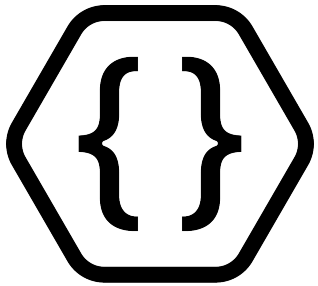
The TEX Talk

Programming Perfect Documents

Matthew Low
UQCS



April 7, 2020

Meme

Motivation

Writing Microsoft Word documents can sometimes be a painful experience. This is mostly caused by the clunky interface and Word's focus on layout rather than content; you need to do a lot of layout work on your own to get an appealing document.

Motivation

LaTeX aims to shift the focus to **content**. LaTeX will handle all of the heavy-lifting (what page should this paragraph should be on? where should this image go?) and let you focus on the content.

Outline

1. How LaTeX works
2. Basic document
3. Environments
4. Text formatting
5. Packages
6. Code listings
7. Mathematics
8. Document formatting and referencing
9. Advanced topics
10. A complete setup
11. Where to next?
12. Questions

What is LaTeX, and what is TeX?

LaTeX is a document **typesetting system**.

TeX is its **programming language**[1].

LaTeX is pronounced LAY-tek or LAH-tek, *not* LAY-teks.

[1] Actually, it's quite complicated and confusing (TeXis also a typesetting system). But this is a good place to start.

How does LaTeX work?

In the most simple terms possible...

How does LaTeX work?

- **Editing TeX code?** Any text editor will do, though VSCode, Vim, emacs with plugins are popular options. There are also IDEs such as TeXstudio/TeXmaker etc.

How does LaTeX work?

- **Editing TeX code?** Any text editor will do, though VSCode, Vim, emacs with plugins are popular options. There are also IDEs such as TeXstudio/TeXmaker etc.
- **Running the LaTeX compiler?** pdfLaTeX, XeLaTeX, LuaLaTeX (we will only cover pdfLaTeX today).

How does LaTeX work?

- **Editing TeX code?** Any text editor will do, though VSCode, Vim, emacs with plugins are popular options. There are also IDEs such as TeXstudio/TeXmaker etc.
- **Running the LaTeX compiler?** pdfLaTeX, XeLaTeX, LuaLaTeX (we will only cover pdfLaTeX today).
- How do I get the compilers, packages, and other stuff? Download a **LaTeX distribution**. Popular options are MiKTeX for Windows, TeX Live for Linux/UNIX, MacTeX for macOS. Alternatively, you can use a website like Overleaf and everything is handled in the browser! **This is recommended for beginners. Free premium accounts for UQ students!**

Before we start diving in...

A TeX file is a program.

This can be good or bad.

You need to write **correct** code, or else LaTeX can't magically compile it for you.

Before we start diving in...

You *will* get syntax errors. Writing documents and getting syntax errors may be a weird experience for you, but learning how to identify and fix those errors over time will help you understand and appreciate greatly the LaTeX system.

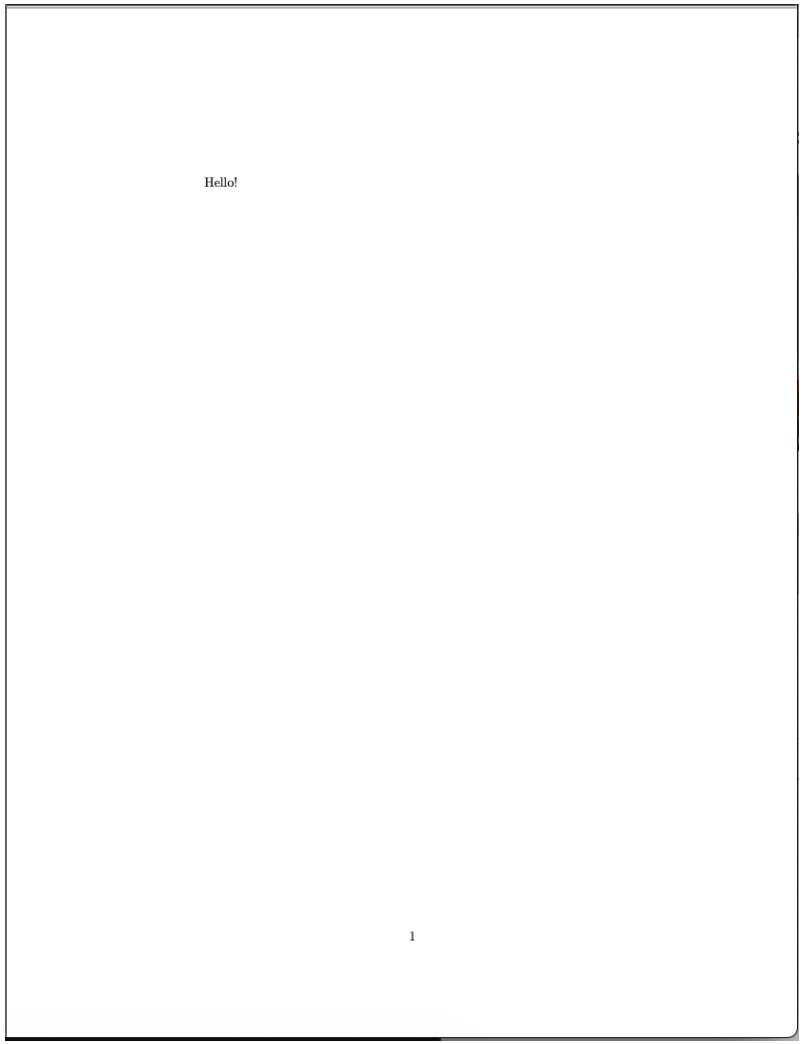
Something simple...

