Computer Science CS134C (Fall 2018)
Duane A. Bailey
Laboratory 1
Introduction to the Python/git workflow
(Monday: due noon Thursday, Tuesday: due noon Friday)

Objective. To become comfortable working with Python and git.

This week we will spend a little time on some little programs. These programs will give you a taste of what it is like to work with Python in a computer scientist's environment. Beneath the Mac O/S is a Unix-based, extensible operating system. We'll be using `git` to check out and turn in our work. We'll introduce you to `emacs`, an editor suitable for a wide variety of tasks. Finally, we'll get our first exposure to `python3`, the primary tool of this semester's study. Your job, this week, is to begin developing a working knowledge of these systems. It will be worth it.

New Environment Setup.

The following should be done the first time you use any new environment. These tasks only need to be done once on any new machine. Over the semester, you may use machines in CS labs, OIT machines, or, perhaps, your own computer. You can tell if you're setup for use in this environment by looking for a `cs134` folder in your Home folder on the Mac (`Shift-Command-H`). If it doesn't exist, you need to set up the environment as follows:

1. If necessary, find and install the `Terminal` application in the dock. Click on the application to start it. Almost all of our work will take place in `Terminal`'s window.

2. Create a directory that will hold all of your CS134 work. Throughout the course we will assume this directory is called `cs134`.[1]

   ```
   mkdir cs134
   ```

3. Now, descend into the `cs134` directory.

   ```
   cd cs134
   ```

4. All of the resources for this course (including the entire web site) are available for download wherever you need to work on this course. It's simple and useful to clone the shared repository (replace 22xyz with your username) into a new folder called `shared` in the `cs134` subdirectory of your home directory. When asked for a password, it's the password associated with your CS account.

   ```
   git clone ssh://22xyz@davey.cs.williams.edu/~cs134/shared.git ~/cs134/shared
   ```

   Here, `davey` is the secure shell server (`ssh`) dedicated to holding all of our collective work. The `cs.williams.edu` is the Computer Science IP domain at Williams. When you connect to a server that has access limited to specific individuals, the `22xyz@...` sequence indicated you'll be accessing the server using the username 22xyz; by default it will try to access the server using the username

[1] You can name it anything you like, but in the future you'll have to remember to change instructions so they work with your particular setup.

you're accessing the local machine with—which is, perhaps, incorrect. The `cs134` reference is the course home (`~`) directory and `shared.git` is the name of the repository on the server.

You can explore around in the shared repository, but you should probably avoid changing anything there. If you want to experiment, copy the files of interest into another directory.

5. Next, you need to clone the `starter` repository. This repository contains a few aids to making a new machine a bit more useable. You can also store files or notes here that you may keep with you as you use other machines.

   ```
   git clone ssh://22xyz@davey.cs.williams.edu/~cs134/22xyz/starter.git ~/cs134/starter
   ```

   Notice that your username appears *twice* in the command line; make sure you change it in both places. In the `cs134` home directory on `davey`, we have a separate directory we share with each student in the course. In this directory are all the private repositories we will need for the course.

6. The first time you use any machine you need to set up git with commands that look like:

   ```
   git config --global user.name "Your Name"
   git config --global user.email "your-email@williams.edu"
   git config --global push.default simple
   git config --global core.editor emacs
   ```

   This creates an the identity you'll use for turning in work. I've collected these commands in a "script" called `configure` in the `~/cs134/starter` folder. Once you edit the `configure` script to reflect your identity, you can have the computer read and execute the commands in `configure` with

   ```
   source ~/cs134/starter/configure
   ```

   Now, you're ready to begin work.

This week's lab. Make sure you've followed the instructions for setting up your environment. This week we'll focus on managing the workflow associated with the course. We'll start by exploring around and making a few changes that personalize our environment.
You can now change your directory to be the `starter` subdirectory of `cs134` with:

```
cd ~/cs134/starter
```

Whenever you begin a session of work, you should make sure you get the latest copy of your work. This is called *pulling* the repository from the server. You should

```
git pull
```

Since we *just* cloned the repository, it's unlikely that anything is out-of-date. Still, we always perform this simple check.

First, we'll edit the `configure` file and personalize it. Change your name and email address so you get credit for your edits. Once you've got things just right, you can (again):

```
source ~/cs134/starter/configure
```

We'll tell `git` that we changed the file:

```
git add configure
git commit -m 'Personalized my configuration file.'
```

The `commit` command will only store the changes of files you've add-ed.

We can send these committed changed back up to `davey`:

```
git push
```

You'll be asked for your CS password. From now on, any time you pull the `starter` repository, the `configure` script will personalize your environment accurately.

Sometimes we'll create new files we want to turn in for credit. Suppose we create a file called `hello.py`. We should add the file and commit changes to our repository:

```
git add hello.py
git commit -m 'My first program!'
```

You should commit every time you think you've made progress. Committing is an important part of managing the progress you make. It is also helpful in backing up your work.

When you're finished with a work session, you should commit one last time and *push* the changes to the server:

```
git commit -am 'Done with work today!'
git push
```

The -a switch causes all tracked files (any file that has ever been add-ed) to be committed. This guarantees you get the latest version of your work every time you pull.

Tasks Required for Credit. You must push to the server the following before noon this Thursday (for Monday labs) or noon Friday (for Tuesday labs).

1. Edit, add, and commit the `configure` file to personalize your username and email settings.
2. Write your first python program, `hello.py`. It should simply print out `Hello, world!`. Welcome, programmer! Add and commit this first file.
3. Create a text file called `ABOUT.txt` that tells me a bit about you. What's your focus at Williams? Where are you from? Where, in your hometown, is the best place to eat? **Make sure the lines of your text are wrapped, and not just one long, continuous line.** Add and commit the `ABOUT.txt` file.

⋆