An Introduction to LaTeX

Version 1.02 July 2015

Institute of Information and Communication Technology
Shahjalal University of Science and Technology

Thank you!

Much of the material used in this introduction comes from

`The LaTeX wikibook`[1] and

`The Not So Short Introduction to` LaTeX 2_ε[2] by Tobias Oetiker

[1] http://en.wikibooks.org/wiki/LaTeX

[2] http://ctan.tug.org/tex-archive/info/lshort/english/lshort.pdf

Preface

LaTeX[1] is a typesetting system that is very suitable for producing scientific and mathematical documents of high typographical quality. It is also suitable for producing all sorts of other documents, from simple letters to complete books. LaTeX uses TeX[2] as its formatting engine.

This short introduction describes LaTeX and should be sufficient for most applications of LaTeX. Refer to [1], [3], [6] for a complete description of the LaTeX system.

This introduction is split into 6 chapters:

Chapter 1 introduces LaTeX and its installation in computer systems.

Chapter 2 presents our first document in LaTeX and its compilation process. It also discusses about the syntax of LaTeX.

Chapter 3 tells you about the basic structure of LaTeX documents.

Chapter 4 goes into the details of typesetting your documents. It explains most of the essential LaTeX commands and environments.

Chapter 5 explains how to typeset formula with LaTeX. Many examples demonstrate how to use one of LaTeX's main strengths.

Chapter 6 shows how to install and typeset Bangla texts in LaTeX.

It is important to read the chapters in order—the book is not that big, after all. Be sure to carefully read the examples, because a lot of the information is in the examples placed throughout the book.

If you need to get hold of any LaTeX related material, have a look at one of the Comprehensive TeX Archive Network (CTAN) sites. The homepage is at `http://www.ctan.org`.

If you have ideas for something to be added, removed or altered in this document, please let me know. I am especially interested in feedback from LaTeX novices about which bits of this intro are easy to understand and which could be explained better.

Mohammad Abdullah Al Mumin `<mumin-cse@sust.edu>`

Professor
Computer Science and Engineering
Shahjalal University of Science and Technology

Table of Contents

List of Tables

Chapter 1

Introduction

1.1 What is LaTeX?

LaTeX is a powerful typesetting system, used for producing scientific and mathematical documents of high typographic quality. Unlike WYSIWYG[1] tools such as MS Word or LibreOffice, it uses plain text files that contain formatting commands. Its big, open source, stable and used by many technical publishing companies.

Donald E Knuth designed a typesetting program called TeX in 1977 especially for complex mathematical text. LaTeX is a macro package that allows authors to use TeX easily, and uses TeX as its formatting engine. It is available for most operating systems; for example, you can use it on low specification PCs and Macs, as well as on powerful UNIX and VMS systems. There are many different implementations of LaTeX. The word LaTeX is pronounced lay-tech or lah-tech. In plain text, the typography is LaTeX.

1.2 Why LaTeX?

The main advantages of LaTeX over normal word processors are the following:

- Professionally crafted layouts are available, which make a document really look as if printed.
- The typesetting of mathematical formula is supported in a convenient way.

[1] What you see is what you get

- Users only need to learn a few easy-to-understand commands that specify the logical structure of a document. They almost never need to tinker with the actual layout of the document.

- Even complex structures such as footnotes, references, table of contents, and bibliographies can be generated easily.

- Free add-on packages exist for many typographical tasks not directly supported by basic LaTeX. For example, packages are available to include PostScript graphics or to typeset bibliographies conforming to exact standards. Many of these add-on packages are described in The *LaTeX Companion*[3].

- LaTeX encourages authors to write well-structured texts, because this is how LaTeX worksby specifying structure.

- TeX, the formatting engine of LaTeX, is highly portable and free. Therefore the system runs on almost any hardware platform available.

1.3 Installing LaTeX

1.3.1 What to Install

To use LaTeX on any computer system, you need several programs.

1. The TeX/LaTeX *program* for processing your LaTeX source files into typeset PDF or DVI documents.

2. A *text editor* for editing your LaTeX source files. Some products even let you start the LaTeX program from within the editor.

3. A PDF/DVI viewer program for previewing and printing your documents.

4. A program to handle PostScript files and images for inclusion into your documents.

For every platforms there are several programs that fit the requirements above. Here we just tell about the ones we know, like and have some experience with.

1.3.2 Cross Platform Editor

While TeX is available on many different computing platforms, LaTeX editors have long been highly platform specific.

Over the past few years I have come to like `Texmaker` quite a lot. Apart from being very a useful editor with integrated pdf-preview and syntax highlighting, it has the advantage of running on Windows, Mac and Unix/Linux equally well. See `http://www.xm1math.net/texmaker` for further information. There is also a forked version of `Texmaker` called `TeXstudio` on `http://texstudio.sourceforge.net/`. It also seems well maintained and is also available for all three major platforms.

You will find some platform specific editor suggestions in the OS sections below.

1.3.3 TeX on Windows

Getting TeX distribution

First, get a copy of the excellent MiKTeX distribution from `http://www.miktex.org/`. It contains all the basic programs and files required to compile LaTeX documents. The coolest feature in my eyes, is that MiKTeX will download missing LaTeX packages on the fly and install them magically while compiling a document.

A LaTeX editor

If you are not happy with our crossplatform suggestion `Texmaker` (section 1.3.2).

`TeXnicCenter` uses many concepts from the programming-world to provide a nice and efficient LaTeX writing environment in Windows. Get your copy from `http://www.texniccenter.org/`. `TeXnicCenter` integrates nicely with MiKTeX.

Recent MiKTeX distributions contain the *TeXworks* editor `http://texworks.org/`. It supports Unicode and requires at least Windows XP.

Document Preview

For PDF you may want to look at any PDF viewer like Foxit reader, Adobe Acrobat reader or Sumatra PDF [2]. I mention Sumatra PDF because it lets

[2] http://blog.kowalczyk.info/software/sumatrapdf/

you jump from any position in the pdf document back into corresponding position in your source document.

Working with graphics

Working with high quality graphics in LaTeX means that you have to use *Encapsulated PostScript* (eps) or PDF as your picture format. The program that helps you deal with this is called GhostScript. You can get it, together with its own front end GhostView, from `http://www.cs.wisc.edu/~ghost/`.

1.3.4 TEX on Linux

If you work with Linux, chances are high that LATEX is already installed on your system, or at least available on the installation source you used to setup. Use your package manager to install the following packages:

- *texlive* the base TEX/LATEX setup.
- *emacs* (with AUCTEX) an editor that integrates tightly with LATEX through the add-on AUCTEX package.
- *ghostscript* a PostScript preview program.
- *xpdf* and *acrobat* a PDF preview program.
- *imagemagick* a free program for converting bitmap images.
- *gimp* a free Photoshop look-a-like.
- *inkscape* a free illustrator/corel draw look-a-like.

If you are looking for a more windows like graphical editing environment, check out *TeXmaker*[3]. Apart from being very a useful editor with integrated pdf-preview and syntax highlighting, it has the advantage of running on Windows, Mac and Unix/Linux equally well.

Most Linux distros insist on splitting up their TEX environments into a large number of optional packages, so if something is missing after your first install, go check again.

[3] http://www.xm1math.net/texmaker

1.3.5 Online Solutions

To get started without needing to install anything, you can use a web-hosted service featuring a full TeXdistribution and a web LaTeX editor.

- *LaTeX Lab*[4] allows real-time simultaneous collaborative editing of text files for anyone with a Google account (and its option to make the document available through a URL makes local download and compilation easily scriptable).
- *Overleaf*[5] is a secure, easy to use online LaTeX editor with integrated rapid preview. Start writing with one click (*no signup* required) and share the link. It supports real time preview, figures, bibliographies and custom styles.

[4] http://docs.latexlab.org
[5] https://www.overleaf.com

Chapter 2

Our First Document

2.1 First Document

Now we can create our first document. We will produce the absolute bare minimum that is needed in order to get some output; the well known *Hello World!* approach will be suitable here.

- Open your favorite text-editor.
- Reproduce the following text in your editor. This is the LaTeX source:

```
% hello.tex - Our first LaTeX example!
\documentclass{article}
\begin{document}
   Hello World!
\end{document}
```

- Save your file as `hello.tex`.

When picking a name for your file, make sure it bears a `.tex` extension.

2.2 Compilation

2.2.1 Compilation process

The general concept is to transform a plain text document into a publishable format, mostly a DVI, PS or PDF file. This process is called *compilation*, which is done by an executable file called a compiler. There are two executables:

- `latex` executable reads a LaTeX `.tex` file and creates a `.dvi`
- `pdflatex` executable reads a LaTeX `.tex` file and creates a `.pdf`

2.2.2 Generating the document

I am assuming that you are at a graphical front-end where you can click LaTeX into compiling your input file. Please *note*: this description assumes that you already have a working LaTeX installation on your computer.

- Select `LaTex => PDF` from output profile.
- Click on `build and view current file` button.

There will be a pause while your document is being converted to a PDF file. When the compiling is complete PDF viewer will open and display your document. The PDF file is automatically saved in the same folder as the `.tex` file.

Once translated this code into a PDF document, you will find the text:

Hello World!

along with the page number at the bottom, which is added automatically in the article class.

2.2.3 Troubleshooting

If there is an error in your document and compiler cannot create the PDF, `Build Output` at the bottom of the screen will stay open. If this happens:

- Read the `Build Output` - the last line will probably include a line number and the command that caused the error.
- Go to the line number in your document and fix the error.
- Click on the `build and view current file` button again.

2.3 The LaTeX syntax

The input for LaTeX is a plain text file. You can create it with any text editor. It contains the text of the document, as well as the commands that tell LaTeX how to typeset the text.

