

# Introduction to the Unix Command Line

ASTR 211, Spring 2021

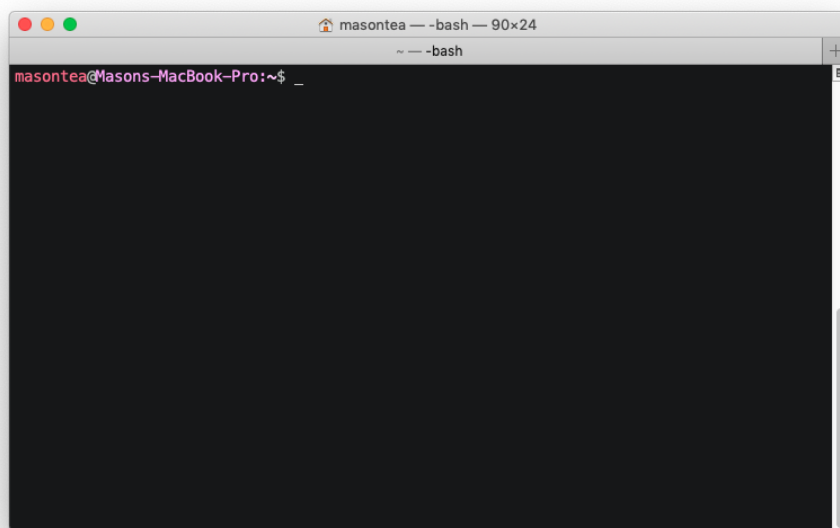
*Mason V. Tea, 1/25/2021*

In this tutorial, we're going to walk through the basics of using the command line on a Unix operating system (like Mac OSX or Linux). We'll be focusing on OSX, but these commands should be valid in Linux frameworks. This won't go into too much detail, and should serve as just enough to get you using the command line for our purposes in the course.

## The terminal

The **terminal** serves as an interface between the computer and the user, where the user can enter text-based commands to perform tasks like moving around the file system, creating files, running programs, so on and so forth.

To open a new terminal window, simply search for "Terminal" in the Spotlight Search bar (top right corner on a Mac). I'd highly recommend dragging-and-dropping this into your app bar for quick access. When you open up the terminal, this is roughly what you'll see:

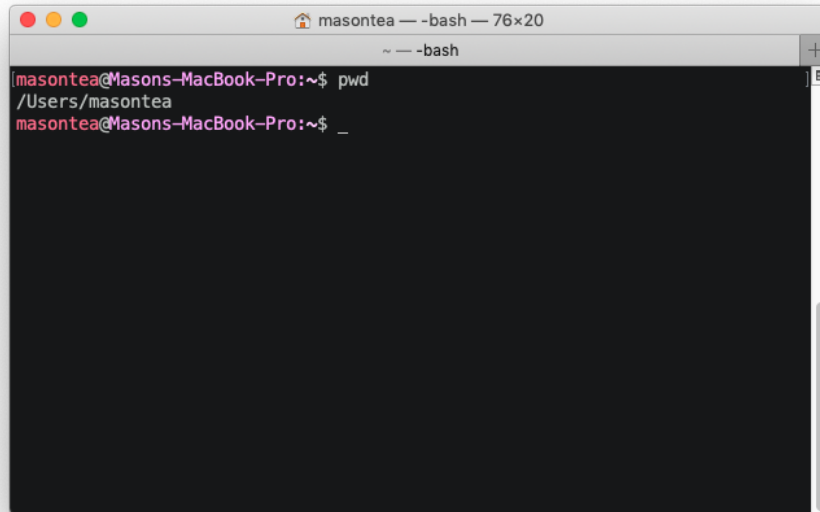


You can see my username and the name of my Macbook with an "@" wedged between them, followed by an odd string of symbols. Entering commands is simple: type your command after the dollar sign and hit enter.

Once you've opened a fresh terminal window, you'll find yourself in your computer's *home directory*, which you can think of as the "root" containing all of your other directories. This will usually be linked to your user account on the computer, so your home directory will really be your *user directory*. (This exists in the root directory of the computer itself, but you shouldn't need to mess with that.)

