

# Version Control

# Agenda

- Finish `flatten` from Monday.
- Quick discussion of `printf`
- Discussion of Version Control
  - What is it, and why use it?
  - Getting Set Up
  - Git & Github from the Command Line
  - Github & Eclipse

# Practice: Flattening a 2D array

For a given **rectangular** (non-jagged) 2D int array **A**, return a new 1D array **B** where **B** has all of the elements from the first row of **A**, then from the second row of **A**, then from the third row of **A**, etc.

# Solution: Flattening a 2D array

```
public static int[] flatten(int[][] A) {
    int numRows = A.length;
    int numCols = A[0].length;

    int[] B = new int[numRows * numCols];

    for (int i = 0; i < numRows; i++) {
        for (int j = 0; j < numCols; j++) {
            B[i * numRows + j] = A[i][j];
        }
    }
}
```

# Formatted Printing

Previously, we'd print out data interspersed with text like this:

```
int age = 19;  
double weight = 198.3839;  
String name = "John Doe";  
System.out.println("Patient " + name + " is " + age + " y/o and weighs " + weight + " lbs.");
```

This is tedious to type and it's easy to make mistakes when typing:

- Forgetting spaces between words
- Missing `+` signs or start/end quotes

# Formatted Printing the Right Way

We can use `System.out.printf()` to do it more succinctly.

```
int age = 19;  
double weight = 198.3839;  
String name = "John Doe";  
System.out.printf("Patient %s is %d y/o and weighs %4.1f lbs.%n", name, age, weight);
```

➔ Patient John Doe is 19 y/o and weighs 198.4 lbs.

# What is `printf`?

`System.out.printf()` takes as input:

- A format String consisting of literal text and *format specifiers*
  - Format specifiers are like slots where the missing data should go, and allow you to dictate how that value is placed in the String
- The remaining arguments whose values are to be placed in the locations specified by the format specifiers.

# Format Specifiers

The syntax for a format specifier is

```
%[flags][width][.precision]conversion-character
```

Common Conversion Characters	Purpose
d	decimal integers ( <code>int</code> )
f	floating point values ( <code>double</code> )
s	<code>String</code>
n	A newline character (a line break)



# Format Specifiers

The flags, width, and precision modifiers can optionally be used to dictate how the arguments get displayed.

Breaking down: `%4.1f`:



- the width of the result is at least `4`, meaning that the number will be padded with space to be at least `4` characters long
- the precision of the value is `1`, meaning that the number will be rounded to one decimal place

## More on `printf()`

This is a method that's likely to be quite useful to you, and there are a lot of parameters to play with.

[This is a great resource from Colorado State that you can explore for more details.](#)

# Version Control Systems

- Version Control Systems are software tools that allow developers to:
  - make incremental changes,
  - save their work at different stages,
  - & work as a team on large projects
- Popular version control systems include:
  - SVN
  - CVS
  -  Git 

# What Does Version Control Let Us Do?

- Get safe and detailed backups of your work
  - Maybe your computer died and you lost your local files
  - Maybe you accidentally introduced a terrible bug in your software without realizing
- Roll back software to previous versions
- Have multiple people contributing changes to the same project

# Introducing Git

Git is an extremely popular VCS. It is not the same thing as GitHub, which is a service for hosting files on the internet using Git.

- Git allows us to maintain different snapshots of all of the code in a **repository** (or "repo" for short).
  - A repository is a folder in which all of our version-controlled files live
- The snapshots of our code are called **commits**.
  - A commit can be thought of as the entire state of our repo at a particular moment in time.

# Gettin' Set Up for Git

If you have an Apple computer, just sit tight for a few minutes.

If you have a computer running Windows, navigate to [cmdr.net](https://cmdr.net). Scroll down about halfway through the page and click "Download Full".

# Gettin' Set Up for GitHub

Git can run entirely locally on your computer, which is still useful for maintaining different versions of your code.

If you want to share your code with a team, or if you want to store your code in a remote location to be more resilient towards computer failure, you can host your repo on GitHub as well.

Go to [github.com](https://github.com) and make an account.

