Jupyter Notebook: An Introduction

- The Jupyter Notebook is an open-source web application that you allows you to author engaging documents that combine live-code with narrative text, equations, images, video, and visualizations.

- By encoding a complete and reproducible record of computation, the documents can be shared with others.

Starting the Jupyter Notebook Server

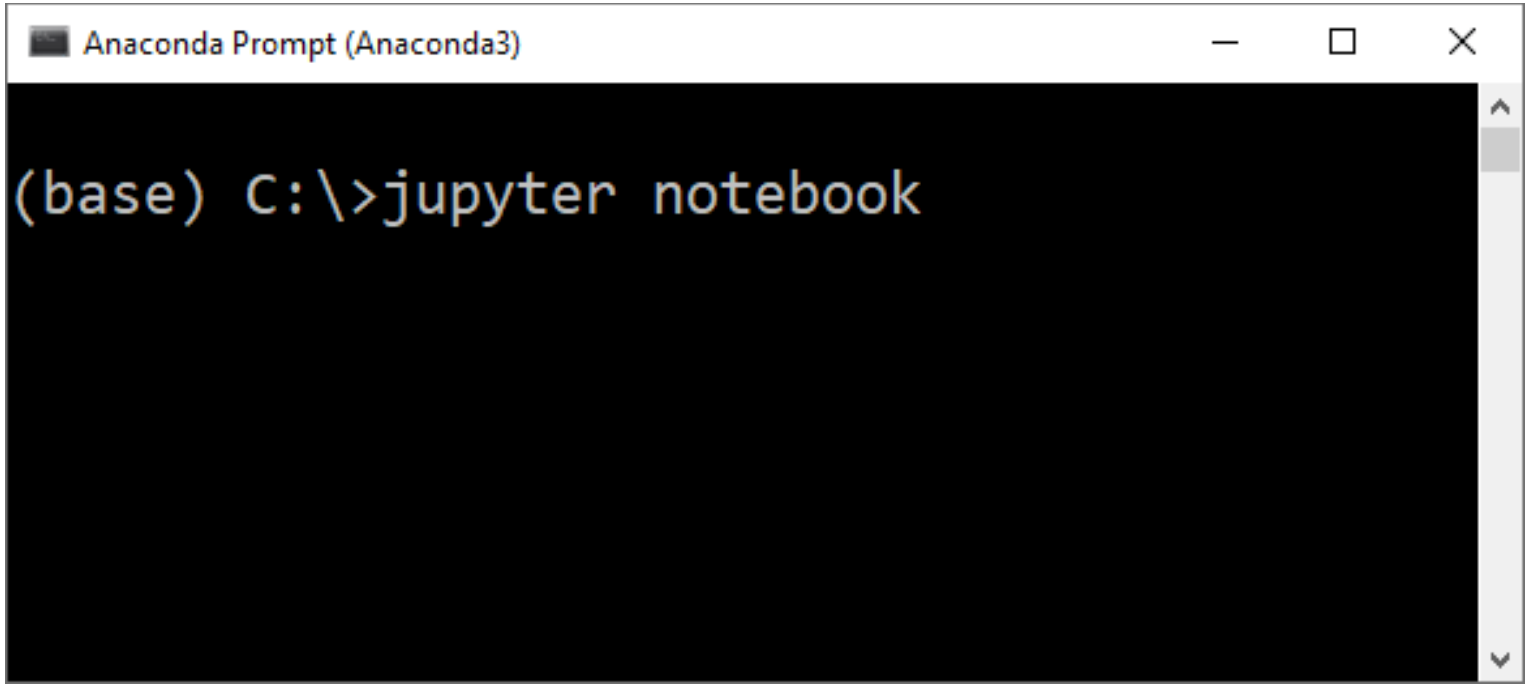
- The easiest way to start up Jupyter is by launching it through Anaconda Navigator

Click here

Starting the Jupyter Notebook Server

- Alternately, you can run the following command in a terminal window:

```
jupyter notebook
```

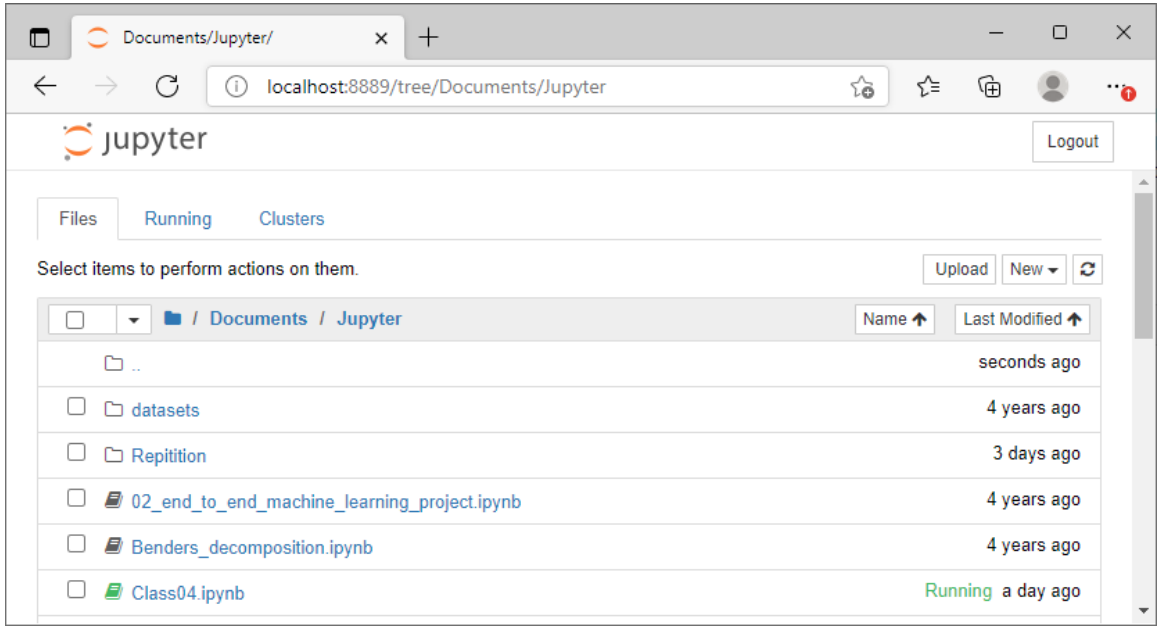


Jupyter Notebook Server

- When you start Jupyter, it will open your default browser to the following URL:

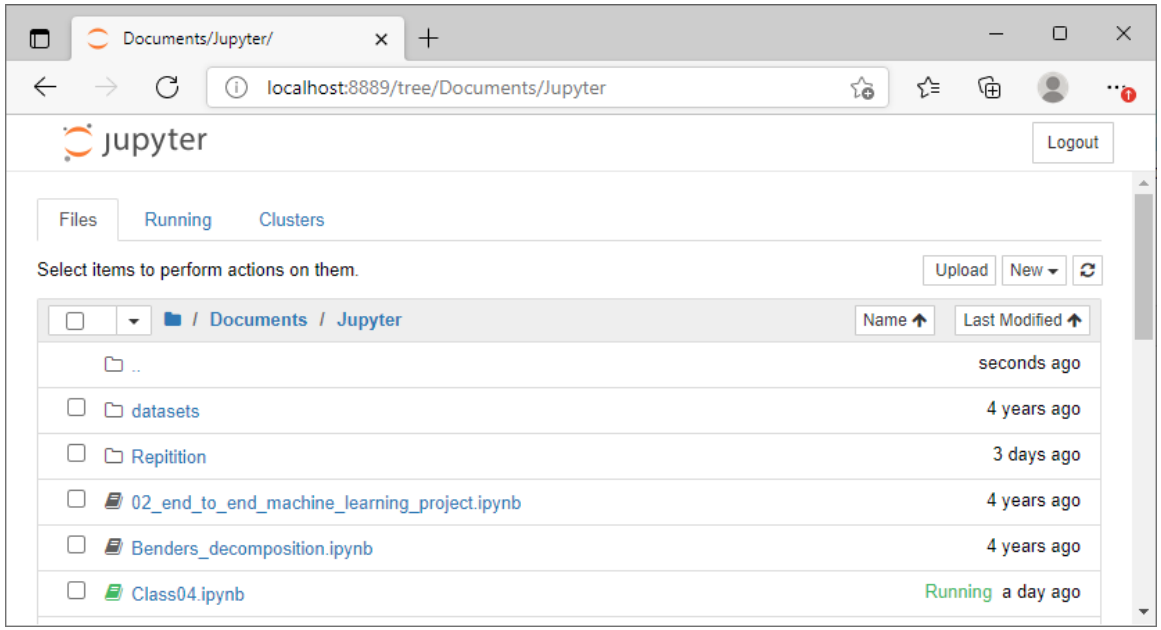
  `http://localhost:8888/tree`

- Your browser should look something like this:



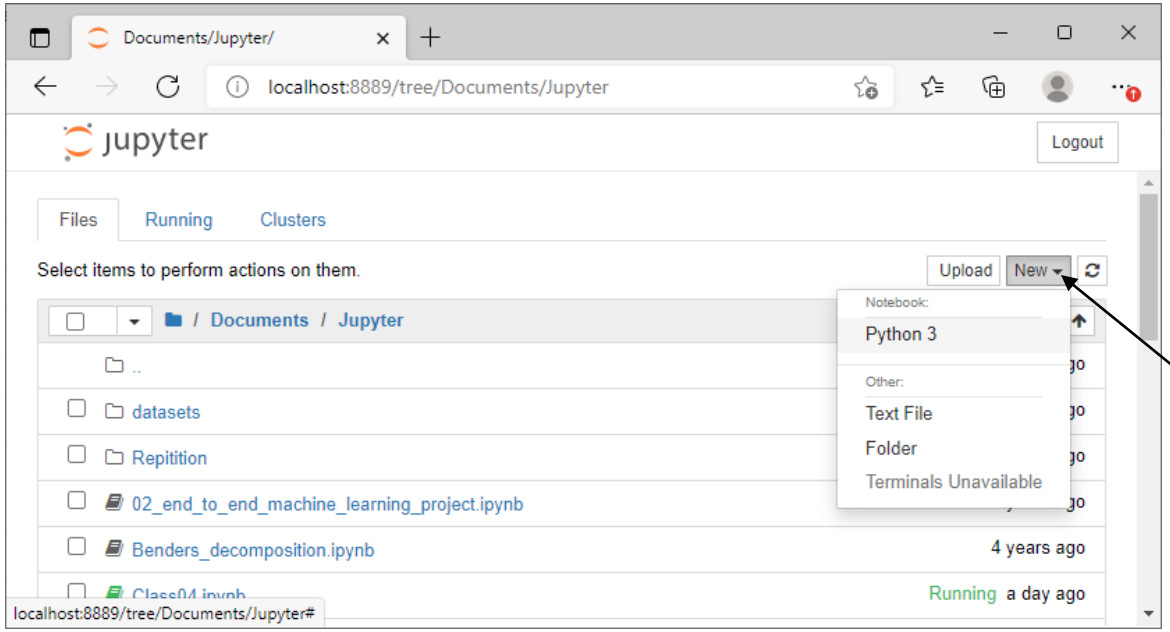
Notebook Server

- Note that right now you are not actually running a Notebook, but instead you are just running a Notebook server.



