1-1-2016

An Introduction to LaTeX

Clayton Hayes
Wayne State University, as6348@wayne.edu

Recommended Citation

Hayes, Clayton, "An Introduction to LaTeX" (2016). *Library Scholarly Publications.* Paper 110.
http://digitalcommons.wayne.edu/libsp/110

An Introduction to LaTeX

C. Hayes

March 11, 2016

What is LaTeX?

LaTeX is a *markup* language for creating attractively typeset documents of a variety of types.

- WYSISYG vs. Markup

What is LaTeX?

LaTeX is a *markup* language for creating attractively typeset documents of a variety of types.

- WYSISYG vs. Markup
- Benefits of using LaTeX

What is LaTeX?

LaTeX is a *markup* language for creating attractively typeset documents of a variety of types.

- WYSISYG vs. Markup
- Benefits of using LaTeX
  - Free and Open Source
  - Handles mathematical equations and symbols very well
  - Puts you in control of basically all aspects of your document

What is LATEX?

LATEX is a *markup* language for creating attractively typeset documents of a variety of types.

- WYSISYG vs. Markup
- Benefits of using LATEX
  - Free and Open Source
  - Handles mathematical equations and symbols very well
  - Puts you in control of basically all aspects of your document
- LATEX is a standard in many of the sciences

What is LaTeX?

LaTeX is a *markup* language for creating attractively typeset documents of a variety of types.

- WYSISYG vs. Markup
- Benefits of using LaTeX
  - Free and Open Source
  - Handles mathematical equations and symbols very well
  - Puts you in control of basically all aspects of your document
- LaTeX is a standard in many of the sciences
  - Journal/arXiv templates
  - NSF grant proposal template
  - AMS has its own package!

What is LaTeX?

LaTeX is a *markup* language for creating attractively typeset documents of a variety of types.

- WYSISYG vs. Markup
- Benefits of using LaTeX
  - Free and Open Source
  - Handles mathematical equations and symbols very well
  - Puts you in control of basically all aspects of your document
- LaTeX is a standard in many of the sciences
  - Journal/arXiv templates
  - NSF grant proposal template
  - AMS has its own package!

When should you *not* use LaTeX?

A Closer Look at TEXample.tex

Open up `texample.tex` in a text editor (like Notepad++).

The main parts of a `tex` file are:

A Closer Look at TEXample.tex

Open up `texample.tex` in a text editor (like Notepad++).

The main parts of a `tex` file are:

- The `documentclass` declaration
- The rest of the *preamble*
- The body of the document, everything between `\begin{document}` and `\end{document}`

The documentclass

```
\documentclass[option1,option2,...]{class}
```

The documentclass

```
\documentclass[option1,option2,...]{class}
```

Every document starts with a `documentclass` declaration, which tells LaTeX what kind of document to create when it compiles your `tex` file.

- The most common class is *article*, and it's a good catchall class
- Other common classes: *report*, *book*, *letter*, and *beamer*
- There are also lots of *options* you can specify:

The `documentclass`

```
\documentclass[option1,option2,...]{class}
```

Every document starts with a `documentclass` declaration, which tells LaTeX what kind of document to create when it compiles your `tex` file.

- The most common class is *article*, and it's a good catchall class
- Other common classes: *report*, *book*, *letter*, and *beamer*
- There are also lots of *options* you can specify:

10pt, 11pt, etc.	Font size (10pt is default)
a4paper, letterpaper,...	Defines the size of your paper
twoside, oneside	Double- or single-sided output

The Preamble

The preamble is everything between the `documentclass` declaration and `\begin`{*`document`*}.

This is where you would specify which `packages` your document will be using, along with any *`options`* for those packages, and any other options or information that isn't necessarily a part of the document's content.

We'll talk more about packages later, but other things that go in the preamble:

- Setting lengths of spaces before/after paragraphs, line height, etc.
- Specifying author/title/date, etc. (important if you will be making a title page)

The Body of the Document

The body of the document is everything between `\begin{document}` and `\end{document}`.

This is where you fill in the actual content of your document.

The Body of the Document

The body of the document is everything between `\begin`{*document*} and `\end`{*document*}.

This is where you fill in the actual content of your document.

You can organize your document using `\section`{ }, `\subsection`{ }, and, in the case of the *report* or *book* document classes, `\chapter`{ }. Your PDF output will include these sections as bookmarks.

Compiling Using the Command Line or Terminal

Let's compile `texample.tex`.

Compiling Using the Command Line or Terminal

Let's compile `texample.tex`.

- For Windows, open the command line: Start → Search for "cmd"
- For Macs, open the terminal: Applications → Utilities → Terminal

Compiling Using the Command Line or Terminal

Let's compile `texample.tex`.

- For Windows, open the command line: Start → Search for "cmd"
- For Macs, open the terminal: Applications → Utilities → Terminal

Navigate to the folder that contains `texample.tex` using `cd`.