## Present working directory (pwd)

Let's say we're not sure if we're in the home directory or not — we can read out the **present working directory** by entering the command `pwd`:

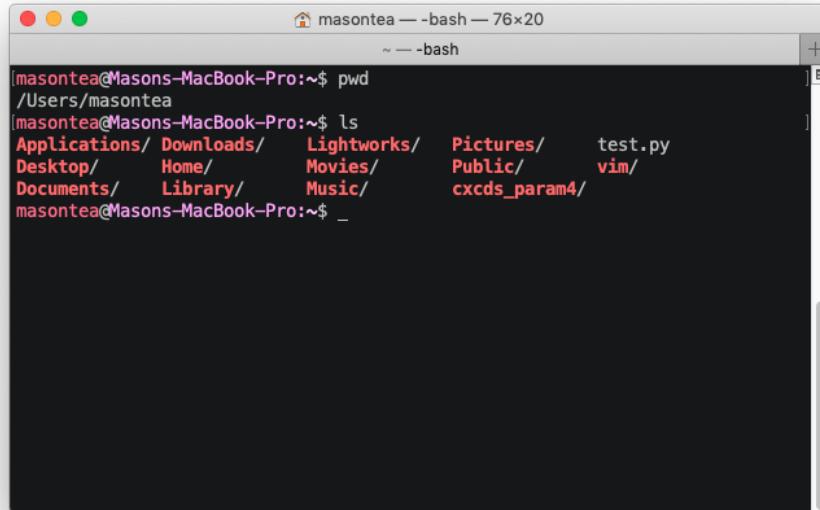
A terminal window titled "masontea -- -bash -- 76x20" with a dark background. The prompt is "masontea@Masons-MacBook-Pro:~\$". The user enters "pwd" and the terminal outputs "/Users/masontea". The prompt then changes to "masontea@Masons-MacBook-Pro:~\$ \_".

```
masontea@Masons-MacBook-Pro:~$ pwd
/Users/masontea
masontea@Masons-MacBook-Pro:~$ _
```

As you can see, I'm in my personal computer's user directory, `masontea`. However, why does it look like our present working directory has two names? This is because the `pwd` command doesn't just read out the name of the directory, but it's **path**. The path of a file is like an address tracing it back to a given directory. So, you can see that we're not only in my personal directory, `masontea`, but also in the `Users` directory that contains it.

## Listing contents (ls)

Now that we know where we are, let's see what's in here. To **list** all the files in the current directory, we use the command `ls`:

A terminal window titled "masontea -- -bash -- 76x20" showing the execution of the 'ls' command. The prompt is "masontea@Masons-MacBook-Pro:~\$". The first command is "pwd", which outputs "/Users/masontea". The second command is "ls", which outputs a list of files and directories: "Applications/", "Downloads/", "Lightworks/", "Pictures/", "test.py", "Desktop/", "Home/", "Movies/", "Public/", "vim/", "Documents/", "Library/", "Music/", and "cxcds\_param4/". The prompt returns to "masontea@Masons-MacBook-Pro:~\$".

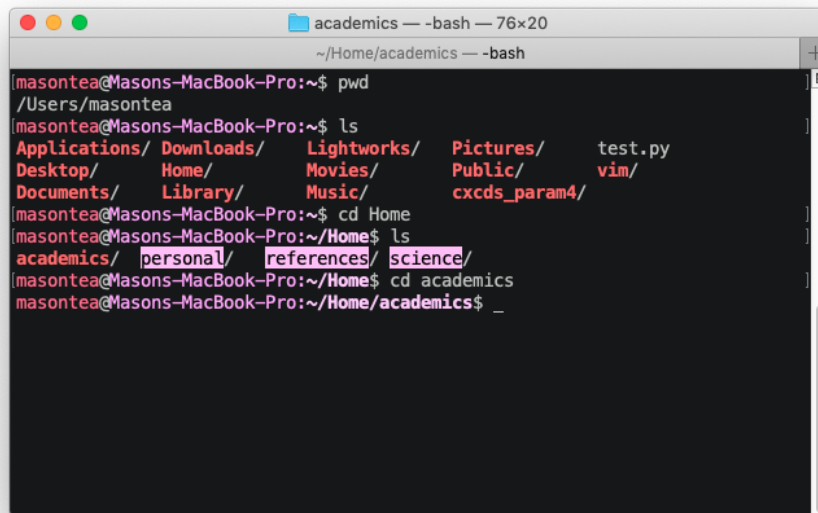
```
masontea@Masons-MacBook-Pro:~$ pwd
/Users/masontea
masontea@Masons-MacBook-Pro:~$ ls
Applications/ Downloads/ Lightworks/ Pictures/ test.py
Desktop/      Home/       Movies/    Public/   vim/
Documents/   Library/   Music/     cxcds_param4/
masontea@Masons-MacBook-Pro:~$ _
```