This is about as simple as it gets for TeX.

```
\documentclass{article}
\begin{document}
Hello!
\end{document}
```


Something simple...

Typing this and compiling will give you a page that looks like this...



What are these bits of code?

- `\documentclass{article}` says that the **document class** should be of type 'article'. There are other classes, such as 'book', 'report' etc. This will become important later when we talk about sections.

What are these bits of code?

- `\documentclass{article}` says that the **document class** should be of type 'article'. There are other classes, such as 'book', 'report' etc. This will become important later when we talk about sections.
- `\begin{document}` starts a new "document". Every thing past here goes on the page, or affects the page.

What are these bits of code?

- `\documentclass{article}` says that the **document class** should be of type 'article'. There are other classes, such as 'book', 'report' etc. This will become important later when we talk about sections.
- `\begin{document}` starts a new "document". Every thing past here goes on the page, or affects the page.
- `\end{document}` should be the last line of every TeX document you write.

Environments

`\begin` and `\end` are delimiters for an **environment**. In LaTeX, almost everything is an environment. You will see this soon.

Document structure

Note the code below.

```
\documentclass{article}
\begin{document}
Hello!
\end{document}
```

Everything above the `\begin{document}` is part of what is called the **preamble**. This is where you setup the document.

Starting off a document

Maybe you want a bit more than "Hello!" in a document. Maybe you want to start with a title? Author? Date? Simple as. Put this in the **preamble**:

```
\title{My Document!}
\author{Me}
\date{\today}
```

`\today` is a command which returns the current date.

