist Nacharbeit erforderlich.

2.5 Makro bearbeiten

Das obige Makro könnte nach ein wenig Überarbeitung wie folgt aussehen:

```
Sub Makro1()
' Makro1 Makro
' Ausgangszelle bestimmen
    Range("A1").Select
' Das aktuelle Datum eingeben
    ActiveCell.FormulaR1C1 = Date
' Das Datum nach rechts auffüllen (Schrittweite 1 Tag)
    Selection.AutoFill Destination:=Range("A1:D1"),
Type:=xlFillDefault
    Range("A1:D1").Select
End Sub
```

Das Makro wurde um zusätzliche Kommentarzeilen ergänzt, die den Zweck des Makros beschreiben. Das heutige Datum wurde durch die in VBA eingebaute Funktion `Date` ersetzt. Die Standardfunktion `Date` liefert das aktuelle Datum.

Setzen Sie im Code einmal den Mauszeiger auf `Date` und drücken Sie dann auf die Funktionstaste F1. Sie erhalten dann aus der eingebauten Online-Hilfe mehr Infos für den jeweiligen Befehl, hier also über `Date`.

2.6 Makro ausführen

Zum Starten eines Makros gibt es mehrere Möglichkeiten:

- Mit der Tastenkombination Alt + F8 kann die im die Abb. 2 gezeigte Maske aufgerufen werden. Zuerst wird das Makro im Listenfeld ausgewählt. Mit anschließendem Klick auf die Schaltfläche AUSFUEHREN wird es gestartet.
- Starten Sie das Makro direkt im Codefenster, indem Sie den Mauszeiger auf die erste Codezeile des Makros setzen und die Taste F5 drücken.
- Starten Sie das Makro über eine Schaltfläche im Tabellenblatt.

Um ein Makro über eine Schaltfläche auf dem Tabellenblatt zu starten, verfahren Sie wie folgt:

Im Register ENTWICKLERTOOLS in der Gruppe STEUERELEMENTE finden Sie die Schaltfläche EINFÜGEN:

Hier sind zwei verschiedene Arten von Steuerelementen auswählbar:

Formularsteuerelemente	Diese können auf einfache Weise mit Makros verknüpft werden und erhalten dadurch ihre Funktionalität.
ActiveX-Steuerelemente	Um ihnen Funktionalität zu verleihen, muss der BVA-Editor eingesetzt werden.

Die angezeigten Formularsteuerelemente umfassen links oben das Symbol einer Schaltfläche.

1 Klicken Sie auf dieses Symbol und positionieren es an der gewünschten Stelle im Tabellenblatt.
2 Klicken Sie im Dialog MAKRO ZUWEISEN auf das Makro MAKRO1 und dann auf die Schaltfläche OK.

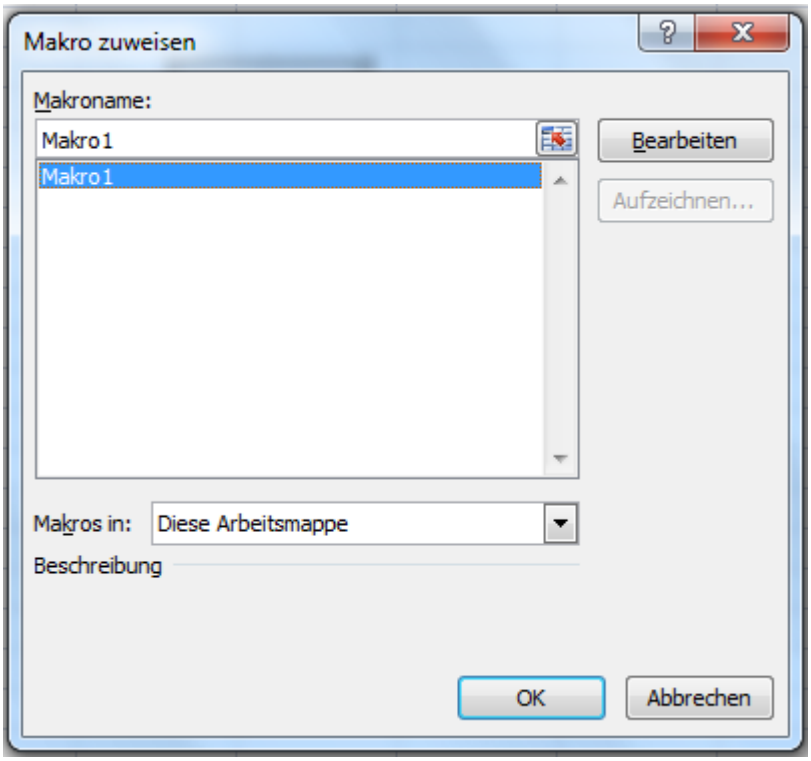
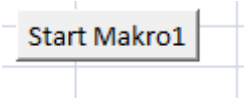


