**Command Line & Git Tutorials**

**I. Command Line Tutorial**

The command line is a text interface for your computer. It's a program that takes in commands, which it passes on to the computer's operating system to run.

From the command line, you can navigate through files and folders on your computer, just as you would with Finder on Mac OS or Windows Explorer on Windows. The difference is that the command line is fully text-based.

The advantage of using the command line is its power. You can run programs, write scripts to automate common tasks, and combine simple commands to handle difficult tasks - making it an important programming tool.

The tutorial below is for is for unix-based systems such as Linux and Mac OS X. For Windows users, you will need to Google equivalent commands. You can do most of this by going to Programs – Anaconda – Anaconda Prompt.

**Opening your terminal**

On a Mac, go to Applications – Utilities – Terminal.

Once it’s open, you may want to pin it to your home screen toolbar since it’s so useful.

**Important Commands**

**1. pwd, as in “print working directory”**

Type pwd and then enter into your terminal. This should list your current working directory. Mine says “/Users/shareshianl” and yours probably says something similar. If you ever want to know which folder (aka directory) that you are currently in, you can type pwd.

**2. ls (lowercase L, lowercase S, as in “list”)**

Type ls and then enter. This lists all of the files in your current directory.

If you type “ls –a” this will list all of the files in your directory, including hidden ones. (If you scroll through, you’ll probably see files that start with a period. These are hidden files that you wouldn’t see from typing “ls” alone.

**3. cd (change directory)**

“cd” changes directories. You probably have a directory called Documents in your current working directory. Let’s move into it. To do so, type “cd Documents”

**4. mkdir (make directory)**

“mkdir” will make a new directory (aka folder) with the name that you specify. Now that we are inside Documents, let’s make a new folder called “terminal\_practice” (Note: You don’t want folder names to contain spaces so we’ll use the underscore instead.”) Type “mkdir terminal\_practice”

Type “ls” again and now you should see a new folder called terminal\_practice located there.

Just because you’ve make the directory doesn’t mean that you are inside it. Type “cd terminal\_practice” to now move inside that folder.

Type “ls” again and you’ll see that this is currently an empty directory.

**5. cd ..**

Let’s say you wanted to navigated back up a directory into Documents. Type “cd ..” and then enter.

Now type “pwd” and you’ll see that you are back in Documents. But actually, we want to be back in terminal\_practice so type cd terminal\_practice once again to get there.

**6. touch**

“touch” makes new files. Let’s make a new file called “helloworld.py” by using touch. Type “touch helloworld.py”

Type “ls” and you should now see the helloworld.py file there.

**6. cp (copy)**

Let’s say we wanted to copy this helloworld.py file into our Documents folder. I’ll type

“cp helloworld.py ~/Documents” (Note: The format will always be “cp filename newlocation”)

**7. rm (remove)**

Let’s navigate up to Documents again by typing “cd ..”. We don’t actually want the helloworld.py file there so let’s remove it. To do so, type “rm helloworld.py”. If you type “ls” again you’ll notice the file is gone.

Note that if you wanted to remove a directory, not just a single file, you’d need to type “rm –r” before the directory name, not just “rm”.

“-r” specifies the recursive option, which deletes everything within the directory.

8. **Up arrow**

Press the up arrow several times. Note that it lists the past commands that you’ve used. You can always navigate up to repeat a command instead of typing it all in.

**9. Tab key**

Let’s say we want to navigate back to the helloworld.py file located inside terminal\_practice. Instead of typing the whole file name, just type “cd term” and then **press the tab key**. You should now see the entire command listed, and you can now press enter.

**II. VIM and other text editors**

Type “ls” to see the helloworld.py file. This file is currently empty – let’s add stuff to it! To add stuff to it, we’ll need to use a text editor. There are many options – VIM, emacs, vi, Atom, Sublime, etc. VIM is one of the ones that comes pre-installed on a Mac so let’s use that.

**1. Opening vim**

Type “vim helloworld.py”

**2. Insert mode**

We need to be in the insert mode in order to edit the file. Type “i” to do so.

**3. Edit the file**

We can now edit the file. Let’s type the following. Note: Don’t use tab - use four spaces to indent.

print(‘Hello, World!’)

for i in range(10):

print(i)

**4. Save the file**

To save the changes, press escape and then type “ :w”

**5. Exit vim**

To exit vim, type “:q”

**6. Running Python in your terminal**

Okay, we don’t need Jupyter Notebooks or Anaconda or PyCharm or anything to run a Python program – we can do it directly from our terminal!

Type “python helloworld.py” and press enter

If you wrote your program correctly, you should see its output on the screen!

**Opening Jupyter Notebooks (or any program!) through your terminal**

We can now open Jupyter Notebooks through your terminal. Type “jupyter notebook” and press enter and give it a few seconds to boot up. (Note: if you still haven’t installed Anaconda on your computer, this won’t work.)

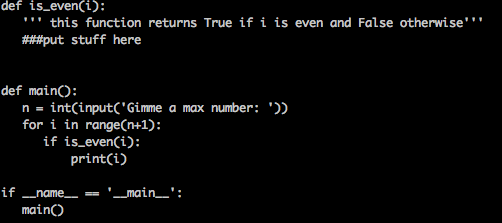
To quit it, type Control-C.

**Terminal Exercises:**

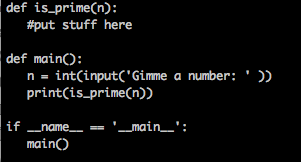
1. Remove the “terminal\_practice” directory using the above techniques.

2. Create a new directory within Documents called terminal.

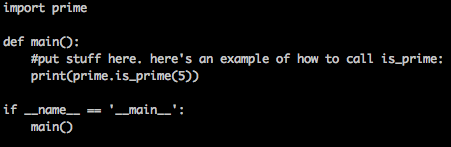
3. Use vim to create a new file within that directory called evens.py that contains an even function. Your program should have the same format as the screenshot below. Use it to print all of the even numbers between 0 and 10 from the command line.



4. Use vim to create a program called prime.py that asks the user for a number and returns True if the number is prime and False if it isn’t. Create a function called is\_prime within this file that helps you, and put the rest within the main function. Then run this program from the command line. Your file should look like this:



5. Use vim to create a program called allprimes.py that asks the user for a number and prints all of the prime numbers between 2 and that number. Use the is\_prime function that you already wrote to help you. Here’s how you’ll import it into allprimes.py:



**III. SOME EXTENSIONS (THIS PART IS OPTIONAL – SKIP THIS UNTIL THE VERY END)**

**1. Sublime Text**

Some of the best coders are so comfortable using the command line that they write ALL of their programs in VIM through the terminal. Crazy, right? If you aren’t that hardcore then I really like Sublime Text. You can open download it here:

<https://www.sublimetext.com/>

You can open it like you would open up any other application through Finder – Applications. You can then create a file as if you were typing into something like Word or Notepad, save it, and then run it from the command line. Just make sure that wherever you saved the Sublime text file is in the same place as your current working directory of your terminal when you go to run your code.

**2. Oh My Zsh**

All of the pros use Oh My Zsh, which makes using your shell (aka terminal) a lot easier to use and prettier. Go to <http://ohmyz.sh/> and follow the instructions for downloading (you can type the curl command directly into your terminal).