

L^AT_EX Quick- Start Guide

Monterrey, N.L., Mexico

Daniel Celis Garza

October 15, 2015

ACKNOWLEDGEMENTS

First, I would like to thank Prof. Marcelo Fernando Videau Vargas for introducing me to L^AT_EX via his fearsome exams.

Furthermore, there are a couple of organisations which made my learning possible such as: Stack Exchange's L^AT_EX forum, <http://tex.stackexchange.com/>; the Wikimedia Foundation's L^AT_EX wikibook <https://en.wikibooks.org/wiki/LaTeX>; Reddit's L^AT_EX sub-reddit, <https://www.reddit.com/r/latex>; and the Comprehensive T_EX Archive Network, <https://www.ctan.org/>. This little guide wouldn't have been possible without their vast stores of knowledge.

DISCLAIMER

The purpose of this guide is not to replace the resources provided by the organisations in the acknowledgements; but to save the reader—presumably new \LaTeX users—the hassle of sifting through them, finding little and understanding less.

Contents

List of Tables	vii
List of Figures	viii
I Introduction to L^AT_EX	1
1 Introduction to L^AT_EX	2
1.1 Quick Description	2
1.2 Getting Started	2
1.3 Best Practices	3
1.4 Special Writing Characters	4
1.5 Reserved L ^A T _E X Characters	4
2 L^AT_EX Features	6
2.1 Preamble	6
2.1.1 Declaring a Document Class	6
2.1.2 Package Declaration	7
2.2 Environments and Commands	8
2.2.1 Font Commands	8
2.2.2 Include and Input Commands	10
2.3 Spacing	10
2.4 Counters	11
2.5 Table of Contents, Appendix, Index & Glossary	12
2.5.1 Table of Contents	12
2.5.2 Appendix	13
2.5.3 Index	13
2.5.4 Glossary	13
2.5.5 User-Defined Commands	14
2.6 Referencing	15
2.7 Non-English Documents	16
2.8 Bibliography	16
2.8.1 Manual	16
2.8.2 B _I B ^A T _E X	17

II General L^AT_EX Documents	18
3 Creating a Document	19
3.1 Titular Information	19
3.2 Paging	20
3.2.1 Page Format	20
3.3 Document Environment	21
3.4 Document Sectioning	21
4 Non-Maths Environments	22
4.1 List Structures	22
4.1.1 Itemize	22
4.1.2 Enumerate Environment	23
4.1.3 In-line Lists	24
4.2 Verbatim Environments	24
4.2.1 Verbatim and Alltt	24
4.2.2 Comment Environment	24
5 Float Environments	25
5.0.1 Figures	26
5.0.2 Subfigures	27
5.1 Tables	29
5.1.1 Multicolumn and Multirow Tables	30
5.1.2 Multipage Tables	31
6 Mathematical Environments	33
6.1 Maths and Display Maths	33
6.2 Common Symbols and Functions	34
6.2.1 Greek Maths Symbols	34
6.2.2 Common Binary Operations	34
6.2.3 Common Relation Symbols	34
6.2.4 Arrows	34
6.2.5 Common Miscellaneous Symbols	35
6.2.6 Common Functions	35
6.2.7 Common Variable-Sized Symbols	36
6.2.8 Non-trivial Delimiters	36
6.2.9 Common Mathematical Accents	36
6.2.10 Limits and Substack Commands	37
6.2.11 Scaled Delimiters	37
6.2.12 Common Maths Typefaces	38
6.3 Align	39
6.3.1 Subequations	39
6.4 Matrices	40
6.4.1 Aligned and Array	40
6.5 Cases	41

7 Referencing and Citing	43
7.1 Referencing	43
7.2 Bibliography	43
7.2.1 Manual	44
7.2.2 B <small>B</small> T <small>E</small> X	44
III L<small>A</small>T<small>E</small>X Presentations	46
IV Conclusions	47
8 Closing Remarks	48

List of Tables

1.1	<code>L^AT_EX distributions.</code>	3
1.2	<code>Special writing characters.</code>	4
1.3	<code>Table of special L^AT_EX characters, their escape method, and function.</code>	5
2.1	<code>Common font styles.</code>	9
2.2	<code>Pre-defined font sizes.</code>	9
2.3	<code>Spacing commands.</code>	10
2.4	<code>Counter styles.</code>	11
2.5	<code>Sophisticate indexing.</code>	13
3.1	<code>Common paging commands.</code>	20
3.2	<code>Common default page styles.</code>	20
5.1	<code>Table with placement specifiers and their description.</code>	25
5.2	<code>Options and their descriptions for the \includegraphics command.</code>	26
5.3	<code>Poorly designed and formatted table.</code>	30
5.4	<code>Multirow and multicolumn table.</code>	31
6.1	<code>Examples of maths greek letters.</code>	34
6.2	<code>Common binary operations.</code>	35
6.3	<code>Common relation symbols.</code>	35
6.4	<code>Arrows.</code>	35
6.5	<code>Common miscellaneous symbols.</code>	35
6.6	<code>Common variable-sized symbols.</code>	36
6.7	<code>Non-trivial delimiters.</code>	36
6.8	<code>Common mathematical accents.</code>	37
6.9	<code>Effect of the \limits command.</code>	37
6.10	<code>Common mathematical typefaces.</code>	39
6.11	<code>Matrix types.</code>	40

List of Figures

5.1 Pictures of memes.	28
--------------------------------	----

Part I

Introduction to L^AT_EX

Chapter 1

Introduction to L^AT_EX

1.1 Quick Description

L^AT_EX (written LaTeX) is an open-source document typesetting language originally created as T_EX by renowned mathematician and computer scientist, Donald E. Knuth, as a means of incorporating the best typesetting practices of academic—predominantly mathematical—journals, in an easy to use package.

T_EX has now been expanded into L^AT_EX—which wraps around many primitives and makes the language more accessible. What started as an academic typesetting language for mathematicians has expanded its scope to include all three basic sciences, computer science, engineering, writing, and even music. It's an extremely flexible tool that can produce all kinds of documents from personal letters to beamer presentations, and everything in between.

The language is characterised by its modularity, flexibility, and ability to produce beautiful, standardised, and reproducible documents. Admittedly, the learning curve can be very steep—especially if one is pressed for time—but the end result is *very* worth it.

L^AT_EX is not a word processor, it's a typesetting language. As such, every L^AT_EX document is comprised of two large sections:

1. Preamble: Where the user loads relevant modules, *packages*, which define a document's characteristics and provide functionality via *backslash commands* (commands for short) and *environments*.
2. Body: Contains the document's actual content.

1.2 Getting Started

There are several freely available distributions depending on your OS, as shown in table 1.1. Since L^AT_EX is a language rather than a word processor, documents can be produced in any plain text editor and compilation carried out via console command. However, this is inefficient and there are better alternatives:

Table 1.1: \LaTeX distributions.

Distribution	OS
\TeX Live	Linux
Mac \TeX	OS X
Mik \TeX & Pro \TeX t	Windows

- Vim \LaTeX : a Vim module for writing \LaTeX documents.
- Emacs/XEmacs + AU \CTeX + Ref \TeX : modules for writing \LaTeX documents on Emacs/XEmacs.
- Dedicated editor: there are a variety of dedicated editors, some of which come with different distributions. $\text{\TeX}works$ for Mik \TeX , $\text{\TeX}Shop$ for Mac \TeX and $\text{\TeX}Studio$ for Pro \TeX t. My personal favourite is $\text{\TeX}Studio$, because it's 1) lightweight, 2) easy to use, 3) compatible with Windows, Linux and OS X, and 4) comes with a lot of quality of life and advanced features; such as automatic quotes, user-defined compilation trains, wizards, macros, folding, etc.

Being a non-interpreted programming language, \LaTeX codes generate documents upon successful compilation. Compiling is done via console commands or keyboard shortcuts. All editors provide their own defaults, but they're also reconfigurable. There is also something known as a *compilation engine*, of which there are a few. The two most modern are $\text{Xe}\text{\LaTeX}$ (or $\text{Xe}\text{\TeX}$), and $\text{Lua}\text{\LaTeX}$ (or $\text{Lua}\text{\TeX}$). The standard is still $\text{PDF}\text{\LaTeX}$ which does not provide full unicode encoding. All the distributions mentioned above come with all three (and more) engines. The default compilation engine can be specified within your editor of choice. This document was created with $\text{Xe}\text{\LaTeX}$, because it is more mature (less buggy, more features) than $\text{Lua}\text{\LaTeX}$, and more competent than $\text{PDF}\text{\LaTeX}$. Which means you have to use $\text{Xe}\text{\LaTeX}$ if you want to compile the source code.

A file with a `.bib` bibliography must also be run through the $\text{BiB}\text{\TeX}$ engine, which also comes in all aforementioned distributions and can be done within your editor of choice. Further details provided in subsection [7.2.2](#).

1.3 Best Practices

Code can get ugly pretty quickly, especially when done wrong. So here is a checklist of things that will make your life easier.

- Indent sensibly. \LaTeX does not interpret manual indents as spaces, that's what the spacing commands are for.
- Break apart troublesome lines. A single line break will not interrupt text flow unless a blank line is added between segments. This helps a lot when debugging and writing long equations.

