

# Git and Github

Ling 250/450: Data Science for Linguistics

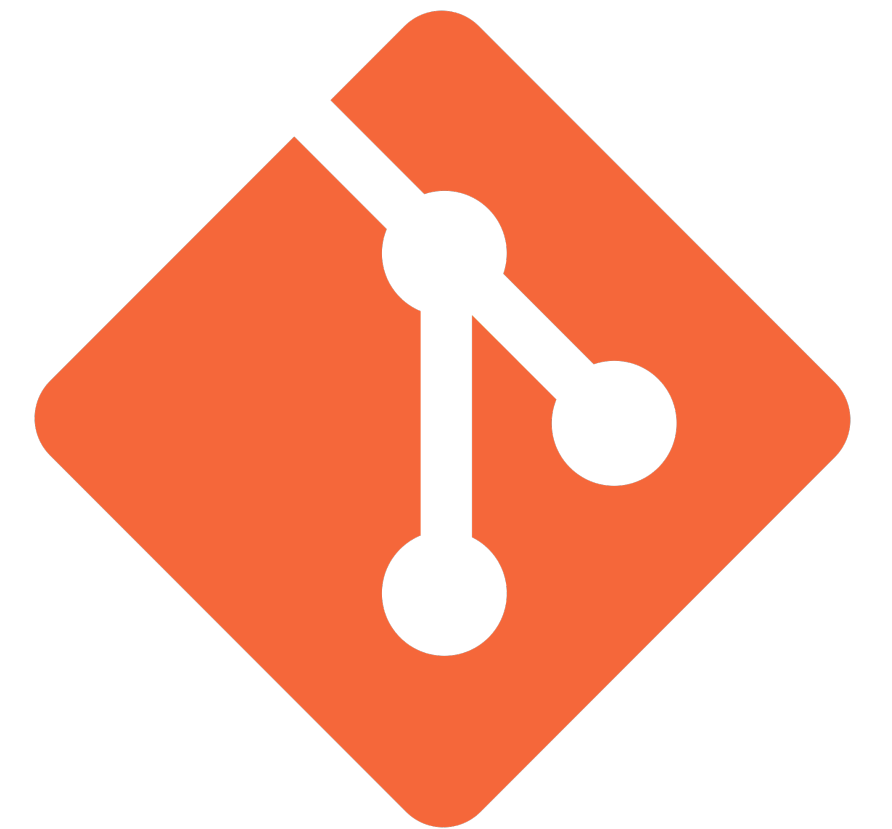
C.M. Downey

Spring 2025

# What even is Git?

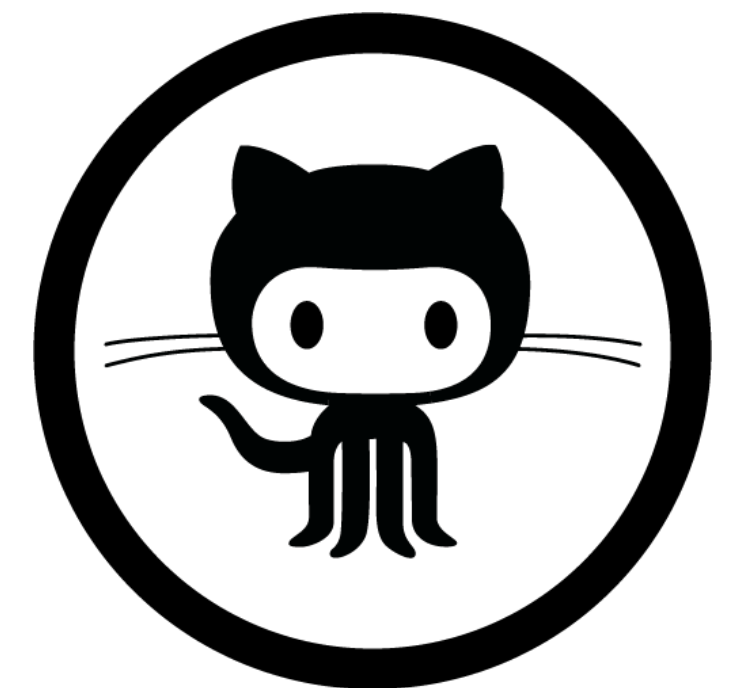
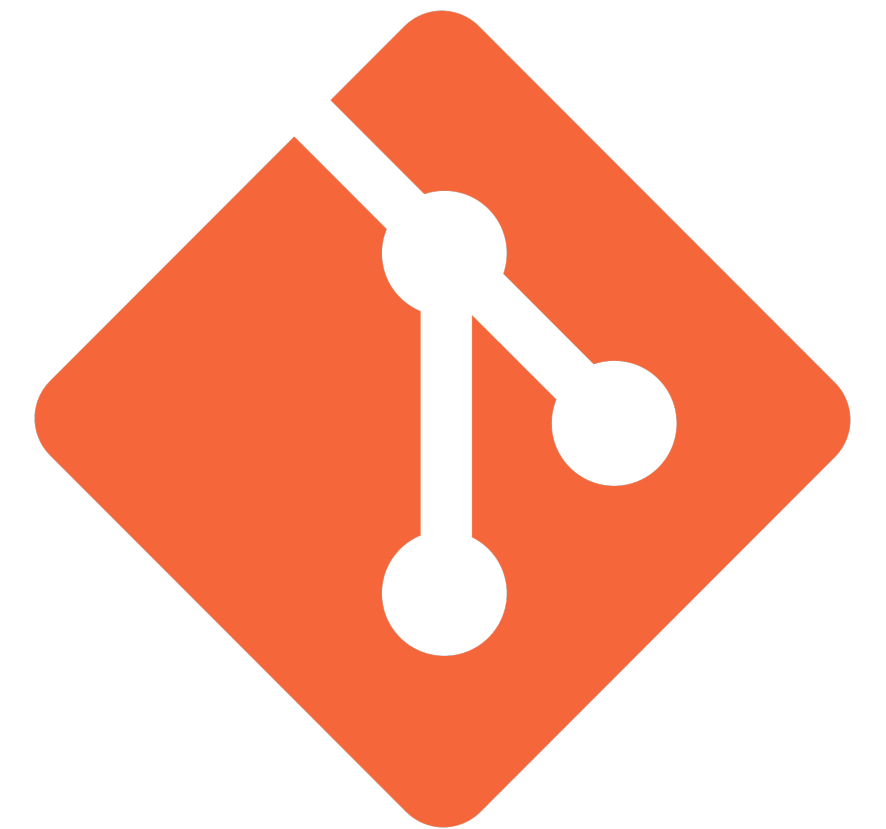
# What even is Git?

- Git
  - At its core, a **system for tracking and saving changes** to files
  - Generally called a **Version Control System**
  - Very complex once you go beyond the basics (which we won't)



# What even is Git?

- Git
  - At its core, a **system for tracking and saving changes** to files
  - Generally called a **Version Control System**
  - Very complex once you go beyond the basics (which we won't)
- Github
  - A website for **hosting Git projects online** (called **repositories**)
  - **Not strictly necessary** for using Git (though it almost always is)
  - Recently: acquired by **Microsoft** (Github, but not Git)



# Main ideas of Git

# Main ideas of Git

- Each change to each file must be **deliberately committed** (saved) to the master copy of the project
- Unlike e.g. Google Docs, where it just saves as you go

# Main ideas of Git

- Each change to each file must be **deliberately committed** (saved) to the master copy of the project
  - Unlike e.g. Google Docs, where it just saves as you go
- All changes are **tracked** and added to a **running history** of changes

# Main ideas of Git

- Each change to each file must be **deliberately committed** (saved) to the master copy of the project
  - Unlike e.g. Google Docs, where it just saves as you go
- All changes are **tracked** and added to a **running history** of changes
- **Different versions** of the same project can **branch off**, and then later be **merged** back together
  - This helps team members work independently **without interfering with each other**



# Why use Git?

# Why use Git?

- In 2025, why not use something like Google docs (**synchronous editing**)?
  - With code, individual changes can have **unintended consequences** (e.g. you might introduce a bug that isn't discovered until much later)
  - With synchronous editing, it is **hard to identify the source** of a problem
  - Systems like Google Docs **don't support branching versions**

# Why use Git?

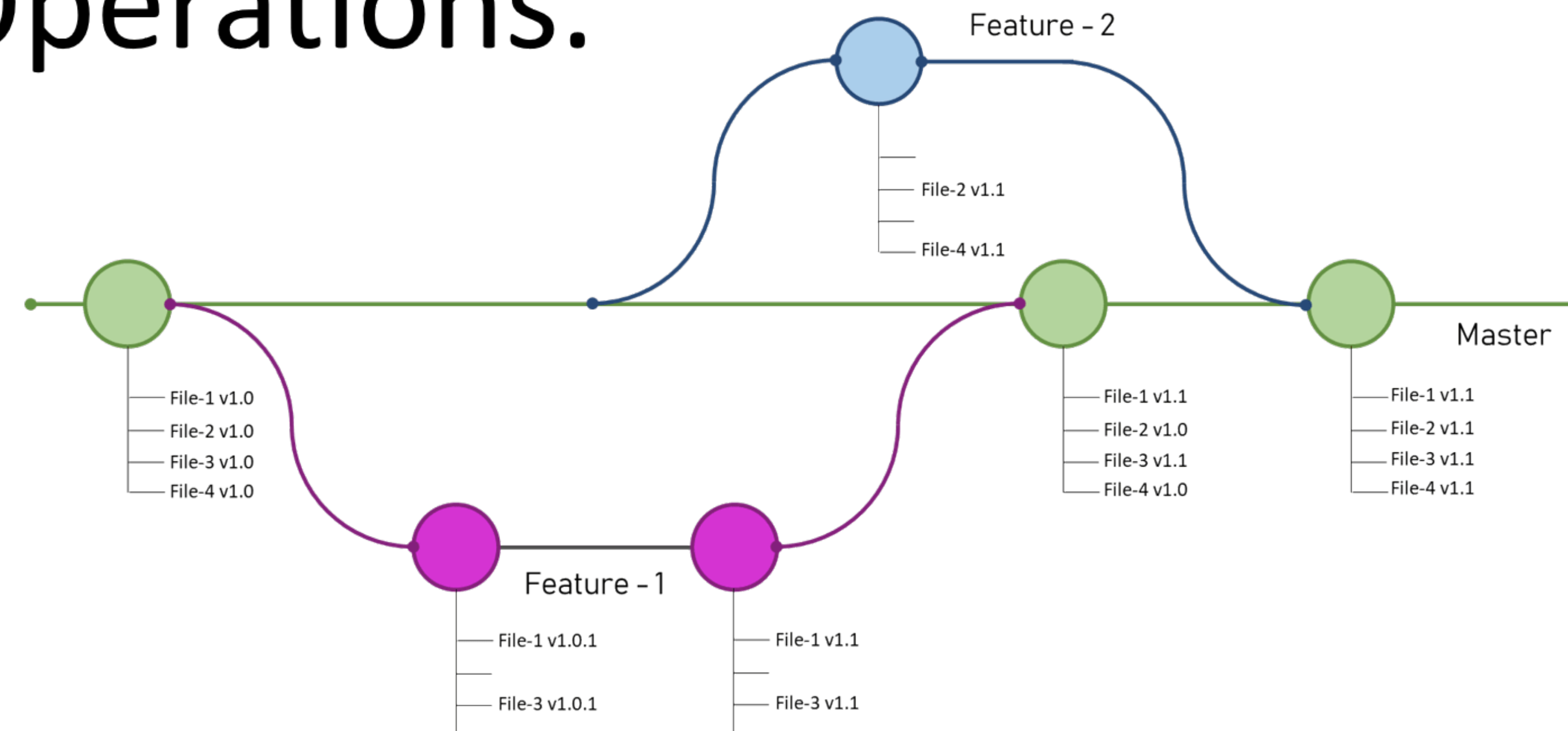
- In 2025, why not use something like Google docs (**synchronous editing**)?
  - With code, individual changes can have **unintended consequences** (e.g. you might introduce a bug that isn't discovered until much later)
  - With synchronous editing, it is **hard to identify the source** of a problem
  - Systems like Google Docs **don't support branching versions**
- Git gives you a **structured history** of the project
  - If you encounter a problem, it is easy to **rewind the project** to a certain point
  - It is also easy to **revert some changes but not others**

# Why use Git?

- In 2025, why not use something like Google docs (**synchronous editing**)?
  - With code, individual changes can have **unintended consequences** (e.g. you might introduce a bug that isn't discovered until much later)
  - With synchronous editing, it is **hard to identify the source** of a problem
  - Systems like Google Docs **don't support branching versions**
- Git gives you a **structured history** of the project
  - If you encounter a problem, it is easy to **rewind the project** to a certain point
  - It is also easy to **revert some changes but not others**
- It is the **de-facto standard** for working with code (across industry and academia)

