

lab00

February 15, 2022

Name: Allan Gongora

Section: 0131

1 Lab 0: Introduction and Practice with Jupyter Notebook

In Lab 0, you will learn how to navigate a Jupyter Notebook (like this one). All of the required lab assignments in this course are published as Jupyter notebooks. You follow the instructions in the notebook to complete the assignment.

Let's get started!


1.1 1. Jupyter Notebooks


This webpage is called a Jupyter notebook. A notebook is a place to write programs and view their results.

1.1.1 1.1. Text cells

In a notebook, each rectangle containing text or code is called a *cell*.

Text cells (like this one) can be edited by double-clicking on them. They're written in a simple format called [Markdown](#) to add formatting and section headings. You don't need to learn Markdown, but you might want to.

After you edit a text cell, select the "run cell" button at the top that looks like  to confirm any changes.

Question 1.1 This paragraph is in its own text cell. Try editing it so that **this** sentence is the last sentence in the paragraph, and then select the "run cell"  button on the top.

```
[1]: print("Hello, World!")
```

Hello, World!

And this one:

```
[2]: print("\N{WAVING HAND SIGN}, \N{EARTH GLOBE ASIA-AUSTRALIA}!")
```

, !

The fundamental building block of Python code is an expression. Cells can contain multiple lines with multiple expressions. When you run a cell, the lines of code are executed in the order in which they appear. Every `print` expression prints a line. Run the next cell and notice the order of the output.

```
[6]: print("First this line,\nthen the whole ,")
      print("and then this one.")
```

```
First this line,
then the whole ,
and then this one.
```

Question 1.2 Change the cell above so that it prints out:

```
First this line,
then the whole ,
and then this one.
```

Hint: If you're stuck on how to print the Earth symbol, try looking at the print expressions above.

1.1.2 Writing Jupyter Notebooks

You can use Jupyter notebooks for your own projects or documents. They are among the world's most popular programming environments for data science. When you make your own notebook, you'll need to create your own cells for text and code.

To add a cell, select the + button in the menu bar. A new cell starts out as text. You can change it to a code cell by selecting it so that it's highlighted, then selecting the drop-down box next to the restart () button in the menu bar, and choosing Code instead of Markdown.

Question 1.3 Add a code cell below this one. Write code in it that prints out:

A whole new cell!

♪♪

(That musical note symbol is like the Earth symbol. Its long-form name is `\N{EIGHTH NOTE}`.)

Run your cell to verify that it works.

```
[7]: print("A whole new cell! ♪♪")
```

```
A whole new cell! ♪♪
```

1.1.3 Errors

Python is a language, and like natural human languages, it has rules. It differs from natural language in two important ways: 1. The rules are *simple*. You can learn most of them in a few weeks and gain reasonable proficiency with the language in a semester. 2. The rules are *rigid*. If you're proficient in a natural language, you can understand a non-proficient speaker, glossing over small mistakes. A computer running Python code is not smart enough to do that.

Whenever you write code, you'll make mistakes. When you run a code cell that has errors, Python will sometimes produce error messages to tell you what you did wrong.

Errors are okay; even experienced programmers make many errors. When you make an error, you just have to find the source of the problem, fix it, and move on.

We have made an error in the next cell. Run it and see what happens.

```
[8]: print("This line is missing something.")
```

This line is missing something.

You should see something like this (minus our annotations):

The last line of the error output attempts to tell you what went wrong. The *syntax* of a language is its structure, and this `SyntaxError` tells you that you have created an illegal structure. “EOF” means “end of file,” so the message is saying Python expected you to write something more (in this case, a right parenthesis) before finishing the cell.

There’s a lot of terminology in programming languages. You’ll learn as you go. If you are ever having trouble understanding an error message, search the discussion forum. If you don’t find an answer, post a question about the error yourself.

Try to fix the code above so that you can run the cell and see the intended message instead of an error.

1.1.4 The Kernel

The kernel is a program that executes the code inside your notebook and outputs the results. In the top right of your window, you can see a circle that indicates the status of your kernel. If the circle is empty (), the kernel is idle and ready to execute code. If the circle is filled in (), the kernel is busy running some code.

