



# Building Presentable Notebooks in Jupyter

A collection of tips, examples, and tools to make content in Jupyter easy to share with broad audiences

## Table of Contents

1. [Building a ToC with Header Links](#)
2. [Headings, Subheadings, and Text](#)
3. [Using HTML to Modify Text](#)
4. [Images, Videos, and Other Content](#)
5. [Colorizing Tabular Data](#)
6. [Interactive Tables](#)
7. [Download as CSV Widget/Button](#)
8. [Data Visualizations](#)
9. [Export as HTML or PDF](#)

## 1. Building a ToC with Header Links

A table of contents can be built in markdown using an ordered or unordered list. An ordered list is a list of items starting with a `number` and a period.

1. First
2. Second
3. Third

An unordered list can begin with an asterisk, hyphen, or plus sign but ends up as a bullet regardless of whichever is used.

- \* Item
- Item
- + Item

The table of contents displays the overall structure of a notebook and where something is located but it can be enhanced by using anchors. Add an anchor to a section heading to set it as a destination.

```
<h2 id="chapter1">Heading</h2>
```

Then that anchor can be referenced within the ToC so that clicking on it will link to the referenced section in the notebook.

```
[Heading](#chapter1)
```

Incorporating this technique makes a notebook, especially a large one, much easier to navigate.

## 2. Headings, Subheadings, and Text

Headings and subheadings can be created via hashes # in Markdown. A general way to go about using different levels of headings could be the following:

```
# Titles
## Major heading
### Subheading
#### Lower level subheading
##### Even lower level subheading
##### Lowest level subheading
```

You can see these same elements in HTML below:

```
<h1>Text</h1>
<h2>Text</h2>
<h3>Text</h3>
<h4>Text</h4>
<h5>Text</h5>
<h6>Text</h6>
```

Text inside sections can be modified in a few ways. Text can be **bold** ( ***bold*** ), *italicized* ( ***italicized*** ), or have ~~strikethrough~~ ( ~~strikethrough~~ ) applied. This is enough for simple notebooks but, for those instances where further modification of text is needed, inline HTML can be used.

## 3. Using HTML to Modify Text

While inline HTML is possible in Markdown, usage is slightly discouraged in the syntax notes because Markdown is meant to be widely approachable and easy to read/write/share.

Markdown's syntax is intended for one purpose: to be used as a format for writing for the web.

Markdown is not a replacement for HTML, or even close to it. Its syntax is very small, corresponding only to a very small subset of HTML tags. The idea is not to create a syntax that makes it easier to insert HTML tags. In my opinion, HTML tags are already easy to insert. The idea for Markdown is to make it easy to read, write, and edit prose. HTML is a publishing format; Markdown is a writing format. Thus, Markdown's formatting syntax only addresses issues that can be conveyed in plain text.

^You can use the greater than symbol `>` in front of text to create a blockquote like above.

Nonetheless, HTML can be used to colorize or highlight text to add an extra emphasis that Markdown does not inherently provide.

Use `<span style="color:red">colorize</span>` to colorize text and `<mark>highlight</mark>` to highlight text.

It is important to understand that using these methods reduces the compatibility of a notebook on different hosting sites. For instance, GitHub does not (currently) display text highlights. The best way to ensure compatibility is to export as an HTML or PDF file (see [Section 9](#)) and view appropriately.

Alternatively, you can upload a notebook on GitHub but view it through [nbviewer](#) or [Binder](#).

## 4. Images, Videos, and Other Content

When using Jupyter Lab, inserting images can be done by dragging a file into a cell and creating an attachment. This will automatically generate the Markdown syntax to display the image like below.

```
![Jupyter Logo](attachment:b148df2f-85ae-4f98-92cc-f37238e31acd.png)
```



Another method would be to use the URL of an image hosted online. [Imgur](#) is a free service that can be used to host images.

```
![Jupyter Logo](https://i.imgur.com/HaiZQxF.png)
```

