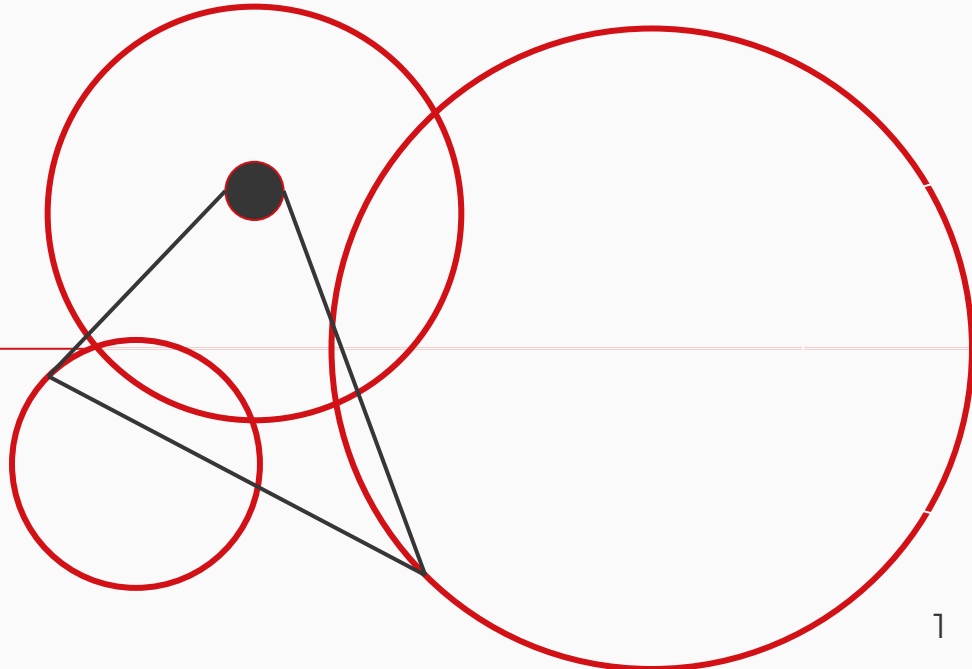
Using Google Colab and GitHub

Fall 2025

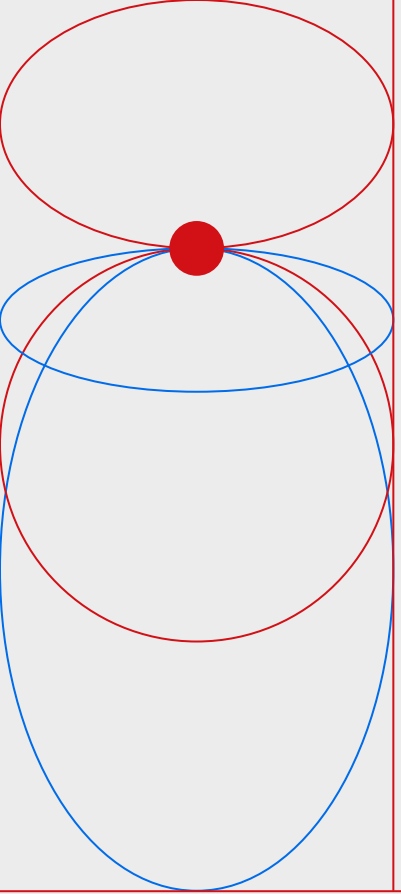


Written by Jessica Huynh-Westfall, PhD
SJSU CS 133

Agenda

CS 133 - Data visualization

- Setting up GitHub
- What is Google Colab?
- Let’s get some practice in



Version control

Using git and GitHub to maintain code

Git is one of the most common version control systems

https://git-scm.com/

Web-based service that provides hosting of your **git** repository.

