

L^AT_EX for the Humanities

Dr. Xander Vertegaal
a.j.j.vertegaal@uu.nl

28th February 2024

Welcome to this introduction to L^AT_EX! After working your way through this tutorial, you will be able to write and typeset your own documents in L^AT_EX. For this tutorial, I will assume that you are using [Overleaf](#), but local installations ([MiK_TE_X](#) for Windows, [Mac_TE_X](#) for Apple) in combination with a code editor (e.g. [Textmaker](#) or [Textstudio](#)) work perfectly well too.

1 Introduction - what is L^AT_EX exactly?

L^AT_EX (/ˈlɑːtɛx/ or /ˈleɪtɛx/) is based on T_EX, a **digital typesetting system** developed in 1978 by prof. dr. Donald Knuth (Stanford University). Knuth was unhappy with the quality of books that were typeset using the then current practice of phototypesetting and based his new digital typesetting program on the quality of old hot metal typesetting (monotype and linotype).

T_EX allows the user to create high-quality books, articles, theses and even PowerPoint-like presentations or exams, taking care of many tedious formatting tasks behind the scenes, while still allowing the authors full control of their documents' look and feel. In T_EX\L^AT_EX you use special **programming commands** to indicate how you want your document to be typeset. Your text and the commands are then sent to a so-called **compiler**, a programme that reads your text and the added commands and turns them into a PDF file. One particularly nice feature of L^AT_EX is that it is **platform-independent**: the same L^AT_EX code will yield the same PDF on any operating system, regardless of whether you are using Windows, Mac OS or Linux.¹

Initially, T_EX was quite difficult to use: simple tasks like creating a section header required a large set of commands to sculpt the text into the desired shape. This is why people started to develop **macro packages** that would make life easier for the average user. These are basically sugar-coated, abbreviated combinations of T_EX commands with a lot of carefully selected default settings.

¹The original T_EX and L^AT_EX would first compile a so-called DVI (DeVice Independent) file that contains all the lay-out information of your document in a way that yields the exact same result regardless of operating system. This DVI file can then be converted into a PDF. Overleaf's pdfL^AT_EX skips the intermediary DVI step and compiles a PDF right away (which is what most people want anyway).

The most popular of these macro packages is \LaTeX , which was developed in 1984 by Leslie Lamport. Nowadays, the raw \TeX commands are hardly ever used directly, but they still form the backbone of almost all \LaTeX functionality. Several more customisations and extensions of Knuth’s original \TeX have been developed (\XeLaTeX , \LuaTeX , \pdfTeX), but \LaTeX is the one most commonly used.

2 Using Overleaf and creating a new document

In your Overleaf “Projects” view, click [New Project](#) \gg [Blank Project](#) to start a new project and give it a nice project name. For now, a “Blank Project” will do—in a later stadium you can also use a project template, upload a project from your computer or import a project from GitHub (for those who are familiar with it).

Exercise 1

Create a new project and call it *LaTeX workshop*.

Once you’ve created a project, you will end up in the editor. The editor consists of three parts.

Left is the **project overview**: here we see all files in your project. As noted above you can have modular projects, where you import files into other files. For now, we have only one file: `main.tex`. Below that, the File Outline presents a list of clickable links to various sections of our document.


In the middle we have the text editor, where we write our text and \LaTeX commands. This will be fed to the compiler.


Right is the **PDF preview** of your current project, so that you can check if everything is going according to plan. You can click [Recompile](#) to tell Overleaf to update to PDF to its latest version (or press `ctrl/⌘ + Enter` or `ctrl/⌘ + S`).

Now lets add some text to the middle of the screen and tell \LaTeX to recompile.

Exercise 2

Add ‘Hello World!’ somewhere in between `\begin{document}` and `\end{document}` and recompile your document.

Lo and behold! You have just created your first \LaTeX document! You can download it as a PDF by clicking the  button, or by clicking [Menu](#) \gg [PDF](#). The source code can be downloaded by clicking [Menu](#) \gg [Source](#).²

²There is a also a button labeled  that allows you to view the ‘raw’ compiler logs. We will return to this later in the tutorial, cf. Section 12.1.

2.1 L^AT_EX commands

Right now, our `main.tex` looks like this. We can see that Overleaf has already added several things for us.

```
\documentclass{article}
\usepackage{graphicx} % Required for inserting images

\title{test}
\author{Xander Vertegaal}
\date{December 2023}

\begin{document}

\maketitle

\section{Introduction}

\end{document}
```

Of special note here are the **commands**. You can recognize them by the `\` that precedes them. Simple commands have the following structure.

```
\commandname[optional-argument]{mandatory-argument}
```

Some commands, like `\maketitle` or `\LaTeX` (which prints the L^AT_EX logo) do not need any arguments and can be run as is. Sometimes, a command changes the formatting of the rest of the document (such as `\centering`, which will center all text after it), in which case they are called **switches**.³

Other commands, like `\documentclass` and `\section`, require additional information to function. These so-called **mandatory arguments** are added in between curly brackets, e.g. `Introduction` in `\section{Introduction}`. Commands can also take **optional arguments**. These are provided using square brackets and normally come in between the function name and the mandatory argument(s). In the case of `\documentclass{article}`, for instance, we can specify that we are in *draft mode*, which speeds up compilation on large documents, by writing `\documentclass[draft]{article}`.⁴

Lastly, there are also **environments**, starting with `\begin{...}` and ending with `\end{...}`. These used to envelop a block of text and apply a certain formatting style to it. For instance, the `\begin{center}... \end{center}` environment is used to center whatever text it surrounds. We will see more of these later on.

³When you use these commands in running text, L^AT_EX will swallow a single space after them. To prevent this, write `\maketitle{}`, as if calling it with an empty mandatory argument.

⁴For multiple mandatory arguments, we add multiple sets of curly brackets. Multiple optional arguments can all be put in one set of square brackets, separated by `,`.