We can see that through my user directory, I have access to things like my `Downloads` and `Pictures` folders that you may be familiar with, as well as the `Desktop` directory which may come as a surprise (you can store files on it, so yes, it's a directory). There are also various folders that I've put there myself, so don't worry if you don't see some of them on your own computer.

Notice that most of these names have a forward slash after them. This means that they are directories, rather than files. The only non-directory file in my own user directory is `test.py`, which has no slash.

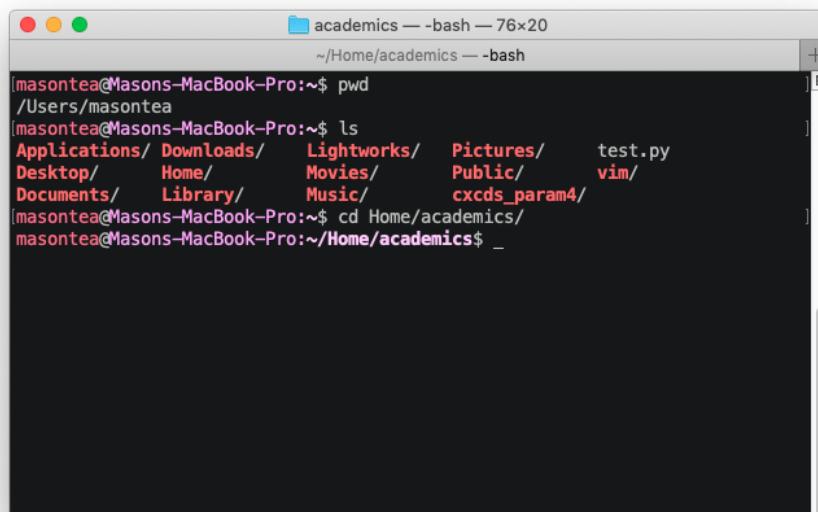
## Changing directories (cd)

Now, for a course like this that will require you to keep track of quite a few files, it's good to create a directory for the class. The first step in making a new directory is moving to the directory where you want it to live. You may just want to do this in your home directory, but you definitely don't have to. For example, I have a directory I've set aside to hold all my school and research files called `Home`, and another directory inside it called `academics` for coursework specifically. Thus, I'm going to **move to that directory** before I make one for this class. I can do this with the `cd` command, along with the name of the directory I want to move to. I can do this multiple times until I arrive at the directory I want:



```
masontea@Masons-MacBook-Pro:~$ pwd
/Users/masontea
masontea@Masons-MacBook-Pro:~$ ls
Applications/ Downloads/ Lightworks/ Pictures/ test.py
Desktop/      Home/         Movies/      Public/      vim/
Documents/   Library/     Music/       cxcds_param4/
masontea@Masons-MacBook-Pro:~$ cd Home
masontea@Masons-MacBook-Pro:~/Home$ ls
academics/  personal/  references/ science/
masontea@Masons-MacBook-Pro:~/Home$ cd academics
masontea@Masons-MacBook-Pro:~/Home/academics$ _
```

Notice that I used the `ls` command in between `cd` commands to make sure the directory I was looking for was there. If you know exactly what you're looking for, however, you can type in the full path from the current directory:

