Lab00

January 23, 2022

1 Lab 0 - The Adventure Begins...

Note: As discussed in the Jupyter Books starting webpage for this lab, you will receive two copies of the second part of Lab 0:
1) the **assignment** version (assigned via GitHub Classroom) and
2) the usual **course materials** version (pulled from the `course-materials` repo). This is the 'usual' version of Lab 0. Most Labs will be available via Jupyter Notebooks `pull`ed from the `course-materials` repo.

In general, completion of labs is completely up to you. Each lab will have a **final** thoughts question or request for a post to the #lab_submission channel on Slack. So the **submission step** for *most* labs is a post on slack.

The overarching goal for this lab is to ensure that your computer is ready for the course. In this lab, we will: * Install git, get you connected to GitHub, and connect you to the course GitHub Classroom * Install the packages that you need for now * Introduce you to `jupyter` notebooks

1.1 Preamble

This lab is unusual in a number of ways. First and foremost, it starts on a website. Every lab following will be found on our GitHub Classroom. This first part of the lab will set you up to access our GitHub Classroom.

The overarching goal for this lab is to ensure that your computer is ready for the course. In this lab, we will:
* Install git, get you connected to GitHub, and connect you to the course GitHub Classroom * Install the packages that you need for now
* Introduce you to `jupyter` notebooks

1.2 Acknowledgements

This lab draws heavily on content from two places:
1. McFee and Kell's Open Source and Reproducible MIR Research Tutorial from the Conference of the International Society of Music Information Retrieval (ISMIR) 2018.
2. Jacob Fiksel's GitHub Classroom Guide for Students

We – your instructor and her pedagogical partner – have updated or adapted language from these two sources for our course. This is our first example of where we are **curating** resources for you. The value-add for our curation for *our* course is that we are 1) updating these documents to be in-line with current practices, and 2) adapting them for our particular course.

1.3 Lab outline

There are 7 steps for this lab. The first six parts are through this website (compiled via `Jupyter Books`) and the last part will be in a `Jupyter notebook` pulled from the class GitHub Classroom through git. Here are the seven steps:

1. Install (or update) both Anaconda and git
2. Get an ssh key, connect to GitHub
3. Prepare your computer to access course materials
4. Clone course-materials to your computer
5. Receive an **assignment** in GitHub Classroom
6. Use conda to install a few packages
7. Run the local version of the lab on your computer

Once we have completed these seven steps, you will be ready to interact with Lab0 in Jupyter Notebooks.

We have already completed steps 1-6 To review these steps, please visit this website.

1.4 Checkpoint 1

If you arrived here, then you have correctly set up git, connected to GitHub, and connected the assignment version of Lab 0. Excellent work!

Remember that you will be receiving an email inviting you to the full GitHub classroom. ***Please accept this!*** All materials for the course – including labs, slides, homework – will be on our GitHub Classroom. Moodle will rarely be used for our course *except* as the place where I will report the grades for the course.

Note: Moodle does not calculate your grade as I will at the end of the term. It's just not functionality that Moodle has. If you want to track in the style that I will at the end of term, please copy this form into your Google drive.

1.5 Welcome to `Jupyter`!

This kind of file is known as a `jupyter notebook`. It is a kind of file where we can both code and write in small *cells*. Our labs will largely be `jupyter notebook`s. The goal of this section is to get used to a few of the functions within a `jupyter notebook`.

Notebooks like these are quite standard in industry and academia alike. We can use them for testing ideas before putting something into production or to communicate about an existing package or library. For each homework assignment I recommend making your own notebook to act as a place for your scratch work.

To learn more about the features of `jupyter notebook`s, check out 'Python for Data Analysis' starting at the bottom of page 18 through page 24.

1.6 Creating a new notebook

When you opened this window, you created a new tab in your internet application. Leaving this tab open, click on the tab that originally opened. In the top right of the window, you will see two buttons "Quit" and "Logout". Do **not** click these. Instead look down to the next set of buttons `Upload` and `New`.

Click on `New` and select `Python 3`. A new notebook should open in your internet application window.

Next to the jupyter logo, it will say *Untitled*. Give your notebook a name. Something like `Lab0-scratch` is great!

1.7 Cells

Everything in `jupyter` is run in cells. In our course, we will primarily be using `code` and Markdown cells. The `code` cells will be running Python underneath and the Markdown files will let you stylize text using a version of Markdown.

1.7.1 Code Cell

When you opened your notebook, there was a grey box that appeared just below the menus. Move your cursor inside that box. Borrowing from Part 7 of McFee and Kell's Tutorial, try the following in your cell: * Type `print("Hello Jupyter")` * In the menu, click "Run"

You should see `Hello Jupyter` as output just below your cell, followed by another grey cell.

