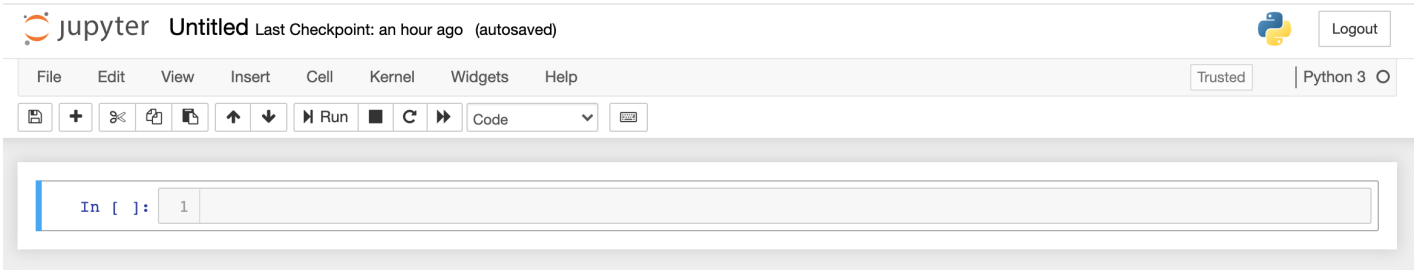
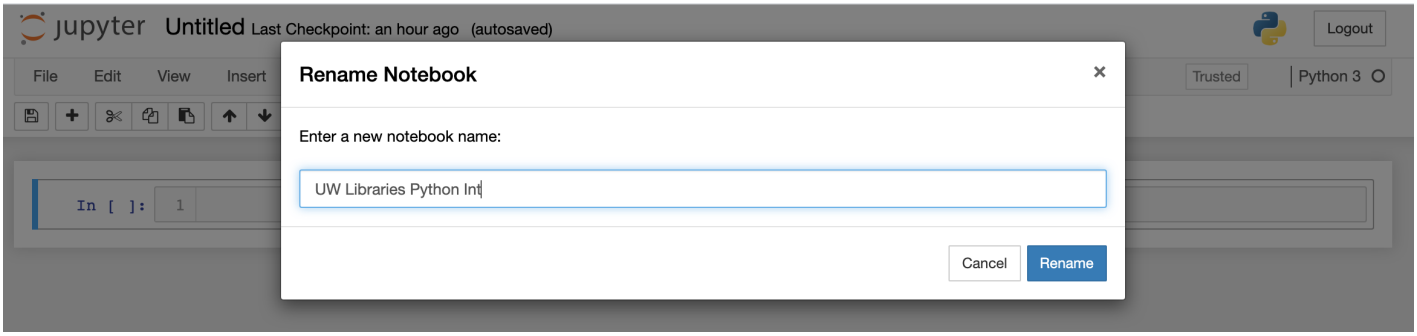
Introduction to Jupyter Notebooks

When you first open a new Jupyter notebook, your screen should look like this:



The blank box at the bottom, enclosed by the green box, is called a "cell" and this is where your notebook content goes.

The first thing I like to do, when starting a new Jupyter notebook, is **name** it. To do this, click on "Untitled" at the top of the page and rename the notebook in the window that pops up:



YOUR TURN: Rename your notebooks (even just making a small change, such as adding a ":" or "-" between the words *Group* and *Intro*)

There are two main types of cells in Jupyter notebooks -- code cells and markdown cells. As shown in the figure above, the default type of cell is a **code** cell -- where you'll put your Python code.

This is a **markdown** cell.

You can change the default type of the cell using the dropdown menu to change the type from "Code" to "Markdown."

If you double click inside this cell (the one you are reading right now), you will see the formatting responsible for the appearance of the content in this cell.

Markdown cells are used to add text, images, and links to Jupyter notebooks. Markdown is a superset of HTML, so markdown cells should recognize HTML commands. They also recognize LaTeX commands, which is very useful if you need to include equations in your notebook.

For more information on features of markdown cells, look at [this guide from Read the Docs (https://jupyter-notebook.readthedocs.io/en/latest/examples/Notebook/Working%20With%20Markdown%20Cells.html)](). Actually, Read the Docs have great information on many features of Jupyter notebooks -- I recommend starting [here (https://jupyter-notebook.readthedocs.io/en/stable/notebook.html)]().

Code cells can be used as if you are working at a Python prompt, as shown below:

```
In [11]: 3+4
Out[11]: 7

In [12]: print("Hello World!")
         Hello World!
```

Conveniently, unlike when working at a Python prompt, you can enter multiple lines of code easily in a code cell:

```
In [13]: x = 2
         y = 4
         z = 6

         product = x * y * z
         triplesum = x + y + z

         print("The product of", x, ",", y, ", and", z,"is", product)
         print("The sum of", x, ",", y, ", and", z,"is", triplesum)
         The product of 2 , 4 , and 6 is 48
         The sum of 2 , 4 , and 6 is 12
```

Hmm... I don't really like that formatting. Notice how spaces automatically got added between each item. Let's try string concatenation, instead:

In [14]:
```
print("The product of" + x + "," + y + ", and" + z +"is" + product)
print("The sum of" + x + "," + y + ", and" + z +"is" + triplesum)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-14-a9a4ff499ace> in <module>
----> 1 print("The product of" + x + "," + y + ", and" + z +"is" + product)
      2 print("The sum of" + x + "," + y + ", and" + z +"is" + triplesum)

TypeError: can only concatenate str (not "int") to str
```

