Getting Started with LaTeX

This is a **LaTeX task**. You must submit both your `.tex` source file and a compiled `.pdf` file. Your response will be assessed on both content and presentation.

LaTeX (pronounced "*lay*-tek" or "*lah*-tek") is a markup language that is commonly used to produce academic and technical writing. Typesetting in LaTeX is quite different to word processors such as Microsoft Word, as the WYSIWYG (what you see is what you get) principle doesn't apply; instead, the document is a plain text file and is machine-interpreted into a form that is more human readable. Much like HTML files are rendered by a browser, LaTeX files (with the extension `.tex`) are *compiled* by an *engine*.

One key outcome of this course is to familiarise yourself with the basics of LaTeX. This exercise will help new students get started.

Why LaTeX?

- It's free to use.
- The `.tex` file format is just plain text, so your documents don't rely on any proprietary software and can easily be worked on across different platforms.
- Both the source code and the compiled document have very small file sizes.
- It allows you to create very complex equations and graphics, as well as all the other document elements provided by a word processor.
- It supports referencing within the document and to a bibliography with minimal effort (particularly useful for a thesis).
- It separates content from presentation, so you can easily change the appearance of an existing document.
- It is highly customisable, allowing you extensive control over how your document appears.
- It is versatile, allowing you to write scientific articles, books, resumés, slides, and more.
- It is highly extensible, with a huge number of free additional packages to help you make just about anything you want.

TeX was originally designed in 1978 by Donald Knuth; he was unsatisfied with the typesetting of the second volume of *The Art of Computer Programming*, so he decided to design a better typesetting system himself!

TeX has since had many extensions. By far the most widely used system is LaTeX, designed in 1984 by Leslie Lamport, which adds many useful macros to base TeX in order to make it easier to use.

Workflow

We highly recommend that you use Overleaf. You can register for a free professional account using your UNSW student email at the link above. Overleaf allows you to edit, compile and display your documents within a browser (and download them in PDF[1] format), and manage large projects. It also supports syncing with various cloud storage providers and version control systems.

To work on your local machine, you'll need a few more tools. We *do not* recommend this workflow, but in case you really want to do this (e.g. for working offline):

- You will need to install a TeX distribution. We recommend TeX Live (or the Mac version, MacTeX), but MiKTeX is a popular alternative.
- You'll also need a program to edit the source code.
  - If you use a 'basic' editor such as Vim or Emacs, you can compile your documents from the command line using `pdflatex`.
  - If you use a more powerful editor such as Sublime Text or VS Code, you can compile your documents after selecting the appropriate build settings.
  - There are also specialised editors such as TeXstudio, TeXworks, TeXShop and LyX, but we find that it's best to use software that you're already familiar with.
- Finally, you'll need a PDF reader.
  - Sublime Text users may wish to use Sumatra PDF on Windows or Skim on Mac to support integrations such as inverse search.

Hello, World!

Your first LaTeX document

In almost every future written task, you can simply edit the stub provided in the Task Resources folder. This task however involves starting new files, as we're building up very simple documents.

Start with an empty file named `hello.tex`, and input the following.

```latex
\documentclass{article}
\begin{document}
Hello, World! This is my first document using LaTeX.
\end{document}
```

Warning: Don't copy and paste from the PDF, as this will add extraneous spaces and newlines. You should instead copy and paste from the source code of the task sheet (`1.02 Hello, World!.tex`), or retype the code.

[1] TeX was first designed for high-quality *printing*, so the traditional compilation was `.tex` $\xrightarrow{\texttt{latex}}$ `.dvi` $\xrightarrow{\texttt{dvips}}$ `.ps`, but for *viewing* documents, we simply use PDF.

The section before `\begin{document}` is called the *preamble*, and it contains configuration options for the document. Here, we've chosen an `article` type document, without altering any settings (yet).

The remainder of the document is called the *body*, and it contains the content of the document. For now, we've only included plain text, but we'll add more structure and formatting later on.

For now, build a PDF from your source code, and admire your first LaTeX document.

Control sequences

You may have noticed that throughout this task sheet, we have stylised the name of the language as LaTeX. This will be our first example of a *control sequence*; a backslash followed by a sequence of letters, which signifies a command.