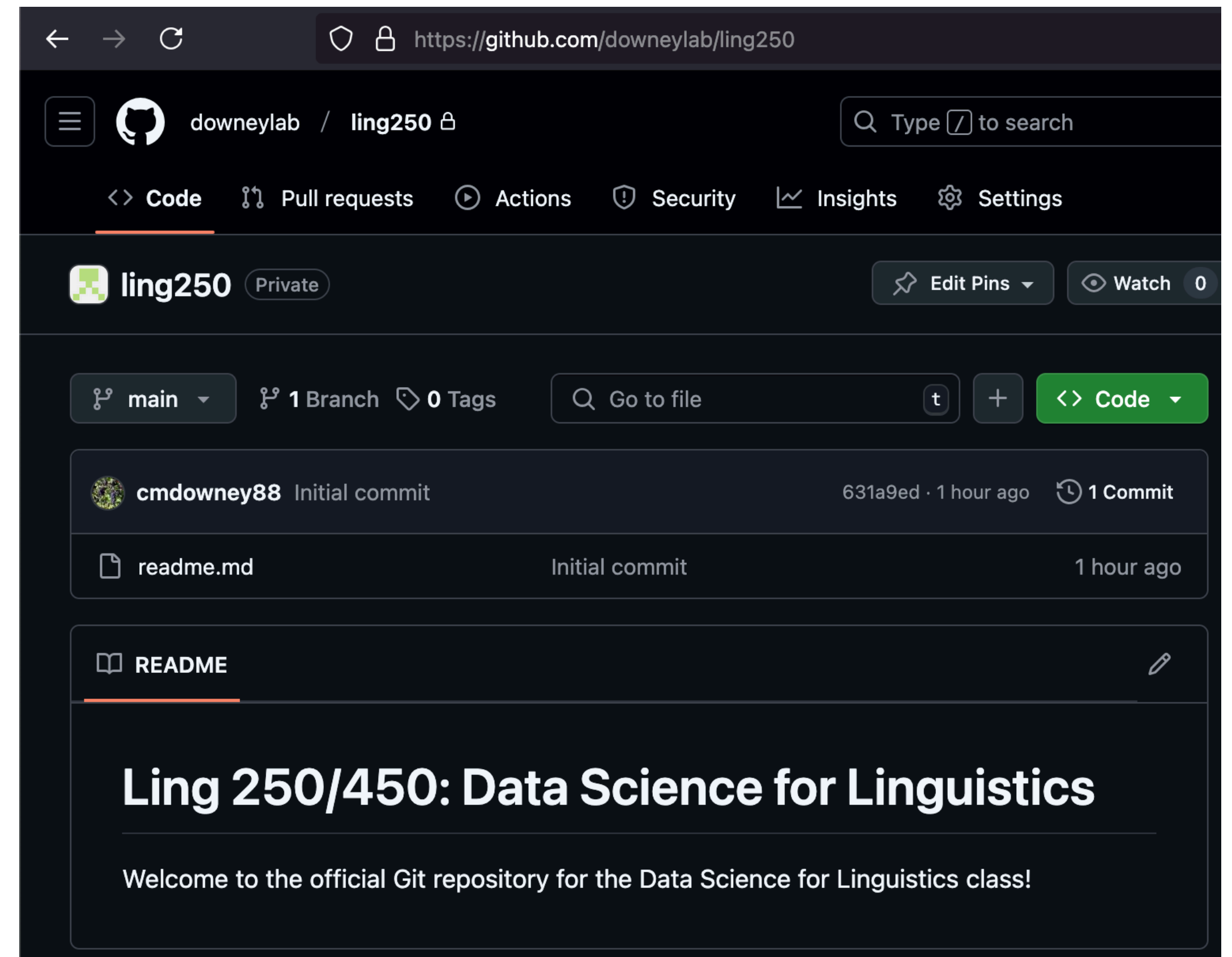
# Example branching structure

## GIT Branch and its Operations.



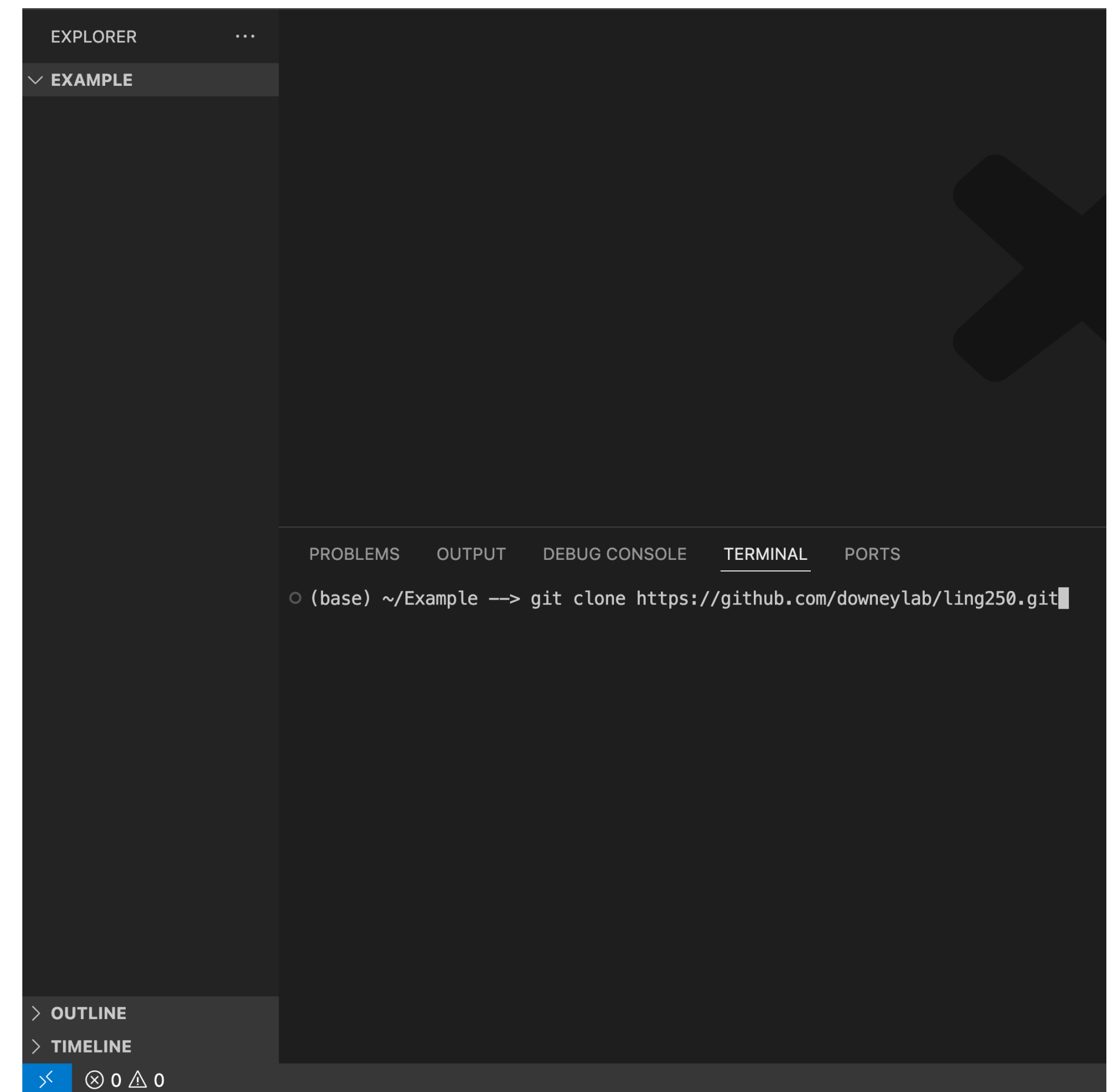
# Repositories

- A repository is simply a **folder of files** that are **tracked by Git**
- A repository can be thought of as a **project**
- Repositories are often made **accessible on Github**
  - Github often serves as the "**central copy**" of the repository
- This class has an **example repository**



# Cloning a repository

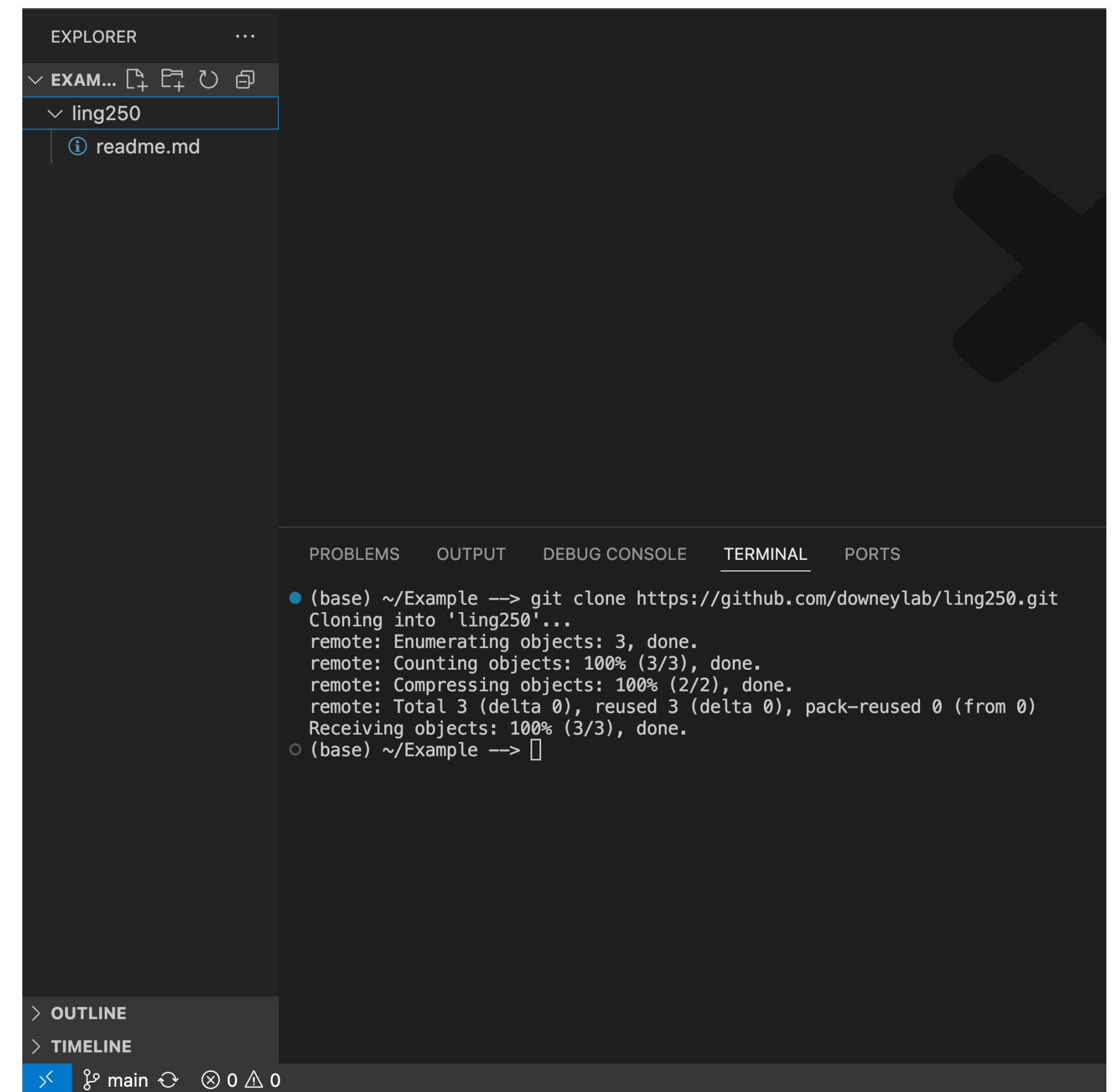
- If you have access, you can **clone** a repository to your computer
  - Done with `git clone <repo_url>`
  - Cloning just means **downloading a copy** to work on
- The cloned copy has the **full history** of the repository





# Cloning a repository

- If you have access, you can **clone** a repository to your computer
- Done with `git clone <repo_url>`
- Cloning just means **downloading a copy** to work on
- The cloned copy has the **full history** of the repository



