An Introductory Guide to LaTeX

Marcus Fedarko

May 10, 2016

1 Introduction

From a 2009 survey of 130 academic journals—ten journals for each of thirteen scientific fields of study—biologists François Brischoux and Pierre Legagneux estimate that 96.9 percent of mathematics papers, 74 percent of physics papers, and 45.8 percent of computer science papers are produced using the typesetting software LaTeX. Brischoux and Legagneux note that the use of LaTeX, particularly with document formatting templates provided by academic journals, avoids the problem of manually reformatting one's submission to fit multiple journal standards: saving a significant amount of time for the tool's users.

LaTeX[1] is a popular typesetting tool that uses code to produce documents. LaTeX can be used to create many kinds of documents that look professional, are inherently portable, and are highly editable: from academic papers to books to resumés.

This code-based paradigm differs from the what-you-see-is-what-you-get (WYSIWYG) approach taken by many other word processing tools like Microsoft Word and OpenOffice Writer in a few ways. One significant consequence of LaTeX's nature as a programmatic tool is the initial difficulty in getting started with the software—and that, hopefully, is where this guide comes in. This document will contain a brief, clear introduction to the tool and will discuss LaTeX compilation, document creation, and a number of useful features in LaTeX.

A brief note: this guide presupposes some basic knowledge of how to use a computer, including saving and opening files, downloading and installing software, and using your device's command line or terminal.

2 Compilation

LaTeX code is just plain text. The act of converting this code to an output file is called *compilation*—if you've used a compiled programming language like Java or C before, this will be familiar. Many compilers exist for LaTeX: your choice in compiler will probably be based on what format you want your output to be in. Here are a few output type options:

- **DVI files**: Device Independent file format (DVI) files were designed to be the output format of TeX, the system upon which LaTeX is based. DVI files are generally used as

[1] Generally pronounced *Lah-tech*, but many different pronunciations exist. (Which pronunciation you end up using doesn't really matter, thankfully.)

an intermediate file type: that is, people tend to use DVI files as a stepping stone for converting to other types of files (e.g. PDF or PS).

Running the command `latex` from your system's terminal or command line, by default, will produce DVI files.

- **PDF files**: Adobe Portable Document Format (PDF) files are a popular document format, used for a variety of purposes. You've probably used or seen a PDF before.

  `pdfLaTeX` is a common compiler that produces PDF files directly from LaTeX code, avoiding the use of DVI files at all. DVI files can also be converted to PDF using DVI-to-PDF tools like `dvipdf`.[2]

- **PS files**: Postscript (PS) files were initially designed as an input language for laser printers. Over time, PS has evolved to be a well-known language for creating vector graphics; PS was actually an influence on the development of PDF.

  PS files can be produced from DVI files using DVI-to-PS conversion tools like `dvips`.

Regardless of your choice of output type and compiler, the general process of working in LaTeX as discussed in this document should remain relatively consistent. This is one of the upsides of using raw text-based document design: the source code of a document is inherently portable.

However, you do have a choice in how you edit LaTeX code. As is discussed in section 7.3, there are a number of specialized LaTeX editors that can help make the compilation and output process more efficient for you. This guide only deals with using a simple plain-text editor on your computer to create and edit LaTeX code files—however, if you'd like to consult and choose from some of the available LaTeX editors before following the rest of this guide, you can certainly do that.[3]

3 Our First Document

In this section, we'll create—in four lines—a simple, working LaTeX document. By the end of this section, you should be familiar with the basic yet crucial parts of LaTeX document design.

Without further ado, here's the source code for such a document:

```
\documentclass[12pt]{article}
\begin{document}
    Hi, \LaTeX!
\end{document}
```

[2] The exact tool you use to produce PDF or PS files will depend on your operating system. As will be mentioned soon, I recommend using a specialized LaTeX editor that takes care of the compilation and output process for you instead of manually producing LaTeX output from the terminal or command line.

[3] You should probably note that, since many specialized editors will take care of compilation for you, the use of such an editor removes the need for some of the compilation steps explicitly listed in the early sections of guide.

This is rendered as:[4]

```
Hi, LaTeX!
```

3.1 Line-by-line explanation

Looking at each line in order (using the line numbers to the left side of the source code)—

1. Here we declare the document class of the document as an *article*, passing in an optional parameter indicating the document body font size as 12 points.[5]. We do this by using the `\documentclass` command. LaTeX commands generally follow this format: a backslash precedes the command name, with *optional arguments* following the command name in square brackets and *required arguments* in curly braces.

