SIAM LaTeX Boot Camp

Kylan Schatz

Department of Mathematics
North Carolina State University

September 1, 2022

What is LaTeX?

Definition

LaTeX is a software system [1], used for typesetting.

What is LaTeX?

Definition

LaTeX is a software system [1], used for typesetting.

- LaTeX ships with its own editor and compiler.
- Many people opt to use an editor of their choice.

What is LaTeX?

Definition

LaTeX is a software system [1], used for typesetting.

- LaTeX ships with its own editor and compiler.
- Many people opt to use an editor of their choice.
- We suggest participants use Overleaf: https://www.overleaf.com/

Syntax Basics

LaTeX uses a syntax similar to Markdown.

Syntax Basics

LaTeX uses a syntax similar to Markdown.

- LaTeX commands begin with a backslash,
  - Parameters in curly braces
  - Options within square braces

Syntax Basics

LaTeX uses a syntax similar to Markdown.

- LaTeX commands begin with a backslash,
  - Parameters in curly braces
  - Options within square braces
- Enter (in-line) math-mode with a dollar sign,

Syntax Basics

LaTeX uses a syntax similar to Markdown.

- LaTeX commands begin with a backslash,
  - Parameters in curly braces
  - Options within square braces
- Enter (in-line) math-mode with a dollar sign,
- Line comments with a percent sign,

Syntax Basics

LaTeX uses a syntax similar to Markdown.

- LaTeX commands begin with a backslash,
  - Parameters in curly braces
  - Options within square braces
- Enter (in-line) math-mode with a dollar sign,
- Line comments with a percent sign,
- Unspecified is plaintext.

Syntax Example

The code snippet:

```
% Binomial coefficient
For $k \leq n \in \mathbb{N}$,
$$\binom{n}{k} = \frac{n!}{k!(n-k)!}.$$
```

Syntax Example

The code snippet:

```
1 % Binomial coefficient
2 For $k \leq n \in \mathbb{N}$,
3 $$\binom{n}{k} = \frac{n!}{k!(n-k)!}.$$
```

Produces the following output:

For $k \leq n \in \mathbb{N}$,

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}.$$

Document Structure

Every document must start with `\documentclass[options]{class}`.

Document Structure

Every document must start with `\documentclass[options]{class}`.

- Common choices of `class` are `article`, `beamer`, or `exam`.
- Each `class` has its own flavor of commands.

Document Structure

Every document must start with `\documentclass[options]{class}`.

- Common choices of `class` are `article`, `beamer`, or `exam`.
- Each `class` has its own flavor of commands.
- Common choices of `option` are `12pt`, `letterpaper`, `twocolumn`, `twoside`, or `landscape`.

Document Structure

Next, import packages with `\usepackage[options]{package}`.

Document Structure

Next, import packages with `\usepackage[options]{package}`.

- Common choices of `package` are `amsmath`, `tikz`, `graphicx`, `geometry`.

Document Structure

Next, import packages with `\usepackage[options]{package}`.

- Common choices of `package` are `amsmath`, `tikz`, `graphicx`, `geometry`.
- Can package your own commands in `.sty`-files.

Document Structure

Next, import packages with `\usepackage[options]{package}`.

- Common choices of `package` are `amsmath`, `tikz`, `graphicx`, `geometry`.
- Can package your own commands in `.sty`-files.

If desired, can specify the document header.

- Change details with `\title{title}`, `\author{author}`, and `\date{date}`.

Document Structure

All content must be contained within `\begin{document}...\end{document}`.

Document Structure

All content must be contained within `\begin{document}...\end{document}`.

- Place document header with `\maketitle`.

Document Structure

All content must be contained within `\begin{document}...\end{document}`.

- Place document header with `\maketitle`.
- Create sections with `\part{part}`, `\section{section}`, etc.

Document Structure

All content must be contained within `\begin{document}...\end{document}`.

- Place document header with `\maketitle`.
- Create sections with `\part{part}`, `\section{section}`, etc.
- Use environments with `\begin{env}...\end{env}`.

