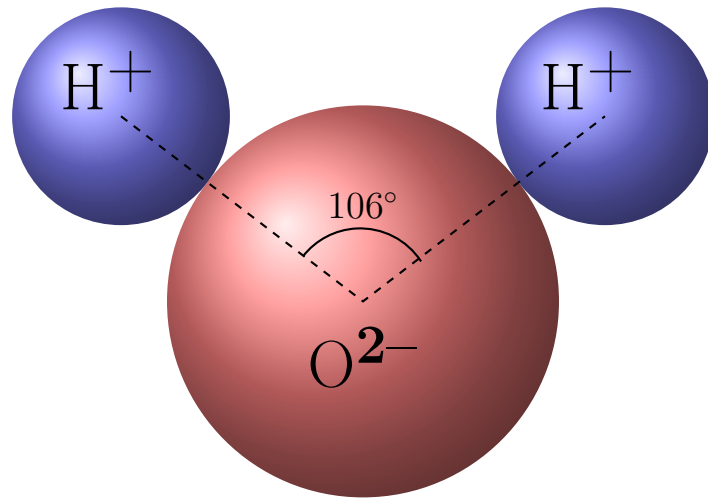


Ecce etiam ego adducam aquas super terram!



L^AT_EX

(opus imperfectum)

Alfred H. Gitter

Version 8.4 vom 2. August 2018

Inhaltsverzeichnis

1	Grundlagen	3
1.1	Einführung	3
1.2	Befehle	6
1.3	Dokumentklassen und Pakete	8
1.4	Beispiel	11
1.5	Gliederung des Dokuments	13
2	Besondere Teilbereiche	15
2.1	Verzeichnisse (Inhalt, Abbildungen, Literatur)	15
2.2	Fuß- und Randnoten, Querverweise, Hyperlinks	16
2.3	Bilder, Tabellen, Balkendiagramme, Kommentare	18
2.4	Aufzählungen und Theorem-Umgebungen	26
2.5	Programm-Code, Zeichnungen, Zähler	28
3	Seite, Absatz und Wort	33
3.1	Seitengestaltung, mehrere Spalten	33
3.2	Rahmen und Minipages	35
3.3	Ausrichtung, Einrückung, Trennungen, Abstände	38
3.4	Typographische Regeln	41
4	Zeichenformatierung	43
4.1	Schrift	43
4.2	Leer- und Sonderzeichen	48
4.3	Physikalische Einheiten	53
5	Mathematik	57
5.1	Mathematik im Fließtext	57
5.2	Sonderzeichen in einer <i>math</i> -Umgebung	61
5.3	Gleichungen	64
6	Grafiken mit TikZ	67
6.1	Strichzeichnungen und Füllungen	67
6.2	Graphen und Funktionen	73
6.3	Elektrische Schaltkreise und Optik	75
6.4	Randnotizen, Kästen, Kalender und Avatare	79
7	Ergänzungen	85
7.1	Physik, Chemie, Bioinformatik, Übungen usw.	85
7.2	Gedichte, Spiele, Strichcode und QR-Code	93
7.3	Präsentation mit <i>beamer</i>	97
7.4	Briefe	103
7.5	L ^A T _E X-Editor <i>Gummi</i>	105

1 Grundlagen

1.1 Einführung

„Da \LaTeX ein überholtes und auslaufendes System zur Formatierung von Text darstellt, entschloss ich mich vor rund dreißig Jahren, stattdessen nur das Textverarbeitungssystem eines namhaften Herstellers zu verwenden. *Vielleicht war das falsch.*“

Die vorliegende Skripte ersetzt nicht Bücher, die \LaTeX fundiert erläutern. Vielmehr gibt sie Hinweise in Form von kommentierten Beispielen, die dem ungeduldigen Anwender helfen können. Die Richtigkeit wird natürlich nicht gewährleistet. Allerdings besteht die Gefahr der Ablenkung. Man sollte zunächst schreiben und die Formatierung \LaTeX überlassen! Danach kann man einige der schönen Extras einfügen, welche hier dargestellt werden. Das Format der Skripte ist, dem Zweck entsprechend, nicht optimal für ein Buch, sondern stellt einen Kompromiss dar. Dieser ermöglicht ein- oder zweiseitige, Papier sparende Ausdrücke einzelner Abschnitte (Save Paper in School) und stellt (für die Freunde von Copy & Paste) beim Lesen am Bildschirm Verweise auf tex-Dateien bereit.

Kurzum, hier ist ein geschenkter Gaul, auf dem ihr reiten mögt oder nicht.

\LaTeX erweitert eine alte, \TeX genannte Programmiersprache zum Textsatz (von Donald E. Knuth von 1977 bis 1989 entwickelt) um benutzerfreundliche Befehle. Das \LaTeX in \LaTeX steht für den Entwickler Leslie Lamport.

Die seit 1994 aktuelle Version von \LaTeX heißt $\LaTeX 2_{\epsilon}$; eine langfristig angelegte Weiterentwicklung läuft unter dem Namen LaTeX3. Die ältere Alternative zu \LaTeX , das Eingabeformat *plain TeX* wird nur noch selten verwendet.

Bei der Textgestaltung mit \LaTeX wird der Inhalt eines Dokuments mithilfe lesbarer Steuerbefehle in ein ansprechendes Druckbild übersetzt. Der Autor gibt, mit im Text eingebetteten, ausgeschriebenen Befehlen, die wesentlichen Gestaltungsmerkmale vor, überlässt Einzelheiten der Formatierung aber dem Programm. Die Befehle dienen zum Beispiel der Auswahl einer Schriftart, der Gliederung oder einer automatischen Nummerierung.

Es gibt verschiedene Ausgabeformate für die Darstellung der mit \LaTeX erzeugten Dokumente. Heute wird PDF am häufigsten verwendet. Ein geeignetes Computerprogramm, welches \LaTeX als Eingabeformat und PDF als Ausgabeformat hat, ist PDF \LaTeX . Wesentliches Teil (Engine) von PDF \LaTeX ist pdf \TeX . Nachfolger von PDF \LaTeX könnte zukünftig Lua \LaTeX (Eingabeformat: \LaTeX , Engine: Lua \TeX , Ausgabeformat: PDF) sein. *Sub reservatione Jacobea* (so Gott will und wir leben).

Im Folgenden gehen wir davon aus, dass PDF \LaTeX (auch pdf \LaTeX oder pdf \LaTeX geschrieben) eingesetzt wird. \LaTeX -Editoren haben dies meistens als Voreinstellung.

Hinweise und Hilfe erhält man unter anderem von der Deutschsprachigen Anwendervereinigung TeX e.V. (DANTE). Die Webseite ist <http://www.dante.de/>

Typographie ist die Wissenschaft oder Kunst des Satzsetzes. Ein Text besteht aus Zeichenketten. Seine Lesbarkeit für Menschen hängt von Form und Anordnung der Zeichen, Satzsatz genannt, ab.

Mit \LaTeX wird ein guter Satzsatz, der den Regeln der Typographie folgt, erreicht, ohne dass der Nutzer eingreifen muss. Der Nutzer kann aber, zum Beispiel durch Wahl der Schriftart, auf die automatisch erstellte Vorlage Einfluss nehmen.

Installation wird hier nicht näher beschrieben. Für Linux-Betriebssysteme kann sie ohne Besonderheiten über die Linux-Paketverwaltung erfolgen. Wenn möglich, sollten schließlich mindestens folgende Linux-Pakete vorliegen: *texlive*, *texlive-lang-german*, *texlive-generic-extra* und *texlive-latex-extra*. Wer genug Speicherplatz hat, sollte das ganze, umfangreiche \LaTeX -System installieren.

Unter Linux können zusätzliche Latex-Pakete, die als Dateien mit der Endung `.sty` (oder `.cls` oder `.trsl`) vorliegen, installiert werden, indem sie in das Verzeichnis `~/texmf/tex/latex` kopiert werden (`~` steht für das Benutzerverzeichnis `/home/BENUTZER`) und danach in einer Konsole (Terminal) der Befehl

```
texhash ~/texmf
```

ausgeführt wird. Falls es das Verzeichnis noch nicht gibt, erzeugt man es mit

```
mkdir ~/texmf/tex/latex
```

Man kann statt einer Datei `PAKET.sty` auch ein Verzeichnis `PAKET`, das die Datei `PAKET.sty` enthält, in das Verzeichnis `~/texmf/tex/latex` kopieren.

Manche Latex-Pakete werden durch zwei Dateien mit den Endungen `.dtx` und `.ins` bereitgestellt. Unter Linux kopiert man beide in ein beliebiges temporäres Verzeichnis, zum Beispiel `~/Downloads`, und führt dann nacheinander die Befehle

```
cd ~/Downloads
```

und

```
latex ./PAKET.ins
```

aus. Damit erhält man eine Datei `PAKET.sty` (oder mehrere) und verfährt weiter wie oben beschrieben. Mehr zu diesem Thema findet man auf der Webseite

https://en.wikibooks.org/wiki/LaTeX/Installing_Extra_Packages

Für \LaTeX unter Kleinweich-Betriebssystemen (*Microsoft Windows*) gibt es verschiedene Distributionen, unter anderem *TeXLive* und *MiKTeX*. Eine Anleitung zur Installation von *TeXLive* findet man auf der Webseite

<https://www.tug.org/texlive/doc/texlive-de/texlive-de.pdf>

Da *TeXLive* sehr umfangreich ist (etwa 3,5 GiB), kann eine direkte Installation über das Internet Probleme bereiten. Besser ist es, von einer DVD zu installieren. Wenn man keine hat, kann man ein [ISO-Abbild](http://ftp.uni-erlangen.de/ctan/systems/texlive/Images/) herunterladen (Webseite: <http://ftp.uni-erlangen.de/ctan/systems/texlive/Images/>) und damit eine *TeXLive*-DVD erzeugen (brennen).

Auch unter Windows 10 und *TeXLive* können zusätzliche Latex-Pakete mithilfe der Konsole installiert werden. Wir gehen davon aus, dass die neuen Pakete (Dateien mit der Endung `.sty`, `.cls`, `.trsl`, `.dtx` oder `.ins`) nach dem Herunterladen im Unterverzeichnis *Downloads* des Benutzerverzeichnisses gespeichert sind. Durch die Tastenkombination *Windowstaste R* öffnet sich ein Fenster mit dem Titel *Ausführen*. Gibt man hier den Befehl `cmd.exe` ein, öffnet sich ein Konsolenfenster. Wichtige Befehle sind dort:

`dir` zeigt eine Liste von Verzeichnissen und Dateien im aktuellen Verzeichnis
`cd verz` macht das Unterverzeichnis *verz* zum aktuellen Verzeichnis
`cd ..` macht das übergeordnete Verzeichnis zum aktuellen Verzeichnis
`mkdir verz` erzeugt ein Unterverzeichnis *verz* im aktuellen Verzeichnis
`latex dat` führt das Programm pdf_{TEX} mit der Datei *dat* aus
`texhash verz` führt das Programm `texhash` mit dem Verzeichnis *verz* aus

Nach dem Öffnen einer Konsole ist das Benutzerverzeichnis des aktuellen Nutzers *Nutzer* das aktuelle Verzeichnis. Mit dem Befehl

```
mkdir texmf\tex\latex
```

erzeugt man ein Unterverzeichnis, in dem neu installierte Latex-Pakete abgelegt werden können. Falls es dies Verzeichnis schon gibt, erhält man eine entsprechende Meldung vom Betriebssystem. Dann kopiert man, (nicht in der Konsole, sondern) wie üblich mit Maus und Windows-Fenstern, Dateien mit der Endung `.sty`, `.cls` oder `.trsl` vom Verzeichnis *Nutzer\Downloads* nach *Nutzer\texmf\tex\latex*. Liegt das Latex-Paket in Form zweier Dateien *dat.dtx* und *dat.ins* vor, geht man in der Konsole mit `cd Downloads` ins Unterverzeichnis *Downloads* und führt dort

```
latex dat.ins
```

aus. Danach liegen im Unterverzeichnis *Downloads* Dateien mit der Endung `.sty`, `.cls` oder `.trsl`, welche man in das Verzeichnis *Nutzer\texmf\tex\latex* kopiert. Nun kann man im Unterverzeichnis *Downloads* alle heruntergeladenen und erzeugten Dateien löschen. In der Konsole geht man mit `cd ..` wieder ins Benutzerverzeichnis *Nutzer* und führt den Befehl

```
texhash texmf
```

aus. Schließlich ist man fertig und kann das Konsolenfenster schließen.

Eingabe von Text und Befehlen kann mit jedem einfachen, allgemein verwendbaren Editor, unabhängig vom Betriebssystem des verwendeten Rechners, geschehen. Nachteilig ist beim Arbeiten mit einem einfachen Editor, dass das gestaltete Dokument nicht sichtbar ist, fehlerhafte Befehle nicht sofort erkannt werden und für häufig verwendete Befehle keine Abkürzung bereitsteht. Deshalb ist es bequemer, mit einem besonderen L^AT_EX-Editor zu arbeiten. Von diesen gibt es viele gute. Ein einfacher L^AT_EX-Editor für Linux-Betriebssysteme (*Gummi*) wird später vorgestellt.

Dateien mit L^AT_EX-Quellcode werden gewöhnlich unter einem Namen mit der Endung `.tex` gespeichert. Druckfertige Dateien haben das Format PDF mit der Endung `.pdf`. PDF-Dateien haben den Vorteil, auf verschiedenen Rechnern, unabhängig vom Betriebssystem, die gleiche Ausgabe auf Bildschirm und Drucker zu erzeugen. (Theoretisch jedenfalls; in der Praxis verhält sich manches PDF-Betrachterprogramm wie ein zirkusreifer Lehmziegel. Um dennoch das gewünschte Druckergebnis in annehmbarer Qualität zu erhalten, kann es helfen, im erweiterten Menü das Drucken als Bild = image zu wählen. Ansonsten sollte man ein anderes PDF-Betrachterprogramm benutzen.)

Der Quellcode einer regelgerechten L^AT_EX-Datei wird als Dokument bezeichnet. Es enthält zwei Teile. Der erste heißt Präambel, der zweite *document*-Umgebung.

Druckfertiges Ergebnis ist die PDF-Datei, welche vom Programm PDF_{Latex} aus dem L^AT_EX-Quellcode erzeugt wurde. L^AT_EX-Editoren erledigen das auf Knopfdruck.

Bei Linux-Betriebssystemen kann man PDF_{La}TeX auch in einer Konsole (Terminal) mit dem Befehl

```
pdflatex datei.tex
```

ausführen, wenn die Datei *datei.tex* im aktuellen Verzeichnis liegt.

Beim Ausdruck von PDF-Dateien auf einem Drucker, der nicht bis zum Seitenrand drucken kann, sollte man im Druck-Menü *randlosen Druck* wählen. Ansonsten werden die Druckseiten automatisch verkleinert ausgegeben. Leider unterstützen nicht alle Laserdrucker den randlosen Druck.

1.2 Befehle

L^AT_EX-Befehle beginnen mit einem \ (Backslash). Danach kommt der Befehlsname, der aus Buchstaben besteht. Anschließend werden oft, in geschweiften oder eckigen Klammern, Argumente übergeben. Argumente (auch Parameter genannt) beeinflussen die Befehle.

Man unterscheidet Pflichtargumente, deren Angabe notwendig ist, und optionale Argumente, mit denen der Befehl (*command*) verändert werden kann, aber nicht muss. Pflichtargumente stehen in geschweiften Klammern *{...}*, optionale in eckigen Klammern *[...]*. Mehrere Argumente in einer Klammer werden durch den Aufzählungsoperator, ein Komma, getrennt.

Das Ende eines Befehlsnamens wird durch ein folgendes Leerzeichen oder Sonderzeichen erkannt. Folgt ein oder mehrere Leerzeichen, werden sie nicht gedruckt. Will man ein Leerzeichen nach einem Befehl darstellen, kann man einen leeren Block *{}* an den Befehlsnamen anfügen. Dies wird hier anhand des Befehls *\LaTeX* gezeigt, welcher (ohne Argumente) das L^AT_EX-Logo ausgibt.

\LaTeX{} ist gut, *\LaTeX* nicht.

L^AT_EX ist gut, L^AT_EXnicht.

Eine zweiseitige, deutsche Übersicht wichtiger Befehle findet man unter der Bezeichnung [latexsheet-de.pdf](#). Der Quellcode liegt in der Datei [latexsheet-de.tex](#) vor. Es gibt auch eine schöne [Befehlsübersicht von Christian Feuersänger](#).

Kommentare verbessern die Verständlichkeit der verwendeten Befehle. Ein Kommentar (*comment*) wird mit % (Prozentzeichen) eingeleitet. Der anschließende Text bis zum Zeilenende wird nicht abgebildet. Mehr dazu in Abschnitt 2.3 auf Seite 25.

Die ersten drei Befehle sind

<code>\documentclass[a4paper, 11pt]{scrartcl}</code>	% Präambel	Dokument
<code>\begin{document}</code>	% Anfang der document-Umgebung	mit drei
<code>Hallo Welt\,!</code>	% (Hier steht der Textinhalt.)	Befehlen,
<code>\end{document}</code>	% Ende der document-Umgebung	Ausgabe:
		Hallo Welt!

Eine detaillierte Vorgabe des Formats eines L^AT_EX-Dokuments geschieht mit dem ersten Befehl, welcher eine Dokumentklasse (hier: *scrartcl*) und die Schriftgröße (hier: *11pt*) bestimmt. In der Regel stehen zumindest (ohne Laden zusätzlicher Programmpakete) die Schriftgrößen 10 pt, 11 pt und 12 pt zur Verfügung. Das Papierformat wird hier mit der Option *a4paper* (für das A4-Format) angegeben.

In der *document*-Umgebung, zwischen `\begin{document}` und `\end{document}`, steht der Textinhalt, hier: Hallo Welt!

Im Text erfolgt die Silbentrennung am Zeilenende automatisch. Die Strukturierung des Texts wird durch eingebettete Befehle vorgenommen. In obigem Beispiel fügt der Befehl `\,` ein kurzes Leerzeichen (Lücke) ein.

Umgebungen sind bestimmte zusammenhängende Bereiche, welche Text und Befehle enthalten. Beispiele sind eine Aufzählung, aber auch die gesamte *document*-Umgebung. Eine Umgebung (*environment*) wird mit `\begin{...}` eingeleitet und mit `\end{...}` abgeschlossen, wobei für `...` der Name der Umgebung eingesetzt wird.

Gleitobjekte sind Umgebungen, die nicht unbedingt dort gedruckt werden müssen, wo sie im Quellcode beschrieben werden. Gleitobjekte sind insbesondere Abbildungen (*figure*-Umgebungen) und Tabellen (*table*-Umgebungen), die auch eine Legende (Beschriftung) haben können. Das Erscheinungsbild einer Seite ist gefälliger, wenn man \LaTeX erlaubt, die Gleitobjekte automatisch zu platzieren. Die Reihenfolge der Gleitobjekte bleibt dabei natürlich unverändert.

Präambel ist der Bereich eines Dokuments, der vor der *document*-Umgebung steht. Die Präambel (*preamble*) enthält den Befehl `\documentclass` und danach weitere Befehle, die zum Beispiel besondere Programmteile (Pakete) laden (siehe unten).

Schalter sind Befehle, die solange wirken können, bis sie durch einen neuen Befehl aufgehoben oder geändert werden. Es ist jedoch meistens besser, den Textbereich, auf den ein Schalter (*switch*) wirkt, von vornherein zu beschränken. Dazu dient eine Umgebung oder die Einfassung von Schalter und Text in einen (durch geschweifte Klammern begrenzten) Block.

Befehle definieren kann sinnvoll sein, um häufig gebrauchte, lange Befehlsfolgen abzukürzen. Mit dem Befehl `\newcommand*`, gefolgt von zwei Pflichtargumenten, wird in der Präambel ein neuer Befehl definiert, vorausgesetzt, dass es einen gleichnamigen Befehl noch nicht gibt. Wenn zum Beispiel das Wort *Freiheit*, mit kleinerer Schriftgröße und kursiv geschrieben, mehrfach im Text vorkommt, kann man statt der langen Befehlsfolge `\begin{small}\textit{Freiheit}\end{small}` den kurzen Befehl `\F` schreiben, nachdem dieser mit

```
\newcommand*\F{\begin{small}\textit{Freiheit}\end{small}\xspace}%  
% der Befehl ohne Argumente \F gibt das Wort Freiheit formatiert aus
```

definiert wurde. Der Name des neuen Befehls, `\F`, ist das erste Pflichtargument und die neue Befehlsfolge ist das zweite Pflichtargument von `\newcommand*`.

Jeder selbst definierte Befehl sollte einen kurzen, doch verständlichen Namen bekommen. Dieser darf nur (kleine oder große) Buchstaben enthalten! Der Zweck des neuen Befehls sollte in einer Kommentarzeile erläutert werden. Alle Zeilen der Befehlsdefinition sollten mit einem `%` (Kommentarzeichen) abgeschlossen werden, um zu verhindern, dass unerwünschte Leerzeichen (durch Zeilenumbrüche) in der Ausgabe auftauchen.

In der Befehlsfolge des zweiten Pflichtarguments steht am Ende noch `\xspace`. Dieser Befehl ist nur dann verfügbar, wenn vorher in der Präambel das Paket *xspace*

mittels des Befehls `\usepackage{xspace}` geladen wurde (siehe unten). Er stellt sicher, dass ein Leerzeichen, welches möglicherweise nach dem neuen Befehl `\F` geschrieben wird, nicht verschwindet.

Falls man, nachdem der Text geschrieben wurde, die *Freiheit* stets groß schreiben möchte, muss man nur den Befehl `\F` ändern und aus jeder *Freiheit* wird *Freiheit*.

Wenn man Zeichenketten eines bestimmten logischen Typs, zum Beispiel Tiernamen, in einem Text durch einen Schriftstil auszeichnen will, ist es sinnvoll, einen Befehl zu definieren, der eine Zeichenkette als Argument entgegennimmt und darstellt. Dafür muss die Folge der Argumente des Befehls `\newcommand*` ergänzt werden:

```
\newcommand*{\tir}[1]{\begin{large}\textbf{#1}\end{large}\xspace}%
% der Befehl \tir{...} mit 1 Argument gibt Tiernamen formatiert aus
```

In eckigen Klammern, nach dem ersten Pflichtargument mit dem Namen des neuen Befehls, steht als optionales Argument die Anzahl der Argumente, welche der neu definierte Befehl erhalten soll, hier: 1. Im zweiten Pflichtargument, mit der neuen Befehlsfolge, wird das Argument, welches der neu definierte Befehl erhalten soll, durch `#1` vertreten. Gäbe es im neu definierten Befehl zwei Argumente, würde das zweite durch `#2` vertreten, und so weiter. Oben definierter Befehl kann folgendermaßen angewendet werden.

Das `\tir{Schwein}` ist groß und fett. Das **Schwein** ist groß und fett.

Entscheidet man sich später, alle Tiernamen lieber klein und kursiv zu schreiben, braucht man nur einmal den *tir*-Befehl zu ändern. *Schwein* gehabt!

Es gibt Befehle, die in einer sogenannten *math*-Umgebung ausgeführt werden müssen (siehe Abschnitt 5.1 auf Seite 57), zum Beispiel die Darstellung des griechischen Buchstabens Rho ρ . Wenn man einen neuen Befehl definiert, der ein ρ zeigt, kann man mithilfe des Befehls `\ensuremath` erreichen, dass der neue Befehl sowohl innerhalb auch außerhalb einer *math*-Umgebung ausgeführt wird.

```
\newcommand*{\R}{\ensuremath{\rho}}%
% der Befehl \R zeigt das kleine Rho
...
$\R$ mit und ohne math-Umgebung \R
```

Das Beispiel zur Darstellung eines ρ ist so einfach, dass man keinen neuen Befehl bräuchte, aber nach dem Vorbild kann man schwierigere Befehlsketten einbauen.

1.3 Dokumentklassen und Pakete

Dokumentklassen erzeugen eine Grundeinstellung für das L^AT_EX-Dokument. Die Formatvorgabe einer Dokumentklasse (*document class*) befreit uns von der mühsamen Festlegung gestalterischer Einzelheiten und das Augenmerk kann sich aufs Wesentliche, den Inhalt, richten.

Eine Dokumentklasse wird mit dem Befehl `\documentclass[...]{...}` festgelegt. Es gibt ältere Standard-Dokumentklassen, die in den USA entwickelt wurden und neuere, sogenannte KOMA-Script-Klassen (deren Name mit *scr* beginnt), welche die in Deutschland übliche Schriftgestaltung unterstützen. Häufig verwendete Dokumentklassen sind *article* beziehungsweise *scrartcl* (kurzer Bericht), *report* beziehungsweise *scrreprt* (langer Bericht) und *book* beziehungsweise *scrbook* (Buch) sowie

beamer (Präsentation). Eine Entsprechung für die *beamer*-Klasse gibt es bei KOMA-Script nicht, aber man kann die Klasse *scrartcl* auch für Präsentationen einsetzen. Die gewünschte Dokumentklasse wird als Pflichtargument des *documentclass*-Befehls in geschweiften Klammern angegeben.

Mit optionalen Argumenten (im eckigen Klammerpaar) kann man die Gestaltung beeinflussen. Ein optionales Argument gibt das Papierformat an, in Deutschland ist das meistens *a4paper*, ein weiteres die Schriftgröße (in der Regel 10 pt, 11 pt oder 12 pt). In den meisten Büchern beginnen Absätze mit einer Einrückung. Alternativ können Absätze durch einen vertikalen Abstand getrennt werden. Dieses Verhalten lässt sich in den KOMA-Script-Klassen mit der Option *parskip=half* (halbzeiliger Abstand) oder *parskip=full* (ganzzeiliger Abstand) einstellen (siehe Abschnitt 1.5 auf Seite 14). Das optionale Argument *twocolumn* würde bewirken, dass die Seiten zwei Spalten bekommen. Will man Vorder- und Rückseite eines Blatts nutzen (zweiseitiger Druck), kann man *twoside* wählen. Beim zweiseitigen Druck werden unter anderem die inneren und äußeren Seitenränder angepasst.

In den Dokumentklassen *article/scrartcl* und *report/scrreprt* wird (ohne das optionale Argument *twoside*) für einseitigen Druck formatiert, in *book/scrbook* für zweiseitigen Druck. Die Klassen *report/scrreprt* und *book/scrbook* schreiben den Titel des Werks auf eine eigene Seite, bei *article/scrartcl* beginnt der weitere Text nach dem Titel auf der gleichen Seite. Ohne besondere Festlegung der Schriftgröße verwenden *article*, *report* und *book* die Größe 10 pt, während *scrartcl*, *scrreprt*, *scrbook* und *beamer* 11 pt nehmen.

Mehr über die KOMA-Script-Klassen gibt es in der [Anleitung von Markus Kohm](#). Nützlich ist auch die [Kurzanleitung für Abschlussarbeiten](#) mit Latex und KOMA-Script von Elke Schubert und Marion Lammarsch.

Pakete sind zusätzliche Teile des L^AT_EX-Systems, welche erst zur Verfügung stehen, wenn sie geladen wurden. Dies geschieht mit dem Befehl `\usepackage[...]{...}`. Pflichtargument ist der Name des zu ladenden Pakets. Optionale Argumente, die das Paket (*package*) beeinflussen, können in den eckigen Klammern gesetzt werden.

Ohne optionale Argumente können mehrere Paketnamen, durch Kommata getrennt, als Pflichtargumente in einem geschweiften Klammerpaar stehen.

Anpassung deutscher Texte geschieht mit folgenden Befehlszeilen. Damit werden Formatvorgaben nach deutschen Gepflogenheiten gesetzt. Die drei Befehlszeilen sollten darum für deutsche Texte immer in dieser Weise eingefügt werden.

```
\usepackage[ngerman]{babel}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
```

Das optionale Argument *ngerman* veranlasst die Verwendung der neuen deutschen Rechtschreibung, zum Beispiel bei der Silbentrennung. Die Kodierung mit UTF-8 erlaubt die direkte Eingabe deutscher Umlaute und so weiter. Mit *T1* werden westeuropäische Fonts (Schriften) verwendet.

Weitere Pakete werden wie in folgenden Beispielen geladen

```
\usepackage{graphicx}      % Einbindung von Bildern aus Bilddateien
\usepackage{xspace}        % Einsetzen eines Leerzeichens bei Bedarf
```

```
\usepackage{amsmath,amsthm,amsfonts} % Erweiterungen für Mathematik
\usepackage[normalem]{ulem} % Unterstreichen von Text, emph kursiv
\usepackage[left=3cm,right=3cm]{geometry} % Anpassung Seitenränder
```

L^AT_EX-Pakete und ihre Dokumentationen werden im CTAN (Comprehensive TeX Archive Network) gesammelt. Die Webseite ist <https://www.ctan.org/>

1.4 Beispiel

Folgender Quellcode zeigt die Struktur eines L^AT_EX-Dokuments. Einrückungen im Quellcode erhöhen seine Lesbarkeit, beeinflussen aber nicht das Druckbild.

```
% Präambel
\documentclass[a4paper, 12pt, parskip=half]{scrartcl}
  \usepackage[ngerman]{babel}
  \usepackage[utf8]{inputenc}
  \usepackage[T1]{fontenc}
  \usepackage[left=2cm,right=2cm,top=1.6cm,bottom=3cm]{geometry}
  \usepackage{booktabs,marvosym,xspace}

\newcommand*{\F}{\begin{Large}\textit{Freiheit}\end{Large}\xspace}%
% der Befehl ohne Argumente \F gibt das Wort Freiheit formatiert aus

% Dokument
\begin{document}

  \title{So heißt mein Werk}
  \author{Karl Mustermann\thanks{E-Mail: kamu@provider.de}}
  \date{Heute ist der \today}
  \maketitle

  \tableofcontents

\section{Erster Abschnitt}
  \subsection{ein Unterabschnitt}
    Text kann zum Beispiel \textbf{fett}
    oder \textit{kurvig} dargestellt werden.
    \par\bigskip
    Einsteins Formel:  $E = m \cdot c^2$ 

    \subsubsection{ein Unter-Unterabschnitt}
      \begin{tabular}{cccc}
        \toprule
        Cross & Frowny & Smiley & Heart \\
        \Cross & \Frowny & \Smiley & \Heart \\
        \bottomrule
      \end{tabular}

\section{Zweiter Abschnitt}
  Text kann auch \begin{Large}Groß\end{Large} und
  \begin{scriptsize}Klein\end{scriptsize} erscheinen.
  \par\bigskip\bigskip
  Die \F jedenfalls sollte groß geschrieben werden!

\end{document}
```

So heit mein Werk

Karl Mustermann*

Heute ist der 23. Juni 2017

Inhaltsverzeichnis

1 Erster Abschnitt	1
1.1 ein Unterabschnitt	1
1.1.1 ein Unter-Unterabschnitt	1
2 Zweiter Abschnitt	1

1 Erster Abschnitt

1.1 ein Unterabschnitt

Text kann zum Beispiel **fett** oder *kurvig* dargestellt werden.

Einsteins Formel: $E = m \cdot c^2$

1.1.1 ein Unter-Unterabschnitt

Cross	Frowny	Smiley	Heart
†	☹	☺	♥

2 Zweiter Abschnitt

Text kann auch Groß und klein erscheinen.

Die *Freiheit* jedenfalls sollte groß geschrieben werden!

*E-Mail: kamu@provider.de

Abbildung 1: Druckbild des Beispiels eines einfachen L^AT_EX-Dokuments

1.5 Gliederung des Dokuments

Ein einfaches Dokument ist als Ganzes in drei Teile gegliedert, nämlich Titel, Inhaltsverzeichnis und Text. Letzterer wird in Abschnitte unterteilt. Das Inhaltsverzeichnis wird, abhängig von den Textabschnitten, durch Kompilierung automatisch erzeugt.

Titel werden (abhängig von der Dokumentklasse) als Titelabschnitt oder Titelseite mit dem Befehl *maketitle* erzeugt, wenn zuvor Angaben zum Verfasser, zum Titel und zum Datum der Erstellung des Werks gemacht wurden. Beispiel:

```
\title{Noch eine \LaTeX\ Kurzeinführung}
\author{Alfred H. Gitter\thanks{E-Mail: alfred.gitter@eah-jena.de}}
\date{\today}
\maketitle
```

Mit dem Befehl `\LaTeX` wird das L^AT_EX-Logo dargestellt. Wenn man im *date*-Befehl den Parameter weglässt, wird kein Datum angezeigt. Mit `|today` wird das aktuelle Datum genommen. Alternativ kann man das Datum mit einer beliebigen Zeichenkette unmittelbar angeben. Mit dem Befehl *thanks* kann man im Titel eine Fußnote herstellen.

Textabschnitte sind die hierarchisch gegliederten Teile des Texts. Die Gliederung eines Textes erfolgt bei den Dokumentklassen *book/scrbook* und *report/scrreport* mit den Hierarchie-Ebenen (und entsprechenden Befehlen) *chapter* (Kapitel, Hierarchieebene 0), *section* (Abschnitt, Ebene 1), *subsection* (Unterabschnitt, Ebene 2) und *subsubsection* (Unterunterabschnitt, Ebene 3). Bei *article/scrartcl* fehlt die Kapitel-Ebene. Die Gliederung eines *article/scrartcl*-Dokuments in Abschnitte könnte (vereinfacht) ungefähr so aussehen:

```
\section{Überschrift des ersten Abschnitts}
\subsection{Überschrift des Unterabschnitts 1.1}
\subsection{Überschrift des Unterabschnitts 1.2}
\subsubsection{Überschrift des Unter-Unterabschnitts 1.2.1}
\section{Überschrift des zweiten Abschnitts}
```

Die Überschriften der Textabschnitte werden bis zu einer unteren Hierarchie-Ebene automatisch nummeriert. Welches die unterste, nummerierte Hierarchie-Ebene ist, hängt von der Dokumentenklasse ab. Man kann dies aber ändern. Der Befehl

```
\setcounter{secnumdepth}{-2}
```

in der Präambel bewirkt, dass keine Überschriften nummeriert werden. Setzt man in obigem Befehl statt -2 die Nummer einer Hierarchieebene ein, wird bis zu dieser nummeriert.

Soll ein bestimmter einzelner Abschnitt (oder Unter- oder Unterunterabschnitt) nicht nummeriert werden, fügt man einen Stern * an den Gliederungsbefehl an, zum Beispiel: `\section*{Überschrift}`. Der Abschnitt mit dem Stern wird nicht im Inhaltsverzeichnis (siehe unten) aufgeführt.

Für einen Anhang, der nach dem Haupttext folgt, gibt es den Befehl *appendix*. Alle Abschnitte, die nach diesem Befehl folgen, werden als Anhangsabschnitte angesehen und erhalten automatisch eine andere Nummerierung als die Abschnitte des Haupttexts:

```
\appendix
\section{erster Anhangsabschnitt}
\section{zweiter Anhangsabschnitt}
```

Die Untergliederung eines Unter-Unterabschnitts (*subsubsection*) kann mit einem *paragraph* genannten Textbereich erfolgen, welcher nicht nummeriert und nicht ins Inhaltsverzeichnis aufgenommen wird. Eine mögliche Überschrift, die als Parameter (in geschweiften Klammern) übergeben wird, erscheint fett am Anfang der ersten Zeile. Ohne Parameter (leere geschweifte Klammern) erhält der *paragraph* keine Überschrift.

Bei Verwendung der KOMA-Script-Klassen werden die Überschriften der Textabschnitte in serifenloser Schrift gesetzt. Will man dies nicht, kann man mit

```
\setkomafont{sectioning}{\normalfont\normalcolor\bfseries}
```

in der Präambel Überschriften mit Serifen erhalten.

Absätze können im Quellcode durch Einfügen einer Leerzeile oder mit dem Befehl `\par` erzeugt werden. Ein einfacher Zeilenumbruch genügt nicht.

Normalerweise beginnt ein Absatz mit einem (horizontalen) Einzug am Beginn der ersten Zeile, ohne vertikalen Abstand vom vorhergehenden Absatz. Ausgenommen von dieser Regel ist der erste Absatz nach einer Überschrift oder einer anderen Textunterbrechung (Leerzeile, Tabelle, Aufzählung und so weiter). Dieser beginnt ohne Einzug (stumpf). Die Länge des Einzugs der ersten Absatzzeile beträgt ein Geviert. (Geviert ist, abhängig vom Kontext, eine typographische Längen- oder Flächeneinheit.) Die letzte Zeile eines Absatzes kann sich bis zum rechten Rand erstrecken.

Sollen stattdessen Absätze durch einen vertikalen Abstand getrennt werden, kann man, bei den Dokumentklassen von KOMA-Script, nach dem Befehl *documentclass* in eckigen Klammern die Option *parskip=half* oder *parskip=full*, für halbzeiligen beziehungsweise ganzzeiligen Abstand, angeben. Zu beachten ist bei *parskip=half* und *parskip=full*, dass die letzte Zeile eines Absatzes mindestens um ein Geviert vom rechten Rand eingerückt wird. Damit wird das Absatzende besser erkennbar. Die Klassenoption *parskip* ist eine Besonderheit von KOMA-Script. Bei den Standard-Dokumentklassen benötigt man für den Absatzabstand das Paket *parskip*.

Die Einrückung am rechten Rand der letzten Zeile eines Absatzes tritt bei den Optionen *parskip=half* und *parskip=full* auch mit einzeiligen Absätzen auf. Umgebungen, die einen oder mehrere Kästen in einem einzeiligen Absatz unterbringen sollen (zum Beispiel Minipage, siehe unten), erzeugen dann eine Warnung (*overflow \hbox*). Man kann diese bei den Dokumentklassen von KOMA-Script verhindern, indem die Option *parskip* vorübergehend ausgeschaltet wird (siehe das Beispiel zur Umgebung *tcbraster* auf Seite 36 in Abschnitt 3.2).

Die Absatzauszeichnung sollte im gesamten Dokument entweder mit Einzug oder mit Abstand erfolgen. Die gleichzeitige Verwendung der beiden wäre übertrieben. Bei kleiner Zeilenlänge (zum Beispiel in mehrspaltigen Texten) ist der Absatzabstand besser, da er eine weitere Verkürzung der ersten Zeile verhindert. Ansonsten wird meistens der Absatzeinzug bevorzugt.

2 Besondere Teilbereiche

2.1 Verzeichnisse (Inhalt, Abbildungen, Literatur)

Das Inhaltsverzeichnis wird dort eingefügt, wo die Befehlszeile

```
\tableofcontents
```

steht. Welche Hierarchie-Ebenen des gegliederten Textes bei der Erstellung eines Inhaltsverzeichnisses einbezogen werden, hängt, sofern die diesbezügliche Voreinstellung nicht geändert wurde, von der Dokumentklasse ab. Mit *scrartcl* geht es bis zu den *subsubsections* (Unterunterabschnitten, Hierarchieebene 3) und mit *scrreport* bis zu den *subsections* (Unterabschnitten, Hierarchieebene 2). Man kann auch dies ändern. Der Befehl

```
\setcounter{tocdepth}{1}
```

in der Präambel bewirkt, dass nur die Kapitel und Abschnitte, aber keine Unterabschnitte (oder feinere Glieder) in das Inhaltsverzeichnis aufgenommen werden. Ersetzt man in obigem Beispiel die 1 durch eine andere Nummer, reicht das Inhaltsverzeichnis bis zur entsprechenden Hierarchieebene.

Jedes Mal, wenn man das Dokument mindestens zweimal kompiliert, wird ein aktuelles Inhaltsverzeichnis automatisch erzeugt. Aufgrund der Abhängigkeit des Inhaltsverzeichnisses von der Formatierung des gesamten Dokuments genügt die einmalige Kompilierung nicht.

Das Abbildungsverzeichnis wird mit der Befehlszeile

```
\listoffigures
```

eingefügt.

Es enthält die Legende oder, soweit vorhanden, die Kurzfassung der Legende, für alle Bilder, die in einer *figure*-Umgebung mit *caption*-Befehl (siehe Seite 19) stehen.

Das Literaturverzeichnis kann am einfachsten mit der *thebibliography*-Umgebung erstellt werden, die in der Regel am Ende des Dokuments steht. Normalerweise lautet die Überschrift des Literaturverzeichnisses *Literatur*. Man kann dies jedoch, vor der *thebibliography*-Umgebung, mit einem Befehl ändern.

```
\renewcommand{\bibname}{Meine Bücherliste}
```

Man beachte, dass in einigen Dokumentklassen *refname* statt *bibname* in diesem Befehl stehen muss.

Wenn man nichts anderes wünscht, lässt man die Einträge automatisch nummerieren. Der `\begin{thebibliography}`-Befehl erwartet in einem weiteren geschweiften Klammerpaar eine Zeichenkette; bei automatischer Nummerierung wird hier die maximale Zahl der Einträge eingetragen, zum Beispiel 99 oder 999.