# You're Going to Need Credentials 🥲

Repositories

New

Find a repository...

sharry29 / 21su

cis110 / 21fa

21fa-cit591 / homeworks

sharry29 / TestingClass

21fa-cit591 / recitation4

21fa-cit591 / cit-591-waiver-lhujhu21

21fa-cit591 / cit-591-waiver-maylaiqm

Show more

Recent activity

When you take actions across GitHub,

## All activity

shivinuppal pushed to cis110/21fa yesterday

2 commits to master

57ab957 rec4 fix

361b2f1 project release date fix

callison-burch pushed to artificial-intelligence-class/artificial-intelligence-class.github.io yesterday

3 commits to master

378c16b Update markov-decision-processes.md

6d49ea8 New TA

1 more commit »

YueYANG1996 pushed to

- Your profile
- Your repositories
- Your codespaces
- Your organizations
- Your enterprises
- Your projects
- Your stars
- Your gists
- Upgrade
- Feature preview
- Help
- Settings
- Sign out



**You're Going to Need Credentials** 🥵

Saved replies

Applications

Developer settings

# You're Going to Need Credentials 🥵

Settings / Developer settings

GitHub Apps

OAuth Apps

Personal access tokens

## Personal access tokens

Generate new token Revoke all

Tokens you have generated that can be used to access the [GitHub API](#).

<b>newMBP</b> — <i>admin:enterprise, admin:gpg_key, admin:org,</i> Last used within the last week	Disable SSO ▾	Delete
<i>admin:org_hook, admin:public_key, admin:repo_hook, delete:packages, delete_repo, gist, notifications, repo, user, workflow, write:discussion, write:packages</i>		
Expires on <i>Wed, Aug 10 2022</i> .		
<b>PC Commits</b> — <i>admin:enterprise, admin:gpg_key,</i> Last used within the last 3 months	Enable SSO ▾	Delete
<i>admin:org, admin:org_hook, admin:public_key, admin:repo_hook, delete:packages,</i>		

## Note

this is for class

What's this token for?

## Expiration \*

30 days



The token will expire on Thu, Nov 18 2021

## Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)

- repo** Full control of private repositories
  - repo:status** Access commit status
  - repo\_deployment** Access deployment status
  - public\_repo** Access public repositories
  - repo:invite** Access repository invitations

# Save That Credential Somewhere Safe!

Or else make a new one every time you need one...

# An Aside to Command Lines

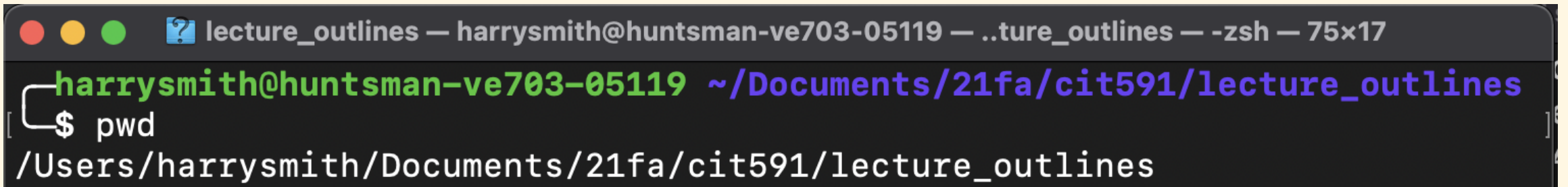
Terminal on MacOS and Cmdr for Windows are both command line interfaces for interacting with your computer.

The model is like this:

- You're always in a *directory* (a folder)
- You can interact with files, directories, and programs
- You can go to other directories as you like.