Document Structure

All content must be contained within `\begin{document}...\end{document}`.

- Place document header with `\maketitle`.
- Create sections with `\part{part}`, `\section{section}`, etc.
- Use environments with `\begin{env}...\end{env}`.
- Use *math mode* for math and special commands.

Document Structure

LaTeX has some reserved characters which must be escaped.

Document Structure

LaTeX has some reserved characters which must be escaped.

- Must escape \ (`\textbackslash`), {, $, &, and %.

Document Structure

LaTeX has some reserved characters which must be escaped.

- Must escape \ (`\textbackslash`), {, $, &, and %.
- Can insert space with `\,` or `\!`.
- Larger space with `\vspace*{space}`, `\hspace*{space}`.
- Line breaks with \\ or `\newline`.

Math Mode

To write math equations, one must use *math mode*.

Math Mode

To write math equations, one must use *math mode*.

- Write in-line math with `$...$`, `\(...\)`.
- Write display math with `$$...$$`, `\[...\]`.

Math Mode

To write math equations, one must use *math mode*.

- Write in-line math with `$...$`, `\(...\)`.
- Write display math with `$$...$$`, `\[...\]`.
- Can also use the `align` or `equation` environments.

Math Mode

Many popular commands have common sense names.

Math Mode

Many popular commands have common sense names.

- Plain text in math mode with `\text{text}`.
- Fractions with `\frac{num}{dem}`.

Math Mode

Many popular commands have common sense names.

- Plain text in math mode with `\text{text}`.
- Fractions with `\frac{num}{dem}`.
- Super and subscripts with `_{sub}` and `^{sup}`.

Math Mode

Many popular commands have common sense names.

- Plain text in math mode with `\text{text}`.
- Fractions with `\frac{num}{dem}`.
- Super and subscripts with `_{sub}` and `^{sup}`.

Most commands have different in-line / display behavior.

- Can force display behaviour with `\displaystyle`.

Math Mode Example

The code snippet:

```
\begin{align}
    & y_0 = y(0), \\
    & y_{n+1} = \Delta x \cdot y'(x_n) + y_n.
\end{align}
```

Math Mode Example

The code snippet:

```
\begin{align}
    & y_0 = y(0), \\
    & y_{n+1} = \Delta x \cdot y'(x_n) + y_n.
\end{align}
```

produces the following output:

$$y_0 = y(0), \tag{1}$$
$$y_{n+1} = \Delta x \cdot y'(x_n) + y_n. \tag{2}$$

Environments

Some special environments help with document structure.

Environments

Some special environments help with document structure.

- Use `align` and `equation` for multi-line display math.

Environments

Some special environments help with document structure.

- Use `align` and `equation` for multi-line display math.
- Use `itemize` and `enumerate` for lists.

Environments

Some special environments help with document structure.

- Use `align` and `equation` for multi-line display math.
- Use `itemize` and `enumerate` for lists.
- Use `tabular` and `array` for tables or matrices.
- Use `figure` for graphics.

Environments

Some special environments help with document structure.

- Use `align` and `equation` for multi-line display math.
- Use `itemize` and `enumerate` for lists.
- Use `tabular` and `array` for tables or matrices.
- Use `figure` for graphics.

Each environment has special commands and control characters.

Environments Example

The code snippet:

```
\begin{itemize}
    \item[1.] Step 1,
    \item[a.] Step 2,
    \item Step 3.
\end{itemize}
```

Environments Example

The code snippet:

```
\begin{itemize}
    \item[1.] Step 1,
    \item[a.] Step 2,
    \item Step 3.
\end{itemize}
```

creates a list with three items whose labels are ”1.”, ”a.”, and ”○”.

Graphics

To use graphics, use the `graphicx` package.

Graphics

To use graphics, use the `graphicx` package.

- Insert images with `\includegraphics[options]{path}`
- Specify graphics path with `\graphicspath{{path}}`

Graphics

To use graphics, use the `graphicx` package.

