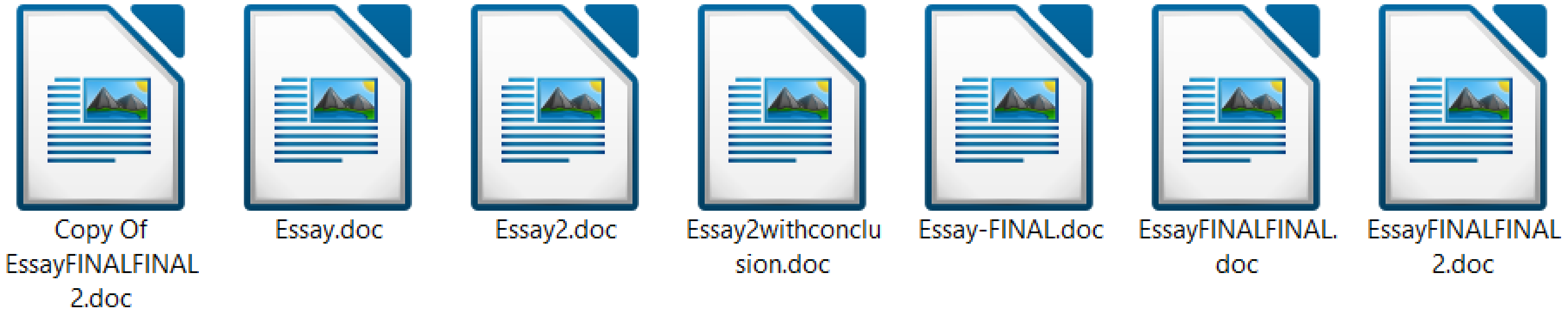
git good

Learn to use git for version control

A UWCS Talk

Has this ever happened to you?



What’s Wrong With This?

- Have to maintain several copies of near identical files
- A hassle to manage manually
  - Multiple files make this even worse
- How do multiple people work on it?
- Even worse if you have multiple files in your project!

Although…

- We care about the history of files sometimes
  - But only specific points
  - We want to be able to find those points easily
- We want to experiment with changes
  - And know we have a safe way to revert
- Tracking changes is important for team projects

Version Control

- Software that tracks changes to source code
- Also called “source control management”
- Examples include SVN, Mercurial, Perforce
  - …and git

Why git?

- git is really **very** popular
  - 85-90% market share
  - Used to manage these projects: the Linux Kernel, Android, Emacs, MySQL, PostgreSQL, Qt, Ruby, Wine, LibreOffice, MediaWiki, jQuery, Django...
  - Used (to at least some extent) by: Apple, Amazon, Microsoft, Meta, Google...

About git

- Created by Linus Torvalds
- Free, open-source
- 17.5 years old
- **https://git-scm.com/**

Aim of This Talk

Give you a basic understanding of git

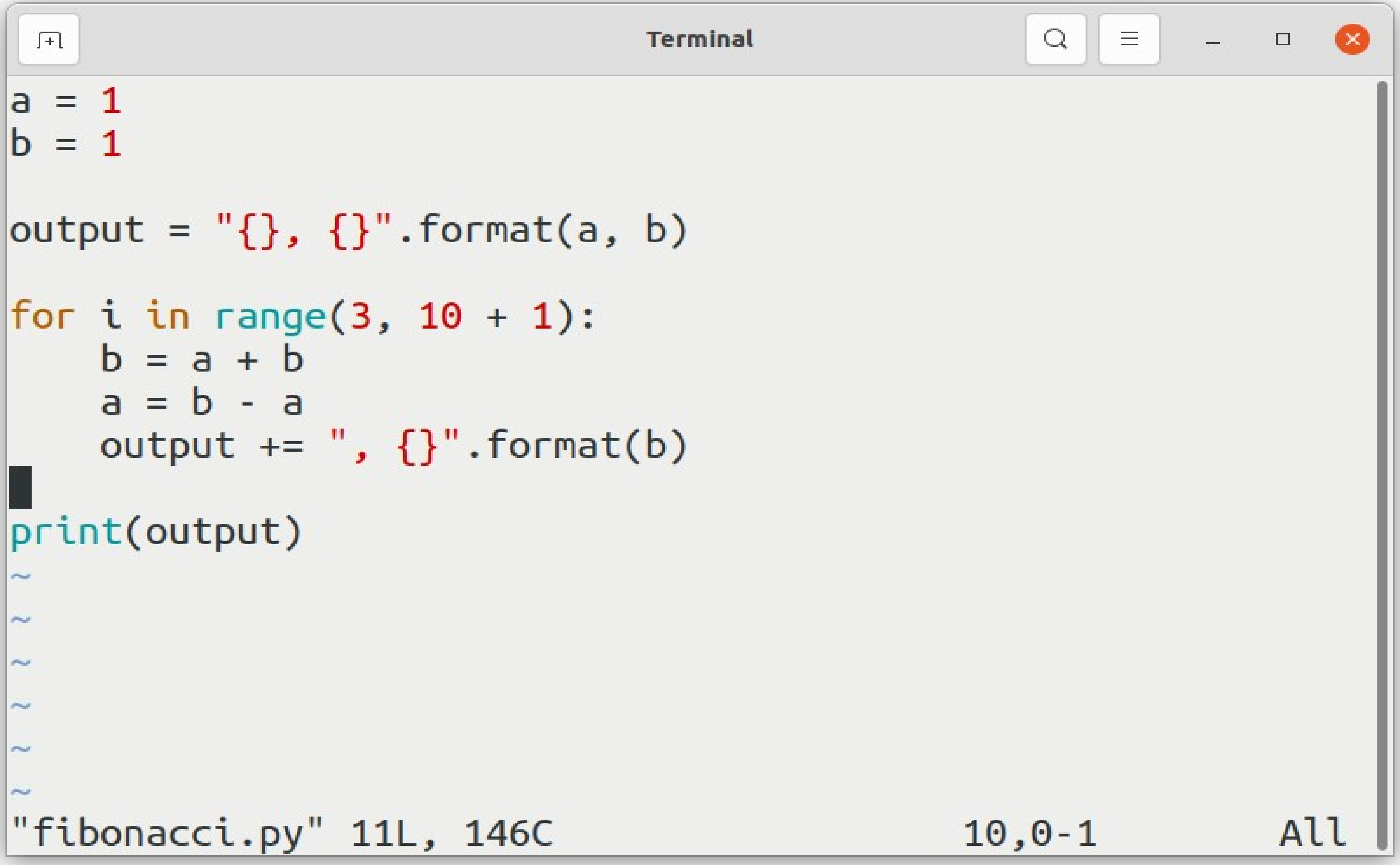
Convince you that git is a skill worth practicing

Command Line

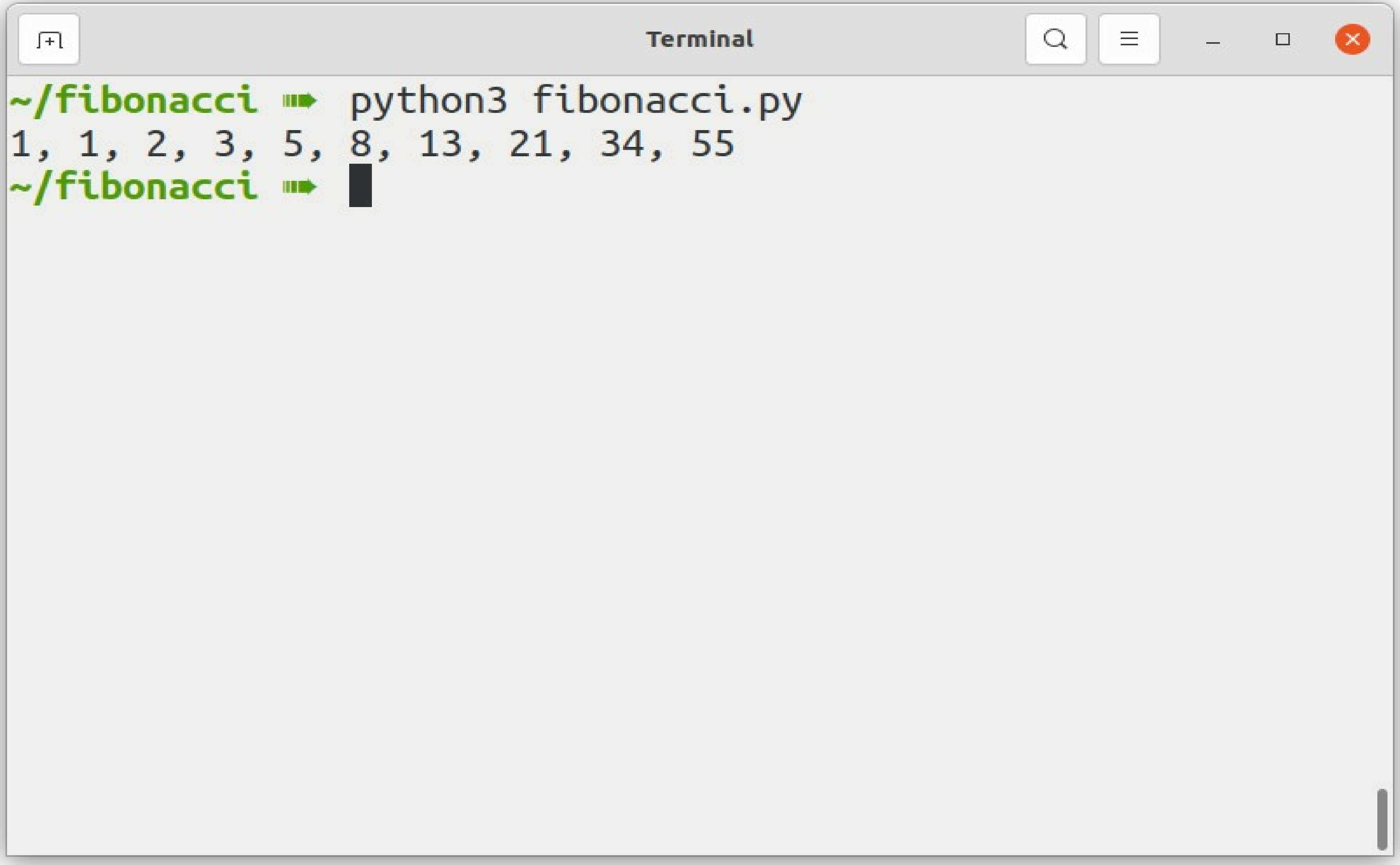
- Git uses subcommands
  - git **<subcommand>** <arguments>
- Each command has many flags
  - man git **<subcommand>**