# Useful Command Line Commands

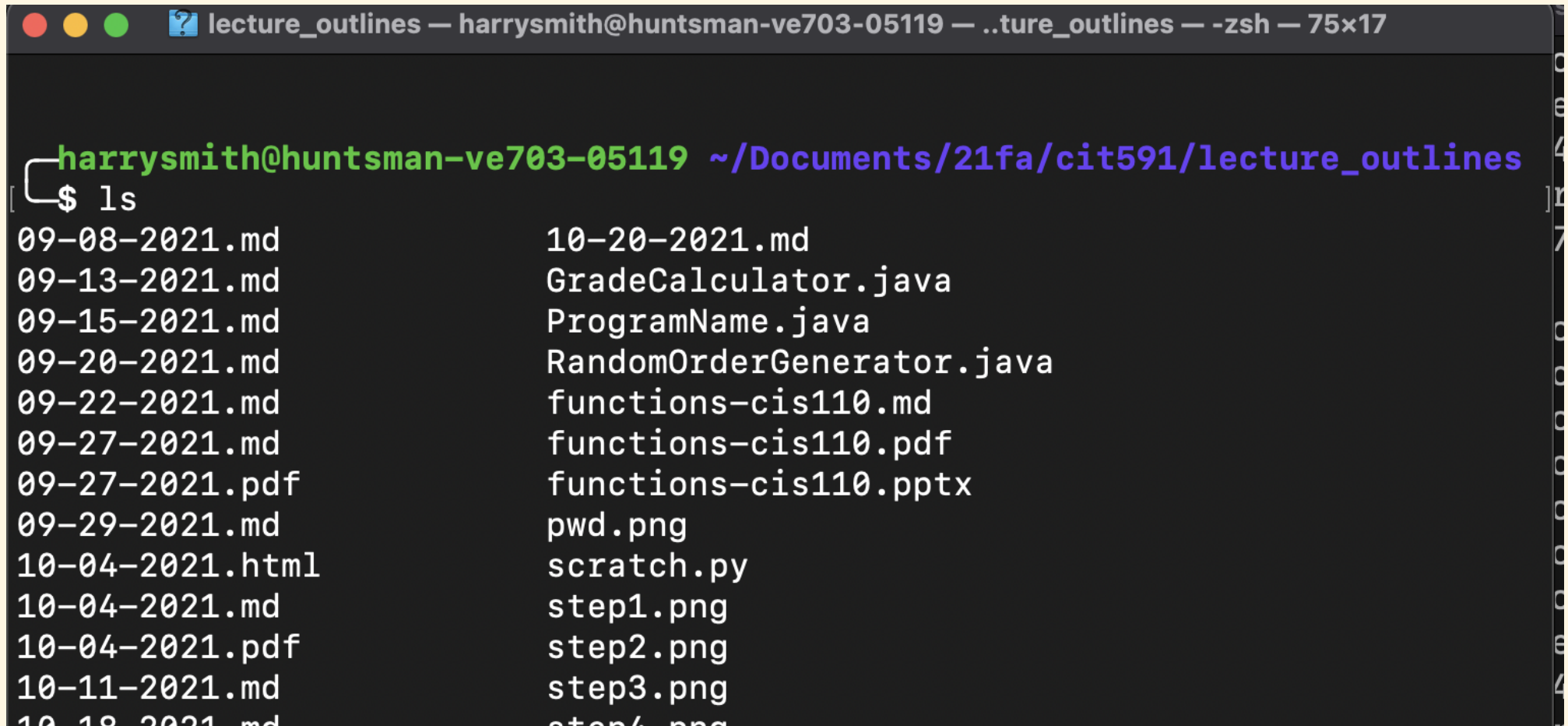
"WHERE AM I???" → `pwd`



```
lecture_outlines — harrysmith@huntsman-ve703-05119 — ..ture_outlines — -zsh — 75x17
harrysmith@huntsman-ve703-05119 ~/Documents/21fa/cit591/lecture_outlines
$ pwd
/Users/harrysmith/Documents/21fa/cit591/lecture_outlines
```