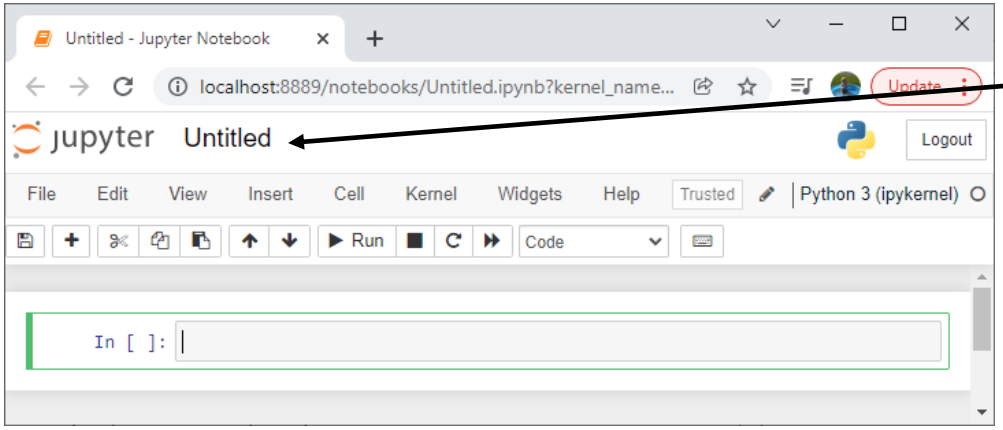
Creating a Notebook

- To create a new notebook, click the `New` button (upper right) and it will open a list of choices (depending on your installation). Let's choose `Python 3`.

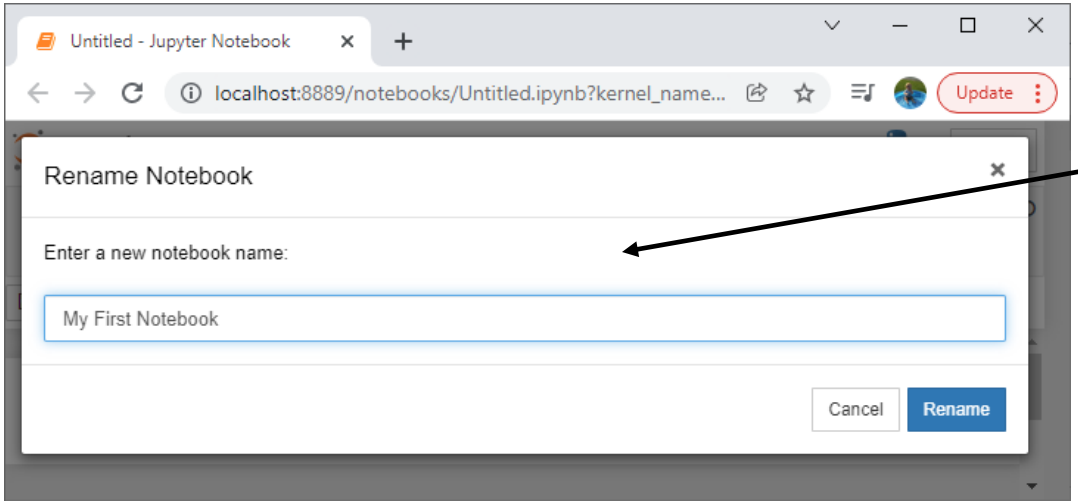


New button

Naming a Notebook



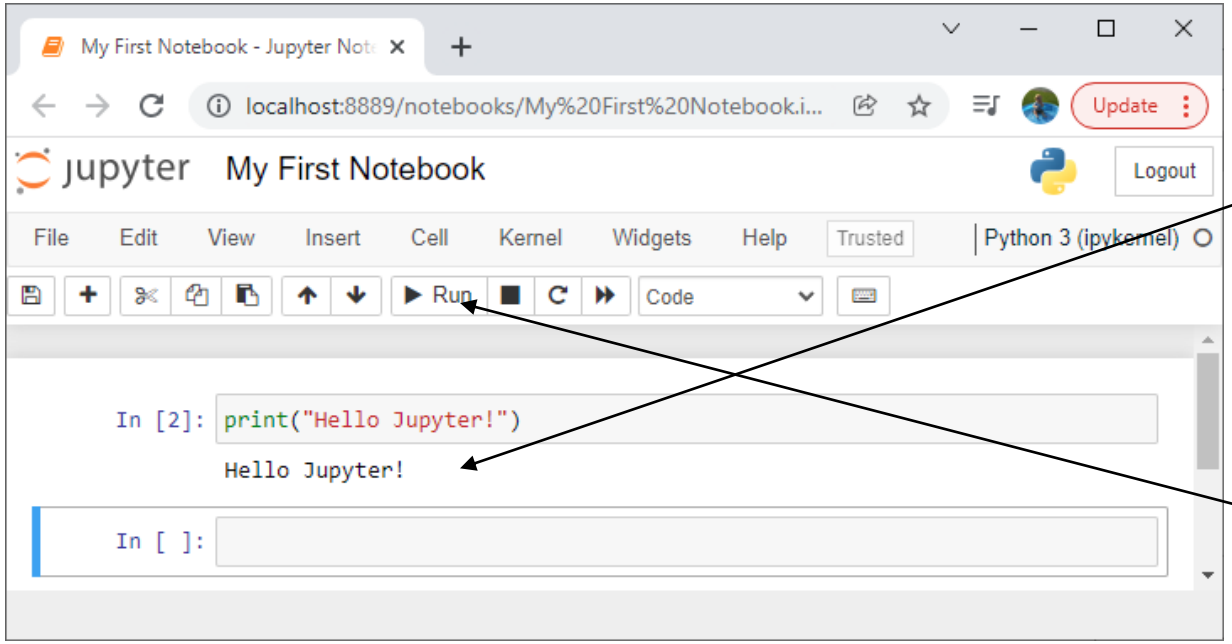
The top of the page is the word *Untitled*. Move your mouse over the word *Untitled* and click on it to rename the notebook to *My First Notebook*.



Note that this will also change the name of the file.

Running Cells

- A notebook's cell defaults to using code whenever you first create one, and that cell uses the *kernel* that you chose when you started the notebook (in our case Python 3).
- Add the following code to the cell: `print("Hello Jupyter!")`
- To execute a cell, you can select the cell and press the *Run* button.



Code in Jupyter Notebooks

- If you have multiple cells in your notebook, and you run the cells in order, you can share your variables and imports across cells.

- This makes it easy to separate your code into logical chunks without the need to reimport libraries or recreate variables or functions in every cell.

- When you run a cell, you will notice that there are some square braces next to the word *In* to the left of the cell. The square braces will auto fill with a number that indicates the order that you ran the cells.

The Menus

- The Jupyter Notebook has several menus that you can use to interact with your Notebook:
  - *File*
  - *Edit*
  - *View*
  - *Insert*
  - *Cell*
  - *Kernel*
  - *Widgets*
  - *Help*

The Menus

- The first menu is the *File* menu. Here you can create a new Notebook or open a preexisting one. This is also where you would go to rename a Notebook.
  - The *Save and Checkpoint* option allows you to create checkpoints that you can roll back to if you need to.

- The *Edit* menu can be used cut, copy, and paste cells. This is also where you would go if you wanted to delete, split, or merge a cell. You can reorder cells here too.

- The *View* menu is useful for toggling the visibility of the header and toolbar. You can also toggle Line Numbers within cells on or off.

The Menus

- The *Insert* menu is just for inserting cells above or below the currently selected cell.

- The *Cell* menu allows you to run one cell, a group of cells, or all the cells. You can also go here to change a cell's type, although the toolbar is more intuitive for that.
  - The other handy feature in this menu is the ability to clear a cell's output. If you are planning to share your Notebook with others, you will probably want to clear the output first so that the next person can run the cells themselves.

Adding Rich Content

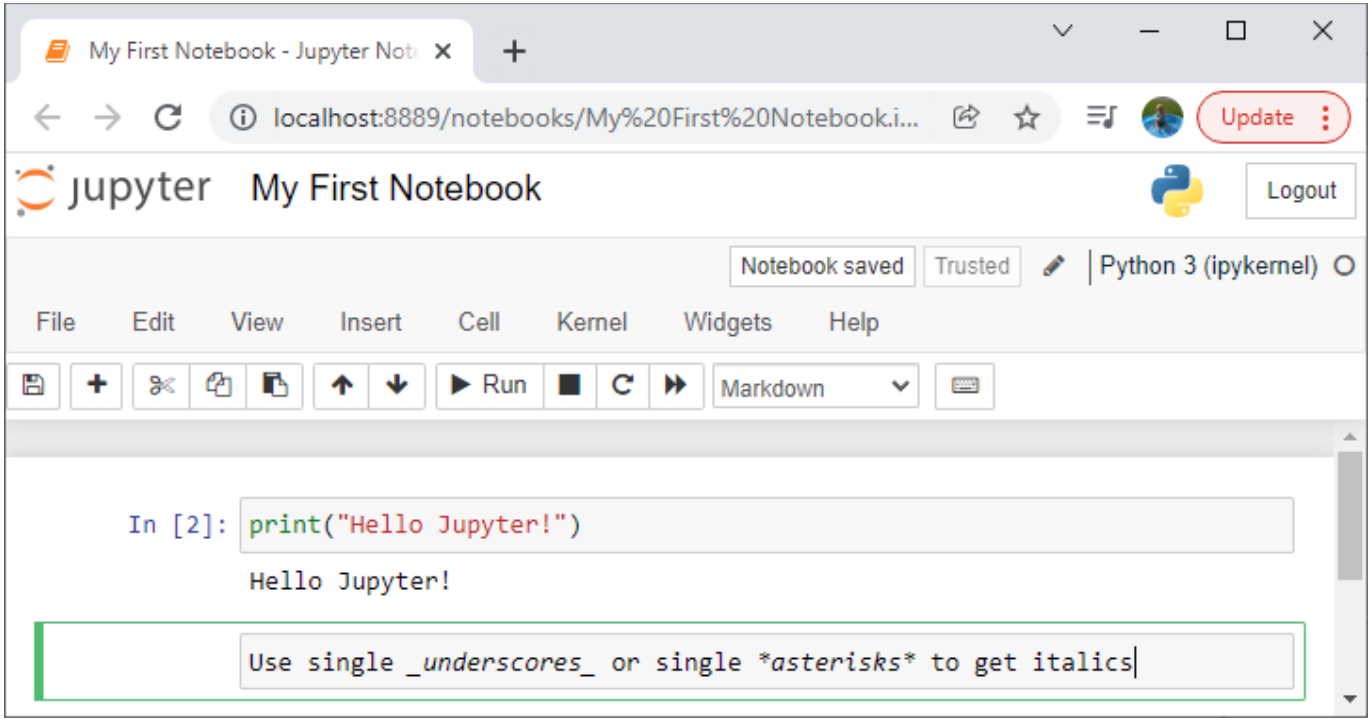
- Jupyter Notebook supports adding rich content to its cells.
- There are technically four cell types: `Code`, `Markdown`, `Raw NBConvert`, and `Heading`.
- The `Heading` cell type is no longer supported. Instead, you are supposed to use `Markdown` for your Headings.
- The `Raw NBConvert` cell type is only intended for special use cases when using the `nbconvert` command line tool. Basically, it allows you to control the formatting in a very specific way when converting from a Notebook to another format.
- The primary cell types that you will use are the `Code` and `Markdown` cell types.