Jeder Eintrag beginnt mit dem Befehl `\bibitem`, der in geschweiften Klammern eine kurze Zeichenkette erhält. Diese ist der Schlüssel, ein eindeutiger Name, mit dessen Hilfe der Eintrag im Text zitiert wird. Der Schlüssel sollte erkennbar zum Eintrag passen. Er könnte aus dem abgekürzten Namen des Autors (drei Kleinbuchstaben), dem Publikationsjahr (zweistellig) und einem Kleinbuchstaben, der die Eindeutigkeit sicherstellt (beginnend mit a), bestehen. Damit ergibt sich folgendes [Beispiel](#).

```
\begin{thebibliography}{99}
\bibitem{glu08a} Hans Glück, Der Elefant im Heuhaufen,
  Verlach, 6. Aufl., Buxtehude 2008.
\bibitem{glu08b} Hans Glück, Der Spatz in der Hand macht
  noch keinen Sommer, Verlach, 8. Aufl., Buxtehude 2008.
\bibitem{eol17a} Ethan Ol, Scotch your drinking problems,
  PUBLisher, Dufftown 2017.
\end{thebibliography}
```

Im Text werden Literatureinträge mit dem Befehl `\cite` zitiert, welcher in geschweiften Klammern einen oder mehrere (durch Kommata getrennte) Schlüssel erhält.

Sprichwörter sind Glücks Sache `\cite{glu08a,glu08b}`.

Zusätzlich kann, mit einem optionalen Parameter in eckigen Klammern, eine Information zum Literatureintrag, beispielsweise eine Seitenzahl, ergänzt werden.

Spiritual balance is a whisky in each hand `\cite[Seite 42]{eol17a}`.

Obige Beispiele mit Zitaten erscheinen gedruckt etwa so:

Sprichwörter sind Glücks Sache [1, 2].

Spiritual balance is a whisky in each hand [3, Seite 42].

Zitate im Text können anders formatiert werden, wenn das Paket [cite](#) von Donald Arseneau geladen wurde. Das Paket [tocbibind](#) von Peter Wilson erlaubt es, das [Literaturverzeichnis im Inhaltsverzeichnis](#) aufzuführen (mit oder ohne Nummerierung).

2.2 Fuß- und Randnoten, Querverweise, Hyperlinks

Fußnoten werden mit dem Befehl `\footnote` an eine Textstelle¹ gesetzt:

Die Flensburger Brauerei wurde 1888, im sogenannten
Dreikaiserjahr\footnote{1888 regierten nacheinander Wilhelm I.,
Friedrich III. und Wilhelm II.}, gegründet.

Die Fußnote wird automatisch nummeriert, ihr Text an das Ende der Seite (oder der folgenden Seite) geschrieben.

Randnoten werden in gleicher Weise (jedoch nicht nummeriert) mit dem Befehl `\marginpar` erstellt. Sie sollten aufgrund der geringen Randbreite kurz gehalten sein. Leider ist die automatische Silbentrennung beim ersten Wort des Randnotentexts ausgeschaltet. Um dennoch eine Silbentrennung zu erreichen, stellt man den Befehl `\hspace{0pt}` vor das erste Wort. Für eine linksbündige Randnote (statt Blocksatz) stellt man `\raggedright` an den Anfang. Zum Beispiel wird mit

¹1888 regierten nacheinander Wilhelm I., Friedrich III. und Wilhelm II.


```
\marginpar{ \raggedright\begin{large}\hspace{0pt}
Randnoten\end{large} sind keine Schand\~boten\,! }
```

die nebenstehende Randnote gebildet.

Rand-
noten
sind
keine
Schand-
boten!

Querverweise verweisen auf eine Zeichenkette im Text oder eine Abbildung oder Gleichung. Ein Querverweis auf eine Zeichenkette wird mit einer Marke ermöglicht, indem man direkt vor der Zeichenkette den Befehl `\label` mit einem willkürlich gewählten Parameter setzt. Für Abbildungen wird eine Marke (*Label*) beim Einfügen des Bilds in eine *figure*-Umgebung erzeugt (siehe Seite 105), für Gleichungen entsprechend (siehe Seite 65). In diesen und anderen Umgebungen sollte der Befehl `\label` ans Ende gesetzt werden.

Die Befehle `\ref` beziehungsweise `\pageref` und der Parametername erlauben es nun, auf Abschnitt beziehungsweise Seitenzahl der Zeichenkette (oder des Bildes oder der Gleichung) hinzuweisen. Beispiel:

```
\label{s1} Die Sonne ist ein Stern in der Milchstraße. Blablabla ...
Über Sterne sprachen wir in Abschnitt \ref{s1}, Seite \pageref{s1}.
```

Eventuell muss man das Dokument mehrfach kompilieren (siehe Abschnitt 2.1 auf Seite 15), damit die Querverweise im Text erscheinen.

Hyperlinks sind Querverweise in einem PDF-Dokument, welches nicht als Ausdruck, sondern (mit einem PDF-Betrachterprogramm) auf dem Bildschirm gelesen wird. Nach Anklicken mit der Maus wird der Leser direkt zum Verweisziel geführt. Wurde das Paket *hyperref* von Sebastian Rahtz und Heiko Oberdiek geladen, werden die Querverweise im L^AT_EX-Dokument zu Hyperlinks. Da *hyperref* einige Einstellungen anderer Pakete ändert, sollte es am Ende der Präambel (nach den anderen Paketen) eingebunden werden.

So kann man zu einer beliebigen (mit `\label` bezeichneten) Marke springen:

```
\usepackage{hyperref}
...
Die Maus\label{maus} ist ein unwillkommenes Haustier.
\par\medskip
blablabla
\newpage
Nützlich ist die Maus (siehe Seite \pageref{maus}) am Computer.
```

Hyperlinks werden auf dem Bildschirm (nicht dagegen im Papierausdruck) standardmäßig durch eine farbige Box gekennzeichnet. Man kann aber beim Laden des Pakets durch Angabe von Optionen das Aussehen der Hyperlinks selbst bestimmen.

Außerdem kann man mit einem Hyperlink auf eine Webseite verweisen. Mit

```
The \href{https://www.nationalbeefassociation.com/}{NBA}: no sports !
```

wird der Text *NBA* in normaler Schrift gezeigt und das Verweisziel (URL) erscheint nicht. Andererseits wird mit

```
Website of the NBA: \url{https://www.nationalbeefassociation.com/}
```

das Verweisziel in nichtproportionaler Schrift gezeigt.

Mithilfe des Pakets *hyperref* kann man auch Information zu Autor und Titel des von L^AT_EX erzeugten Dokuments im PDF-Dokument speichern. Zum Beispiel:

```
...
\usepackage{hyperref}
\hypersetup{ pdfauthor={A. Aguecheek},pdftitle={Beef up your wit !} }
...
```

Der Dokumententitel wird in der Titelleiste des PDF-Betrachterprogramms gezeigt.

2.3 Bilder, Tabellen, Balkendiagramme, Kommentare

Bilder können in das Dokument eingebettet werden, wenn sie in einer Datei vom Typ PNG, JPG oder PDF vorliegen. Da der Bildinhalt nicht in das L^AT_EX-Dokument übertagen wird, muss die Bilddatei bei jeder Kompilierung (siehe Abschnitt 2.1 auf Seite 15) vorhanden sein.

Der entscheidende Befehl für das Einfügen des Bildes aus einer Datei lautet

```
\includegraphics[optionen]{datei}
```

Dabei sind *datei* der Pfad zur Bilddatei und *optionen* Angaben zur Größe des Bildes oder ähnliches. In den *optionen* kann mit *scale=x* die Bildgröße auf das x-fache vergrößert beziehungsweise verkleinert werden, wobei für x der entsprechende Zahlenwert (zum Beispiel 0.5) eingesetzt wird. Stattdessen kann mit *width=x* oder *height=x* die gewünschte Breite oder Höhe des Bildes vorgegeben werden, wobei x eine Länge (zum Beispiel 8cm) ist. Bei der optionalen Größenänderung wird das Seitenverhältnis des Bildes beibehalten. Eine Verzerrung des Bildes durch gleichzeitige Vorgabe von Breite und Höhe ist nicht möglich. Es gibt weitere Optionen zur Veränderung des Bildes, zum Beispiel die Möglichkeiten des Drehens und Zuschneidens, aber derartiges kann besser mit einem Bildbearbeitungsprogramm vor der Einbindung in ein L^AT_EX-Dokument geschehen.

Wenn die Bilddatei nicht existiert, ergibt obiger Befehl jedoch einen Fehler und der Quellcode wird nicht kompiliert. Um die Fehlermeldung zu vermeiden, kann man den Befehl wie folgt ändern:

```
\IfFileExists{datei}
{ \includegraphics[optionen]{datei} } % Bilddatei ist da
{ \textcolor{red}{Datei fehlt} }      % Bilddatei fehlt
```

So wird in jedem Fall kompiliert. Fehlt die Bilddatei, wird anstelle des Bildes der kurze Text *Datei fehlt* in Rot geschrieben.

Eine systematische Einfügung von Bildern in ein Dokument, mit Bildunterschrift, automatischer Nummerierung der Abbildungen und optimierter Platzierung auf der Druckseite erreicht man mithilfe einer *figure*-Umgebung, die so gebildet wird:

```
\begin{figure}
\centering
\includegraphics[width=3cm]{veronica.jpg}
\caption{Langblättriger Ehrenpreis}
\label{veronica}
\end{figure}
```



Abbildung 2: Langblättriger Ehrenpreis

Damit das Bild in der Seitenmitte erscheint, wird am Anfang der *figure*-Umgebung der Befehl `\centering` gesetzt. Er ist ein Schalter (siehe Seite 7), der nur in der Umgebung wirkt, in der er aufgerufen wird. Im Gegensatz zu einer *center*-Umgebung fügt er keinen unerwünschten Leerraum ein.

Der Befehl `\caption{...}` erhält als Argument im geschweiften Klammerpaar die Bildunterschrift (Legende). Er kann als optionales Argument in eckigen Klammern (vor dem geschweiften Klammerpaar) eine Kurzfassung der Legende erhalten, welche nur im Abbildungsverzeichnis erscheint. Mit dem Befehl `\label{veronica}` wird eine Marke für Querverweise (siehe oben) gesetzt.

In Dokumentklassen mit kapitelweiser Nummerierung der Abbildungen erreicht man stattdessen eine durchgehende Nummerierung folgendermaßen in der Präambel.

```
\usepackage{chngcntr}
\counterwithout{figure}{chapter}
```

In Dokumentklassen, bei denen Abbildungen normalerweise abschnittsweise nummeriert werden, muss statt *chapter* in obigem Befehl *section* stehen.

Ein schmales Bild könnte links oder rechts von Text umflossen werden. Leider kann \LaTeX Gleitobjekte, wie eine *figure*-Umgebung, nicht gut mit einem Textumlauf versehen. Das Paket *wrapfig* von Donald Arseneau stellt die *wrapfigure*-Umgebung bereit, welche das im Wesentlichen leistet, aber manuelle Nachbesserung erfordert.

Im folgenden [Beispiel](#) wird mit dem ersten Pflichtargument *r* die Lage des Bildes (l oder L: links, r oder R: rechts) bestimmt. Mit Großbuchstaben (L oder R) wird eine vertikales Gleiten erlaubt, was aber oft nicht richtig gelingt. Das zweite Pflichtargument legt die Breite fest, die dem Bild zur Verfügung steht (mindestens die Bildbreite, die im *includegraphics*-Befehl angegeben wird).

... Adornos, eine reduzierte Farb- und Formsprache.
`\begin{wrapfigure}{r}{0.5\textwidth}`
`\vspace{-2mm}`
`\centering`
`\includegraphics[width=0.5\textwidth]{rebo}`
`\caption{Regenbogen}`
`\vspace{-3mm}`
`\end{wrapfigure}`
 Die Rückbesinnung auf wesentliche und ...

Das Werk zeigt, im Geiste der ästhetischen Moderne Adornos, eine reduzierte Farb- und Formsprache. Die Rückbesinnung auf wesentliche und symbolhafte Stilelemente ergibt ein spannungsreiches abstraktes Ensemble eines Weichbilds der Stadt ohne Menschen. Gelöst von perspektivischen Erwartungen und den tradierten Vorgaben ikonischer Räumlichkeit kontrastiert es unmanieriert mit einem Regenbogen surrealistischer Traumhaftigkeit. Klare Strukturierung gestaltet die ansprechende Synthese.



Abbildung 3: Regenbogen

Der Leerraum über und unter dem Bild und seiner Legende kann mit dem *vspace*-Befehl angepasst werden. Eine Alternative zur *wrapfigure*-Umgebung sind *Minipages* (siehe Abschnitt 3.2 auf Seite 37).

Mit dem Paket *copyrightbox* von Ives van der Flaas lässt sich ein kurzer Text direkt unter oder neben ein Bild platzieren, zum Beispiel ein Hinweis auf den Urheber. Zunächst wird in der Präambel die Schriftart festgelegt. Beim Aufruf des Befehls *copyrightbox*[...]{...}{...} wird im eckigen Klammerpaar die Platzierung des Texts bestimmt (b = unten, l = links, r = rechts). Im ersten geschweiften Klammerpaar wird das Bild eingefügt, im zweiten der Text genannt.

```
\usepackage{copyrightbox}
\makeatletter\renewcommand{%
\CRB@setcopyrightfont}{%
\footnotesize}\makeatother
...
\copyrightbox[b]
{\includegraphics[scale=0.37]
{veronica.jpg} }
{Urheber: A.H.\,Gitter}
```



Urheber: A.H. Gitter

In obigem Beispiel stehen die ersten vier Zeilen in der Präambel. In der vierten Zeile kann statt *\footnotesize* auch eine andere Schriftart festgelegt werden. Die letzten vier Zeilen stehen an der Stelle, wo das Bild eingefügt werden soll.

Eine Beschriftung von Bildern kann am einfachsten mit dem Paket *overpic* von Rolf Niepraschk erfolgen. Es stellt die *overpic*-Umgebung bereit. Die Anwendung wird in folgendem Beispiel anhand der Datei *alveole.jpg* gezeigt.

Zunächst lädt man, ähnlich wie beim *includegraphics*-Befehl, mit der *overpic*-Umgebung ein Bild aus einer Datei (hier: *alveole.jpg*) und skaliert es.

Um die Beschriftung zu positionieren legt man dann vorübergehend ein Gitternetz mit beschrifteten Achsen über das Bild. Dazu dienen die optionalen Argumente *grid* und *tics=10*. Die längere der beiden Achsen hat die Länge 100%. Der Wert, welcher *tics* zugewiesen wird (hier: 10) bestimmt den Abstand der Gitterlinien. Die Beschriftung kann mit allen graphischen Elementen erfolgen, die in einer *picture*-Umgebung möglich sind (siehe Seite 30 in Abschnitt 2.5). Hier wird Text mit dem

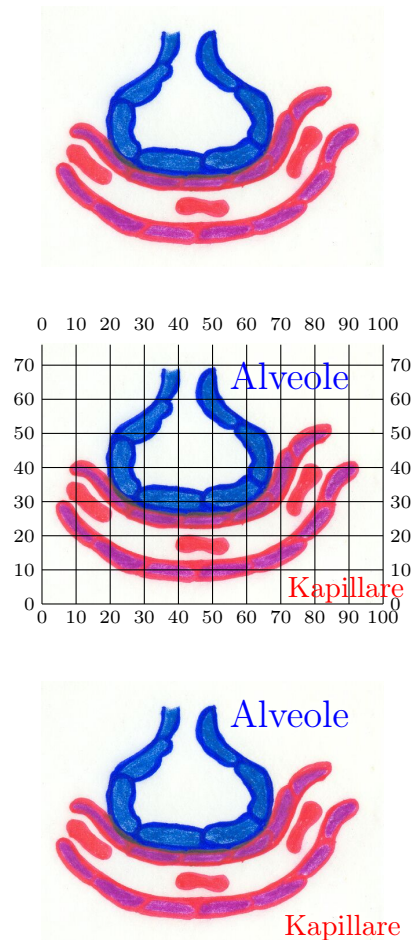
`put`-Befehl gesetzt. In runden Klammern stehen die Koordinaten des Textanfangs und dann folgt in geschweiften Klammern ein formatierter Text.

Nach der Platzierung der Beschriftung entfernt man das Gitternetz durch Löschung der optionalen Argumente `grid` und `tics=10`. Das beschriftete Bild bleibt über.

```
\begin{overpic}[scale=0.6]
{alveole.jpg}
\end{overpic}

\par\vspace{10mm}
\begin{overpic}[scale=0.6%
,grid,tics=10]{alveole.jpg}
\put(52,63)      {
\textcolor{blue}
{\large Alveole}    }
\put(69,2)       {
\textcolor{red}
{\small Kapillare}  }
\end{overpic}

\par\vspace{10mm}
\begin{overpic}[scale=0.6]
{alveole.jpg}
\put(52,63)      {
\textcolor{blue}
{\large Alveole}    }
\put(68,2)       {
\textcolor{red}
{\small Kapillare}  }
\end{overpic}
```



Wird, wie oben, farbiger Text verwendet, muss natürlich das Paket `xcolor` (oder `color`) geladen sein. Ein waagerechter Pfeil (Richtung (1,0)) der Länge 20%, ausgehend von den Koordinaten (10,60), könnte mit

`\thicklines \put(10,60){\vector(1,0){20}} % \thicklines: dicker Pfeil` eingefügt werden.

Wird der Skalierungsfaktor des Bilds nachträglich geändert, bleibt die relative Positionierung der Beschriftung bestehen, doch die Schriftgröße bleibt konstant.

Tabellen können als einfache Anordnung von Text in mehreren Zeilen gestaltet werden. Dafür kann man eine `tabbing`-Umgebung mit Tabulatoren (Sprungmarken innerhalb einer Zeile) verwenden. Zum [Beispiel](#) ergibt

```
\begin{tabbing}
XX\=XXXXXXXX\=XXXXXXXXX\=\kill
\>\>Einwohner\>Bundesland\[-1mm]
\>-----\\
\>Köln\>1,1 Millionen\>Nordrhein-Westfalen\\
```



```
\>München\>1,5 Millionen\>Bayern
\end{tabbing}
```

	Einwohner	Bundesland
Köln	1,1 Millionen	Nordrhein-Westfalen
München	1,5 Millionen	Bayern

In obigem Beispiel werden Tabulatoren mit dem Befehl `>=` in einer Zeile definiert, die nicht gezeigt und gedruckt werden soll und daher mit dem Befehl `\kill` abgeschlossen wird. In den folgenden Zeilen werden Tabulatoren mit dem Befehl `\>` aufgerufen und der Text dort fortgesetzt. Der automatische Zeilenumbruch ist in einer *tabbing*-Umgebung ausgeschaltet. Jede Zeile wird erst mit dem Befehl `\` beendet und der Text in der nächsten Zeile fortgesetzt. Einige Befehle (zum Beispiel `\hline`) können in einer *tabbing*-Umgebung nicht benutzt werden. Wenn man, wie oben gezeigt, eine horizontale Linie einfügt, kann die Option eines (positiven oder negativen) vertikalen Abstands beim `\`-Befehl nützlich sein.

Aufwendigere Tabellen können mit einer *tabular*-Umgebung erzeugt werden. In geschweiften Klammern werden zunächst Anzahl und Ausrichtung der Spalten festgelegt. Für jede Spalte wird mit einem Buchstaben die Ausrichtung bestimmt, wobei *c* für mittig, *l* für linksbündig und *r* für rechtsbündig steht. Für eine vertikale Trennlinie zwischen zwei Spalten wird ein `|` eingefügt. Dann werden die Zeilen und möglicherweise horizontale Trennstriche untereinander geschrieben. Die horizontalen Trennstriche werden durch den Befehl `\hline` erzeugt. In jeder Zeile werden benachbarte Tabellenfelder durch `&` getrennt. Jede Zeile wird mit `\` abgeschlossen. Mit

```
\begin{tabular}{|l|c|r|}
\hline
\multicolumn{2}{|c|}{Verein und Ort} & Tore\\
\hline
Bayern & München & 11\\
Werder & Bremen & 0\\
\hline
\end{tabular}
```

ergibt sich folgende Tabelle

Verein und Ort		Tore
Bayern	München	11
Werder	Bremen	0

Der Befehl `\multicolumn{2}{|c|}{Verein und Ort}` verbindet zwei Zellen, richtet die neue, große Zelle mittig aus (*c*) und fügt links und rechts vertikale Trennstriche an (`|c|`). Im dritten geschweiften Klammerpaar steht der Zelleninhalt.

Tabellen sehen besser aus, wenn sie nur wenige Linien enthalten. Insbesondere auf vertikale Linien kann man oft verzichten. Das Paket *booktabs* von Simon Fear und Danie Els stellt drei horizontale Linien für Tabellen zur Verfügung. Zum Beispiel erhält man mit

```
\begin{tabular}{lcr}
\toprule
\multicolumn{2}{c}{Verein und Ort} & Tore\\
\midrule
\end{tabular}
```

% erzeugt die obere horizontale Linie

% erzeugt eine horizontale Linie in der Tabelle

```

Bayern & München & 0\\
Werder & Bremen & 11\\
\bottomrule           % erzeugt die untere horizontale Linie
\end{tabular}

```

eine viel schönere Tabelle:

Verein und Ort		Tore
Bayern	München	0
Werder	Bremen	11

Tabellen mit fester Spaltenbreite können mit dem Parameter `p`, gefolgt von der Angabe der Spaltenbreite in geschweiften Klammern, erzeugt werden.

```

\begin{tabular}{|p{3cm}|p{3cm}|}
Mozart & Verdi\\
\midrule
La Nozze di Figaro & Aida\\
\end{tabular}

```

Mozart	Verdi
La Nozze di Figaro	Aida

Nachteilig ist, dass der Zelleninhalt stets im Blocksatz ausgerichtet wird. Will man zum [Beispiel](#) eine Tabelle mit zentriertem, links- oder rechtsbündigem Zelleninhalt, benötigt man weitere Pakete und eine Definition neuer Parameter für die *tabular*-Umgebung.

Das Paket [threeparttable](#) von Donald Arseneau ermöglicht eine Tabelle mit Überschrift und Anmerkungen. Zum [Beispiel](#) (mit Paket *booktabs*):

```

\begin{threeparttable}[c]
  \caption{Fußball\tnote{a}}
  \begin{tabular}{lcr}
    \toprule
    \multicolumn{2}{c}{Verein und Ort} & Tore\\
    \midrule
    Bayern & München & 0\\
    Werder & Bremen & 11\tnote{b}\\
    \bottomrule
  \end{tabular}
  \begin{tablenotes}
    \begin{small}
      \item [a] Endspiel DFB-Pokal
      \item [b] davon ein Eigentor
    \end{small}
  \end{tablenotes}
\end{threeparttable}

```

Tabelle 2.1: Fußball^a

Verein und Ort		Tore
Bayern	München	0
Werder	Bremen	11 ^b

^a Endspiel DFB-Pokal

^b davon ein Eigentor

Das optionale Argument des `-`Befehls bestimmt die vertikale Ausrichtung (`t` = top, `c` = center, `b` = bottom). Markierungen für Anmerkungen, die in der Überschrift gesetzt werden, erscheinen nicht im Tabellenverzeichnis. Soll die Tabelle gleiten, muss man die *threeparttable*-Umgebung noch in eine *table*-Umgebung setzen.

Eine Sonderform tabellarischer Anordnungen ist die logische Gliederung durch einseitige Klammern. Das Paket [schemata](#) von Charles P. Schaum ermöglicht zum [Beispiel](#) folgende Darstellungen.

```
\schema
{\schemabox{a}}
{\schemabox{b\\c\\d}}
```

$$a \left\{ \begin{array}{l} b \\ c \\ d \end{array} \right.$$

```
\schema[c]
{\schemabox{b\\c\\d}}
{\schemabox{a}}
```

$$\left. \begin{array}{l} b \\ c \\ d \end{array} \right\} a$$

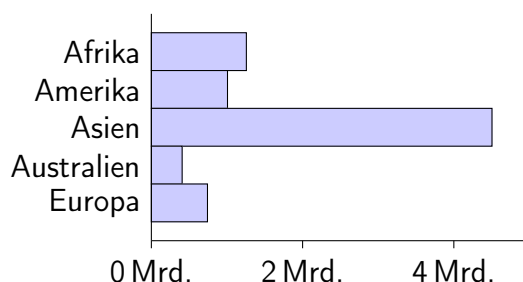
Pflichtargumente des *schema*-Befehls sind die Inhalte links und rechts der Klammer. Sie sollten in eine *schemabox* gepackt werden. Das optionale Argument *[c]* dreht die Klammer um. Innerhalb einer *schemabox* bewirkt `\\` den Beginn einer neuen Zeile. Die Klammergliederungen können verschachtelt werden.

```
\begin{center} \mbox{ \schema {
\schemabox
{
deutsche\\Millionen-\\städte} }
{
\schema {
\schemabox{im Westen:} }
{
\schemabox{
Hamburg\\München\\Köln
} } \bigskip
\schema {
\schemabox{im Osten:} }
{ \schemabox{Berlin} }
}
} \end{center}}
```

$$\begin{array}{l} \text{deutsche} \\ \text{Millionen-} \\ \text{städte} \end{array} \left\{ \begin{array}{l} \text{im Westen:} \left\{ \begin{array}{l} \text{Hamburg} \\ \text{München} \\ \text{Köln} \end{array} \right. \\ \text{im Osten:} \left\{ \begin{array}{l} \text{Berlin} \end{array} \right. \end{array} \right.$$

Balkendiagramme erhält man mit dem Paket *bchart* von Tobias Kuhn. Die Anpassungsmöglichkeiten sind beschränkt. [Beispiele](#):

```
\begin{bchart}[step=2,max%
=5,unit=\,Mrd.,width=5cm]
\bcbar[plain,label=Afrika]%
{1.256268}
\bcbar[plain,label=Amerika]%
{1.006801}
\bcbar[plain,label=Asien]%
{4.504428}
\bcbar[plain,label=Australien]%
{0.40691}
\bcbar[plain,label=Europa]%
{0.742074}
\end{bchart}
```



Das Diagramm wird in einer *bchart*-Umgebung erzeugt. Jeder Balken wird durch einen *bcbar*-Befehl gezeichnet. Die Option *plain* unterdrückt die Angabe des Zahlenwerts neben dem Balken.


```

\textbf{Elektrische Energie%
kosten}\[1mm]
\renewcommand{\bcfontstyle}%
{\rmfamily}
\begin{bchart}[max=0.4,%
plain,width=6cm]
\bcbar[text=\textbf{Polen},%
value=\SI{0.145}{\text{\EUR}%
\per\kWh}]{0.145} \smallskip
\bcbar[text=\textbf{Frankreich%
},value=\SI{0.176}{\text{\EUR}%
\per\kWh}]{0.176} \smallskip
\bcbar[text=\textbf{Deutschlan%
d},value=\SI{0.305}%
{\text{\EUR}\per\kWh}]{0.305}
\bcxlabel{Preis für Haushalte %
(Ende 2017)} \end{bchart}

```

Elektrische Energiekosten

Polen	0,145 €/kWh
Frankreich	0,176 €/kWh
Deutschland	0,305 €/kWh

Preis für Haushalte (Ende 2017)

Statt der vorgegebenen serifenlosen Schrift für das Balkendiagramm wurde durch Neudefinition des *bcfontstyle*-Befehls eine Schrift mit Serifen gewählt. Die Option *plain* der *bchart*-Umgebung unterdrückt die Skalierung der Achse. Mit der *value*-Option wird ein Text festgelegt, welcher (statt des Zahlenwerts der Balkenlänge) neben dem Balken erscheint. Das Dezimaltrennzeichen ist im Paket *bchart* ein Punkt, aber in Verbindung mit dem Paket *siunitx* (siehe Abschnitt 4.3 auf Seite 54) bekommt man ein Dezimalkomma. Für den *EUR*-Befehl zur Darstellung des Euro-Symbols braucht man das Paket *marvosym*. Mit *\smallskip* wird eine vertikale Lücke zwischen die Balken gesetzt. Der *bcxlabel*-Befehl beschriftet die Achse.

Kommentare sind Bereiche des Quelltexts, die in der gedruckten Ausgabe fehlen sollen. Es können Erläuterungen des Codes sein oder Quelltext, der vielleicht später gebraucht wird (und vorerst „auskommentiert“ wird).

Kurze Kommentare werden vom Steuerzeichen % eingeleitet und reichen nur bis zum Zeilenende. Längere Kommentare kann man in einer *comment*-Umgebung unterbringen, wenn das Paket *verbatim* von Rainer Schöpf geladen wurde. **Beispiel:**

```

Hallo % Kommentar
Welt! % noch einer

```

```

Hello,
\begin{comment}
not fit
to print
\end{comment}
World!

```

Hallo Welt!

Hello, World!

Zu beachten ist, dass man nach *\end{comment}* erst in der folgenden Zeile weiterschreiben kann. Auch am Ende einer *verbatim*-Umgebung (siehe Abschnitt 2.5, Seite 28) kann man nach *\end{verbatim}* erst in der folgenden Zeile weiterschreiben, wenn das Paket *verbatim* geladen wurde.

Will man in der tex-Datei einen beliebigen Abschnitt (mit Befehlen und Text) speichern, der nicht von PDFLatex verarbeitet und gedruckt wird, kann man ihn

einfach hinter den Befehl `\end{document}` (das heißt: außerhalb des Dokuments) setzen.

2.4 Aufzählungen und Theorem-Umgebungen

Aufzählungen listen mehrere Textteile (*items*) in einer Umgebung. Eine Aufzählungsumgebung kann beliebig viele *items* (Listenelemente) enthalten. Vor jedem *item* erscheint ein sogenanntes Label, bestehend aus einer Nummerierung (*enumerate*-Liste) oder einem Symbol (*itemize*-Liste) oder einer Zeichenkette (*description*-Liste). Aufzählungen können bis zu vier Ebenen tief geschachtelt werden. Eine geschachtelte *enumerate*-Liste entsteht zum [Beispiel](#) so:

```
\begin{enumerate}
  \item{good}                                1. good
  \item{not good}                             2. not good
    \begin{enumerate}
      \item{bad}                             a) bad
      \item{ugly}                            b) ugly
    \end{enumerate}
\end{enumerate}
```

Die Einrückungen vor den Befehlen sind nicht notwendig; sie dienen nur der Übersichtlichkeit und beeinflussen das Aussehen des Ausdrucks nicht. Statt `\item{Ding}` kann man auch `\item Ding` schreiben, also die geschweiften Klammern weglassen.

Soll die Zählung der Listenelemente nicht mit 1, sondern einer ganzen Zahl n beginnen, setzt man den Listenzähler *enumi* vorher auf $n - 1$, wie in diesem [Beispiel](#):

```
\begin{enumerate}
\setcounter{enumi}{6}
\item Es beginnt mit Sieben,                7. Es beginnt mit Sieben,
\item und endet mit der Acht.               8. und endet mit der Acht.
\end{enumerate}
```

Ebenso kann man eine *itemize*-Liste mit dem Befehl *itemize* erstellen. Dabei erscheint in der ersten Aufzählungsebene normalerweise das Symbol • (bullet) als Label vor jedem *item*. Möchte man ein anderes Symbol oder eine kurze Zeichenkette, gibt man diese als Option in eckigen Klammern nach dem `\item`-Befehl an. [Beispiel](#):

```
\begin{itemize}
  \item{bullets are weaker
        than ballots}                    • bullets are weaker than ballots
  \item[b)]{not a), that is
        no question here}                b) not a), that is no question here
  \item[$\star$]{stars are what
        men are made of}                 ★ stars are what men are made of
\end{itemize}
```

Eine *description*-Liste erzeugt Aufzählungen, deren Labels Zeichenketten sind. Zum [Beispiel](#):

<code>\begin{description}</code>	
<code>\item[Romeo] {Sohn der</code>	Romeo Sohn der Familie Montague
<code>Familie Montague}</code>	
<code>\item[Julia] {Tochter</code>	Julia Tochter der Familie Capulet
<code>der Familie Capulet}</code>	
<code>\end{description}</code>	

Die bisher behandelten Aufzählungen nehmen viel Platz ein, da vor und nach den Listen und zwischen den *items* Zeilenabstände eingefügt werden. Nach dem Laden des Pakets *paralist* von Bernd Schandl stehen weitere Listenumgebungen zur Verfügung (zum Beispiel die nummerierenden Aufzählungen *inparaenum* und *compactenum*), die ohne zusätzliche Zeilenabstände arbeiten. Bei ihnen kann die Art des Labels zu Beginn der Liste als Option verändert werden, wobei einige Zeichen eine besondere Bedeutung haben: a steht für Kleinbuchstaben, A für Großbuchstaben, I für römische und 1 für arabische Zahlen.

Eine *inparaenum*-Liste bettet die *items* in den laufenden Absatz ein. [Beispiel](#):

```
In der Revolution wurden \begin{inparaenum}[(a)]
\item{Freiheit,} \item{Gleichheit und} \item{Brüderlichkeit}
\end{inparaenum}von den Franzosen gefordert.
```

In der Revolution wurden (a) Freiheit, (b) Gleichheit und (c) Brüderlichkeit von den Franzosen gefordert.

Eine *compactenum*-Liste beginnt jedes *item* in einer neuen Zeile. [Beispiel](#):

Heute sind alternativlos	Heute sind alternativlos
<code>\begin{compactenum}[1.]</code>	
<code>\item{Sicherheit}</code>	1. Sicherheit
<code>\item{Korrektheit}</code>	2. Korrektheit
<code>\item{Untertänigkeit}</code>	3. Untertänigkeit
<code>\end{compactenum}</code>	

Entsprechend bilden die Listenumgebungen *compactitem* und *compactdesc* nicht nummerierende Aufzählungen (mit Symbol oder Zeichenkette als Label), welche keine Zeilenabstände einfügen.

Mehrspaltige Listen, zum [Beispiel](#) eine horizontale Auswahlliste mit Lösungsmöglichkeiten einer Übungsaufgabe, kann man mit dem Paket *tasks* von Clemens Niederberger bilden.

```
Wann fand die Seeschlacht von Lepanto statt\,?
\begin{tasks}(3)
\task{1560} \task{1571} \task{1588}
\end{tasks}
```

ergibt

Wann fand die Seeschlacht von Lepanto statt?

a) 1560	b) 1571	c) 1588
---------	---------	---------

Die Zahl in den runden Klammern (hier: 3) gibt die Anzahl der Spalten an, welche in diesem Beispiel mit der Anzahl der Listenelemente übereinstimmt.

Theorem-Umgebungen sind Gruppen von nummerierten, mit einheitlicher Überschrift versehenen Textbereichen besonderer Art, zum Beispiel mathematische Sätze oder Übungsaufgaben. Es sind im Wesentlichen ebenfalls Listen, obwohl die Listenelemente nicht zusammenhängen. Besonders wichtig sind Theorem-Umgebungen [in mathematischen Texten](#), aber auch in anderen Anwendungen können sie nützlich sein, wie folgendes [Beispiel](#) zeigt.

Zunächst wird in der Präambel der Typ der Theorem-Umgebung definiert. Der `newtheorem`-Befehl hat zwei Pflichtargumente, nämlich den Namen des Theorem-Typs im Quellcode (zum Beispiel `beisp`) und die einheitliche Überschrift für alle Theoreme (Listenelemente) dieses Typs.

```
\newtheorem{beisp}{Beispiel}
```

Theoreme des definierten Typs werden danach mit einer Umgebung gebildet, die den Namen des Theorem-Typs hat. Die Nummerierung erfolgt später automatisch.

Tetrapoda sind Wirbeltiere

mit vier Extremitäten.

```
\begin{beisp} Vögel sind
Tetrapoda, weil sie zwei
Beine und zwei Flügel haben.
\end{beisp}
```

Die Extremitäten können Beine, Arme oder Flügel sein.

```
\begin{beisp}[Gegenbeispiel]
Fische sind Wirbeltiere,
aber keine Tetrapoda.
\end{beisp}
```

Tetrapoda sind Wirbeltiere mit vier Extremitäten.

Beispiel 1. *Vögel sind Tetrapoda, weil sie zwei Beine und zwei Flügel haben.*

Die Extremitäten können Beine, Arme oder Flügel sein.

Beispiel 2 (Gegenbeispiel). *Fische sind Wirbeltiere, aber keine Tetrapoda.*

Die Überschrift wird fett geschrieben, der Textinhalt kursiv. Als optionales Argument der Theorem-Umgebung kann (in eckigen Klammern) eine einmalige Ergänzung zur einheitlichen Überschrift folgen (hier: *Gegenbeispiel*).

2.5 Programm-Code, Zeichnungen, Zähler

Programm-Code kann in ein \LaTeX -Dokument eingebettet werden, indem man ihn in eine *verbatim*-Umgebung einschließt. Gewöhnlich fügt diese aber vor- und nachher einen großen vertikalen Abstand ein. Wenn man diese Abstände verkleinern will, kann man das Paket [etoolbox](#) von Philipp Lehman und Joseph Wright laden. Es stellt unter anderem den `preto`-Befehl und den `AtBeginEnvironment`-Befehl zur Verfügung. Nun kann man in der Präambel (nach Laden des `etoolbox`-Pakets) zum [Beispiel](#) die Zeilen

```
\usepackage{etoolbox}
\makeatletter
\preto{\@verbatim}{\topsep=2mm \partopsep=0mm}
\makeatother
```

einfügen. Die Abstände zu den Zeilen vor und nach jeder *verbatim*-Umgebung werden so auf 2 mm verringert. Natürlich sind auch andere Werte möglich. Die Bedeutung der Befehle `\makeatletter` und `\makeatother` wird auf Seite 45 erklärt. Programmcode wird nun mit

Normaler Text. Programmcode:
`\begin{verbatim}`
 Hier steht der Programm-Code.
`\end{verbatim}`
 Es folgt wieder normaler Text.

Normaler Text. Programmcode:
 Hier steht der Programm-Code.
 Es folgt wieder normaler Text.

in normalen Text eingebettet. Der Text erscheint in einer `\verbatim`-Umgebung ohne Einrückung mit nichtproportionaler Schriftart. Zeilenumbrüche werden dargestellt. Innerhalb des Absatzes werden L^AT_EX-Befehle ignoriert (außer `\end{verbatim}`).

Die Befehle `\begin{verbatim}` und `\end{verbatim}` sollten jeweils allein in einer Zeile sein. Davor und danach sollte kein anderes Zeichen stehen, nicht einmal ein Leerzeichen. So vermeidet man Fehler, die unter bestimmten Bedingungen auftreten.

Hübscher kann ein Programmcode dargestellt werden, wenn das Paket *listings* von Jobst Hoffmann geladen wurde. Die Darstellungsweise wird mit dem `\lstset`-Befehl eingestellt und danach folgt Programm-Code in einer oder mehreren *lstlisting*-Umgebungen. In den geschweiften Klammern nach dem `\lstset`-Befehl wird unter anderem die verwendete Programmiersprache als Wert der Zuweisung `language=` angegeben (zum Beispiel bash, C, C++, Java oder Python).

Hier ein [Beispiel](#) für ein Python-Programm:

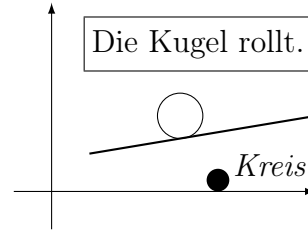
```
\lstset{
  language=Python, % gezeigt wird Python-Programm-Code
  tabsize=4, % Tabulator-Ersetzung durch vier Leerzeichen
  numbers=left, % Zeilennummern links
  xleftmargin=8mm, % Seitenrand links (für Zeilennummern)
  xrightmargin=8mm, % Seitenrand rechts (kleinerer Rahmen)
  numberstyle=\footnotesize\ttfamily, % Zeilennummernstil
  showstringspaces=false, % normale Leerzeichen im String
  frame=trbl, % Rahmen an Seiten top, right, left, bottom
  frameround=tttt % Abrundung der vier Ecken des Rahmens
}
\lstset{iterate=%
  {Ä}{{"A"}}1 {Ö}{{"O"}}1 {Ü}{{"U"}}1 % definiere Umlaute
  {ä}{{"a"}}1 {ö}{{"o"}}1 {ü}{{"u"}}1 % ÄÖÜäöü und das ß
  {ß}{{"ss"}}1 % für deutschen Kommentar
}
\begin{lstlisting}
#!/usr/bin/python3
# Programm zur Begrüßung
text = input("Ihr Name: ")
if text:
print("Hallo "+text)
\end{lstlisting}
```

Damit ergibt sich folgende Darstellung:

```
1 #!/usr/bin/python3
2 # Programm zur Begrüßung
3 text = input("Ihr Name: ")
4     if text:
5 print("Hallo "+text)
```

Zeichnungen können in einer *picture*-Umgebung erstellt werden. Ohne zusätzliche Pakete sind die Möglichkeiten aber begrenzt und man sollte für umfangreichere Zeichnungen das Paket *tikz* (siehe unten) verwenden. [Beispiel](#):

```
\setlength{\unitlength}{1cm}
\begin{picture}(5,4)(-1,-0.9)
\put(-0.5,0){\vector(1,0){4}}
\put(0,-0.5){\vector(0,1){3}}
\thicklines % dickere Linien
\put(0.5,0.5){\line(6,1){3}}
\thinlines % dünnere Linien
\put(1.7,1){\circle{0.6}}
\put(2.2,0.15){\circle*{0.3}}
\put(2.4,0.2){\textit{Kreis}}
\put(0.3,1.6){\framebox(3,0.7)
{Die Kugel rollt.} }
\end{picture}
```

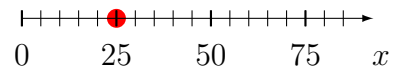


Der erste Befehl setzt die Längeneinheit fest. Danach werden die Längen ohne Einheit geschrieben. Durch Änderung der Längeneinheit lässt sich die Zeichnung vergrößern oder verkleinern. Der Befehl `\begin{picture}(5,4)(-1,-0.9)` legt im ersten runden Klammerpaar Breite und Höhe der Zeichenfläche fest und im zweiten Klammerpaar die Koordinaten der linken unteren Ecke. Dann werden mit dem *put*-Befehl verschiedene Objekte platziert. Im runden Klammerpaar stehen die Koordinaten.