Repositories

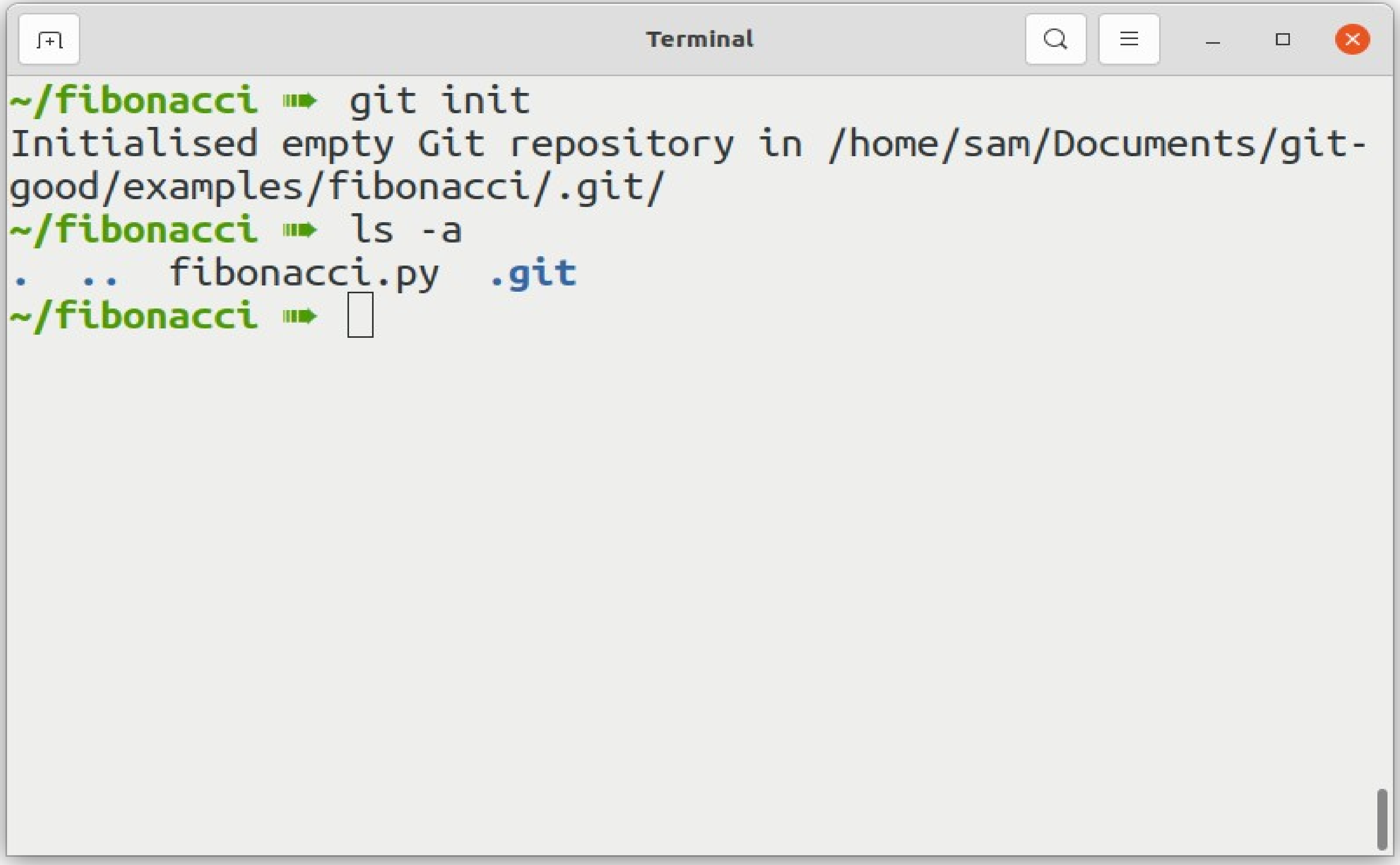
- The **init** subcommand makes a new repository
  - Also called a “repo”
- A repository is a folder being managed by git
  - Think of it as a project
- All information that git needs is stored in the .git folder



```
a = 1
b = 1

output = "{}, {}".format(a, b)

for i in range(3, 10 + 1):
    b = a + b
    a = b - a
    output += ", {}".format(b)

print(output)
~
~
~
~
~
~
"fibonacci.py" 11L, 146C                          10,0-1        All
```



```
~/fibonacci ➠  python3 fibonacci.py
1, 1, 2, 3, 5, 8, 13, 21, 34, 55
~/fibonacci ➠
```



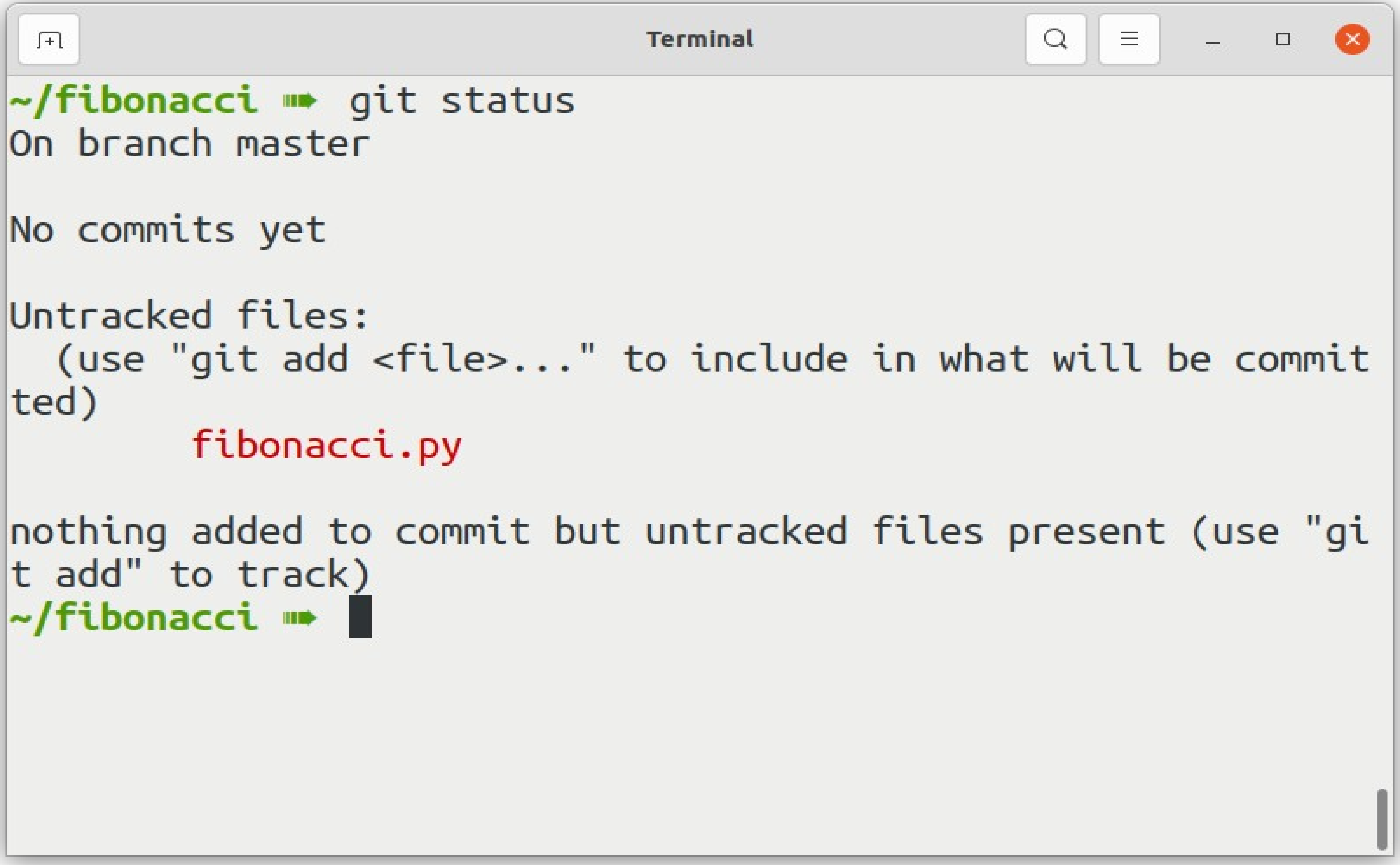
```
~/fibonacci ➠ git init
Initialised empty Git repository in /home/sam/Documents/git-
good/examples/fibonacci/.git/
~/fibonacci ➠ ls -a
.  ..  fibonacci.py  .git
~/fibonacci ➠ 
```