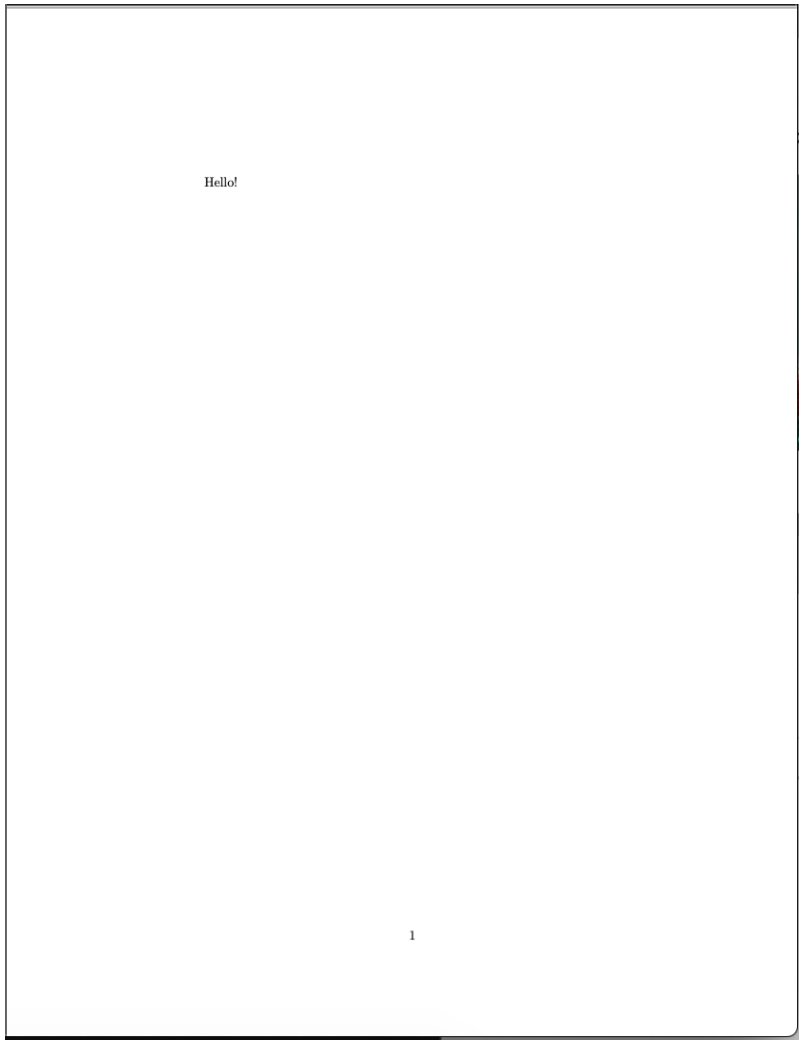
Starting off a document

Our full code now looks like this:

```
\documentclass{article}
\title{My Document!}
\author{Me}
\date{\today}
\begin{document}
Hello!
\end{document}
```

Cool! Let's run it...

Starting off a document



...wait, where's my title?

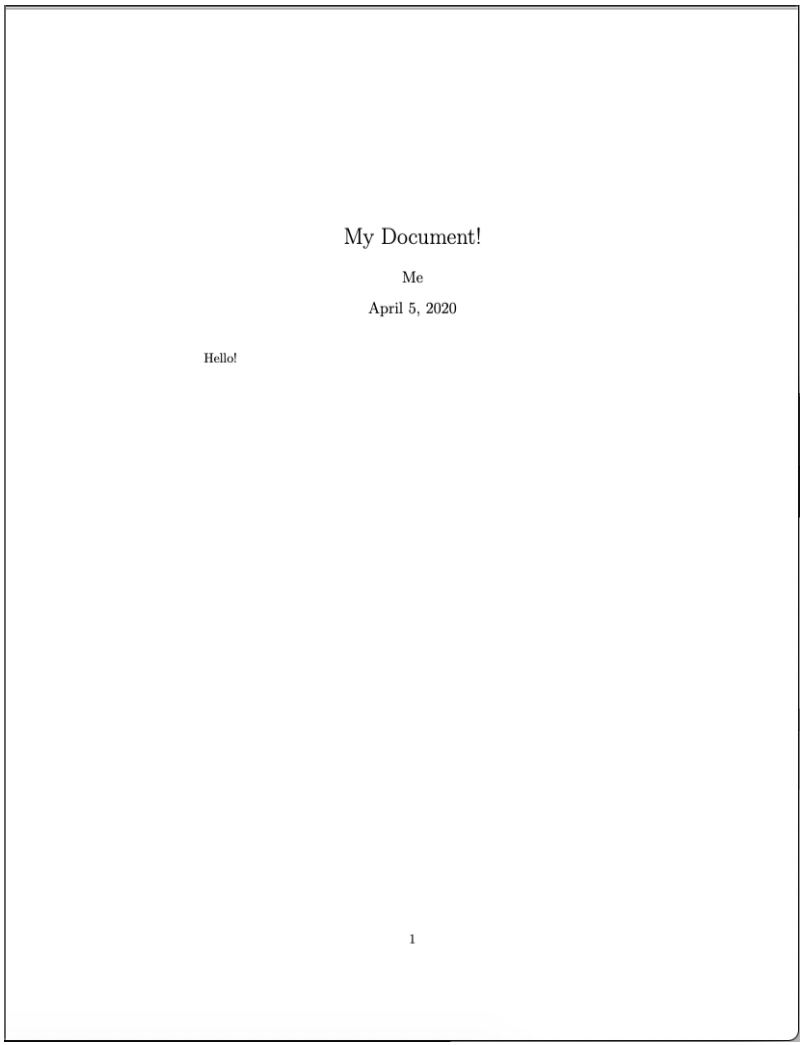
Starting off a document

No worries, just have to actually print the title. Remember, the title, author and date are specified in the preamble, so it won't show up in the document until we say to print it. In this case, we need to add

```
\maketitle
```

after `\begin{document}` to get it to actually show up.

Starting off a document



Better!

Structure

The next thing your document needs is some structure. To do this, we will make use of a command called

`\section`.