# Useful Command Line Commands

"What are the files in this directory?"  `ls`

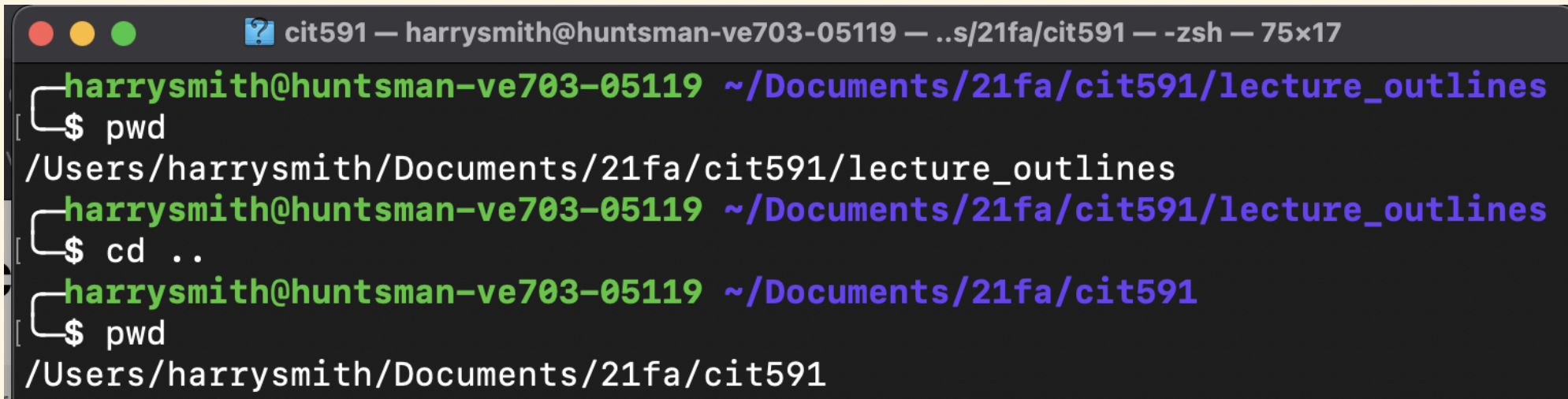


```
lecture_outlines — harrysmith@huntsman-ve703-05119 — ..ture_outlines — -zsh — 75x17

harrysmith@huntsman-ve703-05119 ~/Documents/21fa/cit591/lecture_outlines
$ ls
09-08-2021.md          10-20-2021.md
09-13-2021.md          GradeCalculator.java
09-15-2021.md          ProgramName.java
09-20-2021.md          RandomOrderGenerator.java
09-22-2021.md          functions-cis110.md
09-27-2021.md          functions-cis110.pdf
09-27-2021.pdf         functions-cis110.pptx
09-29-2021.md          pwd.png
10-04-2021.html        scratch.py
10-04-2021.md          step1.png
10-04-2021.pdf         step2.png
10-11-2021.md          step3.png
10-18-2021.md          step4.png
```

# Useful Command Line Commands

"I want to go to the next folder up" ➡ `cd ..`

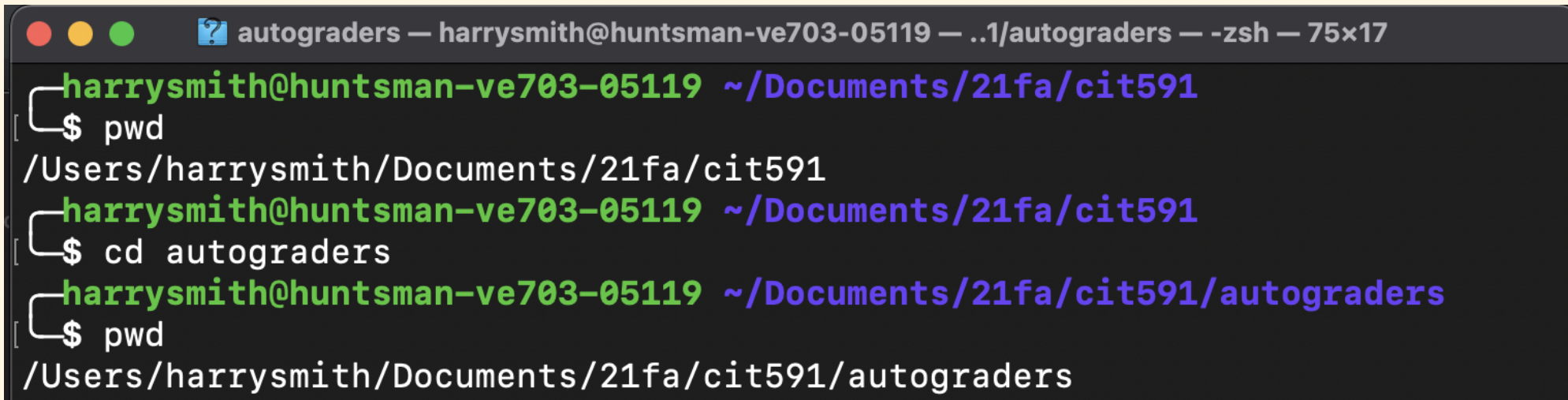


```
cit591 — harrysmith@huntsman-ve703-05119 — ..s/21fa/cit591 — -zsh — 75x17
harrysmith@huntsman-ve703-05119 ~/Documents/21fa/cit591/lecture_outlines
└─$ pwd
/Users/harrysmith/Documents/21fa/cit591/lecture_outlines
harrysmith@huntsman-ve703-05119 ~/Documents/21fa/cit591/lecture_outlines
└─$ cd ..
harrysmith@huntsman-ve703-05119 ~/Documents/21fa/cit591
└─$ pwd
/Users/harrysmith/Documents/21fa/cit591
```