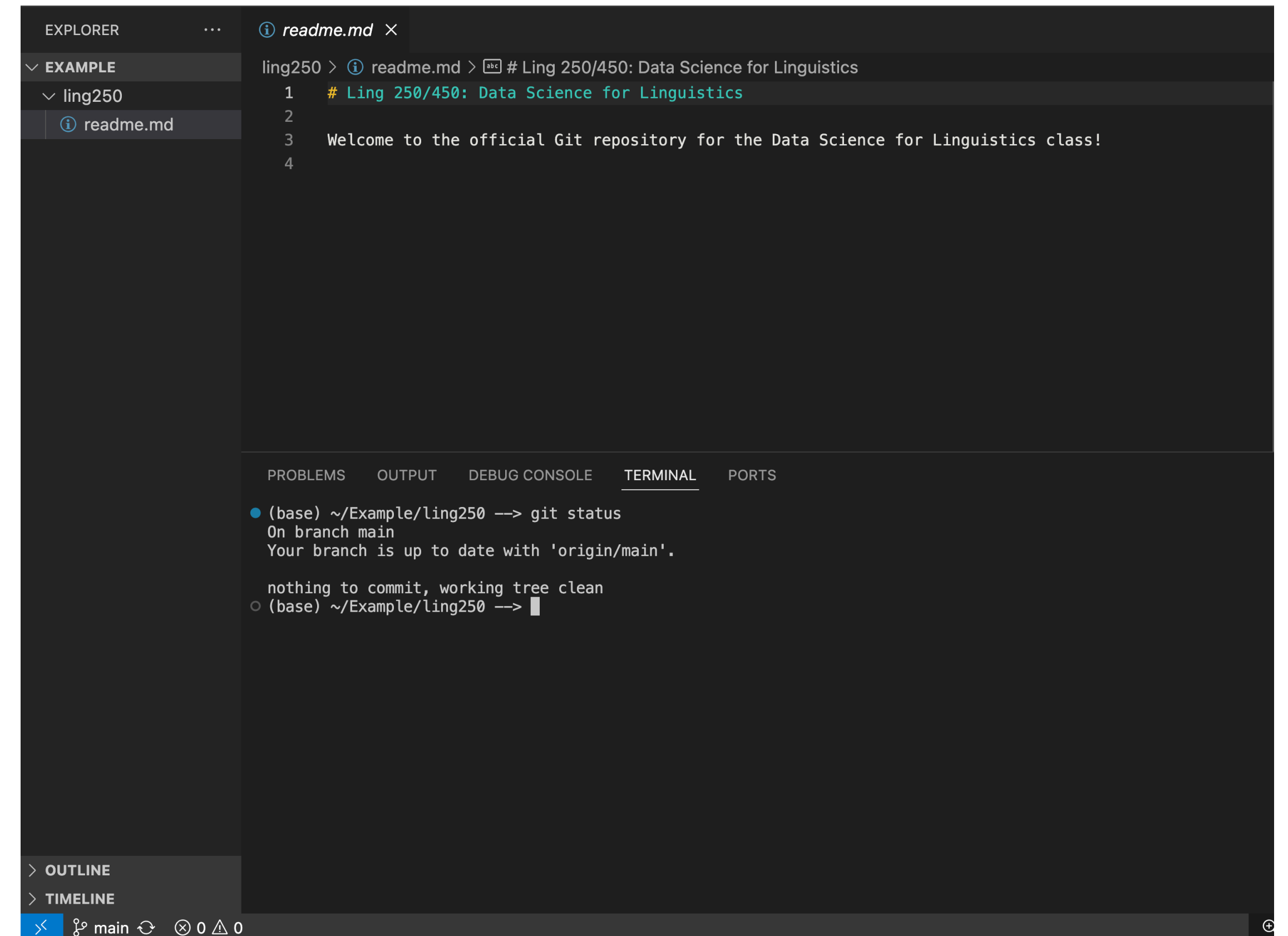
```
EXPLORER
  EXAM...
  ling250
    README.md

TERMINAL
(base) ~/Example --> git clone https://github.com/downeylab/ling250.git
Cloning into 'ling250'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
(base) ~/Example -->
```



# Editing and checking status

- `git status` can be used to **check the status** of the repository
  - Before you change anything, it will say "**working tree clean**"
- You can **freely edit files** (and even press the save button), but it is **not yet saved with Git**
- Git status will call these "**changes not added to commit**"



The screenshot shows the Visual Studio Code interface. The Explorer panel on the left shows a project structure with a folder named 'EXAMPLE' containing a subfolder 'ling250' and a file 'readme.md'. The main editor area displays the content of 'readme.md', which includes a title '# Ling 250/450: Data Science for Linguistics' and a welcome message. Below the editor, the TERMINAL panel is open, showing the output of the command 'git status'. The output indicates that the working tree is clean and the branch is up to date with 'origin/main'.

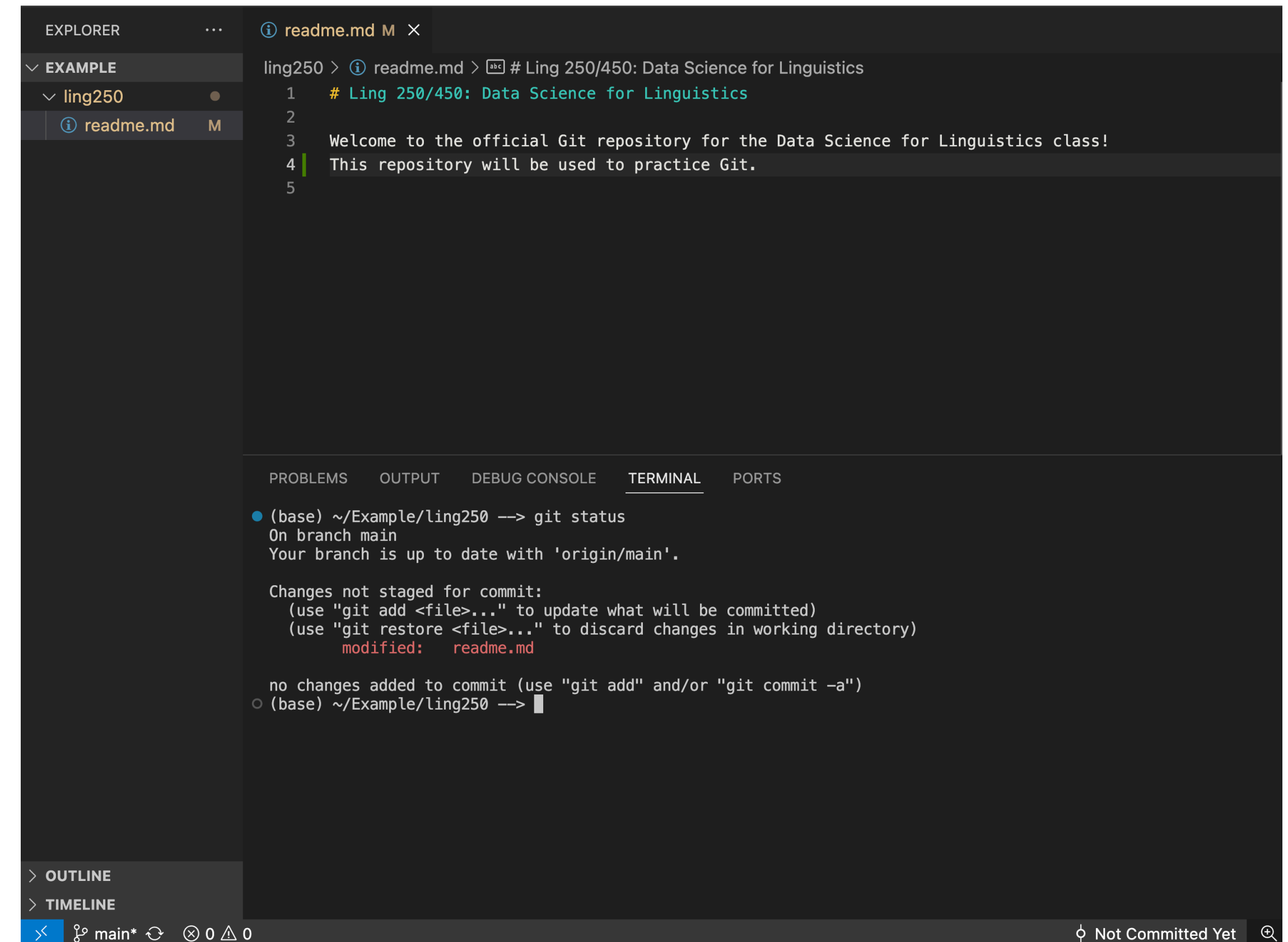
```
ling250 > # Ling 250/450: Data Science for Linguistics
1 # Ling 250/450: Data Science for Linguistics
2
3 Welcome to the official Git repository for the Data Science for Linguistics class!
4

(base) ~/Example/ling250 --> git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
(base) ~/Example/ling250 -->
```

# Editing and checking status

- `git status` can be used to **check the status** of the repository
  - Before you change anything, it will say "**working tree clean**"
- You can **freely edit files** (and even press the save button), but it is **not yet saved with Git**
- Git status will call these "**changes not added to commit**"



The screenshot shows a Visual Studio Code editor with a file explorer on the left showing a project named 'EXAMPLE' with a subdirectory 'ling250' containing a file 'readme.md'. The main editor window shows the content of 'readme.md', which includes a title '# Ling 250/450: Data Science for Linguistics' and a welcome message. Below the editor, the terminal panel is open, displaying the output of the command 'git status'. The output indicates that the working tree is clean and up to date with 'origin/main'. It also shows that there are changes not staged for commit, specifically 'readme.md' which has been modified. The terminal text is as follows:

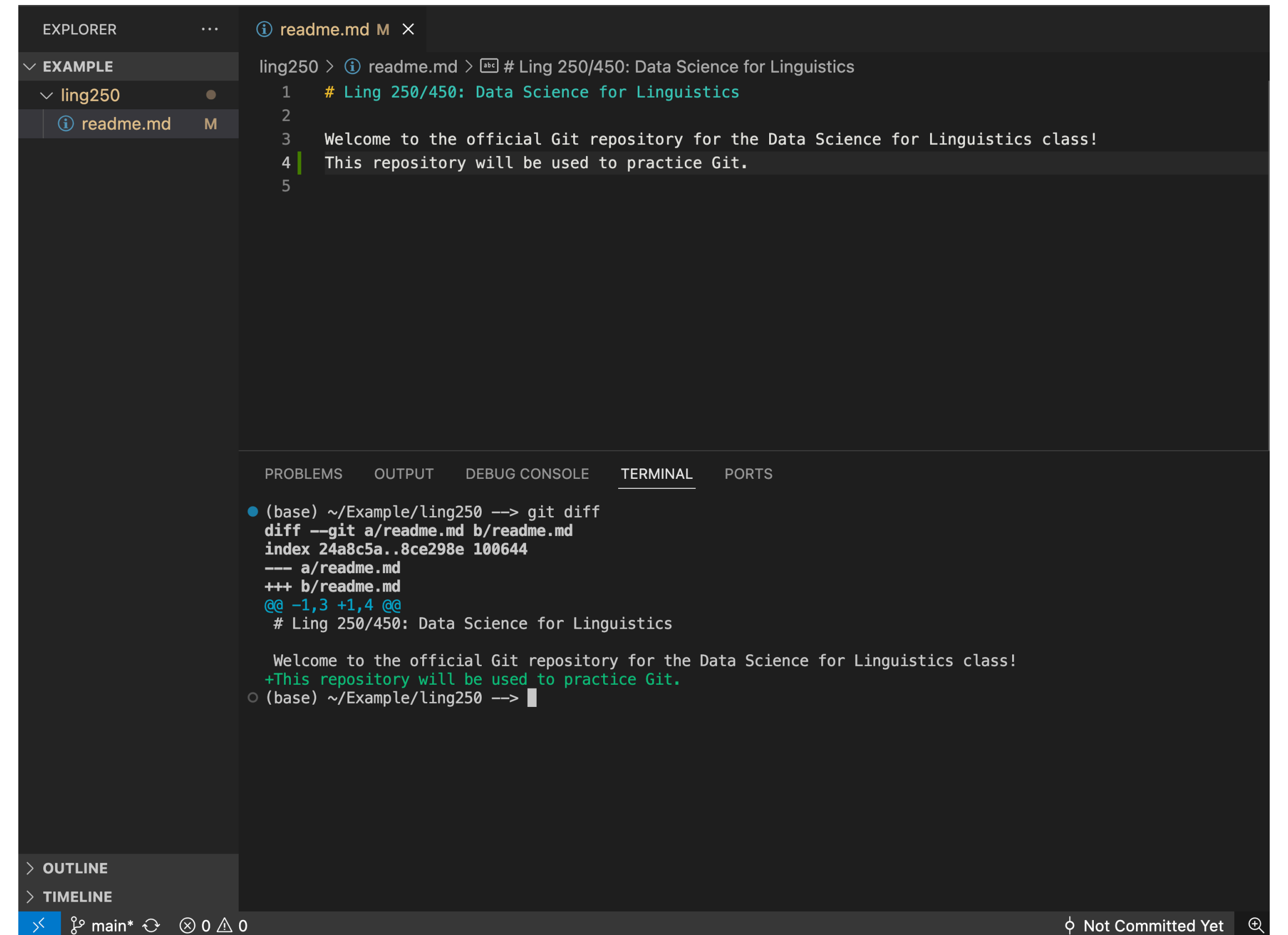
```
(base) ~/Example/ling250 --> git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   readme.md

no changes added to commit (use "git add" and/or "git commit -a")
(base) ~/Example/ling250 -->
```

The status bar at the bottom of the editor shows 'main\*' and '0' changes, and a message 'Not Committed Yet'.