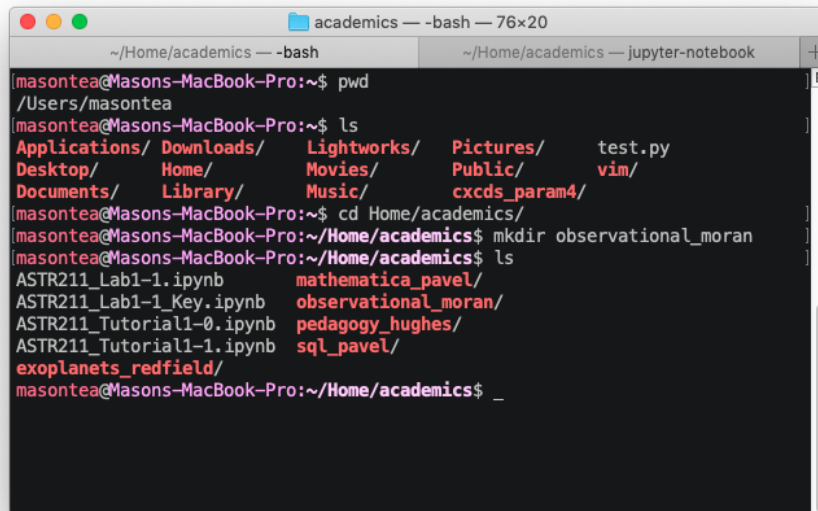


```
masontea@Masons-MacBook-Pro:~$ pwd
/Users/masontea
masontea@Masons-MacBook-Pro:~$ ls
Applications/ Downloads/ Lightworks/ Pictures/ test.py
Desktop/      Home/         Movies/      Public/      vim/
Documents/   Library/     Music/       cxcds_param4/
masontea@Masons-MacBook-Pro:~$ cd Home/academics/
masontea@Masons-MacBook-Pro:~/Home/academics$ _
```

Once you've navigated to a directory, you can see that its path from the home directory is displayed in the address of the current command (in light pink, here).

## Making directories (mkdir)

Now that we're here, we can make our directory for the course. To **make a directory**, we use the `mkdir` command along with the name of the directory we wish to create. I'll call mine `observational_moran`:

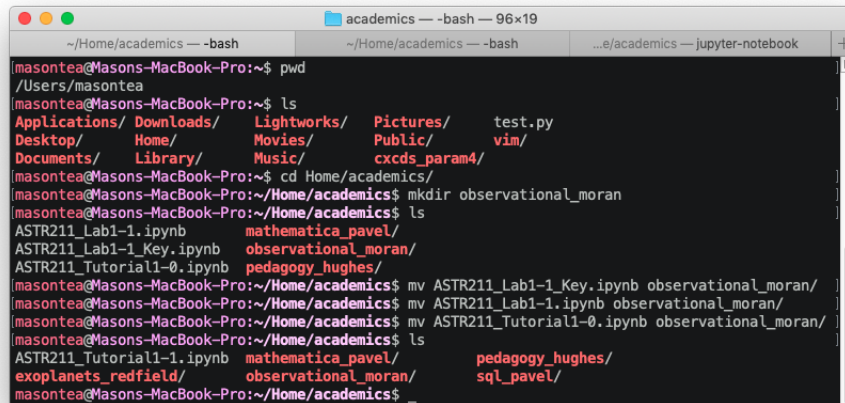
A terminal window titled 'academics — -bash — 76x20' is shown. The user 'masontea' is at the prompt. The terminal shows the following sequence of commands and outputs:

```
masontea@Masons-MacBook-Pro:~$ pwd
/Users/masontea
masontea@Masons-MacBook-Pro:~$ ls
Applications/ Downloads/ Lightworks/ Pictures/ test.py
Desktop/      Home/         Movies/      Public/     vim/
Documents/   Library/     Music/       cxcds_param4/
masontea@Masons-MacBook-Pro:~$ cd Home/academics/
masontea@Masons-MacBook-Pro:~/Home/academics$ mkdir observational_moran
masontea@Masons-MacBook-Pro:~/Home/academics$ ls
ASTR211_Lab1-1.ipynb      mathematica_pavel/
ASTR211_Lab1-1_Key.ipynb  observational_moran/
ASTR211_Tutorial1-0.ipynb pedagogy_hughes/
ASTR211_Tutorial1-1.ipynb sql_pavel/
exoplanets_redfield/
masontea@Masons-MacBook-Pro:~/Home/academics$ _
```

Using the `ls` command, we can see that, after using the `mkdir` command, we've added a new directory called `observational_moran` to `academics`.

