CS4650 Topic 6: Jupyter Notebooks

What is a Jupyter Notebook?

- A *Jupyter Notebook* is a document that combines text, code, visualizations, and equations.
- The code and equations are 'live', in that you can change parameters then recalculate, to immediately see the results.
- This aspect is similar to Excel, in that you can change the values in some cells, which then cause the values in other cells to be recalculated.
- Consider a math textbook written as a Notebook: The reader can read the text, but can also experiment with the formulas to try different values, in addition to seeing the few examples that the writer had given.
- As you develop a data science project, the Notebook can be both your journal and your publishable results.

Installing Jupyter Notebook

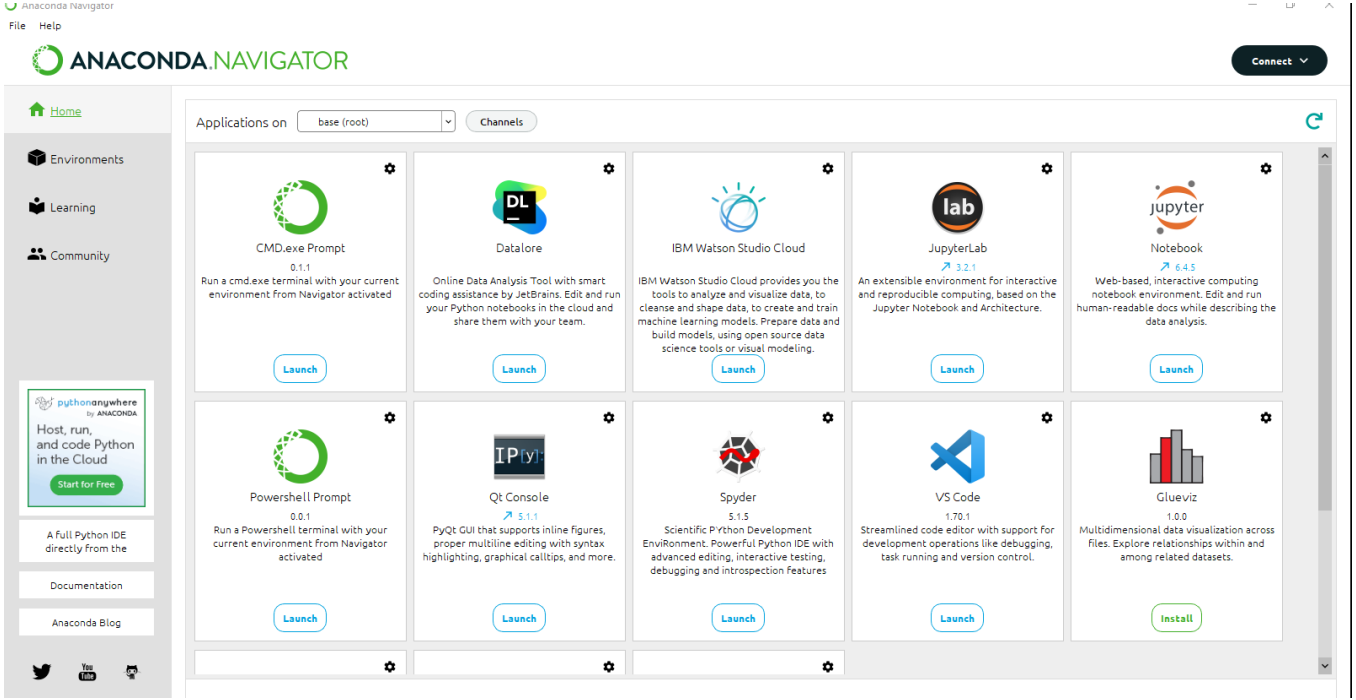
- If you are using Miniconda, using the Terminal, you simply activate the environment, then install the package:

  conda activate Users/dave/desktop/project_1⏎
  conda install jupyter⏎
- To then actually start Jupyter Notebook, enter

  jupyter notebook⏎
- This starts a little web server, then opens a browser which views the Notebook.
- When you are done, to close the web server, go back to the Terminal window and type ^C.

