

Lecture 18:

# LIFE SKILL: CODE REUSE

---

CSC111: Introduction to CS through Programming

R. Jordan Crouser

Assistant Professor of Computer Science

Smith College

# Outline

- Quick review of "coding best practices"
- Some ethical questions
- Using online resources ethically
- How to attribute someone else's code
- Version control: the great frustration mitigator ☺

# Discussion

We've talked a lot about ideas like:

- Lecture 5: “S<sup>4</sup>”: start small | slow | simple
- Lecture 7: organizing your code so it's easy to reuse pieces
- Lecture 13: documenting your code so it's easy to come back to it
- Lecture 17: forking code from other people's repositories

Can you think of any **ethical concerns** about this?



# The balancing act...

INTEGRITY  
(not cheating)

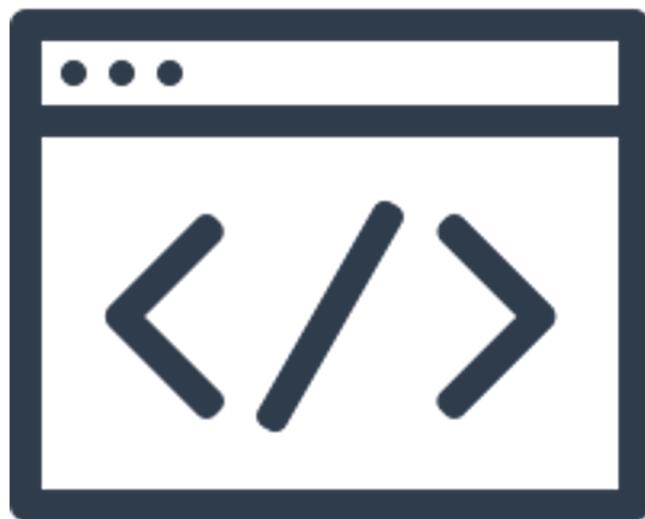


EFFICIENCY  
(not re-implementing)



...so how do you know when it's **okay** to reuse code?

# Let's consider a more familiar case...



Same or different?

## Dictionary

Enter a word, e.g. "pie"



# pla·gia·rism

/'plājə,rizəm/

*noun*

the practice of taking someone else's work or ideas and passing them off as one's own.

*synonyms:* copying, infringement of copyright, **piracy**, **theft**, stealing; *informal* cribbing

"accusations of plagiarism"



Translations, word origin, and more definitions

# Scenario 0: self-reuse, not in a class

- You wrote a program that **solved a particular problem** as part of a project you're working on for fun
- Later on for a **different project** you're working on for fun, you need to solve the same problem
- Questions:
  - Can you reuse the code?
  - Does it matter what the code **does**?
  - Do you need to **attribute** the code?

# Scenario 1: self-reuse, in-class work

- You wrote a program that **solved a particular problem** for a previous assignment in a course
- Later on for a **project you're working on for fun**, you need to solve the same problem
- Questions:
  - Can you reuse the code?
  - Does it matter what the code **does**?
  - Do you need to **attribute** the code?

# Scenario 2: self-reuse, same course

- You wrote a program that **solved a particular problem** for a previous assignment in a course
- In a **later assignment for that same course**, you need to solve the same problem as part of a larger process
- Questions:
  - Can you reuse the code?
  - Does it matter what the code **does**?
  - Do you need to **attribute** the code?
  - Does it matter if you copy/paste or **import** it?

# Scenario 3: self-reuse, different course

- You wrote a program that **solved a particular problem** for a previous assignment in a course
- In **an assignment for a different course**, you need to solve the same problem
- Questions:
  - Can you reuse the code?
  - Does it matter what the code **does**?
  - Do you need to **attribute** the code?
  - Does it matter if it's the **whole assignment**, or just one part?

# Scenario 4: self-reuse, academic work →

- You wrote a program that **solved a particular problem** for an assignment in a course
- You later get a **job as a software engineer**, and you need to solve the same problem
- Questions:
  - Can you reuse the code?
  - Does it matter what the code **does**?
  - Do you need to **attribute** the code?
  - Does it matter if it's the **whole assignment**, or just one part?

# Scenario 5: professors and TAs

- You are trying to **solve a particular problem** for an assignment in a course, but you are stuck
- You ask the professor or TA for advice, they walk you through **how to implement one of the functions**
- Questions:
  - Can you use the code?
  - Does it matter what the code **does**?
  - Do you need to **attribute** the code?

# Scenario 6: peers

- You are trying to **solve a particular problem** for an assignment in a course, but you are stuck
- You ask a friend who took the class last year, they walk you through **how to implement one of the functions**
- Questions:
  - Can you use the code?
  - Does it matter what the code **does**?
  - Do you need to **attribute** the code?

# Scenario 7: online sources

- You are trying to **solve a particular problem** for an assignment in a course, but you are stuck
- You look online to try to understand a concept, someone walks through **how to implement one of the functions**
- Questions:
  - Can you use the code?
  - Does it matter what the code **does**?
  - Do you need to **attribute** the code?

# Common online Q&A resources



# Demo: using StackOverflow

DEMO

TIME

# How to attribute online code

```
# ----- START ATTRIBUTED CODE SECTION -----
# Code created with the help of Stack Overflow
# https://stackoverflow.com/questions/49581417
#
# Question by Alden:
# https://stackoverflow.com/users/9378177/alden
#
# Answer by CD Lane:
# https://stackoverflow.com/users/5771269/cdlane

# ...THE ACTUAL CODE GOES HERE...

# ----- END ATTRIBUTED CODE SECTION -----
```

# Discussion

For educational purposes, does copyright matter?

Isn't everything covered by "**fair use**"?



# Rules for code reuse

- Always **attribute**
- Only use code you **actually understand**
- If it's for a course, **talk to the professor first**
- Understand **the license**, e.g. for StackOverflow



**Attribution-ShareAlike 3.0 Unported (CC BY-SA 3.0)**

# Activity: code reuse

- With your group, pick a simple game (like
- Use **online resources** to find a working python implementation
- **One partner:** make a new repl containing the code you found, add your partners as **collaborators**
- Write up the **attribution**
- Make sure that you **understand each piece of the code** (add lots of comments!)

# Show and tell

What did you **find**?

How does it **work**?



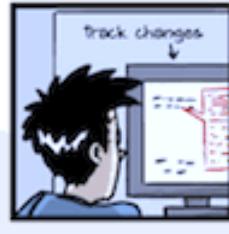
# Version control

- A system that **records changes** to files over time
- Allows you to **restore** specific versions later if you need to

# A common (terrible) pattern

(thanks to phdcomics.com for being hilarious  
and Ben Baumer for the following slides)

## "FINAL".doc



# Slightly better

- 2019-07-25\_1151am\_hw5.py
- 2019-07-25\_0232pm\_hw5.py
- 2019-07-25\_0603pm\_hw5.py
- 2019-07-25\_0947pm\_hw5.py
- 2019-07-25\_1153pm\_hw5.py
- 2019-07-26\_0117am\_hw5.py

...but what if you have multiple files working together?

# Best: version control tools like git

The screenshot shows a GitHub repository page for the user jcrouser named CSC111. The repository is private, has 5 stars, 1 fork, and 1 issue. It contains 2 branches and 0 tags. The master branch is selected. The repository history shows 286 commits from 6 days ago, including updates to assignments, demos, img, labs-old, labs, lectures, site\_libs, .gitignore, FP Workshop.docx, README.html, README.md, \_config.yml, and site.yml. The repository details show no description, website, or topics provided. It also lists releases, packages, and contributors.

