A round up of LaTeX one page guides

Each of the following pages is dedicated to a single LaTeX topic. I chose the one page requirement because it forces me to produce a concise and focused document, and because I imagine it makes it easy for the student who's looking for a specific topic. On the other hand, I've merged all the single page documents into one because sometimes you want everything together.

Contents

Intro to LaTeX in one page

W. Ethan Duckworth, Loyola University Maryland, 2014

- LaTeX is a computer language, and the name of the program that converts that language to printed (or printable) output.

  To use LaTeX you first create an input file, which looks like a file of computer code: it is formatted as simple, plain text, i.e. with no fonts, pictures, margins, etc. The input file contains ordinary words and commands that will be interpreted by the program to produce the output. You create this file in a text editor like Emacs, or Notepad, or a front end like TeXShop, TeXMaker, or writeLaTeX (which each offer a built-in text editor).

  Then you run the program LaTeX, which takes your input file and produces output, typically a PDF file (for Adobe Reader).

- Here's a minimal LaTeX file: everything here (except for "`hello world`") needs to be in every file you make

  Example 1. Input:

```
\documentclass{article}
\begin{document}
Hello world.
\end{document}
```

  Output:

  Hello world.

- Spaces and paragraphs. This example shows how spaces in the input don't necessarily create spaces in the output: paragraphs

  Example 2. Input:

```
Words
automatically
wrap
to fill the left and right edges.
It doesn't  matter where you put
your spaces       or if you hit
return once
followed by more text.

Hitting return twice, leaving a
blank line, starts a new paragraph.
```

  Output:

  Words automatically wrap to fill the left and right edges. It doesn't matter where you put your spaces or if you hit return once followed by more text.

  Hitting return twice, leaving a blank line, starts a new paragraph.

- All math formulas should be typed inside math mode, which you turn on and off by typing certain characters in your input. The simplest way to turn math on and off is to type $. Try the following simple example.

  Example 3. Input:

```
\documentclass{article}
\begin{document}
The sum $3+7$ equals $10$.  Note that $0
\times x=0$ for all $x$.  If we get a
common denominator then we can see that $
\frac 12 + \frac 35 = \frac{5}{10} + \frac
{6}{10} = \frac{11}{10}$.
\end{document}
```

  Output:

  The sum $3 + 7$ equals 10. Note that $0 \times x = 0$ for all x. If we get a common denominator then we can see that $\frac{1}{2} + \frac{3}{5} = \frac{5}{10} + \frac{6}{10} = \frac{11}{10}$.

- Of course, LaTeX is more fun with more complicated examples. Try entering the following to see how it works.

  Example 4. Input:

```
\documentclass{article}
\begin{document}
The solution of $ax^2+bx+c=0$ is given by
$$
x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}.
$$
But the solution of
$$
a\int e^{-x^2}\,dx + b \sum \frac{1}{x^2} +
 c\sqrt{x}=0.
$$
is unknown.
\end{document}
```

  Output:

  The solution of $ax^2 + bx + c = 0$ is given by

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

  But the solution of

$$a \int e^{-x^2}\,dx + b \sum \frac{1}{x^2} + c\sqrt{x} = 0.$$

  is unknown.

- As shown in the previous two examples, you should almost never have any regular text inside of math mode. To mix math and text you should almost always alternate, turning math mode on and then off.

- Other things you may want to learn about:
  - Controlling fonts.
  - What punctuation symbols do.
  - Page formatting.
  - Writing math.
  - Lots of symbols.
  - Tables and equations.
  - Lists
  - Cross referencing.
  - Defining new commands.

LaTeX: Math and Text in one page

W. Ethan Duckworth, Loyola University Maryland, 2014

- Single dollar sign math: text style. By default, whatever you enter in LaTeX is treated as text. A relatively small percentage of your entire document should have $...$ around it to put it in math mode.
  - The first $ turns math on, the second $ turns it off.
  - Most useful for including math within the line of text.

Example 1. Input:

```
Define the relation $R$ on $Z$ such that,
for any $a,b\in Z$, we have $aRb$ if and
only if the distance between $a$ and $b$ is
 at most $2$.
```

Output:

Define the relation R on Z such that, for any $a, b \in Z$, we have aRb if and only if the distance between a and b is at most 2.

- Double dollar sign math: display style. A relatively small percentage of your math material should have $$...$$ around it.
  - The first $$ turns math on, the second $$ turns math off.
  - Contents show up in "display style" (see next bullet point for what this means).

Example 2. Input:

```
Applying integration by parts to $\int x
\cos(x)\,dx$ we see that we get
$$
\int x\cos(x)\,dx = x\sin(x)-\int \sin(x)\,
dx,
$$
which equals $x\sin(x)+\cos(x)+C$.
```

Output:

Applying integration by parts to $\int x\cos(x)\,dx$ we see that we get

$$\int x\cos(x)\,dx = x\sin(x) - \int \sin(x)\,dx,$$

which equals $x\sin(x)+\cos(x)+C$.

- Text versus display. "Display style" means both where the formula is shown (centered, on a line by itself) and how the formula is sized and arranged (some symbols are bigger; limits are placed above and below). Some examples of text style versus display style:

Fractions: $\frac{a}{b}$ versus $\displaystyle\frac{a}{b}$.

Integrals: $\int_a^b$ versus $\displaystyle\int_a^b$.

Sums: $\sum_{n=1}^{\infty}$ versus $\displaystyle\sum_{n=1}^{\infty}$.

Limits: $\lim_{n\to\infty} a_n$ versus $\displaystyle\lim_{n\to\infty} a_n$.

- Forcing text style or display style.
  - Include `\textstyle` within your mathematical formula to force text style.
  - Include `\displaystyle` within your mathematical formula to force display style.
  - Include `\everymath{\displaystyle}` to force display style everywhere in the current scope.

