IEEE & ELSOC's Guide To LaTeX

Matthew Davis Erica Barrett

August 26, 2015

Abstract

This guide aims to assist students who are using LaTeX for the first time. It also describes useful features existing users might find interesting. The is by no means a comprehensive guide to LaTeX. It aims to cover the essentials for beginners, and topics thought to be of interest to ELSOC students, and to be an accompaniment to the ELSOC presentations on learning LaTeX. It is based on the authors' own experiences and biases.

Lots of further information is available on the Internet, particularly `http://en.wikibooks.org/wiki/LaTeX`, which has been has a source of reference in this document.

More learning resources can be found at `www.elsoc.net/notes.php`. Queries, corrections and feedback may be sent to matthew@mdavis.xyz.

Contents

1 What is LaTeX

LaTeX (pronounced lay-tek) is a document creation system for high-quality typesetting. It is most often used for technical or scientific documents, but it can be used for almost any form of publishing, such as letters and resumes.

LaTeX follows the design philosophy of separating presentation from content, so that authors can focus on the content of what they are writing without attending simultaneously to its visual appearance. LaTeX is free software which works on Windows, Mac and Linux machines.

The learning curve is steep compared to programs such as Microsoft Word. However once you surmount that initial curve, you will be able to efficiently create beautiful documents, with easy cross references and bibliographies, and clear equations.

This document is quite long. The essential knowledge which is required to create a basic document is covered by sections[1] 1 to 3. Sections 4 and 5 are also somewhat necessary for most basic assignments. The remainder of the document is less crucial.

Ideally LaTeX should be learnt through experimentation. An important assignment with a strict deadline is not the best choice for a first attempt at LaTeX.

2 Installation, Editors and Compilation

To begin using LaTeX you'll need to install it onto your computer. There are ways of coding LaTeX within a browser without installing it to your computer, which will be discussed in Section 2.7 on page 6. Most likely though, you'll want to download LaTeX to your computer.

You'll need a TeX distribution, a good text editor and a PDF viewer. Optionally, you'll want a bibliography management program too, to manage your references. Bibliographies will be discussed in Section 7 on page 26.

2.1 Installing TeX Distributions

A *TeX distribution* is a set of programs which make it possible to compile LaTeX documents. TeX distributions bundle together all the parts needed for a working TeX system. There are many TeX distributions available for different operating systems.

- MiKTeX for Windows. To download MiKTeX, head to `http://miktex.org/download`. The MiKTeX Setup Wizard guides you through the installation process.
- proTeXt which is based on MiKTeX.
- TeXLive for Windows, Mac and Linux.
  - To install TeXLive and other packages on Linux, open up a terminal and paste the code from Code Sample 1 on the following page.

[1] If you click on the numbers in "section 1" you will be taken to the respective section. This is true of all cross references in this document, and the table of contents.

- MacTeX which is a redistribution of TeXLive for Mac OS X. Download MacTeX.mpkg.zip on the MacTeX page (`http://www.tug.org/mactex/`), unzip it and follow the instructions. The MacTeX page also has some help links.

Code Sample 1: Terminal command for installing LaTeX back-end packages on Linux

```
sudo apt−get install  texlive −full  texlive −fonts−recommended latex−beamer texpower
    texlive−pictures texlive−latex−extra texpower−examples imagemagick texlive−fonts−
    extra
```

2.2 Text Editors

Text editors are where you'll spend most of your time, writing the LaTeX source documents. LaTeX source documents are all text files, and can be opened and modified in almost any text editor. You could use a text editor like Notepad, but dedicated LaTeX editors are more useful than generic plain text editors. This is because they usually have auto-completion of LaTeX commands, spelling and error checking and handy pre-built macros. A few examples of editors are:

- **Mac only**: TeXShop, TeXnicle
- **Linux only**: Kile (based on Kate editor), LaTeXila, Gummi
- **Windows only**: LEd, TeXnicleCenter, WinEdt, WinShell
- **All platforms**: TeXstudio, TeXworks, Gedit (with gedit-latex-plugin), LyX, TeXmaker, Sublime Vim

The authors can recommend TeXstudio, TeXworks, Lyx, Sublime (with LaTeX add-in) and TeXShop, but feel free to experiment with other editors. Each has it's own pros and cons, and deciding between them is a matter of personal preference.

When you download MacTeX from the instructions above, the installation file includes **TeXShop**, BibTeX, TeX Live 2014, among others. A presentation of TeXShop and TeXstudio is given in subsections 2.5 and 2.6 respectively.

Similarly, when you download MiKTeX, it will include the editor **TeXworks**. This is very similar to TeXShop, so it is left to the reader to apply information given on TeXShop and TeXStudio to TeXworks.

In Linux you can download these editors using the software manager that comes with most Linux distributions. Most users would already have a favourite text editor installed.

2.3 Compiling the output

Compilation transforms a plain text document into a publishable format, mosty a DVI, PS or PDF file. This process is done by an executable file called a compiler. The "latex" executable reads a `.tex` file and creates a `.dvi` file. The "pdflatex" executable reads a `.tex` file and creates a `.pdf`. These compilers should be already within your LaTeX distribution you downloaded in subsection 2.1.

The terms *compilation*, *typesetting* and *rendering* are used interchangeably throughout this document.

Beginning the compilation process can be done on the command line or via inbuilt buttons, menu options or keyboard shortcuts in your editor.

GUI compilation

Most editors will have a "Typeset" command to compile the TeX document code into a pretty pdf and view the pdf. Often these editors have a split-screen so you can see your code and the pdf output at the same time. Some programs may have separate compile and preview buttons, rather than an all-in-one Typeset command. This procedure will be discussed for TeXShop in subsection 2.5 and TeXStudio in subsection 2.6.

Command Line compilation

You can alternatively run and compile your LaTeX text files from the command line. This process is discussed for Linux. The workflow for the command line method is the same as programming in C. Create and edit your documents in your text editor, then to compile type the following into your Linux terminal (after navigating to the directory which contains the `.tex` file.):

```
pdflatex helloWorld.tex
```

Alternatively, instead of pdflatex executable you can use xelatex. The outputs look identical. However content from xelatex can be copied and pasted into other applications more easily[2].

2.4 Lyx - GUI editor

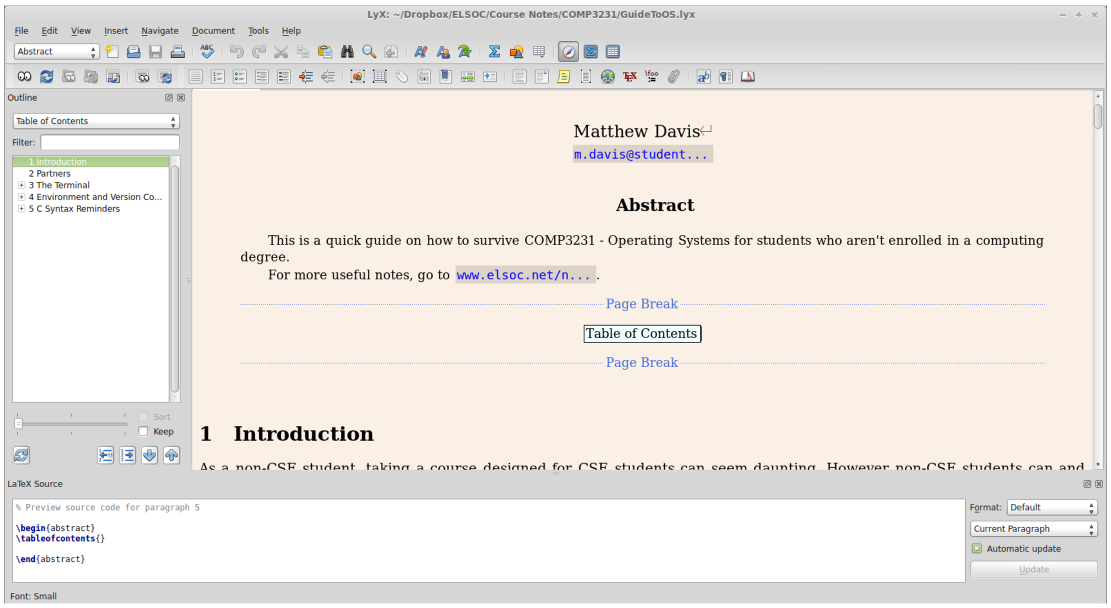
Lyx is a free program, compatible with all platforms, which attempts to provide an intuitive, easy, graphical, code-free interface for creating LaTeX documents. The learning curve for Lyx is far less than for hardcore LaTeX coding, that is done in say TeXShop. You will find it much more similar to Microsoft Word, which you are probably familiar with.

Lyx can do everything basic that can be done with LaTeX code, without the user learning any LaTeX coding syntax. Creating tables, equations and cross references is far easier in Lyx, because the document you see as you work closely resembles the final output, as seen in figure 1. (You can type LaTeX maths into Lyx equations and see them render instantaneously.)

The disadvantage of Lyx is that it is very tedious to do anything advanced, such as placing two images side by side, or not creating a thin space between the top row of a table and the rest. Furthermore, if you only ever use Lyx and never learn LaTeX syntax, you can't use applications such as ShareLaTeX (see section 2.7.1).

You can import and export LaTeX documents in Lyx. You can also insert LaTeX code into anywhere in a Lyx document if that specific feature is not implemented in Lyx. It is usefull to use Lyx just to create a table and fill it with content, and then export and paste into LaTeX documents. This is far less error prone than creating tables with hardcore code from

[2]When pasting text copied from a pdf generated with pdflatex, phantom spaces are often insterted between consecutive characters, and the characters themselves are different to what was copied. Unsure as to reason why.



Matthew Davis
m.davis@student...

Abstract

This is a quick guide on how to survive COMP3231 - Operating Systems for students who aren't enrolled in a computing degree.

For more useful notes, go to www.elsoc.net/n... .