Bei Linien und Vektoren (Pfeile) wird im ersten (runden) Klammerpaar die Steigung durch zwei teilerfremde Zahlen (im Bereich -6 bis 6 beziehungsweise -4 bis 4 für Vektoren) bestimmt und im geschweiften Klammerpaar die Länge. Ab dem Befehl `\thicklines` wird die Linienstärke erhöht und nach `\thinlines` wieder zurückgesetzt. Bei Kreisen (ohne ***) und gefüllten Kreisen (mit ***) wird im Pflichtargument der Durchmesser angegeben. Text, der mit dem *put*-Befehl platziert wird, kann formatiert werden und auch eine *math*-Umgebung enthalten. Bei einem gerahmten Kasten mit Text, *framebox*, werden in einem runden Klammerpaar Breite und Höhe angegeben und in einem geschweiften Klammerpaar der Text.

Das Paket [coordsys](#) von Mogens Lemvig Hansen liefert Befehle, die unter anderem eine skalierte Achse in einer *picture*-Umgebung zeichnen.

```
\setlength{\unitlength}{0.5mm}
\begin{picture}(100,20)(-5,-10)
\put(25,0){\textcolor{red}{\circle*{5}}}
\numblineline[5]{0}{93}
\put(95,0){\sethlabel{x}}
\end{picture}
```



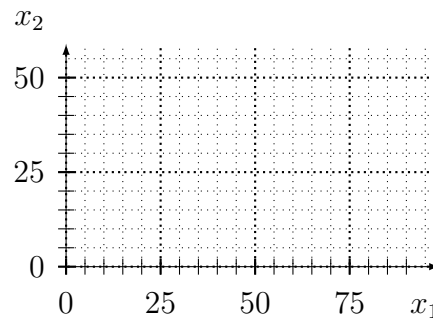
Der rote Kreis in obigem Beispiel erfordert das Paket *color* oder *xcolor*. Der Befehl *numblineline* erhält im optionalen Argument den Skalenteilungswert (Abstand zweier Teilstriche). Die Pflichtargumente geben Anfang und Ende der Achse an. Mit dem *sethlabel*-Befehl wird ein Text für die x-Achsenbeschriftung geschrieben. Er wird automatisch in eine *math*-Umgebung (siehe Seite 57) gesetzt. Die entsprechenden Befehle für eine vertikale Achse sind *numblineline* und *setvlabel*.

Auch ein ebenes kartesisches Koordinatensystem kann gezeichnet werden.


```

\setlength{\unitlength}{0.5mm}
\begin{picture}(110,88)(-10,-15)
\coordsys[5][5](0,0)(99,59)
\coordgrid[5][5](0,0)(99,59)
\put(0,0){\sethlabel{0}}
\put(0,0){\setvlabel{0}}
\put(95,0){\sethlabel{x_1}}
\put(0,65){\setvlabel{x_2}}
\end{picture}

```



Der *coordsys*-Befehl zeichnet die horizontale und die vertikale Achse mit Beschriftung. Er erhält in den optionalen Argumenten die Skalenteilungswerte für die beiden Achsen. Die Pflichtargumente geben Anfang und Ende der Achsen an. Der *coordgrid*-Befehl zeichnet ein Gitternetz. Er hat die gleichen Argumente wie der *coordsys*-Befehl. In obigem Beispiel wurde die Achsenbeschriftung mit der Marke 0 in horizontaler und vertikaler Achse ergänzt. Da das Argument des *sethlabel*-Befehls automatisch in eine *math*-Umgebung (siehe Seite 57) gesetzt wird, kann man Indizes mit $_1$ beziehungsweise $_2$ anhängen.

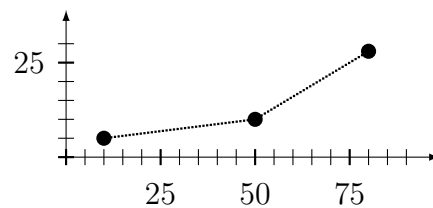
Logarithmische Achsen und Koordinatensysteme mit logarithmischen Achsen erzeugt man in ähnlicher Weise mit dem Paket *logsys* von Mogens Lemvig Hansen.

Das Paket *epic* von Sunil Podar stellt ein paar Erweiterungen für die *picture*-Umgebung bereit. Insbesondere kann man gepunktete Linien zwischen beliebigen Koordinatenpunkten ziehen. [Beispiel](#) (mit Paketen *coordsys* und *epic*):

```

\setlength{\unitlength}{0.5mm}
\begin{picture}(120,70)(-10,-20)
\coordsys[5][5](0,0)(99,39)
\put(10,5){\circle*{4}}
\put(50,10){\circle*{4}}
\put(80,28){\circle*{4}}
\linethickness{0.3mm}
\dottedline(10,5)(50,10)(80,28)
\end{picture}

```



Der *dottedline*-Befehl mit einer Folge von Koordinatenpaaren (in runden Klammern) zeichnet eine gepunktete Linie, deren Stärke vom vorhergehenden *linethickness*-Befehl gegeben wird. Das Argument des *linethickness*-Befehls ist eine Längenangabe mit Einheit (mm). Viel schönere Graphiken kann man mit den aufwändigeren Paketen *tikz* und *pgfplots* erzeugen (siehe unten).

Zähler sind Variablen in \LaTeX , die eine ganze Zahl speichern. Für viele Textbereiche, beispielsweise Abschnitte oder Gleichungen (siehe unten) sind besondere Zähler bereits definiert. Außerdem kann man eigene Zähler definieren.

Die Definition eines eigenen Zählers, hier *zehler* genannt, geschieht am besten bereits in der Präambel, kann aber auch noch in der *document*-Umgebung erfolgen. Mit seiner Definition wird der Zähler auf 0 gesetzt. Durch den Befehl *\setcounter* kann der Zähler auf einen beliebigen ganzzahligen Wert gesetzt werden und mit dem Befehl *\stepcounter* wird der Wert des Zählers um 1 erhöht. Zum Schreiben des aktuellen Zählerwerts in den Text dient der Befehl *\thezehler*, worin an *the* der Name des Zählers gehängt wurde. Will man den Wert des Zählers in einer

bedingten Anweisung verwenden, ohne ihn in den Text zu schreiben, ruft man den Wert des Zählers mit `\value` ab. Im folgenden [Beispiel](#) wird zweimal der Befehl `\ifnum` benutzt, um Text in Abhängigkeit vom Wert des Zählers zu schreiben.

```
\newcounter{zehler}
Zählerwert: \thezehler
```

```
\setcounter{zehler}{3}
\ifnum\value{zehler}=3
Zähler ist gleich 3
\fi
```

```
Zählerwert: 0
Zähler ist gleich 3
Zählerwert: 4
Zähler kleiner als 10
```

```
\stepcounter{zehler}
Zählerwert: \thezehler

\ifnum\value{zehler}<10
Zähler kleiner als 10
\else
Zähler größergleich 10
\fi
```

Der Befehl `\ifnum` erwartet einen Wert (hier: der des Zählers), ein Operatorzeichen (= oder < oder >), einen ganzzahligen Vergleichswert, dann den Text, der geschrieben wird, wenn der Vergleich wahr ist, danach möglicherweise ein `\else` und ein Text, der geschrieben wird, wenn der Vergleich unwahr ist, und als Abschluss `\fi`.

3 Seite, Absatz und Wort

3.1 Seitengestaltung, mehrere Spalten

Seitenränder sind manchmal zu groß (oder zu klein) voreingestellt. Mit dem Paket *geometry* von Hideo Umeki kann man die Seitenränder auf neue Werte setzen.

```
\usepackage[left=3cm,right=3cm,top=2cm,bottom=3cm]{geometry}
```

Natürlich sind auch andere Werte möglich als die im Beispiel angegebenen.

Kopf- und Fußzeile kann man mit dem Befehl `\pagestyle{...}` beeinflussen, der in der Präambel gegeben wird. Dies betrifft auch die Darstellung von Seitenzahlen. Ohne zusätzliche Pakete laden zu müssen, kann man in den geschweiften Klammern die Optionen *plain*, *empty* oder *headings* angeben. Fehlt der *pagestyle*-Befehl, entspricht dies `\pagestyle{plain}` und die Seitenzahl erscheint in der Fußleiste. Mit der Option *empty* werden keine Seitenzahlen gesetzt.

Die Wirkung von *headings* hängt von der verwendeten Dokumentklasse (und den zugehörigen Optionen) ab. Bei der Dokumentklasse *scrartcl*, ohne die Option *twoside*, erscheint der Name des Abschnitts in der Mitte der Kopfzeile und die Seitenzahl in der Mitte der Fußzeile.

Bei der Dokumentklasse *scrartcl*, mit der Option *twoside*, erscheint links in der Kopfzeile der linken (geraden) Seiten des (zweiseitig gedruckten) Dokuments der aktuelle Abschnitt (außer auf der ersten Seite eines Abschnitts) und rechts in der Kopfzeile der rechten (ungeraden) Seiten der aktuelle Unterabschnitt (falls ein solcher im Abschnitt bereits aufgemacht wurde). In den Fußzeilen steht (links auf den linken Seiten beziehungsweise rechts auf den rechten Seiten) die Seitenzahl.

Bei der Dokumentklasse *scrreprt*, ohne die Option *twoside*, erscheint der Name des Kapitels in der Mitte der Kopfzeile (außer auf der ersten Seite des Kapitels) und die Seitenzahl in der Mitte der Fußzeile.

Bei der Dokumentklasse *scrreprt*, mit der Option *twoside*, sowie bei der (zweiseitig gedruckten) Dokumentklasse *scrbook*, erscheint links in der Kopfzeile der linken (geraden) Seiten des (zweiseitig gedruckten) Dokuments das aktuelle Kapitel (außer auf der ersten Seite eines Kapitels) und rechts in der Kopfzeile der rechten (ungeraden) Seiten der aktuelle Abschnitt (falls ein solcher im Kapitel bereits aufgemacht wurde). In den Fußzeilen steht (links auf den linken Seiten beziehungsweise rechts auf den rechten Seiten) die Seitenzahl.

Bei der (zweiseitig gedruckten) Dokumentklasse *book* erscheint in der Kopfzeile der linken (geraden) Seiten links die Seitenzahl und rechts das aktuelle Kapitel. In der Kopfzeile der rechten (ungeraden) Seiten steht links der aktuelle Abschnitt (falls ein solcher im Kapitel bereits aufgemacht wurde) und rechts die Seitenzahl. Die Fußzeile bleibt hier leer. Eine Ausnahme ist die erste Seite eines Kapitels: die Kopfzeile bleibt leer und in der Fußzeile steht mittig die Seitenzahl.

Drehen von Textabschnitten ist mit dem Paket *pdfscape* von Heiko Oberdiek möglich. Was in einer *landscape*-Umgebung steht, wird um 90° gedreht. Bei den Seiten mit gedrehtem Inhalt bleiben Kopf- und Fußzeile unbeeinflusst, werden also nicht mitgedreht. Dies Verhalten ist beim Ausdruck der Seiten erwünscht. Bei der Darstellung im PDF-Betrachterprogramm werden die Seiten mit gedrehtem Inhalt im Querformat dargestellt. So kann man den Inhalt am Bildschirm besser lesen. Das Drehen um 90° kann insbesondere bei großen Tabellen hilfreich sein, die dann besser auf eine Seite passen.

scrlayer-scrpage und fancyhdr sind Pakete, welche vielfältige Gestaltungsmöglichkeiten für Kopf- und Fußzeilen bereitstellen.

Das Paket *scrlayer-scrpage* von Markus Kohm ist an die KOMA-Script-Klassen (*scrartcl*, *scrreprt*, *scrbook*) angepasst. Nach dem Laden desselben kann man dem *pagestyle*-Befehl die Option *scrheadings* übergeben:

```
\documentclass{ ... }
\usepackage{scrlayer-scrpage}
\pagestyle{scrheadings}
...
```

Ohne weitere Befehle des Pakets *scrlayer-scrpage* erscheinen

a) bei den Dokumentklassen *scrartcl* und *scrreprt*, jeweils ohne die Option *twoside*, die Seitenzahlen mittig in der Fußzeile und die Kopfzeile bleibt leer.

b) bei den Dokumentklassen *scrartcl* und *scrreprt*, jeweils mit der Option *twoside*, die Seitenzahlen in der Fußzeile links (auf linken Seiten) oder rechts (auf rechten Seiten) und die Kopfzeile bleibt leer.

c) bei den Dokumentklassen *scrbook* und *book* die Kopf- und Fußzeilen so wie ohne das Paket *scrlayer-scrpage* und mit `\pagestyle{headings}`, siehe oben. Ausnahme: Bei der Dokumentklasse *book* erscheint auf der ersten Seite eines Kapitels keine Seitenzahl.

Mit weiteren Befehlen des Pakets *scrlayer-scrpage* lassen sich Kopf- und Fußzeile fein und umfangreich gestalten. Näheres entnehme man der [Paketbeschreibung](#).

Das Paket *fancyhdr* von Piet van Oostrum ist nicht an die KOMA-Script-Klassen angepasst und eignet sich für die älteren Standard-Dokumentklassen *article*, *report* und *book*.

Wasserzeichen können den Seiten hinzugefügt werden, um kenntlich zu machen, dass nur ein Entwurf (Englisch: draft) vorliegt, oder um die Urheberschaft des Texts festzuhalten. Geeignet hierfür ist das Paket *draftwatermark* von Sergio Callegari. In folgendem [Beispiel](#) wird durch Befehle des Pakets *draftwatermark* in der Präambel der Wasserzeichentext bestimmt, sowie die Helligkeit (0 = schwarz, 1 = weiß) und die Größe des Wasserzeichentexts.

```
\SetWatermarkText{Wasserzeichentext} \SetWatermarkLightness{0.95}
\SetWatermarkFontSize{24.88pt}       \SetWatermarkScale{2.8}
```

Eine Helligkeit von 0.95 bewirkt, dass der Wasserzeichentext fast unsichtbar ist.

Mehrspaltige Bereiche innerhalb eines Dokuments, welches ansonsten einspaltig gesetzt ist, kann man unter anderem mit dem Paket *multicol* von Frank Mittelbach

erzeugen. Das Paket stellt eine *multicols*-Umgebung bereit, die als Pflichtargument die Anzahl der Spalten hat. In der Präambel können, nach dem Laden des Pakets, einige Voreinstellungen geändert werden.

```
\setlength{\columnsep}{0.8cm}           % Spalten-Abstand
\setlength{\columnseprule}{0.4pt}       % Dicke der Spaltentrennlinie
\def\columnseprulecolor{\color{blue}}   % Farbe der Spaltentrennlinie
```

Ohne den zweiten Befehl gibt es keine Spaltentrennlinie. Für eine farbige Spaltentrennlinie wird das Paket *color* oder *xcolor* benötigt. Gleitobjekte, zum Beispiel eine *figure*-Umgebung, und Randnoten sind in den Spalten allerdings nicht erlaubt. Ein [Beispiel](#) zeigt die Anwendung:

```
\begin{multicols}{2}
Im Anfang war das Wort
und das Wort war bei
Gott, und das Wort war
Gott. Im Anfang war es
bei Gott. Alles ist
durch das Wort geworden
und ohne das Wort wurde
nichts, was geworden ist.
\end{multicols}
```

<p>Im Anfang war das Wort und das Wort war bei Gott, und das Wort war Gott. Im Anfang war es bei Gott. Alles ist durch das Wort geworden und ohne das Wort wurde nichts, was geworden ist.</p>	<p>bei Gott. Alles ist durch das Wort geworden und ohne das Wort wurde nichts, was geworden ist.</p>
--	--

3.2 Rahmen und Minipages

Rahmen umranden einen rechteckigen Textbereich. Ein kurzer Text ohne Zeilenumbrüche kann mit dem *fbox*-Befehl gerahmt werden. Der Text kann natürlich formatiert werden. Mit dem Paket *fancybox* von Timothy Van Zandt erhält man in gleicher Weise andere Arten von Rahmen. [Beispiele](#):

<code>\fbox{\textit{abusus non tollit usum}}</code>	<div style="border: 1px solid black; padding: 2px; display: inline-block;"><i>abusus non tollit usum</i></div>
<code>\ovalbox{barba crescit caput nescit}</code>	<div style="border: 1px solid black; border-radius: 10px; padding: 2px; display: inline-block;">barba crescit caput nescit</div>
<code>\Ovalbox{\textbf{quod scripsi, scripsi}}</code>	<div style="border: 2px solid black; border-radius: 10px; padding: 2px; display: inline-block;">quod scripsi, scripsi</div>
<code>\doublebox{\textbf{quis leget haec??}}</code>	<div style="border: 3px double black; padding: 2px; display: inline-block;">quis leget haec?</div>
<code>\shadowbox{mors certa, hora incerta}</code>	<div style="border: 2px solid black; padding: 2px; display: inline-block;">mors certa, hora incerta</div>

Das Paket *framed* von Donald Arseneau, welches das Paket *xcolor* von Dr. Uwe Kern benötigt, enthält die *framed*-Umgebung. Damit können auch mehrzeilige Texte, mit automatischem Zeileneinbruch, gerahmt werden.

```
\begin{framed}
Silver was gone. But this was
not all. The sea-cook had not
gone empty-handed. He had cut
through a bulkhead unobserved
and had removed one of the
sacks of coin, worth perhaps
three or four hundred guineas.
\end{framed}
```

<p>Silver was gone. But this was not all. The sea-cook had not gone empty-handed. He had cut through a bulkhead unobserved and had removed one of the sacks of coin, worth perhaps three or four hundred guineas.</p>	<p>Silver was gone. But this was not all. The sea-cook had not gone empty-handed. He had cut through a bulkhead unobserved and had removed one of the sacks of coin, worth perhaps three or four hundred guineas.</p>
---	---

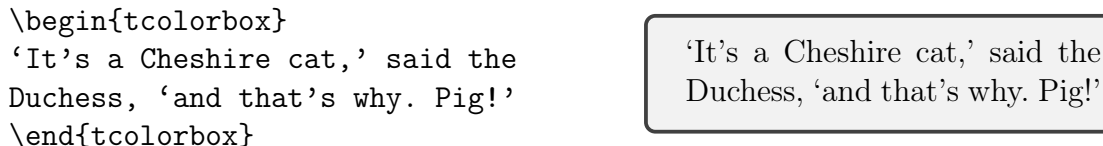
Die *leftbar*-Umgebung zeichnet eine senkrechte Linie links neben den Text.

```
\begin{leftbar}
Oxen and wain-ropes would
not bring me back again to
that accursed island.
\end{leftbar}
```



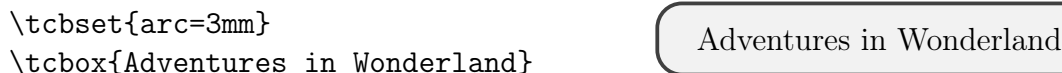
Vielfältige Möglichkeiten zur Gestaltung von Rahmen bietet das Paket *tcolorbox* von Thomas F. Sturm. Die Umgebung *tcolorbox* zeichnet ohne weitere Befehle einen schwarzen, abgerundeten Rahmen in der aktuellen Zeilenbreite (*\linewidth*).

```
\begin{tcolorbox}
‘It’s a Cheshire cat,’ said the
Duchess, ‘and that’s why. Pig!’
\end{tcolorbox}
```



Der *tcbbox*-Befehl zeichnet einen passenden Rahmen um den kurzen Text, der als Pflichtargument übergeben wird. Die Rundung der Ecken kann vorher mit dem Befehl *\tcbset{arc=...}* verändert werden, wobei ... den Innenradius der Bögen angibt (zunächst 1 mm). Der Befehl *\tcbset{colframe=red}* würde einen roten Rahmen ergeben und *tcbset{colback=red!10!white}* einen rötlichen Hintergrund.

```
\tcbset{arc=3mm}
\tcbbox{Adventures in Wonderland}
```



Mit dem Befehl *\tcbuselibrary{raster}* lädt man in der Präambel die Bibliothek *raster* nach dem Paket *tcolorbox*. Sie stellt die Umgebung *tcbbraster* zur Verfügung. Damit kann man eine zusammenhängende Gruppe tabellarisch (in Zeilen und Spalten) angeordneter, automatisch nummerierter Rahmen erzeugen. Sie haben den gleichen Titel, bis auf die laufende Nummer des Rahmens. [Beispiel](#):

```
% nur bei KOMA-Script-Klassen mit der Option parskip (half/full)
%\KOMAoption{parskip}{no}
% Rahmenraster mit 3 Spalten, alle Rahmen haben gleiche Höhe
\begin{tcbbraster}[raster columns=3, raster equal height,
% Abstand der Rahmen voneinander
raster equal skip=6mm,
% Abstände zwischen Rahmen und Titel
bottomtitle=1mm, toptitle=1mm, lefttitle=1mm, righttitle=1mm,
% Abstände zwischen Rahmen und Text
bottom=1.5mm, top=1.5mm, left=1mm, right=1mm,
% Farbe des Rahmens und des Texthintergrunds
colframe=blue,colback=blue!10!white,
% Farbe des Titelhintergrunds
colbacktitle=blue!15!white,
% Farben des Titels und des Texts
coltitle=black,coltext=blue,
% Einheitlicher Titel und laufende Titelnummer (fett)
title={Platonischer Körper \textbf{\thetcbbrasternum}}]
% Textinhalt aller Rahmen, der Reihe nach in einer Umgebung
\begin{tcolorbox}Tetraeder\end{tcolorbox}% 1. Rahmentext
```

```
\begin{tcolorbox}Hexaeder,\\auch %           2. Rahmentext
Würfel genannt\end{tcolorbox}%             Fortsetzung
\begin{tcolorbox}Oktaeder\end{tcolorbox}%    3. Rahmentext
\begin{tcolorbox}Dodekaeder\end{tcolorbox}%  4. Rahmentext
\begin{tcolorbox}Ikosaeder\end{tcolorbox}%   5. Rahmentext
\end{tcbraster}

% nur bei KOMA-Script-Klassen mit der Option parskip (half/full)
%\KOMAOPTION{parskip}{half} % eventuell full statt half
```

ergibt eine Tabelle der regulären, konvexen dreidimensionalen Polyeder.

Platonischer Körper 1	Platonischer Körper 2	Platonischer Körper 3
Tetraeder	Hexaeder, auch Würfel genannt	Oktaeder
Platonischer Körper 4	Platonischer Körper 5	
Dodekaeder	Ikosaeder	

Minipage ist eine Umgebung, die einen Kasten bereitstellt, welcher sich fast wie eine kleine Seite verhält. Allerdings dürfen Minipages keine Gleitobjekte (*figure*, *table*, *lstlisting*) oder Randnoten (*marginpar*) enthalten.

Oft werden zwei Minipages auf der tatsächlichen Seite nebeneinander platziert. So kann man ein Bild, eine Tabelle oder eine Zeichnung neben einen Textabschnitt stellen. Pflichtargument ist die Breite, welche meistens relativ zur Textbreite angegeben wird. Die Summe der Breiten zweier nebeneinander platzierter Minipages darf natürlich die Textbreite nicht überschreiten. **Beispiel:**

```
Klimaforscher ist der älteste
Beruf der Welt.\\[5mm]
\begin{minipage}
{0.6\textwidth}
```

Die Rekonstruktion der Erde vor 120000 Jahren zeigt, wie böse Urmenschen den Planeten erwärmten und so die armen Eisbären in der Eem-Warmzeit schwitzten.

```
\end{minipage}%
```

```
\hfill
```

```
\begin{minipage}
```

```
{0.37\textwidth}
```

```
\includegraphics[%
width=3cm]{erde.png}
```

```
\end{minipage}\\[3mm]
```

Aber wie haben Urmenschen das Treibhaus gemacht?

Klimaforscher ist der älteste Beruf der Welt.

Die Rekonstruktion der Erde vor 120000 Jahren zeigt, wie böse Urmenschen den Planeten erwärmten und so die armen Eisbären in der Eem-Warmzeit schwitzten.



Aber wie haben Urmenschen das Treibhaus gemacht?

Beim Nebeneinandersetzen von Minipages kann es Warnmeldungen geben, wenn im Blocksatz die Zeilen zu viel oder zu wenig gefüllt werden. Derartige Schwierigkeiten wurden vor der ersten und nach der zweiten Minipage verhindert, indem Zeilenumbrüche mit `\\[...]` eingefügt wurden. Die mit ... bestimmten Abstände vor und nach den Minipages, hier 5 mm beziehungsweise 3 mm, sind beliebig.

Ein % nach dem ersten `\end{minipage}` ist wichtig, da der sonst folgende Zeilenumbruch zu einem Leerzeichen zwischen den Minipages führen und damit die Gesamtbreite vergrößern würde. In obigem Beispiel ist die Summe der Breiten beider Minipages $0,97 \times \text{Textbreite}$ und der der Befehl `\hfill` sorgt dafür, dass die Restbreite von $0,03 \times \text{Textbreite}$ zwischen den Minipages mit Leerraum aufgefüllt wird.

Während eine Minipage keine *figure*-Umgebung enthalten darf, kann umgekehrt eine *figure*-Umgebung sehr wohl Minipages enthalten. Damit kann man in einer *figure*-Umgebung Text und Bild nebeneinander setzen.

3.3 Ausrichtung, Einrückung, Trennungen, Abstände

Ausrichtung eines Textbereichs bedeutet, dass der Text entweder linksbündig, rechtsbündig, mittig (zentriert) oder im Blocksatz dargestellt wird. Normalerweise werden Absätze im Blocksatz dargestellt. Mit einer *center*-Umgebung kann ein Textbereich zentriert und mit einer *flushleft*-Umgebung linksbündig werden. Der Schalter `\centering` ist eine Alternative zur *center*-Umgebung.

```
Nach normalem Text im Blocksatz\par
{\centering zentrierter Text\par}
und wieder ein Text im Blocksatz.
\begin{center}Hier zentrierter Text
\end{center}
und schließlich wieder Blocksatz.
```

Nach normalem Text im Blocksatz
zentrierter Text
und wieder ein Text im Blocksatz.

Hier zentrierter Text

und schließlich wieder Blocksatz.

Die *center*-Umgebung fügt vor und nach dem zentrierten Text vertikalen Abstand ein. Will man das nicht, kann man den Schalter *\centering* in einem Block (in geschweiften Klammern) verwenden. Man muss dann den Text mit *\par* abschließen.

Es gibt die Schalter *\raggedright* für linksbündigen Text, *\raggedleft* für rechtsbündigen Text, und *\centering* für zentrierten Text, aber keinen Schalter, um wieder Blocksatz zu schreiben. Daher sollte man die genannten Schalter stets in einem Block `{ }` einschließen. Nach diesem gilt dann wieder der vorgegebene Blocksatz.

Im so ausgerichteten (links- oder rechtsbündigen oder zentrierten) Text fehlt jedoch die automatische Silbentrennung (siehe unten). Abhilfe schafft das Paket *ragged2e* von Martin Schröder. Es definiert entsprechende Umgebungen und Schalter zur Textausrichtung, die die Silbentrennung unterstützen. Außerdem gibt es eine Umgebung und einen Schalter für Blocksatz.

Einrückung eines Textbereichs an der linken Seite kann erreicht werden, indem der Text in eine Umgebung mit *quote* oder *quotation* gesetzt wird. Ein [Beispiel](#):

Es folgt eine lateinische Redensart.

```
\begin{quote}
```

```
O tempora,\o mores !
```

```
\end{quote}
```

```
\begin{quotation}
```

```
O tempora,\o mores !
```

```
\end{quotation}
```

Es folgt eine lateinische Redensart.

```
O tempora,
```

```
o mores !
```

```
O tempora,
```

```
o mores !
```

In einer *quotation*-Umgebung wird die ersten Zeile jedes Absatzes (zusätzlich) eingerückt, bei *quote* nicht.

Um die automatische Einrückung der ersten Zeile eines Absatzes (einmalig) zu unterdrücken, kann man vor den Absatz den Befehl *\noindent* setzen:

Dies ist ein einzeiliger Absatz, der links eingerückt wird. `\par`

`\noindent` Dies ist ein einzeiliger Absatz, der nicht eingerückt wird.

ergibt

Dies ist ein einzeiliger Absatz, der links eingerückt wird.

Dies ist ein einzeiliger Absatz, der nicht eingerückt wird

Trennungen und Abstände beziehen sich auf Absätze oder Textbereiche. Mit dem Befehl *newline* erzwingt man die Fortsetzung des Texts in einer neuen Zeile. Der Absatz wird dadurch jedoch nicht beendet. Der Befehl `\\` hat, außerhalb bestimmter Umgebungen, die gleiche Wirkung wie *newline*. In bestimmten Umgebungen (zum Beispiel *tabular*) hat `\\` eine besondere Bedeutung, bewirkt aber meistens auch einen Zeilenumbruch.

```
Jäh \newline
```

```
kommt die Zeile\\
```

```
an ihr Ende.
```

```
Jäh
```

```
kommt die Zeile
```

```
an ihr Ende.
```

Ein vertikaler Abstand nach einem manuellen Zeilenumbruch kann durch eine Option des `\\`-Befehls erreicht werden, wobei der gewünschte Abstand mit Längeneinheit in eckigen Klammern angegeben wird.

Nach der ersten Zeile folgt `\\[2mm]`

mit Abstand dann die zweite.

Nach der ersten Zeile folgt

mit Abstand dann die zweite.

Der angegebene Abstand darf negativ sein, wodurch die Zeilen zusammenrücken.

Mit dem Befehl *vspace* kann man einen (zusätzlichen) vertikalen Abstand zwischen zwei Absätzen einfügen, dessen Größe auch negativ sein darf. Entsprechend erzeugt *hspace* einen horizontalen Zwischenraum. Beispiel:

Hier gibt es eine <code>\hspace{10 mm}</code> Lücke.	
<code>\par</code>	Hier gibt es eine Lücke.
<code>\vspace{3 mm}</code>	
Der 2. Absatz folgt mit Abstand.	Der 2. Absatz folgt mit Abstand.

Ein horizontaler Zwischenraum, der einer bestimmten Wortbreite entspricht, kann mit dem Befehl *hphantom* eingefügt werden, dessen Argument das Wort ist.

Die Oper <code>\hphantom{Aida}</code> ist von Verdi.	Die Oper ist von Verdi.
--	-------------------------

Entsprechend werden mit dem Befehl *vphantom* vertikale Abstände erzeugt, so wie es das im Argument angegebene (und möglicherweise formatierte) Zeichen oder Wort tun würde (welches aber nicht gedruckt wird). Horizontale Abstände werden dadurch nicht verändert, wohl aber Zeilenabstände.

Mozart komponierte die Oper <code>\La Nozze di \vphantom{\huge F}Figaro.</code>	Mozart komponierte die Oper La Nozze di Figaro.
---	--

Der Befehl `\hfill` ermöglicht es, eine Zeichenkette ans Ende einer Zeile zu schieben:

Kosten: <code>\hfill 30 ct/kWh</code>	Kosten: 30 ct/kWh
---------------------------------------	-------------------

Soll ein Abschnitt am Seitenanfang beginnen, kann vor dem Abschnitt mit dem Befehl *newpage* eine neue Seite angefordert werden. Beispiel:

```
\newpage
\section{Von oben herab}
Erstens kommt es anders,\newline und zweitens als man denkt.\newline
```

Silbentrennung geschieht in der Regel automatisch richtig. In Worte, die L^AT_EX falsch trennt (zum Beispiel *Lastregler*), kann man `\-` einfügen (*Last\regler*), was bei Bedarf am Zeilenende zur Silbentrennung an der angegebenen Stelle führt. Die Trennung bestimmter Worte kann auch mit dem *hyphenation*-Befehl festgelegt werden, der am besten schon in der Präambel steht. [Beispiel](#):

<code>\hyphenation{Hüte Ti-ger-ohr}</code>	
Wie das Adlauge ist das Tigerohr ein sehr gutes Sinnesorgan.	Wie das Adlauge ist das Tiger- ohr ein sehr gutes Sinnesorgan.

Die im *hyphenation*-Befehl aufgeführten Worte werden nur an den mit - angegebenen Stellen getrennt- Daher kann man die Trennung eines Wortes ganz vermeiden (wie in obigem Beispiel beim Wort *Hüte*).

3.4 Typographische Regeln

Auszeichnung soll Wörter innerhalb eines Textes hervorheben, ohne den Lesefluss zu stören. In der Regel ist *kursive* Schrift am besten dafür geeignet. Unterstreichen sollte man nur ausnahmsweise, wo dies durch eine Vorgabe gefordert wird (beispielsweise für Verweise). Sperren ist verboten. GROSSBUCHSTABEN, **fette**, größere oder **serifenlose** Schrift gilt, außerhalb von Überschriften, als zu aufdringlich oder hässlich. KAPITÄLCHEN und **Farbe** sollte man, nur in besonderen Fällen, wohlüberlegt einsetzen. **Nichtproportionale** Schrift eignet sich für den Quellcode von Computerprogrammen. Die praktische Umsetzung der Textauszeichnung wird später, in Abschnitt 4.1 (beginnend auf Seite 43) beschrieben.

Geschützte Leerzeichen verhindern, dass die Zeichen vor und nach dem Leerzeichen an einem Zeilenende getrennt werden. Es gibt nicht nur ein geschütztes Leerzeichen normaler Breite (~), sondern auch ein kleines geschütztes Leerzeichen (\,), siehe Abschnitt 4.2 auf Seite 48. Mit einem geschützten Leerzeichen bleiben Titel (oder Initiale) und Name einer Person zusammen. Ebenso ein Begriff mit zugehöriger Zahl, die vorangeht oder folgt.

7.~Beispiel aus Dr.~Ecks Buch, S.~9 7. Beispiel aus Dr. Ecks Buch, S. 9

Ein kleines geschütztes Leerzeichen trennt Maßzahl und Einheit einer Größe.

A.~Konrads Forelle: 98,5\,cm, 19,8\,kg A. Konrads Forelle: 98,5 cm, 19,8 kg

Prozentangaben enthalten einen kleinen Zwischenraum, außer als Teil eines Wortes:

80\,\% Piraten trinken 40\%igen Rum 80 % Piraten trinken 40%igen Rum

Abkürzungen sollten sparsam verwendet werden, da sie die Lesbarkeit eines Texts meistens verschlechtern. Am Satzanfang vermeidet man sie. Wenn sie nicht allgemein üblich sind, werden sie bei der ersten Benutzung erklärt. In wissenschaftlichen Texten mit vielen Abkürzungen sollte man ein Abkürzungsverzeichnis anlegen.

die Süßwasser-Straßenordnung (SW0) die Süßwasser-Straßenordnung (SWO)

In mehrgliedrigen Abkürzungen werden die Glieder in der Regel durch Punkt und kleines geschütztes Leerzeichen () getrennt:

u.\,a. auch z.\,B. Freunde i.\,w.\,S. u. a. auch z. B. Freunde i. w. S.

Ausnahmen sind mehrgliedrige Abkürzungen, die als Einheit betrachtet werden:

Globuli etc. oder Knödel usw. Globuli etc. oder Knödel usw.

Steht eine Abkürzung mit Punkt am Ende eines Satzes, folgt ihr kein zweiter Punkt. Leerzeichen und Zahl nach einer Abkürzung werden nicht abgetrennt (siehe oben).

Siehe Bsp.~3\,ff. Diese zeigen \dots Siehe Bsp. 3 ff. Diese zeigen ...

Striche können kurz oder lang, mit oder ohne Abstände davor- und dahinter, gezeichnet werden. Betrachten wir zunächst den kurzen Bindestrich (Divis):

rot-blau, Tee-Ei, Code 0-8-1-5

rot-blau, Tee-Ei, Code 0-8-1-5

Der längere Gedankenstrich dient auch für Intervalle, Gegensätze, Auslassungen:

Xaviere kam -- und blieb. \par

Xaviere kam – und blieb.

Schlossallee 6--8\par

Schlossallee 6–8

Peine -- Meine, Eintritt 3,--\,\EUR

Peine – Meine, Eintritt 3,–€

Im Englischen ist der Gedankenstrich noch länger und ohne Abstände:

Xaviere came---to stay.

Xaviere came—to stay.

4 Zeichenformatierung

4.1 Schrift

Damit die Darstellung der Zeichen eines Textes einheitlich erscheint, wird ihre Form durch eine *Schrift* gestaltet. Sofern man nichts anderes bestimmt, gibt L^AT_EX die Schrift *Computer Modern* (CM) vor. Da CM nicht alle westeuropäischen Zeichen (zum Beispiel deutsche Umlaute) enthält, sollte man, durch den Befehl `\usepackage[T1]{fontenc}` in der Präambel, CM durch eine europäische Variante ersetzen. Diese Schrift heißt *European Computer Modern* (EC) oder T1-codierte CM-Schrift.

In EC, und in anderen Schriften, kann man verschiedene Schriftarten unterscheiden, zum Beispiel Proportionalschrift mit Serifen (Beispiel: `MMii`) und ohne Serifen (Beispiel: `MMii`). Außerdem gibt es in EC eine nichtproportionale Schrift mit Serifen (Beispiel: `MMii`). Bei den Schriftarten gibt es weiterhin verschiedene Stile (`normal`, *kursiv*, **fett** ...). Daher gibt es den Zeichensatz einer Schrift (die Menge der verfügbaren Zeichen) in mehreren Ausführungen. Außerdem müssen verschiedene Schriftgrößen bereitgestellt werden.

In der Schrift EC können die verschiedenen Schriftstile nicht in allen Größen dargestellt werden oder die Darstellung ist nicht gut. Einiges kann durch das Paket *fix-cm* ergänzt werden. Will man es verwenden, muss man es mit dem Befehl `\RequirePackage{fix-cm}` laden, der als erster Befehl in der Präambel, also noch vor `\documentclass ...` stehen muss.

Die Verwendung anderer Schriften als *European Computer Modern* (EC) kann sinnvoll sein, wenn sich das Aussehen eines gedruckten Dokuments von anderen L^AT_EX-Dokumenten unterscheiden soll. Darauf soll hier nicht näher eingegangen werden. Da EC einige besondere Schriftarten, welche in mathematischen Ausdrücken benutzt werden (siehe unten), nicht bereitstellt, wird hierfür automatisch die Schrift *Computer Modern* (CM) herangezogen.

Schriftstil meint im Folgenden jede besondere Ausfertigung der Zeichen, zum Beispiel *kursiv*. Um eine kurze Zeichenkette (hier: `vult`) mit **fetter Schrift** (englisch: **boldface**) darzustellen, verwendet man den Befehl `textbf`, zum [Beispiel](#):

Et tu, `\textbf{babulus}`, sinapem addis. Et tu, **babulus**, sinapem addis.

und entsprechend kann man *kursive Schrift* (englisch: **italic**) mit dem Befehl `textit` erzeugen, **nichtproportionale** (englisch **teletypewriter** = Fernschreiber) mit `texttt` und KAPITÄLCHEN mit `textsc` (englisch: **small capitals**). Unterstreichungen sind mit dem Befehl `underline` möglich, aber es findet keine Silben- und Worttrennung in der unterstrichenen Zeichenkette statt. Die verschiedenen Befehle für Stiländerungen können kombiniert werden:

`\textit{mundus \underline{vult} decipi}` *mundus vult decipi*

Das Paket *ulem* von Donald Arseneau ermöglicht weitere Unterstreichungen oder Durchstreichungen, zum Beispiel `\uuline{text}` für doppelt unterstrichenen Text oder `\xout{text}` für schräg durchgestrichene Buchstaben. Leider funktioniert die automatische Silbentrennung in den Befehlen dieses Pakets nicht. Aber man kann (mit `\-`) eine manuelle Trennung vornehmen.