2.2 Document class and document structure

Every \LaTeX document must contain at least the following two elements. If either of these is missing, \LaTeX will throw an error and refuse to compile your document. (Try it out!)

1. `\documentclass{...}` defines the type of document we are creating. This must be the first line of your document. By default, this is **article**, but we can also indicate that we are making a book (**book**), a letter (**letter**) or even a PowerPoint-like presentation (**beamer**). This has some influence on the layout of the document.⁵
2. `\begin{document}...\end{document}` encloses the so-called **document body** (which must not be empty). This environment encloses what will eventually be rendered to the PDF.

Everything before the document body is called the **preamble** (or document head); This is where we define the metadata of our document, such as the title, author and date, and adjust global settings that apply to the entire document.

The preamble is also where **packages** are imported. Packages are extensions to \LaTeX that add extra functionality. For instance, the package **graphicx** allows us to import images into our document. You can simply import new packages using the `\usepackage{nameofpackage}`. \LaTeX will automatically retrieve them from [the Comprehensive \$\text{\TeX}\$ Archive Network \(CTAN\)](#).

Exercise 3

Import the package **fontspec** in your preamble. We will need it later on.

3 Basic text editing

You see that \LaTeX has already taken a lot of decisions for us: it has picked a default font and font size, centered the title and made a nice bold, numbered Section title. We did not have to worry about any of that! Naturally, the standard layout will often not do, which is why we need to be able to modify it ourselves.

3.1 Type style

Type style is specified by three categories: **shape**, **series** and **family**, cf. Table 1. For articles, the default is *upright medium Roman*. If your font supports it, you can combine these categories, e.g. **bold typewriter**, *italic sans*

⁵Journals and conferences often distribute a style file for you to use when writing your paper—strictly speaking they define their own document class in which they define a whole bunch of stuff, such as page layout and bibliography style.

serif. However, it is not possible to use two from the same category at the same time (e.g. upright and italic).

Effect	Command	Switch
Upright shape	<code>\textup{...}</code>	<code>\upshape</code>
<i>Italic shape</i>	<code>\textit{...}</code>	<code>\itshape</code>
<i>Slanted shape</i>	<code>\textsl{...}</code>	<code>\slshape</code>
SMALL CAPS SHAPE	<code>\textsc{...}</code>	<code>\scshape</code>
Medium series	<code>\textmd{...}</code>	<code>\mdseries</code>
Bold series	<code>\textbf{...}</code>	<code>\bfseries</code>
Roman family	<code>\textrm{...}</code>	<code>\rmfamily</code>
Typewriter family	<code>\texttt{...}</code>	<code>\ttfamily</code>
Sans serif family	<code>\textsf{...}</code>	<code>\sffamily</code>

Table 1: Type styles with a sample text.

The commands in the **Command** column are normally used for individual words up to a few sentences (although there is nothing preventing you from wrapping an entire text in a block). For longer texts, however, the switch commands in the third column are easier to read. Note, however, that switch commands change the formatting until it is changed back or until their **scope** ends. A command's scope runs from the part of the text when it is called until the first closing curly brace (`}`) or until the end of the environment it is in.

The switch commands in the third column can also be used as environments, to enclose larger pieces of text, so `\begin{upshape}...\end{upshape}` does the same as `\textup{...}`.

In summary, the text

This is upright. *This is Italic.* This is upright again.

can be gotten by any of the four following lines:

```

% With a regular command
This is upright. \textit{This is Italic.}
This is upright again.

% With switches
This is upright. \itshape This is Italic.
\upshape This is upright again.

% With a 'scoped' switch in between {...}
This is upright {\itshape This is Italic.}
This is upright again.

% With an environment
This is upright.
\begin{itshape}
  This is Italic.
\end{itshape}
This is upright again.

```

Overleaf provides handy shortcuts for some of these, so as not to upset Windows/Mac users: `ctrl/⌘+b` for `\bfseries`, `ctrl/⌘+i` for `\itshape`.⁶

A very useful command is `\emph{...}`, which makes your text stand out from its surroundings. It does this by making the text italic if the surrounding text is upright and vice versa.

Exercise 4

Recreate the following styles, both with switches and commands with arguments: **bold**, SMALL CAPS, *italic typewriter*, **bold sans serif**.

3.2 Font size

Font size is also controlled with switch commands, cf. Table 2. They are used in the same way as the switches mentioned in the previous section. Notice that \LaTeX uses relative font sizes by default. This ensures that the ratio between larger and smaller text remains the same if you change the overall fontsize of your document. You can change the overall fontsize by adding `10pt`, `12pt` etc. as an optional argument to `\documentclass`. \LaTeX will then also adjust the page margins to match the font size.

As with type styles, these switches can also be used as environments, e.g. `\begin{huge}...\end{huge}` will produce the same result as the switch `\huge`.

⁶Note that `ctrl/⌘+u`, surprisingly, capitalises the selected word.

Command	Output
<code>\tiny</code>	sample text
<code>\scriptsize</code>	sample text
<code>\footnotesize</code>	sample text
<code>\small</code>	sample text
<code>\normalsize</code>	sample text
<code>\large</code>	sample text
<code>\Large</code>	sample text
<code>\LARGE</code>	sample text
<code>\huge</code>	sample text
<code>\Huge</code>	sample text

Table 2: Font sizes with a sample text.