2.3.1 Spaces

Whitespace characters, such as blank or tab, are treated uniformly as "space" by LaTeX. Several consecutive "spaces" are treated as one, "space" opening a line is generally ignored, and a single line break also yields "space".

A double line break (an empty line), however, defines the end of a paragraph; multiple empty lines are treated the same as one empty line. The text below is an example. On the above is the text from the input file, and in the box below is the formatted output.

```
It does not matter whether you
enter one or several             spaces
after a word.

An empty line starts a new paragraph.
```

> It does not matter whether you enter one or several spaces after a word.
> An empty line starts a new paragraph.

2.3.2 Reserved Characters

The following symbols are reserved characters that either have a special meaning under LaTeX or are unavailable in all the fonts. If you enter them directly in your text, they will normally not print but rather make LaTeX do things you did not intend.

\# \$ \% \^ \& _ \{ \} \~ \\

As you will see, these characters can be used in your documents all the same by adding a prefix backslash:

```
\# \$ \% \^{} \& \_ \{ \} \~{} \textbackslash{}
```

> # $ % ^ & _ { } ~ \

The backslash character \ cannot be entered by adding another backslash in front of it (`\\`); this sequence is used for line breaking. For introducing a backslash in math mode, you can use `\backslash` instead.

2.3.3 LaTeX groups

A *group* is basically defined by a pair of braces. The range of commands put between braces is limited to them. The `\begingroup` and `\endgroup` commands are equivalent to opening brace and closing brace.

Example:

```
{
 \bf This is bold.
}
This is no longer bold.
```

This is bold. This is no longer bold.

For some commands it is important to restrict their range of action, and that's where groups come to be very useful.

2.3.4 LaTeX commands

LaTeX *commands* are case sensitive, and take one of the following two formats:

- They start with a backslash \ and then have a name consisting of letters only. Command names are terminated by a space, a number or any other "non-letter".
- They consist of a backslash \ and exactly one non-letter.

LaTeX ignores whitespace after commands. If you want to get a space after a command, you have to put either an empty parameter `{}` and a *blank* or a special spacing command after the command name. The empty parameter `{}` stops LaTeX from eating up all the white space after the command name.

```
New \TeX users may miss whitespaces
after a command. % renders wrong
Experienced \TeX{} users are
\TeX perts, and know how to use
whitespaces. % renders correct
```

> New TeXusers may miss whitespaces after a command. Experienced TeX users are TeXperts, and know how to use whitespaces.

Some commands need an argument, which has to be given between curly braces { } after the command name. Some commands support optional parameters, which are added after the command name in square brackets [ ]. The general syntax is:

```
\commandname[option1,option2,...]{argument1}{argument2}...
```

2.3.5 LaTeX environments

Environments in LaTeX have a role that is quite similar to commands, but they usually have effect on a wider part of the document. Their syntax is:

```
\begin{environmentname}
  text to be influenced
\end{environmentname}
```

Between the `\begin` and the `\end` you can put other commands and nested environments. The internal mechanism of environments defines a group, which makes its usage safe (no influence on the other parts of the document). In general, environments can accept arguments as well, but this feature is not commonly used and so it will be discussed in more advanced parts of the document.

Anything in LaTeX can be expressed in terms of *commands* and *environments*.

2.3.6 Comments

When LaTeX encounters a % character while processing an input file, it ignores the rest of the current line, the line break, and all whitespace at the beginning of the next line. This can be used to write notes into the input file, which will not show up in the printed version.

```
This is an % stupid
% Better: instructive <----
example: Supercal%
            ifragilist%
icexpialidocious
```

This is an example: Supercalifragilisticexpialidocious

Note that the % character can be used to split long input lines that do not allow whitespace or line breaks, as with Supercalifragilisticexpialidocious above.

Chapter 3

Basic Document Structure

3.1 Global structure

When LaTeX processes an input file, it expects it to follow a certain structure. Thus every input file must contain the commands

```
\documentclass{...}

\begin{document}
...
\end{document}
```

The area between `\documentclass{...}` and `\begin{document}` is called **the preamble**. It normally contains commands that affect the entire document.

After the preamble, the text of your document is enclosed between two commands which identify the beginning and end of the actual document, i.e., **the document environment**:

```
\begin{document}
...
\end{document}
```

You would put your text where the dots are.

3.2 The Preamble

3.2.1 Document classes

When processing an input file, LaTeX needs to know the type of document the author wants to create. This is specified with the `\documentclass` command. It is recommended to put this declaration at the very beginning.

```
\documentclass[options]{class}
```

Here, `class` specifies the type of document to be created. Table 3.1 lists the document classes explained in this introduction. The `options` parameter customizes the behavior of the document class. The options have to be separated by commas. The most common options for the standard document classes are listed in Table 3.2.

Example: an input file for a LaTeX document could start with the line

```
\documentclass[11pt,twoside,a4paper]{article}
```

which instructs LaTeX to typeset the document as an article with a base font size of 11 points, and to produce a layout suitable for double sided printing on A4 paper.

Table 3.1: Document Classes.

`article`	For articles in scientific journals, presentations, short reports, program documentation, invitations, ...
`proc`	A class for proceedings based on the article class.
`minimal`	Is as small as it can get. It only sets a page size and a base font. It is mainly used for debugging purposes.
`report`	For longer reports containing several chapters, small books, thesis, ...
`book`	For real books.
`slides`	For slides. The class uses big sans serif letters.
`beamer`	For writing presentations.

3.2.2 Packages

While writing your document, you will probably find that there are some areas where basic LaTeX cannot solve your problem. If you want to include graphics, colored text or source code from a file into your document, you need to enhance the capabilities of LaTeX. Such enhancements are called packages. Some packages come with the LaTeX base distribution. Others are provided separately. Modern TeX distributions come with a large number of packages pre-installed. Packages are activated with the

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

command, where package is the name of the package and options is a list of keywords that trigger special features in the package. For example, to use the color package, which lets you typeset in colors, you would type:

```
\documentclass[11pt,a4paper,oneside]{report}

\usepackage{color}

\begin{document}
...
\end{document}
```

You can include several package names in one `\usepackage` command by separating the names with commas, like this:

```
\usepackage{package1,package2,package3}
```

and you can have more than one `\usepackage` command. Some packages allow optional settings in square brackets. If you use these, you must give the package its own separate `\usepackage` command.

Many packages can have additional formatting specifications in optional arguments in square brackets. Read the documentation for the package concerned to find out what can be done. You can pass several options together separated by a comma:

```
\usepackage[option1,option2,option3]{''package_name''}
```

3.3 The `document` environment

3.3.1 Title Creation

At the beginning of most documents there will be information about the document itself, such as the title and date, and also information about the authors, such as name, address, email etc.

A simple example:

```
\documentclass[11pt,a4paper]{report}

\begin{document}
\title{How to Structure a LaTeX Document}
\author{John Doe\\ Magic Department, Richard Miles University
        \and Richard Row, \LaTeX\ Academy}
\date{December 2004}
\maketitle
\end{document}
```

Commonly the date is excluded from the title page by using `\date{}`. It defaults to `\today` if not in the source file. The double backslash (`\\`) is the LaTeX command for forced linebreak.

To form a title page, use

```
\maketitle
```

This should go after the preceding commands.

3.3.2 Abstract

As most research papers have an abstract, there are predefined commands for telling LaTeX which part of the content makes up the abstract. This should appear in its logical order, therefore, after the title creation, but before the main sections of the body. This command is available for the document classes *article* and *report*, but not *book*.

```
\documentclass{article}

\begin{document}

\begin{abstract}
```

```
  Your abstract goes here...
  ...
\end{abstract}
  ...
\end{document}
```

By default, LaTeX will use the word "Abstract" as a title for your abstract.

3.3.3 Sections

The commands for inserting sections are fairly intuitive. Of course, certain commands are appropriate to different document classes. For example, a `book` has chapters but an `article` doesn't. Here are some of the structure commands:

```
\chapter{Introduction}
This chapter's content...

\section{Structure}
This section's content...

\subsection{Top Matter}
This subsection's content...

\subsubsection{Article Information}
This subsubsection's content...
```

Notice that you do not need to specify section numbers; LaTeX will sort that out for you. The spacing between sections, the numbering and the font size of the titles will be set automatically by LaTeX.

LaTeX provides 7 levels of depth for defining sections (see table 3.3). Each section in this table is a subsection of the one above it.

All the titles of the sections are added automatically to the *Table of Contents* (if you decide to insert one). To get an unnumbered section heading which does not go into the Table of Contents, follow the command name with an *asterisk* before the opening curly brace:

```
\subsection*{Introduction}
```

All the divisional commands from `\part*` to `\subparagraph*` have this "starred" version which can be used on special occasions for an unnumbered heading.

3.3.4 Ordinary paragraphs

Paragraphs of text come after section headings. Simply type the text and leave a *blank line* between paragraphs. The blank line means "start a new paragraph here": it does not mean you get a blank line in the typeset output. For formatting paragraph indents and spacing between paragraphs, refer to the **Paragraph Formatting** section in 4.3.

3.3.5 Table of contents

All auto-numbered headings get entered in the Table of Contents (ToC) automatically. You don't have to print a ToC, but if you want to, just add the command

```
\tableofcontents
```

at the point where you want it printed (usually after the Abstract or Summary).

Entries for the ToC are recorded each time you process your document, and reproduced the next time you process it, so you need to re-run LaTeX *one extra time* to ensure that all ToC page number references are correctly calculated.

The commands `\listoffigures` and `\listoftables` work in exactly the same way as `\tableofcontents` to automatically list all your tables and figures. If you use them, they normally go after the `\tableofcontents` command.

3.4 Book structure

The standard LaTeX `book` class follows the same layout described above with some additions. By default a book will be two-sided, i.e. left and right margins will change according to the page number parity. Furthermore current chapter and section will be printed in the header.

