

Worksheet 01

Name: Wilbert Nathanael Limson UID: U11894403

Topics

- Git

Prerequisites (installations)

This is your checklist:

- [V] Access to terminal
- [V] Install Git
- [V] Sign up for a GitHub account
- [V] Choose editor
- [V] Set up ssh keys
- [V] 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:jmDjes1q0zDi8KynXLGQ098JMSRnbIyt0w7vSgEsr2E 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?

Git is a tool that helps you keep track of changes made to files. It's like a time machine for your code, letting you go back to previous versions if needed. While GitHub is a website where you can store your Git repositories online. It lets you share your code with others and work on projects together.

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

with the command `git clone [URL of the repository]`

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

Fork Button

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 doesn't. Your local copy doesn't automatically update when the original repo on GitHub changes. You need to manually update it using Git commands.

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

1. git add . or git add [file name] - This adds your changes to a staging area, preparing them to be saved.
2. git commit -m "Your message" - This saves your changes locally and adds a message explaining what you did.
3. git push - This uploads your saved changes to GitHub.

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?

Use git pull to update your local copy with the most recent version from GitHub. Use git push to update the remote copy on GitHub with your local changes.

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

Conflicts occur when two copies of a repo or branches have changes in the same part of a file, and Git doesn't know which version to keep. To resolve them, you manually edit the files to decide how to merge the changes, then save the file, commit the changes, and push them to the repository.

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

1. Fork the class repository on GitHub to make your own copy.
2. Clone your forked repository to your computer.
3. Create a new branch for your changes.
4. Make your changes and commit them.
5. Push your changes to your forked repository on GitHub.
6. Go to the original class repository on GitHub.
7. Click the "Pull Request" button.
8. Choose your fork and branch with your changes.
9. Fill in the details of your PR and submit it.

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

1. git status: Shows the status of your changes, like what's been added or changed.
2. git branch: Lists all the branches in your repo or lets you create new ones.
3. git checkout [branch name]: Switches you to a different branch.
4. git merge [branch]: Combines the changes from one branch into another.

5. git log: Shows a history of all the commits made.

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/Wilb11/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/380>