Now that you understand the course workflow at a high level, let's see how to actually execute the process using GitHub and Git.

Forking a Repository

The first step will be to fork our class repository. Forking a repository is a way in which, on GitHub, you create a copy of a repository that becomes your own copy on your own GitHub. This allows you to make any changes, and save those changes, in your own repository. To create a fork, navigate to the repository at the following address.

Python Course Overview

On the page that pops up, you will see a Fork button on the upper right of the site. Click that to create your own copy of the repository. Your browser window will show you the home page for your new repository.

Cloning the Repository Locally

The next step is to create a copy of the repository locally on your own machine. This is called *cloning*. Remember that the `clone` command creates a complete copy of the entire repository including all versions of all files. This is different than centralized version control systems such as SVN that only put a working copy on your local machine.

To create a clone, you will need the clone URL for the repository on your GitHub account. In your browser, make sure that you are looking at your own fork of our class repository on GitHub. Look to the right of the screen and find the box labeled "HTTPS clone URL." To the right of the address in the box is a button that copies the address. Go ahead and push it.

Open a command prompt and navigate to your course working directory. Then enter the following command, pasting the URL you copied after `clone`.

git clone https://github.com/MIDS-Python-Bridge-Course/Course-Overview.git

Git should ask you for your GitHub login in order to access your account. If the process does not work correctly, further instructions can be found at this link: https://help.github.com/articles/set-up-git/

This creates a new directory called Course-Overview inside your course directory, initializes a .git directory inside it, and pulls down all the repository data from github. Navigate into this directory and to the week_3 folder.

cd Course-Overview/week_3

You can type `ls` to confirm that all of the week's files are there.

Completing an Exercise

Now that you have your own repository on your local machine, let's practice making some changes. Start IPython Notebooks running in your browser and navigate to the week_3 directory in your course working folder. Click on the file named 3.8.1_GitHub_Exercise.ipynb to open it. You will see a simple practice exercise in this file. Go ahead and edit the code cell to complete the exercise, then save your file.

Next, let's commit the changes to your local repository. Go back to your command terminal and type

```
git status
```

This should confirm that you have a modified file in your repository. Go ahead and add the file.

git add 3.8.1_GitHub_Exercise.ipynb

Then commit your changes.

git commit -m "completed 3.8.1_GitHub_Exercise.ipynb".

Now you have successfully updated your local repository.

Pushing Changes to GitHub

Now that you have completed the assignment and committed your changes locally, it is time to push your changes up to your GitHub repository. First, you should run `git status` to confirm the branch that you are on as well as the state of your code. You cannot push code that is in `staging`; you can only push code that is committed (however, you can run the command without fully committed code). For this exercise, make sure that all your code is currently committed.

The syntax of the push command is straightforward: `git push <remote_name> <branch>`, where `remote_name` and `branch` will be filled in by us. For this exercise we will push to `origin`, and we will push the `master` branch. *Origin* is a name automatically given to the remote GitHub repository we cloned from. We only have one branch, named *master*, so that is the one we will push to.

```
git push origin master
```

When we execute that statement, we will see a printout that should be similar to the one shown below.

```
Counting objects: 10, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (10/10), done.
Writing objects: 100% (10/10), 7.15 KiB | 0 bytes/s, done.
Total 10 (delta 2), reused 0 (delta 0)
To git@github.com:MIDS-Python-Bridge-Course/Course-Overview.git
   6e5921a..f975b78  master -> master
```

We will not concern ourself with exactly what that output tells us; we just want to make sure that it ended positively and did not give us an error. Now let's look at GitHub and find that our code is there.

Congratulations, you have just pushed your first code to GitHub. We will do this a lot throughout this course.

Creating a Pull Request

As you work through this course, you will push changes to your own GitHub repository, but you will never need to change the central repository owned by the instructors. (If you did, other students that fork the central repository afterwards would get your answers.) As you can imagine, however, if you were working with other people on a large coding project, you would often want to make changes to your own repository, then copy those changes to a central repository for all other programmers to use.

If you create changes that you believe belong in the original repository, you can navigate to the central repository in your browser and create a pull request. This notifies the repository owner that you have changes and allows the owner to pull those changes into his or her repository. You can learn more about that here.

GitHub's Pull Request Instructions

A pull request can be useful if you realize you may have misspelled or forgotten something. You can simply push again (to your repository), and those changes will be reflected in the pull request.

Conclusion

This has been a quick introduction to Git and GitHub. We have focused on the features you need to succeed in this course. We will return to the topic of Git later to discuss more advanced features.