1 Introduction

As a non-CSE student, taking a course designed for CSE students can seem daunting. However non-CSE students can and

```
% Preview source code for paragraph 5

\begin{abstract}
\tableofcontents{}

\end{abstract}
```

Figure 1: Lyx - A graphical LaTeX interface

scratch, particularly with merged cells. (See section 4.2 for a full discussion on coding tables in LaTeX).

A full explanation of Lyx is outside the scope of this guide. You can find out more about Lyx at `http://www.lyx.org/`.

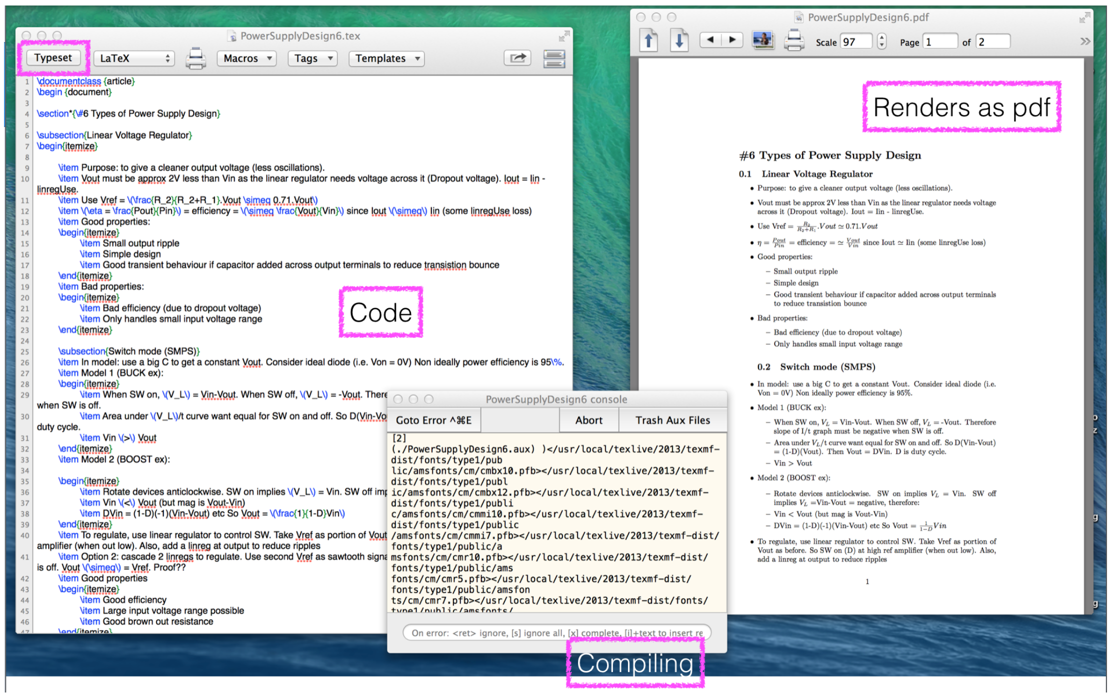
2.5 TeXShop

As seen in figure 2, TexShop has seperate windows for your code, the compiling message, and the final resultant pdf. When you first open up the TexShop application, and select File → New, a blank .tex file will come up. This .tex file is where you write your code. To transform this code into the pdf document, click the Typeset button. This button is in the top right corner of the code window (see Figure 2).

This will cause the compiling window to appear and run. The compiling message will list any errors in your code that is preventing the pdf from being updated. There will be an accompanying line number to assist you with debugging. If you can't see where you've gone wrong, google the error message.

In the code window, there are other buttons on the same horizontal line as Typeset.

- The Macros button allows the user to ask LaTeX to input pre-made code, such as code to write a list, format text and a bunch of other things. If you have code highlighted, then select say bold from the text style option, this code will be inside the `{`, `}`, which instruct LaTeX on which words to bold.
- Tags allow you to navigate the code screen to the heading or subheading line you click from Tags' drop down menu. Very handy in larger documents.



```
\documentclass {article}
\begin {document}

\section*{\#6 Types of Power Supply Design}

\subsection{Linear Voltage Regulator}
\begin{itemize}

    \item Purpose: to give a cleaner output voltage (less oscillations).
    \item Vout must be approx 2V less than Vin as the linear regulator needs voltage across it (Dropout voltage). Iout = Iin -
linregUse.
    \item Use Vref = \(\frac{R_2}{R_2+R_1}.Vout \simeq 0.71.Vout\)
    \item \(\eta = \frac{Pout}{Pin}\) = efficiency = \(\simeq \frac{Vout}{Vin}\) since Iout \(\simeq\) Iin (some linregUse loss)
    \item Good properties:
    \begin{itemize}
        \item Small output ripple
        \item Simple design
        \item Good transient behaviour if capacitor added across output terminals to reduce transistion bounce
    \end{itemize}
    \item Bad properties:
    \begin{itemize}
        \item Bad efficiency (due to dropout voltage)
        \item Only handles small input voltage range
    \end{itemize}

    \subsection{Switch mode (SMPS)}
    \item In model: use a big C to get a constant Vout. Consider ideal diode (i.e. Von = 0V) Non ideally power efficiency is 95\%.
    \item Model 1 (BUCK ex):
    \begin{itemize}
        \item When SW on, \(V_L\) = Vin-Vout. When SW off, \(V_L\) = -Vout. There
when SW is off.
        \item Area under \(V_L\)/t curve want equal for SW on and off. So D(Vin-Vo
duty cycle.
        \item Vin \(>\) Vout
    \end{itemize}
    \item Model 2 (BOOST ex):

    \begin{itemize}
        \item Rotate devices anticlockwise. SW on implies \(V_L\) = Vin. SW off imp
        \item Vin \(<\) Vout (but mag is Vout-Vin)
        \item DVin = (1-D)(-1)(Vin-Vout) etc So Vout = \(\frac{1}{1-D}Vin\)
    \end{itemize}
    \item To regulate, use linear regulator to control SW. Take Vref as portion of Vout
amplifier (when out low). Also, add a linreg at output to reduce ripples
    \item Option 2: cascade 2 linregs to regulate. Use second Vref as sawtooth signa
is off. Vout \(\simeq\) = Vref. Proof??
    \item Good properties
    \begin{itemize}
        \item Good efficiency
        \item Large input voltage range possible
        \item Good brown out resistance
```

```
[2]
(./PowerSupplyDesign6.aux) )</usr/local/texlive/2013/texmf-
dist/fonts/type1/pub
lic/amsfonts/cm/cmbx10.pfb></usr/local/texlive/2013/texmf-
dist/fonts/type1/publ
ic/amsfonts/cm/cmbx12.pfb></usr/local/texlive/2013/texmf-
dist/fonts/type1/publi
c/amsfonts/cm/cmmi10.pfb></usr/local/texlive/2013/texmf-
dist/fonts/type1/public
/amsfonts/cm/cmmi7.pfb></usr/local/texlive/2013/texmf-dist/
fonts/type1/public/a
msfonts/cm/cmr10.pfb></usr/local/texlive/2013/texmf-dist/
fonts/type1/public/ams
fonts/cm/cmr5.pfb></usr/local/texlive/2013/texmf-dist/
fonts/type1/public/amsfon
ts/cm/cmr7.pfb></usr/local/texlive/2013/texmf-dist/fonts/
```

On error: <ret> ignore, [s] ignore all, [x] complete, [i]+text to insert re

#6 Types of Power Supply Design

0.1 Linear Voltage Regulator

- Purpose: to give a cleaner output voltage (less oscillations).
- Vout must be approx 2V less than Vin as the linear regulator needs voltage across it (Dropout voltage). Iout = Iin - linregUse.
- Use Vref = $\frac{R_2}{R_2+R_1}.Vout \simeq 0.71.Vout$
- $\eta = \frac{Pout}{Pin}$ = efficiency = $\simeq \frac{Vout}{Vin}$ since Iout $\simeq$ Iin (some linregUse loss)
- Good properties:
  - Small output ripple
  - Simple design
  - Good transient behaviour if capacitor added across output terminals to reduce transistion bounce
- Bad properties:
  - Bad efficiency (due to dropout voltage)
  - Only handles small input voltage range

0.2 Switch mode (SMPS)

- In model: use a big C to get a constant Vout. Consider ideal diode (i.e. Von = 0V) Non ideally power efficiency is 95%.
- Model 1 (BUCK ex):
  - When SW on, V_L = Vin-Vout. When SW off, V_L = -Vout. Therefore slope of I/t graph must be negative when SW is off.
  - Area under V_L/t curve want equal for SW on and off. So D(Vin-Vout) = (1-D)(Vout). Then Vout = DVin. D is duty cycle.
  - Vin > Vout
- Model 2 (BOOST ex):
  - Rotate devices anticlockwise. SW on implies V_L = Vin. SW off implies V_L =Vin-Vout = negative, therefore:
  - Vin < Vout (but mag is Vout-Vin)
  - DVin = (1-D)(-1)(Vin-Vout) etc So Vout = $\frac{1}{1-D}Vin$
- To regulate, use linear regulator to control SW. Take Vref as portion of Vout as before. So SW on (D) at high ref amplifier (when out low). Also, add a linreg at output to reduce ripples

1

Figure 2: Screen capture of TexShop editor, compiling window and pdf output

- Templates has ready-to-go templates such as the Article, for new learners to use. Highly useful when starting out.