If you do not make use of chapters, it is barely useful to use the `book` class. A general structure of the `book` class is as follows:

```
\begin{document}
```

```
\frontmatter

\maketitle

% Introductory chapters
\chapter{Preface}
% ...

\mainmatter
\chapter{First chapter}
% ...

\appendix
\chapter{First Appendix}

\backmatter
\chapter{Last note}
\end{document}
```

- The `frontmatter` chapters will not be numbered. Page numbers will be printed in roman numerals. Frontmatter is not supposed to have sections, since they will be number 0.n because there is no chapter numbering.
- The `mainmatter` chapters works as usual. The command resets the page numbering. Page numbers will be printed in arabic numerals.
- The `\appendix` macro can be used to indicate that following sections or chapters are to be numbered as appendices. Appendices can be used for the `article` class too:

  ```
  \appendix
  \section{First Appendix}
  ```

  Only use the `\appendix` macro once for all appendices.
- The `backmatter` behaves like the `frontmatter`. It has the same issue with section numbering.

As a general rule you should avoid mixing the command order. Nonetheless all commands are optional, so you might consider using only a few.

Note that the special content like the Table of Contents is considered as an unnumbered chapter.

3.4.1 Page order

This is one traditional page order for books.

Frontmatter

1. Half-title
2. Empty
3. Title page
4. Information (copyright notice, ISBN, etc.)
5. Dedication if any, else empty
6. Table of contents
7. List of figures, tables (can be in the backmatter too)
8. Preface chapter

Mainmatter

1. Main topic

Appendix

1. Some subordinate chapters

Backmatter

1. Bibliography
2. Glossary / Index

3.5 Special pages

Comprehensive papers often feature special pages at the end, like indices, glossaries and bibliographies. Since this is a quite complex topic, we will discuss here only some basic features.

3.5.1 Bibliography

If you are writing only one or two documents and aren't planning on writing more on the same subject for a long time, you might not want to waste time creating a database of references you are never going to use. In this case you should consider using the basic and simple bibliography support that is embedded within LaTeX.

LaTeX provides an environment called `thebibliography` that you have to use where you want the bibliography; that usually means at the very end of your document, just before the `\end{document}` command. Each entry starts with

```
\bibitem[label]{marker}
```

The *marker* is then used to cite the *book, article* or *paper* within the document.

```
\cite{marker}
```

If you do not use the *label* option, the entries will get enumerated automatically. Here is a practical example:

```
LaTeX was originally written by Leslie Lamport \cite{lamport94}.

...

\begin{thebibliography}{56}

\bibitem{lamport94}
  Leslie Lamport,
  LaTeX: a document preparation system,
  Addison Wesley, Massachusetts,
  2nd edition,
  1994.

\end{thebibliography}
```

> LaTeX was originally written by Leslie Lamport [1].

...

Bibliography

[1] Leslie Lamport. LaTeX: A Document Preparation System. Addison Wesley, Massachusetts, 2nd edition, 1994.

The parameter after the `\begin{thebibliography}` command defines how much space to reserve for the number of labels. In the example above, `{56}` tells LaTeX to expect that none of the bibliography item numbers will be wider than the number 99.

3.5.2 Indexing

To enable the indexing feature of LaTeX, the `makeidx` package must be loaded in the `preamble` with:

```
\usepackage{makeidx}
```

and the special indexing commands must be enabled by putting the

```
\makeindex
```

command into the input file `preamble`.

The content of the index is specified with

```
\index{key@formatted_entry}
```

commands, where *formatted_entry* will appear in the index and *key* will be used for sorting. The *formatted_entry* is optional. If it is missing the key will be used. You enter the index commands at the points in the text that you want the final index entries to point to. For example, the text

```
To solve various problems in physics, it can be advantageous
to express any arbitrary piecewise-smooth function as a
Fourier Series composed of multiples of sine and cosine functions.
```

can be re-written as

```
To solve various problems in physics, it can be advantageous
to express any arbitrary piecewise-smooth function as a Fourier Series
```

```
\index{Fourier Series}
composed of multiples of sine and cosine functions.
```

to create an entry called 'Fourier Series' with a reference to the target page. Multiple uses of `\index` with the same *key* on different pages will add those target pages to the same index entry. Table 3.4 explains the syntax with several examples.

To show the index within the document, merely use the command

```
\printindex
```

It is common to place it at the end of the document. The default index format is two columns.

Compiling indices

When the input file is processed with LaTeX, each `\index` command writes an appropriate index entry, together with the current page number, to a special file. The file has the same name as the LaTeX input file, but a different extension (`.idx`). This `.idx` file can then be processed with the `makeindex` program:

`makeindex` *filename*

The `makeindex` program generates a sorted index with the same base file name, but this time with the extension `.ind`. If now the LaTeX input file is processed again, this sorted index gets included into the document at the point where LaTeX finds `\printindex`.

3.6 Big Projects

This is a short step-by-step guide about how to start a document properly, keeping a good high-level structure. This is all about organizing your files using the modular capabilities of LaTeX. This way it will be very easy to make modifications even when the document is almost finished.

3.6.1 Project structure

Create a clear structure of the whole project this way:

1. create a directory only for the project. We'll refer to that in the following parts as the root directory
2. create two other directories inside the root, one for LaTeX documents, the other one for images. Since you'll have to write their name quite often, choose short names. A suggestion would be simply `tex` and `img`.
3. create your document (we'll call it `document.tex`, but you can use the name you prefer) and your own package (for example `mystyle.sty`); this second file will help you to keep the code cleaner.

If you followed all those steps, these files should be in your root directory, using `"/"` for each directory:

```
./document.tex
./mystyle.sty
./tex/
./img/
```

nothing else.

3.6.2 Getting LaTeX to process multiple files

As your work grows, your LaTeX file can become unwieldy and confusing, especially if you are writing a long article with substantial, discrete sections, or a full-length book. In such cases it is good practice to split your work into several files. For example, if you are writing a book, it makes a lot of sense to write each chapter in a separate `.tex` file. LaTeX makes this very easy thanks to two commands:

```
\input{filename}
```

and

```
\include{filename}
```

Compiling the base file

When you compile your document, page references and the like will change according to your use of the `\input` and `\include` commands. Normally LaTeX users only run the compiler on parts of the document to check that an

individual chapter is syntactically correct and looks as the writer intended. A full run is generally only performed for producing a full draft or the final version. In such cases, it is invariably necessary to run LaTeX twice or more to resolve all the page numbers, references, etc. (especially if you are using bibliographic software such as BiBTeX, too).

The simplest way to check that one or more of the various components of your work is syntactically robust, is to comment out the command with a percentage sign, for example:

```
\documentclass{article}
\begin{document}
%\input{Section_1}
%\input{Section_2}
%\input{Section_3}
\input{Section_4}
%\input{Section_5}
\end{document}
```

This code will process your base file with the article conventions but only the material in the file `Section_4.tex` will be processed. If that was, say, the last thing you needed to check before sending off to that major journal, you would then simply remove all the percentage signs and rerun LaTeX, repeating the compiling process as necessary to resolve all references, page numbers and so on.

Separate compilation of child documents

A disadvantage of solely using `\input` and `\include` is that only the base document can be compiled. However, you may decide that you work better on individual sections of text and wish to edit and compile those separate from the main file. There are a few packages available to address this problem.

Standalone

The *standalone* package provides a means for importing the preamble of child documents into the main document, allowing for a flexible way to include text or images in multiple documents (e.g. an `article` and a `presentation`).

In the main document, the package must be loaded as:

```
\usepackage{standalone}
```

in the preamble. Child documents are loaded using `\input` or `\include`.

The child documents contain, for example, the following statements:

```
\documentclass{standalone}
% Load any packages needed for this document
\begin{document}
% Your document or picture
\end{document}
```

The file `mystyle.sty`

Instead of putting all the packages you need at the beginning of your document as you could, the best way is to load all the packages you need inside another dummy package called mystyle you will create just for your document. The good point of doing this is that you will just have to add one single

```
\usepackage{standalone}
```

in your document, keeping your code much cleaner. Moreover, all the info about your style will be within one file, so when you will start another document you'll just have to copy that file and include it properly, so you'll have exactly the same style you have used.

Creating your own style is very simple: create a file called `mystyle.sty` (you could name it as you wish, but it has to end with ".sty"). Write at the beginning:

```
\ProvidesPackage{mystyle}
```

Then add all the packages you want with the standard command `\usepackage{...}` as you would do normally, change the value of all the variables you want, etc. It will work like the code you put here would be copied and pasted within your document.

The main document `document.tex`

Then create a file called `document.tex`; this will be the main file, the one you will compile, even if you shouldn't need to edit it very often because you will be working on other files. It should look like this:

```
\documentclass[12pt,a4paper]{report}

\usepackage{mystyle}
```

```
\begin{document}

\input{./tex/title.tex}
%\maketitle
\tableofcontents
\listoffigures
\listoftables

\input{./tex/intro.tex}
\input{./tex/main_part.tex}
\input{./tex/conclusions.tex}

\appendix
\input{./tex/myappendix.tex}

% Bibliography:
\input{./tex/mybibliography.tex}

\end{document}
```

Table 3.2: Document Class Options.