   The first line of a LaTeX document usually involves declaring the *document class* of the file. Many different document classes exist, all of which fulfill different purposes. Furthermore, LaTeX users can create and use their own document classes as they see fit. The article document class is very commonly used and is an excellent starting class, so we'll only discuss article-class documents for most of this guide; for further reading on different document classes, see the LaTeX wikibook.[6] For a brief discussion of user-defined classes, see section 5.4.1.

2. This line signifies the start of the *document environment*, also referred to as the document body. An *environment* in LaTeX is simply a section of the document that is formatted or otherwise treated a specified way during compilation. We use the `\begin` command here to initialize the document environment: everything between the `\begin` and `\end` commands denoting an environment's range is considered part of that environment.

3. As a part of the document environment, this line contains the text that will be displayed in the output file. The `\LaTeX` command, as used in this line, is just used to display the LaTeX symbol in a fancy way—this trend of using commands to refer to similarly named special symbols continues throughout much of LaTeX.[7]

4. This line signifies the end of the document body, using the `\end` command to end the document environment that was begun by the corresponding `\begin` command.[8]

[4] As a side note: LaTeX automatically indents the first line in a paragraph. If you'd like to avoid this behavior, you can just prefix the start of a paragraph with the `\noindent` command.

[5] The default LaTeX font size is 10 points, indicated by the option `10pt`.

[6] Available at `https://en.wikibooks.org/wiki/LaTeX/Document_Structure#Document_classes`

[7] We'll see more examples of this paradigm in our later discussion in section 4.2 of math mode.

[8] We'll see other examples of environments when we discuss mathematical alignment and lists.

3.2 Writing and compiling the document

You can (and should) try generating this for yourself!

First, you should make sure LaTeX is installed: the tool may be bundled with your operating system, or you may have to download the software by following the instructions at `https://latex-project.org/ftp.html`.

After verifying LaTeX is installed, open up your favorite plaintext editor and create and save a document containing just the above four lines of code—you can name the file any valid filename you'd like, but for clarity the conventional filename suffix for LaTeX source files is `.tex`.

Next, go to the terminal (or command line, depending on your operating system) and run the command `latex name.tex`, where `name.tex` is the filename of whatever you named the file just now.[9]

This should produce a DVI file. You can now convert the DVI file into either PDF (using the command `dvipdf name.dvi`, depending on) or PS (using the command `dvips name.dvi`).

Congratulations! You just created your first LaTeX document!

3.3 Where to go from here

Now that you've created a first document, the hardest part of getting started with LaTeX is over with. Take some time to mess around with the document source code—maybe try using `pdftex` to generate PDF files from the source code without having to generate a DVI file as an intermediate step, or try modifying the text and/or formatting of the document to get a sense for how text editing works in LaTeX.

4 Math Mode

One of the biggest advantages of working in LaTeX is the tool's capability to display mathematical equations. Using, say, Microsoft Word to display equations involves a lot of clicking, a lot of fine-tuning formatting, and a lot of unnecessary hassle. However, once you understand the basics of using *math mode* in LaTeX, formatting equations becomes very simple. Let's take a look at the example below:

```latex
\documentclass[12pt]{article}
\begin{document}
    Look! Here's the sum over $3x + 8$ from $x = 1$ to $10$.
    \[
    \sum_{x = 1}^{10} 3x + 8
    \]
    And here's the sum in inline mode: $\sum_{x = 1}^{10} 3x + 8$.
\end{document} % same as last time
```

[9] This assumes you're running `latex` within the same directory the file is in—if not, you should first move, or `cd`, into the directory containing the source code file.

This is rendered as:

Look! Here's the sum over $3x + 8$ from $x = 1$ to 10.

$$\sum_{x=1}^{10} 3x + 8$$

And here's the sum in inline mode: $\sum_{x=1}^{10} 3x + 8$.

4.1 Line-by-line explanation

Again, looking at each line in order—

1. Same as before, this line declares the document class and font size.

2. Same as before, this line starts the document body.

3. The characters contained within the dollar signs are contained within what we call *inline math mode*—that is, mathematical objects that are designed to be used within text. Mathematical objects in inline math mode are generally somewhat scaled down to make fitting them within text easier.

4. The backslash and square bracket here indicates the start of *display math mode*: mathematical objects that are displayed outside of text.