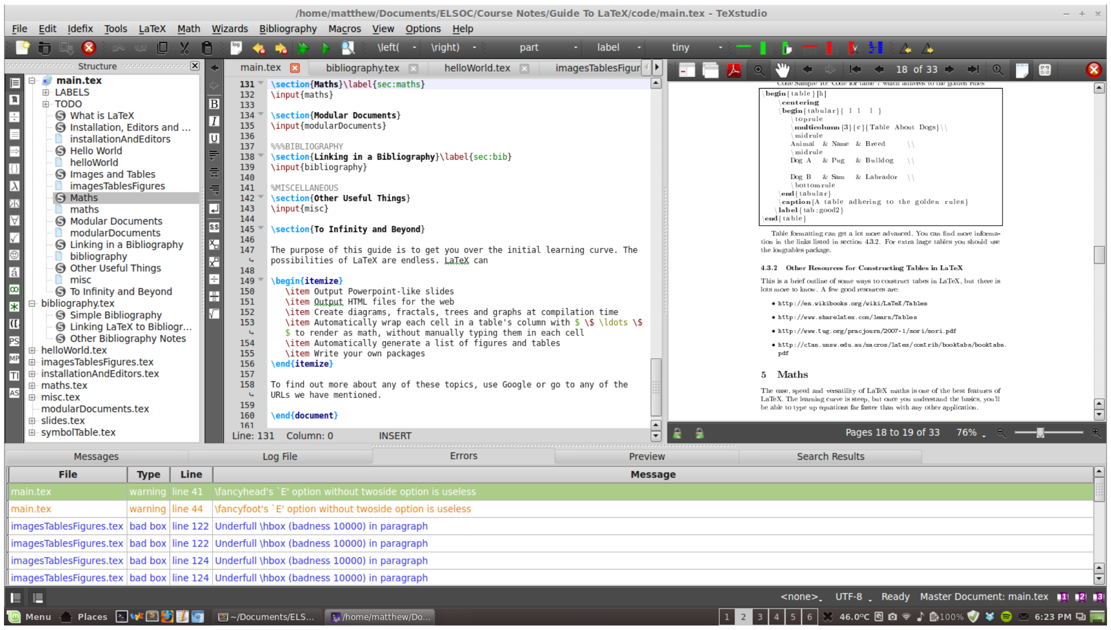
2.6 TeXStudio

As seen in figure 3, TeXstudio can display the pdf and code side by side. Clicking the left padlock in the bottom left of the preview window turns on autoscrolling, so that as you navigate through the code the corresponding section of the pdf is shown. Inversely, turning on the right padlack means scrolling through the pdf causes the code to automatically be scrolled to the corresponding section. This is very usefull.

To compile a document click the double green arrow.

The TeXStudio window also contains a navigation pane, which allows the user to switch between different files and sections easily.

The menus and toolbar allow quick access to common commands, such as for equations or document structure. The *Wizard* menu allows for the graphical creation of tables. The toolbar has buttons for easy creation and deletion of rows and columns. The *Wizard* menu also has tools for easily inserting graphics, adding bibliography entries and creating equation *arrays* (explained in section 5.1). All these tools are far less error prone than writing code from scratch.


/home/matthew/Documents/ELSOC/Course Notes/Guide To LaTeX/code/main.tex - TeXstudio
File Edit Idefix Tools LaTeX Math Wizards Bibliography Macros View Options Help
main.tex
bibliography.tex
helloWorld.tex
imagesTablesFigur
18 of 33
Pages 18 to 19 of 33
76%
Messages
Log File
Errors
Preview
Search Results

```
\section{Maths}\label{sec:maths}
\input{maths}

\section{Modular Documents}
\input{modularDocuments}

%%%BIBLIOGRAPHY
\section{Linking in a Bibliography}\label{sec:bib}
\input{bibliography}

%MISCELLANEOUS
\section{Other Useful Things}
\input{misc}

\section{To Infinity and Beyond}

The purpose of this guide is to get you over the initial learning curve. The
possibilities of LaTeX are endless. LaTeX can

\begin{itemize}
   \item Output Powerpoint-like slides
   \item Output HTML files for the web
   \item Create diagrams, fractals, trees and graphs at compilation time
   \item Automatically wrap each cell in a table's column with $ \$ \ldots \$
   $ to render as math, without manually typing them in each cell
   \item Automatically generate a list of figures and tables
   \item Write your own packages
\end{itemize}

To find out more about any of these topics, use Google or go to any of the
URLs we have mentioned.

\end{document}
```

Line: 131 Column: 0 INSERT

```
\begin{table}[h]
    \centering
    \begin{tabular}{ l l  l }
        \toprule
        \multicolumn{3}{c}{Table About Dogs}\\
        \midrule
        Animal  & Name  & Breed      \\
        \midrule
        Dog A   & Pug   & Bulldog    \\

        Dog B   & Sam   & Labrador   \\
        \bottomrule
    \end{tabular}
    \caption{A table adhering to the golden rules}
  \label{tab:good2}
\end{table}
```

Table formatting can get a lot more advanced. You can find more information in the links listed in section 4.3.2. For extra large tables you should use the longtables package.

4.3.2 Other Resources for Constructing Tables in LaTeX

This is a brief outline of some ways to construct tabes in LaTeX, but there is lots more to know. A few good resources are:

- http://en.wikibooks.org/wiki/LaTeX/Tables
- http://www.sharelatex.com/learn/Tables
- http://www.tug.org/pracjourn/2007-1/mori/mori.pdf
- http://ctan.unsw.edu.au/macros/latex/contrib/booktabs/booktabs.pdf

5 Maths

The ease, speed and versatility of LaTeX maths is one of the best features of LaTeX. The learning curve is steep, but once you understand the basics, you'll be able to type up equations far faster than with any other application.

File	Type	Line	Message
main.tex	warning	line 41	\fancyhead's `E' option without twoside option is useless
main.tex	warning	line 44	\fancyfoot's `E' option without twoside option is useless
imagesTablesFigures.tex	bad box	line 122	Underfull \hbox (badness 10000) in paragraph
imagesTablesFigures.tex	bad box	line 122	Underfull \hbox (badness 10000) in paragraph
imagesTablesFigures.tex	bad box	line 124	Underfull \hbox (badness 10000) in paragraph
imagesTablesFigures.tex	bad box	line 124	Underfull \hbox (badness 10000) in paragraph

<none> UTF-8 Ready Master Document: main.tex

Figure 3: TeXStudio

2.7 Online LaTeX editors

If you don't have LaTeX downloaded on your computer, an alternative is sites like http://www.codecogs.com/eqnedit.php. This website is great for coding a quick maths equation, but not really a whole document. It lets you enter your own LaTeX code into their box, or insert various pre-made symbols from the user buttons on the screen. You can then export it as a number of different graphic file types.

However, the real advantage of online LaTeX editors is apparent in sites such as writeLaTeX, ShareLaTeX and Authorea. They are all free, browser based tools for collaboratively and concurrently working on LaTeX documents. They're all LaTeX equivalents of Google docs. You also don't have to go through the hassle of installing extra packages when you need them.

2.7.1 ShareLaTeX

You can create a free account at https://www.sharelatex.com, although you must pay to have more than 2 collaborators.

2.7.2 writeLaTeX

writeLaTeX is quite similar to ShareLaTeX. The main advantages of writeLaTeX are:

- You can create, edit and save documents even before creating an account.
- The pdf preview is continually updated to match your code.

- There is inbuilt version control. (You can revert back to previous versions of your document.)
- There is no maximum number of collaborators for a free account.
- There is a rich text option which graphically renders some aspects of the code itself, as they would appear in the final document, including equations and headings.

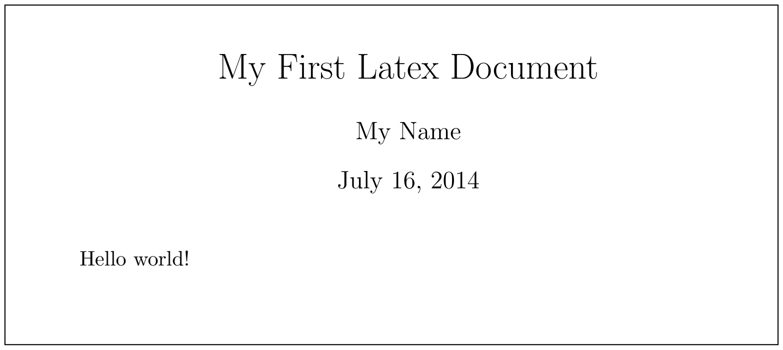
You can get started straight away at `https://www.writelatex.com/`.

3 Hello World

Open up whichever editor you have chosen (except Lyx, that's different[3]). Type in the code from Code Sample 2. Once compiled it should render like figure 4. The exact commands required for compilation are specific to each editor. Go back to section 2.2 on page 2 for more information.

Code Sample 2: The bare necessities of a LaTeXdocument

```
\documentclass{article}

%comment

\ title {My First LaTeX Document}
\author{My Name}

\begin{document}

\maketitle

Hello world!

\end{document}
```



My First Latex Document

My Name

July 16, 2014

Hello world!

Figure 4: Rendered output of Code Sample 2

[3]An explanation of Lyx is not within the scope of this guide.

This write-format-preview cycle is one of the chief ways in which working with LaTeX differs from *what-you-see-is-what-you-get* word-processing. As you become more proficient, you will compile less frequently.

In preparing a LaTeX document, the author specifies the logical structure using simple, familiar concepts such as chapter, section, table, figure, etc., and lets the LaTeX system worry about the formatting, numbering and layout of these structures.

3.1 Comments

Comments are triggered using a percent character. Comments end at the end of the line (see Code Sample 2).

Also, anything after the `\end{document}` line will be ignored.

3.2 Whitespace

The point of LaTeX is that you only have to care about the content, and the compiler will pretify it for you. Consequently, whitespace has limited effect in LaTeX. If you type 2 or more spaces between words, the compiler will only render a single space.

If you type one newline character (enter), the compiler will still stitch the two paragraphs in your code into one in the final document. This allows you to break up your code to make it easier to read. To create a new paragraph you need to insert at least one blank line. For example, Code Sample 3 renders as figure 5.

Code Sample 3: Whitespace and new lines

```
This will  *press Enter*
render as  one continuous  line.

However, if  you press  Enter twice *press  Enter*
*press  Enter*
then this  line  will  be on a new line.
```

```
This will render as one continuous line.
However, if you press Enter twice
then this line will be on a new line.
```

Figure 5: Rendered version of Code Sample 3

3.3 Protected Characters

As you have seen above, the backslash is a special character in LaTeX. It is used to signal to the compiler that you're writing a command, not content. There are other protected characters which you cannot simply type.

