LaTeX Basics

Dylan Yu

January 8, 2022

Table of Contents

1 Introduction

- What is LaTeX?
- Why learn LaTeX?
- Math Equations

What is LaTeX?

LaTeX (pronounced *LAY-tek* or *LAH-tek*) is a tool used to create professional-looking documents.

What is LaTeX?

LaTeX (pronounced *LAY-tek* or *LAH-tek*) is a tool used to create professional-looking documents. It is based on the WYSIWYM (what you see is what you mean) idea, meaning you only have focus on the contents of your document and the computer will take care of the formatting.

Why learn LaTeX?

1. It's useful

Why learn LaTeX?

1. It's useful
    - There's a high amount of customizability

Why learn LaTeX?

1. It's useful
   - There's a high amount of customizability
   - It's extremely stable (so your document won't crash and burn the next time you compile it, unless you broke your computer or something)

Why learn LaTeX?

1. It's useful
   - There's a high amount of customizability
   - It's extremely stable (so your document won't crash and burn the next time you compile it, unless you broke your computer or something)
   - You can collaborate with others via Overleaf, Github, or Dropbox (the later two are preferred by serious users)

Why learn LaTeX?

1. It's useful
   - There's a high amount of customizability
   - It's extremely stable (so your document won't crash and burn the next time you compile it, unless you broke your computer or something)
   - You can collaborate with others via Overleaf, Github, or Dropbox (the later two are preferred by serious users)
2. It's aesthetically pleasing

Why learn LaTeX?

1. It's useful
   - There's a high amount of customizability
   - It's extremely stable (so your document won't crash and burn the next time you compile it, unless you broke your computer or something)
   - You can collaborate with others via Overleaf, Github, or Dropbox (the later two are preferred by serious users)
2. It's aesthetically pleasing
   - In fact, it's the easiest language that also produces beautiful documents

Why learn LaTeX?

1. It's useful
   - There's a high amount of customizability
   - It's extremely stable (so your document won't crash and burn the next time you compile it, unless you broke your computer or something)
   - You can collaborate with others via Overleaf, Github, or Dropbox (the later two are preferred by serious users)
2. It's aesthetically pleasing
   - In fact, it's the easiest language that also produces beautiful documents
   - For example, you could use LaTeX for Google Docs, but it looks bad and you won't get the same functionality

Why learn LaTeX?

1. It's useful
   - There's a high amount of customizability
   - It's extremely stable (so your document won't crash and burn the next time you compile it, unless you broke your computer or something)
   - You can collaborate with others via Overleaf, Github, or Dropbox (the later two are preferred by serious users)
2. It's aesthetically pleasing
   - In fact, it's the easiest language that also produces beautiful documents
   - For example, you could use LaTeX for Google Docs, but it looks bad and you won't get the same functionality
   - You'll see examples of nicely presented articles later

Why learn LaTeX?

Note: a valid reason to learn LaTeX is to make your documents look better, but don't rewrite everything you've ever made in LaTeX; that's pointless.

Why learn LaTeX?

Note: a valid reason to learn LaTeX is to make your documents look better, but don't rewrite everything you've ever made in LaTeX; that's pointless. I also don't think you should learn it (or at least anything beyond the basics) if you're solely typing math equations.

Math Equations

- In-line math mode: use $.

```
$\sqrt{x}=5$
```

$\sqrt{x} = 5$

Math Equations

- In-line math mode: use $.

```
$\sqrt{x}=5$
```

$\sqrt{x} = 5$

- Display math mode: use $$ or \[\] (the latter is *strongly* preferred).

```
\[\sqrt{x}=5\]
```

$$\sqrt{x} = 5$$

(The difference is that the equation is centered here.)

Math Equations

- In-line math mode: use $.

```
$\sqrt{x}=5$
```

$\sqrt{x} = 5$

- Display math mode: use $$ or \[\] (the latter is *strongly* preferred).

```
\[\sqrt{x}=5\]
```

$$\sqrt{x} = 5$$

(The difference is that the equation is centered here.)

There are reasons to use one over the other: for smaller equations, there's no need to use display math mode, but larger equations should be centered.