Styling Your Text

- Jupyter Notebook supports Markdown, which is a markup language that is a superset of HTML.

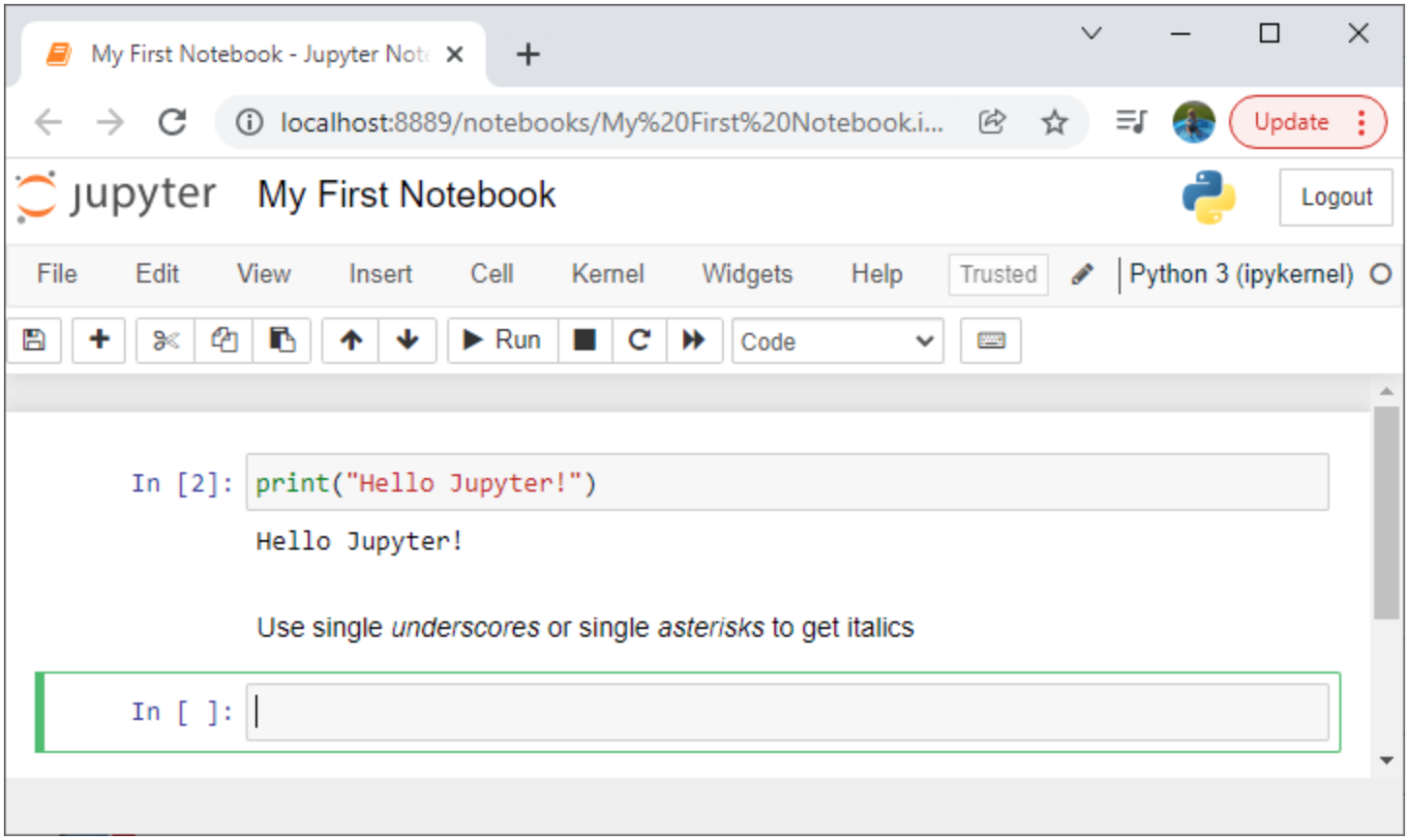
- Set a new cell to `Markdown` and add the following text to the cell:

```
Use single _underscores_ or single *asterisks* to get italics
```



Styling Your Text

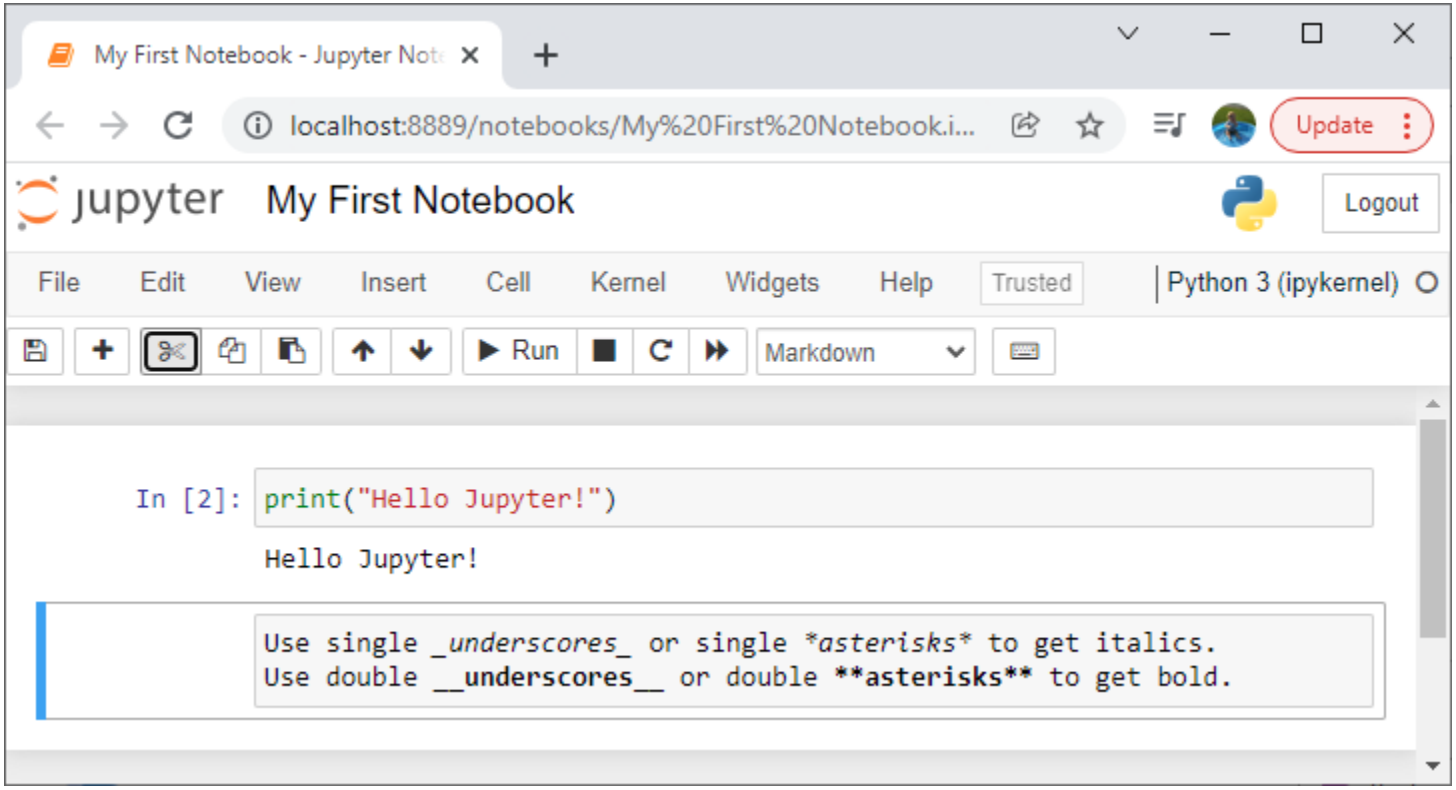
- When you run the cell, the output should look like this:



Styling Your Text

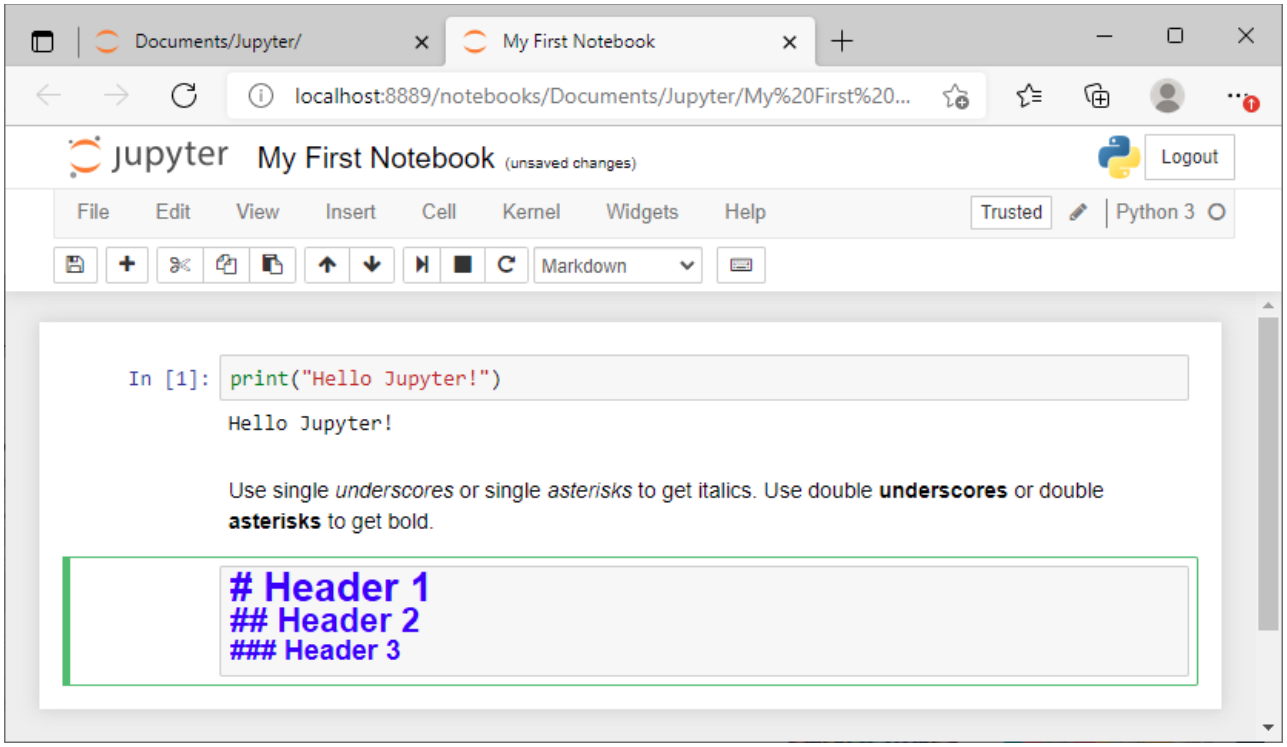
- To edit a `Markdown` cell after it has been run, just double click the cell.