https://github.com/

Version control

Version control track changes

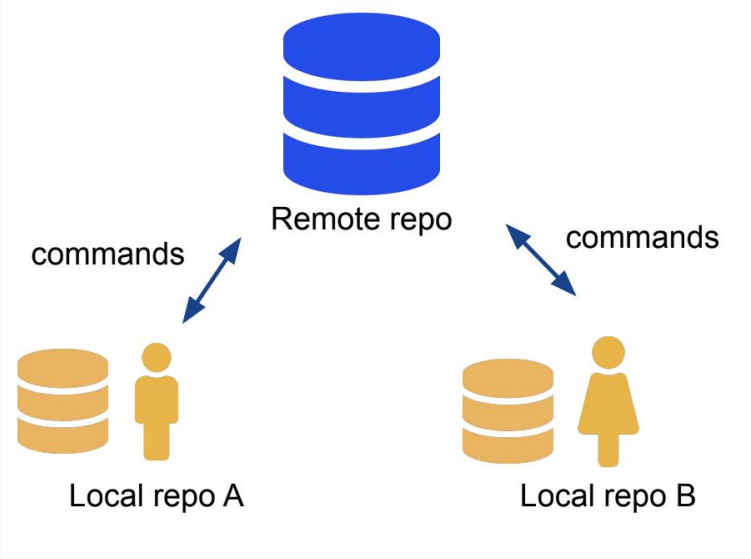
Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later

- track changes with “snapshots”
- allows user to revert to an earlier snapshot

Git and version control

Using Git to manage version control

- Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later
- Git is one of the most common version control systems
- GitHub is a web-based service that provides hosting, utilizing the Git



Using GitHub to store files and code

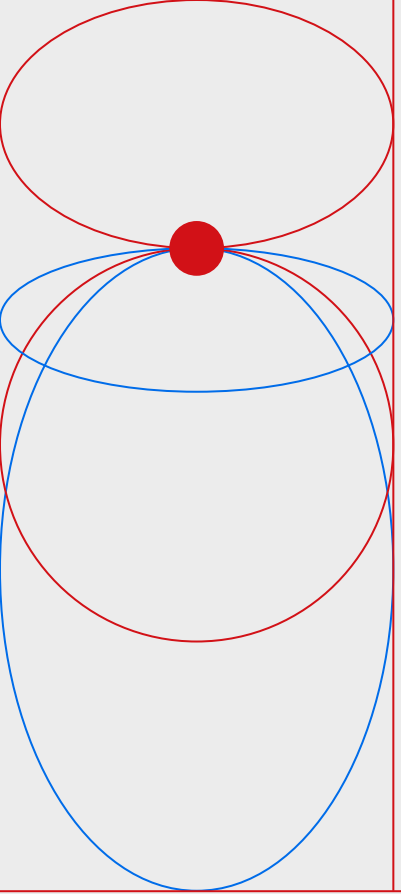
Use GitHub to manage your code and changes

- Store, change, merge and collaborate on files or code
- Use Git for version control
- Visibly tracking iterative changes
- Team can look back at previous versions of the code to trac
  changes and reverse if necessary
- Work with unlimited collaborators on project

Course GitHub to store files and code

Git repo for class files

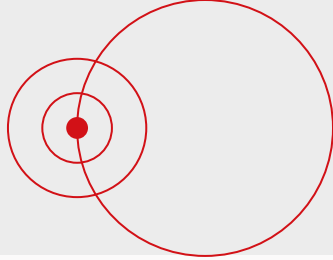
- Clone the repo https://github.com/CS133-DataVisualization/CS133-classfiles
- This repo will be use to download data files, collab notebooks, and other class related files.



Google Colab

Notebook that allows for rich text and executable code

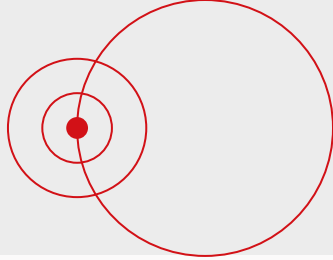
Why are we using Google Colab?



- Cloud-based Jupyter notebook environment
- No setup, easy sharing/collaborating
- Write and execute Python code
- Free GPU and TPU (eg., TensorFlow, PyTorch)
- Import datasets from Kaggle
- Import to GitHub

- Notebooks are ideal for data analysis because they can include:
  - Nicely formatted text (e.g., text that describes the data)
  - Code and code output
  - Spaces for displaying plots

