GW ACM command line (bash) Cheat Sheet

Note: every time we reference *file* or *directory*, we are referencing the **path** to that file/dir!

Moving Around

COMMAND	**WHY?**
cd directory	Change directory- use this command to go to a folder in the current location
cd ..	By passing the "..", you're saying "go back one directory". Use this to move go back
ls dir	List the contents of the current directory
pwd	Print working directory- prints the full path of your current directory
.	Reference to current directory
..	Reference to directory above
~	Reference to home directory (root)

File Basics

COMMAND	**WHY?**
cp inputfile output-location -r	Copy the input file to the specified location. Using "-r" allows for recursively copying contents of directories
mv inputfile output_directory	Move the specified input file or directory to the specified output location
mkdir directory_name	Create a directory in the current location
rmdir directory_name	Delete an *empty* directory
rm filename	Delete a file
rm -r directory_name	Delete recursively- this deletes a directory and its contents. NOTE: you might need to add "sudo" at the front of the command

vim Basics

What is **vim**? It's a text editor that lets you edit files from the command line.

COMMAND	**WHY?**
vim filename	Open vim with the selected file (can be a file in current directory or new file name). Don't forget the ending! (for example: .txt, .csv, .java)

GW ACM command line + Git Workflows Cheat Sheet

Note: every time we reference file or directory, we are referencing the **path** to that file/dir!

Commands for Inside vim

COMMAND	**WHY?**
i	Enter "insert" mode. You can only write/delete text when in insert mode
v	Enter "visual" mode. Lets you make text selections (you can copy&paste)
ESC (escape key)	Exits the current mode or command
:w	"Write" - saves the file
:wq	Write (save) and quit
:q	Quit
:q!	Force quit (doesn't save)

THE GIT WORKFLOW

1. **Make Sure No One Else Made Changes -** Run **git pull** to make sure your repo is up-to-date
2. **Make Some Changes -** Edit some files locally! (Using whatever editor you prefer)
3. **Tell Git You Made Some Changes -** Using **git add file**, tell git that you want to keep track of the changes you just made. Using "git add" tells git that these files should be added to the "stage", and will be a part of your next commit

4. **Tell Other People You Made Some Changes -** When you're done with your commit, use **git commit -m "message"** to write a commit message.

When In Groups/Larger Projects, commit new features to a different branch other than **master** to avoid merge conflicts!

Tips:

- **Write meaningful commit messages!** "Made some changes" is a useless commit message, especially if something breaks...
- **Commit often!** Any new major piece of code should be its own commit, so that if something goes wrong, it's easy to track down what caused the problem
- **Make sure your repo is up-to-date!** (and "git pull" a lot!) If you're making commits to an outdated repo, then you're almost certainly going to run into *merge conflicts*