LaTeX for Mathematics Teachers

Background

I have been told by somebody who has studied both that a main difference between learning to play piano and learning to play guitar is in where the learning curve hits. It is relatively easy, he said, to start playing the piano on day one and producing something that sounds reasonable, but true mastery takes years of daily practice. The guitar, meanwhile, requires a good deal of up-front practice before anything other than atonal noise comes out, but after that point, mastery involves more subtle refinements.

I haven't really studied the guitar enough to know if that's an accurate description, although it does match with my experience with the piano. I *have* worked with both WYSIWYG software and mark-up languages enough to know that the analogy carries over to that realm fairly well.

Before I go into a discussion of mark-up languages in general and LaTeX specifically, I'll provide an overview of WYSIWYG word processors (specifically, Word Office 2010, but the comments apply generally as well). If you're familiar with the philosophical basis of TeX and LaTeX and understand the notion of semantic tagging or CSS, feel free to skip to the next section.

In general, I'll also note that I strive to assume that readers are familiar with the basics of using computer software and are reasonably intelligent, but don't necessarily have much experience with open source or programming.

WYSIWYG

WYSIWYG: W*hat* ***Y****ou* ***S****ee* ***I****s* ***W****hat* ***Y****ou* ***G****et. The document on the screen matches how it will look when printed.*

The odds are high that you've used a WYSIWYG word processor. The most common these days (2012) is Microsoft Word, but perhaps you're more familiar with OpenOffice Writer, Corel WordPerfect, Apple Pages, or something else. What these options have in common is that what you see on the screen is generally what you wind up with on the printed page (WYSIWYG: What You See Is What You Get). The person writing the document has immediate control over the margins, font selection, general formatting, and so on. It is easy for a complete novice to start typing and make adjustments on the fly. For the vast majority of people involved in document preparation, this is the most attractive route. It's easy to see why WYSIWYG has become the standard for word processing.

However, this model does have some limitations. For one thing, because so much is automatic, it can be frustrating (or downright impossible) to overwrite what the system wants to do. For example, by default in Microsoft Word 2010, the straight apostrophe that appears on your keyboard (') is converted to an "appropriate" curly form depending on context. That means that if you type this:

```
    ``This is `it'!''~he announced.~``It's '12!~`Tis the New Year!''
```

you get this:

> "This is 'it'!" he announced. "It's '12! 'Tis the New Year!"

To be technical, the symbol before *'Tis* should be the same as the one before *'12*. Also, if you ever happen to want to start a quotation with a two-digit number, Microsoft Word will give you the incorrect symbol: "We're going to see '39 Steps' at the revival show tonight." These automatic corrections can be overridden, or you can turn off the setting altogether and hand your quotation marks as you please. Also, many people don't care, and Word's automatic selections work the great majority of the time.

A related shortcoming of WYSIWYG is that the producer of the text content also has certain significant responsibilities in formatting. You need to worry about font selection, setting margins, maintaining styles from section to section, and so on. Styles can become particularly hairy, because Word has two different ways of handling these: On the fly, which is what most casual users use, and formally.

Let's say that you want to start each section of your document (as in this document) with an indented label. Each section header will use the same font, which is different from that of the main document. You could do this in Microsoft Word by selecting a header, formatting it as you please, then copy-formatting those changes to each other header in your document. In my experience, this is the way that most casual users do it. Most people then go on to complain because, at some point, they decide to change the formatting of the headers. This entails changing each header. This can be a very time-consuming process, particularly if Word then decides to apply a different font.

The other way to do it in Word 2010 (and other versions) is to use styles. You can start out the same way: Take a section header, and format it the way you want. The next step is to right-click, select Styles and "Save Selection as a New Quick Style", and name it. Now you can select each of your other headers (or any future headers) and select the Style from the list available on the Styles section of the Home tab. This is the best practice for documents with different formats: Each different formatting should have its own style.

The downside to having two different, and conflicting, ways of doing something should be obvious. Even if I understand styles and set them up properly, someone else can munge my file by using the "easier" on the fly method. Files will be peppered with different ways of doing the same thing. If I change the styles on my headers, the ones attached properly to my style will change, while some others might not.

A further difficulty with WYSIWYG software is that it can be challenging to work with certain aspects that are common in technical documents. For instance, it's standard practice to number equations consecutively in a scholarly article, and then refer to those equations by number in the text:

> ... Let us now consider the original problem:
>
> $$\int_0^\pi \sin\theta\,d\theta \tag{1}$$
>
> As we can see from (1), we will be integrating over....