Commits I: What

- A commit is a snapshot of history
- Commits have “names”, which are just hashes
  - Can be referred to by that hash

Working Directory

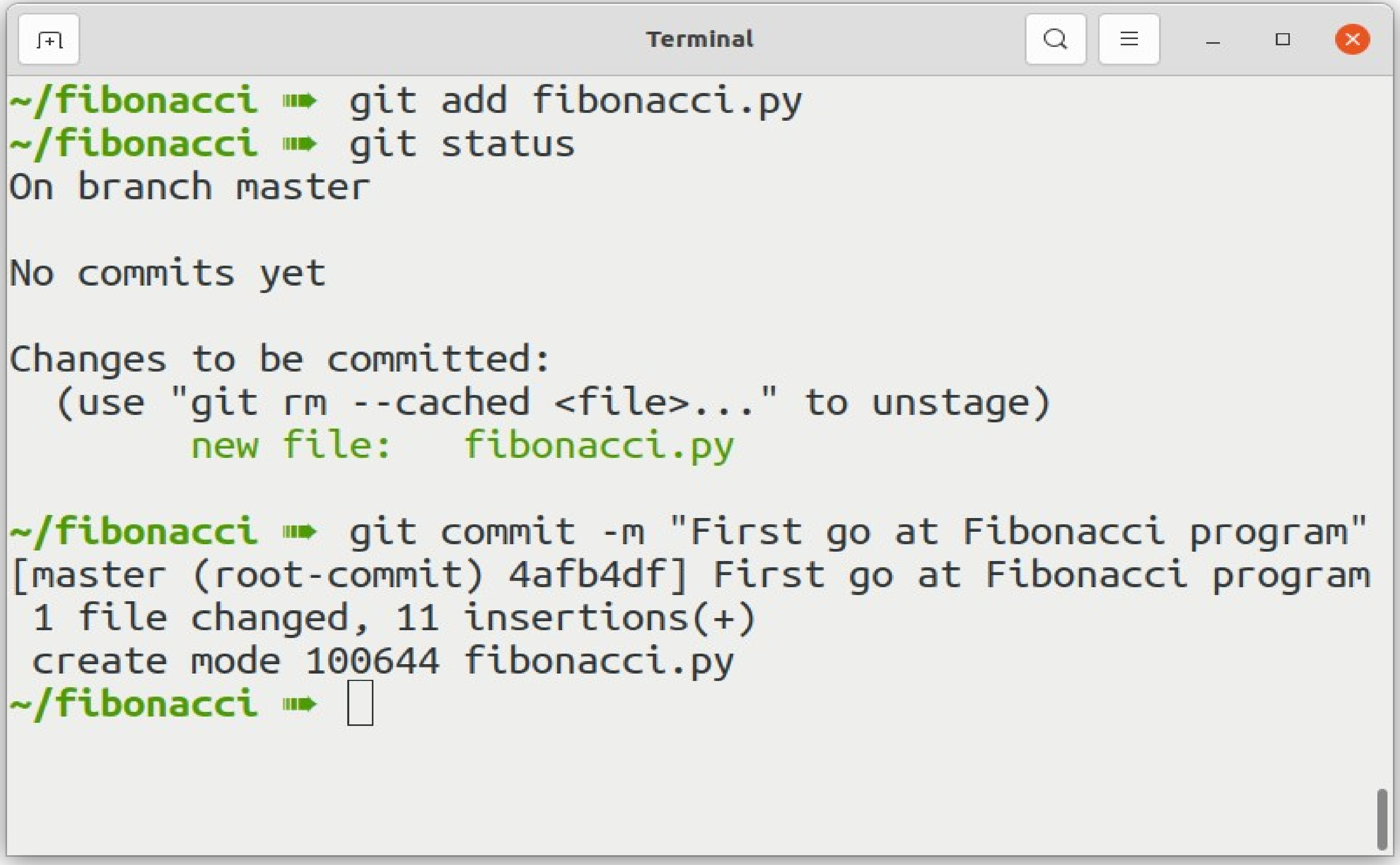
- The “state” of your project
  - If identical to a commit state, we say it’s “clean”
  - If not, we can use the **status** subcommand to find what’s changed



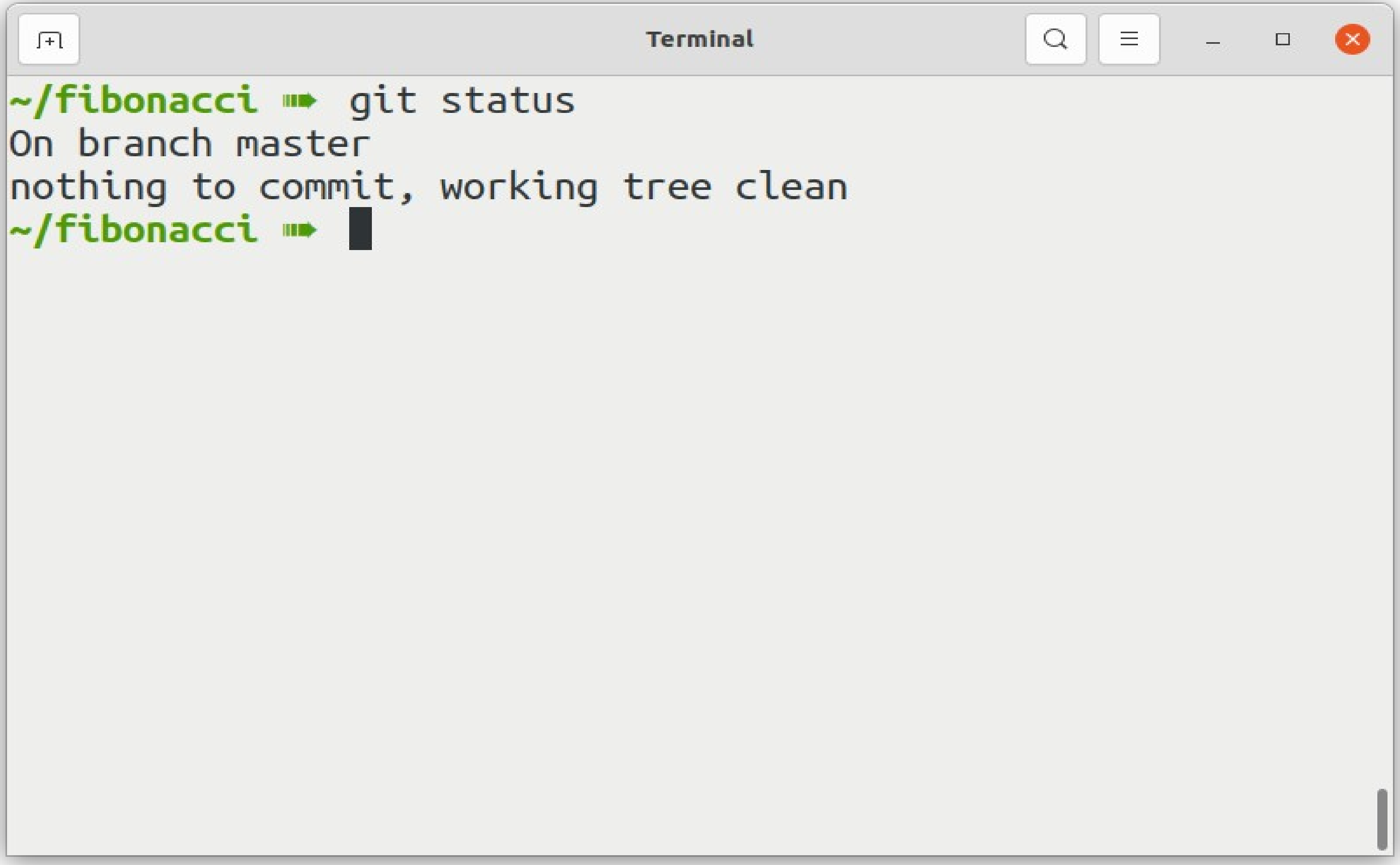
```
~/fibonacci ➠ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be commit
ted)
        fibonacci.py

nothing added to commit but untracked files present (use "gi
t add" to track)
~/fibonacci ➠ 
```