Mit `\usepackage[normalem]{ulem}` verfügbare Durch- und Unterstreichungen:

$\overline{\overline{x}}$	$\underline{\underline{y}}$	$\overline{\widetilde{xy}}$	$\overline{\overline{\underline{xy}}}$	$\overline{\dot{\underline{xy}}}$
<code>\sout{x}</code>	<code>\uline{y}</code>	<code>\uwave{xy}</code>	<code>\dashuline{xy}</code>	<code>\dotuline{xy}</code>

Das Paket *ulem* verändert normalerweise den Befehl `\emph`, so dass er Unterstreichung statt Kursivschrift erzeugt. Die Option *normalem* bewirkt jedoch ein unverändertes Verhalten des *emph*-Befehls.

In Texten mit normaler Schrift sollte die Hervorhebung einer Zeichenkette durch kursive Schrift erfolgen. Die (empfohlene) Hervorhebung *in dieser Weise* wird durch den Befehl *emph* (englisch: emphasize = hervorheben) erreicht. Wenn der Text bereits kursiv ist, dann wird durch normale Schrift hervorgehoben.

Auch in einer *math*-Umgebung (siehe Abschnitt 5.1 auf Seite 57) stehen verschiedene Schriftstile zur Verfügung.

Das Paket *censor* von Steven B. Segletes stellt den *censor*-Befehl zur Verfügung, mit dem einzelne Worte oder kurze Abschnitte innerhalb einer Zeile durch einen gleich langen schwarzen Block ersetzt, also zensiert werden. [Beispiel](#):

`\censor{Jane Shore}` war seine Mätresse. ██████████ war seine Mätresse.

Will man die Zensur aufheben, stellt man den Befehl `\StopCensoring` an den Anfang der *document*-Umgebung. Es gibt außerdem einen `\blackout`-Befehl, um größere Textblöcke zu zensieren, aber der funktioniert nur mit Einschränkungen, darf unter anderem keine Umlaute enthalten.

Schriftgröße meint die Größe der Zeichen und kann in der Einheit *Punkt* (pt) angegeben werden. Die typographische Maßeinheit hat mit dem Satzzeichen Punkt nicht zu tun (außer der Beschreibung kleiner Objekte). Die Länge eines typographischen Punkts in L^AT_EX ($351,459804\ \mu\text{m} \approx \frac{1}{3}\text{ mm}$) unterscheidet sich von anderen typographischen Definitionen dieser Einheit (Didot-Punkt, DTP-Punkt, Pica-Punkt) geringfügig, aber das ist in der Praxis unwichtig. Andere L^AT_EX-Einheiten der Schriftgröße sind *ex* und *em*. Ihre Länge in mm hängt von der Schriftart ab, denn ein *ex* ist die Höhe des Zeichens *x* und ein *em* die Breite eines *M* in der verwendeten Schrift. Außerdem können in L^AT_EX auch die Längeneinheiten mm, cm und in (Inch = 2,54 cm) verwendet werden.

Die Schriftgröße wird zu Beginn des Dokuments festgelegt, kann danach aber verändert werden. Angaben zur Schriftgröße sind, in aufsteigender Reihenfolge: `tiny`, `scriptsize`, `footnotesize`, `small`, `normalsize`, `large`, `Large`, `LARGE`, `huge` und `Huge`. Die zu verändernde Zeichenkette wird in eine entsprechende Umgebung gesetzt, zum Beispiel für `Large`:

`mundus \begin{Large}\vult\end{Large} decipi` `mundus \vult decipi`

Die verschiedenen Schriftgrößen entstehen nicht durch einfache Skalierung (Vergrößerung oder Verkleinerung einer Grundschrift). Vielmehr wurden die Schriften in

den verschiedenen Größen jeweils eigens geformt, sind also nicht geometrisch ähnlich. So ergibt sich ein besseres Schriftbild, aber auch eine merkwürdige Abstufung der Schriftgrößen. Außerdem ist zu beachten, dass es nicht alle Schriftarten in allen Größen gibt.

In den Dokumentklassen *article*, *report*, *book*, *beamer*, *scrartcl*, *scrreprt*, *scrbook* entsprechen benannte Schriftgrößenoptionen folgenden Größen in der Einheit pt:

Option	Dokument-Schriftgröße		
	10 pt	11 pt	12 pt
tiny	5 pt	6 pt	6 pt
scriptsize	7 pt	8 pt	8 pt
footnotesize	8 pt	9 pt	10 pt
small	9 pt	10 pt	10,95 pt
normalsize	10 pt	10,95 pt	12 pt
large	12 pt	12 pt	14,4 pt
Large	14,4 pt	14,4 pt	17,28 pt
LARGE	17,28 pt	17,28 pt	20,74 pt
huge	20,74 pt	20,74 pt	24,88 pt
Huge	24,88 pt	24,88 pt	24,88 pt

Wie man sieht, ist beträgt die normale Schriftgröße tatsächlich 10,95 pt, wenn man im *documentclass*-Befehl die Option *11 pt* setzt.

Es ist auch möglich, eine bestimmte Schriftgröße in der Einheit pt anzugeben.

Das ist die normale Größe

```
\par {\fontsize{8}{18}
\selectfont
und nun das Kleingedruckte\\
mit großem Zeilenabstand.
}\par Wieder normal.
```

Das ist die normale Größe
und nun das Kleingedruckte
mit großem Zeilenabstand.
Wieder normal.

In obigem Beispiel wurde im Kleingedruckten die Schriftgröße auf 8 pt und der Zeilenabstand auf 18 pt gesetzt. Die Wirksamkeit dieser Änderungen beschränkt sich auf den in geschweiften Klammern gefassten Block. Die aktuelle Schriftgröße wird mit der Befehlszeile

```
\makeatletter Schriftgröße: \f@size \,pt \makeatother
```

ausgegeben. Dabei ist `\f@size` ein Makro der zugrunde liegenden Programmiersprache TeX, welches die Schriftgröße in der Längeneinheit pt zurückgibt.

Da TeX-Makros (Befehle) nur Buchstaben enthalten dürfen und @ für TeX normalerweise kein Buchstabe ist, muss das Zeichen @ vorübergehend zum Buchstaben erklärt werden. Dazu dienen die Befehle `\makeatletter` und `\makeatother`. Dies merkwürdige Verhalten von TeX dient der Verhinderung ungewollter Veränderungen von Makros durch unbedarfte Nutzer.

Hoch- und Tiefstellung erreicht man folgendermaßen:

```
Um 10\textsuperscript{30} Uhr \\
trinke ich H\textsubscript{2}O.
```

Um 10³⁰ Uhr
trinke ich H₂O.

In einer *math*-Umgebung $\$. \dots \$$ (siehe unten) bewirkt der Befehl $\backslash sp\{\dots\}$, oder kurz $\wedge\{\dots\}$, eine Hochstellung und $\backslash sb\{\dots\}$, oder kurz $_ \{\dots\}$, eine Tiefstellung.

Um $\$10\backslash sp\{30\}\$$ Uhr $\backslash \backslash$
 trinke ich $H\$\backslash sb\{2\}\O .

Um 10^{30} Uhr
 trinke ich H_2O .

Mehr dazu findet man in Abschnitt 5.1 auf Seite 59.

Farbe kann eine Zeichenkette hervorheben. Während man in Berichten, die auf Papier gedruckt werden, bunten Text oft vermeidet, sind Farbelemente in Präsentationen die Regel. Man benutzt das Paket *color* von David Carlisle oder das umfangreichere Paket *xcolor* von Dr. Uwe Kern. Manche andere Pakete, die Farbungterstützung brauchen, laden *color* oder *xcolor* automatisch.

Man kann vordefinierte Farben verwenden. Mit *color* sind das: black, white, red, green, blue, cyan, yellow und magenta. Mit *xcolor* gibt es zusätzliche:

						
black	blue	brown	cyan	darkgray	gray	green
						
lightgray	lime	olive	orange	pink	purple	magenta
						
red	teal	violet	white	yellow		

Noch mehr vordefinierte Farben sind verfügbar, wenn das Paket *xcolor* mit der Option *[dvipsnames]* geladen wird.

						...
Apricot	Aquamarine	Bittersweet	Black	Blue	BlueGreen	...

Die vollständige *dvipsnames*-Liste und weitere Listen benannter Farben findet man in der Beschreibung des *xcolor*-Pakets.

Beliebige Farben kann man mithilfe eines Farbmodells, zum Beispiel *rgb*, *RGB* oder *cmlyk*, mischen. Für den Druck auf Papier wird das Farbmodell *cmlyk* empfohlen, für den Bildschirm aber *rgb* oder *RGB*. Letztere unterscheiden sich nicht grundsätzlich, sondern verwenden nur unterschiedliche Skalen für die Grundfarben Rot, Grün und Blau. Bei *rgb* liegen die Farbanteile zwischen 0 und 1, bei *RGB* dagegen ganzzahlig zwischen 0 und 255. Beim Farbmodell *cmlyk* wird die Farbzusammensetzung aus den Bestandteilen Cyan, Magenta, Gelb und Schwarz mit Werten jeweils zwischen 0 und 1 gebildet. In der Literatur werden Farben des *CMYK*-Farbmodells (*CMYK* mit Großbuchstaben) durch entsprechende Werte zwischen 0 und 100 angegeben. Bei der Benutzung der Pakete *color* oder *xcolor* und *cmlyk* sind die Werte der *CMYK*-Farbzusammensetzung also durch 100 zu teilen.

Mit dem Befehl *textcolor* wird eine Zeichenkette farbig (zum Beispiel in der Farbe Magenta) geschrieben. Das geht auch in einer *math*-Umgebung (siehe unten).
Beispiele:

```
\textcolor{magenta}{Magenta}: rotblau.  
\textcolor[rgb]{0.55,0.71,0}{Apfelgrün}: schön!  
\textcolor[RGB]{220,20,60}{Purpurrot: schön!}  
$3^2 + \textcolor{red}{x^2} = 5^2$
```

Magenta: rotblau.
 Apfelgrün: schön!
 Purpurrot: schön!
 $3^2 + x^2 = 5^2$

Bei der Mischung von Apfelgrün betragen die relativen rgb-Farbanteile (rot, grün und blau), die stets zwischen 0 und 1 liegen, $r = 0.55$, $g = 0.71$ und $b = 0$. Bei der Mischung von Purpurrot betragen die relativen RGB-Farbanteile (Rot, Grün und Blau), die stets zwischen 0 und 255 liegen, $r = 220$, $g = 20$ und $b = 60$.

Der `definecolor`-Befehl ermöglicht es, Farben mit einem Farbmodell zu definieren. Das erste Pflichtargument gibt den Farbnamen an, das zweite das Farbmodell und das dritte die Zusammensetzung aus den Farben des Farbmodells. Farbdefinitionen stehen meistens am Ende der Präambel.

```
\definecolor{gfb}{cmyk}{0,0,0,1}      % Schwarz
\definecolor{gfr}{cmyk}{0,1,1,0}      % Rot
\definecolor{gfy}{cmyk}{0,0.12,1,0.05} % Gold
```

Während der `definecolor`-Befehl in den beiden Paketen `color` und `xcolor` zur Verfügung steht, definiert `xcolor` als Alternative den Befehl `xdefinecolor`. Dieser ermöglicht die erweiterten Farbmischungen des `xcolor`-Pakets (siehe unten), unterstützt jedoch nicht das Farbmodell *named*.

Mit dem `xcolor`-Paket kann man eine neue Farbe aus bereits definierten Farben mischen. Die Bestandteile werden mit einem Farbausdruck `farbe1!zahl!farbe2` beschrieben, wobei *zahl* den prozentualen Anteil von *farbe1* angibt. Fehlt die letzte Farbe, wird Weiß ergänzt. Der Ausdruck kann erweitert werden. [Beispiele](#):

<code>\textcolor{blue}{Blau} \newline</code>	Blau
<code>\textcolor{blue!45!white}{mit 55\% Weiß} \newline</code>	mit 55% Weiß
<code>\textcolor{blue!45}{ebenso} \newline</code>	ebenso
<code>\textcolor{blue!60!red}{Rotblau} \newline</code>	Rotblau
<code>\textcolor{blue!60!red!30!black}{Dunkelrotblau}</code>	Dunkelrotblau

Der Farbausdruck `blue!45!white` in der zweiten Zeile mischt 45 % Blau mit Weiß, dessen Anteil also 55% beträgt. In der dritten Zeile macht `blue!45` das gleiche, da der Weißanteil automatisch ergänzt wird. Dagegen mischt `blue!60!red` in der vierten Zeile 60 % Blau mit 40 % Rot zu einem Rotblau. In der fünften Zeile mischt `blue!60!red!30!black` zunächst wieder 60 % Blau mit 40 % Rot, und mischt danach 30 % vom Rotblau mit 70 % Schwarz.

Die mit dem `definecolor`-Befehl erzeugten Farben können auch von anderen Paketen, wie TikZ (siehe unten), verwendet werden. Damit kann man zum [Beispiel](#) die deutsche Flagge zeichnen.

```
\begin{tikzpicture}
\fill [gfb] (0mm,12mm) rectangle (30mm,18mm);
\fill [gfr] (0mm,6mm) rectangle (30mm,12mm);
\fill [gfy] (0mm,0mm) rectangle (30mm,6mm);
\end{tikzpicture}
```



Die chinesische Flagge, ebenfalls mit TikZ gezeichnet, wird auf Seite 72 gezeigt.

Eine farbige Hinterlegung von schwarzem Text beeinträchtigt die Lesbarkeit und ist daher nur bei hellen Hintergrundfarben, zum Beispiel Gelb, empfehlenswert.

<code>\colorbox{yellow}{schwarz auf gelb}</code>	schwarz auf gelb
--	------------------

Eine Hinterlegung mit andersfarbigem Rand erhält man mit

<code>\fcolorbox{red}{yellow}{gelb mit rotem Rand}</code>	gelb mit rotem Rand
---	---------------------

Mit dem Paket *xcolor* kann auch eine Spektralfarbe, deren Wellenlänge in Nanometern angegeben ist, durch eine Farbmischung angenähert wiedergegeben werden.

`\textcolor{wave}{470}{Blau (470\,nm)}` Blau (470 nm)

Sollen aneinander grenzende Flächen (auch Text vor weißem oder farbigem Hintergrund) mit verschiedenen Farben belegt werden, ist das heraldische Farbsystem zu empfehlen. Unterschieden werden a) „Metalle“, das sind Gold (Gelb, RGB: 255-220-10) und Silber (Weiß, RGB: 240-240-240), und b) „Heraldische Farben“, dazu gehören Rot (RGB: 255-0-0), Blau (RGB: 0-0-255), Grün (RGB: 0-150-0) und Schwarz (RGB: 0-0-0). Gute Farbpaare bestehen aus „Metall“ und „Heraldischer Farbe“, insbesondere: Weiß-Schwarz, Weiß-Blau, Weiß-Grün, Weiß-Rot, Gelb-Schwarz, Gelb-Rot. Schlechte Paare sind solche, bei denen „Metall“ an „Metall“ oder „Heraldische Farbe“ an „Heraldische Farbe“ grenzt, zum Beispiel Weiß-Gelb und (das leider beliebte) Rot-Blau. Weiß-Gelb wird als kontrastarm empfunden. Im Beispiel oben, mit dem `\fcolorbox`-Befehl, verhindert der rote Rand eine kontrastarme Grenze zwischen Gelb und Weiß. Rotes und blaues Licht unterscheiden sich stark in der Wellenlänge und werden aufgrund der chromatischen Aberration im Auge verschieden stark gebrochen (Dispersion), so dass benachbarte Flächen dieser Farben nicht gleichzeitig scharf erscheinen. [Beispiel](#):

```
\definecolor{gb}{cmyk}{0,0,0.15,0}
\definecolor{sw}{cmyk}{1,0.85,0.85,1}
\colorbox{blue}{\textcolor{red}
{\textbf{Rot auf Blau}}}\[6mm]
\colorbox{sw}{\textcolor{gb}
{\textbf{Hellgelb auf Schwarz}}}
```

Rot auf Blau

Hellgelb auf Schwarz

4.2 Leer- und Sonderzeichen

Leerzeichen bewirken eine Lücke zwischen sichtbaren Zeichen. Manchmal möchte man verhindern, dass zwei durch Leerzeichen getrennte Worte (zum Beispiel *Den Haag*) am Zeilenende getrennt werden. Dann kann man zwischen sie ein geschütztes Leerzeichen `~`, Tilde genannt, setzen. Das Zeichen `~` (welches im Text hochgestellt sein kann: \sim) wird als Leerzeichen normaler Breite gedruckt. Damit kann man auch einen Abstand von ein, zwei oder mehr Leerzeichen erzwingen; `X~X` ergibt zum Beispiel `X X`. Es gibt weitere geschützte Leerzeichen anderer Breite.

Geschützte Leerzeichen verschiedener Breite:

XXXX	XX XX	XX XX	XX XX	XX XX	XX XX	XX XX
<code>XX\,XX</code>	<code>XX\:XX</code>	<code>XX~XX</code>	<code>XX\;XX</code>	<code>XX\quad XX</code>	<code>XX\qquad XX</code>	

In einer *math*-Umgebung (siehe unten) werden schmale geschützte Leerzeichen (nämlich `\,` und `\;`) oft eingesetzt, zum Beispiel um in einem Produkt einen Multiplikationsoperator darzustellen (Beispiel: $a b$ statt ab ohne Lücke).

Es gibt Leerzeichen negativer Breite, die benachbarte Zeichen zusammenrücken lassen. Folgende sind in normalem Text und in einer *math*-Umgebung verfügbar.

Leerzeichen negativer Breite:

XXXX	XXXX	XXXX
<code>XX\negthinspace XX</code>	<code>XX\negmedspace XX</code>	<code>XX\negthickspace XX</code>

`\negmedspace` und `\negthickspace` erfordern das Paket *amsmath*. Statt `\negthinspace` kann man `\!` schreiben (in normalem Text nur nach dem Laden von *amsmath*).

Sonderzeichen sind Zeichen, die nicht direkt über die Tastatur eingegeben werden können, weil es sie dort nicht gibt oder weil die dann als Steuerzeichen gedeutet würden. Die Eingabe von Sonderzeichen erfolgt daher jeweils über eine besondere Zeichenkette (Code). \$ an Anfang und Ende eines Codes kennzeichnen eine *math*-Umgebung (siehe Seite 57). Es gibt im Internet verschiedene, umfangreiche Aufzählungen der in L^AT_EX verfügbaren Sonderzeichen, zum Beispiel die [Liste von Scott Pakin](#). Einige Sonderzeichen und Symbole werden nachfolgend vorgestellt, weitere (mathematische) werden in Abschnitt 5.2 auf Seite 61 dargestellt.

Symbole, die in L^AT_EX als Steuerzeichen dienen, müssen im Quelltext maskiert werden. Sie können mit folgenden Befehlen dargestellt werden:

%	\$	&	#	_	{	}	¶	§	\	~
\%	\\$	\&	\#	_	\{	\}	\P	\S	\$\backslash\$	\$\sim\$
				\$			^			
\textbackslash				\textdollar			\textasciicircum			\textunderscore

Deutsche Anführungszeichen:

„deutsch“	‚halb‘
\glqq deutsch\grqq{}	oder "deutsch" \glq halb\grq{}

Englische und französisch-schweizerische Anführungszeichen:

“double”	‘single’	«franz.-schweiz.»	⟨halb⟩
\lq\lq englisch\rq\rq{}	\lq halb\rq{}	\flqq deutsch\frqq{}	\flq halb\frq{}

Zur Kennzeichnung der direkten Rede werden im Vereinigten Königreich meistens einfache (single quotes), in den USA dagegen in der Regel doppelte Anführungszeichen (double quotes) gesetzt. Sie können auch ``so'` beziehungsweise ```so''` (mit je zwei Zeichen vor und nach dem *so*) erzeugt werden.

Hochgestellte gerade Anführungszeichen werden unter anderem für die Darstellung von Programm-Code benötigt. Zur Darstellung der einfachen geraden Anführungszeichen braucht man das Paket *textcomp* von Sebastian Rahtz:

"doppelte gerade"	'einfache gerade'
\dq doppelte gerade\dq{}	\textquotesingle einfache gerade\textquotesingle{}

Ein typographisch richtiger [Apostroph](#) ' (Auslassungszeichen) wird (für deutschen und englischen Text) durch Eingabe des Zeichens ' auf der Tastatur erzeugt.

Kursive griechische Kleinbuchstaben:

α	β	γ	δ	μ	π	ω
\$\alpha\$	\$\beta\$	\$\gamma\$	\$\delta\$	\$\mu\$	\$\pi\$	\$\omega\$

Für einige griechische Kleinbuchstaben gibt es eine alternative Schreibweise, zum Beispiel ϱ statt ρ . Man bekommt sie durch Voranstellen von *var*: `ϱ`. Dies

betrifft epsilon ϵ , theta θ , pi π , rho ρ , sigma σ und phi ϕ .

Das Paket *upgreek* von Walter Schmidt gibt uns steile (nicht kursive) griechische Kleinbuchstaben:

α	β	ϑ	μ	π
<code>\upalpha</code>	<code>\upbeta</code>	<code>\upvartheta</code>	<code>\upmu</code>	<code>\uppi</code>

Man beachte: Das μ in μm kann mit `\textmu` geschrieben werden (siehe unten).

Steile (nicht kursive) griechische Großbuchstaben:

Δ	Λ	Π	Σ	Φ	Ψ	Ω
<code>\Delta</code>	<code>\Lambda</code>	<code>\Pi</code>	<code>\Sigma</code>	<code>\Phi</code>	<code>\Psi</code>	<code>\Omega</code>

Kursive griechische Großbuchstaben erhält man in einer *math*-Umgebung mit dem Schriftstil *mathit* (siehe unten). So wird Σ mit `\mathit{\Sigma}` erzeugt.

Pfeile (einseitige Pfeile gibt es mit *left* und *right*):

\rightarrow	\Rightarrow	\Leftrightarrow	\Longleftarrow
<code>\rightarrow</code>	<code>\Rightarrow</code>	<code>\Leftrightarrow</code>	<code>\Longleftarrow</code>

Pfeile können mit *not* gestrichen werden, zum Beispiel \nrightarrow mit `\not\rightarrow`.

Pfeile mit auf- oder untersetztem Text, zum Beispiel $x \xrightarrow[Addition]{+8} y$, erhält man mit

`\xrightarrow[Addition]{+8}`

Pfeile, die mit dem Paket *amsfonts* verfügbar sind:

\dashrightarrow	\rightrightarrows	\rightleftarrows	\rightsquigarrow
<code>\dashrightarrow</code>	<code>\rightrightarrows</code>	<code>\rightleftarrows</code>	<code>\rightsquigarrow</code>

Punkte und Kreise:

\dots	\cdots	\vdots	\bullet	\circ	\bigcirc	\bigcirc
<code>\dots</code>	<code>\cdots</code>	<code>\vdots</code>	<code>\bullet</code>	<code>\circ</code>	<code>\bigcirc</code>	<code>\textbigcirc</code>

Akzente:

\grave{x}	\acute{x}	\check{x}	\hat{x}	\breve{x}	\tilde{x}	\bar{x}	\dot{x}	\textcircled{c}
<code>\grave{x}</code>	<code>\acute{x}</code>	<code>\check{x}</code>	<code>\hat{x}</code>	<code>\breve{x}</code>	<code>\tilde{x}</code>	<code>\bar{x}</code>	<code>\dot{x}</code>	<code>\textcircled{c}</code>

Der *textcircled*-Befehl zeichnet einen Kreis um oder über das im Pflichtargument genannte Zeichen: `\textcircled{a}` ergibt \textcircled{a} .

Weiteres Symbole sind:

...	.	£	£	†
<code>\textellipsis</code>	<code>\textperiodcentered</code>	<code>\textsterling</code>	<code>\pounds</code>	<code>\dag</code>
	˘	—	—	
<code>\textbardbl</code>	<code>\textvisiblespace</code>	<code>\textemdash</code>	<code>\textendash</code>	<code>\textbar</code>
©	™	å	Å	Ø
<code>\copyright</code>	<code>\texttrademark</code>	<code>\aa</code>	<code>\AA</code>	<code>\O</code>
‡	*	>	<	•
<code>\ddag</code>	<code>\textasteriskcentered</code>	<code>\textgreater</code>	<code>\textless</code>	<code>\textbullet</code>

Mit dem Paket *textcomp* von Sebastian Rahtz im Text verfügbare Sonderzeichen:

	€	№	μ	Ω
<code>\textlbrokenbar</code>	<code>\textestimated</code>	<code>\textnumero</code>	<code>\textmu</code>	<code>\textohm</code>
˜	♪	°C	★	†
<code>\texttildelow</code>	<code>\textmusicalnote</code>	<code>\textcelsius</code>	<code>\textborn</code>	<code>\textdied</code>
°	½	×	÷	—
<code>\textdegree</code>	<code>\textonehalf</code>	<code>\texttimes</code>	<code>\textdiv</code>	<code>\textminus</code>
¢	\$	¥	±	🍃
<code>\textcent</code>	<code>\textdollaroldstyle</code>	<code>\textyen</code>	<code>\textpm</code>	<code>\textleaf</code>

Achtung: Das Paket *textcomp* definiert einige Symbole neu!

Mit *textcomp* sind im Text-Modus auch einige besondere Klammerpaare vorhanden:

⟨⟩	⌈⌋	{ }
<code>\textlangle</code>	<code>\textrangle</code>	<code>\textlbrackdbl</code>
<code>\textlbrackdbl</code>	<code>\textrbrackdbl</code>	<code>\textlquill</code>
<code>\textrquill</code>		























Mit dem Paket *marvosym* von Martin Vogel im Text verfügbare Sonderzeichen:

€	\$	₴	€	€	€
<code>\EUR</code>	<code>\EyesDollar</code>	<code>\Denarius</code>	<code>\EURcr</code>	<code>\EURhvf</code>	<code>\EURtm</code>
🌐	📞	☎	✉	FAX	✉
<code>\Mundus</code>	<code>\Mobilefone</code>	<code>\Telefon</code>	<code>\Letter</code>	<code>\FAX</code>	<code>\Email</code>
🛑	👩	👨	⚡	😊	☹
<code>\Stopsign</code>	<code>\WomanFace</code>	<code>\ManFace</code>	<code>\Lightning</code>	<code>\Smiley</code>	<code>\Frowny</code>
🚲	🕒	✍	ℹ	♥	🦇
<code>\Bicycle</code>	<code>\ClockLogo</code>	<code>\Writinghand</code>	<code>\Info</code>	<code>\Heart</code>	<code>\bat</code>
♁	♃	♀	♂	☾	☼
<code>\Earth</code>	<code>\Jupiter</code>	<code>\Venus</code>	<code>\Mars</code>	<code>\Moon</code>	<code>\Sun</code>

Die Symbole € (`\EURcr`), € (`\EURhvf`) und € (`\EURtm`) sind an die Schriftarten Courier, Helvetica und Times Roman angepasste Varianten des Euro-Symbols.

Mit dem Paket *fontawesome* im Text verfügbare Sonderzeichen:

♂	♀	🚲	🚗	🚚	🚆	✈
<code>\faMale</code>	<code>\faFemale</code>	<code>\faBicycle</code>	<code>\faCar</code>	<code>\faTruck</code>	<code>\faTrain</code>	<code>\faPlane</code>






						
<code>\faBolt</code>	<code>\faBomb</code>	<code>\faBook</code>	<code>\faBug</code>	<code>\faClockO</code>	<code>\faGears</code>	<code>\faWrench</code>
						
<code>\faDiamond</code>	<code>\faCube</code>	<code>\faFutbolO</code>	<code>\faBello</code>	<code>\faAnchor</code>	<code>\faPaperclip</code>	
						
<code>\faFileO</code>	<code>\faImage</code>	<code>\faLinux</code>	<code>\faApple</code>	<code>\faWindows</code>	<code>\faWikipediaW</code>	
						
<code>\faFlagCheckered</code>	<code>\faHandScissorsO</code>	<code>\faHandRockO</code>	<code>\faHandPaperO</code>			

Das Paket *ifsym* stellt verschiedene Symbolgruppen bereit, die jeweils als Option geladen werden. Mit `\usepackage[electronic]{ifsym}` erhält man zum Beispiel:

				
<code>\PulseHigh</code>	<code>\ShortPulseHigh</code>	<code>\ShortPulseLow</code>	<code>\RaisingEdge</code>	<code>\FallingEdge</code>

Es gibt auch Symbole, mit denen Taktdiagramme zusammengesetzt werden können. Weitere Paketoptionen sind: *alpine*, *clock*, *geometry*, *misc*, *weather*. Man beachte, dass einige Symbolnamen auch von anderen Paketen benutzt werden.

Mit dem Paket *epsdice* im Text verfügbare Sonderzeichen:

				
<code>\epsdice{1}</code>	<code>\epsdice{2}</code>	<code>\epsdice{3}</code>	<code>\epsdice{6}</code>	<code>\epsdice[black]{5}</code>

Mit dem Paket *pifont* im Text verfügbare Sonderzeichen:

①	②	⑨	⑩	✿	☆
<code>\ding{172}</code>	<code>\ding{173}</code>	<code>\ding{180}</code>	<code>\ding{181}</code>	<code>\ding{96}</code>	<code>\ding{73}</code>
➤	♣	♦	♥	♠	✍
<code>\ding{226}</code>	<code>\ding{168}</code>	<code>\ding{169}</code>	<code>\ding{170}</code>	<code>\ding{171}</code>	<code>\ding{46}</code>

Das Paket *svrsymbols* von Apostolos Syropoulos stellt verschiedene phantasievolle Symbole zur Verfügung (auch für Elementarteilchen), die man zum Beispiel für chemische Texte nutzen kann (siehe Abschnitt 7.1 auf Seite 85). Wenn das Paket *keystroke* von Rolf Niepraschk installiert wurde, kann man kleine gerahmte Symbole für Tastatur-Tasten als Sonderzeichen wiedergeben. Beispiele für entsprechende Befehle sind: `\keystroke{C}` (oder statt C ein anderer Buchstabe), `\Return`, `\Ctrl`, `\Shift`, `\Spacebar` und `\DArrow`.

Mit dem Paket *CountriesOfEurope* von Rolf Niepraschk und Herbert Voß kann man einige europäische Länder als Symbole in den Text einfügen. Es wird durch

`\usepackage[scaled=5]{CountriesOfEurope}`

in der Präambel geladen. Der Skalierungsfaktor, hier 5, bestimmt die Größe der Symbole. Im Dokument wird zum Beispiel mit

Dies
`{\CountriesOfEuropeFamily\Spain}`
 ist Spanien.

Dies  ist Spanien.

ein Land (hier Spain = Spanien) in den Text eingefügt. Die Zeichenbreite und -höhe

ist für verschieden große Länder natürlich unterschiedlich, aber die Schriftgröße lässt sich wie bei anderen Zeichen ändern. Diese Ländersymbole gibt es:

Albania, Andorra, Austria, Belarus, Belgium, Bosnia, Bulgaria, Croatia, Czechia, Denmark, Estonia, Finland, France, Germany, GreatBritain, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Liechtenstein, Lithuania, Luxembourg, Macedonia, Malta, Moldova, Montenegro, Netherlands, Norway, Poland, Portugal, Romania, Serbia, Slovakia, Slovenia, Spain, Sweden, Switzerland.

Für die Darstellung der Logos für L^AT_EX und ähnliches gibt es folgende Befehle. Der *hologo*-Befehl erfordert das Paket *hologo* von Heiko Oberdiek.

L ^A T _E X	L ^A T _E X 2 _ε	T _E X	KOMA-Script	pdfL ^A T _E X
<code>\LaTeX</code>	<code>\LaTeXe</code>	<code>\TeX</code>	<code>\KOMAScript</code>	<code>\hologo{pdfLaTeX}</code>

Folgt nach einem der ersten vier Befehle ein Leerzeichen, hängt man eine geschweifte Klammer an den Befehlsnamen, zum Beispiel: `\LaTeX{}`. Das Logo KOMA-Script ist nur in den Dokumentklassen *scrartcl*, *scrreprt* und *scrbook* verfügbar.

Römische Zahlen werden mit dem Paket *romanbar* von H.-Martin Münch und dem Befehl *Romannum{...}* geschrieben, der als Pflichtargument eine natürliche Zahl erhält. Der *romannum*-Befehl schreibt Kleinbuchstaben. [Beispiel](#):

```
\Romannum{2018} \quad \romannum{2018}           MMXVIII   mmxviii
```

Chinesische, japanische oder koreanische Schriftzeichen können, wenn sie im Format UTF-8 codiert sind, mit dem Paket *CJKutf8* von Werner Lemberg eingebettet werden. Die Zeichenkette wird in eine CJK-Umgebung gesetzt. Pflichtargumente sind *UTF8* und die Bezeichnung des Zeichensatzes, zum [Beispiel](#) *gkai* oder *gbsn* für chinesische Kurzzeichen beziehungsweise *bkai* oder *bsmi* für Langzeichen.

Die Volksrepublik China <code>\(chinesisch %</code> <code>\begin{CJK}{UTF8}{gkai}</code> 中华人民共和国% <code>\end{CJK}%</code> <code>) \list ein Staat im östlichen Asien.</code>	Die Volksrepublik China (chinesisch 中华人民共和国) ist ein Staat im östlichen Asien.
--	--

Das Prozentzeichen % am Ende der chinesischen Zeichenkette verhindert, dass ein Leerzeichen vor der folgenden schließenden Klammer) eingefügt wird: Jedes Zeilenende wird unsichtbar durch das Zeilenendezeichen markiert, welches wie ein Leerzeichen wirkt. Das %-Zeichen bricht jedoch die Auswertung der Zeile ab, bevor das Zeilenendezeichen ausgewertet wird.

4.3 Physikalische Einheiten

Physikalische Einheiten sollten nicht kursiv, sondern steil gesetzt werden; auch in einer *math*-Umgebung (siehe unten). Zwischen Zahl und Einheit eines Wertes sollte ein kurzes Leerzeichen \, stehen. Physikalische Größen und mathematische Variable sollten dagegen kursiv sein.

Wenn man nur gelegentlich Werte mit Einheiten schreibt, ist das relativ einfache Paket *units* von Axel Reichert nützlich. Es bietet, neben dem Befehl *nicefrac* (siehe Seite 59), zwei Befehle zur Formatierung von Einheiten oder Werten mit Einheiten,

davon einen für die Darstellung von Einheiten als Bruch. Man kann sie in normalem Text oder in einer *math*-Umgebung $\$....\$$ (siehe unten) verwenden:

$\$v\$$ war $\unitfrac[2\{,\}5]{m}{s}\backslash,.$ v war $2,5\text{ m/s}$.

$\$\unit{N} = \unit{kg}\cdot\unit{m/s^2}\$$ $N = \text{kg} \cdot \text{m/s}^2$

Die Befehle können Einheiten, oder Zahlen mit Einheiten, wiedergeben. Ein steiles (nicht kursives) μ erhält man mit dem Befehl `\textmu`, wenn das Paket *textcomp* geladen wurde; ebenso ein Ω mit dem dem Befehl `\textohm`.

Das Paket *siunitx* von Joseph Wright bringt mehr Darstellungsmöglichkeiten für Zahlen und Einheiten. Bei der Eingabe von Dezimalzahlen können Komma oder Punkt als Trennzeichen dienen. Um a) bei der Ausgabe von Zahlen das Komma als Dezimaltrennzeichen zu bekommen (statt eines Dezimalpunkts) und b) die Einheiten bit und byte, nebst passender Vorsätze (Präfixe) zu haben, lädt man das Paket mit

`\usepackage[output-decimal-marker={,}, binary-units]{siunitx}`

Falls man (anders als im Folgenden) einen Punkt \cdot statt eines Kreuzes \times als Multiplikationsoperator haben möchte, fügt man in die Präambel diesen Befehl ein:

`\sisetup{exponent-product = \cdot, output-product = \cdot}`

Eine physikalische Einheit wird mit dem Befehl `\si{...}` gebildet. Das Pflichtargument enthält Befehle zur Gestaltung einer, möglicherweise zusammengesetzten, Einheit. Eine einfache Einheit kann mit einem dafür vorgesehenen Befehl erzeugt werden. In der Regel gibt es auch eine abgekürzte Variante.

die Einheit <code>\si{\second}</code> für Zeitangaben	die Einheit s für Zeitangaben
die Einheit <code>\si{s}</code> für Zeitangaben	die Einheit s für Zeitangaben

Es ist möglich, im Pflichtargument eine Zeichenkette (ohne `\`) anzugeben.

die Einheit <code>\si{s}</code> für Zeitangaben	die Einheit s für Zeitangaben
---	-------------------------------

Von dieser Möglichkeit wird jedoch abgeraten, weil damit Eingabefehler schlechter erkannt werden und einige Methoden der Formatierung fehlen. Einheiten, die aus einem Bruch bestehen, können unterschiedlich geschrieben werden. [Beispiele](#):

$\$\si{\kilogram\meter\per\square\second}\$$ kg m s^{-2}

$\$\si[per-mode=symbol]{\micro\meter\per\day}\$$ $\mu\text{m/d}$

$\$\si[per-mode=fraction]{\coulomb\per\mole}\$$ $\frac{\text{C}}{\text{mol}}$

Unter anderem sind folgende Namen von Einheiten verfügbar:

arcminute, arcsecond, ampere, angstrom, astronomicalunit, atomicmassunit, bar, barn, becquerel, bel, bit, bohr, byte, candela, clight, coulomb, dalton, day, decibel, degree, degreeCelsius, electronmass, electronvolt, elementarycharge, farad, gray, hartree, hectare, henry, hertz, hour, joule, katal, kelvin, kilogram, knot, liter, litre, lumen, lux, meter, minute, mmHg, mole, nauticalmile, neper, newton, ohm, pascal, percent, planckbar, radian, second, siemens, sievert, steradian, tesla, tonne, volt, watt, weber.

litre ergibt das eigentlich richtige Symbol l, aber *liter* das besser lesbare L, das ebenfalls erlaubt ist. Statt *meter* ist auch (mit gleichem Ergebnis) die US-Schreibweise *metre* möglich. In Prozentangaben sollte, wie bei Einheiten, ein kleines Leerzeichen enthalten sein. Dies wird mit *percent* erreicht, das wie eine Einheit verwendet wird.

Außerdem sind folgende Einheitenvorsätze (Präfixe) definiert:

yocto (10^{-24}), zepto (10^{-21}), atto (10^{-18}), femto (10^{-15}), pico (10^{-12}), nano (10^{-9}), micro (10^{-6}), milli (10^{-3}), centi (10^{-2}), deci (10^{-1}), deca (10^1), hecto (10^2), kilo (10^3), mega (10^6), giga (10^9), tera (10^{12}), peta (10^{15}), exa (10^{18}), zetta (10^{21}), yotta (10^{24}); alternativ zu deca ist auch die Schreibweise deka möglich.

Für viele Einheiten, auch mit Vorsätzen, gibt es Abkürzungen, zum Beispiel:

ng, ug, mg, g, kg, amu; nm, um, mm, cm, dm, m, km; ns; us, ms, s; umol, mmol, mol; pA, nA, uA, mA, A; ul, ml, l; Hz, kHz, MHz, GHz; N, kN; Pa; kohm, Mohm; uV, mV, V, kV; uW, mW, W, kW, MW; J, kJ, eV, keV, MeV, kWh; pF, F; K; dB

Potenzen (vor die Einheiten geschrieben): square, cubic, raiseto{...}, per

$\text{\si{uW}\per\square\m\raiseto{-4}\K}$ $\mu\text{W m}^{-2} \text{K}^{-4}$

Potenzen (nach den Einheiten geschrieben): squared, cubed, tothe{...}

$\text{\si{meter}\cubed\second\tothe{-4}}$ $\text{m}^3 \text{s}^{-4}$

Für die Einheiten bit und byte sind Vorsätze üblich, die Potenzen von 2 sind:

kibi (2^{10}), mebi (2^{20}), gibi (2^{30}), tebi (2^{40}), pebi (2^{50}), exbi (2^{60}), zebi (2^{70})

$1024\text{\,}\text{\si{mebi}\byte} = 1\text{\,}\text{\si{gibi}\byte}$ $1024 \text{ MiB} = 1 \text{ GiB}$

Winkel können mit dem Befehl *ang* angegeben werden.

Winkel $\text{\alpha} = \text{\ang{8.5}}$ Winkel $\alpha = 8,5^\circ$
 $\text{\ang{1;2;3}}$ und $\text{\ang{;10;}}$ $1^\circ 2' 3''$ und $10'$

Der Befehl *of* versieht Einheiten mit einem beschreibenden Index:

$\text{\si{decibel}\of{SPL}}$ dB_{SPL}

Wenn man das Kürzen gleicher Einheiten in Zähler und Nenner eines Bruchs zeigen möchte, sind Durchstreichungen nützlich. Man braucht dafür das Paket *cancel* von Donald Arseneau. [Beispiel](#):

$\text{\si[per-mode=fraction]{\cancel{m}\per\cancel{m}\per{s}}}$ $\frac{\text{m}}{\text{m}^2 \text{s}}$

Eine Zahl ohne Einheit, aber möglicherweise mit einer Zehnerpotenz (markiert durch e, E, d oder D), kann mit dem Befehl *num* formatiert ausgegeben werden.