## Moving files (mv)

You can see a few other directories I have from last semester's classes, as well as some Jupyter Notebook files for this class that have been floating here without a folder. Let's clean those loose files up a bit by adding them to our new class directory. There are a few ways to do this. First is just by **moving** them with the `mv` command, followed by the name of the file you wish to move and the directory you want to move it to:

A terminal window titled 'academics -- -bash -- 96x19' showing a series of commands and their outputs. The user starts in the directory ~/Home/academics. They run 'ls' to list files, then 'cd Home/academics/' to change to the current directory. They run 'mkdir observational\_moran' to create a new directory. Then they run 'ls' again to see the new directory. Finally, they run three 'mv' commands to move files from the root directory into the 'observational\_moran' subdirectory: 'mv ASTR211\_Lab1-1\_Key.ipynb observational\_moran/', 'mv ASTR211\_Lab1-1.ipynb observational\_moran/', and 'mv ASTR211\_Tutorial1-0.ipynb observational\_moran/'. The terminal shows the directory listing before and after each move, confirming the files are now in the subdirectory.

```
masontea@Masons-MacBook-Pro:~$ pwd
/Users/masontea
masontea@Masons-MacBook-Pro:~$ ls
Applications/ Downloads/ Lightworks/ Pictures/ test.py
Desktop/      Home/         Movies/      Public/     vim/
Documents/   Library/     Music/       cxcds_param4/
masontea@Masons-MacBook-Pro:~$ cd Home/academics/
masontea@Masons-MacBook-Pro:~/Home/academics$ mkdir observational_moran
masontea@Masons-MacBook-Pro:~/Home/academics$ ls
ASTR211_Lab1-1.ipynb      mathematica_pavel/
ASTR211_Lab1-1_Key.ipynb observational_moran/
ASTR211_Tutorial1-0.ipynb pedagogy_hughes/
masontea@Masons-MacBook-Pro:~/Home/academics$ mv ASTR211_Lab1-1_Key.ipynb observational_moran/
masontea@Masons-MacBook-Pro:~/Home/academics$ mv ASTR211_Lab1-1.ipynb observational_moran/
masontea@Masons-MacBook-Pro:~/Home/academics$ mv ASTR211_Tutorial1-0.ipynb observational_moran/
masontea@Masons-MacBook-Pro:~/Home/academics$ ls
ASTR211_Tutorial1-1.ipynb  mathematica_pavel/      pedagogy_hughes/
exoplanets_redfield/      observational_moran/    sql_pavel/
masontea@Masons-MacBook-Pro:~/Home/academics$ _
```

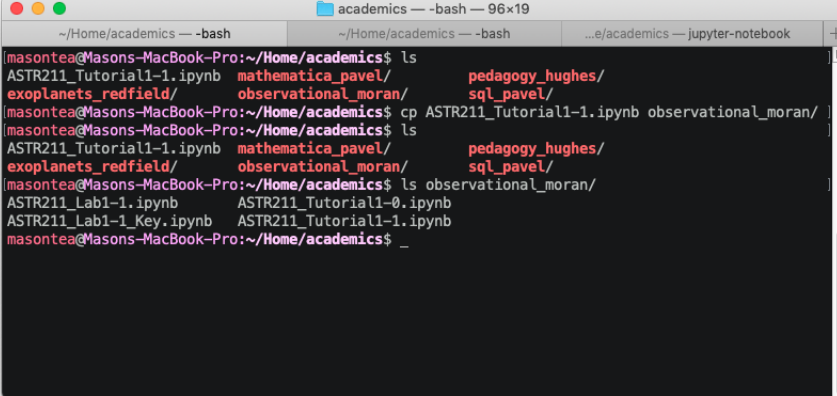
Checking the contents of `academics`, we can see that the three files I used the `mv` command on are nowhere to be found. We can check the contents of `observational_moran` without leaving the directory by using the `ls` command followed by the directory we wish to peer into and, lo and behold, the files we moved are in there.

## Clearing the terminal (clear)

Before we go any further, the terminal window is getting a bit messy with all the commands we've used so far. We can get a fresh window by using the `clear` command. Don't worry — using `clear` only pushes all the old commands up, so you can scroll back up to them later if you want, and you'll stay in the same working directory.