Commits II: How

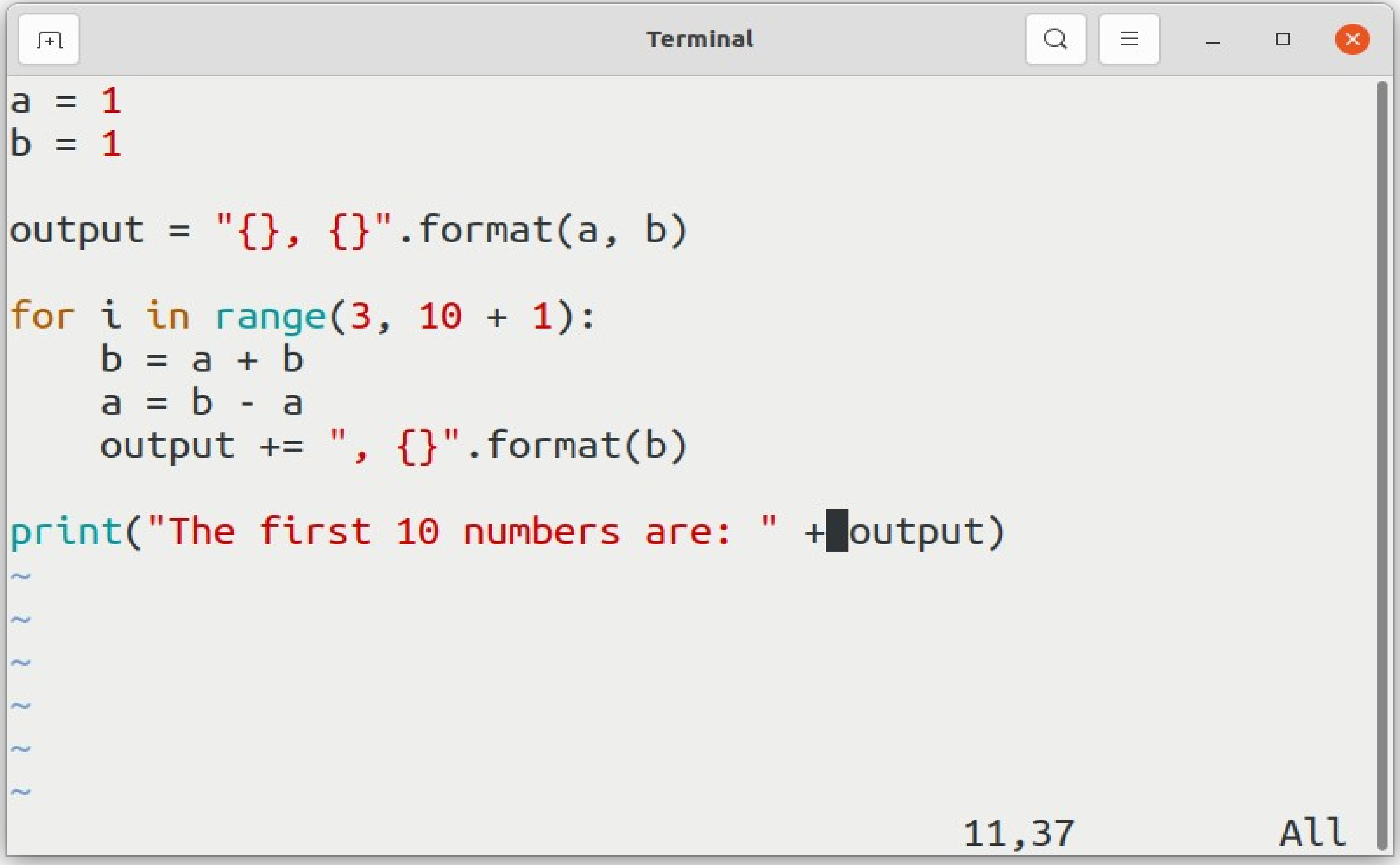
- We have made changes since the last commit
- To make a new commit:
  - Stage your changes using the **add** subcommand
  - Use the **commit** subcommand to make a new commit with these changes incorporated
    - Use the -m flag to title your commit!
    - Use the -v flag to write a longer message!



```
~/fibonacci ➠ git add fibonacci.py
~/fibonacci ➠ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   fibonacci.py

~/fibonacci ➠ git commit -m "First go at Fibonacci program"
[master (root-commit) 4afb4df] First go at Fibonacci program
 1 file changed, 11 insertions(+)
 create mode 100644 fibonacci.py
~/fibonacci ➠ 
```

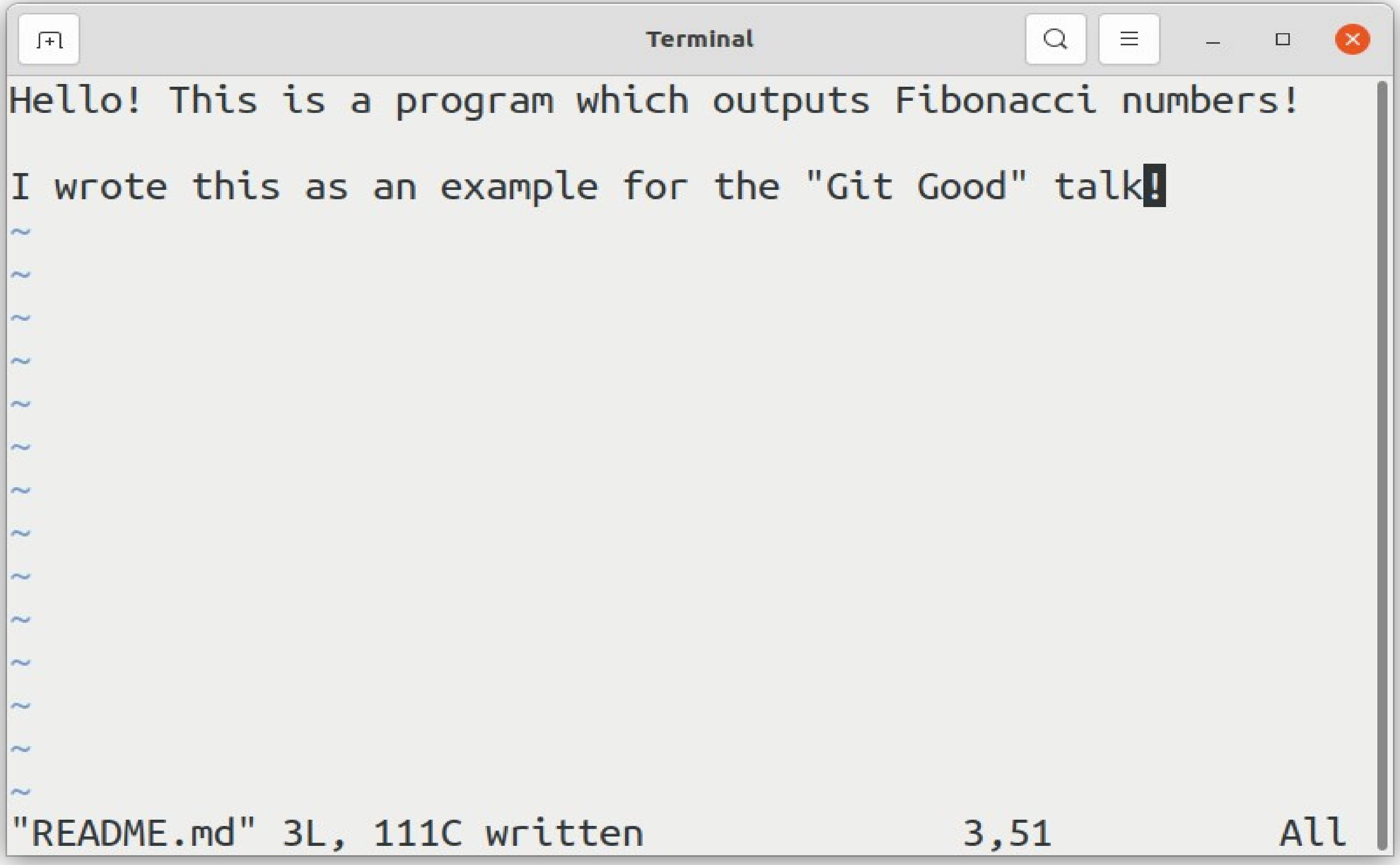


```
~/fibonacci ➠ git status
On branch master
nothing to commit, working tree clean
~/fibonacci ➠ 
```

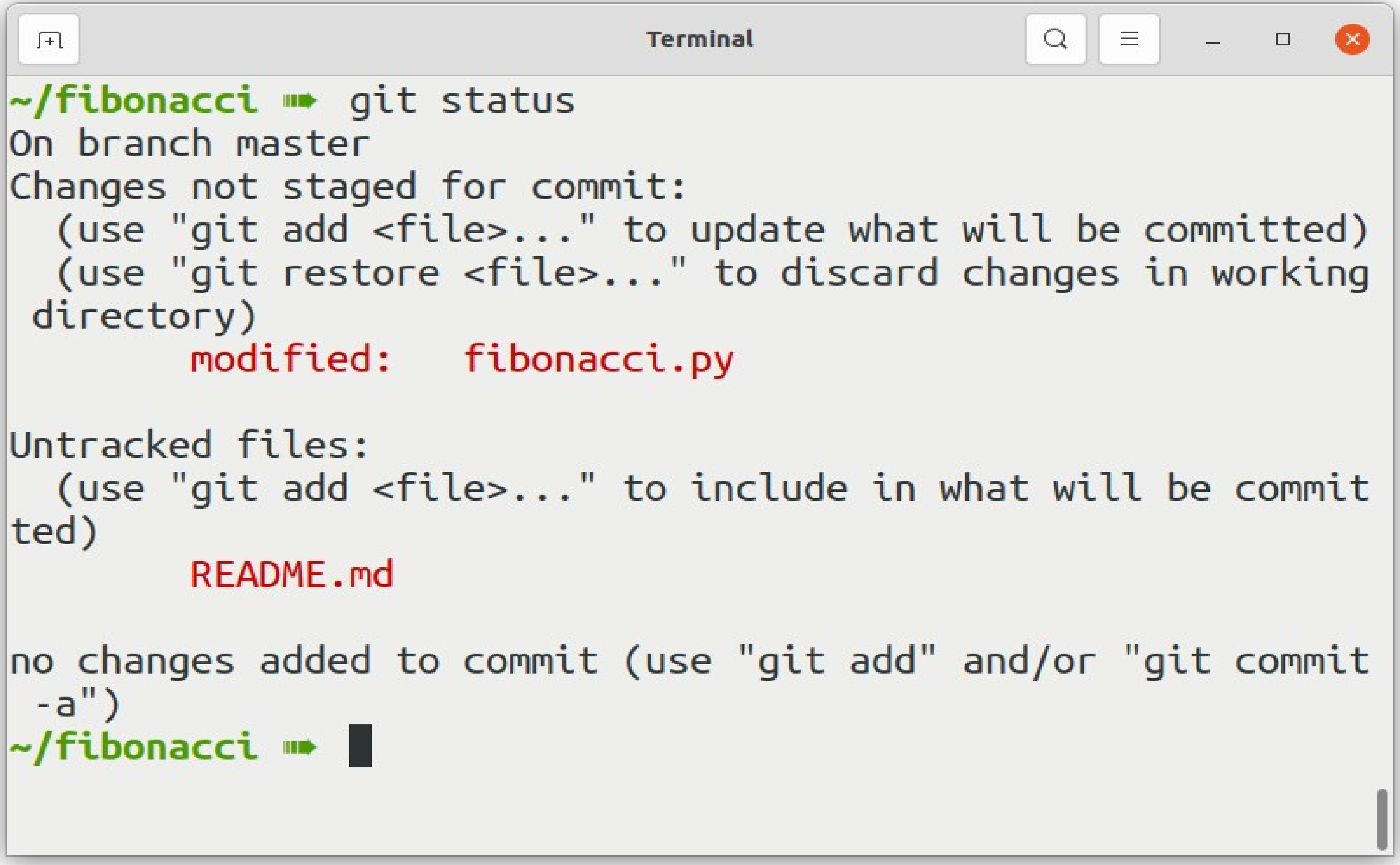


Terminal

```
a = 1
b = 1

output = "{}, {}".format(a, b)

for i in range(3, 10 + 1):
    b = a + b
    a = b - a
    output += ", {}".format(b)

print("The first 10 numbers are: " + output)
~
~
~
~
~
~
                                                   11,37         All
```