5. Here we use the `\sum` command to generate a summation equation. We use the underscore to indicate the subscript of the summation, $x = 1$ (enclosed in curly brackets), and we use the carat symbol to indicate the superscript of the summation, 10 (also enclosed in curly brackets). The body of the summation, $3x + 8$, follows nicely after the command.

6. This line signifies the end of display math mode.

7. To demonstrate the scaling-down of mathematical objects in inline math mode, here's the exact same summation from line 5 presented in inline math mode. Note how the bounds of the summation are moved to the right of the sigma (Σ) symbol to save space.

8. As before, this line ends the document body. Note that the percent sign % is used to start a *comment*: everything to the right of the percent sign on a line is considered a comment, and is ignored by the LaTeX compiler.

4.2 More Math Mode Commands and Syntax

Now that you've seen a basic example of inline and display math mode, adapting what you've learned here to other mathematical formulae shouldn't be too difficult. Most LaTeX math mode commands use similar syntax to `\sum`, and with some brief research it should be simple to find a command suited for what you need.

As a very brief example, `\int` is the LaTeX command for the integral symbol. It uses very similar syntax to `\sum`: ignoring the document beginning/end commands for brevity,

```
Look! Here's the integral over $5x^2$ from $\pi$ to $10y + z$.
\[
\int_{\pi}^{10y + z} 5x^2
\]
And here's the sum in inline mode: $\int_{\pi}^{10y + z} 5x^2$.
```

produces

> Look! Here's the integral over $5x^2$ from π to $10y + z$.
>
> $$\int_{\pi}^{10y+z} 5x^2$$
>
> And here's the sum in inline mode: $\int_{\pi}^{10y+z} 5x^2$.

Aside from how similar this example is to the earlier example of `\sum`, there are a few other things to notice: for one, the use of the carat symbol ˆ for exponentiation is something new. Also, note how we use the pi symbol with just a simple command: similar to the `\LaTeX` command discussed before, many such "special" symbols or characters can be generated in LaTeX with merely the use of a simple, easy-to-remember command.

4.3 Aligning Equations

The last thing we'll talk about in math mode is the task of aligning mathematical equations. This is useful for typesetting, say, the solution to a problem spread out over many lines. This is best illustrated with an example:

$$\sum_{i=0}^{\log_3 n-1} 7^i a\left(\frac{n}{3^i}\right) = \tag{1}$$

$$an\sum_{i=0}^{\log_3 n-1} \left(\frac{7^i}{3^i}\right) = \tag{2}$$

$$an\sum_{i=0}^{\log_3 n-1} \left(\frac{7}{3}\right)^i = \tag{3}$$

$$an\left(\frac{\left(\frac{7}{3}\right)^{\log_3 n} - 1}{\frac{7}{3} - 1}\right) = \tag{4}$$

$$\left(\frac{3}{4}\right)an\left(\left(\frac{7}{3}\right)^{\log_3 n} - 1\right) \tag{5}$$

At first glance, this looks like a fairly intimidating equation. Notice the aligning used, however: the coefficients of *an* are perfectly aligned after they have been “pulled out” from inside the summation, and the equals signs of each line are all aligned with each other. These formatting tricks help to make the series of equations readable.

Here’s what the code used to generate this sequence of equations looks like:

```
\documentclass[12pt]{article}
\usepackage{amsmath}
\begin{document}
\begin{alignat}{2}
& \sum_{i = 0}^{\log_{3} n - 1} 7^i a \bigg(\frac{n}{3^i}\bigg) & = \\
an & \sum_{i = 0}^{\log_{3} n - 1} \bigg(\frac{7^i}{3^i}\bigg) & = \\
an & \sum_{i = 0}^{\log_{3} n - 1} \bigg(\frac{7}{3}\bigg)^i & = \\
an & \bigg(\frac{\big(\frac{7}{3}\big)^{\log_{3} n} - 1}{\frac{7}{3} - 1}\bigg) & = \\
\bigg(\frac{3}{4}\bigg)a n & \Bigg(\bigg(\frac{7}{3}\bigg)^{\log_{3} n} - 1\Bigg) &
\end{alignat}
\end{document}
```

If the code seems sort of intimidating, don’t worry! The code isn’t as difficult to understand as you might think at first.

As you can see, we explicitly *use* a package here to obtain extra functionality: `amsmath`. This package was created by the American Mathematical Society (AMS) as an extension to LaTeX, as you can see from the package’s `ams` prefix. We use `amsmath` to make use of the `alignat` environment, which we specify using the previously-seen `\begin` and `\end` commands.