Compiling Using the Command Line or Terminal

Let's compile `texample.tex`.

- For Windows, open the command line: Start → Search for "cmd"
- For Macs, open the terminal: Applications → Utilities → Terminal

Navigate to the folder that contains `texample.tex` using `cd`.

Try typing in `latex texample.tex`.

Compiling Using the Command Line or Terminal

Let's compile `texample.tex`.

- For Windows, open the command line: Start → Search for "cmd"
- For Macs, open the terminal: Applications → Utilities → Terminal

Navigate to the folder that contains `texample.tex` using `cd`.

Try typing in `latex texample.tex`.

Okay, now try `pdflatex texample.tex`.

Compiling Using TEXmaker

Now let's open `texample.tex` in TEXmaker. Mess around! See what the options are!

Compiling Using TEXmaker

Now let's open `texample.tex` in TEXmaker. Mess around! See what the options are!

So, text editor vs. TEXmaker

Compiling Using TEXmaker

Now let's open `texample.tex` in TEXmaker. Mess around! See what the options are!

So, text editor vs. TEXmaker

- TEXmaker makes things easier as it remembers stuff for you
- It does a bit better with debugging
- The preview is very helpful!

Compiling Using TEXmaker

Now let's open `texample.tex` in TEXmaker. Mess around! See what the options are!

So, text editor vs. TEXmaker

- TEXmaker makes things easier as it remembers stuff for you
- It does a bit better with debugging
- The preview is very helpful!

But...

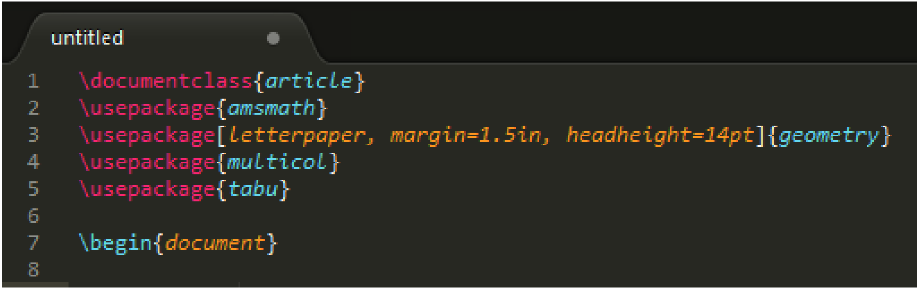
- Using a text editor helps you learn commands and processes by heart
- I just find text editors to be a lot easier on the eyes

Packages

LaTeX can't do everything by default. In fact, you'll find that it can't do all that much by default. That's where packages come in. You call them in the preamble, after `\documentclass` and before `\begin{document}`.

Packages

LaTeX can't do everything by default. In fact, you'll find that it can't do all that much by default. That's where packages come in. You call them in the preamble, after `\documentclass` and before `\begin{document}`.



```latex
\documentclass{article}
\usepackage{amsmath}
\usepackage[letterpaper, margin=1.5in, headheight=14pt]{geometry}
\usepackage{multicol}
\usepackage{tabu}

\begin{document}

```

Anatomy of a Package Call

```
\usepackage[option1,option2,...]{package}
```

Anatomy of a Package Call

```
\usepackage[option1,option2,...]{package}
```

package is the name of the package you wish to use.

option1, etc. are any options that you want to specify with that package. These vary widely from package to package, and are usually laid out in the package's documentation.

Anatomy of a Package Call

```
\usepackage[option1,option2,...]{package}
```

package is the name of the package you wish to use.

option1, etc. are any options that you want to specify with that package. These vary widely from package to package, and are usually laid out in the package's documentation.

From the previous slide, *geometry* is a package that helps fine-tune how your document is laid out.

```
\usepackage[letterpaper, margin=1.5in, headheight=14pt]{geometry}
```

Anatomy of a Package Call

```
\usepackage[option1,option2,...]{package}
```

`package` is the name of the package you wish to use.

`option1, etc.` are any options that you want to specify with that package. These vary widely from package to package, and are usually laid out in the package's documentation.

From the previous slide, `geometry` is a package that helps fine-tune how your document is laid out.

```
\usepackage[letterpaper, margin=1.5in, headheight=14pt]{geometry}
```

Why use this over the options in the documentclass declaration?

Where do Packages Come From?

Anyone who wants to can make a package, so there are a lot of them out there. It wouldn't be feasible for LaTeX to install *all* of them right off the bat. Instead, it installs some core packages and downloads and installs any others that are called on the spot.

Where do Packages Come From?

Anyone who wants to can make a package, so there are a lot of them out there. It wouldn't be feasible for LaTeX to install *all* of them right off the bat. Instead, it installs some core packages and downloads and installs any others that are called on the spot.

Add `\usepackage{geometry}` to the preamble of `texample.tex`, and then try to compile it.

Where do Packages Come From?