`10pt, 11pt, 12pt`	Sets the size of the main font in the document. If no option is specified, `10pt` is assumed.
`a4paper, letterpaper,...`	Defines the paper size. The default size is `letterpaper`; Besides that, `a5paper, b5paper, executivepaper`, and `legalpaper` can be specified.
`fleqn`	Typesets displayed formulas left-aligned instead of centered.
`leqno`	Places the numbering of formulas on the left hand side instead of the right.
`titlepage, notitlepage`	Specifies whether a new page should be started after the document title or not. The `article` class does not start a new page by default, while `report` and `book` do.
`twocolumn`	Instructs LaTeX to typeset the document in two columns instead of one.
`twoside, oneside`	Specifies whether double or single sided output should be generated. The classes `article` and `report` are `single` sided and the book class is double sided by default. Note that this option concerns the style of the document only. The option `twoside` does *not* tell the printer you use that it should actually make a two-sided printout.
`landscape`	Changes the layout of the document to print in landscape mode.
`openright, openany`	Makes chapters begin either only on right hand pages or on the next page available. This does not work with the `article` class, as it does not know about chapters. The `report` class by default starts chapters on the next page available and the `book` class starts them on right hand pages.

Table 3.3: Levels of depth for defining sections.

Command	**Level**	**Comment**
`\part{''part''}`	-1	not in `letters`
`\chapter{''chapter''}`	0	only `books` and `reports`
`\section{''section''}`	1	not in `letters`
`\subsection{''subsection''}`	2	not in `letters`
`\subsubsection{''subsubsection''}`	3	not in `letters`
`\paragraph{''paragraph''}`	4	not in `letters`
`\subparagraph{''subparagraph''}`	5	not in `letters`

Table 3.4: Index Key Syntax Examples.

Example	Index Entry	Comment
`\index{hello}`	hello, 1	Plain entry
`\index{hello!Peter}`	Peter, 3	Subentry under 'hello'
`\index{Sam@\textbf{Sam}}`	**Sam**, 2	Formatted entry
`\index{Jenny textbf}`	Jenny, **3**	Formatted page number
`\index{ecole@\'ecole}`	école, 4	Handling of accents

Chapter 4

Typesetting Text

4.1 Page styles

Page styles in LaTeX terms refers not to page dimensions, but to the running *headers* and *footers* of a document. These headers typically contain document titles, chapter or section numbers/names, and page numbers.

4.1.1 Standard page styles

The possibilities of changing the headers in plain LaTeX are actually quite limited. There are two commands available:

```
\pagestyle{‘‘style’’}
```

will apply the specified style to the current and all subsequent pages, and

```
\thispagestyle{‘‘style’’}
```

will only affect the current page. The possible styles are shown in Table 4.1.

4.2 Text Formatting

4.2.1 Spacing

Line Spacing

If you want to use larger inter-line spacing in a document, you can change its value by putting the

Table 4.1: The Predefined Page Styles of LaTeX.

`empty`	Both header and footer are cleared.
`plain`	Header is clear, but the footer contains the page number in the center.
`headings`	Footer is blank, header displays information according to document class (e.g., section name) and page number top right.
`myheadings`	Page number is top right, and it is possible to control the rest of the header.

```
\linespread{factor}
```

command into the preamble of your document. Use `\linespread{1.3}` for "one and a half" line spacing, and `\linespread{1.6}` for "double" line spacing. Normally the lines are not spread, so the default line spread factor is 1.

The `setspace` package allows more fine-grained control over line spacing. To set "one and a half" line spacing document-wide, but not where it is usually unnecessary (e.g. footnotes, captions):

```
\usepackage{setspace}
%\singlespacing
\onehalfspacing
%\doublespacing
%\setstretch{1.1}
```

To change line spacing within the document, the `setspace` package provides the environments `singlespace`, `onehalfspace`, `doublespace` and `spacing`:

```
This paragraph has \\ default \\ line spacing.

\begin{doublespace}
  This paragraph has \\ double \\ line spacing.
\end{doublespace}

\begin{spacing}{2.5}
  This paragraph has \\ huge gaps \\ between lines.
\end{spacing}
```

Horizontal Space

LaTeX determines the spaces between words and sentences automatically. To add horizontal space, use:

```
\hspace{length}
```

If such a space should be kept even if it falls at the end or the start of a line, use `\hspace*` instead of `\hspace`.

```
This\hspace{1.5cm}is a space
of 1.5 cm.
```

This is a space of 1.5 cm.

Vertical Space

The space between paragraphs, sections, subsections, ... is determined automatically by LaTeX. If necessary, additional vertical space *between two paragraphs* can be added with the command:

```
\vspace{length}
```

This command should normally be used between two empty lines. If the space should be preserved at the top or at the bottom of a page, use the starred version of the command, `\vspace*`, instead of `\vspace`.

Additional space between two lines of *the same* paragraph or within a table is specified with the

```
\\[length]
```

command.

With `\bigskip` and `\smallskip` you can skip a predefined amount of vertical space without having to worry about exact numbers.

4.2.2 Fonts and Sizes

LaTeX chooses the appropriate font and font size based on the logical structure of the document (sections, footnotes, ...). In some cases, one might like to change fonts and sizes by hand. To do this, use the commands listed in

Tables 4.2 and 4.3. The actual size of each font is a design issue and depends on the document class and its options. Table 4.4 shows the absolute point size for these commands as implemented in the standard document classes.

```
{\small The small and
\textbf{bold} Romans ruled}
{\Large all of great big
\textit{Italy}.}
```

The small and **bold** Romans ruled all of great big *Italy*.

Table 4.2: Fonts.

`\textrm{...}`	roman	`\textsf{...}`	sans serif
`\texttt{...}`	`typewriter`		
`\textmd{...}`	medium	`\textbf{...}`	**bold face**
`\textup{...}`	upright	`\textit{...}`	*italic*
`\textsl{...}`	*slanted*	`\textsc{...}`	Small Caps
`\emph{...}`	*emphasized*	`\textnormal{...}`	document font

Table 4.3: Font Sizes.

`\tiny`	tiny font	`\Large`	larger font
`\scriptsize`	very small font		
`\footnotesize`	quite small font	`\LARGE`	very large font
`\small`	small font		
`\normalsize`	normal font	`\huge`	huge
`\large`	large font	`\Huge`	largest

One important feature of LaTeX is that the font attributes are independent. This means that issuing size or even font changing commands, and still keep bold or slant attributes set earlier.

Table 4.4: Absolute Point Sizes in Standard Classes.

size	10pt (default)	11pt option	12pt option
`\tiny`	5pt	6pt	6pt
`\scriptsize`	7pt	8pt	8pt
`\footnotesize`	8pt	9pt	10pt
`\small`	9pt	10pt	11pt
`\normalsize`	10pt	11pt	12pt
`\large`	12pt	12pt	14pt
`\Large`	14pt	14pt	17pt
`\LARGE`	17pt	17pt	20pt
`\huge`	20pt	20pt	25pt
`\Huge`	25pt	25pt	25pt

In *math mode* use the font changing *commands* to temporarily exit *math mode* and enter some normal text. If you want to switch to another font for math typesetting you need another special set of commands; refer to Table 4.5.

Table 4.5: Math Fonts.

`\mathrm{...}`	RomanFont
`\mathbf{...}`	**BoldfaceFont**
`\mathsf{...}`	SansSerifFont
`\mathtt{...}`	`TypewriterFont`
`\mathit{...}`	*ItalicFont*
`\mathcal{...}`	$\mathcal{CALLIGRAPHICFONT}$
`\mathnormal{...}`	$NormalFont$

In connection with the font size commands, curly braces play a significant role. They are used to build *groups*. Groups limit the scope of most LaTeX commands.

```
He likes {\LARGE large and
```

```
{\small small} letters}.
```

He likes large and small letters.

If you want to activate a size changing command for a whole paragraph of text or even more, you might want to use the environment syntax for font changing commands.

```
\begin{Large}
This is not true.
But then again, what is these
days \ldots
\end{Large}
```

This is not true. But then again, what is these days ...

This will save you from counting lots of curly braces.

4.2.3 Text mode superscript and subscript

To superscript text in text-mode, you can use the `\textsuperscript{}` command. This allows you to, for instance, typeset 6th as 6^{th}:

```
Michelangelo was born on March 6\textsuperscript{th}, 1475.
```

Subscripting in text-mode is not supported by LaTeX alone; however, several packages allow the use of the `\textsubscript{}` command.

4.2.4 Dashes and hyphens

LaTeX knows *four* kinds of dashes: a *hyphen* (-), *en dash* (–), *em dash* (—), or a *minus sign* (−). You can access three of them with different numbers of consecutive dashes. The fourth sign is actually not a dash at all, it is the mathematical minus sign:

```
Hyphen: daughter-in-law, X-rated\\
En dash: pages 13--67\\
Em dash: yes---or no? \\
Minus sign: $0$, $1$ and $-1$
```

Hyphen: daughter-in-law, X-rated
En dash: pages 13–67
Em dash: yes—or no?
Minus sign: 0, 1 and −1

4.2.5 Ellipsis (...)

A sequence of three dots is known as an *ellipsis*, which is commonly used to indicate omitted text. There is a special command for these dots. It is called `\ldots`:

```
Not like this ... but like this:\\
New York, Tokyo, Budapest, \ldots
```

Not like this ... but like this:
New York, Tokyo, Budapest, ...

4.3 Paragraph Formatting

4.3.1 Paragraph alignment

Paragraphs in LaTeX are usually fully justified, *i.e.* flush with both the left and right margins. For whatever reason, should you wish to alter the justification of a paragraph, there are *three* environments at hand, and also LaTeX command equivalents.

Alignment	Environment	Command
Left justified	`flushleft`	`\raggedright`
Right justified	`flushright`	`\raggedleft`
Center	`center`	`\centering`

All text between the `\begin` and `\end` of the specified environment will be justified appropriately. The commands listed are for use within other environments. For example, `p` (paragraph) columns in `tabular`.

There is no way (in standard LaTeX) to set full justification explicitly. It means that if you do not enclose the previous 3 commands into a group, the rest of the document will be affected.

So the right way of doing this with commands is

```
\raggedleft{Some text flushed right.}
```

Some text flushed right.

