Sumner Evans and Joseph McKinsey
January 10, 2019

Mines Linux Users Group
Overview

- What is \LaTeX?  
- Installing \LaTeX
- \LaTeX Basics — In this section we will cover all of the essentials for how to use \LaTeX.
- How to be \LaTeXperts — A smattering of cool things you can do with \LaTeX.

Note a lot of this presentation was borrowed from Jack’s presentation on \LaTeX: https://github.com/jackrosenthal/lug-latex-presentation.
What is LaTeX?
Lay-teck or Lah-teck or Lay-tecks? Settling the question once and for all.

### From the TeXbook

English words like ‘technology’ stem from a Greek root beginning with the letters $\tau\varepsilon\chi$...; and this same Greek word means *art* as well as technology...

Insiders pronounce the $\chi$ of TeX as a Greek chi, not as an ‘x’, so that TeX rhymes with the word blecchhh... When you say it correctly to your computer, the terminal may become slightly moist.
Lay-teck or Lah-teck or Lay-tecks? Settling the question once and for all.

From \LaTeX{}: A Document Preparation System

One of the hardest things about \LaTeX{} is deciding how to pronounce it. This is also one of the few things I’m not going to tell you about \LaTeX{}, since pronunciation is best determined by usage, not fiat. \TeX{} is usually pronounced teck, making lah-teck, and lay-teck the logical choices.
\LaTeX{} is a *Typesetting* Program

- Typesetting is the art of putting *types* on a page.
- *Types* are physical or digital representations of letters or symbols.
\LaTeX{} is a *Typesetting* Program

- Typesetting is the art of putting *types* on a page.
- **Types** are physical or digital representations of letters or symbols.

https://en.wikipedia.org/wiki/Typesetting#/media/File:Metal_movable_type.jpg
\LaTeX is a \textit{Typesetting} Program

- Ligatures appear in professional typesetting, such as in the word \textit{find} (rather than \textit{find}).

\[ f(x; y; z) = x^2 y^4 + 81 z \sqrt{4 (x^2 + y^2)} \]
\[ = x^2 y^4 + 81 z^2 \sqrt{x^2 + y^2} \]
LaTeX is a *Typesetting* Program

- Ligatures appear in professional typesetting, such as in the word find (rather than find).
- Typesetting can involve complex mathematics. TeX handles this *very* well:

\[ f : \mathbb{R} \times \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R} \]

\[ f(x, y, z) = \frac{x^2 - y^4 + 81z}{\sqrt{4(x^2 + y^2)}} \]

\[ = \frac{x^2 - y^4 + 81z}{2\sqrt{x^2 + y^2}} \]
\[ -\int_0^{2\pi} \frac{kQ \, d\theta}{2\pi (a^2 + x^2)^{3/2}} (a \sin \theta \, \hat{j}) = 0 \]
Installing \LaTeX
Most distributions have a package called something like `texlive`. Here are the specifics for some of the more common distributions in LUG:

- **Arch Linux**: `pacman -S texlive-most`
- **Ubuntu/Debian**: `apt install texlive-full`
- **Fedora**: `dnf install texlive-scheme-full --enablerepo=updates-testing`
On macOS/OS X, you can install \LaTeX{} using the MacTex distribution (https://www.tug.org/mactex/).

On Windows, you can install \LaTeX{} using the MiKTeX distribution (https://miktex.org/). There are other distributions available, but I have no idea which is best.
When you need to work on group reports concurrently, you have options: Overleaf

It even comes with Emacs and Vim keybindings, dark theme, etc. Just hit the menu and scroll for more options.
Online Editing

When you need to work on group reports concurrently, you have options:

CoCalc

CoCalc has a lot more features (like SageMath, Jupyter Notebook, RMarkdown, X11 applications, terminals, and better tooling).
The online editors give you real-time collaboration for free.

If you don’t want to use an online editing program, or if you want to use your own version control, you can store your \LaTeX source\(^1\) in Git.

*Note, I recommend that you keep the PDF in source control as well so that it is convenient to look at the rendered document in GitLab/GitHub.*

\(^1\)You will see why it is \LaTeX source soon.
LaTeX Basics
To use \LaTeX, you create a source file like `latex.tex`.

Write up some code.

Compile your code into a pdf with `pdflatex latex.tex`.

Inspect for errors (most commonly missed step).

**Going Off-road**

Using other features may require using more or different commands. So while editors commonly have features to compile `.tex` code automatically, it’s not guaranteed.
To use \LaTeX, you create a source file like `latex.tex`.

Write up some code.

Compile your code into a pdf with `pdflatex latex.tex`.

Inspect for errors (most commonly missed step).

Going Off-road

Using other features may require using more or different commands. So while editors commonly have features to compile `.tex` code automatically, it’s not guaranteed.
To use \LaTeX, you create a source file like \texttt{latex.tex}.

Write up some code.

Compile your code into a pdf with \texttt{pdflatex latex.tex}.

Inspect for errors (most commonly missed step).

**Going Off-road**

Using other features may require using more or different commands. So while editors commonly have features to compile .\texttt{tex} code automatically, it’s not guaranteed.
Document Structure — Examples

For the bare basics of a document:

Preamble
\begin{document}
Hello World from \LaTeX !
\end{document}

Body
\documentclass[12pt]{article}
\usepackage[margin=1cm]{geometry}
\begin{document}
\begin{equation}
\sum_{n=0}^{\infty} \frac{x^n}{n!} = e^x
\end{equation}
\end{document}

Output
Hello World from \LaTeX !
\begin{equation}
\sum_{n=0}^{\infty} \frac{x^n}{n!} = e^x
\end{equation}
More likely:

\documentclass[11pt]{amsart}
\usepackage{graphicx}
\usepackage{parskip}
\title{Maze Project--CSCI 406}
\author{TeX Lion}
\begin{document}
\maketitle
\maketitle
\end{document}

Many small commands add up to create a clean, nice document.
Control Sequences

- Immediately after typing `\`, TeX expects a control word or symbol.
- A control word consists of a backslash followed by one or more letters.
- A control symbol consists of a backslash followed by a single nonletter.
- Example: `\input MS` causes TeX to begin reading a file called `MS.tex`.
- Example: TeX converts George P'olya and Gabor Szeg"o to ‘George Pólya and Gabor Szegö.’
- A space after a control word is ignored, to fix this, escape the space after a control word when required.
Control Sequences

- Immediately after typing \, TeX expects a control word or symbol.
- A control word consists of a backslash followed by one or more letters.
  - A control symbol consists of a backslash followed by a single nonletter.
  - Example: \input MS causes TeX to begin reading a file called MS.tex.
- Example: TeX converts George P\'olya and Gabor Szeg\"o to ‘George Pólya and Gabor Szegö.’
- A space after a control word is ignored, to fix this, escape the space after a control word when required.
Control Sequences

- Immediately after typing `\`, \TeX{} expects a control word or symbol.
- A control word consists of a backslash followed by one or more *letters*.
- A control symbol consists of a backslash followed by a single *nonletter*.
- Example: \texttt{	extbackslash input} MS causes \TeX{} to begin reading a file called MS.tex.
- Example: \TeX{} converts George P\textbackslash'olya and Gabor Szeg\textbackslash"o to ‘George Pólya and Gabor Szegö.’
- A space after a control word is ignored, to fix this, escape the space after a control word when required.
Control Sequences

- Immediately after typing `\`, \TeX{} expects a control word or symbol.
- A control word consists of a backslash followed by one or more *letters*.
- A control symbol consists of a backslash followed by a single *nonletter*.
- Example: `\input MS` causes \TeX{} to begin reading a file called `MS.tex`.
- Example: \TeX{} converts `George P’olya and Gabor Szeg"o` to ‘George Pólya and Gabor Szegö.’
- A space after a control word is ignored, to fix this, escape the space after a control word when required.
Control Sequences