Anyone who wants to can make a package, so there are a lot of them out there. It wouldn't be feasible for LaTeX to install *all* of them right off the bat. Instead, it installs some core packages and downloads and installs any others that are called on the spot.

Add `\usepackage{geometry}` to the preamble of `texample.tex`, and then try to compile it.

They can be manually installed, too:

- Windows: Start → MikTeX → Maintenance → MikTeX Package Manager
- Mac: This is actually pretty hard! We can talk about it if need be

Useful Packages

Here are some common and useful packages:

Useful Packages

Here are some common and useful packages:

- *amsmath* is probably most important, and should come installed

Useful Packages

Here are some common and useful packages:

- *amsmath* is probably most important, and should come installed
- *geometry*, as mentioned, helps control the size of the document

Useful Packages

Here are some common and useful packages:

- `amsmath` is probably most important, and should come installed
- `geometry`, as mentioned, helps control the size of the document
- `graphicx` allows you to insert images into your document

Useful Packages

Here are some common and useful packages:

- *amsmath* is probably most important, and should come installed
- *geometry*, as mentioned, helps control the size of the document
- *graphicx* allows you to insert images into your document
- *multicol* lets you have 3+ columns (LaTeX only natively supports 2)

Useful Packages

Here are some common and useful packages:

- *amsmath* is probably most important, and should come installed
- *geometry*, as mentioned, helps control the size of the document
- *graphicx* allows you to insert images into your document
- *multicol* lets you have 3+ columns (LaTeX only natively supports 2)
- *hyperref* gives you more options for hyperlinks in your document

Useful Packages

Here are some common and useful packages:

- *amsmath* is probably most important, and should come installed
- *geometry*, as mentioned, helps control the size of the document
- *graphicx* allows you to insert images into your document
- *multicol* lets you have 3+ columns (LaTeX only natively supports 2)
- *hyperref* gives you more options for hyperlinks in your document

How do you know what packages to use for what?

Formatting Text

Note: LaTeX is *very* case-sensitive.

You can end a paragraph of text with `\par`, or you can line break within a paragraph with `\\`.

Formatting Text

Note: LaTeX is *very* case-sensitive.

You can end a paragraph of text with `\par`, or you can line break within a paragraph with `\\`.

- To make text **bold**, use `\textbf{bold text}`
- For *italics*, use `\textit{italicised text}`
- For underline, use `\underline{underlined text}`

Formatting Text

Note: LaTeX is *very* case-sensitive.

You can end a paragraph of text with `\par`, or you can line break within a paragraph with `\\`.

- To make text **bold**, use `\textbf{bold text}`
- For *italics*, use `\textit{italicised text}`
- For underline, use `\underline{underlined text}`

You can also use `\emph{}` for italics, but what it does sometimes changes based on context.

Formatting Text

Note: LaTeX is *very* case-sensitive.

You can end a paragraph of text with `\par`, or you can line break within a paragraph with `\\`.

- To make text **bold**, use `\textbf{bold text}`
- For *italics*, use `\textit{italicised text}`
- For <u>underline</u>, use `\underline{underlined text}`

You can also use `\emph{}` for italics, but what it does sometimes changes based on context.

LaTeX documents are fully-justified by default, but you can change the text alignment by surrounding it with (for example) `\begin{center}` and `\end{center}` to center it. You can use `flushleft` and `flushright` to left and right justify, respectively.

Formatting Text

The previous slide actually demonstrates the two main methods for affecting text in LaTeX:

Formatting Text

The previous slide actually demonstrates the two main methods for affecting text in LaTeX:

Commands:

The text formatting options in the previous slide are called *commands*, and have the general structure `\command{affected text}`. Commands are used to affect small bits of text, usually inline.

Formatting Text

The previous slide actually demonstrates the two main methods for affecting text in LaTeX:

Commands:

The text formatting options in the previous slide are called *commands*, and have the general structure `\command{affected text}`. Commands are used to affect small bits of text, usually inline.

Environments: The other class of options are called *environments*. They have the general structure of

```
\begin{environment}
affected text
\end{environment}
```

Environments usually set larger blocks of text apart from the normal flow of the document.

Some Notes on Sizing and Spacing

As mentioned above, you set the base text size in the `documentclass` declaration. All of the decisions about spacing and sizing that LaTeX makes are based off of that (default is *10pt*).

Some Notes on Sizing and Spacing

As mentioned above, you set the base text size in the `documentclass` declaration. All of the decisions about spacing and sizing that LaTeX makes are based off of that (default is *10pt*).

Units:

- **pt**, where 1pt is 0.0138in or 0.3515mm
- **mm** or **cm**, millimeters or centimeters
- **in**, inches
- **ex** or **em**, about the width of an 'x' or the height of an 'M'

Some Notes on Sizing and Spacing

As mentioned above, you set the base text size in the `documentclass` declaration. All of the decisions about spacing and sizing that LaTeX makes are based off of that (default is *10pt*).