Math Expressions

- Addition and subtraction stay the same.

Math Expressions

- Addition and subtraction stay the same.
- Do NOT use $*$ and $/$ for multiplication and division! Instead, do the following:

`$\times$, $\cdot$,`
`$\div$`
	$\times$, $\cdot$, $\div$

Math Expressions

- Addition and subtraction stay the same.
- Do NOT use $*$ and / for multiplication and division! Instead, do the following:

`$\times$, $\cdot$, $\div$`	$\times$, $\cdot$, $\div$

- Fractions:

`$\frac{x+y}{2}$`	$\frac{x+y}{2}$

Math Expressions

- Roots:

`$\sqrt{2}$`	$\sqrt{2}$

Math Expressions

- Roots:

`$\sqrt{2}$`	$\sqrt{2}$

- Superscripts and subscripts:

`$a_k \cdot x^k$`	$a_k \cdot x^k$

Math Expressions

- Roots:

`$\sqrt{2}$`	$\sqrt{2}$

- Superscripts and subscripts:

`$a_k \cdot x^k$`	$a_k \cdot x^k$

Remember to use **curly braces**:

`$2^2021$ v.s.`
`$2^{2021}$`
	$2^2021$ v.s. $2^{2021}$

Math Expressions

- If curly braces are used for grouping, how do we display one?

```
$\{ \}$
```

$\{ \}$

Note that spaces don't matter in math mode.

Math Expressions

- If curly braces are used for grouping, how do we display one?

```
$\{ \}$
```

$\{\}$

Note that spaces don't matter in math mode.

- Dots:

```
$\cdots$, $\ldots$,
    $\vdots$
```

$\cdots$, $\ldots$, $\vdots$

Math Expressions

- If curly braces are used for grouping, how do we display one?

```
$\{ \}$
```

$\{\}$

Note that spaces don't matter in math mode.

- Dots:

```
$\cdots$, $\ldots$,
    $\vdots$
```

$\cdots, \ldots, \vdots$

- Sums and products:

```
$\sum_{i=0}^n a_i +
    \prod_{j=1}^k b_j$
```

$\sum_{i=0}^n a_i + \prod_{j=1}^k b_j$

Math Expressions

- Inequalities:

```
$>$, $<$, $\ge$, $\le$
```

$>, <, \ge, \le$

Math Expressions

- Inequalities:

`$>$, $<$, $\ge$, $\le$`	$>, <, \ge, \le$

- Text inside math mode:

`$\text{It is } 6`
`    \text{ p.m.}$`
	It is 6 p.m.

Math Expressions

- Inequalities:

`$>$, $<$, $\ge$, $\le$`	$>, <, \ge, \le$

- Text inside math mode:

`$\text{It is } 6`
`    \text{ p.m.}$`
	It is 6 p.m.

- I won't talk about aligning and numbering equations; look into that by yourself.

Table of Contents

2 Starting a basic document

- Preamble
- Title page

Preamble

There aren't that many necessary parts to a very basic LaTeX document.

```
\documentclass{article}
\begin{document}
The solution to \[\sqrt{x} = 5\] is \[x=25.\]
\end{document}
```

The part before `\begin{document}` is called the preamble. We usually put packages (similar to importing things in Java) there, among other things that we'll discuss later.

Title page

Pretty self explanatory:

```
\documentclass{article}
\title{An example document} % The title
\author{John Doe} % Author’s name
\date{\today} % The date; you can choose a specific date
\begin{document}
\maketitle % Creates the title

The solution to \[\sqrt{x} = 5\] is \[x=25.\]
\end{document}
```

Table of Contents

Bold, italics, and underline

- Bold:

`\textbf{Bold}`	**Bold**

Bold, italics, and underline

- Bold:

`\textbf{Bold}`	**Bold**

- Italics:

`\textit{Italics}`	*Italics*

Bold, italics, and underline

- Bold:

`\textbf{Bold}`	**Bold**

- Italics:

`\textit{Italics}`	*Italics*

- Underline:

`\underline{Underline}`	Underline

Bold, italics, and underline

- Bold:

`\textbf{Bold}`	**Bold**

- Italics:

`\textit{Italics}`	*Italics*

- Underline:

`\underline{Underline}`	Underline

For emphasis, it is not recommended you use the above; instead, redefine the command `\emph` (which is usually italics) to your liking. I usually use `\vocab` for highlights, but `\alert` is also a good name.