Hmm... Well, **that** didn't work! This gives us the chance to look at an error message. Apparently, when using string concatenation, all the elements being concatenated actually have to **be** strings (when using commas to separate elements, we didn't have to worry about this. OK -- trying again:

In [15]:
```
print("The product of" + str(x) + "," + str(y) + ", and" + str(z) +"is" + str(prod
uct))
print("The sum of" + str(x) + "," + str(y) + ", and" + str(z) +"is" + str(triplesu
m))
```

```
The product of2,4, and6is48
The sum of2,4, and6is12
```

So... no error message, but still not what I wanted. Another difference between using "," and "+" is that no spaces are automatically added using the latter -- so you need to add the spaces yourself:

In [16]:
```
print("The product of " + str(x) + ", " + str(y) + ", and " + str(z) +" is " + str
(product))
print("The sum of " + str(x) + ", " + str(y) + ", and " + str(z) +" is " + str(tri
plesum))
```

```
The product of 2, 4, and 6 is 48
The sum of 2, 4, and 6 is 12
```

Much better!

YOUR TURN: Try out some basic Python commands in the code cell below:

In [ ]:
```

```

In the cell above, the output appears when we "run" the cell -- either by clicking the Run button in the menu at the top of the notebook or by using a keyboard shortcut such as "Shift + Return" (moves you to the next cell) or "control + Return" (keeps you in the same cell). When you specifically ask for content to appear in the console, as in the print statement below, you don't see the "Out[#]:" indicator before your output.

When you run a cell, the computer will remember any variables or methods defined in the cell.

The numbers inside the brackets keep track of the order in which you run the **code** cells. If you run the same cell multiple times, its number will increase each time. This can be important because as you work on a notebook and change things, your later code may work because the computer "remembers" commands that you may have changed or deleted. This is why, when you think you're finished with a project, you should "Restart" the Kernel (clearing the memory), to make sure your current notebook actually works the way you think it does.

Some Notes on Formatting

Jupyter notebook markdown cells are great -- you can put just about anything you want in a markdown cell.

An especially nice feature of markdown cells is how easy it is to customize the appearance of your content in the cell. If you already know HTML, it should work in a markdown cell. If you aren't familiar with HTML, Markdown, itself, seems even easier (my opinion) to use. I like this basic introduction to Jupyter notebook Markdown syntax (https://sqlbak.com/blog/jupyter-notebook-markdown-cheatsheet).

Some Basics:

- Easily make headers of different sizes using increasing numbers of #'s (more #'s --> smaller header)
- Make bulleted list using hyphens before each line in the list
- Add internet links by enclosing the link text in square brackets, followed immediately by the url in round brackets
- Make text bold by surrounding it with two underscores or two asterixes before and after
- Make text italic by surrounding it with one underscore or one asterix before and after

It is also possible to change the font color in the cell, although there is not a quick and easy markdown command to do this (that I know of). When I want to change the font color, I use a more traditional HTML approach.

You can also embed videos in Jupyter notebooks:

Perhaps non-intuitively, these are embedded using **code** cells:

In [17]:
```
from IPython.display import YouTubeVideo
YouTubeVideo('DKiI6NfSIe8')
```

Out[17]:

We won't watch the whole video -- it's almost an hour long! But you can watch it afterwards, if you want to.

Inserting Images into Markdown Cells

Jupyter notebooks make adding images very easy:

- Go to the Edit tab at the top of the page.
- Scroll all the way to the end of the dropdown menu.
- Select Insert Image.
- Navigate to the image file you want to insert.

Couldn't be easier!

But what is you want to **resize** your image?

That's possible, but it requires a little more work:

- Insert the image following the guidelines above.
- Modify the markdown code that appears when you insert an image, as shown in the example below.

First -- here's a picture of the best cats in the whole world. It's appropriately **HUGE**!

OK -- maybe *obnoxiously* huge would be more accurate. Let's see how to fix this:

I'm going to insert the image again, and then make some modifications:

(attachment:mysweeties.jpeg)

If you look at the difference in the syntax, you see that after adding the image, I deleted the "!" at the front of the line and, also, the filename inside the square brackets. Then I added some HTML (which I always have to look up using Google search). The width command in the HTML statement changes the size.

(attachment:resize_image.png)

You can also change the position of an image:

(attachment:staffa3.jpg)

For more details on manipulating images, look here (https://stackoverflow.com/questions/57930004/jupyter-notebook-position-embedded-image-in-markdown).

YOUR TURN: Try adding and resizing images from your own computers. You can use this cell (images can only be added in markdown cells).

What exactly IS a Jupyter notebook?

For this section, I'd like you to download your notebooks to your local machines.

(attachment:download.png)

Now, go to your downloads and find the newly downloaded .ipynb file and open it in a text editor.

What does it look like?

Can you find your image file?

A Jupyter notebook is really just a text file! This makes it easy to share -- just send a .ipynb file to someone and they can run it, reproducing any results you found (provided they have access to the same data). This has made Jupyter notebooks very popular for reproducible research.

Sample Jupyter notebooks are available at the NBViewer site (https://nbviewer.jupyter.org/)

If you want to do data science with Jupyter notebooks, there is even an online platform you can use that provides students with free access to Google cloud servers (including GPUs): Google Colab (https://colab.research.google.com)

In [ ]: