Terminal

You will find this neat utility in your applications. It comes by default on computers.
Unfortunately, although git commands are the same under mac and windows, basic terminal commands are not and can sometimes vary significantly
This is why we recommend to pair up with somebody working on mac, if you have windows.

The concepts explained in this workshop are machine-agnostic, so you can take time to toy with your respective machine after if you want and you won't lose out on anything.

What is a Terminal? The short version

"The terminal is an interface in which you can type and execute text based commands."
In a terminal, you **tell** your computer what to do. You type commands and execute them to get the machine to do what you want, instead of pointing and clicking on the graphical interface.

The commands are predefined so that the computer understands them.
They have this form: `'command -options arguments'.` The options are optional (you guessed it).

What is a Terminal? The longer version

If you've never peeked under the hood of a computer/used a terminal, you have gotten used to telling the machine what to do by clicking on things (icons, folders, files …). With a terminal, you are going to be **telling** your computer what to do, literally. We are going to forget about the graphical interface and write down the commands for the things we want the computer to do. We write them down in the terminal, and press **enter** to execute them.

The commands are predefined so that the computer understands them. They take the following form: "*command -options arguments",* with the *"-options"* part optional. Each command has various options you can use to tweak what they do.

An important concept is the **path**. When you use the graphical interface, you never have to worry about 'where you are' (i.e 'which folder am I in'?) in the computer, because it all starts at the desktop and you can use the finder to navigate through various folders. If you want **file_b** in **folder_a**, you're going to use the finder to find **folder_a**, double click to get to **file_b.**
This is straightforward enough, but things change in the terminal. Here, the computer sees every folder as sitting in a 'tree'. The root of the tree is the home folder, branches are chain of folders along with their subfolders and the leaves are files. Every folder on your computer is on this tree, as well as every file.
The path is the text representation of this tree. It looks like this -

The home folder: '/Users/USERNAME'
Desktop folder: '/Users/USERNAME/Desktop/'
Folder_b: '/Users/USERNAME/Desktop/**some_folder/folder_b**/'
Etc.

Navigating the path, you navigate your computer. If you're looking for **file_a,** you would *cd* into folder_b through its path, then type *ls* to check the contents of **folder_b**. This is a key to getting comfortable with the terminal.

The pros – terminal is lightning fast to use and does not take up a lot of memory space. If you know the commands, you can get the machine to do complex manipulations that would be a pain to do by point-and-click, but take 2 commands.
Cons – a learning curve.

Recommended reading if you're interested –
http://blog.teamtreehouse.com/introduction-to-the-mac-os-x-command-line
https://www.learnenough.com/command-line-tutorial
https://en.wikipedia.org/wiki/Command-line_interface

Here are the commands we will use the most, along with a few essential ones -

- **Navigation**
- Where am I? (displays current path): ‘pwd’
- Move up a folder: ‘cd ..’
- Move into a folder: ‘cd <**folder_name**>/’, given folder we are cd-ing into is in current folder
- Open the current folder in finder: ‘open .’

- **Investigation**
- Print out the contents of the folder I’m in: ‘ls’

- **Creation**
- Make a folder: ‘mkdir <**folder_name**>’
- Make a file: ‘touch <**file_name**>’

- **Git**
- Check that it is installed: ‘git –version’
- Initialize it in a folder: ‘git init’
- Check status: ‘git status’
- Commit changes:
    1. ‘git add <**file_to_add**>’
       (shortcut – ‘git add .’)
    2. ‘git commit --message ‘**your message**”
       (shortcut – **‘git commit –m ‘your message’)**

- Push commits: ‘git push <**remote**> <**branch**>’
- Pull commits: ‘git pull **<remote> <branch>**’

- See previous commits: ‘git log’
- Create a new branch and move to it:
    - ‘git branch <**branch_name**>’
    - ‘git checkout <**branch-name**>'

- Delete a branch: 'git branch –d <**branch_name**>’
- Merge a branch into current branch: ‘git merge <**branch_to_merge**>’
- Check the difference between two branches: ‘git diff <**branch_a**> <**branch_b**>’

- List all branches: ‘git branch –a’
- List all remotes: ‘git remote –v’
- Add a remote: ‘git remote add <**remote_name**> <**remote_url**>’
- Clone a repo: ‘git clone <**url**>’