4.3.2 Paragraph indent and break

By default, the first paragraph after a heading follows the standard Anglo-American publishers' practice of no indentation. The size of subsequent paragraph indents is determined by a parameter called `\parindent`. The default length that this constant holds is set by the document class that you use. It is possible to override it by using the `\setlength` command. This will set paragraph indents to 1cm:

```
\setlength{\parindent}{1cm} % Default is 15pt.
```

Whitespace in LaTeX can also be made flexible (what Lamport calls "rubber" lengths). This means that values such as extra vertical space inserted before a paragraph `\parskip` can have a default dimension plus an amount of expansion minus an amount of contraction. This is useful on pages in complex documents where not every page may be an exact number of fixed-height lines long, so some give-and-take in vertical space is useful. You specify this in a `\setlength` command like this:

```
\setlength{\parskip}{1cm plus4mm minus3mm}
```

If you want to indent a paragraph that is not indented, you can use

```
\indent
```

at the beginning of the paragraph. Obviously, this will only have an effect when `\parindent` is not set to zero. If you want to indent the beginning of every section, you can use the `indentfirst` package: once loaded, the beginning of any chapter/section is indented by the usual paragraph indentation.

To create a non-indented paragraph, you can use

```
\noindent
```

as the first command of the paragraph. This might come in handy when you start a document with body text and not with a sectioning command.

4.4 Printing Verbatim

Text that is enclosed between `\begin{verbatim}` and `\end{verbatim}` will be directly printed, as if typed on a typewriter, with all line breaks and spaces, without any LaTeX command being executed.

Within a paragraph, similar behavior can be accessed with

```
\verb+text+
```

The + is just an example of a delimiter character. Use any character except `letters, *` or `space`. Many LaTeX examples in this booklet are typeset with this command.

```
The \verb|\ldots| command \ldots
```

```
   \begin{verbatim}
10 PRINT "HELLO WORLD ";
20 GOTO 10
\end{verbatim}
```

The `\ldots` command ...

```
10 PRINT "HELLO WORLD ";
20 GOTO 10
```

The `verbatim` environment and the `\verb` command may not be used within parameters of other commands.

4.5 List structures

Lists often appear in documents, especially academic, as their purpose is often to present information in a clear and concise fashion. List structures in LaTeX are simply environments which essentially come in *three* flavors: `itemize`, `enumerate` and `description`.

All lists follow the basic format:

```
\begin{list_type}

  \item The first item
  \item The second item
  \item The third etc \ldots

\end{list_type}
```

All three of these types of lists can have multiple paragraphs per item: just type the additional paragraphs in the normal way, with a blank line between each. So long as they are still contained within the enclosing environment, they will automatically be indented to follow underneath their item.

Itemize

This environment is for your standard bulleted list of items.

```
\begin{itemize}
  \item The first item
  \item The second item
  \item The third etc \ldots
\end{itemize}
```

- The first item
- The second item
- The third etc ...

Enumerate

The enumerate environment is for ordered lists, where by default, each item is numbered sequentially.

```
\begin{enumerate}
  \item The first item
  \item The second item
  \item The third etc \ldots
\end{enumerate}
```

1. The first item
2. The second item
3. The third etc ...

Description

The description environment is slightly different. You can specify the item label by passing it as an optional argument (although optional, it would look odd if you didn't include it!). Ideal for a series of definitions, such as a glossary.

```
\begin{description}
  \item[First] The first item
  \item[Second] The second item
  \item[Third] The third etc \ldots
\end{description}
```

First The first item

Second The second item

Third The third etc ...

4.5.1 Nested lists

LaTeX will happily allow you to insert a list environment into an existing one (up to a depth of *four*). Simply begin the appropriate environment at the desired point within the current list. Latex will sort out the layout and any numbering for you.

```
\begin{enumerate}
  \item The first item
  \begin{enumerate}
    \item Nested item 1
```

```
    \item Nested item 2
  \end{enumerate}
  \item The second item
  \item The third etc \ldots
\end{enumerate}
```

1. The first item
   (a) Nested item 1
   (b) Nested item 2
2. The second item
3. The third etc ...

4.6 Tables

Tables are a common feature in academic writing, often used to summarize research results. Mastering the art of table construction in LaTeX is therefore necessary to produce quality papers and with sufficient practice one can print beautiful tables of any kind.

Once you are comfortable with basic LaTeX tables, you might have a look at more advanced packages or the export options of your favorite spreadsheet.

4.6.1 The `tabular` environment

The `tabular` environment can be used to typeset tables with optional horizontal and vertical lines. LaTeX determines the width of the columns automatically.

The first line of the environment has the form:

```
\begin{tabular}[pos]{table spec}
```

The `table spec` argument tells LaTeX the *alignment* to be used in each column and the *vertical lines* to insert.

The number of columns does not need to be specified as it is inferred by looking at the number of arguments provided. It is also possible to add

vertical lines between the columns here. The following symbols are available to describe the table columns (some of them require that the package `array` has been loaded):

`l`	left-justified column
`c`	centered column
`r`	right-justified column
`p{'width'}`	paragraph column with text vertically aligned at the top
`m{'width'}`	paragraph column with text vertically aligned in the middle (requires `array` package)
`b{'width'}`	paragraph column with text vertically aligned at the bottom (requires `array` package)
`|`	vertical line
`||`	double vertical line

By default, if the text in a column is too wide for the page, LaTeX won't automatically wrap it. Using `p{'width'}` you can define a special type of column which will wrap-around the text as in a normal paragraph. You can pass the width using any unit supported by LaTeX, such as 'pt' and 'cm', or *command lengths*, such as `\textwidth`.

The optional parameter `pos` can be used to specify the vertical position of the table relative to the baseline of the surrounding text. You can use the following letters:

`b`	bottom
`c`	center (default)
`t`	top

In the first line you have pointed out how many columns you want, their alignment and the vertical lines to separate them. Once in the environment, you have to introduce the text you want, separating between cells and introducing new lines. The commands you have to use are the following:

`&`	column separator
`\\`	start new row (additional space may be specified after `\\` using square brackets, such as `\\[6pt]`)
`\hline`	horizontal line
`\newline`	start a new line within a cell (in a paragraph column)
`\cline{i-j}`	partial horizontal line beginning in column i and ending in column j

Note, any white space inserted between these commands is purely down to ones' preferences. I personally add spaces between to make it easier to read.

Basic examples

This example shows how to create a simple table in LaTeX. It is a three-by-three table, but without any lines.

```
\begin{tabular}{ l c r }
  1 & 2 & 3 \\
  4 & 5 & 6 \\
  7 & 8 & 9 \\
\end{tabular}
```

1	2	3
4	5	6
7	8	9

Expanding upon that by including some vertical lines:

```
\begin{tabular}{ l | c || r }
  1 & 2 & 3 \\
  4 & 5 & 6 \\
  7 & 8 & 9 \\
\end{tabular}
```

1	2	3
4	5	6
7	8	9

To add horizontal lines to the very top and bottom edges of the table:

```
\begin{tabular}{ l | c || r }
  \hline
  1 & 2 & 3 \\
  4 & 5 & 6 \\
  7 & 8 & 9 \\
  \hline
\end{tabular}
```

1	2	3
4	5	6
7	8	9

And finally, to add lines between all rows, as well as centering:

```
\begin{center}
  \begin{tabular}{ l | c || r }
    \hline
    1 & 2 & 3 \\ \hline
    4 & 5 & 6 \\ \hline
    7 & 8 & 9 \\
    \hline
  \end{tabular}
\end{center}
```

1	2	3
4	5	6
7	8	9

```
\begin{center}
  \begin{tabular}{ | l || c ||| r }
    \hline
    1 & 2 & 3 \\ \hline
    4 & 5 & 6 \\ \hline \hline
    7 & 8 & 9 \\
    \hline
  \end{tabular}
\end{center}
```

1	2	3
4	5	6
7	8	9

```
\begin{tabular}{|r|l|}
  \hline
  7C0 & hexadecimal \\
  3700 & octal \\ \cline{2-2}
  11111000000 & binary \\
  \hline \hline
  1984 & decimal \\
  \hline
\end{tabular}
```

7C0	hexadecimal
3700	octal
11111000000	binary
1984	decimal

4.6.2 Spanning

Rows spanning multiple columns

The command for this looks like this:

`\multicolumn{'num_cols'}{'alignment'}{'contents'}`.

`num_cols` is the number of subsequent columns to merge; `alignment` is either `l, c, r`, or to have text wrapping specify a width `p{5.0cm}` . And `contents` is simply the actual data you want to be contained within that cell. A simple example:

```
\begin{tabular}{ |l|l| }
  \hline
  \multicolumn{2}{|c|}{Team sheet} \\
  \hline
  GK & Paul Robinson \\
  LB & Lucus Radebe \\
  DC & Michael Duberry \\
  DC & Dominic Matteo \\
  \hline
\end{tabular}
```

Team sheet	
GK	Paul Robinson
LB	Lucus Radebe
DC	Michael Duberry
DC	Dominic Matteo

Columns spanning multiple rows

The first thing you need to do is add

`\usepackage{multirow}`

to the preamble. This then provides the command needed for spanning rows:

`\multirow{''num_rows''}{''width''}{''contents''}`.