**Code** **Issues** **Pull requests 1** **Actions** **Projects** **Wiki** **Security** **Insights** **Settings**

**About**

No description, website, or topics provided.

**Readme**

**Releases**

No releases published  
Create a new release

**Packages**

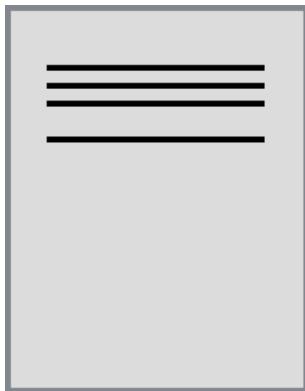
No packages published  
Publish your first package

**Contributors 2**

Avatar	Name	Contributions
	jcrouser R. Jordan Crouser	286
	marshall62 David Marshall	1

# Version control 101

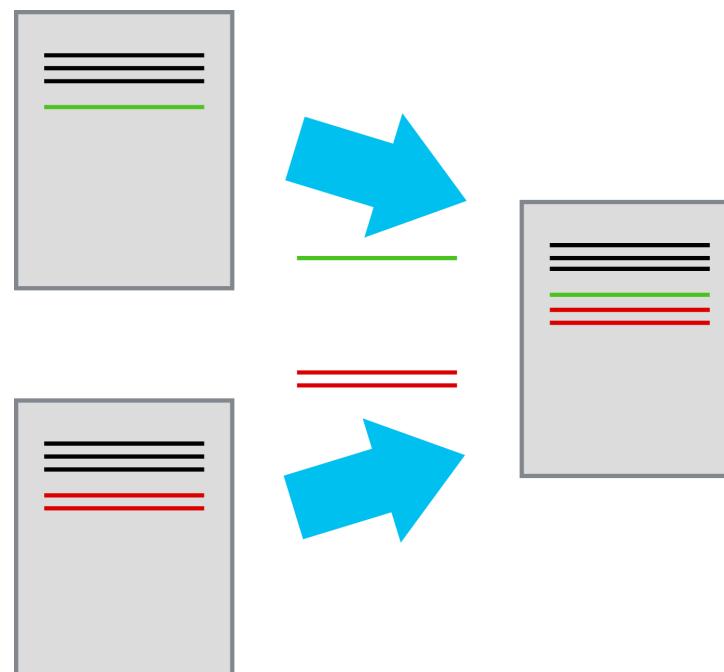
- Start with a base version
- Instead of saving lots of copies, just save just the **changes** (and note where they belong)



Think of it like a tape: can rewind & play back each change

# Collaboration with version control

- **Multiple people** can contribute changes
- “Playing back” different sets of changes = different **versions** of the document



# Git and GitHub

git

- Command-line interface (CLI) that does the work\*
- OS-specific installation
- Needed to use GitHub
- git pull, git push, etc.



- hosting for git projects
- strong community
- additional tools to enhance collaboration
- a common location to share your work

\* you also have a CLI for python!

# Commits

## Commits on Jul 25, 2019

Fixed dead links



jcrouser committed 3 days ago ✓



01f96c3



Added Lab 16



jcrouser committed 4 days ago ✓



7c7feae



Adding Lab 12



jcrouser committed 4 days ago ✓



131090e



## Commits on Jul 24, 2019

Fixed lab date



jcrouser committed 4 days ago ✓



135e881



Added detail on exam scores



jcrouser committed 5 days ago ✓



72ee277



# Diff (“difference”)

Fixed dead links

master

 jcrouser committed 3 days ago 1 parent 7c7feae commit 01f96c315aa6c2d05f9bc59df258057d6af165a2

Showing 9 changed files with 39 additions and 8 deletions.

Unified Split

@@ -339,8 +339,10 @@	339 <li>	339 <li>
340 <a href="labs/lab-11-classes.html">11. Slow Pokémon</a>	340 <a href="labs/lab-11-classes.html">11. Slow Pokémon</a>	341 </li>
341 </li>		342 + <li>
		343 + <a href="labs/lab-12-recursion.html">12. Fractals</a>
		344 + </li>
342 <li class="dropdown-header">12. Sorting</li>	345 <li class="dropdown-header">12. Sorting</li>	
343 - <li class="dropdown-header">13. Fractals</li>		
344 <li class="dropdown-header">14. Making a Scene</li>	346 <li class="dropdown-header">14. Making a Scene</li>	
345 <li class="dropdown-header">15. Fish Tank</li>	347 <li class="dropdown-header">15. Fish Tank</li>	
346 <li class="dropdown-header">16. Interactive Fish Tank</li>	348 <li class="dropdown-header">16. Interactive Fish Tank</li>	

# This seems complicated... why bother?

- Automatic tracking of changes
- No download/upload/emailing of files
- Easy collaboration, with ability to resolve merge conflicts
- Searchability
- Can track issues, releases, and milestones
- Visibility:
  - GitHub repositories are **public**
  - Can request free **private** repositories for education or research

# Alternatives



- **Pros:** easy, seamless
- **Cons:** no simultaneous editing, limited version control (no branches)



- **Pros:** easy, simultaneous editing
- **Cons:** no version control, not actually a .py file (can't run it)

# Getting started with git (a walkthrough)

- **Step 1:** go to `github.com/join` and create an account

The screenshot shows a web browser window for the GitHub Join page (`https://github.com/join`). The page has a dark header with the GitHub logo, navigation links like 'Why GitHub?', 'Enterprise', 'Explore', 'Marketplace', 'Pricing', a search bar, and 'Sign in'/'Sign up' buttons. Below the header, the main title is 'Join GitHub' with the subtitle 'The best way to design, build, and ship software.' The page is divided into three steps: 'Step 1: Set up your account', 'Step 2: Choose your subscription', and 'Step 3: Tailor your experience'. The first step is active. A large form for creating a personal account is shown, with fields for 'Username \*' and 'Email address \*'. A note says, 'This will be your username. You can add the name of your organization later.' To the right, a sidebar lists benefits: 'You'll love GitHub' with points for 'Unlimited public repositories', 'Unlimited private repositories', 'Limitless collaboration', and 'Frictionless development'.

Join GitHub · GitHub

GitHub, Inc. [US] | https://github.com/join

Why GitHub? Enterprise Explore Marketplace Pricing

Search GitHub

Sign in Sign up

## Join GitHub

The best way to design, build, and ship software.

Step 1: Set up your account Step 2: Choose your subscription Step 3: Tailor your experience

Create your personal account

Username \*

This will be your username. You can add the name of your organization later.

