

WIAI L^AT_EX SCRIPT

A quick-start manual for
typesetting academic texts



Imprint

The L^AT_EX Script (version 1.4.3 from November 2025) has been assembled by the Student Council of the Information Systems and Applied Computer Sciences Faculty (Fachschaft WIAI) at the University of Bamberg.

It is licensed under Creative Commons “Attribution-ShareAlike 4.0 International” (CC BY-SA 4.0):



<http://creativecommons.org/licenses/by-sa/4.0/>

Upon request, allowances exceeding the limitations of this license may be granted.

Contents

1	Why use \LaTeX?	3
2	How does \LaTeX work?	4
2.1	What do we need to use \LaTeX ?	5
2.2	The commands	5
3	Basic document structure	7
3.1	Preamble	7
3.1.1	Document class	7
3.1.2	Digression: packages	8
3.1.3	Encoding	8
3.1.4	Language	9
3.2	Document environment	9
3.2.1	Continuous text	9
3.2.2	Comments	9
3.2.3	Sections and chapters	10
3.2.4	Front matter	10
3.2.5	Indices	11
4	Project structure	12
4.1	Main file	12
4.2	Partial files	12
5	Special characters	14
5.1	Spaces	14
5.2	Hyphens and dashes	15
5.3	Quotation marks	16
5.4	Diacritics	16
5.5	More special characters	17
6	Text markup	18
7	Lists	21
8	Mathematical formulas	23
8.1	A few examples	24
8.2	Growing brackets	24
8.3	Lower and upper bounds	25
8.4	Aligning equations	25

8.5	Text in math mode	25
9	Graphics	27
9.1	Inserting graphics	27
9.2	Positioning	27
10	Tables	29
11	References and footnotes	31
11.1	Footnotes	31
11.2	References	31
12	Source code listings	34
12.1	Using minted	35
12.1.1	Installation	35
12.1.2	Changing the compiler command	35
12.1.3	Defining listings	36
12.1.4	Configuring minted	37
12.2	Using listings	38
12.2.1	Installation	38
12.2.2	Defining listings	38
12.2.3	Configuring listings	38
12.2.4	Drawbacks and caveats	40
13	Reference management	41
13.1	The bibliography file	41
13.2	Citing	42
14	Prospects	44
14.1	Packages	44
14.2	Help and information	46

Chapter 1

Why use L^AT_EX?

In the early 1960s, a rather talented American Ph.D. student was asked by a big publishing company whether he wanted to write a book on compilers. He did. Soon after he had begun with his research, he realized that he wanted to start with some foundations of computer science, so he asked the publishers if the book might be a little longer. They replied, “make it as long as you feel necessary.” In 1968, the first volume was published, at that time still printed using mechanical typesetting.

This method was just disappearing then, and being replaced by new methods. However, the author did not like the results of those new methods, so, at the end of the 70s, he began to develop his own typesetting system T_EX (pronounced as [tɛx], [tɛç] or [tɛk], named after the ancient Greek word τέχνη [ˈtɛxɲɛː] meaning *art, craft*).

Today, Donald Knuth (that is the former student’s name) is a retired professor of computer science and his compiler book has grown to become the multi-volume standard work *The Art of Computer Programming*—three volumes of which are still to be written, among them the one on compilers. Unlike the book, however, T_EX is the rare occurrence of a software system that may actually be called *complete* without meaning *dead*.

Two more letters are needed for L^AT_EX: They are the initial two letters of the last name of Leslie Lamport, who, in the 80s, extended T_EX by a collection of small programs that made the entire system usable for us end-users and are responsible for its widespread adoption. The current version dates from the mid 90s.

Why are we telling you all of that? It explains some of the advantages that still distinguish L^AT_EX today: It is a mature, stable, and reliable system that does typesetting in a typographically sophisticated way and mostly automatically.

As the T_EX code is stored in plain text files (cf. chapter 2), even more advantages arise: You can structure your projects clearly (cf. chapter 4), and whenever you undo changes in the source code, you can always rely on getting exactly the same output as before rather than some more or less similar reconstruction. On a larger scale, this does also work in connection with Git or other source code versioning tools. Furthermore, you can trust your source code to be readable long-term, without any specific software. It can always be opened with any program that supports plain text.

Chapter 2

How does L^AT_EX work?

Word processing and document creation programs have to decide how to translate user input into a document layout. There are different concepts to approach this topic. When working with Microsoft Word, the rule is: a document exported as PDF looks exactly like the source document in Word. Where a graphic is placed in Word, it is also found in the PDF. Adjustments to the appearance in Word and other popular programs thus result in a direct visual change. This type of formatting is called *What you see is what you get* (WYSIWYG for short). Content and formatting are closely linked.

L^AT_EX, on the other hand, works according to the principle *What you get is what you mean* (WYGIWYM for short). Content and formatting are separated more clearly. The content is placed in a document in plain text form, together with so-called *commands*. The combination of text content and commands is also called *source code*.

To customize the presentation of the content, we do not set the text appearance itself but add appropriate commands instead. These are processed by a *compiler*, which adjusts the resulting appearance depending on the command. Line by line, the compiler processes text and commands from our source code. When all the source code has been processed by the compiler, we get the final document. There are different export options, but most of the time we output the document as a PDF — just like in Word.

As a brief example for a command, we shall use the emphasis of words or sentences. The command is `\emph{}`. We write the text we want to emphasize inside the curly braces in the source code, like this: `\emph{Good morning!}`. In the resulting PDF, this text will appear in italics: *Good morning!* There is no trace of the command identifier and the special characters. You can see, we are not writing italic text inside the source code, we just tell the compiler that certain words should be emphasized by the use of a command.

This simple example illustrates a strength of the WYGIWYM principle. We mark text elements on a semantic level and can make the associated typographic adjustments centrally — or let L^AT_EX do the configuration itself. For instance, if we want to change the way highlighting is done, we can configure this once. At all places where `\emph{}` is used, the final result will be adjusted accordingly. There is no need to make adjustments at each occurrence of an emphasized word. The principle is similar to style sheets in office programs, although more consistent and powerful.

2.1 What do we need to use L^AT_EX?

If we want to generate a PDF document with L^AT_EX, we need at least two programs. One to create the source code, and a second one to process the source code. The latter is the already mentioned compiler.

In principle, a simple text editing program is sufficient for creating the source code. Most operating systems provide such programs out of the box. Maybe you are already used to applications like Notepad++,¹ these are usable as well. Then there are advanced programs like T_EXstudio² or Texmaker³ which integrate additional functions that facilitate the use of commands. You are free to choose, but we recommend to use T_EXstudio.

As mentioned before, we need a compiler to be able to compile our source code. The compiler is usually part of a collection of *programs* and *packages* which are together called a L^AT_EX *distribution*. The included packages provide various additional commands. For now, we will skip over the many programs.⁴

Multiple different L^AT_EX distributions exist. Some of the well-known ones are MiK_TE_X,⁵ MacT_EX,⁶ and T_EX Live.⁷ It is best to install one of them right away. The installation may take several hours. For this reason, some distributions are available for download in a small and a full version. The full version contains all packages, while the small version downloads packages only when they are needed. Unfortunately, we cannot take away the decision if you would rather wait for the download at the beginning or later while you are working.

2.2 The commands

The commands used in source code follow a general structure:

```
\<command>[<optional_parameters>]{<mandatory_parameters>}
```

A command can use several optional and/or mandatory parameters. Some commands have no mandatory parameters at all. Some examples are shown in table 2.1.

Command	Effect
<code>\newpage</code>	inserts a new page
<code>\textbf{Text}</code>	prints the text in bold font
<code>\usepackage[utf8]{inputenc}</code>	sets the text encoding to UTF-8
<code>\documentclass[a4paper,12pt]{article}</code>	specifies the document class
<code>\frac{3}{4}</code>	inserts a mathematical fraction

Table 2.1: Examples for L^AT_EX-commands

¹Available at <https://notepad-plus-plus.org/>.

²Available at <https://www.textrstudio.org/>.

³Available at <https://www.xmlmath.net/texmaker/>.

⁴We will get to know one of these helper programs later on, in 13, when we are citing literature.

⁵For Windows, macOS and Linux. Available at <https://miktex.org/>.

⁶For macOS and Linux. Available at <https://www.tug.org/mactex/>.

⁷For Windows, macOS, and Linux. Available at <https://www.tug.org/texlive/>.

If a command allows multiple optional parameters that accept similar inputs, it is sometimes necessary to specify which parameter is meant. For example, the command for embedding graphics accepts optional parameters for width and height. If `[12cm, 4cm]` were entered, it would be unclear which value is intended for which parameter. To make the assignment more concrete, it is possible to specify the parameters explicitly:

```
\includegraphics[width=12cm, height=4cm]{picture.png}
```

As the examples already show, many different commands can be used. Some are intended for use in mathematical formulas, others allow the inclusion of graphics. In the beginning, it will take some getting used to. However, you don't have to learn every single command by heart: As soon as you remember the most important commands, you can easily create simple documents and look up everything else.

