Paython Markdown

What is Python Markdown?

Python Markdown is a way of creating fully reproducible documents, in which both text and code can be combined. That’s how we make things:

- bullets
- **bold**
- *italics*
- links

And by the end of this lesson, you should be able to do each of those things too, and more!

Despite these documents all starting as plain text, you can render them into HTML pages, or PDFs, or Word documents, or slides! The symbols you use to signal, for example, **bold** or *italics* is compatible with all of those formats.

Why use Python Markdown?

One of the main benefits is the reproducibility of using Python Markdown. Since you can easily combine text and code chunks in one document, you can easily integrate introductions, hypotheses, your code that you are running, the results of that code and your conclusions all in one document. Sharing what you did, why you did it and how it turned out becomes so simple - and that person you share it with can re-run your code and *get the exact same answers you got.* That’s what we mean about reproducibility. But also, sometimes you will be working on a project that takes many weeks to complete; you want to be able to see what you did a long time ago (and perhaps be reminded exactly why you were doing this) and you can see exactly what you ran AND the results of that code - and Python Markdown documents allow you to do that.

Another major benefit to Python Markdown is that since it is plain text, it works very well with version control systems. It is easy to track what character changes occur between commits; unlike other formats that aren’t plain text. For example, in one version of this lesson, I may have forgotten to bold **this** word. When I catch my mistake, I can make the plain text changes to signal I would like that word bolded, and in the commit, you can see the exact character changes that occurred to now make the word bold.

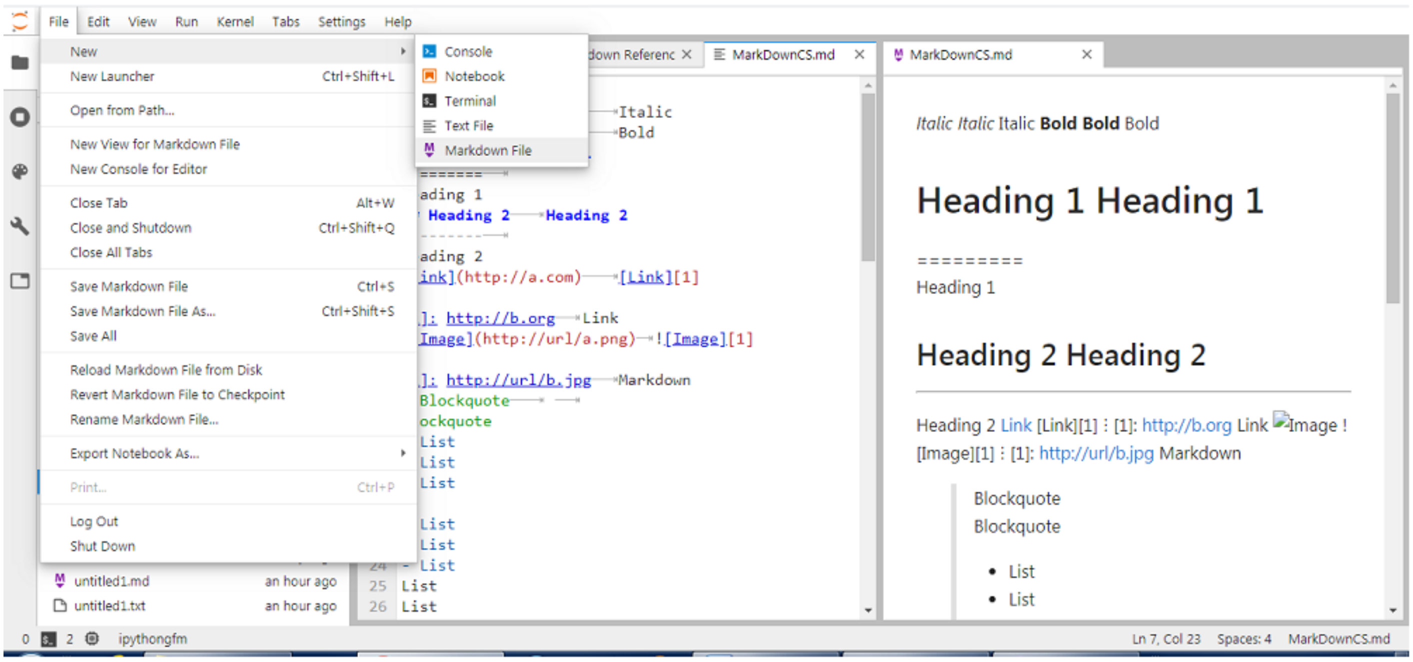
Installation

The easiest way to install Python-Markdown is simply to type the following command from the command line: `pip install markdown`

And that’s it, you are ready to go.

Getting started with Python Markdown

To create an Python Markdown document, in JupyterLab, go to File > New > Markdown File. You will be presented with the following window:



```
Italic
Bold
=======
ading 1
Heading 2 Heading 2
-------
ading 2
ink](http://a.com) [Link][1]
]: http://b.org Link
Image](http://url/a.png) ![Image][1]
]: http://url/b.jpg Markdown
Blockquote
ockquote
List
List
List
List
List
List
List
List
```

Italic Italic Italic **Bold Bold** Bold

Heading 1 Heading 1

=========
Heading 1

Heading 2 Heading 2

Heading 2 Link [Link][1] ⁝ [1]: http://b.org Link Image ! [Image][1] ⁝ [1]: http://url/b.jpg Markdown

> Blockquote
> Blockquote
>
> - List
> - List

Initiating an Python Markdown file

The default template for Python Markdown files

There are three main sections of the Markdown document. The first is the **header** at the top, bounded by the three dashes. This is where you can specify details like the title, your name, the date, and what kind of document you want output. If you filled in the blanks in the window earlier, these should be filled out for you.

Also on this page, you can see **text sections**, for example, one section starts with "## " - We'll talk more about what this means in a second, but this section will render as text when you produce the PDF of this file - and all of the formatting you will learn generally applies to this section.

The easiest way to see how each of these sections behave is to produce the PDF!

What are some easy Markdown commands?

At this point, I hope we've convinced you that Python Markdown is a useful way to keep your code/data and have set you up to be able to play around with it. To get you started, we'll practice some of the formatting that is inherent to Python Markdown documents.

To start, let's look at bolding and italicising text. To bold text, you surround it by two asterisks on either side. Similarly, to italicise text, you surround the word with a single asterisk on either side. `**bold**` and `*italics*` respectively.

We've also seen from the default document that you can make section headers. To do this, you put a series of hash marks (#). The number of hash marks determines what level of heading it is. One hash is the highest level and will make the largest text (see the first line of this lecture), two hashes is the next highest level and so on. Play around with this formatting and make a series of headers, like so:

```
# Header level 1
## Header level 2
### Header level 3...
```

One final thing we will go into detail on is making bulleted lists, like the one at the top of this lesson. Lists are easily created by preceding each prospective bullet point by a single dash, followed by a space. Importantly, at the end of each bullet's line, end with TWO spaces.

- Try
- Making
- Your
- Own
- Bullet
- List!

This is a great starting point and there is so much more you can do with Python Markdown. Thankfully, developers have produced an "Python Markdown cheatsheet" that we urge you to go check out and see everything you can do with Python Markdown! The sky is the limit!

Summary

In this lesson we’ve delved into Python Markdown, starting with what it is and why you might want to use it. We hopefully got you started with Python Markdown, first by installing it, and then by generating and knitting our first Python Markdown document.