Creating those numbers (and referencing them) can be tricky in Word. The most obvious way to create the reference is to simply type the number, but imagine if we then insert another equation earlier in the document. The challenge is in keeping reference numbers updated. As with styles, there's a way to create cross-references that update automatically, but this relies on people knowing how to do this.

For a mathematician and other technically minded person, another challenge with Word and other WYSIWYG options is in creating attractive formulas with little effort. If you try to create equation (1) in Word, you'll be faced with a lot of clicking, and you may not like the final outcome. Equations are one place where TeX-based systems, such as LaTeX, really shine. Equation (1), for instance, was created with this text: `\int\_0^\pi\sin\theta\,d\theta`. If you're not familiar with LaTeX or other mark-up systems, that may look like nonsense now, but remember the piano/guitar analogy: Once you've started to master TeX, it will make perfect sense, and you'll most likely see how it's an improvement over Microsoft Word's equation editor.

Mark-up, WYSIWYM and LaTeX

For most people, WYSIWYG will satisfy their word processing needs just fine. Even many of the short-comings, such as the two ways of maintaining formatting, can be overcome in corporate settings by having strictly enforced document processing policies. However, there are some contexts in which a different tool is more suitable.

***WYSIWYM:** What **Y**ou **S**ee **I**s **W**hat **Y**ou **M**ean. The document on the screen contains notations which indicate how it should be formatted.*

The major modern alternative to WYSIWYG is WYSIWYM (What You See Is What You Mean), a form of mark-up language. In general, You may be familiar with HTML. HTML, like TeX, is a mark-up language. Formatting is explicitly indicated in the text file. If you want text to appear in bold in Microsoft Word, you select the text and press Ctrl-B; the text will now appear in bold on the screen. If you want text to appear in bold in HTML, you can put the tag <strong>[1] before the text (to indicate "start bolding") and </strong> at the end (to indicate "stop bolding"); <em>[2] can be used for italics. For instance:

```
He said <strong>not</strong> to see <em>12 Monkeys</em>.
```

will appear as:

He said **not** to see *12 Monkeys*.

The equivalent LaTeX, by the way, is this:

```
He said \textbf{not} to see \emph{12 Monkeys}.
```

Simple mark-up like this has been around longer than WYSIWYG: It was how the first computer-based word processors indicated simple text formatting changes like bold and italics. However, WYSIWYM adds an additional layer of information because documents contain information about what specific parts of the text mean.

You may also be familiar with CSS, a now-common aspect of HTML programming. CSS stands for Cascading Style Sheets; for our purposes, it's the middle word that's the most important. "Style" here means the same thing it did in the preceding section. The CSS portion (or file) of an HTML document specifies how specific sections of the document are to be treated. Here's a portion of HTML:

```
<h1>Double u Substitution</h1>
<h2>Introduction</h2>
<div>u substitution is an important technique...</div>
```

The related CSS section would specify specific formatting choices for `<h1>` (title), `<h2>` (section), and `<div>` (main text) pieces of text. In a corporate environment where there are consistent formats to be

[1] <b> is also a common choice. However, <strong> is preferred for user accessibility: Screen readers for the visually impaired read text inside of <strong> tags in a different voice, but not text inside of <b> tags. However, unless a website developer has deliberately changed the formatting, there will be no difference on the screen.

[2] Like <b>, <i> does not change how screen readers read the text.

used for every document, what this means is that individual content writers just need to know what tags to use for what type of content (surround a title with `<h1>` tags, and so on). Formatting is then maintained in a different place, usually a centrally located file or set of files. When the corporation decides to change its official title font from Lucida Sans to Garamond, this can be done in a few seconds by changing a single file, rather than by having to laboriously process through thousands (or millions!) of individual files.

Semantic tags: *Text notations which reflect what sort of content the indicated text represents.*

LaTeXworks basically the same way. Like HTML and CSS files, LaTeX files can be written in any plain text editor, like Notepad, although there are certain tools that use color or auto-complete and hence make it easier to use. The files contain notations within the text itself to indicate the semantic nature of the text: Is it a title? A footnote? An equation? These notations are used to type-set the text appropriately.