Chapter 3

Basic document structure

In essence, every L^AT_EX document is composed of two parts: We call the first commands within our L^AT_EX document the *preamble*. It specifies global properties of our document, such as the document class, the encoding, the language, the page format, and additional packages that we want to use. The *document environment*, on the other hand, contains the actual content of our document, i. e., the things that we will later see in our generated PDF file.

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

\begin{document}
Hello World!
\end{document}
```

(a) L^AT_EX-Code

Hello World!

(b) Result

Figure 3.1: Exemplary structure of a simple L^AT_EX document with preamble and document environment

3.1 Preamble

Let's take a closer look at the preamble. A minimal preamble should contain the following specifications:

3.1.1 Document class

We can define a document class by using the command `\documentclass[<parameter>]{<document class>}`. The most commonly used document classes that are supported by default are `article` for short documents, and `report` for longer

ones. Furthermore, you can use `book` for books, `beamer`¹ for presentations, and `letter`² for letters.

In addition to the standard document classes, the KOMA script classes have been developed. They provide alternatives to the document classes mentioned above: In lieu of `article` you can use `scrartcl`, `report` is replaced by `scrreprt`,³ and `scrbook` can be used instead of `book`. As a replacement for `letter`, one can use `scrlttr2`. A complete list of all KOMA script classes is available online.⁴ By using KOMA document classes, the layout of the generated PDF document is changed. On top of that, they provide additional functionalities. The standard document classes are designed according to US-American conventions whereas KOMA classes adhere to European norms, e. g., for writing letters.

Each `\documentclass` command can hold optional parameters in square brackets. `\documentclass[10pt,a5paper,landscape]scrartcl`, for instance, configures a KOMA script article and sets its font size to 10 pt,⁵ the page size to A5,⁶ and the orientation of the page to landscape. The language can be passed as an optional parameter, too (cf. section 3.1.4).

3.1.2 Digression: packages

```
\usepackage[<options>]{<packagename>}
```

Packages provide additional commands and functionalities that we can use within our L^AT_EX source code. There are numerous packages for different use cases (e. g., typesetting formulas, lists, ...). In order to use a package, it must be included within the preamble. To do so, the above-mentioned command is used. The most important L^AT_EX packages can be found in the Comprehensive T_EX Archive Network, short: CTAN.⁷ You can also find documentation for the packages there.

3.1.3 Encoding

```
\usepackage[utf8]{inputenc}
\usepackage[t1]
```

One use case for packages is specifying the encoding of our L^AT_EX document. The character encoding⁸ determines the available character set. The standard encoding in L^AT_EX is ASCII.⁹ It is an American character encoding and therefore does not contain German umlauts or most other special characters, which makes it unsuitable at least for non-english use cases. Instead, UTF-8¹⁰ can be used as a universal character encoding.

¹We do not cover making presentations in L^AT_EX in this tutorial. However, if you are interested in the topic, we recommend this introduction on Overleaf: <https://www.overleaf.com/learn/latex/Beamer>

²We also do not cover letters in this script. An introduction can be found on WikiBooks: <https://en.wikibooks.org/wiki/LaTeX/Letters>

³Those vowels are indeed missing, do not try to insert them.

⁴Available at: <https://komascript.de/komascriptbestandteile>

⁵The standard font size is 12 pt.

⁶The default case would be A4.

⁷Available at: <https://www.ctan.org/>

⁸cf. https://en.wikipedia.org/wiki/Character_encoding

⁹cf. <https://en.wikipedia.org/wiki/ASCII>

¹⁰cf. <https://en.wikipedia.org/wiki/UTF-8>

In L^AT_EX, we need to specify two character encodings: The input encoding (`\inputenc`), which refers to our source code, and the font encoding (`\fontenc`), which determines what the content of our PDF document looks like.¹¹ T1 is an encoding that tries to cover most European languages with a limited number of characters.

3.1.4 Language

```
\usepackage[ngerman]{babel}
```

The package `babel` provides language-specific information (e. g., on hyphenation, special characters, changing fonts, translation of labels¹² like “Chapter,” “Table of Contents,” or “Figure.” The desired language can be passed as an optional parameter. `ngerman`, for instance, is used for the new German spelling. Some packages require that the language is already passed as an optional parameter in the `\documentclass` command. In this case, just leave out the optional parameter for the language within the `babel` inclusion command.

We can also use multiple languages in our document. To do so, we pass the languages, separated by commas, as an optional parameter to the `babel` inclusion command. Within our document, we can switch between languages with the `\selectlanguage{<language>}` command. Alternatively, foreign-language text can be declared by using the following command:

```
\foreignlanguage{<language>}{<text>}
```

3.2 Document environment

The actual content of the PDF document needs to be put inside of an environment starting with `\begin{document}` and ending with `\end{document}`.

3.2.1 Continuous text

The easiest content that we can integrate into the document environment is continuous text. We can write it directly into our source code. Line breaks and multiple consecutive spaces are ignored by L^AT_EX. Blank lines create a new paragraph, that is indented by default.¹³ Manual line breaks can be enforced with two backslashes (`\`). This should be avoided, though.

3.2.2 Comments

Some characters are reserved for L^AT_EX-specific commands, for instance, the percent sign. Using a percent sign tells the L^AT_EX compiler to ignore the rest of the line, so the text after the percent character will not appear in the generated PDF document. This is called a *comment*, and it can be useful to take notes while working on a document without affecting the document itself.

¹¹Details on `fontenc` can be found at: <https://tex.stackexchange.com/questions/108417/font-encoding-in-latex>

¹²cf. section 11.2

¹³The automatic indentation of new paragraphs can be prevented by using the command `\noindent`.

There are a few more of these reserved characters, as we will see and learn to deal with in section 5.5.

3.2.3 Sections and chapters

Continuous text can be structured by headings that divide the document into sections and chapters. Needless to say, \LaTeX provides us with commands for that. The commands that are depicted in fig. 3.2 can be used with any document class.

<hr/> <pre> \section{Level 1} Lorem ipsum % ... \subsection{Level 2} Lorem ipsum % ... \subsubsection{Level 3} Lorem ipsum % ... \paragraph{Level 4} Lorem ipsum % ... \subparagraph{Level 5} Lorem ipsum % ... </pre> <hr/>	<hr/> <p>1 Level 1</p> <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit.</p> <p>1.1 Level 2</p> <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit.</p> <p>1.1.1 Level 3</p> <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit.</p> <p>Level 4 Lorem ipsum dolor sit amet, consectetur adipisicing elit.</p> <p>Level 5 Lorem ipsum dolor sit amet, consectetur adipisicing elit.</p> <hr/>
(a) \LaTeX -Code	(b) Result

Figure 3.2: Heading Levels

Some document classes provide additional commands. In a **report**, you get `\chapter{Chapter}`, and in a **book**, additionally `\part{Part}`. You can mark the command with an asterisk if you want to omit the numbering of a section and exclude it from the table of contents:¹⁴

An alternative title for the table of contents can be declared as an optional parameter in square brackets between the command and the actual title:

```
\section[Title in the TOC]{Actual Chapter Title}
```

3.2.4 Front matter

A simple front matter can be created by using the command `\maketitle`. The values to be inserted into the front matter must be specified within the preamble. Multiple authors are joined by `\and`. If the date is not specified by the `\date` command, the current date will be inserted by default. The design of the front matter depends on the specified document class.

¹⁴cf. section 3.2.5

```

\documentclass[english]{article}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[english]{babel}

```

```

\title{The World of Truffles}
\author{Fooboar Rüssel \and Fachschaft WIAI}
\date{\today}

```

```

\begin{document}
\maketitle
\end{document}

```

The World of Truffles

Fooboar Rüssel

Fachschaft WIAI

November 14, 2022

(a) L^AT_EX-Code

(b) Result

Figure 3.3: The front matter

3.2.5 Indices

The command `\tableofcontents` generates an automatically numbered table of contents by making use of the above-mentioned commands for dividing our text into sections and chapters (this can be seen in fig. 4.1 on page 13).

The numbering style and depth, and many other options can, of course, be specified manually.¹⁵ For L^AT_EX to create our table of contents properly, the project has to be compiled twice.

Besides the table of contents, you can also generate a `\listoffigures` (list of figures) and a `\listoftables` (list of tables). The captions of your figures and tables will appear within those indices.¹⁶

¹⁵We recommend the following blogpost: <https://texblog.org/2011/09/09/10-ways-to-customize-tocloft/>

¹⁶cf. chapter 9 (Graphics) and chapter 10 (Tables) for more information on captions

Chapter 4

Project structure

In the previous chapters, we have only seen very short \LaTeX examples. \LaTeX can of course also be used to create larger documents and projects, such as a thesis. In order not to lose the overview in the source code and to avoid that source files become too long, a reasonable structuring of a larger \LaTeX project is advisable. For this purpose, the source code is divided into different files, which will be discussed in more detail in the following sections.

4.1 Main file

In large projects, we typically use one main file, which is often called `main.tex`. It is, in a sense, the structural skeleton of the project, as it contains the basic structure including the preamble. The title, table of contents, as well as the individual chapters of a work are integrated in this main file (cf. fig. 4.1). The inclusion of the individual sections is typically done by the `\input{...}` command ¹ with the file to be included as an argument.

4.2 Partial files

Partial files are files that are included within another file, most often the main file. In a thesis, for example, these can represent individual chapters. You are free to decide how granular the division of the content into individual files should be. The files that are included by the main file do not contain a preamble, since this is already present in the main file. Neither do the commands `\begin{document}` and `\end{document}` appear again.

¹There exists another command called `\include{...}` that works slightly differently. It requires you to specify the file to be included without a file extension. Besides this, a line break is added before the content of the partial file. Lastly, you cannot nest `\include{...}` statements. When you try to include a file that also includes a file, a compiler error will be thrown. `\input{...}` on the other hand is capable of such nested imports. In this script, we will present the `\input{...}` command only.