# git add



The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left shows a project structure with a folder named 'ling250' containing a file 'readme.md' that has been modified. The main editor area displays the content of 'readme.md', which includes a title and a welcome message. The bottom panel shows the 'TERMINAL' tab with the output of a 'git diff' command, highlighting the changes made to the file.

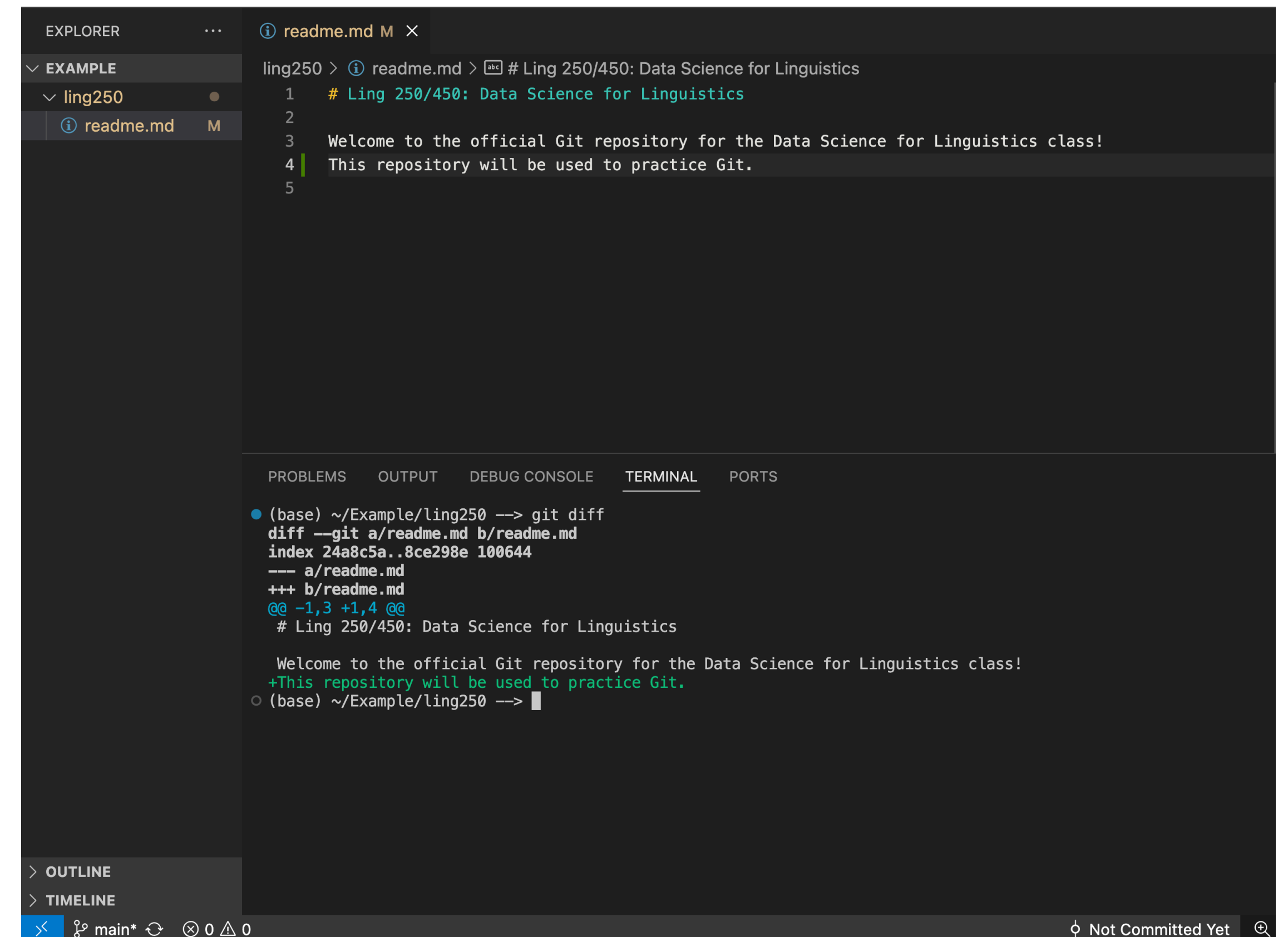
```
ling250 > # Ling 250/450: Data Science for Linguistics
1  # Ling 250/450: Data Science for Linguistics
2
3  Welcome to the official Git repository for the Data Science for Linguistics class!
4  This repository will be used to practice Git.
5
```

```
(base) ~/Example/ling250 --> git diff
diff --git a/readme.md b/readme.md
index 24a8c5a..8ce298e 100644
--- a/readme.md
+++ b/readme.md
@@ -1,3 +1,4 @@
 # Ling 250/450: Data Science for Linguistics

 Welcome to the official Git repository for the Data Science for Linguistics class!
+This repository will be used to practice Git.
(base) ~/Example/ling250 -->
```

# git add

- `git diff` can be used to view **specific changes** from the **last saved version** (called the last **commit**)



The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left shows a project structure with a folder named 'EXAMPLE' containing a subfolder 'ling250', which in turn contains a file 'readme.md' marked as modified (M). The main editor area displays the 'readme.md' file with the following content:

```
1 # Ling 250/450: Data Science for Linguistics
2
3 Welcome to the official Git repository for the Data Science for Linguistics class!
4 This repository will be used to practice Git.
5
```

Below the editor, the TERMINAL panel is active, showing the output of a `git diff` command:

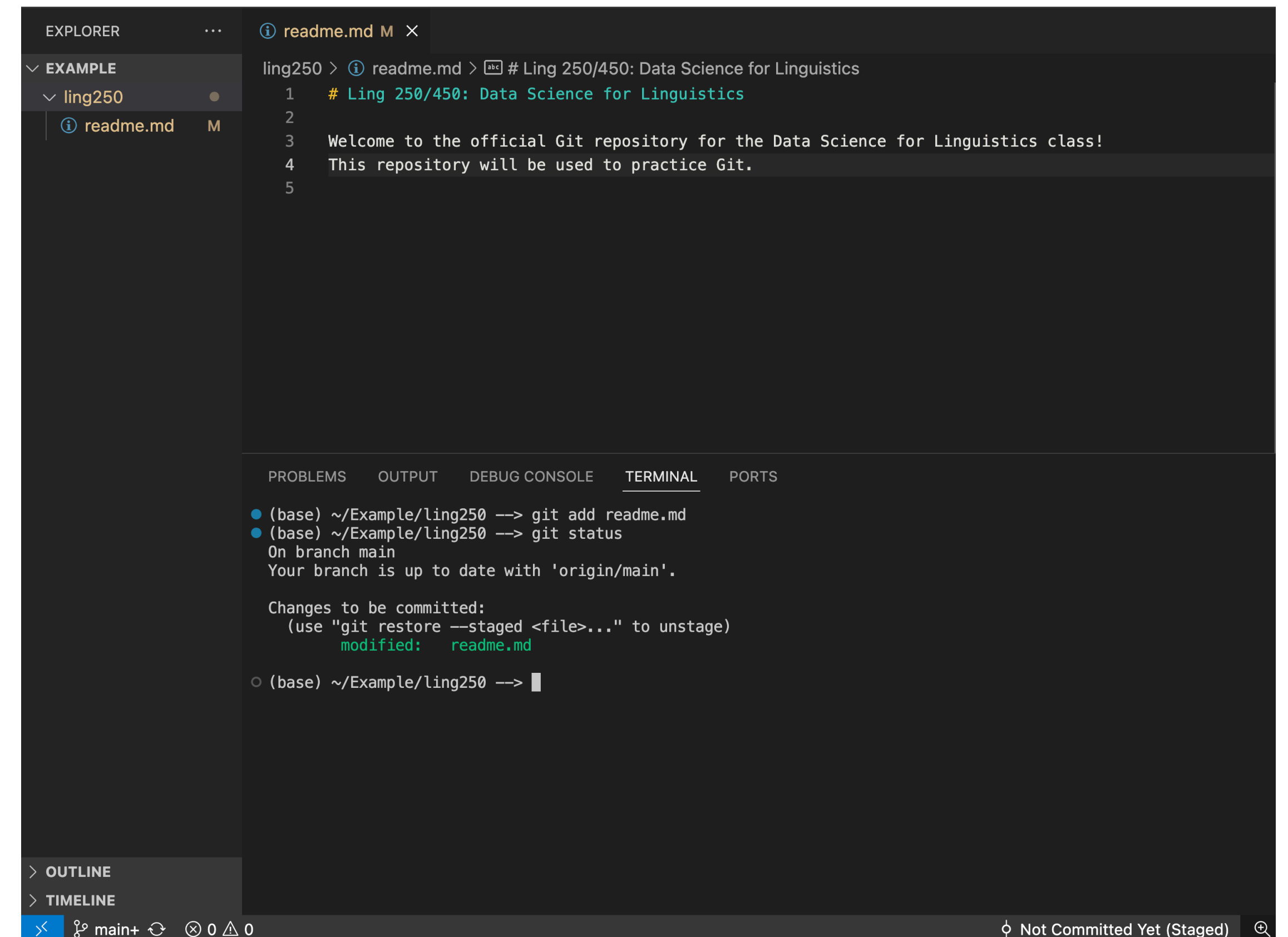
```
(base) ~/Example/ling250 --> git diff
diff --git a/readme.md b/readme.md
index 24a8c5a..8ce298e 100644
--- a/readme.md
+++ b/readme.md
@@ -1,3 +1,4 @@
 # Ling 250/450: Data Science for Linguistics

 Welcome to the official Git repository for the Data Science for Linguistics class!
+This repository will be used to practice Git.
o (base) ~/Example/ling250 -->
```

The status bar at the bottom indicates the current branch is 'main\*' and that there are 0 changes. A message 'Not Committed Yet' is also visible.

# git add

- `git diff` can be used to view **specific changes** from the **last saved version** (called the last **commit**)
- If you're happy with the changes, `git add` will "**stage**" the **file** for committing
- This essentially "**marks**" the **current version** of the file to be saved



The screenshot shows a Visual Studio Code editor with a file explorer on the left showing a project named 'EXAMPLE' with a subdirectory 'ling250' containing a 'readme.md' file. The main editor area shows the content of 'readme.md' with the following text:

```
1 # Ling 250/450: Data Science for Linguistics
2
3 Welcome to the official Git repository for the Data Science for Linguistics class!
4 This repository will be used to practice Git.
5
```

Below the editor, the 'TERMINAL' panel is open, showing the output of the following commands:

```
(base) ~/Example/ling250 --> git add readme.md
(base) ~/Example/ling250 --> git status
On branch main
Your branch is up to date with 'origin/main'.

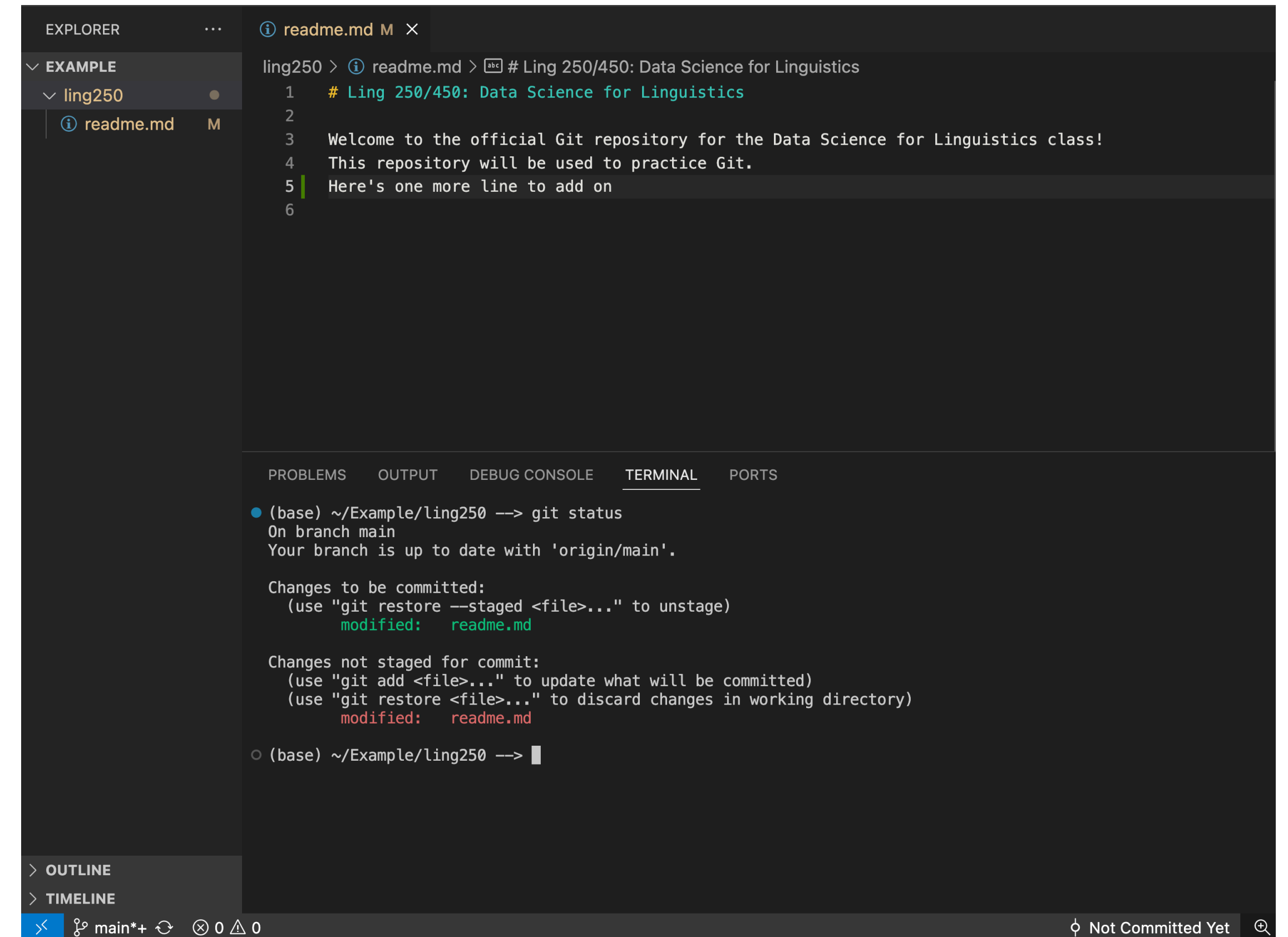
Changes to be committed:
  (use "git restore --staged <file>.." to unstage)
        modified:   readme.md

(base) ~/Example/ling250 -->
```

The status bar at the bottom indicates 'main' and '0' changes, with a message 'Not Committed Yet (Staged)'.

# git add

- `git diff` can be used to view **specific changes** from the **last saved version** (called the last **commit**)
- If you're happy with the changes, `git add` will **"stage" the file** for committing
  - This essentially **"marks" the current version** of the file to be saved
- Changes made **after** `git add` are **not** added automatically!



The screenshot shows a Visual Studio Code editor with a file explorer on the left showing a project named 'EXAMPLE' with a subdirectory 'ling250' containing a file 'readme.md'. The main editor window displays the content of 'readme.md', which includes a title '# Ling 250/450: Data Science for Linguistics' and a welcome message. Below the editor, the 'TERMINAL' panel is open, showing the output of the `git status` command. The output indicates that the repository is on the 'main' branch and is up to date with 'origin/main'. It lists 'Changes to be committed' as 'modified: readme.md' and 'Changes not staged for commit' as 'modified: readme.md'. The status bar at the bottom indicates 'Not Committed Yet'.

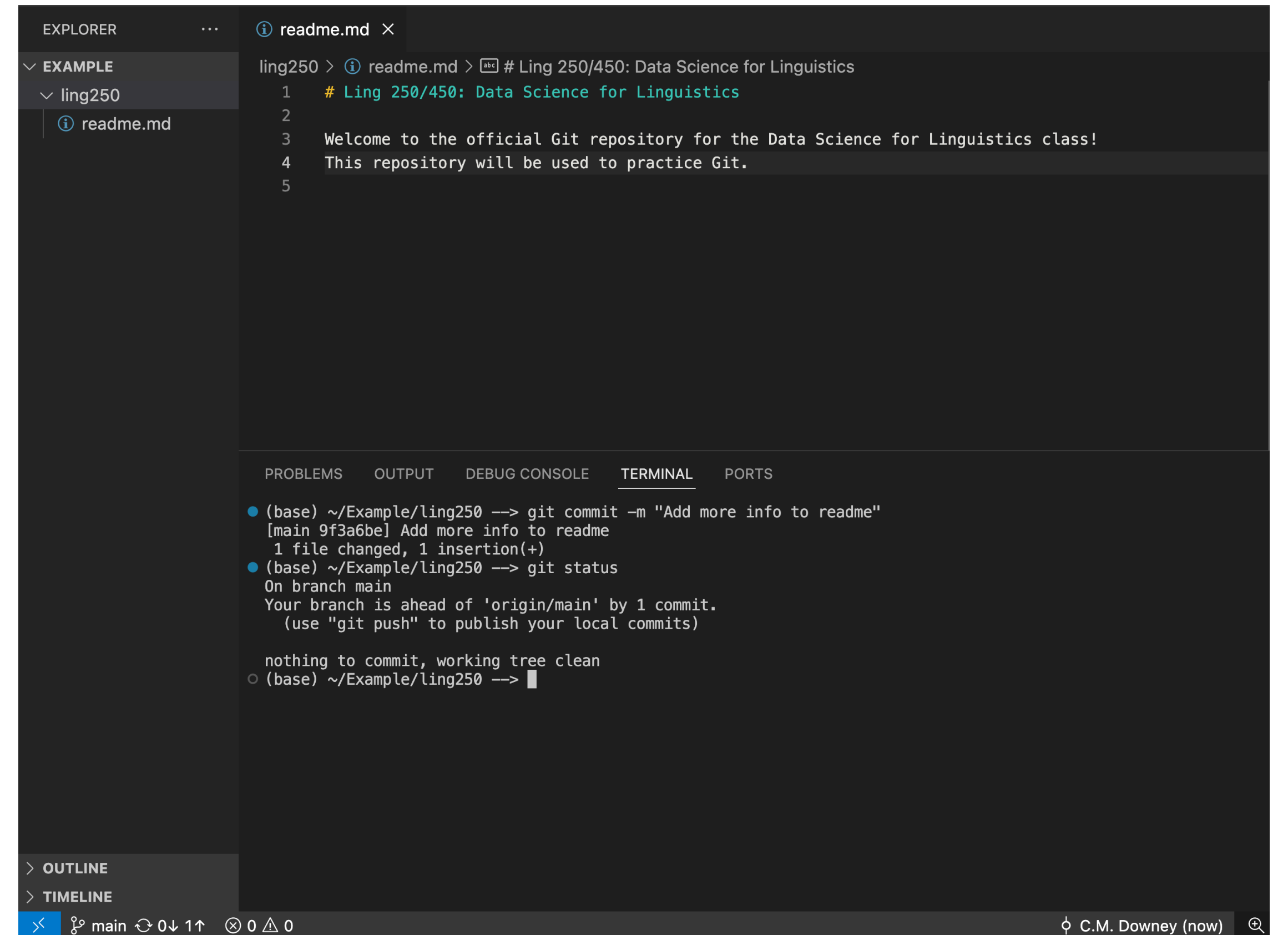
```
ling250 > git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   readme.md

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   readme.md

o (base) ~/Example/ling250 -->
```

# git commit



```
ling250 > # Ling 250/450: Data Science for Linguistics
1  # Ling 250/450: Data Science for Linguistics
2
3  Welcome to the official Git repository for the Data Science for Linguistics class!
4  This repository will be used to practice Git.
5

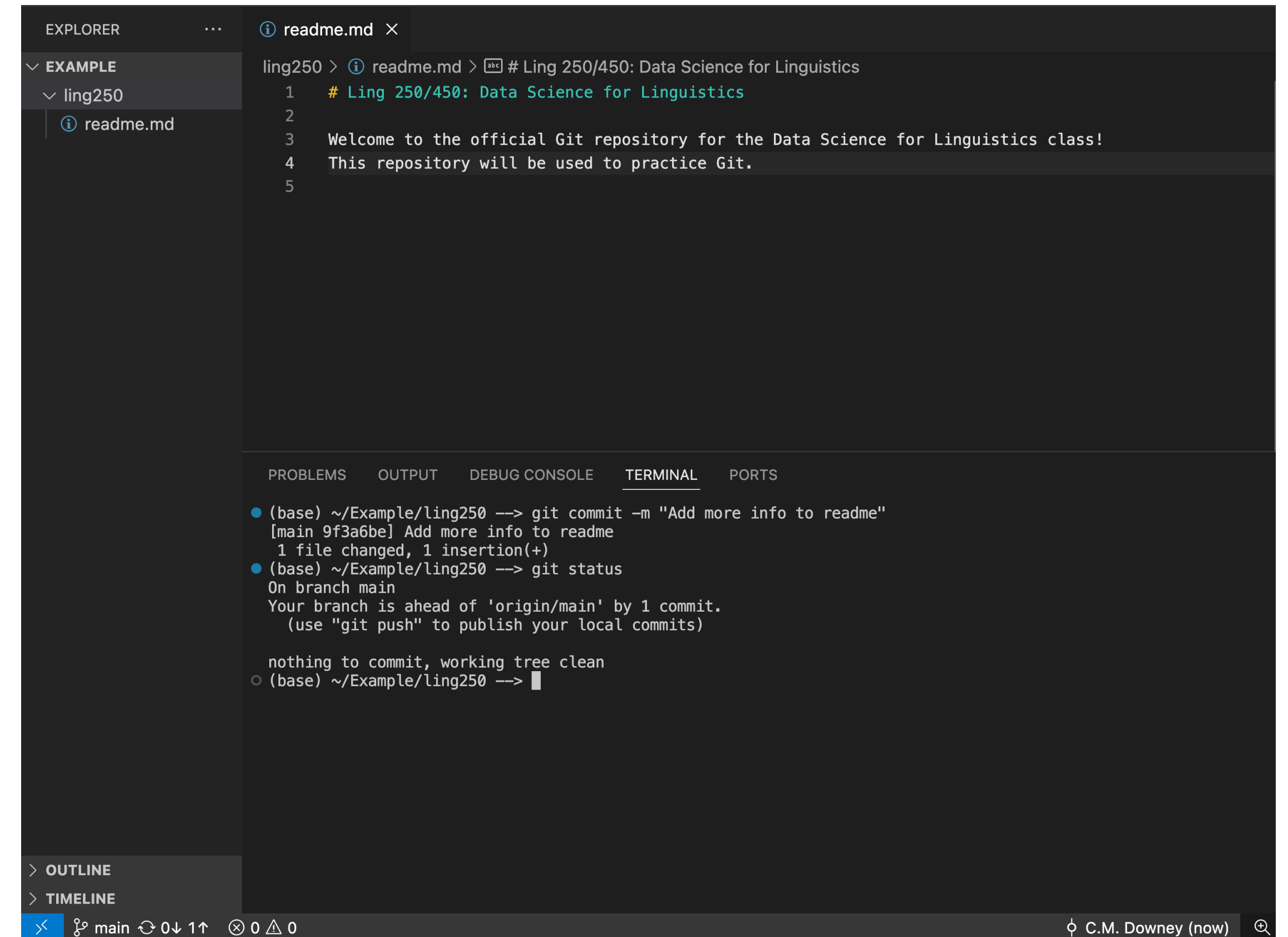
(base) ~/Example/ling250 --> git commit -m "Add more info to readme"
[main 9f3a6be] Add more info to readme
1 file changed, 1 insertion(+)
(base) ~/Example/ling250 --> git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

nothing to commit, working tree clean
(base) ~/Example/ling250 -->
```



# git commit

- git commit **saves** your staged changes as a **reference-able point in history**



The screenshot shows the Visual Studio Code interface with a file explorer on the left, a code editor in the center, and a terminal at the bottom. The file explorer shows a project named 'EXAMPLE' with a subdirectory 'ling250' containing a file 'readme.md'. The code editor shows the content of 'readme.md' with the following text:

```
1 # Ling 250/450: Data Science for Linguistics
2
3 Welcome to the official Git repository for the Data Science for Linguistics class!
4 This repository will be used to practice Git.
5
```

The terminal at the bottom shows the output of the following commands:

```
(base) ~/Example/ling250 --> git commit -m "Add more info to readme"
[main 9f3a6be] Add more info to readme
1 file changed, 1 insertion(+)
(base) ~/Example/ling250 --> git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

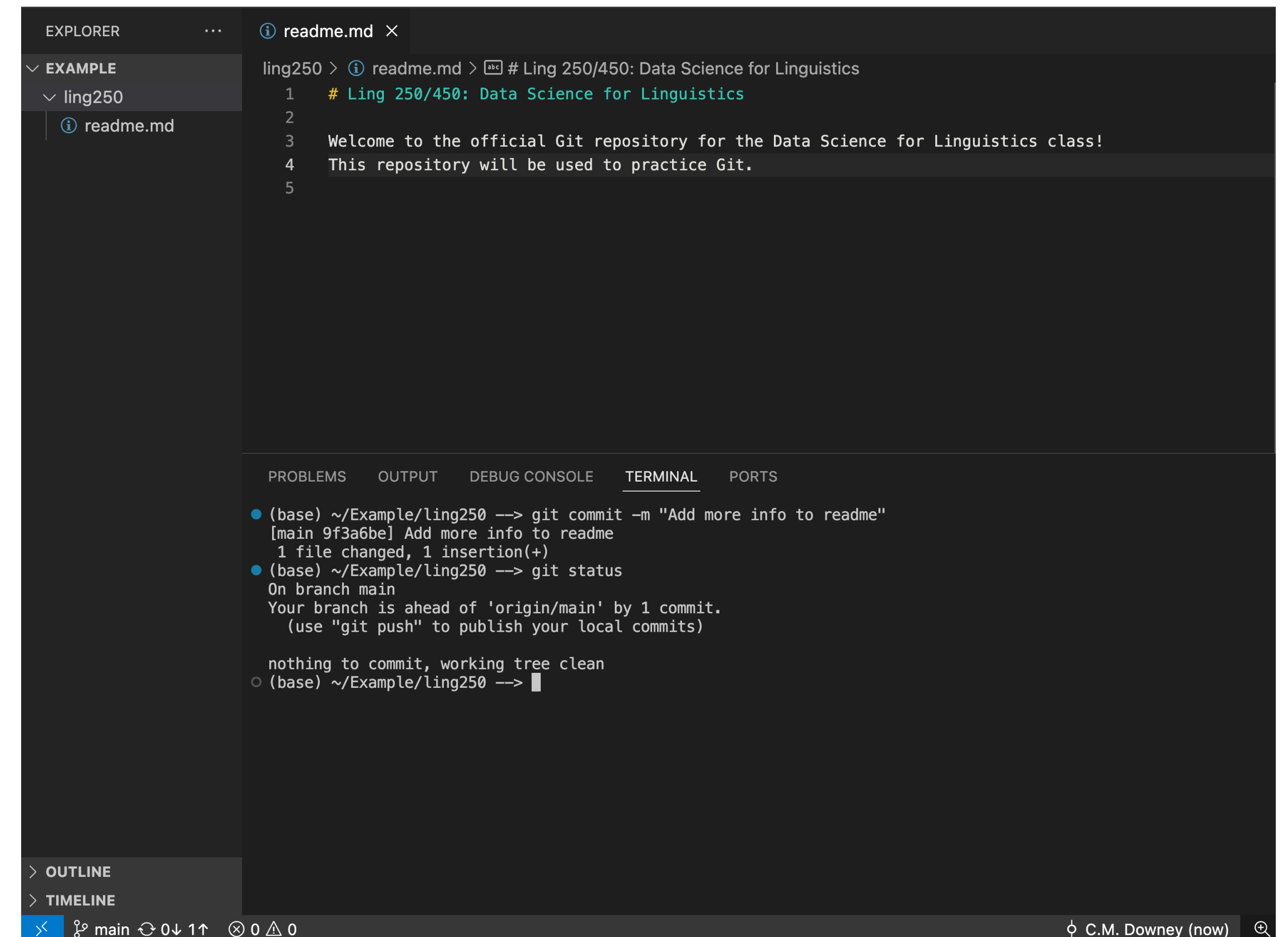
nothing to commit, working tree clean
(base) ~/Example/ling250 -->
```

The status bar at the bottom indicates the current branch is 'main' and shows the commit hash '9f3a6be'.



# git commit

- git commit **saves** your staged changes as a **reference-able point in history**
- I recommend using `git commit -m` to **add a description** of the changes
- If you don't use -m, it will **take to you a text editor called Vim** to write a message anyway. To **escape Vim**, type `:cq!`



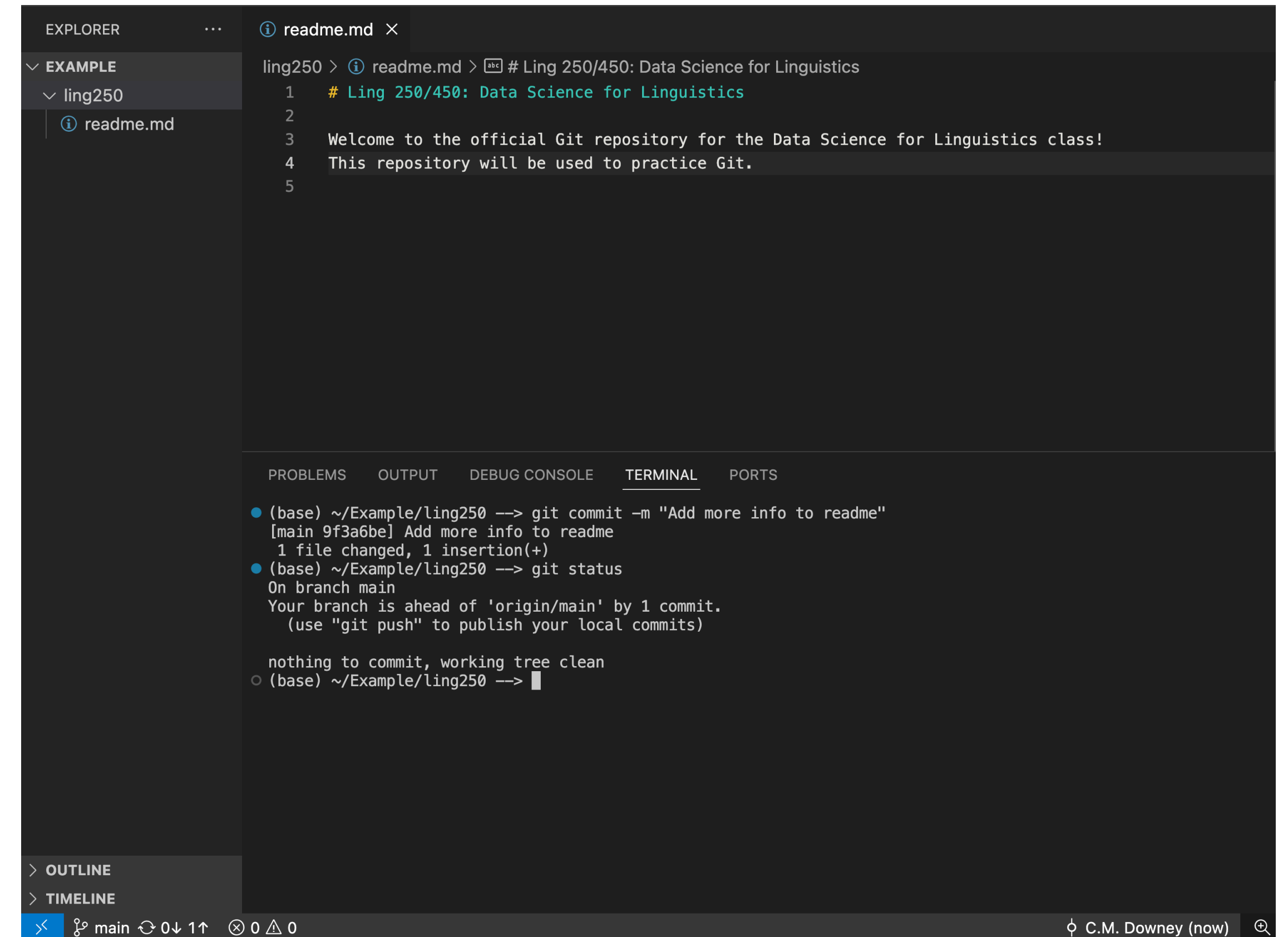
```
ling250 > # Ling 250/450: Data Science for Linguistics
1 # Ling 250/450: Data Science for Linguistics
2
3 Welcome to the official Git repository for the Data Science for Linguistics class!
4 This repository will be used to practice Git.
5
```

```
(base) ~/Example/ling250 --> git commit -m "Add more info to readme"
[main 9f3a6be] Add more info to readme
1 file changed, 1 insertion(+)
(base) ~/Example/ling250 --> git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

nothing to commit, working tree clean
(base) ~/Example/ling250 -->
```

# git commit

- git commit **saves** your staged changes as a **reference-able point in history**
- I recommend using `git commit -m` to **add a description** of the changes
  - If you don't use -m, it will **take to you a text editor called Vim** to write a message anyway. To **escape Vim**, type `:cq!`
- Notice it now says our branch is "**ahead by 1 commit**". This is compared to the **online (Github) version**



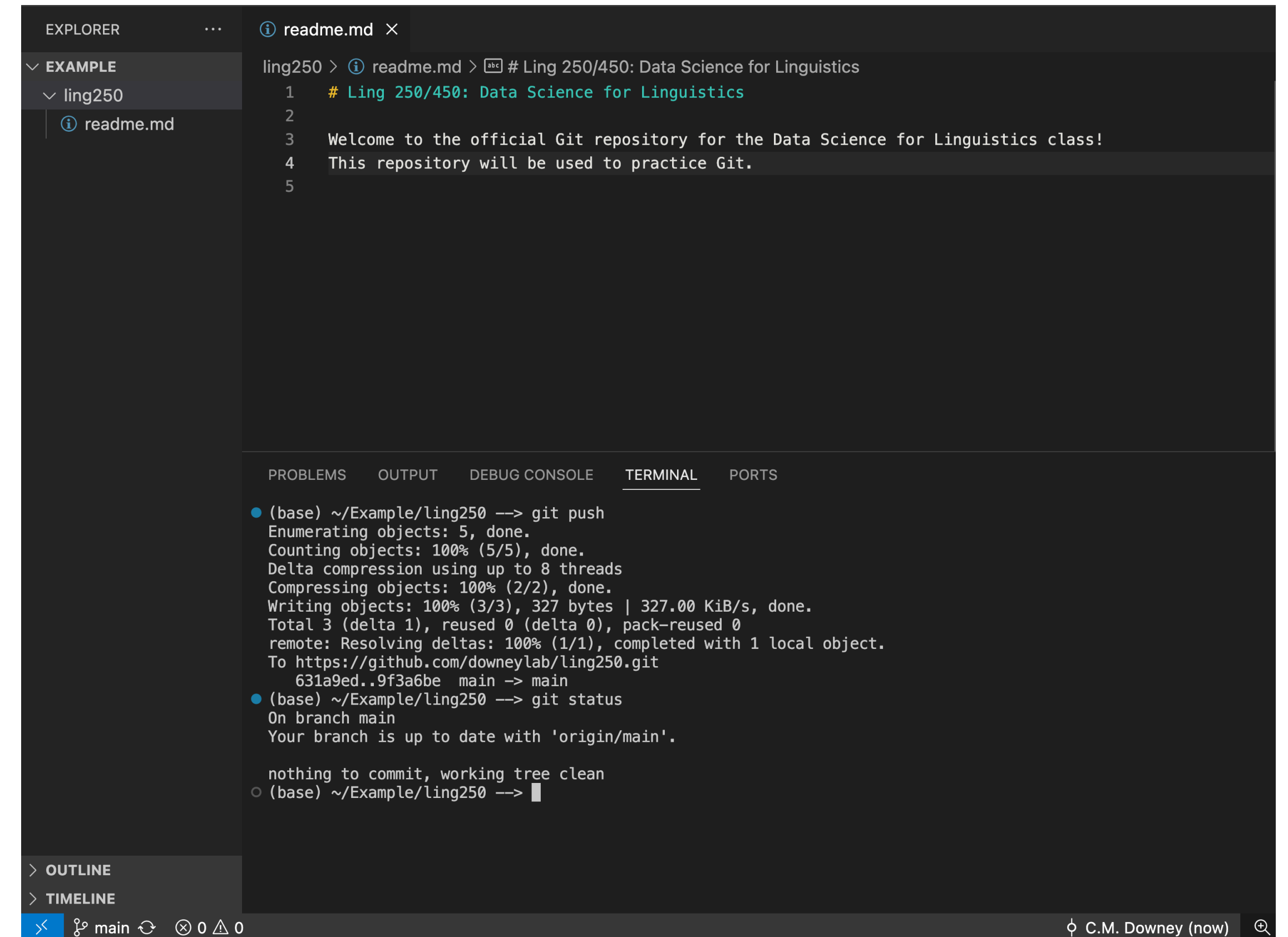
```
ling250 > # Ling 250/450: Data Science for Linguistics
1 # Ling 250/450: Data Science for Linguistics
2
3 Welcome to the official Git repository for the Data Science for Linguistics class!
4 This repository will be used to practice Git.
5

(base) ~/Example/ling250 --> git commit -m "Add more info to readme"
[main 9f3a6be] Add more info to readme
1 file changed, 1 insertion(+)
(base) ~/Example/ling250 --> git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

nothing to commit, working tree clean
(base) ~/Example/ling250 -->
```

# git push

- `git push` **publishes your commits** to the online repository
  - i.e. it makes them available for others
  - The online repository is often called the "**upstream**" version or `origin`
- If the upstream repository has **someone else's commits**, it will **block you** from pushing until you merge those changes with yours (more later)



The screenshot shows a VS Code editor with a file explorer on the left showing a project structure with 'EXAMPLE' and 'ling250' folders, and a 'readme.md' file. The main editor area shows the content of 'readme.md' with a title '# Ling 250/450: Data Science for Linguistics' and a welcome message. Below the editor, the 'TERMINAL' tab is active, displaying the output of a 'git push' command. The output shows the process of enumerating, counting, compressing, and writing objects, followed by a successful push to the remote repository 'https://github.com/downeylab/ling250.git'. The terminal also shows the output of a 'git status' command, indicating that the branch is up to date with 'origin/main'.

