Beginner's Guide to LaTeX

Why LaTeX?

LaTeX makes beautiful documents, especially for mathematics. It is far more powerful than your standard text editor, and can be extended even further with packages for everything you can think of.

But how does it work?

You write your document in `plain text` with commands that describe its structure and meaning. The LaTeX program then processes your text and commands to produce a formatted document... a lot like HTML and CSS!

The point of such a markup language is to separate content from visual design. If you work in a WYSIWYG editor like MS Word, you may be tempted to waste time fiddling about with making everything look good; LaTeX gives you the opportunity to write all your content undistracted.

Getting Started

A minimal LaTeX document requires a document class, as well as beginning and end tags.

```
\documentclass{article}
\begin{document}
    Your first document!
    % The rest of your content goes here
\end{document}
```

All commands start with a backslash (\\). Every document starts with a `\documentclass` command. The argument in curly braces ({ }) tells LaTeX what kind of document we are creating. In this case, it is an `article`. A percent sign (%) starts a *comment* - LaTeX ignores all characters after a % on a given line.

Getting Started with Overleaf

Overleaf is a website for creating LaTeX documents. It will "compile" your LaTeX and automatically show you the results.

Click here to open the worksheet in Overleaf and complete exercises 1 and 2.

Handling Errors

LaTeX can run into issues when compiling your document, just like any programming language. When this happens, it stops and throws an error, and will not produce any output until this error is fixed.

For example, if you misspell a command, say `\textbf` as `\txetbf` LaTeX will stop with an "Undefined control sequence" error, as "txetbf" is not a command it knows.

In Overleaf, the errors panel is accessed by clicking the page button next to "Recompile". If you have errors, there will be a red icon on this button, and the errors will be listed.

If you encounter an error, don't panic! Look at the LaTeX you have written and try to figure out where it went wrong. If you have multiple errors, start with the first one - it's possible this is leading to all of the others.

Typesetting Maths

As you saw earlier, dollar signs are special characters in LaTeX. This is because they're used to mark mathematics in text (inline notation). Special characters can be escaped with a backslash; so, to insert a dollar sign, you can write `\$`.

For example, "Let a and b be distinct positive integers, and let $c = a - b + 1$" can be written as "`Let $a$ and $b$ be distinct positive intergers, and let $c = a - b + 1$`" Dollar signs are always used in pairs - once to begin and once to end the inline equation. Spacing is also automatically handled in maths environments; any spaces you use will be ignored.

Here's some of the basic notation in maths:

- Use a caret (^) for superscripts and an underscore (_) for subscripts:
  - `$y = c_2 x^2 + c_1 x + c_0$`
  - compiles to $y = c_2 x^2 + c_1 x + c_0$
- These characters only consume one token by default.

  - `$x^123$`
  - compiles to $x^1 23$

  You can use curly braces ({ }) to group superscripts and subscripts into single tokens:

  - `$F_n = F_{n - 1} + F_{n - 2}$`
  - compiles to $F_n = F_{n-1} + F_{n-2}$

- There are commands for Greek letters, beginning with a backslash followed by the name of the character:
  - `$\Omega = \sum_{k = 1}^{n} \omega_k$`
  - compiles to $\Omega = \sum_{k=1}^{n} \omega_k$
- Roots can be written with `\sqrt{}`;
  - `$\sqrt{x+1}$`
  - compiles to $\sqrt{x+1}$

  The base of the root can be specified with an optional argument in square brackets.

  - `$\sqrt[2n]{x+1}$`
  - compiles to $\sqrt[2n]{x+1}$

Generally, commands that take mandatory arguments do so in curly brackets, while optional arguments are given in square brackets;

For larger, more complicated equations, we can display these on their own line with *displaystyle* delimiters. For these, you can use either `\[ ... \]` or `\begin{equation} ...\end{equation}`.

For example,

```
\[ x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \]
```

Will display as

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Exercise 3:

Typeset the following expressions on the worksheet:

$$\sum_{p=1}^{n} (-1)^{p-1} \frac{n!}{p!}$$

$$\frac{\Delta f}{\Delta t} = \lim_{h \to 0} \frac{f(t+h) - f(t)}{h}$$

Extension:

$$\frac{\partial^2 \Psi}{\partial x^2} + \frac{8\pi^2 m}{h^2}(E - V)\Psi = 0$$

Environments

`equation` is an environment – it gives a command context for the output it should be producing.