- Divide and conquer. Use information theory ($\log_2 x$) to optimise your debugging. Divide things progressively in half (comment out segments), narrowing down your search by halving your search space with each iteration.
- Structure your approach to equation writing. Simplify equations as much as possible. Typeset small, independent sections and incorporate them where they are needed *after* they have been checked for mistakes. Repeat process until the complete equation has been typeset.
- Use sensible labels. I like to use “p”–part, “c”–chapter, “s”–section, “sb”–subsection, “ssb”–subsubsection, “f”–figure, “sf”–subfigure, “t”–table, “e”–equation, “se”–subequations. Followed by a colon and a short but descriptive name.
- Automate as much as possible. Define user commands for things you will use often.
- Customise your editor of choice. Take time to set up your theme, shortcuts, custom macros, auto-complete, compilation train, flags, dictionary, etc. Make sure you learn your most used shortcuts, macros, and commands.

1.4 Special Writing Characters

There exist special writing characters which have specific, non-trivial \LaTeX characters. Table 1.2 contains some of the most common ones, their \LaTeX character, and usage.

Table 1.2: Special writing characters.

Name	Print	Code	Usage
English Half-Quotation	“	‘’	Nested quotation.
English Quotation	””	‘‘’’	Speech or technical term.
Hyphen	-	-	Compound words.
En Dash	—	--	Ranges, scores, conflict/connection.
Em Dash	—	---	Cursory explanation, no pre/post-space.
Double Em Dash	—	-----	Interrupted word, no pre-space.
Ellipses	...	ldots	Pause, pre/post-space optional (format).

1.5 Reserved \LaTeX Characters

Like many programming languages, \LaTeX contains certain reserved characters used for different native processes. Table 1.3 shows the symbol, its escape method (how to print them), and function.

Table 1.3: Table of special L^AT_EX characters, their escape method, and function.

Symbol	Escape	Function
%	%	Comment.
\	<code>textbackslash</code>	Escape other characters, initiate macro.
&	&	Align delimiter.
~	or \$ \sim \$	Non-breaking space. It's a character, not a space.
#	#	Macro numbered argument.
\$	\$	Maths environment (<i>maths</i>).
\$\$	\\$\\$	Display maths environment (\$\$ <i>maths</i> \$\$).
^	\char`^	Superscript (maths environment).
-	_	Subscript (maths environment).
\\	<code>textbackslash</code> (x2)	Line break.
{}	\{\}	Argument.

Chapter 2

L^AT_EX Features

2.1 Preamble

The preamble is where a document's overall characteristics are defined. Things such as required packages, page setup and layout, user-defined and user-edited macros, as well as titular information are all defined in the preamble.

2.1.1 Declaring a Document Class

The first line of the preamble is where the user declares the document type, default font size, and paper size.

It is well known that only barbarians utilise non-International Standard Paper Sizes (ISO 216 & 269)¹. According to the all-knowing and less-wrong-than-encyclopaedia-britannica, wikipedia.org [? ? ?], Mexico is one such barbaric place. One can be an enlightened rebel—and use the A-series sizes—or a barbaric conformist and use non-standard sizes like so:

```
\documentclass[12pt,letterpaper]{article}
```

There are many document classes but the most common ones are: 1) `article`, 2) `report`, 3) `book`, 4) `letter`, and 5) `beamer`. These provide the overall document format. In some cases, they also automatically load certain packages and provide custom commands and/or environments. The most obvious example of this is the `beamer` class, which automatically loads the `hyperref` package and provides custom commands and environments such as `frametitle` and `frame`, respectively.

The following are common options of the `documentclass` command, default values appear in **bold**:

- Equation numbering (left or right): `leqno`, `reqno`.
- Document language: `german`, `greek`, `spanish`, `english`. Can be overridden with the `babel` and `polyglossia` packages.

¹https://en.wikipedia.org/wiki/Paper_size

- Page orientation: `landscape`, `portrait`.
- Text columns: `twocolumn`, `onecolumn`.
- Font size: Any font size, `12pt`.
- Paper size: Any official ISO and non-ISO paper size as well as a custom paper size override, `letterpaper`.

Modern L^AT_EX editors have wizards that automatically declare document class and load essential packages via graphical user interface (GUI). They let the user decide which common options and packages they want to load without actually having to write all the code and memorise all relevant options.

2.1.2 Package Declaration

After declaring a document class, users must declare all relevant packages. This is done as follows:

```
\usepackage[options]{name}
```

The order in which some packages are loaded may affect their behaviour as dependencies need to be loaded before the packages that use them. A package's load order and dependencies are often specified either in its documentation and/or wikibook entry.

Some of the most used packages include:

- `babel`: sets punctuation (hyphenation), part, chapter, section, subsection, subsubsection, figure, subfigure, table, etc. to the language of your choice. `polyglossia` is the X^EL^AT_EX alternative, but `babel` still has more regional options such as `[spanish,mexico]`.
- `fontspec`: loads utf-8 encoded fonts, natively supports accents—even in `.bib` files—and works on X^EL^AT_EX and Lua^TE_X. It replaces PDFL^AT_EX's `inputenc` and `latin` packages; both of which require options in order to use unicode characters, and have trouble with accented unicode characters in `.bib` files.
- `xcolor`: provides tons of colour customisation and declaration, as well as a wide variety of predefined colours. Contains many options which expand the default colour palette.
- `hyperref`: provides hyperlinks for references, footnotes, citations, and URLs. Contains various options which allow the user to customise their behaviour and appearance.
- `geometry`: provides page layout specifications for customisable margins and paper sizes.

This document contains more packages in `preamble.tex`, read the comments for more information. Further information on packages and L^AT_EX in general can be found on the web, particularly at the L^AT_EX wikibook², T_EX Stack Exchange³, and Sub-Reddit⁴.

2.2 Environments and Commands

A big part of L^AT_EX's appeal is the fact that it uses explicit environments and commands which are loaded by the document class and packages declared in the preamble. Environments are sections of code with special rules, and may contain arguments and/or options. In contrast, commands⁵ have a certain function, but do not change the rules of a section of code. They may also contain options and/or arguments. It's worth mentioning that not all environments and commands need or even use options and/or arguments. Options are always found within chevrons `[options]`, while arguments within braces `{arguments}`, options *always* go before arguments.

Environments are specified as:

```
\begin{environment}  
...  
\end{environment}
```

while commands are simply written as `command`. Options and arguments—if needed—are added immediately after declaration. Commands also gobble one space immediately to their right, avoid this by adding a nonbreaking space, `~`, between it and the following text, e.g. `command text`. Punctuation can be placed immediately following a command without problem.

Some commands and environments have *starred* versions, where an asterisk, `*`, is placed immediately after their name `command*`, `beginenvironment*` and `endenvironment*`. Starred versions change to an alternate behaviour, usually something to do with numbering or indexing.

2.2.1 Font Commands

There are a myriad of fonts and font types so only the most common ones will be included here.

The standard L^AT_EX font is the famed Computer Modern font family, which despite its superiority over plebian fonts, still has its minor flaws. Which the Latin Modern font family was designed to fix. Computer Modern is still the default font in most L^AT_EX distributions, so Latin Modern must be loaded with the `\modern`

²<https://en.wikibooks.org/wiki/LaTeX>

³<http://tex.stackexchange.com/>

⁴<https://www.reddit.com/r/latex>

⁵In a way, packages are commands which load other commands.

package at the very end of the package declaration because it redesigns many fonts and symbols loaded by other packages.

There are also many font styles, the most common ones are found in table 2.1. Some of them can be nested to produce a combination of effects, but only if the combination exists within the document's font family.

Table 2.1: Common font styles.

Command	Result
<code>emphABCdef123</code>	<i>ABCdef123</i>
<code>textsfABCdef123</code>	ABCdef123
<code>textttABCdef123</code>	ABCdef123
<code>textitABCdef123</code>	<i>ABCdef123</i>
<code>textslABCdef123</code>	ABCdef123
<code>textscABCdef123</code>	ABCDEF123
<code>uppercaseABCdef123</code>	ABCDEF123
<code>textbfABCdef123</code>	ABCdef123

There are predefined font sizes whose absolute size varies depending on document class and default font size. They are declared as:

```
{\fontsizecommand <text or environment>}
```

Table 2.2 contains these commands and what each does in a report with 12pt font size.

Table 2.2: Pre-defined font sizes.

Command	Effect
<code>tiny</code>	ABCdef123
<code>scriptsize</code>	ABCdef123
<code>footnotesize</code>	ABCdef123
<code>small</code>	ABCdef123
<code>normalsize</code>	ABCdef123
<code>large</code>	ABCdef123
<code>Large</code>	ABCdef123
<code>LARGE</code>	ABCdef123
<code>huge</code>	ABCdef123
<code>Huge</code>	ABCdef123

Arbitrary font sizes can be selected by using:

```
{\fontsize{<size>}{<line space>}\selectfont <text or environment>}
```

where the size can be given as pt, or any properly abbreviated imperial or metric measure.

2.2.2 Include and Input Commands

`input` and `include` are two *very* special and useful commands. They take a `.tex` file as argument without extension.⁶

`inputfilename` is very low level, and it simply inserts `filename.tex`'s content into the `.tex` file where it is called. It can be nested, i.e. `input` files can be found within other `input` and `include` files, and be used in the preamble—which is useful when documents share core packages and configurations.

