Introduction to Programming Homework 7

Due Friday Nov 11

You will turn in your homework **via GitHub**! **Please do Exercise 1 before starting on the other problems.** After Exercise 1, Exercise 2 has the most detailed `git` instructions, so if you choose to do Exercise 3 first, be sure to read the `git` instructions in Exercise 2.

Exercise 1 (Git and GitHub)

You will set up a GitHub account and start using `git` in this exercise.

For macOS or Linux

Open up a terminal window (`/Applications/Utilities/Terminal.app` on macOS) and see if you have `git` installed by typing

```
git version
```

followed by pressing ENTER. If you see something like `git version 2.9.3 (Apple Git-75)`, then you have `git` installed. If you get an error, then go to https://git-scm.com/downloads to download and install `git`.

For Windows

If you have Windows 10 installed, follow these instructions to setup the Ubuntu bash shell built into the latest version of Windows 10 : https://msdn.microsoft.com/commandline/wsl/install_guide?f=255&MSPPError=-2147217396.

If you have an earlier version of Windows, download and install Babun from https://babun.github.io/. From now on, when I refer to the terminal app, please use either the `bash` program or `Babun` on Windows.

Author and email (for everyone)

Now you should all be able to open up your terminal app, type `git version`, and get a git version number. If you can't get past this step, please email me.

Once this works, we need to let `git` know your name and email address so that all your commits will be appropriately marked. Type these commands out in the terminal :

```
git config --global user.name "Your Name"

git config --global user.email "your_email@wherever.something"
```

Now, we can move on to creating a GitHub account and the repository for this homework.

GitHub (for everyone)

Go to https://github.com and create a free account. Once you have done so, please go to the following link to join our classroom page :

https://classroom.github.com/assignment-invitations/ad9d115831ef34e2d7af2067ed4b92f0

Once you have done so, you will be told that your assignment has been created and you can follow the link to it. There you will see an address of the form :

```
https://github.com/itp-uni-lux/homework-7-YOUR-GITHUB-USERNAME.git
```

We will now create a folder for your repository using the **terminal app**.

- use `pwd`, `ls`, and `cd` to move to a directory on your disk where you store your homework files for this class
- use

  ```
  mkdir homework_7
  ```

  to create a folder called `homework_7`
- go into your newly created folder using

  ```
  cd homework_7
  ```

- check that you are in the right folder by using `pwd` just to be safe
- run

  ```
  git init
  ```

  to **initialize a blank repository**
- use the command

  ```
  echo "# homework-7-YOUR-NAME" >> README.md
  ```

  to create a text file called README.md
- run

  ```
  git status
  ```

  to see what is going on
- we will now make `git` track your file by calling :

  ```
  git add README.md
  ```

- run

  ```
  git status
  ```

  to note the difference
- we will not make the first commit :

  ```
  git commit -m "Created README.md"
  ```

- run

```
git log
```

to see what you just did

We are now in a state where your have a **local** `git` repository on your disk in a folder called `homework_7`. You have also added a file called `README.md` that contains the text "# homework-7-YOUR-NAME" (without quotation marks).

We would now like to **push** this repository to the GitHub cloud. To do this, we need to tell `git` where this **remote** repository is stored. We will use the link given to us by GitHub the the following command :

```
git remote add origin https://github.com/itp-uni-lux/homework-7-YOUR-GITHUB-USERNAME.git
```

Now we can push our commits to GitHub with :

```
git push -u origin master
```

Got back to https://github.com (or straight to the repository link with https://github.com/itp-uni-lux/homework-7-YOUR-GITHUB-USERNAME) where you will find your new README.md file!

Your second commit (for everyone)

Edit the README.md file on your computer using your favorite text editor (I still recommend using Atom). You can add to the file whatever you want (e.g. add a line saying "I am learning to use git!"). Don't forget to save.

Got back to the terminal app and make sure you are back in your `homework_7` directory. From there

- check what files have been edited with

  ```
  git status
  ```

- check the differences with

  ```
  git diff
  ```

- to **stage** your changes for the next commit run :

  ```
  git add README.md
  ```

- commit your changes with :

  ```
  git commit -m "Updated README.md for my second commit!"
  ```

- **push** your updates with :

  ```
  git push origin master
  ```

- let's take a look at your (pretty) commit log by running :

  ```
  git log --graph --decorate --all
  ```

We are now all set to begin the next programming exercises.

Exercise 2 (Conditionally Convergent Series)

git setup

In your terminal app **create a new branch** using the command

```
git checkout -b cond_conv
```

you can now type

```
git branch
```

to check that you are indeed in the **new** `cond_conv` branch.

coding

Go back to your favorite text editor and create a file called `ccs.py`

- **a.** Recall that if $\{a_i\}_{i\in\mathbb{N}} \in \mathbb{R}$ and
$$\sum_{i=0}^{\infty} a_i \text{ converges but } \sum_{i=0}^{\infty} |a_i| \text{ diverges}$$
then for every $\beta \in \mathbb{R}$ we can find a bijective map $\sigma_\beta : \mathbb{N} \to \mathbb{N}$ such that
$$\beta = \sum_{i=0}^{\infty} a_{\sigma_\beta(i)}$$

Given `beta` and a **generator function** `cond_conv_seq()` write a generator function `rearrange(cond_conv_seq,beta)` that yields $a_{\sigma_\beta(i)}$ in order.