Regarding the actual aligning process done within `alignat`: the required argument to the environment is the number of alignment points within the equations. That is, each line (representing an equation) within the environment should contain exactly that many alignment points. The ampersand symbol, &, is used to indicate a point of alignment. As you can see, the first & takes care of aligning the *an* coefficients on the left side of the equations—and the second & takes care of aligning the equals signs on the right side of the equations.[10]

For reference, the \\ character sequence of two backslashes is used to explicitly end a line in LaTeX. Also, the size of parentheses in LaTeX can be increased with the (`\big`), (`\Big`), (`\bigg`), and (`\Bigg`) commands. See what we did there?[11]

Take some time to look over the code and see how things work. You should be able to

[10] The line numbers off to the right side of the equations are automatically included in the `alignat` environment. Appending an asterisk to the environment name to get `alignat*` removes the presence of these line numbers—this trend, of asterisks removing enumeration, is actually fairly common throughout LaTeX commands and environments.

[11] There are also similar commands, like `\small`, that decrease the size of certain characters. However, in math mode, the need to increase the size of certain characters is generally far more common than the need to decrease the size of certain characters.

understand or at least take an educated guess at what most of the code does, even if the code doesn't look that pretty.

5 Other Useful Features

There are a few other features in LaTeX that are worth knowing about for most people. This section will broadly discuss the use of four key concepts: lists (ordered and unordered), document sections, user-defined commands, and user-defined document functionality. Feel free to skip over certain subsections, although each of these features should be useful in some way to the majority of LaTeX users.

5.1 Lists

Lists are a useful type of structure in almost any kind of document. Users of Microsoft Word or other might have flashbacks to the struggles caused by the software automatically detecting (or, sometimes, incorrectly not detecting) lists, enforcing arbitrary formatting and making editing documents a hassle. Rest assured that using lists in LaTeX is a lot easier, and far more flexible.

5.1.1 Unordered Lists

As always, let's open this section with an example:

```latex
\documentclass[12pt]{article}
\begin{document}
    Here's my shopping list for final exams:
    \begin{itemize}
        \item Bread
        \item Peanut butter
        \item Sufficient motivation (\dots lost track of this)
        \begin{itemize}
            \item Chocolate
            \item Chicken Noodle Soup
        \end{itemize}
    \end{itemize}
\end{document}
```

This is rendered as:

Here's my shopping list for final exams:

- Bread
- Peanut butter
- Sufficient motivation (...lost track of this)
  - Chocolate
  - Chicken Noodle Soup

We make use of the `itemize` environment here, using the `\begin` and `\end` commands as with the `document` and `alignat` environments earlier. Within the `itemize` environment, we can use an `\item` command to indicate an item within the unordered list. We can also nest other lists within lists, as seen in the third item in the outer list.[12]

5.1.2 Ordered Lists

Ordered lists, as you might have expected, are very similar in LaTeX to unordered lists. Instead of the `itemize` environment, you can just use the `enumerate` environment. Here's a similar example to the previous list:

```latex
\documentclass[12pt]{article}
\begin{document}
    Priorities for final exams:
    \begin{enumerate}
        \item Study for algorithms
        \item Write that final paper
        \item Study for programming languages
        \item Survive
        \begin{enumerate}
            \item Eat lots of chocolate
            \item Sleep as much as possible
        \end{enumerate}
    \end{enumerate}
\end{document}
```

This is rendered as:

[12] You can only nest lists up to four levels in standard LaTeX—beyond that, you will need to use something like the `easylist` package to facilitate arbitrary-depth nested lists.

Priorities for final exams:

1. Study for algorithms
2. Write that final paper
3. Study for programming languages
4. Survive
   (a) Eat lots of chocolate
   (b) Sleep as much as possible

5.1.3 Wrapping up with lists

As you can see, lists in LaTeX are pretty simple to use. Although we've only given them a cursory treatment here, LaTeX lists are also very flexible, both through the use of external packages and through built-in options that give you a lot of room to change the behavior of lists. These details are beyond the scope of this guide, but reviewing some of the further reference material in section 7 is an excellent way to get exposed to the advanced functionality of lists.