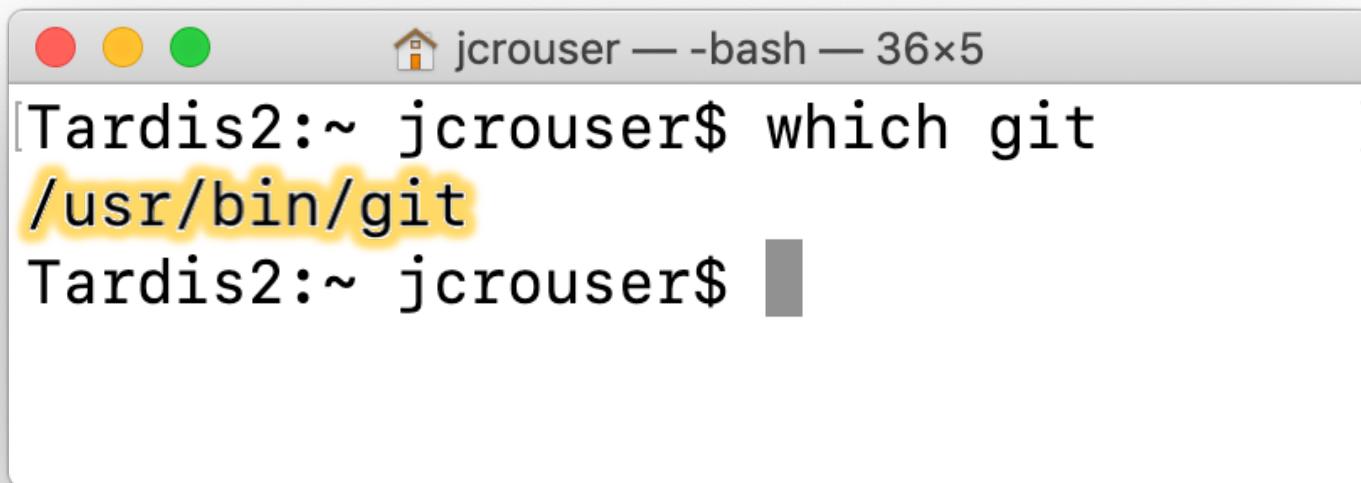
Email address \*

You'll love GitHub

- Unlimited public repositories
- Unlimited private repositories
- ✓ Limitless collaboration
- ✓ Frictionless development

# Getting started with **git** (a walkthrough)

- **Step 2:** make sure git is installed (it should be)
  - open the **Terminal app** (not IDLE)
  - type **which git** at the prompt and hit return

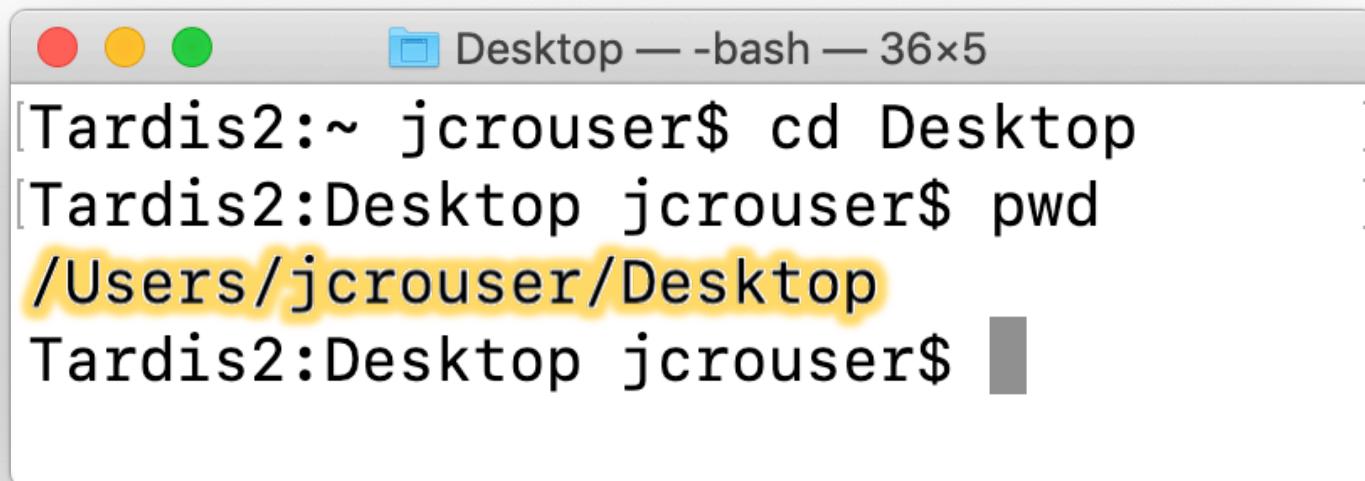


A screenshot of a macOS Terminal window. The window title is "jcrouser — -bash — 36x5". The terminal prompt is "[Tardis2:~ jcrouser\$ which git". The command "which git" was run, and the output "/usr/bin/git" is displayed. The path "/usr/bin/git" is highlighted with a yellow glow effect. The terminal prompt "[Tardis2:~ jcrouser\$" is visible at the bottom.

```
[Tardis2:~ jcrouser$ which git
/usr/bin/git
Tardis2:~ jcrouser$ ]
```

# Getting started with **git** (a walkthrough)

- **Step 3:** navigate to someplace useful, like the Desktop
  - type **cd Desktop** (“change directory”) at the prompt and hit return
  - If you want, type **pwd** (“print working directory”) to see the path



```
[Tardis2:~ jcrouser$ cd Desktop
[Tardis2:Desktop jcrouser$ pwd
/Users/jcrouser/Desktop
Tardis2:Desktop jcrouser$ ]]
```

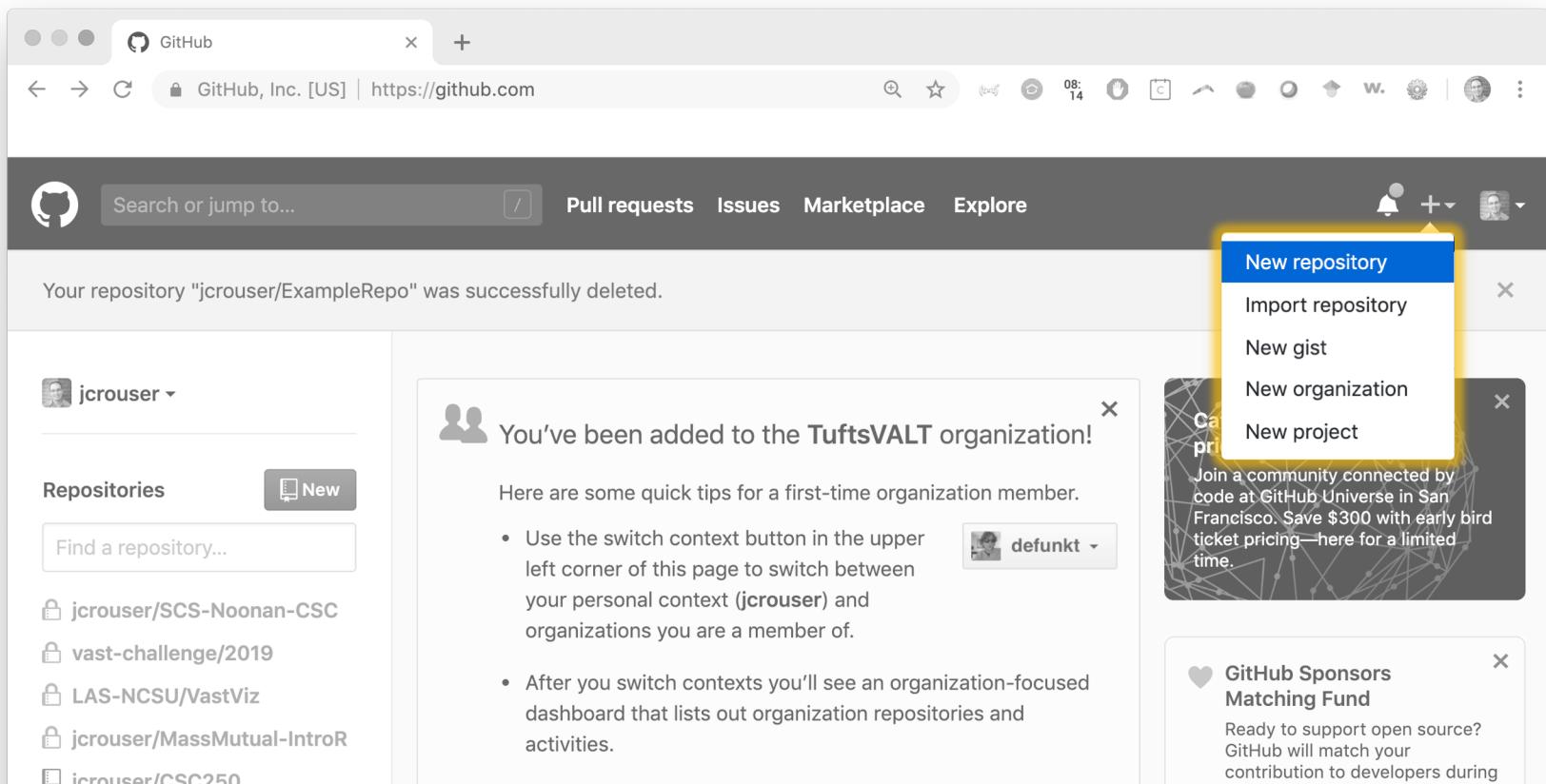
The screenshot shows a macOS terminal window with a light gray background and a title bar that reads "Desktop — -bash — 36x5". The window contains the following text from the command line:

- [Tardis2:~ jcrouser\$ cd Desktop
- [Tardis2:Desktop jcrouser\$ pwd
- /Users/jcrouser/Desktop
- Tardis2:Desktop jcrouser\$ ]]

The path "/Users/jcrouser/Desktop" is highlighted with a yellow glow.

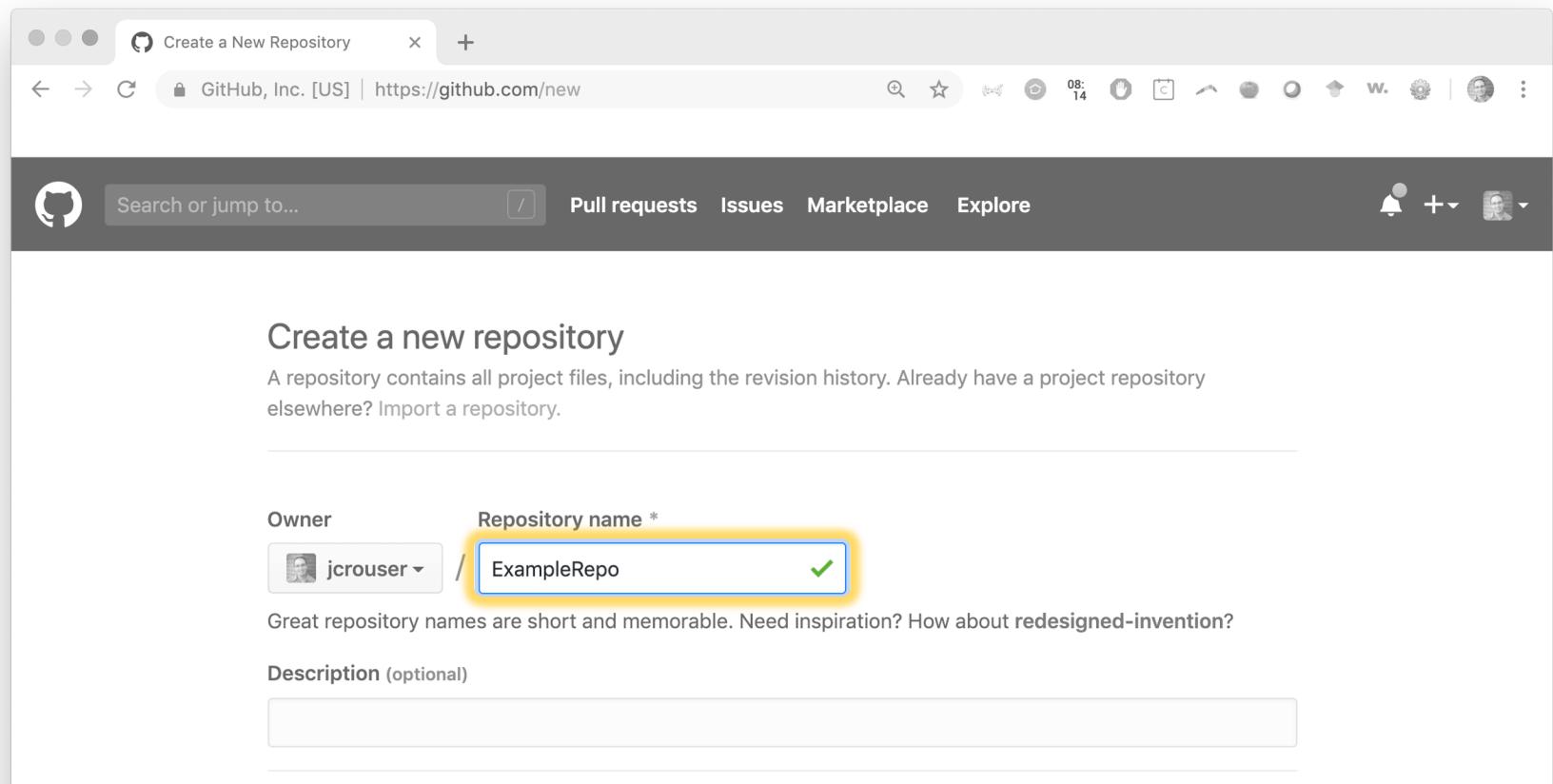
# Getting started with git (a walkthrough)

- **Step 4a: to make a new repository**
  - Click the “+” icon in the upper right corner of the page
  - Select “New repository” from the menu