- Immediately after typing `\`, \TeX\ expects a control word or symbol.
- A control word consists of a backslash followed by one or more \textit{letters}.
- A control symbol consists of a backslash followed by a single \textit{nonletter}.
- Example: `\input` \texttt{MS} causes \TeX\ to begin reading a file called \texttt{MS.tex}.
- Example: \TeX\ converts George \texttt{P\textquoteright olya} and Gabor Szeg\texttt{	extquoteright o} to ‘George Pólya and Gabor Szegő.’
- A space after a control word is ignored, to fix this, escape the space after a control word when required.
Control Sequences

- Immediately after typing \ , \TeX \ expects a control word or symbol.
- A control word consists of a backslash followed by one or more letters.
- A control symbol consists of a backslash followed by a single nonletter.
- Example: \texttt{\input MS} causes \TeX \ to begin reading a file called MS\texttt{.tex}.
- Example: \TeX \ converts George P\texttt{\'olya and Gabor Szeg\"o} to ‘George Pólya and Gabor Szegö.’
- A space after a control word is ignored, to fix this, escape the space after a control word when required.
In most cases, you tend to use curly braces to specify arguments to control sequences.

```
\texttt{command[options]}{argument1}{argument2}
```

Title, Author, Date

The commands `\title{Title}` and `\author{Sumner}` in the preamble define information that `\maketitle` uses to then construct the title section.
In most cases, you tend to use curly braces to specify arguments to control sequences.

\[\text{\texttt{command}}[\text{\texttt{options}}]\{\text{\texttt{argument1}}\}\{\text{\texttt{argument2}}\}\]

**Title, Author, Date**

The commands \texttt{\textbackslash title\{Title\}} and \texttt{\textbackslash author\{Sumner\}} in the preamble define information that \texttt{\textbackslash maketitle} uses to then construct the title section.
\texttt{This} produces teletype typeface output.

We can also see how \{\} allows grouping of sections.

\{\small Text\} \{\footnotesize getting\} \{\tiny smaller\}

Text getting smaller
Do you remember that line: \texttt{\documentclass[11pt]\{article\}}

It specifies the fine details of the formatting somewhere else:

- Fonts: types, sizes, and more
- Margins and page numbers
- Default packages and required presets
- New commands
- Arbitrary code
- Section numbering

You can create your own, which can be really helpful...
Do you remember that line: `\documentclass[11pt]{article}`

It specifies the fine details of the formatting somewhere else:

- Fonts: types, sizes, and more
- Margins and page numbers
- Default packages and required presets.
- New commands
- Arbitrary code
- Section numbering

You can create your own, which can be really helpful...
Do you remember that line: `\documentclass[11pt]{article}`

It specifies the fine details of the formatting somewhere else:

- Fonts: types, sizes, and more
- Margins and page numbers
- Default packages and required presets.
- New commands
- Arbitrary code
- Section numbering

You can create your own, which can be really helpful...
Do you remember that line: `\documentclass[11pt]{article}`

It specifies the fine details of the formatting somewhere else:

- Fonts: types, sizes, and more
- Margins and page numbers
- Default packages and required presets.
- New commands
- Arbitrary code
- Section numbering

You can create your own, which can be really helpful...
Document Classes — What are they?

Do you remember that line: `\documentclass[11pt]{article}`

It specifies the fine details of the formatting somewhere else:

- Fonts: types, sizes, and more
- Margins and page numbers
- Default packages and required presets.
- New commands
- Arbitrary code
- Section numbering

You can create your own, which can be really helpful...
Do you remember that line: `\documentclass[11pt]{article}`

It specifies the fine details of the formatting somewhere else:

- Fonts: types, sizes, and more
- Margins and page numbers
- Default packages and required presets.
- New commands
- Arbitrary code
- Section numbering

You can create your own, which can be really helpful...
Do you remember that line: `\documentclass[11pt]{article}`

It specifies the fine details of the formatting somewhere else:

- Fonts: types, sizes, and more
- Margins and page numbers
- Default packages and required presets.
- New commands
- Arbitrary code
- Section numbering

You can create your own, which can be really helpful...
Do you remember that line: `\documentclass[11pt]{article}`

It specifies the fine details of the formatting somewhere else:

- Fonts: types, sizes, and more
- Margins and page numbers
- Default packages and required presets.
- New commands
- Arbitrary code
- Section numbering

You can create your own, which can be really helpful...
Document Classes — 95% of use cases

But luckily there’s a lot of presets.

- article
- IEEEtran
- proc
- minimal
- report
- book
- memior
- letter
- exam
- leaflet
- beamer
- amsart
- amsproc
- amsbook
- ...

Most journals (such as IEEE, AMS, and ACM) will provide document classes and examples (templates) for “proper” formatting.

You can also specify options like
\documentclass[12pt,a4paper,titlepage,twocolumn]{article}
But luckily there’s a lot of presets.

- article
- IEEEtran
- proc
- minimal
- report
- book
- memior
- letter
- exam
- leaflet
- beamer
- amsart
- amsproc
- amsbook
- ...

Most journals (such as IEEE, AMS, and ACM) will provide document classes and examples (templates) for “proper” formatting.

You can also specify options like
\documentclass[12pt,a4paper,titlepage,twocolumn]{article}
Environments allow for more advanced grouping and nesting. Like:

\begin{itemize}
    \item article
    \item IEEEtran
\end{itemize}

and

\begin{enumerate}
    \item article
    \item IEEEtran
\end{enumerate}
Environments allow for more advanced grouping and nesting. Like:

\begin{itemize}
    \item article
    \item IEEEtran
\end{itemize}

and

\begin{enumerate}
    \item article
    \item IEEEtran
\end{enumerate}
Environments allow for more advanced grouping and nesting. Like:

\begin{itemize}
  \item article
  \item IEEEtran
\end{itemize}

and

\begin{enumerate}
  \item article
  \item IEEEtran
\end{enumerate}
Use the `equation` environment for numbered equations.

\[
\begin{equation}
\lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}
\end{equation}
\]

Output

\[
\lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}
\] (1)
Use the `equation*` environment \([ \]
\) or the more common \([ \)
\) for unnumbered equations.

\[
\lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x},
\]

**Output**

\[
\lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}
\]

\(^2\) usually denotes the unnumbered counterpart, e.g. `section*`
Use $ $ or \(( \)\) for inline math mode.

If $f(x)$ is square integrable, then $\lim_{x \to \pm \infty} f(x) = 0$.

Output
If $f(x)$ is square integrable, then $\lim_{x \to \pm \infty} f(x) = 0$.

There’s way too many symbols and commands to cover here. Overleaf has good documentation and beginner’s guide to guide you from fractions to triple integrals.
You may notice that \( \lim \) looks different.

Inline math mode implicitly uses \textstyle. By calling \texttt{\textbackslash displaystyle}, you can recover the style inline.

If \( f(x) \) is square integrable, then \( \displaystyle \lim_{x \to \pm \infty} f(x) = 0 \).

Let \( f(x) = \displaystyle \frac{\sin x}{x} \).

\begin{itemize}
  \item Output
  \begin{itemize}
    \item If \( f(x) \) is square integrable, then \( \lim_{x \to \pm \infty} f(x) = 0 \).
    \item Let \( f(x) = \frac{\sin x}{x} \).
  \end{itemize}
\end{itemize}

We’ll see some more vital environments later with \texttt{amsmath}. 
You may notice that $\lim$ looks different.

Inline math mode implicitly uses $\textstyle$. By calling $\displaystyle$, you can recover the style inline.