Example 3. Input:

```
The formula $\displaystyle\frac12\sum_{n
=1}^{10} n^2$ is forced one way.  The
formula
$$
\textstyle \frac 12\sum_{n=1}^{10} n^2
$$
is forced the other way.
```

Output:

The formula $\displaystyle\frac12\sum_{n=1}^{10} n^2$ is forced one way. The formula

$$\textstyle \frac 12\sum_{n=1}^{10} n^2$$

is forced the other way.

- Text within mathematics. Do you absolutely have to include some text within a math formula? You might, for instance inside a display style forumula, but don't do it for than a word or two at a time. Use the `\text` command (and load the package `amstext`).

Example 4. Input:

```
$$
A = \{x\in Z : x^2 \ge 10 \text{ and }x^2
\le 100\}.
$$
```

Output:

$$A = \{x \in Z : x^2 \ge 10 \text{ and } x^2 \le 100\}.$$

- Writing style guidelines.
  1. Every letter that stands for a variable, every letter that stands for a set, every mathematical expression, every minus sign, etc., must be in math mode, not ordinary text mode.
  2. Every mathematical expression (even things in display mode) needs to be part of an English sentence that begins with a capitalized English word and ends with a period.
  3. Don't try to control every line by putting `\\` in to end the line, or putting every statement of your proof on a line by itself. Type mostly sentences, with a relatively small percentage of stuff in $...$ and an even smaller percentate in $$...$$, and let LaTeX wrap the lines, which, after all, are mostly words.

LaTeX: Punctuation Symbols and Their Meaning

W. Ethan Duckworth, Loyola University Maryland, 2014

- Ordinary characters. All the letters, a–z, A–Z, and numbers 0–9 produce ordinary output, just what you'd expect. Also

  ! @ * ( ) [ ] / ? . , | : ; − + =

  Note that < and > are only half ordinary: they work, as shown here, but only in math mode. Outside of math mode you get ¡ and ¿.

- Special input characters:

  \ starts each command: adding \ to the beginning of `underline{this}` makes this. You can print \ with `\textbackslash`.

  $ turn text-mode math on/off: "`$x^2$`" makes x^2. You can print $ with `\$`

  $$ turn display math on/off: "`$$x^2$$`" makes

  $$x^2$$

  You can print $$ with `\$\$`.

  % internal comment: entering "`% you can't see this`" makes . You can print % with `\%`.

  _ subscript: "`$x_1$`" makes x_1. You can print _ with `\_`.

  ^ superscript: "`$x^2$`" makes x^2. You can print ^ with `\textasciicircum`.

  & column marker in arrays/tables. For instance,

  ```
  \begin{tabular}{cc}
   a & b \\
   c & d
  \end{tabular}
  ```

  makes $\begin{tabular}{cc} a & b \\ c & d \end{tabular}$. You can print & with `\&`.

  # stands for command arguments when you define a new command. For instance,

  ```
  \newcommand{\foo}[1]{$x^{#1}$}
  ```

  followed by `\foo{2}` makes x^2. You can print # with `\#`.

  { } group command inputs. For instance, `$x_123$` makes x_123, but `$x_{123}$` makes x_{123}. You can print { and } with `\{` and `\}`.

  ~ creates a nonbreaking space. For instance, `Mr.~Doe` makes Mr. Doe, but will never allow a line break between Mr. and Doe.

- Quotation marks. LaTeX makes fancy looking quotation marks.

  Example 1. Input:

  ```
  The marks on the left and right look
  different like ``this''.
  ```

  Output:

  The marks on the left and right look different like "this".

  Your text editor should automatically insert the correct marks at the correct time. In other words, if you just hit shift-[" / '], then it will probably insert the right kind of quotation marks.

  However, if you have to enter the quotion marks manually, note that " is made by hitting [~ / `] twice, and " is made by hitting [" / '] twice (or entering shift-[" / '] once).

- Other fancy symbols.

  - I've made a one page quick reference guide to the most common symbols. You'll see there how to make things like $\int$, $\sqrt{x}$, $\sum \frac{1}{n^2}$, etc.
  - LaTeX has access to a few thousand special symbols. Google "The Comprehensive LATEX Symbol List" to see more.

A one page symbol guide for LaTeX 2ε (almost all need math mode)

Binary relations and operations

$+$	`+`	$\cap$	`\cap`	$\times$	`\times`	$\perp$	`\perp`	$\bigcap$	`\bigcap`	$\sum$	`\sum`
$-$	`-`	$\cup$	`\cup`	$\div$	`\div`	$\in$	`\in`	$\bigcup$	`\bigcup`	$\prod$	`\prod`
$>$	`>`	$\wedge$	`\wedge`	$*$	`\ast`	$\ni$	`\ni`				
$<$	`<`	$\vee$	`\vee`	$\circ$	`\circ`	$\propto$	`\propto`				
$\leq$	`\le`	$\subset$	`\subset`	$\otimes$	`\otimes`	$\approx$	`\approx`				
$\geq$	`\ge`	$\subseteq$	`\subseteq`	$\oplus$	`\oplus`	$\cong$	`\cong`				
$\neq$	`\ne`	$\supset$	`\supset`	$\circ$	`\circ`	$\equiv$	`\equiv`				
$\cdot$	`\cdot`	$\supseteq$	`\supseteq`	$\bullet$	`\bullet`	$\pm$	`\pm`				

Variable-size symbols (these take limits)

$\int$	`\int`	$\bigoplus$	`\bigoplus`	$\bigotimes$	`\bigotimes`		

Some trigonometric and similar symbols

sin	`\sin`	lim	`\lim`	inf	`\inf`
cos	`\cos`	log	`\log`	dim	`\dim`
tan	`\tan`	gcd	`\gcd`	ker	`\ker`
ln	`\ln`	sup	`\sup`	arg	`\arg`

