Just Another Introduction to LaTeX: A (Very) Short Course

Prepared by Jonathan P. Olmsted

Updated: September 29, 2014

Introduction to this Short Course

Purpose. The aim of this document is to serve as a set of guided lessons in creating LaTeX documents and to serve as an example of the kind of documents that can be produced with the LaTeX family of tools. It is not and does not aim to be comprehensive.

About. This document was originally prepared by the author during his tenure as **the starlab** fellow for the incoming graduate student cohort in the Political Science department at the University of Rochester. Previous versions of the short course and the corresponding materials are available at **the starlab**'s website (http://www.rochester.edu/college/psc/thestarlab/). This document began as an adaptation of Arthur Spirling's version. It's current version may or may not have diverged considerably.

In addition to the supplemental files used for the various exercises included in this version, all files required to create this document are available at https://github.com/olmjo/LaTeX-Intro.

Rights. This document is released under the Creative Commons Attribution license ⓑⓨ .

Comments. If you have any comments, questions, or concerns regarding this document please contact Jonathan Olmsted jpolmsted@gmail.com.

Contents

Part I
An Introduction to Using LaTeX

1 | What is LaTeX (lay-teck) or LaTeX, but not latex or Latex?

- LaTeX is an open, document–preparation system used by many students, academics, and publishers whose content involves technical topics or complex structure (*e.g.* math professors, engineers, you).

- LaTeX is a kind of markup language. This means LaTeX files explicitly articulate both the *content* and the *structure*. Other markup languages include HTML, XML and Markdown. If you go to the New York Times webpage (www.nytimes.com) and view the HTML source for the main page it will look (incredibly) different from how that web page is rendered by your browser. [1] The web page is written in the HTML markup language, but then a separate program renders the page. The main point here is that "creation" and "consumption" are conceptually distinct. LaTeX is much more similar to HTML source than it is to a word processor (*e.g.* MS Word, OpenOffice Writer, Google Documents). When you work with a LaTeX file, you describe how the file should look, but this is not displayed for you in real-time.

2 | Why Would You Use LaTeX?

LaTeX has a number of distinct advantages:

- A LaTeX document can be very pleasing to the eye without the need for any customization.

- When your focus is on producing content, but are neither a publisher nor a graphic designer, LaTeX lets you focus on the content: LaTeX separates content from appearance.

- LaTeX's ability to incorporate math is un-rivaled.

- You will never get locked out from the fruits of your intellectual labor because of licenses or old software. The software is open and the files you create are "flat text" files. If someone has a computer, they can read your LaTeX files.

- Also, because it relies on plain text files, collaboration among multiple contributors to a single project can be managed *much* more readily.

- The markup code for LaTeX math is one of the canonical ways of representing math in plain-text environments (*e.g.* email, instant messaging).

3 | How Does LaTeX Work?

The process of creating a LaTeX document is quite different from creating one in a word processor. There is, undeniably, a learning curve.

Although most of the technical details can safely be ignored, some understanding of the different components comprising a functioning LaTeX workflow is helpful at the outset. And, understanding how pieces fit together makes asking for help much easier. A basic representation of the workflow is given in Figure 1.

The basic steps to creating a document are:

---

[1] In the Google Chrome browser you can view source by navigating to the "Tools" menu item and choosing "View Source".

Figure 1: Example LATEX Workflow
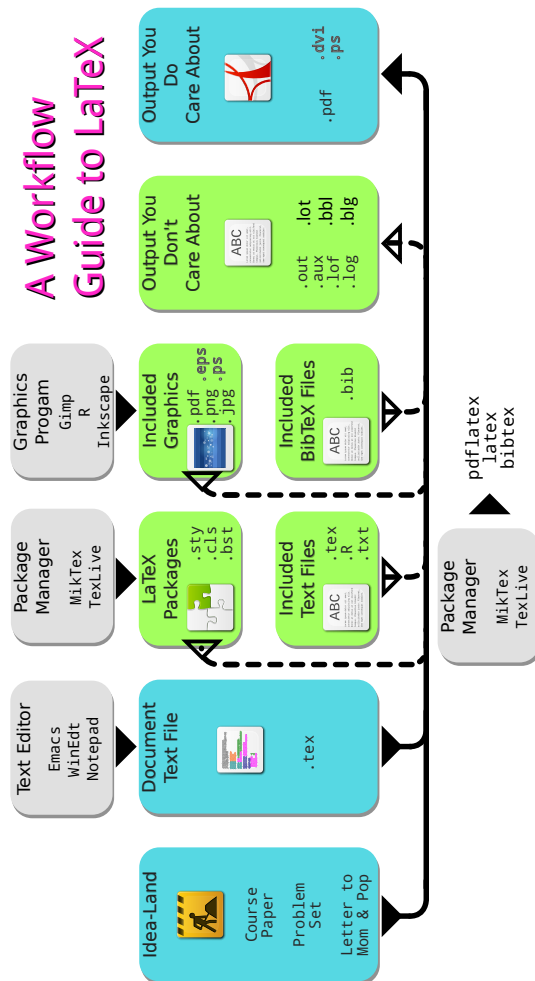
An example workflow using LATEX to produce the output you care about. Blue regions represent the components you will focus on the most on a daily basis. Grey regions represent the software that will help you get from start to finish. Green regions represent intermediate or sometimes option files that are less focal.

5

1. Have an idea!

2. Create a plain text document containing the content in a text editor. These plain text files a typically given a file extension of `.tex`.