`includefilename` is a bit more complicated and *much* more useful when it comes to sectioning. This command forces a `clearpage` before and after inserting `filename.tex`'s content, so it's usually reserved for inserting chapters, as they already start on a new page. It also does some very convenient `TeX-fu` with `.aux` files, only compiling those whose corresponding `.tex` files have been modified, rather than the whole document's. This makes the first complete compilation noticeably slower than normal, but considerably hastens subsequent ones.

2.3 Spacing

Table 2.3: Spacing commands.

Command	Function
<code>indent</code>	Forces indentation.
<code>noindent</code>	Removes automatic indentation.
<code>hspace</code>	Add horizontal space in the desired units.
<code>vspace</code>	Add vertical space in the desired units.
<code>hfill</code>	Fill the page horizontally with space.
<code>vfill</code>	Fill the page vertically with space.
<code>~</code>	Non-breaking space. Treated as a letter rather than a space.
<code>,</code>	Thin space (maths and text).
<code>;</code>	Thick space (maths).
<code>:</code>	Medium space (maths).
<code>!</code>	Negative space (maths).
<code>quad</code>	Em dash of space.
<code>qqquad</code>	Two Em dashes of space.
<code>\\"</code>	Line break.
<code>\\"[]</code>	Line break with spacing in the desired units.

Note: Spacing units can be metric or imperial and use the standard abbreviated form. They can be negative and subtract spacing, or positive and add spacing.

One of the paramount things about `LATEX` is the emphasis on proper spacing. `LATEX` does not take Tab indents or single line breaks into account because they

⁶They also accept relative or absolute paths with the file.

help in making code readable and easier to debug, so there are specific spacing commands for that. Table 2.3 shows the most commonly used spacing commands and their function.

2.4 Counters

Counters are variables whose values increase whenever an event is triggered. There are standard counters for all sectioning commands, equations (all maths environments except in-line and display maths automatically increase it), floats (figure, table), paragraph, subparagraph, footnote, mpfootnote, and page. They are named after the object they keep track on—for example the figure counter is called `figure`, and the page counter, `page`. There are also counters for the `enumerate` and `itemize` environments. They cover the four default nesting levels, from first to last they are: 1) `enumi`, 2) `enumii`, 3) `enumiii`, and 4) `enumiv`.

Calling a counter is done with the command `\the\counter{name}`, which prints the counter as it is defined, and can be used to create custom formats with `\renewcommand` (more on renewing commands later). The command `\value{\counter{name}}` will return the counter's numerical value, which can be used in calculations and setting other counters to some related value.

Default counters are pseudo-numerical—except for footnote symbol counters. Table 2.4 contains the default counter styles. Setting a counter's style is done by typing `\counter{style}\counter{name}`, and has to be done within `\renewcommand` so the changes are global.

Table 2.4: Counter styles.

Command	Function
<code>\arabic{\counter{name}}</code>	Arabic numbering.
<code>\alph{\counter{name}}</code>	Lower case alphabetic.
<code>\Alph{\counter{name}}</code>	Upper case alphabetic.
<code>\roman{\counter{name}}</code>	Lower case roman numerals.
<code>\Roman{\counter{name}}</code>	Upper case roman numerals.
<code>\fnsymbol{\counter{name}}</code>	Footnote symbol.

As an example, say we want our pages to be numbered as follows `\meta{chapter number} - \meta{page number}`. We would need to add

```
\renewcommand{\thepage}{\thechapter-\arabic{page}}
```

where we want this page numbering style to begin.

Counters can also be manually modified. Stepping a counter by one is done with `\stepcounter{\counter{name}}`. Adding or subtracting an arbitrary number to a counter is done with `\addtocounter{\counter{name}}{number}`. Setting a counter to a given number is done with `\setcounter{\counter{name}}{number}`. Setting a counter in relation to some other counter is done with

```
\setcounter{<counter_name>}{\number\value{<other_counter_name>}}
```

They can also be created with the `newcounter` command. And counters which are reset to zero every time another counter is increased are created with `newcounter` `name[trigger counter]`. New counters must be created in the preamble.

2.5 Table of Contents, Appendix, Index & Glossary

If they are used within the beamer document class, remove all paging commands.

2.5.1 Table of Contents

Some of the most painful aspects of creating documents in word processors are this section's namesake. Fortunately, L^AT_EX has us covered. In fact adding a complete tables of contents (toc) is as easy as adding the following code where the user wants them to appear.

```
\newpage
\tableofcontents      % Table of contents.
\cleardoublepage
\phantomsection
% Add table index to the table of contents (toc).
\addcontentsline{toc}{chapter}{List of Tables}
\listoftables         % Table index.
\cleardoublepage
\phantomsection
% Add figure index to the toc.
\addcontentsline{toc}{chapter}{List of Figures}
\listoffigures        % Figure index.
```

The `tableofcontents` command prints the table of contents, `listoftables` prints a list of tables, `listoffigures` prints one for figures, and `addcontentsline` adds an entry to the table of contents as if it were a section type (part, chapter, section, subsection or subsubsection), under the desired name. The `cleardoublepage` and `phantomsection` commands ensure the correct page numbers are referenced. In the given example, the main table of contents prints the location of the lists of tables and figures as if they were chapters named “List of Tables” and “List of Figures”.

Anything can be added to the table of contents with the `addcontentsline` command, including appendices, indices, and unnumbered (starred) sectioning commands. Glossaries can be added with the `glossaries` package.

2.5.2 Appendix

Adding an appendix is as easy as writing `appendix` where the appendix is to be printed. It only needs to be added once, all subsequent sectioning commands will be formatted accordingly.

2.5.3 Index

Creating an index requires the `makeidx` package, adding the `makeindex` command in the preamble, and placing `printindex` where the index is to be printed. Index entries are specified with the `index` command. Table 2.5 shows how advanced index entries are specified. Accents cannot be added as unicode characters, and

Table 2.5: Sophisticated indexing.

Command	Function
<code>index{a}</code>	Plain entry.
<code>index{a!b}</code>	Sub entry under a.
<code>index{a}</code>	Plain entry.
<code>index{b@\textstyle{b}}</code>	Formatted entry.
<code>index{a!b@\textstyle{b}}</code>	Formatted sub entry under a.
<code>index{a \textstyle{b}}</code>	Formatted page number.
<code>index{a see {b}}</code>	Cross-reference with <i>see</i> .
<code>\index{a seealso{b}}</code>	Cross-reference with <i>see also</i> .

must instead be added as `<accent>` followed by the word to be indexed. Advanced index entries can be nested as needed.

There is a way to create more than one index, but for most purposes, a single one will suffice. Regardless, the indexing page of the `LATeX` wikibook⁷ contains more information on the matter.

2.5.4 Glossary

The glossary requires the `glossaries` package, and adding the `makeglossaries` in the preamble. Added functionality can be added by specifying the package options: `xindy`, which offers better indexing than `makeindex` (default); and `toc`, which adds the glossary to the table of contents. In order to make glossary entries hyperlinks, the package must be loaded *after* `hyperref`.

Defining a glossary term is done as follows:

```
\newglossaryentry{<label>}{  
    name = <name>,
```

⁷<https://en.wikibooks.org/wiki/LaTeX/Indexing>

```

symbol = <symbol>,
description = {<description>},
sort = <how to sort the entry>,
plural = <plural>
}

```

Where all entries but name are optional. If name and/or symbol require mathematical environments, they should be placed within ensuremath. The description can have in-line maths. The longnewglossaryentrylong description command should be used in cases where the description is over one paragraph long. It takes the same first and second arguments but the description is placed as the third, not as part of the second. Acronym definition is done with the newacronymlabelabbreviationfull word command. If acronyms require a different list, add the acronym option to the glossaries package.

Referencing glossary entries is done with glslabel, by adding symbol, desc and pl between gls and the argument, it will reference the symbol, description and plural respectively. By capitalising G in the normal and plural versions, the printed name will have its first letter capitalised. An entry can also be referenced with alternate text by using the command glslinklabelalternate text.

There is much more information in the documentation for the glossaries package and the L^AT_EX wikibook's glossary entry⁸.

2.5.5 User-Defined Commands

L^AT_EX is in fact *Turing complete*, meaning it has the full capability of a programming language. Therefore, one can define and use programming language staples such as loops, ifthen statements, calculations, variables, etc. L^AT_EX's collective name for these is *macro*. In reality, only someone who truly hates himself/herself would actually seriously use L^AT_EX as a programming language. Most people just stop at defining new variables or redefining old ones. These variables are known to L^AT_EX as commands and environments. Redefining environments is reserved for package makers and *very* advanced users, so they are well outside this guide's scope.