Markdown language



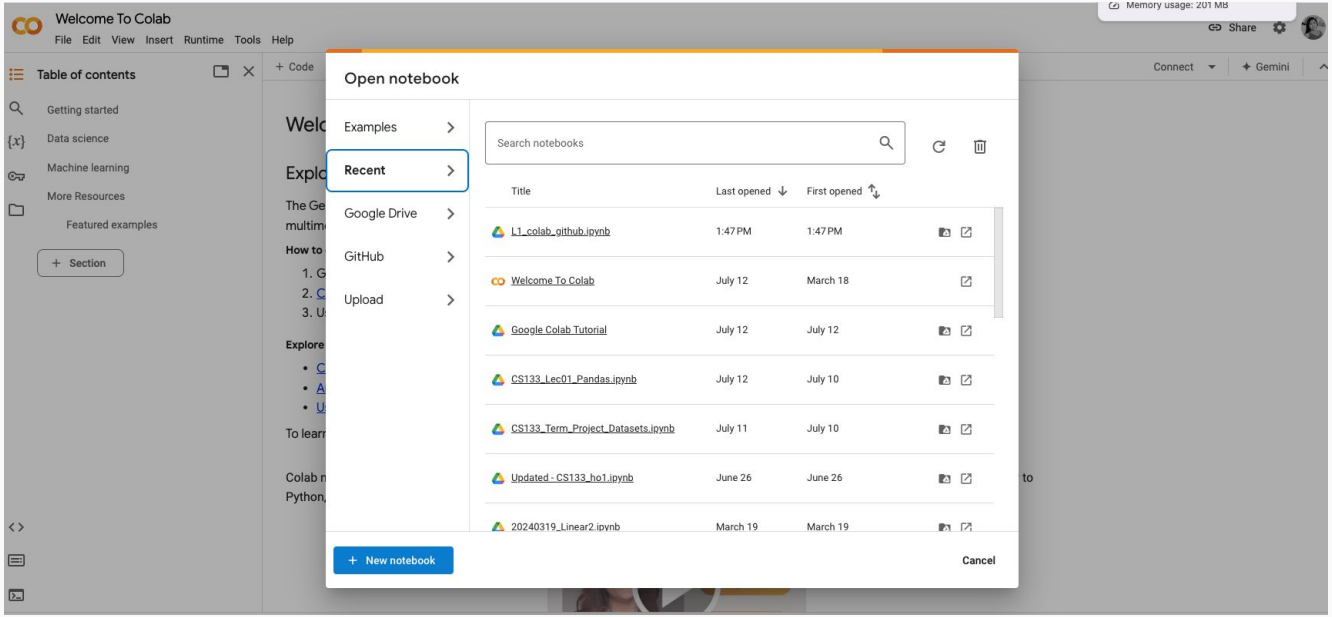
What is markdown?

Formatting rich text cells as simple markup language. The Markdown source shows the source text and the rendered version.

Colab has two type of cells; text and code.

Getting started

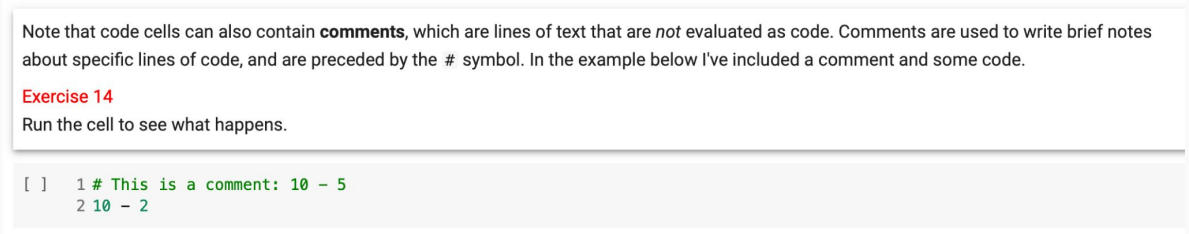
- Need Google account, GitHub account and Kaggle (optional)
- Open https://colab.research.google.com and click on + New notebook



Notebook example

- Pull the GitHub repository to make a copy of the in-class hands-on notebook.

