Code Submission Guidelines

College: CUNY School of Professional Studies
Course-Name: Software Application Programming I
Course-Code: IS 210

Overview

This is a rough outline of the pull request workflow process used throughout this course in the submission of coding assignments. For a more in-depth analysis of the pull request workflow, see the Week Two materials in the course site.

Workflow Steps

To get access to the starter code, you must first *Fork* the starter repo on *GitHub* and then clone the newly created repository to your working machine. The link to the starter repository is found in the body of the assignment in the Course site.

> ***Warning***
>
> Be careful about which repository you are forking. Make sure you fork only your specific course section repository.

Once you've forked the starter code to your personal GitHub repository, open a terminal in your development environment.

Getting the Starter Code

```
$ git clone HTTPS_CLONE_URL
```

Where `HTTPS_CLONE_URL` is the HTTPS Clone Url found on your *personal* fork of the starter repo. Please be cautious and check that you're cloning your repo and not the parent repository. To check, make sure that your username is in the Clone URL:

https://github.com/YOUR-USERNAME/some-repository-name.git

Enter the Newly Created Local Repository

Use the `cd` command to enter the starter repository directory.

```
$ cd some-repository-name
```

Where `some-repository-name` is the name of the git repository you just clonded. This will change with each lesson but is found in the Clone URL as the part after the last slash (`/`) and before `.git`.

Use `ls` to see the available files:

Example:

```
$ ls
LICENSE new_python.py README.rst
```

Begin Work

You may now begin work. Use whatever editor your prefer to work on and run your code. You may also use Git Bash to run python files, eg:

```
$ python new_python.py
Some value
```

Remember, you can call your program with `python -i` to start an interpreter after the program runs. This will let you investigate the value of variables which will now be accessible from the python interactive command line. This is a helpful way to debug your work in progress.

Example `new_python.py`:

```
my_var = 'Some value'
my_new_var = my_var * 2
print my_new_var
```

```
$ python -i myprogram.py
Some valueSome value
```

```
>>> print my_var
Some Value
>>> print my_new_var
Some valueSome value
```

You may also launch the IDLE Python editor, the preferred editor of this course, from Git Bash.

```
$ idle new_python.py
```

This works the same whether you're accessing an existing Python file or want to create a new Python file called `new_python.py`.

Running Tests

After confirming your code works, it's always a good idea to run tests locally to see if the tests catch something you didn't. In most cases, you can use the test script provided with your repository:

```
$ ./runtests.sh
```

You can also PyLint files directly if syntax is all that you're after:

```
$ pylint myprogram.py
```

Where `myprogram.py` is the name of the file you with to lint.

Saving your Work

While you are welcome to use any pattern you wish, I recommend saving your work after each task by issuing a commit. Then, when you're finished for a coding session, issuing a push to save the work in GitHub, just in case something happens do your developer machine in the interim.

To save your work, first check what files have changed in your repository.

```
$ git status
On branch master
Your branch is ahead of 'origin/master' by 3 commits.
  (use "git push" to publish your local commits)

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   old_python.py

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        new_python.py
```

Now add the files you've recently worked on to staging. The `add` command adds changes, not files, so it must be used to add new and existing files alike.

```
$ git add new_python.py old_python.py
```

Run `git status` again to check that the files have been added.

```
$ git status
On branch master
Your branch is ahead of 'origin/master' by 3 commits.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   new_python.py
        modified:   old_python.py
```

Everything looks good, so commit your changes.

```
$ git commit -m "Here's my commit message about what I did."
```

This saves your work locally. Now lets push it to our remote repository.

```
$ git push origin
```

You may repeat the steps in this section as many times as you need or want as you iterate your work or respond to test results.

Submission

Code should be submitted to GitHub by means of opening a pull request.

As-of Lesson 02, each student will have a branch named after his or her GitHub username. Pull requests should be made against the **base** branch that matches your GitHub username. Pull requests made against other branches will be closed.

For a refresher on how to open a pull request, please see the pull request video in the Week Two course materials.

In order to receive full credit you must complete the assignment as-instructed and without any violations (reported in the build status). There will be automated tests for this assignment to provide early feedback on program code.

When you have completed this assignment, please post the link to your *pull request* in the body of the assignment on Blackboard in order to receive credit.