Guide to Terminal on Mac

CodeCademy "Learn the Command Line"

- Navigating, Viewing, and Changing the File System
  - `pwd` outputs the name of the current working directory
  - `ls` lists all files and directories in the working directory
    - `ls -a` lists all contents of a directory, including hidden files and directories
    - `ls -l` lists all contents in long format
    - `ls -t` orders files and directories by the time they were last modified
  - `cd` switches you into the directory you specify
  - `mkdir` creates a new directory in the working directory
  - `touch` creates a new file inside the working directory
  - `cp` copies files
  - `mv` moves and renames files
  - `rm` removes files
  - `rm -r` removes directories
- Redirecting Input and Output
  - `cat` may be used to read, concatenate, and create files
  - `>` redirects standard output of a command to a file, overwriting previous content
  - `>>` redirects standard output of a command to a file, appending new content to old content
  - `<` redirects standard input to a command
  - `|` redirects standard output of a command to another command
  - `sort` sorts lines of text alphabetically
  - `uniq` filters duplicate, adjacent lines of text
  - `grep` searches for a text pattern and outputs it
  - `sed` searches for a text pattern, modifies it, and outputs it
  - `head` returns a few lines of a any input
  - `echo` returns a string
    - `$VAR` prints the value of a variable
- Configuring the Environment
  - `nano` opens a text editor in the command line
  - `clear` empties the terminal screen
  - `~` represents the user's home directory.
  - `.` indicates a hidden file.
  - `source` makes the changes available right away in the session we are in
  - `~/.bash_profile` script file
    - `alias COM="COMMAND"` creates a shortcut
    - `export VAR="VALUE"` creates an environmental variable
    - `PS1` is the command prompt characters
  - `PATH` list of directories containing scripts for the command line to execute
  - `USER` is the name of the current user
  - `env` returns a list of environment variables
- Other Stuff
  - `*` wildcards are useful for operating on groups of files and directories
    - examples: `file *`, or `ls *.txt`, or `rm *phrase*`
  - `r` makes a command recursive
  - `i` makes a command case independent
- Philosophy
  - order is (1) commands (2) arguments (3) options
  - multiple options may be used together
  - multiple commands may be used together through pipelining `|`
  - working in a different directory is possible by specifying its path
  - there is a difference between outputting something you want and writing something you want