- Each bit of text or code is referred to as a **cell**

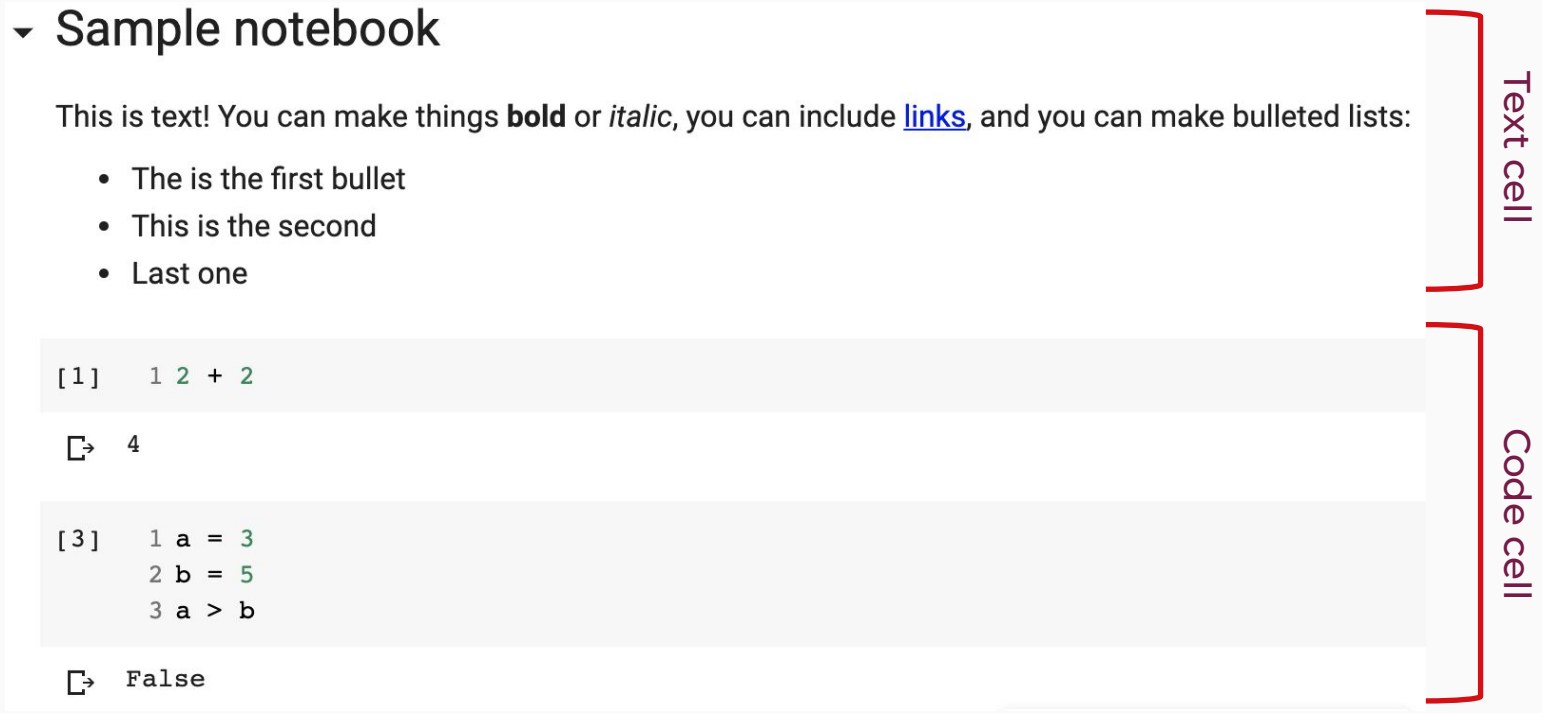


Text cell

Code cell

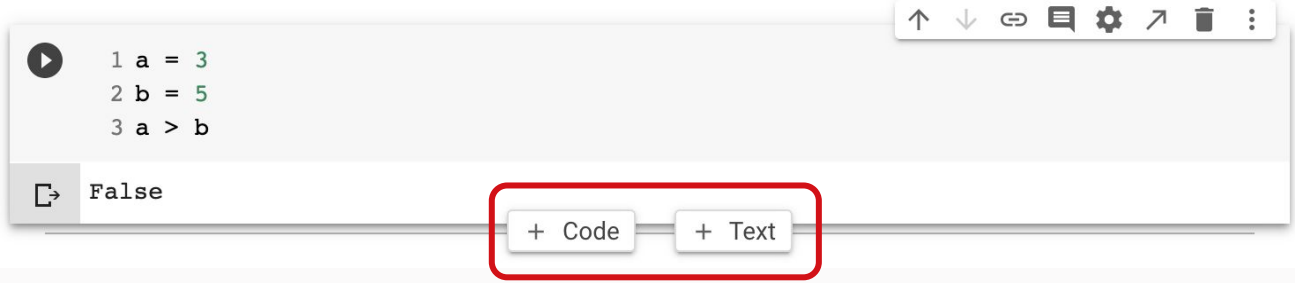
Google Colab sample notebook

Each bit of text or code is referred to as a cell

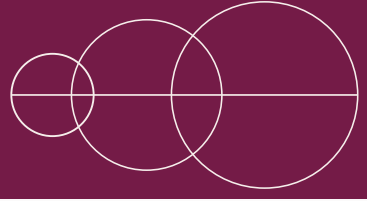


Creating new cells

- Some exercises will ask you to create a new cell
  - Single click on an existing text/code cell, and then hover your mouse towards the bottom edge of the cell, in the center
  - You can select either a Code cell or Text cell