```

\documentclass{article}
\usepackage[english]{babel}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\title{A Title}
\begin{document}
  \maketitle
  \tableofcontents
  \input{section-1.tex}
  \input{section-2.tex}
\end{document}

```

(a) L^AT_EX-Code

A Title

July 23, 2021

Contents

1 This is Section 1
 Here is a paragraph with content for section 1.

2 This is Section 2
 Here is a paragraph with content for section 2.

(b) Result

Figure 4.1: Typical structure in a main file L^AT_EX

Chapter 5

Special characters

5.1 Spaces

The special character used most frequently is a simple space between words. \LaTeX creates it whenever the source code contains spaces or single newlines (by hitting the enter key once) between other characters. The word space is not the only one, though—there are a few more types of spaces. Figure 5.1 illustrates how they are used.

Thin spaces are used quite often, e.g., in abbreviations and between numbers and units: 10\,s. Normal-width non-breaking spaces can, i.e., help to keep a title and name in one line around a potential line break: Dr.\~Fooboar.

(a) \LaTeX -Code

Thin spaces are used quite often, e.g., in abbreviations and between numbers and units: 10 s. Normal-width non-breaking spaces can, i. e., help to keep a title and name in one line around a potential line break: Dr. Fooboar.

(b) Result

Figure 5.1: Different spaces in \LaTeX

English Spacing In documents written in English, \LaTeX uses traditional English Spacing by default. That is, double spaces after each sentence. We can prevent this by using the command `\frenchspacing` above the first paragraph. With `\nonfrenchspacing`, we can reset the behavior. When using English Spacing, \LaTeX tries to recognize abbreviations and to use normal spaces after those. We should still check this and—where necessary—enforce word spaces (`._`) or sentence spaces (`\@.`).

5.2 Hyphens and dashes

There are different kinds of horizontal lines being used as punctuation, most frequently the hyphen (-), the en dash (–) and the em dash (—).¹ These three characters are created in L^AT_EX by different numbers of consecutive hyphens in the source code, as shown in table 5.1.

Code	Character	Usage
-	Hyphen	a small-sized stroke
--	En dash	a range mark (8–10 am) or an interruptor at sentence level – the latter surrounded by spaces and used in most European languages (including British English)
---	Em dash	an interruptor at sentence level — mostly used in American English and without or with thin spaces

Table 5.1: Hyphens and dashes in L^AT_EX

The hyphen is obviously also used for hyphenation, but for this purpose, we should not insert it explicitly in our source code. In most places, L^AT_EX does the hyphenation automatically if we are using the correct **babel** configuration.² In case something goes wrong, we can intervene using the codes from table 5.2:³

Code	Explanation	English example	German example
\-	Exclusive hyphenation: The word may only be hyphenated at this position (can also be used multiple times within one word—all of those positions are then allowed).		Vers- endung, Ver- sendung
"-	Additional hyphenation: The word may be hyphenated here, as well as at the default positions.		Mecklen- burg-Vorpommern
-	Exclusive hyphen: Is only used for compound words and prevents the automatic hyphenation for the rest of the word (which is typographically desirable).	stand- by	Hals-Nasen-Ohren- Arzt
"=	Non-exclusive hyphen: Is used for long compound words when the text wrapping would not work without additional automatic hyphenation.		Sonnen- auf- und -untergang
""	Cut-off point without a hyphen: Allows something like a URL to wrap without inserting a potentially misleading hyphen. Also useful in combination with parentheses.	module (sub-)	Prüfung (Teil-)
"~	Non-wrapping hyphen: Keeps the hyphen together with the following word, very useful for suspended hyphens when parts of compound words are omitted.	high-quality and -priced products	von Satzanfang bis -ende

¹They are named after the letters N and M, that occupy roughly the same space. The letter M is also about as wide as it is tall, and the amount is called a *Geviert* in German, hence the German terms *Halbgeviertstrich* and *Geviertstrich* for the two dashes.

²c. f. section 3.1.4.

³Those also require the **babel** package.

Code	Explanation	English example	German example
------	-------------	-----------------	----------------

Table 5.2: Exceptions for hyphenation

5.3 Quotation marks

Quotation marks can generally be created using the codes from table 5.3. The decisive factor is the appearance, not the semantics, which is why the French Guillemets are used the wrong way around in German (the *french left quotation mark* on the right and vice versa).

Language	First level		Second level	
	Code	Result	Code	Result
English (A. E.)	<code>“...”</code>	“...”	<code>‘...’</code>	‘...’
English (B. E.)	<code>‘...’</code>	‘...’	<code>“...”</code>	“...”
German	<code>„...“</code>	„...“	<code>‚...‘</code>	‚...‘
German (alternatively)	<code>»...«</code>	»...«	<code>>...<</code>	>...<

Table 5.3: Quotation marks

We can get more flexibility using the `csquote` package, which provides the `“<quote>”` command. It chooses the correct quotation marks depending on the language being used; also, nested `enquotes` automatically switch back and forth between first and second level. When enabling the `autostyle=true` option on package import, `\foreignquote{<language>}{<quote>}` selects varying quotation marks for each quotation.

5.4 Diacritics

If we are able to insert letters with diacritics via our keyboard — e. g., the German Umlauts or common accents — we can do this within our source code, as well: The characters will remain intact in the output. If not, we can also create the diacritics via escape codes. Table 5.4 shows just a few examples — the letters can of course be swapped out, but there is still a huge amount of different diacritics.

<code>\‘o</code>	–	ò	<code>\cc</code>	–	ç	<code>\du</code>	–	ù
<code>\’{o}</code>	–	ó	<code>\k{a}</code>	–	ä	<code>\r{a}</code>	–	â
<code>\{o}</code>	–	ô	<code>\l{}</code>	–	ł	<code>\u{o}</code>	–	ö
<code>\" {o}</code>	–	ö	<code>\={o}</code>	–	ō	<code>\v{s}</code>	–	š
<code>\H{o}</code>	–	õ	<code>\b{o}</code>	–	ö	<code>\t{oo}</code>	–	öö
<code>\ {o}</code>	–	õ	<code>\. {o}</code>	–	ó	<code>\o</code>	–	ø

Table 5.4: Diacritics

5.5 More special characters

Some special characters, like the percent sign, are reserved as part of the \LaTeX syntax and cannot be used as normal characters. These and many, many more can be created by their own commands. Please note that some of them only work in maths environments (c.f. chapter 8), others might require additional packages.

Sign	Code	Remarks
¿/¡	?`/!`	
ˆ	\textasciicircum	
˜	\textasciitilde	
*	\textasteriskcentered	
\	\textbackslash	
©	\textcopyright	
†	\textdagger	
...	\textellipsis	
</>	\textless/\textgreater	
‰	\textperthousand	
§	\textsection	
δ, π, Σ	\delta, \pi, \Sigma, ...	only in maths environments
‡	\texttreshlig	requires the <code>tipa</code> package
♪	\textmusicalnote	requires the <code>textcomp</code> package

Table 5.5: Some special characters

Whenever you need a certain symbol and don't know its name, *Detexify*⁴ comes to the rescue: You can draw the symbol and get all necessary information. From cuneiforms to technical symbols, there is absolutely *everything*, as you can see scrolling through the *Comprehensive \LaTeX Symbol List*.⁵

⁴<http://detexify.kirelabs.org/classify.html>

⁵<http://tug.ctan.org/info/symbols/comprehensive/symbols-a4.pdf>

Chapter 6

Text markup

Text highlighting

Text markup can be done in two ways: semantically or visually. We recommend that you use semantic markup whenever possible. In contrast to visual markup, it only states *why* something is special and entrusts to L^AT_EX *how* it is going to look. The simplest semantic markup, that was also used in the previous sentence, is an emphasis: `\emph{...}`. By default, this command sets text in italics. When it is nested, the second level of emphasis is set straight again. This kind of formatting is only perceived when reading the text and does not attract attention beforehand, as colored or bold text does (which is more appropriate for higher-level structuring purposes).

Some types of visual markup are listed in table 6.1, but you should use them very carefully. In principle, they can also be nested, however, for some combinations, the corresponding fonts will be missing. Many other programs try to distort existing fonts to imitate the missing one. As this does not look particularly good, L^AT_EX will not do it. So do not be surprised when your carefully nested selection of four different markups is just ignored and does not do anything at all.

Markup	Command	Rendering
bold	<code>\textbf{bold face}</code>	bold face
italics	<code>\textit{italics}</code>	<i>italics</i>
small caps	<code>\textsc{small caps}</code>	SMALL CAPS
monospaced	<code>\texttt{typewriter text}</code>	typewriter text
slanted	<code>\textsl{slanted}</code>	<i>slanted</i> (please, don't!)
underlined	<code>\underline{underlined}</code>	<u>underlined</u>
subscript	<code>\textsubscript{subscript}</code>	x _{subscript}
superscript	<code>superscript</code>	x ^{superscript}

Table 6.1: Visual markup commands

Usually, you should not need these commands too often, as they appear by themselves when you are using semantic markup. For instance, the `hyperref` package provides the `\url{...}` command. This command does not only use a

mono-spaced font for URLs, it also makes them clickable and, if necessary, wraps them without adding hyphens.

The same applies for different font sizes. You can specify the body text font size with an option at the document class:

```
\documentclass[9pt]{article}
```

Building upon this, L^AT_EX generates different font sizes that can be called via the commands in table 6.2. It is, however, best to restrict those to title pages and similar things. For the rest, you can trust the default settings and avoid the visual clutter.

Command	Rendering
<code>{\tiny tiny}</code>	tiny
<code>{\footnotesize footnote size}</code>	footnote size
<code>{\small small}</code>	small
<code>{\normalsize normal}</code>	normal
<code>{\large large}</code>	large
<code>{\Large larger}</code>	larger
<code>{\LARGE largest}</code>	largest
<code>{\huge largest of all}</code>	largest of all
<code>{\Huge megalomania}</code>	megalomania

Table 6.2: Font size commands

Paragraph alignment

By default, \LaTeX sets continuous text in full justification. However, we can also switch to ragged alignment by using the commands `\raggedright`, `\raggedleft`, and `\centering`. These commands influence the environment that they are used in, e.g., the `document` environment. Correspondingly, the text within the whole document is affected. Alternatively, we can use dedicated environments in order to influence the formatting of certain paragraphs (fig. 6.1).

```
\begin{flushleft} This text is situated
↪ inside a \texttt{flushleft}
environment. Lorem ipsum dolor sit amet,
↪ consectetur adipisicing elit, sed do
eiusmod tempor incididunt ut labore et dolore
↪ magna aliqua. \end{flushleft}

\begin{flushright} This text is situated
↪ inside a \texttt{flushright}
environment. Lorem ipsum dolor sit amet,
↪ consectetur adipisicing elit, sed do
eiusmod tempor incididunt ut labore et dolore
↪ magna aliqua. \end{flushright}

\begin{center} This text is situated inside a
↪ \texttt{center} environment.
Lorem ipsum dolor sit amet, consectetur
↪ adipisicing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua.
↪ \end{center}
```

(a) \LaTeX -Code

This text is situated inside a `flushleft` environment. Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

This text is situated inside a `flushright` environment. Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

This text is situated inside a `center` environment. Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

(b) Result

Figure 6.1: Ragged alignment

Indentation and spacing

Usually, we illustrate a new paragraph by indenting the first line of it (`\parindent`). Alternatively, paragraph spacing, i.e., vertical space between paragraphs, can be used (`\parskip`). For both variants, there are adjustable parameters:

```
\setlength{\parindent}{0pt}
\setlength{\parskip}{1em}
  plus .5em % admissible stretch
  minus .5em % admissible shrink
}
```

We can use `\noindent` to turn off the indentation for only one paragraph. For the first paragraph after a heading, there is usually no indentation.

Chapter 7

Lists

Inherently, L^AT_EX supports three types of lists: unordered, ordered, and description lists. For each of these types, there are dedicated environments whose structure is very similar.

```
\begin{itemize}
  \item lasagna noodles
  \item crushed tomatoes
  \item herbs \begin{enumerate}
    \item oregano
    \item basil
    \item rosemary
  \end{enumerate}
\end{itemize}
```

(a) L^AT_EX-Code

-
- lasagna noodles
 - crushed tomatoes
 - herbs
 - 1. oregano
 - 2. basil
 - 3. rosemary
-

(b) Result

Figure 7.1: Exemplary unordered list

In fig. 7.1a, different list items (`\item`) are framed by an `itemize` environment. Within a list item, we can open a new list environment to create a nested list. If we replace `itemize` by `enumerate`, the list becomes ordered without changing the source code any further. To make use of the less common description lists, we have to specify the environment `description`. Moreover, the items need to be extended by the term to be defined, put in brackets (fig. 7.2).

The exuberant spacing between list items can be reduced by utilizing the `paralist`¹ package. To do so, we simply adjust the environment name: `compactitem` replaces `itemize`, `compactenum` replaces `enumerate`, and `compactdesc` replaces `description`. If the list ought to appear within a text body, `paralist` provides the extra environments `inparaenum` and `inparaitem`.

We can customize bullet and list numbering styles via the `enumitem`² pack-

¹<https://www.ctan.org/pkg/paralist>

²<https://www.ctan.org/pkg/enumitem>

```

\begin{description}
  \item [Béchamel sauce] Béchamelsauce, also
    ↪ known as white sauce, % ...
  \item [Lasagne] Lasagne (singular lasagna)
    ↪ are a type of wide, % ...
\end{description}

\end{document}

```

(a) L^AT_EX-Code

Béchamel sauce Béchamel sauce, also known as white sauce, is made from a white roux (butter and flour) and milk. It has been considered, since the seventeenth century, one of the mother sauces of French cuisine. (cf. https://en.wikipedia.org/wiki/B%C3%A9chamel_sauce)

Lasagne Lasagne (singular lasagna) are a type of wide, flat pasta, possibly one of the oldest types of pasta. Lasagne, or the singular lasagna, commonly refers to an Italian cuisine dish made with stacked layers of this flat pasta alternated with sauces and ingredients such as meats, tomato sauce and other vegetables, cheese (which may include ricotta and parmesan), and seasonings and spices such as garlic, oregano and basil. (cf. <https://en.wikipedia.org/wiki/Lasagne>)

(b) Result

Figure 7.2: Exemplary description list

age. `\begin{enumerate}[label=\roman*]` produces a list with Roman numerals. The parameter `[label=\alph*]` inserts alphabetical numbering.

Chapter 8

Mathematical formulas

Mathematical formulas are always set in *math mode*, which, within a paragraph (referred to as *inline*), can be turned on or off with a dollar sign. There is also a *block* environment (cf. fig. 8.1). Important packages for mathematical features are the `amsmath`, `amsthm`, and `amssymb` packages of the American Mathematical Society, as well as `mathtools`. As with many other environments, adding an asterisk turns off the numbering.

The imaginary unit i is defined as $i = \sqrt{-1}$.

```
\begin{equation}
2 \sqrt{\frac{\pi^2}{3}} \cdot c_2
\end{equation}
```

(a) L^AT_EX-Code

The imaginary unit i is defined as $i = \sqrt{-1}$.

$$2\sqrt{\frac{\pi^2}{3}} \cdot c_2 \quad (8.1)$$

(b) Result

Figure 8.1: Exemplary math environments

8.1 A few examples

Source code	Result
<code>\sqrt{16}</code>	$\sqrt{16}$
<code>\frac{3}{4}</code>	$\frac{3}{4}$
<code>e^{\pi}</code>	e^{π}
<code>\sum_{i=1}^n x^2</code>	$\sum_{i=1}^n x^2$
<code>12 \leq 4 x^2 + 13</code>	$12 \leq 4x^2 + 13$
<code>n \choose k</code>	$\binom{n}{k}$

Table 8.1: Frequently used commands (square root, fraction, power, sum, in-equation, binomial coefficient). By `^{\dots}` and `_{\dots}`, the content is set in super- or subscript.

Source code	Result
<code>(x), [x], \lbrace x \rbrace, \lvert x \rvert</code>	$(x), [x], \{x\}, x $
<code>\exists, \forall, \in, \notin, \infty</code>	$\exists, \forall, \in, \notin, \infty$
<code>\alpha, \beta, \Gamma, \Delta, \varepsilon, \pi</code>	$\alpha, \beta, \Gamma, \Delta, \varepsilon, \pi$
<code>\rightarrow, \leftarrow, \Rightarrow, \Leftarrow, \Leftrightarrow</code>	$\rightarrow, \leftarrow, \Rightarrow, \Leftarrow, \Leftrightarrow$
<code>(A \cup B) \cap C</code>	$(A \cup B) \cap C$
<code>(A \vee B) \wedge C</code>	$(A \vee B) \wedge C$
<code>(A \cdot B) \times C</code>	$(A \cdot B) \times C$

Table 8.2: Brackets, quantifiers, greek letters, arrows, operators

8.2 Growing brackets

Especially in combination with fractions, brackets should grow according to their content. This can be achieved by pre-pending each bracket (`(`, `)`, `[`, `]`, `\lbrace` and `\rbrace`) with a position marker (`\left` or `\right`).