# Getting started with git (a walkthrough)

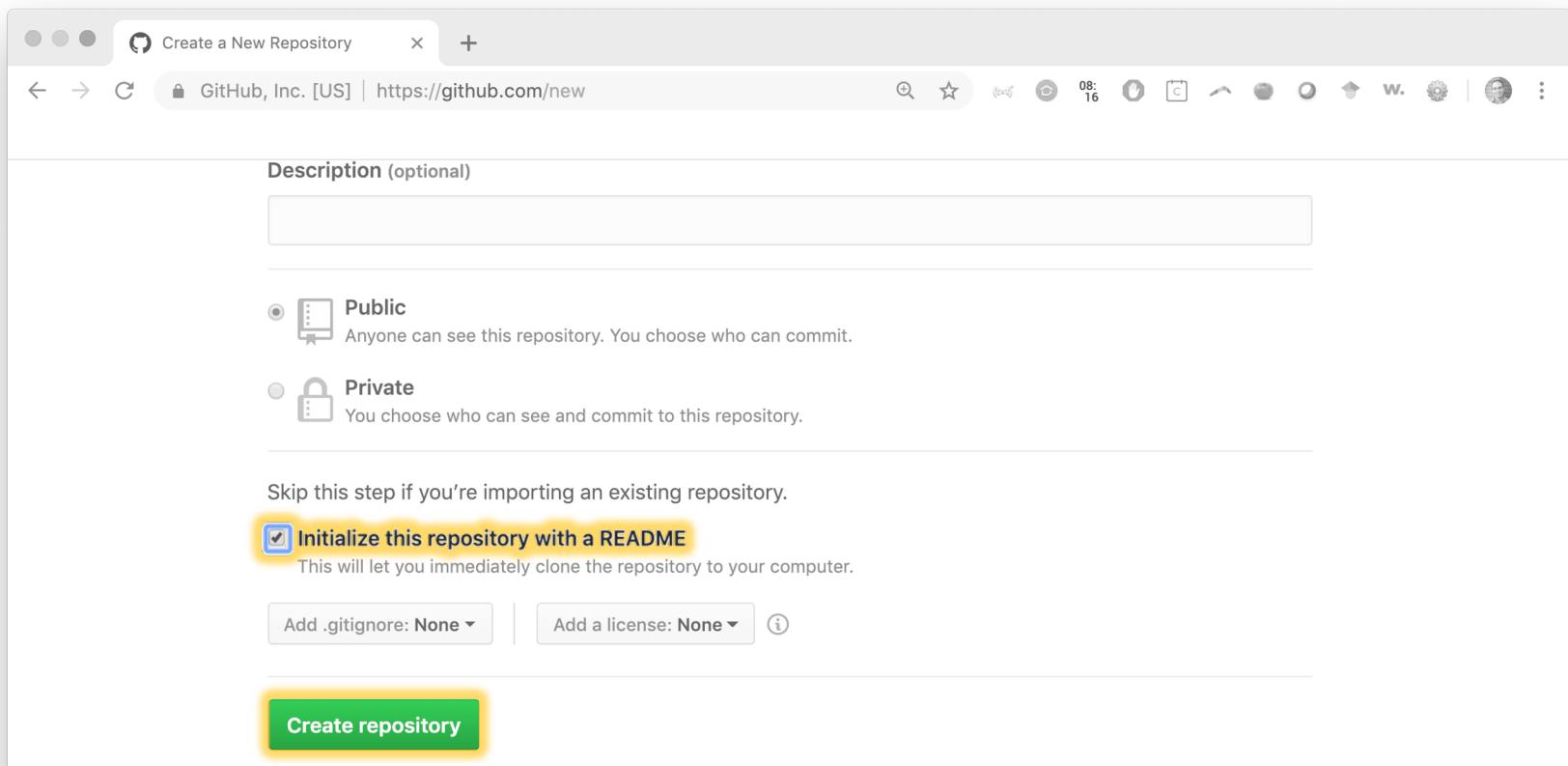
- **Step 4b:** set up your **new** repository
  - Give your repository a name (like **ExampleRepo**)



The screenshot shows a web browser window for GitHub's 'Create a New Repository' page. The URL in the address bar is <https://github.com/new>. The page has a dark header with the GitHub logo, a search bar, and navigation links for Pull requests, Issues, Marketplace, and Explore. Below the header, the main title is 'Create a new repository'. A sub-instruction says: 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.' On the left, there's a 'Owner' dropdown set to 'jcrouser'. To its right is a 'Repository name \*' input field containing 'ExampleRepo', which is highlighted with a yellow rectangular selection. A green checkmark icon is to the right of the input field. At the bottom, a note says: 'Great repository names are short and memorable. Need inspiration? How about redesigned-invention?' There is also an 'Description (optional)' input field at the very bottom.

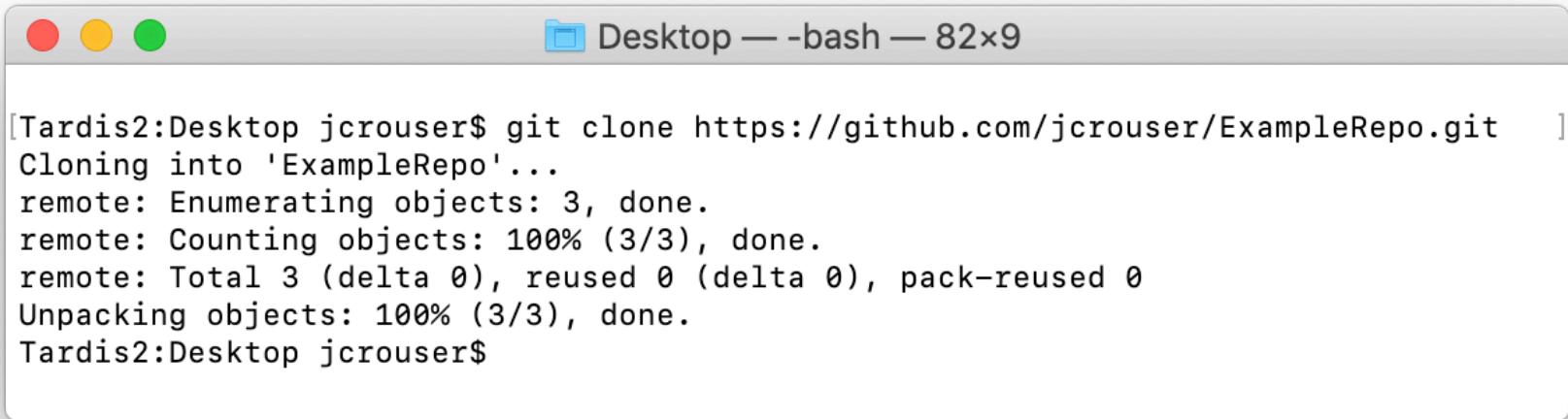
# Getting started with `git` (a walkthrough)

- **Step 4c: set up your new repository**
  - Check the box “Initialize this repository with a README”
  - Click the green “**Create Repository**” button



# Getting started with **git** (a walkthrough)

- **Step 5:** clone your repository to your **computer**
  - go back to the **Terminal** (command line)
  - **git clone https://github.com/yourname/ExampleRepo.git**

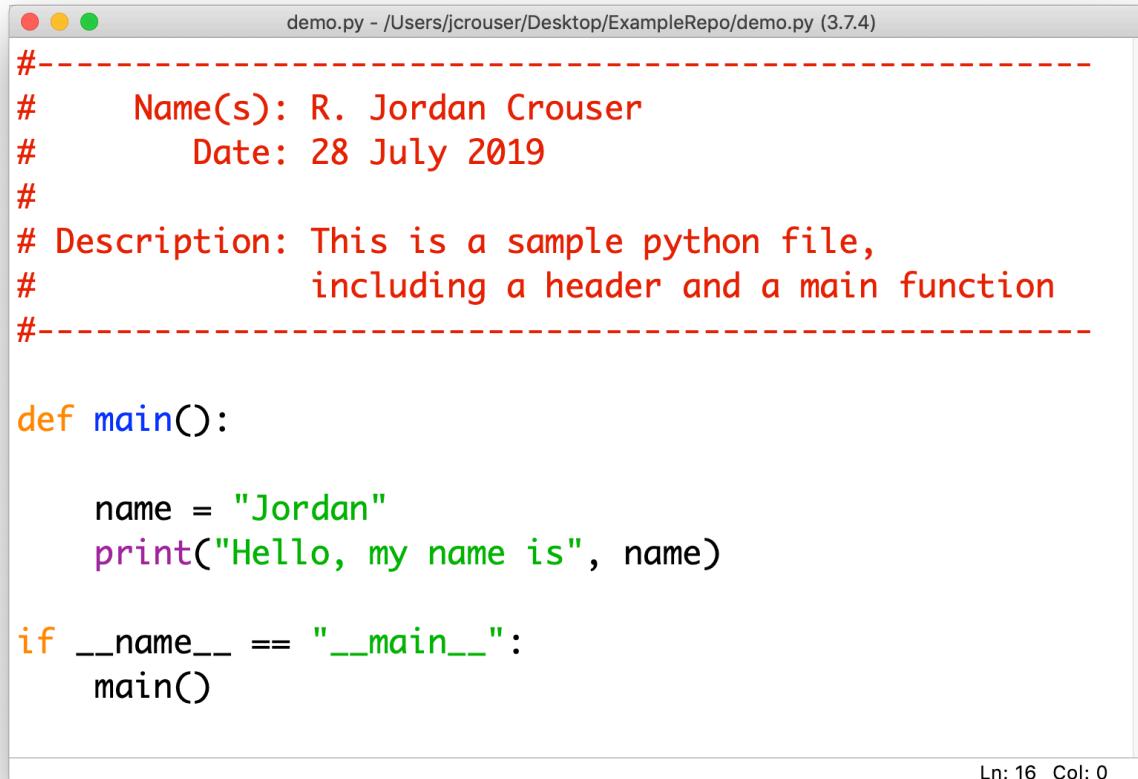


```
[Tardis2:Desktop jcrouser$ git clone https://github.com/jcrouser/ExampleRepo.git
Cloning into 'ExampleRepo'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
Tardis2:Desktop jcrouser$ ]
```