3. Process the plain text document with the LATEX "program".

4. LATEX pulls in any additional files that are asked for—but they have to be asked for—and creates, among other files, the document you will print or email. Most of the time, this is a PDF file.

5. View the result/rendering of your LATEX document by opening the PDF file.

6. Be happy.

Notice that this is different from the word processor workflow. In that case, Steps 2, 4, and 5 are basically integrated. Step 3 is unnecessary. Step 6 may or may not happen.

What now follows is some more explanation of the kind of software will be needed for each step.

Step 2 You could use any number of other text editors. Many Windows users like WinEdt or Sublime. Other options include nerdier text editors like Emacs. It's best to find a text editor with well-documented LATEX integration. This will facilitate the next steps after you write your file.

We typically denote LATEX files with the `.tex` file extension. Although it isn't necessary in some cases, most software and operating systems will make your life easier if you are willing to name LATEX files what they expect you to name them.

Step 3 If you have chosen your text editor well, you can achieve this from within that program. For example, WinEdt has a button you can just click to process your `.tex` file.

Also, LATEX-ing your document can proceed through one of two different branches. I will focus on the `pdflatex` branch and not the `latex` branch. `pdflatex` will convert your `.tex` file into a `.pdf` document. Included images must be `.jpg`, `.png`, `.pdf`, or `.svg` files. These inputs and PDF output are far more common now, so that is the focus here.

Step 4 Sometimes markup in the LATEX file will require contributed packages which tell the LATEX how the content should be rendered. This is one of the critical benefits of using a system like LATEX: you can utilize functionality that many other people have already provided rather than re-invent it or do without.

However, making sure these packages are placed in the right spot on your system and up-to-date can be involved and opaque to the newcomer. LATEX package managers simplify this. On Window machines, MikTeX is often used. By and large, you can and should ignore this fact in the beginning of your LATEX experience. What is important is to realize the following:

$$\text{LATEX} \neq \text{a text editor} \neq \text{a package manager (e.g., MikTeX)}.$$

Step 5 Once you have a PDF file, you can view it in your preferred PDF viewer like Adobe Reader. A well-chosen text editor will make this viewing easy, too.

Step 6 No software necessary.

4 | Exercise: Your First Document

1. Go to https://github.com/olmjo/LaTeX-Intro/tree/master/Extra_Materials.

2. Save the file FakeFile.tex to your desktop.

3. Open this file in your text editor.

4. Run pdflatex on the .tex file and view it.

5. Notice how many ancillary files this process creates.

The contents of this file are as follows:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%% PREAMBLE %%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%% CLASS %%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\documentclass[11pt,letterpaper]{article}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%% PACKAGES %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\usepackage{graphicx}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%% SETTINGS %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\pagestyle{plain}

\author{}
\title{Questions and Answers}
\date{\today}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%% CONTENT %%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\begin{document}

\maketitle

\section{Hello World.}
Hello, right back at you. Also, not a question.

\section{What is this?}
This is a document.

\section{Why is this?}
Buttons were pushed, naturally. Come back later.
```

7

```
\section{No, but \textit{why} is this?}
That's deep!

\end{document}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Listing 1: `FakeFile.tex`

The resultant PDF file should look like Figure 2.

You have successfully created your first LATEX document and simultaneously learned to always keep LATEX files/projects in separate directories because the output will create clutter.

Figure 2: `FakeFile.pdf` Output
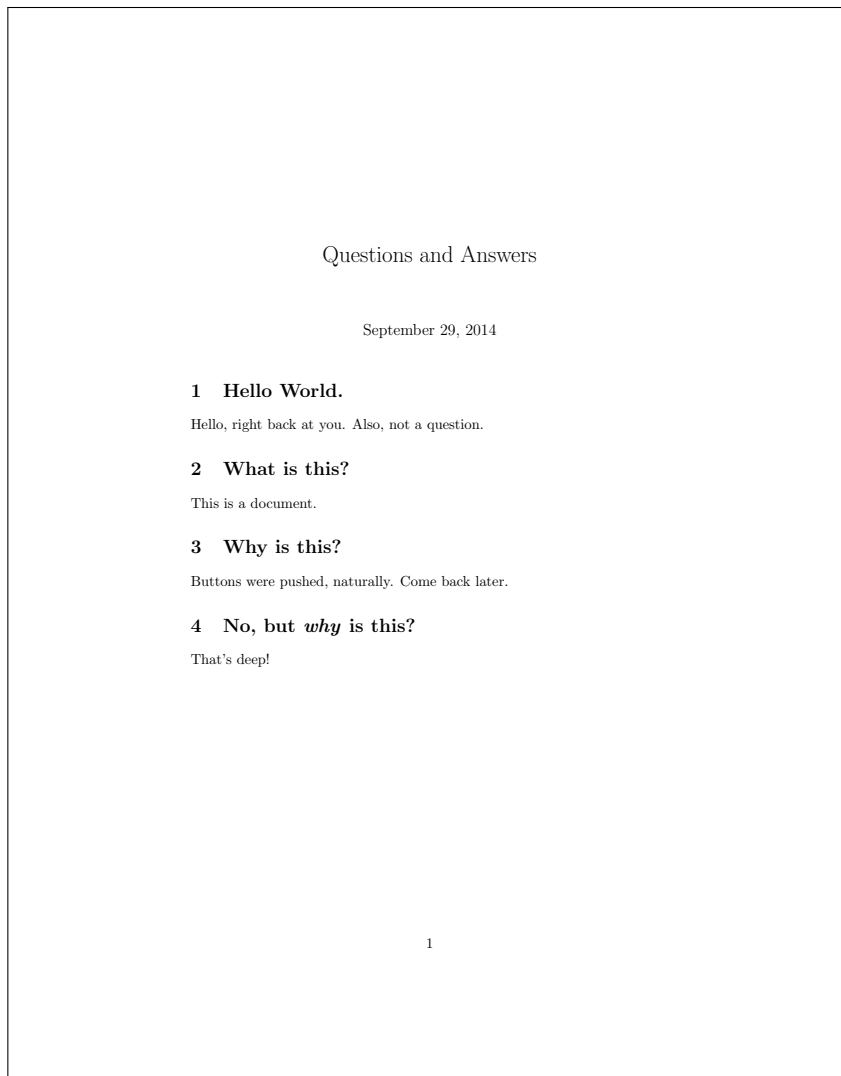
This figure shows the PDF output created by the input LATEX file `FakeFile.tex`.