All macros must be defined in the preamble. Commands can be defined as having standalone content; combinations of other standalone commands; as having arguments; and any imaginable combination. They are defined as follows:

```

\newcommand{\cmd1}{content}
\newcommand{\cmd2}[n]{content_1#1 content_2#2... content_n#n}
\newcommand{\cmd3}{\command_1 \command_2... \command_3}
\newcommand{\cmd4}[n]{\command_1{#1} \command_2{#2}... \command_1{#n}}

```

When using a command, its arguments will be specified within braces, {}, of which there will be as many as there are arguments in the command's definition. Argu-

⁸<https://en.wikibooks.org/wiki/LaTeX/Glossary>

ments are denoted as, #<n>, and refer to the command's n^{th} argument left to right. For instance:

```
\newcommand{\helwo12}[2]{#1 #2}
\newcommand{\helwo21}[2]{#2 #1}
```

respectively produce the commands `helwo12` and `helwo21`. If one were to write `helwo12Hello!World!`, the output would be "Hello! World!", but if one were to write `helwo21Hello!World!` the output would be "World! Hello!".

Redefining existing commands is somewhat more complex, because it requires the user to know the command's internal parameters. Fortunately, these are found in each command's L^AT_EX wikibook page. As a general example, redefining existing commands is done in the following manner:

```
\renewcommand{\existingcommand}
{
    \component_1 value_1
    \component_2 value_2
    ...
    \component_n value_n
}
```

Commands do not need to be strictly renewed in the preamble, but it's generally advisable to do so. There are times when renewing commands in-document may be necessary—such as special page numbering.

For more information on L^AT_EX macros visit the wiki book's macro page⁹.

2.6 Referencing

Referencing in L^AT_EX is extremely easy to do, all it requires is a label—`label`—and reference—`ref` for non-equations, and `eqref` for equations—commands. Referencing floats also requires the label to come after a non-starred caption.

As long as an item can be *uniquely* identified, it can be labeled and referenced. Labels are placed immediately following sectioning, ToC, appendix, and indexing commands; before line breaks in equations; and after the caption in floats. The `hyperref` package loads further referencing capability, such as hyperlinks and colour-coded links depending on hyperlink type.

The package `cleveref` does not work on Overleaf, but adds incredibly useful functionality such as smart references, reference formats, reference ranges, compressed references, capitalisation, etc. It doesn't work when `amsmath` and `hyperref` are simultaneously loaded. Fortunately `mathtools` is vastly superior to `amsmath` in every respect, and does *not* present this bug; so use it in place of `amsmath` and enjoy the wonders of `cleveref`.

⁹<https://en.wikibooks.org/wiki/LaTeX/Macros>

2.7 Non-English Documents

L^AT_EX is also capable of creating non-English documents. In fact, language packs are freely available in community packages and extensions. The babel and polyglossia¹⁰ packages provide non-english language capability. They change sectioning and float languages, hyphenation patterns, quotations, and preference for decimal dot vs. comma. There are also regional options for many languages: such as iberic and mexican spanish; iberic or brazilian portugues, etc. Unfortunately polyglossia still lacks some regional adjustments, including latin-american and mexican spanish, which means many users should stick to babel for the time being. Babel languages are declared as follows:

```
\usepackage[language1,language2...,languagen]{babel}
```

Where the last language will be the default one. Regional adjustment are placed immediately after the main language, for instance:

```
\usepackage[spanish,mexico]{babel}
```

It is also possible to create multilingual documents, so linguists and language academics are well catered for. More information on this can be found in babel's user manual.

2.8 Bibliography

One of the appeals of L^AT_EX is bibliography management. There are two ways of doing so:

- Manually: better known as the hard way.
- B_IB_TE_X: better known as the only way. Google scholar also provides .bib files by clicking Cite under an article and selecting B_IB_TE_X at the bottom left of the popup window.

They can be referenced using the `cite[]`, where the optional chevrons can be omitted, and the bibliographic label is placed inside the braces.

2.8.1 Manual

Doing things manually means the user has to hard-code the style into the bibliography. The user adds the `thebibliography` environment, which takes a numeric argument for the maximum number of bibliographic items:

¹⁰Preferred for X_EL^AT_EX.

```
\begin{thebibliography}{99}
    \bibitem{<biblabel1_>} ...
    \bibitem{<biblabel2_>} ...
    ...
    \bibitem{<biblabel_n>}
\end{thebibliography}
```

2.8.2 BBTEX

The preferred way of handling bibliographies is by using the `natbib` package together with `BBTEX`. `natbib` offers further citing options and commands which are too numerous and readily available¹¹ to be placed here. Though I'd like to mention `nocite`, because it adds the specified entry to the bibliography, but doesn't print the reference in-text. If the argument is an asterisk, `*`, all entries in the `.bib` file are added to the bibliography regardless of whether they were cited or not.

Using `BBTEX` requires a separate `.bib` file, where entries are defined according to specific formats depending on the type of document or file to be cited, the first string of which is the bibliographic label. The following commands are then placed after all the document's content and before `\end{document}`:

```
\bibliographystyle{<style>}
\bibliography{<filename>}
```

where the `<style>` is replaced by a bibliographic style, and `<filename>` is the `.bib` file's name without extension (also accepts relative and absolute paths). There are many bibliographic styles found in `natbib`'s manual or its wikibook entry¹². They are not added here because their nuances are too subtle and involved for the purposes of this document. Other packages—especially journal-specific packages—add specialist styles.

In order to fully compile a file with a `BBTEX` bibliography, the following compilation train must be followed:

1. `XELATEX`(or engine of your choice).
2. `BBTEX`.
3. `XELATEX`(or engine of your choice).
4. `XELATEX`(or engine of your choice).

This will ensure all references are parsed and the bibliography will be updated to show all the right entries with the correct style. Some editors can automate this process in a single shortcut.

¹¹https://en.wikibooks.org/wiki/LaTeX/More_Bibliographies

¹²https://en.wikibooks.org/wiki/LaTeX/Bibliography_Management#Natbib

Part II

General L^AT_EX Documents

Chapter 3

Creating a Document

3.1 Titular Information

Titular information is declared in the preamble after the package declaration. It varies greatly between document classes. However, key pieces of information which most documents must have, i.e. title, author and date; academic documents also require author affiliations, and some documents require subtitles. It's important to note that all these are added at the user's behest.¹ This information is added last in the preamble, just before the document proper.

```
\title{Document Title}
\subtitle{Document Subtitle}
\author{auth_1\thanks{affil_1}, auth_2\thanks{affil_2},...}
\date{<date>}
```

The date can be dynamically updated to the OS' by using `today` as the date command's argument. The affiliation is added as a footnote by the `thanks` command.

The preferred way to add author affiliations is by using the `authblk` package:

```
\author[x1]{author_1}
\affil[x1]{affil_1}
\author[x2]{author_2}
\affil[x2]{affil_2}
...
\author[xn]{author_n}
\affil[xn]{affil_n}
```

This package makes keeping track of authors and affiliations much easier because it relies on labels rather than a command's position. It also tidies and simplifies having multiple affiliations per author, and multiple authors per affiliation.

Printing the titular information requires the command `maketitle` to be inside the document environment.

¹Contrary to word processors, L^AT_EX will *not* automatically do anything unless the user expressly tells it to.

3.2 Paging

Page numbering and formatting are two of the most important, yet most overlooked things in document typesetting. There are a few paging commands and styles but the common ones are found in table 3.1. The `pagenumberingstyle` name changes the page numbering to one of the default styles and resets the counter to its default value of 1.

Table 3.1: Common paging commands.

Command	Function
<code>pagebreak</code>	New page with spread out paragraphs.
<code>newpage</code>	New page with normal paragraphs.
<code>clearpage</code>	New page that cannot automatically have floats.
<code>cleardoublepage</code>	Sale as <code>clearpage</code> but on two pages.

There are also paging styles which modify a single page's style. The most common default page styles and their function are found in table 3.2. They are declared within the `pagestyle` (all pages), or `thispagestyle` (this page only) commands.

Table 3.2: Common default page styles.

Style	Function
<code>empty</code>	Blank header and footer.
<code>plain</code>	Blank header, footer contains a centered page number.
<code>headings</code>	Blank footer, header contains page number & relevant info.

There are other page styles defined in packages such as `fancyhdr`, whose manuals and downloads can be found on the net.

3.2.1 Page Format

The standard L^AT_EX page formatting is usually sufficient for everyday use. However there are times—such as when writing a thesis template and short but comprehensive L^AT_EX tutorial—when you need special formatting options. Such options follow the structure:

- `setlength{command}{length}`: set a length command to a new value `length`.
- `addtolength{command}{length}`: add a value, `length`, to a length parameter.
- `linespread{len}`: set inter-line spacing to the value `length`.

Added lengths can be negative, so it is possible to subtract lengths. It's also possible to use factors of one length to define another by placing the factor before a specific length command, e.g. `-0.5\textwidth` divides the text width by -2.

There are `length` commands for all manner of things, from margin padding to paragraph skip length—and the need to modify them is niche at best, so most are excluded from this guide. More information on lengths can be found in the `lengths` entry of the `LATEX` wikibook². The most used by far are `textwidth`—text width, which is often different from the page width due to text padding and `multicolumns`—and `width`—page width with margins—because of how useful they are in scaling images.

3.3 Document Environment

The document environment is where the document's actual content is found. It is the master environment within which all other environments must be called. Thus the first line after the preamble is *always* `begin{document}` and the `.tex` file's very last line is *always* `end{document}`.