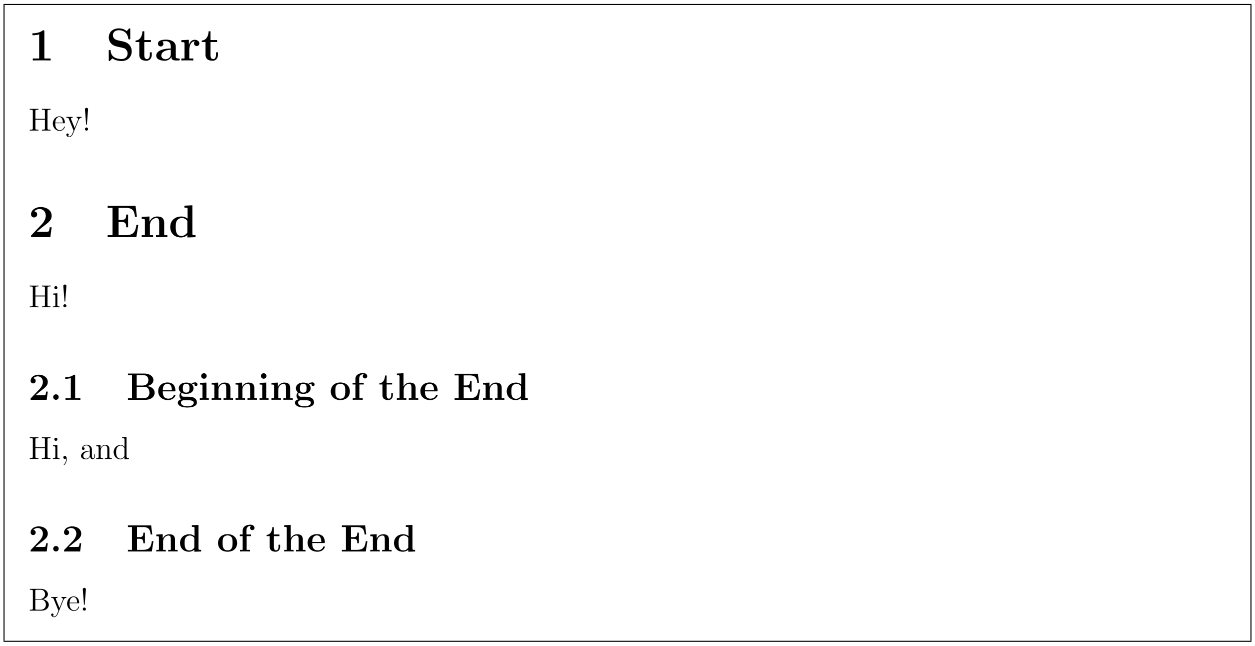
5.2 Document Sections

As you've probably noticed in this document, LaTeX supports the ability to mark parts of a document as within user-specified *sections*. By default, LaTeX only supports three levels of sections, defined by the `\section`, `\subsection`, and `\subsubsection` commands (which each take one required argument, of that particular section's name). If necessary, however, you can use something like the `titlesec` package to circumvent this depth restriction.

A quick, silly example of using sections in a document:

```
\documentclass[12pt]{article}
\begin{document}
    \section{Start} Hey!
    \section{End} Hi!
        \subsection{Beginning of the End} Hi, and
        \subsection{End of the End} Bye!
\end{document}
```

This is rendered as:



1 Start

Hey!

2 End

Hi!

2.1 Beginning of the End

Hi, and

2.2 End of the End

Bye!

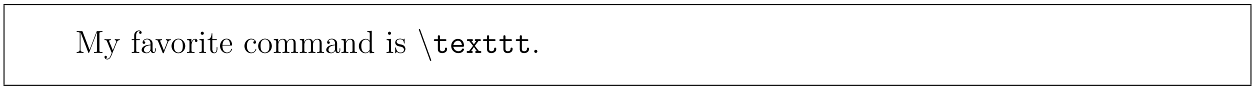
5.3 User-Defined Commands

It's not necessary for normal LaTeX use, but you can define new commands to abstract certain repeated sections of LaTeX code in a reusable command. This is done with the `\newcommand` command.[13]

As a somewhat self-referential example, in the process of writing this guide I found it often necessary to refer to LaTeX commands by name. Ideally, this involves writing the command name in mono-spaced font (done via the `\texttt` command), prefixed by a backslash (to indicate it as a LaTeX command). This is done by writing, say, `\texttt{\textbackslash commandname}` to produce `\commandname`. That's a lot of repetitive things to write out, so I created a simple command, `\dc`,[14] that does this automatically:

```
\documentclass[12pt]{article}
\begin{document}
    \newcommand{\dc}[1]{\texttt{\textbackslash #1}}
    My favorite command is \dc{texttt}.
\end{document}
```

This is rendered as



My favorite command is `\texttt`.