# Useful Command Line Commands

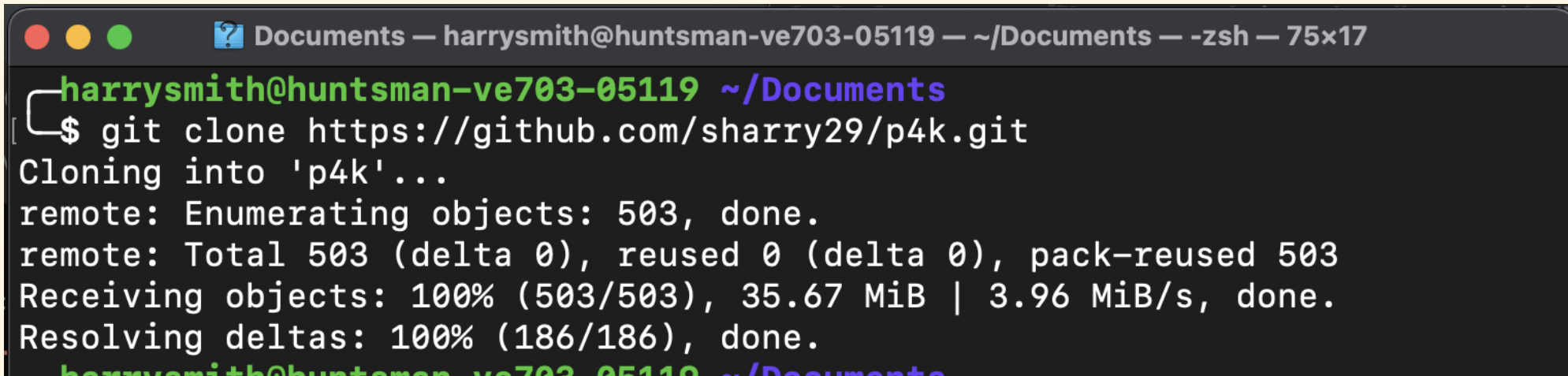
"I want to go down into this other folder"  `cd <folder-name>`



```
autograders — harrysmith@huntsman-ve703-05119 — ../autograders — -zsh — 75x17
harrysmith@huntsman-ve703-05119 ~/Documents/21fa/cit591
└─$ pwd
/Users/harrysmith/Documents/21fa/cit591
harrysmith@huntsman-ve703-05119 ~/Documents/21fa/cit591
└─$ cd autograders
harrysmith@huntsman-ve703-05119 ~/Documents/21fa/cit591/autograders
└─$ pwd
/Users/harrysmith/Documents/21fa/cit591/autograders
```