In general, if you want a special character to appear in your text, you'll need to put a backslash before it. For example `\%`, `\{`, `\}`, `\$`, `\_` and `\#` appear as %, {, }, $, _ and # respectively. For a backslash, type `\textbackslash`. To type a caret (^) use `\textasciicircum`. Typing these symbols within equations is a different matter. See section 5 on page 20 for more info.

To type subscripts and superscripts, you'll have to create equations, as outlined in section 5 on page 20.

3.4 Content Structure

You can create headings using the commands listed in Table 1. For example, this paragraph is part of subsection 3.4 Content Structure, which is part of section 3 Hello World. The code for this structure is given in Code Sample 4.

Code Sample 4: Code for a the structure of this section

```
\section{Hello World}

...

\subsection{Protected Characters}
Paragraph text

\subsection{Content Structure}
Paragraph text
```

The name of each heading needs to appear inside the curly braces. The advantage of LaTeX is that you don't need to worry about what numbers each heading will have. The compiler will sort that out for you. You can move around sections, insert new headings, delete old ones, and the compiler will simply renumber everything accordingly, including cross references (discussed in section 3.8). To make un-numbered headings, insert an asterix immediately before the open bracket. e.g. `\section*{...}`.

To add a table of contents, simply paste `\tableofcontents` after `\maketitle`. This table will be automatically populated by the compiler.

Significance	Numbered	Un-numbered
Largest	\part{}	\part*{}
	\section{}	\section*{}
	\subsection{}	\subsection*{}
	\subsubsection{}	\subsubsection*{}
	\paragraph{}	\paragraph*{}
Smallest	\subparagraph{}	\subparagraph*{}

Table 1: Heading commands

3.5 Document Class

In Code Sample 2, the document class is defined to be the generic "article" format. The handy thing is that if you want to change the appearance of your document, you can substitute article for another class file that exists, such as *report* or *letter*. For more document class types, see section 2.1 at `http://en.wikibooks.org/wiki/LaTeX/Document_Structure`.

3.6 Changing the Default with [options]

Most LaTeX commands look like: `\commandName{argument(s)}`. The extended version looks like `\commandName[options]{argument(s)}`. You can change the defaults using `\documentclass[<options>]{articl`
Options include changing the font size, the paper size, the number of columns etc. Again, see Section 2.1 at `http://en.wikibooks.org/wiki/LaTeX/Document_Structure`.

3.7 Packages

Everything before `\begin{document}` is called the *preamble*.

In the preamble of most tex documents, after `\documentclass{...}`, there are several `\usepackage{...}` commands. These are used to tell the compiler to include certain packages (much like `#include`-ing `.h` files when programming in C). If you get the error "Undefined control sequence" when compiling, that either means you misspelt a command which starts with a backslash, or you need to include another package.

To find documentation for a given package, go to `http://ctan.org/search.html#byDescription`.

3.8 Cross References

One of the best features of LaTeX is the ability to automatically number cross references. In most document writers, if you add another section to your document, you need to manually change all instances of "as shown in figure 6.1 on page 8" to "as shown in figure 7.1 on page 9". This is time consuming, and for large documents you're bound to miss a reference. In LaTeX you can give tables, images, sections and equations unique non-numerical names, and then reference them using those names.

To try this out, add the following between `\maketitle` and `\end{document}` in Code Sample 2.

```
\section{How To Reference}\label{referencing}
```

The contents of the `\label{...}` construct can include spaces, and can be as long as you like. This allows you to create unambiguous, memorable labels, such as `\label{graph of cheese vs voltage}`.

To reference the label we just created, add the following after \maketitle.

```
As seen in section \ref{referencing}.
```

This will initially appear in the pdf as

```
As seen in section 1.
```

Such references can be placed within any body text, including within tables and captions, using the same syntax. The exact placement of the labels will be covered later in this document when each respective object is explained.

3.9 Style

Like in any programming language, coding with good style is highly recommended. It is suggested you get in the habit of creating labels that identify what you are referencing. For example, if you are referencing "Section 1: Dinosaurs", then your reference label might be `\ref{sec:dinosaurs}`. If you're referencing a figure of a butterfly, your reference label might be `\ref{fig:butterfly}`, and so on.

It can also be helpful to recursively indent complex code, such as an equation within a table. Code Sample 5 on the following page is an example of good indentation.

4 Images and Tables

Most documents have a combination of text, maths equations, images, code and tables. Maths equations are dealt with in Section 5 on page 20 and code samples are discussed in section 8.4 on page 35. This section will discuss how you can incorporate images and tables into your document.

4.1 Figures

In general, *figures* are what images and many other non-text objects are wrapped in. Figures can have a caption and a label. Images alone cannot.

A *float* is an object[4] (e.g. a figure) which "floats" around. Instead of placing a float precisely where it was declared in the code, the compiler optimises the layout of the text and float by placing the float somewhere nearby.

4.1.1 Inserting figures into the Document

A possible way to insert a figure into LaTeX is shown in Code Sample 5. This specific example involves a figure which contains an image. The result is seen in Figure 6.

The first argument to `\includegraphics` in Code Sample 5 specifies that the figure should be 70% of the width of the text a page. It is equally acceptable to specify the width of the figure in inches, rather than as a proportion of the textwidth, such as `width=3.0in`. There are many other possible units of measurement. The essential argument of `\includegraphics` is the name of the graphics file to look for, in this case, `frogWithRocks.jpg`. The code will also work if you drop the `.jpg` extension. The caption command is straight forward, and the label option has the same functionality discussed in Section 3.8 on the previous page - Cross References. Ensure

[4]More specifically, a *property* of an object.

Code Sample 5: Code to insert a figure

```
\begin{figure}[htbp]
    \centering
    \includegraphics[width=0.7\textwidth]{frogWithRocks.jpg}
    \caption{Laid Back Frog}
    \label{fig:frog}
\end{figure}
```

Figure 6: Laid Back Frog

that the label command appears immediately after the caption, otherwise the compiler may get confused.

In Code Sample 5, the `[htbp]` tells LaTeX it can pretty much place the image anywhere. `h` stands for *here*, `t` for *top*, `b` for *bottom* and `p` for a separate *page* of only figures. The advantage of LaTeX is that the user doesn't have to worry about the precise layout. The compiler will calculate the optimal position for floats which minimises awkward page breaks. If you're unhappy with the way LaTeX has chosen to layout your document, there's a few handy commands.

- The *placeins* package provides the command `\FloatBarrier`. Say I definitely wanted the above frog picture to appear inside Section 3 - Frogs, I could type Code Sample 5, then ensure I add `\FloatBarrier` on the line before the next section. i.e. All floats written above or below `\FloatBarrier` appear above or below that point in the final document respectively. To use this option, add `\usepackage{placeins}` in the document preamble.
- The *float* package introduces a placement option H, which enforces the placement of the figure exactly at that point. To use this option, add `\usepackage{float}` in the document preamble.

4.1.2 Addressing the Graphics File

Your LaTeX document will have a .tex extension. Your graphic file may have a .png, .svg or .jpg extension[5]. The .tex file needs to know the location of the image file so it can be pulled into the document. The simplest option is to store all your images in the same directory as your main .tex file. Alternatively you will have to set `\graphicspath` to tell LaTeX were to look. This will need to be set anywhere in the preamble.

You can use absolute referencing or relative referencing to tell LaTeX where to look. In every case (even the simple case) you will need to add the graphicx package to your document preamble.

```
\usepackage{graphicx}
```

Relative referencing is much better practice. That way, your code is transferable to another computer, and will work a year later when you go to recompile your code but have since added/delted certain parent folders.

So, a cleaner and more portable option is relative address such as the code:

```
\graphicspath{{./images/}}
```

Double curly braces are required. There should be a directory named "images" in the same directory as your main tex file for this specific relative addressing code to work, since . means current folder. So you are telling LaTeX, look in this current folder for a folder named images. Lower/upper case does matter. `../` means parent folder, which could be called rather than the current folder.

Store all your `.png` files in this "images" folder[6].

[5]Several other formats are accepted. Files ending in .bmp are not accepted.
[6]Of course you can choose your own directory structure, as long as it is sensible and consistent.

4.1.3 Subfigures

A handy package is \usepackage{subcaption}. This allows subfigures to be created, so you could create a float that looks like figure 7. The code to create this float is presented in Code Sample 6.

Each subfigure can have it's own caption and label. So I could reference just the frog, as in "see fig 7a", by saying "see fig \ref{fig:frogSubfig}". You can also give the float a global caption, in this case "Side by side images".

(a) A frog

(b) A Nintenbro

Figure 7: Side by side images

In Code Sample 6 where the optional desired spacing comment is, you can add the following to spread out your subfigures:

- ∼ can be used to insert a single space in between the subfigures.
- `\hspace{‘‘length’’}` can insert a generic space you specify. In Figure 7 `\hspace{0.5in}` is used.
- `\hfill` automatically expands/contracts space to evenly space the figures across `\textwidth`.
- A blank line will force the subfigure onto a new line

Code Sample 6: Code to insert side by side figures

```
%code to put 2 images side by side in a figure
\begin{figure}[bp]
        \centering
        %Image 1
        \begin{subfigure}[htbp]{0.4\textwidth}
                \includegraphics[width=\textwidth]{frogWithRocks.jpg}
                \caption{A frog}
                \label{fig:frogSubfig}
        \end{subfigure}
        %optional desired spacing
        %Image 2
        \begin{subfigure}[htbp]{0.4\textwidth}
                \includegraphics[width=\textwidth]{linkGuitar}
                \caption{A Nintenbro}
                \label{fig:link}
        \end{subfigure}

        \caption{Side by side images}%global caption
        \label{fig:sideBySide}
```

4.2 Tables

The construction and adjustment of tables is probably the most frustrating part of LaTeX. Anything more than a basic table is a pain. An option for constructing complicated tables is to make them in Lyx (see section 2.4 on page 3), then export into LaTeX (file→export→LaTeX or view→source). This is particularly useful for tables with merged cells. After that you need to edit the code to make the table pretty (see section 4.2.4 on page 19).