If $f(x)$ is square integrable, then $\lim_{x \to \pm \infty} f(x) = 0$.

Let $f(x) = \frac{\sin x}{x}$.

Output

If $f(x)$ is square integrable, then $\lim_{x \to \pm \infty} f(x) = 0$.

Let $f(x) = \frac{\sin x}{x}$.

We’ll see some more vital environments later with amsmath.
You may notice that \( \lim \) looks different.

Inline math mode implicitly uses \textstyle. By calling \displaystyle, you can recover the style inline.

If \( f(x) \) is square integrable, then \( \lim_{x \to \pm \infty} f(x) = 0 \).

Let \( f(x) = \frac{\sin x}{x} \).

\begin{tabular}{|l|}
\hline
\textbf{Output} \\
\hline
If \( f(x) \) is square integrable, then \( \lim_{x \to \pm \infty} f(x) = 0 \).

Let \( f(x) = \frac{\sin x}{x} \).
\hline
\end{tabular}

We’ll see some more vital environments later with amsmath.
You may notice that \( \lim \) looks different.

Inline math mode implicitly uses \textstyle. By calling \texttt{displaystyle}, you can recover the style inline.

If \( f(x) \) is square integrable, then \( \displaystyle \lim_{x \to \pm \infty} f(x) = 0 \).

Let \( f(x) = \displaystyle \frac{\sin x}{x} \).

**Output**

If \( f(x) \) is square integrable, then \( \lim_{x \to \pm \infty} f(x) = 0 \).

Let \( f(x) = \frac{\sin x}{x} \).

We’ll see some more vital environments later with \texttt{amsmath}.
**Whitespace**

- **LaTeX** treats most forms of whitespace as just a single space.

- Paragraphs come from blank lines (or rarely `\par`).

Command sequences fine tune whitespace for math and text.

<table>
<thead>
<tr>
<th>In increasing width,</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>\</code>, <code>\&gt;</code> <code>\</code>; <code>\.</code> <code>\:</code> <code>\space</code> <code>\enspace</code> <code>\hphantom{xyz}</code> <code>\quad</code></td>
</tr>
<tr>
<td><code>\qquad</code> <code>\hfill</code></td>
</tr>
</tbody>
</table>

`\` marks a new line, common only in math mode and tables.
• *LaTeX* treats most forms of whitespace as just a single space.

• Paragraphs come from blank lines (or rarely \textbackslash par).

Command sequences fine tune whitespace for math and text.

In increasing width,

\textbackslash , \textbackslash > \textbackslash ; \textbackslash . \textbackslash : \textbackslash \space \textbackslash ~ \textbackslash \enspace \textbackslash \hphantom{xyz} \textbackslash \quad
\leftrightarrow \textbackslash \qquad \textbackslash \hfill

\textbackslash \textbackslash marks a new line, common only in math mode and tables.
\texttt{\LaTeX} treats most forms of whitespace as just a single space.

- Paragraphs come from blank lines (or rarely \texttt{\par}).

Command sequences fine tune whitespace for math and text.

\begin{center}
\textbf{In increasing width,}
\end{center}

\begin{verbatim}
, \> \; \. \: \space \~ \enspace \hphantom{xyz} \quad
\rightarrow \qqquad \hfill
\\ marks a new line, common only in math mode and tables.
\end{verbatim}
Due to the many reserved characters and lack of default unicode support, many characters have are instead invoked with commands.

H"uhnchen

Cómo estás

"Don't do this"

``Do this''
\LaTeXX uses a rather complicated algorithm for determining how to lay out flexible elements in your document.

- \LaTeXX has a badness value from 0 to 10,000, where 0 is perfect and 10,000 is infinitely bad.
- When a line is perfect in spacing between words and no hyphenation, the badness will be zero.
- As words get too tight or too narrow or too far apart, the badness increases.
- Hyphenation in words adds a lot of badness!
- \LaTeXX then optimises the badness of each line, trying to get it as low as possible.
\LaTeX uses a rather complicated algorithm for determining how to lay out flexible elements in your document.

- \LaTeX has a badness value from 0 to 10,000, where 0 is perfect and 10,000 is infinitely bad.
- When a line is perfect in spacing between words and no hyphenation, the badness will be zero.
- As words get too tight or too narrow or too far apart, the badness increases.
- Hyphenation in words adds a lot of badness!
- \LaTeX then optimises the badness of each line, trying to get it as low as possible.
\LaTeX{} uses a rather complicated algorithm for determining how to lay out flexible elements in your document.

- \LaTeX{} has a badness value from 0 to 10,000, where 0 is perfect and 10,000 is infinitely bad.
- When a line is perfect in spacing between words and no hyphenation, the badness will be zero.
- As words get too tight or too narrow or too far apart, the badness increases.
- Hyphenation in words adds a lot of badness!
- \LaTeX{} then optimises the badness of each line, trying to get it as low as possible.
\LaTeX \text{ uses a rather complicated algorithm for determining how to lay out flexible elements in your document.}

- \LaTeX \text{ has a badness value from 0 to 10,000, where 0 is perfect and 10,000 is infinitely bad.}

- When a line is perfect in spacing between words and no hyphenation, the badness will be zero.

- As words get too tight or too narrow or too far apart, the badness increases.

- Hyphenation in words adds a lot of badness!

- \LaTeX \text{ then optimises the badness of each line, trying to get it as low as possible.}
\LaTeX uses a rather complicated algorithm for determining how to lay out flexible elements in your document.

- \LaTeX has a badness value from 0 to 10,000, where 0 is perfect and 10,000 is infinitely bad.
- When a line is perfect in spacing between words and no hyphenation, the badness will be zero.
- As words get too tight or too narrow or too far apart, the badness increases.
- Hyphenation in words adds a lot of badness!
- \LaTeX then optimises the badness of each line, trying to get it as low as possible.
**Badness**

**LaTeX** uses a rather complicated algorithm for determining how to lay out flexible elements in your document.

- **LaTeX** has a badness value from 0 to 10,000, where 0 is perfect and 10,000 is infinitely bad.
- When a line is perfect in spacing between words and no hyphenation, the badness will be zero.
- As words get too tight or too narrow or too far apart, the badness increases.
- Hyphenation in words adds a lot of badness!
- **LaTeX** then optimises the badness of each line, trying to get it as low as possible.
Often (especially in academic papers) you need section breaks. \LaTeX{} supports automatically numbered sections out-of-the-box.

\section{Introduction}

\lipsum[1]

\subsection{Related Work}

\lipsum[1]

\subsubsection{Cool Stuff}

\lipsum[1]

\section{Methods}

\lipsum[1]

\section*{No Number}

\lipsum[1]
Pictures are handled using the `graphicx` package. First, include the following in your preamble.

```
\usepackage{graphicx}
```

Then, in the body where you want the figure

```
\includegraphics[width=0.5\textwidth]{graphics/tree.jpg}
```

Options

Note that there are many other options (or you can specify no options), and the filename extension is not necessary.
Pictures are handled using the `graphicx` package.
First, include the following in your preamble.

\usepackage{graphicx}

Then, in the body where you want the figure

\includegraphics[width=0.5\textwidth]{graphics/tree.jpg}

**Options**

Note that there are many other options (or you can specify no options), and the filename extension is not necessary.


Tables

Tables are created using the **`tabular`** environment.

\begin{tabular}{|l|l|}
  \hline
  Program & Review \\
  \hline
  \LaTeX & Good \\
  Microsoft Word & Sucks \\
  \hline
\end{tabular}

<table>
<thead>
<tr>
<th>Program</th>
<th>Review</th>
</tr>
</thead>
<tbody>
<tr>
<td>\LaTeX</td>
<td>Good</td>
</tr>
<tr>
<td>Microsoft Word</td>
<td>Sucks</td>
</tr>
</tbody>
</table>
Tables

Tables are created using the `tabular` environment.

\begin{tabular}{|l|l|}
\hline
Program & Review \\
\hline
\LaTeX & Good \\
Microsoft Word & Sucks \\
\hline
\end{tabular}

The first parameter is the table specification. Mainly, it is used to specify column alignments.