Greek letters

α	`\alpha`	ϑ	`\vartheta`	σ	`\sigma`	Γ	`\Gamma`
β	`\beta`	ι	`\iota`	τ	`\tau`	Δ	`\Delta`
γ	`\gamma`	κ	`\kappa`	υ	`\upsilon`	Θ	`\Theta`
δ	`\delta`	λ	`\lambda`	ϕ	`\phi`	Λ	`\Lambda`
ϵ	`\epsilon`	μ	`\mu`	φ	`\varphi`	Ξ	`\Xi`
ε	`\varepsilon`	ν	`\nu`	χ	`\chi`	Π	`\Pi`
ζ	`\zeta`	ξ	`\xi`	ψ	`\psi`	Σ	`\Sigma`
η	`\eta`	π	`\pi`	ω	`\omega`	Υ	`\Upsilon`
θ	`\theta`	ρ	`\rho`			Φ	`\Phi`
						Ψ	`\Psi`
						Ω	`\Omega`

Some arrows

$\leftarrow$	`\leftarrow`	$\Rightarrow$	`\Rightarrow`	$\hookrightarrow$	`\hookrightarrow`
$\Leftarrow$	`\Leftarrow`	$\Longrightarrow$	`\Longrightarrow`	$\leftrightarrow$	`\leftrightarrow`
$\longleftarrow$	`\longleftarrow`	$\uparrow$	`\uparrow`	$\Leftrightarrow$	`\Leftrightarrow`
$\Longleftarrow$	`\Longleftarrow`	$\Uparrow$	`\Uparrow`	$\Longleftrightarrow$	`\Longleftrightarrow`
$\rightarrow$	`\rightarrow`	$\mapsto$	`\mapsto`		
$\to$	`\to`	$\longmapsto$	`\longmapsto`		

Miscellaneous symbols

$\ldots$	`\dots`	$\ddots$	`\ddots`	$\forall$	`\forall`	∂	`\partial`
$\cdots$	`\cdots`	$\neg$	`\neg`	$\triangle$	`\triangle`	$\therefore$	`\therefore`
$\ldots$	`\ldots`	$\sim$	`\sim`	∇	`\nabla`	$\emptyset$	`\emptyset`
$\vdots$	`\vdots`	$\exists$	`\exists`	$\angle$	`\angle`		

Constructions

X_{22}^{11}	`X_{22}^{11}`
$\frac{abc}{def}$	`\frac{abc}{def}`
$\widetilde{abc}$	`\widetilde{abc}`
$\overline{abc}$	`\overline{abc}`
$\underline{abc}$	`\underline{abc}`
$\widehat{abc}$	`\widehat{abc}`
$\sqrt{abc}$	`\sqrt{abc}`
$\sqrt[n]{abc}$	`\sqrt[n]{abc}`

$\underbrace{abcdef}$	`\underbrace{abcdef}`
$\underbrace{abcdef}_{n}$	`\underbrace{abcdef}_{n}`
$\overbrace{abcdef}$	`\overbrace{abcdef}`
$\overbrace{abcdef}^{n}$	`\overbrace{abcdef}^{n}`
$\left\{\int x^2\,dx\right\}$	`\left\{\int x^2\,dx\right\}`
f'	`f'`
$a \not< b, a \notin A$, etc.	`a\not < b, a\notin A, etc.`

Delimiters (resizeable with `\left` and `\right` as above)

output	input
{ }	`\{ \}`
( )	`( )`
[ ]	`[ ]`
$\|$ $\|$	`|` `|`
⟨ ⟩	`\langle` `\rangle`

Some fancy fonts (use `\usepackage{amsfonts}`)

$\mathfrak{A}, \mathfrak{B}, \mathfrak{C}, \ldots, \mathfrak{a}, \mathfrak{b}, \mathfrak{c}, \ldots$	`\mathfrak{A,B,C,...,a,b,c,...}`
$\mathcal{A}, \mathcal{B}, \mathcal{C}, \ldots$	`\mathcal{A,B,C,...}`
$\mathbb{A}, \mathbb{B}, \mathbb{C}, \ldots$	`\mathbb{A,B,C,...}`

LaTeX font control

W. Ethan Duckworth, Loyola University Maryland, 2014

The standard warning applies to most of the commands here: if you are entering them very often within a single document, then you are doing it wrong. Most of the time changes are made automatically, for instance in math mode, or section headings. If you need `\LARGE \mathbb{R}` more than once, you should create your own command `\R` that has the font commands built in.

- Whole document default font size:
  `\documentclass{article}` = 10pt
  `\documentclass[11pt]{article}` = 11pt
  `\documentclass[12pt]{article}` = 12pt

- Font size for part of document.

  {\tiny Example} (that's \tiny !)
  {\scriptsize Example}
  {\footnotesize Example}
  {\small Example}
  {\normalsize Example}
  {\large Example}
  {\Large Example}
  {\LARGE Example}
  {\huge Example}
  {\Huge Example}

- Font styles

  Input:
  ```
  \textrm{Roman}
  \textbf{Bold face}
  \textit{Italic}
  \textsc{Small Cap}
  \textsf{Sans Serif}
  \texttt{Teletype}
  \textsl{Slanted}
  ```

  Output:

  Roman
  Bold face
  Italic
  Small Cap
  Sans Serif
  `Teletype`
  Slanted

- Emphasizing. Don't use `\textit`, use `\emph`.

  Input:
  ```
  \emph{This} word is emphasized.
  ```

  Output:

  This word is emphasized.

- Underlining.

  - There is a built in command `\underline{easy to use}` that's <u>easy to use</u>. But this command won't underline across line breaks.
  - Or you can use the add-on package `ulem` (which also defines double underline, strike-through, etc).