Abb. 3: Befehlsschaltfläche mit Makro verbinden

3 Geben Sie der Schaltfläche einen geeigneten Namen.



Steuerelemente auf Tabellenblättern können nur dann mit den ihnen verbundenen Makros gestartet werden, wenn das entsprechende Tabellenblatt aktiviert ist. Wenn Makros unabhängig von einem Tabellenblatt ausgeführt werden sollen, müssen Sie z. B. ein Symbol in die Symbolleiste für den Schnellzugriff einfügen.

Neben den oben genannten Steuerelementen können auch

- Grafiken
- selbst erstellte Zeichnungen oder
- Bildschirmabdrucke (engl. Screenshots)

verwandt werden, um sie mit Makros zu verbinden.

Erforderliche Arbeitsschritte:

❶	Durch das Klicken auf ein Symbol im Register EINFÜGEN in der Gruppe ILLUSTATIONEN kann eine Grafik an einer beliebigen Stelle eines Tabellenblatts eingefügt werden.
❷	Wird die eingefügte Grafik mit der rechten Maustaste angeklickt, kann der Kontextmenüpunkt MAKRO ZUWEISEN aufgerufen werden.

2.7 Arbeitsmappe mit Makros speichern

Standardmäßig werden Makros zusammen mit der jeweiligen Arbeitsmappe gespeichert. Makros können nur ausgeführt werden, wenn die entsprechende Arbeitsmappe geöffnet ist.

- Klicken Sie in der Symbolleiste für den Schnellzugriff auf das Disketten-Symbol. Das Dialogfenster SPEICHERN UNTER wird geöffnet.
- Ändern Sie bei Bedarf den vorgeschlagenen Speicherort.
- Geben Sie im Listenfeld DATEINAME den gewünschten Dateinamen ein.

- Wählen Sie über das Listenfeld DATEITYP das Dateiformat EXCEL-ARBEITSMAPPE MIT MAKROS.
- Bestätigen Sie mit der Schaltfläche SPEICHERN.

Bei dem oben genannten Dateiformat wird als Dateinamenserweiterung **.xlsm** verwendet.

3. Speicherorte für Makros festlegen

Beim Starten einer Makroaufzeichnung besteht die Möglichkeit, einen Speicherort für das neue Makro festzulegen (siehe Abb. 1, Punkt 3).

DIESE ARBEITSMAPPE	• Excel speichert das Makro in der aktuellen Arbeitsmappe. • Das Makro steht anderen Arbeitsmappen zur Verfügung, wenn DIESE ARBEITSMAPPE geöffnet ist.
NEUE ARBEITSMAPPE	• Im Hintergrund wird eine neue leere Arbeitsmappe geöffnet. Die Aufzeichnung erfolgt zwar in der aktuell geöffneten Arbeitsmappe, aber das Makro wird in der neuen Arbeitsmappe gespeichert. • Das gespeicherte Makro steht nur zur Verfügung, wenn diese NEUE ARBEITSMAPPE geöffnet ist.
PERSÖNLICHE MAKROARBEITSMAPPE	• Beim Schließen von Excel erhalten Sie eine Rückfrage. Sie können diese mit SPEICHERN bestätigen, um das Makro in der PERSÖNLICHEN MAKROARBEITSMAPPE zu speichern. • Die Arbeitsmappe wird unter dem Namen PERSONAL.XLSB im Dateiformat Microsoft Office Excel-Binärarbeitsblatt abgelegt. • Die darin gespeicherten Makros können in **allen** geöffneten Arbeitsmappen ausgeführt werden. Die Makros werden im Dialogfenster MAKRO mit den Makronamen PERSONAL.XLSB!Makroname aufgelistet.