$x = \text{\num{-5,25e-3}}$ $x = -5,25 \times 10^{-3}$
 $\text{\num[group-separator = \text{\text{\~}}]{1234567}}$ 1 234 567
 $\text{\num[negative-color = red]{-1,23E88}}$ $-1,23 \times 10^{88}$
 $\text{\num[retain-explicit-plus]{+42}}$ +42

Im zweiten Beispiel steht in eckigen Klammern eine Option, die eine Zifferngruppierung zur besseren Lesbarkeit längerer Zahlen bewirkt, hier mit einem Leerzeichen. Im dritten Beispiel bewirkt die Option, dass negative Zahlen rot gedruckt werden und in vierten wird das positive Vorzeichen ausnahmsweise ausgeschrieben.

Eine beliebige Zahl kann auch farbig gedruckt werden. Eine Liste von Zahlen (getrennt durch ;) wird mit dem Befehl *numlist* ausgegeben. Statt einer Zahl kann mit *num* oder *numlist* das Symbol π (`\pi`) oder das Symbol \dots (`\dots`) stehen. Ein Produkt wird mithilfe des Multiplikationszeichens (Kleinbuchstabe x) ausgegeben. Komplexe Zahlen werden mit der imaginären Einheit *i* oder *j* eingegeben und bei der Ausgabe wird, sofern nicht anders bestimmt, ein nachgestelltes i geschrieben.

<code>\num[color = teal]{42}\</code>	42
<code>\numlist{4;9;16;25}\</code>	4, 9, 16 und 25
<code>\numlist{\pi;2\pi;3 \pi;\dots}\</code>	π , 2π , 3π und \dots
<code>\num{0.1 x 2.5 x 4} = \num{1}\</code>	$0,1 \times 2,5 \times 4 = 1$
<code>\num{3,3+i5} = \num{3,3+5i}\</code>	$3,3 + 5i = 3,3 + 5i$
<code>\num[output-complex-root = j]{3,3+5j}\</code>	$3,3 + 5j$
<code>\num[complex-root-position=before-number]{3,3+5j}</code>	$3,3 + i5$

Der Wert einer physikalischen Größe besteht aus Zahl und Einheit. Er kann mit dem Befehl *SI* ausgegeben werden. In einer ersten geschweiften Klammer wird die Zahl angegeben (wie beim Befehl *num*) und in einer zweiten geschweiften Klammer wird die Einheit genannt (wie beim Befehl *si*). Möglich ist die Beeinflussung der Darstellung durch Optionen in einer eckigen Klammer vor den geschweiften Klammern.

<code>\\$E = \SI{3,8E6}{\mega\volt\per\meter}\$</code>	$E = 3,8 \times 10^6 \text{ MV m}^{-1}$
<code>\\$ \SI[parse-numbers=false]{\sqrt{2}}{\m}\$</code>	$\sqrt{2} \text{ m}$

Im zweiten Beispiel erlaubt die Option in eckigen Klammern die Eingabe einer formatierten Zahl, wie in einer *math*-Umgebung (siehe unten), hier eine Wurzel.

Das Ergebnis einer Messung gibt den Wert einer Größe mit einer Messungenauigkeit wieder. Diese hat die gleiche Einheit wie der Messwert. Sie kann, für die letzten Ziffern des Messwerts, in runden Klammern angegeben werden. Wenn man eine Lücke zwischen Messwert und Messungenauigkeitsklammer wünscht, gibt man die Option *uncertainty-separator={\,}* an. Alternativ kann die Messungenauigkeit durch das Zeichen \pm vom Messwert getrennt werden. Dies wird mit der Option *separate-uncertainty* erreicht.

<code>\SI{3,5(1)}{\volt}</code>	3,5(1) V
<code>\SI[uncertainty-separator={\,}]{3,5(1)}{\volt}</code>	3,5 (1) V
<code>\SI[separate-uncertainty]{3,5(1)}{\volt}</code>	(3,5 \pm 0,1) V

Will man die letztgenannte Schreibweise als Standard wählen, fügt man in die Präambel (nach dem Laden des Pakets *siunitx*) den Befehl

`\sisetup{separate-uncertainty}`

ein. Nun kann man auf die Angabe der entsprechenden Option verzichten. Außerdem kann die Messungenauigkeit im Quelltext durch ein vorangestelltes \pm angegeben werden, statt durch Klammerung. [Beispiel](#):

<code>\SI{3,5(1)}{\volt}</code>	(3,5 \pm 0,1) V
<code>\SI{3,5 +- 0,1}{\volt}</code>	(3,5 \pm 0,1) V

Eine Liste von Werten mit der gleichen Einheit wird mit dem Befehl *SIl* in normalem Text, nicht im *math*-Modus (siehe unten), ausgegeben.

<code>\SIl{1;2;3}{\meter}</code>	1 m, 2 m und 3 m
----------------------------------	------------------

5 Mathematik

5.1 Mathematik im Fließtext

Mathematische Ausdrücke, die zwischen normalem Text (Fließtext) stehen, können in einer *math*-Umgebung geschrieben werden (*inline math mode*). Es gibt drei Möglichkeiten dafür: a) `$... $`, b) `\begin{math} ... \end{math}` und c) `\(... \)`, wobei der Inhalt ... im *math*-Modus ausgewertet wird, also anders als normaler Text, welcher im Textmodus ausgewertet wird. Besagte drei Möglichkeiten haben meist die gleiche Wirkung, aber in besonderen Textbereichen ist die Wirkung unterschiedlich (zum Beispiel in Befehlen mit „beweglichem Text“, wie bei Fußnoten). In der Regel ist `$... $` zu empfehlen. In einer *math*-Umgebung werden im Quellcode eingefügte Leerzeichen nicht gedruckt und können zur lesbaren Gestaltung des Quellcodes eingesetzt werden.

Größere mathematische Ausdrücke werden nicht in den Fließtext eingebettet sondern abgesetzt in einer *displaymath*-Umgebung gebildet. Es gibt zwei Möglichkeiten dafür: a) `\begin{displaymath} ... \end{displaymath}` und b) `\[... \]`. In einer *displaymath*-Umgebung werden einige Symbole größer dargestellt als in einer *math*-Umgebung. Ansonsten verhält sich der *math*-Modus in beiden Umgebungen gleich.

Beispiele:

<code>\(\sum_{i=1}^{10} i = 55 \)</code>	$\sum_{i=1}^{10} i = 55$
<code>\quad \$\lim_{n \rightarrow \infty} x_n\$</code>	$\lim_{n \rightarrow \infty} x_n$
<code>\par\bigskip</code>	
Abgesetzt: <code>\[\sum_{i=1}^{10} i = 55\]</code>	$\sum_{i=1}^{10} i = 55$
<code>\[\lim_{n \rightarrow \infty} x_n \]</code>	$\lim_{n \rightarrow \infty} x_n$

Die *equation**-Umgebung entspricht im Wesentlichen der *displaymath*-Umgebung, erfordert aber das Paket *amsmath*. Für nummerierte Gleichungen gibt es die Umgebungen *equation* und *align* (siehe Abschnitt 5.3 auf Seite 64).

Bei der Klammerung größerer Ausdrücke genügt die normale Klammergröße oft nicht. Die Verwendung größenangepasster Klammern wird in Abschnitt 5.3 auf Seite 65 beschrieben.

Schriftstile in einer *math*-Umgebung stehen über besondere Befehle zur Verfügung, ohne dass Pakete geladen werden müssten.

ohne Befehl: $\$ABCabcd\backslash beta\backslash gamma\backslash Phi\backslash Psi\ 2389\$$	$ABCabcd\beta\gamma\Phi\Psi2389$
$\$ \backslash mathnormal\{ABCabcd\backslash beta\backslash gamma\backslash Phi\backslash Psi\ 2389\}\$$	$ABCabcd\beta\gamma\Phi\Psi_{2389}$
$\$ \backslash mathrm\{ABCabcd\backslash beta\backslash gamma\backslash Phi\backslash Psi\ 2389\}\$$	$ABCabcd\beta\gamma\Phi\Psi2389$
$\$ \backslash mathbf\{ABCabcd\backslash beta\backslash gamma\backslash Phi\backslash Psi\ 2389\}\$$	$ABCabcd\beta\gamma\Phi\Psi2389$
$\$ \backslash mathsf\{ABCabcd\backslash beta\backslash gamma\backslash Phi\backslash Psi\ 2389\}\$$	$ABCabcd\beta\gamma\Phi\Psi2389$
$\$ \backslash mathhtt\{ABCabcd\backslash beta\backslash gamma\backslash Phi\backslash Psi\ 2389\}\$$	$ABCabcd\beta\gamma\Phi\Psi2389$
$\$ \backslash mathit\{ABCabcd\backslash beta\backslash gamma\backslash Phi\backslash Psi\ 2389\}\$$	$ABCabcd\beta\gamma\Phi\Psi2389$
$\$ \backslash mathcal\{ABCXYZ\}\$$	$ABC\mathcal{XYZ}$

Der kalligraphische Stil *mathcal* kann nur lateinische Großbuchstaben darstellen. Beim *mathit*-Stil gibt es bei ff eine Ligatur (Buchstabenverbindung), zum Beispiel im Wort *Suffix* statt *Su ffix*, was bei Funktionsnamen wichtig sein kann. Mit dem Paket [amsfonts](#) der American Mathematical Society (AMS) gibt es außerdem den Stil *mathfrak*, der aber keine griechischen Großbuchstaben erzeugt: $\mathfrak{ABCabcd}\beta\gamma2389$. Um in *math*-Umgebungen kursive Zeichen fett zu drucken, lädt man das Paket *bm*.

$\$ \backslash mathrm\{v\}, \backslash bm\{\backslash mathrm\{v\}\}\$$ und $\$ \backslash bm\{\backslash varphi\}\}\$$ v, \boldsymbol{v} und $\boldsymbol{\varphi}$

Unterschiedliche Schriftstile für mathematische Symbole sollen die mathematischen Objekte leichter erkennbar machen. Es gibt es verschiedene Empfehlungen dazu, zum Beispiel diese: Variablen im weiteren Sinne (einschließlich Vektoren, Tensoren und Matrizen) werden kursiv gesetzt, alles andere (einschließlich Zahlen, Einheiten, Operatoren, Konstanten und Namen unveränderlicher Objekte) geradestehend (steil). Dies gilt sowohl für lateinische als auch für griechische Buchstaben. Demnach ist $u = 2 \times \pi \times r$ richtig. Es gilt auch für Indizes, also h_i (i geradestehend) für eine Anfangshöhe und h_i (i kursiv) für das i -te Element einer Folge. Auch Vektoren, Tensoren und Matrizen sind Variable und werden kursiv geschrieben. Im Unterschied zu Skalaren werden sie fett gedruckt, nicht aber ihre Indizes. Vektor-Operatoren, zum Beispiel ∇ , sind geradestehend und fett. Vektoren werden mit Kleinbuchstaben, Tensoren und Matrizen mit Großbuchstaben geschrieben.

Rahmen können auch in in einer *math*-Umgebung mit dem *fbox*-Befehl (siehe Abschnitt 3.2 auf Seite 35) erzeugt werden. Der Inhalt der *fbox* ist zwar zunächst im Textmodus, aber sie kann auch eine *math*-Umgebung enthalten.

$\$a^2 + \backslash fbox\{\$x^2\$} = c^2\$$ $a^2 + \boxed{x^2} = c^2$

Der Abstand des Rahmens vom umgebenden Text kann mit dem *hspace*-Befehl vergrößert werden, dessen Pflichtargument den Zusatzabstand angibt.

wenn $\backslash hspace\{2mm\}\backslash fbox\{\$x=y^2\$}\backslash hspace\{2mm\}$ gilt wenn $\boxed{x = y^2}$ gilt

Entsprechend erhält man eine farbige Hinterlegung mit dem *colorbox*-Befehl:

$\$a^2 + \backslash colorbox\{yellow\}\{\$x^2\$} = c^2\$$ $a^2 + \boxed{x^2} = c^2$

Der *fcolorbox*-Befehl fügt noch einen andersfarbigen Rahmen hinzu:

$\$a^2 + \backslash fcolorbox\{red\}\{yellow\}\{\$x^2\$} = c^2\$$ $a^2 + \boxed{x^2} = c^2$

Unterstreichungen sind auch im *math*-Modus mit dem *underline*-Befehl möglich:

$\$a^2 + \backslash underline\{10\ x^2\} = c^2\$$ $a^2 + \underline{10x^2} = c^2$

Hoch- und Tiefstellung eines Zeichens wird durch \wedge beziehungsweise $_$ eingeleitet. Zeichenketten müssen als Block in geschweiften Klammern stehen

$$\text{\$x_1 +x_{99}= x_{50}\{\}\^2\$} \qquad x_1 + x_{99} = x_{50}^2$$

Auf der rechten Seite der Gleichung wurde vor der Hochstellung ein leerer Block $\{\}$ eingefügt, damit tief- und hochgestellte Zeichen nicht in einer Spalte (x_{50}^2), sondern horizontal versetzt (x_{50}^2) stehen. Hoch- und Tiefstellungen können auch vor einem Zeichen stehen, indem man einen leeren Block voranstellt. Mehrfache Hoch- oder Tiefstellungen sind möglich.

$$\text{\$}\^{\sim{13}}_{{\sim{6}}}\text{\$C}\^{\sim{\sim}}\text{\$f''} = \text{\$a}\^{\sim{-x^2}}\text{\$} \qquad {}^{13}_6\text{C} \quad f'' = a^{-x^2}$$

Die Ableitungsstriche bestehen in einer *math*-Umgebung aus Apostrophzeichen. Man erhält sie genauso f'' mit $\text{\$f}\^{\sim{\prime}\sim{\prime}}\text{\$}$. Eine höher gestellte, aber nicht empfohlene Alternative f'' gibt es mit $\text{\$f}\^{\sim{\prime\prime}}\text{\$}$.

Der Apostroph ist in einer *math*-Umgebung ein sogenanntes aktives Zeichen. Dies bedeutet, dass eine Folge von Apostrophen zu einem Symbol für eine mehrfache Ableitung wird. Das Symbol einer zweifachen Ableitung ist also nicht zweimal das Symbol einer einfachen Ableitung.

Brüche und Dezimalzahlen. Der Befehl $\text{\backslashfrac}\{\dots\}\{\dots\}$ erzeugt einen Bruch. Mit dem Paket *units* (oder seiner Teilmenge *nicefrac*) kann man Brüche in einer normalen Textzeile oder in einer *math*-Umgebung mithilfe des Befehls $\text{\backslashnicefrac}\{\dots\}$ kompakter schreiben. Es folgen einige [Beispiele](#).

Hässlich ist $\text{\$18/37\$}$ und	Hässlich ist 18/37 und der
der Bruch $\text{\$\frac{18}{37}\$}$ ist zu hoch,	Bruch $\frac{18}{37}$ ist zu hoch, aber
aber $\text{\$\nicefrac{18}{37}\$}$ passt besser,	$\frac{18}{37}$ passt besser, im Text-
im Textmodus ($\text{\backslashnicefrac{18}{37}}$) auch.	modus ($\frac{18}{37}$) auch.

Soll ein Bruch in einer *math*-Umgebung so groß geschrieben werden wie in einer *displaymath*-Umgebung, braucht man den *displaystyle*-Befehl.

$$\text{\$\displaystyle\frac{18}{37}\$} \text{ statt } \text{\$\frac{18}{37}\$} \qquad \frac{18}{37} \text{ statt } \frac{18}{37}$$

Die Zeilenabstände vergrößern sich dadurch natürlich. Das Kürzen von Brüchen lässt sich mit Durchstreichungen veranschaulichen, wenn das Paket *cancel* von Donald Arseneau geladen wurde.

$$\text{\$\frac{70}{130} = \frac{\cancel{2} \cdot \cancel{5}}{\cancel{2} \cdot 7 \cdot \cancel{5} \cdot 13}\$} \qquad \frac{70}{130} = \frac{\cancel{2} \cdot \cancel{5} \cdot 7}{\cancel{2} \cdot \cancel{5} \cdot 13}$$

Wenn man Kettenbrüche mit einem *frac*-Befehl schreibt, werden die Teilzähler und Teilnenner höherer Glieder immer kleiner. Eine gleichbleibende Größe erhält man dagegen, wenn das Paket *amsmath* geladen ist, mit dem *cfrac*-Befehl, wie folgendes [Beispiel](#) zeigt:

```
\[ \sqrt{2} =
1 + \frac{1}{2 + \frac{1}{2 + \frac{1}{2 + \dots}}}]
```

$$\sqrt{2} = 1 + \frac{1}{2 + \frac{1}{2 + \frac{1}{2 + \dots}}}$$

```
\[ \sqrt{2} =
1 + \cfrac{1}{2 + \cfrac{1}{2 + \cfrac{1}{2 + \cfrac{1}{\ddots}}}} \]
```

$$\sqrt{2} = 1 + \frac{1}{2 + \frac{1}{2 + \frac{1}{2 + \frac{1}{\ddots}}}}$$

Das Sonderzeichen \ddots wird mit dem Befehl `\ddots` geschrieben.

In der *math*-Umgebung wird ein Komma normalerweise als Aufzählungsoperator mit nachfolgendem kleinem Abstand dargestellt. Das gilt auch für Dezimalzahlen, die, wie im Deutschen üblich, mit Dezimalkomma geschrieben werden sollen. In deutschen Dezimalzahlen muss man daher `{,}` als Dezimalkommazeichen schreiben. Obiges Beispiel verwendet das schmale Leerzeichen `\,` als Multiplikationsoperator.

Matrizen und geschweifte Klammern können in der *math*-Umgebung leicht gestaltet werden, wie folgende [Beispiele](#) zeigen. Mit dem Quellcode

```
$\begin{smallmatrix} a_1&a_2 \\ a_3&a_4 \end{smallmatrix} \text{ und} \\ \left( \begin{smallmatrix} x \\ y \\ z \end{smallmatrix} \right)$
```

werden kleine Matrizen ohne und mit Klammern, zum Beispiel $\begin{smallmatrix} a_1 & a_2 \\ a_3 & a_4 \end{smallmatrix}$ und $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$, in der Textzeile erzeugt. Dagegen werden geklammerte große Matrizen mit

```
$\begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix}$
```

erzeugt, was $\begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix}$ geschrieben wird. Mit *matrix* statt *pmatrix* entstehen ent-

sprechend ungeklammerte Matrizen und mit *vmatrix* Determinanten, $\begin{vmatrix} a_1 & a_2 \\ a_3 & a_4 \end{vmatrix}$.

Die Umgebung *cases* ermöglicht eine Fallunterscheidung. Mit

```
$x = \begin{cases} a^{-1} & \text{für } a \neq 0 \\ 0 & \text{sonst} \end{cases}$
```

erhält man $x = \begin{cases} a^{-1} & \text{für } a \neq 0 \\ 0 & \text{sonst} \end{cases}$

Die Befehle `\overbrace` und `\underbrace` erzeugen waagerechte geschweifte Klammern. Zum Beispiel ergibt

```
$\underbrace{1\cdot 2\cdot 3}_6 = \overbrace{1+2+3}^6$
```

$$\underbrace{1 \cdot 2 \cdot 3}_6 = \overbrace{1 + 2 + 3}^6$$

Einfache Rechenoperationen kann man im L^AT_EX-Dokument unter anderem mit dem Paket *fltpoint* von Eckhart Guthohrlein durchführen. Der Befehl `\fpDecimalSign{,}` beziehungsweise `\fpDecimalSign{.}` in der Präambel legt Komma beziehungsweise Punkt als Dezimaltrennzeichen fest.

ungsweise Punkt als Dezimaltrennzeichen fest. Ein Befehlsname kann als Variable dienen, der in der Präambel ein Wert zugewiesen wird.

```
\fpDecimalSign{,} % Dezimaltrennzeichen: ,
\newcommand*{\wertX}{8,8}% Variable: \wertX
```

In der *document*-Umgebung stehen dann verschiedene arithmetische Operatoren zur Verfügung: Addition geht mit dem Befehl `\fpAdd{...}{...}{...}`. Weitere binäre arithmetische Operatoren sind entsprechend `fpSub`, `fpMul` und `fpDiv`. Unäre Operatoren sind `\fpNeg{...}{...}` und entsprechend `fpAbs`. In der ersten geschweiften Klammer steht jeweils ein Befehlsname, welcher als Variable das Ergebnis aufnimmt (im nächsten Beispiel: `\wertA`). In den folgenden ein oder zwei geschweiften Klammern stehen die Operanden, entweder als Zahl oder als Variable (Befehlsname), der ein Wert zugewiesen wurde.

```
\fpDiv{\wertA}{108}{4} Ergebnis: \wertA \par Ergebnis: 27
\fpAdd{\wertB}{\wertX}{2,5} Ergebnis: \wertB Ergebnis: 11,3
```

Außerdem gibt es einen unären Operator zur Rundung einer Zahl, bei dem als weiteres Argument die Zehnerpotenz angegeben wird, zu der gerundet werden soll.

```
\fpRound{\wertC}{3,141}{-1} Ergebnis: \wertC Ergebnis: 3,1
```

Hier noch ein [Beispiel](#):

```
Variable x hat den Wert \wertX.\
\fpSub{\wertY}{\wertX}{6,66} Variable x hat den Wert 8,8.
Ungerundet: x - 6,666 = \wertY\ Ungerundet: x - 6,666 = 2,14
\fpRound{\wertZ}{\wertY}{0} und gerundet ergibt das 2.
und gerundet ergibt das \wertZ.
```

Literatur zu Mathematik mit Latex gibt es auf der Webseite der American Mathematical Society, <http://www.ams.org/publications/authors/tex/amslatex>. Die Seite

<http://ftp.fau.de/ctan/info/short-math-guide/short-math-guide.pdf> ist ebenfalls hilfreich mit einem *Short Math Guide* for \LaTeX .

5.2 Sonderzeichen in einer *math*-Umgebung

wurden teilweise bereits in Abschnitt 4.2 (ab Seite 49) aufgelistet. Man kann zahlreiche Sonderzeichen schreiben (zum Beispiel griechische Buchstaben, siehe oben), die bei der normalen Texteingabe fehlen. Der Quellcode

es gelten `\$c^2=a^2+b^2-2\,a\,b\cos\gamma` und `\$\uppi\approx 3{,}14` ergibt als gedruckte Zeile (der Befehl `\uppi` benötigt das Paket *upgreek*)

es gelten $c^2 = a^2 + b^2 - 2ab \cos \gamma$ und $\pi \approx 3,14$

Es folgen weitere [Beispiele](#). Vorausgesetzt wird das Paket *amsmath*.

Mathematische Operatoren:

–	±	·	*	×	÷	⊗	⊕
--	<code>\pm</code>	<code>\cdot</code>	<code>\ast</code>	<code>\times</code>	<code>\div</code>	<code>\otimes</code>	<code>\oplus</code>

Brüche, Summen und Produkte:

$$\frac{a}{b} \quad a/b \quad \sum_{i=1}^3 x_i \quad \prod_{i=1}^3 i$$

Wurzel, Exponentialfunktion, Binomialkoeffizient und Grenzwert:

$$\sqrt[2]{x} \quad x^{88} \quad \binom{n}{k} \quad \lim_{x \rightarrow \infty}$$

Index, Exponentialfunktion, Logarithmen, Minimum, Maximum, modulo:

$$x_1 \quad \exp \quad \log \quad \ln \quad \min \quad \max \quad \text{mod}$$

Differentiale und Differentialquotienten:

$$dx \quad \frac{dy}{dx} \quad \partial$$

Integrale (*iint* und *iiint* sind nur mit `\usepackage{amsmaths}` verfügbar):

$$\int_{t_0}^{t_{max}} t^2 dt \quad \oint \quad \iint \quad \iiint$$

Mit dem *limits*-Befehl können die oberen und unteren Grenzen eines Integrals über und unter, statt rechts vom Integralzeichen gesetzt werden. [Beispiel](#):

$$\int\limits_a^b f(x) \, dx$$

Entsprechend ist dies auch mit Summen und Produkten möglich.

Relationen:

$$\neq \quad \leq \quad \geq \quad \ll \quad \gg \quad \approx \quad \equiv$$

$$\sim \quad \nsim \quad \propto \quad \in \quad \notin \quad \ni \quad \ngtr$$

Mit `\stackrel{def}{=}` setzt man eine Zeichenkette auf eine Relation: $x \stackrel{def}{=} y$.

Logik:

$$\neg \quad \wedge \quad \vee \quad \rightarrow \quad \leftarrow \quad \forall \quad \exists$$

Geometrie:

$$\triangle \quad \square \quad \parallel \quad \perp \quad \angle \quad \measuredangle$$

Mengenlehre:

$$\cup \quad \cap \quad \setminus \quad \subseteq \quad \not\supseteq \quad \emptyset$$

Trigonometrische Funktionen:

\sin	\arccos	\cot	\arctan	\sinh	\sec	\csc
$\backslash\sin$	$\backslash\arccos$	$\backslash\cot$	$\backslash\arctan$	$\backslash\sinh$	$\backslash\sec$	$\backslash\csc$

Die Funktion arccot fehlt in der *math*-Umgebung.

Komplexe Zahlen:

$r\angle\varphi$	\Re	\Im	$ z = 1$	$\overline{a+b}$
$\backslashr\angle\backslashvarphi$	$\backslash\Re$	$\backslash\Im$	$\backslashleft z\right =1$	$\backslash\overline{a+b}$

Die imaginäre Einheit ist i mit i oder j mit j .

Weitere mathematische Symbole:

∇	Δ	∞	$ $	\parallel	\mid	\hbar
$\backslash\nabla$	$\backslash\Delta$	$\backslash\infty$	$\backslash\vert$	$\backslash\Vert$	$\backslash\mid$	$\backslash\hbar$

Betrag und Norm:

$ +1 = -1 $	$\ -x\ = \ x\ $
$\backslash\lvert+1\rvert=\backslash\lvert-1\rvert$	$\backslash\lVert-x\rVert=\backslash\lVertx\rVert$

Diakritische Zeichen:

\acute{x}	\grave{x}	\bar{x}	\check{x}
$\backslash\acute{x}$	$\backslash\grave{x}$	$\backslash\bar{x}$	$\backslash\check{x}$
\tilde{x}	\widetilde{abcde}	\hat{x}	\widehat{abcde}
$\backslash\tilde{x}$	$\backslash\widetilde{abcde}$	$\backslash\hat{x}$	$\backslash\widehat{abcde}$
\dot{x}	\ddot{x}	\vec{x}	\overrightarrow{AB}
$\backslash\dot{x}$	$\backslash\ddot{x}$	$\backslash\vec{x}$	$\backslash\overrightarrow{AB}$

Weitere „Bedeckungen“ sind \ddot{x} mit \ddot{x} , $\dot{\vec{x}}$ mit $\dot{\vec{x}}$ und $\ddot{\vec{x}}$ mit $\ddot{\vec{x}}$, \breve{x} mit \breve{x} , sowie \mathring{x} mit \mathring{x} . Mithilfe des *ddot*-Befehls kann man auch deutsche Umlaute in einer *math*-Umgebung bilden (obwohl das nicht empfehlenswert ist):

$\ddot{A}\ddot{O}\ddot{U}\ddot{a}\ddot{o}\ddot{u}$ $\ddot{A}\ddot{O}\ddot{U}\ddot{a}\ddot{o}\ddot{u}$

Für diakritische Zeichen auf i und j gibt es diese Buchstaben auch ohne Punkt: \imath mit \imath und \jmath mit \jmath .

Mit `\usepackage{amsfonts}` verfügbare Symbole für Zahlenmengen:

\mathbb{N}	\mathbb{Z}	\mathbb{R}	\mathbb{C}
$\backslash\mathbb{N}$	$\backslash\mathbb{Z}$	$\backslash\mathbb{R}$	$\backslash\mathbb{C}$

Mit `\usepackage{amsfonts}` verfügbare hebräische Buchstaben und andere Symbole:

\aleph	\beth	\gimel	\daleth	\boxdot	\boxminus
\boxplus	\boxtimes	\Subset	\Supset	\Cap	\Cup
\circledcirc	\leqq	\geqq	\lneqq	\nexists	\nparallel

Ein Doppelpunkt kann als Relation $a : b$ mit $\$a:b\$$ geschrieben werden oder mit anderem Zeichenabstand als Interpunktion $f : x \rightarrow y$ mit $\$f\colon x\to y\$$. Mathematische Symbole, die einen Doppelpunkt enthalten, werden schön mit dem Paket *colonequals* von Heiko Oberdiek dargestellt, sowohl in normalem Text als in einer *math*-Umgebung (als Relation). Wer den Unterschied zwischen $:=$ und \coloneqq nicht sieht, braucht das Paket natürlich nicht. **Beispiele:**

$::$	\coloneqq	\equiv	\colonminus	\colonapprox
$\backslash coloncolon$	$\backslash colonequals$	$\backslash equalscolon$	$\backslash colonminus$	$\backslash colonapprox$

Das Paket *shuffle* von Antoine Lejay und Julian Gilbey definiert zwei Symbole für das Shuffle-Produkt, welches bei algebraischen Rechnungen verwendet wird.

Shuffle-Produkt	$\quad \backslash shuffle$	Shuffle-Produkt	\sqcup
Vollst. Shuffle-Produkt	$\quad \backslash cshuffle$	Vollst. Shuffle-Produkt	$\sqcup\sqcup$

Halloween hat eine beliebte Symbolik, mit der sich auch fürchterliche Mathematik betreiben lässt. Dazu dient das Paket *halloweenmath* von G. Mezzetti. Vorausgesetzt, die nötigen Pakete können mit

```
\usepackage[pdftex]{pict2e}
\usepackage{halloweenmath}
```

in der Präambel (nach dem Paket *amsmath*) geladen werden, stehen die im folgenden **Beispiel** gezeigten gruseligen Symbole in einer *math*-Umgebung zur Verfügung.

Kürbisse: $\backslash pumpkin+\pumpkin=\bigpumpkin$	Kürbisse: $\odot + \odot = \odot$
$\par\smallskip \$\mathwitch f(x^2)=y \quad$	$\mathwitch f(x^2)=y \quad a+b \succ c$
$a+b\rightbroom c \quad \par\medskip \\bigskull	
$\quad \$\xrightswishingghost\{\text{Geist}\}$	Geist
$\quad \mathghost \quad \par\smallskip \\mathcloud	$\skull \quad \ghost \quad \ghost$
$\quad \mathbat \quad \quad \mathwitch* \$$ mit Katze	$\cloud \quad \cat \quad \ghost$ mit Katze

Das Paket enthält noch weitere Varianten der soeben vorgestellten Symbole.

5.3 Gleichungen

erlauben die Einbettung komplexer mathematischer Aussagen in das Dokument. Dafür wird mit $\backslash begin\{equation\}$ und $\backslash end\{equation\}$ eine *equation*-Umgebung gebildet, innerhalb der die Schreibweise der *math*-Umgebung entspricht, die in Abschnitt 5.1 (Seite 57) beschrieben wird. Da Gleichungen in einer eigenen Zeile stehen, können sie umfangreichere mathematische Ausdrücke wiedergeben. Einige Befehle, zum

Beispiel $\sum_{u \in o}$, werden in einer Gleichung ausladender geschrieben als in einer *math*-Umgebung.

Gleichungen werden automatisch nummeriert. Wurde das Paket *amsmath* mit `\usepackage{amsmath}` geladen, hat man zusätzliche mathematische Schreibweisen zur Verfügung und kann auch nicht nummerierte Gleichungen bilden, indem man ein `*` an *equation* hängt. Folgende [Beispiele](#) verwenden einige der oben aufgeführten mathematischen Sonderzeichen.

```
\begin{equation*} % nicht nummerierte Gleichung
\text{Es sei } \Delta x = \left(\frac{b-a}{n}\right) \end{equation*}
```

ergibt als Druckbild:

$$\text{Es sei } \Delta x = \left(\frac{b-a}{n}\right)$$

Durch den Befehl `\text` kann normaler Text eingebettet werden. Mit `\left(` und `\right)` werden runde Klammern gesetzt, deren Größe automatisch an den Inhalt angepasst wird. Entsprechendes gilt für eckige `[]` und geschweifte Klammern `{ }`.

```
\begin{equation}\label{riemann} % nummerierte Gleichung mit Marke
\int_a^b f(x)\,dx = \lim_{n \rightarrow \infty} \sum_{k=1}^n f(x_k) \Delta x
\end{equation}
```

ergibt als Druckbild:

$$\int_a^b f(x) \, dx = \lim_{n \rightarrow \infty} \sum_{k=1}^n f(x_k) \Delta x \quad (5.1)$$

Hier wurde zum Trennen von Faktoren mit `\`, ein schmales Leerzeichen gesetzt. Mithilfe der Marke, die mit dem Befehl `\label{riemann}` gebildet wurde, kann später auf die Gleichung verwiesen werden. Der Quellcode

siehe Gleichung `\eqref{riemann}` zur Flächenberechnung

ergibt

siehe Gleichung (5.1) zur Flächenberechnung

Es gibt verschiedene Möglichkeiten mehrzeilige Gleichungen zu schreiben. Man benötigt dafür das Paket *amsmath* und in allen Fällen gilt: Mit `\\` wird eine Zeile beendet und in der nächsten Zeile eine weitere Gleichung begonnen. In der letzten Zeile wird `\\` weggelassen.

Ist eine Gleichung länger als eine Zeile, kann man sie (statt in eine *equation*- oder *equation**-Umgebung) in eine *multline*- (für eine nummerierte Gleichung) beziehungsweise *multline**-Umgebung (ohne Nummerierung) stellen. Die erste Zeile wird linksbündig, die folgenden zentriert, die letzte rechtsbündig gesetzt. [Beispiel](#):

```
\begin{multline}
y(x) = 88x^8 + 7x^7 + 666x^6 + 5x^5 + 42x^4 \\
+ 33x^3 + 22x^2 + 11x - 123
\end{multline}
```

$$y(x) = 88x^8 + 7x^7 + 666x^6 + 5x^5 + 42x^4 + 33x^3 + 22x^2 + 11x - 123 \quad (5.2)$$

Eine Alternative ist die *split*-Umgebung. Sie wird in eine *equation*- oder *equation**-Umgebung gesetzt. Hier kann man die Ausrichtung steuern, indem man die Stellen, welche direkt untereinander stehen sollen, mit dem Steuerzeichen `&` markiert. In folgendem [Beispiel](#) werden die Zeilen am Zeichen `=` ausgerichtet.

```
\begin{equation} \begin{split}
y&=x^3\,,(23x^8+13x^4+18x^2+21)\backslash\backslash & y = x^3(23x^8 + 13x^4 + 18x^2 + 21) \\
&=21x^3+18x^5+13x^7+23x^{11} & = 21x^3 + 18x^5 + 13x^7 + 23x^{11} \quad (5.3)
\end{split} \end{equation}
```

Um mehrere Gleichungen zentriert untereinander zu schreiben, kann man sie in eine *gather*-Umgebung (mit Nummerierung) oder eine *gather**-Umgebung (ohne Nummerierung) setzen. [Beispiel](#):

```
\begin{gather}
5 x_2 + 7 x_3 = 9 k \backslash\backslash & 5x_2 + 7x_3 = 9k \quad (5.4) \\
3 x_1 + 5 x_2 = x_3 + 10 & 3x_1 + 5x_2 = x_3 + 10 \quad (5.5)
\end{gather}
```

Sollen mehrere Gleichungen untereinander geschrieben und so ausgerichtet werden, dass beispielsweise die Gleichheitszeichen untereinander stehen, verwendet man eine *align*-Umgebung (mit Nummerierung) oder eine *align**-Umgebung (ohne). [Beispiel](#):

```
\begin{align}
& k \& = 8 \backslash\text{nonumber} \backslash\backslash & k = 8 \\
2 x_1 + 8 x_3 \& = 3 k \backslash\backslash & 2x_1 + 8x_3 = 3k \quad (5.6) \\
5 x_2 + 7 x_3 \& = 9 k \backslash\backslash & 5x_2 + 7x_3 = 9k \quad (5.7) \\
3 x_1 + 5 x_2 \& = x_3 + 10 & 3x_1 + 5x_2 = x_3 + 10 \quad (5.8)
\end{align}
```

Die Ausrichtung erfolgt an den Zeichen, welche dem Steuerzeichen & folgen. Der Befehl `\nonumber` am Ende einer Zeile verhindert deren Nummerierung.

Das Paket *witharrows* von François Pantigny erlaubt es, untereinander geschriebene Gleichungen ohne Nummerierung mit einem Pfeil am rechten Rand und einer Erklärung zu versehen, wie folgendes [Beispiel](#) zeigt.

```
\begin{WithArrows}
x \& = (a + b)^2 & x = (a + b)^2 \\
\Arrow{ausmultiplizieren} \backslash\backslash & = a^2 + 2ab + b^2 \quad \rangle \text{ausmultiplizieren} \\
& \& = a^2 + 2ab + b^2
\end{WithArrows}
```

Da die Pfeile mit TikZ (siehe unten) gezeichnet werden, können sie (ebenso wie der erklärende Text) anders (beispielsweise farbig) gestaltet werden.

Mit dem Paket *framed* von Donald Arseneau (siehe Abschnitt 3.2 auf Seite 35) können auch Gleichungen eingerahmt (*framed*-Befehl) oder farbig hinterlegt werden (*snugshade*-Befehl oder *shaded*-Befehl). Für letzteres muss man mit dem Befehl

```
\definecolor{shadecolor}{named}{yellow} % Farbe für shaded-Umgebungen
```

in der Präambel, nach Laden des ebenfalls benötigten Pakets *xcolor*, die Hintergrundfarbe (hier: gelb) festlegen. [Beispiele](#):

```
\begin{framed} \begin{equation}
m = \frac{m_0}{\sqrt{1 - (v/c)^2}}
\end{equation} \end{framed}
```

$$m = \frac{m_0}{\sqrt{1 - (v/c)^2}} \quad (5.9)$$

```
\begin{snugshade} \begin{equation}
m = \frac{m_0}{\sqrt{1 - (v/c)^2}}
\end{equation} \end{snugshade}
```

$$m = \frac{m_0}{\sqrt{1 - (v/c)^2}} \quad (5.10)$$

6 Grafiken mit TikZ

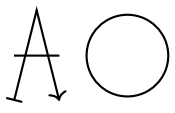
Innerhalb eines L^AT_EX-Dokuments kann man mit entsprechenden Befehlen Vektorgrafiken erzeugen, wenn das Paket *tikz* von Till Tantau und Christian Feuersänger mit `\usepackage{tikz}` geladen wurde. TikZ stützt sich auf das Graphikpaket PGF und ist sehr umfangreich. Hier folgt nur eine Einführung anhand von Beispielen, aber es gibt zum Glück eine hervorragende [Anleitung](#).

TikZ stellt eine Zeichenfläche bereit, die durch ein zweidimensionales kartesisches Koordinatensystem beschrieben wird, dessen Ursprung (0,0) in der linken unteren Ecke der Fläche liegt. Wenn nichts anderes angegeben wird, ist 1 cm die Längeneinheit. Man sollte bei der Platzierung grafischer Elemente andere Längeneinheiten möglichst vermeiden und die Gesamtgröße der Grafik bei Bedarf durch Skalierung mit der Option *scale* anpassen. Sehr kleine Grafiken, zum [Beispiel](#) ●, können mit `\tikz{ ... }` in eine Textzeile eingebaut werden:

... Beispiel `\tikz{\fill[red] (0,0) circle[radius=1mm];}`, können ...

Das Paket *tikz* stellt die Farben des Pakets *xcolor* zur Verfügung (siehe Abschnitt 4.1 auf Seite 46). Aufwendigere Grafiken werden übersichtlicher in einer Umgebung beschrieben, welche mit `\begin{tikzpicture}` beginnt und mit `\end{tikzpicture}` endet.