3.4 Document Sectioning

A standard document can be sectioned into parts, chapters, sections, subsections and subsubsections. However, the user can add new and edit existing sectioning commands to change their format, or even add more arguments. It is quite well known—in *most* reputable moustache-twirling circles—that should you require more sectioning levels than the `LATEX` default, you are most likely doing it *wrong*, and will forever be cursed with spaghetti code. Don't let it happen to you, structure your thoughts, documents, and code properly. The standard sectioning commands are declared as follows:

- `\part` `name`: Parts. Primarily used in book writing.
- `\chapter` `title`: Chapters.
- `\section` `title`: Sections.
- `\subsection` `title`: Subsections.
- `\subsubsection` `title`: Subsubsections.

Their starred versions prevents them from being numbered, and stops them from being indexed unless they are manually added to the table of contents (see section [2.5](#)).

²<https://en.wikibooks.org/wiki/LaTeX/Lengths>

Chapter 4

Non-Maths Environments

There are many environments that respect the text flow and are not specifically designed for mathematical or special notation. This section contains descriptions for some of the most common ones.

4.1 List Structures

Both stratified listing environments have four standard nesting levels (`enum<x>` counters). Bullet shapes, enumeration characters, and list levels can be specified and customised in the environment's options, `beginenvironment[options]1`. The `enumitem` package expands this capability.

4.1.1 Itemize

The `itemize` environment lists items as bullet-point lists. Bullet points can be redefined with `renewcommand` or with functionality offered by the `enumitem` package. The environment is used as follows:

```
\begin{itemize}
    \item Nesting level 1.
    \begin{itemize}
        \item Nesting level 2.
        \begin{itemize}
            \item Nesting level 3.
            \begin{itemize}
                \item Nesting level 4.
            \end{itemize}
        \end{itemize}
    \end{itemize}
\end{itemize}
```

And outputs:

¹https://en.wikibooks.org/wiki/LaTeX>List_Structures

- Nesting level 1.
 - Nesting level 2.
 - * Nesting level 3.
 - Nesting level 4.

Printed indentations are a result of the environment, not code indentation (see section [2.5](#)).

4.1.2 Enumerate Environment

The enumerate environment places items as numbered lists. The numbering can be redefined with `renewcommand` or with functionality offered by the `enumitem` package. It's declared as follows:

```
\begin{enumerate}
  \item Nesting level 1.
    \begin{enumerate}
      \item Nesting level 2.
        \begin{enumerate}
          \item Nesting level 3.
            \begin{enumerate}
              \item Nesting level 4.
              \item Nesting level 4.
            \end{enumerate}
          \end{enumerate}
        \end{enumerate}
      \end{enumerate}
    \end{enumerate}
  \end{enumerate}
```

And outputs:

1. Nesting level 1.
 - (a) Nesting level 2.
 - i. Nesting level 3.
 - A. Nesting level 4.
 - B. Nesting level 4.
 - ii. Nesting level 3.
 - (b) Nesting level 2.
2. Nesting level 1.

Printed indentations are a result of the environment, not code indentation (see section [2.5](#)).

4.1.3 In-line Lists

There are a couple of ways to do this, with either the `paralist` or `enumitem` package. For `paralist`, the counter tokens are A, a, I, i and 1; and respectively represent the `Alpha`, `alph`, `Roman`, `roman`, and `arabic` counter styles. Conversely, `enumitem` uses them explicitly. Both environments can take optional font shape switches and commands, as well as other symbols for item label customisation. In `paraenum` this is done like so:

```
\begin{inparaenum}[\itshape A\upshape]
    \item Item 1.
    \item Item 2.
\end{inparaenum}
```

in `enumitem` the way to do this is:

```
\begin{enumitem*}[label=\itshape\Alpha*\upshape]
    \item Item 1.
    \item Item 2.
\end{enumitem*}
```

They both produce: A) Item 1. B) Item 2.

4.2 Verbatim Environments

Say you want to make a template, package, or `LATEX` guide, so you need to textually print `LATEX` commands so people can actually read them. Or you simply want to comment large sections of code. Then the `verbatim` package is essential.

4.2.1 Verbatim and Alltt

The `verbatim` environment *explicitly* prints whatever is written within it (spaces included), regardless of what it is. It's useful for printing `LATEX` commands in independent lines. There is also a way to print in-line verbatim, `\verb|content|`, where the first character after `verb` is the delimiter—which can be any character except for `*`. The `alltt` environment is somewhat different in that it accepts *some* commands within it².

4.2.2 Comment Environment

The `comment` environment is self-explanatory. Anything written within it will be treated as a comment and will therefore not be in the output file.

²https://en.wikibooks.org/wiki/LaTeX/Paragraph_Formatting

Chapter 5

Float Environments

Floats are the names L^AT_EX gives environments which are uncoupled from normal text, i.e. can freely float around the document, namely figures and tables. L^AT_EX generally does a relatively good job at placing floats where they make sense. Unfortunately, it sometimes does an atrocious one—especially in documents with highly customised layouts—so the user has to intervene. Fortunately, there exist optional placement specifiers which give the user control over float placement. Table 5.1¹ shows possible placement specifiers and their meaning.

Table 5.1: Table with placement specifiers and their description.

Specifier	Description
h	Place <i>approximately</i> where it occurs in the source code.
t	Place at the <i>top</i> of the page.
b	Place at the <i>bottom</i> of the page.
p	Place in a <i>dedicated</i> page.
H	Place <i>exactly</i> where it occurs in the source code, needs <i>float</i> package.
!	Overrides L ^A T _E Xparameters for determining good float positions.
!htbp	Overrides L ^A T _E Xparameters and finds the best position out of all placings.

Along with these, there's the command, *FloatBarrier*, which forces all previous floats to be rendered before it. And the *placeins* package, often used with the *section* option, which forces floats to be placed within the section they are called.

Generally, floats require a caption, which is added by the command *caption*. As a rule, table captions in tables are found at the top, while figure captions at the bottom. There is also a starred version, *caption**, that eliminates the float number, useful when tables or figures need notes. In order to reference a float, it must be assigned a label, *label*, after the caption. If a float has no caption but needs a label, an empty caption can be added before labeling, *caption*². Floats

¹Edited from https://en.wikibooks.org/wiki/LaTeX/FLOATS,_FIGURES_and_Captions

²Do *not* use starred captions when labeling, otherwise there will be nothing—but emptiness and deep internal sadness—for the label to reference.

Table 5.2: Options and their descriptions for the `\includegraphics` command.

Option	Description
<code>width=x</code>	Image width x .
<code>height=y</code>	Image height y .
<code>keepaspectratio</code>	<code>true</code> , <code>false</code> . Keep aspect ratio with max width x and height y .
<code>scale=x</code>	Scale by factor x .
<code>angle=x</code>	Rotate by x (standard rotation).
<code>trim=l b r t</code>	Trim left l, bott b, right r, top t (broken in X ^E T _E X).
<code>clip</code>	Has to be <code>true</code> for trim to work.
<code>resolution=x</code>	Image resolution in x dpi.

are also generally centered on a page/column, so the `centering` alignment switch is usually the first line in a float environment. In multicolumn documents, the normal environments may overlap with the text and each other. Starred versions, `{figure*}` and `{table*}` avoid this issue by reserving the whole page width (the value of `width`) for the float. Unfortunately, not all placement options work with starred floats.

5.0.1 Figures

Documents often require the user to add images as figures or standalone inputs. As with many things, it is common knowledge that an angel loses its wings every time someone inserts a `.jpg` into a L^AT_EXdocument. So for the love of ceiling cat



, use vectorised images such as `.eps`, `.ps`, and `.pdf`. The only exceptions are scatter plots and heat maps, where it's advisable to use `.bmp` or `.png` instead—as they are more accurate in such cases.

The `figure` environment is generally used in the following way:

```
\begin{figure}[<placementSpecifier>]
    \centering
    \includegraphics[options]{imagename.extension}
    \caption{caption}
    \label{f:label}
\end{figure}
```

`includegraphics[]` can be used on its own to include in-line images. Its options and their descriptions are found in table 5.2³.

³Edited from https://en.wikibooks.org/wiki/LaTeX/Importing_Graphics.

5.0.2 Subfigures

The `subfigure` environment can only be declared within the `figure` environment, and requires the `subcaption` package. Its usage is as follows:

```
\begin{figure}
    \centering
    \begin{subfigure}[t]{0.45\textwidth} % t = align subfigure on top.
        \includegraphics[width=\textwidth]{dank.jpg}
        \caption{A dank meme.}
        \label{f:dankmeme}
    \end{subfigure}
    ~ % add desired spacing between images, e. g. ~, \quad, \quad, \hfill etc.
    % (or a blank line to move the subfigure to a next line.)
    \begin{subfigure}[b]{0.45\textwidth} % align subfigure at the bottom.
        \includegraphics[width=\textwidth]{ceiling-cat.jpg}
        \caption{Ceiling cat.}
        \label{f:ceilingcat}
    \end{subfigure}
    \end{subfigure}

    \begin{subfigure}[c]{0.45\textwidth} % b = align subfigure on its centre.
        \includegraphics[width=\textwidth]{watoge.jpg}
        \caption{Watoge}
        \label{f:watoge}
    \end{subfigure}
    ~

    \begin{subfigure}[c]{0.45\textwidth} % b = align subfigure on its centre.
        \includegraphics[width=\textwidth]{MLG.png}
        \caption{MLG.}
        \label{f:mlg}
    \end{subfigure}
    \caption{Pictures of memes.}\label{f:memes}
\end{figure}
```

Which produces the output found in figs. [5.1](#), [5.1a](#), [5.1b](#), [5.1c](#) and [5.1d](#):



(a) A dank meme.



