The Markdown Mini-Tutorial

Adapted from The Markdown Tutorial, By Ryan Hodson

This tutorial is a plain text document that needs to be formatted with Markdown. As you work through each section, you will learn a little background about each Markdown element, and then you will be prompted to add your own Markdown formatting. Don't be alarmed if you see some markup that you don't understand yet. Everything will be explained by the end of the tutorial. The instructions will guide you along the way.

<!-- This is what instructions will look like. -->

By the end of the tutorial, you will have transformed this document into a nicely formatted Markdown reference for later use. Let's get started.

****

Markdown and HTML
Notice that instructions look just like HTML comments. That's because HTML formatting can be inserted directly into a Markdown document. Try adding `<strong>` tags to the following line:

<!-- Add tags around the following line. -->

This wimpy string of text is soon to become a BOLD string of text.

Markdown is (usually) smart enough to know when you are trying to add HTML to a document, versus when you just want to add a < or a > or a & character. For example,

The less than sign in 3 < 4 is NOT recognized as HTML
The ampersand in AT&T is NOT recognized as HTML
Items wrapped in HTML tags are recognized as HTML
HTML escape sequences like & and © are recognized as HTML.

Generally, you shouldn't have to worry about whether or not a character is escaped properly, Markdown should do it for you. One common exception is when you write about HTML and you want to show a <tag>, you will have to escape it by hand. Later, we will cover a more convenient way to write about code.

For now, just remember that whenever you come across some kind of formatting that you can't make in Markdown, you can just switch to HTML.

****

Header Elements
A header element is tranformed into '`<h1>, <h2>,...,<h6>`' tags in HTML. There are two ways to make a header in HTML:

<!-- Throughout the tutorial, you may see sequences like `foo`. This is a way of escaping special characters that we will discuss later.

Right now, just look at what is in between the ` ` marks. For example, when you see `#`, just think of it as a # sign all by itself. -->

Atx Style Headers
The first method is by using a `#` (pound sign) before the text of the header. The number of `#` characters that you use denotes the level of the header tag. So, `#` translates to `<h1>`, `##` translates to `<h2>` and so on, up to `<h6>`. You may enclose the header by the same number of `#` signs. This is not required, but it does make your plain text easier to read. The following is an `<h3>` header.

This is important information

<!-- Try adding an <h1> header to 'Header elements' and an <h2> header to 'Atx Style Headers'. -->

Setext Style Headers
The second method is by 'underlining' the text of the header with `-` (minus sign) or `=` (equal sign) characters. The `=` translates to `<h1>`, while `-` translates to `<h2>`. Higher lever header tags are not supported. The following is an example of an `<h1>` header.

This info is more important that the last info.
==========

<!-- Try making 'Setext Style Headers' into an <h2> header. -->

Note that you don't have to underline the entire title, one `-` or `=` will suffice. However, underlining all of it makes your plain text more readable.

****

<!-- Add an <h1> header to the following line. You can use either method. -->

Paragraphs and Line Breaks

<!-- Remember that '<' and '>' in the next paragraph are recognized as HTML entities -->

Paragraphs, denoted by '<p>' tags in HTML, are a very natural entity in Markdown syntax. To create a paragraph, simply leave a blank line after the text that you want to mark as a paragraph. For example,

<!-- Try adding a blank line between the two paragraphs. -->

THIS IS PART OF MY FIRST PARAGRAPH. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque sed arcu est. Aliquam augue lorem, fringilla non sagittis a, eleifend at purus. Sed lectus mauris, semper interdum suscipit quis, aliquet vel diam. Cras a metus vitae ligula dignissim placerat.
THIS IS PART OF MY SECOND PARAGRAPH. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque sed arcu est. Aliquam augue

lorem, fringilla non sagittis a, eleifend at purus. Sed lectus mauris, semper interdum suscipit quis, aliquet vel diam. Cras a metus vitae ligula dignissim placerat.

Line breaks are handled a little differently that you might expect. HTML `<br />` tags are NOT inserted at every newline in your plain text. This means that simply typing a 'return' will not produce a `<br />` tag in the resulting HTML. To insert an HTML break tag, you need to end a line with TWO spaces, then press return. For example,

<!-- Try adding two spaces to the end of the next line -->

This line ends with two spaces.
This should be on the second line.

This line does not end with two spaces.
This should still be on the same line.

The implication for this behavior is the ability
to format your plain text with newlines, without
forcing the newlines into your HTML document
(e.g., your text will be a solid block in your
HTML, filling the entire width of the enclosing
`<div>` tags, while retaining line breaks in
the plain text version). This paragraph is a
case-in-point.