- l, r, c specify left, right, and center justification, respectively.
- | and || specify single and double vertical lines, respectively.
Tables are created using the `tabular` environment.

\begin{tabular}{|l|l|}
\hline
Program & Review \\
\hline
\LaTeX & Good \\
Microsoft Word & Sucks \\
\hline
\end{tabular}

Inside the table body, the following have significance:

- `&` — separates columns.
- `\\` — creates newline.
- `\hline` — creates horizontal line.
Tables

Tables are created using the `tabular` environment.

\begin{tabular}{|l|l|}
\hline
Program & Review \\
\hline
\LaTeX & Good \\
Microsoft Word & Sucks \\
\hline
\end{tabular}

There are many other options and cool things you can do with tables. For example, you can have *paragraph columns* which allow for word wrapping.

Look up the documentation for details.
Automatically Numbered Figures

With \LaTeX, you can create *automatically* numbered figures.

\begin{figure}
  \centering
  \includegraphics[width=0.2\textwidth]{graphics/tree}
  \caption{This is a tree.}
\end{figure}

**Output**

Figure 1: This is a tree.
Automatically Numbered Figures

With \LaTeX, you can create \textit{automatically} numbered \textbf{figures}.

\begin{figure}
  \centering
  \includegraphics[width=0.2\textwidth]{graphics/tree}
  \caption{This is a tree.}
\end{figure}

\textbf{Output}

\textbf{Figure 1:} This is a tree.
With \LaTeX{}, you can create *automatically* numbered tables.

\begin{table}
\centering
\caption{This is a table.}
\begin{tabular}{|l|l|}
\hline
... \\
\hline
\end{tabular}
\end{table}

**Output**

**Table 1:** This is a table.

<table>
<thead>
<tr>
<th>Program</th>
<th>Review</th>
</tr>
</thead>
<tbody>
<tr>
<td>\LaTeX{}</td>
<td>Good</td>
</tr>
<tr>
<td>Microsoft Word</td>
<td>Sucks</td>
</tr>
</tbody>
</table>
With \LaTeX, you can create *automatically* numbered tables.

\begin{table}
  \centering
  \caption{This is a table.}
  \begin{tabular}{|l|l|}
    \hline
    \textbf{Program} & \textbf{Review} \\
    \hline
    \LaTeX & Good \\
    Microsoft Word & Sucks \\
    \hline
  \end{tabular}
\end{table}
In addition to automatically numbering sections, tables, and figures, \LaTeX{} can automatically generate a Table of Contents, List of Tables, List of Figures:

\tableofcontents
\listoffigures
\listoftables

Note on Unnumbered Sections

By default, the Table of Contents will only include \textit{numbered} sections. To add an unnumbered section, you have to add this above your \texttt{section*}.

\addcontentsline{toc}{section}{Unnumbered Section}
\section*{Unnumbered Section}
In addition to automatically numbering sections, tables, and figures, \LaTeX can automatically generate a Table of Contents, List of Tables, List of Figures:

\tableofcontents
\listoffigures
\listoftables

Note on Unnumbered Sections

By default, the Table of Contents will only include \textit{numbered} sections. To add an unnumbered section, you have to add this above your section*.

\addcontentsline{toc}{section}{Unnumbered Section}
\section*{Unnumbered Section}
Tables of Contents — Output

Contents

1 Dummy section 1 ........................................ 2
2 Dummy section 2 ........................................ 2

List of Figures

1 Dummy figure 1 ........................................ 2
2 Dummy figure 2 ........................................ 2

List of Tables

1 Dummy table 1 ........................................ 2

texblog.org/2013/04/29/latex-table-of-contents-list-of-figurestables-and-some-customizations
To reference another table, figure, section, or equation, you can add a \texttt{\label} to the element and then elsewhere use the \texttt{\ref} control sequence to reference it.

\begin{figure}
  \caption{Cool Thing}
  \label{fig:cool}
\end{figure}

See Figure \texttt{\ref{fig:cool}} for details.

\textbf{Figure 2:} Cool Thing

See Figure 2 for details.
To reference another table, figure, section, or equation, you can add a `\label` to the element and then elsewhere use the `\ref` control sequence to reference it.

```latex
\begin{figure}
  \caption{Cool Thing}
  \label{fig:cool}
\end{figure}
```

See Figure \ref{fig:cool} for details.
To reference another table, figure, section, or equation, you can add a \texttt{\label} to the element and then elsewhere use the \texttt{\ref} control sequence to reference it.

\begin{figure}
  \caption{Cool Thing}
  \label{fig:cool}
\end{figure}

See Figure \ref{fig:cool} for details.

\textbf{\texttt{\label} placement}

It is important to put the \texttt{\label} \textit{after} the caption. Don’t ask why.
To reference another table, figure, section, or equation, you can add a \texttt{\label} to the element and then elsewhere use the \texttt{\ref} control sequence to reference it.

\begin{figure}
  \caption{Cool Thing}
  \label{fig:cool}
\end{figure}

See Figure \texttt{\ref{fig:cool}} for details.

This sucks! My reference shows up as “??”!

To fix this, recompile. \LaTeX{} uses an intermediate file to determine what figures/tables/sections exist.
How to learn more?

- **Official Documentation**
- **LaTeX WikiBook**
- **Lists of Symbols**
How to learn more?

- Official Documentation
- LaTeX Wikibook
- Lists of Symbols

The True MVP
Overleaf Documentation
Detexify
Essential Resources

How to learn more?

- Official Documentation
- LaTeX Wikibook
- Lists of Symbols
How to learn more?

- Official Documentation
- LaTeX Wikibook
- Lists of Symbols

The True MVP

Overleaf Documentation

Detexify
Loading the Package

After installing the package, the TikZ-Feynman package can be loaded with `\usepackage{tikz-feynman}` in the preamble. It is recommend that you also specify the version of TikZ-Feynman to use with the `compat` package option: `\usepackage[compat=1.0.0]{tikz-feynman}`. This ensures that any new versions of TikZ-Feynman do not produce any undesirable changes without warning.

A First Diagram

Feynman diagrams can be declared with the `\feynmandiagram` command. It is analogous to the `\tikz` command from TikZ and requires a final semi-colon (`;`) to finish the environment. For example, a simple s-channel diagram is:

```
feynmandiagram [horizontal=a to b] {  i1 -- [fermion] a -- [fermion] i2,  a -- [photon] b,  f1 -- [fermion] b -- [fermion] f2;  }
```
Detexify

Want a Mac app?

Lucky you. The Mac app is finally stable enough. See how it works on Vimeo. Download the latest version here.

Restriction: In addition to the LaTeX command the unlicensed version will copy a reminder to purchase a license to the clipboard when you select a symbol.