(b) Ceiling cat.



(c) Watoge



(d) MLG.

Figure 5.1: Pictures of memes.

5.1 Tables

Tables in L^AT_EX can either be the source of great pride or great pain. Done right, they look classy and slick... done wrong, they are tacky and I hate them⁴. Before designing a table, you must answer me these questions three⁵:

1. Readability: is it easily understandable?
2. Minimalism: does it contain the minimum required information—does it minimise redundancy?
3. Simplicity: how many borders does it have—does it really need all those vertical and/or horizontal borders?

The package `booktabs` provides improved versions of the horizontal line commands, `hline` and `cline`.⁶ They are: `toprule`, `midrule`, `cmidrule`, and `bottomrule`. They all provide additional spacing to make tables more readable, and accept options to modify their trimming and thickness. A table is declared in the following way:

```
\begin{table}[<placementSpecifier>]
  \centering
  \caption{<caption>}
  \label{<t>:<label>}
  \begin{tabular}{<n columns (with alignment), vertical borders>}
    \toprule
    heading 1 & ... & heading n
    \midrule
    a_11 & ... & a_1n \\
    .
    .
    .
    a_m1 & ... & a_mn \\
    \bottomrule
  \end{tabular}
\end{table}
```

In line with the three principles of table design, one must avoid borders as much as possible without sacrificing readability. For most purposes, tables only need horizontal lines at the top, bottom, and for separating heading from the body. There is also `cmidrule`_{i-j} which spans columns *i* to *j*. The `array` package extends table functionality by providing extra arguments for table customisation. The columns are represented by `l`, `c` and `r` for left, center and right alignment, they can

⁴Stickin' it to 'The Man'.

⁵You seek the holy grail.

⁶If there's a need for non-tabular horizontal lines—such as redefining footnote rules—you should still use `\hline`.

be placed immediately after one another or separated by spaces. Vertical borders are denoted by | between columns. The actual columns are then separated by ampersands &. If the number of columns is n , the number of ampersands is $n - 1$.

To showcase what we have learned of tables up to now, table 5.3 is an example of a very poorly formatted table. Its generating code is:

```
\begin{table}
  \centering
  \caption{Poorly designed and formatted table.}
  \label{t:example}
  \begin{tabular}{|c|||r||}
    \cline{2-3}
    & Measurement & Magnitude & Units \\
    \hline
    Speed & 15 & m/s \\
    Time &  $\infty$  & s \\
    \hline
    \hline
  \end{tabular}
\end{table}
```

Table 5.3: Poorly designed and formatted table.

Measurement	Magnitude	Units
Speed	15	m/s
Time	∞	s

The documentation for `booktabs` contains further information on table design, formatting, and package features.

5.1.1 Multicolumn and Multirow Tables

Sometimes tables have fields which span more than one row or column (sometimes both). When this is unavoidable, the `multirow` package provides the commands:

```
\cs{multirow}{num_rows}{width}{contents}
\cs{multicolumn}{num_columns}{column_alignment and borders}{contents}
```

An asterisk `"*"` as width in `multirow` stands for the contents' natural width. `multicolumn` gobbles all relevant columns, so there's no need to add the gobbled ampersands. If `multirow` is used, all subsequent rows in the same column(s) must be left empty and take up the same number of columns. Table 5.4 was sourced directly from the wiki entry for `tables`⁷ because it's an excellent non-trivial minimal working example:

⁷<https://en.wikibooks.org/wiki/LaTeX/Tables>

```

\begin{table}
    \centering
    \caption{Multirow and multicolumn table.}
    \label{t:multirowcol}
    \begin{tabular}{cc|c|c|c|c|l}
        \cline{3-6}
        & & \multicolumn{4}{c}{Primes} \\ \cline{3-6}
        & & 2 & 3 & 5 & 7 \\ \cline{1-6}
        \multicolumn{1}{|c|}{\multirow{2}{*}{Powers}} & & 3 & 2 & 0 & 1 & \\ \cline{2-6}
        \multicolumn{1}{|c|}{504} & & 2 & 3 & 1 & 0 & \\ \cline{2-6}
        \multicolumn{1}{|c|}{\multirow{2}{*}{Powers}} & & 540 & 2 & 3 & 1 & 0 & \\ \cline{2-6}
        \multicolumn{1}{|c|}{gcd} & & 2 & 2 & 0 & 0 & min \\ \cline{2-6}
        \multicolumn{1}{|c|}{\multirow{2}{*}{lcm}} & & 3 & 3 & 1 & 1 & max \\ \cline{2-6}
    \end{tabular}
\end{table}

```

Table 5.4: Multirow and multicolumn table.

		Primes			
		2	3	5	7
Powers	504	3	2	0	1
	540	2	3	1	0
Powers	gcd	2	2	0	0
	lcm	3	3	1	1
		min			
		max			

5.1.2 Multipage Tables

Sometimes a project requires tables which span more than a single page. For cases such as these, the `longtable` package provides the `longtable` environment, which offers customisation commands relevant to multipage tables⁸. `longtable` does not need the `tabular` environment, it is declared as follows:

```
\begin{longtable}{<n columns (with alignment), vertical borders>}\caption{caption}\label{t:longtable}\topruleheading 1 & ... & heading n\midrulea_11 & ... & a_1n \\
```

⁸<http://mirrors.rit.edu/CTAN/macros/latex/required/tools/longtable.pdf>

```
a_m1 & ... & a_mn \\  
\\bottomrule  
\\end{longtable}
```

`longtable` also works with the functionality offered by the `multirow` package. However, multiple compilations (4+) may be required to get long tables with multirows and/or multicolumns looking as they should.

Chapter 6

Mathematical Environments

The primary reason why Don Knuth started developing \TeX was to create an easy and powerful tool for typesetting academic mathematical documents. There are absurdly many mathematical tools and symbols in \LaTeX , but this document will only touch the most common ones non-mathematicians use. The packages `mathtools`, `bm`, `amssymb`, and `pxfonts` respectively provide mathematical packages and symbols, corrected bold maths fonts, other mathematical symbols, and more fonts and symbols. Mathematical symbols, functions and typefaces cannot exist outside a mathematical environment, though some have non-maths—or text—versions, usually `textsymbol name` or `textcommand name`.

6.1 Maths and Display Maths

The two most basic mathematical environments are the inline and display maths environments. The inline maths environment is delimited by single dollar signs `$ inline maths $`, which yields *inlinemath*. Inline maths has its limitations because it is text-bound, which can cause problems when using mathematical notation that extends beyond the bounds of a normal line of text. For example
$$\frac{\sum_{i=0}^n i}{\sum_{i=0}^n 2i} = \frac{1}{2}$$
 generated with,

```
\frac{\sum\limits_{i=0}^n i}{\sum\limits_{i=0}^n 2i} = \frac{1}{2}
```

Which pushes line spacing beyond acceptable levels. In cases such as these, when equation numbering is not an issue, one can use the display maths environment delimited by double dollar signs, `$$ display maths $$`, or the preferred method (better spacing) of using backslash followed by chevrons, `\[display maths \]`. Display maths uncouples the notation from the text and places it *exactly* where the command is found the point it's called from. It also lets large symbols scale

accordingly. Using the previous equation as an example,

$$\frac{\sum_{i=0}^n i}{\sum_{i=0}^n 2i} = \frac{1}{2},$$

it's easy to see the need for both. It's also important to note that all mathematical environments gobble spaces. Use `\text{}`, `\text{ }`, or any other spacing command to add spaces. If you need text to be printed within a mathematical environments use the `text` command and make sure to explicitly add spaces at the beginning and end of the normal text to ensure proper spacing. For example: `\text{x}\text{ apples}+y\text{ pears}`, `xapples+ypears` vs. `\text{x}\text{ apples }+y\text{ pears}`, `x apples + y pears`.

6.2 Common Symbols and Functions

There are a myriad of symbols and functions for virtually *all* areas of maths. This document touches on the ones non-mathematicians will generally use. If anyone is curious about more symbols make sure to check out Scott Pakin's 164-page list of L^AT_EX symbols [?].

6.2.1 Greek Maths Symbols

All maths greek letters are backslash commands of the letter's name. There are capitalised versions which are written the same, but start with a capital letter. Some letters also have variations which are denoted as `var<letter_name>`. Table 6.1 contains an example of this. There are also bold versions of some symbols,

Table 6.1: Examples of maths greek letters.

ϕ	φ	Φ
<code>phi</code>	<code>varphi</code>	<code>Phi</code>

which is specified by the `boldsymbol` command. When a bold symbol does not exist there's the `pmb` (poor man's bold) which prints slightly offset symbols to give the appearance of bold typeface.

