Homework 1

Due on: Friday, 01.29.2021 at 11:59 pm ET.

Welcome to ECE 20875, Python for Data Science! In this class, you will learn the basics of various topics in data science, and, along the way, learn how to write code in Python.

Goals

This homework has several objectives:

1. Get familiar with git and GitHub: cloning, committing, tagging, etc.
2. Get familiar with the GitHub Classroom submission system.
3. Write a simple Python script demonstrating Python fundamentals.

Background

Version Control

Git is a version control tool. What is version control? It is a means of retaining organized versions of your code as you work on a project. This is especially important in industry for code organization, collaborative projects, and large projects spanning many files. Further information on how Github might be used in collaborative projects can be found at https://guides.github.com/ if interested. You will not be concerned with branches in this class.

When working with repositories for this class, there are three main parts to working with git:

1. The remote repository (on Github)
2. The local repository (on your device/ Scholar account)
3. Your working files (on your device/ Scholar account)

Note: Git is the tool you use to do version control. Github is where your remote repository is hosted.

When you do a `git add` followed by `git commit`, you are updating your local repository. When you do a `git push`, you are updating the remote repository on Github (supervisors or collaborators can now see your changes).

The key commands you will need to use in this assignment (and at other times) are:

1. `git clone`: This sets up a local repository on your machine by "cloning" (copying) a remote repository; in this case, a repository set up on GitHub.
2. `git add`: When you have changed a file, you can "stage" it for the next version (i.e., tell git that you want the changes in this file to appear in the next version) using this command. (Note, when you want to just add/stage changes made to all files in a directory, consider "git add --all". Changes may include deletion of certain files.)

3. `git commit` : Creates a new version of your code. This new version is, essentially, all the files you called `git add` on, plus all the *other* files in the previous version (that you have not `add` ed). This new version only exists in your local repository.
4. `git push` : Updates the remote repository (on GitHub) with all the changes that you have `commit` ted to your local repository. Note that uncommitted changes will not appear on GitHub.
5. `git pull` : Updates your local repository with any changes that are on GitHub and have not yet been reflected in your local repository. Note that there may be some conflicts between what GitHub knows about a file and what your local repository knows, and this command will notify you if that happens (fixing those conflicts requires more work). You should always run `git pull` and resolve any conflicts before running `git push` .
6. `git status` : This shows you the status of any files in your local repository. In particular, it shows you any files that you have modified, but not yet `add` ed, and any files that you have `add` ed but not yet `commit` ted.

We will be using the latest version of your code (files present after your most recent commit) at 11:59pm on the submission deadline for grading.

Push your code to GitHub often. Not only does that prevent you from losing any code if you accidentally delete anything, it helps us help you debug, by giving us access to your latest code.

NOTE: You should verify that your code files are showing up correctly on GitHub once you have `push` ed them from your local machine.

Python

Familiarize yourself with basic python scripting from the lecture notes. The Python language documentation is online at https://docs.python.org/3/

This assignment has the potential to involve:
Initializing variables. Recall that unlike C, C++, etc., Python does not require a data type declaration.
Arithmetic (+,-,*,/,%,**,//), logical (`and` , `or` , `not`), and comparison (<,>,<=,>=,==,!=) operations.
Decision (if,elif,else) structures.
A basic understanding of lists and how to index them.

A caution: DO NOT use `&` and `|` . They are for bitwise operations, which should not be used for conditional statements. Using them could lead to very hard bugs to find. As an example, `2 & 1` will give `0` , which will be read as `False` whereas `2 and 1` gives `1` , which is read as `True` . Instead of `&` and `|` you should use `and` and `or` in Python.

Remember that indentation is Python's inherent way of organizing code blocks.

Ex:

```
if True:
    print('HW1 is truly fantastic.')
print('I concur.')
```

Getting Started

Clone your HW1 repository by clicking on the GitHub classroom link distributed through Piazza. This will create a repository on GitHub with your user name. Use `git clone` to clone that repository locally. If you face issues with this, then you might want to look into using 'ssh key' to clone (directions at end of README).

There should be three files in your repository when you begin:

1. This README file.
2. `problem1.py` , a python file with one line of code, which contains a single year.
3. `problem2.py` , a python file that has questions you must answer using the format specified in the file.

Instructions

After completing the 'Getting Started' section above, you will complete 'problem1.py', 'problem2.py' and verify it works using the provided test cases. You may then submit your code to Github using the git commands above and following the 'What to Submit and How' section below.

Problem 1

Modify 'problem1.py' by adding additional lines of code so that it solves the following problem: Determine if the year given at the top of the file in 'problem1.py' is a leap year. A leap year is when the year value is divisible by 4, but not divisible by 100 unless it is also divisible by 400. For example, 500 is not a leap year, but 1200 is a leap year. Print `True` if the `year` variable is a leap year or `False` if the `year` variable is not a leap year.

Note: Your code should be correct if this year variable is changed to any positive integer year. We plan to test your code with other years.

For example, if we set the year variable to `8` , your code should print `True` . If we set the year variable to `2010` , your code should print `False` . If we set the year variable to `13` , your code should print `False` . (You can temporarily change the value of the year variable to test your code, but you **SHOULD** return it to its original value `2021` before submission.)

Some other test cases are the years: 800, 2020, 583, 1100, 1994, 2015, and 3132.

Problem 2

Follow the instructions in `problem2.py` to answer the questions inside the file. Please do not change the order or modify any of the pre-filled commands except where specified.

What to Submit and How

The only files you need to modify for this homework are `problem1.py` and `problem2.py` .

Once you have a version of your code (that you have `commit` ted using `git commit` and `push` ed using `git push`) that you are happy with, you are done! Do not make changes to code that has been provided to you except where specified.

NOTE: You should verify that your code files are showing up correctly on GitHub once you have `push` ed them from your local machine.

You might want to try using the ssh key if you face issues while submitting homework1, see the final section of the readme for information on this.

Resubmitting:

If you want to update your submission before the deadline then re-commit and re-push your files. Ensure that you do both `git commit` and `git push` on your updated files so that they appear on GitHub.

SSH key (if needed)

Directions for adding ssh key on GitHub: Here is a link to instructions if you need more help: https://help.github.com/en/github/authenticating-to-github/adding-a-new-ssh-key-to-your-github-account

You may want to use ssh keys to clone your repository if cloning using https gives you problems. To do this, you need to first generate an ssh key pair on your local machine. On your local machine, open a terminal and type:

1. ssh-keygen -t rsa -C "your_email@example.com" make sure "your_email@example.com" is the same email used on GitHub
2. Press "Enter" when prompted where to store keys. This will result is saving the keys to a default location.
3. Enter a secure pass phrase that you will remember.
4. That's it! Keys will be saved in .ssh/

Now, you must enter your public key on GitHub. To do this, type 'cd .ssh/' on your local machine and then copy your PUBLIC key. Then, login to your GitHub repo, click on your profile picture in the upper right hand corner, click on Settings, click on SSH and GPG Keys, click New SSH Key, create a title so you will know which computer the ssh key corresponds to, paste the PUBLIC key, and finally click Add SSH Key.