- Insert images with `\includegraphics[options]{path}`
- Specify graphics path with `\graphicspath{{path}}`
- Use the `figure` environment for positioning, captions, and tags.

Graphics Example

The code snippet:

```
\begin{figure}[h]
    \centering
    \includegraphics[width=0.4\textwidth]{my-pic}
    \caption{My beautiful selfie.}
    \label{me}
\end{figure}
```

Graphics Example

The code snippet:

```
\begin{figure}[h]
    \centering
    \includegraphics[width=0.4\textwidth]{my-pic}
    \caption{My beautiful selfie.}
    \label{me}
\end{figure}
```

places the picture at path `my-pic` *here*, centered, scaled to 0.4-times text width, and adds the caption "My beautiful selfie."
Additionally, we may reference the photo with `\ref{me}`.

References and Citations

In LaTeX, one can make inter-document references automatically.

References and Citations

In LaTeX, one can make inter-document references automatically.

- Create a reference point with `\label{name}`.
- Refer back to that point using `\ref{name}`.

References and Citations

In LaTeX, one can make inter-document references automatically.

- Create a reference point with `\label{name}`.
- Refer back to that point using `\ref{name}`.
- Rename a label with `\tag{name}`

References and Citations

In LaTeX, one can make inter-document references automatically.

- Create a reference point with `\label{name}`.
- Refer back to that point using `\ref{name}`.
- Rename a label with `\tag{name}`
- Use `\cite{name}` for bibliography items.

If you import the `href` package, references are clickable.

References and Citations

One way to create a bibliography is with `bibtex`.

References and Citations

One way to create a bibliography is with `bibtex`.

- Bib. items are stored in a `.bib`-file.
- Create an item with `@type{name,...}`.

References and Citations

One way to create a bibliography is with `bibtex`.

- Bib. items are stored in a `.bib`-file.
- Create an item with `@type{name,...}`.
- Cite bib. items with `\cite{name}`.

References and Citations

One way to create a bibliography is with `bibtex`.

- Bib. items are stored in a `.bib`-file.
- Create an item with `@type{name,...}`.
- Cite bib. items with `\cite{name}`.
- Make references page from `bibtex` file with `\bibliography{path}`.

Reference Example

The code snippet:

```
\begin{equation}\label{SODE} \tag{$\Delta$}
    my'' + by' + ky = f(t)
\end{equation}
```

Reference Example

The code snippet:

```
\begin{equation}\label{SODE} \tag{$\Delta$}
    my'' + by' + ky = f(t)
\end{equation}
```

creates the following output:

$$my'' + by' + ky = f(t), \tag{Δ}$$

which we may reference with \eqref{SODE}, resulting in (Δ).

Bibliography Example

The code snippet (in, say, `biblio.bib`):

```
@book{DF,
    author={David S. Dummit, Richard M. Foote},
    title={Abstract Algebra}
}
```

Bibliography Example

The code snippet (in, say, `biblio.bib`):

```
@book{DF,
    author={David S. Dummit, Richard M. Foote},
    title={Abstract Algebra}
}
```

creates a bib. entry which we may cite with `\cite{DF}`.
Calling `\bibliography{biblio}` creates the bibliography page, with all *used* citations.

Theorems

LaTeX has many tools to typeset theorems.

Theorems

LaTeX has many tools to typeset theorems.

- To access, must use the package `amsthm`.
- Built in are the `theorem` and `proof` environments.

Theorems

LaTeX has many tools to typeset theorems.

- To access, must use the package **amsthm**.
- Built in are the **theorem** and **proof** environments.
- Create a theorem with
  `\begin{env}[Name]...\end{env}`

Theorems

LaTeX has many tools to typeset theorems.