    Example 1. Input:

    ```
    \documentclass{article}
    \usepackage[normalem]{ulem}
    \begin{document}
    \uline{This will underline the text and
     allow it to break across the line.}
    \end{document}
    ```

    Output:

    <u>This will underline the text and allow it to break across the line.</u>

- Some font alphabets.

	Input
Greek	`$\alpha$, $\beta$, \dots, $\omega$`
Caligraphic	`$\mathcal{A,B,\dots,Z}$`
Script	`$\mathscr{A,B,\dots,Z}$`
Blackboard	`$\mathbb{A,B,\dots,Z}$`
Fraktur	`$\mathfrak{A,B,\dots,Z,a,b,\dots,z}$`

Output	Extra package needed
$\alpha, \beta, \ldots, \omega$	none
$\mathcal{A}, \mathcal{B}, \ldots, \mathcal{Z},$	none
$\mathscr{A}, \mathscr{B}, \ldots, \mathscr{Z}$	`\usepackage{mathrsfs}`
$\mathbb{A}, \mathbb{B}, \ldots, \mathbb{Z}$	`\usepackage{amsfonts}`
$\mathfrak{A}, \mathfrak{B}, \ldots, \mathfrak{Z}, \mathfrak{a}, \mathfrak{b}, \ldots, \mathfrak{z}.$	`\usepackage{amsfonts}`

- Whole document font families. If you don't like the default font (Computer Modern Roman), you can put one of the following options (in addition to many others) in your preamble.

  - `\renewcommand{\familydefault}{\sfdefault}` To change the whole document to Computer Modern Sans Serif. The quick brown fox jumped over the lazy dog.
  - `\usepackage{helvetica}` To change the sans serif font to Helvetica. The quick brown fox jumped over the lazy dog.
  - `\usepackage{times}` To change the whole document to Times Roman. The quick brown fox jumped over the lazy dog.
  - `\usepackage{palatino}` To change the whole document to Palatino. The quick brown fox jumped over the lazy dog.
  - `\usepackage{charter}` To change the whole document to Charter. The quick brown fox jumped over the lazy dog.
  - `\usepackage{bookman}` To change the whole document to Bookman. The quick brown fox jumped over the lazy dog.
  - `\usepackage{chancery}` *To change the whole document to Chancery. The quick brown fox jumped over the lazy dog.*

LaTeX Lists

W. Ethan Duckworth, Loyola University Maryland, 2014

- `itemize`: Lists without counting.

  Example 1. Input:

```
\begin{itemize}
\item apple
\item berry
\begin{itemize}
\item strawberry
\item blueberry
\end{itemize}
\end{itemize}
```

  Output:

  - apple
  - berry
    - strawberry
    - blueberry

  Most of this page is formatted in an `itemize` list.

- `enumerate`: lists with counting.

  Example 2. Input:

```
\begin{enumerate}
\item apple
\item berry
\begin{enumerate}
\item strawberry
\item blueberry
\end{enumerate}
\end{enumerate}
```

  Output:

  1. apple
  2. berry
     (a) strawberry
     (b) blueberry

- `description`: lists headed by a word

  Example 3. Input:

```
\begin{description}
\item[apple] a crisp, red or green fruit.
\item[berry] a small, succulent fruit.
\begin{description}
\item[strawberry] a ``false'' berry that is
 red, oblong, and covered
  in seeds.
\item[blueberry] a true berry that is blue,
 spherical, and contains a
  relatively small number of seeds.
\end{description}
\end{description}
```

  Output:

  apple a crisp, red or green fruit.
  berry a small, succulent fruit.
  strawberry a "false" berry that is red, oblong, and covered in seeds.
  blueberry a true berry that is blue, spherical, and contains a relatively small number of seeds.

- Customizing labels for enumerate, using built-in commands.

  Example 4. Input:

```
\begin{enumerate}
\renewcommand{\labelenumi}{\roman{enumi}.}
\renewcommand{\labelenumii}{\Alph{enumii}.}
\item apple
\item berry
\begin{enumerate}
\item strawberry
\item blueberry
\end{enumerate}
\end{enumerate}
```

  Output:

  i. apple
  ii. berry
      A. strawberry
      B. blueberry

- Customizing labels for `itemize`, using built-in commands.

  Example 5. Input:

```
\begin{itemize}
\renewcommand{\labelitemi}{$\circ$}
\renewcommand{\labelitemii}{$\dagger$}
\item apple
\item berry
\begin{itemize}
\item strawberry
\item blueberry
\end{itemize}
\end{itemize}
```

  Output:

  - apple
  - berry
    - † strawberry
    - † blueberry

- Customizing labels using the `enumitem` package.

  The previous examples can be recreated as follows

```
\usepackage{enumitem}
\setenumerate[1]{label=\roman*.}
\setenumerate[2]{label=\Alph*.}
\setitemize[1]{label=$\circ$}
\setitemize[2]{label=$\dagger$}
\end{enumitem}
```

- Customizing lengths using the `enumitem` package.

  The following will change the length of various vertical spaces to 0 inches.

```
\usepackage{enumitem}
\setlist{itemsep=0in, parsep=0in, topsep=0
in}
```

LaTeX: Manual Spacing in One Page

W. Ethan Duckworth, Loyola University Maryland, 2014

Note that like most formatting commands, you should not be entering the following commands very often. You should rely on previously defined environments, or create your own new environments, with the spacing built in.

- Horizontal spacing

  Input:

```
a$\!$b      ab      a\,b      a\ b      a\quad b      a\qquad b      a\hspace{0.5in}b
```

  Output:

a$\!$b ab a\,b a\ b a\quad b a\qqquad b a\hspace{0.5in}b

  Three things to note: (1) `\!` only works in math mode. (2) `\hspace` is continuous, you can plug in any lengh you want. (3) to make `\hspace` work at the beginning of a line you need to use `\hspace*`.