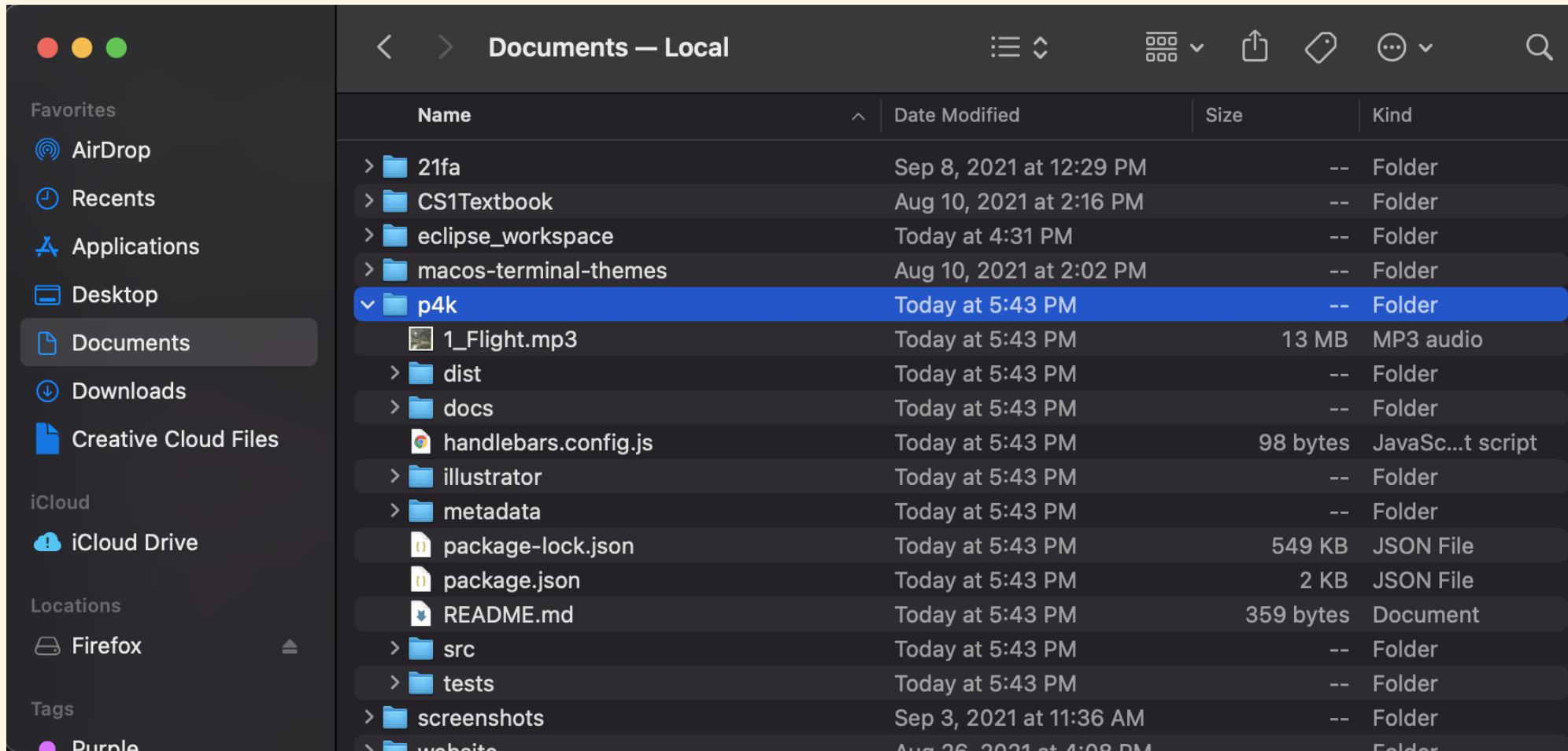
# Using Git on the Command Line

`git clone <repo-uri>` lets you download the files from a remote repository.

A terminal window screenshot showing the execution of the `git clone` command. The window title is "Documents — harrismith@huntsman-ve703-05119 — ~/Documents — -zsh — 75x17". The prompt is `harrismith@huntsman-ve703-05119 ~/Documents`. The command entered is `$ git clone https://github.com/sharry29/p4k.git`. The output shows the cloning process: "Cloning into 'p4k'...", "remote: Enumerating objects: 503, done.", "remote: Total 503 (delta 0), reused 0 (delta 0), pack-reused 503", "Receiving objects: 100% (503/503), 35.67 MiB | 3.96 MiB/s, done.", and "Resolving deltas: 100% (186/186), done.". The prompt at the bottom is partially visible as `harrismith@huntsman-ve703-05119 ~/Documents`.

```
Documents — harrismith@huntsman-ve703-05119 — ~/Documents — -zsh — 75x17
harrismith@huntsman-ve703-05119 ~/Documents
$ git clone https://github.com/sharry29/p4k.git
Cloning into 'p4k'...
remote: Enumerating objects: 503, done.
remote: Total 503 (delta 0), reused 0 (delta 0), pack-reused 503
Receiving objects: 100% (503/503), 35.67 MiB | 3.96 MiB/s, done.
Resolving deltas: 100% (186/186), done.
harrismith@huntsman-ve703-05119 ~/Documents
```