Part II
Basic Content in LaTeX

5 | LATEX's Interpretation of Plain Text

- The entire `.tex` file you write will contain only characters on your keyboard. That means, somehow, the characters on your keyboard you need to represent: a, A, α, $\mathbb{A}$, à, A, $_a$, and Ä.

- The file that you write will be composed entirely of the characters in Table 1. This is true no matter what kind of crazy symbols are ultimately included, such as .

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
0 1 2 3 4  6 7 8 9
! @ # $ % ^ & * ( ) _ +
{ } [ ] | \ l ; : ' " < , > . ? /
```

Table 1: Characters To Use in `.tex` Files

Special Characters

- The following characters are reserved or *special*:

```
% # $ ^ & _ { } ~ \
```

Table 2: Special Characters in `.tex` Files

- By special, we mean that each of these characters in the LATEX file doesn't represent itself in your output file. If you type "%" into your text editor and build your output, you will not get a percent sign. Rather, these *special* characters mean something very specific to LATEX.

- In order to generate these symbols in your *output* you need to use other, non-literal representations.

  Although detailed descriptions of these symbols is beyond the purpose of this section, accept the following brief comments:

Character	Purpose
%	used for including comments and preventing LATEX from interpreting file contents
#	used to define LATEX commands (or macros)
$	used to start LATEX math mode
^	used for superscripts in math mode
_	used for subscripts in math mode
&	used for alignment in LATEX math and tables
{ }	used to pass arguments to LATEX commands
~	used to represent a special kind of whitespace
\	used to start every LATEX command

Table 3: Purpose of Special Characters

Commands

- Using LATEX efficiently often requires one be familiar with various LATEXcommands (or macros). In general, this comes only with practice. Commands start with a "\" and they are case-sensitive. Commands can be used to generate particular glyphs (think shapes on the page) or alter the content in some way.

- One command is "\LaTeX{}" which produces "LATEX". Similarly, "\textbf{election}" produces "**election**".

- You'll never learn a significant proportion of all the LATEX commands that people use. However, you'll eventually memorize the ones you use over and over and know how to look up the rest.

- The mandatory arguments to a command are passed inside curly braces and the option arguments to a macro are passed inside square brackets. The result is something like

  ```
  \command[optional1=1, optional2=2]{mandatory=Always}
  ```

  where the (optional) argument `optional1` is being passed the value 1, the (optional) argument `optional2` is being passed the value 2, and the (mandatory) argument `mandatory` is passed the value "Always".

Whitespace and Spacing

- 2 or more carriage returns, "\\", and "\linebreak" break lines

- "~" is a non-breaking space

- "\par" will create a new paragraph for the succeeding text, regardless of surrounding whitespace

- "\linebreak" and "\pagebreak" are appropriately named

- multiple consecutive spaces are interpreted as one

11

So,

```
And in the naked light I saw \\ Ten thousand people, maybe more.

      People talking without speaking, \\
      People~hearing~without~listening,

People writing
songs that voices never share \par And no one dared

Disturb      the       sound       of       silence.
```

produces the following text:

> And in the naked light I saw
> Ten thousand people, maybe more.
> People talking without speaking,
> People hearing without listening,
> People writing songs that voices never share
> And no one dared
> Disturb the sound of silence.

6 | `.tex` Input File Structure

- LaTeX `.tex` files have a particular structure. If the file you attempt to compile has an error, your document will either fail to be produced or it will not be compiled in accordance with your intentions.

- First, you must declare the *document class*:

```
\documentclass[letterpaper, 10pt]{article}
```

or

```
\documentclass[12pt]{letter}
```

and this is part of the *preamble*. For most purposes, `article` is sufficient.

- Second, the remainder of the preamble contains explicit calls to outside packages if you require their functionality, the creation of new macros, and setting document properties. After the `\documentclass` command we might see the following lines:

```
\usepackage{fancyhdr}

\pagestyle{fancy}
\cfoot{\thepage}
\title{A Document}
```

which provides the functionality from the `fancyhdr` package, sets the "`pagestyle`" to "fancy", places the page number in the center of the footer, and sets the title to "A Document".

This is also part of the *preamble*.

- Third, the main content is typed within the "`document`" environment, as in:

```
\begin{document}

%% place content here

\end{document}
```

- Lastly, any characters occurring after the close of the `document` environment will be ignored by LaTeX. This includes special characters and commands.

In its entirety, we might have the following `.tex` file:

```
\documentclass[12pt]{article}

\title{An Essay about a Guy Named Rob}

\begin{document}

\maketitle

Rob is a cheery chap whose trumpeted lips are sometimes chapped.