- To access, must use the package **amsthm**.
- Built in are the `theorem` and `proof` environments.
- Create a theorem with `\begin{env}[Name]...\end{env}`
- Make your own theorem type with `\newtheorem{env}{name}`
- Specify theorem style with `\theoremstyle{style}`

Theorem Example

The code snippet:

```
1 \begin{theorem}[FT Finite Abelian Groups]
    \label{FTFAG}
2    Let $G$ finite Abelian. Then, $G \cong
       \bigoplus_{i=1}^k \mathbb{Z} / n_i\mathbb{Z}$.
3 \end{theorem}
```

Theorem Example

The code snippet:

```
\begin{theorem}[FT Finite Abelian Groups]
    \label{FTFAG}
    Let $G$ finite Abelian. Then, $G \cong
        \bigoplus_{i=1}^k \mathbb{Z} / n_i\mathbb{Z}$.
\end{theorem}
```

results in the following named theorem:

Theorem (FT Finite Abelian Groups)

Let G finite Abelian. Then, $G \cong \bigoplus_{i=1}^{k} \mathbb{Z}/n_i\mathbb{Z}$,

which we may reference with `\ref{FTFAG}`.

Fancy Text

LaTeX provides many options to stylize text.

Fancy Text

LaTeX provides many options to stylize text.

- Include the `amsfonts` package for more tools.
- Built in are `\mathbb`, `\mathcal`, `\mathrm`, `\mathfrak`.

Fancy Text

LaTeX provides many options to stylize text.

- Include the `amsfonts` package for more tools.
- Built in are `\mathbb`, `\mathcal`, `\mathrm`, `\mathfrak`.
- Italics and boldface with `\textit`, `\textbf`

Fancy Text

LaTeX provides many options to stylize text.

- Include the `amsfonts` package for more tools.
- Built in are `\mathbb`, `\mathcal`, `\mathrm`, `\mathfrak`.
- Italics and boldface with `\textit`, `\textbf`
- Add diacritics with `\tilde`, `\hat`, `\overline`, `\dot`, etc.

Macros

LaTeX has powerful tools for creating your own macros and commands.

Macros

LaTeX has powerful tools for creating your own macros and commands.

- Create a command with `\newcommand{\name}[nargs][arg1]{...}`
  - `nargs` is the number of arguments the command takes
  - Specify default first argument with `arg1`
  - Reference arguments with `#narg`

Macros

LaTeX has powerful tools for creating your own macros and commands.

- Create a command with `\newcommand{\name}[nargs][arg1]{...}`
  - `nargs` is the number of arguments the command takes
  - Specify default first argument with `arg1`
  - Reference arguments with `#narg`
- Overwrite existing command with `\renewcommand`

Macros

LaTeX has powerful tools for creating your own macros and commands.

- Create a command with `\newcommand{\name}[nargs][arg1]{...}`
  - `nargs` is the number of arguments the command takes
  - Specify default first argument with `arg1`
  - Reference arguments with `#narg`
- Overwrite existing command with `\renewcommand`
- Use `\declaremathoperator` for *text* commands with no arguments

Macros Example

If we declare the following commands at the start of our document:

```
1 \DeclareMathOperator{\spn}{span}
2 \newcommand{\s}[1]{\left \{ #1 \right \}}
```

Macros Example

If we declare the following commands at the start of our document:

```
\DeclareMathOperator{\spn}{span}
\newcommand{\s}[1]{\left \{ #1 \right \}}
```

we may use them later in our document:

```
$$\mathfrak{sl}(2) = \spn\s{h, e, f}, \dots$$
```

Macros Example

If we declare the following commands at the start of our document:

```
\DeclareMathOperator{\spn}{span}
\newcommand{\s}[1]{\left \{ #1 \right \}}
```

we may use them later in our document:

```
$$\mathfrak{sl}(2) = \spn\s{h, e, f}, \dots$$
```

to produce the following output:

$$\mathfrak{sl}(2) = \operatorname{span}\{h, e, f\}, \dots$$

Closing

Questions?

Closing

Questions?

To continue your LaTeX journey...

Bibliography

- The LaTeX project.
  https://www.latex-project.org.
- Overleaf documentation.
  https://www.overleaf.com/learn.
- Several LaTeX project templates.
  https://www.overleaf.com/read/qdfmzmmmpwvq.
- This beamer presentation.
  https://www.overleaf.com/read/rxkjkqhhtmcz.