- look on the Desktop and you'll see a new folder called **ExampleRepo**
- take a look inside!

# Getting started with git (a walkthrough)

- **Step 6:** adding a new file to the repository
  - make a file called **demo.py**
  - save it in **ExampleRepo**



A screenshot of a code editor window titled "demo.py - /Users/jcrouser/Desktop/ExampleRepo/demo.py (3.7.4)". The code editor displays the following Python script:

```
#-----
#      Name(s): R. Jordan Crouser
#      Date: 28 July 2019
#
# Description: This is a sample python file,
#               including a header and a main function
#-----

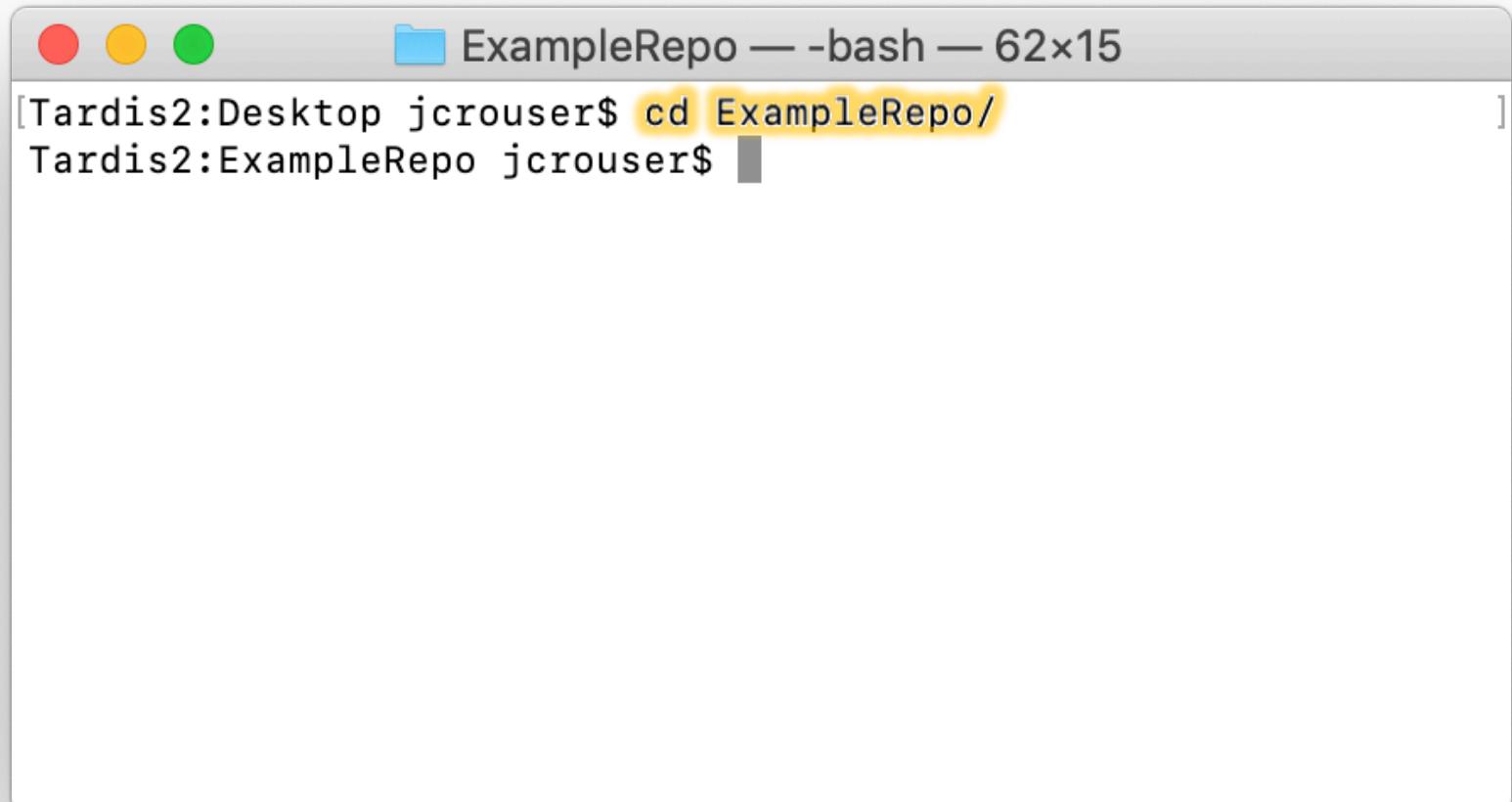
def main():
    name = "Jordan"
    print("Hello, my name is", name)

if __name__ == "__main__":
    main()
```

The code is color-coded: comments are in red, strings are in green, and keywords like `def` and `if` are in blue. The status bar at the bottom right shows "Ln: 16 Col: 0".

# Getting started with **git** (a walkthrough)

- **Step 7a: your first commit**
  - type **cd ExampleRepo** to move into the new directory



A screenshot of a macOS terminal window. The window title is "ExampleRepo — -bash — 62x15". The terminal prompt shows the user's path as "[Tardis2:Desktop jcrouser\$ cd ExampleRepo/" followed by a cursor. The command "cd ExampleRepo/" is highlighted with a yellow selection bar. The background of the terminal is white, and the text is black.