****

<!-- Add an <h1> header to the following line. -->

Emphasis
Markdown uses `*` (asterisk) and `_` (underscore) characters as indicators of emphasis. Enclose the emphasized text with one character for italic text (HTML `<em>` tags), two characters for bold text (HTML `<strong>` tags), and three characters for text that is both italicized and bold. You can use either one, *but* you must make sure that the opening and closing character are the same (no mixing and matching `*` and `_` in the same span of text).

This text has three levels of emphasis.

If you want to include a literal `*` or `_`, you must escape it with a backslash.

<!-- Try adding various levels of emphasis to words in the next sentence. -->

The quick brown fox jumps over the lazy dog.

****

<!-- Add an <h1> header to the following line. -->

Horizontal Rules
Horizontal rules are denoted by `<hr />` in HTML. To create a horizontal rule in markdown, place three or more consecutive `*` (asterisk), `-` (minus sign), or `_` (underscore) characters on an empty line. Optionally, they may have spaces between them. For example, `* * *` will produce the same horizontal rule as `----------`. Generally, I tend to fill a whole line with `*` because it makes it easier to see the horizontal rule in plain text.

<!-- There is a horizontal rule missing between this section and the next. Try placing one before the 'Lists' section. -->

<!-- Make the 'Lists' title into an <h1> header. -->

Lists
Markdown supports both unordered and ordered lists, denoted in HTML by `<ul>` and `<ol>`, respectively. Unordered lists can use `*`, `-`, or `+` to define a list. This is an unordered list:

* Apples
* Bananas
* Oranges

<!-- Try making the following items into a list with a '-' or a '+'. Remember to leave a space after the '-' or '+'. -->

Fruits that keep doctors away
Fruits that are for monkeys
Fruits that don't rhyme with anything

Ordered lists are numbered lists in plain text:

1. John Lennon
2. Ringo Starr
3. Paul McCartney
4. George Harrison

The actual numbers that you use are inconsequential. As long as you start the list with a '1', the resulting HTML will be an ordered list.

<!-- Try making the following items into an ordered list using any order of numbers, but remember to start with a '1' -->

Britney Spears
Christina Aguilera
Hilary Duff
Hannah Montana

<!-- Add an <h2> header to the following line. -->

Paragraphs in List Items
Lists items can also contain paragraphs ('<p>' tags). To make a

list item into a paragraph, add a blank line between each item. Note that this is the same way that you would make a regular paragraph outside of a list.

<!-- Try making each of the following list items into paragraphs. The first one has been done for you. All Star Trek information is supplied by Wikipedia.org. -->

- Christopher Pike

- Spock
- Number One
- J. M. Colt
- Phillip Boyce
- Jose Tyler
- Garrison

List items can have more than one paragraph. Subsequent paragraphs in the same list item should begin with 4 spaces or one tab.

<!-- Try adding multiple paragraphs to the following list items by indenting the second paragraph. Also add two levels of emphasis to the name of each Star Trek officer. The first two have been done for you. -->

+ __James T. Kirk__ was the Captain. He was played by William Shatner. Kirk was the Commanding Officer and Chief of Starfleet Operations.

    If I knew anything more about Captain Kirk, I would write about it in this second paragraph.

+ **Spock** was a Commander. He was played by Leonard Nimoy. Spock was Executive Officer, Science Officer, Commanding Officer, and Ambassador.

    Note that I used 'hanging indents' to align the text of the previous paragraph, as well as this paragraph. This is a common practice for producing readable plain text documents, and will have no effect on the HTML output.

+ Leonard "Bones" McCoy was a Commander and later an Admiral. He was played by DeForest Kelley. Bones was the Chief Medical Officer.

If I knew anything else about Leonard McCoy, I would write about it in this second paragraph.

+ Montgomery "Scotty" Scott was a Lt. Commander. He was played by James Doohan. Scotty was Chief Engineer and Second Officer.

If I knew anything else about Scotty, I would put it in this second paragraph. Wasn't he Irish? Or maybe it was Scottish...