6.1 Strichzeichnungen und Füllungen

In folgendem [Beispiel](#) wird mit drei *draw*-Befehlen eine Grafik  gezeichnet. Jeder *draw*-Befehl wird mit einem Semikolon abgeschlossen. Der erste *draw*-Befehl zeichnet eine einzelne Strecke, der zweite einen Pfad, der aus zwei Strecken zusammengesetzt ist. Die Zahlenpaare in runden Klammern geben kartesische Koordinaten an. Ohne Option zur Linienart erzeugt -- eine Strecke mit durchgezogener Linie. Die Option `|->` in eckigen Klammern nach *draw* erzeugt einen Querstrich am Anfang und einen Rechtspfeil am Ende des Pfades. Der dritte *draw*-Befehl zeichnet einen Kreis, dessen Mittelpunkt bei (5,2) liegt und dessen Radius 18 mm beträgt.

```
\begin{tikzpicture} [thick, scale=0.3]
\draw (0,2) -- (2,2) ;
\draw[|->] (0,0)--(1,4)--(2,0) ;
\draw (5,2) circle (18mm);
\end{tikzpicture}
```



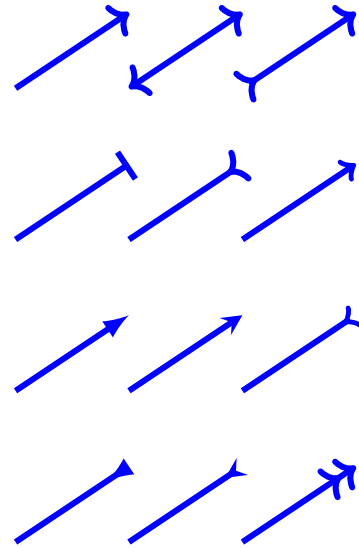
In den eckigen Klammern nach `\begin{tikzpicture}` stehen Optionen, mit denen das Erscheinungsbild verändert wird: *thick* ergibt dickere Striche und mit *scale* wird ein Maßstabsfaktor angegeben, um die Gesamtgröße der Grafik anzupassen.

Die Pfeilspitzen sind in der Standardeinstellung von TikZ ziemlich klein. Ihre Größe wird zwar an die Linienbreite angepasst, aber man sollte doch in der Präambel, nach `\usepackage{tikz}`, die Zeile

```
\usetikzlibrary{arrows.meta}
```

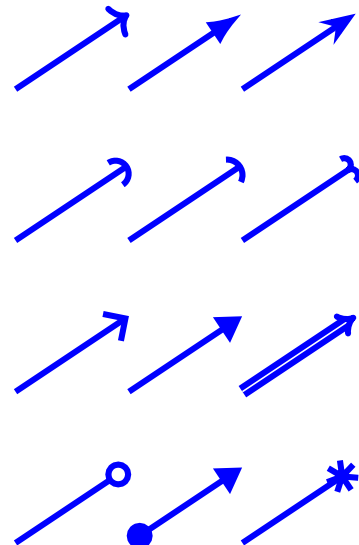
einfügen (dies wurde im vorliegenden Manuskript getan) und erhält dann einige etwas größere Pfeilspitzen. Folgendes [Beispiel](#) zeigt diejenigen Pfeile, welche TikZ bereits ohne *arrows.meta* bereitstellt, deren Spitzen aber mit *arrows.meta* teilweise größer gezeichnet werden.

```
\begin{tikzpicture} [line width=0.8mm,blue]
\draw[->] (0,0) -- (1.5,1) ;
\draw[<->] (1.5,0) -- (3,1) ;
\draw[>->] (3,0) -- (4.5,1) ;
\draw[-|] (0,-2) -- (1.5,-1) ;
\draw[-<] (1.5,-2) -- (3,-1) ;
\draw[-to] (3,-2) -- (4.5,-1) ;
\draw[-latex] (0,-4) -- (1.5,-3) ;
\draw[-stealth] (1.5,-4) -- (3,-3) ;
\draw[-to reversed] (3,-4) -- (4.5,-3);
\draw[-latex reversed] (0,-6)--(1.5,-5);
\draw[-stealth reversed] (1.5,-6)--(3,-5);
\draw[->>] (3,-6) -- (4.5,-5) ;
\end{tikzpicture}
```



Außerdem hat man mit *arrows.meta* mehr Pfeilarten, zum [Beispiel](#):

```
\begin{tikzpicture} [line width=0.8mm,blue]
\draw[-To] (0,0) -- (1.5,1) ;
\draw[-Latex] (1.5,0) -- (3,1) ;
\draw[-Stealth] (3,0) -- (4.5,1) ;
\draw[-Arc Barb] (0,-2) -- (1.5,-1) ;
\draw[-Parenthesis] (1.5,-2) -- (3,-1) ;
\draw[-Hooks] (3,-2) -- (4.5,-1) ;
\draw[-Straight Barb] (0,-4) -- (1.5,-3) ;
\draw[-Triangle] (1.5,-4) -- (3,-3) ;
\draw[-Implies,double] (3,-4) -- (4.5,-3);
\draw[-{Circle[fill=none]}] (0,-6)--(1.5,-5);
\draw[Circle-Triangle] (1.5,-6)--(3,-5);
\draw[-{Rays[n=7]}] (3,-6) -- (4.5,-5) ;
\end{tikzpicture}
```



Überdies kann man die Pfeilspitzen noch selbst verändern, etwa so:

```
\begin{tikzpicture}[thick]
\draw[ ->,
>={ Stealth[width=3mm,length=4mm] } ]
(0,0) -- (2,1) ;
\end{tikzpicture}
```



Die Option *Stealth* gibt die Form des Pfeils an; eine Alternative ist *Latex*. Die Parameter *width* und *length* kann man natürlich nach Belieben ändern.

Für die Linienstärke sind folgende Namen möglich: *ultra thin*, *very thin*, *thin*, *semithick*, *thick*, *very thick* und *ultra thick*. Alternativ kann die Linienstärke auch durch einen Wert festgelegt werden, zum Beispiel mit *line width=2mm*.

Ein Pfad oder *path* ist eine Aneinanderreihung von Strecken oder (mit aufwändigerer Formatierung) kurvenförmigen Linien. Mit *cycle* kann er zu seinem Anfang zurückgeführt werden und bildet dann einen Zyklus. Die von einem Zyklus eingeschlossene Fläche kann gefärbt werden. [Beispiel](#):

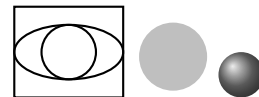
```
\begin{tikzpicture} [thick, scale=0.8]
\path[fill=orange]
(0,0)--(0,2)--(2,1)--(2,0)--cycle;
\path[draw, line width=3pt, shade]
(3,0)--(5,2)--(6,0)--cycle;
\end{tikzpicture}
```



Der links gezeigte trapezförmige Zyklus hat keine Umrandung, da die Option *draw* fehlt. Die Option *fill* färbt das Trapez einheitlich in der angegebenen Farbe (orange), während *shade* den grauen Farbverlauf in der rechts gezeigten Fläche erzeugt. Mit der Option *draw* werden die Grenzlinien der Dreiecksfläche gezeichnet. Die Liniendicke kann mit der Option *line width* bestimmt werden, hier in der Längeneinheit pt. Mehr zu Längeneinheiten findet man in Abschnitt 4.1 auf Seite 44.

Nun zeichnen wir ein Rechteck, dessen untere linke Ecke bei (0,0) ist, und dessen obere rechte Ecke bei (48mm, 40mm). Darin liegt eine Ellipse mit Mittelpunkt (2.4,2), horizontalem Radius 2.4 und vertikalem Radius 1.2. Ganz innen befindet sich ein Kreis mit Mittelpunkt bei (2.4,2) und einem Radius von 1.2. Weiterhin wird mit dem *fill*-Befehl ein farbig ausgefülltes Objekt, hier ein Kreis, dargestellt. Schließlich erzeugen wir mit dem *shade*-Befehl einen Farbverlauf, der hier einen grauen Ball erscheinen lässt.

```
\begin{tikzpicture} [thick, scale=0.3]
\draw (0,0) rectangle (48mm,40mm) ;
\draw (2.4,2) ellipse (24mm and 12mm) ;
\draw (2.4,2) circle (12mm) ;
\fill [lightgray] (7,1.8) circle (15mm) ;
\shade[ball color=gray] (10,1) circle (1);
\end{tikzpicture}
```



Ein Kreisbogen (englisch *arc*) wird zum [Beispiel](#) so (hier als Pfeil) gezeichnet:

```
\begin{tikzpicture} \draw[->] (0,0) arc [
start angle=10, end angle=330,
radius=5mm ] ; \end{tikzpicture}
```



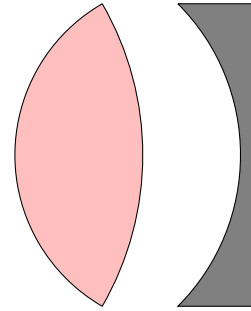
In der runden Klammer stehen die x- und y-Koordinate des Anfangspunkts des Bogens. Der Anfangswinkel *start angle* gibt den Winkel der Strecke an, die Kreismittelpunkt und Anfangspunkt des Bogens verbindet (auch wenn die Strecke nicht gezeichnet wird). Dabei wird der Winkel mathematisch positiv (linksdrehend) von der x-Achse aus gemessen. Entsprechendes gilt für den Endwinkel *end angle*. Der Radius *radius* wird, wie üblich, als Länge gegeben. Statt des Endwinkels kann auch die Differenz von End- und Anfangswinkel *delta angle* angegeben werden. Eine Kurzschreibweise für obigen Zeichenbefehl ist: `\draw[->] (0,0) arc (10:330:5mm);`

Im folgenden [Beispiel](#) werden zwei optische Linsen gezeichnet. Beide haben einen vertikalen Durchmesser von 4 cm. Links ist eine sphärische bikonvexe Sammellinse mit $R_1 = \frac{4}{3}\sqrt{3}$ cm und $R_2 = -4$ cm, rechts eine sphärische konkavplane Zerstreuungslinse mit $R_1 = -2\sqrt{2}$ cm. Beide Linsen werden als geschlossener Linienzug gezeichnet, der rosa beziehungsweise grau gefüllt wird.

```

\begin{tikzpicture}
\draw [fill=pink] (0,2) arc[start angle=120,%
    end angle=240,radius=(4/3)*sqrt(3)]
    arc[start angle=-30,%
    end angle=30,radius=4];
\draw [fill=gray] (1,2)--(2,2)--(2,-2)--(1,-2)
    arc[start angle=-45,%
    end angle=45,radius=2*sqrt(2)];
\end{tikzpicture}

```



Die Verbindung zwischen zwei Punkten muss keine gerade Strecke, sondern kann eine gekrümmte Linie sein. In folgendem Beispiel werden die Winkel, unter denen eine Linie ihren Anfangspunkt verlässt beziehungsweise ihren Endpunkt erreicht, mit einer Winkelangabe in Grad vorgegeben. Da `--` stets eine gerade Strecke ergibt, schreibt man stattdessen *to* zwischen Anfangs- und Endpunkt.

```

\begin{tikzpicture} [thick]
\draw[out=90, in=90, dashed] (0,0) to (2,1) ;
\draw[out=30, in=-90, dotted] (1,0) to (3,1) ;
\end{tikzpicture}

```



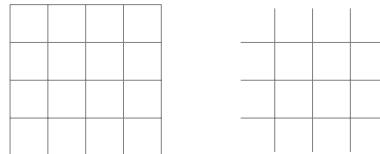
Die Optionen *dashed* und *dotted* erzeugen eine gestrichelte beziehungsweise gepunktete Linie. Weitere Linienarten sind *loosely dashed*, *densely dashed*, *loosely dotted* und *densely dotted*.

Bei der Entwicklung zusammengesetzter Grafiken kann es hilfreich sein, ein Raster mit definierten Abständen zu zeigen. Vor und nach *grid* werden die Koordinaten der linken unteren beziehungsweise der rechten oberen Ecke festgelegt. Die Schrittweite der Rasterlinien wird mit der Option *step* angegeben.

```

\begin{tikzpicture}[help lines]
\draw [step=0.5]
(-1,-1) grid (1,1) ;
\draw [step=0.5]
(2.05,-0.95) grid (3.95,0.95) ;
\end{tikzpicture}

```



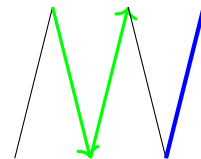
Im Beispiel hat das linke Raster einen Außenrand, das rechte jedoch nicht, weil deren Begrenzungen keine Vielfachen der Schrittweite sind. Die Option *help lines* bewirkt, dass Linien dünn und grau gezeichnet werden.

In folgendem Beispiel wird die Formatierung der Grafik in einem Teilbereich geändert. Dieser beginnt mit `\begin{scope}` und endet mit `\end{scope}`.

```

\begin{tikzpicture}[thin, scale=0.5]
\draw (0,0) -> (1,4) ;
\begin{scope}[very thick, green, ->]
\draw (1,4) -- (2,0) ;
\draw (2,0) -- (3,4) ;
\end{scope}
\draw (3,4) -- (4,0) ;
\draw[ultra thick, blue] (4,0) -- (5,4) ;
\end{tikzpicture}

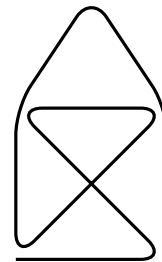
```



In eckigen Klammern stehen nach `\begin{scope}` Optionen, die nur im Teilbereich gelten sollen (hier *very thick*, *green* und `->` für eine dicke grüne Linie mit Pfeilspitze rechts). Die Optionen *ultra thick* und *blue* (die, in eckigen Klammern, direkt hinter einem `draw`-Befehl stehen) wirken nur auf die Strecke, die durch diesen `draw`-Befehl beschrieben wird.

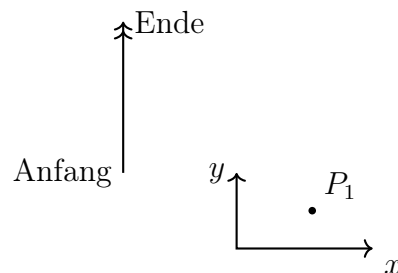
Relative Koordinaten, bezogen auf die letzte Position, können angegeben werden, indem zwei Pluszeichen vor das Koordinatenpaar gesetzt werden. Mit $(1,1) -- ++(1,1)$ wird eine Linie von $(1,1)$ zum Punkt $(2,2)$ gezogen. Folgendes [Beispiel](#) zeigt eine Strichzeichnung mit gerundeten Ecken.

```
\begin{tikzpicture}[
very thick,
rounded corners=10pt,
scale=0.5 ]
\draw (0,0)  --  ++(4,0) --  ++(-4,4)
--  ++(4,0) --  ++(-4,-4) --  ++(0,4)
--  ++(2,3) --  ++(2,-3) --  ++(0,-4) ;
\end{tikzpicture}
```



Mit dem Befehl `node` kann man einem, Knoten genannten, Ort im Koordinatensystem der Grafik eine Zeichenkette zuweisen, welche an dieser Stelle erscheint. Dies soll anhand einer (mit Doppelpfeil versehenen) Strecke gezeigt werden, deren Anfang und Ende wir beschriften, sowie anhand eines beschrifteten Punkts in einem kleinen Achsenkreuz.

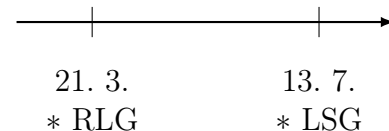
```
\begin{tikzpicture} [thick]
\draw [->>] (0,1) -- (0,3) ;
\node [left] at (0,1) {Anfang} ;
\node [right] at (0,3) {Ende} ;
\draw [->] (1.5,1)--(1.5,0)--(3.3,0);
\node [below right] at (3.3,0) {$x$};
\node [left] at (1.5,1) {$y$};
\fill (2.5,0.5) circle (0.5mm) ;
\node[above right] at (2.5,0.5) {$P_1$};
\end{tikzpicture}
```



Die Beschriftung eines Knotens erfolgt meistens nach oben (*above*), unten (*below*), links (*left*) oder rechts (*right*) versetzt, um die Linie oder das Objekt am Ort des Knotens nicht zu überschreiben.

Mehrzeilige Beschriftungen, wie in folgendem [Beispiel](#), sind mithilfe von `\\` möglich, wenn die Ausrichtung bestimmt wird (*align=center* oder *align=left*).

```
\begin{tikzpicture}
\draw[-latex,thick] (0,0)--(5,0);
\draw (1,-.2)--(1,.2) (4,-.2)--(4,.2);
\node[align=center,below] at (1,-0.5)
{21. 3.\\ \textasteriskcentered~RLG};
\node[align=center,below] at (4,-0.5)
{13. 7.\\ \textasteriskcentered~LSG};
\end{tikzpicture}
```



Verschiedene geometrische Formen, zum Beispiel Sterne und Zylinder, sind verfügbar wenn man in der Präambel, nach `\usepackage{tikz}`, die Zeile

`\usetikzlibrary{shapes.geometric}`

einfügt. Damit kann man zum [Beispiel](#) die chinesische Flagge oder eine zylindrische Scheibe zeichnen. (Die deutsche Flagge wurde bereits auf Seite 47 gezeigt.)

```
\begin{tikzpicture}
[inner sep=0pt,star point ratio=2.617]
  \definecolor{cfr}{HTML}{FF0000}
  \definecolor{cfy}{HTML}{FFFF00}
\fill[cfr] rectangle (30mm, 20mm);
\node[star,fill=cfy, minimum size=6mm,
rotate=0] at (5mm,15mm) {};
\node[star, fill=cfy, minimum size=2mm,
rotate=50] at (10mm,18mm) {};
\node[star, fill=cfy, minimum size=2mm,
rotate=25] at (12mm,16mm) {};
\node[star, fill=cfy, minimum size=2mm,
rotate=0] at (12mm,13mm) {};
\node[star, fill=cfy, minimum size=2mm,
rotate=50] at (10mm,11mm) {};
\end{tikzpicture}
```



```
\begin{tikzpicture}
\node[cylinder,draw=black,fill=blue!15,
minimum width=3cm, minimum height=1cm,
aspect=0.8, shape border rotate=90] {};
\end{tikzpicture}
```



Sollen Objekte mehrfach gezeichnet werden, kann es sinnvoll sein, eine Schleife zu programmieren. Die Schleife beginnt mit dem Befehl `\foreach` und einer Variablen, zum Beispiel `\x`, gefolgt von `in` und einer Liste in geschweiften Klammern. Danach folgt ein Befehl, der mit einem Semikolon abgeschlossen wird, oder ein Befehlsblock in geschweiften Klammern.

```
\begin{tikzpicture}
\foreach \x in {1,1.5,2,3} \draw (\x,0) circle (0.3);
\end{tikzpicture}
```



Endliche Zahlenfolgen, deren Glieder mit konstantem Abstand angeordnet sind, können mit `...` beschrieben werden.

```
\begin{tikzpicture}
\foreach \x in {1,3,...,9}
{
\draw (\x/2,0) circle (0.3);
\node at (\x/2,0) {\x};
}
\end{tikzpicture}
```

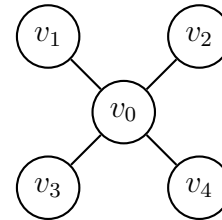


Linien zur Bemaßung in technischen Zeichnungen stellt das Paket [dimline](#) von Sébastien Gross als Ergänzung von *tikz* bereit.

6.2 Graphen und Funktionen

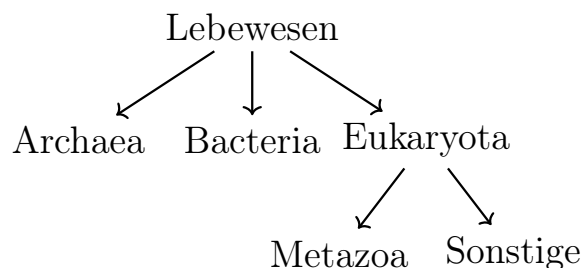
Graphen im Sinne der Graphentheorie sind mathematische Objekte, die Knoten und Verbindungen enthalten. Ein Graph wird gezeichnet, indem zunächst alle Knoten benannt und dargestellt, und dann die Verbindungen gesetzt werden.

```
\begin{tikzpicture} [thick,
every node/.style={circle,draw}]
\node (v0) at (0,0) {$v_0$} ;
\node (v1) at (-1,1) {$v_1$} ;
\node (v2) at (1,1) {$v_2$} ;
\node (v3) at (-1,-1) {$v_3$} ;
\node (v4) at (1,-1) {$v_4$} ;
\draw (v0)--(v1) (v0)--(v2)
      (v0)--(v3) (v0)--(v4) ;
\end{tikzpicture}
```



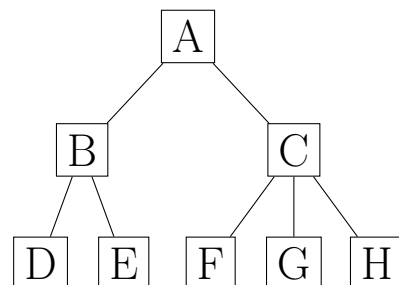
In obigem [Beispiel](#) steht nach `\node` in Klammern ein Knoten-Name, den man beim Setzen der Verbindungen benutzt. Folgendes [Beispiel](#) zeigt ein Baumdiagramm mit gerichteten Verbindungen.

```
\begin{tikzpicture}
[ sibling distance=23mm,
font=\large,
line width=0.3mm,
style={->} ]
\node {Lebewesen}
child { node {Archaea} }
child { node {Bacteria} }
child {
node {Eukaryota}
child {node {Metazoa}}
child {node {Sonstige}}
};
\end{tikzpicture}
```



Ein weiteres [Beispiel](#) eines Baumdiagramms, hier mit rechteckigen Knoten.

```
\begin{tikzpicture} [ font=\Large,
every node/.style={rectangle,draw},
level 1/.style={sibling distance=28mm},
level 2/.style={sibling distance=11mm} ]
\node {A}
child { node {B}
child { node {D} } child { node {E} } }
child { node {C}
child { node {F} } child { node {G} }
child { node {H} } } ;
\end{tikzpicture}
```



Baumdiagramme mit Wurzel (ein ausgezeichnete Knoten an der Spitze) lassen sich gut mit dem Paket *forest* von Sašo Živanović zeichnen, das auf *tikz* aufbaut. Ein Baum wird in einer *forest*-Umgebung definiert und automatisch möglichst kompakt, das heißt: nicht unnötig breit, dargestellt. Die Baumstruktur wird durch verschachtelte eckige Klammern wiedergegeben, wie folgendes [Beispiel](#) zeigt.

```
\begin{forest}
```

```
  [ $V_1$
```

```
    [ $V_{11}$
```

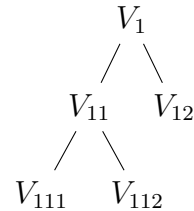
```
      [ $V_{111}$ ]
```

```
      [ $V_{112}$ ]
```

```
    ]
```

```
  [ $V_{12}$ ]      ]
```

```
\end{forest}
```



Zur richtigen Darstellung des Baumdiagramms muss die L^AT_EX-Datei zweimal übersetzt werden. Die Beschriftung der Knoten kann durch Befehle verändert werden. Soll sie Zeichen mit besonderer Bedeutung (zum Beispiel eckige Klammern, Gleichheitszeichen oder Leerzeichen) enthalten, schließt man sie in geschweifte Klammern ein.

Die Knoten können mit den Optionen gestaltet werden, die *tikz* bereitstellt. Das Erscheinungsbild lässt sich mit Optionen verändern, die der Knotenbeschriftung, und einem Komma, folgen. Im [Beispiel](#)

```
\begin{forest}
```

```
  [ {$V_1$ = [Wurzel]},
```

```
    for tree={calign=first}
```

```
      [ V11, circle, draw
```

```
        [ V111, tier=t1 ]
```

```
        [ V112, tier=t1 ] ]
```

```
      [ V12, tier=t1      ]
```

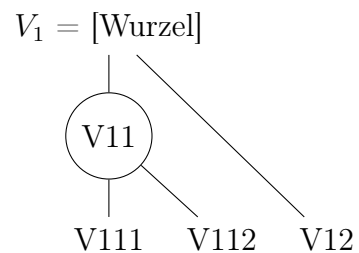
```
    ]
```

```
  \node at (current bounding box.south)
```

```
    [below=3mm]
```

```
    {Ein Beispiel für \emph{forest}.};
```

```
\end{forest}
```



Ein Beispiel für *forest*.

bewirkt die Option *for tree={calign=first}*, dass im ganzen Baum die ersten Kindknoten senkrecht unter ihrem Vorfahren platziert werden. Die mit der Option *tier=t1* versehenen Knoten werden in einer Zeile angeordnet. Dabei ist *t1* ein willkürlich gewählter Name, der für die Knoten der Zeile gleich sein muss. Das Baumdiagramm kann mit einer Unterschrift ergänzt werden, wie oben gezeigt.

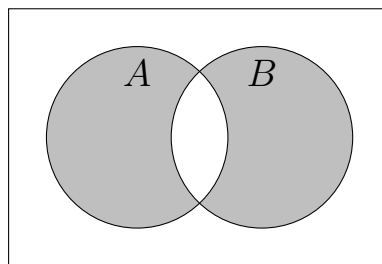
Venn-Diagramme stellen Relationen von Mengen bildlich dar. Das Paket *venn-diagram* von Nicola Talbot zeichnet mit den Umgebungen *venndiagram2sets* und *venndiagram3sets* Venn-Diagramme mit zwei beziehungsweise drei Mengen. [Beispiel](#):

```

\begin{large}
$(A \setminus B) \cup (B \setminus A)$ \quad
\begin{venndiagram2sets}
\fillonlyA \fillonlyB
\end{venndiagram2sets}
\end{large}

```

$$(A \setminus B) \cup (B \setminus A)$$

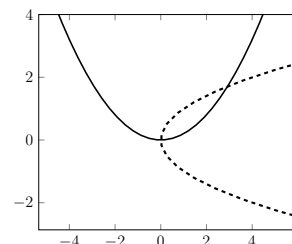


Mathematische Funktionen können berechnet und dargestellt werden, wenn man zusätzlich das Paket *pgfplots* von Christian Feuersänger lädt. Das folgende [Beispiel](#) zeigt mit wenigen Befehlen eine ansprechende Darstellung zweier Kurven.

```

\begin{tikzpicture} [thick, scale=0.5]
\begin{axis}[xmax=6,ymax=4, samples=30]
\addplot[very thick] (x,0.2*x*x);
\addplot[dashed, ultra thick] (x*x,x);
\end{axis}
\end{tikzpicture}

```



Mit einer *axis*-Umgebung wird sehr einfach ein Koordinatenkreuz erzeugt. Es genügt die Angabe der maximalen Achsenwerte und der Anzahl von Funktionswerten, die (pro Funktion) dargestellt werden. Innerhalb der *axis*-Umgebung wird in nur einer Zeile sowohl die Gleichung einer Funktion als auch die zugehörige Formatierung der Kurve festgelegt. Die in eckigen Klammern angegebene Formatierung bestimmt eine Farbe (ohne Angabe bedeutet: *black*) und Linienstärke und -art (ohne Angabe bedeutet: *solid*).

Verschiedene (auch farbige) Symbole für Datenpunkte in Liniendiagrammen stellt das Paket *oplotsymb* von B. Michel Döhring bereit. Es beruht auf TikZ. Die Symbole können als Sonderzeichen in normalem Text oder in einer *math*-Umgebung benutzt werden. Neben den Grundformen sind zahlreiche Varianten verfügbar.

Symbole für wissenschaftliche Diagramme, die mit *oplotsymb* verfügbar sind:

\triangle	\circ	\pentagon	\star	\diamond	\hexagon	\square
<code>\trianglepa</code>	<code>\circlet</code>	<code>\pentago</code>	<code>\starlet</code>	<code>\rhombus</code>	<code>\hexago</code>	<code>\squad</code>

Einige Varianten von `\trianglepa`:

∇	\triangleright	\triangleleft	\triangleup	\blacktriangle
<code>\trianglepb</code>	<code>\trianglepr</code>	<code>\trianglepl</code>	<code>\trianglepadot</code>	<code>\trianglepafill[blue]</code>

Als optionales Argument kann man die Farben des Pakets *xcolor* verwenden.

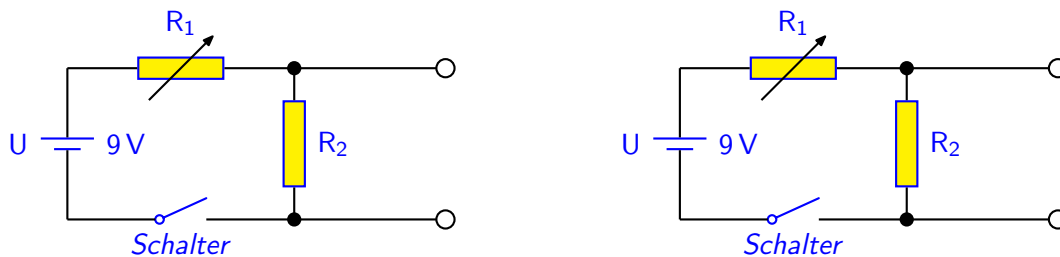
6.3 Elektrische Schaltkreise und Optik

Um Schaltkreise mit TikZ zu zeichnen, kann man die Programmbibliothek *circuits.ee.IEC* oder das Paket *circuitikz* hinzuladen. Erstere ist vielleicht einfacher anzuwenden, letzteres bietet mehr Schaltsymbole, zum Beispiel auch Transistoren.

circuits.ee.IEC ist eine Programmbibliothek für in Deutschland übliche elektrische Symbole, welche man mit dem Befehl `\usetikzlibrary{circuits.ee.IEC}` lädt. Zu Beginn der *tikzpicture*-Umgebung wird die Option *circuit ee IEC*, sowie weitere Optionen für den Stil der Zeichnung angegeben. In der *tikzpicture*-Umgebung stehen die Schaltkreissymbole als *tikz*-Knoten (nodes) zur Verfügung. Zum [Beispiel](#) ergibt

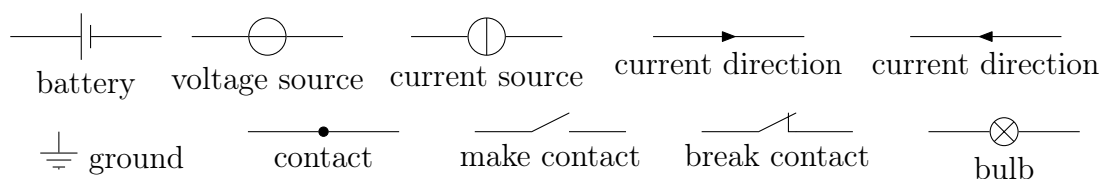
```
\begin{tikzpicture} [                                % Anfang des Optionsblocks
circuit ee IEC,                                % elektrische Schaltungssymbole benutzen
font=\sffamily\small,                        % Schrift ohne Serifen, Größe: small
thick,                                        % Linienbreite: thick
every node/.style=blue, % Stil für nodes, auch Schaltungssymbole
large circuit symbols, % Größe der Schaltungssymbole: large
every info/.style=blue, % Stil der Beschriftungen (info, info')
every resistor/.style={fill=yellow}, % Stil der Widerstände
set make contact graphic= var make contact IEC graphic % Schalter
]                                              % Ende des Optionsblocks
\draw (0,2) to [battery={info={9\,V},info'={U}}] (0,0) ;
\draw (0,2) to [resistor={adjustable}] (3,2) ;
\node at (1.5,2.6) {R$\mathsf{_{1}}$} ;
\draw (3,0) to [resistor={info'={R$\mathsf{_{2}}$}}] (3,2) ;
\draw (0,0) to [make contact={info'={\textit{Schalter}}}] (3,0) ;
\draw[fill=black] (3,0) circle (0.8mm)
(3,2) circle (0.8mm) ;
\draw (3,0)--(5,0) (3,2)--(5,2) ;
\draw[fill=white] (5,0) circle (1.2mm)
(5,2) circle (1.2mm) ;
\end{tikzpicture}
```

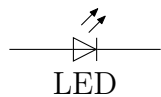
den unten rechts gezeigten Schaltkreis. Der linke Schaltkreis entsteht, wenn der Befehl `\begin{tikzpicture}` nur mit der Option `[circuit ee IEC]` gegeben wird.



Der Befehl zur Erzeugung eines Schaltkreissymbols kann den Schlüssel *info* oder *info'* enthalten, dessen Wert eine Zeichenkette ist, welche als Text über, unter, links oder rechts vom Symbol gezeigt wird (abhängig von der Ausrichtung des Symbols und von *info* beziehungsweise *info'*). Alternativ kann die Beschriftung eines Symbols auch durch einen *node* erfolgen, wie allgemein in *tikz*.

Unter anderem sind folgende Schaltkreissymbole bereits definiert:





Eine Leuchtdiode erhält man mit folgendem Befehl:

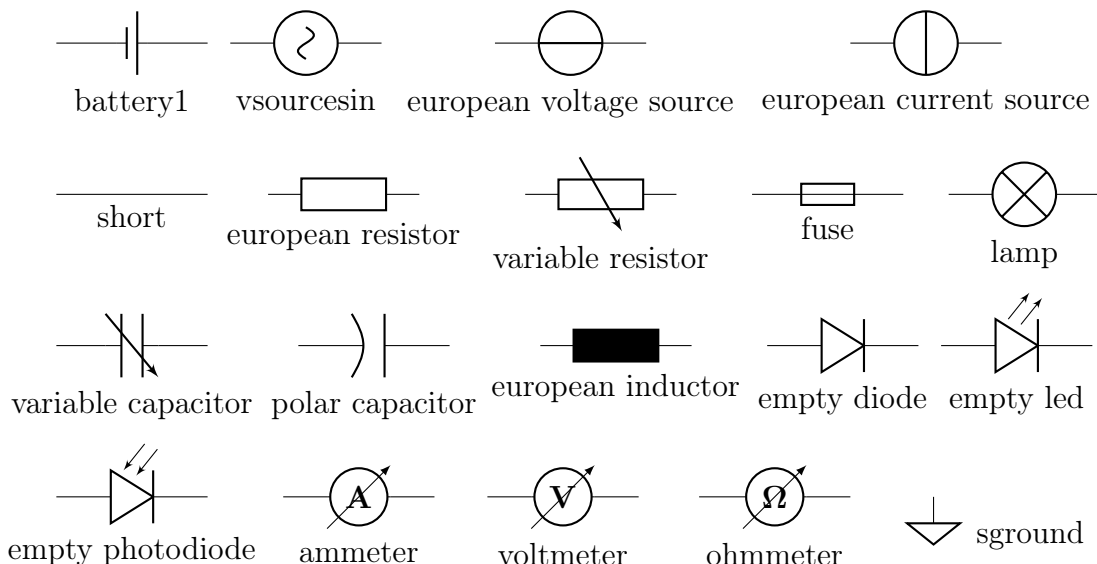
```
\draw (0,0) to [diode={light emitting, info'={LED}}] (2,0) ;
```



Die Schaltkreissymbole kann man selbst definieren:

```
\tikzset{circuit declare symbol = ameter} % Amperemeter definieren
\tikzset{set ameter graphic=
{draw,generic circle IEC,minimum size=5mm,info=center:A}}
\tikzset{circuit declare symbol = vmeter} % Voltmeter definieren
\tikzset{set vmeter graphic=
{draw,generic circle IEC,minimum size=5mm,info=center:V}}
```

CircuiTikZ ist ein Paket von M. A. Redaelli, S. Lindner und S. Erhardt, das (einschließlich der Option für die in Deutschland üblichen elektrischen Symbole) mit `\usepackage[european]{circuitikz}` geladen wird. Damit werden Schaltkreise in einer *circuitikz*-Umgebung gezeichnet. Da das Paket auf TikZ aufbaut, werden die gleichen Befehle benutzt, aber es gibt besondere *nodes* für Schaltsymbole. Zu beachten ist, dass die Symbol-Kurzbezeichnungen des Pakets *circuitikz* nicht verfügbar sind, wenn die Programmbibliothek *circuits.ee.IEC* geladen wurde. Da man *circuits.ee.IEC* mit *circuitikz* nicht braucht, sollte man darauf verzichten. Bei den folgenden Beispielen werden besagte Kurzbezeichnungen jedoch nicht verwendet. Unter anderem sind folgende Schaltkreissymbole bereits definiert:



Weitere Informationen zu CircuiTikZ gibt es in der [Beschreibung des Pakets](#).

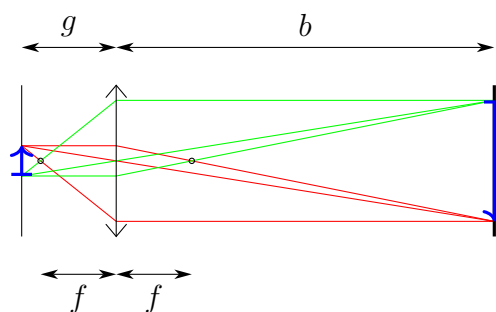
Optik unterstützt das Paket *optics* von Michel Fruchart. Verschiedene, einfach gezeichnete graphische Objekte dienen als Symbole für optische Bauelemente, die als besondere *tikz*-Knoten (nodes) in einer *tikzpicture*-Umgebung, zusammen mit

anderen Teilen von TikZ, eingesetzt werden. In der Präambel lädt man TikZ und die notwendigen Programmbibliotheken mit

```
\usepackage{tikz}
\usetikzlibrary{shapes,shapes.misc,shapes.geometric}
\usetikzlibrary{optics}
```

Im folgenden [Beispiel](#) wird die Abbildung eines Gegenstands auf ein vergrößertes Bild durch eine Sammellinse dargestellt. Ist $f = \ell$ die Brennweite, wobei ℓ eine beliebige Länge ist, beträgt die Gegenstandsweite $g = 1,25 \ell$ und die Bildweite $b = 5 \ell$. Rot beziehungsweise grün sind die jeweils drei Hauptstrahlen des Lichts gezeichnet, das vom oberen beziehungsweise unteren Ende des Gegenstands ausgeht.

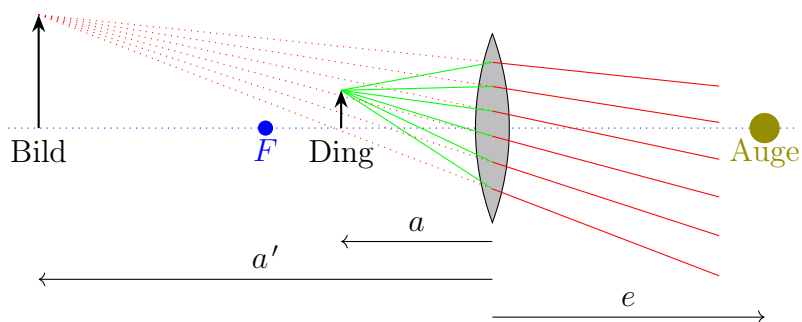
```
\begin{tikzpicture}[use optics]
\draw[red] (-1.25,0.2)--(5,-0.8) (-1.25,0.2)--(0,-0.8) -- (5,-0.8);
\draw[red] (-1.25,0.2)--(0,0.2)--(5,-0.8);
\draw[green] (-1.25,-0.2)--(5,0.8) (-1.25,-0.2)--(0,0.8)--(5,0.8);
\draw[green] (-1.25,-0.2)--(0,-0.2)--(5,0.8);
\node[thin optics element] (G) at (-1.25,0) {};
\node[lens,focal length=1cm,draw focal points={circle}]
(L1) at (0,0) {};
\node[screen] (S) at (5,0) {};
\draw[|->,blue,very thick] (-1.25,-0.2) -- (-1.25,0.2);
\draw[|->,blue,very thick] (5,0.8) -- (5,-0.8);
\coordinate (ao1) at (0,1.5cm);
\draw[>=technical,<->] (ao1 -| G) -- (ao1 -| L1)
node[midway,above] {$g$};
\draw[>=technical,<->] (ao1 -| L1) -- (ao1 -| S)
node[midway,above] {$b$};
\coordinate (ao2) at (0,-1.5cm);
\coordinate (f1) at (-1cm,0);
\coordinate (fr) at (1cm,0);
\draw[>=technical,<->] (ao2 -| f1) -- (ao2 -| L1)
node[midway,below] {$f$};
\draw[>=technical,<->] (ao2 -| L1) -- (ao2 -| fr)
node[midway,below] {$f$};
\end{tikzpicture}
```



Optische Bauelemente des Pakets *optics* sind Sammellinse, Zerstreuungslinse, Spalt, Doppelspalt, Spiegel (plan, konvex, konkav), Polarisator, Strahlteiler, Geradsichtprisma, allgemeines dünnes oder dickes optisches Element, Wärmeschutzfilter, Bildschirm, optisches Gitter, Raster, halbdurchlässiger Spiegel, Blende, Lampen und Sensoren, Laser.