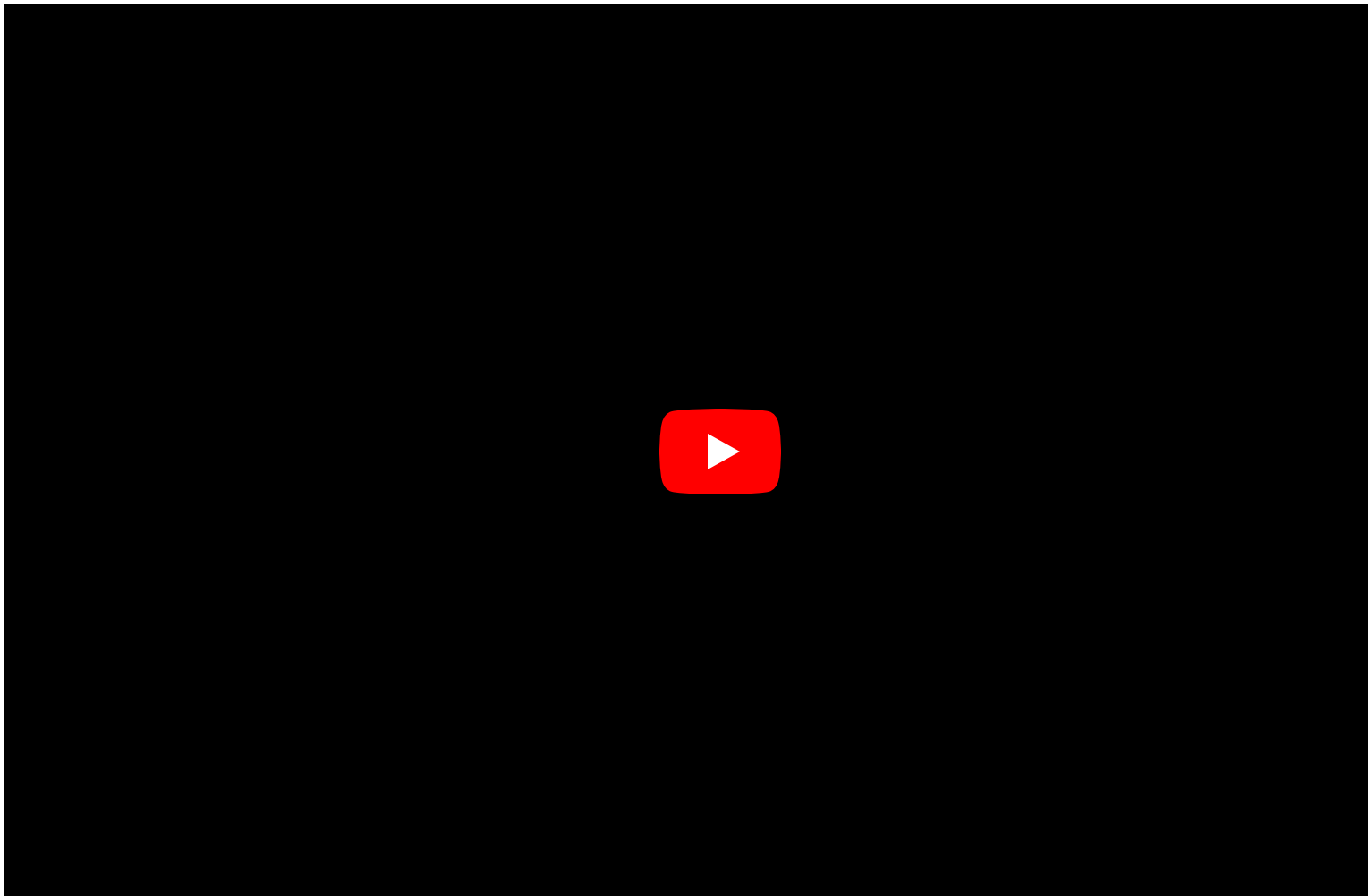
# Using iframe

If images/gifs are not enough, there is always the option to embed elements via `iframe` .

Many sites will provide embed code that can just be copied and pasted like this YouTube example.

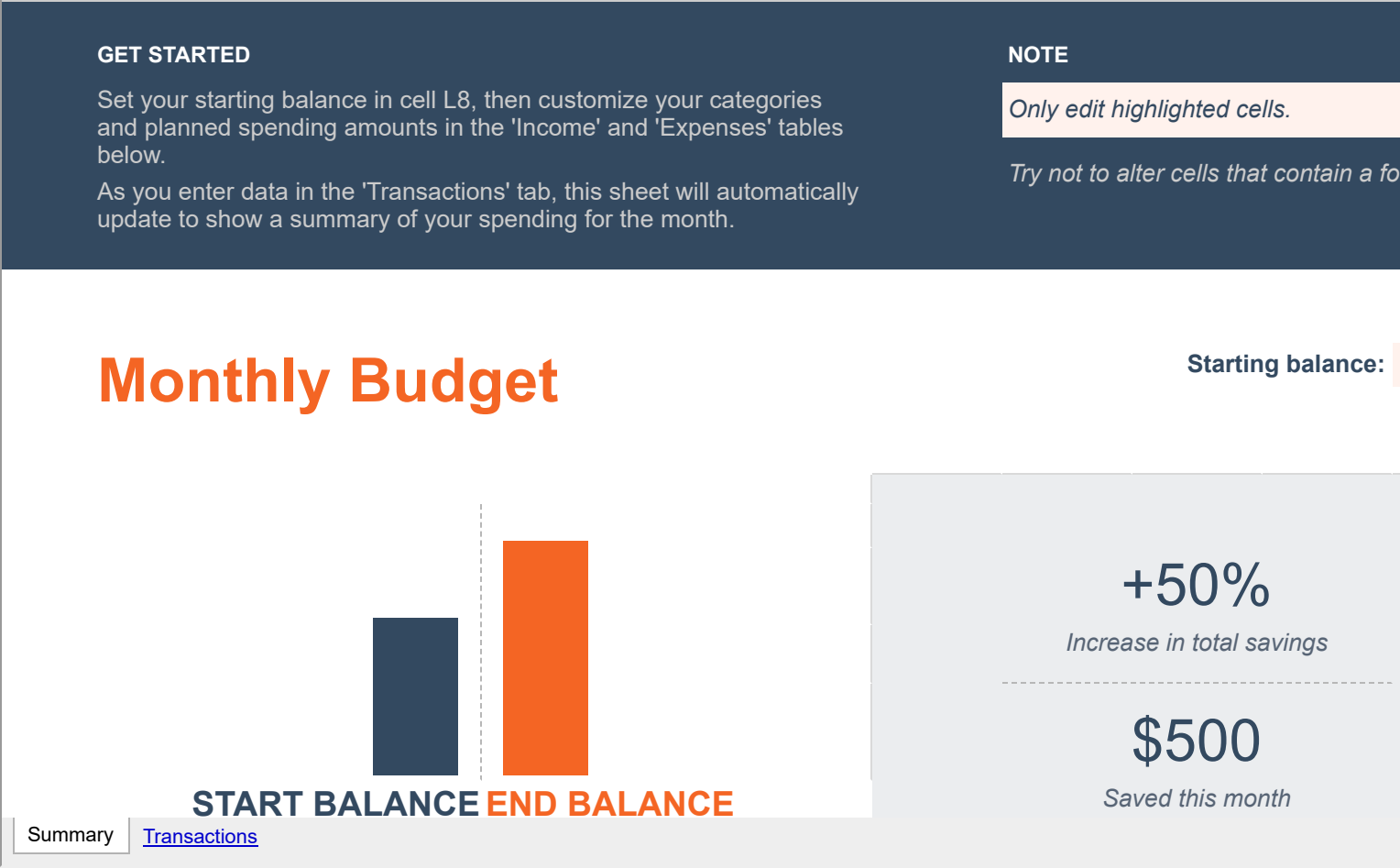
```
<iframe width="560" height="315" src="https://www.youtube.com/embed/yEic9z-Ad3k" title="YouTube video player" frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture" allowfullscreen></iframe>
```