# Now I have these files!



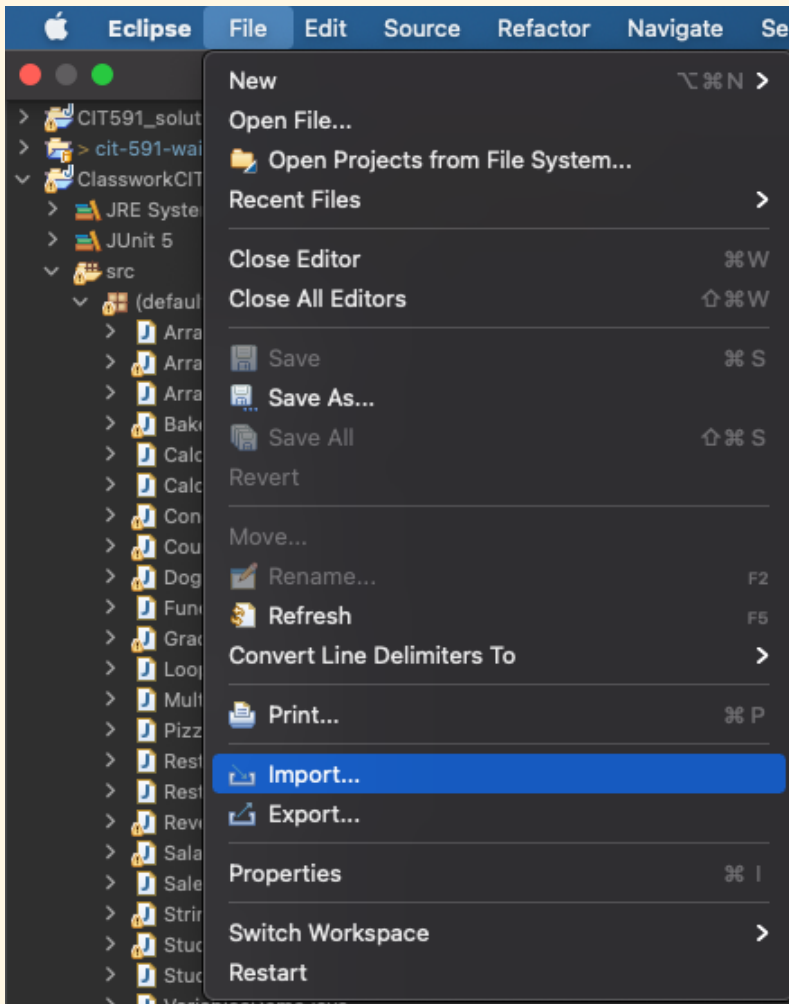
# Using Git on the Command Line: Committing to Remote Repo

From within the directory from a cloned repo...

```
# add files to this commit  
git add <file_name>  
# make a new commit "snapshot"  
git commit -m "your commit message here, describing changes" -a  
# add your commit to the remote repo  
git push
```

This is a bread-and-butter technique.

# Using Git within Eclipse (Cloning a Repo)



# Using Git within Eclipse (Cloning a Repo)

```
Git -> Projects from Git (with smart import) -> Clone URI
```

Then enter the URI, your username, and your access token.

Make sure to check "Store in Secure Store"!!!

# Git Collaboration Partner Activity

Find a partner to collaborate with

# Git Collaboration Partner Activity

Make a new folder with your name somewhere on your computer that you know how to find!

(Do this using File Explorer or Finder)



# Git Collaboration Partner Activity

Add some stuff to this folder, maybe a text file or a photo you don't mind sharing.

(Do this using File Explorer or Finder)

# Git Collaboration Partner Activity

Navigate to the folder IN YOUR TERMINAL.

You can copy the file path from Finder/File Explorer and then

```
cd <COPIED-DIRECTORY>
```

# Git Collaboration Partner Activity

Navigate to the folder IN YOUR TERMINAL.

You can copy the file path from Finder/File Explorer and then

```
cd <COPIED-DIRECTORY>
```

# Git Collaboration Partner Activity

From inside your folder in the Terminal,

```
git init .
```

# Git Collaboration Partner Activity

On [GitHub.com](https://github.com), create a new private repo named with your favorite color.

# Git Collaboration Partner Activity

Push your local repo to GitHub with the following:

```
git remote add origin <REPO-URL>  
git branch -M main  
git push -u origin main
```

# Git Collaboration Partner Activity

Go to **Settings -> Manage Access** and click **Add People** to add your partner by username.

Once added, accept each other's invitations.

# Git Collaboration Partner Activity

```
cd ..  
git clone <PARTNER_REPO_URL>
```



# Git Collaboration Partner Activity

Make a change in a file in the Repo you just cloned.

# Git Collaboration Partner Activity

From within the directory from the cloned repo...

```
# add files to this commit  
git add <file_name>  
# make a new commit "snapshot"  
git commit -m "your commit message here, describing changes" -a  
# add your commit to the remote repo  
git push
```

This is a bread-and-butter technique.