- To make the text bold, use a double underscore or double asterisk.



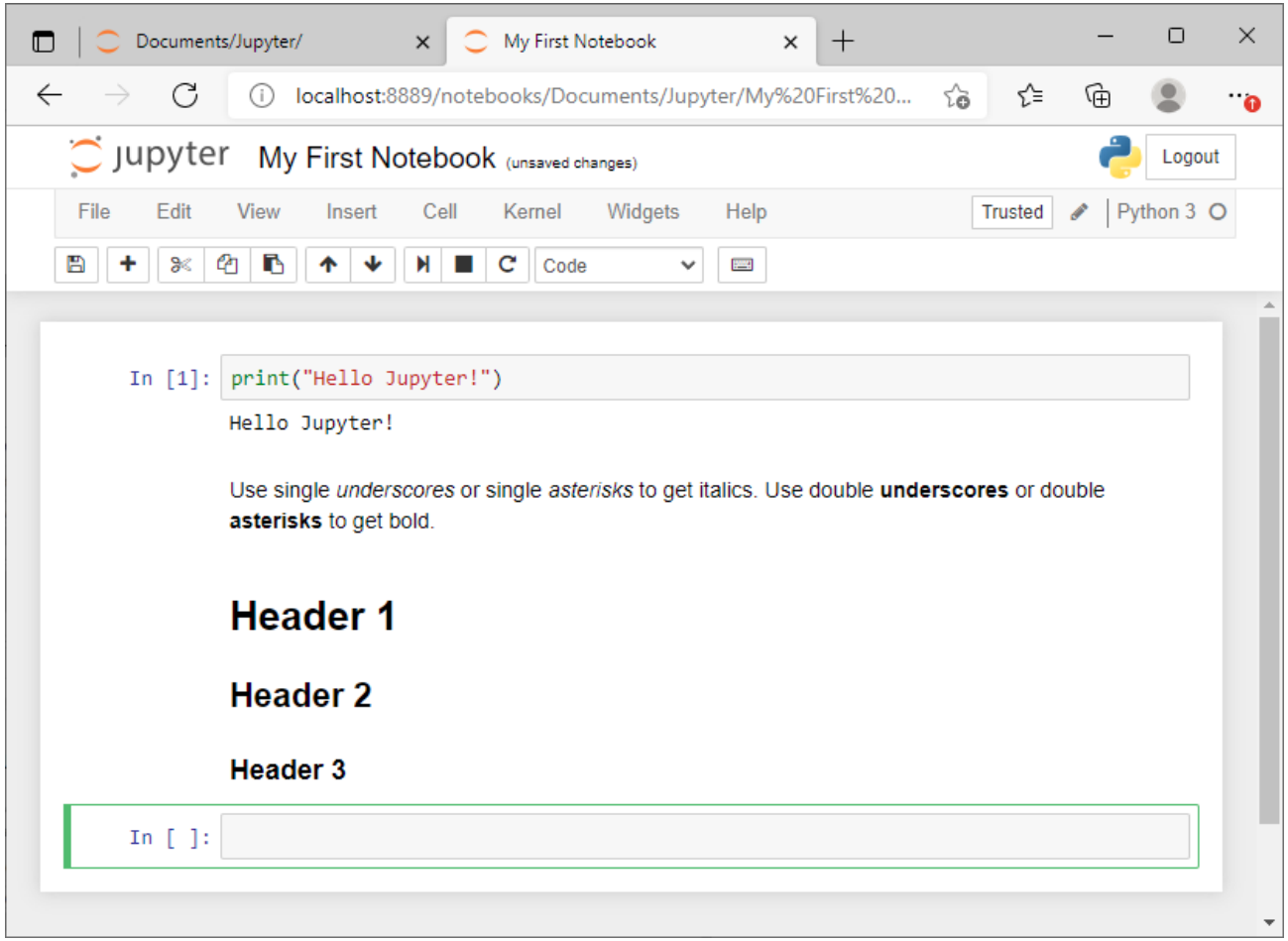
Headers

- Headers are created in `Markdown` using the pound sign.
- The more pound signs you use, the smaller the header. Jupyter Notebook preview it for you:



Headers

- When you run the cell, it will look like this:



Additional Options

- You can switch to *monospace* font with a back single quotation mark (`): `` `string` ``
  - This is useful if you want to include commands or Python output in the Markdown cell.

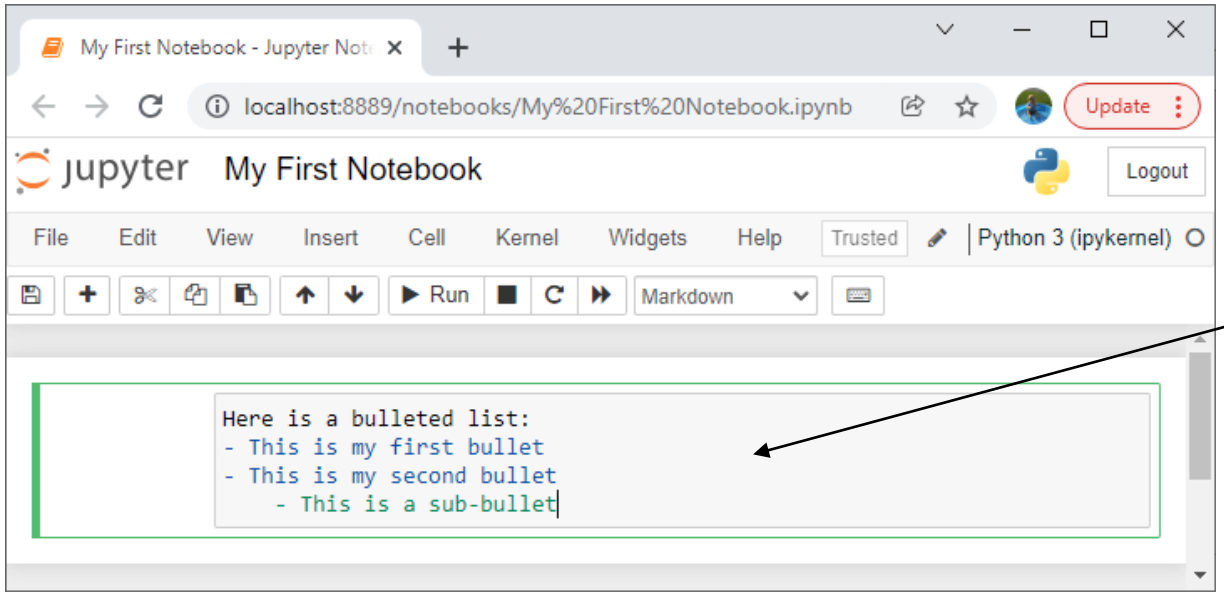
- Sometimes markdown doesn't make line breaks when you want them. To force a line break, use the following: `<br>`

- Use the greater than sign (>) followed by a space to achieve an indent until the next carriage return.

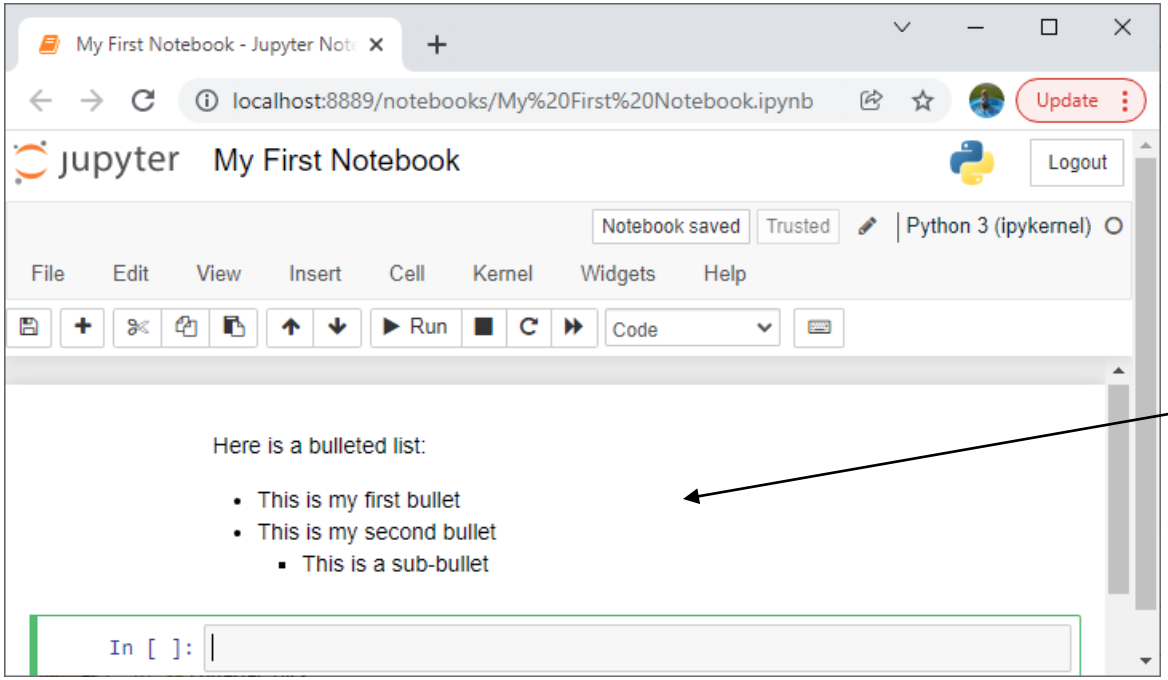
Bulleted List

- To create bullet points, use one of the following methods. Note that each bullet point must be on its own line.
  - An asterisks (*) followed by one or two spaces
  - A hyphen (-) followed by one or two spaces

- To create a sub-bullet, press Tab before entering the bullet point before using one of the methods described above.