Optische Zeichnungen können mit Tikz auch ohne das Paket *optics* erstellt werden. Oben (Seite 70) wurden optische Linsen abgebildet. Im folgenden [Beispiel](#) wird der Verlauf von Lichtstrahlen bei Verwendung einer sphärischen Sammellinse als Lupe dargestellt (maßstabsgetreu für $n_{\text{Glas}} = 1,6$, nicht das Auge). Die zum virtuellen Bild fortgesetzten Strahlen sind gepunktet.

```
\begin{tikzpicture}
% opt. Achse, Dingbrennpunkt (x-Koordinate = -3)
\draw[blue,dotted] (-6.4,0)--(4,0); % optische Achse
\fill[blue] (-3,0) circle (0.1) node[below] {$F$}; % F
% symmetrische (L=R) bikonvexe Linse, Brennweite 3
\pgfmathsetmacro{\WL}{asin(2.5/(2*3.6))} % d=2.5, r=3.6
\draw [fill=lightgray] (0,2.5/2) % oberster Linsenpunkt
arc[start angle=180-\WL,delta angle=2*\WL,radius=3.6]
arc[start angle=-\WL,delta angle=2*\WL,radius=3.6];
% Lichtstrahlen
\foreach \WI in {-7,...,-2} % 6 Strahlen von -7*3 bis -2*3 Grad
{ \pgfmathsetmacro{\DY}{tan(\WI*3)} % Bild: x = -6, y = 1.5
\draw[red,dotted] (-6,1.5)--(0,1.5+6*\DY); % vom Bild zur Linse
\draw[red] (0,1.5+6*\DY)--(3,1.5+9*\DY); % Linse zum Auge
\draw[green] (-2,0.5)--(0,1.5+6*\DY); } % vom Ding zur Linse
% Ding (x-Koordinate = -2, y = 0.5) und Bild (x = -6, y = 1.5)
\draw[Stealth-,black,thick] (-2,0.5)--(-2,0) node[below] {Ding};
\draw[Stealth-,black,thick] (-6,1.5)--(-6,0) node[below] {Bild};
% Auge (x-Koordinate = 3.6, Radius = 0.2)
\fill[olive] (3.6,0) circle (0.2) node[below] {Auge}; % Augenkreis
% Streckenpfeile (Dingweite a, Bildweite a', Augenabstand e)
\draw[<-] (-2,-1.5)--(0,-1.5) node[midway,above] {$a$};
\draw[<-] (-6,-2)--(0,-2) node[midway,above] {$a'$};
\draw[->] (0,-2.5)--(3.6,-2.5) node[midway,above] {$e$};
\end{tikzpicture}
```



6.4 Randnotizen, Kästen, Kalender und Avatare

Randnotizen können in einem unfertigen Manuskript Stellen markieren, die noch bearbeitet werden müssen. Mit dem, auf Tikz aufbauendem Paket [todonotes](#) von Henrik Skov Midtiby kann man, in einem Kasten mit farbigem Hintergrund, Randnotizen setzen und man kann auch auf fehlende Bilder hinweisen.

Gallia est omnis divisa in partes tres,
`\todo[color=green!30]{Belgae?}\`aliam Aquitani,
 tertiam qui ipsorum lingua Celtae, nostra Galli appellantur.

Gallia est omnis divisa in partes tres, _____
 aliam Aquitani, tertiam qui ipsorum lingua Celtae, nostra Galli appellantur.

Belgae?

Die im optionalen Argument festgelegte Hintergrundfarbe sollte, wie im Beispiel, transparent sein, damit der Vordergrundtext gut lesbar ist. In Fußnoten und einigen Umgebungen (zum Beispiel minipage) funktionieren die *todonotes* leider nicht.

Gerahmte Kästen in verschiedenen Formen und Farben kann man mit TikZ um Text oder mathematischen Formeln setzen. Mit der Zeile

```
\usetikzlibrary{shapes.symbols}
```

lädt man zunächst eine geeignete Programmbibliothek in der Präambel (nach dem Laden von *tikz*) und kann danach die bereitgestellten Formen nutzen. Es folgen einige [Beispiele](#).

```
\begin{tikzpicture}
\node [cloud,draw,cloud %
ignore aspect]{\Huge Wetter};
\end{tikzpicture}
```



```
\begin{tikzpicture}
\node [starburst,starburst %
points=28, starburst point %
height=5mm,draw,thick,red,%
fill=red!10,inner sep=3mm] {%
\Large \textbf{Revolution}};
\end{tikzpicture}
```



```
\begin{tikzpicture}
\node [cloud,cloud puffs=18.8,cloud ignore aspect,minimum width%
=8cm,minimum height=3cm,align=center,draw,blue,fill=yellow!10]{%
\begin{minipage}{0.65\textwidth}
\begin{large}
\textsc{Additionstheoreme für Sinus und Cosinus} \vspace{-2mm}
\begin{align}
\sin(x_1+x_2) &= \sin x_1 \cos x_2 + \sin x_2 \cos x_1 \\
\cos(x_1+x_2) &= \cos x_1 \cos x_2 - \sin x_1 \sin x_2
\end{align}
\end{align}\vspace{-6mm}
\end{large}
\end{minipage}
};
\end{tikzpicture}
```


ADDITIONSTHEOREME FÜR SINUS UND COSINUS

$$\sin(x_1 + x_2) = \sin x_1 \cos x_2 + \sin x_2 \cos x_1 \quad (6.1)$$

$$\cos(x_1 + x_2) = \cos x_1 \cos x_2 - \sin x_1 \sin x_2 \quad (6.2)$$

Gibt man für die Anzahl der Puffen (Auswölbungen der Wolke) einen Wert an, der nicht ganzzahlig ist (hier: *cloud puffs*=18.8), wird die Wolke unsymmetrisch. Ohne die Option *cloud ignores aspect* ergäbe sich eine rundliche Wolke (Höhe = Breite).

```
\begin{tikzpicture}
\node [rectangle, rounded corners,draw=blue,very thick,%
inner sep=3mm] (kasten){%
\begin{minipage}{0.925\textwidth} \rule{0pt}{7mm}%
He had had a nice, good, idle time all the while---plenty of
company---and the fence had three coats of whitewash on it!
If he hadn't run out of whitewash he would have bankrupted
every boy in the village.
\end{minipage}
};
\node[right=8mm, rectangle,rounded corners,fill=yellow!20,%
draw=blue,very thick,text=blue,inner xsep=3mm,%
inner ysep=1.8mm] at (kasten.north west)%
{\vphantom{\Large W}\textbf{Efficient Delegation}};
\end{tikzpicture}
```

Efficient Delegation

He had had a nice, good, idle time all the while—plenty of company—and the fence had three coats of whitewash on it! If he hadn't run out of whitewash he would have bankrupted every boy in the village.

Die Einfügung von `\vphantom{\Large W}` in die Zeichenkette der Überschrift bewirkt hier einen größeren Abstand von Überschrift und darüber liegender Linie, sodass der Überschriftenkasten ausgewogener erscheint.

Das leider fehlerhafte Paket *pgfornament* von Alain Matthes enthält allerlei Verzierungen, mit denen man zum [Beispiel](#) schöne Trennlinien bekommt. Gewarnt sei vor merkwürdigen Fehlern, die bei manchen PDF-Betrachterprogrammen auftreten können, wenn *pgfornament* benutzt wird und bestimmte andere Pakete ebenfalls geladen wurden.

Kästen mit Logo können mit dem Paket *bclogo* von Patrick Fradin und Maxime Chupin erzeugt werden, das mit

```
\usepackage[tikz]{bclogo}
```

geladen wird und die *bclogo*-Umgebung für einen schönen Textkasten bereitstellt.

```
\begin{bclogo}[logo=\bcplume,
noborder=true, couleurBarre=red]
{~Cicero: In Catilinam}
O tempora, o mores! Senatus haec
intellegit, consul videt; hic tamen
vivit. Vivit? Immo vero etiam in
senatum venit! \end{bclogo}
```
















Cicero: In Catilinam

O tempora, o mores! Senatus
haec intellegit, consul videt;
hic tamen vivit. Vivit? Immo
vero etiam in senatum venit!

Optionale Argumente der *bclogo*-Umgebung sind unter anderem *logo=...* mit einem Logo (siehe unten) als Wert, *noborder=...* mit *true* (ohne Rahmen) oder *false* (mit Rahmen) als Wert, und *couleurBarre=...* mit einem Farbwert für den linken Balken. Mit *barre=none* statt *couleurBarre=...* wird der Balken weggelassen.

Unter anderem sind folgende Logos verfügbar:

					
<code>\bcplume</code>	<code>\bchorloge</code>	<code>\bccalendrier</code>	<code>\bcpoisson</code>	<code>\bcfleur</code>	<code>\bcbombe</code>
					
<code>\bcdallemagne</code>	<code>\bcdfrance</code>	<code>\bcditalie</code>	<code>\bcinfo</code>	<code>\bcbook</code>	<code>\bcquestion</code>

Die Logos können auch ohne *bclogo*-Umgebung in normalen Text eingefügt werden, sind dort aber nicht wie Zeichen skalierbar. Zum Beispiel gibt `\bclampe` .

Wenn man über die Platonischen Körper schreibt, sind folgende Logos interessant:

				
<code>\bccube</code>	<code>\bcdodecaedree</code>	<code>\bcicosaedre</code>	<code>\bcoctaedre</code>	<code>\bctetraedre</code>

Kalender können mit der TikZ-Programmbibliothek *calendar* gezeichnet werden. Um deutsche Namen für Monate und Wochentage zu erhalten, lädt man in der Präambel vor `\usetikzlibrary{calendar}` das Paket *translator* mit der Sprachoption, die auch als Dokumentklassenoption angegeben wurde (in der Regel *ngerman*):

```
\usepackage[ngerman]{translator}
\usepackage{tikz}
\usetikzlibrary{calendar}
```

Dann lässt sich im Dokument ein Kalender wie in folgendem [Beispiel](#) erstellen.

```

\begin{tikzpicture}
[every day/.style={anchor=center}]
\calendar[
name=ahgcal, week list, day text={\%d=},
dates=\year-\month-01 to \year-\month-last,
month label above centered,
month text={\textit{\%mt \%y0}} ]
if (Sunday) [red!70!black]
if (equals=\year-\month-\day)
{\draw (0,0) circle (8pt);} ;
\end{tikzpicture}

```

August 2018

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25
26	27	28	29	30
31				

Avatare sind graphische Figuren, die Menschen, Tiere oder Fabelwesen darstellen und Menschen im Internet symbolhaft vertreten. Derartige Graphiken sind vielleicht nicht notwendig, aber lustig. Hierfür gibt es unter anderem das Paket *tikzpeople* von Nils Fleischhacker. Die Figuren werden als Knoten in einer *tikzpicture*-Umgebung gezeichnet. Der Typ wird als optionales Argument angegeben. Definiert sind:

alice, bob, bride, builder, businessman, charlie, chef, conductor, cowboy, criminal, dave, devil, duck, graduate, groom, guard, jester, judge, maninblack, mexican, nun, nurse, person, physician, pilot, police, priest, sailor, santa, surgeon.

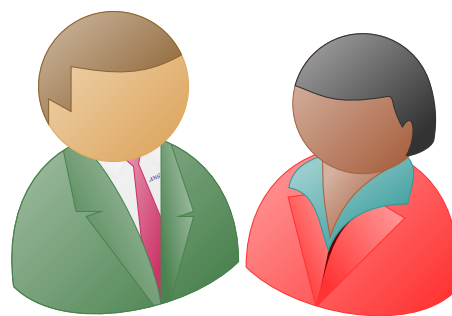
Die Figuren können durch Optionen ergänzt oder verändert werden, nämlich: evil (böse, zum Beispiel mit Kinnbart und Hörnern), female (längere Haare), good (mit Heiligenschein), mirrored (guckt von rechts nach links, statt von links nach rechts), monitor (mit Computerbildschirm), saturated (mit gesättigten Farben), shield (mit Schild), sword (mit Schwert). Bestimmten Figuren kann jeweils nur eine Teilmenge aller Optionen zugeordnet werden. Gesichtszüge (Mund, Nase Augen) fehlen den Figuren. Verschiedene Teile können anders gefärbt werden (Beispiel: hair=brown). Die Größe der Figuren kann mit der Option *minimum size=...* verändert werden. Als Pflichtargument muss für die Figur eine Beschriftung angegeben werden, zum Beispiel: {Romeo}, die aber auch leer sein kann: {}.

Nun endlich zwei **Beispiele**:

```

\begin{tikzpicture}
\node[businessman, tie=purple,
monogramtext=AHG, % auf dem Hemd
minimum size=3cm] at (0,0.16) {Romeo};
\node[alice, mirrored, shirt=red,
undershirt=teal, minimum %
size=2.8cm] at (3,0) {Julia};
\end{tikzpicture}

```



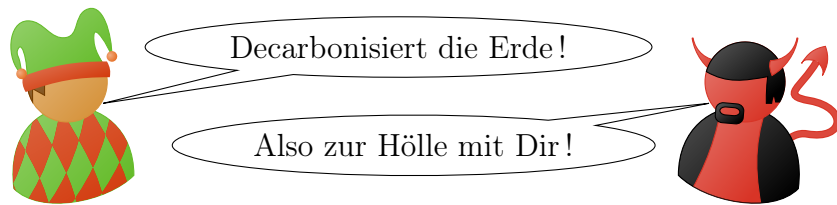
Romeo

Julia

```

\begin{center} \begin{tikzpicture}
\node[jester, saturated, minimum size=1.6cm] (a) at (0,0) {};
\node[ellipse callout, draw, yshift=1cm, xshift=4.3cm,
callout absolute pointer={(a.mouth)}] {Decarbonisiert die Erde\,!};
\node[devil, mirrored, saturated, minimum size=1.6cm] (b) at (8.8,0) {};
\node[ellipse callout, draw, yshift=-0.3cm, xshift=4.5cm,
callout absolute pointer={(b.mouth)}] {Also zur Hölle mit Dir\,!};
\end{tikzpicture} \end{center} % in Präambel: \usetikzlibrary{shapes}

```



Schneemänner können ganzjährig mit dem Paket `scsnowman` von Hironobu Yamashita in mehreren Varianten gezeichnet werden. Zum [Beispiel](#):

Schneemann

```
\scsnowman
```

```
[scale=6, adjustbaseline,  
mouthshape=frown,sweat=blue]
```

Schneemann



Optionale Argumente sind *body*, *sweat*, *hat*, *arms*, *muffler*, *buttons* und *snow*. Man kann ihnen eine Farbe zuweisen. Die Option *mouthshape* hat die möglichen Werte *smile*, *tight* und *frown*. Die Option *scale* mit einem Zahlenwert bestimmt die Größe und *adjustbaseline* setzt den Schneemann auf die Grundlinie der Textzeile.

Gummienten sind ja wohl das Letzte, was hier erwähnt werden soll. Das Paket `tikzducks` von samcarter malt bunte Entchen. Hier ein einfaches [Beispiel](#):

```
\begin{tikzpicture}  
\duck[water=blue]  
\end{tikzpicture}
```



Wir fragen unsere chinesischen Freunde: Wann gibt es ein Paket für Pandabären?







7 Ergänzungen

7.1 Physik, Chemie, Bioinformatik, Übungen usw.

Physikalische Grundlagen können mit dem Paket *mandi* von Paul J. Heafner beschrieben werden. Es enthält besondere Befehle für physikalische und astronomische Gleichungen. Leider fehlt eine Anpassung an deutsche Schreibweisen und die Kompatibilität mit anderen Paketen ist nicht gut. Nützlich sind zum [Beispiel](#) die (mit Symbol und Wert) definierten physikalischen und astronomischen Konstanten.

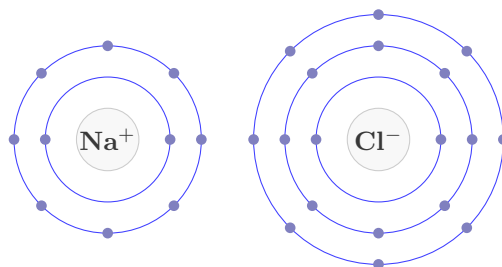
Pakete für physikalische Größen (Seite 53), Optik (Seite 77) und elektrische Schaltkreise (Seite 75) wurden bereits vorgestellt (siehe oben).

Phantasievolle chemische Symbole werden vom Paket *svrsymbols* (von Apostolos Syropoulos) als Sonderzeichen in einer *math*-Umgebung zur Verfügung gestellt. Hier einige [Beispiele](#):

Atom <code>~~ \backslashatom$\\$</code>	Atom 
Wassermolekül <code>~~ \backslashwater$\\$</code>	Wassermolekül 
ionische Bindung <code>~~ \backslashionicbond$\\$</code>	ionische Bindung 
metallische Bindung <code>~~ \backslashmetalbond$\\$</code>	metallische Bindung 
kovalente Einfachbindung <code>~~ \backslashcovbond$\\$</code>	kovalente Einfachbindung 
Wasserstoffbrückenbindung <code>~~ \backslashhbond$\\$</code>	Wasserstoffbrückenbindung 

Atommodelle mit elektronenbesetzten Schalen stellt das Paket *bohr* von Clemens Niederberger dar. [Beispiel](#):

```
\setbohr{ name-options-set
={font=\footnotesize},
electron-radius = {2pt} }
\bohr{10}{ $\mathbf{Na^+}$ }
\quad % Bohr-Schalenmodell
\bohr{18}{ $\mathbf{Cl^-}$ }
```



Der erste Befehl in obigem Beispiel setzt die Schriftgröße für die Kernbezeichnung und den Radius der Elektronen fest. Der Befehl `\bohr{...}{...}` erhält im ersten Pflichtargument die Zahl der Elektronen, im zweiten den Text, welcher den Kern bezeichnet.

Das Paket *elements*, welches vom Paket *bohr* automatisch geladen wird, stellt die Elektronenkonfiguration der Atome bereit und erlaubt auch, andere Elektronenkonfigurationen im gleichen Format zu schreiben. [Beispiel](#):

```
\elconf{Cl} \par % e-Konfig. Chlor 1s22s22p63s23p5
\writeelconf{2,2+6,2+6} % Chlorid 1s22s22p63s23p6
```

In obigem Beispiel werden die Elektronenkonfigurationen von Chlor (dem Programm bekannt) und Chlorid (als Argument des Befehls `\writeelconf{...}`) ausgegeben.

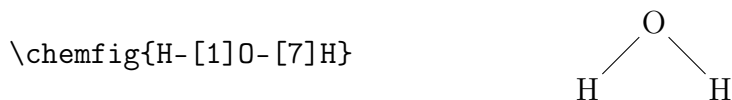
Summenformeln und chemische Gleichungen können mit dem Paket *mhchem* von Martin Hensel dargestellt werden. Es benötigt einige andere Pakete, die automatisch geladen werden. Umgekehrt wird *mhchem* automatisch vom Paket *chemexec* geladen. Mit *chemexec* sollte man daher auf das Laden von *mhchem* verzichten. Da die neuen Versionen von *mhchem* nicht kompatibel zu älteren sind, sollte man beim Laden eine Versionsnummer als Option angeben:

```
\usepackage[version=4]{mhchem}
```

Der Befehl `\ce{...}` des Pakets *mhchem* erlaubt es, einzelne Summenformeln oder ganze Gleichungen zu schreiben, wie folgende [Beispiele](#) zeigen.

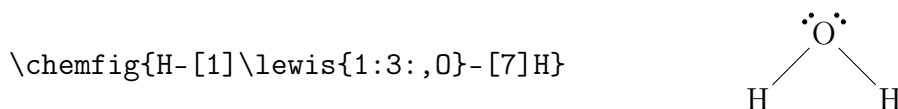
<code>\ce{H3O+} \ce{NaCl(aq)} \ce{^{6}_{13}C}</code>	$\text{H}_3\text{O}^+ \text{NaCl(aq)} \text{}^{6}_{13}\text{C}$
<code>\ce{Ba^2+ + SO4^2- -> BaSO4 v }</code>	$\text{Ba}^{2+} + \text{SO}_4^{2-} \longrightarrow \text{BaSO}_4 \downarrow$
<code>\ce{H2O -> H2 ^ + 1/2 O2 ^}</code>	$\text{H}_2\text{O} \longrightarrow \text{H}_2 \uparrow + \frac{1}{2} \text{O}_2 \uparrow$
<code>\ce{O=O} \quad \ce{H-C#N}</code>	$\text{O}=\text{O} \quad \text{H}-\text{C}\equiv\text{N}$
<code>\ce{H2C=CH2 ->[\mathrm{H_2}] H3C-CH3}</code>	$\text{H}_2\text{C}=\text{CH}_2 \xrightarrow{\text{H}_2} \text{H}_3\text{C}-\text{CH}_3$
<code>\ce{A <--> B} \quad \ce{A <=> B}</code>	$\text{A} \rightleftharpoons \text{B} \quad \text{A} \rightleftharpoons \text{B}$
<code>\ce{A <=>> B} \quad \ce{A <=> B}</code>	$\text{A} \rightleftharpoons \text{B} \quad \text{A} \rightleftharpoons \text{B}$

Chemische Strukturformeln können mit dem Paket *chemfig* von Christian Tellechea gezeichnet werden. [Beispiele](#):



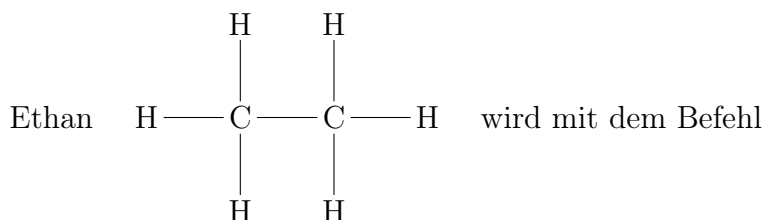
Die Molekülbeschreibung besteht aus den Buchstaben für die Atome (oder Atomgruppen) und den Bindungen (- Einfach-, = Doppel-, ~ Dreifachbindung). Für Bindungen kann in eckigen Klammern ein Winkel angegeben werden. Am einfachsten ist die Verwendung vordefinierter Winkel (0: 0°, 1: 45°, 2: 90°, ..., 7: 315°).

Die Elektronenformel (Lewis-Struktur) erhält man, wenn statt des Atomsymbols der Befehl `\lewis{Winkel Elektronen Winkel Elektronen,...,Atomsymbol}` steht.



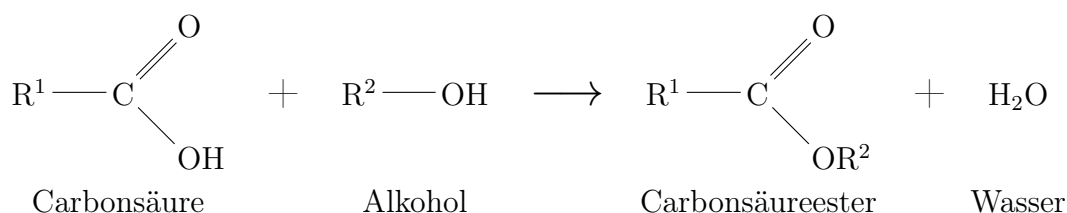
Elektronen können durch einen Punkt (.) oder Doppelpunkt (:) bezeichnet werden. Wird keine Angabe zu den Elektronen (weder . noch :) gemacht, erscheint ein Strich (für ein Elektronenpaar). Es werden vordefinierte Winkel wie für Bindungen angegeben (siehe oben).

Verzweigungen werden durch Ausdrücke in runden Klammern angegeben.



`\chemfig{H-C(-[2]H)(-[6]H)-C(-[2]H)(-[6]H)-H}`

gezeichnet. Die Reaktionsgleichung



erhält man mit den Befehlen

```

\chemname{\chemfig{R^{1}-C(-[7]OH)=[1]O}}{Carbonsäure}
\quad \begin{Large}+\end{Large} \quad
\chemname{\chemfig{R^{2}-OH}}{Alkohol}
\quad \begin{Large}$\longrightarrow$\end{Large} \quad
\chemname{\chemfig{R^{1}-C(-[7]OR^{2})=[1]O}}{Carbonsäureester}
\quad \begin{Large}+\end{Large} \quad
\chemname{\chemfig{H_2O}}{Wasser}

```

Der Befehl `\chemname{Molekül}{Name}` setzt den gewählten Namen unter das Molekül. Hoch- oder Tiefstellung einer Zeichenkette *zk* nach einem Atomsymbol erreicht man, wie in einer *math*-Umgebung, mit `^{\{zk\}}` beziehungsweise `_{\{zk\}}`. In obigem Beispiel wurde die Nummerierung der Organylgruppen, R^1 und R^2 , hoch gesetzt. So kann man auch Ladungssymbole (+ und -) hochsetzen.

Chemische Übungsblätter können leichter mit dem Paket [chemexec](#) von Clemens Niederberger geschrieben werden. Wenn der Befehl `\aufgabe{\dots}` (siehe unten) zur Verfügung stehen soll, muss das Paket mit

```
\usepackage[exercise]{chemexec}
```

geladen werden. Sollen englische statt deutsche Bezeichnungen geschrieben werden, ist beim Laden zusätzlich die Option *english* anzugeben.

```
\usepackage[exercise,english]{chemexec}
```

Die im Paket definierten Umgebungen und einige Befehle sind auch gut für Übungen in anderen Fachgebieten geeignet. Das Paket kann jedoch nicht in der Dokumentklasse *beamer* verwendet werden. Zunächst einige nützliche Befehle, die chemische Sonderzeichen ergeben:

Ladungen <code>\om{}</code> und <code>\op</code> , <code>Ca\op[2]</code>	Ladungen \ominus und \oplus , $\text{Ca}^{2\oplus}$
<code>\el{}</code> <code>\Hpl{}</code> und Hydroxid <code>\Hyd{}</code>	e^\ominus H^\oplus und Hydroxid OH^\ominus
Oxidationszahlen <code>\ox{+1}{K}</code> <code>\ox{-1}{Cl}</code>	Oxidationszahlen $\overset{+1}{\text{K}}\overset{-1}{\text{Cl}}$
Säurereste <code>\carbonat{}</code> <code>\nitrat{}</code> <code>\nitrit{}</code>	Säurereste CO_3 NO_3 NO_2
<code>\phosphat{}</code> <code>\phosphit{}</code> <code>\sulfat{}</code> <code>\sulfit{}</code>	PO_4 PO_3 SO_4 SO_3

Wenn die Befehle `\om`, `\op` oder `\ox` in einer Umgebung oder einem Befehl des Pakets *mhchem* benutzt werden, muss vor ihnen ein Leerzeichen sein.

Die *beispiel*-Umgebung erzeugt einen Rahmen, in dem ein nummeriertes Beispiel gesetzt wird.


```
\begin{beispiel}
Wenn dir die Hose reißt und eine Dame
sagt, sie wisse deine Offenheit zu
schätzen, dann ist das Ironie.
\end{beispiel}
```

Beispiel 1:

Wenn dir die Hose reißt und eine Dame sagt, sie wisse deine Offenheit zu schätzen, dann ist das Ironie.

Sollen zwei Beispiele im gleichen Rahmen gezeigt werden, trennt man sie mit dem Befehl `\bsp`. Die `definition`-Umgebung erzeugt einen Kasten, in dem nach der Überschrift *Definition* eine Definition steht.

```
\begin{definition}
Ironie ist ein sprachlicher
Ausdruck, aus dem eine andere
Aussage abgeleitet werden kann
als das wörtlich Gesagte.
\end{definition}
```

DEFINITION Ironie ist ein sprachlicher Ausdruck, aus dem eine andere Aussage abgeleitet werden kann als das wörtlich Gesagte.

Die `exkurs`-Umgebung mit einem Pflichtargument `{...}` erzeugt einen Rahmen aus zwei farbigen Balken (oben und unten), in dem die Überschrift *EXKURS: ...* (wobei ... der im Pflichtargument angegebene Titel ist) und der Exkurstext stehen. Die Exkurse werden in das Inhaltsverzeichnis aufgenommen.

```
\begin{exkurs}{Wetter} % Exkurs übers Wetter (Zitat von Mark Twain)
Alle schimpfen aufs Wetter, aber keiner tut was dagegen. \end{exkurs}
```

Der Befehl `\aufgabe{}` mit leerem Pflichtargument gibt eine nummerierte Überschrift *Aufgabe ...* aus. Wird im Pflichtargument eine Zeichenkette gegeben, wird diese statt des Wortes *Aufgabe* in der Überschrift wiedergegeben.

```
\aufgabe{}
Wie lange dauerte der
Siebenjährige Krieg?%
\aufgabe{Siebengescheites}
Was ergibt sieben
mal sieben?
```

1. Aufgabe

Wie lange dauerte der Siebenjährige Krieg?

2. Siebengescheites

Was ergibt sieben mal sieben?

Der Befehl `\loesung{...}` definiert im Pflichtargument die Lösung zur vorhergehenden Aufgabe. Man kann auch ein optionales Argument mit dem Aufgabentitel (oder einer anderen Zeichenkette) angeben. Der Befehl `\makeloesung`, der nur einmal aufgerufen werden kann, gibt alle Lösungen aus.

```
\aufgabe{Erster Kreuzzug}
Wann lebte
Gottfried von Bouillon?
\loesung[Erster Kreuzzug]
{Als er nichts anderes
zu essen hatte.}
\makeloesung
```

3. Erster Kreuzzug

Wann lebte Gottfried von Bouillon?

3. Erster Kreuzzug

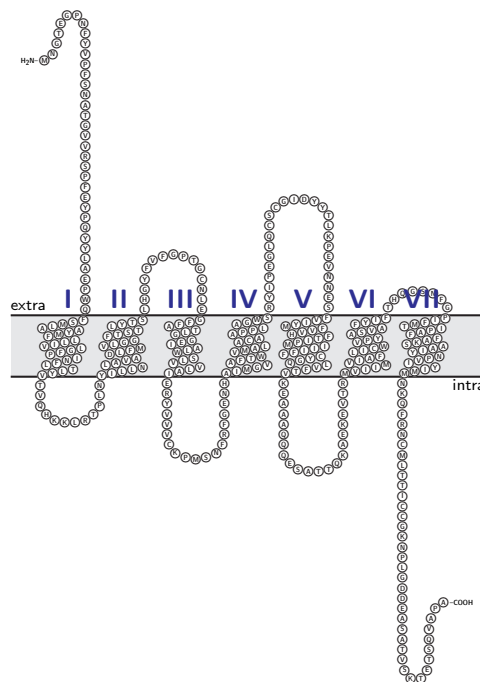
Als er nichts anderes zu essen hatte.

Wird beim `loesung`-Befehl ein optionales Argument angegeben, erhält die Lösung dieses als Überschrift, ansonsten ist das Wort *Lösung* die Überschrift.

Ein Alternative zu `chemexec` ist das Paket `exercise`, siehe unten.

Membranproteine können mit dem Paket *textopo* von Eric Beitz dargestellt werden. Im folgenden *Beispiel* werden die sieben Transmembrandomänen des menschlichen Proteins *Rhodopsin* abgebildet.

```
\begin{textopo}
\membranecolors{Black}{Gray10} % Farben
\scaletopo{1} % Verkleinerung des Plots
\sequence{ MNGTEGPNFYVPFSNATGVVRSPFEYPQYYLAEPWQ
  [ FSLAAYMFLLIVLGFPINFLTYV ]      % 1. TMD
TVQHKKLRTPLN
  [ YILLNLAVADLFMVLGGFTSTLYTS ]    % 2. TMD
LHGYFVFGPTGCNLE
  [ GFFATLGGEIALWSLVVLAI ]          % 3. TMD
ERYVVVCKPMSNFRFGENH
  [ AIMGVAFTWVMALACAAPPLAGWS ]      % 4. TMD
RYIPEGLQCSCGIDYYTLKPEVNNES
  [ FVIYMFVVHFTIPMIIFFCYGQLVFTV ]  % 5. TMD
KEAAAQQQESATTQKAEKEVTR
  [ MVIIMVIAFLICWVPYASVAFYIF ]      % 6. TMD
THQGSNFG
  [ PIFMTIPAFFAKSAAIYNPVIYIMM ]     % 7. TMD
NKQFRNCMLTTICCGKNPLGDDEASATVSKTETSQVAPA }
\Nterm{extra} % N-Terminus extrazellulär
\end{textopo}
```



Mit `\membranecolors{...}{...}` werden die Farben der Membran (Rand und Inneres) gewählt und mit dem Befehl `\scaletopo{1}` wird die Graphik verkleinert. Der Befehl `\sequence{...}` enthält die Aminosäuresequenz im einbuchstabigen Code, wobei

Transmembrandomänen in eckige Klammern gefasst sind. Der Befehl `\Nterm{extra}` bestimmt, dass das Amino-Ende extrazellulär liegt.

Bioinformatische Sequenzalignments sind Vergleiche von Zeichenketten (informatische Sequenzen), bei denen ähnliche Teilsequenzen sichtbar werden. In der Bioinformatik werden biologische Sequenzen durch Zeichenketten dargestellt, $A = a_1a_2a_3 \dots a_m$, $B = b_1b_2b_3 \dots b_n$, wobei die Zeichen Elemente eines Alphabets Σ sind. Beim Alignment werden die Zeichenketten zeilenweise, unter Hinzufügung von Leerzeichen, mit konstanter Zeichenbreite geschrieben, so dass die Bereiche größter Ähnlichkeit direkt untereinander stehen. Das Paket *gotosh* von Takuto Asakura führt ein paarweises Alignment mit dem Gotoh-Algorithmus durch, welcher die Methode der dynamischen Programmierung anwendet.

In folgendem [Beispiel](#) sind zwei Nukleotidsequenzen $A = \text{ATCGGCGCACGGGGGA}$ und $B = \text{TTCCGCCACACA}$ über dem Alphabet $\Sigma = \{A, C, G, T\}$ gegeben. Das Alignment erhält man nun einfach durch

```
\Gotoh{CCGTATTCACTAC}
{AGTATCAATACCGGGGCGAC}
\texttt{\GotohResultA}\
\texttt{\GotohResultB}%
\[\3mm] Score \GotohScore
```

CCGTATTCA CTAC
. AGTATCAATACCGGGGCGAC
Score -17

Der *Gotoh*-Befehl hat als Pflichtargumente die beiden Zeichenketten. Er speichert das Ergebnis des Alignments in den Befehlen `\GotohResultA` und `\GotohResultB`. Diese können dann mit normalen L^AT_EX-Befehlen untereinander geschrieben werden. Es ist sinnvoll, dafür eine nichtproportionale (dicktengleiche) Schrift zu benutzen, hier mit dem *texttt*-Befehl. Die Kosten des Alignments (Score) finden sich im Befehl `\GotohScore`. Den Parametern des Gotoh-Algorithmus (*match*, *mismatch*, *d* und *e*) können vor dem *Gotoh*-Befehl mit dem *GotohConfig*-Befehl Werte zugewiesen werden. Darauf wurde in obigem Beispiel verzichtet, so dass die Standardwerte eingesetzt sind.

Eine graphische Darstellung bioinformatischer Sequenzalignments ist mit dem Paket *texshade* von Eric Beitz möglich. Werden die beiden Pakete *texshade* und *textopo* verwendet, lädt man zuerst *texshade*, danach *textopo*. Ein multiples Sequenzalignment (Alignment mehrerer Sequenzen) wird mit Programmen wie [CLUSTALW](#) erzeugt. Das Ergebnis kann in einer Datei gespeichert sein. Das Paket *texshade* kann Alignmentdateien lesen, die im MSF-format oder im ALN-format geschrieben wurden. In folgenden [Beispielen](#) gehen wir von einer Datei *ins_A.aln* aus, welche sich im gleichen Verzeichnis befindet wie die *tex*-Datei. Diese Alignmentdatei enthält die Aminosäuresequenz der A-Kette des Insulins von fünf verschiedenen Säugerarten.

```

\begin{texshade}
{ins_A.aln}
\seqtype{P}
\shadingmode
{identical}
\threshold
[100]{50}
\showlegend
\hidenumbering
%\hideconsensus
\end{texshade}

```

Mensch	GIVEQCC	TSI	CS	LY	Q	LEN	NYCN
Schwein	GIVEQCC	TSI	CS	LY	Q	LEN	NYCN
Rind	GIVEQCC	ASV	CS	LY	Q	LEN	NYCN
Alligator	GIVEQCC	HNT	CS	LY	Q	LEN	NYCN
Neunauge	GIVEQCC	HRK	CS	IY	DM	EN	NYCN
Konsensus	!!!!!!!	*	!!*!!	*!!*!!	!!!!!!		

X nicht konserviert
X $\geq 50\%$ konserviert
X alle identisch

Die vom Paket bereit gestellte *texshade*-Umgebung erhält als Pflichtargument den Namen oder Pfad zur Alignmentdatei. Der Befehl `\seqtype{P}` gibt an, dass es sich um Aminosäuresequenzen handelt (Alternative: *N* für Nukleotidsequenzen). Der Befehl `\shadingmode{identical}` bestimmt, dass spaltenweise die Anzahl der Übereinstimmungen von Zeichen gekennzeichnet wird. Die Hintergrundfarben der Zeichen hängen von der prozentualen Übereinstimmung der Zeichen ab und werden mit dem Befehl `\threshold[100]{50}` festgelegt. Die Legende dieser Festlegung wird mit `\showlegend` gezeigt und `hidenumbering` unterdrückt die Anzeige der Zeichenzahl pro Zeile. Mit dem Befehl `hideconsensus` (welcher hier durch % auskommentiert wurde) könnte man die Anzeige der Konsensuszeile abschalten.

```

\begin{texshade}
{ins_A.aln}
\seqtype{P}
\shadingmode
[1]{diverse}
\hidenumbering
\end{texshade}

```

Mensch	GIVEQCCTSI	CS	LY	Q	LEN	NYCN
Schwein					
Rinda.v					
Alligatorhnt					
Neunaugehrk..i.dm					

```

\begin{texshade}
{ins_A.aln}
\seqtype{P}
\shadingmode
[hydropathy]
{functional}
\shadeallresidues
\showlegend
\hidenumbering
\end{texshade}

```

Mensch	GIVEQCC	TSI	CS	LY	Q	LEN	NYCN
Schwein	GIVEQCC	TSI	CS	LY	Q	LEN	NYCN
Rind	GIVEQCC	ASV	CS	LY	Q	LEN	NYCN
Alligator	GIVEQCC	HNT	CS	LY	Q	LEN	NYCN
Neunauge	GIVEQCC	HRK	CS	IY	DM	EN	NYCN

X sauer (-)
X basisch (+)
X polar ungeladen
X hydrophob unpolar

Übungsblätter können mit verschiedenen Paketen erstellt werden, zum Beispiel mit *chemexec*, siehe oben. Einfach zu handhaben ist auch das Paket *exercise* von Paul Pichaureau. Es passt die Namen an die verwendete Sprache an und stellt je eine Umgebung für Übungen und Lösungen zu Verfügung. Wenn man es mit

```
\usepackage[lastexercise]{exercise}
```

lädt, schreibt man die Lösung direkt nach der Übung auf. Jede Übung wird mit einem *label* gekennzeichnet. Am Anfang der *document*-Umgebung, vor den Übungen, wird, anhand der *label*, ausgewählt, welche Übungen ausgedruckt werden sollen. Damit kann man aus einer Aufgabensammlung leicht ein besonderes Übungsblatt zusammenstellen. Die Auswahl erfolgt beispielsweise mit

```
\ExerciseSelect[label={4ma15,25qua}] % Übungen 4ma15 und 25qua
```

Wenn man den standardmäßigen Namen *Übung* durch *Aufgabe* ersetzen möchte, fügt man anschließend den Befehl

```
\renewcommand{\ExerciseName}{Aufgabe}
```

hinzu. Mit dem *ExerciseSelect*-Befehl werden nur die ausgewählten Übungen, nicht aber die Lösungen gedruckt. Will man dagegen alle Aufgaben und Lösungen ausgeben, lässt man den *ExerciseSelect*-Befehl weg (oder kommentiert ihn aus).

Danach folgen die Übungen und Lösungen. Als optionales Argument sollte man in jeder *Exercise*-Umgebung ein *label* haben. Außerdem kann man einen Titel angeben und die Schwierigkeit der Übung mit einer Anzahl Sternchen bezeichnen. Hier ein kurzes [Beispiel](#) ohne *ExerciseSelect*-Befehl:

<pre>\begin{Exercise} [label=4ma15, difficulty=1] Was ist $4 \cdot 5$? \end{Exercise} \begin{Answer} 20 \end{Answer}</pre>	<p style="text-align: center;">* Übung 1</p> <p>Was ist $4 \cdot 5$?</p> <p style="text-align: center;">Lösung zu Übung 1</p> <p>20</p>
<pre>\begin{Exercise}[label=25qua, %name=Rechenaufgabe, title={Quadrat}, difficulty=2] Was ergibt 25^2 ? \end{Exercise} \begin{Answer} 625 \end{Answer}</pre>	<p style="text-align: center;">** Übung 2 Quadrat</p> <p>Was ergibt 25^2 ?</p> <p style="text-align: center;">Lösung zu Übung 2</p> <p>625</p>

Hyperlinks können benutzt werden. Man muss jedoch das Paket *hyperref* vor dem Paket *exercise* laden.