There is also

`\subsection`, `\subsubsection`, ..., `\chapter`, `\part` etc.

The availability of these depends on your document class. For example, articles don't have chapters; books do.

Table of contents

To do a table of contents, type the code

```
\tableofcontents
```

in your documents. It will automatically be generated from the sections that you specified with `\section`, `\subsection` etc.

We're done!

Now, we technically have everything we need to start writing our 500-page novel. But, of course, there may be a few fancier things than paragraphs that we may want to add.

Images? Tables? Lists? Code? Math?

Images and figures

To put an image on the page, you need to have the graphicx package imported. A package is essentially an "add-on" that adds more features, environments and options in your document. We'll see more of these as we go on.

To import graphicx, type in the preamble:

```
\usepackage{graphicx}
```

We can create an image by simply putting

```
\includegraphics{path_to_image}
```

in the document. But this might make a massive image, or a tiny one! And it's not centered!

Images and figures

To fix this, we can create a center environment:

```
\begin{center}
    \includegraphics{path_to_image}
\end{center}
```


Images and figures

To fix this, we can create a center environment:

```
\begin{center}
    \includegraphics{path_to_image}
\end{center}
```

and also specify a width with an **option** to `\includegraphics`:

```
\begin{center}
    \includegraphics[width=0.6\textwidth]
                    {path_to_image}
\end{center}
```

`\textwidth` is a "length" for the width of the current area of text. There are many other lengths specified in TeX.

Figures

Another method of putting an image on the screen is using a figure environment. This can be done by wrapping the `\includegraphics` in it like so:

```
\begin{figure}
    \centering
    \includegraphics[width=0.6\textwidth]
                    {path_to_image}
    \caption{This is a wonderful image!}
\end{figure}
```


Figures

Another method of putting an image on the screen is using a figure environment. This can be done by wrapping the `\includegraphics` in it like so:

```
\begin{figure}
    \centering
    \includegraphics[width=0.6\textwidth]
                    {path_to_image}
    \caption{This is a wonderful image!}
\end{figure}
```

The `\centering` is equivalent to wrapping in a `\begin{center}` and `\end{center}`, but it is a toggle. Since it is called inside an environment, it stops centering after the environment is finished. If you typed `\centering` in the main body of the document, it would center everything (including text) until the end.

Figures

Another method of putting an image on the screen is using a figure environment. This can be done by wrapping the `\includegraphics` in it like so:

```
\begin{figure}
    \centering
    \includegraphics[width=0.6\textwidth]
                    {path_to_image}
    \caption{This is a wonderful image!}
\end{figure}
```

The `\centering` is equivalent to wrapping in a `\begin{center}` and `\end{center}`, but it is a toggle. Since it is called inside an environment, it stops centering after the environment is finished. If you typed `\centering` in the main body of the document, it would center everything (including text) until the end.

`\caption{}` does exactly what it says.

Float

You may notice that figures don't really go where you want them to go. This is all part of LaTeX's smarts.

To fix this, simply import the package `\usepackage{float}`

and put a `[H]` after `\begin{figure}`, like so:

```
\begin{figure}[H]
    \centering
    \includegraphics[width=0.6\textwidth]
                    {path_to_image}
\end{figure}
```

This will force the image to go exactly where you specified.

Tables

Tables are quite simple. To make a 3 column table where all columns are centre-aligned:

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

`&` is the "alignment" character in TeX, and `\\` are the "linebreak" characters. You'll see these a lot in maths.

`l` for left aligned, `c` for centered, `r` for right-aligned. If you want to center the whole table, use a figure or `\begin{center}`.

Ordered and unordered lists

Ordered lists are called **enumerates**. Unordered lists are called **itemizes**. To do them, simply:

```
\begin{enumerate}
    \item Hello
    \item Hi
\end{enumerate}
```

or

```
\begin{itemize}
    \item Hello
    \item Hi
\end{itemize}
```


Font sizes

Font sizes are based off of the **base font size**. This is set in the document class by default to 10pt. If you want 11pt, or 12pt, provide the command 11pt or 12pt as an argument to the document class.

Bold, italic and underline

To apply formatting to text, it is often useful to wrap it in some command such as `\textbf`. For example:

- To **bold** text, `\textbf{the text}` or `\bfseries` to switch.
- To *italicise* text, `\textit{the text}` or `\itshape` to switch.
- To underline text, `\underline{the text}` (it doesn't have a switch, but you can define one.)