The arguments are pretty simple to deduce (* for the *width* means the content's natural width).

```
...
\usepackage{multirow}
```

```
...

\begin{tabular}{ |l|l|l| }
\hline
\multicolumn{3}{ |c| }{Team sheet} \\
\hline
Goalkeeper & GK & Paul Robinson \\ \hline
\multirow{4}{*}{Defenders} & LB & Lucus Radebe \\
 & DC & Michael Duburry \\
 & DC & Dominic Matteo \\
 & RB & Didier Domi \\ \hline
\multirow{3}{*}{Midfielders} & MC & David Batty \\
 & MC & Eirik Bakke \\
 & MC & Jody Morris \\ \hline
Forward & FW & Jamie McMaster \\ \hline
\multirow{2}{*}{Strikers} & ST & Alan Smith \\
 & ST & Mark Viduka \\
\hline
\end{tabular}
```

Team sheet		
Goalkeeper	GK	Paul Robinson
Defenders	LB	Lucus Radebe
	DC	Michael Duburry
	DC	Dominic Matteo
	RB	Didier Domi
Midfielders	MC	David Batty
	MC	Eirik Bakke
	MC	Jody Morris
Forward	FW	Jamie McMaster
Strikers	ST	Alan Smith
	ST	Mark Viduka

The main thing to note when using `\multirow` is that a blank entry must be inserted for each appropriate cell in each subsequent row to be spanned.

Spanning in both directions simultaneously

Here is a nontrivial example of how to use spanning in both directions simultaneously and have the borders of the cells drawn correctly:

```
\usepackage{multirow}

\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} } &
\multicolumn{1}{ |c| }{504} & 3 & 2 & 0 & 1 &     \\ \cline{2-6}
\multicolumn{1}{ |c  }{}                          &
\multicolumn{1}{ |c| }{540} & 2 & 3 & 1 & 0 &     \\ \cline{1-6}
\multicolumn{1}{ |c  }{\multirow{2}{*}{Powers} } &
\multicolumn{1}{ |c| }{gcd} & 2 & 2 & 0 & 0 & min \\ \cline{2-6}
\multicolumn{1}{ |c  }{}                          &
\multicolumn{1}{ |c| }{lcm} & 3 & 3 & 1 & 1 & max \\ \cline{1-6}
\end{tabular}
```

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

The command `\multicolumn{1}{` is just used to draw vertical borders both on the left and on the right of the cell. Even when combined with `\multirow{2}{*}{...}`, it still draws vertical borders that only span the first row. To compensate for that, we add `\multicolumn{1}{` in the following rows spanned by the multirow. Note that we cannot just use `\hline` to draw horizontal lines, since we do not want the line to be drawn over the text that spans several rows. Instead we use the command `\cline{2-6}` and opt out the first column that contains the text "Powers".

Here is another example exploiting the same ideas to make the familiar and popular "2x2" or double dichotomy:

```
\begin{tabular}{ r|c|c| }
```

```
\multicolumn{1}{r}{}
 &  \multicolumn{1}{c}{noninteractive}
 & \multicolumn{1}{c}{interactive} \\
\cline{2-3}
massively multiple & Library & University \\
\cline{2-3}
one-to-one & Book & Tutor \\
\cline{2-3}
\end{tabular}
```

	noninteractive	interactive
massively multiple	Library	University
one-to-one	Book	Tutor

4.7 Importing Graphics

There are two possibilities to include graphics in your document. Either create them with some special code, a topic which will not be discussed here or import productions from *third party tools*, which is what we will be discussing here.

4.7.1 The `graphicx` package

LaTeX can't manage pictures directly, so we will need some extra help: we have to load the `graphicx` package in the preamble of our document:

```
\usepackage{graphicx}
```

4.7.2 Supported image formats

Using `pdflatex` will be usually much more simple for graphics inclusion as it supports widespread formats such as PDF, PNG and JPG. The only format you can include while compiling with `latex` is *Encapsulated PostScript* (EPS).

Table 4.6: Key Names for graphicx Package.

`width`	scale graphic to the specified width
`height`	scale graphic to the specified height
`angle`	rotate graphic counterclockwise
`scale`	scale graphic

4.7.3 Including graphics

After you have loaded the `graphicx` package in your preamble, you can include images with `\includegraphics`, whose syntax is the following:

```
\includegraphics[key1=val1, key2=val2, ..., keyn=valn]{imagename}
```

The argument in the curly braces is the name of the image. Write it without the extension. This way the LaTeX compiler will look for any supported image format in that directory and will take the best one (EPS if the output is DVI; JPEG, PNG or PDF if the output is PDF).

The optional parameter accepts a comma separated list of *keys* and associated *values*. The keys can be used to alter the width, height and rotation of the included graphic. Table 4.6 lists the most important keys.

The following example code may help to clarify things:

```
\includegraphics[scale=0.1]{./img/tulips}
```

It includes the graphic stored in the file `tulips.png`. The graphic is scaled down to 0.1. The angle, width and height parameters can also be specified in absolute dimensions.

4.8 Floats

Floats are containers for things in a document that cannot be broken over a page. LaTeX by default recognizes "table" and "figure" floats, but you can

define new ones of your own. Floats are there to deal with the problem of the object that won't fit on the present page, and to help when you really don't want the object here just now.

4.8.1 The `figure` environment

To create a figure that floats, use the `figure` environment.

```
\begin{figure}[placement specifier]
 \centering
... figure contents ...
 \caption{this is a figure}
 \label{fig:myfirstfigure}
\end{figure}
```

Floats are used to allow LaTeX to handle figures and add a label and caption to the figure. The *placement specifier* parameter (Table 4.8) gives the author a greater degree of control over where certain floats are placed. Here is an example that shows the figure in section 4.7.3 becomes float:

```
\begin{figure}
 \centering
   \fbox{\includegraphics[scale=0.1]{./img/tulips}}
 \caption{Tulips}
 \label{fig:tulips}
\end{figure}
```

Figure 4.1: Tulips

4.8.2 The `table` environment

To tell LaTeX we want to use our table as a float, we need to place a `tabular` environment in a `table` environment, which is able to float and add a label and caption.

Please understand: you do not have to use floating tables. If you want to place your tables where they lie in your source code and you do not need any label, do not use `table` at all! This is a very common misunderstanding among newcomers.

The environment names may now seem quite confusing. Let's sum it up:

- `tabular` is for the content itself (columns, lines, etc.).
- `table` is for the location of the table on the document, plus caption and label support.

```
\begin{table}[placement specifier]
  \centering
  \begin{tabular}{|l|}
    ... your table ...
  \end{tabular}
  \caption{This table shows some data}
  \label{tab:myfirsttable}
\end{table}
```

In the table, we used a label, so now we can refer to it just like any other reference:

```
\ref{tab:myfirsttable}
```

Centering the table horizontally works like everything else, using the `\centering` command just after opening the `table` environment, or by enclosing it with a `center` environment. Here is an example that shows a table in section 4.6.1 becomes float:

```
\begin{table}
\caption{A simple table}
\centering
\begin{tabular}{ l c r }
  1 & 2 & 3 \\
  4 & 5 & 6 \\
  7 & 8 & 9 \\
\end{tabular}
\label{tab:simple_table}
\end{table}
```

Table 4.7: A simple table

1	2	3
4	5	6
7	8	9

Table 4.8: Float Placement Specifiers.

`h`	where the table is declared (here)
`t`	at the top of the page
`b`	at the bottom of the page
`p`	on a dedicated page of floats
`!`	override the default float restrictions

You can set the optional parameter `placement specifier` to define the position of the floats (`table`, `figure`), where it should be placed. The characters shown in table 4.8 are all possible placements. Using sequences of it define your "wishlist" to LaTeX.

Default is `tbp`, which means that it is by default placed on the top of the page. If that's not possible, it's placed at the bottom if possible, or finally with other floating environments on an extra page.

4.8.3 Captions

It is always good practice to add a caption to any figure or table. Fortunately, this is very simple in LaTeX. All you need to do is use the `\caption{''text''}` command within the float environment. LaTeX will automatically keep track of the numbering of figures and tables, so you do not need to include this within the caption text.

4.8.4 Lists of figures and tables

Captions can be listed at the beginning of a paper or report in a "List of Tables" or a "List of Figures" section by using the `\listoftables` or `\listoffigures` commands, respectively. The caption used for each table and figure will appear in these lists, along with the their numbers, and page numbers that they appear on.

4.9 Labels and Cross-referencing

In LaTeX you can easily reference almost anything that is numbered (sections, tables, figures, formulas), and LaTeX will take care of numbering, updating it whenever necessary. The commands to be used do not depend on what you are referencing, and they are:

```
\label{marker}
```

give the object you want to reference a *marker*, you can see it like a name.

```
\ref{marker}
```

can reference the object you have *marked* before. This prints the number that was assigned to the object.

```
\pageref{marker}
```

will print the number of the page where the object is.

For example, you could write something like:

```
See figure~\ref{fig:tulips} on page~\pageref{fig:tulips}.
```

> See figure 4.1 on page 52.

LaTeX will calculate the right numbering for the objects in the document; the *marker* you have used to label the object will not be shown anywhere in the document. Then LaTeX will replace the string `"\ref{marker}"` with the right number that was assigned to the object. When you use references, you have to compile your document *twice* to see the proper output.

4.10 Footnotes

The footnote facility is easy to use. The command you need is:

```
\footnote{text}
```

Do not leave a space between the command and the word where you wish the footnote marker to appear, otherwise LaTeX will process that space

and will leave the output not looking as intended.

```
Creating a footnote is easy.\footnote{An example footnote.}
```

Creating a footnote is easy.[1]
.
.
.

[1] An example footnote.

LaTeX will obviously take care of typesetting the footnote at the bottom of the page. Each footnote is numbered sequentially - a process that is automatically done for you.

Chapter 5

Typesetting Mathematics

Typesetting mathematics is one of LaTeX's greatest strengths. It is also a large topic due to the existence of so much mathematical notation.

5.1 The `amsmath` package

If your document requires only a few simple mathematical formulas, plain LaTeX has most of the tools that you will need. If you are writing a scientific document that contains numerous complicated formulas, the `amsmath` package [1] introduces several new commands that are more powerful and flexible than the ones provided by LaTeX. To use the package, include:

```
\usepackage{amsmath}
```

in the preamble of the document.