```
\begin{equation*}
4 \cdot \left( \frac{1}{2} + \frac{3}{12} \cdot \right.
\rightarrow \left. \left( 2 + \frac{1}{86} \cdot \right.
\rightarrow \left. \left( \frac{1}{2} + 24 \right.
\rightarrow \left. \left. \right. \right) \right) \right)
\end{equation*}
```

$$4 \cdot \left(\frac{1}{2} + \frac{3}{12 \cdot \left(2 + \frac{1}{86 \cdot \left(\frac{1}{2} + 24 \right)} \right)} \right)$$

(a) L^AT_EX-Code

(b) Result

Figure 8.2: Example for growing brackets

8.3 Lower and upper bounds

The `\limits` command renders lower and upper bounds of integrals above and below the integral sign. Sums, products, and limits do this automatically (cf. fig. 8.3). For inline formulas, `\limits` are of bounded suitability.

```

\begin{equation*}
  \sum_{i=1}^{n^2}(x+2)
\end{equation*}
\begin{equation*}
  \prod_{j=1}^{100}(3 \cdot x)
\end{equation*}
\begin{equation*}
  \lim_{x \rightarrow \infty}(14x^3 - 12)
\end{equation*}
\begin{equation*}
  \int\limits_{-12}^4(14x^3 - 12)
\end{equation*}

```

$$\sum_{i=1}^{n^2}(x+2)$$

$$\prod_{j=1}^{100}(3 \cdot x)$$

$$\lim_{x \rightarrow \infty}(14x^3 - 12)$$

$$\int\limits_{-12}^4(14x^3 - 12)$$

(a) L^AT_EX-Code

(b) Result

Figure 8.3: Lower and upper bounds of sums, products, limits and integrals

8.4 Aligning equations

The `align` environment allows to align multiple equations horizontally, e. g., at the = sign (fig. 8.4). As in tables, the `&` sign is used to specify anchorage points. Line breaks are denoted by two backslashes.

```

\begin{align*}
  13 \cdot (4a - 3)^2 &= 13 \cdot 16a^2 - 24a + 9 \\
  &= 208a^2 - 312a + 117
\end{align*}

```

$$13 \cdot (4a - 3)^2 = 13 \cdot 16a^2 - 24a + 9$$

$$= 208a^2 - 312a + 117$$

(a) L^AT_EX-Code

(b) Result

Figure 8.4: Equations aligned at equals signs

8.5 Text in math mode

Sometimes sets have to be defined in terms of textual descriptions or longer function names. The L^AT_EX math mode assumes that letters are variables rather

than text, which creates problems when they are indeed supposed to be entire words. For this case, there is the `\text{}` command (c.f. fig. 8.5).

```

\begin{align*}
\left\{x \mid \text{tiefe}(x) \geq 20\right. \\
\rightarrow \left. \right\} \\
\left\{x \mid \text{tiefe}(x) \geq 20\right. \\
\rightarrow \left. \right\}
\end{align*}

```

$$\{x \mid \text{tiefe}(x) \geq 20\}$$

$$\{x \mid \text{tiefe}(x) \geq 20\}$$

(a) L^AT_EX-Code

(b) Result

Figure 8.5: Problems arising from intensional set notation and their solution

Chapter 9

Graphics

Since in \LaTeX we work with plain text, we cannot simply embed graphics into our text as we may be used to from other word processing programs. Instead, we reference external image files by a command. The figure is then embedded and positioned at compile time.

9.1 Inserting graphics

In order to be able to reference graphics, the package `graphicx` has to be included. For inserting a figure, we can use the following commands:

```
\begin{figure}
    \includegraphics{<file path>}
    \caption[<short title>]{<caption>}
\end{figure}
```

The command `includegraphics` can be used to change the image size. The desired height and width of the figure can be indicated separately, as illustrated by the following example:

```
\includegraphics[width=0.5\textwidth,height=5cm]{<file path>}
```

With the `caption` command you can also add a caption to the image. Note that you can provide an alternative caption in square brackets, if you want.

9.2 Positioning

One interesting aspect of the what-you-get-is-what-you-mean paradigm is the way how graphics can be positioned. The compiler creates multiple layouts and evaluates them. By default, graphics are placed at the potentially optimal position that is calculated by the compiler. By moving them around, typographic blemishes like widows and orphans¹ can be avoided.

As a consequence, graphics are not necessarily placed between the two text blocks that we specify, but at another position. In order to reference a picture,

¹The first (last) line of a paragraph appears alone as last (first) line on the previous (next) page, cf. https://en.wikipedia.org/wiki/Widows_and_orphans.

that possibly is placed on another page, we can use labels, which are covered in section 11.2. On top of that, we can limit the positioning of our image more or less rigorously by adding optional parameters to the **figure** environment. The available positioning shortcuts can be found in table 9.1.

Shortcut	Position
h	here, if possible
t	on top of the page (<i>top</i>)
b	at the bottom of the page (<i>bottom</i>)
p	on its own page (<i>page</i>)
H	definitely here (requires package float)

Table 9.1: Shortcuts for positioning graphics

```
\begin{figure}[<position shortcut>]
  \centering
  \includegraphics{<file path>}
\end{figure}
```

Besides the vertical positioning, also the horizontal orientation may be of importance. By default, graphics are left-justified. The command **\centering** centers all following objects in the current environment. If we want the centering to affect only one object, we can alternatively wrap it with **\begin{center}** and **\end{center}**.

Chapter 10

Tables

There are two fundamental environments for tables. The first one, called `table`, is responsible for integrating the table as a whole into the document. Positioning is done the same way as it is with graphics. The `\caption` command is the same, as well.

At the table *content*, the similarities end: While graphics are embedded from external files and not interpreted by L^AT_EX, the inner structure of tables has to be encoded explicitly. This is done with the `tabular` environment that expects a column definition as an obligatory parameter. The column definition consists of one letter per column specifying its text alignment: `l` for left-justified, `r` for right-justified, `c` for centered.

Within the `tabular` environment, the actual table content follows. Table rows are separated by `\\`, just like line breaks, and cells by an `&`.

For typographically pleasing tables, we recommend the `booktabs` package.¹ This package brings along the commands `\toprule`, `\midrule` and `\bottomrule` that draw appropriate horizontal lines above, within and below the table, respectively.

Vertical lines can be inserted as a vertical bar character (`|`) in the column definition, but this is *not* recommended.² If you want to remove the additional whitespace that surrounds the columns by default (e.g., to the left of the first column and to the right of the last), you can add the string `@{}` at the corresponding places within the column definition.

A complete table can then look like the one shown in fig. 10.1.

Excess length For tables exceeding one page or requiring line breaks within individual cells, the `longtable` package can be used additionally (also supported by `booktabs`). The `longtable` environment combines the `table` and `tabular`

¹All commands previously mentioned also work without this package, but the result will look far less professional.

²The whitespace within the table separates the columns clearly enough, additional lines just make for visual clutter.

```

\begin{table}[h]
  \centering
  \begin{tabular}{llr}
    \toprule
    Language & Author & Year \\
    \midrule
    C++ & Bjarne Stroustrup & 1985 \\
    Java & James Gosling & 1998 \\
    Python & Guido van Rossum & 1991 \\
    \bottomrule
  \end{tabular}
  \caption{Well-known programming
    ↪ languages}
\end{table}

```

(a) L^AT_EX-Code

Language	Author	Year
C++	Bjarne Stroustrup	1985
Java	James Gosling	1998
Python	Guido van Rossum	1991

Well-known programming languages

(b) Result

Figure 10.1: A complete table

environments. With it, you get the following basic structure:

```

\begin{longtable}
  % content
  \caption{<caption>}
\end{longtable}

```

Excess width If you want to present a very wide table instead, it is preferable on pages in portrait orientation to turn the table by 90°. This can be done with the help of the `rotating` package. The only difference between this and a normal table is that the `table` environment gets replaced by a `sidewaystable` environment. Positioning and the `tabular` stay unaffected.

More possibilities Of course, L^AT_EX offers lots of additional features for sophisticated tables, e. g., row- or column-spanning cells. For something like this — or just to save some typing — we recommend the *Tables Generator*,³ that allows you to click your tables together in WYSIWYG style and provides you with L^AT_EX code ready for copying into your document.

³<https://tablesgenerator.com/>

Chapter 11

References and footnotes

11.1 Footnotes

Whenever we want to include footnotes into our \LaTeX document, we can use the command `\footnote<text>`. At the position where we use the command, the correct number will be inserted automatically, and the text within the curly braces will appear in the footer. In combination with the package `hyperref`, URLs within footnotes become clickable.¹ We can see examples for that in the whole document.

The package `footmisc` provides us with additional options for how to display footnotes. They can be passed as optional parameters to the command `\usepackage`:

- `\usepackage[perpage]{footmisc}` ensures that the count of footnotes begins at 1 for each new page.
- `\usepackage[para]{footmisc}` lets the footnotes within the footer appear as continuous text (i. e., the footnotes can also appear next to each other).
- `\usepackage[symbol]{footmisc}` causes a numbering with symbols (e. g., †, ‡) instead of numbers.

11.2 References

If we want to make references, like “... , which you can see in figure 21, ...”, \LaTeX by default provides us with the command `\ref{<label>}`. No more adapting of the numbering for graphics, tables, etc. needed! The command expects a unique label as argument, that needs to be assigned to the referenced element. After that, wherever we call the command, the number of our referenced object appears in the text.

A smarter package for references is `cleveref`.² It provides us with the command `\cref{<label>}`, which can also handle multiple labels separated by commas. This automatically generates elegant references like “sections 1 to 3, and

¹if we use the command `\url...`

²with only one “r”!