3.3 Quotation marks

A quick note about quotation marks. While MS Word automatically changes quotation marks to their right form (i.e. facing left or facing right), \LaTeX does not. In \LaTeX , a single opening quotation mark is created using ``` (backtick or accent grave), a single closing quotation mark with `'` (apostrophe or accent aigu). Double quotation marks are printed by simply doubling quotation marks, e.g.:

```John''` or `''John''`

This yields “John” or ”John”. (In most cases, you will want the first one.)

#### Exercise 5

Recreate the following text: “*Hello world!*” This is large. This is tiny and sans serif.

### 3.4 Text alignment and positioning

By default, text in  $\text{\LaTeX}$  is fully justified, yielding nicely rectangular blocks of text. To switch to different alignments, use the commands below, cf. Table 3.

Alignment	Environment	Switch
Left	<code>\begin{flushleft}...\end{flushleft}</code>	<code>\raggedright</code>
Right	<code>\begin{flushright}...\end{flushright}</code>	<code>\raggedleft</code>
Center	<code>\begin{center}...\end{center}</code>	<code>\centering</code>

Table 3: Alignment commands

A word of warning: after using a switch like `\centering` and `\raggedright` in your main text, standard  $\text{\LaTeX}$  has no option of reverting back to fully justified alignment. For this, you will need the package `ragged2e`, which allows you to use the environment `\begin{justify}...\end{justify}` and the switch command `\justify`.

#### Exercise 6

Try to reproduce the following text:

**This text is centered and bold.**

*This text is right-aligned, italicised and ‘Large’!*

### 3.5 Whitespaces and indentation

While doing the previous exercise, you may have noticed that  $\text{\LaTeX}$  does not interpret a newline as a new line. This means that if I type

```
John has a cat.
The cat is also called John.
```

$\text{\LaTeX}$  will print these two sentences on the same line. This is convenient for keeping your  $\text{\LaTeX}$  code nice and clean, but not for people who want to start a new paragraph. In order to force a newline, type `\newline`. This will start a new line, but not a new paragraph.

```
John has a cat.\newline
The cat is also called John.
```

You can start a new paragraph by simply leaving a empty line between two bits of text.

```
John has a cat.

The cat is also called John.
```

$\text{\LaTeX}$  automatically adds indentation to every new paragraph. Try it out to see the difference.

To create *vertical whitespace* between two lines, just like the one above this paragraph, there are a few commands at your disposal.

- `\baselineskip` inserts the space of one text line between two paragraphs.
- `\smallskip` inserts a tiny space between two paragraphs.
- `\medskip` inserts a medium space between two paragraphs.
- `\bigskip` inserts a large space between two paragraphs.



- `\vspace{...}` inserts a fixed amount of space between two paragraphs in centimeters, millimeters, inches etc., such as `\vspace{2cm}`. You can also enter negative length here to (e.g. `\vspace{-2cm}`) to bring two pieces of text closer together.
- `\vfill` puts the following paragraph at the bottom of the page and inserts as much whitespace as needed between that paragraph and the preceding one.

The precise height of these skips varies depending on the font (size) and the content of the page.  $\text{\LaTeX}$  can stretch or shrink these skips a bit if it encounters problems while filling the page. This is why the first four **rubber lengths** are generally more appropriate than `\vspace{...}`.

For *horizontal spaces*, the following commands are used.

- `\_` (i.e. a slash and a space): adds a single word space.
- `\quad`: adds a space equal to the point size (font height), approximately the width of an uppercase ‘M’.
- `\qqquad`: adds the width of two `\quads`.
- `\enspace`: adds half a quad, approximately the width of a lowercase ‘n’.
- `\hspace...` adds a space of a user-defined length, in centimeters, millimeters, inches etc., such as `\hspace{3in}`. Again, negative lengths can also be used.
- `\hphantom{...}`: provides horizontal whitespace as wide as the text in between brackets. For example: `\hphantom{text}` gives the width of the word ‘text’ (without printing the word itself).
- `\hfill`: puts the next word at the end of the line and inserts as much whitespace as needed between that word and the preceding one.

Again, the space produced by the first four spacing commands is relative to your actual font size, so the effect should stay the same, even if you decide to change the font size of your document. For this reason, they are generally preferred over the absolute command `\hspace`.

$\text{\LaTeX}$  will play around with the space between characters, words and sentences to allow for an optimal filling of the line. It will especially put a bit more space to occur after a full stop . for aesthetic reasons. This may have unwanted effects, especially when you are using abbreviations such as e.g. or dr. in the middle of a sentence, as  $\text{\LaTeX}$  will treat it as the end of a sentence. To avoid this, force  $\text{\LaTeX}$  to put a normal space between words there by writing a slash followed by a space, for instance: `dr.\_Shackleton`.

Description	L <sup>A</sup> T <sub>E</sub> X command	Output
Grave accent	<code>\`{o}</code>	ò
Acute accent	<code>\'{o}</code>	ó
Circumflex	<code>\^{o}</code>	ô
Umlaut/trema/dieresis	<code>\"o</code>	ö
Double acute	<code>\H{o}</code>	ő
Tilde	<code>\~{o}</code>	õ
Cedilla	<code>\c{s}</code>	ç
Ogonek	<code>\k{o}</code>	ę
Barred L	<code>\l</code>	ł
Macron	<code>\={o}</code>	ō
Bar under the letter	<code>\b{o}</code>	⒐
Dot over the letter	<code>\.{o}</code>	ȝ
Dot under the letter	<code>\d{o}</code>	ḡ
Ring over letter	<code>\r{a}</code>	å
Breve	<code>\u{o}</code>	ö
Caron/háček	<code>\v{o}</code>	ǎ
Tie over two letters	<code>\t{oo}</code>	ōo
Slashed o	<code>\o</code>	ø
Dotless i or j	<code>\i</code> or <code>\j</code>	ı or ʝ

Table 4: Diacritics in L<sup>A</sup>T<sub>E</sub>X

### 3.6 Diacritics and special characters

Diacritics and special characters in L<sup>A</sup>T<sub>E</sub>X can be entered with (you guessed it) commands, cf. Table 4.

Putting diacritics on capital letters works in the same way, and you can even combine multiple diacritics if your font allows it: `\̈n` (`\={\d{n}}`).<sup>7</sup>

You may already have hotkeys or keyboard layouts set up to directly enter these characters and other modified characters in the text editor. At this point, L<sup>A</sup>T<sub>E</sub>X's age shines through. It was developed in the 1980s, when the internet was still in its infancy and the world was a much less globalised place. As such, it was not designed to handle non-Latin scripts. If you try to copy and paste the character *q* or a non-western character such as 猫 in your text editor, it will fail, even if your font supports it. If this happens to you, it may be a good idea to switch to a more modern compiler. X<sub>Y</sub>L<sup>A</sup>T<sub>E</sub>X is able to interpret a wider range of characters than the traditional L<sup>A</sup>T<sub>E</sub>X or pdfL<sup>A</sup>T<sub>E</sub>X compilers. To switch to X<sub>Y</sub>L<sup>A</sup>T<sub>E</sub>X, select **XeLaTeX** at **Menu** **Compiler** in Overleaf.

<sup>7</sup>The standard font provided by L<sup>A</sup>T<sub>E</sub>X is not well-suited for this. As detailed here <https://tex.stackexchange.com/questions/159291/multiple-diacritics-on-one-character>, freely downloadable fonts such as **Brill** or **Charis SIL** have better support for this.

### Exercise 7

Try to reproduce the following (non-sensical) combinations:  $\ddot{p}$ ,  $\bar{g}$ ,  $\dot{i}$ ,  $*$ .

## 3.7 Changing fonts

You may have noticed that some characters do not display very well in  $\text{\LaTeX}$ , especially signs involving multiple diacritics, or words spelled with non-Latin scripts. This is because the font that  $\text{\LaTeX}$  uses by default (*Computer Modern*) is excellent for mathematical formulae, but cannot display all combinations of diacritics, let alone text written in non-roman writing systems, such as Greek, Arabic, Amharic or Chinese. To be able to print these characters, we need to change to a different font, and in order to change to a modern font (OpenType or TrueType), we need the package `fontspec`, which in turn requires the  $\text{\XeLaTeX}$  compiler (see Section 3.6 above).<sup>8</sup>

In Section 3.1 we saw the three fonts (also known as *font families*) that standard  $\text{\LaTeX}$  provides, and we learned that we can use `\textrm{...}`, `\texttt{...}` and `\textsf{...}` to switch to them. To override the fonts behind these commands, use the following.

1. `\setmainfont{<font name>}` to set the main or ‘roman’ font.
2. `\setsansfont{<font name>}` to set the ‘sans serif’ font.
3. `\setmonofont{<font name>}` to set the ‘monospace’ or ‘typewriter’ font.

For instance, to change *the roman font* to the font *Charis SIL*, we can use `\setmainfont{Charis SIL}`. The `\textrm{...}` command will now print in *Charis SIL*.

Often, however, you will want to use more than one roman or sans-serif font family. In that case, you can define your own font family using:

```
\newfontfamily{<command>}{}
```

This creates a new switch command. For instance, if we want to create a command `\charis` that lets us switch to *Charis SIL* whenever we need, we could write the following in our preamble.<sup>9</sup>

```
\newfontfamily{\charis}{Charis SIL}.
```

<sup>8</sup>You are highly recommended to use the  $\text{\XeLaTeX}$  compiler for all your documents, as it is more modern and has better support for non-Latin scripts.

<sup>9</sup>For a one-off switch, not recommended for common use, you can use `\fontspec{<font name>}`. This will switch all following characters to the specified font, until you switch back to the main font using `\normalfont`. Note, however, that `\fontspec` forces  $\text{\LaTeX}$  to load the font into its memory *ad hoc* each time it is called, leading to longer compilation times. It is therefore almost always preferable to use `newfontfamily`, which only needs to load a font once.

L<sup>A</sup>T<sub>E</sub>X can be made aware of **font names** in various ways.

- Overleaf has a wide range of fonts already loaded in. You can view the full list (with examples) in [this document \(Overleaf.com\)](#). If you are running a downloaded version of L<sup>A</sup>T<sub>E</sub>X on your own system, these will likely not work.
- If you are running L<sup>A</sup>T<sub>E</sub>X on your own device, you can use fonts that are already installed on your computer. For Windows, these are found in `C:/Windows/Fonts`. You can use these fonts by specifying the font name as it appears in your font manager, e.g. `\setmainfont{Times New Roman}`. Check out the [documentation \(CTAN.org\)](#) of the `fontspec` package for more information. These fonts are not available on Overleaf.
- Some fonts are available (both on Overleaf and in your own L<sup>A</sup>T<sub>E</sub>X installation) as **packages**. For instance, adding `\usepackage{carolmin}` enables the command `\cmfamily`, which we can use to **create beautiful Carolingian minuscule**<sup>10</sup>. You can find a list of all available font packages in the [L<sup>A</sup>T<sub>E</sub>X Font Catalogue \(tug.org\)](#).
- Lastly, you can add the font files to your project and use them directly. This is a bit more complicated, but allows you to use fonts that are not available on Overleaf. For more information, see the [documentation \(CTAN.org\)](#) of the `fontspec` package.

#### Exercise 8

Try switching the main font to Charis SIL (which has excellent support for diacritics) for a little bit. Try to create the character á now, and then switch back to the standard font, where á looks funky.

## 4 Defining your own commands

Since L<sup>A</sup>T<sub>E</sub>X is a programming language, you can (and should) define your own commands, especially if you find yourself using the same commands over and over again.

New commands that do not require any arguments can be defined as follows.

```
\newcommand{name}{definition}
```

The **name** is the name of your command (starting with a `\`). When the compiler finds this command, it will replace it with the code in the **definition**.<sup>10</sup> Suppose, for instance, that we want to make a command for the word *supercalifragilisticexpialidocious* because we cannot remember how to spell it. We can define a new command `\supcal` as follows.

<sup>10</sup>If there is already a command with your picked name, L<sup>A</sup>T<sub>E</sub>X will throw an error. You can redefine existing commands using `\renewcommand` instead of `\newcommand`.

