Worksheet 01

Name: Daniyal Ahmed

UID: U11469883

Topics

- Git

Prerequisites (installations)

This is your checklist:

- ☑ Access to terminal
- ☑ Install Git
- ☑ Sign up for a GitHub account
- ☑ Choose editor
- ☑ Set up ssh keys
- ☑ Configure git

Step 1: Work Environment: Access to Terminal

- Mac/Linux: use **Terminal**
- Windows:
  - Option 1: Power Shell
  - Option 2: Git Bash (recommended)

Step 2: Install Git

- Mac:
  - Git
- Windows:
  - Git for Windows (Git Bash)
- Linux:
  - Install Git on Linux

Confirm Git is installed by typing `git --version` on your terminal

Step 3: Sign up for a GitHub Account

Go to github.com

Step 4: Choose a Graphical Editor

- Try Visual Studio Code
  - Visual Studio Code
- OR one of these other editors
  - Sublime Text 3
  - Atom
  - Notepad++ (for Windows)

Step 5: SSH Setup

Mac & Linux Users

Go to home directory (in terminal)

```
% cd ~
% pwd
/Users/gallettilance
```

Go to `.ssh` directory

```
% pwd
/Users/gallettilance
% cd .ssh
% pwd
/Users/gallettilance/.ssh
```

Note: If you do not have the `.ssh` directory, you can create it

- if you are in your home directory:
  - `mkdir .ssh`
- if you are not in your home directory:
  - `mkdir ~/.ssh`

Generate `id_rsa` keypair files if needed

- **Note:** these `id_rsa` files contain a special password for your computer to be connect to network services (Ex: GitHub, AWS).
- Check to see if these files exist by typing `ls -alt`
- If you do not have these two files (`id_rsa` and `id_rsa.pub`), create them by typing:
  - `ssh-keygen`
  - Hit `enter` **3 times**

```
% pwd
/Users/gallettilance/.ssh
% ls
% ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/gallettilance/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```

```
Your identification has been saved in /Users/gallettilance/.ssh/id_rsa.
Your public key has been saved in /Users/gallettilance/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:jmDJes1qOzDi8KynXLGQ098JMSRnbIyt0w7vSgEsr2E gallettilance@RESHAMAs-
MacBook-Pro.local
The key's randomart image is:
+---[RSA 2048]----+
|   .=+           |
|.  .==           |
|.o  +o           |
|..+= oo          |
|.E.+X.  S        |
|+o=o=*oo.        |
|++.*o.+o.        |
|..*.oo           |
|o= o+o           |
+----[SHA256]-----+
% ls
total 16
-rw-------  1   1675 Dec 17 12:20 id_rsa
-rw-r--r--  1    422 Dec 17 12:20 id_rsa.pub
%
```

Navigate to the `.ssh` directory

```
cd ~/.ssh
```

open `id_rsa.pub` using your editor of choice and copy its contents. Add `ssh` key to GitHub by following these steps:

- go to your GitHub account (create one if you don't have one, and save your user name and password somewhere easily accessible for you.)
- click on your avatar/profile picture (upper right of screen)
- go to `Settings`
- on left of screen, select `SSH and GPG keys`
- Select `New SSH key`
- for "Title": entitle it "GitHub key"
- for "Key": paste key from clipboard here
- click `Add SSH key`
- save, exit, confirm GitHub password as requested

Windows Users

Follow How to Create SSH Keys with PuTTY on Windows

For Windows 10 or 11

- Press Windows+R
- Enter cmd

- In the opened Command Prompt, type in “ssh-keygen”
- Press Enter
- You can choose to enter a passphrase, it will not be displayed.
- Go to the shown path to find your file named id_rsa.pub Ex. C:\Users\user/.ssh/id_rsa.pub
- Open the file with a notepad and copy everything
- Go to github and click settings at top right
- Go to SSH and GPG keys, click New SSH key and paste your SSH key here.
- Click Add SSH key. You might be asked to enter Github password.
- Go back to your Command Prompt and type in “ssh -T git@github.com”
- Enter your SSH passphrase.
- You will see “You’ve successfully authenticated” in the following message. Which means you are now connected to Github.

Step 6: Configure Git

Configure user name and email (lets Git know who you are)

```
git config --global user.name "First Last"
git config --global user.email "myname@email.com"
```

To verify these additions, type:

```
git config --list
```

Default Editor

The default editor will be Vim. You may want to look up how to edit, save, and close vim as this can't be done with just point and click (you must use the vim commands).

Git / GitHub (In Class)

a) what is the difference between git and github?

Github is a company that hosts git, where git itself is a version control system that helps store different versions of projects and allows you to update your project as you progress. Because it is a version control system, it allows you to go back to a previous version incase you had some aspect about that version you preferred.

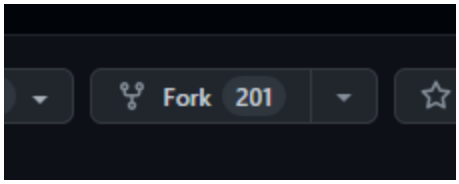
b) what command would you use to copy a repo locally?

You can use Git clone

Hence:

git clone

c) what button would you use to make a copy of a repo in GitHub?

you can fork a git repo by clicking this button: 

It is usually in the upper right hand corner

d) let's say you have a copy of a repo in GitHub but that repo changes, does your copy on your laptop change too? why / why not?

No it does not, when you clone a repo from github you download that version of that repo and only that version. If the repo has changed and you need to push your version to Github, you will encounter an error because it messes up the timeline since git doesn't know where exactly to place your version

e) what are the three commands you use to create a new save point in your git repo and back it up to GitHub?

In [ ]:
```
git add
git commit -m "Type your message here"
git push
```

The first git command adds any file you need to the upload stream The Second git command commits, here you tell what changes you made to the files that you are uploading new versions of finally you push them to git hub with git push

f) how would you make your local and remote copies change so that they have the most up-to-date version of the repo they are copied from?

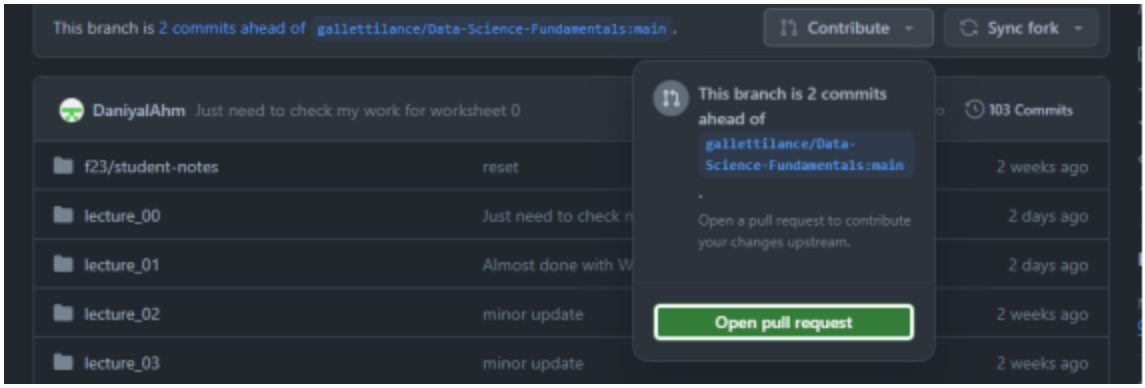
In [ ]:
```
"you can use "
git pull <url>
"alternatively if you need to pull from some specific branch then what you can do i
git pull origin <branchname>
```

g) why are there sometimes conflicts between copied repos / branches? How do you resolve them?

- If you copy an earlier version of your repo and git push it git will get confused on where to put it and simply not allow it. one way to counter this is by creating more branches to push your work to, however this is more of a work around than a complete solution since you are pushing to a new branch not the same one. Yes one can argue that you can 'merge' branches however this takes time and can be painstakingly slow at times.
- Another way, this could happen is you are working on a group project and both you and your peer clone a repo and make changes to it, but your peer pushes before you, simply put it once you try to push your version after the peer git will not allow it since you are technically working an older version of the repo. One Counter to this both of you make forks of the repo, work on them individually and come up with a way to merge them

h) describe all the steps needed to make a PR to contribute your notes to the class repository.

Step 1: you fork a copy of the class repo and then you may want to take notes in the folder that correlates to the lecture, thus you either take notes in that folder or copy your notes there Step 2: You must git add the file you took the notes in for example "git add lecture_00_notes.pdf" Step 3: you must commit the notes to your own forked copy of the repo for example "git commit -m Here are the notes for the first lecture if anybody needs them" Step 4: you must then "git push" Step 5: Then on git hub you can click the contribute button on github, it looks like this:



It will allow you to compare the changes as well as add comments and when you are done you can click the button that says "Create Pull request"

i) Write here some other commands we used in class and what they mean / how to use them:

In [ ]:
```
git add <file> #Add files to be pushed
git remove <file> # files to be removed from the current commit
git commit # Commit files for them to be pushed
git init #make the current directory into a repo directory
git status # status of git repo
```