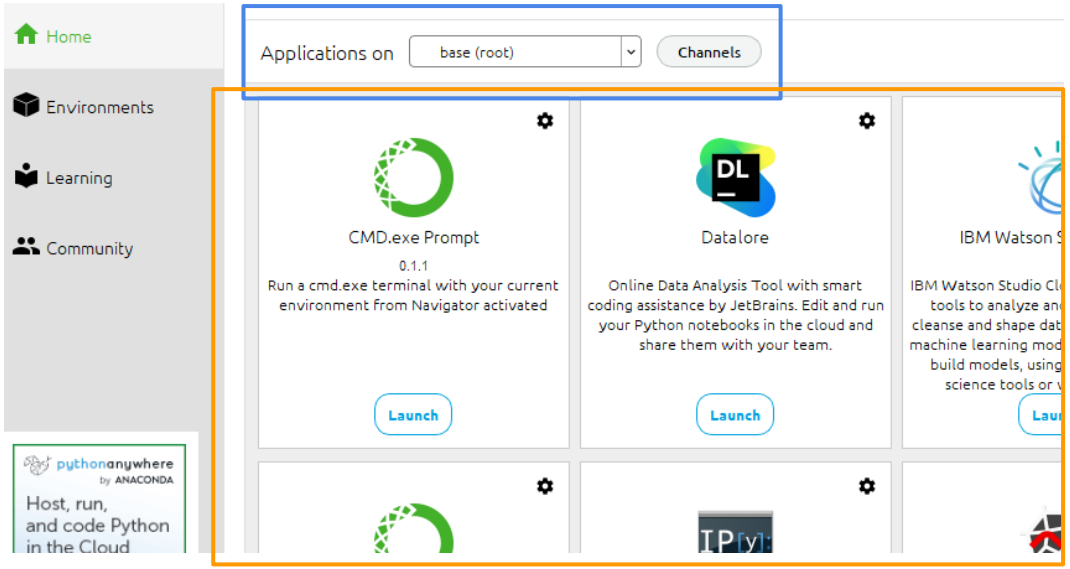
Installing Jupyter Notebook

- If you are using Anaconda, this is the view of your browser:



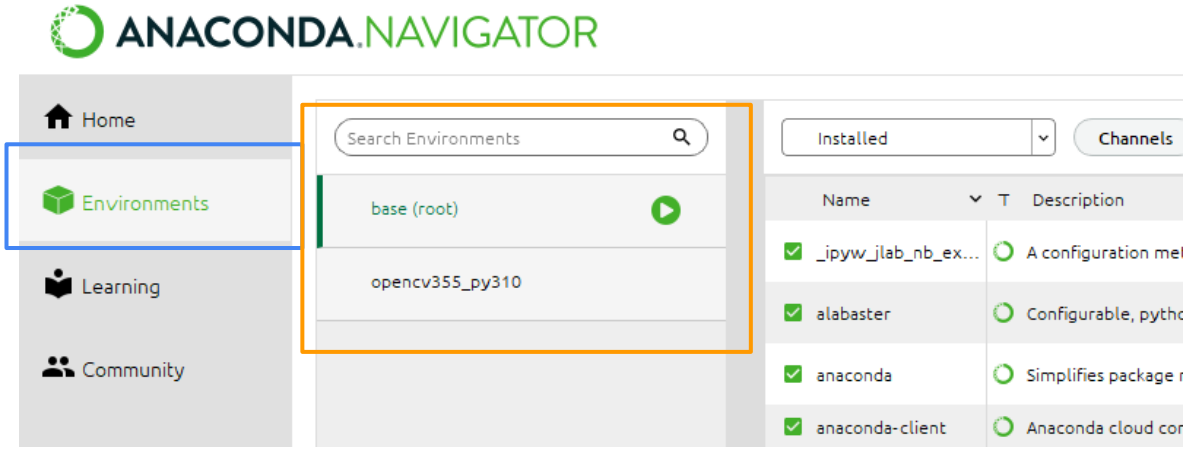
Installing Jupyter Notebook

- Anaconda has environments (just like Miniconda).
- The blue box shows that the (base) environment is activated.
- The orange box, most of the screen, shows the packages that are installed. (Note that Anaconda calls them *applications.*)