- Vertical spacing

  Input:

```
end.

Begin

end.
\smallskip

Begin

end.
\medskip

Begin

end.
\bigskip

Begin

end.
\vspace{0.33in}

Begin
```

  Output:

end.

Begin

end.

Begin

end.

Begin

end.

Begin

end.

Begin

  Four things to note: (1) "end" and "Begin" represent the end and beginning of paragraphs. The commands shown here need to appear in the gap between two paragraphs. (2) `\vspace` can take any length as it's input. (3) To make `\vspace` work at the top of the page you need to use `\vspace*`.

- Flexible horizontal spacing: stretches to make equal spacing.

  Input:

```
(a) \hfill (b) \hfill (c) \hfill (d)\hfill \mbox{}
```

  Output:

(a) \hfill (b) \hfill (c) \hfill (d)\hfill \mbox{}

  To make `\hfill` work at the *end* of a line you need to put an invisible box after it.

- Flexible vertical spacing: stretches to make equal spacing.

  Input:

```
\mbox{}
\vfill

(a)
\vfill

(b)
\vfill

(c)
\vfill
```

  Output:

\mbox{}
\vfill

(a)
\vfill

(b)
\vfill

(c)
\vfill

  To make `\vfill` work at the *beginning* of the page you need to put an invisible box before it. One application of this command is to make a title page, using `\mbox{}\vfill ***** \vfill` where ***** would be replaced by your title.

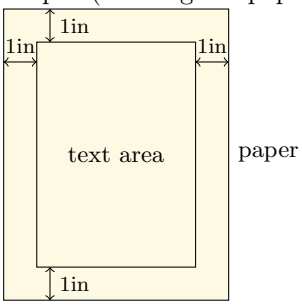
LaTeX page formatting

W. Ethan Duckworth, Loyola University Maryland, 2014

- Setting margins with built in lengths.
  Input (for 8.5×11 paper):

```
\topmargin=0in
\headheight=0in
\headsep=0in
\textheight=9in
\oddsidemargin=0in
\textwidth=6.5in
```

  Output (showing the paper and text area):



  Note three things. (1) The left margin is controlled by `\oddsidemargin`, the right margin is defined implicitly by adding the left margin and the text width. (2) The top margin is conrolled by `\topmargin` (and `\headheight` and `\headsep`), the bottom margin is defined implicitly by these lengths and `\textheight`. (3) The left margin and the top margin are measured from a default position of 1 inch. Thus, `\oddsidemargin=0in` results in 1 inch real margin, `\oddsidemargin=-0.5in` results in 0.5 inch real margin, etc.

- Margins with `geometry` package:
  `\usepackage[text={6.5in,9in}]{geometry}`
  one inch margins on left, right, top and bottom.

- Paragraphs: indentation and space between.
  `\parindent=0.1in` paragraph indentation = 0.1 inches.
  `\parskip=0.2in` space between paragraphs = 0.2 inches.

- Line spacing with built in commands (put in preamble)
  For one and a half spacing:

```
\renewcommand{\baselinestretch}{1.25}
```

  For double spacing:

```
\renewcommand{\baselinestretch}{1.66}
```

- Line spacing with package `setspace`
  For one and a half spacing:

```
\usepackage[onehalfspacing]{setspace}
```

  For double spacing:

```
\usepackage[doublespacing]{setspace}
```

- Two columns for whole document:
  `\documentclass[twocolumn]{article}`

- Two columns for current and following pages:
  `\twocolumn`

- n columns for any region using `multicol` package.

Example 1.

```
\documentclass{article}
\usepackage{multicol}
\begin{document}
This is one column of text all the way
across the page.
\begin{multicols}{2}
This text will go into two columns once we
get enough text so that it will wrap and
fill both columns.  This is meaningless but
 I must keep going.
\end{multicols}
\end{document}
```

Output:

This is one column of text all the way across the page.

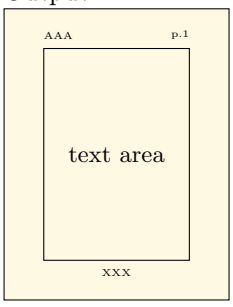
This text will go into two columns once we get enough text so that it will wrap and fill both columns. This is meaningless but I must keep going.

- Page numbers.
  `\pagestyle{empty}` causes no page number to print
  `\pagestyle{plain}` puts page number at bottom center
  `\pagestyle{headings}` puts page number at top right (and section headings top left)

- Headings with `fancyhdr` package
  Input:

```
\documentclass{article}
\usepackage{fancyhdr}
\pagestyle{fancy}
\lhead{AAA}
\rhead{p.\thepage}
\cfoot{XXX}
\renewcommand{\headrulewidth}{0pt}
\begin{document}
```

  Output:



  There are also commands for `\chead`, `\lfoot` and `\rfoot`

LaTeX tables and equations

W. Ethan Duckworth, Loyola University Maryland, 2014

- `tabular` makes something with rows and columns.

Example 1. Input:

```
\begin{tabular}{rcll}
  xxx & yyyy & zzzz & wwww \\
  t   & u    & v    & s
\end{tabular}
```

Output:

xxx	yyyy	zzzz	wwww
t	u	v	s

(1) The letters {rcll} specify how many columns there are (four, one for each letter) and how each is justified (r = right, c = center, l = left). (2) Separate each column with & and each row with \\.

- `array` is just like tabular, but it's used in math mode.