# Getting started with **git** (a walkthrough)

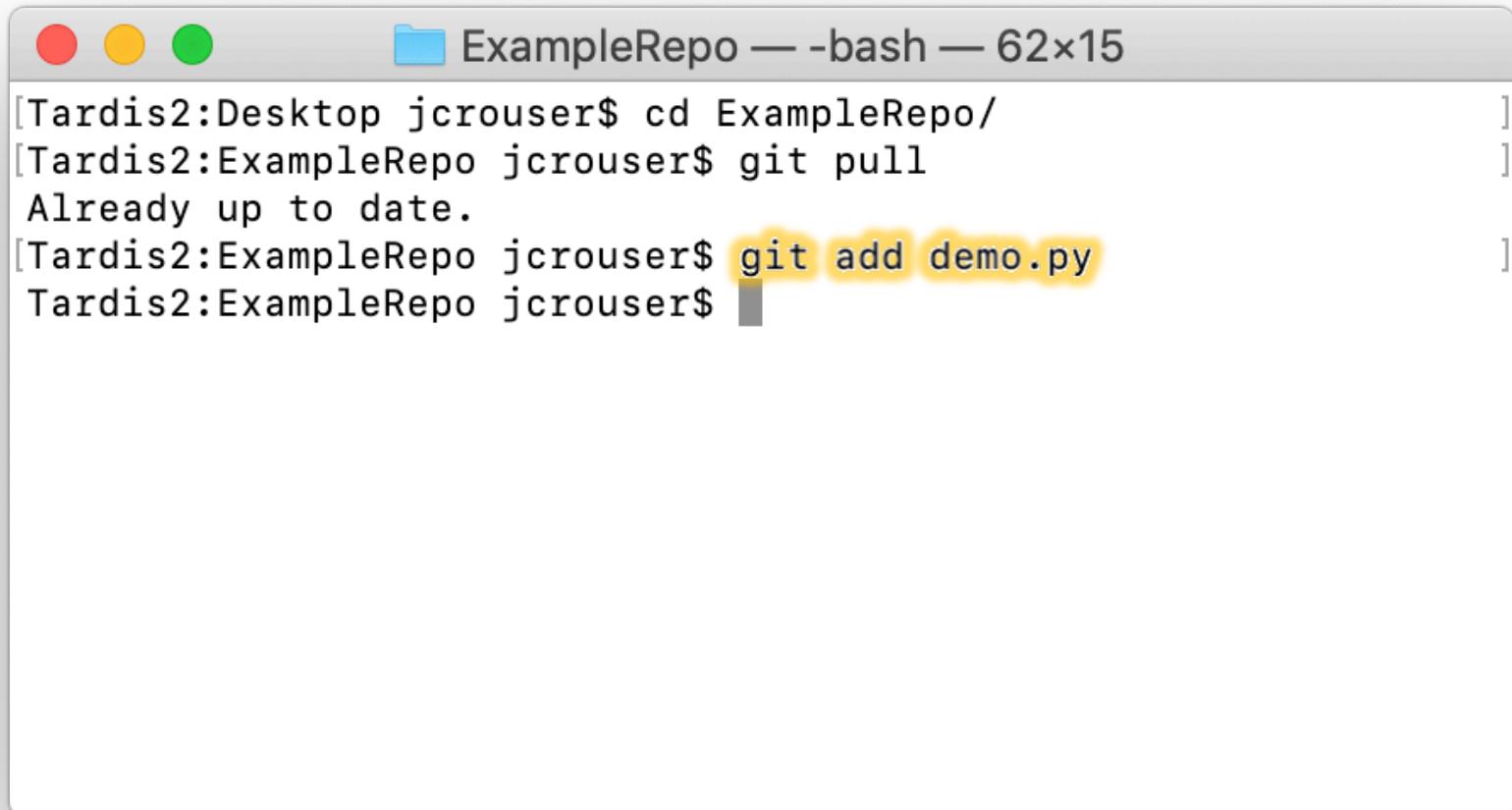
- **Step 7b:** your first **commit**
  - type **git pull** to make sure your local version is up-to-date



```
[Tardis2:Desktop jcrouser$ cd ExampleRepo/
[Tardis2:ExampleRepo jcrouser$ git pull
Already up to date.
Tardis2:ExampleRepo jcrouser$ ]]
```

# Getting started with **git** (a walkthrough)

- **Step 7c: your first commit**
  - type **git add demo.py** to “stage” your changes



The screenshot shows a terminal window with the title bar "ExampleRepo — -bash — 62x15". The window contains the following text:

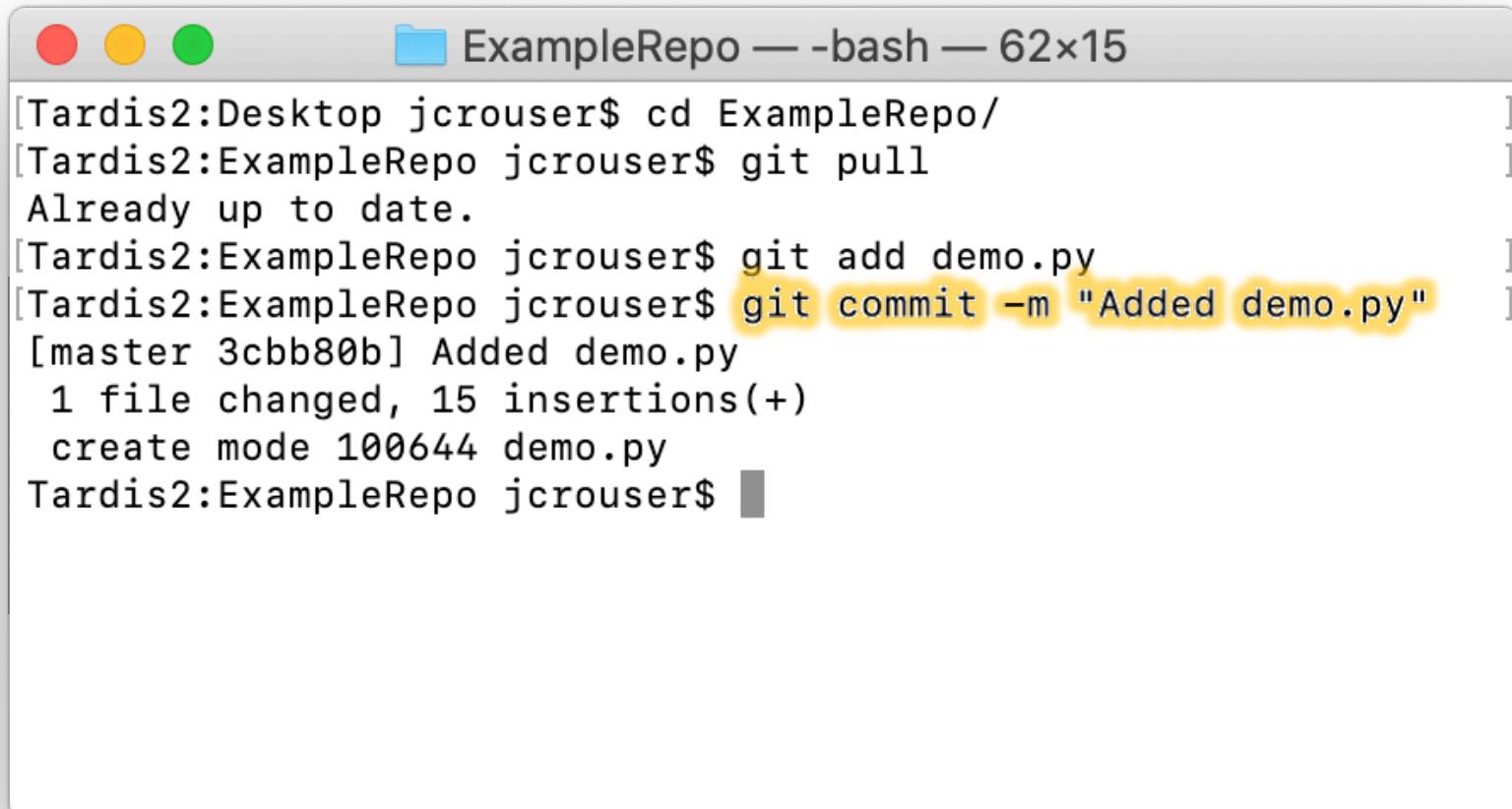
```
[Tardis2:Desktop jcrouser$ cd ExampleRepo/  
[Tardis2:ExampleRepo jcrouser$ git pull  
Already up to date.  
[Tardis2:ExampleRepo jcrouser$ git add demo.py  
Tardis2:ExampleRepo jcrouser$
```

The command `git add demo.py` is highlighted with a yellow glow effect.

