Computer Science CS134 (Fall 2020)

Daniel Aalberts, Duane Bailey, & Molly Feldman

Laboratory 1

Introduction to the Python/git workflow

(Due 5pm Thursday)

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. Beneath most modern operating systems is a Unix-based, extensible operating system. We'll be using `git` to check out and turn in our work. We'll think about what makes an editor suitable for programming. 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.

Setting up your environment. Before we begin our first lab exercise, it is important that we make sure that we have all the utilities we need. We assume that you have completed all of the configuration described in the handout *Setting Up Your Computer*. You can find this handout on the course website, `http://www.cs.williams.edu/~cs134`.

When you are working from off-campus (all of us will, at some point this semester), you will need access to the Williams VPN, by installing Cisco's `AnyConnect` software. See OIT's Software page `http://oit.williams.edu/software` for installation instructions.

Establishing your identity. We have sent you your Computer Science credentials: a username and password that allows you to access our facilities. Please, make sure you change your password. Those credentials identify who you are, with our servers. We also sent you a two digit code that will be your anonymous identifier throughout the semester. At times, we may ask you for your two digit id to identify who you are, with our graders.

We also need to help your machine identify who you are, as well. Before you turn in your first assignment, we must make sure you have established your `git` identity. If you have not already done this during setup, carefully type the following commands into the shell, replacing Joe Cool's identity with your own:

```
git config --global user.name 'Joe Cool'
git config --global user.email 'jc8@williams.edu'
git config --global push.default simple
git config --global core.editor nano
```

Your identity is used to digitally sign your submitted work.

Organizing your work. Unix *directories* correspond to *folders* in your Mac or Windows operating system. You will be creating many files on your computer for this class. We suggest that you put all of your work in a single folder, called `cs134`, located in your *home directory*. From within the `Terminal` window (on Macs) or the `Ubuntu` app window (on Windows)[1], do the following:

[1] From this point on, when we refer to the `Terminal`, we're referring to either the `Terminal` app window on Macs, or the `Ubuntu` app window on Windows machines.

1. Create your cs134 directory:

   ```
   mkdir cs134
   ```

2. Now, descend into the cs134 directory:

   ```
   cd cs134
   ```

3. At this point, we're going to establish secure connections with our git server, evolene:

   ```
   curl http://www.cs.williams.edu/~cs134/evolene.pem -o ~/cs134/evolene.pem
   git config --global http.sslCAInfo ~/cs134/evolene.pem
   ```

   This downloads the certificate for evolene and ensures that git uses it whenever we exchange information with the server. Do not change or move the file evolene.pem, or you will find you will no longer be able to connect to the server. Again: this only needs to be done once this semester.

4. If you are off campus, start the VPN server. (It should connect to ssl-vpn.williams.edu.)

5. Now that we're set-up, we're going to retrieve the files we need to complete this week's lab. This will create a new directory in your cs134 folder with the name lab01 containing your lab work. Type

   ```
   git clone https://evolene.cs.williams.edu/cs134-labs/22xyz3/lab01.git lab01
   ```

   Here, evolene.cs.williams.edu is the full name of the git server dedicated to holding all of our collective work. The cs134-labs reference is the CS134 lab directory on that server (many courses make use of evolene), and lab01.git is the name of the repository. The identifier 22xyz3 should be replaced with your CS identity. It is the folder that contains all of *your* repositories on the server. You will be asked for your CS credentials.

   If you've made it this far, congratulations! Now, you're ready to begin work.

This week's lab. This week we'll get introduced to the weekly workflow associated with the course.

1. You can now change your directory to be the starter subdirectory of cs134 with:

   ```
   cd ~/cs134/lab01
   ```

   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 `hello.py` file, adding the following Python commands:

```
# My first python program!
print('Hello, world!')
```

Save the file. Now, at your prompt (represented by the $, here) run the Python script in the shell:

```
$ python3 hello.py
Hello, world!
```

Congratulations, you're a programmer! We'll tell `git` that we changed the file:

```
git add hello.py
git commit -m "I'm a programmer!"
```

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

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

```
git push
```

You'll be asked for your CS password. From now on, any time you pull the `lab01` repository, the `hello.py` file will reflect the changes you just made.

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

```
git add goodbye.py
git commit -m 'Another great 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 for backing up your work.

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

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

The `-a` switch causes 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.

2. Use your editor to write a paragraph about yourself in a file called `AboutMe.txt`. Who are you? What do you enjoy doing? Where are you from? Where can you get good food in your hometown? Add and commit this work:

    ```
    git add AboutMe.txt
    git commit -m 'Added restaurant recommendation.'
    ```

3. The `GradeSheet.txt` file describes the expectations we have for each lab, and is where you can find our comments after we grade your work. You might find it useful to look at this file before you turn in you work to make sure you have not missed an important part of the assignment.

4. Every week you will need to certify that your work is your own. Edit `honorcode.txt` and, if appropriate, write your name below the Honor Code statement. If you received help from other students in the course, you must indicate who that was, as well. Now, commit that work:

    ```
    git add honorcode.txt
    git commit -m 'Signed the honor code.'
    ```

5. **Final Task.** Now, push your work up to the server, to be graded, by Thursday at 5pm.

    ```
    git push
    ```

    Make sure that you always push your work up to the server. If you don't do this, we cannot grade your work!

What to expect. Your work is due on Thursday, at 5pm. Over the course of the next few days, we provide feedback and grade your work. After we have graded your work, you can use

```
git pull
```

to download our comments, found in the `GradeSheet.txt` file.

⋆