Example 2. Input:

```
$$
\begin{array}{rcll}
  xxx & yyyy & zzzz & wwww \\
  t   & u    & v    & s
\end{array}
$$
```

Output:

$$\begin{array}{rcll} xxx & yyyy & zzzz & wwww \\ t & u & v & s \end{array}$$

- Equations using `array`

Example 3. Input:

```
$$
\begin{array}{rcl}
  x & = & \frac{-2\pm \sqrt{4+4}}{2}\\
    & = & \frac{-2\pm 2\sqrt{2}}{2}\\
    & = & -1\pm \sqrt{2}
\end{array}
$$
```

Output:

$$\begin{array}{rcl} x & = & \frac{-2\pm \sqrt{4+4}}{2}\\ & = & \frac{-2\pm 2\sqrt{2}}{2}\\ & = & -1\pm \sqrt{2} \end{array}$$

- Equations using `align*` (and \usepackage{amsmath})

Example 4. Input:

```
\documentclass{article}
\usepackage{amsmath}
\begin{document}
\begin{align*}
  x & =  \frac{-2\pm \sqrt{4+4}}{2}\\
    & =  \frac{-2\pm 2\sqrt{2}}{2}\\
    & =  -1\pm \sqrt{2}
\end{align*}
\end{document}
```

Output:

$$\begin{aligned} x &= \frac{-2\pm \sqrt{4+4}}{2}\\ &= \frac{-2\pm 2\sqrt{2}}{2}\\ &= -1\pm \sqrt{2} \end{aligned}$$

- Differences between `array` and `align*`. (1) `align*` needs about half as many &'s: you only put them between the left hand side and the = (or + or > etc.). (2) The spacing is somewhat better in `align`. (3) If you use `align` instead of `align*` you get equation numbering. (4) By entering `\allowdisplaybreaks` in the preamble, you can let LaTeX break the page in the middle of the equations. (5) `array` gives you more control over each column (but requires that you describe each column).

- Matrices using `array` with \left[and \right].

Example 5. Input:

```
$$
\left[ \begin{array}{ccc}
         a & b & c \\
         d & e & f \\
         g & h & i
       \end{array}          \right]
$$
```

Output:

$$\left[\begin{array}{ccc} a & b & c \\ d & e & f \\ g & h & i \end{array} \right]$$

- Matrices using `bmatrix` (and \usepackage{amsmath})

Example 6. Input:

```
\documentclass{article}
\usepackage{amsmath}
\begin{document}
$$
\begin{bmatrix}
  a & b & c \\
  d & e & f \\
  g & h & i
\end{bmatrix}
$$
```

Output:

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

LaTeX Cross Referencing

W. Ethan Duckworth, Loyola University Maryland, 2014

- LaTeX can keep track of cross references: things like equation numbers, section numbers, theorem numbers, page numbers, and bibliography references. But it needs your help: you have to label things to keep track of.

- Using `\label`, `\ref` and `\pageref`. Suppose in the following example that we have already had 3 previous sections.

Example 1. Input:

```
\section{How to use labels}
\label{using_labels}
Imagine a lot of stuff here.

\section{Chapter review}
Recall that in
Section~\ref{using_labels}, on
page~\pageref{using_labels}, we learned all
 about everything.
```

Output:

4 How to use labels

Imagine a lot of stuff here.

5 Chapter review

Recall that in Section 4, on page 1, we learned all about everything.

- Two things to note: (1) When you mark something with `\label`, the numbers are stored in an external file named `foo.aux` (where your main file is `foo.tex`). For the numbers to be accurate, you need to `latex` your file twice. (2) It's good, but not mandatory, to enter `~` between things like "Section" and `\ref` so that a space will appear, but LaTeX will not ever break the line here.

- Other things that can be labelled. Imagine in the next example that we have had 17 previous theorems, and 5 previous equations.

Example 2. Input:

```
\begin{theorem}
  \label{parity_results}
  We have:
  \begin{enumerate}
  \item \label{even_even}
    even + even = even
  \item in other words
    \begin{equation}
      \label{general}
      0 + 0  = 0 \mod 2
    \end{equation}
  \end{enumerate}
\end{theorem}

Theorem~\ref{parity_results} had results
about parity.
Part~\ref{even_even} showed that the sum of
 even numbers is even, and Equation~\ref{
general} stated this result in terms of
modular arithmetic.
```

Output:

Theorem 18. *We have:*

1. *even + even = even*
2. *in other words*

$$0 + 0 = 0 \mod 2 \tag{6}$$

Theorem 18 had results about parity. Part 1 showed that the sum of even numbers is even, and Equation 6 stated this result in terms of modular arithmetic.

- Bibliographies.

Example 3. Input:

```
\documentclass{article}
\usepackage{amsrefs}
\begin{document}
You should read
\cite{duckworth1}.

Imagine we're now at the end of the
document.
\begin{bibdiv} % bibliography heading
\begin{biblist} % list of citations
\bib{duckworth1}{article}
{ author = {Duckworth, W. Ethan},
  title = {Using LaTeX},
  journal = {Practical TeX},
  volume = {15},
  number = {111},
  pages = {1734--1745} }
\end{biblist}
\end{bibdiv}
```

Output:

You should read [1].
Imagine we're now at the end of the document.

References

[1] W. Ethan Duckworth, *Using LaTeX*, Practical TeX **15**, no. 111, 1734–1745.

- One thing to note: you would create a `\bib{foo}` entry for each entry in your bibliography.

- Other `amsrefs` features: there are other `\bib` types, such as `book`. There are other keys, such as `editor`, `address` and `date`. There are other options for how the list of references is formatted, such as `alphabetic` or `author-year`. For more information read the AMS References documentation `amsrdoc.pdf`.

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 $\frac{dy}{dx} = 2x$.

  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 $\frac{dV}{dr} = 2\pi r$.