# Getting started with `git` (a walkthrough)

- **Step 7d: your first `commit`**

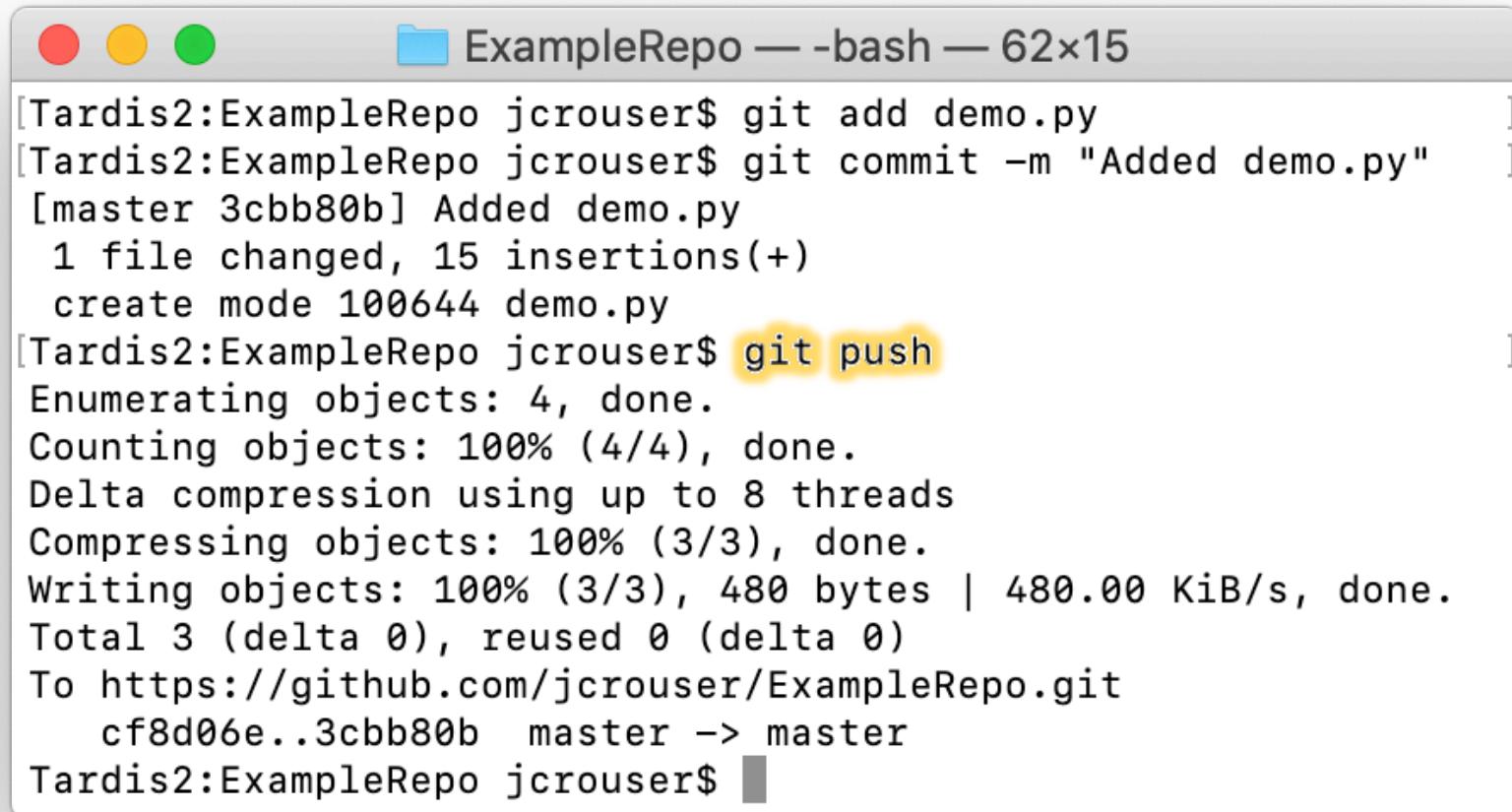
- type `git commit -m "Added demo.py"` to commit the changes, including a message about what you did



```
[Tardis2:Desktop jcrouser$ cd ExampleRepo/
[Tardis2:ExampleRepo jcrouser$ git pull
Already up to date.
[Tardis2:ExampleRepo jcrouser$ git add demo.py
[Tardis2:ExampleRepo jcrouser$ git commit -m "Added demo.py"
[master 3cbb80b] Added demo.py
 1 file changed, 15 insertions(+)
  create mode 100644 demo.py
Tardis2:ExampleRepo jcrouser$ ]]
```

# Getting started with **git** (a walkthrough)

- **Step 7e:** your first **commit**
  - type **git push** to push the changes to the repository



```
[Tardis2:ExampleRepo jcrouser$ git add demo.py
[Tardis2:ExampleRepo jcrouser$ git commit -m "Added demo.py"
[master 3cbb80b] Added demo.py
 1 file changed, 15 insertions(+)
  create mode 100644 demo.py
[Tardis2:ExampleRepo jcrouser$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 480 bytes | 480.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/jcrouser/ExampleRepo.git
  cf8d06e..3cbb80b  master -> master
Tardis2:ExampleRepo jcrouser$ ]]
```

# Getting started with git (a walkthrough)

- **Step 8:** go take a look at your repository!

The screenshot shows a GitHub repository page for 'jcrouser/ExampleRepo'. The page includes a header with navigation links like Pull requests, Issues, Marketplace, and Explore. Below the header, there's a search bar and sections for Code, Issues (0), Pull requests (0), Projects (0), Wiki, Security, Insights, and Settings. A note says 'No description, website, or topics provided.' There are buttons for Unwatch (1), Star (0), Fork (0), and Edit. Key statistics are shown: 4 commits, 1 branch, 0 releases, and 1 contributor. A dropdown menu shows the branch is 'master'. Buttons for New pull request, Create new file, Upload files, Find File, and Clone or download are available. The commit history lists three entries: 'jcrouser Added demo.py' (latest commit, 5 minutes ago), 'README.md' (Initial commit, 24 minutes ago), and 'demo.py' (Added demo.py, 5 minutes ago). A 'README.md' file is also listed at the bottom.

jcrouser / ExampleRepo

No description, website, or topics provided.

Manage topics

4 commits 1 branch 0 releases 1 contributor

Branch: master ▾ New pull request Create new file Upload files Find File Clone or download ▾

jcrouser Added demo.py

README.md Initial commit 24 minutes ago

demo.py Added demo.py 5 minutes ago

README.md

# Basic git procedure

1. **pull** any recent changes (`git pull`)
2. **make edits** on your computer
3. **commit** your local changes (`git add` / `git commit`)
  - this step might be repeated many times
4. **pull** again to avoid merge conflicts (`git pull`)
5. **push** your commit(s) to GitHub (`git push`)

# GitHub advice

- This is all you really need to know for now
- Make small, frequent commits!
- Always **pull** before you **push**
- Slack message us if you get into trouble
- Cheatsheets / more advanced stuff (like pull requests):
  - <https://education.github.com/git-cheat-sheet-education.pdf>
  - <https://www.git-tower.com/blog/git-cheat-sheet/>

# Up next

- ✓ Monday: Life skill #4 - code reuse
- Wednesday: **Recursion pt. 2**
- Friday: Classes pt. 1 – Attributes and Methods
- **Midterm due Friday at 9am**