Math 291: Lecture 7

Dr. Fagerstrom

Minnesota State University Moorhead
web.mnstate.edu/fagerstrom
fagerstrom@mnstate.edu

March 1, 2018

Table of Contents

Outline

Theorem-Like Environments

- LaTeX has several pre-defined that allow us to quickly typeset a variety of structures, without having to manually set the typeface, numbering, and other aspects ourselves.

Theorem-Like Environments

- LaTeX has several pre-defined that allow us to quickly typeset a variety of structures, without having to manually set the typeface, numbering, and other aspects ourselves.
- Recall: Environments start with the command:
  \begin{environment name}
  and end with the command:
  \end{environment name}.

Theorem-Like Environments

- LaTeX has several pre-defined that allow us to quickly typeset a variety of structures, without having to manually set the typeface, numbering, and other aspects ourselves.
- Recall: Environments start with the command:
  \begin{environment name}
  and end with the command:
  \end{environment name}.
- LaTeX also allows us to create environments using certain commands.

Theorem-Like Environments

- One such command in LaTeX is:
  `\newtheorem{Env name}{Title}[subctr]`

Theorem-Like Environments

- One such command in LaTeX is:
  `\newtheorem{Env name}{Title}[subctr]`
- In this command, 'Env name' is the name used to call the environment.

Theorem-Like Environments

- One such command in LaTeX is:
  `\newtheorem{Env name}{Title}[subctr]`
- In this command, 'Env name' is the name used to call the environment.
- 'Title' is the title or name that is actually printed (along with with a "counter") when the document is compiled.

Theorem-Like Environments

- One such command in LaTeX is:
  `\newtheorem{Env name}{Title}[subctr]`
- In this command, 'Env name' is the name used to call the environment.
- 'Title' is the title or name that is actually printed (along with with a "counter") when the document is compiled.
- 'subctr' is the counter of some other environment or structural element.
  - If omitted, numbering is maintained throughout the document.
  - If present, it must be a standard LaTeX counter, and will create numbers such as '4.1', where 4 is the last number of the standard counter and 1 is the first use of this environment since the 4 was updated. Ex: Theorem 4.1 in section 4 of your document.

Theorem Example

- Open a new document containing your standard preamble.

Theorem Example

- Open a new document containing your standard preamble.
- In the preamble of your document, type:
  `\newtheorem{thm}{Theorem}`

Theorem Example

- Open a new document containing your standard preamble.
- In the preamble of your document, type: `\newtheorem{thm}{Theorem}`
- Then, type in a similar command to define the "axiom" environment: `\newtheorem{ax}{Axiom}`

Theorem Example

An Example:

- Next, type in the following and then build:

```
\begin{thm}[The Fundamental
Theorem of Calculus]
$$\int_{a}^{b} f(x)\, dx = F(b)-F(a)$$
\end{thm}
```

Theorem Example

An Example:

- Next, type in the following and then build:

```
\begin{thm}[The Fundamental
Theorem of Calculus]
$$\int_{a}^{b} f(x)\, dx = F(b)-F(a)$$
\end{thm}
```

- Now type in commands and build to produce the following:

Axiom 1.

All dogs chase postal workers.

Axiom 2.

All postal workers deliver mail.

Theorems, continued

- Notice that LaTeX keeps track of the numbering for you.

Theorems, continued

- Notice that LaTeX keeps track of the numbering for you.
- When you add or remove theorems, the numbering throughout the document is automatically updated.

Theorems, continued

- Notice that LaTeX keeps track of the numbering for you.
- When you add or remove theorems, the numbering throughout the document is automatically updated.
- Note that when you call a new environment, you can give it an extra title on a one-by-one basis, as for the FTC in the exercise.

Theorems, continued

- Notice that LaTeX keeps track of the numbering for you.
- When you add or remove theorems, the numbering throughout the document is automatically updated.
- Note that when you call a new environment, you can give it an extra title on a one-by-one basis, as for the FTC in the exercise.
- It is possible to refer to the numbers, and those reference labels will be automatically updated as well.

Exercise

- Define a new environment "thm2".

Exercise

- Define a new environment "thm2".
- Then try using this new environment within an enumerated list to create the following:

Exercise

- Define a new environment "thm2".
- Then try using this new environment within an enumerated list to create the following:
  1. This is the first enumerated item.
  2. This is the second enumerated item.
  3. This is the third enumerated item.

Theorem 3.1.

This is the first numbered theorem after item three.

Theorem 3.2.

This is the second numbered theorem after item three.

4. Blah, blah

Theorem 4.1.

This is yet another theorem statement - How did it get numbered?.

Theorems, continued

- Note that we referred to a specific enumeration level in our environment definition.
- Other counters can be used, like sections and chapters in larger documents.

Theorems, continued

- There is another version of this that can create sub-environments that continue the numbering of the main environment.

Theorems, continued

- There is another version of this that can create sub-environments that continue the numbering of the main environment.
- The syntax is:
  `\newtheorem{Subenv name}[main env]{Sub-Title}`

Theorems, continued

- There is another version of this that can create sub-environments that continue the numbering of the main environment.
- The syntax is:
  `\newtheorem{Subenv name}[main env]{Sub-Title}`
- With this, you can create a sequence of theorems such as: Thm 1, Thm 2, Cor 3, Cor 4, Thm 5, Thm 6, ...

Theorems, continued

- There is another version of this that can create sub-environments that continue the numbering of the main environment.
- The syntax is:
  `\newtheorem{Subenv name}[main env]{Sub-Title}`
- With this, you can create a sequence of theorems such as: Thm 1, Thm 2, Cor 3, Cor 4, Thm 5, Thm 6, ...
- There are more options with the amsthm package, described in a bit.

Outline

Package: amsthm

If you don't want to take the time to define environments yourself, you can use the `amsthm` package.

Package: amsthm

If you don't want to take the time to define environments yourself, you can use the `amsthm` package.

- Along with the standard environments, this package defines a `newtheorem*` version, used for unnumbered theorems

Package: amsthm

If you don't want to take the time to define environments yourself, you can use the `amsthm` package.

- Along with the standard environments, this package defines a `newtheorem*` version, used for unnumbered theorems
- It also defines three environment styles:
  - plain (bold title, then italics in the body)
  - definition (bold title, then normal text in the body)
  - remark (italicized title, then normal text in the body)

Package: amsthm

If you don't want to take the time to define environments yourself, you can use the `amsthm` package.

- Along with the standard environments, this package defines a `newtheorem*` version, used for unnumbered theorems
- It also defines three environment styles:
  - plain (bold title, then italics in the body)
  - definition (bold title, then normal text in the body)
  - remark (italicized title, then normal text in the body)
- You activate a style with the \theoremstyle{style} command. It then remains that style until you re-define the style.
- You can also create your own styles with a \newtheoremstyle command.

Package: amsthm

If you don't want to take the time to define environments yourself, you can use the `amsthm` package.

- Along with the standard environments, this package defines a `newtheorem*` version, used for unnumbered theorems
- It also defines three environment styles:
  - plain (bold title, then italics in the body)
  - definition (bold title, then normal text in the body)
  - remark (italicized title, then normal text in the body)
- You activate a style with the \theoremstyle{style} command. It then remains that style until you re-define the style.
- You can also create your own styles with a \newtheoremstyle command.
- You can still manually define other theorem environments as if you didn't load the amsthm package.

Package: amsthm, continued

- The "amsthm" package also defines the `\swapnumbers` command (in preamble before any `\newtheorem` commands), which puts the numbers *before* the theorem (as in: **1 Theorem**).

Package: amsthm, continued

- The "amsthm" package also defines the `\swapnumbers` command (in preamble before any `\newtheorem` commands), which puts the numbers *before* the theorem (as in: **1 Theorem**).
- Finally, it defines a `proof` environment (`\begin{proof} \end{proof}`). This environment:
  - is unnumbered
  - it starts with *Proof*
  - it ends with: □.

Outline

Manually Defining Your Own Commands:

- The syntax for defining a command is:
  \newcommand{\name}[#args][opt]{def}

Manually Defining Your Own Commands:

- The syntax for defining a command is:
  \newcommand{\name}[#args][opt]{def}

  Note: LaTeX will not allow you to redefine a command that has already been defined internally.

Manually Defining Your Own Commands:

- The syntax for defining a command is:
  \newcommand{\name}[#args][opt]{def}

  Note: LaTeX will not allow you to redefine a command that has already been defined internally.
- **Example:** \newcommand{\di}{\displaystyle}

Manually Defining Your Own Commands:

- The syntax for defining a command is:
  \newcommand{\name}[#args][opt]{def}

  Note: LaTeX will not allow you to redefine a command that has already been defined internally.
- **Example:** \newcommand{\di}{\displaystyle}
- When placed in the preamble, the command is globally defined (applies to the entire document).
- When placed within an environment, it is defined only within that environment.
- When placed elsewhere in the body of a document, it can only be used from then on.

New Commands with Arguments

- The "#args" part of the newcommand syntax indicates the number of arguments that are required to be supplied when using the command (each argument should be put within a separate "{}").

New Commands with Arguments

- The "#args" part of the newcommand syntax indicates the number of arguments that are required to be supplied when using the command (each argument should be put within a separate "{}").
- Each argument will be referred to separately in the definition of the command by using: #1, #2, etc.