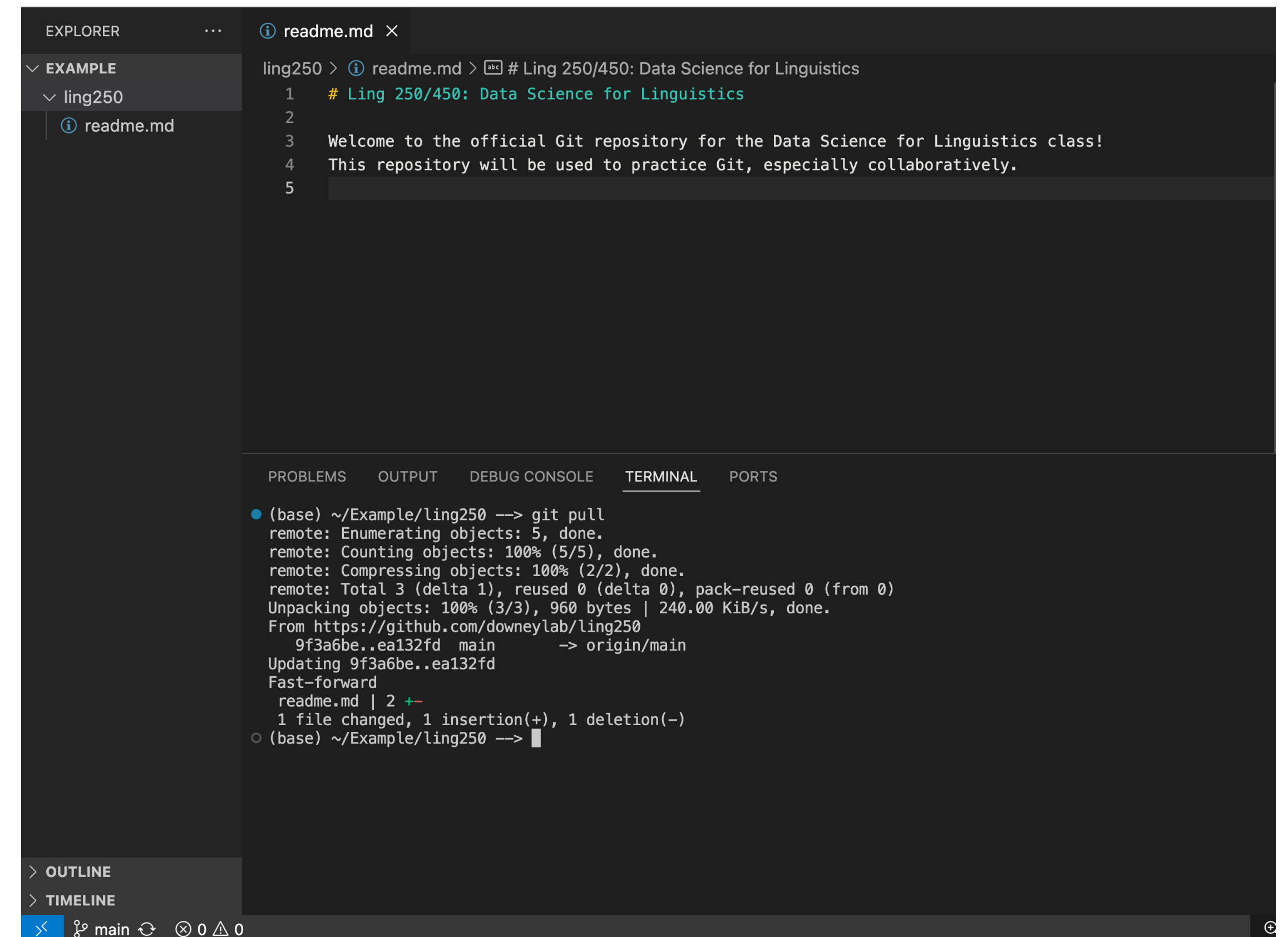
```
ling250 > # Ling 250/450: Data Science for Linguistics
1 # Ling 250/450: Data Science for Linguistics
2
3 Welcome to the official Git repository for the Data Science for Linguistics class!
4 This repository will be used to practice Git.
5

(base) ~/Example/ling250 --> git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 327 bytes | 327.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/downeylab/ling250.git
 631a9ed..9f3a6be  main -> main
(base) ~/Example/ling250 --> git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
(base) ~/Example/ling250 -->
```

# git pull

- `git pull` **downloads new commits** from the upstream repository
- Good practice: always **pull before you start new work!**
- It's easier to work on top of the **most current version** rather than writing **potentially conflicting commits**
- A **rule of thumb** could be to pull whenever `git status` says "**working tree clean**"



The screenshot shows the Visual Studio Code interface. On the left, the Explorer pane shows a project structure with 'EXAMPLE' and 'ling250' folders, and a 'readme.md' file inside 'ling250'. The main editor area shows the 'readme.md' file with the following content:

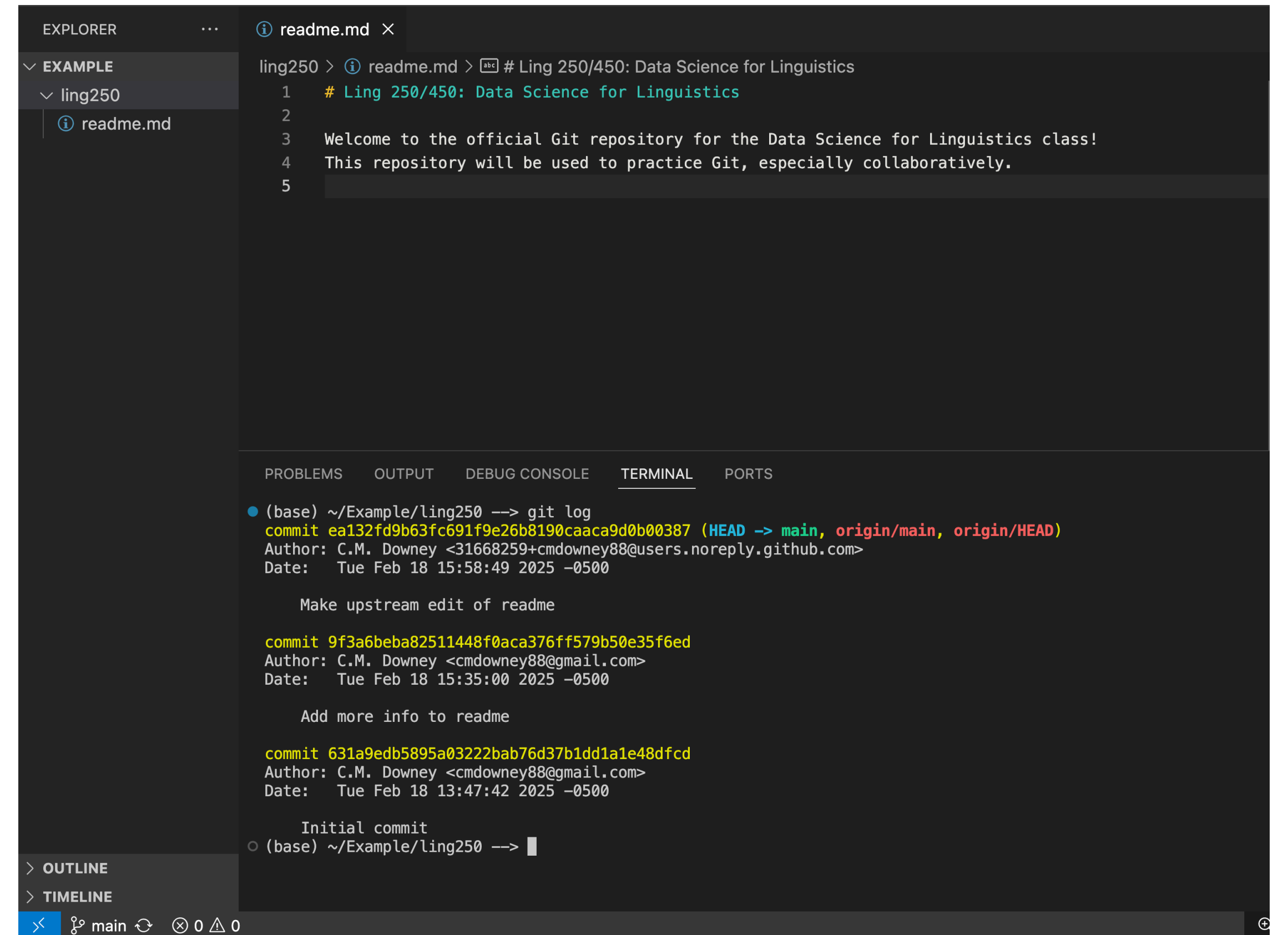
```
1 # Ling 250/450: Data Science for Linguistics
2
3 Welcome to the official Git repository for the Data Science for Linguistics class!
4 This repository will be used to practice Git, especially collaboratively.
5
```

Below the editor, the TERMINAL pane shows the output of a `git pull` command:

```
(base) ~/Example/ling250 --> git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 960 bytes | 240.00 KiB/s, done.
From https://github.com/downeylab/ling250
 9f3a6be..ea132fd  main    -> origin/main
Updating 9f3a6be..ea132fd
Fast-forward
 readme.md | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
(base) ~/Example/ling250 -->
```

The status bar at the bottom indicates the current branch is 'main' and the working tree is clean (0 changes).

# Viewing commit history



The screenshot shows the Visual Studio Code interface. The Explorer panel on the left shows a project structure with 'EXAMPLE' and 'ling250' folders, and a 'readme.md' file. The main editor area shows the 'readme.md' file with the following content:

```
1 # Ling 250/450: Data Science for Linguistics
2
3 Welcome to the official Git repository for the Data Science for Linguistics class!
4 This repository will be used to practice Git, especially collaboratively.
5
```

The bottom panel shows the 'TERMINAL' tab with the following output:

```
(base) ~/Example/ling250 --> git log
commit ea132fd9b63fc691f9e26b8190caaca9d0b00387 (HEAD -> main, origin/main, origin/HEAD)
Author: C.M. Downey <31668259+cmdowney88@users.noreply.github.com>
Date: Tue Feb 18 15:58:49 2025 -0500

    Make upstream edit of readme

commit 9f3a6beba82511448f0aca376ff579b50e35f6ed
Author: C.M. Downey <cmdowney88@gmail.com>
Date: Tue Feb 18 15:35:00 2025 -0500

    Add more info to readme

commit 631a9edb5895a03222bab76d37b1dd1a1e48dfcd
Author: C.M. Downey <cmdowney88@gmail.com>
Date: Tue Feb 18 13:47:42 2025 -0500

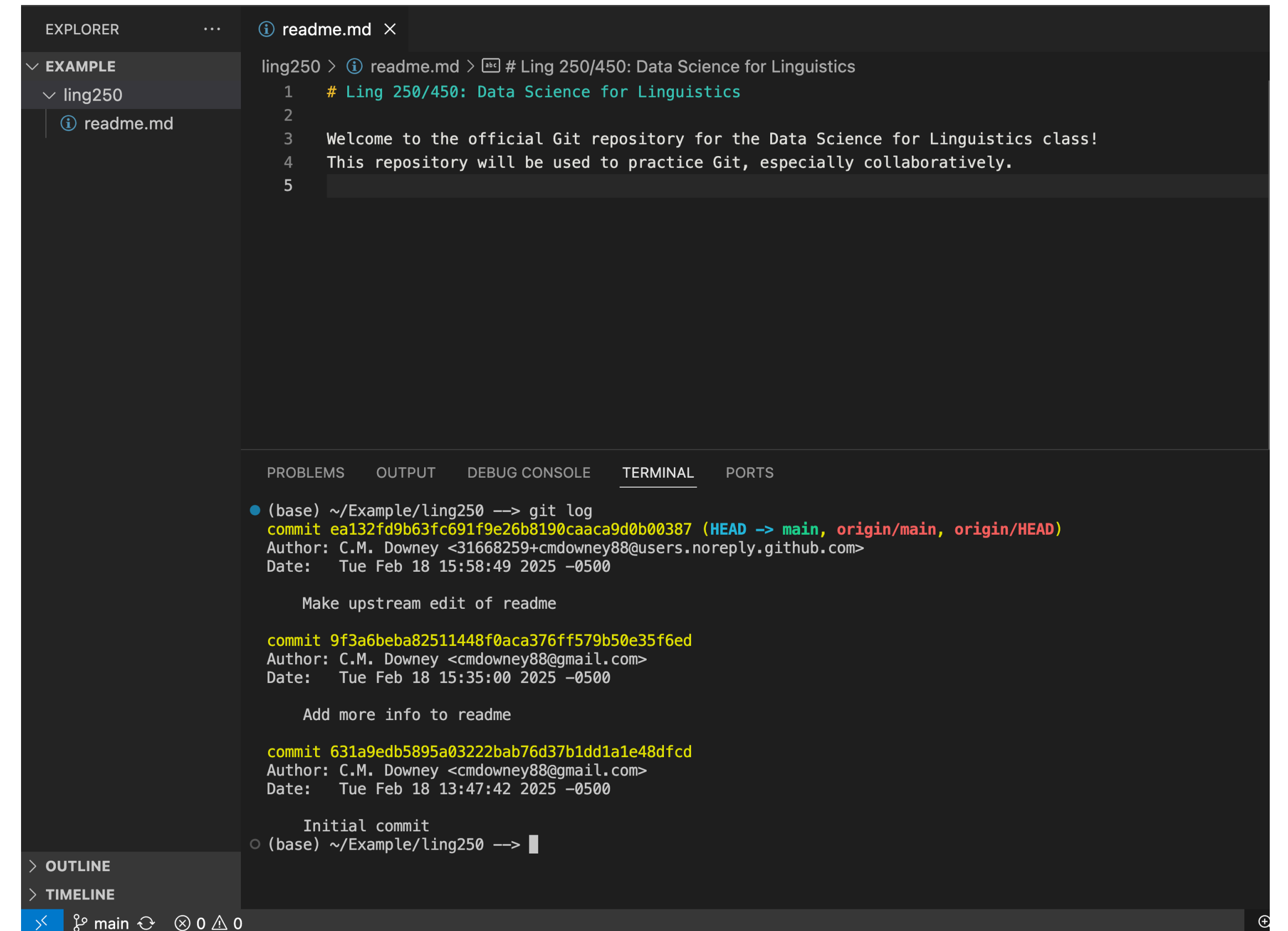
    Initial commit
(base) ~/Example/ling250 -->
```