## Copying files (cp)

Now, let's say we want to keep the last file in `academics`, but we want a **copy** of it in the `observational_moran` directory as well. We can accomplish this using the `cp` command, which works the same way as `mv` while copying instead of moving the files:

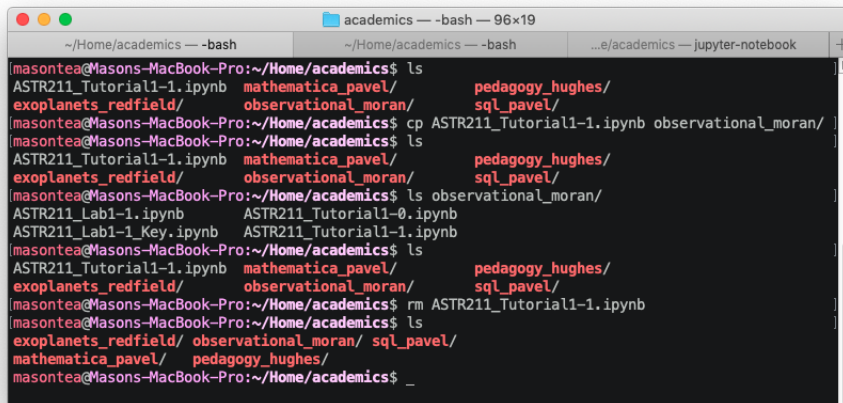


```
masontea@Masons-MacBook-Pro:~/Home/academics$ ls
ASTR211_Tutorial1-1.ipynb  mathematica_pavel/  pedagogy_hughes/
exoplanets_redfield/    observational_moran/  sql_pavel/
masontea@Masons-MacBook-Pro:~/Home/academics$ cp ASTR211_Tutorial1-1.ipynb observational_moran/
masontea@Masons-MacBook-Pro:~/Home/academics$ ls
ASTR211_Tutorial1-1.ipynb  mathematica_pavel/  pedagogy_hughes/
exoplanets_redfield/    observational_moran/  sql_pavel/
masontea@Masons-MacBook-Pro:~/Home/academics$ ls observational_moran/
ASTR211_Lab1-1.ipynb      ASTR211_Tutorial1-0.ipynb
ASTR211_Lab1-1_Key.ipynb  ASTR211_Tutorial1-1.ipynb
masontea@Masons-MacBook-Pro:~/Home/academics$ _
```

Checking the contents of both folders, we can see that the file is in both.

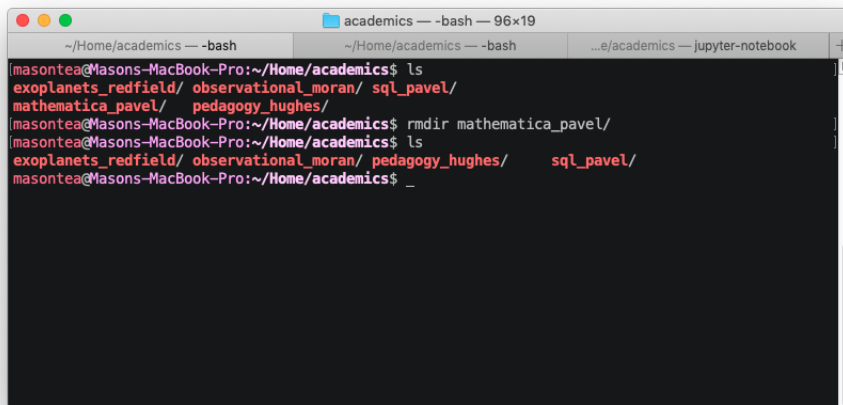
## Removing files (rm) and directories (rmdir)

Now, I don't actually want two copies of this file, so I'm going to **remove** the one still in `academics` with the `rm` command, like so:

A terminal window titled 'academics -- bash -- 96x19' showing a series of commands and their outputs. The user is in the directory ~/Home/academics. The commands and outputs are: 1. 'ls' showing a list of files and directories including 'ASTR211\_Tutorial1-1.ipynb', 'mathematica\_pavel/', 'pedagogy\_hughes/', 'exoplanets\_redfield/', 'observational\_moran/', and 'sql\_pavel/'. 2. 'cp ASTR211\_Tutorial1-1.ipynb observational\_moran/' copying the file to the 'observational\_moran' directory. 3. 'ls' showing the file is now in 'observational\_moran/'. 4. 'ls observational\_moran/' showing the file 'ASTR211\_Tutorial1-1.ipynb'. 5. 'rm ASTR211\_Tutorial1-1.ipynb' removing the file from the current directory. 6. 'ls' showing the file is no longer in the current directory. The prompt ends with a tilde '~'.

