Intro

Welcome to CS 61A! I'm happy you're here.   :-)

One reason I love this class is because it's accessible to beginners. However, it does work with some assumptions that can be hard to pick up on, if you've never programmed before. I want to take some time to talk about those assumptions. They're not part of the class, and not covered on the tests. These are just some things I wish someone told me, when I first enrolled.

The Terminal

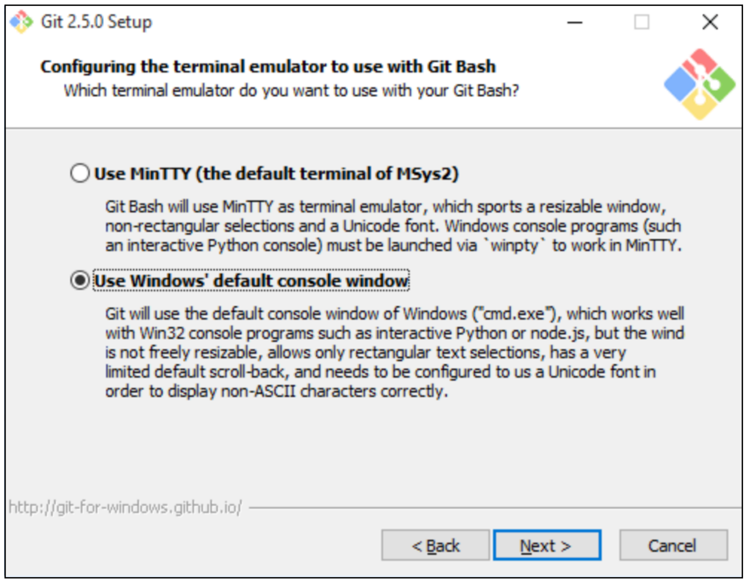
The terminal is an app on your computer. It's maybe the most useful one, because you can use it to tell your computer to do pretty much whatever you want. We'll just go over the basics here.

1. Setup

First, we need to find the terminal app.

On a Mac: Hold ⌘ (command) and press spacebar; this will open up the Spotlight Search. Type "Terminal", and you should see the app icon pop up. Double click it to open.

On Windows: This is a bit tricker, since Microsoft is a rebel. You'll need to install Git Bash, from https://git-scm.com/downloads. During installation, be sure to select the option "Use Windows' default console window". Here is a screenshot, for reference:



Git 2.5.0 Setup

Configuring the terminal emulator to use with Git Bash
Which terminal emulator do you want to use with your Git Bash?

○ **Use MinTTY (the default terminal of MSys2)**

Git Bash will use MinTTY as terminal emulator, which sports a resizable window, non-rectangular selections and a Unicode font. Windows console programs (such an interactive Python console) must be launched via `winpty` to work in MinTTY.

[✓ **Use Windows' default console window**]

Git will use the default console window of Windows ("cmd.exe"), which works well with Win32 console programs such as interactive Python or node.js, but the wind is not freely resizable, allows only rectangular text selections, has a very limited default scroll-back, and needs to be configured to us a Unicode font in order to display non-ASCII characters correctly.

http://git-for-windows.github.io/

< Back   Next >   Cancel

Now that you're set up with the terminal, make sure you can readily access it. For Mac users, I recommend you right click the icon in your home bar, and select "Options > Keep in Dock". You will be using this app *a lot* in this class, and even more if you are pursuing CS as a major.

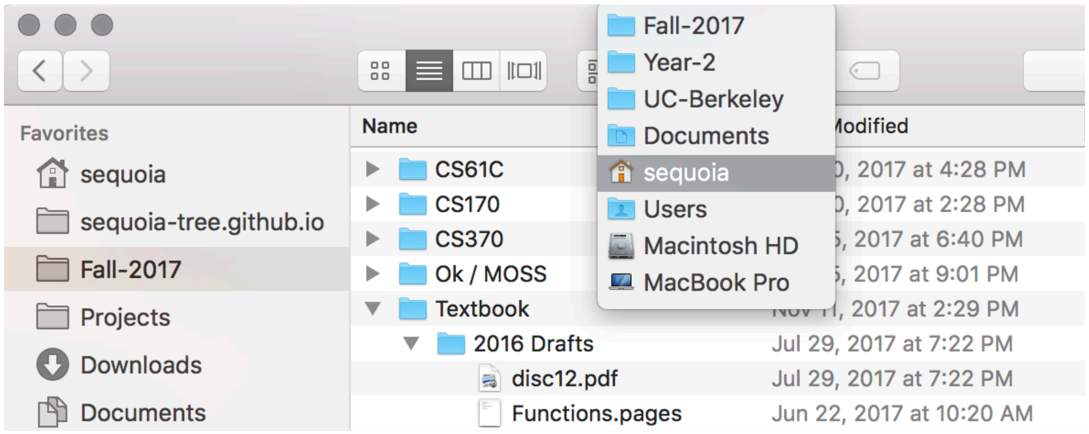
2. Computer Directories

When you open the terminal, you probably see something like this:

```
MacBook-Pro:~ sequoia$ █
```

This is what's called your *bash prompt*. It's where you type text into the terminal. You don't have to know any of these terms for class, but they will be useful to know now and then. Notice the squiggly that appears right after the colon. This indicates what *directory* you're in. ("Directory" is just another word for "folder".) In particular, the squiggly always denotes your *home directory*. This is the main folder on your computer, and your whole computer profile is stored inside it. In fact, we can go find your home directory using Finder or the Windows start menu.

On a Mac: You have probably never encountered your home directory in Finder. Since most people never need to access it, Apple sort of hid it from you. The good news is, it's easy to find! In the side bar, you should see some folders like "Documents", "Downloads", etc. Click on one of those. Then, right click the icon that appears at the top of the Finder window. You'll see a little menu appear, as in this screenshot:



Click the next folder down, with the picture of the house. This will take you to your home directory in Finder. This is the same folder you just opened in the terminal a minute ago! After entering your home directory, left-click the house icon at the top of your Finder window, and drag it over to the menu at the left of your Finder window. If you drop it there, you can always access your home directory in a snap. In the screenshot above, I put my home directory right above the folder called "sequoia-tree.github.io".

On Windows: From the Windows start menu, you should be automatically see a folder called "My Computer", or in newer versions, "This PC". That's actually your home directory.

To recap: When you open up the terminal, you are in your home directory. This is the main directory in your computer, which contains everything in your computer profile. In particular, your home directory includes the folders called "Documents", "Downloads", etc.

Now that we know what the home directory is, we are almost ready to navigate through the computer using only the terminal. The next step is to learn two really helpful commands. The first one is `ls`, short for "list". Here's what I see when I type it into my bash prompt:

```
MacBook-Pro:~ sequoia$ ls
Applications/         Dropbox/              Pictures/
Boostnote/            Languages/            Public/
Desktop/              Library/              Pyagram/
Documents/            Movies/               anaconda/
Downloads/            Music/                sequoia-tree.github.io/
```

These are all the contents of my home directory. You can see my Documents folder, my Downloads, Applications, and some other miscellaneous items. `ls` just lists the contents of whatever directory I'm in.

The last piece of the puzzle is moving between directories. How do we go from home to Documents, for instance? This is where the `cd` command comes in. It's short for "change directory". Here's an example:

```
MacBook-Pro:~ sequoia$ cd Documents
MacBook-Pro:~/Documents sequoia$ █
```

Notice that after the colon, I now see `~/Documents`. That's the computer's way of saying I'm in Documents, which is in my home directory. Whatever folder you end up in, your computer will always display the *path* from home. You can always get home by running `cd ~` or `cd ~/`. The slash is used to separate the names of different folders in your path, so it usually doesn't matter whether there's a slash the end of the line. Let's try out our cool new skills:

```
MacBook-Pro:~ sequoia$ cd Documents
MacBook-Pro:~/Documents sequoia$ ls
(misc)/                       UC-Berkeley/
Barstow-College/              devious.py

MacBook-Pro:~/Documents sequoia$ cd UC-Berkeley
MacBook-Pro:~/Documents/UC Berkeley sequoia$ █
```

We say the *working directory* is whatever folder you are currently in. In this example, my working directory was initially my home directory. Next it was my Documents folder, and after that it was the folder called "UC-Berkeley".

Sometimes you may also want to go "back", for example from Documents to your home directory. You can do that like so:

```
MacBook-Pro:~/Documents sequoia$ cd ..
MacBook-Pro:~ sequoia$ █
```

Again, you don't need to know these terms for class. Don't worry about them appearing on a test, or anything like that. These are just some useful things you should know before the semester gets too far along. Here's a quick cheat sheet for the terms we learned:

bash prompt	Where you type text into the terminal.
directory	A folder.
home directory	The main folder on your computer, denoted by ~ or `~/`.
working directory	The folder you're currently in, while using the terminal.
path	The long way to write what folder you're in. For example `~/Documents/UC-Berkeley/Year-2/Summer-2017`

3. Your 61A Folder

In this class there are going to be lots of files to keep track of, so it will be wise to make a folder for all your assignments. Since I obsessively nest folders inside other folders, I keep all my 61A stuff in `~/Documents/UC-Berkeley/Year-1/Fall-2016/CS61A`. I also have subfolders for Homework, Lab, and Projects, but that's just my preference. Take a moment now to make your own CS 61A folder wherever you think is convenient. (I notice most folks like to keep in on the Desktop, so that they don't even have to open Finder to access it.)

Over the course of the class, you will download your assignments from the CS 61A website. Make sure to keep them all in the folder you just set up, so that you can find all your work in one organized location on your computer. I have worked with a lot of students who did not heed this advice, and they ended up with multiple copies of a single homework floating around, without knowing which was the one they submitted. Please keep organized, so you don't fall into that trap.

After you set up the directory in Finder, try navigating to it using the `cd` command in the terminal. You might realize it's a bit of a hassle, and certainly not something you want to be doing several times a day, let alone several times a week. There are two good solutions.

Continue to the next page.

Shortcut 1: Drag and Drop

Try opening up a new terminal window. *Without pressing enter*, type "`cd `" into the bash prompt. Include the space at the end. Also navigate to your new CS 61A folder using Finder. Drag and drop it onto the terminal window. You'll notice the file path automatically shows up. Press enter to run the command and `cd` into your CS 61A directory.

Shortcut 2: Aliasing

This second method is a bit more technical, but maybe also cooler. It's totally up to you whether you want to do it or not. Also note I have only tried this on a Mac and I'm sure the process is a bit different for Windows computers. First go to your home directory in the terminal. We're going to access an "invisible" file called `.bash_profile`, which gets run every time you open up the terminal. Invisible files are just files that Apple hid from you because most people don't need to access them.

```
MacBook-Pro:~ sequoia$ open .bash_profile
```

This should open `.bash_profile` in a text editor like TextEdit. Now, on any blank line, go ahead and set up what's called an *alias*. Here's an example from my `.bash_profile`:

```
alias ucb='cd ~/Documents/UC-Berkeley/Year-1/Fall-2016'
```

Notice there are no spaces around the equals sign. Also, the quotes are *non-formatted* single quotes. You can copy and paste them to after typing them into the terminal. Using the example above, just replace "`ucb`" with whatever you want your custom command to be, as long as it is only one word, and replace the path with wherever you put your CS 61A folder. Then save your changes to `.bash_profile` and restart the terminal app. Just like that, you made your own custom terminal command! Here's an example using the one I defined above:

```
MacBook-Pro:~ sequoia$ ucb
MacBook-Pro:~/Documents/UC-Berkeley/Year-1/Fall-2016$ █
```

4. Tips & Tricks

Often, you'll want to run a command in the terminal that you recently typed before. You can save time by cycling through your previous commands using the up-arrow key.

Python

Ok, I admit learning about the terminal is pretty dry stuff. Now let's get set up with the *much cooler* aspect of the class: Python. This is a computer programming language named after the species of tropical snake that invented it. Let's install it on your computer.

On a Mac: Download the Python installer at https://www.python.org/downloads
On Windows: Download the Python installer at https://www.python.org/downloads/windows Make sure to check the box labelled "Add Python 3.6 to PATH".
On Ubuntu: From the terminal, run this command: `sudo apt-get install python3`

After you run the installer, you should be all set up! Let's test it out. In the terminal, run this command:

```
python3
```

If you're using Windows, the command might simply be `python`. You should see a Python interpreter appear, which looks something like this:

```
Python 3.6.1 (v3.6.1:69c0db5050, Mar 21 2017, 01:21:04)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> █
```

Verify that at the beginning of the first line, you see version `3.5.1` or later. If you're on a Mac, running just `python` may open an interpreter for version `2.7.10`, so make sure you always use the command `python3` or `python3.6`.

In this chapter we won't talk about all the awesome stuff you can do in Python. For now, just try this in the interpreter to make sure everything is working:

```
>>> import this
```

You should see a little easter-egg put in by the creators of Python. It likely won't make much sense right now, but maybe by the end of the class you'll understand what it's all about.

Continue to the next page.

Picking a Text Editor

The very last thing for today will be picking a text editor. Code is sort of like an essay, but the kind of essay your computer can understand. Technically you could write all your code in the terminal, but that would be like writing an essay in pen on your first draft: hard to edit, and pretty frustrating. Furthermore, we can't use the text editors you're familiar with (Pages, Microsoft Word, etc.) because they store formatted text instead of unformatted text. So, what do we use to write code?

There are a few editors to choose from. You'll probably hear about Atom and Sublime Text. These are both great for CS 61A. They aren't that different so pick whichever you prefer. My personal recommendation is Atom. You can download them from the following links:

https://atom.io
https://www.sublimetext.com/3

Using either of these editors, it should be pretty straightforward to make a new Python file. This will be a document ending with the extension `".py"`. We've already covered a lot today, so you don't have to do this now. For future reference though, know that you can execute any Python file from the terminal like so:

```
MacBook-Pro:~/Documents sequoia$ python3 my_file.py
```

In order for this to work, you have to be in the same directory as the file you're trying to run. For the example above, `my_file.py` would have to be in my Documents folder.

This last remark is very important: Make sure you always save the file you're writing, before you try running it. This is just like how you want to save an essay you're writing, before you try printing it.

Continue to next page.

Closing Remark

CS 61A can be a super fun class. It's certainly one of my favorites so far. But it can also be challenging at times. There are a lot of resources available to you, so please don't hesitate to use them. On the course website you can access years of practice and course content, and always feel free to email any of the tutors or TAs for extra help. Speaking of tutors and TAs, we get lonely when people don't sign up for tutoring or come to office hours. Drop by if you could use a few pointers!

That said, I strongly believe CS shouldn't just be confined to the classroom. While maybe you're not yet ready to start your own project, there are loads of neat puzzles online that you can solve using code, to help you hone your skills. When I was in CS 61A, I was a fan of Project Euler, which you can access at https://projecteuler.net/archives. We'll also cover some cool stuff in the coming sections of this book — ranging from generating the Fibonacci sequence to building cellular automata. Using the CS skills you develop in the next week or so, you can solve puzzles and problems in just about any field that interests you.