+ Hikaru Sulu was a Lieutenant. He was played by George Takei. Hikaru was the Helm Officer.

```
If I knew anything else about Lieutenant Sulu, I would put it in
this second paragraph.

+ Uhura was a Lieutenant. She was played by Nichelle Nichols. Uhura
  was the communications officer.

If I knew anything else about Uhura, I would put it in this second
paragraph.


<!-- There is another horizontal rule missing before the following
section. Try adding one before the 'Code' section. -->



<!-- Add an <h1> header to the following line. -->

Code
Many times when writing about code, you will want to include examples.
There are two ways to include code snippets in Markdown:

<!-- Make this into an ordered list with two items. -->

As inline code
As a code block

<!-- Add an <h2> header to the following line. -->

Inline code

<!-- The \` in the following line is an escaped backtick. Just think of
it as a ` (backtick) character all by itself when you see it. -->

Inline code appears on the same line as the surrounding text, but
formatted as 'code' (that usually means a monospace font of some
sort). Inline code is surrounded by \` \` (backticks). Be careful not
to confuse backticks with apostrophes. Backticks are on the same key
as the ~ in most Western keyboard layouts. When translated to HTML,
inline code is enclosed by `<code>` tags.

<!-- Try formatting the R function call as inline code. -->

For example, If I were writing a tutorial on PHP and wanted to write
about the function tapply(), I could do it using
backticks.

<!-- Add one level of emphasis to the word 'LOT' in the next sentence. --
>

But, what if you want to write about a LOT of code?

## Code Block ##

<!-- In the next paragraph, '<pre><code>...</code></pre>' should be
```

```
marked as inline code. Try formatting it as such using backticks. -->

The other way to include code in Markdown is as a whole code block. In
HTML, this translates to <pre><code>...</code></pre> tags. To include
a code block, simply indent the code by at least 4 spaces, or one tab.
One level of indentatation will be removed from code blocks when they
are translated to HTML.

<!-- Try making this R function into a code block by adding
four more spaces before each line. -->

psi.1 <- function(x) {
  psi <- ifelse(x^2 > 1, abs(x), x^2)
  return(psi)
}

HTML characters and markdown characters will be properly escaped
within code blocks (and within inline code), so you can just indent
the original code, and it will come out exactly as you see it in plain
text.

    As an example,
    * This list looks like Markdown syntax in an HTML page.
    * That's because Markdown isn't parsed between code blocks.
    * So, it's easy to use Markdown to write about Markdown.
    * Look, I can even include <strong>HTML Tags</strong>, and they will
      still be visible in an HTML page.

<!-- Add another horizontal rule before the next section. -->



# Links #
There are two types of links in Markdown:

<!-- Make the following items into an ordered list. -->

Inline links
Reference links.

## Inline Links ##
Markdown inline links are equivalent to HTML `<a href='foo.com'>`
links, they just have a different syntax. Links have the following
format:

<!-- Add 4 spaces or a tab to the next line to make it into a code
block -->

[Link Text](url "Title")

<!-- Make each element into an unordered list item. Put two levels of
emphasis on the name of each element as well. -->

Link Text - The text that will appear on the HTML page.
```

```
url - The url of the link.
Title - The text that will appear when the user hovers over the link.
        This is an optional parameter. It is equivalent to the `title`
        attribute of an HTML link.

Let's look at an example:

[Visit my site](http://example.com "Visit example.com")

<!-- The '/about/' path in the next paragraph should be an inline code
span. Format it as such using backticks. -->

The url can also be a relative url. For example, /about/ will go to
the about page in the same domain as the current page. This is the
same behavior as the HTML `href` attribute. Now try adding a link to
your site.

<!-- Add a link to your site. Use anything your want for the link
text, url, and optional title. -->

My website:

## Reference Links ##

<!-- Add one level of emphasis to 'love' in the next sentence and
'referencing' in the second sentence. -->

I love reference links. Markdown lets you define links by referencing
the url of the link instead of hardcoding it into every occurrence of
a link. This is the equivalent of using a variable to store a value in
a programming language. Reference links make it easy to update urls
when they change. You only have to change the referenced value, and
all of the links that use it will magically update. This is the syntax
for a reference link:

<!-- Make the next two lines into a code block. -->

The link: [Link Text][id]
The reference: [id]: url "Title"

<!-- Make each element into an unordered list item. Put two levels of
emphasis on the name of each element as well. -->

Link Text - The text that will appear on the HTML page.
id - The name of the reference.
url - The url of the link.
Title - The text that will appear when the user hovers over the link.
        This is an optional parameter. It is equivalent to the `title`
        attribute of an HTML link.

First, add the link to your document as you normally would. Then,
anywhere else in the document, add the reference line containing the
actual url of the link. Let's look at an example for reference links:
```

<!-- Example begins (don't change this part) -->

[URL]: http://example.com

You should really [visit my site][My URL]. Let's talk about something else for awhile. Ok, now you really need to [visit my super awesome website][URL].