The optional argument to `\newcommand` is a number indicating how many arguments the command takes. The arguments of the command you define are assigned numbers based on

[13] Users of C/C++ may recognize this feature as being somewhat analogous to the use of macros in those languages.

[14] Short for "display command."

the argument ordering.[15] Arguments can be referred to in the defined command's *output*, the second required argument of `\newcommand`, by using a pound sign (or a hashtag, depending on what generation you identify with) followed by the number of the desired argument.

So, following the command definition shows us that `\dc` takes one argument, the command name, which is output in a mono-spaced font and prefixed by a backslash. Every time we call `\dc`, it will be replaced with its argument output in that particular manner.

This is obviously a fairly simple example of a user-defined command, but this should serve as a decent starting point if you're interested in learning more about creating your own commands. Oftentimes I don't realize the usefulness of creating a new command until I start writing a document and notice redunancies like the one rectified by `\dc`.

5.4 User-Defined Document Functionality: Classes, "Templates," and Packages

As Brischoux and Legagneux noted, the use of predefined style information in LaTeX is a very powerful feature that can save a significant amount of time. Many such files exist that fulfill many different needs. To start, though, we'll discuss what forms a LaTeX "template" can actually take.

5.4.1 Classes (`.cls` files) and "templates"

As we discussed when we created our first document earlier in this guide, LaTeX classes can be used to create a variety of document types. LaTeX users can also create their own classes and save them as `.cls` files, facilitating the creation of documents based on user-defined class files.

Actually creating new classes goes beyond the scope of this guide, but implementing user-defined classes is an important skill to know about. Most "style guides" provided by organizations (e.g. academic conferences or journals) will have instructions provided with the requisite files. However, in general, to implement a user-defined class:

1. Download and, if necessary, extract the `.cls` file.

2. Move the `.cls` file to the directory the `.tex` file of your document is in—or, if you'd like to use the class in multiple LaTeX documents, you can move the `.cls` file to the proper location for class files for your LaTeX installation.

3. Declare the document class of your document to be the name of the `.cls` file, but without the file extension. As an example, for the document class `cooldocument.cls`, the first line of your `.tex` file should be `\documentclass{cooldocument}`.

That should finish the task. Many "style guides" provide a starter `.tex` file for you, which—in the most accurate sense of the word—is what we could call a *template*.[16] Using such a file

[15] Such that argument 1 is #1, argument 2 is #2, etc.

[16] For an example of such a "style guide," see the Association for Computing Machinery's guides here: `http://www.acm.org/publications/acm-latex-style-guide`

removes the need for you to follow steps 2 and 3 above.[17]

5.4.2 Packages (`.sty` files)

Remember the `amsmath` package we used earlier, when we were discussing math mode? Packages like that are used to provide extra functionality to LaTeX documents. Broadly speaking, while classes define the type of a document, packages extend this in a generally class-independent manner. Packages can just be included in a document via the `\usepackage` command in the *document preamble*: the space after the `\documentclass` has been defined but before the `\begin{document}` command has been provided.[18] The `\usepackage` command takes the name of the package to be included as an argument. Packages use the file extension `.sty`.

Many packages, like `amsmath`, are bundled with some distributions of LaTeX: this means that you don't have to bother with downloading them yourself, and you only have to specify your use of the package via `\usepackage`.

However, less-frequently-used packages may not be readily included with LaTeX. As an example, to include the (very fictional) package `coolstuff.sty` in your document, you should first ensure that the `coolstuff.sty` file is located in the proper location for packages for your LaTeX installation. (The exact location will vary depending on how you installed LaTeX earlier.) After that, the process is the same as before: include the line `\usepackage{coolstuff}` in your document preamble.

6 Conclusion

In just a few pages, we've covered the basics of LaTeX. We've discussed compilation, document creation, math mode, lists, sectioning, creating commands, and user-defined templates in some form or another.

However, this guide has barely scratched the surface of what's possible in LaTeX. There's so much to know about the software that trying to get started can leave many people paralyzed with indecision—I know that's how my first experiences with LaTeX went, and that's one of the main reasons for this guide.