Alternatively, you can use `http://www.tablesgenerator.com/`. It is a graphical online tool which can be used to easily create LaTeX tables, including merged cells. There is also a *booktabs* option, to make tables look stylish. (For more info, see subsection 4.2.4.)

4.2.1 Simple Table

Say we wanted to create the simple 3×3 table shown in Table 2. The code to achieve this is presented in Code Sample 7.

The `\begin{tabular}` statment begins a table environment. In Code Sample 7, its argument is `{| l | l | l |}`. Each `|` indicates a vertical border. The l, i.e. the letter “ell”, not the

Animal	Name	Breed
Dog A	Pug	Bulldog
Dog B	Sam	Labrador

Table 2: Simple Table

Code Sample 7: Code to create simple table

```
\begin{tabular}{| l | l | l |}
   \hline
   Animal & Name & Breed    \\
   \hline
   Dog A  & Pug  & Bulldog  \\
   \hline
   Dog B  & Sam  & Labrador \\
   \hline
\end{tabular}
```

number "one", tells LaTeX to left-align the text in that column. In a similar function to `|`, `\hline` creates a horizontal border. Each '`&`' separates what text should appear in each column. The \\ denotes a new row.

If you wanted the text to be centre-aligned, rather than left-aligned, substitute the `l` for a `c`. Similarly, `r` for right-aligned. LaTeX defaults to left-align if you don't specify. There are options for top, middle and bottom alignment.

4.2.2 Adjusting Table Attributes

One helpful piece of code, is the ability to force the width of a column. Usually LaTeX will auto-size the columns. However, for tables with many columns, LaTeX doesn't wrap text so the tables runs off the page (i.e. the rightmost part of the table is lost). Hence the occasional need to force the width of the column, which forces wrapping. This is achieved by writing `p{5cm}` instead of `l`. It may take some trial-and-error to get the result you want, i.e. try 5cm, then try 4cm etc. "`p`" will align your cell text at the top of the cell. An example is shown in Table 3 and the code in Code Sample 8.

Animal	Name	Breed	Location
Dog A	Pug	Bulldog	Randwick, New South Wales
Dog B	Sam	Labrador	Greystanes, New South Wales

Table 3: Simple Table with Fixed Column Width and Various Text Alignments

If you want to add more space between the rows, you can use `\renewcommand{\arraystretch}{<some number>}`. By default this is 1, but if we change it to 1.5, i.e. use `\renewcommand{\arraystretch}{1.5}`, then Table 2 now appears as shown in Table 4.

To includes captions and labels for your tables, you'll need to embed them in the `\begin{table}` environment. This is also how you make a table float. This adds the "Table 1: Simple Table" caption seen on page 15, and the label allows you to cross reference the table. So many accuratley, Code Sample 8: Code to create fixed width columns and text-aligned cells seen in table 3

```
\begin{tabular}{| l | r | c | p{2cm}|}
   \hline
   Animal & Name & Breed & Location \\
   \hline
   Dog A  & Pug  & Bulldog & Randwick, New South Wales \\
   \hline
   Dog B  & Sam  & Labrador & Greystanes, New South Wales \\
   \hline
\end{tabular}
```

Animal	Name	Breed
Dog A	Pug	Bulldog
Dog B	Sam	Labrador

Table 4: Simple Table stretched

Table 2 was created with the code seen in Code Sample 9. Note that the label has to appear directly after the caption.

Code Sample 9: Code to create and label simple table

```
\begin{table}
\centering
\begin{tabular}{| l | l | l |}
   \hline
   Animal & Name & Breed    \\
   \hline
   Dog A  & Pug  & Bulldog  \\
   \hline
   Dog B  & Sam  & Labrador \\
   \hline
\end{tabular}
\caption{Simple Table}
\label{tab:dogs}
\end{table}
```

4.2.3 Merging Cells

If you want to include merged cells, you can use the \multicolumn function. This function takes 3 arguments - the number of columns to span, the alignment and presence of vertical columns, and cell text. The code is shown in Code Sample 10 on the next page and rendered in Table 5.

Table About Dogs		
Animal	Name	Breed
Dog A	Pug	Bulldog
Dog B	Sam	Labrador

Table 5: Simple Table with Multi-column header row

Code Sample 10: Code to add a multi-column cell as seen in table 5

```
\begin{tabular}{| l | l | l |}
   \hline
   \multicolumn{3}{|c|}{Table About Dogs}\\
   \hline
   Animal & Name & Breed    \\
   \hline
   Dog A  & Pug  & Bulldog  \\
   \hline
   Dog B  & Sam  & Labrador \\
   \hline
\end{tabular}
```

Similarly, you can use `\multirow`, as shown in Code Sample 11, which renders as table 6. The first argument to the `\multirow` command is the number of cells to span. The 2nd is the width of the column. An asterix sets the width automatically. The `\cline` command draws a horizontal line across the columns specified, beginning in column i and ending in column j. So in Code Sample 11, this line is draw only in column 2. Using `\hline` wouldn't be suitable due to the multirow "Dog" cell.

You need to add `\usepackage{multirow}` to your preamble.

Code Sample 11: Code to add a multi-row cell, seen in table 6

```
\begin{tabular}{|c|c|}
    \hline
    Animal & Name\\
    \hline
    \multirow{2}{*}{Dog} & Steve\\
    \cline{2-2}
    & Carl\\
    \hline
\end{tabular}
```

Animal	Name
Dog	Steve
	Carl

Table 6: Table with cell spanning multiple rows

4.2.4 Stylish Tables

Creating tables in LaTeX is fiddly. Creating tables which look good is another matter. To do so, you need to stick to the golden rules.

- Never use vertical lines.
- Avoid double lines

These rules may sound odd. However if you compare tables 7 and 8 it is evident that these rules make tables look more professional.

Table About Dogs		
Animal	Name	Breed
Dog A	Pug	Bulldog
Dog B	Sam	Labrador

Table 7: A table breaking the golden rules

Table About Dogs		
Animal	Name	Breed
Dog A	Pug	Bulldog
Dog B	Sam	Labrador

Table 8: A table adhering to the golden rules

To avoid using vertical lines in the table, simply omit the `|` characters from the `\begin{tabular}{}` command, as seen in code samples 10 and 11.

The other difference between table 8 and 7 is that table 8 uses different rules (horizontal lines). First, include the booktabs package. Also swap the first, last and remaining instances of `\hline` with `\toprule`, `\bottomrule` and `\midrule` respectively. The code for table 8 is shown in Code Sample 12, which is a revised version of Code Sample 10.

Code Sample 12: Code for table 8 which adheres to the golden rules

```
\begin{table}[h]
    \centering
    \begin{tabular}{ l l  l }
        \toprule
        \multicolumn{3}{c}{Table About Dogs}\\
        \midrule
        Animal & Name & Breed    \\
        \midrule
        Dog A  & Pug  & Bulldog  \\

        Dog B  & Sam  & Labrador \\
        \bottomrule
    \end{tabular}
    \caption{A table adhering to the golden rules}
    \label{tab:good2}
\end{table}
```

Table formatting can get a lot more advanced. You can find more information in the links listed in section 4.2.5. For extra large tables you should use the longtables package.

4.2.5 Other Resources for Constructing Tables in LaTeX

The text in the above section was a brief outline of some ways to construct tabes in LaTeX, but there is lots more to know. A few good resources are:

- `http://en.wikibooks.org/wiki/LaTeX/Tables`
- `http://www.sharelatex.com/learn/Tables`
- `http://www.tug.org/pracjourn/2007-1/mori/mori.pdf`
- `http://ctan.unsw.edu.au/macros/latex/contrib/booktabs/booktabs.pdf`

5 Maths

The ease, speed and versatility of LaTeX maths is one of the best features of LaTeX. The learning curve is steep, but once you understand the basics, you'll be able to type up equations far faster than with any other application.

For very basic maths, you won't need to include any packages. For anything moderately complicated, add `\usepackage{amsmath}` to your preamble. Some special symbols require extra packages.

You can include redundant whitespace for code clarity, and LaTeX will still space things properly.

5.1 Equation Types

Equations can be written in two types of formats, inline and displayed. They can both contain the same content. Code sample 13 outlines the different options for equation syntax. Inline equations appear within the text, such as $y = mx + b$. To create an inline equation, simply type LaTeX maths between two dollar signs. A less common alternative is to use `\ ( ... \ )`.

Displayed equations appear on their own line. The syntax options for displayed equations are shown in Code Sample 13. Displayed equations can be numbered, and referenced, such as equation 1. To reference equations, use `\ref{equation label}` or `\eqref{equation label}`. This renders as 1 and (1) respectively.

$$y = mx + b \tag{1}$$

Code Sample 13: different types of equation formats

```
Text $y = m x + b$ more text \( y = m x + b \) text.
```

```
%un−numbered display equation
\[
y = m x + b
\]

%another un−numbered display equation
\begin{equation∗}
y = m x + b
\end{equation∗}

%another un−numbered display equation
$$y = m x + b$$

%numbered display equation
\begin{equation}
y = m x + b \label{equation label}
\end{equation}
```

It is possible to write multiple lines of working with aligned equals signs. This is achieved by wrapping the equations in `\begin{align}` and `\end{align}`. A double backslash is used to signal the end of each line of working. An ampersand (&) is placed before the symbol which is to be aligned. (e.g. the equals sign.) You can number and reference all, none or some of the lines. Code sample 14 renders as this:

$$\begin{align} a &= b + c - d \nonumber \\ &= g + h \tag{2} \\ &\leq i \nonumber \end{align}$$

Code Sample 14: multi-line equations

```
\begin{align}
a & =      b+c−d \nonumber               \\
  & =      g+h   \label{middle equation} \\
  & \leq   i     \nonumber
\end{align}
```

If you want to number none of the lines, you can omit the `\nonumber` commands, and replace both instances of align with align*.