```
\begin{figure}[H]
  \includegraphics[width=% ...
  \caption{Our mascot Foobar}
  \label{img:foobar}
\end{figure}
```

As you can see in Figure `\ref{img:foobar}`,
↪ Foobar has already undertaken a
↪ deep-dive into the `\LaTeX{}` ecosystem.

(a) \LaTeX -Code



Abbildung 1: Our mascot Foobar

As you can see in Figure 1, Foobar has already undertaken a deep-dive into the \LaTeX ecosystem.

(b) Result

Figure 11.1: Example for a reference

5.”³ Furthermore, `\ref{<label>}` automatically inserts a suited abbreviation, e.g., “fig.” for figures.

We can reference graphics, tables, sections, chapters, source code listings, and equations. Many packages use the label in order to find out the object type of the referenced element. For this reason, it is common to insert a prefix before each label (table 11.1).

Prefix	Object type	Prefix	Object type
fig:	figures	tbl:	tables
sec:	sections	subsec:	subsections
ch:	chapters	itm:	items
eq:	equations	lst:	source code listings

Table 11.1: Prefixes for labels

Note that if we use `\cref{<label>}` — for some document classes — the generated passages only appear in the desired language (e.g., German) when the language is specified already within the document class command:

```
\documentclass[ngerman]{article}
```

Except for sections, captions⁴ *always need to be specified and positioned before the label*. Otherwise, they cannot be referenced later on in the text. Labels for sections are inserted directly after the command:

```
\section{Comments}\label{sec:hints}
```

³for the source code `\cref{sec:section1,sec:section2, sec:section3,sec:section5}`

⁴`\caption{...}`

Chapter 12

Source code listings

There are many ways in L^AT_EX to display source code. We have come to appreciate the package `minted`, which causes some additional installation overhead, but generates very appealing source code renderings.

However, especially on macOS, the installation of `minted` has caused a lot of headache in the early days of this workshop. That's why we will also have a look at an alternative called `listings`.

A note on colors. Both of these packages require you to define colors. We can recommend using the `xcolor` package. There are very helpful resources¹ available online such that we will not go into details here.

Source code listings within this script

This script uses the package `listings` as default way for displaying source code listings, since, as mentioned above, it causes less headache, especially on macOS. As we introduce `minted` as an alternative package, naturally, this script can also display source code listings rendered by `minted`.

To switch between the packages, you can change the listings mode to `minted` like so: First of all, follow the respective installation instructions. Afterwards, create a new file called `listings-mode.tex` in the root directory of this project, and insert the following line into the newly created file:

```
\newcommand\listingsmode{minted}
```

After compiling the script, you should see differently rendered source code listings. If you want to return to `listings` as package for rendering your source code listings, just replace `minted` with `default` in the `listings-mode.tex` file. Source code listings within this script are then displayed using the `listings` package again.

¹If you would like to specify your own colors, these pages might help you: https://www.overleaf.com/learn/latex/Using_colours_in_LaTeX for a list of pre-defined colors in the `xcolor` package, <https://mmoredo.github.io/latex-color-converter/> for defining your own colors

12.1 Using minted

12.1.1 Installation

Using `minted` requires a working installation of the programming language Python 3 (henceforth referred to as Python). On some operating systems, Python comes pre-installed, in which case entering the command `python --version` or `python3 --version` in a terminal of your choice² should print out the installed Python version.

If Python is yet to be installed, then you can find the installation files on the project website³. There are extensive articles that cover all relevant steps to install Python on Windows,⁴ Linux,⁵ or macOS.⁶

After a successful installation, you should be able to execute the aforementioned command in a terminal, confirm by pressing Enter, and see approximately the following result:

```
$ python --version
Python 3.8.5
```

If the version number is equal to the one stated here, or higher, then everything should be set up correctly. Next, enter the command `pip install Pygments`⁷ in the same terminal window to install the Pygments package for Python. Once the installation is complete, you are ready to include the L^AT_EX package `minted` into your documents by adding `\usepackage{minted}` to your preamble.

12.1.2 Changing the compiler command

There is one last adjustment needed before you can actually compile your documents—we will have to adjust the compile command. Out of the box, your editor will probably execute the following command after you clicked the green compile arrow:

```
$ pdflatex main-exercises.tex
```

The exact command can be found and configured in T_EXstudio under Options → Configure T_EXstudio → Commands. It is stated next to the PdfL^AT_EX label. The file that is to be compiled will replace the placeholder `%.tex` upon compilation. Typically, there are two additional options configured. You can recognize them by the hyphen before their names (`-synctex=1` and `-interaction=nonstopmode`). These options are called flags and they configure the program `pdflatex`. We will have to add another flag. Place the string `-shell-escape` before the file placeholder (`%.tex`):

²Opening a terminal on Windows: `Win + R` → Type “cmd” → Enter

³Available at <https://www.python.org/downloads/>.

⁴Pawandeep, How to install Python in Windows?. Tutorialspoint. March 10, 2021. Available at <https://www.tutorialspoint.com/how-to-install-python-in-windows>. Windows users will have to adjust the system path during the installation process. Forgetting this step has been the number one installation problem in past workshops.

⁵<https://docs.python-guide.org/starting/install3/linux/>

⁶<https://docs.python-guide.org/starting/install3/osx/>

⁷On some operating systems, you might have to use the command `pip3 install Pygments`

```
$ pdflatex -synctex=1 -interaction=nonstopmode -shell-escape %.tex
```

After a click on “Okay” the configuration is finished. Other editors usually provide similar options to configure the compilation command. We recommend you to have a look at the settings or to use of a search engine to figure it out.

An important note on the shell-escape flag. `minted`’s syntax highlighting is done by a Python package, which adds the necessity to communicate between the compiler and the Python runtime. The shell-escape flag adds this communication path. When enabled, \LaTeX can execute any command in a terminal, which can be very useful. Nonetheless, it would also be possible to execute malign code on your computer via \LaTeX , especially when you are compiling unknown documents from the Internet. Therefore, do not compile downloaded documents with the shell-escape flag if you do not trust the authors and did not check the packages and commands they include.

12.1.3 Defining listings

We are finally ready to marvel at the aesthetic quality of the listings `minted` produces. You can define listings using a dedicated environment:

```
\begin{minted}{haskell}
  [x^2 | x <- [1..200], even x]
\end{minted}
```

(a) \LaTeX -Code

```
[x^2 | x <- [1..200], even x]
```

(b) Result

Figure 12.1: Exemplary source code listing

```
% Shorthand
Despite it is a shorthand,
↪ \mint{html}|<h2>LaTeX im Studium</h2>|
↪ will be rendered on its own line.

% Inline variant
The method call
↪ \mintinline{java}{o.doSomething();} on
↪ the other hand will appear inline.
```

(a) \LaTeX -Code

```
Despite it is a shorthand,
<h2>LaTeX im Studium</h2>
will be rendered on its own line.
The method call o.doSomething(); on the other hand
will appear inline.
```

(b) Result

Figure 12.2: Shorthand and inline listing

There is also a shorthand and an inline variant of the command (cf. fig. 12.2). To avoid redundancy, it may be practical to import source code directly from the source file. To accomplish this, we only have to pass the programming language and the file path to the `\inputminted` command (cf. fig. 12.3).