Der zuletzt gewählte Eintrag im Listenfeld MAKRO SPEICHERN IN bleibt für die nächste Makroaufzeichnung erhalten.

4. Auto-Makros für Arbeitsmappen

Als Auto-Makros werden solche bezeichnet, die in bestimmten Situationen automatisch ausgeführt werden, beispielsweise, wenn eine Arbeitsmappe geschlossen oder geöffnet wird. Excel bietet mehrere Möglichkeiten, Makros automatisch auszuführen:

Makroname	Beschreibung
`Auto_Open`	Wird beim Öffnen einer Arbeitsmappe automatisch ausgeführt
`Auto_Close`	Wird beim Öffnen einer Arbeitsmappe automatisch ausgeführt

Tab: 1: Auto-Makros für Arbeitsmappen

Arbeitsschritte zur Erstellung eines primitiven Auto-Makros:

❶	Erstellen Sie eine neue Arbeitsmappe in Excel.
❷	Klicken Sie im Register EMTWICKLUNGSTOOLS in der Gruppe CODE auf die Schaltfläche MAKRO AUFZEICHNEN. Vergeben Sie den Namen `Auto_Open` für das neue Makro. Wählen Sie als Speicherort für das Makro den Eintrag DIESE ARBEITSMAPPE.
❸	Schreiben Sie als Text „*Hallo Welt!*“ in die Zelle A1 und beenden Sie die Makroaufzeichnung.
❹	Löschen Sie den Inhalt der Zelle A1.
❺	Speichern und schließen Sie die Arbeitsmappe.
❻	Öffnen Sie die Arbeitsmappe.
Ergebnis: Es wird automatisch der Text *„Hallo Welt!“* in Zelle A1 eingefügt.	

Das zugehörige Auto-Makro enthält folgende VBA-Befehle:

```
Sub Auto_Open()
   Range("A1").Select
   ActiveCell.FormulaR1C1 = "Hallo Welt!"
End Sub
```

5. Makrosicherheit

In neueren Excel–Versionen können Standard-Arbeitsmappen (**.xlsx**) keine Makros enthalten. Beim Öffnen dieser Arbeitsmappen besteht folglich auch keine Gefährdung durch Makroviren.

Wenn Sie eine Excel-Datei öffnen, die Makros enthält **(.xlsm**), kann es vorkommen, dass Ihnen ein Dialogfeld angezeigt wird mit der Meldung, dass die Makros deaktiviert wurden. Alternativ kann Ihnen ein Dialogfeld angeboten werden, das Ihnen erlaubt, die Makro zu aktivieren oder zu deaktivieren.

Makros können bösartige Makroviren enthalten. Deshalb wird die Ausführung von Makros aus nicht vertrauensvollen Quellen von Excel standardmäßig unterdrückt.

Excel (und die anderen Microsoft Office-Programme) kennen vier Sicherheitsstufen für die Ausführung von Makros (siehe Abb. 4)