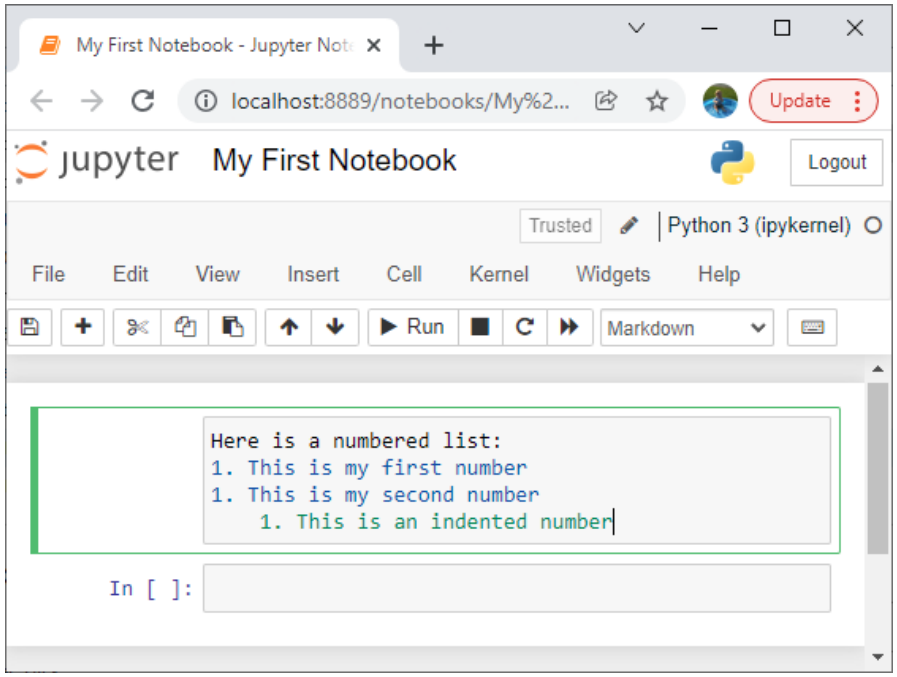
Before running the Markdown cell

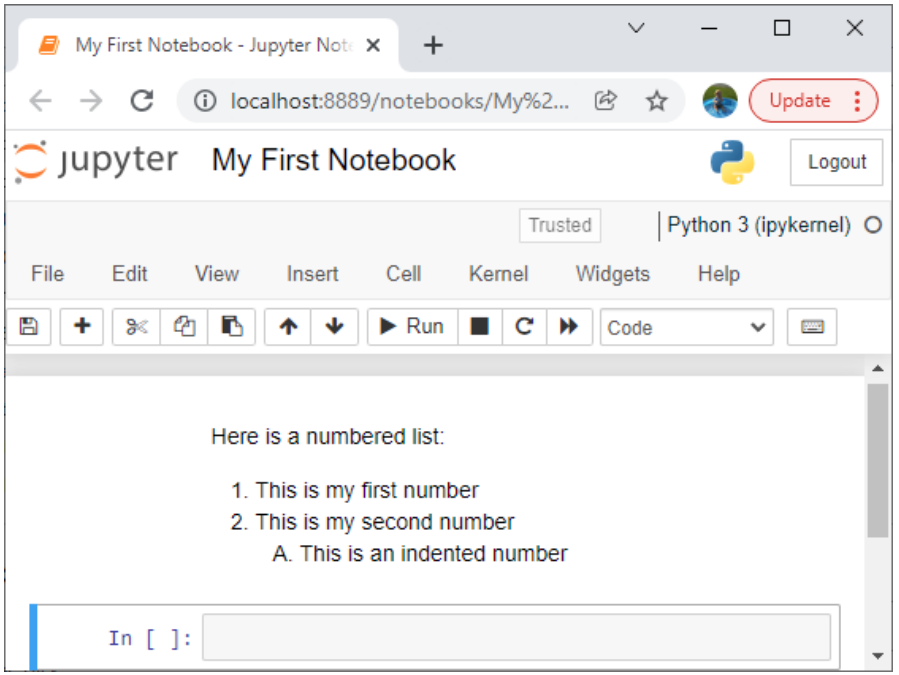


After running the Markdown cell

Numbered Lists

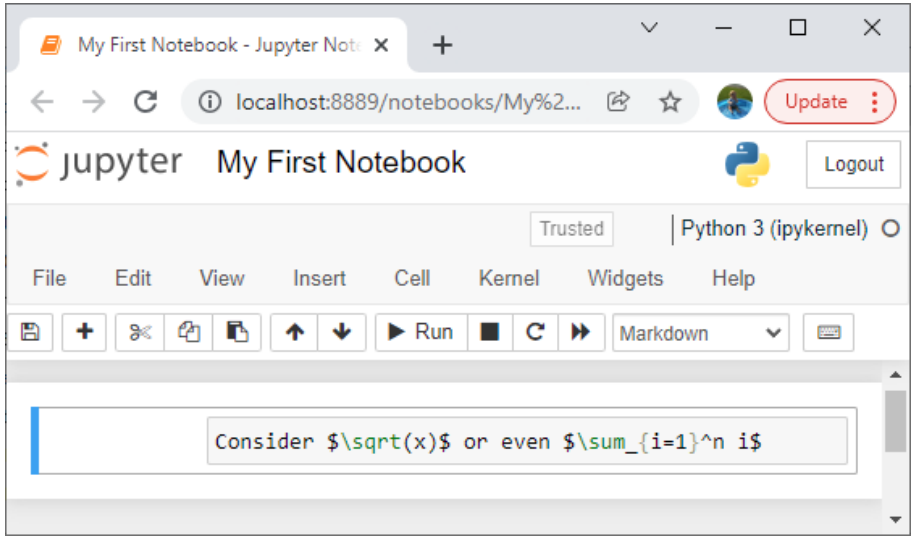
- To create a numbered list, enter `1.` followed by a space. For simplicity, you use `1.` before each entry and the list will be numbered correctly when you run the cell.

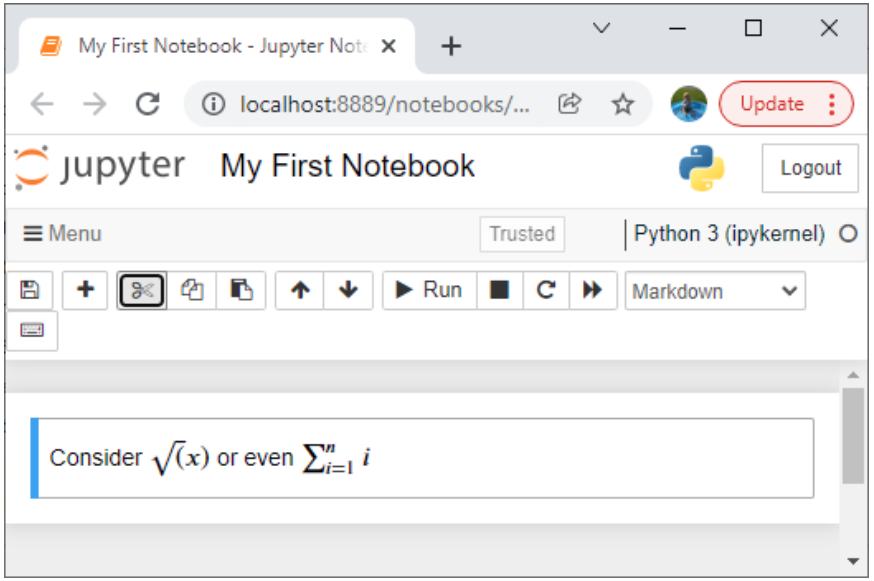




Mathematical Symbols

- You can create mathematical expressions using LaTeX syntax by using ($) symbol:





Additional Features

- Other available features include:
  - Inserting graphics
  - Using HTML code
  - Inserting web links
  - Inserting tables
  - …and more

Git & Github

- Git is an open source **version control** language, developed in 2005.
- GitHub is a cloud-based hosting service that lets you manage Git repositories
  - Extremely widely used in the industry
  - Similar to Dropbox, OneDrive, but is more sophisticated for collaborative work: version control

Git & Github

- The files are stored in a “repository”, or “repo” that are simply folders containing the files for a project.
- Suppose you want to contribute to the TheGreatestApp repo.
  - You first “fork” the repo, which is a way to “clone” the original repo files, which creates an identical repo under your account’s name.