You can put important equations inside a box by surrounding the expression with `\boxed{}`. For example, equation 3 is boxed, and the code for that equation is shown in Code Sample 15.

$$\boxed{y = mx + b} \tag{3}$$

Code Sample 15: Code for equation 3

```
\begin{equation}
\boxed{y = m x + b} \label{boxed equation}
\end{equation}
```

5.2 Symbols

Like the rest of LaTeX, special expressions in maths are prepended with a backslash. Table 9 lists a few examples of symbols. For example, `$ \pm \alpha $` renders as $\pm\alpha$.

To find the LaTeX expression for a symbol you don't know, you can go to `http://detexify.kirelabs.org/classify.html`, and draw the symbol you want. There is a similar app for Android called Detexify.

5.3 Brackets

You can type normal bracket characters and square brackets inside LaTeX equations, such as `()` and `[ ]`. However if the expression inside them is tall, you may want brackets which stretch to fit their content. You can do this by prepending `\left` and `\right` to the bracket. You also need to prepend a backslash before curley braces if you want them to be visable. For example `$\left( a^{b^{c}} \right)$` renders as $\left(a^{b^{c}} \right)$. You can even mix bracket types. `$\left\{ a^{b^{c}} \right]$` renders as $\left\{ a^{b^{c}} \right]$.

You can also do the same for absolute value signs with \lvert and \rvert. For example $\lvert \frac{\Sigma}{\Sigma} \rvert $ renders as $\lvert \frac{\Sigma}{\Sigma} \rvert$.

5.4 Fractions, Powers and Roots and Subscripts

LaTeX maths works recursively. You can have powers inside square roots inside fractions inside powers. To type powers, simply use a caret (^). If you want more than one symbol inside the power, wrap the power inside curly braces. For example `$x^a$` and `$ x^{a+b} $` render as x^a and x^{a+b} respectively. Writing subscripts is the same, but instead of a caret, use an underscore.

Symbol	LaTeX	Symbol	LaTeX
$\geq$	`\geq`	α	`\alpha`
$\leq$	`\leq`	β	`\beta`
$\neq$	`\neq`	γ	`\gamma`
∞	`\infty`	δ	`\delta`
$\pm$	`\pm`	Δ	`\Delta`
$\mp$	`\mp`	∇	`\nabla`
$\Vert$	`\Vert`	$\rightarrow$	`\rightarrow`

Table 9: Common Math Symbols

Expression	LaTeX
$\frac{\partial x}{\partial t}$	`\frac\partial x\partial t`
$\int_{x=1}^{\pi}$	`\int_{x=1}^{\pi}`
$\Sigma_{n=-1}^{\infty}$	`\Sigma_{n=-1}^{\infty}`
$\lim_{x\rightarrow\infty}$	`\lim_{x\rightarrow\infty}`
$\sqrt{a^{2}+b^{2}}$	`\sqrt{a^{2}+b^{2}}`
$\tan^{-1} x$	`\tan^{-1} x`

Table 10: Common Math Expressions

For example `$ x_{a+b} $` renders as x_{a+b}. To create superscripts and subscripts within normal text, you need to place the superscript or subscript content inside dollar signs, and use carets and underscores.

To type square roots, use `$\sqrt{a+b}$` to get $\sqrt{a+b}$. To type nth roots, use `$\sqrt[a]{b+c}$` to get $\sqrt[a]{b+c}$. To type fractions, use `$\frac{a}{b}$` to get $\frac{a}{b}$. Table 10 lists some more complicated expressions.

5.5 Piecewise Functions

It is possible to write piecewise functions. The code for equation 4 is shown in Code Sample 16. An ampersand (&) is used to separate each value from the respective condition. Once again, a double backslash triggers a new line. Note that `\mbox{}` has been used to write plain text within an equation.

$$f(x)=\begin{cases}1 & x \geq 0\\ 0 & \mbox{otherwise}\end{cases} \tag{4}$$

Code Sample 16: Code for piecewise function, equation 4

```
f(x)=\begin{cases}
1 & x \geq 0\\
0 & \mbox{otherwise}
\end{cases}
```

5.6 Matricies

Code sample 17 outlines how to create the matrix in equation 5. A double backslash is used to trigger a new line. Ampersands (&) are used to trigger a new column on the current line.

To wrap the matrix in curley braces {}, use `\begin{Bmatrix}...\end{Bmatrix}`, for normal parenthesis use `\begin{pmatrix}...\end{pmatrix}`, and for no brackets use `\begin{matrix}...\end{matrix}`.

$$\begin{bmatrix} a & b \\ c & d \\ e & f \end{bmatrix} \tag{5}$$

Code Sample 17: Code for the matrix in equation 5

```
\begin{bmatrix}
a & b\\
c & d\\
e & f
\end{bmatrix}
```

5.7 Putting It All Together

Equation 6 is an example that uses various symbols, fractions, powers and other features. The code for equation 6 is shown in Code Sample 18.

$$f_n(x) = \int_{t=-\infty}^{\infty} \left(\frac{\partial g^2 \left(\sqrt{x}, t\right)}{\partial x} \right) dt \tag{6}$$

Code Sample 18: LaTeX code for equation 6

```
\begin{equation}
\label{complicated_eq}
f_{n}(x) = \int_{t = - \infty}^{\infty}{
    \ left (
        \frac{\partial g^{2} \left(\sqrt{x} , t \right)}
              {\partial x}
    \right)
    dt
} \right) dt
\end{equation}
```

5.8 Plugins

There is a plugin you can install on desktop Chrome browsers called "LaTeX for Facebook". Once installed, it automatically renders any text between two dollar signs in Facebook posts, messages and comments as maths equations. This can be very useful when posting maths on Facebook. However it only renders for people using the plugin.

There is also a plugin you can get for Mozilla Thunderbird, called "LaTeX it!" which allows you to type LaTeX math into emails (between dollar signs), then with the click of a button it renders them and embeds image files into the email so that anyone can see the nice pretty equations, instead of horrible ascii.

LaTeX maths is somewhat ubiquitous. There are probably many other similar plugins for other applications.

6 Modular Documents

6.1 Include command

For large documents, it can be cleaner to split your code into several `.tex` files and combine them in a top level `.tex` file. The command to achieve this is `\include{}`. An example top level `.tex` file is shown in Code Sample 19.

Code Sample 19: Code to combine lots of .tex files into one pdf

```
%Preamble - mostly \usepackage{}..

%Start Document
\begin{document}

%Title Page/ Title

%Table of Contents, List of Figures/ Tables

%Content split into many .tex files
\include{ file1 }
\include{<name of tex file without the .tex extension>}
\include{<name of tex file without the .tex extension>}

\end{document}
```

We can see the bulk of the document's code is written in other `.tex` files, and called via `\include` statements.

An example of an included `.tex` file is `file1.tex` shown in Code Sample 20. There's nothing special about `file1.tex`, it's as if you had written the code where the `\include` statement is in the top level `.tex` file.

Code Sample 20: Example of .tex file to be included by the top level .tex file

```
%file1.tex

\section{Section Name}
%Body of Code
```

To create an updated output pdf, it is the top level `.tex` file that must be Typeset (rendered). LaTeX will throw an error if you try to Typeset any of the included `.tex` files. That is because the included `.tex` files don't have the essential commands like `\begin{document}` or `\end{document}`.

6.1.1 Includeonly command

If you only want the output pdf to show a selection of the included `.tex` files you can add the `\includeonly{}` statement into the top level `.tex` file. A code example of this is shown in Code Sample 21.

Code Sample 21: Code example for using includeonly command

```
\includeonly{ file1 } %only contents of  file1 .tex  displayed  in  output pdf
%\includeonly{file1, file2 } %Uncomment this statement to display only the contents of file1.
     tex  and  file2 .tex  in  output pdf
\include{ file1 }
\include{ file2 }
\include{ file3 }
\include{ file4 }
```

6.2 Input command

The include command forces a new page. This is usually desirable when include each section of a large document. However, if say you wanted to insert a small section of code from an external `.tex` file and not start a new page you could use the command `\input{}`. Similarly to include, the mandatory argument is the tex file to be inserted.

7 Linking in a Bibliography

This section deals with the occasions when a bibliography and/or in-text referencing is required. Incorporating references can be fiddly at first. You can either produce a bibliography by manually listing the entries of the bibliography or producing it automatically using the external BibTeX program of LaTeX.

7.1 Simple Manual Bibliography

7.1.1 Adding bib items

The simple case is when you have only a few references. LaTeX provides an environment called "thebibliography". You begin, end, and add items to "thebibliography" environment in a similar way to the "itemize" environment discussed in subsection 8.2.

To add two bibliography items to your document in this fashion, add Code Sample 22 just before `\end{document}`.

Code Sample 22: Code to add 2 bibliography items

```
\begin{thebibliography}{<any single digit number>}

\bibitem{itemID}
  author,
   title .
  publisher,
   ...

\bibitem{itemID}
  <bibliogrpahic information>
```

```
\bibitem{itemID}
  <bibliogrpahic information>

\end{thebibliography}
```

Everything between the `\begin` and `\end` tags in Code Sample 22 is treated as being data for the bibliography. The mandatory argument after the begin statement, "<any single digit>" tells LaTeX that only bibliography labels of one character in length are needed. Hence, no more than nine references can be included in total in the document. If you want more than nine, then put as the argument <any two digit number>, which allows up to 99 references.

The <itemID> needs to be unique, as this string identifies each particular bibliography item. There's lots of different methods of systematic labelling, and a common one is the surname of the first author, followed by the last two digits of the year the item was published.

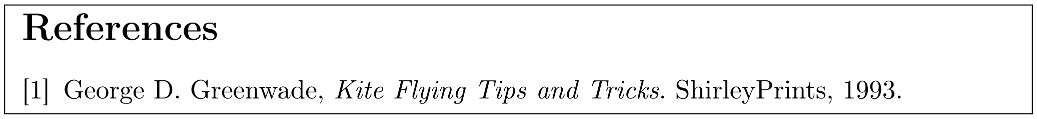
The different parts of the reference (such as author, title, etc.) are on different lines simply for readability.