Comments

You might've noticed the percent symbol before; these are called comments. Some languages (like Java) use // for comments. LaTeX uses %.

```
% The solution to
    \[\sqrt{x} = 5\] is
    \[x=25.\]
```

There's no display because it's a comment.

Images

Inserting images is easy:

```
\includegraphics{images/latex-thonk.png}
```

Images

Inserting images is easy:

```
\includegraphics{images/latex-thonk.png}
```

Centering them, adjusting them, etc. are a bit harder, but still manageable:

Figure: Thonk

Lists

With numbers:

1. One

Without numbers:

- Dot

Sections

In math articles, we usually like to outline our topic. They work as "headers."

```
\section{A section}
Divide your topic into smaller parts.
\subsection{A subsection}
Even smaller.
\subsubsection{A subsubsection}
\textit{Even smaller.}
\paragraph{A paragraph} Gives a title to a paragraph.
```

You can use `\tableofcontents` to display the table of contents (duh).

Tables and figures

Take a look at the example below; don't worry too much about the [H] for now.

```
\begin{figure}[H]
\centering
\begin{tabular}{c|c}
\textbf{Kanye} &
    \textbf{Drake}\\
\hline
MBDTF & IYRTITL\\
CD & TML
\end{tabular}
\caption{A table inside
    a figure
    environment.}
\end{figure}
```

Kanye	**Drake**
MBDTF	IYRTITL
CD	TML

Figure: A table inside a figure environment.

Table of Contents

4 Commands

- \newcommand
- \newenvironment

\newcommand

Defining new commands with `\newcommand` is like creating a method in Java. You give it some name, and insert a few parameters, and output something.

```
\newcommand{\introduce}[1]{Hi my name is #1.}
```

\newcommand

Defining new commands with `\newcommand` is like creating a method in Java. You give it some name, and insert a few parameters, and output something.

```
\newcommand{\introduce}[1]{Hi my name is #1.}
```

The first part, `\introduce`, is the name of the command. The second part, `[1]` is the number of parameters. Unlike with many other languages, there's no need for data types. The third part is what the command should output, where the first parameter (if there is one) is denoted with #1, the second (if there is one) is denoted with #2, etc.

\newcommand

Defining new commands with `\newcommand` is like creating a method in Java. You give it some name, and insert a few parameters, and output something.

```
\newcommand{\introduce}[1]{Hi my name is #1.}
```

The first part, `\introduce`, is the name of the command. The second part, `[1]` is the number of parameters. Unlike with many other languages, there's no need for data types. The third part is what the command should output, where the first parameter (if there is one) is denoted with #1, the second (if there is one) is denoted with #2, etc. Running the above command, we get

`\introduce{Dylan}`	Hi my name is Dylan.

\newenvironment

We use \newenvironment when we have something larger (e.g. with paragraphs) as a parameter.

```
\newenvironment{boldquote}{My quote is:\newline}
{\newline\textit{--- Me.}}
```

\newenvironment

We use \newenvironment when we have something larger (e.g. with paragraphs) as a parameter.

```
\newenvironment{boldquote}{My quote is:\newline}
{\newline\textit{--- Me.}}
```

The first part, boldquote, is the name of the environment. The second part, My quote is:\newline, is what appears before the text you will type. The third part, \newline\textit{--- Me.}, is what appears after the text you will type. You can think of this like a *sandwich*: first is the \begin stuff, then it's your text, and finally it's the \end stuff.