%% note to self: add some additional character development

\end{document}

$ breaking \ rules # $ without }any{ effect
```

Listing 2: `fakefile2.tex`

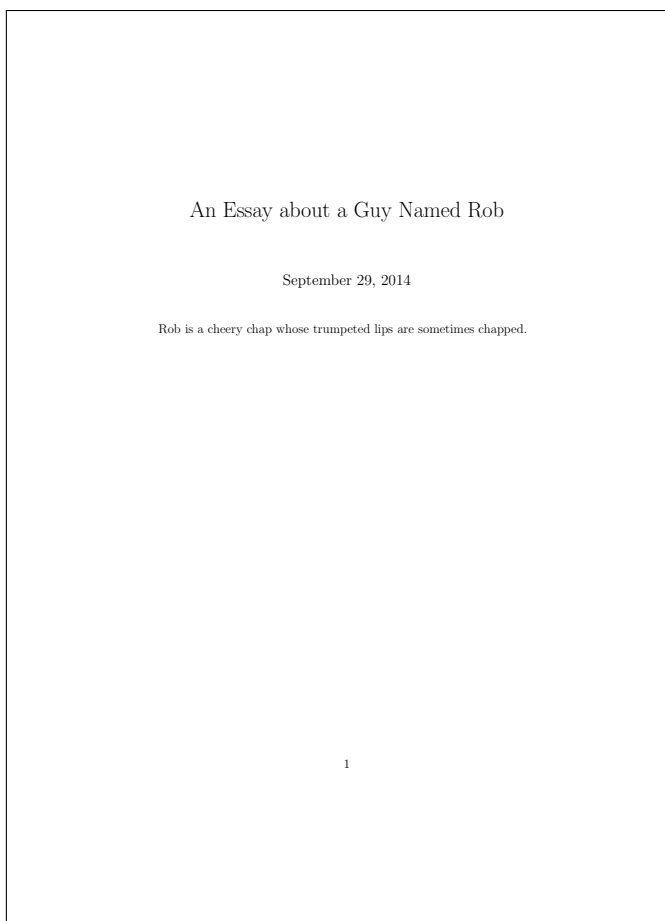Go ahead and compile this document after you type it out.

An Essay about a Guy Named Rob

September 29, 2014

Rob is a cheery chap whose trumpeted lips are sometimes chapped.

1

Figure 3: Output from Example Input: `fakefile2.tex`

7 | Output File Structure

Headings

- LaTeX accepts the definition of a document hierarchy and displays headings appropriately. The *article* class accepts the following headings whose order indicates how far down they are in the order: `\part{}`, `\section{}`, `\subsection{}`, `\subsubsection{}`, `\paragraph{}`, and `\subparagraph{}`. If an "*" is placed after the name as in "`\part*{}`" then the heading will not be numbered and will not be included in the table of contents, if it exists. Otherwise it will be numbered and included in the table of contents. The one mandatory argument is the text of the heading.

- Any headings after the `\appendix` command will be altered to reflect that they do not belong to the main body of the text and, rather, the appendix.

Add a section and a subsection* heading to your document. Name them after your two favorite colors. Compile the document.

8 | Environments

- All of the content that will be displayed in your final document is contained in at least one environment. Environments are contexts which affect how the content is rendered and interpreted. Environments are indicated by the `\begin{env}` and `\end{env}` commands. We have already seen the document environment.

- We will discuss various environments (and there are many) during the remainder of the week, but the basics are all the same. If you begin an environment, you must end it. When nesting environments, the most recently opened environment must be closed before an earlier environment can be closed.

- The `enumerate` environment creates ordered lists. The `itemize` environment creates unordered lists. The `table` environment is used to encapsulate table-like content.

  The insertion might look like

  ```
  \begin{tiny}
  ``Stellaaaaaaaaa!''
  \end{tiny}
  ```

At the end of your content begin and end a "tiny" environment. Inside this environment type your favorite thing to shout. Use proper latex quotes (*i.e.* "..." and not "..."). Compile the document.

9 | Error Debugging

You will, invariably, make errors. However, you'll probably never be the first person to make any particular error. The four most common errors are

1. you type a command incorrectly

2. you forget to `\usepackage` the package that supplies a command which LaTeX interprets as if you typed a command incorrectly

3. you don't `\end{env}` what you `\begin{env}` or you abuse the order of nested environments

4. you have un-balanced braces/parenthesis/brackets depending on what LaTeX is looking for.

Place the `tiny` environment inside a `center` environment. Now, change one of the `\begin` commands to `\benign`. Compile it. This is not benign! Look at the error message. Correct this mistake and swap the two `end` statements. Compile the document now. Look at the error message.

10 | Solving Problems and Getting Help

- Don't panic.

- Check the usual suspects.

- Google a description of your problem in addition to "latex ctan." Including "latex" without "ctan" is a bad, bad, bad, bad, bad idea!

- Comment out potential sources of the problem until you identify it. Slowly add things back in until you have removed only the offending portion. Can this be fixed?

- Consult one of the many great LaTeX sites out there.[2]

- Ask a friend!

---

[2]Google either "latex ctan wikibook" or "latex comprehensive symbol". These two are great.

Part III

Creating Full Documents in LATEX

11 | Starting Point

- This section builds off of the previous one which introduced several LATEX commands.

- By the end of the last section we had working documents with environments and headings that were pleasing to the eye.

- We will quickly create a very basic document to pick up where we left off.

- Open your text editor and create a new LATEX file with a `*.tex` file extension wherever is most sensible to you.

- Create a basic document with the following "code" and check that it compiles.