  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}
```

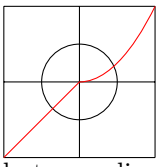
  defines a new environment called `foo`. The number of arguments is n. The definition is broken into two parts: `bar` is what happens when you enter `\begin{foo}` and `baz` is what happens when you enter `\end{foo}`.

LaTeX Graphics

W. Ethan Duckworth, Loyola University Maryland, 2014

- Internal graphics with Tikz.

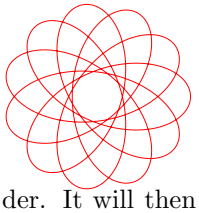
```
\usepackage{tikz}
\begin{document}
\begin{tikzpicture}
\draw (0,0) grid (2,2);
\draw (1,1) circle (0.5cm);
\draw[red] (0,0) -- (1,1) parabola (2,2);
\end{tikzpicture}
```



The package Tikz is part of the larger package PGF. In this package you describe the picture using commands and a kind of programing language. It is powerful, and produces good looking results, but complicated pictures may require more work than you want to do. For more information see the user manual for PGF.

- External graphics.

```
\usepackage{graphicx}
\begin{document}
\includegraphics[width=1in]{hypocycloid}
```



In the example shown, LaTeX will look for a file `hypocycloid` in the same directory as the main LaTeX input file. More specifically, it will look for `hypocycloid.png`, `hypocycloid.pdf`, or `hypocycloid.jpg`, in this order. It will then include the graphic file in the output, scaling it to have a width of 1 inch.

All of this assumes that the graphics file `hypocycloid` has been previously created, probably by some other program. Also, for the file types described, I've assumed that you're using `pdflatex`, i.e. you're set to create PDF files directly from your LaTeX file. For more information see the user manual for `graphics`, or the file `epslatex`, available online.

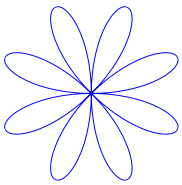
- EPS files. Using `pdflatex` you cannot directly include EPS (Encapsulated PostScript) files. But, in most current LaTeX versions they can be converted automatically, in the background, to PDF files, which are then included. The resulting PDF file gets a name such as `foo-eps-converted-to.pdf`. For more information see the `epstopdf` package or the TeXLive documentation.

- Positioning graphics. Most of the time you should put a graphic between paragraphs, maybe in `\begin{center}... \end{center}`.

- Floating graphics. In professionally made books, the pictures are allowed to float. The purpose of floating is to prevent a picture from going off the bottom of the page or creating large gaps of space. (The other way to prevent this is to manually move text and graphics around, which isn't feasible with long documents.) Roughly speaking, instead of telling LaTeX "put this graphic here" you say "put this graphic somewhere on this page" and then you tell your reader "The graphic in Figure 1 shows the answer". For more information, see the file `epslatex`, available online.

Figure 1: Input and output of a figure

```
  Believe it or not, \emph{this}
\begin{figure}
\caption{Input and output of a figure}
\label{figure:graph_answer}
\includegraphics[width=1in]{rose_petal}
\end{figure}
is where I entered the code shown in Figure~
\ref{figure:graph_answer}
above.
```



Believe it or not, *this* is where I entered the code shown in Figure 1 above.

- Controlling floats. Floats position figures (and tables, and other things) automatically, but sometimes you'd like to modify or overrule the automatic options. Various settings and packages let you do this. Read the page `http://www.tex.ac.uk/cgi-bin/texfaq2html?label=floats` for more information.

- It's also possible to wrap text around a figure, as in the first two pictures in this document, although this does not work automatically in lists. See the package `wrapfig` for documentation.

- You'll get the best looking results by using graphics made in some high quality, vector format such as EPS, or produced by Tikz, or in PDF provided this was made from a vector graphic format. In other words, anything *except* JPEG, TIFF, GIF, screenshots, or any other bitmap format. Next to the rest of the text produced by LaTeX, all bitmap formats will look inferior.

- Pretty much everything I make comes from two graphics sources: Tikz and Maple, a computer algebra program.

  I use Tikz when the picture is relatively simple. I use Maple when the picture involves specific functions or calculations. It's possible to combine Tikz and Maple in a single picture, but I usually create each picture entirely in one or the other.

- One final recommendation: Do your graphics in such a way that it's easy to make changes and revisions later. To wit: save the Maple code, work in batch mode when possible, give the files meaningful names.

A Simple Set of LaTeX Custom Commands

W. Ethan Duckworth, Loyola University Maryland, 2014

Here is a sampling of commands I've created in the context of classroom material. They are fairly simple and pretty typical of the kind of commands most users of LaTeX create. Most of them are shortcuts: they make it easier to enter code, read the code, and adjust the formatting of the input.

```
% Taking derivatives
\newcommand{\dd}[2]{\dfrac{d#1}{d#2}}
\newcommand{\dydx}{\dd yx}
\newcommand{\ddx}{\dd {}x}
\newcommand{\ddt}{\dd {}t}

% Taking partial derivatives
\newcommand{\pp}[2]{\frac{\partial#1}{\partial
#2}}
\newcommand{\ppx}{\pp{}x}
\newcommand{\ppy}{\pp{}y}
\newcommand{\ppz}{\pp{}z}

% evaluating anti-derivatives
\newcommand{\eval}{\Big|}

% formatting some important single letters
\newcommand{\e}{\mbox{\large$e$\rule{0in}{1.5ex
}}}
\renewcommand{\epsilon}{\varepsilon}
\renewcommand{\phi}{\varphi}

% Labelling L'Hospital's Rule
\newcommand{\LH}{\stackrel{\text{LH}}{=}}
% Asking if two things are equal
\newcommand{\eq}{\stackrel{\text{?}}{=}}
% making a larger decimal point
\newcommand{\bd}{\mbox{\Large .}}

% For using in integrals, like \int x\dx
\newcommand{\dx}{\,dx}
\newcommand{\dy}{\,dy}
\newcommand{\dz}{\,dz}
\newcommand{\dt}{\,dt}
\newcommand{\du}{\,du}
\newcommand{\dv}{\,dv}
\newcommand{\dtheta}{\,d\theta}