Parameters like `height` and `width` can be changed to further customize the embedded element.



Just as with videos, `iframe` can be used to embed other web content in notebooks. There are plenty of options to explore but one noteworthy example is embedding a live Google Sheet. By selecting "Publish to the web" in Google Sheets, the code to embed is made available.

```
<iframe src="https://docs.google.com/spreadsheets/d/e/2PACX-1vQvEpJdNx17lFsoJEJuUKuAoNjB2GfnJSnsPC8CM_OqXNeK08uwLmrRVt0j0A6uswk0VFKJck6N9rqC/pubhtml?widget=true&headers=false"></iframe>
```



## 5. Colorizing Tabular Data

If you are using Jupyter with Python, it is virtually guaranteed that you are using Pandas to perform data manipulation. But what if you want to liven up the data being displayed in output cells to be more engaging and presentable?

Pandas has a `Styler` class to colorize tabular data displayed within notebooks to do just that.

The `Styler` creates an HTML `<table>` and leverages CSS styling language to manipulate many parameters including colors, fonts, borders, background, etc. This allows a lot of flexibility out of the box, and even enables web developers to integrate DataFrames into their exiting user interface designs.

A couple of examples can be found below but the [full documentation](#) shows many more.

### Simple Style Function

	A	B	C	D
0	-0.569107	-0.161194	-1.797622	-0.074493
1	1.129418	1.203531	0.271461	0.391286
2	1.182594	-1.888372	-0.242168	-0.540415
3	0.059033	0.260116	0.648308	1.213310
4	0.590691	0.828380	-0.654768	0.593242
5	-1.214150	1.435144	-0.106310	-0.803603
6	-0.039938	-0.093800	-1.316536	0.152418
7	0.998963	-0.973248	-1.049432	0.241919
8	0.325723	0.117656	-0.632185	-0.193620
9	0.547118	-0.054539	-0.184790	0.746192

## Seaborn Colormap Background Gradient

	A	B	C	D
0	-0.569107	-0.161194	-1.797622	-0.074493
1	1.129418	1.203531	0.271461	0.391286
2	1.182594	-1.888372	-0.242168	-0.540415
3	0.059033	0.260116	0.648308	1.213310
4	0.590691	0.828380	-0.654768	0.593242
5	-1.214150	1.435144	-0.106310	-0.803603
6	-0.039938	-0.093800	-1.316536	0.152418
7	0.998963	-0.973248	-1.049432	0.241919
8	0.325723	0.117656	-0.632185	-0.193620
9	0.547118	-0.054539	-0.184790	0.746192

## Bars in DataFrame

	A	B	C	D
0	-0.569107	-0.161194	-1.797622	-0.074493
1	1.129418	1.203531	0.271461	0.391286
2	1.182594	-1.888372	-0.242168	-0.540415
3	0.059033	0.260116	0.648308	1.213310
4	0.590691	0.828380	-0.654768	0.593242
5	-1.214150	1.435144	-0.106310	-0.803603
6	-0.039938	-0.093800	-1.316536	0.152418
7	0.998963	-0.973248	-1.049432	0.241919
8	0.325723	0.117656	-0.632185	-0.193620
9	0.547118	-0.054539	-0.184790	0.746192

## 6. Interactive Tables

The `itables` package in Python takes DataFrames to another level by adding features like pagination, sorting, filtering, and searching.

Refer to the [full documentation](#) for more information and advanced parameters but a few common examples are demonstrated below.

### Default Table

Show 

10 ▼

 entries

Search:

A ▲▼	B ▲▼	C ▲▼	D ▲▼
1.764052	0.400157	0.978738	2.240893
1.867558	-0.977278	0.950088	-0.151357
-0.103219	0.410599	0.144044	1.454274
0.761038	0.121675	0.443863	0.333674
1.494079	-0.205158	0.313068	-0.854096
-2.55299	0.653619	0.864436	-0.742165
2.269755	-1.454366	0.045759	-0.187184
1.532779	1.469359	0.154947	0.378163
-0.887786	-1.980796	-0.347912	0.156349
1.230291	1.20238	-0.387327	-0.302303

Showing 1 to 10 of 1,000 entries

Previous

1

2345...100Next

### Pagination Replaced with Vertical Scroll

Search:

A	B	C	D
1.764052	0.400157	0.978738	2.240893
1.867558	-0.977278	0.950088	-0.151357
-0.103219	0.410599	0.144044	1.454274
0.761038	0.121675	0.443863	0.333674
1.494079	-0.205158	0.313068	-0.854096
-2.55299	0.653619	0.864436	-0.742165
2.269755	-1.454366	0.045759	-0.187184
1.532779	1.469359	0.154947	0.378163
-0.887786	-1.980796	-0.347912	0.156349
1.230291	1.20238	-0.387327	-0.302303
-1.048553	-1.420018	-1.70627	1.950775
-0.509652	-0.438074	-1.252795	0.77749

Showing 1 to 1,000 of 1,000 entries

### Include a Table Caption

Show 

10

 entries

Search:

This DataFrame has 1000 Rows and 4 Columns

A	B	C	D
1.764052	0.400157	0.978738	2.240893
1.867558	-0.977278	0.950088	-0.151357
-0.103219	0.410599	0.144044	1.454274
0.761038	0.121675	0.443863	0.333674
1.494079	-0.205158	0.313068	-0.854096
-2.55299	0.653619	0.864436	-0.742165
2.269755	-1.454366	0.045759	-0.187184
1.532779	1.469359	0.154947	0.378163
-0.887786	-1.980796	-0.347912	0.156349
1.230291	1.20238	-0.387327	-0.302303

Showing 1 to 10 of 1,000 entries

Previous

1

2345...100Next



# Search by Column

A B C D

Loading... (need [help?](#))

## Colorizing Cell Content via JavaScript

Show 

10

 entries

Search:

A	B	C	D
1.764052	0.400157	0.978738	2.240893
1.867558	-0.977278	0.950088	-0.151357
-0.103219	0.410599	0.144044	1.454274
0.761038	0.121675	0.443863	0.333674
1.494079	-0.205158	0.313068	-0.854096
-2.55299	0.653619	0.864436	-0.742165
2.269755	-1.454366	0.045759	-0.187184
1.532779	1.469359	0.154947	0.378163
-0.887786	-1.980796	-0.347912	0.156349
1.230291	1.20238	-0.387327	-0.302303

Showing 1 to 10 of 1,000 entries

Previous

1

2

3

4

5

...

100





Next

## 7. Download as CSV Widget/Button

Sometimes it is not just enough to present data, but make it available to others to download. In the future, `itables` may be updated to include "Copy", "CSV", "PDF" and "Excel" buttons according to its author. For now, an alternative approach can be applied using `base64` and HTML widgets. The major caveat being that this method will only work on small to medium sized DataFrames.

In the examples below, each DataFrame is displayed with an accompanying "Download as CSV" option.

Search:

A 	B 	C 	D 
-0.569107	-0.161194	-1.797622	-0.074493
1.129418	1.203531	0.271461	0.391286
1.182594	-1.888372	-0.242168	-0.540415
0.059033	0.260116	0.648308	1.21331
0.590691	0.82838	-0.654768	0.593242
-1.21415	1.435144	-0.10631	-0.803603
-0.039938	-0.0938	-1.316536	0.152418
0.998963	-0.973248	-1.049432	0.241919
0.325723	0.117656	-0.632185	-0.19362
0.547118	-0.054539	-0.18479	0.746192

Showing 1 to 10 of 10 entries





[Download as CSV](#)

Show 

10 

 entries

Search:

A 	B 	C 	D 
1.764052	0.400157	0.978738	2.240893
1.867558	-0.977278	0.950088	-0.151357
-0.103219	0.410599	0.144044	1.454274
0.761038	0.121675	0.443863	0.333674
1.494079	-0.205158	0.313068	-0.854096
-2.55299	0.653619	0.864436	-0.742165
2.269755	-1.454366	0.045759	-0.187184
1.532779	1.469359	0.154947	0.378163
-0.887786	-1.980796	-0.347912	0.156349
1.230291	1.20238	-0.387327	-0.302303