```
\inputminted{java}
{listings/source-code-listings/Test.java}
```

(a) L^AT_EX-Code

```
public class Test {
    public static void main(/*...*/) {
        System.out.println("Welcome, " +
            "fellow LaTeX learners!");
    }
}
```

(b) Result

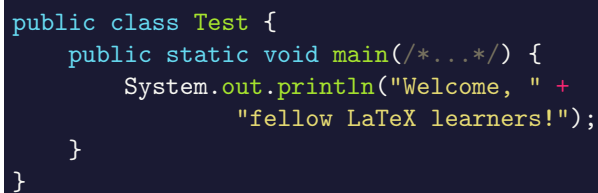
Figure 12.3: Including from a separate file

12.1.4 Configuring minted

Optional parameters allow us to add line numbers, line breaks, and colors. Moreover, there are numerous themes available (fig. 12.4). The introduction on Overleaf and the package documentation⁸ give an extensive overview.

```
\usemintedstyle{monokai}
\definecolor{bg}{rgb}{0.1,0.1,0.2}
\inputminted[
    breaklines=true,
    bgcolor=bg
]{java}{listings/source-code-listings/Test.java}
```

(a) L^AT_EX-Code



(b) Result

Figure 12.4: Themes and further options

⁸Available at https://www.overleaf.com/learn/latex/Code_Highlighting_with_minted and <https://ctan.kako-dev.de/macros/latex/contrib/minted/minted.pdf>, respectively.

12.2 Using listings

12.2.1 Installation

`listings` is a \LaTeX package of itself and does not require any additional installation as long as the `xcolor` package is installed as well. We can therefore jump straight into typesetting source code.

12.2.2 Defining listings

Similar to `minted`, you can define listings using a dedicated environment (cf. fig. 12.5) and you can use another command to import source code directly from external files (cf. fig. 12.6).

```
\begin{lstlisting}[language=Haskell]
  [x^2 | x <- [1..200], even x]
\end{lstlisting}
```

(a) \LaTeX -Code

```
[x^2 | x <- [1..200], even x]
```

(b) Result

Figure 12.5: Exemplary source code listing

```
\lstinputlisting[language=Java]
{listings/source-code-listings/Test.java}
```

(a) \LaTeX -Code

```
public class Test {
    public static void main(/*...*/) {
        System.out.println("Welcome, " +
            "fellow_LaTeX_learners!");
    }
}
```

(b) Result

Figure 12.6: Including from a separate file

Both commands need a language specification⁹ as an optional parameter to apply syntax highlighting.

12.2.3 Configuring listings

The `listings` package can also be tweaked in an almost infinite number of ways. Numerous optional parameters give you a lot of freedom.

Once again, Overleaf and the package documentation¹⁰ come in handy.

⁹See this guide on Overleaf for a list of all languages: https://www.overleaf.com/learn/latex/Code_listing#Reference_guide

¹⁰Available at https://www.overleaf.com/learn/latex/Code_listing#Code_styles_and_colours and <https://ctan.org/pkg/listings>, respectively.

```

\definecolor{codegray}{rgb}{0.5,0.5,0.5}
\definecolor{codegreen}{rgb}{0.65,0.89,0.18}
\definecolor{codeyellow}{rgb}{0.9,0.86,0.45}
\definecolor{codeblue}{rgb}{0.36,0.76,0.85}
\definecolor{backcolour}{rgb}{0.1,0.1,0.2}

\lstinputlisting[
  language=Java,
  backgroundcolor=\color{backcolour},
  keywordstyle=\color{codeblue},
  identifierstyle=\color{codegreen},
  stringstyle=\color{codeyellow},
  basicstyle=\small
    \ttfamily\color{white},
  commentstyle=\color{codegray},
  numberstyle=\tiny\color{codegray},
  numbers=left,
  numbersep=5pt,
  showstringspaces=false,
]{listings/source-code-listings/Test.java}

```

(a) L^AT_EX-Code

```

1 public class Test {
2     public static void main(/*...*/) {
3         System.out.println("Welcome, " +
4             "fellow LaTeX learners!");
5     }
6 }

```

(b) Result

Figure 12.7: Themes and further options

12.2.4 Drawbacks and caveats

As mentioned earlier, `listings` is not our first choice for source code listings. Its renderings are of rather peculiar appearance.

With some amount of configuration, we can overcome the most disturbing default settings. Although `listings` is not shipped with any pre-defined themes, you can define your own and use them throughout your project with the `lstdefinestyle` command.¹¹

The package is also a bit older than its alternative, causing UTF-8 special characters to break. If this happens to you, have a look at the `literate` option of the `listings`¹² commands.

¹¹Have a look at it in the package documentation. For the very impatient, here is a solarized theme for `listings`: <https://github.com/jez/latex-solarized>

¹²The \LaTeX WikiBook offers a good starter for the `literate` option covering the most frequent special characters: https://en.wikibooks.org/wiki/LaTeX/Source_Code_Listings#Encoding_issue

Chapter 13

Reference management

For typesetting our first thesis in L^AT_EX, the last core functionality to learn is citing literature. Our references are gathered in a bibliography file. Once we reference one of its entries from our L^AT_EX document, BibT_EX (a program similar to the standard pdf_lat_ex compiler) can insert automatically generated citations. It will format them in a bibliography style of our choice.

13.1 The bibliography file

Our **bibliography collection** consists of multiple literature entries in a pre-defined format, such that they can be processed by BibT_EX. An exemplary item can be seen in fig. 13.1.