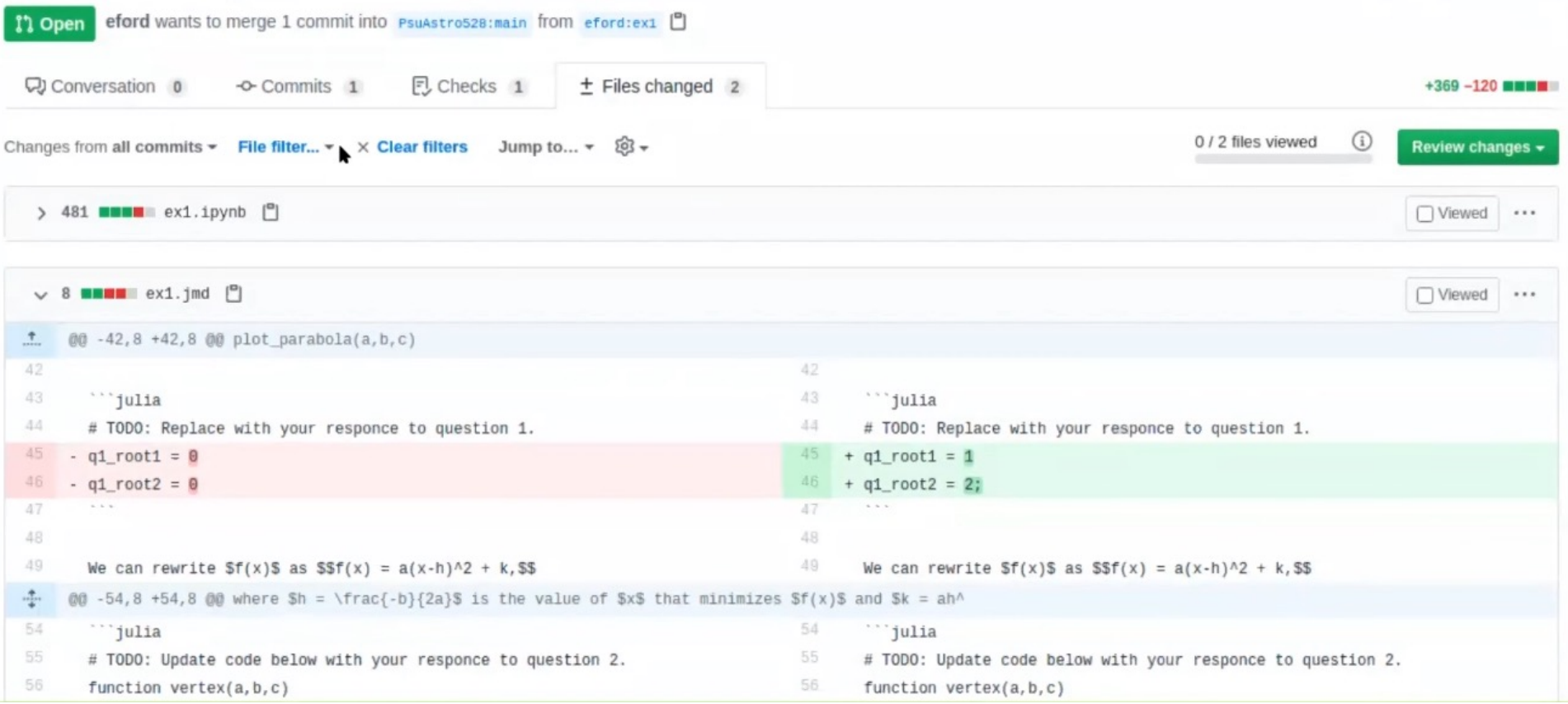
Git & Github

- You work on your great idea to improve the app locally, and edit the files in your clone repo.
- Once you are done, you first “commit” your changes.
  - “Commit”ing basically is a manual way of overwriting files.
- Then you push your changes to your clone repo such that the files are modified and uploaded.

Git & Github

  - Note that nothing has happened to the original repo so far.
- You finally send a “pull request” to the owner of the original repo.
  - A pull request automatically shows the account owner what changes you have made in a nice intuitive side-by-side window.
  - You can add a message at this stage and communicate.

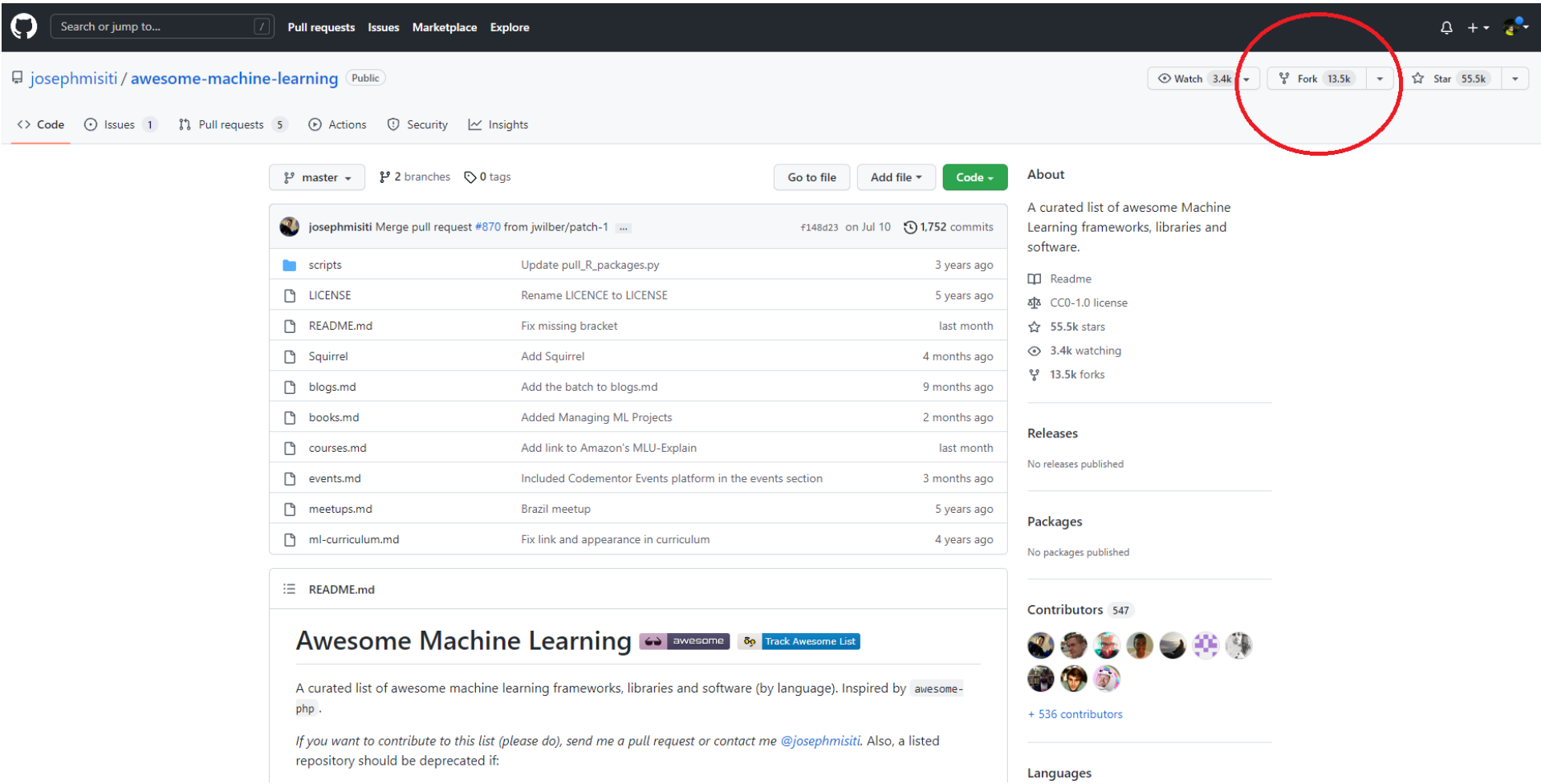
Git & Github


Open
eford wants to merge 1 commit into PsuAstro528:main from eford:ex1
Conversation 0
Commits 1
Checks 1
Files changed 2
+369 −120
Changes from all commits
File filter...
Clear filters
Jump to...
0 / 2 files viewed
Review changes
481 ex1.ipynb
Viewed
8 ex1.jmd
Viewed
@@ -42,8 +42,8 @@ plot_parabola(a,b,c)
```julia
# TODO: Replace with your responce to question 1.
- q1_root1 = 0
- q1_root2 = 0
+ q1_root1 = 1
+ q1_root2 = 2;
We can rewrite $f(x)$ as $$f(x) = a(x-h)^2 + k,$$
@@ -54,8 +54,8 @@ where $h = \frac{-b}{2a}$ is the value of $x$ that minimizes $f(x)$ and $k = ah^
```julia
# TODO: Update code below with your responce to question 2.
function vertex(a,b,c)

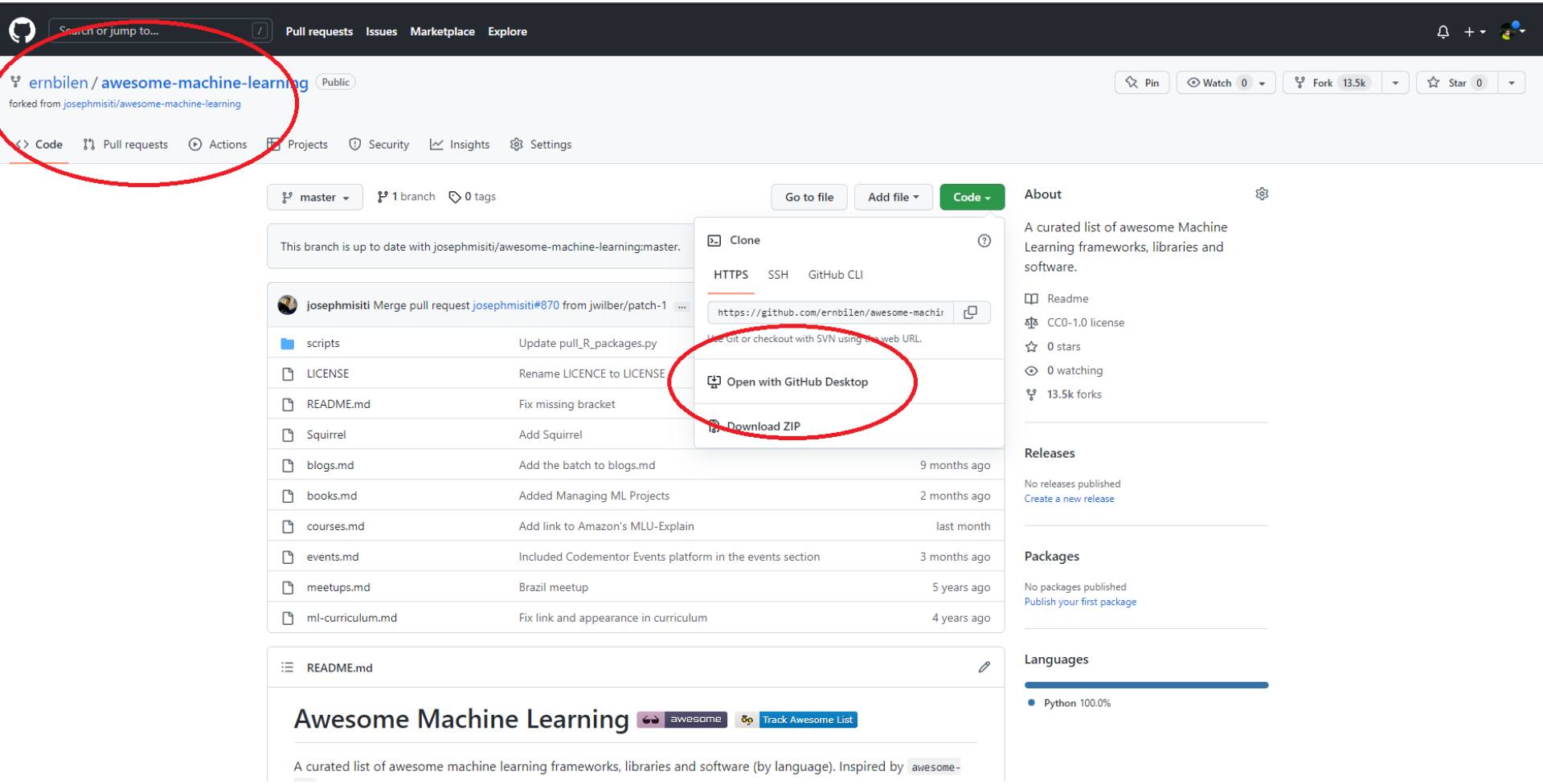
# Git & Github

- After setting your own local repo via “fork”
  - commit
  - push
  - pull request