Installing Jupyter Notebook

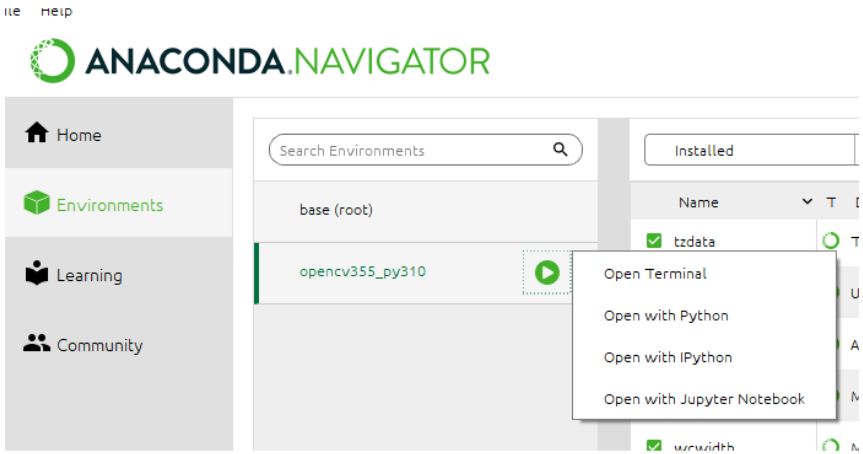
- By clicking on the Environments button (blue box), a list of the environments is shown (orange box).
- New environments can be created.



- Selecting an environment will then activate that new environment.
- The screen will then show the packages that are installed in that environment.

Installing Jupyter Notebook

- Also note, if you click on the circled green arrow of an environment, a pop-up menu appears.
- If you select 'Open Terminal', a Terminal is opened, at this project's directory. This is the same Terminal you would see using Miniconda.



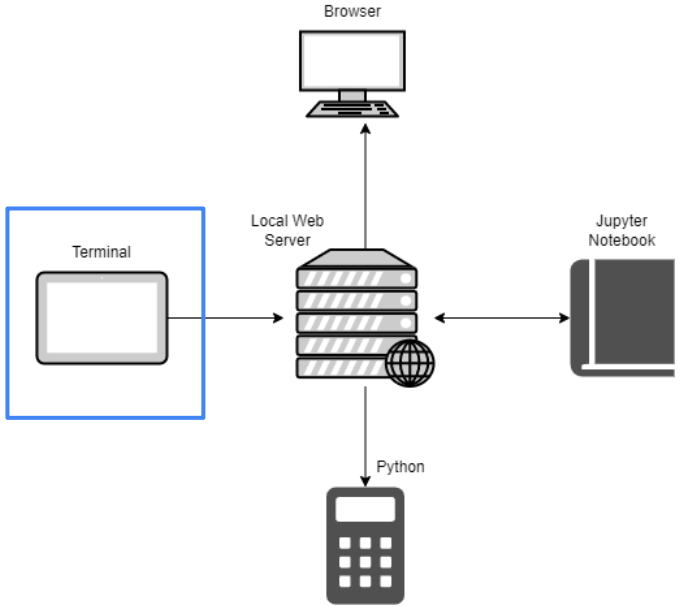
- If you select 'Open with Jupyter Notebook', Jupyter will be started.

Running Jupyter

- Whichever path we took, using Miniconda or Anaconda, we have this view on the screen.
- This is not a Notebook (yet), this is just the Dashboard. In here we can create a new Notebook, or open an existing Notebook.