\newenvironment

We use `\newenvironment` when we have something larger (e.g. with paragraphs) as a parameter.

```
\newenvironment{boldquote}{My quote is:\newline}
{\newline\textit{--- Me.}}
```

The first part, `boldquote`, is the name of the environment. The second part, `My quote is:\newline`, is what appears before the text you will type. The third part, `\newline\textit{--- Me.}`, is what appears after the text you will type. You can think of this like a *sandwich*: first is the `\begin` stuff, then it's your text, and finally it's the `\end` stuff. Running the above environment, we get

```
\begin{boldquote}
I am cool.
\end{boldquote}
```

My quote is:
I am cool.
— *Me.*

Table of Contents

Asymptote

As you'd expect, code-generated figures are generated by. . . code.

Asymptote

As you'd expect, code-generated figures are generated by. . . code. Here's a figure by Evan Chen:

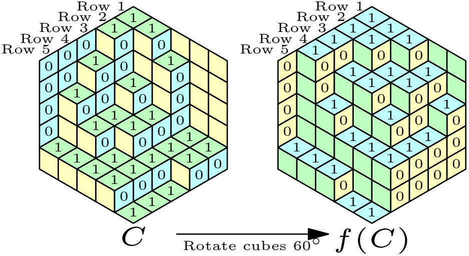


Figure: USA TST 2013/3

Asymptote

As you'd expect, code-generated figures are generated by. . . code. Here's a figure by Evan Chen:

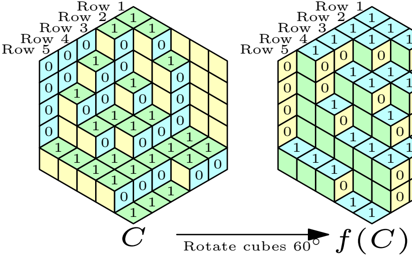


Figure: USA TST 2013/3

A lot of people who do olympiad math use Asymptote (because it is compatible with AoPS).

Ti*kz*

Ti*kz* and Asymptote can do the same things (probably) but I prefer Ti*kz* because it is more stable (at least on Overleaf).

Ti*kz*

Ti*kz* and Asymptote can do the same things (probably) but I prefer Ti*kz* because it is more stable (at least on Overleaf). Here's an example figure using Ti*kz*:

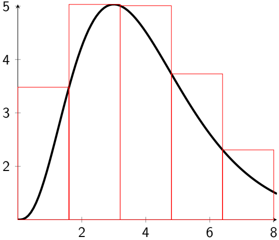


Figure: The right Riemann sum of $3e^{-x}x^3 + 1$.

Table of Contents

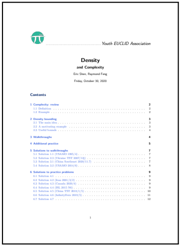
6 Beauty
- Handouts
- Style Files
- More fun

Handouts

There are many beautiful handouts that can be made with LaTeX.

Handouts

There are many beautiful handouts that can be made with LaTeX.



Youth EUCLID Association

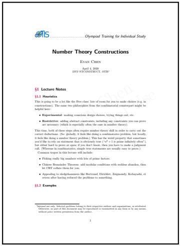
Density

and Complexity

Eric Shen, Raymond Feng

Friday, October 30, 2020

Contents



Olympiad Training for Individual Study

Number Theory Constructions

Evan Chen

April 4, 2020
DNY-NTCONSTRUCT, OTIS[*]

§1 Lecture Notes

§1.1 Heuristics

This is going to be a lot like the Free class: lots of room for you to make choices (e.g. in constructions). The same two philosophies from the combinatorial counterpart might be helpful here:

- **Experimental**: making conscious design choices, trying things out, etc.
- **Restrictive**: adding abstract constraints, including any constraints you can prove are necessary (which is especially often the case in number theory).

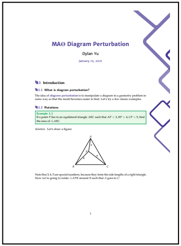
This time, both of these steps often require number theory skill in order to carry out the correct deductions. (So: globally, it feels like doing a combinatorics problem, but locally, it feels like doing a number theory problem.) This has the weird property that sometimes you'd like to rely on statement that is obviously true ("n^2+1 is prime infinitely often"), but either hard to prove or open; if you don't know, then you have to make a judgment call. (Whereas in combinatorics, simple true statements are usually easy to prove.)