As I discussed in the previous section, this sort of semantic information *can* be included (and, in my view, *should* be included) in a properly prepared Microsoft Word or WYSIWYG document, through the appropriate use of styles. However, while the nature of Microsoft Word tends to encourage the casual user towards the more expeditious use of local formatting, LaTeX either requires or at least strongly encourages using appropriate semantic marking.

If this were the sole difference between Microsoft Word 2010 and LaTeX, though, this document wouldn't exist. I personally wouldn't be using LaTeX at all. I know how to use styles in Word. However, what makes LaTeX so compelling for people who work with mathematics on a regular basis is the way in which it handles formulas and mathematics in general.

Getting Started

Installing LaTeX

The first thing you'll need is some sort of distribution of LaTeX. There are other versions of TeX, by the way; you may ultimately prefer XeLaTeX or LuaTeX, for instance. TeX is the family name for a variety of very related type-setting systems, each of which has certain strengths and weaknesses. For instance, LaTeX doesn't have full Unicode support; to get "curly" quotation marks, you have to type ` `` ` and `''`; XeLaTeX also recognizes the Unicode versions (“ and ”). On the other hand, LaTeX is more widely used at this point, so having XeLaTeX symbols in your document may get in the way of sharing files with others. As you explore, decide on the one that's best for you.

Because TeX and its derivatives are based on the open source philosophy of free software maintained by a global community of programmers, it's not as straightforward to install as a single, closed software package like Microsoft Word is. There are a few ways of installing LaTeX on your computer; you can rely on your favorite search engine, or try the options at the LaTeX project (`http://www.latex-project.org/ftp.html`).

One thing you'll notice early on is that there's a dizzying array of individual "packages" that can be installed. Microsoft Word and other commercial software packages tend to just install all of its features at once, leaving you with a bloated program that takes up hard drive space with a bunch of stuff you don't need. If you don't care about memory (and these days, it's a diminishing concern), that's not a big deal. However, open source software tends to be built around the notion that people should only have to install what they'll actually be using. Also, because so many different programmers are involved, there are different packages that do similar things.

Packages: *Add-ons in LaTeX that allow for certain advanced formatting.*

Depending on your needs as a mathematician, you may be fine with just AMSMath, which should be installed by default; you may also need AMSSymb, also available by default. In fact, you can do a lot of mathematics formatting without any packages at all. If you're working in a situation where you want to submitted documents regularly to an academic journal, you'll probably want to avoid doing too much of your own formatting anyway; the original point, after all, is that you (the writer) will be producing content with is marked up in such a way that the publisher (the journal you're submitting to, for instance) can deal with the formatting.

However, if you want to adjust your own documents, or if you are personally tasked with formatting documents for publication, you may want to explore some of these packages. For instance, I use the `boxedminipage` and `wrapfigure` packages to produce the inset call-out boxes on some of these pages; I use the `fancyhdr` package to produce the fancier footer you see on the pages. For the purposes of this document, I'll assume you're looking to figure out how to generate basic documents, though; you can learn the other stuff, if you're interested, later on.

Another thing you may find helpful is a GUI, a graphical user interface. While LaTeX files can easily be edited within a text editor, it can get annoying generating the formatted file to make sure that things are looking the way you want them to. This is especially true early on in the learning process, where you may not trust yourself as much and want to check your edits often. LaTeX, like XHTML (and unlike HTML), is fairly unforgiving of programming errors. If you leave off a closing tag in HTML, HTML will do its best to figure it out; LaTeX will simply return an error message (sometimes fairly cryptic). So it helps immensely to have an interface that both speeds up the process of creating formatted files (PDF, for instance) and helps pinpoint any errors.

I use Texmaker (`http://www.xm1math.net/texmaker/`); I prefer it to LyX (`http://www.lyx.org/`), but by all means explore both and any others you might find, and do what works best for you. I find the simplicity of Texmaker's interface to be helpful without being invasive. If you have multiple monitors (which I do), you can leave the generated PDF window open in a different monitor. It doesn't update the PDF view as you type, but re-generating the file is quick to do with a button click (just click the arrow to the left of "Quick Build" at the top of the screen).