**BE VERY CAREFUL WITH THIS COMMAND!** Removing a file gives *no warning* and *completely removes the file from the computer* rather than putting it in the trash. It'll be gone forever, so make sure you want to delete a file before you use `rm`!

Removing directories works the same way, but with a different command, because deleting entire directories by accident is an incredibly scary prospect. The `rm` command doesn't work on directories, and rather requires the `rmdir` command. For example, I created these folders in `academics` before I finalized my schedule last semester, and didn't actually end up taking the Mathematica class. So, let's go ahead and get rid of that folder:

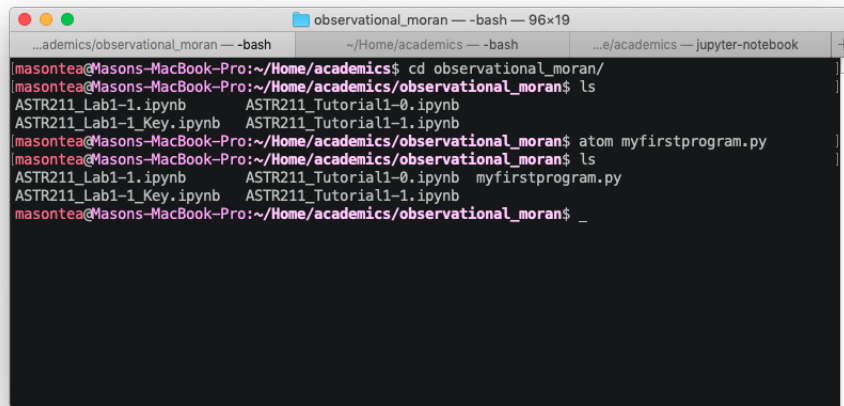
A terminal window titled 'academics -- bash -- 96x19' showing a series of commands and their outputs. The user is in the directory ~/Home/academics. The commands and outputs are: 1. 'ls' showing a list of files and directories including 'exoplanets\_redfield/', 'observational\_moran/', 'sql\_pavel/', 'mathematica\_pavel/', and 'pedagogy\_hughes/'. 2. 'rmdir mathematica\_pavel/' removing the directory. 3. 'ls' showing the directory is no longer in the current directory. The prompt ends with a tilde '~'.

Boom, the directory is gone. Again, there was *no warning* and the directory, along with all of its contents, was *permanently deleted* by doing this. So, **BE DOUBLE CAREFUL WITH THIS COMMAND!!**



## Creating (text) files

Let's go ahead and enter our class directory and create your first (my fifth) file here. Most of the files you create by hand will be text files containing code, so you'll use your favorite text editor to do this. Mine is Atom, so if I wanted to make a new file, I would type atom followed by the name of the file I want to create. In this case, let's make a python file called `myfirstprogram.py`:

A terminal window titled 'observational\_moran' showing a series of commands and their outputs. The user navigates to the 'observational\_moran' directory and lists files. Then, they use the 'atom' command to create a file named 'myfirstprogram.py'. Finally, they list the directory again, showing the new file has been added.

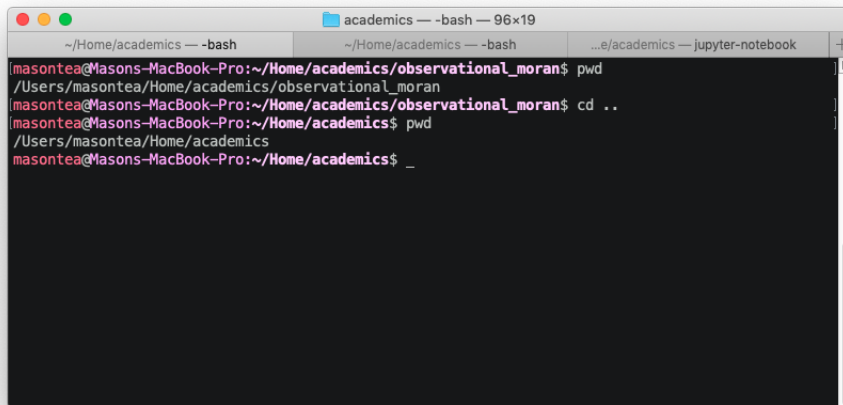
```
masontea@Masons-MacBook-Pro:~/Home/academics$ cd observational_moran/
masontea@Masons-MacBook-Pro:~/Home/academics/observational_moran$ ls
ASTR211_Lab1-1.ipynb      ASTR211_Tutorial1-0.ipynb
ASTR211_Lab1-1_Key.ipynb ASTR211_Tutorial1-1.ipynb
masontea@Masons-MacBook-Pro:~/Home/academics/observational_moran$ atom myfirstprogram.py
masontea@Masons-MacBook-Pro:~/Home/academics/observational_moran$ ls
ASTR211_Lab1-1.ipynb      ASTR211_Tutorial1-0.ipynb  myfirstprogram.py
ASTR211_Lab1-1_Key.ipynb  ASTR211_Tutorial1-1.ipynb
masontea@Masons-MacBook-Pro:~/Home/academics/observational_moran$ _
```