Units:

- **pt**, where 1pt is 0.0138in or 0.3515mm
- **mm** or **cm**, millimeters or centimeters
- **in**, inches
- **ex** or **em**, about the width of an 'x' or the height of an 'M'

In the preamble, you can use the command `\setlength`{ } to define spacing around environments in your document. For example: `\setlength{\parskip}{10pt}` tells LaTeX that you want 10pt of space between paragraphs.

Some Notes on Sizing and Spacing

As mentioned above, you set the base text size in the `documentclass` declaration. All of the decisions about spacing and sizing that LaTeX makes are based off of that (default is *10pt*).

Units:

- **pt**, where 1pt is 0.0138in or 0.3515mm
- **mm** or **cm**, millimeters or centimeters
- **in**, inches
- **ex** or **em**, about the width of an ‘x’ or the height of an ‘M’

In the preamble, you can use the command `\setlength`{ } to define spacing around environments in your document. For example: `\setlength`{`\parskip`}{10pt} tells LaTeX that you want 10pt of space between paragraphs.

Other lengths you can set are things like `\parindent`, the indentation on each paragraph, or `\abovedisplayskip` and `\belowdisplayskip` for the space above and below the display math environment.

Some Notes on Sizing and Spacing

You can change text size using different commands, like:

Some Notes on Sizing and Spacing

You can change text size using different commands, like:

- `\tiny{tiny text}`
- `\small{small text}`
- `\normalsize{normal text}` (the default)
- `\large{large text}`
- `\huge{huge text}`

Some Notes on Sizing and Spacing

You can change text size using different commands, like:

- `\tiny{tiny text}`
- `\small{small text}`
- `\normalsize{normal text}` (the default)
- `\large{large text}`
- `\huge{huge text}`

Here's a chart of what they look like.

Some Notes on Sizing and Spacing

You can change text size using different commands, like:

- `\tiny{tiny text}`
- `\small{small text}`
- `\normalsize{normal text}` (the default)
- `\large{large text}`
- `\huge{huge text}`

Here's a chart of what they look like.

You can also insert horizontal or vertical space using `\hspace{length}` or `\vspace{length}`, though you should be careful of using them too much.

Some Notes on Sizing and Spacing

You can change text size using different commands, like:

- `\tiny{tiny text}`
- `\small{small text}`
- `\normalsize{normal text}` (the default)
- `\large{large text}`
- `\huge{huge text}`

Here's a chart of what they look like.

You can also insert horizontal or vertical space using `\hspace{length}` or `\vspace{length}`, though you should be careful of using them too much.

Two other useful commands are `\quad` and `\qquad`, which insert horizontal space of length 1em and 2em, respectively.

Math Mode

The *amsmath* package is the backbone of using LaTeX to do math. As mentioned previously, it should already be installed. You still have to include `\usepackage`[*amsmath*], though.

Math Mode

The *amsmath* package is the backbone of using LATEX to do math. As mentioned previously, it should already be installed. You still have to include `\usepackage[amsmath]`, though.

The “math environment” comes in two different forms:

- **Inline** mode, which will format the math within existing lines of text

Math Mode

The *amsmath* package is the backbone of using LaTeX to do math. As mentioned previously, it should already be installed. You still have to include `\usepackage[amsmath]`, though.

The "math environment" comes in two different forms:

- **Inline** mode, which will format the math within existing lines of text
- **Display** mode, which sets the math apart and centers it on the page

Math Mode

The *amsmath* package is the backbone of using LaTeX to do math. As mentioned previously, it should already be installed. You still have to include `\usepackage`[*amsmath*], though.

The “math environment” comes in two different forms:

- **Inline** mode, which will format the math within existing lines of text
- **Display** mode, which sets the math apart and centers it on the page

Let’s have a look at `jmasm_template.tex` and its corresponding output.

Entering Math Mode

The commands to enter the math environment are as follows:

Entering Math Mode

The commands to enter the math environment are as follows:

- **Inline**:
  - Using `\begin{math}` and `\end{math}`
  - Surround the math with `\( math stuff \)`
  - Surround the math with single dollar signs: $ *math stuff* $

Entering Math Mode

The commands to enter the math environment are as follows:

- **Inline**:
  - Using `\begin{math}` and `\end{math}`
  - Surround the math with `\( math stuff \)`
  - Surround the math with single dollar signs: `$` *math stuff* `$`
- **Display**:
  - Using `\begin{displaymath}` and `\end{displaymath}`
  - Surround the math with `\[ math stuff \]`
  - Surround the math with double dollar signs: `$$` *math stuff* `$$`

Entering Math Mode

The commands to enter the math environment are as follows:

- **Inline**:
  - Using `\begin{math}` and `\end{math}`
  - Surround the math with `\( math stuff \)`
  - Surround the math with single dollar signs: `$` *math stuff* `$`
- **Display**:
  - Using `\begin{displaymath}` and `\end{displaymath}`
  - Surround the math with `\[ math stuff \]`
  - Surround the math with double dollar signs: `$$` *math stuff* `$$`

Technically using $$ is frowned upon (it's TEX, not LATEX), but practically speaking it is fine.