Other Uses of LaTeX

If you want to insert mathematics equations into a web page using LaTeX formatting, I recommend MathJax (`http://www.mathjax.org/`). Just add the following to the head of your HTML files:

```
<script type=\textquotedbl /javascript\textquotedbl ~src=
\textquotedbl http://cdn.mathjax.org/mathjax/latest/MathJax.js?
config=TeX-AMS HTML\textquotedbl ~/>
```

Inline mathematics is enclosed in `\(` and `\)`, while standalone mathematics is enclosed in `\[` and `\]`.

Another advantage to knowing how to create mathematical equations in LaTeX is that Wolfram Alpha (/urlhttp://www.wolframalpha.com/) understands it. Hence, rather than trying to figure out how explain your problem in plain English within the confines of the standard keyboard, you can ask your question clearly:

```
Integrate the sine of theta from zero to pi
Calculate $\int_0^\pi\sin\theta\,d\theta$
```

In either case, Wolfram Alpha informs us that the answer is 2, and gives us some other information as well. In this example, it was easy to explain the problem in plain English, but you can hopefully see how it could become difficult with more complexity.

The Minimal LaTeX file

Once you have a version of LaTeX installed, you can begin creating documents right away. Here is an example of a minimal template:

```
\documentclass{article}
\usepackage{amssymb,amsmath}
\begin{document}
    This is a basic \LaTeX ~file.
\end{document}
```

If you compile this (in Texmaker, by clicking the arrow to the left of Quick Build), you should get a PDF file containing a single line (This is a basic LaTeX file.) on a numbered page.

The first line, `\documentclass{article}`, tells the compiler to use the formatting for articles. There are six document classes that come with LaTeX2e , each of which comes with different default font sizes, page number positioning, and other formatting. If you're not sure which to use, use "article".

The second line, `\usepackage{amssymb,amsmath}`, indicates that you plan to use the AMS Symb and AMS Math packages in this document. You can refer to packages you don't actually use, but if you use commands in your text that refer to packages you haven't indicated, you'll get an error. You can list multiple packages this way, separated by commas, or you can list each package on a separate line, as in:

```
\usepackage{amssymb}
\usepackage{amsmath}
```

Note that in the first two lines, the arguments for the command are included in curly braces ({}). A casual way of reading the first two lines is: "This **document** is an *article*. It uses the **packages** *AMS Symb and AMS Math*." The commands are `documentclass` and `usepackage`. For many commands, arguments are included in curly braces; sometimes, other arguments for the command are included in square brackets ([]).

Another common structure for LaTeX, one that allows for text to have paragraph breaks, is the `begin-end` structure, known as the **environment**. That's shown on the last three lines of the sample:

```
\begin{document}
   This is a basic \LaTeX ~file.
\end{document}
```

The argument for the `begin-end` (environment) structure specifies what sort of thing is inside of it. Each LaTeX document must have a document body in it. In this case, the contents of the document body are `This is a basic LaTeX file.`

At this point, with this template, you can begin typing your document, replacing `This is a basic LaTeX file.` with whatever you want. As you get more comfortable and if you want to start modifying your own formatting, you can then add packages as appropriate.

White space

I have a few more comments before we get started with some actual examples of mathematics formatting. Very importantly, LaTeX doesn't generally care about white space. Remember that the author is not supposed to be worried about formatting. You can run all of your text on one line (whether or not you have word-wrapping turned on), or you can press Enter at the end of each line. You can single-space or double-space after each sentence. LaTeX will ignore any extra spaces you add and do its own formatting. These will all create the same document:

```
\begin{document}
   This is a basic \LaTeX ~file.
\end{document}

\begin{document} This is a basic \LaTeX ~file. \end{document}

\begin{document}
   This is a basic
   \LaTeX ~file.
\end{document}
```

This does mean you need to learn some tricks to get white space when you want it. To force a line break (without starting a new paragraph), add `\\` at the end of a line. To create a new paragraph, insert a completely blank line:

```
\begin{document}
   This goes to the next line \\
   but doesn't start a new paragraph.

   This starts a new paragraph.
\end{document}
```

The white space after certain commands (such as `LaTeX`) will be removed. A tilde(˜) is used to force a single space. Because the standard spacing after a sentence in LaTeX is sometimes slightly wider than the spacing between words, you may want to use the tilde to maintain that narrower word spacing in people's names (e.g., type `Mr.~Smith` instead of `Mr. Smith`).