5.2 Mathematics environments

The mathematics environments can be distinguished into *two* categories depending on how they are presented:

- *text* — text formulas are displayed *inline*, that is, within the body of text where it is declared, for example, I can say that a + a = 2a within this sentence.
- *displayed* — displayed formulas are separate from the main text.

[1] http://www.ams.org/publications/authors/tex/amslatex

The table 5.1 summarizes the mathematics environments. Unlike most other environments, however, there are some handy shorthands to declaring your formulas.

Table 5.1: The mathematics environments name and shorthands.

Type	**Text formulas**	**Displayed equations**	**Displayed and automatically numbered equations**
Environment	`math`	`displaymath`	`equation`
LaTeX shorthand	`\(...\)`	`\[...\]`	
TeX shorthand	`$...$`	`$$...$$`	
Comment			`equation*` suppresses numbering

Suggestion: Using the `$$...$$` should be avoided, as it may cause problems, particularly with the AMS-LaTeX macros. Furthermore, should a problem occur, the error messages may not be helpful.

The `equation*` and `displaymath` environments are functionally equivalent, both suppresses numbering.

If you are typing text normally, you are said to be in *text mode*, but while you are typing within one of those mathematical environments, you are said to be in *math mode*, that has some differences compared to the *text mode*:

1. *Most spaces and line breaks do not have any significance*, as all spaces are either derived logically from the mathematical expressions, or have to be specified with special commands such as `\,`, `\quad` or `\qquad` (we'll get back to that later, in section 5.3.10)

2. *Empty lines are not allowed.* Only one paragraph per formula.

3. *Each letter is considered to be the name of a variable and will be typeset as such.* If you want to typeset normal text within a formula (normal upright font and normal spacing) then you have to enter the text using `\text{...}` commands.

5.3 Building Blocks of a Mathematical Formula

In this section, we describe the most important commands used in mathematical typesetting.

5.3.1 Greek letters

Lowercase Greek letters are entered as `\alpha, \beta, \gamma`, . . . , and *Uppercase* letters are entered as `\Gamma, \Delta`, . . .

```
\alpha, \beta, \gamma, \Gamma, \pi, \Pi, \phi, \varphi, \Phi
```

$$\boxed{\alpha, \beta, \gamma, \Gamma, \pi, \Pi, \phi, \varphi, \Phi}$$

5.3.2 Powers and indices

Powers and *indices* are equivalent to superscripts and subscripts in normal text mode. The caret (^) character is used to raise something, and the underscore (_) is for lowering. If more than one expression is raised or lowered, they should be grouped using curly braces ({ and }).

```
k_{n+1} = n^2 + k_n^2 - k_{n-1}
```

$$\boxed{k_{n+1} = n^2 + k_n^2 - k_{n-1}}$$

For powers with more than one digit, surround the power with {}.

```
n^{22}
```

$$\boxed{n^{22}}$$

An underscore (_) can be used with a vertical bar (|) to denote evaluation using subscript notation in mathematics:

```
f(n) = n^5 + 4n^2 + 2 |_{n=17}
```

$$\boxed{f(n) = n^5 + 4n^2 + 2|_{n=17}}$$

5.3.3 Operators

An *operator* is a function that is written as a word: e.g. trigonometric functions (sin, cos, tan), logarithms and exponentials (log, exp). LaTeX has many of these defined as commands:

```
\cos (2\theta) = \cos^2 \theta - \sin^2 \theta
```

$$\cos(2\theta) = \cos^2\theta - \sin^2\theta$$

For certain operators such as *limits*, the subscript is placed underneath the operator:

```
\lim_{x \to \infty} \exp(-x) = 0
```

$$\lim_{x \to \infty} \exp(-x) = 0$$

For the *modular operator* there are two commands: `\bmod` and `\pmod`:

```
a \bmod b
```

$$a \bmod b$$

```
x \equiv a \pmod b
```

$$x \equiv a \pmod b$$

5.3.4 Fractions and Binomials

A fraction is created using the `\frac{numerator}{denominator}` command. Likewise, the *binomial coefficient* may be written using the `\binom` command:

```
\frac{n!}{k!(n-k)!} = \binom{n}{k}
```

$$\frac{n!}{k!(n-k)!} = \binom{n}{k}$$

You can embed fractions within fractions:

```
\frac{\frac{1}{x}+\frac{1}{y}}{y-z}
```

$$\frac{\frac{1}{x}+\frac{1}{y}}{y-z}$$

Note that when appearing inside another fraction, or in inline text $\frac{a}{b}$, a fraction is noticeably smaller than in displayed mathematics. The `\tfrac` and `\dfrac` commands force the use of the respective styles, `\textstyle` and `\displaystyle`. Similarly, the `\tbinom` and `\dbinom` commands typeset the binomial coefficient.

5.3.5 Roots

The `\sqrt` command creates a square root surrounding an expression. It accepts an optional argument specified in square brackets ([and]) to change magnitude:

```
\sqrt{\frac{a}{b}}
```

$$\sqrt{\frac{a}{b}}$$

```
\sqrt[n]{1+x+x^2+x^3+\ldots}
```

$$\sqrt[n]{1+x+x^2+x^3+\ldots}$$

5.3.6 Sums and integrals

The `\sum` and `\int` commands insert the sum and integral symbols respectively, with limits specified using the caret (^) and underscore (_). The typical notation for sums is:

```
\sum_{i=1}^{10} t_i
```

$$\sum_{i=1}^{10} t_i$$

The limits for the integrals follow the same notation. It's also important to represent the integration variables with an upright d, which in math mode is obtained through the `\mathrm{}` command, and with a small space separating it from the integrand, which is attained with the `\,` command.

```
\int_0^\infty \mathrm{e}^{-x}\,\mathrm{d}x
```

$$\int_0^\infty \mathrm{e}^{-x}\,\mathrm{d}x$$

The `\substack` command allows the use of `\\` to write the limits over multiple lines:

```
\sum_{\substack{
   0<i<m \\
   0<j<n
  }}
 P(i,j)
```

$$\sum_{\substack{0<i<m\\0<j<n}} P(i,j)$$

If you want the limits of an integral to be specified above and below the symbol (like the sum), use the `\limits` command:

```
\int\limits_a^b
```

$$\int\limits_a^b$$

5.3.7 Brackets, braces and delimiters

There are a variety of delimiters available for use in LaTeX:

```
( a ), [ b ], \{ c \}, | d |, \| e \|,
\langle f \rangle, \lfloor g \rfloor,
\lceil h \rceil, \ulcorner i \urcorner
```

$$(a), [b], \{c\}, |d|, \|e\|, \langle f\rangle, \lfloor g\rfloor, \lceil h\rceil, \ulcorner i\urcorner$$

Automatic sizing

Very often mathematical features will differ in size, in which case the delimiters surrounding the expression should vary accordingly. This can be done automatically using the `\left, \right`, and `\middle` commands. Any of the previous delimiters may be used in combination with these:

```
\left(\frac{x^2}{y^3}\right)
```

$$\left(\frac{x^2}{y^3}\right)$$

```
P\left(A=2\middle|\frac{A^2}{B}>4\right)
```

$$P\left(A=2\middle|\frac{A^2}{B}>4\right)$$

Curly braces are defined differently by using `\left\{` and `\right\}`,

```
\left\{\frac{x^2}{y^3}\right\}
```

$$\left\{\frac{x^2}{y^3}\right\}$$

If a delimiter on only one side of an expression is required, then an invisible delimiter on the other side may be denoted using a period (.).

```
\left.\frac{x^3}{3}\right|_0^1
```

$$\left.\frac{x^3}{3}\right|_0^1$$

Manual sizing

In certain cases, the sizing produced by the `\left` and `\right` commands may not be desirable, or you may simply want finer control over the delimiter sizes. In this case, the `\big, \Big, \bigg` and `\Bigg` modifier commands may be used:

```
( \big( \Big( \bigg( \Bigg(
```

These commands are primarily useful when dealing with nested delimiters.

```
\frac{\mathrm d}{\mathrm d x} \big( k g(x) \big)
```

$$\frac{\mathrm{d}}{\mathrm{d}x}\big(kg(x)\big)$$

5.3.8 Matrices and arrays

A basic matrix may be created using the `matrix` environment: in common with other table-like structures, entries are specified by row, with columns separated using an ampersand (&) and a new rows separated with a double backslash (\\)

```
\begin{matrix}
  a & b & c \\
  d & e & f \\
  g & h & i
 \end{matrix}
```

$$\begin{matrix} a & b & c \\ d & e & f \\ g & h & i \end{matrix}$$

To specify alignment of columns in the table, use starred version:

```
\begin{matrix}
  -1 & 3 \\
  2 & -4
 \end{matrix}
 =
 \begin{matrix*}[r]
  -1 & 3 \\
  2 & -4
 \end{matrix*}
```

$$\begin{matrix} -1 & 3 \\ 2 & -4 \end{matrix} = \begin{matrix*}[r] -1 & 3 \\ 2 & -4 \end{matrix*}$$

The alignment by default is `c` but it can be any column type valid in `array` environment.

However matrices are usually enclosed in delimiters of some kind, and while it is possible to use the `\left` and `\right` *commands*, there are various other predefined environments which automatically include delimiters. There are *six* versions with different delimiters:

`matrix` (none), `pmatrix` (, `bmatrix` [, `Bmatrix` {, `vmatrix` | and `Vmatrix` ||.