6.2.2 Common Binary Operations

6.2.3 Common Relation Symbols

6.2.4 Arrows

Arrows follow a few rules, capitalising the first letter makes the arrow have a two line shaft. The direction is specified first, if the arrow has two heads, left goes

Table 6.2: Common binary operations.

+	-	\pm	\mp	\times	\div	.
+	-	pm	mp	times	div	cdot

Table 6.3: Common relation symbols.

$=$	\neq	\equiv	\approx	\sim	\cong	$<$	$>$	
=	neq	equiv	simeq	approx	sim	cong	<	>
\leq	\geq	\ll	\gg	\in	\ni	\propto	\perp	\parallel
leq	geq	ll	gg	in	ni	proto	perp	parallel

before right and up before down. There are two standard types, arrows (always stated as singular) and harpoons (singular except left right harpoons). Harpoons can only be horizontal and the orientation of the spike is specified last. Table 6.4 contains some examples. Arrows can also be specified as long, which goes first in the command.

Table 6.4: Arrows.

\Leftrightarrow	\Downarrow	\Leftarrow
Leftrightarrow	updownarrow	leftrightharpoons
\leftarrow	\iff	\rightarrow
leftharpoonup	Longleftrightarrow	longrightarrow

6.2.5 Common Miscellaneous Symbols

Table 6.5: Common miscellaneous symbols.

\dots	\dots	\vdots	\ddots	∞	\hbar	\emptyset
ldots	cdots	vdots	ddots	infty	hbar	emptyset
\exists	∇	\neg	\aleph	\Im	\angle	∂
exists	nabla	neg	Re	Im	angle	partial
\therefore \therefore						

6.2.6 Common Functions

\LaTeX has commands for all basic functions. They are backslash commands defined as the function's shortened name without abuse of notation, e.g. inverse trigonometric functions are written as `arcfunction`. In fact, most functions are defined exactly how they are written in rigorous pen & paper maths.

6.2.7 Common Variable-Sized Symbols

Some symbols can change size depending on where they are in-document. Sometimes however, they're not big enough for their argument. This is solved by the nest-able `mathlarger` command, provided by the `relsize` package. The number of integrals is defined by the number of *i*'s in the command—up to 4 for normal integrals, 3 for cyclic ones¹. Linear integrals can also have dots.

Table 6.6: Common variable-sized symbols.

Σ	\prod	\int	\oint	\iint	\iiint	$\frac{a}{b}$	$\frac{a}{b}$
sum	prod	int	oint	idotsint	oiint	fracab	dfracab

6.2.8 Non-trivial Delimiters

Table 6.7: Non-trivial delimiters.

$\ $	$ $	\langle	\rangle	\lfloor	\rfloor	\lceil	\rceil
\mid	$\mid\mid$	$\langle\!\langle$	$\rangle\!\rangle$	$\lfloor\!\lfloor$	$\rfloor\!\rfloor$	$\lceil\!\lceil$	$\rceil\!\rceil$

6.2.9 Common Mathematical Accents

There are many mathematical accents, however we will only mention the most common ones used in science. The majority of the most common ones are provided by the `AMSTeX` packages. The `mathdots` package redefines the spacing between dots for three and four dots. The number of dots is defined by the number of *d*'s in the `dots` command. There also exist dotless versions of *i* and *j*, provided by `imath` and `jmath` respectively—useful for properly adding hats to orthogonal unit vectors.

Some accents also have wide versions. Some bar accents (including arrows) are denoted as `overbar` accent, while other wide accents are denoted as `wideaccent`. Table 6.8 contains some common mathematical accents and their respective commands.

Furthermore, `overbrace` and `underbrace` may have sub/superscripts automatically placed directly below and above them without resorting to

```
z = \overbrace{\underbrace{x}_{\text{Real}} + i \underbrace{y}_{\text{Imaginary}}}^{\text{Complex number.}}
```

$$z = \underbrace{x}_{\text{Real}} + i \underbrace{y}_{\text{Imaginary}}$$

Complex number.

¹Triple cyclic integrals need the `pxfonts` or `txfonts` package

Table 6.8: Common mathematical accents.

\hat{a}	\check{a}	\bar{a}	\vec{a}	\dot{a}	\ddot{a}	\tilde{a}
hata	checka	bara	veca	dota	ddota	tildea
\mathring{a}		\widehat{abc}		\sqrt{abc}		$\sqrt[n]{abc}$
<code>\mathring{a}</code>		<code>\widehat{abc}</code>		<code>\sqrt{abc}</code>		<code>\sqrt[n]{abc}</code>
\underline{abc}				$\overbrace{\underbrace{abc}}_{\substack{\leftarrow \\ \rightarrow}}$		$\overleftarrow{\overleftarrow{\overleftarrow{abc}}}$
<code>\underline{abc}</code>				<code>\overbrace{\underbrace{abc}}_{\substack{\leftarrow \\ \rightarrow}}</code>		<code>\overleftarrow{\overleftarrow{\overleftarrow{abc}}}</code>

L^AT_EX will let the sub and superscripts take spacing priority, if this is a problem, placing the whole sub/superscript inside the `mathclap` command will eliminate the extra spacing.

6.2.10 Limits and Substack Commands

Some symbols may have arguments and/or sub/superscripts in different places. Placing the `limits` command immediately after a symbol, moves the immediately following sub/superscript below and above it, respectively—having both at the same time is not a requirement. The `substack` command allows subscripts to be stacked. Table 6.9 shows this.

Table 6.9: Effect of the `\limits` command.

No limits	limits	Command
\sum_i	\sum_i	<code>\$\sum\limits_{i}</code>
\sum^j	\sum^j	<code>\$\sum\limits^j\$</code>
\sum_i^j	\sum_i^j	<code>\$\sum\limits_{i}^j\$</code>
$\sum_{\substack{1 \leq i \leq n \\ i \neq j}}$	$\sum_{\substack{1 \leq i \leq n \\ i \neq j}}$	<code>sum\limits_{\substack{1 \leq i \leq n \\ i \neq j}}</code>

6.2.11 Scaled Delimiters

It's very common to have large equations which require large delimiters, these are obtained by adding `left`, `right` and `middle` (optional) before the actual delimiter, they also scale the delimiter according to the size of the equation. It's important to

remember that braces still need a backslash for L^AT_EX to print them, `\left{ \right}`. Sometimes the delimiter is unilateral (only one is printed), but their scaled versions must be closed (i.e. have both left and right components), in such cases a single full stop, `.` becomes the closing delimiter. The following equations are good examples of this.

```
\begin{aligned}
& \exp(-\sum\limits_i^N \frac{E_i}{k_b T}) \\
& \exp\left(-\sum\limits_i^N \frac{E_i}{k_b T}\right) \\
& \int\limits_a^b \exp\left(-\frac{E}{k_b T}\right) dE = -k_b T \exp\left(-\frac{E}{k_b T}\right) \Big|_a^b \\
& \int\limits_a^b \exp\left(-\frac{E}{k_b T}\right) dE = -k_b T \exp\left(-\frac{E}{k_b T}\right) \Big|_a^b
\end{aligned}
```

$$\exp\left(-\sum_i^N \frac{E_i}{k_b T}\right) \quad (6.1)$$

$$\exp\left(-\sum_i^N \frac{E_i}{k_b T}\right) \quad (6.2)$$

$$\int_a^b \exp\left(-\frac{E}{k_b T}\right) dE = -k_b T \exp\left(-\frac{E}{k_b T}\right) \Big|_a^b \quad (6.3)$$

$$\int_a^b \exp\left(-\frac{E}{k_b T}\right) dE = -k_b T \exp\left(-\frac{E}{k_b T}\right) \Big|_a^b \quad (6.4)$$

6.2.12 Common Maths Typefaces

Oftentimes, different mathematical objects have specific notations. For most applications no packages are needed, but loading both `bm` and `mathtools` packages cover most other cases. Table 6.10 provides some examples.

Table 6.10: Common mathematical typefaces.