```
git diff  # shows what has changed between the last commit and now
git push # pushes to commit
git switch #where we switch to another branch
```

A bit more explanation:

- git add, adds the file to the current upstream, these are files you want to push to GitHub
- Alternatively if you changed your mind and want to work on a file a little bit more before you commit it then you can use git remove to remove the file from being commited
- git inint is a useful tool to initialize a folder or directory on your computer as a git repository, this way git will let you push and pull from that folder itself.
- git status tells you what files that you have made changes to need to be pushed and what files are in the commit
- git diff Shows what files are changed between now and when you git added them
- git push pushes the files to git hub, if you don't use git add and git commit before this then you will get a misleading message that says "Everything up to date"
- git swtich helps you switch to another branch in your repo

Exercise

a) Create a public repo on your github called "polynomial". Create a folder on your laptop called "polynomial" and initialize it as a git repo. Add a remote called "origin" pointing to the github repo you just created. Create a file called "polynomial.py" with the following code:

```
class X:
    def __init__(self):
        pass

    def __repr__(self):
        return "X"

class Int:
    def __init__(self, i):
        self.i = i

    def __repr__(self):
        return str(self.i)

class Add:
    def __init__(self, p1, p2):
        self.p1 = p1
        self.p2 = p2
```

```
    def __repr__(self):
        return repr(self.p1) + " + " + repr(self.p2)

class Mul:
    def __init__(self, p1, p2):
        self.p1 = p1
        self.p2 = p2

    def __repr__(self):
        if isinstance(self.p1, Add):
            if isinstance(self.p2, Add):
                return "( " + repr(self.p1) + " ) * ( " + repr(self.p2) +
" )"
            return "( " + repr(self.p1) + " ) * " + repr(self.p2)
        if isinstance(self.p2, Add):
            return repr(self.p1) + " * ( " + repr(self.p2) + " )"
        return repr(self.p1) + " * " + repr(self.p2)


poly = Add( Add( Int(4), Int(3)), Add( X(), Mul( Int(1), Add( Mul(X(),
X()), Int(1)))))
print(poly)
```

add and commit this file with the following message "cs 506 exercise part a". Push these changes to github.

b) In this exercise, you will write code to define and evaluate polynomial expressions. You should write out polynomials yourself to test various use cases / edge cases. Using the code above as an example, write classes for:

- division (called `Div`)
- subtraction (called `Sub`)

you may modify the `Mul` class if needed. Ensure that the rules of parentheses are properly encoded in your `repr` functions. When you're done with this part, add and commit the changes with the message "cs 506 exercise part b". If you need to modify the code from this exercise at a later time, you must use the interactive rebase so that all changes related to this section are in the relevant commit.

c) Write an `evaluate` method to each class that can evaluate the polynomial for a given value of `X`.

```
poly = Add( Add( Int(4), Int(3)), Add( X(), Mul( Int(1), Add( Mul(X(),
X()), Int(1)))))
print(poly.evaluate(-1))
```

When you're done with this part, add and commit the changes with the message "cs 506 exercise part c". If you need to modify the code from this exercise at a later time, you must use the interactive rebase so that all changes related to this section are in the relevant commit.

d) Provide the link to this github repo below

https://github.com/DaniyalAhm/polynomial

Exercise

Fork the course repo. Clone that fork locally. Ensure there is a remote called origin pointing to your fork and add a remote called upstream pointing to the course repo. Create a new branch called "worksheet_01". Checkout this new branch. In the `student_notes` folder, create a new file called `<your_last_name>_<your_first_name>_worksheet_01.txt`. In this file, answer the following question:

> A friend presents you with a coin they claim to be fair. You flip the coin 5 times and it lands on Heads every single time. You flip the coin another 5 times, same result. How many times must this happen for you to start doubting the fairness of the coin? Explain your reasoning a bit.

add and commit this change with the message "contributing class notes". Push the changes to the origin/worksheet_01 branch. Create a Pull Request against the course repo from this branch on github. Provide a link below to this PR.

Go back to the main branch so you can repeat this process in future worksheets or when you want to add any class notes for extra credit.

https://github.com/gallettilance/Data-Science-Fundamentals/pull/377