# Git & Github

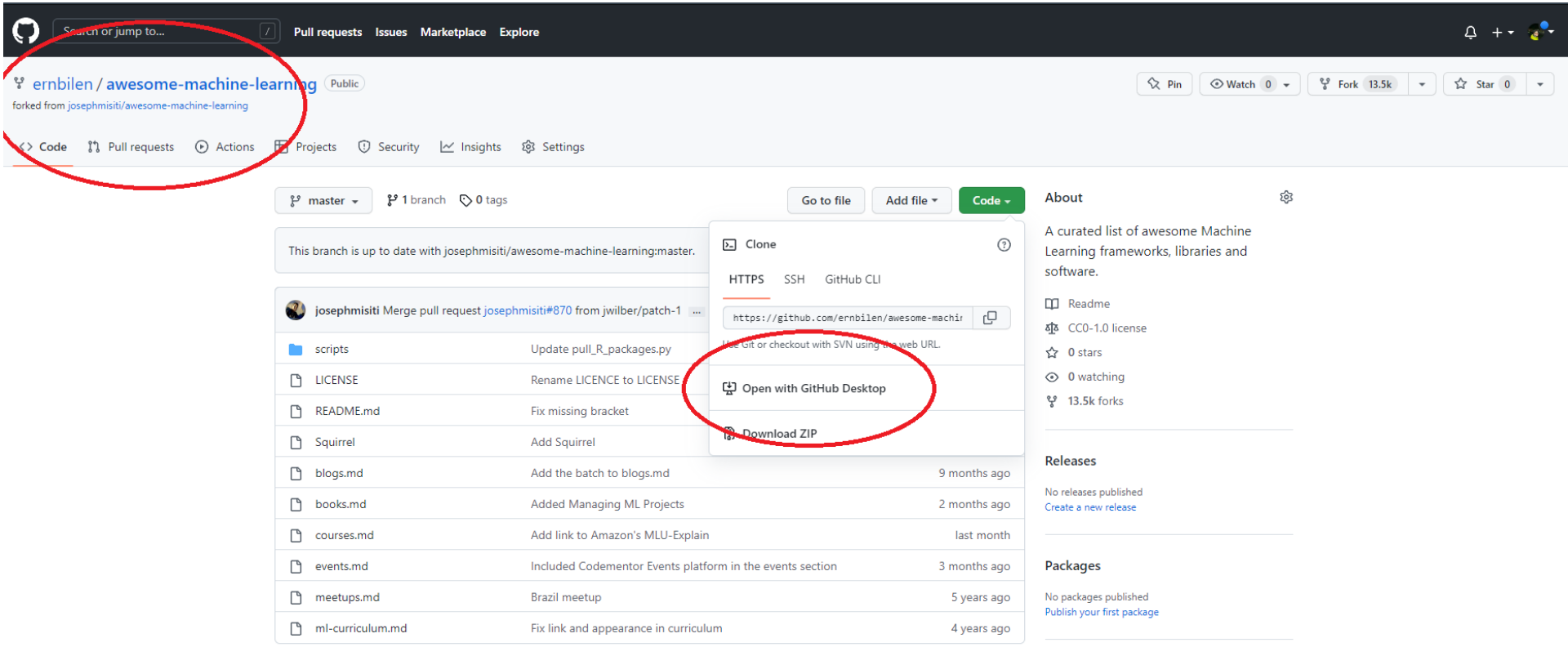

Search or jump to...
Pull requests
Issues
Marketplace
Explore
josephmisiti / awesome-machine-learning
Public
Watch 3.4k
Fork 13.5k
Star 55.5k
Code
Issues 1
Pull requests 5
Actions
Security
Insights
master
2 branches
0 tags
Go to file
Add file
Code
josephmisiti Merge pull request #870 from jwilber/patch-1 ...
f148d23 on Jul 10
1,752 commits
scripts Update pull_R_packages.py 3 years ago
LICENSE Rename LICENCE to LICENSE 5 years ago
README.md Fix missing bracket last month
Squirrel Add Squirrel 4 months ago
blogs.md Add the batch to blogs.md 9 months ago
books.md Added Managing ML Projects 2 months ago
courses.md Add link to Amazon's MLU-Explain last month
events.md Included Codementor Events platform in the events section 3 months ago
meetups.md Brazil meetup 5 years ago
ml-curriculum.md Fix link and appearance in curriculum 4 years ago
README.md
Awesome Machine Learning
awesome
Track Awesome List
A curated list of awesome machine learning frameworks, libraries and software (by language). Inspired by awesome-php .
If you want to contribute to this list (please do), send me a pull request or contact me @josephmisiti. Also, a listed repository should be deprecated if:
About
A curated list of awesome Machine Learning frameworks, libraries and software.
Readme
CC0-1.0 license
55.5k stars
3.4k watching
13.5k forks
Releases
No releases published
Packages
No packages published
Contributors 547
+ 536 contributors
Languages

# Git & Github


Search or jump to...
Pull requests
Issues
Marketplace
Explore
ernbilen / awesome-machine-learning
Public
forked from josephmisiti/awesome-machine-learning
Pin
Watch 0
Fork 13.5k
Star 0
Code
Pull requests
Actions
Projects
Security
Insights
Settings
master
1 branch
0 tags
Go to file
Add file
Code
This branch is up to date with josephmisiti/awesome-machine-learning:master.
Clone
HTTPS
SSH
GitHub CLI
https://github.com/ernbilen/awesome-machi
Use Git or checkout with SVN using the web URL.
Open with GitHub Desktop
Download ZIP
josephmisiti Merge pull request josephmisiti#870 from jwilber/patch-1
scripts Update pull_R_packages.py
LICENSE Rename LICENCE to LICENSE
README.md Fix missing bracket
Squirrel Add Squirrel
blogs.md Add the batch to blogs.md 9 months ago
books.md Added Managing ML Projects 2 months ago
courses.md Add link to Amazon's MLU-Explain last month
events.md Included Codementor Events platform in the events section 3 months ago
meetups.md Brazil meetup 5 years ago
ml-curriculum.md Fix link and appearance in curriculum 4 years ago
README.md
Awesome Machine Learning
awesome
Track Awesome List
A curated list of awesome machine learning frameworks, libraries and software (by language). Inspired by awesome-
About
A curated list of awesome Machine Learning frameworks, libraries and software.
Readme
CC0-1.0 license
0 stars
0 watching
13.5k forks
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
Languages
Python 100.0%

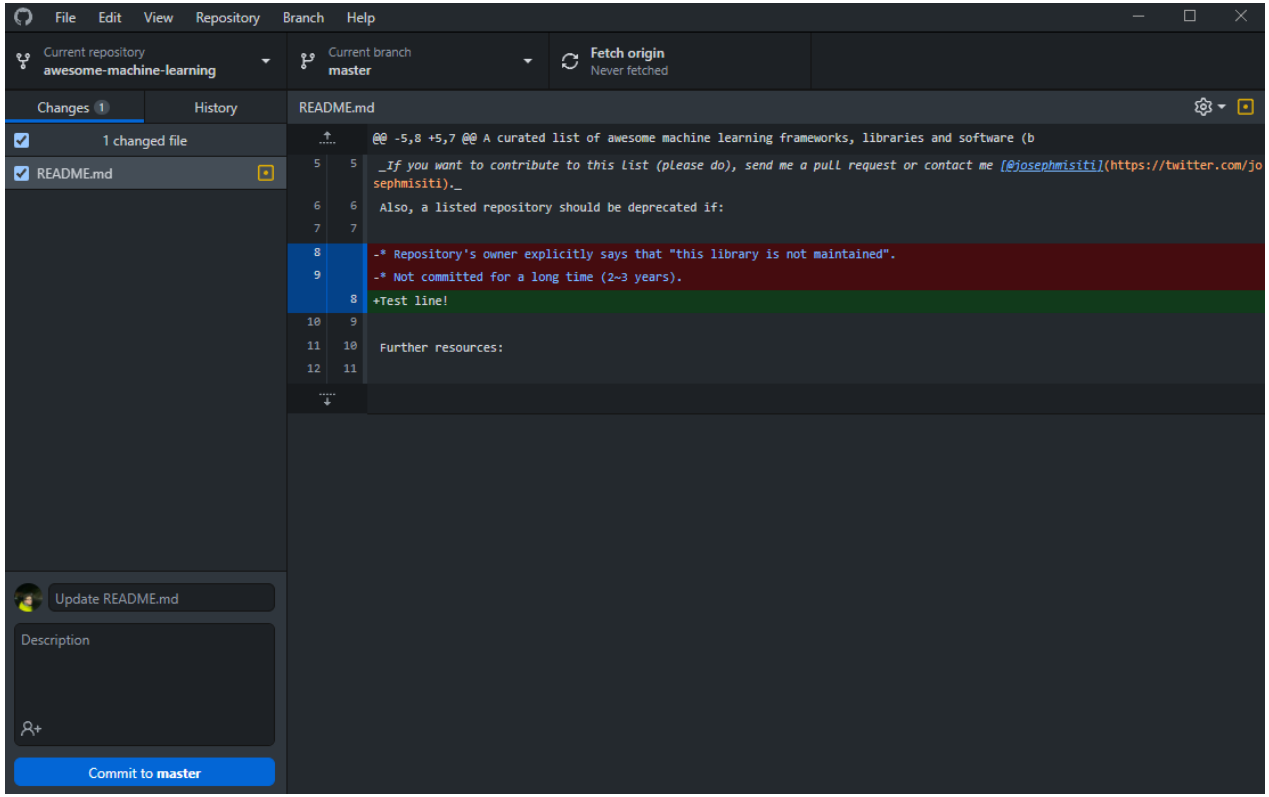
# Git & Github

- You can use browser, GitHub Desktop App, or command line to interact with your repo