After saving your file, it will show up in the folder. If you use a different text editor, you need only replace `atom` with the name of yours. If we want to **open** a file in the current directory, we use the `open` command followed by the file name. It will open with your default text editor if its a text file, default PDF viewer if its a PDF, et cetera.

Sidenote: You can just open your text editor by typing its name into the command line. You can do this with most programs, just like your text editor.

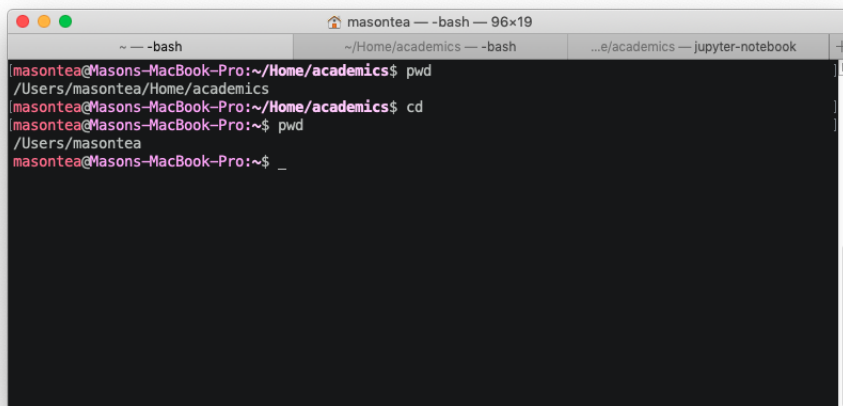
## Moving backwards through directories (more with cd)

So far, we've only moved "forward," so to speak, through directories. If we want to go back into the previous folder, we use the `cd` command again, this time followed by `..` (two dots), which is basically just Unix speak for "one folder back." So, let's go back to `academics`:



```
masontea@Masons-MacBook-Pro:~/Home/academics/observational_moran$ pwd
/Users/masontea/Home/academics/observational_moran
masontea@Masons-MacBook-Pro:~/Home/academics/observational_moran$ cd ..
masontea@Masons-MacBook-Pro:~/Home/academics$ pwd
/Users/masontea/Home/academics
masontea@Masons-MacBook-Pro:~/Home/academics$ _
```

We're only two folders away from our user directory now, so we could just use `cd ..` two more times to go back. But if we were 26 folders deep in some directory, it would make more sense to use the `cd` command on its own to bring us back, without typing 52 double-dots to escape. Either way, let's use `cd` alone to head back to the user directory:



```
masontea@Masons-MacBook-Pro:~/Home/academics$ pwd
/Users/masontea/Home/academics
masontea@Masons-MacBook-Pro:~/Home/academics$ cd
masontea@Masons-MacBook-Pro:~$ pwd
/Users/masontea
masontea@Masons-MacBook-Pro:~$ _
```

## Motivation

Now that we've gone to all the trouble of moving around the file system with the terminal, I'll point out that all of this can be done with the file explorer as well. Once you get the hang of it, moving around in the terminal ends up being much faster than clicking around in the file explorer, dragging and dropping, et cetera. However, if you're not into it, you can definitely use what you're used to. You can also use both, like me — sometimes it's more convenient to look at files in a window all their own. In that case, I typically navigate with the terminal and then use the `open` command on whatever directory I want to look at, rather than clicking around in the Finder window.

The way you do it starting out doesn't really matter, but in my experience as an astronomer, it's much better to know how to use it than not. From installing programs and packages to downloading data or code, it comes in handy sooner or later. (For example, when we set up all of our python-related tools.) Either way, this should be enough to get anyone started!