Showing 1 to 10 of 1,000 entries

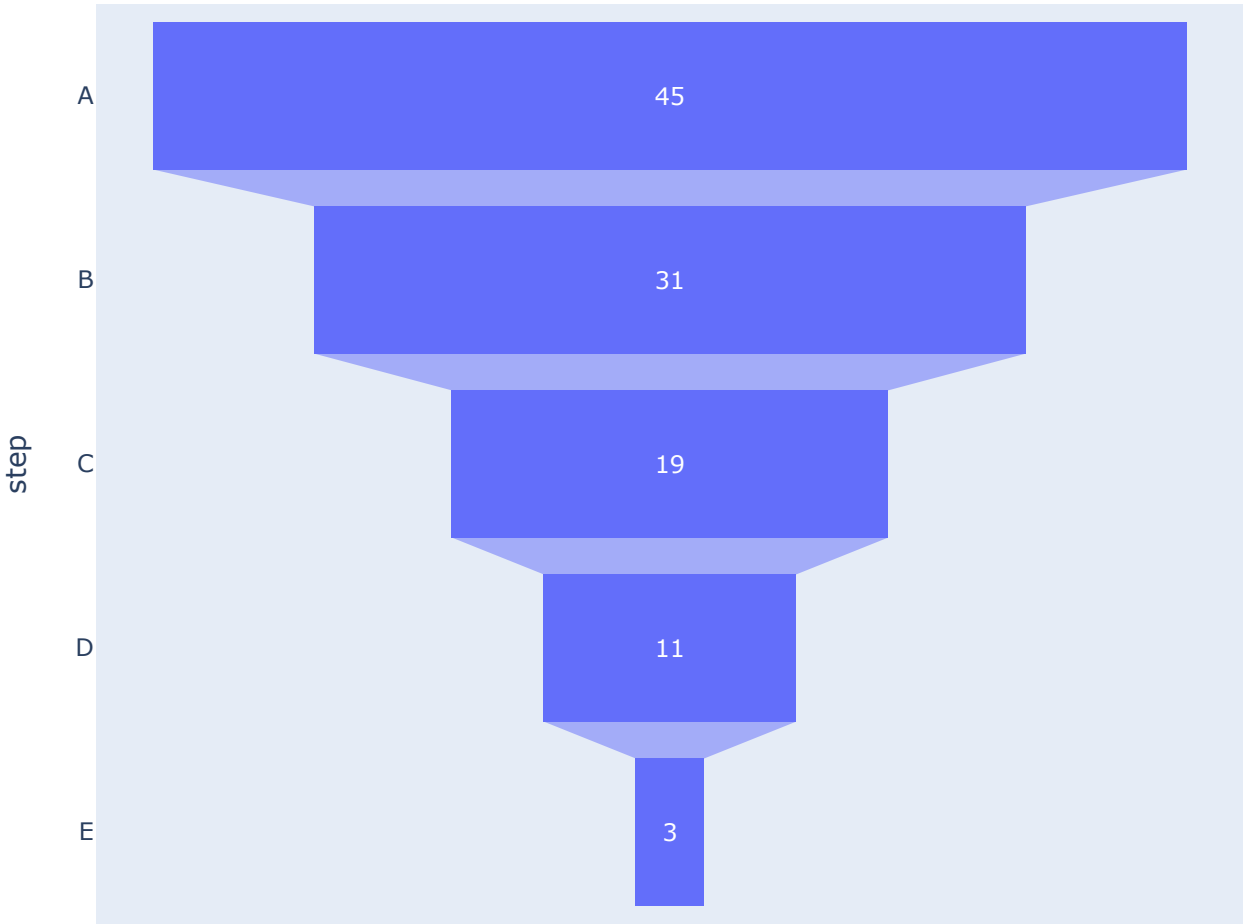
[Download as CSV](#)

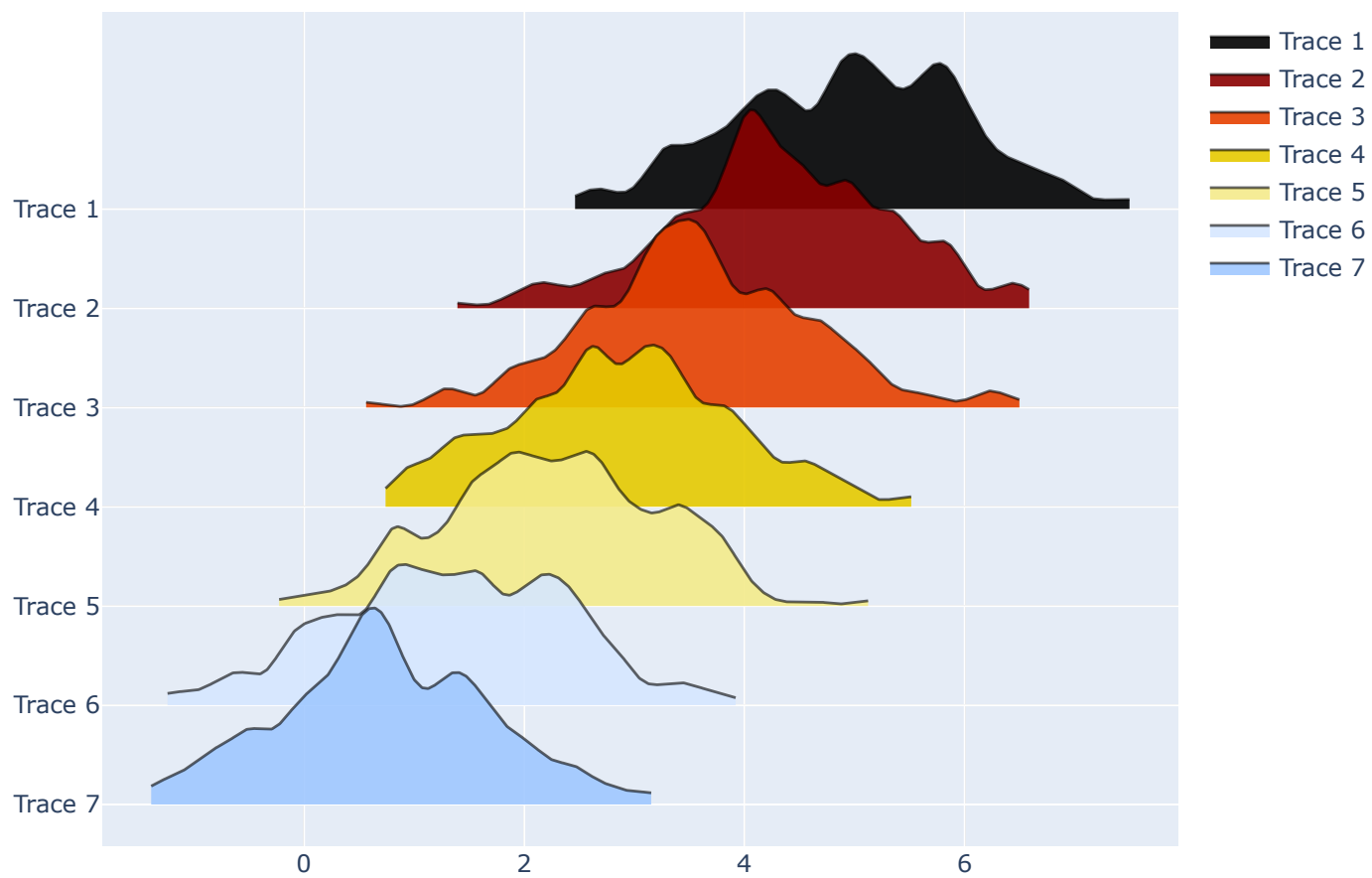
## 8. Data Visualizations

A notebook's ability to contain data visualizations is one of the major reasons why it is so prominently used for data science projects by enthusiasts and professionals alike.