Also, as noted earlier, LaTeX doesn't recognize Unicode characters. It simply gets confused. It's very powerful when you want to build specific characters; you just have to tell it how you want them constructed. For instance, `\"a\`a\'a\~a\^a\d{a}\r{a}\c{a}` generates äàáãâạåą. If you have a document that contains Unicode already, though, such as something you've copied from Microsoft Word (with its curly quotation marks), you'll have to convert them before LaTeX can handle them correctly. If this is something you're likely to do often, you may want to look into XeLaTeX instead.

There are other formatting tricks you'll learn as you explore. In some cases, as with any software package, it may seem like it's impossible to do a particular thing (for instance, since LaTeX converts `` and " to ‘ and ’, it's not obvious how to create the non-coverted forms if you need them)[3]. However, rest assured, if it's something that can appear as part of a text document, there's most likely a way to do it in LaTeX.

Mathematics

Inline vs Display

TeX has two commonly used ways of displaying mathematics: Inline, where equations appear as part of the paragraph stream ($a+b=c$) and display, where equations appear on their own line:

$$a+b=c$$

To create inline equations, surround the equation with either single dollar signs (`$a+b=c$`) or with the backslash-escaped parentheses (`\(a+b=c\)`). It doesn't matter which you use. Using the dollar signs means faster typing, but keep in mind that the default in MathJax is to only use the backslash-escaped parentheses.

To create display equations, you can either use double dollar signs (`$$a+b=c$$`) or backslash-escaped brackets (`\[a+b=c\]`); again, use the one you prefer, although the latter is currently preferred. Additionally, there are several environments that use display mode by default. These include `align` and `equation`, which also allow you to add a number to the equation. We'll look at examples below.

Variables and basic operators

You'll notice from the preceding subsection that any text in a mathematics displaying mode, be it inline or display, will be in italics. That's the standard way of displaying variable names in mathematics. When you need non-italic text (such as for function names), there are ways to do that as well; this will be discussed later.

Basic operators can be included either by typing them or by using the appropriate command, de-

[3]Unfortunately, it's different for each symbol. For ``, type `\`{}\`{}`; for ", use `\textquotedbl`.

pending on the symbol you want. Addition and subtraction can be obtained using the obvious keyboard symbols + and -. Note that the latter appears as a hyphen (-) in text mode but is converted to a minus sign ($-$) in a displaying mode.

Multiplication and division both have multiple symbols in common use. For multiplication, you can use `\cdot`, `*`, or `\times`, which produce $\cdot$, $*$, and $\times$, respectively. For division, you can use `/`, `\div`, or `\frac{a}{b}`, which produce $/$, $\div$, and $\frac{a}{b}$, respectively. Note that `\frac{}{}` produces more readable results in display mode than in inline mode:

$$x = \frac{a}{b}$$

You can force it to use the larger characters (and hence take up more vertical room) in inline mode by including `\everymath{\displaystyle}` in the preamble (that is, before `\begin{document}`).

You can indicate exponents by using the ^ symbol to automatically superscript the following character: `a^b` displays a^b. If you want to include multiple characters in the superscripted part, use braces around the part you want superscripted: `a^{b\cdot c}` displays $a^{b\cdot c}$.[4] If you want the ^ symbol itself to appear, use `\hat{}~` or `\text{\textasciicircum}`.

Note that all whitespace in mathematics display modes is ignored, so if you want any spacing, you'll have to put it in using either ~ (for a wider space) or `\,` (for a narrower space).

Numbering Equations

When displaying equations on their own line, it is common to want to number the equations so they can be easily referred to in the text:

$$a^2 + b^2 = c^2 \tag{2}$$

If you just use two dollar signs to indicate an equation for display, you won't get this number. The two most common ways to get this numbering is to use `equation` or `align`; another important option is `subequations`.

Use `equation` if you want to show a single line. Thus, an input of

```
\begin{equation}
   x = 3
\end{equation}
```

[4]With certain commands that take a single argument, such as `^{}`, the braces are optional if the argument is a single character. Hence `a^b` and `a^{b}` display the same thing.

will result in

$$x = 3 \tag{3}$$

If you have multiple lines and you'd like to line them up based on an equal sign or some other symbol, use `align`. To indicate where the information should line, use &; indicate new lines with two backslash symbols. For instance,

```
\begin{align}
   x &= a + b \\
   &= 3
\end{align}
```

will result in

$$x = a + b \tag{4}$$

$$= 3 \tag{5}$$

If you want to create a third column of aligned information (because, for instance, you want to include short comments), just use another &. This third column will be right-justified. For instance,

```
\begin{align}
   x &= a + b &(given) \\
   &= 3 &a=1,~b=2
\end{align}
```

will result in

$$x = a + b \qquad\qquad (given) \tag{6}$$

$$= 3 \qquad\qquad a = 1,\ b = 2 \tag{7}$$

If you want the third column left-justified instead, use two ampersands (&) instead of one.

If you want equations which are grouped together to be numbered in a group (that is, in the form 1a, 1b, etc.), use `subequations` as well as `align`. For instance,

```
\begin{subequations}
\begin{align}
   x &= a + b \\
   &= 3
\end{align}
\end{subequations}
```

will result in

$$x = a + b \tag{8a}$$

$$= 3 \tag{8b}$$

To create a cross-reference for an equation to be used in the text, add the command `\label{labelname}` before the relevant line in the equation, then refer to it using `\eqref{labelname}`.[5] For instance,

```
\begin{subequations}
\begin{align}
   \label{givenx} x &= a + b \\
   &= 3
\end{align}
\end{subequations}
When we refer back to the given information shown in \eqref{givenx}....
```

will result in

$$x = a + b \tag{9a}$$

$$= 3 \tag{9b}$$

When we refer back to the given information shown in (9a)....

Note that `labelname` can be any unique label name. Also note that if you're using Texmaker, you may need to run Quick Build twice (rather than just once) to get the appropriate cross-reference numbers.

[5]You can also use `\ref{labelname}` if you don't want the parentheses.

Lines, rays, and segments are typically indicated with an overbar with arrowheads. These are done in LaTeX using `\overleftrightarrow{AB}`, `\overrightarrow{AB}`, and `\overline{AB}`; these generate, respectively, $\overleftrightarrow{AB}$, $\overrightarrow{AB}$, and $\overline{AB}$.

Some other useful symbols for trigonometry are `\angle`, `\triangle`, and `^{\circ}`, which generate, respectively, $\angle$, $\triangle$, and $^{\circ}$. Review the following example:

> Consider $\triangle ABC$. If $\angle A = 90^{\circ},~\angle B = 30^{\circ}, \text{ and } m\overline{AB}=1 \text{, then } m\overline{BC} = 2$.

This is generated by the following in the base LaTeX text file:

```
Consider $\triangle ABC$. If $\angle A = 90^{\circ},
~\angle B = 30^{\circ}, \text{ and } m\overline{AB}=1
\text{, then } m\overline{BC} = 2$.
```

This example demonstrates one way to include text in a non-italic font in a mathematical display mode: Put it inside of `\text{ }`. Remember to put spaces in as appropriate within the braces, since spaces in mathematical display mode are generally removed.

There are also a number of standard mathematical abbreviations that normally appear in a non-italic font, and which can be included by preceding them with a \. These include all the standard abbreviations of trigonometric functions, such as `\sin` and `\cos` (as well as others, such as `\log` and `\ln`). Let's continue the previous example:

> This is because $\sin 30^{\circ} = \frac{1}{2}$.

that is,

```
This is because $\sin 30^{\circ} = \frac{1}{2}$.
```

The degree symbol is a superscripted circle, just as exponents are superscripted numbers. If you want a subscripted number, precede it with an underline instead of a caret:

```
$x_1 + x_2 = x_3$
```

creates

$x_1 + x_2 = x_3$

As with superscripts, if you want to indicate more than one character as a subscript, enclose the characters in braces. You can have both a superscript and a subscript on the same character:

```
$x_1^2 + x^2_2 = x_3$
```

creates

$x_1^2 + x^2_2 = x_3$

Note that it doesn't matter which order you put the superscript and subscript in.

If you want to use Greek characters, simply use a backslash and spell out the character's most common name; if you want it capitalized, capitalize the first letter of the name. For example:

```
$\pi\sin\theta = \frac{\pi}{2}$

$\Sigma_{i=1}^4\,x_i = 10$
```

creates

$\pi\sin\theta = \frac{\pi}{2}$
$\Sigma_{i=1}^4\,x_i = 10$

However, the more common way of showing the parameters for the sum and product operators is to put them above and below the operator, instead of next to it. We'll discuss that in the next subsection.