7.1.2 Bibitems rendered

Code Sample 23 would render as seen in Figure 8.

Code Sample 23: Code to add 2 bibliography items

```
\begin{thebibliography}{9}

\bibitem{greenwade93}
  George D. Greenwade, %author
  \textit{Kite Flying Tips and Tricks}. %title
  ShirleyPrints, %publisher
  1993. %year

\end{thebibliography}
```



References

[1] George D. Greenwade, *Kite Flying Tips and Tricks.* ShirleyPrints, 1993.

Figure 8: Output of reference created using bibitems

Bibitems output exactly as they were written, hence why figure 8 has Author, title. Publisher, year., since that was the order it appeared in the bibitem text.

7.1.3 Manual Bibliography Formatting

When using bibitems, you must format the item's entries manually. For example, a common scheme is to italicise the title, so you could use `\textit{title}`, which is done in Code Sample

23. If you want the surname to appear first, you would need to manually adjust the bibitem text to

```
Greenwade, George D.,
\textit{Kite Flying Tips and Tricks}...
```

For this method of citation, each style of referencing (IEEE, Harvard, etc) will have their own style conventions, which you would need to apply manually. There is a way to automatically apply the right style, which is outlined in Section 7.2.

7.1.4 In-text citing

Now that you have a few bibliography items, you can cite them in the body of your code. To do this, go to the point where you want the citation to appear, and use the following `\cite{<itemID>}`.

For example, using the following line of code:

```
In \cite{greenwade93}, Greenwade suggests wind that is too strong or too light is
    difficult to fly in.
```

It would appear as:

In [1], Greenwade suggests wind that is too strong or too light is difficult to fly in.

When LaTeX compiles the document, the citation is cross-referenced with the bibitems, the matching itemID is found, and `\cite{itemID}` is replaced with a bracketed number. Replacement by bracketed number is the default. If greenwade93 was the second bibitem in the list of bibitems, then the above would have rendered as “In [2],...”.

7.2 Bibliography using BibTeX

What if you have lots and lots of references? Or if you don't want to manually format all your references? Or if you don't want to manually sort them into alphabetical order because your chosen reference style dictates that? Or if your references are stored in EndNote? Then using BibTeX is a good idea.

Back in section 2 on page 1, you would have downloaded LaTeX and most likely a few other packages in the single install. One of these was probably BibTeX.

The in-text citing procedures stays the same, however, in BibTeX thebibliography environment is not utilised. Instead your list of references is stored in an external .bib file. We will need to call this .bib file(s) in our .tex file.

7.2.1 The .bib file

You can create a .bib file in an ordinary text editor, such as the one you use to write your LaTeX documents. When you save the new file, make sure you give it the extension .bib.

A .bib file will contain all your references, think of it as containing all your "bibitems". The general format of your references is:

```
    @<entry type>{<itemID>,
    <field name> = {<text>},
     ...
    <field name> = {<text>}
    }
```

Figure 9: Syntax for layout of references in a .bib file

An example reference is show in Code Sample 24. In this code sample the <entry type> was article. Other standard entry types are shown in Table 11, which cover most forms. The itemID has the same purpose as described above in Section 7.1. The <text> can either be enclosed either by curly brackets ({<text>}) or double quotes ("<text>"). If the text is just numbers, neither {} nor "" are needed.

Code Sample 24: Code to store bibliography details of an article

```
@article{greenwade93,
    author  = "George D. Greenwade",
     title  = "The Comprehensive Tex Archive Network (CTAN)",
    year    = "1993",
    journal = "TUGBoat",
    volume  = "14",
    number  = "3",
    pages   = "342--351"
}
```

Entry Name	Description
article	Article from a journal
book	Published book
booklet	Printed work without a publisher
conference	Identical to inproceedings
inbook	Part, chapter, section etc of a book
incollection	A chapter of a book with its own author and title
inproceedings	An article in a conference proceedings
manual	Technical documentation
mastersthesis	A master's thesis
misc	Non-standard work
phdthesis	PhD thesis
proceedings	Conference proceedings
techreport	Report published by an institution
unpublished	Unpublished work with an author and title

Table 11: Table of standard entry types for references in .bib files

Not all fields are applicable to every reference entry type. For example, a reference to a journal might include the volume and number of the journal, which is usually not meaningful for a book. For any one particular entry type there will be Required fields, Optional Fields and Ignored Fields. To expand:

Required : Omitting the field will produce an error message and may result in a badly formatted bibliography entry. If the required information is not meaningful, you are using the wrong entry type.

Optional : The field's information will be used if present, but can be omitted without causing any formatting problems.

Ignored : The field is ignored.

A list of what fields are required/ optional for each entry type can be read at `http://bib-it.sourceforge.net/help/fieldsAndEntryTypes.php`. This weblink also provides information about each field.

If there is more than one author, their names are joined with an "and" in between. Commas are used to distinguish first and last name. The following field-value pair details 2 authors:

```
author = Orti, E. and Bredas, J.L. and Clarisse, C.
```

The bib file may contain more entries than are actually cited in the text. The redundant entries do not appear in the generated bibliography.

There are online bibtex generator (e.g. `http://truben.no/latex/bibtex/`) which will take field input, and output the data into the appropiate format (see figure 9).

Anything that is not inside a recognized entry is a comment i.e. anything before the symbol and after the final } for that entry.

7.2.2 Creating the Bibliography Section in Your Document

Now that we've created a bib file, to get the bibliography to appear on the page we need to add three commands into the document - the `\bibliographystyle{}` command, the `\bibliography{}` command and the `\cite{}` command (which we used in Section 7.1.

- The `\bibliographystyle{}` command requires a style name to be entered between the `{}`. There are several styles you can use. You can find out more about these styles at `https://www.sharelatex.com/learn/Bibtex_bibliography_styles`
- The `\bibliography{}` command needs the file name of the `.bib` file containing the details of the sources we are citing. If you've split your source details over multiple bib files you can enter multiple filenames into this command by separating them with a comma.
- The `\cite{}` command has the same functionality discussed in Section 7.1 - go to the point where you want the citation to appear, and use the `\cite{}` command.

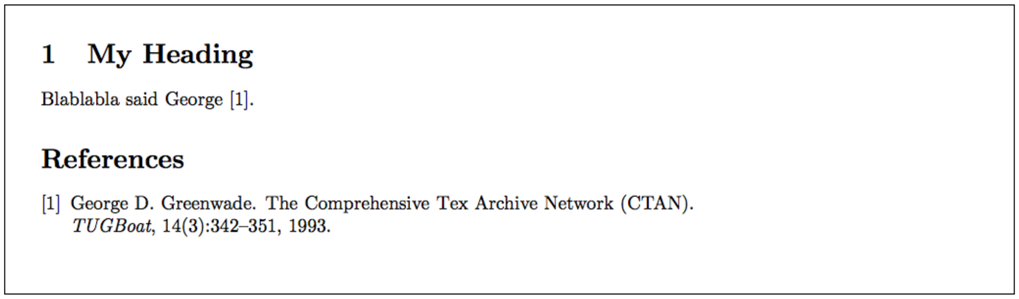
An example of a document which calls a file "mybib.bib" and cites a reference within the text is seen in Code Sample 25.

Code Sample 25: Code to create document and use bibliographic details from mybib.bib file to cite a reference

```
\documentclass{article}
```

```
\begin{document}

\section{My Heading}

Blablabla said George \cite{greenwade93}.

\bibliography{mybib}
\bibliographystyle{plain}
\end{document}
```

Code Sample 25 will render as shown in Figure 10.



1 My Heading

Blablabla said George [1].

References

[1] George D. Greenwade. The Comprehensive Tex Archive Network (CTAN). *TUGBoat*, 14(3):342–351, 1993.

Figure 10: Document with in-text citing and a references list

7.2.3 Compiling through LaTeX and then BibTeX

Getting the output shown in Figure 10 requires a few steps. You have to now compile both your .tex file and your .bib file(s).

TeXShop Example Usually your editor will be set to compiling LaTeX tex files. See Figure 11. This drop down menu is where you can navigate to BibTeX instead of LaTeX. To compile after changing your BibTeX file, you need to:

1. Run LaTeX
2. Run BibTeX
3. Run LaTeX
4. Run LaTeX

For normal updating (non-bibilography changes) you only need to do step 1.

The above example should be similar enough in other editors.

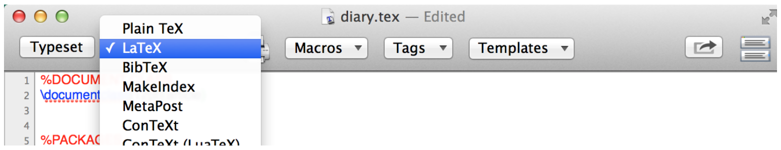
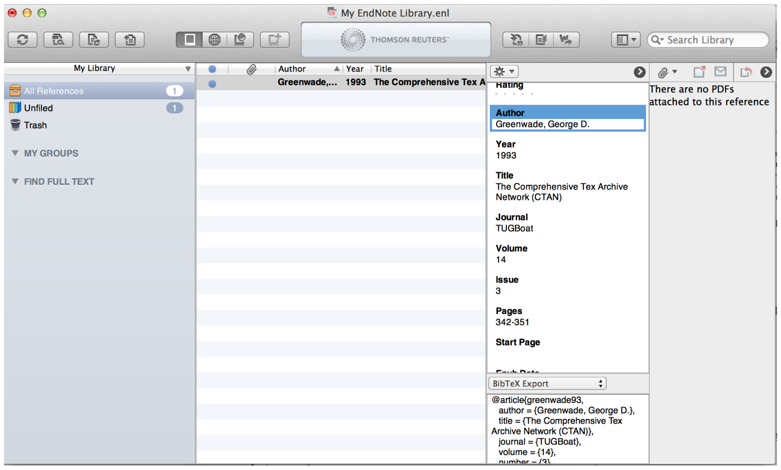


Figure 11: Screen capture of TeXShop top panel

7.3 Other Bibliography Notes

7.3.1 Comparison of Reference Management Software

There's a large number of reference management software you could use to manage your references. In a reference management software, you can compile a list of all your references and fill in details about them (like who the author is etc). You can add references as you go along, and most softwares allow you to export a reference to a format shown in 9. Figure 12 shows a screen shot of the EndNote software, which is one of many softwares available.