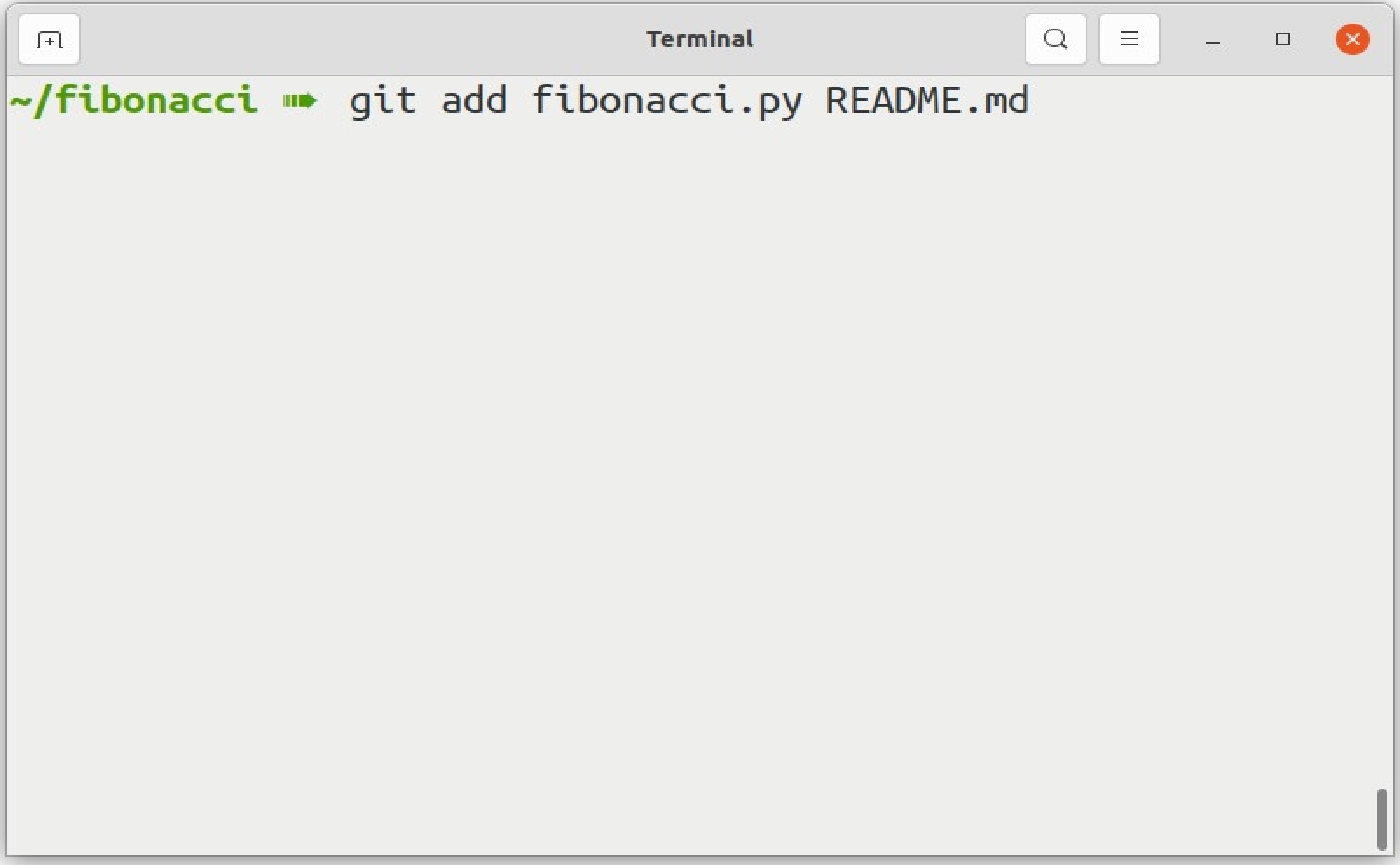


```
Hello! This is a program which outputs Fibonacci numbers!

I wrote this as an example for the "Git Good" talk!
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"README.md" 3L, 111C written                     3,51          All
```

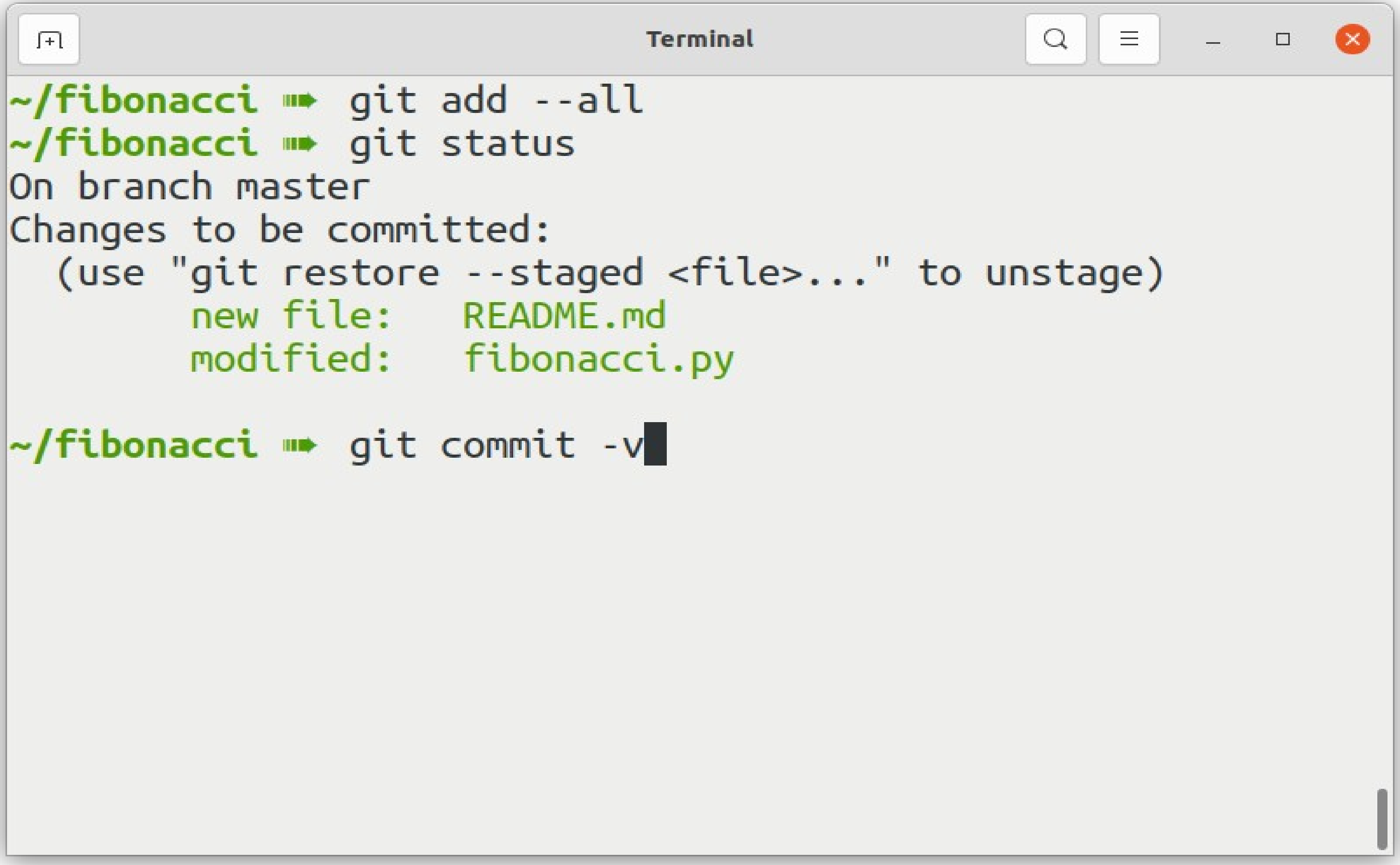


Terminal

```
~/fibonacci ➠ git status
On branch master
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:   fibonacci.py

Untracked files:
  (use "git add <file>..." to include in what will be commit
ted)
        README.md

no changes added to commit (use "git add" and/or "git commit
 -a")
~/fibonacci ➠ 
```