Command	Result
<code>\mathtt{ABCdef123}</code>	$ABCdef123$
<code>\mathbb{ABCdef123}</code>	$ABC\mathbb{U}\mathbb{K}\mathbb{K}$
<code>\mathbf{ABCdef123}</code>	$\mathbf{ABCdef123}$
<code>\mathcal{ABCdef123}</code>	$\mathcal{ABC}\mathcal{T}\{\infty\in\mathfrak{A}$
<code>\mathsf{ABCdef123}</code>	$\mathsf{ABCdef123}$
<code>\mathit{ABCdef123}</code>	$\mathit{ABCdef123}$
<code>\mathrm{ABCdef123}</code>	$\mathrm{ABCdef123}$
<code>\mathfrak{ABCdef123}</code>	$\mathfrak{ABCde}\mathfrak{g}$
<code>\bm{ABCdef123}</code>	$\bm{ABCdef123}$

6.3 Align

The `align` environment is the one you should be using for all your equation typesetting needs. It does everything the `equation` and `eqnarray` packages do, but better. The starred variant removes all equation numbers. Its behaviour is exactly the same as `tabular`, but does not require the user to declare the number of columns or alignment within them. It also automatically defines all the lines it encloses as a mathematical environment, removing the need to add dollar signs. Individual equations can be labeled by placing `label<label>` before a line break, and referenced with `eqref`. Individual equation numbers can be removed with the `nonumber` command before a line break. The following equations are aligned on the equal sign:

```
\begin{aligned}
F(x) &= \int f(x), \mathrm{d}x + \mathrm{C} \\
f(x) &= \frac{\mathrm{d}F(x)}{\mathrm{d}x} \nonumber \\
g(x, y, z) &= 0
\end{aligned}
```

$$F(x) = \int f(x) \, dx + C \tag{6.5}$$

$$f(x) = \frac{dF(x)}{dx}$$

$$g(x, y, z) = 0 \tag{6.6}$$

6.3.1 Subequations

Subequations are declared by the environment of the same name, which wraps around other equation environments such as, `align`, `eqnarray` and `equation`. They have same behaviour as normal equations, including labeling and referencing. Their numbering can also be customised via `renewcommand`. Using the aforementioned example within the `subequations` environment yields:

```
\begin{subequations}
\begin{aligned}
F(x) &= \int f(x), \mathrm{d}x + \mathrm{C} \label{e:int}\\
f(x) &= \frac{\mathrm{d}F(x)}{\mathrm{d}x} \nonumber\\
g(x,y,z) &= 0
\end{aligned}
\end{subequations}
```

$$F(x) = \int f(x) \, dx + C \tag{6.7a}$$

$$f(x) = \frac{dF(x)}{dx}$$

$$g(x, y, z) = 0 \tag{6.7b}$$

6.4 Matrices

Matrices can be represented in many ways depending on the context, or to denote certain matrix operations. Pre-defined matrices require the `mathtools` package. The default types are shown in table 6.11, which is edited from the `mathematics` entry in the L^AT_EX wikibook². Non-starred versions center the columns by default, starred variants allow user-defined alignments. The `matrix` environment must be declared within a `maths` environment. Matrices can also be created using the `array` environment, which is essentially the `maths`-mode version of the `tabular` environment.

Table 6.11: Matrix types.

Command	Delimiter
<code>pmatrix</code>	()
<code>bmatrix</code>	[]
<code>Bmatrix</code>	{ }
<code>vmatrix</code>	— —
<code>Vmatrix</code>	

6.4.1 Aligned and Array

There are times when annotating equations is useful for the reader, or desired by the author. In cases such as these, the `aligned` provides a quick solution, though it only provides a single ampersand for aligning items. In cases where more than one ampersand is needed use `array` instead, which is the `maths` equivalent of `tabular`.³

²<https://en.wikibooks.org/wiki/LaTeX/Mathematics>

³They are used and declared in exactly the same way; only `tabular` is used within normal text, and `array` within `maths` environments.

```

\begin{align}
\left. \begin{aligned}
f(x,y,z) &= \alpha x + \beta y + \gamma z + \delta \\
f(x,y) &= \alpha x + \beta y + \gamma \\
f(x) &= \alpha x + \beta
\end{aligned} \right. \text{Linear equations.}
\end{align}

```

$$\left. \begin{aligned} f(x,y,z) &= \alpha x + \beta y + \gamma z + \delta \\ f(x,y) &= \alpha x + \beta y + \gamma \\ f(x) &= \alpha x + \beta \end{aligned} \right\} \text{Linear equations.} \quad (6.8)$$

6.5 Cases

The `cases` environment allows the user to define piecewise functions without the pain of recurring to `aligned` and delimiter scaling. Similarly to the `eqnarray` and `aligned` environments, it provides a single ampersand for aligning. Take eq. (6.9) for example. There also exists `dcases`, which provides a display version of the environment—it properly scales symbols—as shown in eqs. (6.10) and (6.11). It also has a starred version shown in Eq. (6.12), which sets everything to the right of the ampersand to normal text—therefore all maths in this region must be within single dollar signs.

```

\begin{align}
|f(x) + b| + c &= \begin{cases} -f(x) - b + c & \text{if } x < 0 \\ f(x) + b + c & \text{otherwise} \end{cases} \\
g(x) &= \begin{cases} \int f(x), \mathrm{d}x & \text{if } x < 0 \\ 0 & \text{otherwise} \end{cases} \\
g(x) &= \begin{dcases} \int f(x), \mathrm{d}x & \text{if } x < 0 \\ 0 & \text{otherwise} \end{dcases} \\
g(x) &= \begin{dcases*} \int f(x), \mathrm{d}x & \text{if } \$x < 0\$ \\ 0 & \text{otherwise} \end{dcases*}
\end{align}

```

$$|f(x) + b| + c = \begin{cases} -f(x) - b + c & \text{if } x < 0 \\ f(x) + b + c & \text{otherwise} \end{cases} \quad (6.9)$$

$$g(x) = \begin{cases} \int f(x) dx & \text{if } x < 0 \\ 0 & \text{otherwise} \end{cases} \quad (6.10)$$

$$g(x) = \begin{cases} \int_0^x f(x) dx & \text{if } x < 0 \\ 0 & \text{otherwise} \end{cases} \quad (6.11)$$

$$g(x) = \begin{cases} \int_0^x f(x) dx & \text{if } x < 0 \\ 0 & \text{otherwise} \end{cases} \quad (6.12)$$

Chapter 7

Referencing and Citing

7.1 Referencing

Referencing in L^AT_EX is extremely easy to do, all it requires is a label—`label`—and reference—`ref` for non-equations, and `eqref` for equations—commands. Referencing floats also requires the label to come after a non-starred caption.

As long as an item can be *uniquely* identified, it can be labeled and referenced. Labels are placed immediately following sectioning, ToC, appendix, and indexing commands; before line breaks in equations; and after the caption in floats. The `hyperref` package loads further referencing capability, such as hyperlinks and colour-coded links depending on hyperlink type.

The package `cleveref` does not work on Overleaf, but adds incredibly useful functionality such as smart references, reference formats, reference ranges, compressed references, capitalisation, etc. It doesn't work when `amsmath` and `hyperref` are simultaneously loaded. Fortunately `mathtools` is vastly superior to `amsmath` in every respect, and does *not* present this bug; so use it in place of `amsmath` and enjoy the wonders of `cleveref`.

7.2 Bibliography

One of the appeals of L^AT_EX is bibliography management. There are two ways of doing so:

- Manually: better known as the hard way.
- B_IB_TE_X: better known as the only way. Google scholar also provides .bib files by clicking Cite under an article and selecting B_IB_TE_X at the bottom left of the popup window.

They can be referenced using the `cite[]`, where the optional chevrons can be omitted, and the bibliographic label is placed inside the braces.

7.2.1 Manual

Doing things manually means the user has to hard-code the style into the bibliography. The user adds the `thebibliography` environment, which takes a numeric argument for the maximum number of bibliographic items:

```
\begin{thebibliography}{99}
  \bibitem{<biblabel1>} ...
  \bibitem{<biblabel2>} ...
  ...
  \bibitem{<biblabel_n>}
\end{thebibliography}
```

7.2.2 BBITEX

The preferred way of handling bibliographies is by using the `natbib` package together with BBITEX. `natbib` offers further citing options and commands which are too numerous and readily available¹ to be placed here. Though I'd like to mention `nocite`, because it adds the specified entry to the bibliography, but doesn't print the reference in-text. If the argument is an asterisk, `*`, all entries in the `.bib` file are added to the bibliography regardless of whether they were cited or not.

Using BBITEX requires a separate `.bib` file, where entries are defined according to specific formats depending on the type of document or file to be cited, the first string of which is the bibliographic label. The following commands are then placed after all the document's content and before `enddocument`:

```
\bibliographystyle{<style>}
\bibliography{<filename>}
```

where the `<style>` is replaced by a bibliographic style, and `<filename>` is the `.bib` file's name without extension (also accepts relative and absolute paths). There are many bibliographic styles found in `natbib`'s manual or its wikibook entry². They are not added here because their nuances are too subtle and involved for the purposes of this document. Other packages—especially journal-specific packages—add specialist styles.

In order to fully compile a file with a BBITEX bibliography, the following compilation train must be followed:

1. XELATEX(or engine of your choice).
2. BBITEX.
3. XELATEX(or engine of your choice).
4. XELATEX(or engine of your choice).

¹https://en.wikibooks.org/wiki/LaTeX/More_Bibliographies

²https://en.wikibooks.org/wiki/LaTeX/Bibliography_Management#Natbib

This will ensure all references are parsed and the bibliography will be updated to show all the right entries with the correct style. Some editors can automate this process in a single shortcut.

Part III

L^AT_EX Presentations

Part IV

Conclusions

Chapter 8

Closing Remarks

Well, it's been quite a journey, hasn't it? I originally intended this to be ~ 10 pages long, but it seems I gravely underestimated. It contains all the stuff I've learned in my 2.5 years of using L^AT_EX—except what I know of beamer (that may come later) and minted (doesn't work on Overleaf). I have tried to include all the tips and techniques that have cost me so much time and late nights to learn—all in an effort to prevent it from happening to you. I hope my objectives are met when you read this document.

Finally, I'd like to ask something of you. The world is built one little brick at a time. But the process is often marred by greed and selfishness. Lets not be like that. I did this out of the kindness of my heart, expecting nothing in return but a promise you will make only to yourselves: that you will do your best to spontaneously help others whenever you can.

With that, I give you my thanks and wish you all the best.