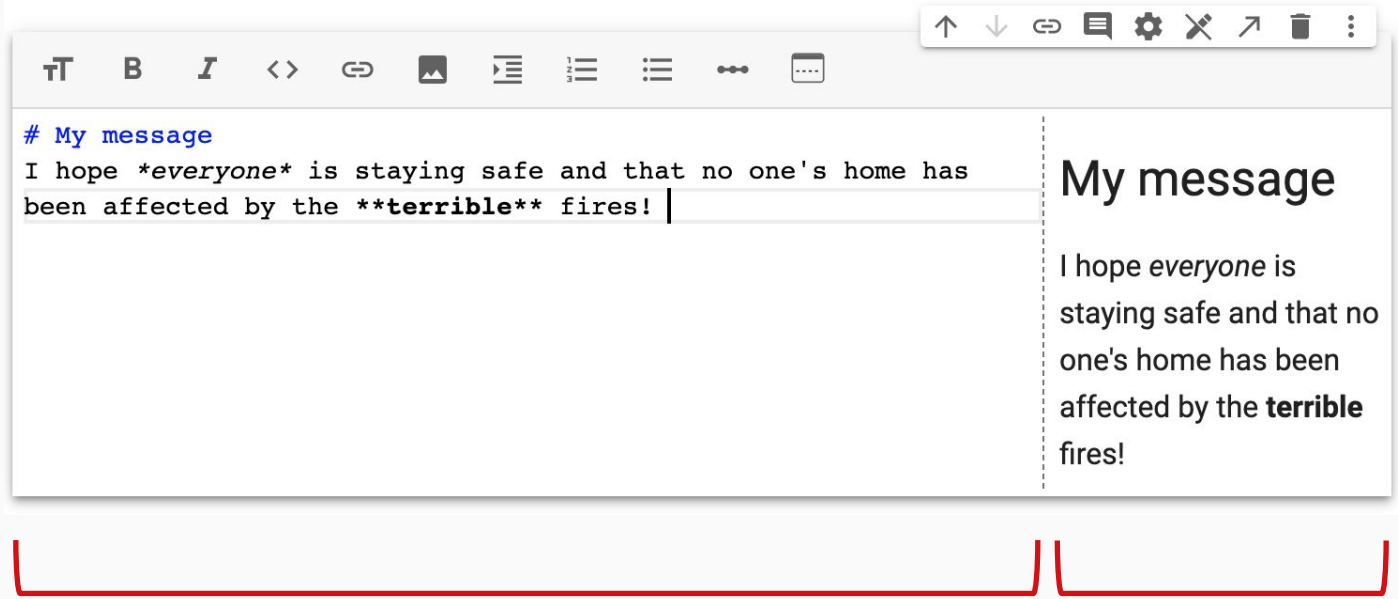
Formatting text cells



- Colab has buttons for things like:
  - Changing font size
  - Making things **bold** and *italic*
  - Creating numbered/bulleted lists
  - Creating links

- Note: you can make these formatting changes by typing, i.e., without clicking buttons
  - As you work through the hands-on, you'll learn how to do this