Calculus and More

We indicate the parameters for limits using the subscript marker. LaTeX knows to put the parameters under the limit operator instead of next to it when in display (as opposed to inline) mode. Hence,

```
$$\lim_{x\to+\infty}\,\frac{1}{x}=0$$
```

displays as:

$$\lim_{x\to+\infty}\frac{1}{x}=0$$

Note that we can nest superscripts and subscripts, which is useful for limits.

```
$$\lim_{x\to 0^-}\,x^2=\lim_{x\to 0^+}\,x^2
=\lim_{x\to 0}\,x^2=0$$
```

displays as:

$$\lim_{x\to 0^-}\,x^2=\lim_{x\to 0^+}\,x^2=\lim_{x\to 0}\,x^2=0$$

Similarly, if we want to use the sum and product operators such that the parameters appear below and above the operators, we use `sum` and `prod`:

```
$$\sum_{n=1}^4\,n=10;~\prod_{n=1}^4\,n=24$$
```

displays as:

$$\sum_{n=1}^4\,n=10;~\prod_{n=1}^4\,n=24$$

Likewise, we can use the `int` command to create an integral:

```
$$\int_0^4\,2x\,dx=16$$
```

displays as:

$$\int_0^4\,2x\,dx=16$$

Let's say we wanted to show the intermediate step. Simply using the standard pipe character may yield less than satisfactory results.

```
$$\int_0^4\,2x\,dx=x^2|_0^4=16$$
```

displays as:

$$\int_0^4 2x\,dx = x^2|_0^4 = 16$$

The 0 and the 4 are a bit crowded. To get a better range symbol, we'll have to rely on a bit of a trick.

Before explaining how to get the range symbol we want, I'll tangent to discuss parentheses. Standard parentheses don't resize, regardless of what's inside of them. Instead, if you want parentheses that scale based on the contents, use `left(` and `right)`. Compare:

```
$$\left(\frac{1}{2}\right) \text{ vs. } (\frac{1}{2})$$
```

$$\left(\frac{1}{2}\right) \text{ vs. } (\frac{1}{2})$$

There are a few delimiters that can be resized using `left` and `right`, but these have to come as a pair, otherwise LaTeX returns an error. The resizeable pipe is `right|`, but it requires the dummy placeholder `left.` somewhere to the left of it, with at least one character in between. To keep things tidy, consider putting the placeholder immediately after the equal sign:

```
$$\int_0^4\,2x\,dx=\left.x^2\right|_0^4=16$$
```

$$\int_0^4\,2x\,dx=\left.x^2\right|_0^4=16$$

Let's look back at the integration shown at the beginning: `\int_0^\pi\sin\theta\,d\theta`. Hopefully this will make more sense now.

Some other symbols you might find useful include:

```
$\exists \forall \in \cap \cup \wedge \vee \therefore
\Rightarrow \sqrt{x}$
```

$\exists \forall \in \cap \cup \wedge \vee \therefore \Rightarrow \sqrt{x}$

If you want to use a symbol to refer to a subset of numbers, format the appropriate letter with `mathbb{}`:

```
$\mathbb{NWZRI}$
```

$\mathbb{NWZRI}$

Thus, you could write:

```
$y = \sqrt{x} \Rightarrow x\wedge y \in \mathbb{R}:
x \wedge y \in (0,\infty)$
```

$y = \sqrt{x} \Rightarrow x \wedge y \in \mathbb{R}: x \wedge y \in (0,\infty)$

and:

```
$y = \frac{1}{\sqrt{x^2+x-6}}-8 \Rightarrow
x\wedge y \in \mathbb{R}:x \in (-\infty,-3]\cup[2,\infty)
\wedge y \in [-8,\infty)$
```

$y = \frac{1}{\sqrt{x^2+x-6}}-8 \Rightarrow x\wedge y \in \mathbb{R}:x \in (-\infty,-3]\cup[2,\infty) \wedge y \in [-8,\infty)$

As you can probably imagine, there are many, many characters that you can generate using LaTeX. AMS Math and AMS Symb contain a very wide variety of mathematical symbols; more specialized characters appear in other packages. An exhaustive, organized list, The Comprehensive LaTeX Symbol List, is available at `http://www.tex.ac.uk/tex-archive/info/symbols/comprehensive/symbols-a4.pdf`.

There are also numerous tutorials and examples available online. Hopefully, this guide has given you some basic foundations on which to build, as well as showing you the advantages of using LaTeX specifically for creating mathematically related formulas and documents.