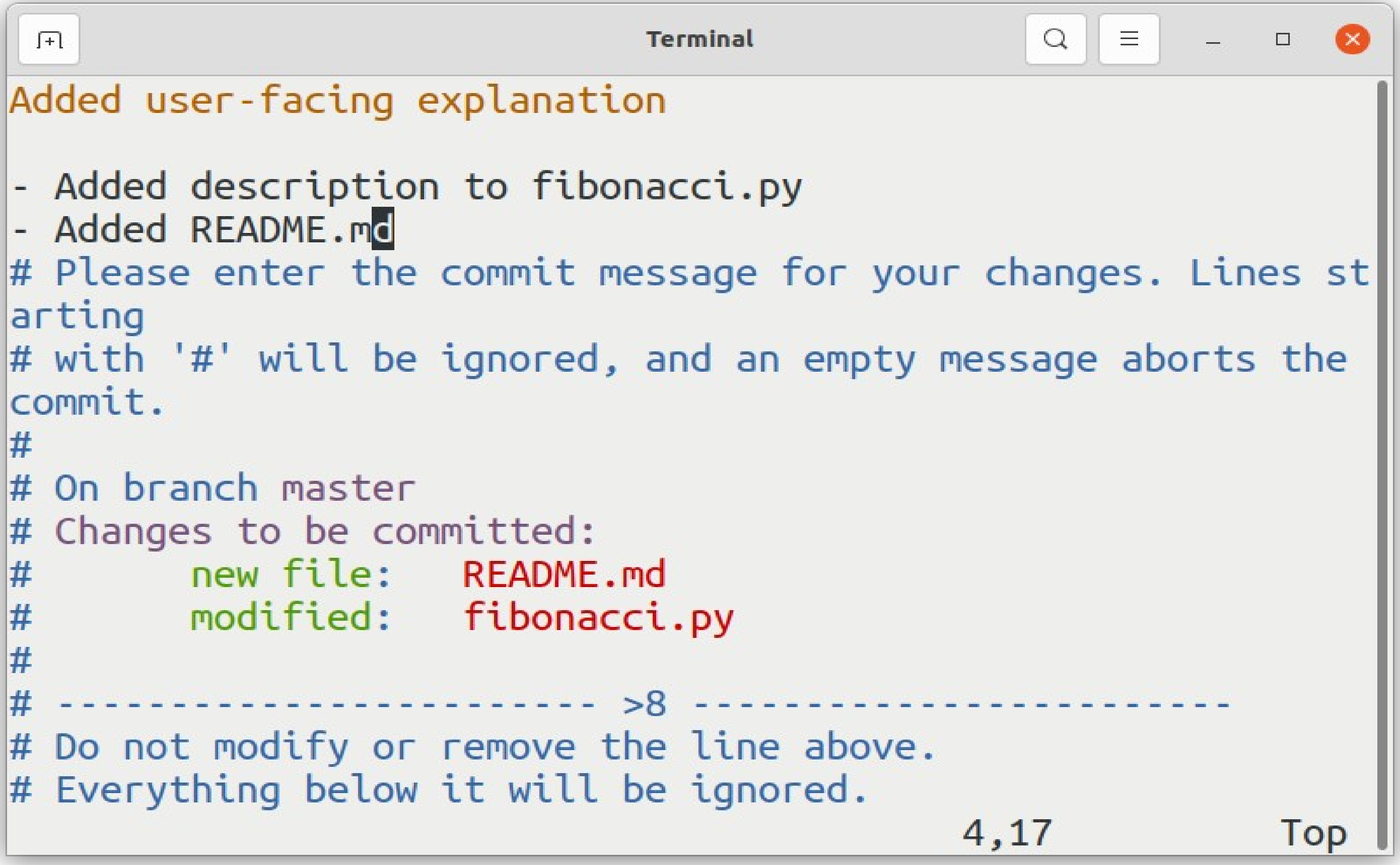


```
~/fibonacci ➠  git add fibonacci.py README.md
```

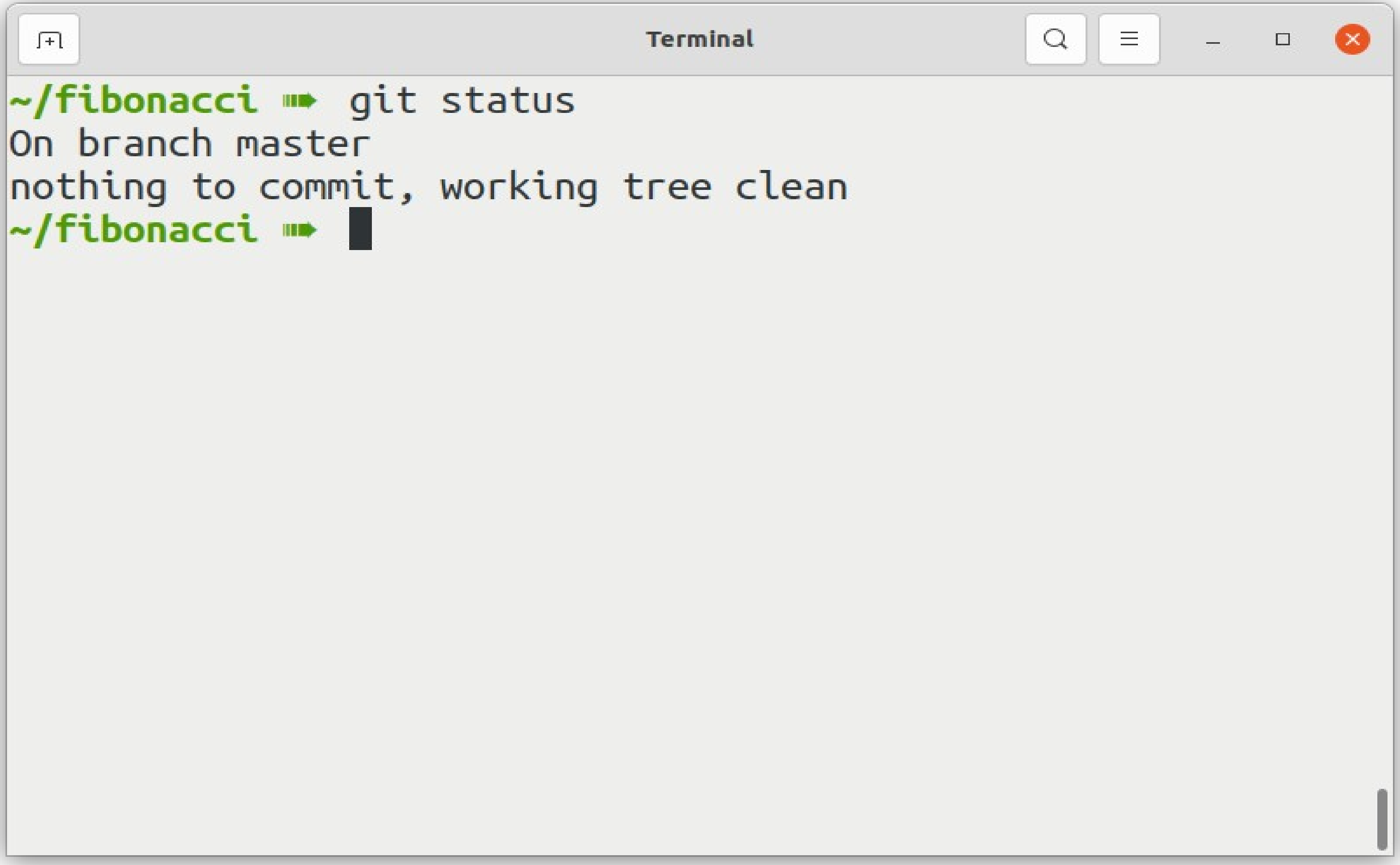


Terminal

```
~/fibonacci ➠ git add --all
~/fibonacci ➠ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   README.md
        modified:   fibonacci.py

~/fibonacci ➠ git commit -v
```



```
Added user-facing explanation

- Added description to fibonacci.py
- Added README.md
# Please enter the commit message for your changes. Lines st
arting
# with '#' will be ignored, and an empty message aborts the
commit.
#
# On branch master
# Changes to be committed:
#       new file:   README.md
#       modified:   fibonacci.py
#
# ------------------------ >8 ------------------------
# Do not modify or remove the line above.
# Everything below it will be ignored.
                                            4,17          Top
```