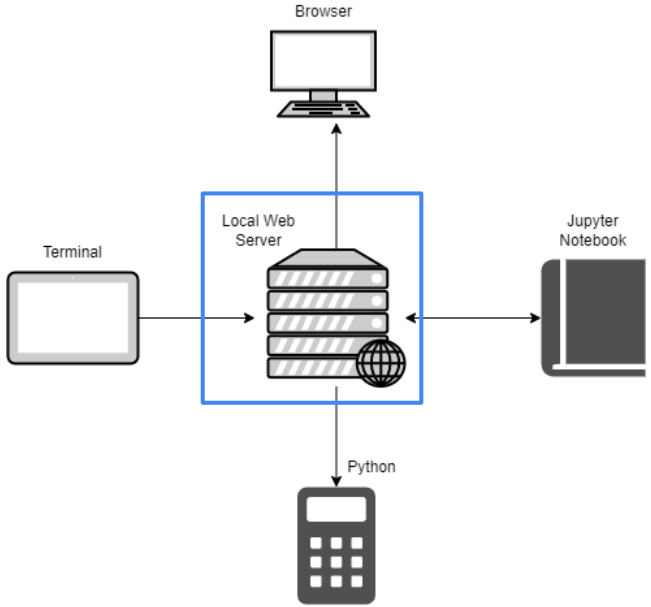
Behind The Scenes

- This is a little behind-the-scenes look at how Jupyter Notebook works.
- Everything starts from the Terminal window, where you ran the 'jupyter notebook' command.
- Alternatively, you could have clicked on the 'Jupyter Notebook' button in Anaconda, but that essentially runs a hidden Terminal.



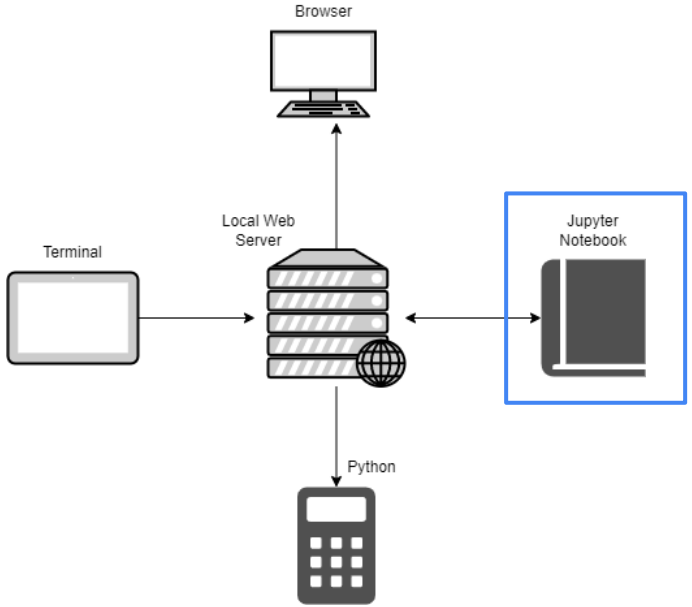
Behind The Scenes

- Running the 'jupyter notebook' command starts a Local Web Server.
- This is a private web server, so browsers can connect to this server to fetch web pages, just like when you use a browser on the web.
- This is private, so no other computers can see this (more or less).



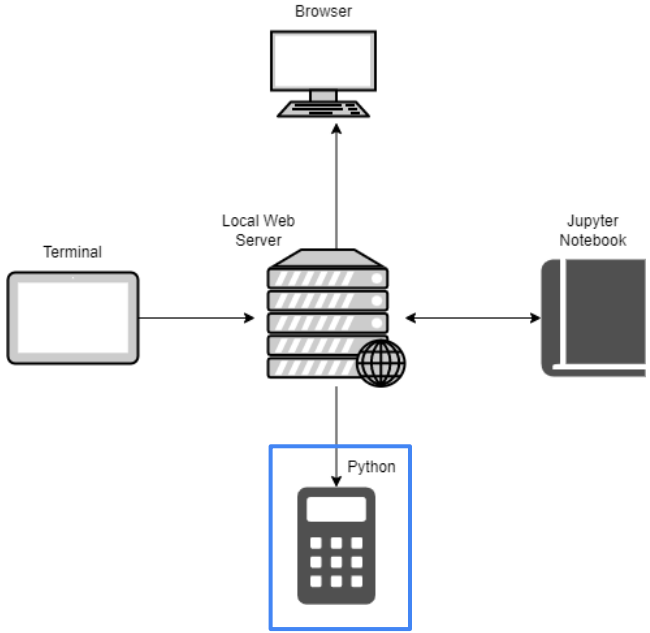
Behind The Scenes

- The local web server attaches to a file on your computer, which is the actual Jupyter Notebook.
- The program can view the contents and edit the contents.
- It first looks to see which *kernel* to run, which indicates which language your notebook uses.
- The most common kernel is Python, although Java, C++, and a hundred more are available.