```
\newcommand{\supcal}{supercalifragilisticexpialidocious}
```

In order to define a new command with arguments, you put the amount of required arguments (in square brackets) between the first and second arguments. Then, you can refer to these required arguments in your definition with #1, #2, etc.

For example, a new command that takes two arguments and prints the first in bold and the second in italics can be defined as follows:

```
\newcommand{\mycoolcommand}[2]{\textbf{#1} \textit{#2}}
```

Calling `\mycoolcommand{John}{Mary}` will now print: **John** *Mary*.

Slightly more useful: we can turn the switch command we created to render text in the Charis SIL font into a command.<sup>11</sup>

```
\newcommand{\charis}[1]{\{\charissil #1}}
```



### Exercise 9

Try making a command called `\subscriptor{...}{...}` that takes two arguments. It should put the second one in italicised subscript to the first, so that `\subscriptor{Hands}{down}` becomes *Hands<sub>down</sub>*.

### Exercise 10

Create a command `\hugeify{...}` that takes one argument and makes it huge. Make sure the text around it is not affected.

## 5 Commenting out

If there is a line that you do not want in your final PDF but you do not want to delete it either, you can tell L<sup>A</sup>T<sub>E</sub>X to ignore it when it is compiling—this is called **commenting out**. In L<sup>A</sup>T<sub>E</sub>X this is done with a %: all text on the same line following this symbol is ignored. Overleaf also has a handy shortcut for toggling this:  + .

When working on larger documents, commenting out sections you are not working on can decrease compilation time. It can also help you find errors in your code: if L<sup>A</sup>T<sub>E</sub>X fails and you do not know where the error is, comment out parts of your document incrementally until you find the offending line.

However, if you want a simple percentage sign in your text, you need to type `\%`, otherwise L<sup>A</sup>T<sub>E</sub>X will interpret it as a comment.

<sup>11</sup>As a general note, it is good practice to create commands for recurring types of text that requires special formatting, such as words in a special language, definitions or blocks of code. For instance, a user-defined command `\author{...}` that is defined as `\textsc{#1}` to print out an author's name in small caps, indicates much more clearly what you are writing and allows you to change all instances of author names later, without incidentally changing other instances where you use small caps. In practice, you should hardly have to use the type style commands such as `\textbf{...}`, `\textsc{...}` etc. directly in your document body.

## 6 Sectioning and table of contents

In our `main.tex` file, we started off with a line saying `\section{Introduction}`. This command prints a section header with the title *Introduction*. But apart from that,  $\text{\LaTeX}$  keeps track of every section title and on which page it starts for later to use in the table of contents.  $\text{\LaTeX}$  also automatically numbers sections (which can be referenced, cf. Section 9) and updates these numbers if sections are added or deleted.

There are also commands for subsections and sub-subsections, chapters and parts. They all have an internal level: sectioning commands with higher levels are embedded in sectioning commands with lower levels (so a chapter, with level 0, can contain sections, with level 1; sections cannot contain chapters, etc.). If necessary, you can also define your own sectioning command on a new level (for instance if you for some reason need a sectioning command that can contain parts). If you would like to adjust the style associated with a particular sectioning command, have a look at the package `titlesec`.

Command	Level
<code>\part{...}</code>	-1
<code>\chapter{...}</code>	0
<code>\section{...}</code>	1
<code>\subsection{...}</code>	2
<code>\subsubsection{...}</code>	3
<code>\paragraph{...}</code>	4
<code>\subparagraph{...}</code>	5

Table 5: Sectioning commands

Then to make a table of contents, just type `\tableofcontents` wherever you want it printed (for instance directly after your `\maketitle`). A dedicated table of contents for figures and tables can also be instantiated using the commands `\listoffigures` and `\listoftables`: more on figures and tables in Sections 8 and 10.

Two more things. Sectioning commands can take an optional argument (so between the command and the curly brackets) which is the title of the section (or chapter etc.) as it should be printed in the table of contents. Then, if you don't want a section (or chapter etc.) to receive a number, you can put a `*` between the command and the curly brackets. However, this will also make that it is not in the table of contents!<sup>12</sup> So:

<sup>12</sup>Given the fact that anything is possible in  $\text{\LaTeX}$ , if you don't want a section a number but you do want to have it appear in the table of contents, it is possible using the command `\addtocontents{toc}{...}`.

```

\section{A}
This is section 1, called A both here and in the TOC.

\section[B]{C}
This is section 2, called C here, but B in the TOC.

\section*{D}
This is an unnumbered section, called D.
It won't show up in the TOC.

```

#### Exercise 11

Play around with sectioning commands and the table of contents a bit to acquaint yourself with it.

## 7 List environments

L<sup>A</sup>T<sub>E</sub>X provides three kinds of lists out of the box, but you can define your own lists if you want to.

**Ordered lists** (numbered lists) are introduced by the `\enumerate` environment, **unordered lists** (bullet point lists) by the `\itemize` environment. You can embed these environments to create sublists.

Items inside list environments are introduced with `\item`. By default, unordered lists use a variety of bullet points, horizontal lines and asterisks for various levels, while ordered lists use Roman/Arabic numerals. For ways to configure this, see the [Overleaf tutorial](#) and the `enumitem` package for more details.

You can also insert an arbitrary symbol for use in a particular item by passing it as an optional argument to the `\item` command. See below for an example.

```

\begin{enumerate}
 \item The first item in an ordered list.
 \begin{itemize}
 \item The second item in an unordered sublist.
 \end{itemize}
 \item The third item. Nothing special.
 \item[\textbf{€}] The fourth item with a custom symbol.
\end{enumerate}

```

1. The first item in an ordered list.
  - The second item in an unordered sublist.
2. The third item. Nothing special.
- € The fourth item with a custom symbol.

Lastly, the **description** environment deserves special attention. It is used to create a list of terms and their definitions. It is introduced by the `\description` environment, and each item is introduced by `\item[<term>]`. The term is printed in bold, followed by the definition. For example:

```
\begin{description}
 \item[Term 1] This is the definition of term 1.
 \item[Term 2] This is the definition of term 2.
\end{description}
```

**Term 1** This is the definition of term 1. This is a longer sentence to show how text wraps around the definition and is indented relative to the term and the following item.

**Term 2** This is the definition of term 2.

#### Exercise 12

Try to recreate the following list:

1.  $\text{\LaTeX}$  is fun,
2. and so is ice cream.  
    ? Is this true?  
    ! I don't know.
3. Ice cream has the following components.  
    **Milk** contains lots of calcium.  
    **Sugar** is bad for your teeth. Floss regularly.

## 8 Images and figures

You can import images using the `graphicx` package that Overleaf has already loaded in for us.

#### Exercise 13

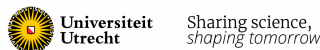
Upload an image file of your choice with the `Upload` button in the top left of your screen. Supported image files are PDF, JPEG and PNG (with `graphicx` at least).

Once you have uploaded your file, `\includegraphics{...}` can be used to insert it. If the image is a bit too big or too small, you can adjust its scale, width and height as an optional argument (see below). Most often, you will want to define the width or size of a picture relative to the text size (or `\textwidth`),



so that the picture size changes along with the text dimensions.<sup>13</sup> Invoking the following command prints Utrecht University's logo to half the width of the text:

```
\includegraphics[width=.5\textwidth]{uu-logo.png}
```



As you can see, this image is not aligned to the middle of the page. It also lacks a caption and we cannot currently reference it. For these reasons, images are commonly placed in a `figure` environment. A figure is a so-called **float**, a bit of content that L<sup>A</sup>T<sub>E</sub>X can move around to determine its optimal placement within the document. Figures are numbered automatically and a list of figures can be printed with the command `listoffigures`. You would use a figure environment as follows:

```
\begin{figure}[ht]
 \centering
 \includegraphics[width=.5\textwidth]{assets/uu-logo.png}
 \caption{This is the university's logo.}
\end{figure}
```

The `ht` will have L<sup>A</sup>T<sub>E</sub>X try to print the figure ‘here’ (`h`), and if that is not possible to print it at the top of the page (`t`). Other possible positioning options are `b` (bottom of the page) and `p` (which will print the image on a dedicated page for figures). You can insist on a particular positioning option by adding an exclamation mark, e.g. `[h!]`.

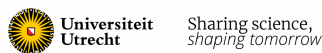


Figure 1: This is the university's logo.

#### Exercise 14

Upload an image from your computer and place it in your document using the `figure` environment. Also try a few positioning options and try adjusting the size of the image.

<sup>13</sup>For scaling pictures relative to their original size, use `[scale = 0.5]` or similar measurements.

## 9 References

You can reference sections, chapters, tables, figures, equations, items in a list etc. from anywhere in your document by giving them a label. For example, this section was referenced earlier on, and in order to do that we gave this section a label:

```
\section{Referencing sections and figures}\label{secrefs}
```

We can reference its number (9) using `\ref{secrefs}`. For sectioning commands and items in a list, just put the label anywhere inside the section or item. Tables, figures and equations you can give a label inside the environment. You can also reference the page on which something is or starts using `\pageref{...}`.

If you want your references to be clickable in the final PDF document, import the `hyperref` package in your preamble.<sup>14</sup>

```
This section starts on page \pageref{secrefs}.
```

This yields:

This section starts on page 18.

L<sup>A</sup>T<sub>E</sub>X automatically updates all of these references if you decide to remove or add a table, section or figure, saving you a lot of tedious updating.

### Exercise 15

Recreate (in sans-serif): ‘The logo of our university is displayed in Figure 1 on page 17.’ using `\label{...}`, `\ref{...}` and `\pageref{...}`. Make sure that the links are clickable.

## 10 Tables

Tables in L<sup>A</sup>T<sub>E</sub>X take some getting used to and introduce a couple of new concepts, but they are not very difficult. An example table is produced below, cf. Table 6.

Country	Capital	Inhabitants
Netherlands	Amsterdam	17.02 M
Germany	Berlin	82.67 M

Table 6: A table about two countries.

---

<sup>14</sup>Note that `hyperref` is one of the few packages that cares about the order in which it is imported. Try to import it as the *last* in your preamble, as it redefines a lot of commands.

```

\begin{table}[ht]
 \centering
 \begin{tabular}{|r|cl|}
 \hline
 Country & Capital & Inhabitants \\
 \hline
 Netherlands & Amsterdam & 17.02 M \\
 Germany & Berlin & 82.67 M \\
 \hline
 \end{tabular}
 \caption{A table about two countries.}
 \label{tblcountries}
\end{table}

```

There are many new elements here that we have not seen before. Let's go over them one by one.

`\begin{table}...\end{table}` creates a **float** environment, just like **figure** above.

`\begin{tabular}...\end{tabular}` contains the actual table.

`{|r|cl|}` specifies the alignment of the columns and the presence of vertical lines. The `r` means that the first column is right-aligned, the `c` that the second column is centred, and the `l` that the third column is left-aligned. The vertical lines are specified by the `|` symbols. So, there are vertical lines in the table on the left, between the first and second column and on the right.

`\hline` prints a horizontal line.

`&` separates columns.<sup>15</sup>

`\\` separates rows. These are easy to forget, so make sure you have added them after every row!

`\caption{...}` adds a caption to the table. This can also be put right after `\begin{table}` if you prefer your caption to appear above the table.

Overleaf provides handy tools to create tables in a more visual way without the need to remember all of this. However, the explanation of what the code does is still provided here for future reference.

#### Exercise 16

Try to reproduce the following (right-aligned) table.

<sup>15</sup>In the example here, the `&`s are all neatly aligned. This is recommended to keep your code readable, but not required. By default, `LATEX` will ignore more than one space.

Name	Inaugurated
Barack Obama	2009
Donald Trump	2017
Joe Biden	2021

Table 7: US Presidents

## 10.1 Advanced tables

### 10.1.1 booktabs

The `booktabs` package provides three simple commands that are alternatives to the `\hline` command for creating horizontal lines in your table: `\toprule` (for the top line of your table) `\midrule` (after the header) and `\bottomrule` at the bottom of your table). The results can be seen in Table 8 (below). Vertical lines do not go well in combination with the horizontal lines drawn by `booktabs` (try it out and see why!), but many typographical style guides advise against using vertical lines in tables anyway.

### 10.1.2 Merging cells and columns

You can merge cells in a table with the command `\multicolumn`.