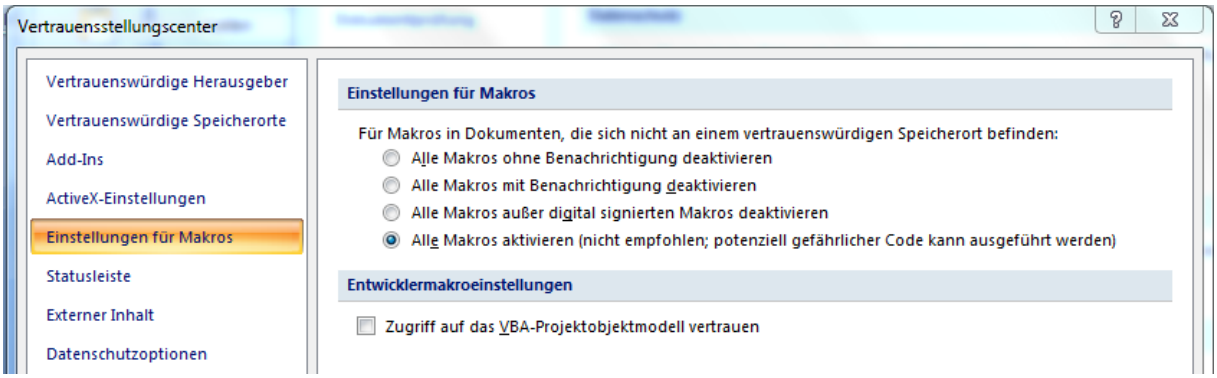


Abb. 4 Einstellungen für Makros im Vertrauensstellungscenter

Die Sicherheitsstufe kann u. a. eingestellt werden über das Register ENTWICKLERTOOLS, Gruppe CODE, Schaltfläche MAKROSICHERHEIT.

Das Thema Makrosicherheit spielt insbesondere eine Rolle bei der Weitergabe von Makros.

6. Übungen

6.1 Gelbe Zellen

Ein Mitschreibmakro mit dem Namen `GelbeZellen` soll markierte Zellen im `Range("A1:B5")` eines Tabellenblatts automatisch gelb formatieren.

- Vergleichen Sie den Code des aufgezeichneten Makros mit folgendem Listing.
- Ergänzen Sie Ihr Listung mit Kommentaren.

```
Sub Makro1()
'
' Makro1 GelbZellen
'
    Range("A1:B5").Select
    With Selection.Interior
        .Pattern = xlSolid
        .PatternColorIndex = xlAutomatic
        .Color = 65535
        .TintAndShade = 0
        .PatternTintAndShade = 0
    End With
End Sub
```

6.2 Spalte A formatieren

Erstellen Sie das Mitschreibmakro `FormatiereSpalte` in einer neuen Arbeitsmappe:
Aufzuzeichnende Arbeitsschritte:

- Spalte A markieren,
- Schriftschnitt „fett“,
- Schriftfarbe „rot“
- Füllfarbe „gelb“

```
Sub FormatiereSpalte()
'
' FormatiereSpalte Makro
'
    Columns("A:A").Select
    Selection.Font.Bold = True
    With Selection.Font
        .Color = -16776961
        .TintAndShade = 0
    End With
    With Selection.Interior
        .Pattern = xlSolid
        .PatternColorIndex = xlAutomatic
        .Color = 65535
        .TintAndShade = 0
        .PatternTintAndShade = 0
    End With
End Sub
```

Stellen Sie den Ausgangszustand von Spalte A wieder her.
Führen Sie das Makro `FormatiereSpalte` aus.

Ändern und kommentieren Sie das Makro `FormatiereSpalte wie folgt:`

```
Sub FormatiereSpalte()
    ' Spalte A markieren
    Columns("A:A").Select
    ' Schriftart fett
    Selection.Font.Bold = True
    ' Schriftfarbe rot
    With Selection.Font
        .Color = vbRed
    End With
    ' Füllfarbe gelb
    With Selection.Interior
        .PatternColorIndex = xlAutomatic
        .Color = vbYellow
    End With
End Sub
```

Frage: Erfüllt das geänderte Makro die gleiche Aufgabe wie zuvor?
Schließen Sie die neue Arbeitsmappe, ohne diese zu speichern.