Author
Greenwade, George D.

Year
1993

Title
The Comprehensive Tex Archive Network (CTAN)

Journal
TUGBoat

Volume
14

Issue
3

Pages
342-351

Start Page

There are no PDFs attached to this reference

```
@article{greenwade93,
  author = {Greenwade, George D.},
  title = {The Comprehensive Tex
Archive Network (CTAN)},
  journal = {TUGBoat},
  volume = {14},
```

Figure 12: Screen Shot of EndNote, a Reference Management Software

The following wikipedia page (`http://en.wikipedia.org/wiki/Comparison_of_reference_management_software` details a long list of such software and compares their different attributes. This wikipedia page looks at factors such as:

- cost of software
- operating system(s) that support the software
- file formats the software can export. Most of them are able to export in the .bib format.

Some of the reference management software that appear good to the authors are *Mendeley*, *BibDesk*, and *EndNote*. EndNote can be downloaded from the UNSW IT page for free, by entering your zPass. This initial setup must take place while you are using the campus wifi.

7.3.2 Citation Style

It may be desired that your in-text citation is something other than square bracketed numbers. You may want the author's name to appear, not a number. The reader is then advised to check out the natbib package. By default natbib uses the author-year system, rather than numbers. If you use the natbib package, it has different bibliography styles. For example, plain (as used in Code Sample 25) must be changed to plainnat.

7.3.3 BibLaTeX

Rather than using BibTeX, users may want to investigate BibLaTeX. It provides things like citation filtering, sectionated bibliographies, chapter bibliographies, full cites, ibidem, gender differentiation, full localization, on the fly modification etc. It's more powerful that BibTeX and has been developed more recently.

8 Other Useful Things

8.1 Foot Notes

You can create footnotes[7] by using `\footnote{...}`. What you want at the bottom of the page should appear inside the curly braces. Place this expression exactly[8] where you want the superscript number to appear.

In other applications such as Microsoft Word, you can create a footnote by manually typing a superscript number, then manually typing the same superscript number in the footer, followed by the footnote contents. However in MS Word if you insert another footnote, you need to manually renumber all later footnotes. If you write new paragraphs and the superscript footnote number moves to another page, you'll need to manually move the footnote content to the new page. When writing LaTeX documents, you don't have to worry about that. LaTeX will do all the numbering and layout for you.

There are some limitations regarding the location and content of footnotes. You can include inline equations[9] in footnotes.

To reference the same footnote more than once, place a label inside the curley brackets of the footnote. Then for each additional reference to that same footnote, simply call $\backslash\text{ref}\{\text{foot:label}\}$ as a superscript. i.e. $\$\hat{}\{\backslash\text{ref}\{\text{foot:label}\}\$$.

8.2 Lists

LaTeX's equivalents of bulleting and Numbering are the functions itemize, enumerate and description. Code sample 26 shows how they are used, and figure 13 shows the rendered output.

[7] A footnote is a little bit of text at the bottom of the page.
[8] This can be mid sentence.
[9] They do appear somewhat smaller than normal equations. $\frac{a^2}{b^3}$

Code Sample 26: Code example for creating lists

```
\begin{itemize}
   \item Dot Point 1
   \begin{itemize}
      \item Sub Dot Point 1
   \end{itemize}
   \item Dot Point 2
\end{itemize}

\begin{enumerate}
    \item Item No. 1
    \item Item No. 2
\end{enumerate}

\begin{description}
    \item [term 1]  definition  of term 1
    \item [term 2]  definition  of term 2
\end{description}
```

- Dot Point 1
  - Sub Dot Point 1
- Dot Point 2

1. Item No. 1
2. Item No. 2

term 1 definition of term 1
term 2 definition of term 2

Figure 13: Rendered version of Code Sample 26

To create indented sub-lists, a second list must be created inside the first. There are commands you can add to the preamble or the list itself which will customise the symbol used as the dot.

The description command is used to highlight the first word(s) of a definition or explanation.

8.3 Text Formatting

If you would like to format text as bold, italics, or underlined, use the following commands:

- `\textbf{Wallace}` → **Wallace**
- `\textit{and}` → *and*
- `\underline{Gromit}` → Gromit

To force a space between words, or a new line, use a tilde (˜) or a double backslash respectively. This should only be neccessary in complicated code. There are other commands for forcing page breaks, line breaks etc.

8.4 Code Samples

To add code samples to your document, add `\usepackage{listings}` to your preamble. Then add the code from Code Sample 27 for each code sample.

If you want your code samples to be called "code samples" instead of "listings", also add

```
\renewcommand{\lstlistingname}{Code Sample}
```

to your preamble (after loading the package).

Code Sample 27: Code for inserting code samples

```
\begin{ lstlisting }[caption={caption},
                    label={memorable label},
                    language=C,%or java, Matlab, TeX, HTML, Verilog...
                    frame=single,%outline the code
                    breaklines=true, %break apart long lines onto multiple  lines
                    numbers=left,]%line numbers
    printf("hello  world!\n");
\end{ lstlisting }
```

If your document has many code samples, you can set default options for language, border etc in the preamble. Use something like Code Sample 28. You can overwrite the defaults by restating a setting in each specific code sample.

Code Sample 28: Code for setting defaults for code samples

```
\ lstset {
        language=TeX,
        frame=single,               %outline the code
        breaklines=true,            %break apart long lines onto multiple  lines
        numbers=left,               %line numbers
        float=ht,                   %make it a float, try to put it here
        showstringspaces=false,     %hide visable 'space' markers from strings
}
```

You can also include code from separate files. The method for doing so is shown in Code Sample 29. If the firstline or lastline options are omitted LaTeX will include from the start or to the end respectively.

Code Sample 29: Code for inserting code samples from external files

```
\lstinputlisting [caption={caption},
                  label={code:label},
                  firstline  = 123,
                  lastline  = 321,
                  float =ht]
                  {fileName.extension}
```

8.5 Adding pdfs

LaTeX has the functionality to include some or all pages of external `.pdf` files into your document.

To embed a (single page) pdf as an image, simply use `\incudegraphics{}`, as you would do for a `.jpg` or `.png` file.

To include standalone pages from external `.pdf` files, use `\includepdf{file1.pdf}`. It is important `file1.pdf` is saved in the same folder as your `.tex` file this statement appears in. If not, use `\includepdf{<relativeFilePath>/file1.pdf}`. Most LaTeX commands have options, which can be set using the syntax `\<functionName>[Options]{Arguments}`. To include only say the first page of `file1.pdf`, the code would be `\includepdf[pages={1}]{file1.pdf}`. You will also need to add `\usepackage{pdfpages}` to your preamble.

8.6 Double Quotation Marks

LaTeX renders all double quotes as closing quotes. To write an opening double quote, type two consecutive single open quotes (the top left of your keyboard). Alternatively, add Code Sample 30 to your preamble. Do not mix the two methods.

Code Sample 30: Preamble addition for automatically converting double quotes to open double quotes when needed

```
\usepackage [english]{babel}
\usepackage [autostyle, english = american]{csquotes}
\MakeOuterQuote{"}
```

8.7 Better Cross References

You can reference an object and its page number by using `\vref{...}`. To use this command you need to add `\usepackage{varioref}` to the preamble. For example:

```
As seen in figure \vref{label}.
```

This will appear in the pdf as:

`As seen in figure x on page y.`

8.8 Clickable Links

Add `\usepackage{url}` to your preamble. Now you can add nicely formatted URLs by using `\url{www.example.com}`, which renders as `www.example.com`.

To make cross references and URLs clickable, add the `hyperref` package to your preample. It must be the last added package[10], otherwise you may encounter problems.

```
\usepackage[hidelinks]{hyperref}
```

By default this package places a hideous coloured box around the links. The `hidelinks` option removes this box and makes then appear unboxed, black and un-underlined. There are options which allow you to specify colour, underline and other aspects of formatting. This package also adds some usefull meta-data to your pdf, such as adding the sections, subsections etc. to the navigation menu.

8.9 Todo notes

You can add brightly coloured to-do notes to your document which appear in the rendered document. These can be used whilst you're still working on your document. They're supposed to all be addressed and removed by the time you complete the document. The bright colours mean that they are harder to forget about than normal comments within your code, especially if you have many files, and have multiple collaborators.

Example todo note

Example inline todo note

Code sample 31 shows how to create a todo notes.

Code Sample 31: Code for todo notes

```
Blah \todo{Example todo note in the margin} blah

\todo[inline]{Example todo note on it's own line}
```

You also need to the following to your preamble.

```
\usepackage[ colorinlistoftodos ]{todonotes}
```

At any location in the document, a list of the inserted notes can be generated with the following command.

```
\ listoftodos
```

Todo list

[10] There are a few rare exceptions.

9 To Infinity and Beyond

The purpose of this guide is to get you over the initial learning curve. The possibilities of LaTeX are endless[11]. LaTeX can

- Output Powerpoint-like slides. (The *Beamer* document class.)
- Output HTML files for the web.
- Create diagrams, fractals, trees and graphs at compilation time. (the *pstricks*, *pgfplots* and *tikz* packages.)
- Automatically generate a list of figures and tables
- Write your own packages
- Declare and manipulate variables

To find out more about any of these topics, simply search the web or go to any of the URLs we have mentioned. There are countless learning resources out there for you to explore.

[11] It's actually Turing complete.