% For specially formatted fractions
\newcommand{\textfrac}[2]{\frac{\text{#1}}{
\text{#2}}}
\newcommand{\change}[2]{\frac{\text{change in
#1}}{\text{change in #2}}}
% Better appearance for a "skinny frac"
% like \frac 1x
\newcommand{\sfrac}[2]{\frac{\ #1\ }{#2}}

% domain, image and identity functions
\newcommand{\dom}{\mathop{\rm dom}\nolimits}
\newcommand{\im}{\mathop{\rm im}\nolimits}
\newcommand{\id}{\mathop{\rm id}\nolimits}

% Real numbers, etc.
\usepackage{amsfonts}
\newcommand{\R}{\mathbb{R}}
\newcommand{\Z}{\mathbb{Z}}
\newcommand{\Q}{\mathbb{Q}}
\newcommand{\C}{\mathbb{C}}

% Script letters: usually for collections
\usepackage{mathrsfs}
\renewcommand{\L} {\mathscr{L}}
\newcommand{\A}   {\mathscr{A}}
\renewcommand{\P} {\mathscr{P}}
\newcommand{\scrC}{\mathscr{C}}

% for "such that"
\newcommand{\st}{:}
% inverse functions
\newcommand{\inv}{^{-1}}
% Making answer blanks
\newcommand{\blank}[1]{\underline{\hspace*{#1}}
}

\usepackage{amssymb}
% stands for "not divides"
\newcommand{\notdiv}{\nmid}

% for labeling parts of proofs
\newcommand{\forwards}{``$\Rightarrow$''}
\newcommand{\contra}{\ensuremath{\Rightarrow
\Leftarrow}}
\newcommand{\backwards}{``$\Leftarrow$''}

% Theorems etc., read amsthdoc.pdf
\usepackage{amsthm}
\newtheorem{theorem}{Theorem}[section]
\newtheorem{prop}[theorem]{Proposition}
\theoremstyle{definition}
\newtheorem{example}[theorem]{Example}
\newtheorem{definition}[theorem]{Definition}
```

Guideline on creating shortcuts:

- readabilty is much more important than ease of input. Therefore, I don't create `\l` as a shortcut for `\left`, but I do create a shortcut of `\dydx` for `\frac{dy}{dx}`.
- Some of these commands produce subtle improvements in formatting. For intstance

  e^{-(x-\mu)^2/\sigma} produces $e^{-(x-\mu)^2/\sigma}$

  and

  \e^{-(x-\mu)^2/\sigma} produces $e^{-(x-\mu)^2/\sigma}$.

LaTeX Fancy Custom Command Examples

W. Ethan Duckworth, Loyola University Maryland, 2014

Here is a sampling of commands I've created in the context of classroom material. Some of them may be directly useful to you; others may help you imagine what's possible and create your own. To see the actual definitions, look at the source code for this file.

- `\circle`

```
$$
\frac{\circle{31x^5}+4x^2-10x+5}
     {\circle{x^5}-2x^3+10,000}
$$
```

produces

$$\frac{\textcircled{31x^5}+4x^2-10x+5}{\textcircled{x^5}-2x^3+10,000}$$

- `\map`

```
f:\map{A}{B}{x}{x^2}
```

$$f:\begin{array}{ccc} A & \longrightarrow & B \\ x & \longmapsto & x^2 \end{array}$$

- `\define`

```
The number $m$ is called the \define{slope}
, and $y=mx+b$ is called a
\define[equation!linear]{linear equation}.
```

The number m is called the **slope**, and $y = mx + b$ is called a **linear equation**.

As used, this makes two index entries: one equivalent to `\index{slope}` and one equivalent to `\index{equation!linear}`.

- `\squigdownarrow`

This makes a squiggly, downwards arrow like so

$$\lim_{n\to\infty}\sum_{i=0}^{n} x^2\Delta x \;\rightsquigarrow\; \int x^2\,dx$$

- `\chain`

```
{\ddx \sqrt{4x^2+x}}
{\frac{1}{2\sqrt{\inside{4x^2+x}}}}
{(8x+1)}
```

$$\frac{d}{dx}\sqrt{4x^2+x} = \frac{1}{2\sqrt{4x^2+x}}\cdot(8x+1)$$

deriv. of outside — don't change inside — deriv. of inside

For more examples, try the following:

```
\chain[(1ex,0ex)]
   {\ddx \sin(x^2+1)}
   {\cos(\inside{x^2+1})}
   {2x}
\chain[(0.5ex,0.5ex)]
   {\ddx e^{-x^2}}
   {e^{\inside{-x^2}}}
   {(-2x)}
```

- `\TIbutton{$\div$}`
  `\TImenu{MATH}`

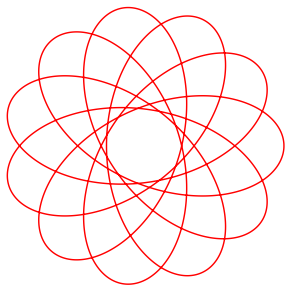
makes ÷ and MATH.

These are meant to look like a button and a menu on a TI calculator.

- `\rvect{a,b,c,d,e,f,g}`
  `\cvect{a,b,c,d,e,f,g}`

$$\begin{bmatrix} a & b & c & d & e & f & g \end{bmatrix} \qquad \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \end{bmatrix}$$

- `\begin{fpanel}{\includegraphics[width=1.5in]{hypocycloid}}`
  `This is the graph of a hypocycloid. It was made in Maple, then I exported it as EPS, and then I included it here.`
  `\end{fpanel}`



This is the graph of a hypocycloid. It was made in Maple, then I exported it as EPS, and then I included it here.

This is for making comic-book like panels, with text at the bottom. It has an argument for a picture (or some kind of box). The picture sets the width of the panel, and then the body of the environment contains the text, that will wrap below the panel.