Common tropes in this lecture will include:

- Picking really big numbers with lots of prime factors.
- Chinese Remainder Theorem: add modular conditions with reckless abandon, then let CRT collate them for you.
- Appealing to sledgehammers like Bertrand, Dirichlet, Zsigmondy, Kobayashi, et cetera after having reduced the problems to something.

§1.2 Examples

[*] Internal use only. Selected problems belong to their respective authors and organizations, as attributed. Otherwise, no part of this document may be reproduced or transmitted in any form or by any means, without prior written permission from the author.



MAΘ Diagram Perturbation

Dylan Yu

January 10, 2021

1 Introduction

1.1 What is diagram perturbation?

The idea of *diagram perturbation* is to manipulate a diagram in a geometry problem in some way so that the result becomes easier to find. Let's try a *few* classic examples.

1.2 Rotations

Example 1.1
If a point P lies in an equilateral triangle ABC such that $AP = 3$, $BP = 4$, $CP = 5$, find the area of $\triangle ABC$.

Solution. Let's draw a figure:

Note that 3, 4, 5 are special numbers, because they form the side lengths of a right triangle. Now we're going to rotate $\triangle APB$ around B such that A goes to C:

Figure: The first handout is by Raymond Feng and Eric Shen; the second handout is by Evan Chen; the third is by me.

Style files

There are many pretty style files, which are basically just custom-made designs that act like a "template" for your article. Some nice ones are evan.sty by Evan Chen (he doesn't have an example document so this is just one of his handouts),

Style files

There are many pretty style files, which are basically just custom-made designs that act like a "template" for your article. Some nice ones are evan.sty by Evan Chen (he doesn't have an example document so this is just one of his handouts), lucky.sty by Dennis Chen, and

Style files

There are many pretty style files, which are basically just custom-made designs that act like a “template” for your article. Some nice ones are evan.sty by Evan Chen (he doesn’t have an example document so this is just one of his handouts), lucky.sty by Dennis Chen, and dylanadi.sty by me (sorry for the self-advertisement).

Style files

There are many pretty style files, which are basically just custom-made designs that act like a “template” for your article. Some nice ones are evan.sty by Evan Chen (he doesn’t have an example document so this is just one of his handouts), lucky.sty by Dennis Chen, and dylanadi.sty by me (sorry for the self-advertisement).

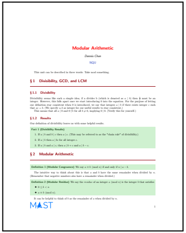
Oh and for fun take a look at my dark mode.

More fun

Just some random assortment of things that I think look nice:

More fun

Just some random assortment of things that I think look nice:



Modular Arithmetic

Dennis Chen

NQU

This unit can be described in three words: Take mod something.

§ 1 Divisibility, GCD, and LCM

§ 1.1 Divisibility

Divisibility seems like such a simple idea; if a divides b (which is denoted as $a \mid b$) then $\frac{b}{a}$ must be an integer. However, this falls apart once we start introducing 0 into the equation. For the purpose of letting our definition stay consistent when 0 is introduced, we say that integers $a \mid b$ if there exists integer c such that $ac = b$. (We specify a, b as integer for our useful results to stay consistent.)

This means that all $a \mid 0$ and $0 \nmid b$ for all $b \neq 0$, implying $0 \mid 0$. (Verify this for yourself.)

§ 1.2 Results

Our definition of divisibility leaves us with some helpful results.

Fact 1 (Divisibility Results)

1. If $a \mid b$ and $b \mid c$ then $a \mid c$. (This may be referred to as the "chain rule" of divisibility.)
2. If $a \mid b$ then $a \mid bc$ for all integer c.
3. If $a \mid b$ and $a \mid c$, then $a \mid b + c$ and $a \mid b - c$.

§ 2 Modular Arithmetic

Definition 1 (Modular Congruence) We say $a \equiv b \pmod{n}$ if and only if $n \mid a - b$.