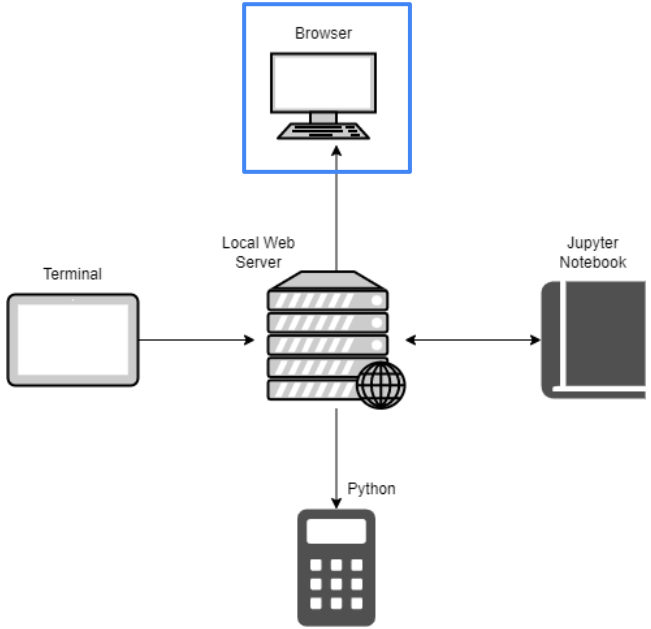
Behind The Scenes

- Based on the kernel specified in the Notebook, the Local Web Server runs an engine that 'does' that language.
- Here, the server started the Python engine.
- When the user asks the 'Notebook' to evaluate a *cell*, the contents of that cell are sent to the engine, and the results of that run are written back to the Notebook.
- We will see more of this in a bit!



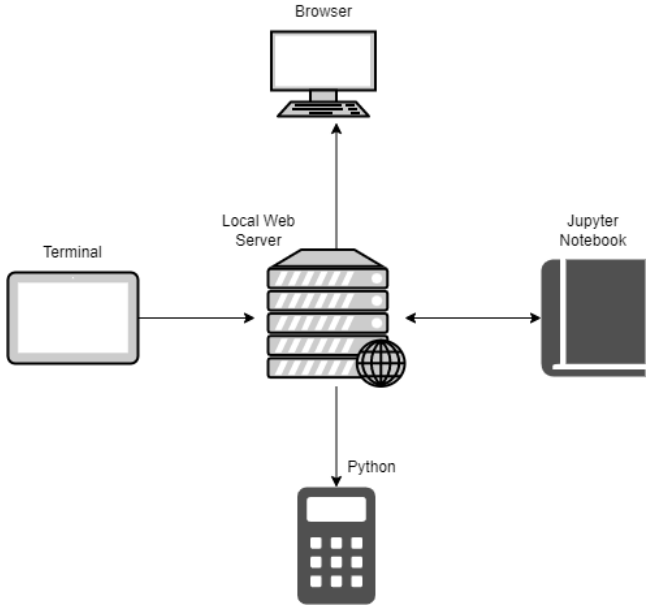
Behind The Scenes

- Finally, the Local Web Server starts a browser, connecting the browser to this server.
- This then shows you the 'Notebook'.



Behind The Scenes

- When we talk about the 'Notebook', we might be referring to one of three things:
  - The actual Notebook file
  - What the user sees and interacts with on the screen of the browser
  - This whole collection of programs and data.
- To shut this down, we need to terminate the Local Web Server.
- This is done by typing ^C in the Terminal window.