```
@article{turing1990,  
  title={The chemical basis of morphogenesis},  
  author={Turing, Alan Mathison},  
  journal={Bulletin of mathematical biology},  
  volume={52},  
  pages={153--197},  
  year={1990},  
  publisher={Springer}  
}
```

Figure 13.1: Exemplary bibliography entry

The type of the bibliography entry is specified after the opening @ sign (article, book, proceedings, ...). What follows is a list of important attributes like title and author. Whether they are required or not depends on the type of the entry. In any case, we will need the first entry after the opening braces: the BibT_EX key. This is the identifier that we will use to reference the entry in our L^AT_EX document. BibT_EX keys can be chosen freely, but have to be unique. Typically, they will consist of a combination of authors, publication dates, and topics.

Bibliography files can be compiled manually, yet it is more common to use programs like JabRef¹ or Zotero². While JabRef operates directly on your bibliography file, Zotero projects³ can be exported to bibliography files to use them in L^AT_EX documents.

Bibliography entries are provided by many academic search engines, including Google Scholar (cf. fig. 13.2). When using them, make sure that the entries are cohesive across your reference collection and complete with regard to their attributes. A high-quality (although, unfortunately, incomplete) source for BibT_EX entries is the dblp computer science bibliography.⁴

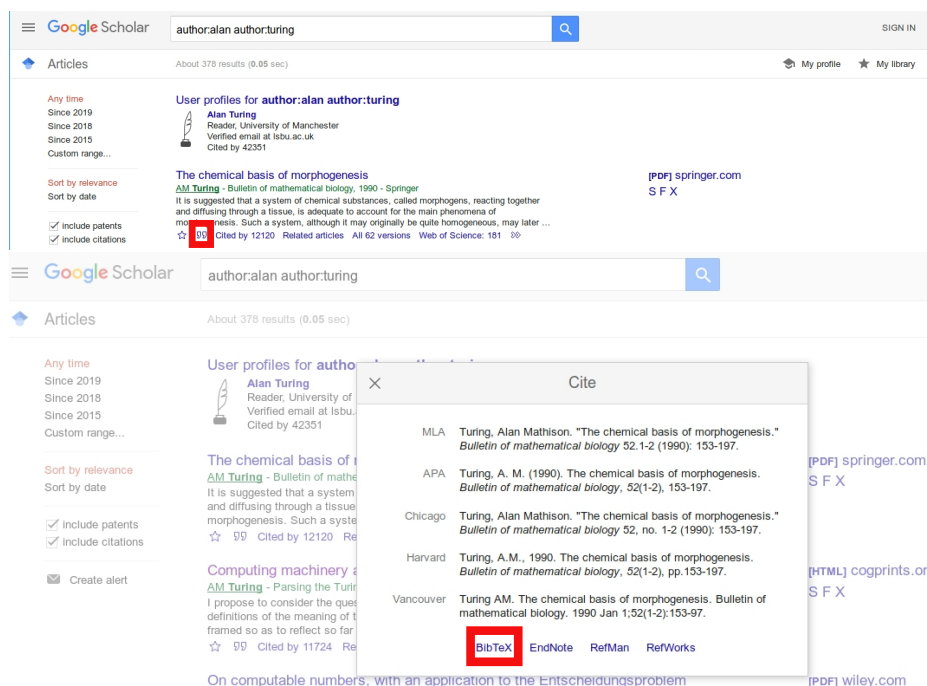


Figure 13.2: Loading BibT_EX entries from Google Scholar

13.2 Citing

BibT_EX extends L^AT_EX by several commands (cf. table 13.1). Make sure to include the `natbib` package for this purpose.

¹Cf. <https://www.jabref.org/>.

²Cf. <https://www.zotero.org/>.

³Vgl. <https://guides.library.iit.edu/c.php?g=720120&p=6296986>.

⁴Available at <https://dblp.org/search>.

Function	Command
Citing authors	<code>\citeauthor{<source>}</code>
Citing sources	<code>\cite{<source>}</code>
Citing pages	<code>\cite[p. 15]{<source>}</code>
Custom citations	<code>\cite[<prefix>][<suffix>]{<source>}</code>
Including the bibliography	<code>\bibliography{<bibliographyfile>}</code>
Setting the bibliography style	<code>\bibliographystyle{<style>}</code>

Table 13.1: Commands for citations

The `<source>` of a citation is always a BibTeXkey. The list of available citation styles⁵ includes `alpha`, `natdin`, and `apa`. The table of references will always appear where the `\bibliography{...}` command was put. The `\cite` command comes with many variants.⁶

<hr/> <p>This text thrives on prominently-placed citations <code>\cite[e.\,g.,]{Frank1957}</code> of much more prominent literary works <code>\cite[cf.][p. 7]{Orwell1957}</code>, of which <code>\cite{Hawking1988}</code> is only one.</p> <pre>\bibliographystyle{natdin} \bibliography{collection}</pre> <hr/>	<hr/> <p>This text thrives on prominently-placed citations (e. g., Frank, 1957) of much more prominent literary works (cf. Orwell, 1957, p. 7), of which Hawking (1988) is only one.</p> <h2>References</h2> <p>[Frank 1957] FRANK, Anne: <i>Das Tagebuch der Anne Frank</i>. Frankfurt/Main : Fischer, 1957</p> <p>[Hawking 1988] HAWKING, Stephen W.: <i>Eine kurze Geschichte der Zeit. Die Suche nach der Urkraft des Universums</i>. 1988</p> <p>[Orwell 1957] ORWELL, George: <i>1984</i>. Konstanz : Diana-Verlag, 1957</p> <hr/>
(a) L ^A T _E X-Code	(b) Result

Figure 13.3: Exemplary citations in the `natdin` style.

⁵Head to Overleaf for a rather complete list: https://www.overleaf.com/learn/latex/Biblatex_citation_styles

⁶cf. <https://www.economics.utoronto.ca/osborne/latex/BIBTEX.HTM>

Chapter 14

Prospects

Obviously, in this script, we were not able to show you the least of what L^AT_EX has to offer. Therefore, in this last section, we gathered some information to help you to go further into depth by yourself.

14.1 Packages

We already have presented a selection of packages. However, there are thousands more of them. In the following sections we have put together some packages for frequently needed features:

Indices can be created automatically with `makeidx`.¹ By using `\index{...}`, one can mark entries for the index. With `\printindex`, an index with references is compiled out of them.

Vector graphics (fig. 14.1a) can be “drawn” directly in the L^AT_EX source code with `TikZ` (recursive acronym for *TikZ ist kein Zeichenprogramm*, in English: *TikZ is not a drawing program*).² Caution: This package is very powerful, but not necessarily beginner-friendly. Before creating vector graphics from scratch, we recommend you to experiment with some of the examples at T_EXample³. Also, for certain use cases, there are special packages that are easier to handle than “raw” `TikZ`:

Parse trees that divide sentences into their grammatical components (fig. 14.1b) can be created with `qtree`.⁴

Proof trees, that are often used in logics (fig. 14.1c), can be drawn with the package `prftree`.⁵

Chemical structural formulas (fig. 14.1d) can, amongst others, be created with `chemfig`.⁶

¹<https://www.ctan.org/pkg/makeidx>

²<https://www.ctan.org/pkg/pgf>

³<https://texample.net/tikz/examples/>

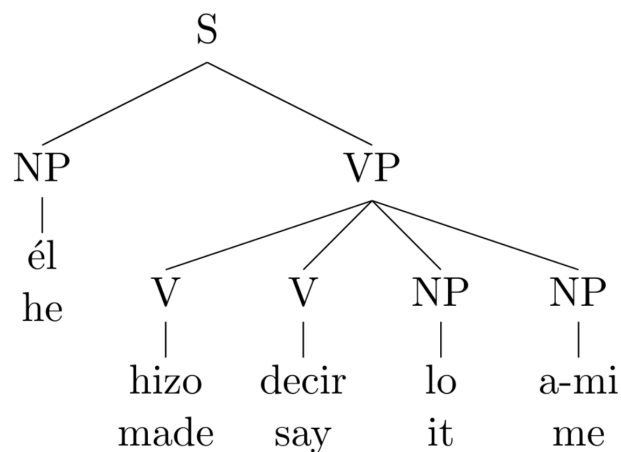
⁴<https://ctan.org/pkg/qtree>

⁵<https://www.ctan.org/pkg/prftree>

⁶<https://www.ctan.org/pkg/chemfig>



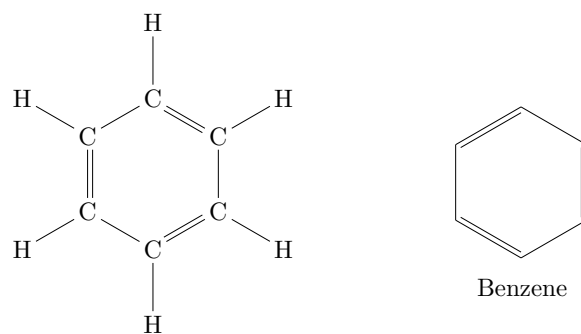
(a) Vector graphics with **TikZ**
<https://texample.net/tikz/examples/coffee-cup/>



(b) Parse trees with **qtree**
<https://www.ling.upenn.edu/advice/latex/qtree/>

$$\frac{\frac{\frac{[A \rightarrow (B \rightarrow C)] \quad [A]}{B \rightarrow C} \quad \frac{[A \rightarrow B] \quad [A]}{B}}{C}}{A \rightarrow C}}{(A \rightarrow B) \rightarrow (A \rightarrow C)} \\
 \frac{}{(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))}$$

(c) Proof trees with **prftree**
<https://ftp.gwdg.de/pub/ctan/macros/latex/contrib/prftree/>



(d) Chemical structural formulas with **chemfig**
<http://latex-cookbook.net/cookbook/examples/benzene-ring/>

Figure 14.1: Examples for some packages

Colors for your documents are provided by `xcolor`.⁷

Notes that you cannot overlook can be created with `todonotes`.⁸ With this package, you can mark what you still have to change within your document.

Pages of other PDF files can be integrated into the source code with the `pdfpages`⁹ package. It comes in very handy whenever one needs the output of external programs in the document, for example, in the appendix. Just compile the document one more time and the appendix is up to date again, if the external program has changed something.

Nested graphics and the positioning of captions at almost any place are provided by `subcaption`.¹⁰ We also made extensive use of this package.

Tables can be designed much more flexibly than what we have shown here. The following packages can help you with that: `colortbl`,¹¹ `tabularx`,¹² `multirow`,¹³ `makecell`.¹⁴

`beamer`, which is not a package, but another document class, can be used to create **slide shows** with L^AT_EX. Information on the document class and examples are available at Overleaf,¹⁵ which brings us to the next section:

Please
do not
change.
This is an
example.

14.2 Help and information

Wikibooks provides you with a much more detailed introduction into L^AT_EX. Note that the German version¹⁶ is less complete than the English one.¹⁷ If required, both refer to additional packages.

Whenever you need information on certain packages, **CTAN**¹⁸ is your place to go. For each package, you can find the official documentation as a PDF file there. Within this file, the first paragraphs are usually the most interesting. They are followed by implementation details, that you normally do not need.

If the official documentation is too theoretical, and you prefer a more hands-on approach, **Overleaf**¹⁹ can help you out. Primarily, it is a collaborative online L^AT_EX editor. However, you can find multiple templates²⁰ for different types of documents (CVs, theses, ...) there.

If you are looking for examples dedicated to TikZ, **T_EXample**²¹ provides you with multiple of them.

⁷<https://www.ctan.org/pkg/xcolor>

⁸<https://www.ctan.org/pkg/todonotes>

⁹<https://www.ctan.org/pkg/pdfpages>

¹⁰<https://www.ctan.org/pkg/subcaption>

¹¹<https://www.ctan.org/pkg/colortbl>

¹²<https://www.ctan.org/pkg/tabularx>

¹³<https://www.ctan.org/pkg/multirow>

¹⁴<https://www.ctan.org/pkg/makecell>

¹⁵<https://www.overleaf.com/learn/latex/Beamer>

¹⁶<https://de.wikibooks.org/wiki/LaTeX-Kompedium>

¹⁷<https://en.wikibooks.org/wiki/LaTeX>

¹⁸<https://ctan.org/>

¹⁹<https://www.overleaf.com/>

²⁰<https://www.overleaf.com/latex/templates>

²¹<https://texample.net/>

For concrete questions, the question-answering platform **Stackexchange** is a good place to go: There even is a T_EX community there.²²

Of course, you can always contact us with your questions:

- via mail to `fachschaft-wiai.stuve@uni-bamberg.de`,
- via phone at +49951 863 1219,
- or just come to our bureau at WE5/02.104.

²²<https://tex.stackexchange.com/>