You may run into problems where your kernel is stuck for an excessive amount of time, your notebook is very slow and unresponsive, or your kernel loses its connection. If this happens, try the following steps: 1. At the top of your screen, select **Kernel**, then **Interrupt**. 2. If that doesn’t help, select **Kernel**, then **Restart**. If you do this, you will have to run your code cells from the start of your notebook up until where you paused your work. 3. If that doesn’t help, restart your server. First, save your work by selecting **File** at the top left of your screen, then **Save and Checkpoint**. Next, select **Control Panel** at the top right. Choose **Stop My Server** to shut it down, then **My Server** to start it back up. Then, navigate back to the notebook you were working on.

1.2 Submitting Assignments

Once you’re finished, select “Save and Checkpoint” in the File menu. Your name and course section number should be in the first and last cell of the assignment. Be sure you have run all cells with code and that the output from that is showing.

Double check that you have completed all of the free response questions as the auto-grader does NOT check that and YOU are responsible for knowing those questions are there and completing them as part of the grade for this quiz. When ready, click “Print Preview” in the File menu. Print a copy from there in pdf format. (This means you right click and choose print and choose “save as pdf” from your printer options.) You will need to submit the pdf in Canvas by the deadline.

The gopher grader output and/or output from your coding are essential to helping your instructor grade your work correctly and in a timely manner.

Files submitted that are missing the required output will lose some to all points so double check your pdf before submitting.

1.2.1 Completing a Lab

All assignments in the course will be distributed as notebooks like this one. At the top of each assignment, you'll see a cell like the one below that imports autograder tests. Run it to import the autograder tests.

```
[13]: pip install gofer-grader
```

```
Requirement already satisfied: gofer-grader in /opt/conda/lib/python3.7/site-  
packages (1.1.0)  
Requirement already satisfied: pygments in /opt/conda/lib/python3.7/site-  
packages (from gofer-grader) (2.11.2)  
Requirement already satisfied: tornado in /opt/conda/lib/python3.7/site-packages  
(from gofer-grader) (6.1)  
Requirement already satisfied: jinja2 in /opt/conda/lib/python3.7/site-packages  
(from gofer-grader) (3.0.3)  
Requirement already satisfied: MarkupSafe>=2.0 in /opt/conda/lib/python3.7/site-  
packages (from jinja2->gofer-grader) (2.0.1)  
Note: you may need to restart the kernel to use updated packages.
```

```
[14]: # Don't change this cell, just run it  
# Import autograder tests  
from gofer.ok import check
```

When you finish a question, you need to check your answer by running the check command below. It's OK to grade multiple times; Gofer will only try to grade your final submission for each question. There are no hidden autograder tests. If you pass all the given autograder tests for a question, you will receive full credit for that question.

```
[11]: check("tests/q0.py")
```

```
[11]: <gofer.ok.OKTestsResult at 0x7f1d1f026350>
```

The notebook resides on a server that is run by the course staff, and so we have access to it as well. Once you're finished with a lab, use the File menu within the notebook page (below the Jupyter logo) to "Save and Checkpoint" and you're done. You may also check your notebook in its entirety with the following command.

```
[12]: import glob  
from gofer.ok import grade_notebook  
if not globals().get('__GOFER_GRADER__', False):  
    display(grade_notebook('lab00.ipynb', sorted(glob.glob('tests/q*.py'))))
```

```
Hello, World!
, !
First this line,
then the whole ,
and then this one.

['tests/q0.py']
Question 1:

<gofer.ok.OKTestsResult at 0x7fd1c2abed0>

1.0
```

2 Further Reference

This notebook is an introductory notebook to help learn about what Jupyter Notebooks are, how you can use it, shortcuts for some common actions, and information about Gofer Grader which is used in this Data 8X course.

2.1 What is a Jupyter Notebook?

A Jupyter Notebook is a file/notebook that can contain both code and regular text (like this paragraph here!) as well as math formulas, images, links, and more. The fact that a Jupyter Notebook can contains all of these various elements makes it a valuable tool for people, especially when doing data analysis, since the analysis description and the results can be contained in the same document.

There are two ways to run and execute a Jupyter notebook: 1. On a local Desktop (which requires no internet access) 2. On a remote server (many Berkeley classes like Data 8X and Data 100 use this option so you can save your work and submit it online)

Every notebook has something called a **kernel**, which acts as the operation part of the notebook. With the kernel, we can execute the lines of code that are in a notebook. When you open a notebook, the kernel associated with that notebook is automatically launched as well. When we execute or run a notebook (more on this down below), the kernel performs the computations and then returns the results. To see when the kernel is running/executing, you can look at the top right hand corner of your notebook, at a circle. If the circle is gray/filled in, the kernel is currently executing cells. If the circle is not filled in, the kernel is not currently executing cells. In the diagram below, the circle shows us that the kernel is not executing anything at the moment.