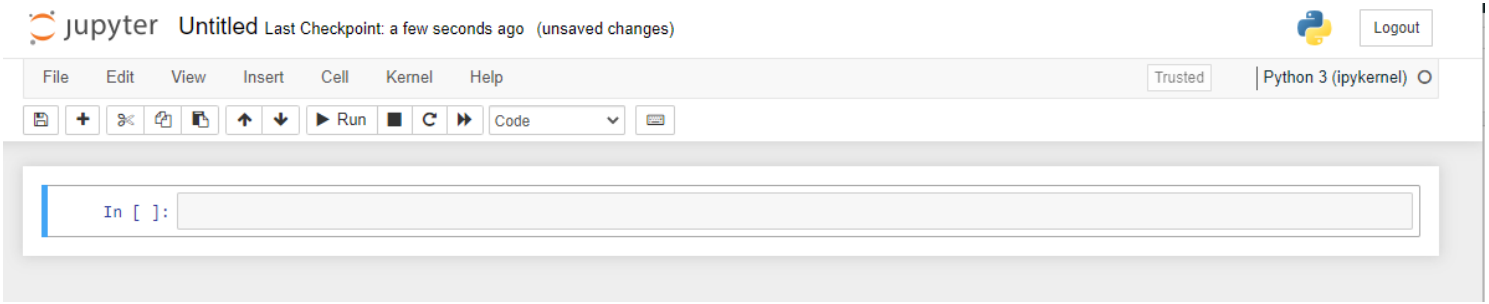
Running Jupyter

- Back to the browser!
- This currently shows a list of files in the startup directory.

Running Jupyter

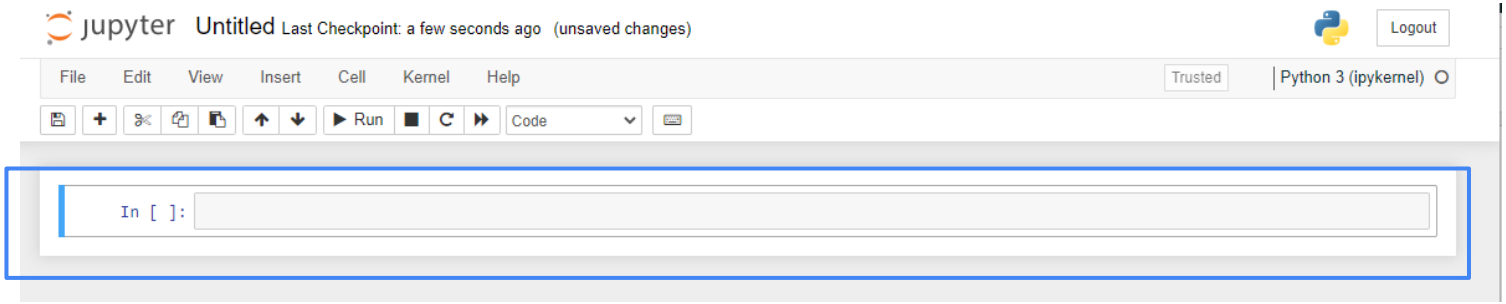
- Let's create a Notebook. Click on the 'New' dropdown, then pick 'Python 3' (to be the *kernel*).

Running Jupyter



- This creates a new Notebook, naming it 'Untitled'.
- Click on the name, then enter a new name for the Notebook. I called mine "Sample 1".
- If you then go back to the dashboard, you will see an entry 'Sample 1.ipynb'.
- Later you can get back to this Notebook by clicking this file.

Running Jupyter



- The key feature of a Notebook is a Cell.
- A Cell is where you can type Python code, or you can enter text, including headlines, paragraphs, itemized lists, and add hyperlinks to images.

Running Jupyter

- Right now, the first Cell is empty, and it is expecting code:

  In [ ]: |_____|

- Enter some Python code into the Cell:

  In [ ]: [print('Hello World!')__|

  then click the '▷Run' button.

- The Notebook will then number that cell, print the result of that run, then open a new cell:

  In [1]: [print('Hello World!')__|

  Hello World!

  In [ ]: |_____|

Running Jupyter

- Edit the contents of the Cell, then run it again:

  In [2]: [print('Goodbye World!')__|

  Goodbye World!

  In [ ]: |_____|
- Notice that the output of the cell will change to reflect the new code.
- Also notice that the label of the line has changed to 'In [2]'. This is the second run of the kernel.
- Before a cell is run, it's label is 'In [ ]'.
- While a cell is running, it's label is 'In [*]'.
- After a cell has run, it's label contains the *execution count*.