After reading this document you should be ready to get started on creating something in LaTeX. What you create matters less than going through *the process* of creating: doing research on further LaTeX functionality, running into errors in your document, fixing them, and being able to say at the end of the journey that you typeset a document by yourself is fantastic experience for getting comfortable with LaTeX.

For inspiration (and for reference), I've included a number of interesting and helpful references in the next section, from reference material to task-specific guides to LaTeX editors. Hopefully you'll find something there that piques your interest and inspires you to create something!

[17] Furthermore, some user-defined classes may take certain optional arguments that you could look into making use of. Reading the documentation/instructions provided by the class' creator is useful here.

[18] See the code from section 4.3 for an example of a document preamble containing a `\usepackage` command.

7 Further Information

This section contains a number of references you may find helpful in applying LaTeX to your everyday work.

7.1 General Reference Material

- CTAN, `https://www.ctan.org/`: The Comprehensive TeX Archive Network (abbreviated CTAN) is a “set of sites” that distribute materials related to TeX, the typesetting system upon which LaTeX is based. These materials include but are not limited to packages that add functionality to TeX (at the time of writing, over 5,000 such packages are distributed by CTAN), distributions of TeX and TeX’s descendant software (including LaTeX), and a significant amount of related documentation. Over 2,000 contributors have helped with this project, including Donald Knuth, Professor Emeritus of Computer Science at Stanford University and the creator of TeX.

- LaTeX Wikibook, `https://en.wikibooks.org/wiki/LaTeX`: The LaTeX Wikibook is probably one of the most useful sources of documentation on LaTeX. The book is composed of the curated insights of a staggering amount of collaborators from many different fields. The book is accessible and searchable online through a simple user interface, and printable and PDF versions of the over-700-page book are also available.

- ShareLaTeX Documentation, `https://www.sharelatex.com/learn/Main_Page`: ShareLaTeX is a popular online LaTeX editor that is discussed in section 7.3. The ShareLaTeX website contains comprehensive documentation on many features of LaTeX, from equation typesetting to table and figure creation.

7.2 More Specific Guides

Although general information is useful, sometimes it helps to find a detailed guide to a particular application of LaTeX to a certain field of study or a certain type of document. I’ve compiled a few such sources here.

- Typing Math, `http://www.math.uiuc.edu/~hildebr/tex/course/intro2.html`: Dr. A. J. Hildebrand is a Professor Emeritus of mathematics at the University of Illinois. This article, created by Dr. Hildebrand, is a useful guide and reference on typesetting mathematical equations in LaTeX. The document is a slightly more in-depth treatment of math mode than in this guide, so Dr. Hildebrand’s guide is a great next step if you’d like to learn some more about typesetting mathematical equations.

- Writing a thesis with LaTeX, `https://tug.org/pracjourn/2008-1/mori/mori.pdf`: Intended for an audience that already knows the basics of LaTeX, this guide by Dr. Lapo Mori is an excellent source of information for students planning on (or interested in) writing a thesis using LaTeX.

- Template-based introductory guide to LaTeX for Economics, `http://faculty.gvsu.edu/ogural/LaTeX%20for%20economists.pdf`: Dr. Laudo M. Ogura, an economics professor at Grand Valley State University, authored this introductory guide to LaTeX for use by incoming economics doctoral students. The guide should be a useful resource for readers interested in composing social science documents in LaTeX.

7.3 Specialized LaTeX Editors

Although we assume the use of a normal text editor to edit LaTeX code in this guide, there are a number of other options for editing LaTeX code that can make the process significantly easier. Many such editors, for example, will automatically compile your code—removing the need for you to go through the manual compilation process.

What editor you use is entirely up to you. However, to get you started, here's just a few options I've found:

- Overleaf, `https://www.overleaf.com/`: An online collaborative LaTeX editor.
- ShareLaTeX, `https://www.sharelatex.com/`: Another online collaborative LaTeX editor.
- TeXworks, `http://www.tug.org/texworks/`: A graphical LaTeX editor that is open-source software. TeXworks is available for Windows, OS X, and Linux. (Disclaimer: I mostly use TeXworks.)
- Texmaker, `http://www.xm1math.net/texmaker/`: An open-source LaTeX editor for Windows, OS X, and Linux.
- TeXstudio, `http://www.texstudio.org/`: An open-source LaTeX editor for Windows, OS X, and Linux that originated as a fork of Texmaker with the goal of adding more features to the editor.