<!-- Example ends -->

In the example, the url is referenced twice in the text. If you ever need to change the url of your site, you would only have to change the line with http://example.com for both links to point to your new site. 'URL' is the reference id for `http://example.com`. You can use anything that you want for the reference id.

<!-- Try making the reference for the links in the following text. It can come before or after the text. -->

[Cats][Cat Site] are really awesome. I'm allergic to them, but I still like them. They make me all stuffy and teary whenever I pet them, but they are just so soft and fluffy. If you want to know more about cats, visit [this website that is dedicated to cats][Cat Site].

<!-- Add two levels of emphasis to the word 'NOT' in the next sentence. -->

One thing to remember about reference links is that the reference id is NOT case-sensitive. So, don't try to define two links that only differ by a capital letter (that's bad practice anyway).

<!-- Add another horizontal rule before the next section. -->

<!-- Add an <h1> header to the following line. -->

Block Quotes

<!-- In the second sentence of the next paragraph, add one level of emphasis to the phrase 'block quotes'. -->

Many people often have the need to reference somebody else's statements in their writing. Markdown makes this easy by using block quotes. Block quotes are denoted by a `>` (greater than) character before each line of the block quote. Let's look at an example:

<!-- The following sentences do not appear on their own lines in an HTML page. Add HTML breaks to each line by adding two spaces after each sentence. -->

> Somebody said something mean about me.
> I don't know who it was or what they said.
> For some reason, I'm still trying to quote them using this Markdown

```
  block quote.

You can have multiple paragraphs in a single block quote, just add a
`>` before the beginning of each paragraph.

<!-- Add 2 levels of emphasis to the first sentence in each paragraph. --
>

> This is part of my first paragraph. Lorem ipsum dolor sit amet,
consectetur adipiscing elit. Pellentesque sed arcu est. Aliquam augue
lorem, fringilla non sagittis a, eleifend at purus. Sed lectus mauris,
semper interdum suscipit quis, aliquet vel diam. Cras a metus vitae
ligula dignissim placerat.

> This is part of my second paragraph. It is in the same block quote.
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque
sed arcu est. Aliquam augue lorem, fringilla non sagittis a, eleifend
at purus. Sed lectus mauris, semper interdum suscipit quis, aliquet
vel diam. Cras a metus vitae ligula dignissim placerat.

It is a common practice to put a `>` before each line as in the next
block quote. This is purely for aesthetic purposes, as it makes the
block quote more apparent in plain text.

> This is a block quote where each line begins with a > character. The
> sole purpose of this is to make the plain text document more readable.
> You can be lazy and only indent the first line if you want. This
> example also introduces nested block quotes:
>
> > **This is a nested block quote**. Nested block quotes are useful for
> > conveying situations like, "Sandy told me that Sheila said that
> > Pamela's sister wants to go to the dance with Johnny."
>
> This is part of the original block quote. Remember to add a blank line
> between each nested block quote.


<!-- Add another horizontal rule before the next section. -->



<!-- Add an <h1> header to the following line. -->

Images
Yay! Finally some visuals. Unfortunately, images are included using
plain text, so your document isn't going to look like it has pictures
in it until it is output as HTML.

Images look an awful lot like Markdown links, they just have an extra
`!` (exclamation mark) in front of them. Here is the formal syntax:

<!-- Format the following line as inline code (not a code block). -->

![Alt Text](url "Title")
```

<!-- Make each element into an unordered list item. Put two levels of emphasis on the name of each element as well. -->

Alt Text - The text that will appear on the HTML page.
url - The url of the image.
Title - The text that will appear when the user hovers over the image.
        This is an optional parameter. It is equivalent to the `title`
        attribute of an HTML image.

For example, here the Vanderbilt logo formatted with Markdown syntax:

```
![Vandy
logo](http://sitemason.vanderbilt.edu/files/gwYypa/vu06b.gif/main.gif)
```

Now try adding your own image.

<!-- Try adding another image to this document. Use anything you want for the alt text, url, and optional title. -->

You can also use reference urls for images. This is the exact same as using reference links, but with a prepended `!` character. For example, this would be how to display my logo using a reference link.

<!-- Make the following two lines into a code block. -->

```
![Vandy logo][id]
[id]: http://sitemason.vanderbilt.edu/files/gwYypa/vu06b.gif/main.gif
"Vanderbilt Logo"
```

You will find reference urls for images very useful if you reuse the same image throughout a single document.

<!-- Add another horizontal rule before the final section. -->

Conclusion
Congratulations! You have completed **The Markdown Tutorial**. By now, you should have a solid grasp of Markdown syntax, as well as a nicely formatted Markdown reference for future use.