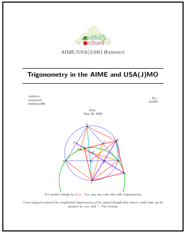
The intuitive way to think about this is that a and b have the same remainder when divided by n. (Remember that negative numbers also have a remainder when divided.)

Definition 2 (Modular Residue) We say the *residue* of an integer $a \pmod{n}$ is the integer b that satisfies

- $0 \leq b < n$
- $a \equiv b \pmod{n}$.

It can be helpful to think of b as the remainder of a when divided by n.

MAST 1



AIME/USA(J)MO Handout

Trigonometry in the AIME and USA(J)MO

Authors:
naman12
freeman66

For:
AoPS

Date:
May 26, 2020

Yet another beauty by Evan. Yes, you can solve this with trigonometry.

"I was trying to unravel the complicated trigonometry of the radical thought that silence could make up the greatest lie ever told." - Pat Conroy

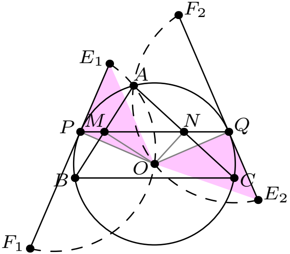


Figure: From left to right: a handout by Dennis Chen, a handout by Amol Rama and I, and a secret spiral similarity in USA TSTST 2018/5.

More fun

There aren't picture for these, since they are more technical:

More fun

There aren't picture for these, since they are more technical:

- VON by Evan Chen, an olympiad problem database written in Python, but prints out documents in LaTeX

More fun

There aren't picture for these, since they are more technical:

- VON by Evan Chen, an olympiad problem database written in Python, but prints out documents in LaTeX
- Napkin by Evan Chen, an open source math textbook written in LaTeX

More fun

There aren't picture for these, since they are more technical:

- VON by Evan Chen, an olympiad problem database written in Python, but prints out documents in LaTeX
- Napkin by Evan Chen, an open source math textbook written in LaTeX
- Dynamic Programming in Computing Contests by Arpan Banerjee, a book on dynamic programming written in LaTeX

More fun

There aren't picture for these, since they are more technical:

- VON by Evan Chen, an olympiad problem database written in Python, but prints out documents in LaTeX
- Napkin by Evan Chen, an open source math textbook written in LaTeX
- Dynamic Programming in Computing Contests by Arpan Banerjee, a book on dynamic programming written in LaTeX
- An Introduction to USACO by Darren Yao, a beginner's guide to USACO written in LaTeX; the cover was designed by me (in LaTeX, of course)

More fun

There aren't picture for these, since they are more technical:

- VON by Evan Chen, an olympiad problem database written in Python, but prints out documents in LaTeX
- Napkin by Evan Chen, an open source math textbook written in LaTeX
- Dynamic Programming in Computing Contests by Arpan Banerjee, a book on dynamic programming written in LaTeX
- An Introduction to USACO by Darren Yao, a beginner's guide to USACO written in LaTeX; the cover was designed by me (in LaTeX, of course)

As you can tell, Evan Chen loves LaTeX.

Table of Contents

7 Parting words
- Google
- Further Reading

Google

Google is your best friend when it comes to learning LaTeX.

Google

Google is your best friend when it comes to learning LaTeX. If you are confused by something, Google it, and if you can't figure out what exactly you're trying to ask, join a LaTeX Discord server/TeX StackExchange (I prefer the former but the latter will yield better results) and ask the question there.

Further Reading

1. Learn LaTeX in 30 minutes; these slides were heavily based on this article

Further Reading

1. Learn LaTeX in 30 minutes; these slides were heavily based on this article
2. A Beginner's Guide to LaTeX

Further Reading

1. Learn LaTeX in 30 minutes; these slides were heavily based on this article
2. A Beginner's Guide to LaTeX
3. An Example LaTeX Document; you can also take a look at the sources of Evan Chen's handouts for more examples

Further Reading

1. Learn LaTeX in 30 minutes; these slides were heavily based on this article
2. A Beginner's Guide to LaTeX
3. An Example LaTeX Document; you can also take a look at the sources of Evan Chen's handouts for more examples
4. Notes on Programming in TeX; I haven't read through the whole thing, but it should be pretty comprehensive for your everyday needs

Further Reading

A list of other stuff to learn (that I can remember):

- using ‘ ‘ and ’ ’ for quotes; you should **never** use " (double quotes) for quotes!