Complex Cells

- Cells can contain multiple lines of code:

In [1]:

```
def square(x):
        return x * x

x = 2
y = square(x)
print('%d squared is %d' % (x, y))
```

```
2 squared is 4
```

In [ ] :

```

```

Multiple Cells

- Cells can contain multiple lines of code:

In [1]:
```
def square(x):
        return x * x
x = 2
```

In [2]:
```
y = square(x)
print('%d squared is %d' % (x, y))
```

```
2 squared is 4
```

In [ ] :
```

```

- The functions and variables defined in Cell 1 are still available when Cell 2 is run.
- What matters is the order of when the Cells are run.
- Usually Cells run top to bottom.

Markdown

- Webpages are designed using HTML, a *markup language*: Mixed in with the text of the webpage are commands (markup) that indicate how the text is arranged and displayed.
- For example, some markup may indicate that some text is actually a headline, some other text is a paragraph, and some other text are items in a bulleted list.
- Jupyter uses *Markdown*, which is a lightweight language for formatting plain text.
- A Cell in a Notebook may be tagged to contain Markdown rather than Java code.

Markdown

- Plain text is entered as is, just plain text.
- A line that starts with a single '#' is a level 1 heading:

  # This is a heading

  becomes

  This is a heading
- A line that starts with '##' is a level 2 heading:

  ## And this becomes

  becomes

  And this becomes

Markdown

- There are two ways to make things **bold** or *italic*:

  Here **is** one *way* to do it

  And __here__ is _another_ way

  becomes

  Here **is** one *way* to do it

  And **here** is *another* way
- As you enter these in a Markdown Cell, the text changes appearance a bit. But when you '▷Run' the Cell, then the changes happen 'for real'.

Markdown

- You can make itemized lists:

  * This makes an unordered list
  * These use bullets
  1. You can also make ordered lists
  2. With numbers

  becomes

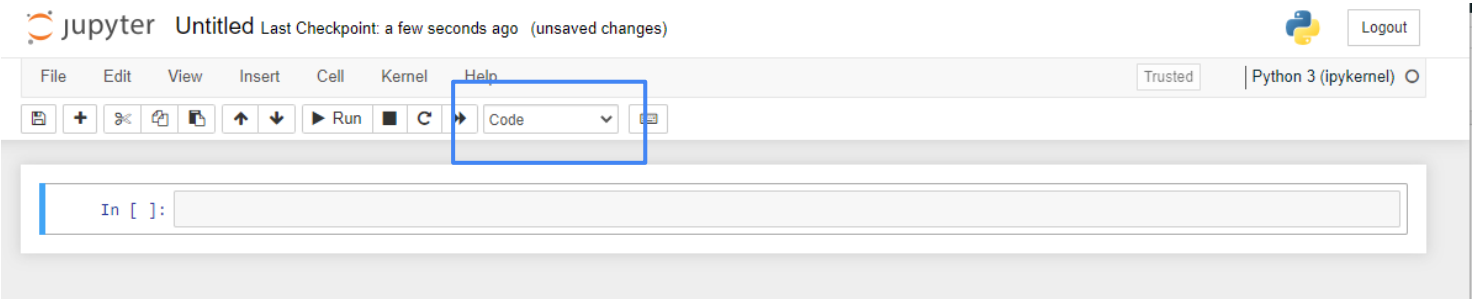
  • This makes an unordered list
  • These use bullets
  1. You can also make ordered lists
  2. With numbers

Markdown

- You can include an image in your Notebook.
- If the image is on-line, you can use it's URL:

  ![Alt text](https://www.example.com/image.jpg)
- The [Alt Text] is displayed if the Notebook cannot find the image or if the user turns off viewing images.
- You can use a local URL of the image, if it is on your computer, and if you share the Notebook, you will also include the image (at the same relative address).
- You can also insert an image, but this turns the image into encoded text. The image gets reconstructed, but your Notebook gets much larger!

Turning on Markdown



- So how do you indicate that a Cell is either Code or Markdown?
- The blue box shows a drop-down where you can make this selection.

Wrapup

- This gives you an introduction to Jupyter Notebooks.
- We will be using them a lot in this class.
- Some of the lectures will be based on Notebooks.
- Some of your homework will be using Notebooks.
- Be sure to install either Anaconda or Miniconda on your computer, then experiment with Notebooks.
- The lectures will give you more information about Notebooks as we proceed.