New Commands with Arguments

- The "#args" part of the newcommand syntax indicates the number of arguments that are required to be supplied when using the command (each argument should be put within a separate "{}").
- Each argument will be referred to separately in the definition of the command by using: #1, #2, etc.
- The command `\ensuremath` ensures the command will **always** be carried out in math mode (whether you call the command inside $ signs or not).

Examples of Commands with Arguments

- Type the following into your document:

```
\newcommand{\repdec}[1]
{\ensuremath{0.\overline{#1}=\frac{#1}{99}}}
```

Examples of Commands with Arguments

- Type the following into your document:

```
\newcommand{\repdec}[1]
{\ensuremath{0.\overline{#1}=\frac{#1}{99}}}
```

- Then call your new command by entering:

```
$\repdec{43}$
```

Examples of Commands with Arguments

- Type the following into your document:

```
\newcommand{\repdec}[1]
{\ensuremath{0.\overline{#1}=\frac{#1}{99}}}
```

- Then call your new command by entering:
  `$\repdec{43}$`
  What happens when you build? Notice that this is a command with a single argument (input).

Examples of Commands with Arguments

- Type the following into your document:

```
\newcommand{\repdec}[1]
{\ensuremath{0.\overline{#1}=\frac{#1}{99}}}
```

- Then call your new command by entering:
  `$\repdec{43}$`
  What happens when you build? Notice that this is a command with a single argument (input).
  Try changing the input value and see what happens to the output.

Examples of Commands with Arguments

- As another example, here is a command requiring 4 inputs:

Examples of Commands with Arguments

- As another example, here is a command requiring 4 inputs:

```
\newcommand{\cfrac}[4]
{\ensuremath{\frac{\frac{#1}{#2}}
{\frac{#3}{#4}}}}
```

Examples of Commands with Arguments

- As another example, here is a command requiring 4 inputs:

```
\newcommand{\cfrac}[4]
{\ensuremath{\frac{\frac{#1}{#2}}
{\frac{#3}{#4}}}}
```

- Add this command definition to your sample document.
- Then, test out your new `\cfrac` command using some different input values. What does this command do?

Example of a command with an optional argument

- The `\newcommand` also allows you to define commands with one *optional* argument (an argument that is available for use but not absolutely required).

Example of a command with an optional argument

- The `\newcommand` also allows you to define commands with one *optional* argument (an argument that is available for use but not absolutely required).
- For example, try adding the following to your sample document:

```
\newcommand{\subvec}[3][x]
{\ensuremath{#1_{#2}, \ldots, #1_{#3}}}
```

Example of a command with an optional argument

- The `\newcommand` also allows you to define commands with one *optional* argument (an argument that is available for use but not absolutely required).
- For example, try adding the following to your sample document:

```
\newcommand{\subvec}[3][x]
{\ensuremath{#1_{#2}, \ldots, #1_{#3}}}
```

- The first of the three arguments is optional, and a default value has been supplied.
- If a new value for this optional argument is **not** supplied, the default value of x will be used. Otherwise, the new input value will be used.
- Remember, the optional argument, if used, is in square brackets.

Example of a command with an optional argument

- Try calling this command three times using the following inputs:
  `\subvec[A]{1}{n}`
  `\subvec[y]{1}{n}`
  `\subvec{1}{n}`

The Renewcommand Command

- The `\renewcommand` command allows us to redefine or alter an existing command.

The Renewcommand Command

- The \renewcommand command allows us to redefine or alter an existing command.
- Here is a command that Dr. Fagerstrom uses when she runs out of alphabet on her review sheets:

```
\setcounter{enumi}{0}
\renewcommand{\labelenumi}{(\alph{enumi}\alph{enumi})}
```

The Renewcommand Command

- Try using this command to create the following enumeration:
  (a) First
  (b) Second
  (aa) Third
  (bb) Fourth

The Renewcommand Command

- Try using this command to create the following enumeration:
  (a) First
  (b) Second
  (aa) Third
  (bb) Fourth
- **Be careful** when using `renewcommand`. You can use it to accidentally overwrite standard LaTeX commands!

Outline

New Environments

- Finally, the following syntax can be used to define a new LaTeX environment:

New Environments

- Finally, the following syntax can be used to define a new LaTeX environment:

```
\newenvironment{envname}[narg][opt]{begdef}{enddef}
```

New Environments

- Finally, the following syntax can be used to define a new LaTeX environment:

  ```
  \newenvironment{envname}[narg][opt]{begdef}{enddef}
  ```

- 'begdef' is the stuff that is printed when the environment is opened, and 'enddef' is the stuff that is printed when the environment is closed.
- We can also define and make use of new counters using the command `newcounters`.
- These options are part of what makes LaTeX highly customizable and useful.