1.7.2 Markdown Cell

We are going to turn this new gret cell into a Markdown cell. In the menu, a few buttons over from "Run", you should see a drop down menu currently set to "Code". Click on this menu and select "Markdown." Now this cell will not engage with Python (unless of course, you re-select "Code".)

In the **Markdown** cell, try the following: * Type `print("Hello Jupyter")` * In the menu, click "Run"

What happens?

Just like the code cells, we can always edit these boxes by double-clicking on the resulting text. I think of the Run button as a sort of publish button for my markdown text.

1.8 Saving our work

This section will borrow from a section of Part 3 as adapted to our class. As before, any text in purple (with edits in black) is directly lifted from McFee and Kell's Tutorial with permission from the authors.

We created a new notebook, hooray! But we've only created it in our local version. Let's commit this to our local git repository, and then update our code on GitHub.

First, make a new branch to work on. Working in a branch means that we can do whatever we want without breaking our main code. Do `git checkout -b new-notebook`.

Next, add and commit your files: `git add Lab0-scratch.ipynb`, then git commit -m "Adding my first jupyter notebook"
* ***Note***: If you want to commit all changes to all files, you can use `git add -A`.

Rad! We've saved our work locally. To push your code to GitHub, do `git push origin new-notebook`.

(You can think of all this git and GitHub magic like you are mailing something to a friend. `git checkout -b` creates the letter you're going to mail. `git add` selects what you're going to put into your letter. `git commit` puts the changes into your letter, and then `git push` sends your paper to the post office.)

We want to get in the habit of keeping our online work up to date with our local work. So let's go look at GitHub, and merge this branch there.
* Visit GitHub, and find your fork of this repository: https://github.com/
* You'll see a list of branches - select `new-notebook`.
* You'll see a small button that says **New pull request**. Press it!
* You'll get taken to a page that says Comparing changes, and should see a dropdown that says Base fork: _____.
* **You may need to change this!**
* You need to select left dropdown and change it to point to your fork, with your username. Once you do this the left drop down should read `base: main`. Next we need to update the right dropdown. You want this to read `compare: new-notebook`. * After this, you'll be taken to a nice web form. Put in the reasons for your changes, and press **Create Pull Request**.

Hooray, you've made a Pull Request! Pull Requests (PRs) are the best way to contribute code to a project. You can review & comment on the proposed changes before merging them, or request that things be improved. To finish the mail metaphor, a Pull Request is like adding some notes to your letter about what it is supposed to do and why you are sending it.

Here's an important aside: when discussing code in PRs, please be nice! It's really easy for a neutral comment to be read as mean, and it's really easy to alienate people by being mean.
You should do your best to very polite and kind when dealing with people on GitHub (and probably on computers in general).

With that said, let's merge our PR! This will add the changes from your branch to the main branch.

- Press **Merge**. GitHub will merge your commit into main.
- Now your online repo is up to date, so let's update your local.
- In your command line, run `git checkout main`.
- Next, do git pull origin main to update your local copy. You'll see that git adds your commit from GitHub to your local copy.
- Finally, do git branch -d update-conda-env to delete your local branch – you've merged the commits to main, so you can delete it.

We've just gone over the basic git and GitHub workflow that you should, in general, stick with: (Stated here as a list for emphasis) 1. make a branch, 2. make some changes, 3. run tests – (we will get to this in Lab 1) –, 4. commit, 5. push, 6. merge, 7. clean up.

This means that you never have untested code in main, and it means that main is always functional. It also means that the online, "deployed" version of main is your primary version, and that you (and maybe many other collaborators) just have a local version that you're working on.

We'll come back to why this method is useful when we talk about tests (in Lab 1). If you like, you can read more about this way of working with code on GitHub.

1.8.1 Final Thoughts

To finish up this lab, create at least 3 code cells and at least 3 markdown cells. Fill your cells with a variety of python commands (in the code cells) and markdown commands (in the markdown cells). Commit your changes to your homework repository. I will check these to ensure that everything went smoothly.

Then create a post to **#lab_submission** channel something cool that you learned from this lab.

If your have questions from this lab, post them to #lab_questions with the same preamble (i.e. starting with **Lab0**). If you have the same question, please use one of the emoji's to upvote the question. If you would like to answer someone's question, please use the thread function. This will tie your answer to their question.

1.8.2 Resources Consulted

1. McFee and Kell's Open Source and Reproducible MIR Research Tutorial from the Conference of the International Society of Music Information Retrieval (ISMIR) 2018.

2. Jacob Fiksel's GitHub Classroom Guide for Students

3. How to Use the Equivalent of the "ls" Command in Windows
4. Command Prompt: 11 basic commands you should know (cd, dir, mkdir, etc.)