```
\multicolumn{number of columns}{alignment}{text}
```

The first argument specifies the number of columns to merge, the second the alignment of the text in the merged cell, and the third the text to put in the merged cell.

In order to merge rows, you will need the package `multirow`, which provides the command `\multirow`.

```
\multirow{number of rows}{width}{text}
```

The first argument specifies the number of rows to merge, the second the width of the cell (use `*` for the natural width of the text), and the third the text to put in the merged cell.

An example involving both `multirow` and `multitable` is presented here, cf. Table 8.

Vegetables	Verdict	Price
Onion	Tasty	€ 1.00
Carrot	Yummy	€ 1.50
Tomato		€ 2.00
Parsnip	Not tried yet.	

Table 8: My vegetable garden

```

\begin{table}[ht]
\centering
\begin{tabular}{lll}
\toprule
\textbf{Vegetables} & \textbf{Verdict} & \textbf{Price} \\
\midrule
Onion & & € 1.00 \\
Carrot & & € 1.50 \\
Tomato & & € 2.00 \\
Parsnip & & Not tried yet. \\
\bottomrule
\end{tabular}
\caption{My vegetable garden}
\label{tblmultirow}
\end{table}

```

## 11 Bibliography management

L<sup>A</sup>T<sub>E</sub>X can save you a lot of problems when managing your bibliography and citing references. The available tools are highly customisable and powerful, so we will only be scratching the surface here, using the BIBL<sup>A</sup>T<sub>E</sub>X package.<sup>16</sup>

All of your bibliographical references are stored in a file with the `.bib` extension. An example for such an entry is given here.

```

@article{lander1966counterexample,
 title={Counterexample to {E}uler's conjecture on sums of
 like powers},
 author={Lander, L.-J. and Parkin, T.-R. and others},
 journal={Bulletin of the American Mathematical Society},
 volume={72},
 number={6},
 pages={1079},
 year={1966}
}

```

In this bibliography entry, we define with `@article` that the reference is an article. There are many different entry types, including `@book`, `@phdthesis` and `@unpublished` for papers that are yet unpublished. What follows is the **reference key**: `lander1966counterexample`, which is used to cite this publication in your text.<sup>17</sup> Make sure your `.bib` file does not contain any duplicate keys. Then we define the **metadata**, such as title, author and the name of the

<sup>16</sup>Other bibliography packages for L<sup>A</sup>T<sub>E</sub>X you might come across are `natbib` and `bibtex`.

<sup>17</sup>You can pick whatever you want as your key, but it is common practice to use a combination of the last name of the first author, the year of publication and one or two keywords from the publication title.

journal the article was published in.<sup>18</sup> Different entry types can have different metadata: check [this page for an overview \(Wikibooks\)](#). It is okay not to include all metadata: if you don't know the volume of the journal it was published in, simply leave it out the `.bib` file, BIB<sub>La</sub>T<sub>E</sub>X takes care of it when citing it.

#### Exercise 17

Create a new file in Overleaf, call it `bib.bib` and copy the reference above into this file.

Now we need to import BIB<sub>La</sub>T<sub>E</sub>X, and L<sub>A</sub>T<sub>E</sub>X needs to be made aware of your brand new bibliography file. We do this by adding the following lines to our preamble.

```
\usepackage[style=authoryear,backend=biber]{biblatex}
\addbibresource{bib.bib}
```

The optional argument `[style=authoryear]` tells L<sub>A</sub>T<sub>E</sub>X which bibliography style to use. Other options are `apa`, `numeric`, among many others. Have a look at [this page](#) for a list.<sup>19</sup>

#### Exercise 18

Go to Google Scholar, search for *vertegaal filling in the facts*, and get the BIB<sub>La</sub>T<sub>E</sub>X citation by clicking on the two quotes underneath the first hit. Click the `BibTeX` button to obtain the reference. Copy it into your `.bib` file.

Now that BIB<sub>La</sub>T<sub>E</sub>X is fully set up, we can start citing our sources! The commands you are most likely to use are the following, cf. Table 9.

Command	Result
<code>\cite[20]{vertegaal2017filling}</code>	Vertegaal, 2017, p. 20
<code>\parencite[20]{vertegaal2017filling}</code>	(Vertegaal, 2017, p. 20)
<code>\footcite[20]{vertegaal2017filling}</code>	<sup>20</sup>
<code>\textcite[20]{vertegaal2017filling}</code>	Vertegaal (2017, p. 20)

Table 9: Citing commands

<sup>18</sup>Note the use of the comma to separate last from first names, and of `and` to separate entire names. If you end with `and others`, BIB<sub>La</sub>T<sub>E</sub>X will automatically replace this with *et al.* if your style requires it. Also note the use of `~` to tell L<sub>A</sub>T<sub>E</sub>X not to break the line at the space between the authors' initials.

<sup>19</sup>The optional argument `[backend=biber]` tells L<sub>A</sub>T<sub>E</sub>X which program to use to compile your bibliography. `biber` is the most recent and powerful program, but `bibtex` is still widely used.

<sup>20</sup>Vertegaal, 2017, p. 20.

### Exercise 19

Cite pages 250–253 in Vertegaal 2017.

When citing multiple references at once, just pass multiple reference keys as arguments at the same time, separated by a comma (no space!). If you want to add things like “see” or “pp. 48–49”, you can do that as optional arguments:<sup>21</sup>

```
\parencite[see][pp.~48--49]{lander1966counterexample}
```

yields

(see Lander, Parkin et al., 1966, pp. 48–49)

Note the use of `~` to prevent L<sup>A</sup>T<sub>E</sub>X from breaking the line at the space between the page numbers.

Finally, to print your bibliography, just type `\printbibliography` wherever you want your bibliography to be printed.

### Exercise 20

Find a reference on Google Scholar of a paper or book that you like, and add it to `bib.bib`. Refer to it elsewhere in your document. Change the reference style to `[style=authoryear-icomp]` and see what happens.

## 12 Troubleshooting

### 12.1 Interpreting errors

When compiling your document, you may run into *errors* or *warnings* produced by the L<sup>A</sup>T<sub>E</sub>X compiler. Overleaf tells you exactly how many error there are: the button next to the **Recompile** button is the log—the number on that is the amount of errors. If it’s red, there are errors, if it’s yellow, there are only warnings.

**Errors** are fatal mistakes in your code, causing L<sup>A</sup>T<sub>E</sub>X to crash without fully compiling your document. These should always be fixed. **Warnings** are less serious; they are meant to indicate that the final result of compilation may look different from what you intended. Always check them out, but sometimes they can be ignored. A good tip: you can click on an error to jump to the offending line in your code.

But what do these error messages actually mean? Usually the log messages give you a clue what is going on, so reading those really helps. One frequently recurring warning **Overfull hbox** is dealt with in more detail below.

<sup>21</sup>Helpful (if pedantic) tip: typing `--` (two hyphens) creates an *en dash*, the width of ‘N’, used for inclusive ranges such as page numbers or dates: ‘5–12 June’. Three hyphens—that is `---` without spaces—create a so-called *em dash*, commonly defined as the width of ‘M’. This is used for parenthetical remarks, as in the previous sentence.

If there are errors that you don't understand, do check [the Overleaf tutorial page on errors](#), or enter your question into Google. In 90% of the cases, you are not alone. Usually, one of the first hits will lead you to [StackExchange](#), which is where L<sup>A</sup>T<sub>E</sub>X users can ask questions to experts and L<sup>A</sup>T<sub>E</sub>X developers. If you cannot find your answer there, post a question, and you will most likely have your answer within 24 hours. (Read the [guidelines](#) first before you do, however.)

For questions about specific packages, be sure to read the so-called **package documentation** (the package's manual and description, so to say) first. You can easily find that using a Google search (e.g. 'graphicx documentation'). This will almost always bring you to the CTAN website (see Section 2.2), where most L<sup>A</sup>T<sub>E</sub>X packages are hosted.<sup>22</sup>

## 12.2 Over- or underfull hboxes

By default, L<sup>A</sup>T<sub>E</sub>X is quite fussy when it comes to fitting text on a line. Sometimes it does not manage to uphold its standards, and it will tell you so in the log, with a message like `Overfull \hbox (51.48561pt too wide) in paragraph at lines 25--29..` Since these warnings are such a common source of frustration, they are treated in more detail here.

An `Overfull \hbox` error indicates that L<sup>A</sup>T<sub>E</sub>X has had to stretch the space in between words and sentences beyond its preset limits ('underful hbox') or that it could not fit all words on one line, leaving some to stick out into the paper margin ('overfull hbox').<sup>23</sup>

Not all of these errors are disruptive or 'ugly', so don't panic and always check them out before submitting anything important to a journal or your supervisor.

If you are keen on solving these errors, there are a few things you can do. Here they are, in order of increasing desperation.

1. Import the `babel` package and `microtype` packages. Also when you do not have any `hbox` errors, these are recommended. `Babel` loads hyphenation settings for specific languages, and `microtype` stretches the kerning and character widths ever so slightly based on a given font's properties. These are often just enough to fix a nasty `hbox` warning.
2. If L<sup>A</sup>T<sub>E</sub>X still cannot break off a long word, help it along by indicating where the potential breaks are, using `\-`. For example, we can teach L<sup>A</sup>T<sub>E</sub>X how to break the Dutch word *angstschreeuw* by writing it as `angst\-\schreeuw`.<sup>24</sup>
3. Try to rephrase. By moving words around, L<sup>A</sup>T<sub>E</sub>X can often find a better way to fit everything on the line.

---

<sup>22</sup>More technical users may also type `texdoc graphicx` in their terminal to open the documentation.

<sup>23</sup>The term *hbox* stands for 'horizontal box' and refers to the line/box L<sup>A</sup>T<sub>E</sub>X attempts to fill.

<sup>24</sup>You can also define hyphenation rules for the entire document, by adding the command `\hyphenation{angst\-\schreeuw}` to the preamble.



4. Set a higher `emergencystretch`. This increases the amount of whitespace  $\text{\LaTeX}$  can add to a line before it is considered ‘overfull’. For example, `\setlength{emergencystretch}{3em}` will allow  $\text{\LaTeX}$  to add 3em of whitespace to a line before it is considered ‘overfull’. Use the smallest value that solves your problem, and be sure to wrap this in a `\begin{group}` and `\endgroup` command, so that the change is only temporary.
5. Switch to sloppy mode using `\sloppy`. This will allow  $\text{\LaTeX}$  to be very loose with its placement and generally leads to ugly spacing, so only use this if you are really desperate!

### 12.3 Where to go for help?

Getting good at  $\text{\LaTeX}$  is a skill you simply have to train. A personal tip to improve your confidence is to take an existing document and convert it, as best as you can, to  $\text{\LaTeX}$ . This will force you to look up a lot of things, and you will learn a lot in the process.

If you would like us to help you with your code, do not hesitate to contact our team of developers at the [Centre for Digital Humanities](#) or come in during one of our [walk-in hours](#). We are always happy to help!

Remember that while  $\text{\LaTeX}$  has a steep learning curve, it is a very powerful tool that will save you a lot of time and frustration in the long run. The question ‘Can I do X in  $\text{\LaTeX}$ ’ is almost always to be answered with ‘Yes, you can!’. It is just a matter of finding out how.