The status bar at the bottom shows 'main' branch with 0 changes.



# Viewing commit history

- `git log` will list the **commit history** in the **command line**



The screenshot shows a Visual Studio Code editor with a file explorer on the left and a terminal window at the bottom. The file explorer shows a project named 'EXAMPLE' with a subdirectory 'ling250' containing a 'readme.md' file. The terminal window shows the output of the `git log` command, listing three commits in reverse chronological order. The first commit is the initial commit, followed by two subsequent commits where the README file was updated. The terminal output is as follows:

```
(base) ~/Example/ling250 --> git log
commit ea132fd9b63fc691f9e26b8190caaca9d0b00387 (HEAD -> main, origin/main, origin/HEAD)
Author: C.M. Downey <31668259+cmdowney88@users.noreply.github.com>
Date: Tue Feb 18 15:58:49 2025 -0500

    Make upstream edit of readme

commit 9f3a6beba82511448f0aca376ff579b50e35f6ed
Author: C.M. Downey <cmdowney88@gmail.com>
Date: Tue Feb 18 15:35:00 2025 -0500

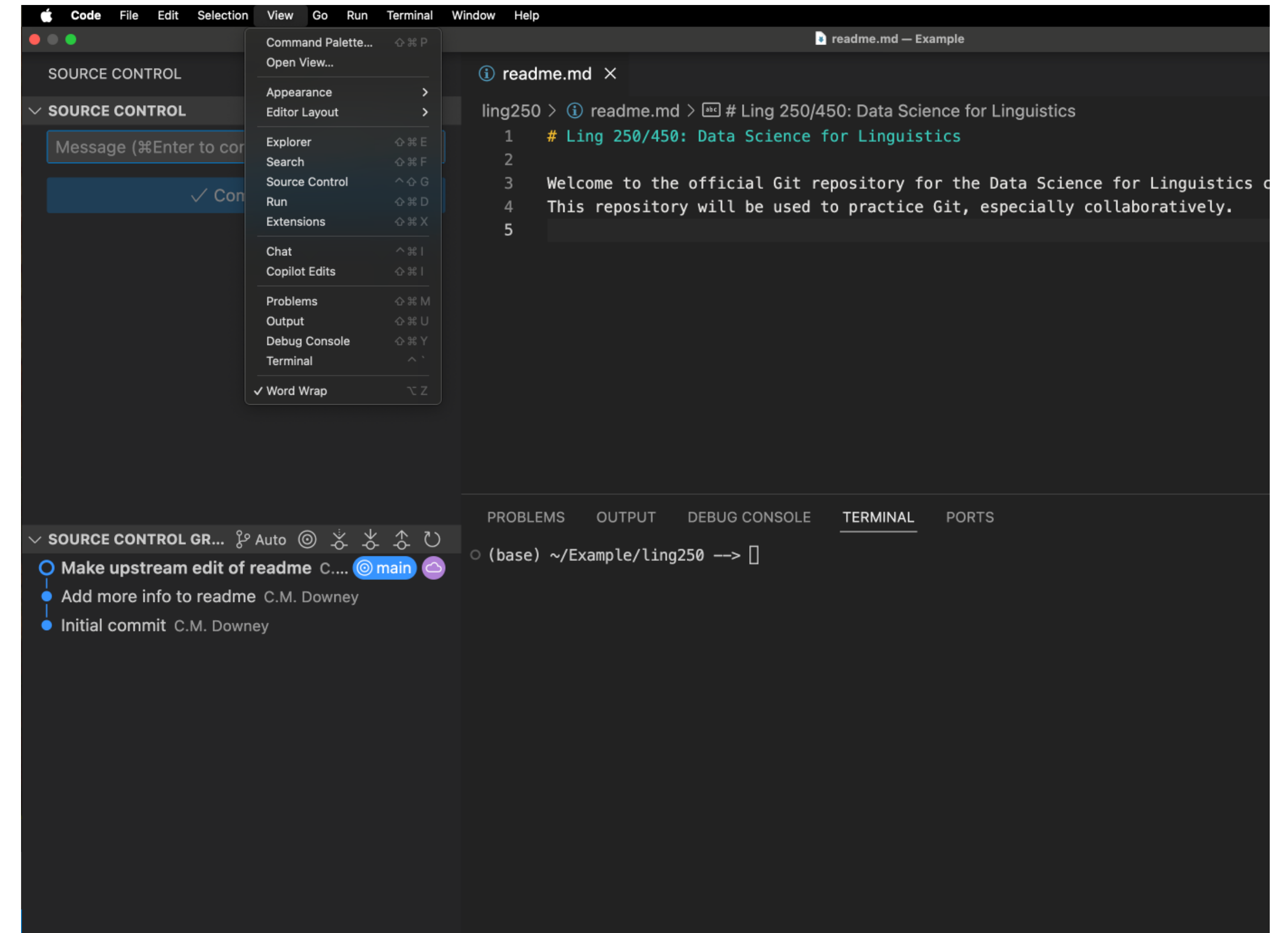
    Add more info to readme

commit 631a9edb5895a03222bab76d37b1dd1a1e48dfcd
Author: C.M. Downey <cmdowney88@gmail.com>
Date: Tue Feb 18 13:47:42 2025 -0500

    Initial commit
○ (base) ~/Example/ling250 --> |
```

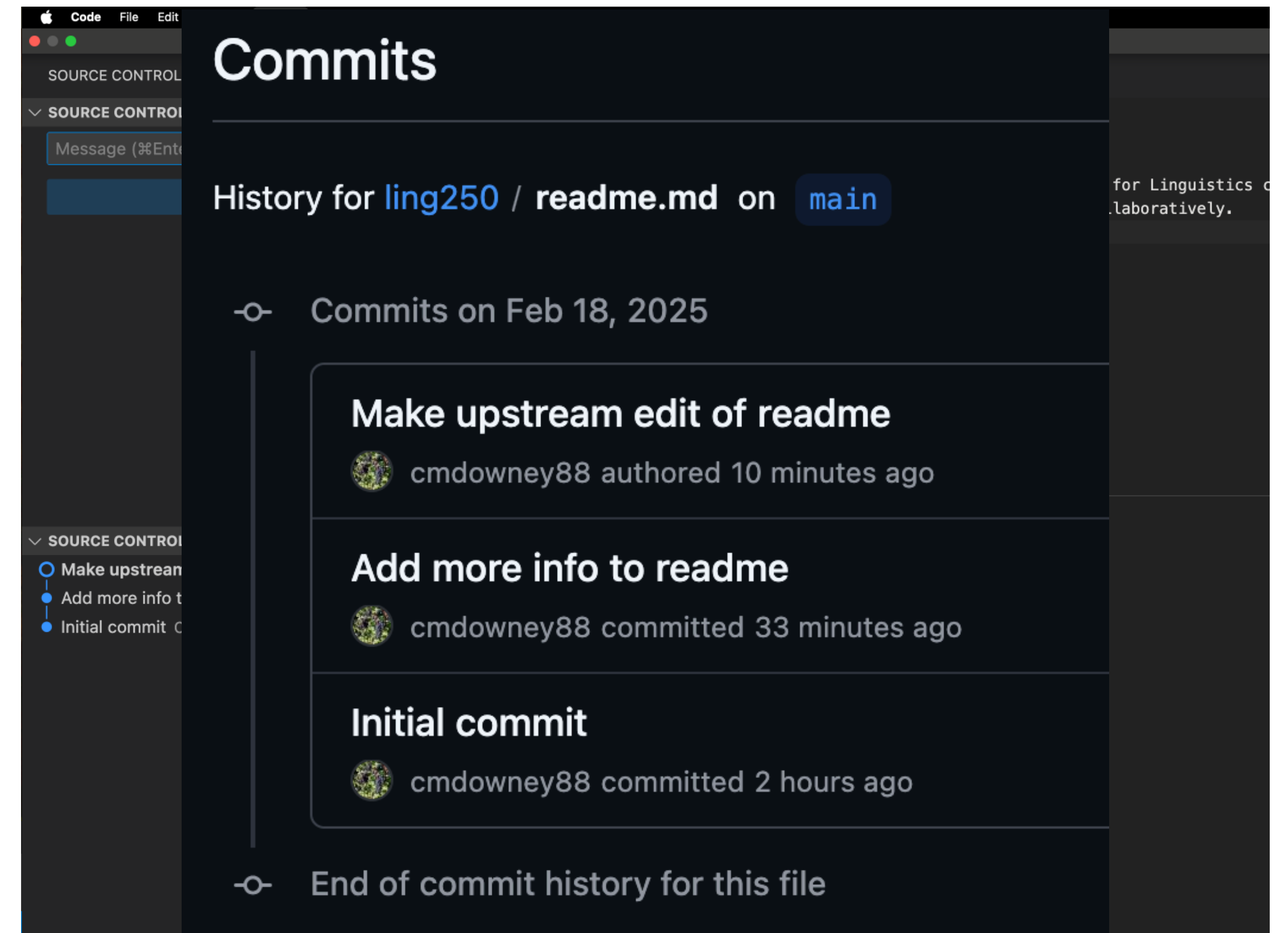
# Viewing commit history

- `git log` will list the **commit history** in the **command line**
- Clicking View > Source Control in VSCode will open a **timeline view**
- **Warning:** this will also open some **graphical interface buttons** for Git, which I **recommend against** when you're first learning Git



# Viewing commit history

- `git log` will list the **commit history** in the **command line**
- Clicking View > Source Control in VSCode will open a **timeline view**
  - **Warning:** this will also open some **graphical interface buttons** for Git, which I **recommend against** when you're first learning Git
- Commit history can also be **viewed on Github**





# General tips for committing

# General tips for committing

- **Commit often!!**
  - "Commit" sounds serious, but think of it as **saving your work**
  - A **common pitfall** is to try to make every commit refined
  - Committing often allows Git to **reverse buggy changes** without losing other work

# General tips for committing

- **Commit often!!**
  - "Commit" sounds serious, but think of it as **saving your work**
  - A **common pitfall** is to try to make every commit refined
  - Committing often allows Git to **reverse buggy changes** without losing other work
- Break changes to **separate files** into **separate commits**
  - (unless the changes are **closely tied**)
  - You can use `git add` to only add one file to be committed at a time

# Commit messages

# Commit messages

- Make your commit messages **informative!!**
  - Uninformative messages are a **bad habit**. Some common examples:  
"debugged", "fixes", "changes", "latest version", "update"
  - The message should **concisely** but **concretely** indicate **what was changed**
  - This is another reason to prefer **smaller, frequent commits**

# Commit messages

- Make your commit messages **informative!!**
  - Uninformative messages are a **bad habit**. Some common examples:  
"debugged", "fixes", "changes", "latest version", "update"
  - The message should **concisely** but **concretely** indicate **what was changed**
  - This is another reason to prefer **smaller, frequent commits**
- Possible exception: solving **mysterious bugs**
  - I.e. "I have no idea what the problem was but this solved it"
  - The message should still indicate **what part** of the code was debugged