There are many data visualization options available when using Python but the examples here are made with **Plotly** since it can be used to make both interactive and static data visualizations. Learn more about Plotly by visiting [their website](#).

## Interactive Plotly Visualizations





Votes

3500

3000

2500

2000

1500

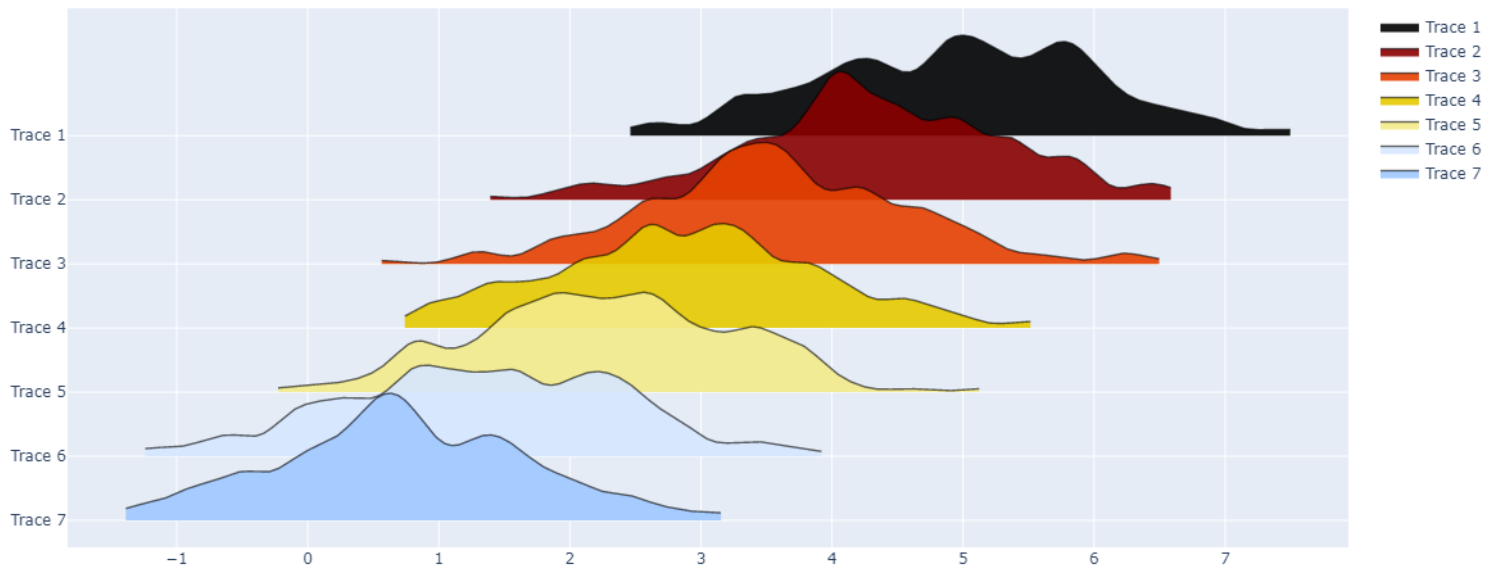
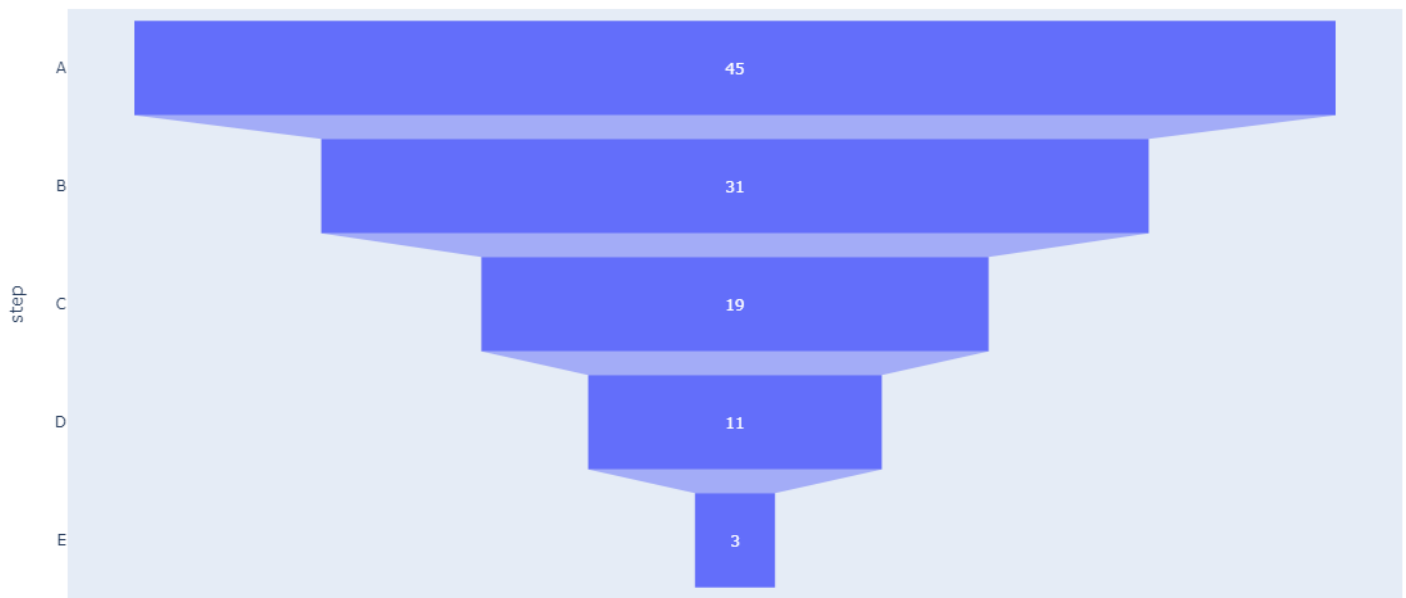
1000