You can purchase a license here:

Buy Detexify for Mac
How to be \LaTeX\!Xperts
\LaTeX has a lot of odd holes, so the AMS gave us:

<table>
<thead>
<tr>
<th>The Trifecta</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{amsmath}</td>
</tr>
<tr>
<td>\texttt{amssymb}</td>
</tr>
<tr>
<td>\texttt{amsthm}</td>
</tr>
</tbody>
</table>

A trio so inseparable, no one is sure if they should ever import just one.
**LaTeX** has a lot of odd holes, so the AMS gave us:

<table>
<thead>
<tr>
<th>The Trifecta</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>amsmath</td>
<td>amssymb</td>
</tr>
<tr>
<td></td>
<td>amsthm</td>
</tr>
</tbody>
</table>

A trio so inseparable, no one is sure if they should ever import just one.
Tables are annoying and uncomfortable: so \texttt{amsmath} provides many matrix environments:

\[
\begin{bmatrix}
0 & 2 & 3 & 5 \\
1 & 0 & 6 & 3 \\
2 & 2 & 0 & 2 \\
4 & 9 & 3 & 0
\end{bmatrix}
\begin{pmatrix}
1 & 2 \\
3 & 4
\end{pmatrix}
\]
Don’t even bother trying to use \texttt{eqnarray}.

\begin{align*}
\frac{\partial y}{\partial x} &= \frac{\partial}{\partial x} \left( x \log z - 3 w \right) \\
&= \log z \in \mathbb{R}
\end{align*}

\textsuperscript{3}Here, I also use the resizable \texttt{\left ( \right )} and blackboard symbols \texttt{\mathbb{}}.
• Better Equation Numbering
• Oversets and undersets
• Underbraces and overbraces
• $\text$ command
• Better labeled arrows

• Cases environment
• Math-mode Page Breaks
• Boxed equations
• Custom operators
• Substacks
• short inter text

\footnote{Actually part of the \texttt{mathtools} package, which is a less common extension of \texttt{amsmath}}
Advanced Spacing Hackery

Normally, \LaTeX is great at automatically spacing things for you. But occasionally it just fails. In these cases, you have two options:

1. Try and manipulate the options to get it to do what you want.
2. Use $\texttt{\hspace{<dimension>}}$ and $\texttt{\vspace{<dimension>}}$.

Usage

$\texttt{\hspace{<dimension>}}$ or $\texttt{\vspace{<dimension>}}$

Note that $<\text{dimension}>$ can be negative. Useful when there’s too much padding and you have no idea how to fix it.
Normally, \LaTeX{} is great at automatically spacing things for you. But occasionally it just fails. In these cases, you have two options:

1. Try and manipulate the options to get it to do what you want.

2. Use `\hspace` and `\vspace`.

**Usage**

`\hspace{<dimension>}` or `\vspace{<dimension>}`

Note that `<dimension>` can be negative. Useful when there’s too much padding and you have no idea how to fix it.
Advanced Spacing Hackery

Normally, LaTeX is great at automatically spacing things for you. But occasionally it just fails. In these cases, you have two options:

1. Try and manipulate the options to get it to do what you want.
2. Use \hspace and \vspace.

Usage

\hspace{<dimension>} or \vspace{<dimension>}

Note that <dimension> can be negative. Useful when there’s too much padding and you have no idea how to fix it.
Normally, \LaTeX{} is great at automatically spacing things for you. But occasionally it just fails. In these cases, you have two options:

1. Try and manipulate the options to get it to do what you want.
2. Use $\hspace{}$ and $\vspace{}$.

**Usage**

\begin{verbatim}
\hspace{<dimension>} or \vspace{<dimension>}
\end{verbatim}

Note that $<\text{dimension}>$ can be negative. Useful when there’s too much padding and you have no idea how to fix it.
One of the great things about \LaTeX{} is that you can use it to make presentations. You can get all of the benefits of \LaTeX{} in your presentations!

To make a presentation in \LaTeX{}, use the beamer document class.

All of your knowledge of how to use \LaTeX{} for documents applies.

- You can use sections to make divisions in your presentation. (Note that \LaTeX{}-subsections are ignored.)
- You have to put everything into frame environments.
- To make a title slide for your presentation, create a slide with just \maketitle inside.
One of the great things about \LaTeX{} is that you can use it to make presentations. You can get all of the benefits of \LaTeX{} in your presentations!

To make a presentation in \LaTeX{}, use the \texttt{beamer} document class.

All of your knowledge of how to use \LaTeX{} for documents applies.

- You can use sections to make divisions in your presentation. (Note that [sub]+sections are ignored.)
- You have to put everything into frame environments.
- To make a title slide for your presentation, create a slide with just \texttt{\maketitle} inside.
One of the great things about \LaTeX{} is that you can use it to make presentations. You can get all of the benefits of \LaTeX{} in your presentations!

To make a presentation in \LaTeX{}, use the \texttt{beamer} document class.

All of your knowledge of how to use \LaTeX{} for documents applies.

- You can use sections to make divisions in your presentation. (Note that \texttt{[sub]++sections} are ignored.)
- You have to put everything into \texttt{frame} environments.
- To make a title slide for your presentation, create a slide with just \texttt{\maketitle} inside.
One of the great things about \LaTeX{} is that you can use it to make presentations. You can get all of the benefits of \LaTeX{} in your presentations!

To make a presentation in \LaTeX{}, use the \texttt{beamer} document class.

All of your knowledge of how to use \LaTeX{} for documents applies.

- You can use \texttt{sections} to make divisions in your presentation. (Note that \texttt{[sub]+sections} are ignored.)
- You have to put everything into \texttt{frame} environments.
- To make a title slide for your presentation, create a slide with just \texttt{\maketitle} inside.
One of the great things about \LaTeX{} is that you can use it to make presentations. You can get all of the benefits of \LaTeX{} in your presentations!

To make a presentation in \LaTeX{}, use the beamer document class.

All of your knowledge of how to use \LaTeX{} for documents applies.

- You can use sections to make divisions in your presentation. (Note that [sub]+sections are ignored.)
- You have to put everything into frame environments.
- To make a title slide for your presentation, create a slide with just \texttt{\maketitle} inside.
One of the great things about \LaTeX{} is that you can use it to make presentations. You can get all of the benefits of \LaTeX{} in your presentations!

To make a presentation in \LaTeX{}, use the \texttt{beamer} document class.

All of your knowledge of how to use \LaTeX{} for documents applies.

- You can use \texttt{sections} to make divisions in your presentation. (Note that \texttt{[sub]+sections} are ignored.)
- You have to put everything into \texttt{frame} environments.
- To make a title slide for your presentation, create a slide with just \texttt{\maketitle} inside.
Here’s a basic example Beamer presentation.

\documentclass{beamer}
\begin{document}
\begin{frame}{Frame Title}
Hello World!
\pause % Advance slide to continue
This won't show till you click.
\begin{block}{Cool Information}
This shows in a fancy blue block
\end{block}
\end{frame}
\end{document}
Hello World!

Hello World! This won’t show till you click.

Cool Information
This shows in a fancy blue block
By default, Beamer is pretty ugly. You can select from a multitude of themes, or even spin your own.

This presentation uses a modified version of the metropolis theme. Jack Rosenthal created this version of the theme for LUG.

Other options include Luebeck, Rochester, and Pittsburgh.

To set the theme, just put

\usetheme{Luebeck} \% replace with your preferred theme

after all of the \usepackage lines.
By default, Beamer is pretty ugly. You can select from a multitude of themes, or even spin your own.

This presentation uses a modified version of the metropolis theme. Jack Rosenthal created this version of the theme for LUG.

Other options include Luebeck, Rochester, and Pittsburgh.

To set the theme, just put

\usetheme{Luebeck} \% replace with your preferred theme

after all of the \usepackage lines.
By default, Beamer is pretty ugly. You can select from a multitude of themes, or even spin your own.

This presentation uses a modified version of the metropolis theme. Jack Rosenthal created this version of the theme for LUG.

Other options include Luebeck, Rochester, and Pittsburgh.

To set the theme, just put

\usetheme{Luebeck} % replace with your preferred theme after all of the \usepackage lines.
By default, Beamer is pretty ugly. You can select from a multitude of themes, or even spin your own.

This presentation uses a modified version of the metropolis theme. Jack Rosenthal created this version of the theme for LUG.