Let's now edit the "Hello, World!" document. Place a backslash before the 'L' in the final word, and recompile.

```
\documentclass{article}
\begin{document}
Hello, World! This is my first document using \LaTeX.
\end{document}
```

One of the most common compilation errors is "Undefined control sequence", which is reported when you use a control sequence that is invalid, i.e. one that is not defined in the base language or any packages you've included, usually because you have mistyped the control sequence. Note that control sequences are case-sensitive.

An aside about spacing

Whitespace in LaTeX is different to other typesetting systems. Whitespace in a `.tex` file is more *semantic* than literal. Placing multiple spaces or a single line break in running text is equivalent to a single space.

```
\documentclass{article}
\begin{document}
Hello,   World!
This is my first document using \LaTeX.
\end{document}
```

A double line break starts a new paragraph.

```
\documentclass{article}
\begin{document}
Hello, World! This here is a very very very very long sentence which should ideally be
    more than one line in the PDF, but conveys no useful information.

This is my first document using \LaTeX. This is yet another very very very very long
   sentence for no reason than to make the default indentation settings apparent.
\end{document}
```

New paragraphs are indented on the first line by default.[2]

One case to watch out for: whitespace characters are not letters, so they can also mark the end of a control sequence instead of producing actual whitespace.

```
\documentclass{article}
\begin{document}
Hello, World! This is my first \LaTeX document.
\end{document}
```

When compiled, no space will appear between the words "LaTeX" and "document". We can either manually insert the space (using backslash-space: `\LaTeX\ document`), or insert an empty group (using braces enclosing nothing to terminate the control sequence: `\LaTeX{} document`).

```
\documentclass{article}
\begin{document}
Hello, World! This is my first \LaTeX\ document.
\end{document}
```

Aside: How might you typeset a backslash?

More on control sequences

`\LaTeX` is a very simple control sequence; it produces a constant output. The real power of control sequences comes from their ability to take arguments. Let's use italics to draw attention to the fact that this is our very first document.

```
\documentclass{article}
\begin{document}
Hello, World! This is my \textit{first} \LaTeX\ document.
\end{document}
```

Everything within the braces is acted upon by the `\textit` control sequence. We could also achieve the same effect using `\emph`, which designates text to be emphasised in a way that is suitable for the context.

```
\documentclass{article}
\begin{document}
Hello, World! This is my \emph{first} \LaTeX\ document.
\end{document}
```

We can nest control sequences, for example, to re-emphasise just what a big accomplishment it is to use LaTeX for the first time.

```
\documentclass{article}
\begin{document}
Hello, World! \emph{This is my \emph{first} \LaTeX\ document.}
\end{document}
```

[2] See here for how to change this, if you want.

What stands out within italicised text? Upright text!

You can also use control sequences to make text bold (`\textbf`), underlined (`\underline`) and more, which each take one argument as above.

However, some control sequences take two or more arguments. To make text a different colour, we'll use the `\textcolor` control sequence, which is defined in the `xcolor` package. We'll first include that package in the preamble, and now we can use the commands defined there in the body.

```latex
\documentclass{article}
\usepackage{xcolor}
\begin{document}
Hello, World! This is my \textcolor{red}{first} \LaTeX\ document.
\end{document}
```

Environments

Some effects, particularly those which are larger in scope or significance, are defined using *environments*. An environment is enclosed by `\begin` and `\end`. You have already seen one environment: `document` processes its contents as the body of the document.

```latex
\documentclass{article}
\usepackage{xcolor}
\begin{document}
Hello, World! This is my \textcolor{red}{first} \LaTeX\ document.
\begin{center}
I can centre-align text using an environment!
\end{center}
\end{document}
```

We will see many important environments in later tasks, allowing us to make tables, insert graphics, draw our own graphics and much more.

Comments

As with any other language, readability is important. It can be useful to include comments for portions of the document that are not to be typeset. The percent character '%' denotes the beginning of an inline comment: the rest of that line will not affect the document.

```latex
\documentclass{article}
\usepackage{xcolor} % pretty colours

\begin{document}
This is my \textcolor{red}{first} \LaTeX\ document. % yay!
\end{document}
```

Block comments are not a feature of the base language. However, the `verbatim` package provides a `comment` environment for this purpose.

```latex
\documentclass{article}
\usepackage{xcolor} % pretty colours
\usepackage{verbatim}

\begin{document}
This is my \textcolor{red}{first} \LaTeX\ document.
\begin{comment}
TODO:
- add a title
- talk about future learning
\end{comment}
\end{document}
```

Aside: how would you typeset a percentage symbol?

