LaTeX Packages and New Commands

W. Ethan Duckworth, Loyola University Maryland, 2014

- Packages add more capabilities to LaTeX. Load them in the preamble (i.e. between `\documentclass{article}` and `\begin{document}`).

  Example 1. Input:

```
\documentclass{article}
\usepackage{amsmath}
\begin{document}
The following command is defined in amsmath
$$
\begin{bmatrix}
a & b \\
c & d
\end{bmatrix}
$$
\end{document}
```

  Output:

  The following command is defined in amsmath

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

- Here are a handful of the more common packages.
  - `amsmath`: AMS (American Mathematical Society) Math. Adds commands for various common constructions, including aligning equations and making matrices.
  - `amsfonts`: AMS Fonts. Defines the fonts for things like Blackboard Bold and Fraktur.
  - `amsthm`: AMS Theorem. Makes it easy for the user to create and control their own environments like `theorem`, `example`, `lemma`, `definition`, etc.
  - `amssymb`: AMS Symbols. Defines more symbols.
  - `amsrefs`: AMS References. Defines commands for easily entering bibliographic data.
  - `fancyhdr`: Fancy Header. Defines commands for the user to create and control complicated headers and footers.
  - `geometry`: (Page) Geometry. Allows the user to easily set the margins, text area, distances for headers and footers, etc.
  - `graphics`: Graphics. Defines commands for including (external) graphics into a LaTeX file.
  - `setspace`: Set (Line) Spacing. Allows the user to make the document double-spaced or one-and-a-half spaced.
  - `tikz`: Tikz. Allows the user to create simple pictures, within a LaTeX document, with programming-type commands.
- If you define a new command, you should usually do this in the preamble.

  Example 2. Input:

```
\documentclass{article}
\newcommand{\dydx}{\frac{dy}{dx}}
\begin{document}
If $y=x^2$ then $\dydx = 2x$.
\end{document}
```

  Output:

  If $y = x^2$ then $\dfrac{dy}{dx} = 2x$.

  In this example, `\dydx` is the name of the new function being defined. Then `{\frac{dy}{dx}}` is what the function is defined as.

- Commands with inputs, i.e. arguments.

  Example 3. Input:

```
\newcommand{\dd}[2]{\frac{d#1}{d#2}}
If $V=\pi r^2$ then $\dd Vr = 2\pi r$.
```

  Output:

  If $V = \pi r^2$ then $\dfrac{dV}{dr} = 2\pi r$.

  Here "`[2]`" represents the number of inputs that the function will have. These inputs are then represented by `#1` and `#2` in the definition.

- Changing commands. You can change commands that already exist. For instance

```
\renewcommand{\labelenumi}{\Roman{enumi}.}
```

  will change an enumerate list to be in upper case Roman numerals.

- Fancier constructions can be useful too.

  Example 4. Input:

```
\newcommand{\map}[4]{%
  \begin{array}[t]{rcl}
    #1 & \longrightarrow & #2\\
    {#3} & \mapstochar \longrightarrow & #4
\end{array}}
$f:\map{A}{B}{x}{x^2}$
```

  Output:

$$f: \begin{array}[t]{rcl} A & \longrightarrow & B \\ x & \longmapsto & x^2 \end{array}$$

- Environments. There is an analogue of `\newcommand` for environments too:

```
\newenvironment{foo}[n]{bar}{baz}
```

  defines a new environment called `foo`. The number of arguments is n. The definition is broken into two parts: `bar` is what happens when you enter `\begin{foo}` and `baz` is what happens when you enter `\end{foo}`.