Further Reading

A list of other stuff to learn (that I can remember):

- using ‘ ‘ and ’ ’ for quotes; you should **never** use " (double quotes) for quotes!
- aligning/numbering equations

Further Reading

A list of other stuff to learn (that I can remember):

- using ‘ ‘ and ’ ’ for quotes; you should **never** use " (double quotes) for quotes!
- aligning/numbering equations
- lists with different types of bullets and numbering (e.g. with letters, with roman numerals, etc.)

Further Reading

A list of other stuff to learn (that I can remember):

- using ‘ ‘ and ’ ’ for quotes; you should **never** use " (double quotes) for quotes!
- aligning/numbering equations
- lists with different types of bullets and numbering (e.g. with letters, with roman numerals, etc.)
- using [H] v.s. [h] v.s. [!ht] v.s. etc. for figures

Further Reading

A list of other stuff to learn (that I can remember):

- using ‘ ‘ and ’ ’ for quotes; you should **never** use " (double quotes) for quotes!
- aligning/numbering equations
- lists with different types of bullets and numbering (e.g. with letters, with roman numerals, etc.)
- using [H] v.s. [h] v.s. [!ht] v.s. etc. for figures
- cross-referencing

Further Reading

A list of other stuff to learn (that I can remember):

- using ‘ ‘ and ’ ’ for quotes; you should **never** use " (double quotes) for quotes!
- aligning/numbering equations
- lists with different types of bullets and numbering (e.g. with letters, with roman numerals, etc.)
- using [H] v.s. [h] v.s. [!ht] v.s. etc. for figures
- cross-referencing
- bibliographies

Further Reading

A list of other stuff to learn (that I can remember):

- using ‘ ‘ and ’ ’ for quotes; you should **never** use `"` (double quotes) for quotes!
- aligning/numbering equations
- lists with different types of bullets and numbering (e.g. with letters, with roman numerals, etc.)
- using [H] v.s. [h] v.s. [!ht] v.s. etc. for figures
- cross-referencing
- bibliographies
- optional parameters for `\newcommand` and `\newenvironment`

Further Reading

A list of other stuff to learn (that I can remember):

- other math mode commands

Further Reading

A list of other stuff to learn (that I can remember):

- other math mode commands
- mathematical symbols

Further Reading

A list of other stuff to learn (that I can remember):

- other math mode commands
- mathematical symbols
- drawing figures (I only skimmed over it)

Further Reading

A list of other stuff to learn (that I can remember):

- other math mode commands
- mathematical symbols
- drawing figures (I only skimmed over it)
- presentations (in Beamer, like this one!)

Further Reading

A list of other stuff to learn (that I can remember):

- other math mode commands
- mathematical symbols
- drawing figures (I only skimmed over it)
- presentations (in Beamer, like this one!)
- various document classes (e.g. `article`, `scrartcl`, `book`, `beamer`, etc.)

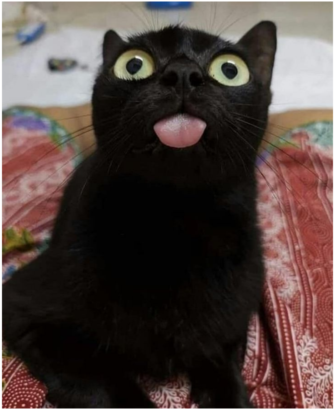
Further Reading

A list of other stuff to learn (that I can remember):

- other math mode commands
- mathematical symbols
- drawing figures (I only skimmed over it)
- presentations (in Beamer, like this one!)
- various document classes (e.g. `article`, `scrartcl`, `book`, `beamer`, etc.)
- converting to a local setup (download TEX Live, then some source code editor; please read these FAQ's on Evan's website)

Cat pic

idk



Thanks to Dennis Chen for proofreading these slides.