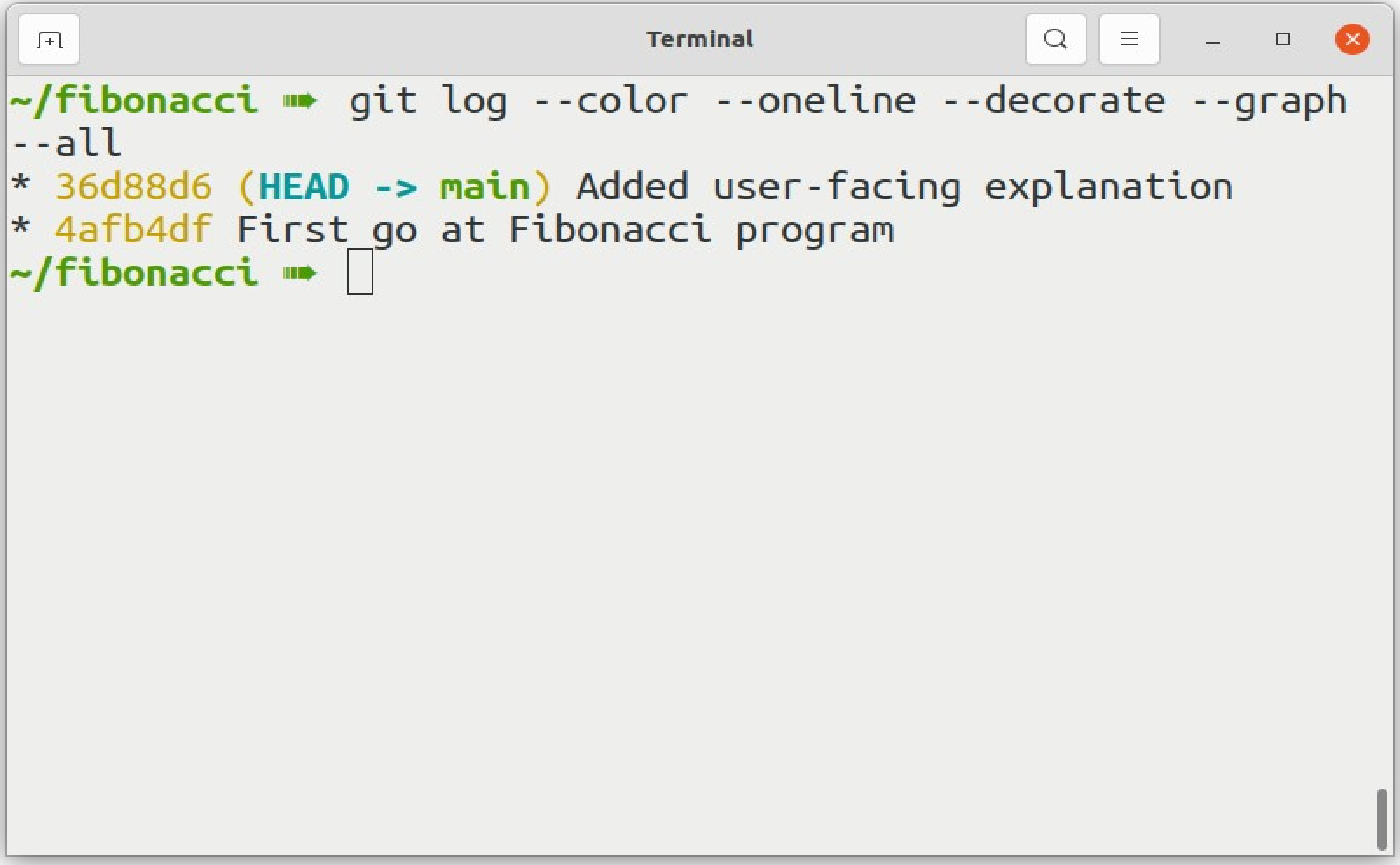


Terminal

```
~/fibonacci ➠ git status
On branch master
nothing to commit, working tree clean
~/fibonacci ➠ 
```

Log

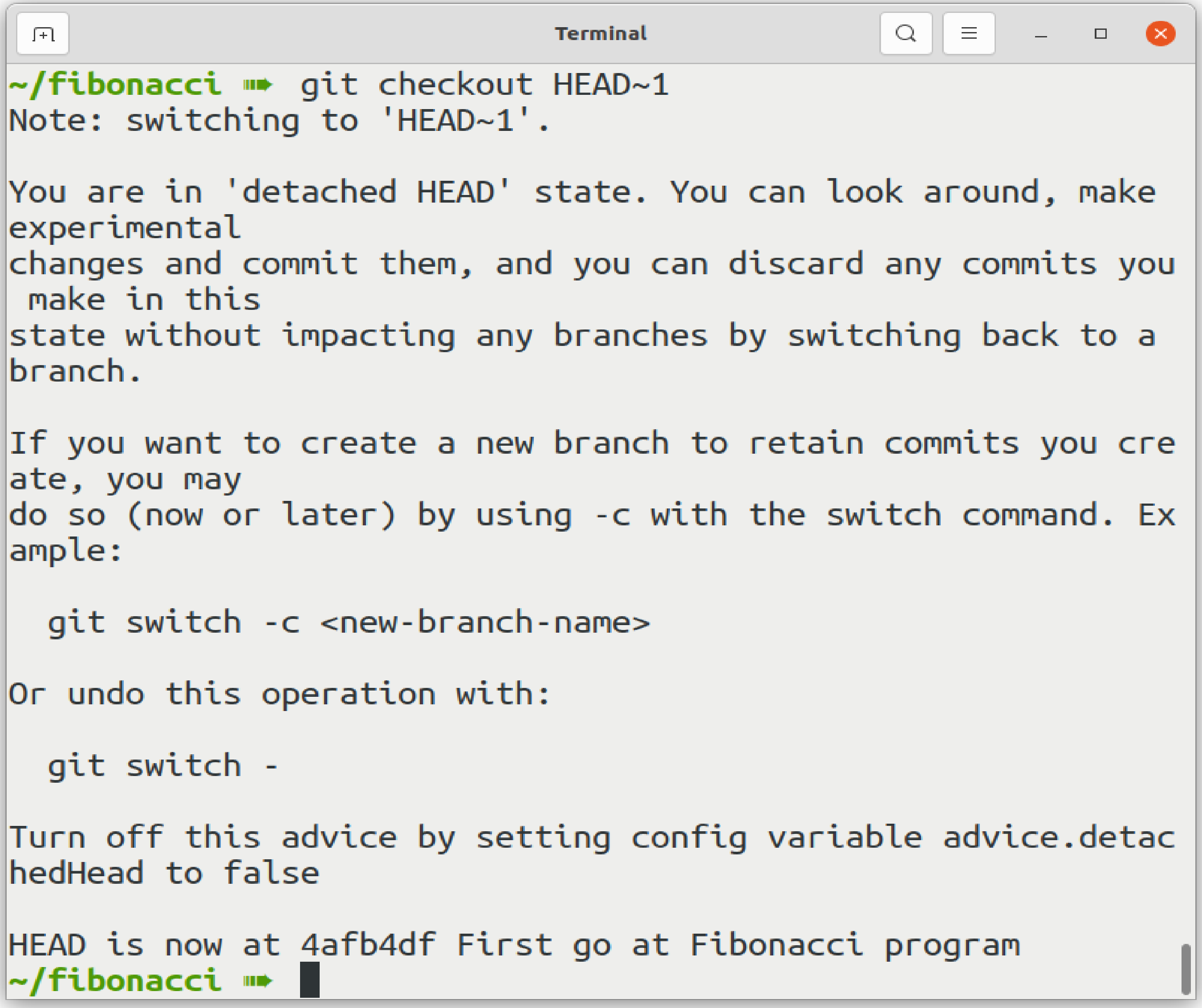
- You can see the history of your repo using the **log** subcommand
- Plenty of ways to make it look nicer
  - --color, --oneline, --graph, --decorate, --all



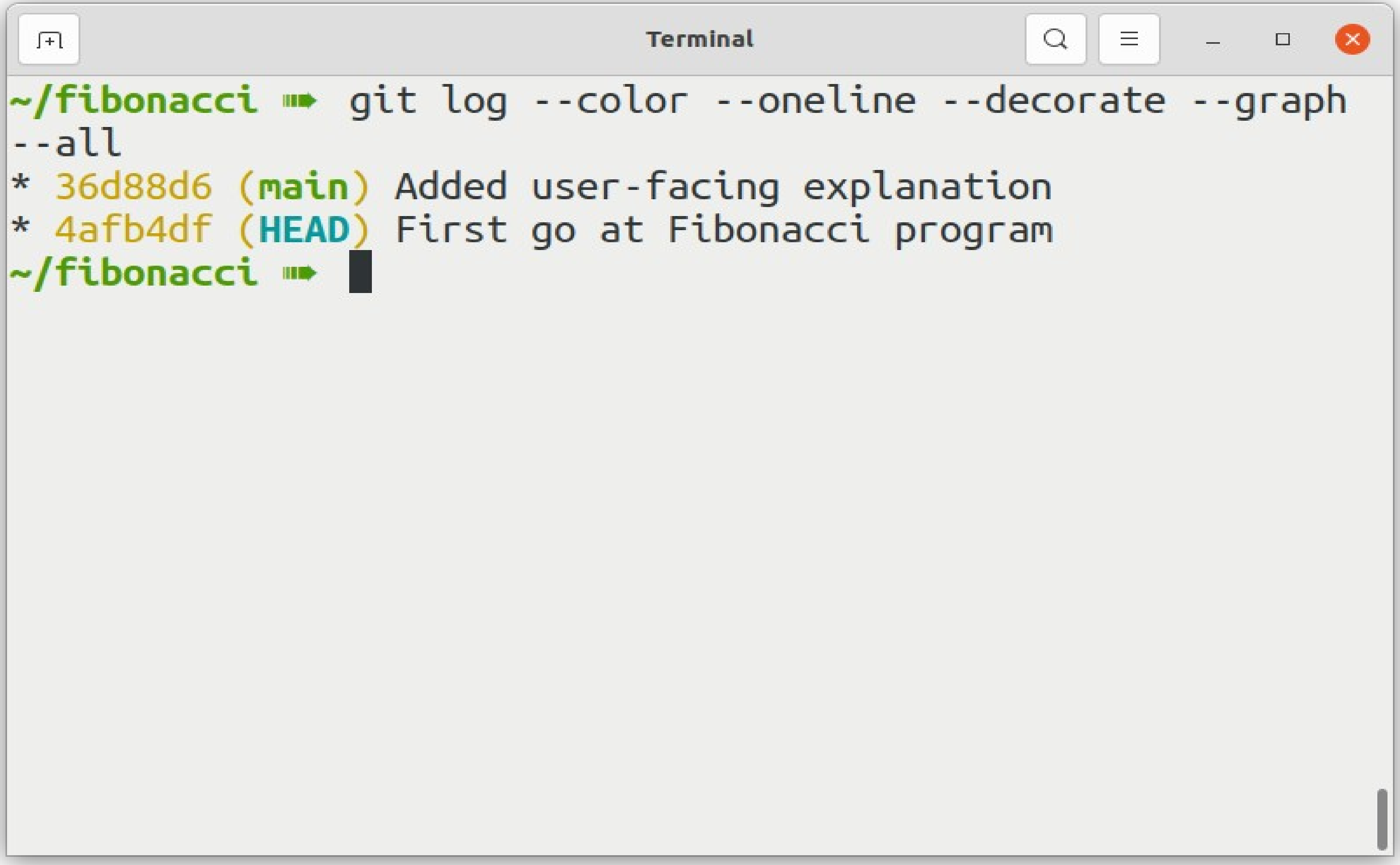
```
~/fibonacci ➠ git log --color --oneline --decorate --graph
--all
* 36d88d6 (HEAD -> main) Added user-facing explanation
* 4afb4df First go at Fibonacci program
~/fibonacci ➠
```