Math Commands

Some common math symbols:

`\alpha`, `\beta`, etc.	α, β, etc.	`x^{2}, y_{1}, a^{b}_{c}`	x^2, y_1, a^b_c
`\Gamma`, `\Delta`, etc.	Γ, Δ, etc.	`\frac{x}{y}`	$\frac{x}{y}$
`\ldots`, `\cdots`	$\ldots, \cdots$	`\sqrt[x]{y}`	$\sqrt[x]{y}$
`\ddots`, `\vdots`	$\ddots, \vdots$	`\sum_{i=1}^{n}`	$\sum_{i=1}^n$
`\leq`, `\geq`	$\leq, \geq$	`\int_{0}^{\infty}`	$\int_0^\infty$
`\bigcap`, `\bigcup`	$\bigcap, \bigcup$	`\prod_{n=1}^{N}`	$\prod_{n=1}^N$
`\exists`, `\forall`	$\exists, \forall$	`\in`	$\in$
`\times`, `\div`, `\pm`	$\times, \div, \pm$	`\cdot`, `\bullet`, `\circ`	$\cdot, \bullet, \circ$
`\sin`, `\cot`, etc.	$\sin, \cot$, etc.	`\lim_{x \to \infty}`	$\lim_{x\to\infty}$

Math Commands

Some common math symbols:

`\alpha`, `\beta`, etc.	α, β, etc.	`x^{2}, y_{1}, a^{b}_{c}`	x^{2}, y_{1}, a^{b}_{c}
`\Gamma`, `\Delta`, etc.	Γ, Δ, etc.	`\frac{x}{y}`	$\frac{x}{y}$
`\ldots`, `\cdots`	$\ldots, \cdots$	`\sqrt[x]{y}`	$\sqrt[x]{y}$
`\ddots`, `\vdots`	$\ddots, \vdots$	`\sum_{i=1}^{n}`	$\sum_{i=1}^{n}$
`\leq`, `\geq`	$\leq, \geq$	`\int_{0}^{\infty}`	$\int_{0}^{\infty}$
`\bigcap`, `\bigcup`	$\bigcap, \bigcup$	`\prod_{n=1}^{N}`	$\prod_{n=1}^{N}$
`\exists`, `\forall`	$\exists, \forall$	`\in`	$\in$
`\times`, `\div`, `\pm`	$\times, \div, \pm$	`\cdot`, `\bullet`, `\circ`	$\cdot, \bullet, \circ$
`\sin`, `\cot`, etc.	$\sin, \cot$, etc.	`\lim_{x \to \infty}`	$\lim_{x\to\infty}$

You can combine all of these to make pretty much anything!

Math Commands

Some of those math commands will change in appearance depending on if you're in display or math mode. For example:

Here is a sum in inline mode: $\sum_{i=1}^{n} \frac{1}{a_n}$

Math Commands

Some of those math commands will change in appearance depending on if you're in display or math mode. For example:

Here is a sum in inline mode: $\sum_{i=1}^{n} \frac{1}{a_n}$

Here it is in display mode:

$$\sum_{i=1}^{n} \frac{1}{a_n}$$

Math Commands

Some of those math commands will change in appearance depending on if you're in display or math mode. For example:

Here is a sum in inline mode: $\sum_{i=1}^{n} \frac{1}{a_n}$

Here it is in display mode:

$$\sum_{i=1}^{n} \frac{1}{a_n}$$

A limit in inline mode: $\lim_{x \to \infty} f(x) = \sqrt[3]{a_0}$

Math Commands

Some of those math commands will change in appearance depending on if you're in display or math mode. For example:

Here is a sum in inline mode: $\sum_{i=1}^{n} \frac{1}{a_n}$

Here it is in display mode:

$$\sum_{i=1}^{n} \frac{1}{a_n}$$

A limit in inline mode: $\lim_{x \to \infty} f(x) = \sqrt[3]{a_0}$

And in display mode:

$$\lim_{x \to \infty} f(x) = \sqrt[3]{a_0}$$

Math Commands

A quick note, there is a package you can use to get slanted fractions:

```
\usepackage{xfrac}
```

With *xfrac*, you can use the `\sfrac` command to make slanted fractions:

`$ \sfrac{150}{29} $` ${}^{150}\!/_{29}$

`$$ \sfrac{150}{29} $$` ${}^{150}\!/_{29}$

It's useful for saving space, but I wouldn't try putting any math symbols that take up too much space in there.

Notes on Escaping Characters and Delimiters

Because some characters are used as part of commands in LaTeX, we need to be careful about using them both in and out of math mode.

Notes on Escaping Characters and Delimiters

Because some characters are used as part of commands in LaTeX, we need to be careful about using them both in and out of math mode.