Schriftliche Prüfungen kann man mit der von Philip Hirschhorn geschaffenen Dokumentklasse [exam](#) erstellen. Das folgende [Beispiel](#) erscheint hier ziemlich gedrängt. Die Anwendung ist jedoch einfach.

```
\documentclass[addpoints,a4paper,ngerman,12pt]{exam}
\usepackage[left=2cm,right=1.5cm]{geometry} % Seitenränder
\usepackage{babel} \usepackage[utf8]{inputenc}
```

```

\usepackage[T1]{fontenc}          \usepackage{sectsty}
% Deutsche Worte statt der vorgegebenen englischen
\pointpoints{Punkt}{Punkte}      \hpgword{Seite:}
\hpword{Punkte:}                  \hsword{Ergebnis:}
\hqword{Aufgabe:}                 \htword{Summe}
% Schriftgröße und Zeilenabstand für Abschnittsüberschriften
\sectionfont{\fontsize{12}{15}\selectfont} % benötigt sectsty
\newcommand{\vd}{\vspace{10mm}}    % Abstand nach einer Aufgabe
% Festlegung des allgemeinen Formats von Kopf- und Fußzeilen
\header{Klausur Umbratikulogie}{1. April 2017}{Prof. Barcala}
\headrule % horizontale Linie unter der Kopfzeile
\footer{}{}{\thepage\,/\,\numpages} % Fußzeile mit Seitenzahl
\begin{document}
\noprintanswers % ohne die Lösungen, mit Lösungen: \printanswers
\parbox{16cm}{ \makebox[7.8cm]{Name:\enspace\hrulefill} % Name
\makebox[8.1cm]{Vorname:\enspace\hrulefill} } % Vorname ins Feld
\begin{questions}
\section*{Künstliche Intelligenz}
\question[2] Was ist Intelligenz? % 2 Punkte für die Aufgabe
\begin{solution} Das, was der Test misst. \end{solution} \vd
\section*{Natürliche Dummheit}
\question[3] Wer erfand das Rad? % 3 Punkte für die Aufgabe
\begin{checkboxes} % Mehrfachauswahl-Aufgabe (Multiple Choice)
\choice Moses \choice Newton \choice Einstein \end{checkboxes}
\end{questions} \vspace{3mm}
\gradetable[h][questions] % Tabelle mit Punkten pro Aufgabe
\end{document} % pages statt questions: Punkte pro Seite

```

Vorteilhaft sind: a) die automatische Nummerierung der Aufgaben, b) eine automatisch aktualisierte Tabelle mit den Punktzahlen pro Aufgabe (oder pro Seite), und c) die Möglichkeit, Lösungen im Quellcode einzufügen und mit einem besonderen Befehl sichtbar zu machen. Um die Punktetabelle automatisch zu aktualisieren, muss drei Mal kompiliert werden. Man kann auch Aufgaben in Teilaufgaben zerlegen (hier nicht gezeigt).

7.2 Gedichte, Spiele, Strichcode und QR-Code

Gedichte können mit der Umgebung *verse* dargestellt werden. [Beispiel](#):

Es folgt ein Gedicht:

```

\begin{verse}
Good Guys aimed to please,\
Bad Guys used the club.

Life is live and cruel,\
Man, ' trusts man, is fool.

The Good Guys rest in peace,\
And Bad Guys rest in pub.
\end{verse}

```

Es folgt ein Gedicht:

```

Good Guys aimed to please,
Bad Guys used the club.

Life is live and cruel,
Man, ' trusts man, is fool.

The Good Guys rest in peace,
And Bad Guys rest in pub.

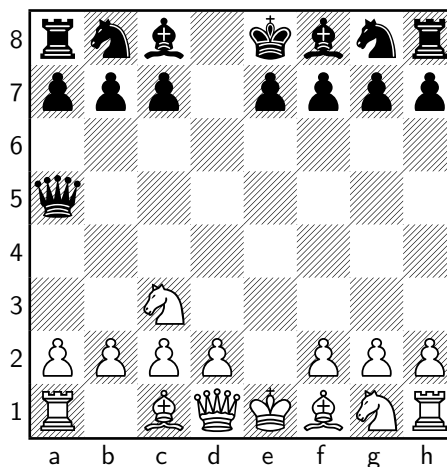
```

Die Zeilen des Gedichts, außer der letzten Zeile jeder Strophe, werden mit `\\` abgeschlossen. Die Strophen werden durch Leerzeilen getrennt. Das Gedicht erscheint eingerückt und zwischen den Strophen ist ein kleiner Abstand. Die ohne weiteres verfügbare *verse*-Umgebung kann, durch Laden des Pakets *verse* (von Peter R. Wilson), ersetzt werden. Damit erhält man mehr Darstellungsmöglichkeiten.

Schach wird mit dem Paket *skak* von Torben Hoffmann dargestellt. Es kann entweder den Verlauf einer Partie wiedergeben oder eine vorgegebene Stellung. Die in einer Partie oder Schachaufgabe erreichte Stellung kann als Schachbrettzeichnung abgebildet werden. Bei der Beschreibung einer Partie werden die eingegebenen Züge gespeichert, so dass der Verlauf einer Partie wiedergegeben und die erreichte Stellung jederzeit gezeigt werden kann. Der Befehl `\newgame` beginnt ein Spiel und mit `\mainline` können die Züge aufgeschrieben werden. Die Offiziere werden bei der Angabe der Züge für Weiß und Schwarz mit englischen Abkürzungen gekennzeichnet: König K, Dame Q, Turm R, Läufer B und Springer N. Eine Variante wird mit `\variation` notiert und `\lastmove` gibt den letzten Zug aus. Die erreichte Stellung wird mit `\showboard` abgebildet. Folgendes [Beispiel](#) demonstriert die Anwendung der Befehle.

```
\newgame % neues Spiel
\mainline{1.e2e4 d5}
Skandinavisch \par
Der Zug \lastmove{}
ist aggressiv. \par
% der Zug d7d5 wird
% kurz d5 geschrieben
\mainline{2.e4xd5}
nicht \variation{2.e5}
\par \mainline{2...Qd5
3.Nc3 Qa5}
% Offiziere mit Groß-
% buchstaben, Q=Queen
% K=King, R=Rook, B=
% Bishop, N=Knight
\par \bigskip
\showboard % Stellung
```

1 e4 d5 Skandinavisch
Der Zug 1...d5 ist aggressiv.
2 exd5 nicht 2 e5
2... ♔d5 3 ♘c3 ♕a5



Eine Schachaufgabe mit vorgegebener Stellung wird mit

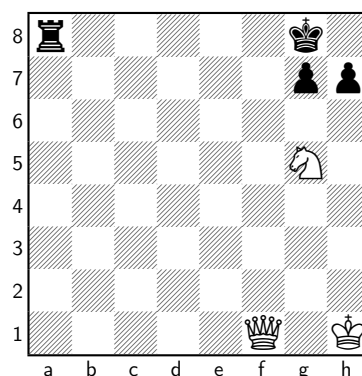
```
\newgame
\fenboard{r5k1/6pp/8/6N1/8/8/8/5Q1K w - - 0 1}
\scalebox{0.8}{\showboard}
```

dargestellt (Abbildung, siehe unten). Der Befehl `\newgame` steht am Anfang einer neuen Zugfolge oder Stellung und der Befehl `\fenboard{...}` beschreibt eine Stellung in der Forsyth-Edwards-Notation (FEN), welche hier nicht erläutert werden soll. Zu beachten ist, dass bei der FEN (abweichend von obiger Zug-Notation) die Figuren von Weiß mit K, Q, R, B, N und P (Bauer) und die von Schwarz mit k, q, r, b, n und p bezeichnet werden. Die Abbildung des Schachbretts erfolgt wieder mit dem Befehl `showboard`. Da die Abbildungsgröße meistens angepasst werden soll, skaliert man sie (im Beispiel auf 0,8 mal die Originalgröße) mit dem Befehl `scalebox{...}{...}`,

der verfügbar ist, wenn das Paket *graphicx* (oder das ältere *graphics*-Paket) geladen wurde.

Schachaufgabe von Siegbert Tarrasch.

Weiß am Zug setzt in 5 Zügen matt.



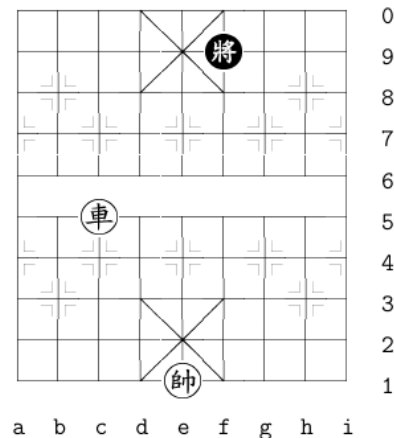
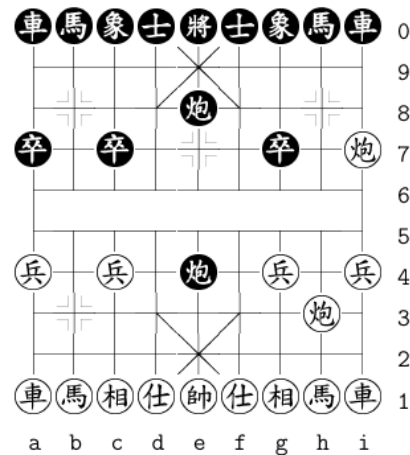
Xiangqi oder chinesisches Schach kann mit dem Paket *xq* von Stephan Weinhold und Sebastian Pipping aufgezeichnet werden. Leider hat das Paket einige Fehler, so dass es beispielsweise mit der Dokumentklasse *article*, nicht aber mit *scrartcl* läuft. Das unten gegebene [Beispiel](#) schließt daher den *documentclass*-Befehl ein. Zunächst wählt man eine Sprache für die Zeichen, welche den Spielfiguren zugeordnet werden. Möglich sind unter anderem Deutsch (*german*) und Englisch (*english*). Die roten Spielfiguren und ihre Zeichen (in Deutsch beziehungsweise Englisch) sind: Feldherr (F, K), Mandarin (M, A), Elefant (E, E), Wagen (W, R), Kanone (K, C), Pferd (P, H) und Soldat (S, P). Die entsprechenden schwarzen Spielfiguren werden mit Kleinbuchstaben bezeichnet. Der Befehl `\newgame` beginnt ein neues Spiel. Züge werden mit dem Befehl `\move` eingegeben, wobei roter und schwarzer Zug nacheinander mit jeweils vier Zeichen angegeben werden. Jeweils zwei Zeichen stehen für Start- und Zielkoordinaten. Deren Bedeutung ergibt sich aus der Abbildung des Spielfelds. Soll der rote oder schwarze Zug fehlen, wird stattdessen *xxxx* geschrieben. Der Befehl `\cr` beziehungsweise `\cb` fügt dem folgenden Zug von Rot beziehungsweise Schwarz einen Kommentar hinzu. Das Spielfeld und die erreichte Stellung werden mit `\showboard` ausgegeben.

Der Befehl `\textpiece` bindet Spielsteine in den Text ein. Mit `\resetboard` erzeugt man ein leeres Spielbrett, auf dem man mit `\piece` Spielsteine setzen kann. Danach wird das Brett mit `\showboard` gezeigt. Folgendes [Beispiel](#) verdeutlicht die Anwendung der Befehle.


```

\documentclass{article}
\usepackage{xq}
\begin{document}
\mylanguage{german}
\newgame % Spiel beginnen
\move b3e3 b8e8 % 1. Zug
\move e3e7 e8e4 % 2. Zug
\cr{ ?} % Kommentar, 3. rot
\move e7i7 xxxx % 3. rot
\cb{ !} % Komm., 3. schwarz
\move xxxx h8e8 % 3. schwarz
\showboard \textbf{Matt\,!}
%
\par Der rote Feldherr
\textpiece{F} sieht anders
aus als der schwarze
\textpiece{f} . \par
%
\resetboard % leeres Brett
\piece ff9 % schwarzer und
\piece Fe1 % roter Feldherr
\piece Wc5 % und ein Wagen
\showboard % zeige Stellung
\end{document}

```



Sudoku ist ein mathematisches Rätselspiel auf Grundlage einer gezeichneten quadratischen Tabelle mit 9 Teilquadrate. Jedes Teilquadrat besteht aus 3×3 Feldern, die mit den Ziffern 1 bis 9 gefüllt werden sollen. Einige Ziffern sind bereits vorgegeben. Die fehlenden Ziffern sollen durch logische Überlegungen gefunden werden. Mit dem Paket *sudoku* von Paul Abraham kann man solch eine Rätseltabelle abbilden, zum Beispiel:

```

\scalebox{0.65}{
\begin{sudoku-block}
| | | 8|6| | | | |
| | | | | | | 5|2|
3| | | 7| | | | 4|
6|9| | | 2| | | 7|
| | 1| 3| | | | |
| 2|4| | 1| 8|9|
| | 7| 4| 6| | |
| | 8| 7|9|4| | |
2| 1| | | 7| | |
\end{sudoku-block}
}

```

			8	6				
							5	2
3			7				4	
6	9				2			7
		1	3					
	2	4			1		8	9
		7		4		6		
		8		7	9	4		
2		1				7		

Strichode ist eine maschinenlesbare Schrift aus unterschiedlich breiten, parallelen Strichen. Das Paket *makebarcode* von Zdeněk Wagner ermöglicht es, verschiedene

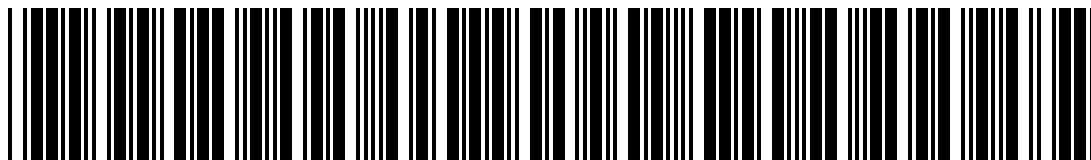
eindimensionale Strichcodes zu erzeugen. Beim Laden des Pakets kann man als optionale Argumente den Typ des Strichcodes und dessen Größe angeben. Mit

```
\usepackage[code=Code39,X=0.5mm,ratio=3,H=2cm]{makebarcode}
```

lädt man *makebarcode* mit Standard Code 39, Modulbreite 0,5 mm, Breitenverhältnis (zwischen schmalen und breiten Balken) 3, Höhe 2 cm. Der Zeichensatz (Alphabet) enthält die englischen Großbuchstaben, die Ziffern 0 bis 9, das Leerzeichen und die sechs Zeichen `+ - . $ / %`. Die Länge ist grundsätzlich nicht beschränkt. Bei der Wiedergabe des Codes im Klartext werden manchmal Start und Stopp als `*` geschrieben. Vorteile von Standard Code 39 sind die weite Verbreitung (bei den meisten Strichcodelesegeräten) und gute Maschinenlesbarkeit. Nachteile sind der kleine Zeichensatz und die geringe Informationsdichte (großer Platzbedarf). Zum [Beispiel](#) wird mit

```
\barcode{ALFRED H. GITTER}
```

der Strichcode

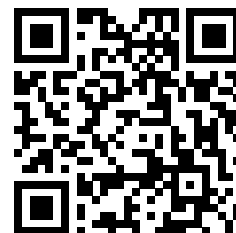


ausgegeben, welcher die Zeichenkette ALFRED H. GITTER darstellt.

QR-Code ist eine quadratische Zeichnung, die aus kleinen schwarzen Quadraten besteht und eine Zeichenkette codiert. Mobiltelefone und Computer mit Kamera verfügen oft über ein Programm, welches QR-Code lesen und decodieren kann. Beispielsweise können so Internetadressen bequem von einem Papierausdruck abgelesen und in einem Browser geöffnet werden. Maximal kann ein QR-Code knapp 3 kB an Information beinhalten. QR-Code kann mit dem Paket [qrcode](#) von Anders O. F. Hendrickson erzeugt werden. [Beispiel](#):

QR-Code mit 3 cm Seitenlänge
für eine Wikipedia-Webseite

```
\qrcode[height=3cm]{  
https://de.wikipedia.org/wiki/QR-Code  
}
```



7.3 Präsentation mit *beamer*

Für die Bild-Projektion während eines Vortrags empfiehlt sich die Dokumentklasse *beamer*. Der folgende Quellcode zeigt [ein kurzes Beispiel](#). Die Abfolge der Befehle wird durch die zugehörigen Erläuterungen unterbrochen.

Die Präambel enthält einen einleitenden *documentclass*-Befehl, in dem die Dokumentklasse *beamer* gewählt wird. Im Gegensatz zu anderen Dokumentklassen wird bei *beamer* der Text linksbündig (nicht im Blocksatz) und ohne automatische Silbentrennung dargestellt. Aufgrund der geringen Textmenge pro Seite kann man bei Bedarf Trennungen manuell einsetzen.

```
\documentclass[14pt]{beamer} % Dokumentklasse
```

Die Schriftgröße wird hier mit dem relativ großen Wert 14 pt festgelegt, da die Textmenge bei der Präsentation gering, die Entfernung des Lesers jedoch sehr groß sein kann. Zur Auswahl stehen bei *beamer* mehr Schriftgrößen als bei anderen Standard-Dokumentklassen, nämlich 8 pt, 9 pt, 10 pt, 11 pt, 12 pt, 14 pt, 17 pt und 20 pt. Ohne besondere Festlegung der Schriftgröße verwendet *beamer* die Größe 11 pt. Das Format einer Seite, bei einer Projektion (aus historischen Gründen) auch Folie genannt, braucht nicht angegeben zu werden. Es beträgt 128 mm × 96 mm, aus dem sich ein Seitenverhältnis von 4:3 ergibt.

Wir fügen nun eine Zeile ein, die den Befehl `\Tiny` anweist, nicht Schriftgröße 4, sondern (wie der Befehl `\tiny`) Schriftgröße 5 zu verwenden.

```
\let\Tiny=\tiny % Warnung (size <4> not available) verhindern
```

Damit verhindern wir die Warnung, dass Schriftgröße 4 nicht zur Verfügung steht. Da diese Schriftgröße ohnehin nicht gebraucht wird, ist die Sache eigentlich belanglos, aber wir möchten keine überflüssige Warnung (*Font shape ‘OT1/cmss/m/n’ in size <4> not available*) bekommen.

Dann werden, wie in anderen Dokumentklassen, Pakete geladen.

```
\usepackage[ngerman]{babel} % Pakete laden
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{graphicx, multimedia}
\usepackage{tikz, amsfonts, amsmath}
```

Mit den ersten drei Befehlen wird die deutsche Sprachanpassung vorgenommen. Dann werden Pakete für die Einbindung von Bildern, Filmen, Grafiken, mathematischer Symbole und Gleichungen geladen. Die Pakete *xcolor* für farbigen Texte und *hyperref* für Hyperlinks werden von der Dokumentklasse *beamer* automatisch geladen. Damit stehen die Farben des Pakets *xcolor* zur Verfügung, die in Abschnitt 4.1 (Seite 46) genannt wurden.

Das Layout einer Präsentation wird im Wesentlichen durch Wahl eines Themas vorgegeben, wozu der Befehl `\usetheme` dient.

```
\usetheme{default} % Layout-Vorgabe
```

Hier wird das einfache Standard-Thema *default* gewählt, welches eine Seite ohne Rand-Spalten darstellt und schwarz auf weißem Hintergrund schreibt. Will man für die Seitenüberschrift einen farbigen, statt eines weißen Hintergrunds (Standard-einstellung *default*), kann man den Befehl `usecolortheme{option}` anfügen, wobei *option* zum Beispiel *seahorse* oder *wolverine* sein kann. Derartige Malkunst lenkt jedoch vom Inhalt ab.

Einige Standardvorgaben sollte man ändern. Mit

```
\setbeamertemplate{navigation symbols}{} % Navi-Zeile löschen
```

wird die Einblendung der *navigation symbols* ausgeschaltet, die ansonsten am unteren Rand jeder Seite erscheinen würden. Dann wird mit

```
\setbeamerfont{page number in head/foot}{size=\small} % S.zahlgröße
```

die Schriftgröße für die Seitenzahl auf *small* gesetzt und mit

```
\setbeamertemplate{footline}[frame number] % Seitenzahl am Fuß
```

in der Fußzeile (*footline*) die Angabe der aktuellen Seite und der Seiten-Anzahl mit der Option *frame number* angefordert. Nun wird das Format der später automatisch eingefügten Abschnittsseiten (*sectionpage*) festgelegt.

```
\setbeamertemplate{section page} % Format der
{ \begin{centering} % Abschnitts-Seite
  \usebeamerfont{section title}\insertsection\par
  \end{centering} }
```

In einer (*centering*-Umgebung mittig ausgerichtet wird hier nur der Abschnittstitel in der dafür standardmäßig vorgesehenen Schriftart eingesetzt. Ähnlich wie in anderen Dokumentklassen wird die Titelseite vorbereitet.

```
\title{Dieser Tag ein Leben} % Format der
\author{Melcher Melcherson} % Titelseite
% \institute{Familie Melcherson aus Stockholm}
\date{}
```

Bei wissenschaftlichen Vorträgen wird unter dem Redner (*author*) noch das Institut genannt, aus dem dieser kommt. Hier entfällt das (die betreffende Zeile ist auskommentiert). Zu Beginn jedes Abschnitts soll automatisch eine *sectionpage* eingefügt werden, die den Titel des Abschnitts mitteilt. Dies wird mit

```
\AtBeginSection{\frame{\sectionpage}} % Abschnitts-Seite
```

veranlasst. Nun haben wir (bis auf später erläuterte, mögliche Ergänzungen) das Ende der Präambel erreicht.

Der Textinhalt beginnt mit der *document*-Umgebung.

```
\begin{document} % Dokument-Anfang
```

Nach dem Anfang der *document*-Umgebung folgt die Titelseite.

```
\begin{frame} \titlepage \end{frame} % Titelseite
```

Eine Präsentationsseite heißt *frame* und wird daher in einer *frame*-Umgebung erzeugt. Hier wird die Titelseite aus den in der Präambel gemachten Angaben automatisch gestaltet. Die Präsentation kann mit dem *section*-Befehl in verschiedene Abschnitte unterteilt werden. Der Titel eines Abschnitts wird in geschweiften Klammern angegeben.

```
\section{der Ferien erster Teil} % Abschnitt 1
```

Eine Liste aller Abschnittstitel kann, bei geeignetem Layout (nicht mit der hier empfohlenen *default*-Einstellung), am Rand jeder Seite in einer Navigationsspalte gezeigt werden.

Es ist in Vorträgen üblich, einzelne Teile einer Seite nacheinander sichtbar zu machen. Dies lässt sich einfach erreichen, indem die Anzeige mit dem Befehl `\pause` unterbrochen wird, bis der Vortragende durch Drücken der Eingabetaste oder der linken Maustaste die Fortsetzung bewirkt.

```
\begin{frame}{Die Familie}
  Meine Kinder:
  \begin{itemize}
```

```

\pause \item{Malin}
\pause \item{Johann}
\pause \item{Niklas}
\pause \item{Pelle}
\begin{itemize}\item{Jocke}\end{itemize}
\end{itemize}
\end{frame}

```

In obigem Beispiel werden die Namen der Kinder der Familie Melcherson nacheinander sichtbar gemacht. Im *handout*-Modus (siehe unten) werden die *pause*-Befehle nicht beachtet und alles ohne Unterbrechung auf eine Seite geschrieben. Die Einblendung einzelner Teile der Seite nennt man Overlay. Die schrittweise Darstellung des Seiteninhalts soll die Aufmerksamkeit des Lesers auf den gerade besprochenen Teil lenken. Der Leser wird dadurch jedoch bevormundet und die Einblendung lenkt *per se* den Leser vom Inhaltlichen ab. Daher sollte man Overlays, jedenfalls in wissenschaftlichen Vorträgen, sparsam verwenden.

Oft braucht man zwei Spalten auf einer Seite, zum Beispiel um Text neben ein Bild zu platzieren. Die Dokumentklasse *beamer* stellt hierfür (im Gegensatz zu anderen Dokumentklassen) die *columns*-Umgebung bereit, innerhalb derer man Spalten mit jeweils einer *column*-Umgebung erzeugen kann.

```

\begin{frame}{Die Vegetation}
\begin{columns}
\begin{column}{0.5\textwidth}
\IfFileExists{bild1.jpg} % prüfe, ob die Bilddatei existiert
{ \includegraphics[scale=0.5]{bild1.jpg} } % Bilddatei ist da
{ \textcolor{red}{Bild-Datei fehlt} } % Bilddatei fehlt
\end{column}
\begin{column}{0.3\textwidth}
\centering % centering-Befehl zentriert (in dieser Umgebung)
blau blüht der Ehrenpreis\\
neben dem Schreinerhaus
\end{column}
\end{columns}
\end{frame}

```

In obigem Beispiel werden zwei Spalten gebildet. In der linken Spalte wird mit dem *includegraphics*-Befehl ein Bild aus der Datei `bild1.jpg` geladen. In der rechten steht ein erklärender Text, der mit *\centering* mittig gesetzt wird.

Nach dem Beginn des zweiten Abschnitts soll ein Bild gezeigt werden, das aus urheberrechtlichen Gründen nicht im Handout (siehe unten) erscheinen darf. Um die Befehle zur Darstellung des Bildes nur bei der Präsentation wirken zu lassen, schließt man sie in die geschweiften Klammern nach einem *only<beamer>*-Befehl ein.

```

\section{der Ferien zweiter Teil} % Abschnitt 2
\begin{frame}{Blick über die große Ostsee}
\centering % centering-Befehl zentriert (in dieser Umgebung)
\only<beamer>{
% \hspace{-0.9cm} % der hspace-Befehl verrückt das Bild horizontal
\IfFileExists{bild2.jpg} % prüfe, ob die Bilddatei existiert
{ \includegraphics[scale=0.7]{bild2.jpg} } % Bilddatei ist da
}

```

```

        { \textcolor{red}{Bild-Datei fehlt} }      % Bilddatei fehlt
      }
\end{frame}

```

In obigem Beispiel wird ein Bild aus der Datei `bild2.jpg` eingefügt. Wenn das Bild sehr groß dargestellt werden soll, kann man es nach links verschieben und den leeren Textrand nutzen. Dies würde durch den `hspace`-Befehl mit einer negativen Längenangabe erreicht werden. Manchmal möchte man ein Bild seitenfüllend, also mit maximaler Höhe und Breite, darstellen. Dabei nimmt man in Kauf, dass das Bild verzerrt wird, wenn das Seitenverhältnis Höhe/Breite der Bildvorlage nicht mit dem der Seite übereinstimmt.

```

\IfFileExists{bild3.jpg}      % prüfe, ob die Bilddatei existiert
{ { \setbeamertemplate{background canvas}{\includegraphics[%
  height = \paperheight,width = \paperwidth]{bild3.jpg}}
\begin{frame}[plain] \end{frame} } }      % Bilddatei ist da
{ \textcolor{red}{Bild-Datei fehlt} } }      % Bilddatei fehlt

```

Obige Befehle dehnen das Bild automatisch. Da das Bild die ganze Seite ausfüllt, ist kein Platz für eine Überschrift oder die Anzeige der Seitenzahl. Die Seitenzahl wird aber um 1 erhöht.

Filme kann man mit dem `movie`-Befehl in die Seite einbetten, wenn der PDF-Viewer diese selbst abspielt oder auf einen vom Betriebssystem des Rechners bereitgestellten Mediaplayer zugreifen kann, welcher das gewünschte Film-Format unterstützt. Ein Film sollte natürlich nur im *beamer*-Modus, nicht im Handout erscheinen.

Besonderheiten bestehen in der Dokumentklasse *beamer* bei der Einbindung bestimmter Umgebungen in eine *frame*-Umgebung. Betroffen sind unter anderem die *verbatim*-Umgebung, die *lstlisting*-Umgebung des Pakets *listings*, und die *comment*-Umgebung des Pakets *comment*. In diesen Fällen muss der *frame*-Umgebung die Option *fragile* hinzugefügt werden:

```

\begin{frame}[fragile]{Test}
\begin{verbatim}
\LaTeX
\end{verbatim}
\end{frame}

```

Backup-Seiten werden bereit gehalten, um bei Bedarf zusätzlich gezeigt zu werden. In einer Präsentation soll auf jeden Fall eine bestimmte Mindestanzahl von Seiten gezeigt werden. Die möglicherweise darüber hinaus gezeigten Seiten (*Backup*) sollen bei der Berechnung der Seiten-Anzahl nicht berücksichtigt werden. Um einen Bereich mit solchen Backup-Seiten bilden zu können, werden in der Präambel (vor `\begin{document}`) zwei neue Befehle definiert. Zunächst der Befehl `\beginbackup`

```

\newcommand{\beginbackup}      % Beginn der
{ \newcounter{framenumbervorappendix}      % Backup-Seiten
  \setcounter{framenumbervorappendix}{\value{framenumber} } }

```

und dann der Befehl `\backupend`

```

\newcommand{\backupend}      % Ende der Backup-Seiten

```

```
{ \addtocounter{framenumbervorappendix}{ -\value{framenumber} }
  \addtocounter{framenumber}{ \value{framenumbervorappendix} } }
```

Am Ende des regulären Vortragsteils beginnt mit unserem neu definierten Befehl `\beginbackup` der, Backup genannte, Zugabe-Teil der Präsentation.

```
\beginbackup % Backup
\begin{frame}<handout:0>{Bootsfahrt im Schärengraben}
  \only<beamer>{
    \movie[autoplay, width=200pt, height=150pt]{
      {film.mp4}
    }
  }
\end{frame}
\backupend
```

Mit dem ebenfalls oben definierten Befehl `\backupend` endet das Backup. Die Seiten des Backup, welche nicht im Handout erscheinen sollen, werden mit der Option `<handout:0>` begonnen (siehe unten).

```
\end{document} % Dokument-Ende
```

Nach dem Backup endet die *document*-Umgebung.

Handout ist eine gedruckte Kurzfassung der Präsentation. Dafür wird der *documentclass*-Befehl um die Option *handout* erweitert. Will man mehrere, zum Beispiel 8 (möglich sind auch 2, 4, oder 16), Präsentationsseiten auf einer Druckseite unterbringen, kann man das Paket *pgfpages* verwenden (eine Alternative ist das Paket *pdfpages* von Andreas Matthias). Folgende Befehle erzeugen ein Handout auf DIN-A4-Seiten im Hochformat.

```
\documentclass[14pt,handout]{beamer}
\usepackage{pgfpages}
\pgfpagesuselayout{8 on 1}[a4paper,border shrink=5mm]
```

Will man 4 oder 16 Präsentationsseiten auf eine DIN-A4-Seite drucken, sollte die Option *landscape* im *pgfpagesuselayout*-Befehl ergänzt werden.

Wenn, wie oben (Seite 100), ein Teil der Präsentationsseite nur im *beamer*-Modus gezeigt wird, dann wird im *handout*-Modus immernoch die Seite mit Überschrift (und eventuell weiterem Text) dargestellt. Soll dagegen eine ganze Seite im Handout fehlen, fügt man an den Seitenbeginn die Option `<handout:0>` an, zum Beispiel (für obige Seite) `\begin{frame}<handout:0>{der Ferien zweiter Teil}`. Umgekehrt würde mit der Option `<beamer:0>` eine Seite nur im Handout erscheinen und im *beamer*-Modus fehlen. Damit kann man zwei Ausführungen einer Seite erstellen, eine für die Präsentation und die Alternative für das Handout.

Mehr Information zur Dokumentklasse *beamer* gibt es im ausführlichen [Benutzerhandbuch](#) von Till Tantau, Joseph Wright und Vedran Miletic, in einer [Präsentation](#) von Ki-Joo Kim und in [Beispielen](#) von Sascha Frank.

Eine Alternative zu *beamer* ist zum [Beispiel](#) die Dokumentklasse *elpres*, welche im Paket *elpres* von Volker Kiefel definiert wird. Damit kann man Präsentationen einfacher erstellen. Es fehlen die umfangreichen Gestaltungsmöglichkeiten des Layouts und andere Extras von *beamer*. Aber vielleicht braucht man die ja nicht.

7.4 Briefe

Mehrere Pakete stellen besondere *Dokumentklassen für Briefe* bereit, zum Beispiel *letter* von Leslie Lamport, Frank Mittelbach und Rainer Schöpf. Im folgenden wird die Dokumentklasse *scrlltr2* verwendet, welche Teil des Pakets *KOMA-Script* von Markus Kohm ist. Sie bietet zahlreiche Anpassungsmöglichkeiten, aber wir beschränken uns auf eine einfache, weitgehend automatische Formatierung. Der folgende Quellcode zeigt ein *Beispiel* mit Erläuterungen.

Da ein Brief oft als Vorlage für den nächsten Brief dient, ist es sinnvoll, alle Textbausteine als Variablen zu definieren, die wir möglicherweise in einem Brief verwenden könnten. Bei sehr umfangreicher Korrespondenz mag es sinnvoll sein, derartige Definitionen in einer besonderen Datei auszulagern. Wir bescheiden uns hier aber mit der einfachen Variante *Alles in einer Datei*.

In den Optionen des *documentclass*-Befehls wird nur entschieden, welche Textbausteine im aktuellen Brief gezeigt werden. Die Wertzuweisung erfolgt später. Man beginnt die Präambel mit

```
\documentclass[paper=a4,fontsize=12pt%
,fromalign=center    % andere Werte: left, right
,fromphone           % Telephonnummer zeigen
,frommobilephone     % Handynummer zeigen
,fromemail           % E-Mail-Adresse zeigen
%,fromurl            % URL der Homepage zeigen
%,fromlogo           % Logo (Bild) zeigen
]{scrlltr2}
```

wobei die Schriftgröße hier auf 12 pt festgelegt wurde, um eine gute Lesbarkeit auch für Menschen mit schlechterem Sehvermögen zu erreichen. Die Platzierung der Absenderdaten im Oberteil der ersten Seite wird mit der Variablen *fromalign* bestimmt. Die Nennung weiterer Variablen bewirkt, dass die zugehörigen Textbausteine gezeigt werden. Durch Voranstellung des %-Zeichens (Auskommentieren) wird eine Zeile unwirksam und der zugehörige Textbaustein nicht gezeigt. Entsprechend wird mit folgender Zeile (ohne %) das Bankkonto am Ende der ersten Seite angegeben.

```
%\setkomavar{firstfoot}{\usekomavar{frombank}}
```

Anschließend werden die Pakete für deutsche Sprachunterstützung geladen.

```
\usepackage[ngerman]{babel}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
```

Will man eine andere Schrift, lädt man das entsprechende Paket.

```
\usepackage{libertine} % benutze freie Schriftart Linux Libertine
```

Will man Bilder, zum Beispiel ein Logo, einfügen, braucht man das *graphicx*-Paket.

```
\usepackage{graphicx}
```

Nun erfolgt die Wertzuweisung an die Variablen für Textbausteine, welche Name, Adresse und Kommunikationsdaten des Absenders beschreiben.

```
\setkomavar{fromname}{Jim Knopf}
\setkomavar{fromaddress}{Bahnhofstraße 1\\12345 Lummerland}
```

```

\setkomavar{fromphone}{(\,01\,23\,) 20\,76\,33}
\setkomavar{frommobilephone}{01\,51\,25\,76\,24\,57}
\setkomavar{fromfax}{(0123) 45679}
\setkomavar{fromemail}{jimknopf@mollymail.lul}
\setkomavar{fromurl}[Homepage: ]{http://www.eah-jena.de/$\sim$gitter}

```

In der Absenderadresse erzeugt `\\` einen Zeilenumbruch. Mit `\sim` wird in der URL-Zeichenkette das Zeichen `~` (Tilde) erzeugt. Um ein Logo aus einer Bilddatei *logo* einfügen zu können, wird der Dateiname bekannt gemacht:

```

\setkomavar{fromlogo}{\includegraphics{logo}} % Bilddatei für Logo

```

Die Bankverbindung des Absenders wird in der Variablen *frombank* gespeichert.

```

\setkomavar{frombank}{IBAN: DE49 3002 0900 1805 6122 25\\
bei der TARGOBANK (Düsseldorf), BIC: CMCIDEDD}

```

In die *document*-Umgebung wird eine *letter*-Umgebung gesetzt, welche den Brieftext enthält. Pflichtargument ist die Adresse des Empfängers.

```

\begin{document}
\begin{letter}{Wilde 13\\c/o Mahlzahn\\Alte Straße 133\\Kummerland}

```

In der Empfängeradresse erzeugt `\\` einen Zeilenumbruch. Der Briefftext muss mit dem *opening*-Befehl beginnen, welcher die Anrede setzt. Briefe an Ämter, Firmen und Autoritäten sollte man formal höflich beginnen:

```

\opening{Sehr geehrte Damen und Herren,}

```

Wird ein Mensch mit akademischen Graden oder anderen Ehrentiteln im Brief angesprochen, stellt man den höchstrangigen Titel und Grad dem Namen voran. Im Zweifelsfall wird aufgerundet! Private Briefe kann man meistens mit *Lieber ...* oder *Liebe ...* beginnen. Altmodische Anrede- und Grußformeln wirken im Schriftverkehr nicht nachteilig und sind allemal besser als ein *Hallo!* wie in der Telefonwarteschleife.

Dann folgt der eigentliche Briefftext.

Ich wäre Ihnen sehr verbunden, wenn Sie mir den gegenwärtigen Aufenthaltsort der Prinzessin Li Si aus Mandala mitteilen würden, sofern er Ihnen bekannt sein sollte.

In einem Brief sollte man Bildelemente grundsätzlich nur verwenden, wenn sie a) einen wohldefinierten Zweck erfüllen, b) persönlich und originell erscheinen oder c) die übergeordnete Leitung es so will. Emoticons gehören in der Regel nicht dazu. Die abschließende Grußformel wird mit dem *closing*-Befehl gesetzt. Meistens ist *Mit freundlichen Grüßen*, gut. Oft ist ein vorangestellter Dank noch besser.

```

\closing{Ich danke für Ihre Bemühungen und verbleibe\\
mit freundlichen Grüßen,}

```

Die Grußformel kann also auch aus mehreren (mit `\\` getrennten) Zeilen bestehen. Anlagen können mit dem *encl*-Befehl aufgeführt werden.

```

\encl{Bild der Prinzessin, von Herrn Pi Plu} % Anlagen

```

Geht der Brief an mehrere Empfänger, kann man sie im Verteiler nennen.

```

\cc{Die Wilde 13\\Kopie an Kaiser Pung Ging} % Verteiler

```


Am Ende werden *letter*- und *document*-Umgebung geschlossen.

```
\end{letter}  
\end{document}
```

Um eine einfache (deutsche oder englische) Rechnung zu schreiben, kann man zum [Beispiel](#) die Umgebung *invoice* des Pakets [invoice2](#) von Simon Dierl einfügen. Die Verwendung des Pakets (in L^AT_EX 2_ε) wird leider durch Programmfehler erschwert.

7.5 L^AT_EX-Editor *Gummi*

Die Erstellung von L^AT_EX-Dateien wird durch besondere Editoren erleichtert. Erfahrene Nutzer können einige ausprobieren und brauchen dafür hier keine Erklärungen. Für Einsteiger ist der L^AT_EX-Editor *Gummi* zu empfehlen, da er klein und schnell ist und, aufgrund seines geringen Funktionsumfangs, wenig Einarbeitung erfordert. *Gummi* kann auf verschiedenen Linux-Betriebssystemen über die Paketverwaltung installiert werden. Für Kleinweich-Betriebssysteme (*Microsoft*) steht *Gummi* derzeit nicht zur Verfügung und man muss einen anderen Editor wählen.

Gummi zeigt in einem besonderen Fenster das gestaltete Dokument, wie es schließlich als PDF-Datei vorliegen wird. Sobald man einen fehlerhaften Befehl eingibt, wird die PDF-Vorschau nicht mehr gezeigt und die Zeile mit dem Fehler rot hervorgehoben. (Wenn jedoch eine schließende geschweifte Klammer } vergessen wird, kann *Gummi* die Fehlerzeile nicht hervorheben.)

Einen Textbereich kann man mittig, links- oder rechtsbündig ausrichten, indem man ihn mit der Maus markiert und dann in der Werkzeugleiste (oben links) das entsprechende Symbol anklickt. Entsprechend wird ein markierter Textbereich fett, kursiv oder unterstrichen dargestellt, indem man das entsprechende Symbol anklickt. Das Einfügen von Abbildungen in einer *figure*-Umgebung wird durch Klick auf das Auswahlfeld *Bild* am unteren Rand des Editor-Fensters eingeleitet. Es erscheint ein Formularfenster, in dem man die Bilddatei auswählt, die Bildunterschrift (Titel) einträgt und eine Skalierung der Bildgröße wählt. Die Skalierung kann später, durch Änderung des Werts im Quelltext, noch korrigiert werden. Die Marke oder *Label* ist ein eindeutiger kurzer Name, mit dem man an beliebiger Textstelle auf das Bild verweisen kann (siehe Abschnitt 2.2 auf Seite 17). Die so eingefügten Abbildungen werden automatisch nummeriert. Tabellen beziehungsweise Matrizen können durch Klick auf das Auswahlfeld *Tabelle* beziehungsweise *Matrix* am unteren Rand des Editor-Fensters eingefügt werden. Nach dem Eintragen der Parameter und Klick auf *Anwenden* werden die zugehörigen Befehle eingesetzt.

Zur Kompilierung klickt man im Reiter *Dokument* auf den Menü-Punkt *Dokument kompilieren*. Man beachte, dass einige Änderungen erst nach zwei- oder mehrfachem Kompilieren sichtbar werden. Zur Übersetzung in eine druckfertige PDF-Datei klickt man im Reiter *Datei* auf den Menü-Punkt *Exportieren als PDF*.