Document class features

This text is nice, but it's rather small. The default font size is 10 point; let's increase it to 12. We can do this by providing an argument to the document class. Some packages can also be included with optional arguments.

```latex
\documentclass[12pt]{article}
\usepackage{xcolor}

\begin{document}
Hello, World! This is my \textcolor{red}{first} \LaTeX\ document.
\end{document}
```

Now, let's add a title to our document. The `article` document class allows us to define certain special variables in the preamble and then generate the title from them in the body.

```latex
\documentclass[12pt]{article} % larger font
\usepackage{xcolor} % pretty colours

\title{Hello, World!}
\author{Your name here}
\date{Term X, 20YY}

\begin{document}
\maketitle
This is my \textcolor{colour}{first} \LaTeX\ document. I am \emph{very} excited to
   learn more features!
\end{document}
```

If the `\date` control sequence is omitted from the preamble, then the system date will be used. You can provide an empty string as the argument to `\date` if you want to leave out the date completely.

Lists

We'd like to be able to make ordered and unordered lists, again in much the same way as the `<ol>` and `<ul>` tags in HTML, or numbered and bulleted lists in a word processor. LaTeX achieves this using the `enumerate` and `itemize` environments respectively.

```latex
\documentclass[12pt]{article} % larger font
\usepackage{xcolor} % pretty colours

\title{Hello, World!}
\author{Your name here}
\date{Term X, 20YY}

\begin{document}
\maketitle
This is my \textcolor{colour}{first} \LaTeX\ document. I am \emph{very} excited to
   learn more features!

Some animals I like are:
\begin{enumerate}
    \item my favourite animal,
    \item my second favourite animal, and
    \item my third favourite animal.
\end{enumerate}

Some of the books I've read include:
\begin{itemize}
    \item this book,
    \item that book, and
    \item that other book.
\end{itemize}
\end{document}
```

Note that `algos-tasks.sty` reconfigures the default numbering style from 1., 2., …to (a), (b), …. This style file is included in the solution stubs in the Task Resources, and you can use this styling in future tasks to set out your answers to each part.

Copy the above snippet to `hello.tex`, then edit it with your name, the current term, your favourite colour and some animals and books.

Reminder: Don't copy and paste from the PDF, as this will add extraneous spaces and newlines. You should instead copy and paste from the source code of the task sheet (`1.02 Hello, World!.tex`), or retype the code.

Advice. You should submit both the compiled PDF and the source code from the final example, with your name and the current term, and a colour of your choice.

Document structure

A LaTeX document consists of two main parts; the *preamble* and the *body*.

Preamble

The preamble contains information about how the document is to be configured.

The preamble begins by specifying a *document class*, which is the type of document you want to produce. Some common examples include `article`, `book`, `letter`, `beamer` (for presentations) and `standalone` (for graphics). The format is `\documentclass[options]{class}`.

Next, we include packages. These are the equivalent of libraries or headers in programming languages such as C, adding commands and functionalities that are not conveniently available in the base language. We will commonly include packages such as `amsmath` and `amssymb` (for mathematical symbols), `xcolor` (for colours), `listings` (for code formatting) and `hyperref` (for links), but there are thousands of packages on CTAN. Packages can also be included with optional arguments.

We can also specify custom commands and macros, which will be discussed in a later task.

We can even include custom packages. In the LaTeX templates for tasks in this course, you will see that the preamble always includes `\usepackage{algos-tasks}`, which is a style (`.sty`) file containing various packages and macros that we use to typeset the tasks (and you might also find convenient in your responses).

Body

In the body of the document, LaTeX is always in one of three modes.

- We won't discuss LR mode.
- Paragraph mode is the default, used for processing ordinary text.
- Math mode is used for equations, and has two submodes:
  - *inline* math mode is used for equations in running text, and
  - *display* math mode is used for equations on their own line.

Inline math mode is delimited by either `\(...\)` or `$...$`, and display math mode is delimited by either `\[...\]` or `$$...$$`.

Aside: How might you typeset a dollar sign?