When writing down arbitrary sized matrices, it is common to use *horizontal*, *vertical* and *diagonal* triplets of dots (known as *ellipses*) to fill in certain columns and rows. These can be specified using the `\cdots, \vdots` and `\ddots` respectively:

```
A_{m,n} =
 \begin{pmatrix}
  a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\
  a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\
  \vdots  & \vdots  & \ddots & \vdots  \\
  a_{m,1} & a_{m,2} & \cdots & a_{m,n}
 \end{pmatrix}
```

$$A_{m,n} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}$$

In some cases you may want to have finer control of the alignment within each column, or want to insert lines between columns or rows. This can be achieved using the `array` environment, which is essentially a math-mode version of the `tabular` *environment*, which requires that the columns be pre-specified:

```
\begin{array}{c|c}
  1 & 2 \\
  \hline
  3 & 4
 \end{array}
```

$$\begin{array}{c|c} 1 & 2 \\ \hline 3 & 4 \end{array}$$

5.3.9 Adding text to equations

There are a number of ways that text can be added properly. The typical way is to wrap the text with the `\text{...}` command.

```
50 \text{ apples} \times 100 \text{ apples}
 = \text{lots of apples}^2
```

$$50 \text{ apples} \times 100 \text{ apples} = \text{lots of apples}^2$$

5.3.10 Controlling horizontal spacing

LaTeX has defined two commands that can be used anywhere in documents (not just maths) to insert some horizontal space . They are `\quad` and `\qquad`

A `\quad` is a space equal to the current font size. So, if you are using an 11pt font, then the space provided by `\quad` will also be 11pt (horizontally, of course.) The `\qquad` gives *twice* that amount.

LaTeX also provides some small spaces to be utilized:

`\,` for 3/18 quad,
`\:` for 4/18 quad,
`\;` for 5/18 quad,

`\!` for -3/18 quad

NB you can use more than one command in a sequence to achieve a greater space if necessary.

```
\int y\, \mathrm{d}x
```

$$\int y\,\mathrm{d}x$$

```
\int y\: \mathrm{d}x
```

$$\int y\:\mathrm{d}x$$

```
\int y\quad \mathrm{d}x
```

$$\int y\quad \mathrm{d}x$$

5.4 Single Equation

Mathematical equations within a text (*text formulas*) are entered between $ and $:

```
Add $a$ squared and $b$ squared
to get $c$ squared. Or, using
a more mathematical approach:
$a^2 + b^2 = c^2$
```

> Add a squared and b squared to get c squared. Or, using a more mathematical approach: $a^2 + b^2 = c^2$

```
\TeX{} is pronounced as
$\tau\epsilon\chi$\\[5pt]
100~m$^{3}$ of water\\[5pt]
This comes from my $\heartsuit$
```

TEX is pronounced as $\tau\epsilon\chi$

100 m^3 of water

This comes from my ♡

If you want your larger equations to be set apart from the rest of the paragraph, it is preferable to *display* them rather than to break the paragraph apart. To do this, you enclose them between `\begin{equation}` and `\end{equation}`.

```
Add $a$ squared and $b$ squared
to get $c$ squared. Or, using
a more mathematical approach:

\begin{equation}
a^2 + b^2 = c^2
\end{equation}
```

Add a squared and b squared to get c squared. Or, using a more mathematical approach:

$$a^2 + b^2 = c^2 \tag{5.1}$$

If you dont want LATEX to number the equations, use the starred version of equation using an asterisk, `equation*`, or even easier, enclose the equation in `\[` and `\]`:

```
Add $a$ squared and $b$ squared
to get $c$ squared. Or, using
a more mathematical approach
\begin{equation*}
a^2 + b^2 = c^2
\end{equation*}

or you can type less for the
same effect:
\[ a^2 + b^2 = c^2 \]
```

Add a squared and b squared to get c squared. Or, using a more mathematical approach

$$a^2 + b^2 = c^2$$

or you can type less for the same effect:

$$a^2 + b^2 = c^2$$

While `\[` is short and sweet, it does not allow switching between numbered and not numbered style as easily as `equation` and `equation*`.

Note the difference in typesetting style between *text* style and *display* style equations:

```
This is text style:
$\lim_{n \to \infty}
\sum_{k=1}^n \frac{1}{k^2}
= \frac{\pi^2}{6}$.

And this is display style:
\begin{equation}
\lim_{n \to \infty}
\sum_{k=1}^n \frac{1}{k^2}
= \frac{\pi^2}{6}
\end{equation}
```

This is text style: $\lim_{n\to\infty} \sum_{k=1}^{n} \frac{1}{k^2} = \frac{\pi^2}{6}$.

And this is display style:

$$\lim_{n\to\infty} \sum_{k=1}^{n} \frac{1}{k^2} = \frac{\pi^2}{6} \tag{5.2}$$

5.5 Multiple Equations

In the most general situation we have a sequence of several equalities that do not fit onto one line. Here we need to work with vertical alignment in order to keep the array of equations in a nice and readable structure.

The `align` environment is used for arranging equations of multiple lines. As with matrices and tables, `\\` specifies a line break, and & is used to indicate the point at which the lines should be aligned.

```
\begin{align}
 f(x) &= (x+a)(x+b) \\
 &= x^2 + (a+b)x + ab
\end{align}
```

$$f(x) = (x+a)(x+b) \tag{5.3}$$
$$= x^2 + (a+b)x + ab \tag{5.4}$$

Note that the `align` environment must not be nested inside an `equation` (or similar) environment. Instead, `align` is a replacement for such environments; the contents inside an `align` are automatically placed in math mode.

`align*` suppresses numbering. To force numbering on a specific line, use the `\tag{...}` command before the linebreak.

`align` is similar, but automatically numbers each line like the `equation` environment. Individual lines may be referred to by placing a `\label{...}` before the linebreak. The `\nonumber` or `\notag` command can be used to suppress the number for a given line:

```
\begin{align}
 f(x) &= x^4 + 7x^3 + 2x^2 \nonumber \\
 &\qquad {} + 10x + 12
\end{align}
```

$$f(x) = x^4 + 7x^3 + 2x^2 + 10x + 12 \tag{5.5}$$

Notice that we've added some indenting on the second line. Also, we need to insert the double braces ({}) before the + sign, otherwise latex won't create the correct spacing after the + sign. The reason for this is that without the braces, latex interprets the + sign as a unary operator, instead of the binary operator that it really is.

More complicated alignments are possible, with additional &'s on a single line specifying multiple "equation columns", each of which is aligned. The following example illustrates the alignment rule of `align*`:

```
\begin{align*}
 f(x)  &= a x^2+b x +c   &   g(x)  &= d x^3 \\
 f'(x) &= 2 a x +b       &   g'(x) &= 3 d x^2
\end{align*}
```

$$f(x) = ax^2 + bx + c \qquad g(x) = dx^3 \tag{5.6}$$
$$f'(x) = 2ax + b \qquad g'(x) = 3dx^2 \tag{5.7}$$

5.6 Equation numbering

The equation environment automatically numbers your equation:

```
\begin{equation}
 f(x)=(x+a)(x+b)
\end{equation}
```

$$f(x) = (x + a)(x + b) \tag{5.8}$$

You can also use the `\label` and `\ref` (or `\eqref` from the `amsmath` package) commands to label and reference equations, respectively. For equation number 1, `\ref` results in 1 and `\eqref` results in (1):

```
\begin{equation} \label{eq:someequation}
5^2 - 5 = 20
\end{equation}

this references the equation \ref{eq:someequation}.
```

$$5^2 - 5 = 20 \tag{5.9}$$

this references the equation 5.9.

```
\begin{equation} \label{eq:erl}
a = bq + r
\end{equation}

where \eqref{eq:erl} is true if $a$ and $b$ are integers with $b \neq c$.
```

$$a = bq + r \tag{5.10}$$

where (5.10) is true if a and b are integers with $b \neq c$.

5.7 List of Mathematical Symbols

The list of nearly all symbols available for LaTeX can be found at
`http://www.ctan.org/tex-archive/info/symbols/comprehensive/symbols-letter.pdf`

Appendix A

Further Reading

LaTeX Project
`http://www.latex-project.org/`
Official website - has links to documentation, information about installing LaTeX on your own computer, and information about where to look for help.

The Comprehensive TeX Archive Network
`http://ctan.org`
Central place for all kinds of material around TeX. Most of the packages are free and can be downloaded and used immediately.

LaTeX Wikibook
`http://en.wikibooks.org/wiki/LaTeX/`
Comprehensive and clearly written. A downloadable PDF is also available.

The Not So Short Introduction to LaTeX 2_ε
`http://ctan.tug.org/tex-archive/info/lshort/english/lshort.pdf`
A good tutorial for beginners.

The LaTeX Companion
Book for advanced users
Contains descriptions on hundreds of packages, along with information of how to write your own extensions to LaTeX 2_ε.

Bibliography

[1] Leslie Lamport. *LaTeX: A Document Preparation System.* Addison-Wesley, Reading, Massachusetts, second edition, 1994, ISBN 0-201-52983-1.

[2] Donald E. Knuth. *The TeXbook*, Volume A of *Computers and Typesetting*, Addison-Wesley, Reading, Massachusetts, second edition, 1984, ISBN 0-201-13448-9.

[3] Frank Mittelbach, Michel Goossens, Johannes Braams, David Carlisle, Chris Rowley. *The LaTeX Companion, (2nd Edition).* Addison-Wesley, Reading, Massachusetts, 2004, ISBN 0-201-36299-6.

[4] Michel Goossens, Sebastian Rahtz and Frank Mittelbach. *The LaTeX Graphics Companion.* Addison-Wesley, Reading, Massachusetts, 1997, ISBN 0-201-85469-4.

[5] Tobias Oetiker. *The Not So Short Introduction to LaTeX 2ε.* `http://ctan.tug.org/tex-archive/info/lshort/english/lshort.pdf`

[6] *The LaTeX wiki book.* `http://en.wikibooks.org/wiki/LaTeX`

[7] *BanglaTeX: Bangla with XeLaTeX.* Dr. Newton M A Hakim `http://www.dharapat.com/`

Index