Left align, right align and centre

You can align text both as a switch and as an environment.

- To **left-align**, use the `\begin{flushleft}`, or use the `\raggedright` switch.
- To **right-align**, use the `\begin{flushright}`, or use the `\raggedleft` switch.
- To **centre-align**, use the `\begin{center}`, or use the `\centering` switch.
- To **justify** (default), use the `\justify` switch.

Packages

Let's look at more packages.

Essential packages

We've already covered `graphicx` and `float`. Now, I'll cover

- `listings` for writing code and syntax highlighting;
- `amsmath, amsthm, amssymb` for writing mathematics (theorems and proofs);
- `geometry` for adjusting page dimensions;
- `biblatex` for bibliographies

Code listings

To write code, use the `listings` package, and write in your document:

```
\begin{lstlisting}
def hello_world():
    print("Hello world!")
\end{lstlisting}
```

This will print the code in its own nice little environment.

Code listings

To write code, use the `listings` package, and write in your document:

```
\begin{lstlisting}
def hello_world():
    print("Hello world!")
\end{lstlisting}
```

This will print the code in its own nice little environment. You might want to syntax highlight too: you can do this by providing the option `[language=python]` in this case, with an output like:

```
def hello_world():
    print("Hello␣world!")
```

You can see how it *slightly* bolds `def` and `print`, and puts a space character in. Syntax highlighting varies by language, and you can do a whole bunch more than this. Just google.

External code

Do you want code from another file?Just use `\lstinputlisting`.

```
\lstinputlisting{name_of_code.py}
```

This will place the code from the file `name_of_code.py` in that position.

Writing math

So, you want to write some mathematics. LaTeX is **the** tool for typesetting mathematics. Its syntax is super simple, and very easy to get the hang of. I'll go through some basics here.

Writing math

So, you want to write some mathematics. LaTeX is **the** tool for typesetting mathematics. Its syntax is super simple, and very easy to get the hang of. I'll go through some basics here.

For this section onwards, we assume we have imported the `ams` packages `amsmath` (basics), `amssymb` (more symbols) and `amsthm` (theorem environments).

Writing math

So, you want to write some mathematics. LaTeX is **the** tool for typesetting mathematics. Its syntax is super simple, and very easy to get the hang of. I'll go through some basics here.

For this section onwards, we assume we have imported the `ams` packages `amsmath` (basics), `amssymb` (more symbols) and `amsthm` (theorem environments).

Firstly, there are two types of math environments: **inline** and **display**. Inline is done using `$$` and `\(\)`, whereas display math is typically done using `$$$$` and `\[\]`. Most people use `$$` and `\[\]` for technical reasons (spacing).

Inline math goes inline, like this $\sin^2\theta + \cos^2\theta = 1$. Display math goes "out-of-line", like this

$$\sin^2\theta + \cos^2\theta = 1.$$

Writing math

Plus and minus are just typed on the keyboard. Multiplication ($\times$) is written using `\times`. Fractions ($\frac{a}{b}$) are written using `\frac{a}{b}`.

Writing math

Plus and minus are just typed on the keyboard. Multiplication ($\times$) is written using `\times`. Fractions ($\frac{a}{b}$) are written using `\frac{a}{b}`.

Common operators and symbols are written using `\nameofsymbol`. If you are applying some operator or function to some variables, you might need curly brackets. For example, `\sqrt{a}` gives $\sqrt{a}$.

Writing math

Plus and minus are just typed on the keyboard. Multiplication ($\times$) is written using `\times`. Fractions ($\frac{a}{b}$) are written using `\frac{a}{b}`.

Common operators and symbols are written using `\nameofsymbol`. If you are applying some operator or function to some variables, you might need curly brackets. For example, `\sqrt{a}` gives $\sqrt{a}$.

Trigonometric functions are written as expected (`\sin`, `\cos`, `\tan` give $\sin, \cos, \tan$). Exponentials and logarithms are also self-explanatory.

Writing math

Plus and minus are just typed on the keyboard. Multiplication ($\times$) is written using `\times`. Fractions ($\frac{a}{b}$) are written using `\frac{a}{b}`.

Common operators and symbols are written using `\nameofsymbol`. If you are applying some operator or function to some variables, you might need curly brackets. For example, `\sqrt{a}` gives $\sqrt{a}$.

Trigonometric functions are written as expected (`\sin`, `\cos`, `\tan` give $\sin, \cos, \tan$). Exponentials and logarithms are also self-explanatory.