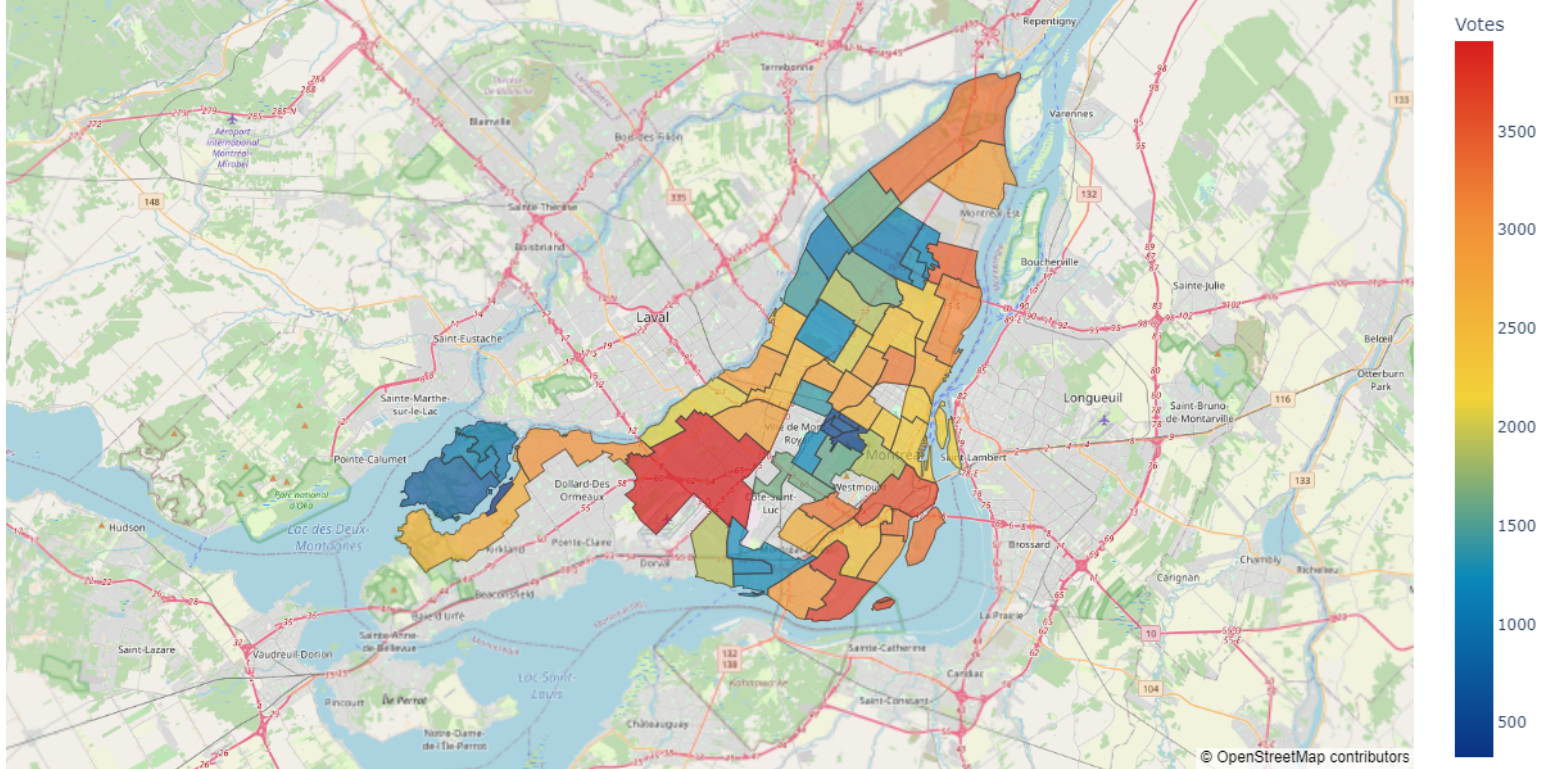
500



## Static Plotly Visualizations

All of the visualizations above can be rendered statically by using `show(renderer='png')` so that the default renderer is not used.