Other options include Luebeck, Rochester, and Pittsburgh.

To set the theme, just put

\usetheme{Luebeck} \% replace with your preferred theme

after all of the \usepackage lines.
We showed an example of \texttt{\textbackslash pause} earlier, and we showed that it allows you to hide parts of the slide.

However, it can be tedious to add \texttt{\textbackslash pauses} after each element in a list. To remedy this, there is a cool shorthand:

\begin{itemize}[<+->]
  \item Foo
  \item Bar
\end{itemize}

Result:

- Foo
- Bar
We showed an example of \texttt{\textbackslash pause} earlier, and we showed that it allows you to hide parts of the slide.

However, it can be tedious to add \texttt{\textbackslash pauses} after each element in a list. To remedy this, there is a cool shorthand:

\begin{itemize} [<+->]
  \item Foo
  \item Bar
\end{itemize}

Result:

• Foo
• Bar
We showed an example of \texttt{\textbackslash pause} earlier, and we showed that it allows you to hide parts of the slide.

However, it can be tedious to add \texttt{\textbackslash pauses} after each element in a list. To remedy this, there is a cool shorthand:

\begin{itemize}[^{<-+-}>]
  \item Foo
  \item Bar
\end{itemize}

Result:

- Foo
- Bar
We showed an example of `\pause` earlier, and we showed that it allows you to hide parts of the slide.

However, it can be tedious to add `\pauses` after each element in a list. To remedy this, there is a cool shorthand:

```latex
\begin{itemize}
  \item Foo
  \item Bar
\end{itemize}
```

Result:

- Foo
- Bar
We showed an example of \texttt{\textbackslash pause} earlier, and we showed that it allows you to hide parts of the slide.

However, it can be tedious to add \texttt{\textbackslash pauses} after each element in a list. To remedy this, there is a cool shorthand:

\begin{itemize}[<+->]
  \item Foo
  \item Bar
\end{itemize}

\textbf{Result:}

- Foo
- Bar
One of the coolest things that you can do with LaTeX is have automatically generated citations.

Unfortunately, there are approximately 1000 different ways of doing this, and they are not entirely compatible with one another.

However, when it works correctly, it’s amazing.
One of the coolest things that you can do with \LaTeX is have automatically generated citations.

Unfortunately, there are approximately 1000 different ways of doing this, and they are not entirely compatible with one another.

However, when it works correctly, it’s amazing.
The general idea is that you store citations in a machine-readable format called *BibTeX*.

Normally, you put these citations in a `.bib` file. Each entry in the file looks something like this:

```biblatex
@presentation{Evans-McKinsey:LaTeX:LUG,
  author       = "Evans, Sumner and McKinsey, Joseph",
  title        = "\LaTeX",
  publisher     = "Mines LUG",
  address       = "Golden, CO",
  year          = {2018},
  url           = {https://gitlab.com/sumner/lug-latex}
},
```

There are many different types of entries. This website has a more comprehensive list of the entry types: https://libguides.nps.edu/citation/ieeeebibtex.
The general idea is that you store citations in a machine-readable format called *BibTeX*.

Normally, you put these citations in a `.bib` file. Each entry in the file looks something like this:

```latex
@presentation{Evans-McKinsey:LaTeX:LUG,
  author = "Evans, Sumner and McKinsey, Joseph",
  title = "\LaTeX",
  publisher = "Mines LUG",
  address = "Golden, CO",
  year = {2018}
  url = {https://gitlab.com/sumner/lug-latex}
},
```

There are many different types of entries. This website has a more comprehensive list of the entry types:

https://libguides.nps.edu/citation/ieeебibtex.
The general idea is that you store citations in a machine-readable format called \textit{BibTeX}. Normally, you put these citations in a \texttt{.bib} file. Each entry in the file looks something like this:

\begin{verbatim}
@presentation{Evans-McKinsey:LaTeX:LUG,
  author = "Evans, Sumner and McKinsey, Joseph",
  title = "\LaTeX",
  publisher = "Mines LUG",
  address = "Golden, CO",
  year = {2018}
  url = {https://gitlab.com/sumner/lug-latex},
},
\end{verbatim}

There are many different types of entries. This website has a more comprehensive list of the entry types: 
\url{https://libguides.nps.edu/citation/ieeeebibtex}. 
Citations — Pro Tip

Many academic websites allow you to download citations as .bib files. For example, this is the interface for IEEE Xplore Digital Library:
Once you have the citation in your .bib file, you can cite within your document like so:

To be, or not to be \cite{Shakespeare:Hamlet}.

Then, where you want the actual bibliography in your document, add this line to \texttt{automagically} generate your bibliography:

\bibliography{path/to/references.bib}

\textbf{A note on options}

There are a multitude of options which you can configure (e.g. citation and bibliography styles). There are also many different types of \cite variants which allow you to style the citations further.
Once you have the citation in your .bib file, you can cite within your document like so:

To be, or not to be `\cite{Shakespeare:Hamlet}`.

Then, where you want the actual bibliography in your document, add this line to `automagically` generate your bibliography:

```
\bibliography{path/to/references.bib}
```

A note on options

There are a multitude of options which you can configure (e.g. citation and bibliography styles). There are also many different types of `\cite` variants which allow you to style the citations further.
Once you have the citation in your .bib file, you can cite within your document like so:

To be, or not to be \cite{Shakespeare:Hamlet}.

Then, where you want the actual bibliography in your document, add this line to automagically generate your bibliography:

\bibliography{path/to/references.bib}

A note on options

There are a multitude of options which you can configure (e.g. citation and bibliography styles). There are also many different types of \cite variants which allow you to style the citations further.
When you compile a document and want to use BibTeX, you need to do a sequence something like:

\texttt{xelatex your-file.tex}
\texttt{bibtex your-file}
\texttt{xelatex your-file.tex}
\texttt{xelatex your-file.tex}

The reason for this is that BibTeX uses the .aux files to figure out what citations you need. Nobody understands why you need to have two compilations after calling \texttt{bibtex} though...\textsuperscript{5}

\textsuperscript{5}something something partial .aux compilation
When you compile a document and want to use BibTeX, you need to do a sequence something like:

```
xelatex your-file.tex
bibtex your-file
xelatex your-file.tex
xelatex your-file.tex
```

The reason for this is that BibTeX uses the .aux files to figure out what citations you need. Nobody understands why you need to have two compilations after calling `bibtex` though...
When you compile a document and want to use BibTeX, you need to do a sequence something like:

\texttt{xelatex your-file.tex}
\texttt{bibtex your-file}
\texttt{xelatex your-file.tex}
\texttt{xelatex your-file.tex}

The reason for this is that BibTeX uses the \texttt{.aux} files to figure out what citations you need. Nobody understands why you need to have two compilations after calling \texttt{bibtex} though...\(^5\)

\(^5\)something something partial \texttt{.aux} compilation
Citations — Pitfalls

• Getting group members to use BibTeX for their citations.
• Adhering to strict nonstandard bibliography formats.
• Formatting non-ascii is more difficult.
• Harder to justify the effort with less than 4 entries.
• May not play nice with your page setup.
Citations — Pitfalls

- Getting group members to use BibTeX for their citations.
- Adhering to strict nonstandard bibliography formats.
  - Formatting non-ascii is more difficult.
  - Harder to justify the effort with less than 4 entries.
  - May not play nice with your page setup.
Citations — Pitfalls

- Getting group members to use BibTeX for their citations.
- Adhering to strict nonstandard bibliography formats.
- Formatting non-ascii is more difficult.
- Harder to justify the effort with less than 4 entries.
- May not play nice with your page setup.
• Getting group members to use BibTeX for their citations.
• Adhering to strict nonstandard bibliography formats.
• Formatting non-ascii is more difficult.
• Harder to justify the effort with less than 4 entries.
• May not play nice with your page setup.
Citations — Pitfalls

- Getting group members to use BibTeX for their citations.
- Adhering to strict nonstandard bibliography formats.
- Formatting non-ascii is more difficult.
- Harder to justify the effort with less than 4 entries.
- May not play nice with your page setup.
You can use the \texttt{tikz} library to do generate figures in \LaTeX. Figures you can create include charts, plots, graphs, trees, and other types of figures.