To write lowercase, write `\delta` (δ) for example (note the lowercase 'a'). For uppercase, `\Delta` (Δ).

Writing more advanced math

Powers and subscripts are written using `_{}` and `^{}`. For example, `x^2` gives x^2 and `x_1` gives x_1.

Writing more advanced math

Powers and subscripts are written using `_{}` and `^{}`. For example, `x^2` gives x^2 and `x_1` gives x_1.

For operators with limits (limits, integrals), you denote the lower limit by `_{}` and upper limit by `^{}`, for example `\int_{-\infty}^{\infty}` $= \int_{-\infty}^{\infty}$.

Writing more advanced math

Powers and subscripts are written using `_{}` and `^{}`. For example, `x^2` gives x^2 and `x_1` gives x_1.

For operators with limits (limits, integrals), you denote the lower limit by `_{}` and upper limit by `^{}`, for example `\int_{-\infty}^{\infty}` $= \int_{-\infty}^{\infty}$.

Parentheses with fractions do not "stretch" by default. If you want them to, wrap the parentheses (or whatever other brackets) in `\left` and `\right`. For example, `\[\left(\frac{1}{2}\right)\]` will give

$$\left(\frac{1}{2}\right)$$

whereas `\[(\frac{1}{2})\]` will give

$$(\frac{1}{2}).$$

Writing more advanced math

Matrices are done using the **matrix** environment, **inside a math environment**. You can wrap matrices in a `\left(` and `\right)` to obtain "pmatrices", or you can use the package `amsmath` to use the **pmatrix** environment directly. Matrices are done the same way as tables, but without column specification:

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

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

Mathematical alphabets

You may find yourself needing to use fancier mathematical alphabets in certain situations. Note that these alphabets are different to `\textbf` etc. that we defined above as these can only be used in math mode. There are plenty more than just these.

Mathematical alphabets

You may find yourself needing to use fancier mathematical alphabets in certain situations. Note that these alphabets are different to `\textbf` etc. that we defined above as these can only be used in math mode. There are plenty more than just these.

- `\mathrm` is roman (upright) ABC
- `\mathit` is italic (the default, keep this in mind when writing operators as upright is convention) ABC
- `\mathsf` is sans-serif (this document is sans so it looks "normal") ABC
- `\mathbf` is bold $\mathbf{ABC}$
- `\mathbb` is blackboard bold $\mathbb{ABC}$
- `\mathcal` is calligraphic $\mathcal{ABC}$
- `\mathfrak` is Fraktur $\mathfrak{ABC}$

Now, you can write this!

$$\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \mathrm{d}x = 1$$

can be written using the LaTeX code

```
\[
  \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}}
  e^{\frac{-(x-\mu)^2}{2\sigma^2}} \mathrm{d}x = 1
\]
```

Want to practice writing math? Go to TeXnique! It's a fun web game which gives you a bunch of equations to type within a time limit. https://texnique.xyz/

Aligned math

If you have multiple lines of math you want aligned, you can use an **align*** or **align** environment. The * indicates no numbered labels.

For example, typing

```
\begin{align*}
        \sin^2 \theta + \cos^2\theta &= 1 \\
        \tan^2 \theta + 1 &= \sec^2 \theta \\
        1 + \cot^2 \theta &= \csc^2 \theta
\end{align*}
```

will give you

$$\begin{aligned} \sin^2 \theta + \cos^2 \theta &= 1 \\ \tan^2 \theta + 1 &= \sec^2 \theta \\ 1 + \cot^2 \theta &= \csc^2 \theta \end{aligned}$$

Note the alignment character &.

Theorem environments

To create a theorem environment, we first have to define one. Put this in the preamble:

```
\newtheorem{theorem}{Theorem}
```

You can then create a theorem anywhere in the text by typing

```
\begin{theorem}
        UQCS is a wonderful organisation.
\end{theorem}
```

Theorem
UQCS is a wonderful organisation.

Proof environment

Proofs are already defined for us, just use the proof environment. It even puts a cute square box at the end!

```
\begin{theorem}
  UQCS is a wonderful organisation.
\end{theorem}
\begin{proof}
  Left as an exercise for the reader.
\end{proof}
```

Theorem
UQCS is a wonderful organisation.

Proof.
Left as an exercise for the reader. □

Mathematics in LaTeX: a summary

There is a **lot** of math stuff in LaTeX. Just keep practicing and you'll be fluent in no time!

Footnotes

To put a footnote in your document, simply type `\footnote{This is a footnote!}`[2]

[2] This is a footnote!