When you first open your Jupyter Notebook, you are given an empty cell that you are free to edit. Cells can be of different types based on the content you want to write in it. Any Python code that is written will be in a “Code” cell, while paragraphs and images will be in a “Markdown” cell.

Cells can also be in Edit Mode or Command Mode. When a cell is in edit mode, in the top right corner of the menu, there will be a pencil icon (as shown in the image above). When the cell is in command mode, this icon will not be there. Based on which mode the cell is in, we can do different actions/shortcuts.

2.1.1 Actions you can do in a Jupyter Notebook

There are many different actions that you can do once you are within a notebook, a lot of which is in the top toolbar (containing icons) or the “File Edit View ...” menu. In this section, we will go over some common actions you can take to use your notebook.

2.1.2 Change Type of Cell

Based on what purpose you are using your Jupyter Notebook for, you may want one type of cell or various types of cells within your notebook. The most common types of cells are Markdown cells and Code cells, which exist among Heading cells and Raw NBConvert cells. For any cells in your notebook, you can change what type of cell it is by selecting the cell you want to change and then going to the top toolbar and looking for the drop down menu with all the types options (as shown in the diagram below).

2.1.3 Add a Cell

In order to add a cell to your notebook, select an existing cell and then navigate to the toolbar and click on the plus symbol. A cell will be added, below the cell you selected before. The cell will default be a “Code” type cell.

2.1.4 Delete a Cell

To delete a cell from your notebook, select the cell you want to delete (by clicking on the cell) and then navigate to the menu at the top and do **Edit -> Delete Cells**.

2.1.5 Run Cells

To run a single cell (so the kernel can evaluate what is in the cell), select the cell and then navigate to the menu and do: **Cell -> Run Cells**.

To run a single cell and then select the next cell after the evaluation is complete, select the cell and then navigate to the menu and do: **Cell -> Run Cells and Select Below**. This can also be done by selecting the cell, going to the toolbar and clicking on the step icon (the triangle with a line on the right side).

To run a single cell and then add a new cell after the evaluation is complete, select the cell and then navigate to the menu and do: **Cell -> Run Cells and Insert Below**.

To run all the cells in the notebook, navigate to the menu and do: **Cell -> Run All**.

To run all the cells above a certain cell, click on the cell then navigate to the menu and do: **Cell -> Run All Above**.

To run all the cells below a certain cell, click on the cell then navigate to the menu and do: **Cell -> Run All Below**.

2.1.6 Stop Executing

Suppose you run a cell (which is taking a long time to finish), and you want to stop the kernel from executing it. Simply closing the window that your notebook is in will not make the kernel stop executing. To do this, you can go to the menu and select **Kernel -> Interrupt**. Alternatively,

you can go to toolbar and press the stop icon (the full black square). Doing either of these actions will stop the kernel from executing the current cell.

2.1.7 Other Kernel Actions

There are other actions (besides interrupting) that you can do with your kernel. These actions are in the menu under **Kernel**.

Kernel -> Restart: This restarts the kernel and refreshes all the current variables in the kernel (variables are lost). The notebook cells are NOT run automatically but the output from the cells from before the restart are still there on the screen.

Kernel -> Restart & Clear Output: This restarts the kernel and refreshes all the current variables in the kernel (variables are lost). The notebook cells are NOT run automatically and the output from the cells from before the restart are cleared from the screen.

Kernel -> Restart & Run All: This restarts your kernel (and refreshes all the outputs/variables previously computed) and then runs all the cells in the notebook from the top.

Kernel -> Reconnect: This helps to reconnect to the kernel in the case that your kernel disconnects or is having a problem. If your kernel is not working, you can try this action so that you can connect and execute with the kernel again.

Kernel -> Shutdown: This shuts down the kernel so it will not be running or executing or taking up memory/storage.

2.1.8 Markdown Cells

Markdown cells can contain text, photos, videos, even code chunks that cannot be executed. To insert these elements and change the text (headings, bold, italics, links, etc), there is a language called Markdown Language.

With Markdown you can:

Make text **bold** by surrounding the text with double asterisks or double underscores.

Make text *italicized* by surrounding the text with single asterisks or single underscores.

