

## Example Project

# How to write in Latex

A helpful guide to get started and to show some common use cases

Max Mustermann 1234567

Mira Musterfrau 9876543

~~01.01.2020~~

May 24, 2024

**Professor:** your Professor

---

## **Declaration of Authorship**

We hereby certify that the work we are submitting is entirely of our own making except where otherwise indicated. We are aware of regulations concerning plagiarism, including disciplinary actions that may result from it. Any use of the works of any other author, in any form, is properly acknowledged at their point of use.

---

Max Mustermann

---

Mira Musterfrau

---

# Abstract

If you need an abstract for your document, you can write it wherever you see fit by using the `\begin{abstract}...\end{abstract}` environment, like demonstrated here. It acts as an unnumbered chapter. You can choose if you want it in the TOC using the `abstract=totoc` and `abstract=nottotoc` options of the documentclass.

If you prefer your abstract to be on a clean page, you can use `\thispagestyle{plain}` to get only a page number or `\thispagestyle{empty}` to get no header or footer.

If you use the `\keywords{list, of, keywords}` command in your preamble, the given keywords will also be printed here. You may use `abstract=nokeywords` as a documentclass option to disable this.

**Keywords**     some, informative, keywords

# Contents

<b>1</b>	<b>What is LaTeX</b>	<b>1</b>
1.1	Following this document . . . . .	1
1.2	Requirements to use LaTeX . . . . .	1
1.3	Running LaTeX . . . . .	2
1.4	LaTeX commands . . . . .	2
1.5	Getting more information . . . . .	2
<b>2</b>	<b>Starting a new document</b>	<b>3</b>
<b>3</b>	<b>Formatting text</b>	<b>4</b>
3.1	Texts and paragraphs . . . . .	4
3.2	Headings . . . . .	4
3.3	Text spacing . . . . .	5
3.4	Breaking pages . . . . .	5
3.5	Text styling . . . . .	5
3.6	Special characters . . . . .	5
3.6.1	LaTeX command characters . . . . .	5
3.6.2	Invisible characters . . . . .	6
	<b>List of Figures</b>	<b>7</b>
	<b>List of Tables</b>	<b>8</b>

# 1 What is LaTeX

So you decided to get stated with LaTeX. Great! So let's talk a bit about the basic concept and differences it comes with.

Up to this point you probably used a Word Processor like MS Word. The kind of workflow you know from there is often referred to as *What you see is what you get*. You see the exact final layout as you type it, press some colourful buttons to insert stuff and if it doesn't want to do something you need, you're screwed.

LaTeX on the other hand falls into the category of *What you see is what you mean*, which describes all forms of markup languages. This means you create your LaTeX document as a simple plain text file without any from of formatting and mix in a bunch of commands telling what you mean. For example: "This is supposed to be a chapter heading", "make this bold" or "insert an image here". This source file will than be passed to a document processor (the LaTeX program), which will, depending on its settings, create the document for you. The advantage is, that you can use the same markup with all sorts of formattings and target file types.

This is why working with LaTeX will require some getting used to and you will find yourself wanting to compile every five seconds to see the document update. Try to restrain yourself and concentrate on writing. You will find yourself working much faster.

## 1.1 Following this document

To see how the LaTeX source code and the resulting PDF correspond, I recommend you open this documents source code and PDF file next to each other and scroll through them simultaneously.

If you already have a working LaTeX setup, most editors support *SyncTex*, which allows you to jump between source code and PDF file and vice versa. You have to compile yourself, which will create a file called `example.synctex.gz` in your project directory. Now you can `<CTRL>+Click` in the PDF and the corresponding line of source code will be highlighted.

The shortcut to jump from the source code into the PDF will depend on your Editor, but for VS Code its right `Click→SyncTex from cursor` or `CTRL+ALT+J`.

## 1.2 Requirements to use LaTeX

As LaTeX files are just plain text file, you can edit them with any text editor (even windows notepad works, but that's just terrible). However, I would strongly recommend a more suitable editor. I use Visual Studio Code (which is a multi porpoise text editor that support all major programming languages) but you could also use something like Texmaker, which is an editor specifically for LaTeX. There is also the online editor Overleaf, which saves you the trouble of setting up your own LaTeX installation and provides everything you need in the cloud.

As I have already mention above, you also need the LaTeX program. It comes bundled with packages and other additional software inside a Tex-distribution. There are two major ones, Texlive and MiKTeX. I recommend MiKTeX, but it essentially doesn't matter which one you choose.

Once you have the distribution installed, test it by running `pdflatex --version` in any terminal windows and it should return you some information about the installed version and setup.

## 1.3 Running LaTeX

To create a PDF file from your LaTeX source code, you can always navigate to the project folder in a terminal window and run `pdflatex filename.tex`. However, if you have a decent editor installed, it will provide you with a button and do this for you.

With these project files you also received a makefile, which demonstrates how to compile this example file successfully from the terminal. The README file also has some tips and information for you.

If you use VS Code, this project also contains settings for LaTeX and recommended extensions. If you open the folder for the first time you will be asked if you want to install them and should than be able to compile this file.

## 1.4 LaTeX commands

Now lets look at the LaTeX command. Every one will begin with a `\` followed by a letters only command name, like this: `\command`. Most commands also accept input, which is put after it into curly brackets: `\command{argument}`. They can accept multiple arguments either in multiple sets of curly brackets or as a comma separated list, depending on the command.

Some commands also accept optional arguments. These are passed inside square brackets between the command name and the curly brackets, like this: `\command[optional]{argument}`.

## 1.5 Getting more information

So what can you do if you get stuck or just want more information. The simple answer is: Google is your friend. Most questions have already been answered. For example on [Tex Strackexchange](#). Also, Overleaf has a great section for learning LaTeX.

An of course you can always check the documentation, which you can find on [CTAN](#).

## 2 Starting a new document

## 3 Formatting text

To begin I want to show you the basics of how to get text onto the page and structure it. You can also see how I created this exact text as an example.

### 3.1 Texts and paragraphs

Writing text for a LaTeX document is very easy. You just put the text in. LaTeX doesn't care about line breaks or whitespace, so its up to your preferences how the source code looks. You could just write the whole document as one super long line, but it makes sense to break it up and keep it readable. One common way is to put every sentence on a new line. Alternatively lots of editors can break lines for you after you reach a certain width (that's how this source code is formatted).

LaTeX will automatically space and break your text to optimal use the available space while still looking good. You can however assist it when it struggles. Putting hyphens (-) into a word tells LaTeX to break it there. If it's a word you use a lot, you can use `\hyphenation{very-long-word}` in your preamble to tell LaTeX how to split it everywhere.

Lots of examples will tell you that `\\` is a line break. While this is correct you shouldn't use it to break your text block and start a new line. A block of text is a paragraph and should be ended with the `\par` command. For ease of use LaTeX will automatically use this command if you leave a blank line. (see this source code)

If you want to further separate paragraphs visually (when you finish a train of thought for example) you can use the commands `\smallskip`, `\medskip` and `\bigskip`

### 3.2 Headings

The exact commands available will vary depending on you your documentclass, but they will always be a single command that expects any text inside curly brackets, for example `\section{text}`. The different commands form a hierarchy you can nest into each other, keeping track of its parent element. That means you don't have to worry about any formatting or numbering, LaTeX will handle that for you.

When using an *article* documentclass, the commands available are `\section{}`, `\subsection{}` and `\subsubsection{}`. Should you need more nesting levels, you are usually overcomplicating things, but you could additionally use the `\paragraph{}` command, which gives you a slightly bigger, bold first word for your paragraph.

The *report* documentclass adds the additional command `\chapter{}` as the highest heading level. You can still use the previous three commands for the nested headings. A chapter automatically starts on a new page, so it should be at least two pages long. You also get the command `\part{}`, which creates a separate page for the part's title. These should only be used in very long documents.



### 3.3 Text spacing

By default, these classes add no spacing between paragraph, but sometimes you want to visually enforce a breakpoint in your argumentation. For that you can add some space in between to paragraph by using one of the commands `\bigskip`, `\medskip` or `\smallskip`. How much space you want depends on your taste, but you should keep it consistent. Here is an example:

This text has a big space before it,  
Here I used just some medium spacing  
and this is a small space.

### 3.4 Breaking pages

Sometimes you will find yourself in situations, where you don't like where LaTeX splits your text to the next page. So first, take some advice: Don't worry about it for now. Your text will probably change a few times before its final. Just leave it.

If you are at the final stage, you can do a beautifying pass. Now you can use `\pagebreak` to tell LaTeX about better places to break the text.

Should you still not be happy (this happens especially with multiple images/tables in close proximity) you most likely have to little text and should redesign your document. But if you absolutely want to print it that way, you can use `\clearpage` to force all figures/tables to be put onto the page and then start a new page.

You might also need just a little more space only a page to just fit one more sentences. For that you can use the command `\enlargethispage{N\baselineskip}` with  $N$  being the number of lines you need. Use this sparingly however, as the bottom margin is there for a reason and you shouldn't intrude on the footer too much.

### 3.5 Text styling

When writing text, you will need to *emphasize* certain parts of the text. The easiest way is to use the `\emph{}` command around you text. You can also nest it to emphasize *even more*.

If you want to change to a specific font-type, you can do that like this:

<code>\underline{text}</code>	<u>Underlined</u>
<code>\textbf{text}</code>	<b>Bold Font</b>
<code>\textit{text}</code>	<i>Italic Font</i>
<code>\textrm{text}</code>	Roman Font
<code>\texttt{text}</code>	Typewriter Font
<code>\textsc{text}</code>	Small Caps Font

You might also want to change your text colour, which is what the color package is for. It provides the `textcolor{colour}{text}` command, which allows you to change your text colour.

### 3.6 Special characters

#### 3.6.1 LaTeX command characters

As in most programming languages, some characters are used for LaTeXes commands and can't be used in text directly. Here is a table explaining them all:

<i>character</i>	<i>special meaning</i>	<i>how to get character</i>
<code>\</code>	beginning of a command	<code>\textbackslash</code>
<code>{</code> and <code>}</code>	denote a code block	<code>\{</code> and <code>\}</code>
<code>%</code>	beginning of a comment	<code>\%</code>
<code>#</code>	macro parameter character	<code>\#</code>
<code>\$</code>	beginning/end of math mode	<code>\\$</code>
<code>~</code>	non-breaking space	<code>\textasciitilde</code>
<i>only inside math mode:</i>		
<code>_</code>	subscript	<code>\_</code>
<code>^</code>	superscript	<code>\textasciicircum</code>

### 3.6.2 Invisible characters

To properly typeset your text you may need a number of special characters under specific circumstances:

<i>explanation</i>	<i>command</i>	<i>example</i>
non-breaking space	<code>~</code>	Max Mustermann (Names shouldn't be broken)
3/4 non-breaking space	<code>\;</code>	10 000 (separate thousands)
medium non-breaking space	<code>\:</code>	z. B. (abbreviations)
1/2 non-breaking space	<code>\,</code>	1 V (number + unit)
normal space	<code>\space</code>	in case some command eats up all your space

## List of Figures

## List of Tables