In Text Mode:
Watch out for: `& % $ # _ { } ~ ^ \`.

We have `\textasciitilde` for ˜, `\textasciicircum` for ˆ, and `\textbackslash` for \.

For the others, you can just put a \ in front of them, e.g. `\&` will give you &.

Quotation marks are also done differently, with `‘‘text’’` giving you “text” and `‘text’` giving you ‘text’ (those are supposed to be grave accent marks).

Notes on Escaping Characters and Delimiters

Because some characters are used as part of commands in LaTeX, we need to be careful about using them both in and out of math mode.

In Text Mode:
Watch out for: `& % $ # _ { } ~ ^ \`.

We have `\textasciitilde` for ~, `\textasciicircum` for ^, and `\textbackslash` for \.

For the others, you can just put a \ in front of them, e.g. `\&` will give you &.

Quotation marks are also done differently, with ` ``text'' ` giving you “text” and `` `text' `` giving you ‘text’ (those are supposed to be grave accent marks).

In Math Mode:
Parenthesis ( ), brackets [ ], and absolute value $||$ can be done with the keyboard. You still need to be careful of braces $\{\}$ as above. You can get the norm $\|x\|$ with `\|` (i.e. `\|x\|`).

Your delimeters can scale, though!

Scaling Delimeters

Let's look at parenthesis in display mode:

```
$$ (\frac{x}{y})^{2} $$
```

Scaling Delimeters

Let's look at parenthesis in display mode:

```
$$ (\frac{x}{y})^{2} $$
```

$$(\frac{x}{y})^{2}$$

Scaling Delimeters

Let's look at parenthesis in display mode:

```
$$ (\frac{x}{y})^{2} $$
```

$$(\frac{x}{y})^{2}$$

Instead, we do the following:

```
$$ \left(\frac{x}{y}\right)^{2} $$
```

Scaling Delimeters

Let's look at parenthesis in display mode:

```
$$ (\frac{x}{y})^{2} $$
```

$$(\frac{x}{y})^{2}$$

Instead, we do the following:

```
$$ \left(\frac{x}{y}\right)^{2} $$
```

$$\left(\frac{x}{y}\right)^{2}$$

Scaling Delimeters

All the other rules apply, though, so if we want

$$\left\{ \frac{1}{a_n} \right\}_{n=1}^{\infty}$$

...

Scaling Delimeters

All the other rules apply, though, so if we want

$$\left\{\frac{1}{a_n}\right\}_{n=1}^{\infty}$$

...we have to use `\left\{` and `\right\}`, i.e.

```
$$ \left\{\frac{1}{a_n}\right\}_{n=1}^{\infty} $$
```

CC-BY 4.0

Other Display Math Environments

There are two other display math environments that may be of interest:

\begin{equation} and \end{equation} will automatically number the displayed math within the document:

Other Display Math Environments

There are two other display math environments that may be of interest:

`\begin{equation}` and `\end{equation}` will automatically number the displayed math within the document:

$$y = \frac{3}{2}x + \frac{2}{7} \tag{1}$$

Other Display Math Environments

There are two other display math environments that may be of interest:

\begin{*equation*} and \end{*equation*} will automatically number the displayed math within the document:

$$y = \frac{3}{2}x + \frac{2}{7} \tag{1}$$

While \begin{*align**} and \end{*align**} will align several equations using the & symbol:

Other Display Math Environments

There are two other display math environments that may be of interest:

`\begin{equation}` and `\end{equation}` will automatically number the displayed math within the document:

$$y = \frac{3}{2}x + \frac{2}{7} \tag{1}$$

While `\begin{align*}` and `\end{align*}` will align several equations using the & symbol:

```
f(x) &= 3x+2 \\
   x &\geq 0
```

Other Display Math Environments

There are two other display math environments that may be of interest:

`\begin{equation}` and `\end{equation}` will automatically number the displayed math within the document:

$$y = \frac{3}{2}x + \frac{2}{7} \tag{1}$$

While `\begin{align*}` and `\end{align*}` will align several equations using the & symbol:

```
f(x) &= 3x+2 \\
   x &\geq 0
```

$$\begin{aligned} f(x) &= 3x+2 \\ x &\geq 0 \end{aligned}$$

Other Display Math Environments

There are two other display math environments that may be of interest:

`\begin{equation}` and `\end{equation}` will automatically number the displayed math within the document:

$$y = \frac{3}{2}x + \frac{2}{7} \tag{1}$$

While `\begin{align*}` and `\end{align*}` will align several equations using the & symbol:

```
f(x) &= 3x+2 \\
   x &\geq 0
```

$$\begin{align*} f(x) &= 3x+2 \\ x &\geq 0 \end{align*}$$

The asterisks suppress numbering, and we could've used them on *equation*, too.