Once you have finished, **add** and **commit** your code in `git`.

- **b.** Write a function called `test_rearrange(cond_conv_seq, error)` that picks a random float `beta` and returns `True` if after some finite number of steps N, $\left|\beta - \sum_{i=0}^{N} a_{\sigma_\beta(i)}\right| < error$ using your generator function from part **a.** Run a few tests using

```
def cond_conv_seq() :
  """ Generates (-1)^(n+1)/n. """
  n = 1.
  sign = 0
  while True :
      yield (-1)**sign/n
      n += 1.
      sign = 1 - sign
```

Once you have finished, **add** and **commit** your got to `git`.

git final steps

Got back to terminal and run

```
git status
```

to make sure you **you don't have any uncommitted changes.**

Take a look at your git repository history with

```
git log --graph --decorate --all
```

You will see at the top `(HEAD -> cond_conv)` and two commits down you will see `(origin/master, master)`. This means that your code is two commits ahead of `master`. Now that you have written working code, let's **merge** your `cond_conv` branch into `master`. You can only merge **into** a branch you have currently checked out, therefore, we will need to switch to the master branch first.

We switch to the `master` branch by calling

```
git checkout master
```

You will be switched to the `master` branch (you can check this by calling `git branch`). Now we **merge** the `cond_conv` **into** the `master` branch with :

```
git merge cond_conv
```

To see what happened, we can run

```
git log --graph --decorate --all
```

again and you should see `(HEAD -> master, cond_conv)` at the top right. It remains now to **push** our **local work** to GitHub with :

```
git push origin master
git push origin cond_conv
```

Note : the second command pushes your branch `cond_conv` to GitHub, without it, I would not be able to see that you ever created the branch locally!

Remark : if you see an image such as

you can exit from this screen by typing :

```
:wq
```

followed by pressing ENTER. Note that this is the character ':', 'w', 'q' followed by ENTER.

Exercise 3 (Random Text Files)

git setup

In your terminal app **create a new branch** using the command

```
git checkout -b random_text
```

you can now type

```
git branch
```

to check that you are indeed in the **new** `random_text` branch.

coding

Go back to your favorite text editor and create a file called `random_text.py`

- **a.** Recall that a partition of a positive number n is a tuple of numbers $(a_1, \ldots, a_k)$ such that $a_1 \geq a_2 \geq a_k \geq 1$ and $a_1 + \cdots a_k = n$. Write a function `partitions(n)` which returns the list of all partitions of `n` (i.e. returns a list of tuples). If you can, make your function a generator.

Once you have finished, **add** and **commit** your code in `git`.

- **b.** Write a function called `generate_random_text(file_name)` which will write random text to a file called `file_name` using the following algorithm :
  - pick a random number `num_lines` from $\{1, \ldots 100\}$
  - for each line pick a random number `num_letters` from $\{1, 20\}$
  - pick a random partition `p` of `num_letters` using part **a**
  - for each `i = 0 ... len(p) - 1` generate a random word `w[i]` of length `p[i]`
    - you can use the module `string` the data `string.ascii_letters` for sampling the letters for `w[i]`
    - make sure you choose randomly **with replacement**
    - to create a string out of a list of characters, use `''.join(char_list)`
  - write a the words `w[i]` on a line of the file `file_name` **separated by a space**
  - make sure you have written `num_lines` lines to the file using this algorithm

Once you have finished, **add** and **commit** your code in `git`.

- **c.** Write a function called `count_capitals(file_name)` which will read a file at the path `file_name` and return a dictionary mapping line numbers (starting with line 1) to the number of capital letters in each line. You can use `str.isupper` to test if a character is uppercase or not.

For example, if `file_name` contains

```
o Agy fId ZGIBwAGM bnSxLFLCcHZpjab
M TiRmMn weyfZVKT wNftXrUrjuLmECV
```

then your code should return

```
{ 1 : 16, 2 : 15 }
```

Once you have finished testing your code. Then, **add** and **commit** your code in `git`.

git final steps

Got back to terminal and run

```
git status
```

to make sure you **you don't have any uncommitted changes.**

Take a look at your git repository history with

```
git log --graph --decorate --all
```

Now that you have written working code, let's **merge** your `random_text` branch into `master`.

```
git checkout master

git merge random_text
```

To see what happened, we can run

```
git log --graph --decorate --all
```

It remains now to **push** our **local work** to GitHub with :

```
git push origin master
git push origin random_text
```

Note : the second command pushes your branch `random_text` to GitHub, without it, I would not be able to see that you ever created the branch locally!