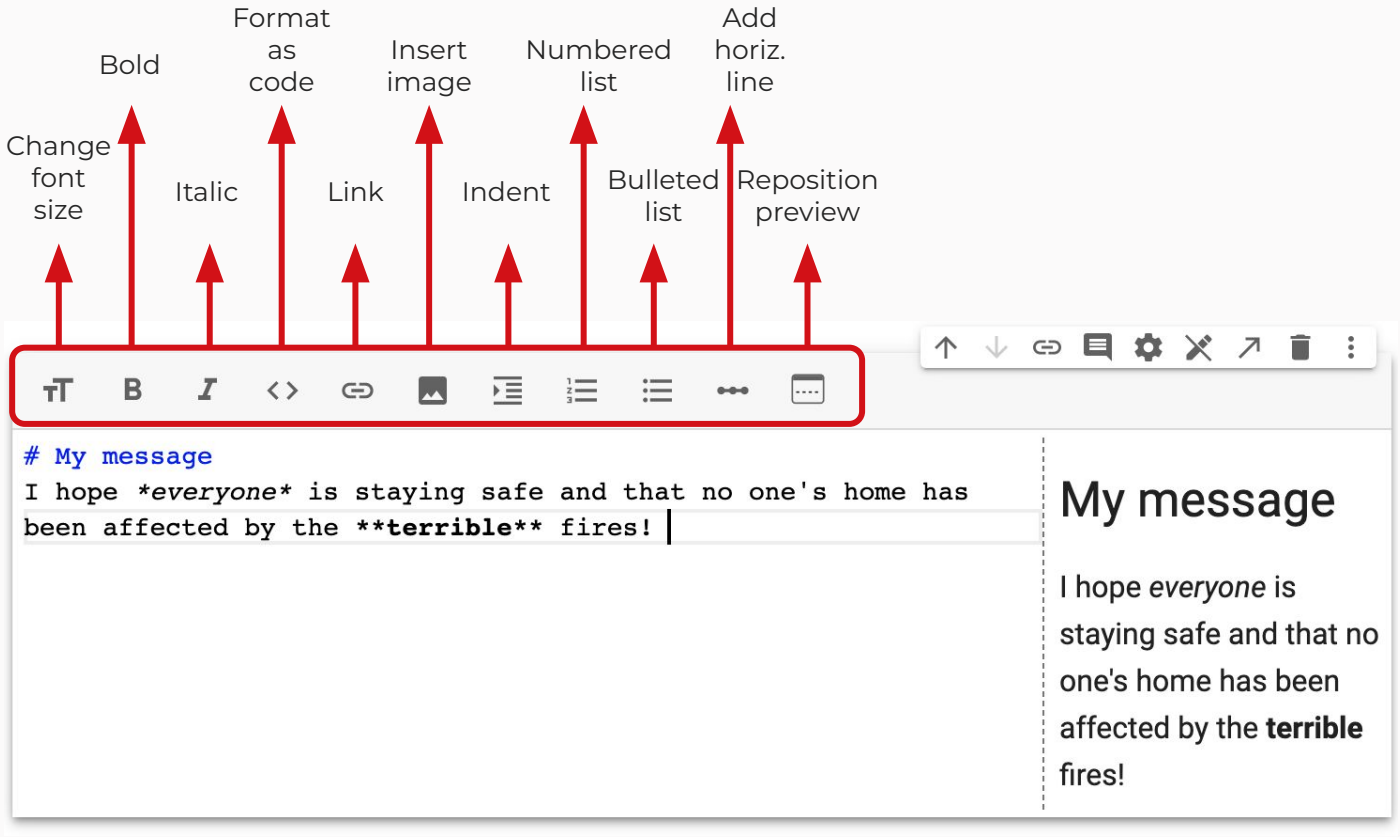
Formatting text cells



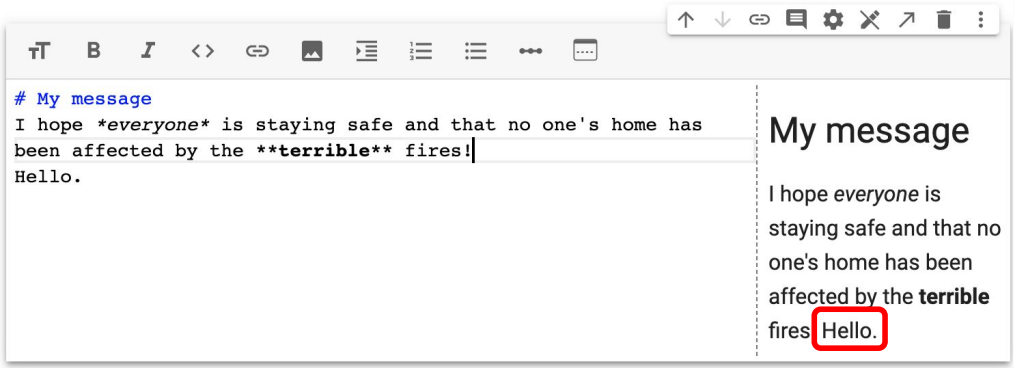
You'll type here, on the left

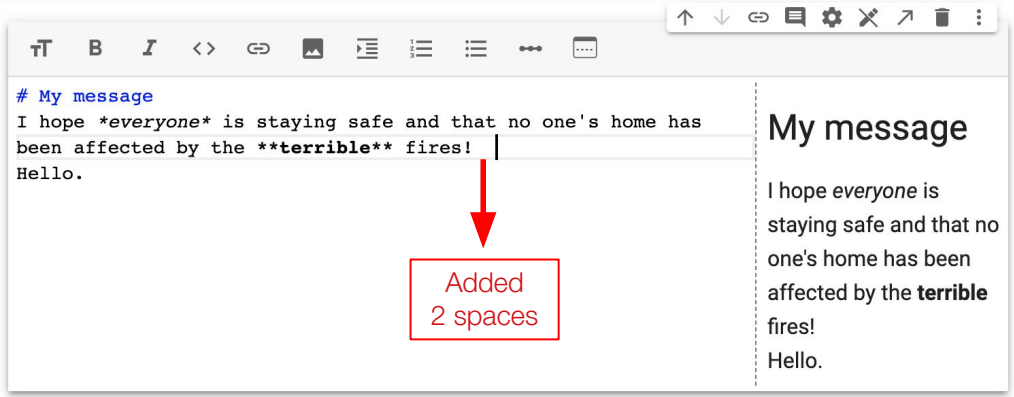
On the right is a preview

Formatting text cells



Formatting text cells





To create a new line inside a text cell, use 2 spaces ␣ ␣ at the end of the line, then hit enter

“Running” a cell

When editing a text cell, the left side looks rather ugly. To make things look nice (like the preview), we need to “run” the cell.

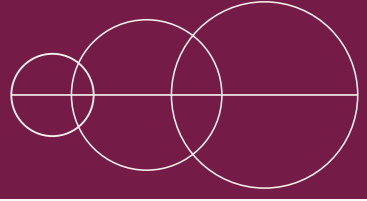
Simultaneously hit **Shift** and **Enter** to run a cell

-or-

Hit the  (play) button

This is true for code cells too.

Code cells to run python scripts



Code cells can run simple and complex Python code

An example is using Python as a calculator to run simple mathematical expressions.

- Type in a simple mathematical expression and then run the cell
- The output (found under the code cell) will be the answer

Running mathematical expression

To run simple mathematical expression

```
[1]   1 2 + 2