6.3 Summe bilden

Erstellen Sie das Mitschreibmakro `SummeBilden` in einer neuen Arbeitsmappe.
Aufzuzeichnende Arbeitsschritte: Die Summe der Inhalte der Zellen A1, A2 und A3 soll in Zelle B4 anzeigt werden.

```
Sub SummeBilden()
'
' SummeBilden Makro
'
    Range("B4").Select
    ActiveCell.FormulaR1C1 = "=SUM(R[-3]C[-1]:R[-1]C[-1])"
    Range("B5").Select
End Sub
```

Workshop 1: VBA-Programmierung mit MS Excel

6.4 Schaltfläche erstellen

Erstellen Sie die Schaltfläche `TagesZeit`, mit der Sie das aktuelle Datum und die aktuelle Zeit im aktuellen Tabellenblatt angezeigt bekommen (Hinweis: Excel-Funktion =JETZT())

Benötigtes Makro:

```
Sub Tageszeit()
' Tageszeit Makro
    Range("A1").Select
    ActiveCell.FormulaR1C1 = "=NOW()"
End Sub
```

Hinweis: Die VBA-Funktion `NOW()` wird im deutschsprachigen Excel als `JETZT()` bezeichnet. Excel ist also lokalisiert, VBA dagegen nicht.
Merke: Die Programmierung mit VBA erfordert Englischkenntnisse.

Was bedeutet …	
Makrorecorder	Ist ein integraler Bestandteil von Excel, um vom Benutzer durchgeführte Arbeitsschritt in Form eines sogen. *Mitschreibmakros* aufzuzeichnen. Die Aufzeichnung erfolgt mit Sprachelementen der Programmiersprache *VBA*.
Mitschreib-makro	Bezeichnet das automatische Mitschreiben der Tastatureingaben mit dem in Excel eingebauten *Makrorecorder*. Beginn und Ende der Aufschreibung bestimmt der Anwender. Die Benennung eines Mitschreibmakros obliegt ebenfalls Anwender.
Makro	Wird häufig mit dem Begriff *Prozedur* gleichgesetzt. Ein *Mitschreibmakro* ist ein *Unterbegriff* von Makro.
VBA	VBA ist eine objektorientierte Programmiersprache, deren grundlegende Sprachelemente in allen MS-Office-Anwendungen (EXCEL, WORD, ACCESS, OUTLOOK, POWERPOINT) angewandt werden können. **VBA verwendet nur englische Schlüsselworte.** Die Bezeichnung von Prozeduren, Funktion, Variablen, Konstanten usw. kann allerdings in Deutsch erfolgen.
VBA-Editor	Ist Bestandteil der integrierten Entwicklungsumgebung von Excel. Makros, auch Mitschreibmakros, werden mit dem VBA-Editor editiert.
Schlüsselwörter	Das sind englische Wörter, die in der Programmiersprache VBA **reserviert** sind und deshalb nur für den vorgesehen Zweck verwandt werden dürfen.
Prozedur	Bezeichnet in VBA ein *Unterprogramm*, das mit dem Schlüsselwort **`Sub`** eingeleitet und mit **`End Sub`** beendet wird. Eine Prozedur kann andere Prozeduren oder Funktionen aufrufen.
Programm-generator	Ein Programmgenerator erzeugt ablauffähige Programme, die mehr Logik enthalten als eine Aneinanderreihung einfacher Anweisungen wie sie der Makrorecorder erzeugt. **Der Makrorecorder in Excel ist kein Programmgenerator**! VBA-Code, der mit dem Makrorecorder aufgezeichnet wird, kann allerdings vom Anwender geändert und ergänzt werden. Das erfordert aber die syntaktisch richtige Anwendung der VBA-Sprachelemente.
Eingebaute Funktionen	Bei der VBA-Programmierung kann auf die in Excel eingebauten Funktionen zurückgegriffen werden. **Excel enthält enorm viele eingebaute Funktionen**, zum Beispiel finanzmathematische und statistische Funktionen aber auch String- und Datumsfunktionen und dergleichen mehr. Die eingebauten Funktionen erleichtern die Erstellung von Anwendungen erheblich, insbesondere für VBA-Einsteiger.