Tables

It is best to start out by just looking at an example of a table:

```
\begin{tabular}{|r|cl|}
(1,1) & (1,2) & (1,3) \\
\hline
(2,1) & (2,2) & (2,3)
\end{tabular}
```

(1,1)	(1,2)	(1,3)
(2,1)	(2,2)	(2,3)

Tables

It is best to start out by just looking at an example of a table:

```
\begin{tabular}{|r|cl|}
(1,1) & (1,2) & (1,3) \\
\hline
(2,1) & (2,2) & (2,3)
\end{tabular}
```

(1,1)	(1,2)	(1,3)
(2,1)	(2,2)	(2,3)

{r|cl|} tells LaTeX several things:

- The number of letters determines the number of columns (three)
- The letters themselves determine the text alignment in each column (r for right, c for center, l for left)
- Vertical lines (|) determine where vertical borders are inserted into the table

Ampersands (&) separate cells, `\hline` inserts a horizontal border, and \\ends a row.

Arrays

You can enter inline math mode inside individual table cells (display mode is not advised):

```
\begin{tabular}{|c|c|}
$\frac{1}{2}$ & $a \to \infty$ \\
Math and & Text \\
\end{tabular}
```

Arrays

You can enter inline math mode inside individual table cells (display mode is not advised):

```latex
\begin{tabular}{|c|c|}
$\frac{1}{2}$ & $a \to \infty$ \\
Math and & Text \\
\end{tabular}
```

$\frac{1}{2}$	$a \to \infty$
Math and	Text

Arrays

You can enter inline math mode inside individual table cells (display mode is not advised):

```
\begin{tabular}{|c|c|}
$\frac{1}{2}$ & $a \to \infty$ \\
Math and & Text \\
\end{tabular}
```

$\frac{1}{2}$	$a \to \infty$
Math and	Text

You can also include an entire table in math mode using the *array* environment instead of *tabular*. If you wanted to include plain text, you would just use the `\text{}` command.

```
$$ \begin{array}{|c|c|}
\frac{1}{2} & a \to \infty \\
\text{Math and} & \text{Text} \\
\end{array} $$
```

Arrays

You can enter inline math mode inside individual table cells (display mode is not advised):

```
\begin{tabular}{|c|c|}
$\frac{1}{2}$ & $a \to \infty$ \\
Math and & Text \\
\end{tabular}
```

$\frac{1}{2}$	$a \to \infty$
Math and	Text

You can also include an entire table in math mode using the *array* environment instead of *tabular*. If you wanted to include plain text, you would just use the \text{} command.

```
$$ \begin{array}{|c|c|}
\frac{1}{2} & a \to \infty \\
\text{Math and} & \text{Text} \\
\end{array} $$
```

$\frac{1}{2}$	$a \to \infty$
Math and	Text

Matrices and Vectors

Matrices and vectors work pretty similarly to Tables, but they have their own special environments. There are different environments depending on the delimeters you want around your matrix. The two most common are *pmatrix* (for parenthesis) and *bmatrix* (for square brackets). If you don't want any delimeters at all, you can just use *matrix*.

```
\begin{bmatrix}
1 & 2 & 3 \\
\frac{1}{2} & f(x) & \infty
\end{bmatrix}
```

Matrices and Vectors

Matrices and vectors work pretty similarly to Tables, but they have their own special environments. There are different environments depending on the delimeters you want around your matrix. The two most common are *pmatrix* (for parenthesis) and *bmatrix* (for square brackets). If you don't want any delimeters at all, you can just use *matrix*.

```
\begin{bmatrix}
1 & 2 & 3 \\
\frac{1}{2} & f(x) & \infty
\end{bmatrix}
```

$$\begin{bmatrix}
1 & 2 & 3 \\
\frac{1}{2} & f(x) & \infty
\end{bmatrix}$$

Matrices and Vectors

Matrices and vectors work pretty similarly to Tables, but they have their own special environments. There are different environments depending on the delimeters you want around your matrix. The two most common are *pmatrix* (for parenthesis) and *bmatrix* (for square brackets). If you don't want any delimeters at all, you can just use *matrix*.

```
\begin{bmatrix}
1 & 2 & 3 \\
\frac{1}{2} & f(x) & \infty
\end{bmatrix}
```

$$\begin{bmatrix} 1 & 2 & 3 \\ \frac{1}{2} & f(x) & \infty \end{bmatrix}$$

You *can* use matrices in inline math mode, but that doesn't mean you should. If it's in the middle $\mathbf{A} = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix}$ of a paragraph it's going to mess your spacing up a bit.

Lists

There are two list environments, *itemize* and *enumerate*.

Both use the same general structure:

```
\begin{enumerate}
\item Item one
\item Item two
\end{enumerate}
```

Lists

There are two list environments, `itemize` and `enumerate`.

Both use the same general structure:

```
\begin{enumerate}
\item Item one
\item Item two
\end{enumerate}
```

1. Item one
2. Item two

Lists

There are two list environments, *itemize* and *enumerate*.