```

```
4
```

```
[3]   1 a = 3
      2 b = 5
      3 a > b
```

```
False
```

More mathematical expression

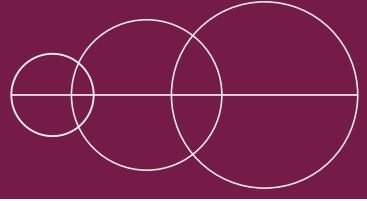
LaTeX for rendering mathematical equations and symbol

```
### You can also display math symbol using `$$`
$y = \sqrt{k}$ + $x^2$ - $sin(y)$
```

You can also display math symbol using `$$`

$$y = \sqrt{k} + x^2 - sin(y)$$

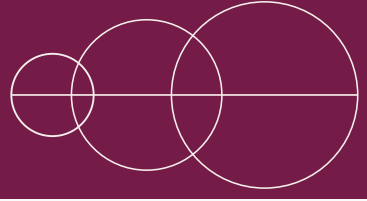
Common LaTeX commands



Fraction	\frac{numerator}{denominator}		
Subscript	x_i	Superscript	x^2
Greek	\alpha, \beta, \sigma		
Symbols	\infty, \int, \sum		
Functions	\sin, \cos, \log		
Matrix	\begin{matrix} ... \end{matrix}		

Cheat sheet for writing mathematics with LaTeX

Comments within code cells



Write comments about the code

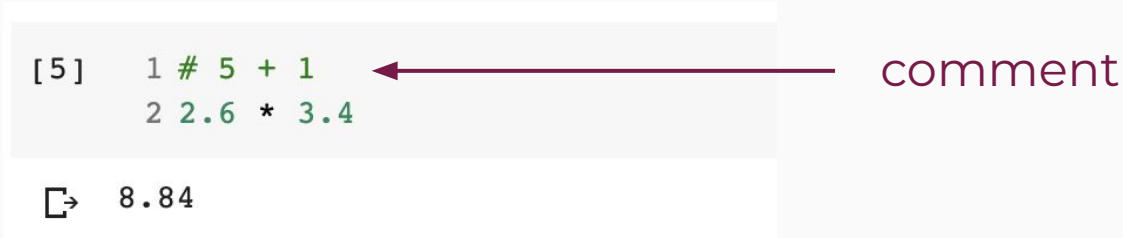
Within a code cell, you can also include comments about the code

Comments: Lines of text within a code cell that are not evaluated as code

- Used for writing brief notes about certain lines of code
- Preceded by the # symbol
- For multi-line comments use ‘’’ ‘’’