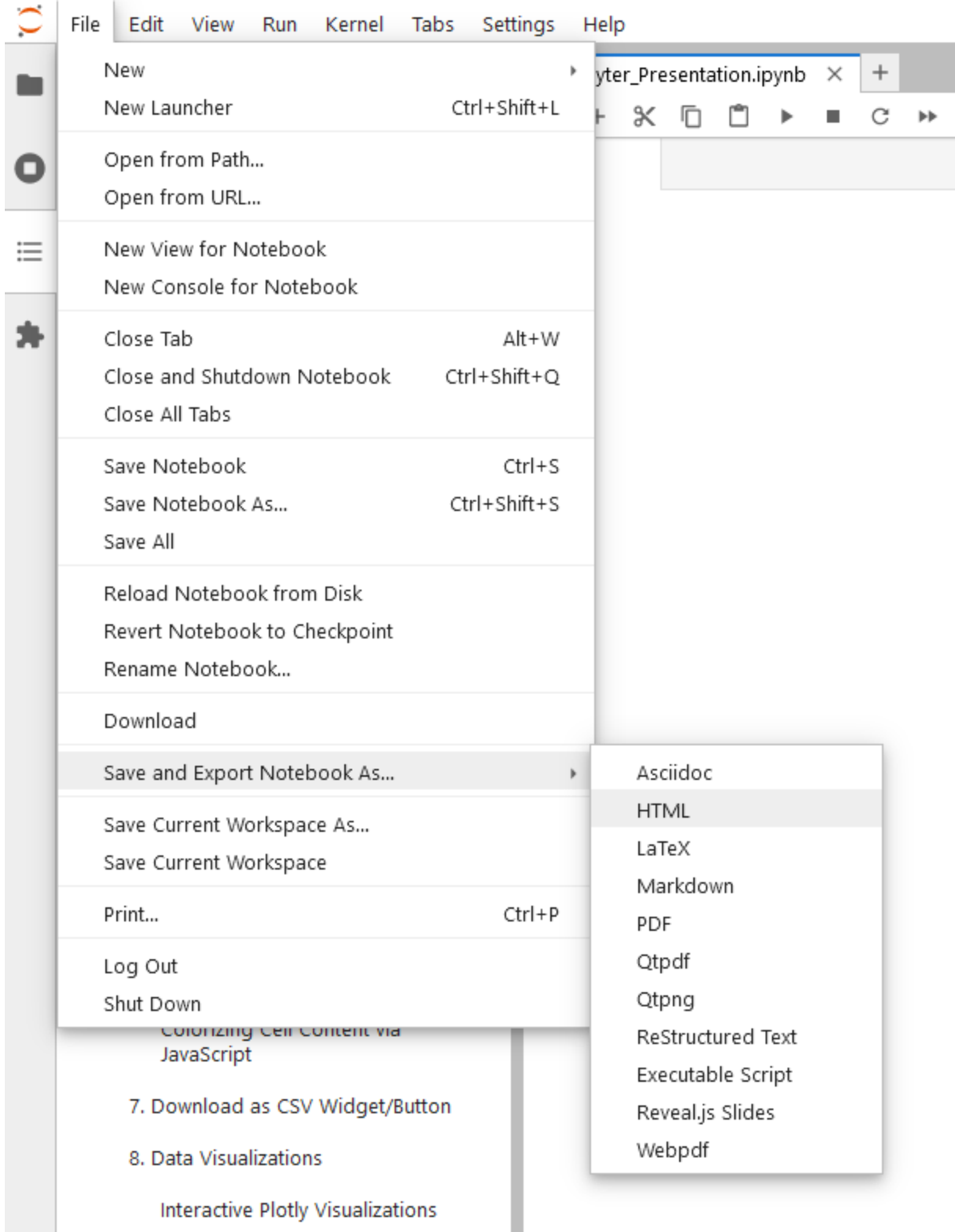
## 9. Export as HTML or PDF

A notebook can be shared as a `.ipynb` file but only familiar audiences will be able to view and use it. For others, it is helpful to export as HTML or PDF.

### Save and Export Notebook As...

Jupyter allows you to export notebooks as various formats but export options are completely dependent on your `nbconvert` configuration so it will be helpful to review the [documentation](#) to learn more.

Selecting an export format via this interface invokes `nbconvert` but it is also usable as a command line tool.



## HTML

The command line syntax to run the `nbconvert` script for HTML is:

```
jupyter nbconvert --to HTML <name of notebook>.ipynb
```

This will generate an HTML version of the notebook in the same location as the notebook.

## PDF (Web)



The simplest way to export a notebook as a PDF file would be to use `--to webpdf` which first renders to HTML and then exports as PDF. The webpdf exporter requires the `pyppeteer` library to function successfully so ensure that it is installed first.

```
jupyter nbconvert --to webpdf <name of notebook>.ipynb
```

This will generate a PDF version of the notebook that is entirely static so it will be absolutely necessary to plan the content of a notebook around this hindrance if a PDF export is needed

## Bulk Convert Notebooks

In situations where there are more than a few notebooks to convert, the tedious act of listing them out individually can be avoided by using the asterisk wildcard `*` in front of the `.ipynb` file extension.

```
jupyter nbconvert --to <format> *.ipynb
```

This will bulk convert **ALL** notebooks in the current directory into whichever format is specified.

## Additional Options

There are additional flags that can be specified like `--template` and `--theme` to dictate the overall aesthetic of the export but there are also some specific ways to augment an export.

- Use `--no-input` to remove all code cells.
- Use `--no-prompt` to remove all input and output prompts.
- Use `--embed-images` to embed images as base64 data URLs.
- Use `--execute` to execute a notebook prior to export.

To create a presentation-ready export from *within* a notebook.

```
!jupyter nbconvert --execute --to <format> --no-prompt --no-input --embed-images <name of notebook>.ipynb
```

Leading with an exclamation `!` is used to execute commands from the underlying operating system when working inside Jupyter.