Various mathematical symbols are typeset using control sequences, some of which can only be used in math mode.

```
\documentclass[12pt]{article}
\usepackage{amsmath} % for maths
\title{Typesetting Mathematics}
\author{Your name here}
```

```latex
\begin{document}
\maketitle
The area of a circle is given by $A = \pi r^2$.

The quadratic formula states that the solutions to \(ax^2 + bx + c = 0\) are \[ x = \
   frac{-b \pm \sqrt{b^2-4ac}}{2a}. \]
\end{document}
```

It is often desirable to write a sequence of equations with alignment. For this, we'll use the `align*` environment[3] from the `amsmath` package. The ampersand '`&`' is the alignment marker, and '`\\`' marks a new line.

```latex
\documentclass[12pt]{article}
\usepackage{amsmath}
\title{Typesetting Mathematics}
\author{Your name here}

\begin{document}
\maketitle
The area of a circle is given by $A = \pi r^2$.

The quadratic formula states that the solutions to \(ax^2 + bx + c = 0\) are \[ x = \
   frac{-b \pm \sqrt{b^2-4ac}}{2a}. \]

For example, if $a = 2$, $b = -5$ and $c = 3$, then we have
\begin{align*}
x &= \frac{-(-5) \pm \sqrt{(-5)^2 - 4(2)(3)}}{2(2)}\\
&= \frac{5 \pm \sqrt{25-24}}{4}\\
&= \frac{5 \pm 1}{4}\\
&= \frac{3}{2} \text{ or } 1.
\end{align*}
\end{document}
```

Note the use of `\text` to typeset the word "or" in plain text, rather than in the same style as the variable name x, and the explicit spacing around this word.

Sigma Notation

When describing algorithms, we often use compact sigma notation to describe repeated operations. For example, the operation

```
sum = 0
for i = 1 to n:
    sum = sum + A[i]
```

would be written as $\sum_{i=1}^{n} A[i]$. In LaTeX, this is written as

[3] The unstarred version assigns an equation number, which we will only rarely need.

```latex
\documentclass[12pt]{article}
\usepackage{amsmath} % for maths
\title{Typesetting Mathematics}
\author{Your name here}
\begin{document}
\maketitle
The following sigma notation represents a for loop that sums all the elements of an
   array $A$ of length $n$.
\[\sum_{i=1}^{n} A[i].\]
\end{document}
```

The `\sum` command takes two arguments, `lower` and `upper` each enclosed in `{...}` that define the lower and upper bound for the summation's range. However, where the elements are not ordered (i.e. a set), we should omit the `upper` argument. For example,

$$\sum_{e\in E} \text{weight}(e)$$

denotes the sum of all the edge weights in a graph (all edges $e \in E$).

Other tips you might want to be wary of:

- When quoting, be wary of orienting your inverted commas correctly. Using two inverted commas (`'`) before and after a quote will output two inverted commas in the same direction. You should use a backtick (`` ` ``) to open a quote, and a regular inverted comma to close the quote.
- It's often good to have a cheat sheet ready for commonly used control structures. There are many available online, and it might be worth writing your own as you go along. If you're unsure what command gives a particular symbol, Detexify is an helpful tool available online.
  - It's often worth asking staff about certain norms; say, usage of the `\left` and `\right` operators, and not using the asterisk for multiplication.

For this question, we want to practice using the `\sum` command. Copy the above snippet to `math.tex` and add the answers to the questions to `math.tex` in the `enumerate` environment by using the `\enumerate` command.

(a) We want to write the expression

$$\sum_{i=1}^{2n} f(i).$$

We wrote the following LaTeX code:

```
\sum_i=1^2n f(i)
```

What output does this produce? Identify and correct the errors in the code.

(b) Using sigma notation, write an expression for the total absolute difference between each pair of adjacent elements in an array A of length n. For example, given the array $A = [1, 4, 2, 3]$, the total absolute difference is $|1 - 4| + |4 - 2| + |2 - 3| = 6$.

Note: `\abs{...}` is not a control sequence defined in base LaTeX or the maths packages, but rather a macro that we've defined in `algos-tasks.sty`. You can instead use `\abs*{...}` for bars which scale with the height of the enclosed content.

Reminder: Don't copy and paste from the PDF, as this will add extraneous spaces and newlines. You should instead copy and paste from the source code of the task sheet (`1.02 Hello, World!.tex`), or retype the code.

Advice. You should submit both the compiled PDF and the source code for this question, and answer all the subparts. Detailed instructions for submission are available as comments in `hello.tex` and `math.tex` in the Task Resources folder.