In general, just look up what you are trying to do for examples.

Examples:
You can use the \texttt{tikz} library to do generate figures in \LaTeX. Figures you can create include charts, plots, graphs, trees, and other types of figures.

In general, just look up what you are trying to do for examples.

Examples:
Procedural Figures

You can use the \texttt{tikz} library to do generate figures in \LaTeX. Figures you can create include charts, plots, graphs, trees, and other types of figures.

In general, just look up what you are trying to do for examples.

Examples:
To create tikz pictures, first add the following lines to the preamble:

\usepackage{tikz}
\usepackage{pgfplots} \% sometimes not necessary
\usetikzlibrary{calc,arrows.meta,shapes,graphs}
    \% put whatever tikz libraries you want to use here

Then, in general, you will write all of your tikz code in a \texttt{tikzpicture} environment:

\begin{tikzpicture}
    \% tikz code goes here
\end{tikzpicture}

There’s a lot that could be covered, but really, just copy and paste from the internet until it works.
To create \texttt{tikz} pictures, first add the following lines to the preamble:

\begin{verbatim}
\usepackage{tikz}
\usepackage{pgfplots} \% sometimes not necessary
\usetikzlibrary{calc,arrows.meta,shapes,graphs}
\hspace{1cm} \% put whatever \texttt{tikz} libraries you want to use here
\end{verbatim}

Then, in general, you will write all of your \texttt{tikz} code in a \texttt{tikzpicture} environment:

\begin{verbatim}
\begin{tikzpicture}
\hspace{1cm} \% \texttt{tikz} code goes here
\end{tikzpicture}
\end{verbatim}

There’s a lot that could be covered, but really, just copy and paste from the internet until it works.
To create \texttt{tikz} pictures, first add the following lines to the preamble:

\begin{verbatim}
\usepackage{tikz}
\usepackage{pgfplots} % sometimes not necessary
\usetikzlibrary{calc,arrows.meta,shapes,graphs}
  % put whatever tikz libraries you want to use here
\end{verbatim}

Then, in general, you will write all of your \texttt{tikz} code in a \texttt{tikzpicture} environment:

\begin{verbatim}
\begin{tikzpicture}
  % tikz code goes here
\end{tikzpicture}
\end{verbatim}

There’s a lot that could be covered, but really, just copy and paste from the internet until it works.
• **Graphs** — Graphviz. Uses the `.dot` format, and automatically lays out your graphs for you.\(^6\)

• **Plots** — Python’s matplotlib. Do all your plotting using Python, rather than LaTeX.

\(^6\)Unless you are doing something crazy like trying to graph 10,000 nodes in a dense social network...
Instead of just using `\appendix` and hoping for the best, the appendix package allows options and environments.

\usepackage[\toc,\page]{appendix}
\begin{appendices}
\end{appendices}

It also supports appendix titles and more.
Most customizable and handsome pseudocode package created.

The hardest part is customizing it to look good in the preamble:

Example Preamble

```latex
\usepackage[linesnumbered,commentsnumbered,ruled,vlined]{algorithm2e}
\newcommand\mycommfont[1]{{\footnotesize\ttfamily\textcolor{blue}{#1}}}
\SetCommentSty{mycommfont}
\SetKwComment{tcc}{}{\#}
\SetKwComment{tcp}{}{\#}
...
Most customizable and handsome pseudocode package created.

The hardest part is customizing it to look good in the preamble:

**Example Preamble**

\usepackage[linenumbered,commentsnumbered,ruled,vlined]{algorithm2e}
\newcommand\mycommfont[1]{
  \footnotesize\ttfamily\textcolor{blue}{#1}}
\SetCommentSty{mycommfont}
\SetKwComment{tcc}{#}{
  \#}{
\SetKwComment{tcp}{\#}{
  \#}{
...
\begin{algorithm}[H]
\DontPrintSemicolon
\KwIn{$(b_w, b_l, b_d)$ and $(w_w, w_l, w_d)$, $(score, games)$}
\KwOut{Probability of scoring $(score)$ or better.}
Initialize $(games \times score)$ array called $(dp)$ to 2s;
\For{$g \in (0, \ldots, games)$}{
\tcp{This is a full line comment}
\eIf{$310$}{
Do x and y. \tcc{a little comment.}
}{
Do z. \tcc{This is a comment}
}
\tcc{This is a comment}
\}
\Return $(dp[\text{games}][\text{score}])$;
\caption{DP Probability Algorithm}
\end{algorithm}
Algorithm 1: DP Probability Algorithm

**Input:** $b_w, b_d$ and $w_w, w_d, \text{score, games}$

**Output:** Probability of scoring $\text{score}$ or better.

1. Initialize $(\text{games} \times \text{score})$ array $dp$ to 2s
2. for $g \in (0, \ldots, \text{games})$ do
3.    # This is a full line comment
4.    if 310 then
5.        Do x and y.  # a little comment.
6.    else
7.        Do z.
8.    # This is a comment
9. return $dp[\text{games}][\text{score}]$
You can link within a document and to the World Wide Web.\footnote{It also automatically from \texttt{refs} to \texttt{labels}.}

\url{https://google.com} or \href{https://google.com}{this}.

<table>
<thead>
<tr>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="https://google.com">https://google.com</a> or this.</td>
</tr>
</tbody>
</table>
The cleanest code highlighting in the west:  

\begin{minted}{python}
    def hello():
        print("hello")
\end{minted}

\textbf{Output}

\begin{verbatim}
def hello():
    print("hello")
\end{verbatim}

\footnote{Sadly, you can’t nest minted environments.}
\usepackage{parskip}

\usepackage{parskip}

No paragraph indentation. Just empty lines.
Pastes other .pdfs directly into your pdf.\footnote{Except in Beamer. If you try to do this in a Beamer presentation is blows up for some reason. Just use \texttt{includegraphics}.}

\begin{verbatim}
\usepackage[final]{pdfpages}
\includepdf[pages=-]{file.pdf}
\end{verbatim}
For those poor souls who have to care about units in their math, there is a better way than \text{\{\}.}

\[
\begin{align*}
x &= \SI{500}{\cubic\metre\per\hour} \\
\neq \SI{1023}{\m^3\per\hour}
\end{align*}
\]

Output

\[x = 500 \text{ m}^3 \text{ h}^{-1} \neq 1023 \text{ m}^3 / \text{h}]

You can easily define your own units and shortcuts.
For those poor souls who have to care about units in their math, there is a better way than \text{}.

\[
x = \SI{500}{\cubic\metre\per\hour} \neq \SI{1023}{\m^3\per\hour}
\]

Output
\[
x = 500 \, \text{m}^3 \, \text{h}^{-1} \neq 1023 \, \text{m}^3 / \text{h}
\]

You can easily define your own units and shortcuts.
For those poor souls who have to care about units in their math, there is a better way than $\text{text}{}$.