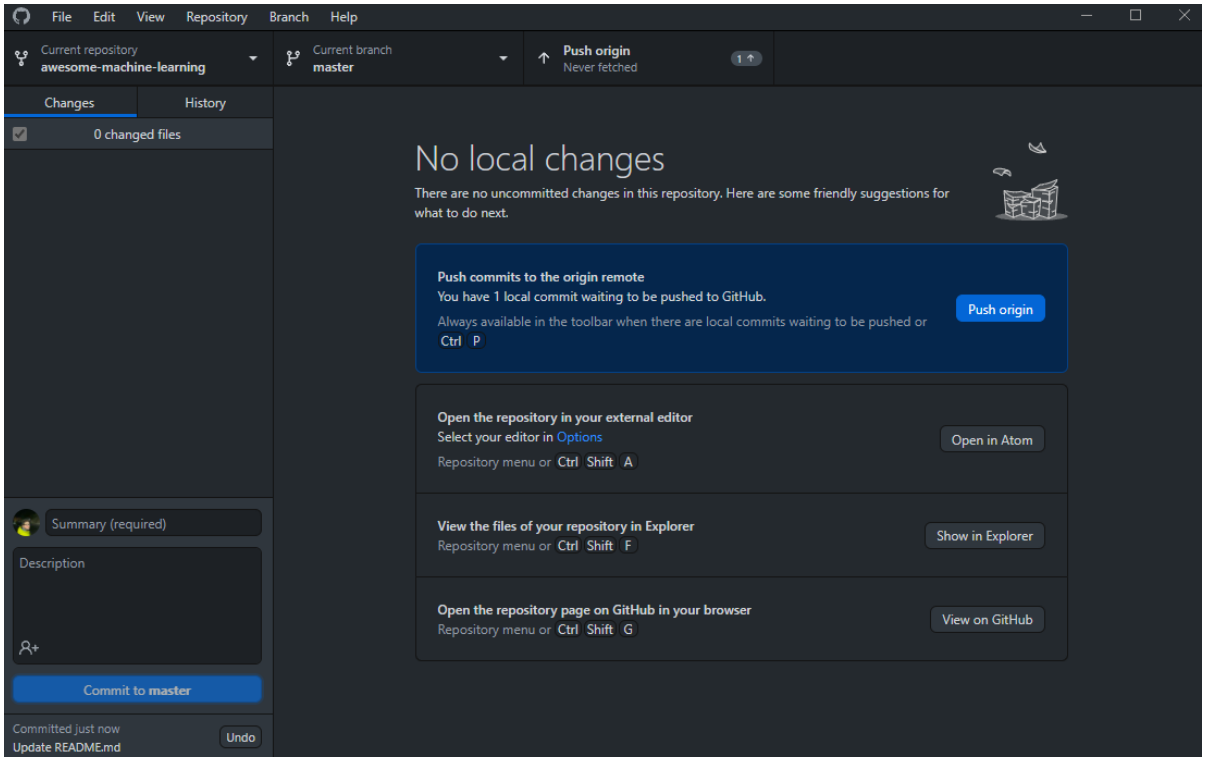
# Git & Github

- Edits will be displayed on the interface, after which you can press “commit to master”, i.e. to your local repository.



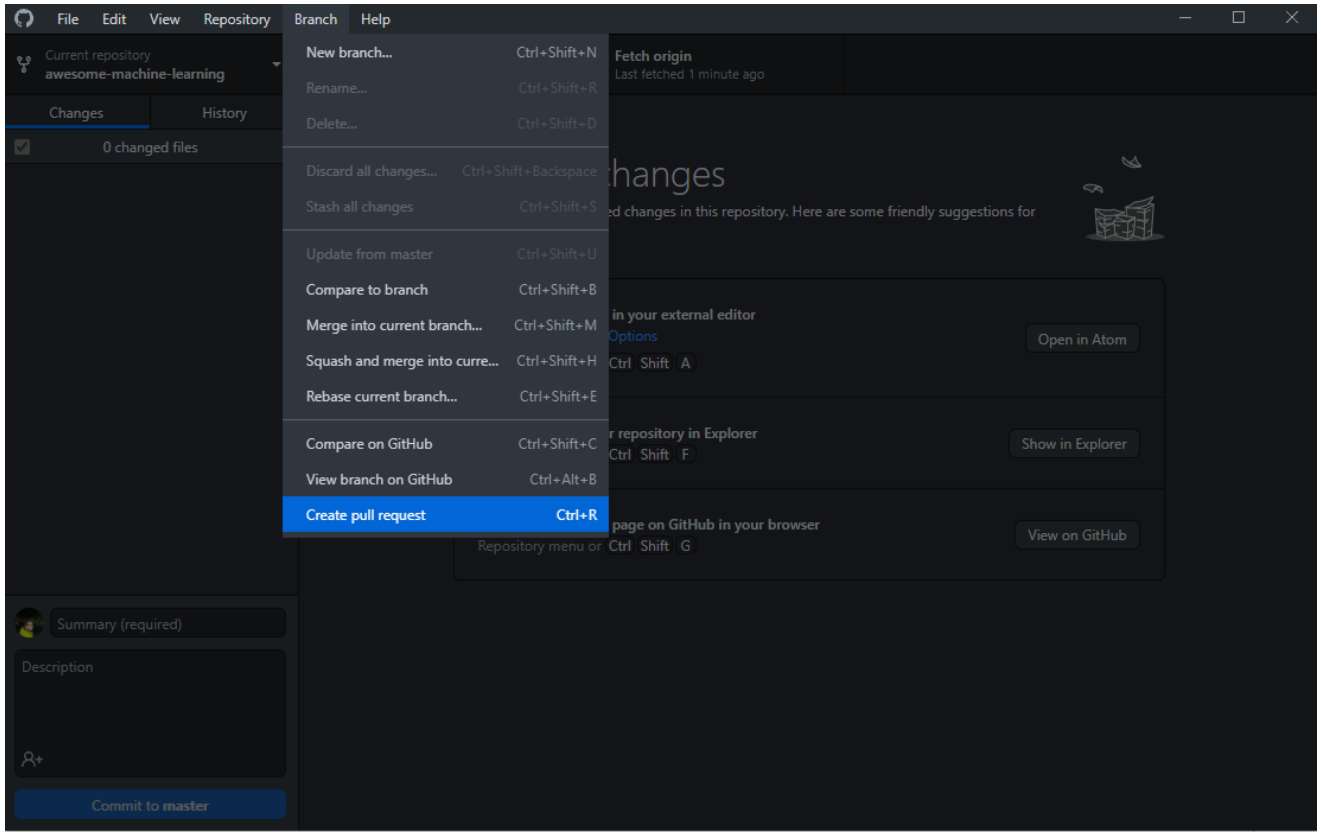
# Git & Github

- You are now ready to “push commits to the origin remote”, this time to your “cloud” repository.



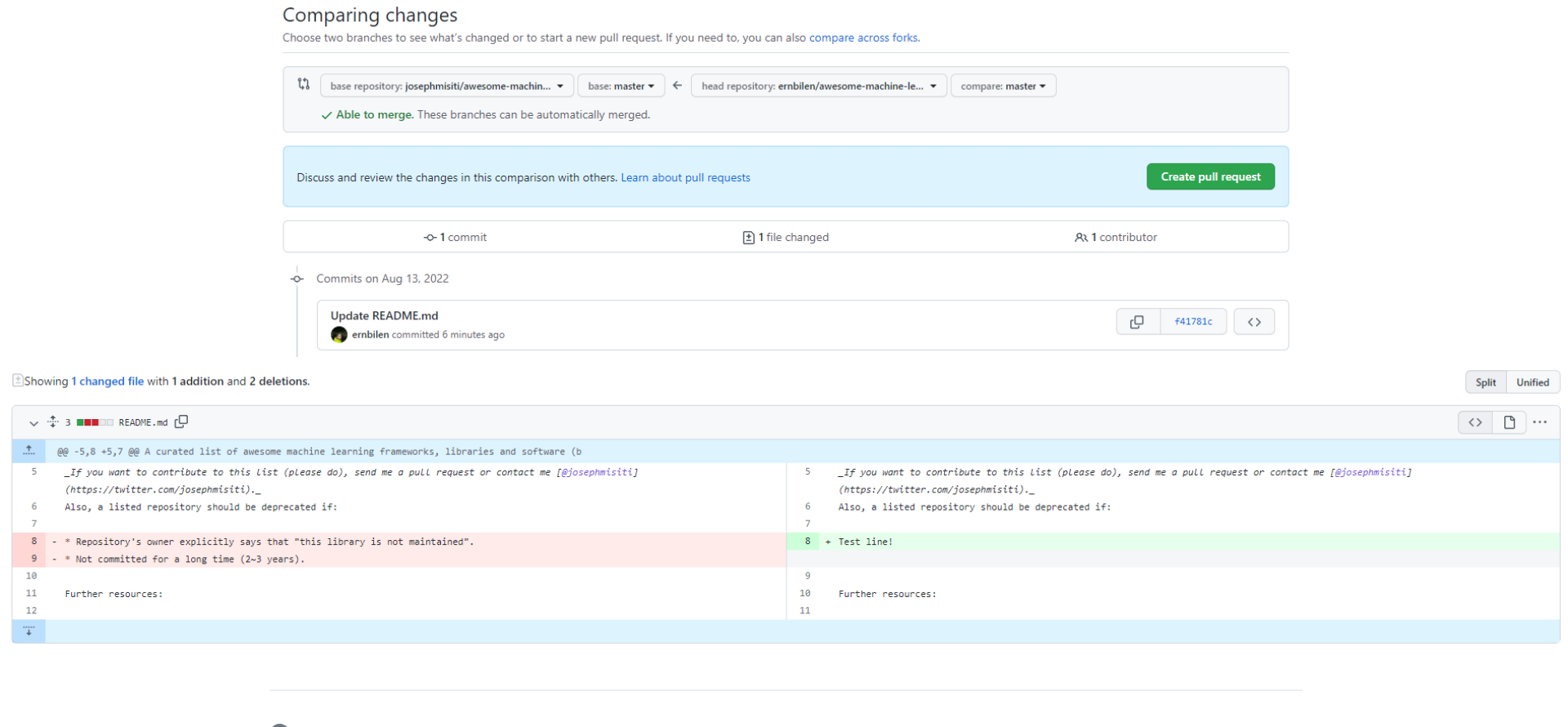
# Git & Github

- To send your changes to the owner of the original repo, click on “create pull request”.



# Git & Github

- You can write a comment and explain your reasoning in updating the code on this screen.



# Git & Github: command line

- After setting your own local repo via “fork” or “clone”
  - add files (only for command line) `>git add --all`
  - commit `>git commit -m “descriptive message”`
    - You can run `>git status` to check what files are changed
  - push `>git push`
  - pull request `>git pull`

For more git commands:
https://about.gitlab.com/images/press/git-cheat-sheet.pdf

# Github Classroom

- We will use Github Classroom to make things more efficient and easier. (free service)
- ~~After setting your own local repo via "fork" or "clone"~~
  - You will receive an invitation email for each assignment.
  - Accepting the invitation automatically forks my repo and creates your own local clone repo.
  - You can work on your assignment in your local repo, do any edits.

# Github Classroom

- Once you have your local repo, you can use the Github Desktop app, or any browser to upload your assignment via
  - commit
  - push
  - ~~pull request:~~ Also automated. Once you push your files to your local repo, everything gets delivered automatically.

# We are done! Any questions?

# We are done! Any questions?