- Create bulleted lists
- By adding
- An asterisk and a space and then your text

Insert a [link](#) into your text, by surrounding the text with brackets and immediately putting your url with parentheses around it.

Add headings by starting a line with one or more # followed by a space and your heading text, ### Like ## this # example

You can also embed code for illustration purposes (not execution) by surrounding the code with a single backtick, like `x = 5`.

It is also possible to include mathematical equations by surrounding the expression with single dollar signs. The way you write the math formulas follows LaTeX rules. To have the expression on

its own line, surround the expression with double dollar signs. $e^x = \sum_{i=0}^{\infty} \frac{1}{i!} x^i$

$$e^x = \sum_{i=0}^{\infty} \frac{1}{i!} x^i$$

There are many more commands and tricks you can do with Markdown, check out this cheat sheet for more tips: <https://medium.com/ibm-data-science-experience/markdown-for-jupyter-notebooks-cheatsheet-386c05aeebed>.

2.1.9 Save Notebook

To save your notebook in the location where it is currently, you can click on the save icon in the leftmost side of the toolbar or go to File -> Save and Checkpoint in the Menu. Checkpoints are created with this command so you are able to go back to the checkpoint if needed (similar to Google Docs and how you can see previous edits and versions of the document).

2.1.10 Cut, Copy, Paste Cells

Similar to how you can cut, copy, and paste text in other editors like Google Docs or Word, you can do the same with cells in Jupyter Notebook. To do any of these commands you can select the cells and then navigate to the toolbar - the scissor icon is the Cut command, the double pages icon next to it is the Copy command, and the clipboard and page icon next to that is the Paste icon. Alternatively, you can do the same commands by going to the Menu and clicking Edit (where there are commands Cut Cells, Copy Cells, Paste Cells Above, Paste Cells Below, Paste Cells and Replace).

2.1.11 Move Cells Up and Down

It is also possible to move cells up or down if you want to change their location within the notebook. This can be done by selecting the cell(s) and clicking the up or down icons (based on where you want the cell to be). Each time the buttons is pressed, the cell selected will move one cell in that direction. This can also be done with Edit -> Move Cell Up or Move Cell Down.

2.2 Jupyter Notebook Shortcuts

In the previous section, we talked about various actions we can do within a Jupyter Notebook. All of these actions required going to the Toolbar or the Menu at the top of the notebook. Doing this for every single action can be tiring, so luckily for us, there are a number of neat shortcuts to help us do the same actions.

Some of these shortcuts require you to be in the Edit Mode or the Command Mode of a cell. Remember, you are in Edit Mode if you can see the pencil icon in the top right corner of the Menu.

To activate command mode, press **Esc** and to activate Edit Mode, press **Enter**.

Below the shortcuts are grouped based on what mode you are in.

2.2.1 Shortcuts for Both Modes

- Run the current cell and select below: **Shift + Enter**
- Run the selected cell: **Ctrl + Enter**

- Run the current cell and insert cell below: **Alt + Enter**
- Save and checkpoint: **Ctrl + S**

2.2.2 Shortcuts while in Command Mode

- Show all shortcuts: **H**
- Select cell above: **Up**
- Select cell below: **Down**
- Extend selected cells above: **Shift + Up**
- Extend selected cells below: **Shift + Down**
- Insert cell above: **A**
- Insert cell below: **B**
- Cut selected cells: **X**
- Copy selected cells: **C**
- Paste cells above: **Shift + V**
- Paste cells below: **V**
- Delete selected cells: **D, D** (press D key twice)
- Undo deleting cell: **Z**
- Save and checkpoint: **S**
- Change to Code cell type: **Y**
- Change to Markdown cell type: **M**
- Scroll notebook up: **Shift + Space**
- Scroll notebook down: **Space**

2.2.3 Shortcuts while in Edit Mode

Many of these are similar to other text editors because we are doing them in Edit Mode. * Code completion or indent: **Tab** * Indent: **Ctrl +]** * Dedent: **Ctrl + [** * Select all: **Ctrl + A** * Undo: **Ctrl + Z** * Redo: **Ctrl + Y** * Go to left of line: **Ctrl + Left** * Go to right of line: **Ctrl + Right** * Move cursor down: **Down** * Move cursor up: **Up**

There are other shortcuts that also exist, many of them are documented online as well as in the Jupyter Notebook documentation.

Name: Allan Gongora

Section: 0131