Bibliographies

To create a bibliography, use the package `biblatex`, create a `.bib` file (BibLaTeX file) and link it in the preamble using the syntax

```
\addbibresource{name_of_bib_file.bib}
```

This allows you to reference any entries in the `.bib` file using `\cite{}` in the body of the `.tex` file, where the label of the reference in the `.bib` file is inserted in the curly brackets. To print the bibliography somewhere in the document, simply type

```
\printbibliography
```


Sample .bib file

Here's an example of a `.bib` file, courtesy of Overleaf:

```
@article{einstein,
    author =        "Albert Einstein",
    title =         "{Zur Elektrodynamik bewegter K{\"o}rper}. ({German})
    [{On} the electrodynamics of moving bodies]",
    journal =       "Annalen der Physik",
    volume =        "322",
    number =        "10",
    pages =         "891--921",
    year =          "1905",
    DOI =           "http://dx.doi.org/10.1002/andp.19053221004",
    keywords =      "physics"
}

@online{knuthwebsite,
    author    = "Donald Knuth",
    title     = "Knuth: Computers and Typesetting",
    url       = "http://www-cs-faculty.stanford.edu/~uno/abcde.html",
    keywords  = "latex,knuth"
}
```

Many bibliography managers can export to BibLaTeX.

Include and input

This is really important and really cool. You can put other `.tex` files in your current `.tex` file by using the commands

```
\include{the_sub_tex_file.tex}
```

or

```
\input{the_sub_tex_file.tex}
```

This allows you to modularise all of your documents to subfiles, and even strip out a common preamble to apply to all of your documents![3]

[3] The difference between input and include is that input purely inserts the sub-file's TeX in that position in the document, whereas include does some fancy stuff with .aux files to speed up compilation, and also adds page breaks.

Margins and page size

Want to adjust the margins of your page? More importantly, would you like to print your documents at A4 size (since US Letter is the default in LaTeX, you will want to do this for 1-to-1 printing without scaling)?

Simply import the `geometry` package, and use the `\geometry` command in your preamble to specify parameters. For example:

```
\geometry{left=2cm, right=2cm,
          bottom=3cm, top=3cm, a4paper}
```

will put margins on the side at 2cm, margins on the top and bottom at 3cm and make the page A4 sized.

Headers and footers using fancyhdr

Want custom headers and footers? Use the `fancyhdr` package and then, for example, put this in the preamble:

```
\pagestyle{fancy}
\fancyhf{}
\rhead{Right head text}
\lhead{Left head text}
\rfoot{Page number \thepage}
```


TikZ

This is a package for making purely-LaTeX diagrams and graphs. For example, the diagram in one of the earlier slides

was made using the code

```
\begin{tikzpicture}
  \begin{pgfonlayer}{nodelayer}
    \node [style=rectangle] (0) at (0, 0) {\TeX \ code (\texttt{.tex})};
    \node [style=rectangle] (1) at (3.7, 0) {\LaTeX \ compiler};
    \node [style=rectangle] (2) at (7, 0) {PDF output};
  \end{pgfonlayer}
  \begin{pgfonlayer}{edgelayer}
    \draw [style=arrow] (0) to (1);
    \draw [style=arrow] (1) to (2);
  \end{pgfonlayer}
\end{tikzpicture}
```


titlesec/tocloft/sectsty/enumitem/titling

These are all formatting packages for very fine-grained control of output.

I won't discuss them today, but make sure you have a look at them if you ever feel dissatisfied with the default formatting of titles, sections, table of contents or lists. They go *very* in-depth.

beamer

This entire slideshow was made in LaTeX, using the `beamer` document class.

Specialist packages

There are hundreds and thousands of packages for very specialised needs. These include:

- `pgfplots` for more control of plots;
- `physics` for typesetting physics;
- `bm`, `bbm`, `eufrak` for more fancy alphabets;
- `ipa` for IPA symbols in linguistics;
- `tikzcd` for typesetting commutative diagrams;
- `chemfig` for chemical formulae;

...and many many more!

A lecture note setup

Let's consolidate this all together. This is an example lecture note setup, all in LaTeX.

```
|-- assignments
|   |-- math3401-complex
|   \-- stat3001-math-stats
|-- courses
|   |-- comp4403-compilers
|   |-- math3401-complex
|   |-- stat3001-math-stats
|   \-- stat3004-stochastic
|-- header.tex
|-- other
|   \-- the-tex-talk
|-- styles.tikzstyles
\-- tikzit.sty
```


A lecture note setup

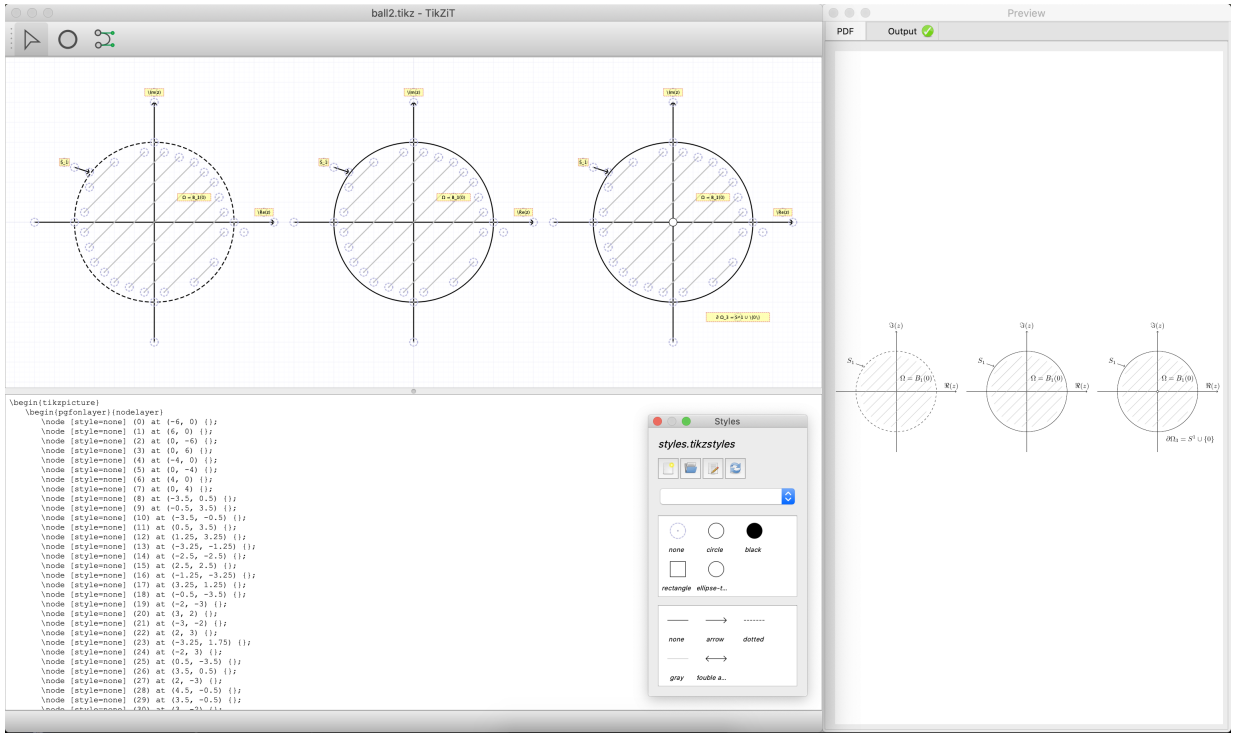
In each folder:

```
|-- 01.aux
|-- 01.tex
|-- 02.aux
|-- 02.tex
...
|-- 15.aux
|-- 15.tex
|-- math3401-complex.pdf
|-- math3401-complex.tex
\-- tikz
    |-- accumulation.tikz
    |-- analytic.tikz
    ...
    \-- zw.tikz
```


A lecture note setup

`math3401-complex.tex` has `\input{../../header.tex}` at the very top. It then includes all of the subfiles `01.tex`, `02.tex` etc. using `\include*{}`.[4].

I also use the application TikZit to draw diagrams without having to type raw TikZ. https://tikzit.github.io/



[4]This is a modified version of include, allowing for the compile speedup of include without page-breaks after every file.

Congratulations!

You can now write modular, clean documents without worrying (too much!) about page layout.

You can write documents (papers, theses, even your assigments) easily and beautifully, and develop your own templates to suit your needs.

Resources

LaTeX is a massive (and wonderful) rabbit hole. Jump in!

- If you have any problems, visit the **TeX Stack Exchange** website (like Stack Overflow, but for TeX questions). https://tex.stackexchange.com
- Read the LaTeX **Wikibook**! https://en.wikibooks.org/wiki/LaTeX
- If you have any more questions, ask in the **UQCS Slack** channel `#latex`, or you can DM me `@mcl`.

Questions?