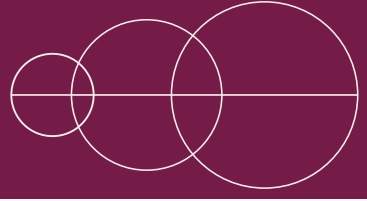
Comments within code cells

When running the following cell, we see the output of the code (line 2) and not the *comment* (line 1)



Note: Please put a space after the # sign when writing comments

Text vs. Comments

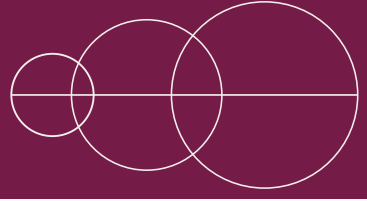


When should you use a text cell, and when should you write a comment in a code cell?

- **Comments**
  - Generally very short (less than a sentence)
  - Used to provide notes about specific lines of code

- **Text cells**
  - Appropriate for longer text or text that is formatted

Text vs. Comments



Practice notebook

In this notebook, I'll be practicing some simple Python commands. For example:

- Conditional execution
- Iteration

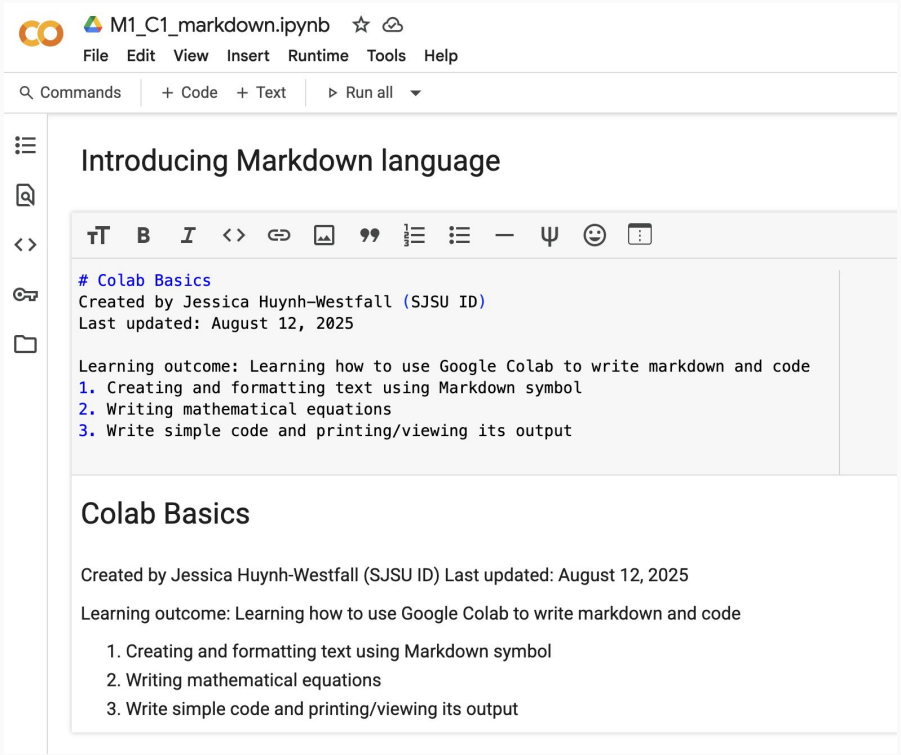
You'll be learning both of these skills later in the semester!

```
# conditional execution
x = 3
if x > 1:
    print('Greater than 1')
else:
    print('Less than or equal to 1')

# iteration
my_list = [1, 2, 3]
for num in my_list:
    print(num)
```

```
Greater than 1
1
2
3
```

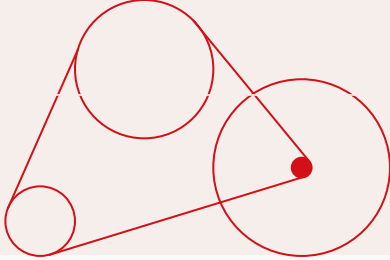
Notebook documentation



At the top of every notebook, it's helpful to include details like:

- Notebook title, using heading 1
- Name of person who created the notebook (or your name and SJSU ID, if it's an assignment)
- Date that you last updated the notebook
- Purpose of the notebook

Class activity



1. We will work on M1_C1_markdown.ipynb. Download from the class repository.
2. Work on the in-class CL1.1 and CL1.2
3. Enter the code you wrote for CL1.2 into the CL1_2 quiz on Canvas. Access code: “math”