```
x = \SI{500}{\cubic\metre\per\hour}
\neq \SI{1023}{\m^3\per\hour}
```

**Output**

\[ x = 500 \text{ m}^3 \text{ h}^{-1} \neq 1023 \text{ m}^3 / \text{h} \]

You can easily define your own units and shortcuts.
For those poor souls who have to care about units in their math, there is a better way than \text{}.

\[
x = \SI{500}{{\cubic\metre\per\hour}} \\
\neq \SI{1023}{{\m^3\per\hour}}
\]

**Output**

\[x = 500 \, \text{m}^3 \, \text{h}^{-1} \neq 1023 \, \text{m}^3 / \text{h}\]

You can easily define your own units and shortcuts.
\LaTeX can be a very lengthy, so one option to mitigate that is to define your own commands in the preamble:\(^{10}\)

\textbf{The `\texttt{newcommand}` command}

\texttt{\textbackslash newcommand\{\textbackslash yournewcommand\}[2]\{Something #1, and #2\}}

\textbf{Output}

\texttt{yournewcommand\{test\}\{this\} \Rightarrow Something test, and this}

\(^{10}\text{Use `\texttt{renewcommand}` to redefine something.}\)
LaTeX can be a very lengthy, so one option to mitigate that is to define your own commands in the preamble:\(^{10}\)

**The \texttt{\newcommand} command**

\newcommand{\yournewcommand}{[2]{Something #1, and #2}}

**Output**

\yournewcommand{test}{this} ⇒ Something test, and this

\(^{10}\)Use \texttt{\renewcommand} to redefine something.
Defining Your Own Commands — A Real Small Example

\newcommand{\Z}{\mathbb{Z}}

Small Example

Let $x \in \Z$.

Let $x \in \mathbb{Z}$.
\newcommand{\mathbb{Z}}{Z}

Small Example

Let $x \in \mathbb{Z}$.

Let $x \in \mathbb{Z}$
\newcommand{\tikzmark}[1][]{%
  \tikz[baseline=-0.55ex,overlay,remember picture]
  \node[inner sep=0pt,](#1),
  \vphantom{T};
}

\newcommand{\braced}[3][]{%
  \begin{tikzpicture}[overlay,remember picture]
    \draw
    (\(\#2\).south west-|T1.south west) --
    node[anchor=west,left,xshift=-1.8ex,text=olive]{\(\#3\)}
    (#1.north west-|T1.south west);
  \end{tikzpicture}
}
\begin{center}
tikzmark{E1}
This could be any thing with nice boundaries.

Hopefully with multiple lines.
tikzmark{E2}
end{center}
\braced{E1}{E2}{Sentence}

<table>
<thead>
<tr>
<th>Sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td>This could be any thing with nice boundaries.</td>
</tr>
<tr>
<td>Hopefully with multiple lines.</td>
</tr>
</tbody>
</table>
Let’s say you need a new environment, maybe for exam questions or proofs. You can define a completely new environment with \newenvironment

\newenvironment{<name>}{<begin code>}{<end code>}

\renewenvironment redefines an environment. Square brackets can define arguments.
\newenvironment{question}[1]\{\par\textbf{Question #1.}}\par\}{}
\newenvironment{question}{\par\textbf{Question #1.}}{\par}{ }

**Small Example**

\begin{question}{2-3}
  \begin{itemize}
    \item This could be where you put your answer.
  \end{itemize}
\end{question}

**Output**

Question 2-3.

This could be where you put your answer.
\newenvironment{question}[1]{\par\textbf{Question #1.}}{\par}

Small Example

\begin{question}{2-3}
  This could be where you put your answer.
\end{question}

Output

Question 2-3.

This could be where you put your answer.
With the \texttt{amsthm} package, you can define your own theorems very easily with some basic \texttt{theoremstyles}. Notable features include:

- Numbered Environments.
- Quickly creating several simple environments: remarks, lemmas, etc.
- Built in styles.
With the `amsthm` package, you can define your own theorems very easily with some basic `theoremstyles`. Notable features include:

- Numbered Environments.
- Quickly creating several simple environments: remarks, lemmas, etc.
- Built in styles.
With the `amsthm` package, you can define your own theorems very easily with some basic `theoremstyles`. Notable features include:

- Numbered Environments.
- Quickly creating several simple environments: remarks, lemmas, etc.
- Built in styles.
You probably have a lot of boiler plate in your preamble. You have two options:

- Create a package (.sty): your commands can be used anywhere.
- Create a document class (.cls): they are part of a type of document.

Both options force you to use a greater subset of \LaTeX\ commands.

I will focus on creating a document class, since college tends to have a lot of boilerplate for assignments. You can find my own document class at https://github.com/josephmckinsey/hw-document-class.
Defining Your Own Document Classes/Packages

You probably have a lot of boiler plate in your preamble. You have two options:

- Create a package (.sty): your commands can be used anywhere.
- Create a document class (.cls): they are part of a type of document.

Both options force you to use a greater subset of \LaTeX{} commands.

I will focus on creating a document class, since college tends to have a lot of boilerplate for assignments. You can find my own document class at https://github.com/josephmckinsey/hw-document-class.
You probably have a lot of boiler plate in your preamble. You have two options:

- Create a package (.sty): your commands can be used anywhere.
- Create a document class (.cls): they are part of a type of document.

Both options force you to use a greater subset of \LaTeX commands.

I will focus on creating a document class, since college tends to have a lot of boilerplate for assignments. You can find my own document class at https://github.com/josephmckinsey/hw-document-class.
You probably have a lot of boiler plate in your preamble. You have two options:

- Create a package (.sty): your commands can be used anywhere.
- Create a document class (.cls): they are part of a type of document.

Both options force you to use a greater subset of \LaTeX commands.

I will focus on creating a document class, since college tends to have a lot of boilerplate for assignments. You can find my own document class at [https://github.com/josephmckinsey/hw-document-class](https://github.com/josephmckinsey/hw-document-class).
Defining Your Own Document Classes

You have a section for identification:

\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{<class-name>}[<date> <other information>]

Then you can import your own packages with:\textsuperscript{12}

\RequirePackage[<options>]{<package>}[<date>]

And classes with:

\LoadClass[<options>]{<package>}[<date>]

\textsuperscript{12}You can also pass along options with
\RequirePackageWithOptions {Package}
Defining Your Own Document Classes

You have a section for identification:

\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{<class-name>}[<date> <other information>]

Then you can import your own packages with:\textsuperscript{12}

\RequirePackage[<options>]{<package>}[<date>]

And classes with:

\LoadClass[<options>]{<package>}[<date>]

\textsuperscript{12}You can also pass along options with
\RequirePackageWithOptions{<Package>}

{\textsuperscript{12}}
You have a section for identification:

\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{<class-name>}[<date> <other information>]

Then you can import your own packages with:

\RequirePackage[<options>]{<package>}[<date>]

And classes with:

\LoadClass[<options>]{<package>}[<date>]

\textsuperscript{12}You can also pass along options with

\RequirePackageWithOptions {Package}
The `lug.cls` document class:

\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{lug}[2016/08/31]
\LoadClass{beamer}
\usetheme[numbering=none,
            progressbar=frametitle,
            block=fill]{metropolis}
\setbeamercovered{dynamic}
\RequirePackage{graphicx}
\RequirePackage{ifxetex}
\ifxetex\RequirePackage{fontspec}\fi
\RequirePackage{minted}
\RequirePackage{xcolor}
\RequirePackage{hyperref}
...
Questions?
Resources

- For examples of presentations in \LaTeX{}, just go to the LUG website and you can see the source from all of the previous presentations for inspiration.

- clsguide.pdf is a great 30-pg intro to document classes and package writing.

- Overleaf

- Detexify

- \LaTeX{} Wikibook
This presentation was from the Mines Linux Users Group. A mostly-complete archive of our presentations can be found online at https://lug.mines.edu.

Individual authors may have certain copyright or licensing restrictions on their presentations. Please be certain to contact the original author to obtain permission to reuse or distribute these slides.