Workshop 1: VBA-Programmierung mit MS Excel

7 Anhang

In der 1. Sitzung des Excel-Workshops wurde von einem Teilnehmer die Frage gestellt, wie man den Farbwert einer Zelle in einem Tabellenblatt erkennen und wie man in Abhängigkeit davon bestimmte Aktionen ausführen kann.

Im Folgenden wird die Lösung gezeigt. Die Grundlagen zum vollen Verständnis der Lösung werden erst in den kommenden Sitzungen vermittelt.

Die Prozedur `FarbpaletteAnzeigen` erzeugt in den Zellen A1 bis A56 des Tabellenblatts `Tabelle1` genau 56 Farben und weist in Spalte B den jeweils dazugehörigen Farbindex aus.

```
Sub FarbpaletteAnzeigen()
   ' Farbpalette des Innenbereichs von Zellen im Tabellenblatt darstellen
   Const conColors As Integer = 56  ' Anzahl Farben
   Dim intColor    As Integer       ' Farbindex
   Dim intSchleife As Integer       ' Schleifenzähler
   ' Mit Excel-Objekt (Tabellenblatt) bestimmen ...
   With ActiveWorkbook.Worksheets("Tabelle1")
      ' Zählschleife über 7 x 8 = 56 Farben
      For intSchleife = 1 To conColors
         intColor = intSchleife
         ' Farbe ausgeben
         .Cells(intSchleife, 1).Interior.ColorIndex = intColor
         ' Farbindex ausgeben
         .Cells(intSchleife, 2).Value = intColor
      Next intSchleife
   End With
   MsgBox "Fertig!", vbExclamation
End Sub
```

Die Prozedur `ZellbereichDurchlaufen` untersucht die Zellen A1 bis A56 des Tabellenblatts `Tabelle1` hinsichtlich bekannter Farbnamen. In Abhängigkeit davon könnte in weitere Prozeduren verzweigt werden, die gewünschte Aufgaben ausführen.

```
Sub ZellbereichDurchlaufen()
   ' Benannte Farben zählen
   Dim rngZellBereich   As Excel.Range ' Zellbereich
   Dim objZelle         As Object      ' einzelne Zelle
   Dim strFarbName      As String      ' Farbname
   Dim intZaehler       As Integer     ' Farbzähler
   ' Verweis auf Zellbereich setzen
   Set rngZellBereich = ActiveWorkbook.Worksheets("Tabelle1").Range("A1:A56")
   ' Zellbereich durchlaufen
   For Each objZelle In rngZellBereich
      ' Rückgabewert der Funktion 'FarbeErkenne' zwischenspeichern
      strFarbName = FarbeErkennen(objZelle)
      ' Wenn der Farbname bekannt ist, dann ...
      If strFarbName <> "unbenannt" Then
         intZaehler = intZaehler + 1   ' Farbzaehler erhöhen
      End If
   Next objZelle
   MsgBox "Der Zellbereich " & rngZellBereich.Address & _
      " enthält " & intZaehler & " benannte Farben."
End Sub
```

Die Funktion `FarbeErkennen` erhält von der aufrufenden Prozedur eine Zelle (`objCell`) übergeben und gibt den Namen der zugehörigen Farbe zurück:

```
Function FarbeErkennen(objCell As Object) As String
   Dim strColorName As String ' Farbname
   Application.Volatile    ' veränderliche Funktion
   With objCell            ' mit übergebener Zelle
      ' Fallstruktur für Farbindex
      Select Case .Interior.ColorIndex
         Case Is = 1
            strColorName = "schwarz"
         Case Is = 2
            strColorName = "weiß"
         Case Is = 3
            strColorName = "rot"
         Case Is = 4
            strColorName = "grün"
         Case Is = 5
            strColorName = "blau"
         Case Is = 6
            strColorName = "gelb"
         Case Is = 7
            strColorName = "rosa"
         Case Is = 8
            strColorName = "türkis"
         Case Else
            strColorName = "unbenannt"
      End Select
   End With
   'Rückgabewert der Funktion setzen
   FarbeErkennen = strColorName
End Function
```