Inline equations can be denoted by `\begin{math}...\end{math}` (aliased with $ as seen earlier), and display differently to `\begin{equation}...\end{equation}`. For example, the summation sign $\sum_{n=1}^{\infty}$ will be smaller in an inline equation than a displaystyle one, and limits will be displayed on the right to fit in prose. Compare with:

$$\sum_{n=1}^{\infty}$$

The `\begin` and `\end` commands are used to create many different environments. For example, the `itemize` and `enumerate` environments create bulleted and numbered lists respectively.

```
\begin{itemize}
    \item Item 1;
    \item Item 2.
\end{itemize}
```

- Item 1;
- Item 2.

Theorem and Proof Environments

When typesetting maths in LaTeX, you will come across times you need to format a theorem or a proof - thankfully, LaTeX can handle this too. Numbered theorems can be defined by `\newtheorem{theorem}{Theorem}`, which takes 2 arguments:

- The first argument is the name of the environment used
- The second argument is the word printed in bold at the beginning of the environment.

Once this has been defined in the preamble of your LaTeX document (where you import your packages), you can use this with the standard `\begin{}...\end{}` environment syntax.

```
\begin{theorem}
    \dots
\end{theorem}
```

Similarly, proofs can be displayed nicely. The amsthm package (more information on packages below) provides the functionality for this, with `\begin{proof}...\end{proof}`.

This will give the italic "*Proof*" text at the start of the proof, as well as the end proof symbol at the end.

For example:

Theorem 1. *Assume n to be an integer. If n^2 is odd, then n is odd.*

Proof. By contraposition. Suppose n is an integer. If n is even, then n^2 is even. Since n is an even number, we can substitute $n = 2k$. Substituting n for $2k$ in n^2 gives us $n^2 = (2k)^2 = 4k^2 = 2(2k^2)$ and is therefore even. As k is an integer, as is $2k^2$. Therefore, we have proved the contrapositive to be true, meaning the original statement must also be true, hence if n^2 is odd then n must be odd. $\square$

Spacing

Note that in LaTeX, spacing has

Packages

So far, everything we have used has been built into LaTeX. However, LaTeX has libraries, called *packages*, of extra commands and environments.

We have to load each of the packages we want to use with a `\usepackage` command in the preamble (before the `begin{document}` command).

amsmath

The American Mathematical Society has a package called `amsmath`, imported with `\usepackage{amsmath}` command in the preamble. This package provides many useful features, and the documentation for it is here. Some examples of `amsmath` environmenst are given below.

Compare

```
\begin{equation}
    \Omega = \sum_{k - 1}^{n} \omega_k
\end{equation}
```

produces

$$\Omega = \sum_{k-1}^{n} \omega_k \tag{1}$$

with:

```
\begin{align*}
    0 &\leq(x-y)^2 \\
      &= x^2 - 2xy - y^2 \\
      &= (x+y)^2 - 4xy
\end{align*}
```

$$\begin{aligned} 0 &\leq (x-y)^2 \\ &= x^2 - 2xy - y^2 \\ &= (x+y)^2 - 4xy \end{aligned}$$

The `align` environment aligns each line at the & symbol, with linebreaks marked by double backslashes (\\\\).

Note that the & symbol must be placed to the **left** of any relation/operator symbols for correct spacing and alignment.

Also note that the final line is **not** marked with a double backslash. Doing so will add an extra linebreak and cause issues with spacing since the environment already adds spacing to the final line. Also to **not** use a double backslash for a linebreak in text mode prose, and just leave a blank line instead.

```
\begin{gather}
    \nabla\cdot\vec{u}=0\\
    \rho\frac{d\vec{u}}{dt}=-\nabla p-\mu\nabla^2\vec{u}+\mathbf{F}
\end{gather}
```

$$\nabla \cdot \vec{u} = 0 \tag{2}$$

$$\rho\frac{d\vec{u}}{dt} = -\nabla p - \mu\nabla^2\vec{u} + \mathbf{F} \tag{3}$$

The `gather` environment centres several separate lines. This should be used for collecting together several related equations.

There are also “starred” variants of all of these that remove the equation numbering:

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

$$a = b + c \tag{4}$$

$$a = b + c$$

EX 4: details are on the worksheet.