⚠ Before you start ⚠

Duplicate this Jupyter Notebook in your `week-02` folder (right-click -> Duplicate) and then your last name to the beginning of it (ie. `blevins-jupyter-notebook-intro.ipynb` - otherwise you risk having all your work overwritten when you try to sync your GitHub repository with your instructor's repository.

⚠ *No, seriously: check the name of this file. Is it the copy you made? (ie. `blevins-jupyter-notebook-intro.ipynb`). If so, you can proceed* ⚠

Getting Started With Jupyter Notebooks

We're going to be using Jupyter Notebooks to write and run Python code. Jupyter notebooks are made up of cells - a chunk of text or code, kind of like a paragraph. There are a few different types of cells: `Markdown` vs. `Code`.

A Markdown Cell

Markdown is basically a fancy form of text (like you'd type in a Microsoft Word file or Google Doc). In Jupyter Notebooks, you use Markdown cells to explain and document your code cells. You can do things like make a words **bold** or *italics*, add headers, insert hyperlinks, etc.

Let's practice - here's a reference sheet for common Markdown formatting

1. Write your name in the cell and make it bold
2. Write the title of this class in the cell and turn it into a hyperlink to the course website.
3. Run the cell to see if it's rendering correctly

Kun Cheng

A Code Cell

Unlike the above Markdown cells, Jupyter Notebook also has `code` cells. For our purposes, we're going to be primarily filling these cells with Python code. You run/execute a **code** cell using `Shift-Enter` or by pressing the `Run (play)` ▶ button in the toolbar. Make sure your cursor is inside the following code cell by clicking inside it, and then try running the code cell:

In [10]:
```
print("Welcome to HIST 4261/5261!")
```

```
Welcome to HIST 4261/5261!
```

Notice that when you `Run` the above code cell, it uses the `print` function to print out whatever is inside the parentheses. With a lot of programming languages, print doesn't mean "print something on paper"; it means "show me this on the screen." Try writing your own similar code in the next empty cell that, when you run it, **will print a greeting of some kind for your instructor**:

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

```
Hello every one!
```

When you run a `code` cell, it tries to execute the code. But what happens if you make a mistake in your code and spell something wrong? You'll get an error when you try to run it. To see what I mean, I've made a typo in the following code cell. Try running the cell to see what an error looks like:

In [14]:
```
print("Debugging code means dealing with lots of error messages!"
```

```
  Cell In[14], line 1
    print("Debugging code means dealing with lots of error messages!"
                                                                      ^
SyntaxError: incomplete input
```

Compare the above code cell with the print statements in the earlier code cells. **Can you spot the error?** Try to fix it in the cell below - run the cell to see if you were successful:

In [ ]:
```
print("Debugging code means dealing with lots of error messages!")
```

Working With Cells

Again, each cell in a Jupyter Notebook is like a short paragraph or sentence. Learning how to work with them is important. Some things to keep in mind:

- Edit vs. Command mode:
  - When you're in Edit mode - ie. you're inside the cell - you are actively typing and changing its contents (ie. text/code). To activate edit mode, double click or hit the `Enter` key - think of this as activating "edit mode" just for that cell. To leave edit mode, hit the `Esc` key.
  - When you're in Command mode, you can do things TO the overaching cell, but not WITHIN the cell - ie. you can run it, delete it, copy it, move it, etc.
- Adding a new cell: click the `+` sign in the menu bar, or click immediately to left of a cell (activating Command mode) and use keyboard shortcut `b` to insert a cell "below" or `a` to insert a cell above that cell.
- Markdown vs. Code cells: always check the type of cell - you should see either `Markdown` or `Code` in the dropdown on the top menu bar, and you can change the type through that dropdown menu. Code cells also have brackets `[ ]:` to the left of them.

- You can move cells around to rearrange them - hover over the left side of the cell and click, hold, and drag it up or down.
- You can delete a cell by right-clicking to the left of the cell and selecting `Delete` or clicking to the left of the cell and typing `d` twice (`d` `d`).

Add Your Own Cells

Insert new cells immediately after this one.

- Cell 1 (Markdown): Write a sentence about your favorite kind of animal in Markdown.
- Cell 2 (Code): Use the same `print()` code you used above to have Python print out an identical sentence.

My favorite animals are cats and dogs. I have a Corgi at home,his name is Toby.

In [ ]:
```
print("My favorite animals are cats and dogs. I have a Corgi at home,his name is
```

Submitting Jupyter Notebooks as Coding Homeworks

Once you've finished, we're going to pretend this is one of your coding homeworks and go through the process of submitting it to Canvas. Follow the instructions I've made for submitting homework and then submit your files to the corresponding Sample Homework assignment page on Canvas.

- Save your notebook
- `Kernel` -> `Restart Kernel and Run All Cells`
- `File` -> `Print` and try to find an option to Save/Print to PDF. Depending on your operating system and browser, this might be `Destination` -> `Save as PDF`, `Select Printer` -> `Microsoft Print to PDF` (instructions for different browsers). Name the file with the same naming convention as your .ipynb file (ie. `hw-01-yourlastname.pdf`) and save the resulting PDF file (ending in `.pdf`) into the same folder.

Note: use the naming convention `yourlastname-jupyter-notebooks-intro.ipynb` & `yourlastname-jupyter-notebooks-intro.pdf`. This will help us troubleshoot the pipeline for submitting to Canvas and make sure that this pipeline also works on my end.

In [ ]:
```
```