```
\documentclass{article}

% This is just a comment.
\begin{document}

\section{The Content}

\subsection{Short Content}

\subsection{Long Content}

\appendix

\section{Some Book-keeping}

\end{document}
```

Listing 3: `fakefile3.tex`

- Now, download the two dummy text files from https://github.com/olmjo/LaTeX-Intro/tree/master/extra. They are named `loremipsum-short.txt` and `loremipsum.txt`.

- Place these two files in the same folder/directory as your new LATEX file.

- As it stands, the document is quite bare. Modify the contents in the following way:

    - Add `\input{./loremipsum-short.txt}` on a new line directly after `\section{Short Content}`. That part of your `.tex` file should look like:

    ```
    \section{Short Content}

    \input{loremipsum-short}
    ```

    As long as the downloaded file was placed in the appropriate place, you can safely ignore what this command does.

17

– Similarly, add `\input{./loremipsum.txt}` on a new line directly after `\section{Long Content}`, as in

```
\section{Long Content}

\input{loremipsum}
```

As long as the downloaded file was placed in the appropriate place, you can safely ignore what this command does.

- Compile this document again. You should have a document with a considerable amount of filler text in two of the sections.

12 | Fonts

In LaTeX we can make some very basic changes to the font in with ease. These changes can be made with one of two approaches. The *environment* approach or the *command* approach.

In the first, whatever font we change will be inside an environment. In the second, whatever font we change will be the mandatory argument to a command.

Font Styles

When displaying some text in **bold face** or SMALL CAPS, we are changing font styles.

The basic and most common changes are here:

Style	Environment	Command
normal/default	`\begin{textnormal} ...\end{textnormal}`	`\textnormal{...}`
bold	`\begin{bf} ...\end{bf}`	`\textbf{...}`
italics	`\begin{it} ...\end{it}`	`\textit{...}`
SMALLCAPS	`\begin{sc} ...\end{sc}`	`\textsc{...}`
underline	none :-(	`\underline{...}`
Roman	`\begin{textrm} ...\end{textrm}`	`\textrm{...}`
Sans Serif	`\begin{textsf} ...\end{textsf}`	`\textsf{...}`
teletype	`\begin{tt} ...\end{tt}`	`\texttt{...}`

- Place the `\input` command under the "Short Content" section inside a font style environment of your choice. Compile the document.

```
\section{Short Content}

\begin{textsf}
  \input{loremipsum-short}
\end{textsf}
```

- Before this environment, but under the "Short Content" section type and complete the following sentence: "If I had a boat, I'd ...".

```
\section{Short Content}

If I had a boat, I'd go out on the ocean.
\begin{textsf}
  \input{loremipsum-short}
\end{textsf}
```

- Finally, change the style of the font for that sentence and only that sentence by using the command approach. This time, choose a different style from the environment change you made above.

  As an example,

```
\textbf{If I had a boat, \textit{I'd go out on the ocean}}.
```

  gives

> **If I had a boat, *I'd go out on the ocean*.**

Sizes

The size the font can be modified in ways similar to how the font style was changed.

Style	Environment	Command
tiny	\begin{tiny} ...\end{tiny}	\tiny{...}
scriptsize	\begin{scriptsize} ...\end{scriptsize}	\scriptsize{...}
footnotesize	\begin{footnotesize} ...\end{footnotesize}	\footnotesize{...}
small	\begin{small} ...\end{small}	\small{...}
normalsize	\begin{normalsize} ...\end{normalsize}	\normalsize{...}
large	\begin{large} ...\end{large}	\large{...}
Large	\begin{Large} ...\end{Large}	\Large{...}
LARGE	\begin{LARGE} ...\end{LARGE}	\LARGE{...}
huge	\begin{huge} ...\end{huge}	\huge{...}
Huge	\begin{Huge} ...\end{Huge}	\Huge{...}

- Place the \input command under the "Long Content" section inside a font size environment of your choice, but choose a size that is smaller than the normal size.

```
\section{Long Content}

\begin{tiny}
  \input{loremipsum}
\end{tiny}
```

- Compile your document.

19

13 | List Environments

There are three main list environments that you will likely use: `itemize`, `enumerate`, and `description`.

`itemize` The `itemize` environment allows you to make bullet-ed lists without conveying any order. These can be nested to create any structure desired. However, complex nesting often results in user-mistakes.

`enumerate` The `enumerate` environment allows you to make numbered lists which convey a precise order. The numbering is automatic.[3] These can be nested to create any structure desired and the "numbering" system used at each level reflects how far in the tree the entry is.

`description` The `description` environment is slightly different. In fact, I am using it here to describe the different kinds of list environments. Instead of each entry being marked with an automatic bullet or number, the user must provide the "term" being described.

- For either `itemize` or `enumerate` your code looks like this, where "`environment`" is swapped out for the correct environment:

```
\begin{environment} % itemize or enumerate
\item An item
\item Another item
\item Still more items!
\end{environment} % again, either itemize or enumerate
```

- For the `description` environment your code is slightly different:

```
\begin{description}
\item[An item] description 1
\item[Another item] description 2
\item[Still more items!] description 3
```

- Create a list in the "Some Book-keeping" section. In this list you should have two items—two of your favorite books. The environment can be any of the three. If you are using the `description` environment include the author's name as the description if you know it. In this case, make sure the book title is in square brackets.

14 | Metadata: titles, dates, authors, abstracts

`\maketitle`

- Although our content is already impressively long, the document is lacking all metadata. To address this we will formally introduce the `\maketitle` command

---

[3]Like everything in LATEX, if it is important enough, you can override default. But, the guiding ethos of LATEX is that authors not spend long periods of time worrying about format rather than content.

- In the preamble of the file (that is, before `\begin{document}`, but after `\documentclass`), enter the following code:

```
\title{My First Opus}
\date{\today}
\author{Woodrow Wilson}
```

- Compile the document. Notice that *nothing* changes.

- Now, add `\maketitle` on a new line directly after `\begin{document}` as in

```
\title{My First Opus}
\date{\today}
\author{Woodrow Wilson}

% This is just a comment.
\begin{document}

\maketitle
```

- Compile again.

- Comment out the line with the `\date` directive. Re-compile. What results from the following code?

```
\title{My First Opus}
% \date{\today}
\author{Woodrow Wilson}

% This is just a comment.
\begin{document}

\maketitle
```

- Uncomment that same line, delete the directive's single mandatory argument and compile. What happens from this code, instead?

```
\title{My First Opus}
\date{}
\author{Woodrow Wilson}

% This is just a comment.
\begin{document}

\maketitle
```

- In order to use the \maketitle command, a title must be defined. The author and date, however are optional.

\thanks

- Woodrow Wilson is a humble man and acknowledges the contribution that non-Woodrow Wilson entities play in his success. To add this to the title we use \thanks.

- Include the following line at the end of "Wilson" within the author command as part of the mandatory argument: \thanks{The author thanks Prospect House for having a sufficiently pretty garden.}

```
\author{Woodrow Wilson\thanks{The author thanks
 Prospect House for having a sufficiently pretty garden.}
}
```

- You can use \thanks to include institution information or contact information, too. Or, you can use \footnote which behaves the same way.

titlepage

- Because this opus is so substantial, it can be overwhelming to assault the reader with content on the first page. In this case, we'd like to have just a title page.

- Place the \maketitle directive in an environment by the name of titlepage

- Compile the document and see that there is now a title page separated from the content itself.

abstract

- Even with a title, we might want to convey the content of our document in more detail.

- In this case, we'd use an abstract.

- After `maketitle` but within the `titlepage` environment, create a new environment (which is completely nested) called `abstract`. Within the `abstract` environment, write two sentences about what we have covered in this tutorial, so far. Include the word LaTeX, properly typeset.

- Compile the document.

15 | Fancy (and not Fancy) Headers

- By default, the only content in the header or footer of our document is the page number.

- We can change this by using a package called `fancyhdr`.

- Tell LaTeX to use this package. What is the command? Where does it go?

- In the preamble, also include `\pagestyle{plain}`. Compile the document. What changes?

- Now use `\pagestyle{fancy}` instead. Compile the document. What changes?

- You can change the contents of the headers and footers by using commands of the form `\rfoot{}` for r (right), l (left), and c (center) ; and for `foot` (the footer) and `head` (the header).

- Place the state or non-US country in which you were born somewhere in the footer. Compile the document.

- Notice that the first page, the title page, has no header. Even when we use fancy headers and get the pretty content on that first inch of the page, any page with `\maketitle` is unchanged, by default.

  We can change this by including `\thispagestyle{fancy}` immediately after `\maketitle`.

16 | Putting it Together

```
\documentclass{article}

\usepackage{fancyhdr}

% This is just a comment.

\title{My First Opus}
\date{\today}
\author{Woodrow Wilson\thanks{The author thanks
    Prospect House or having a sufficiently pretty garden.}
}

\pagestyle{fancy}
\rfoot{Virgina}

\begin{document}

\begin{titlepage}
  \maketitle

  \begin{abstract}
    The tutorial has covered basic \LaTeX{} commands useful for creating full
```

```
    documents. There is clearly a learning curve in learning LaTeX{}, but the
    output sure is pretty.
  \end{abstract}
\end{titlepage}

\section{The Content}

\subsection{Short Content}

\textbf{If I had a boat, \textit{IâĂŹd go out on the ocean}}.

\begin{textsf}
  \input{./loremipsum-short.txt}
\end{textsf}

\subsection{Long Content}

\begin{tiny}
  \input{./loremipsum.txt}
\end{tiny}

\appendix

\section{Some Book-keeping}

\begin{itemize}
\item Where the Sidewalk Ends -- Shel Silverstein
\item The Philosophy Gym -- Stephen Law
\end{itemize}

\begin{description}
\item[Where the Sidewalk Ends] Shel Silverstein
\item[The Philosophy Gym] Stephen Law
\end{description}


\end{document}
```

My First Opus

Woodrow Wilson[*]

September 29, 2014

Abstract

The tutorial has covered basic LaTeX commands useful for creating full documents. There is clearly a learning curve in learning LaTeX, but the output sure is pretty.

[*]The author thanks Prospect House or having a sufficiently pretty garden.

1

---

1 The Content

1.1 Short Content

If I had a boat, *Id go out on the ocean*.

../loremipsum-short.txt

1.2 Long Content

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed metus lorem, dignissim sed adipiscing in, lobortis et quam...

A Some Book-keeping

- Where the Sidewalk Ends – Shel Silverstein
- The Philosophy Gym – Stephen Law

Where the Sidewalk Ends Shel Silverstein

The Philosophy Gym Stephen Law

Virginia

1

Figure 4: First Two Pages of a Full Document

Part IV
Including Mathematics in LaTeX Documents

17 | Additional Packages

- In order to avoid a lot of redundant and complicated code, it is helpful to load two particular packages each time you create a document with mathematics in it.

- A bare-bones document requires only a `\documentclass{}` line, followed by the `\begin{document}` and `\end{document}` lines.

- We will now add `\usepackage{amsmath, amsthm, amssym, amsfonts}` to the *preamble* of the document. This tells LaTeX to recognize certain commands defined in these packages.

- We'll never use most of the functionality they add, but we'll often use some of it!

- Create the following empty LaTeX document in WinEdt in which we'll practice mathematical commands.

```
\documentclass{article}

\usepackages{amsmath, amsthm, amssymb, amsfonts}

% This is just a comment.

\begin{document}

\end{document}
```

18 | Math Modes

- In order to display mathematical content in LaTeX, we will use a new set of commands that we wouldn't need for, say, writing a letter. However, LaTeX doesn't allow you to issue these commands just anywhere.

- Rather, you need to be either in math mode or in a mathematical *environment* where the math mode is implied.

- There are two math modes which don't utilize special *environments*: *in-line* math and *display math*.

- These two are analogous to the way we include quotes in our own written work.

In-line Sometimes the quotes are treated like any other text and printed in the same sized font without any special indentation.

Display   Sometimes the quotes begin on new lines, are indented, and quite distinct from the other text.

- As a note, some commands and characters can only be entered in math mode. However, many, like the word "computer" can be entered in normal mode and math mode. Yet, if you think "computer" is different from "*computer*", then you must beware which mode you use.

- After this section, it will be assumed that all math command sequence are typed in math mode. Otherwise, they will almost always fail. The math mode openings and closings are omitted.

In-line Math

- The *in-line* math mode is opened by typing a single dollar sign ($) and then closed by typing a single dollar sign.

- The content is placed in between the two $ symbols.

- When we type `$55/i = \pi^{-0.3}$`, we get $55/i = \pi^{-0.3}$. Any lines immediately following the in-line text are entirely unaffected, including this one. And this one, too.

Display Math

- The *display* math mode is opened by typing `\[` and closed by typing `\]`. Another approach is to use double dollar signs as in the in-line mode (`$$ ...$$`). However, the square bracket approach is considered "better".

- Again, the content to be displayed in the math mode is placed between the opening and closing key sequences.

- Now, when we type `\[55/i = \pi^{-0.3}\]`, we get

$$55/i = \pi^{-0.3},$$

which is the same as when we type `$$55/i = \pi^{-0.3}$$` and we get

$$55/i = \pi^{-0.3}.$$

- Clearly the appearance of the *display* math is different from the *in-line* math. However, sometimes the differences go beyond indentation and size. Both $\int_0^\infty \sum_{n=1}^\infty \frac{1}{n}\, dx$ and

$$\int_0^\infty \sum_{n=1}^\infty \frac{1}{n}\, dx$$

have the same math mode code, but they are placed in the two different modes. In this examples, the limits of integration are the most obvious difference in how these expressions are rendered.

19 | Superscripts and Subscripts

- To add a superscript to an expression use a hat, "`^`", after the term with the exponent. So, `X^3` gives X^3.

- If your exponent has more than one character to be rendered in the case of X^{3+y}, then we must surround the entire exponent in *curly braces* (`{ }`). For this previous expression we'd type `X^{3+y}`. In this sense, the *curly braces* group terms. When in doubt, use curly braces to set the scope of your exponent. Had we omitted the curly braces, we'd see $X^3 + y$ which is quite different.

- The principle for subscripting is identical, but we use the underscore, "`_`", instead. Typing `X_{3+y}` gives X_{3+y}.

- Super– and subscripting can be recursive. However, one must be very careful with braces in this case. We can typeset e^{x_i} with `e^{x_{i}}`. Now, the braces around the `i` term are optional. Remove them, what happens to the rendering? Remove the braces around the x_i term. Compile the document. What happens to the rendering?

- Both superscripts and subscripts can be used simultaneously. In Social Choice, we often care about the median voter's preference, denoted x_i^m. This can be achieved with either `x_i^m` or `x_{i}^{m}`. The latter is more typing, but forces you to be explicit about the scope which tends to reduce the number of errors.

20 | Operators

$+$	`+`	$\bigcup$	`\bigcup`	$\sim$	`\sim`
$=$	`=`	$\bigcap$	`\bigcap`	$\approx$	`\approx`
$-$	`-`	$\forall$	`\forall`	$\Leftrightarrow$	`\Leftrightarrow`
$/$	`/`	$\exists$	`\exists`	$\leftrightarrow$	`\leftrightarrow`
$<$	`<`	$\subset$	`\subset`	$\Leftarrow$	`\Leftarrow`
$>$	`>`	$\subseteq$	`\subseteq`	$\leftarrow$	`\leftrightarrow`
$\leq$	`\leq`	$\subsetneq$	`\subsetneq`	$\Rightarrow$	`\Rightarrow`
$\geq$	`\geq`	$\supset$	`\supset`	$\rightarrow$	`\rightarrow`
$\vee$	`\vee`	$\supseteq$	`\supseteq`	$/$	`\not`
$\wedge$	`\wedge`	$\supsetneq$	`\supsetneq`	$\neg$	`\neg`
$\bigvee$	`\bigvee`	$\int$	`\int`	$\div$	`\div`
$\bigwedge$	`\bigwedge`	$\sum$	`\sum`	∞	`\infty`
$\cup$	`\cup`	$\prod$	`\prod`	$\emptyset$	`\emptyset`
$\cap$	`\cap`	∂	`\partial`	$\in$	`\in`

21 | Greek Letters

- Greek letters are uncomfortably common in the work we do, so you will get used to including them in your documents.

- Although there are neither ζ nor χ keys on your keyboard, we can include these symbols in the math environment in a very intuitive way. Type `\name-of-letter` or `\Name-of-letter` for the lower-case and upper-case version of the Greek letter.

- So, `\delta` and `\Delta` give δ and Δ, respectively.

- Not every letter has a special upper-case version provided. Although `\beta` gives β `\Beta` is undefined.

- Not every letter has a special character, regardless of case. For example, the Greek omicron is just like the Roman 'o', so o gives o which will be sufficient.

- With Greek letters a particular issue arises often, although it isn't specific to Greek letters. LATEX commands do not need a leading space before the \. Yet, they do need a trailing space so that LATEX knows the name of the command is over. So, `X\beta` ($X\beta$) is rendered identically to `X \beta` ($X\beta$) because LATEX has its own way of interpreting whitespace. However, `X\beta + \epsilon` ($X\beta + \epsilon$) works whereas `X \beta+ \epsilon` will not because `\beta+` is an undefined control sequence (i.e. we just made up that command).

22 | Other Letter-y Things

Sometimes we need to present change the presentation of standard letters or symbols in math mode. Here are several common examples. The way these work is that whatever symbols are to have their face changed are passed to these commands as arguments. Notice how the "math" faces remove white space

Modification	Command	Example	Common Uses
Normal Math Face	—	$ABCXY$	most maths
Roman Face	`\textrm{}`	ABC XY	text within equation
Bold Math Face	`\mathbf{}`	**ABCXY**	vectors and matrices
Blackboard Math	`\mathbb{}`	$\mathbb{ABCXY}$	special sets of numbers
Calligraphic Math	`\mathcal{}`	$\mathcal{ABCXY}$	arbitrary sets

23 | Special Text-like Mathematical Expressions

Certain functions, operators, and constructs pop up in math frequently which are basically abbreviates for words. The cosine function—as in $\cos(\pi) = 1$—and the limit of an expression—as in $\lim_{x\to\infty}\frac{1}{x}$—are two such examples. If we just typed `$cos$` (cos) or `$lim$` (lim) we get results quite different. An incomplete list of the expressions for which the control sequences are defined is below.

det `\det`	min `\min`	cos `\cos`	exp `\exp`
lim `\lim`	inf `\inf`	sin `\sin`	Pr `\Pr`
max `\max`	sup `\sup`	tan `\tan`	arg `\arg`

24 | Delimiters

Many times, we need to express groupings through the use of delimiters. For example, $(100 - 100)^{100}$ is quite different than $100 - 100^{100}$. For parentheses we can just enter `(x)`. For curly braces we enter `{x}`. And, lastly, for square brackets we use `[x]`. These produce (x), $\{x\}$, and $[x]$, respectively. In longer expressions, though, the results can look funny. In

$$[\sum_{x=1}^{3}\{(\int_0^x (e^y \, dy))\}],$$

the size of the delimiters is off. However, in

$$\left[\sum_{x=1}^{3}\left\{\left(\int_0^x (e^y \, dy)\right)\right\}\right],$$

the problems seem to be gone. If we type `\left(`, `\left\{`, or `\left[` and close the expression off with the approprate `\right)`, `\right\}`, or `\right]`, the sizes are determined by the size of the internal expression as opposed to being constant.

25 | Accents

Often times we need to modify slightly a character it denote that the construct has slightly changed. For example, we want

- not the true parameter β but the estimate $\hat{\beta}$,

- not the random variable y, but the sample mean $\bar{y}$, or

- not the singleton x, but a vector $\vec{x}$.

A partial list of these accents is below.

$\acute{a}$ `\acute{a}`	$\dot{e}$ `\dot{e}`	$\tilde{\imath}$ `\tilde{\imath}`
$\bar{b}$ `\bar{b}`	$\ddot{f}$ `\ddot{f}`	$\vec{\jmath}$ `\vec{\jmath}`
$\breve{c}$ `\breve{c}`	$\grave{g}$ `\grave{g}`	$\widehat{xyz}$ `\widehat{\xyz}`
$\check{d}$ `\check{d}`	$\hat{h}$ `\hat{h}`	$\widetilde{xyz}$ `\widetilde{\xyz}`

26 | Assignment!

Questions

Write LaTeX code for the following expressions. Confirm that your compiled document matches these.

1.

$$\neg\left((T \vee F) \wedge F\right) \vee (T)$$

2.

$$e^{ix} = \cos x + i \sin x$$

3.

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \epsilon_i$$

4.

$$\left(\sum_{x=1}^{\infty}(1/x)\right) - \left(\sum_{x=1}^{\infty}(1/x)^2\right)$$

Answers

1.

```
\[
\neg\left(\left(T \vee F\right)
\wedge F\right) \vee \left( T \right)
\]
```

2.

```
\[
e^{ix} = \cos x + i \sin x
\]
```

3.

```
\[
y_i = \beta_0 + \beta_1 x_i +
\beta_2 x_i^2 + \epsilon_i
\]
```

4.

```
\[
\left(\sum_{x=1}^{\infty} (1/x) \right) -
\left(\sum_{x=1}^{\infty}
(1/x)^2 \right)
\]
```