Both use the same general structure:

```
\begin{enumerate}
\item Item one
\item Item two
\end{enumerate}
```

1. Item one
2. Item two

LaTeX will automatically make a sublist if you nest an *itemize* or *enumerate* environment within another list environment:

```
\begin{enumerate}
\item Item one
\begin{enumerate}
\item Subitem one
\item Subitem two
\end{enumerate}
\item Item two
\end{enumerate}
```

Lists

There are two list environments, *itemize* and *enumerate*.

Both use the same general structure:

```
\begin{enumerate}
\item Item one
\item Item two
\end{enumerate}
```

1. Item one
2. Item two

LaTeX will automatically make a sublist if you nest an *itemize* or *enumerate* environment within another list environment:

```
\begin{enumerate}
\item Item one
\begin{enumerate}
\item Subitem one
\item Subitem two
\end{enumerate}
\item Item two
\end{enumerate}
```

1. Item one

   1.1 Subitem one

   1.2 Subitem two

2. Item two

The graphicx package

If you want to include images in your document, *graphicx* is the way to do it. When exporting to a PDF using `pdflatex`, supported files are JPG, PNG, PDF, and EPS (using the *epstopdf* package).

The command to include an image using *graphicx* looks like this:

```
\includegraphics[attr1=val1, attr2=val2,...]{imagename}
```

The graphicx package

If you want to include images in your document, *graphicx* is the way to do it. When exporting to a PDF using `pdflatex`, supported files are JPG, PNG, PDF, and EPS (using the *epstopdf* package).

The command to include an image using *graphicx* looks like this:

```
\includegraphics[attr1=val1, attr2=val2,...]{imagename}
```

Examples of attributes you can define:

- width=5in, the preferred width
- height=3cm, the preferred height
- scale=0.5, a multiplicative scalar for the size

The graphicx package

If you want to include images in your document, *graphicx* is the way to do it. When exporting to a PDF using `pdflatex`, supported files are JPG, PNG, PDF, and EPS (using the *epstopdf* package).

The command to include an image using *graphicx* looks like this:

```
\includegraphics[attr1=val1, attr2=val2,...]{imagename}
```

Examples of attributes you can define:

- width=5in, the preferred width
- height=3cm, the preferred height
- scale=0.5, a multiplicative scalar for the size

By default, LaTeX just looks for images in whichever folder your `tex` file is in, but you can specify a specific path in the preamble for LaTeX to use when looking for images:

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

Online LaTeX Editors

There are a few online options for working with LaTeX:

- ShareLaTeX.com allows you to create and compile LaTeX documents online if you sign up for a free account. Collaborating with others requires a paid account, however.

Online LaTeX Editors

There are a few online options for working with LaTeX:

- ShareLaTeX.com allows you to create and compile LaTeX documents online if you sign up for a free account. Collaborating with others requires a paid account, however.
- Overleaf.com allows you to create and compile LaTeX documents all online, and you can save them if you set up a (free) account. More importantly, it allows you to collaborate with other using just your free account

Online LaTeX Editors

There are a few online options for working with LaTeX:

- ShareLaTeX.com allows you to create and compile LaTeX documents online if you sign up for a free account. Collaborating with others requires a paid account, however.
- Overleaf.com allows you to create and compile LaTeX documents all online, and you can save them if you set up a (free) account. More importantly, it allows you to collaborate with other using just your free account

Navigate to http://ow.ly/ZgKH6 (and let's see how many of you it'll let on there at once).

Online LaTeX Editors

There are a few online options for working with LaTeX:

- ShareLaTeX.com allows you to create and compile LaTeX documents online if you sign up for a free account. Collaborating with others requires a paid account, however.
- Overleaf.com allows you to create and compile LaTeX documents all online, and you can save them if you set up a (free) account. More importantly, it allows you to collaborate with other using just your free account

Navigate to http://ow.ly/ZgKH6 (and let's see how many of you it'll let on there at once).

This also seems like a great time to talk about commenting!

Further Information

Where should you go for further information?

Further Information

Where should you go for further information?

- The LaTeX research guide: http://guides.lib.wayne.edu/latex

Further Information

Where should you go for further information?

- The LaTeX research guide: http://guides.lib.wayne.edu/latex
- The TeX/LaTeX Stack Exchange: http://tex.stackexchange.com/

Further Information

Where should you go for further information?

- The LaTeX research guide: http://guides.lib.wayne.edu/latex
- The TeX/LaTeX Stack Exchange: http://tex.stackexchange.com/
- The Comprehensive TeX Archive Network (CTAN): https://www.ctan.org/

Further Information

Where should you go for further information?

- The LaTeX research guide: http://guides.lib.wayne.edu/latex
- The TeX/LaTeX Stack Exchange: http://tex.stackexchange.com/
- The Comprehensive TeX Archive Network (CTAN): https://www.ctan.org/
- Google! Yes, really.