Checkout

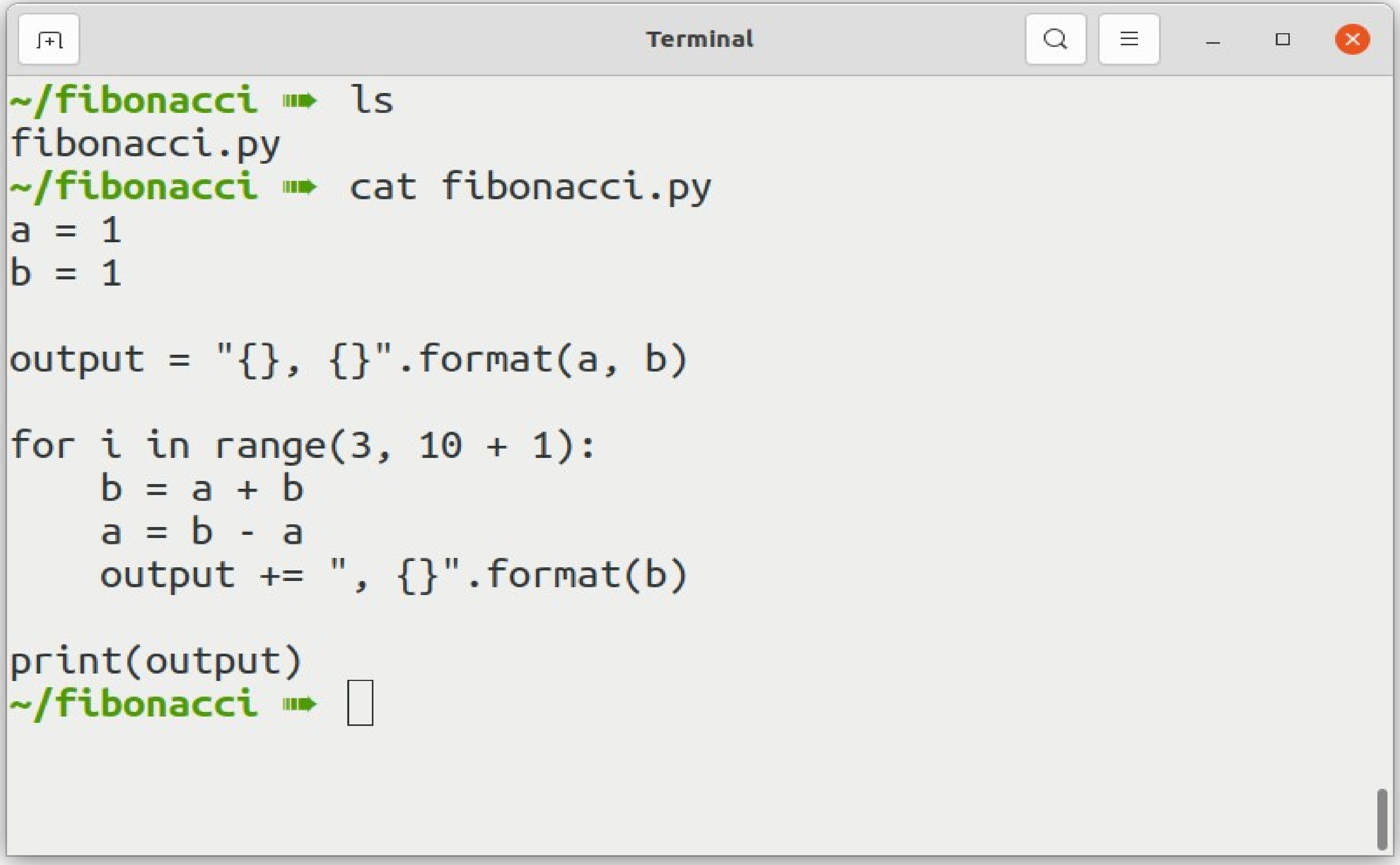
- You can travel around the history of your repo!
- Use the **checkout** subcommand to visit a commit by using its hash
  - Make sure your working directory is clean
- HEAD is used to refer to where you currently are
- You can go “back” from a commit using ~, e.g. HEAD~1



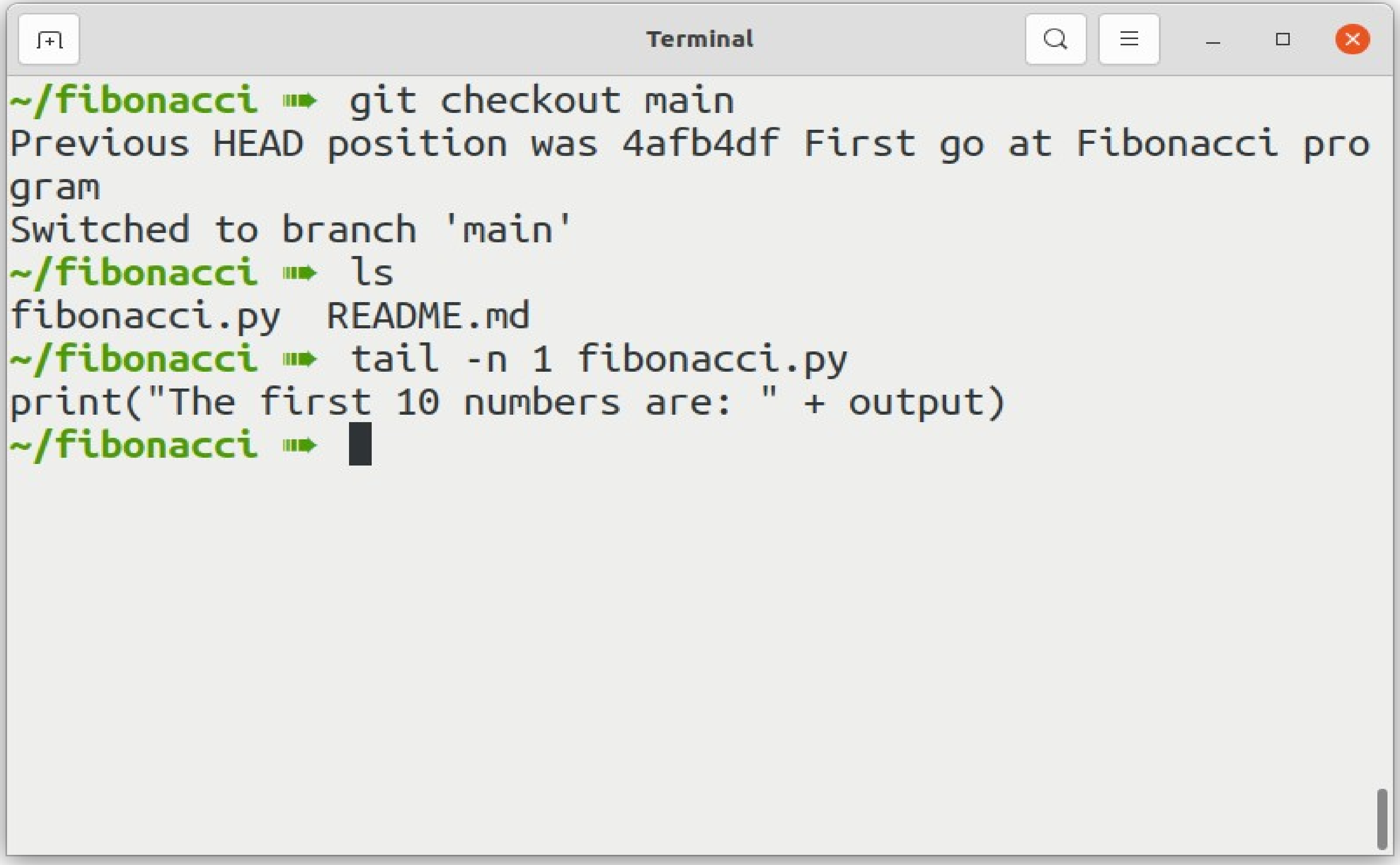
```
~/fibonacci ➠ git checkout HEAD~1
Note: switching to 'HEAD~1'.

You are in 'detached HEAD' state. You can look around, make
experimental
changes and commit them, and you can discard any commits you
 make in this
state without impacting any branches by switching back to a
branch.

If you want to create a new branch to retain commits you cre
ate, you may
do so (now or later) by using -c with the switch command. Ex
ample:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detac
hedHead to false

HEAD is now at 4afb4df First go at Fibonacci program
~/fibonacci ➠ 
```



```
~/fibonacci ➠ git log --color --oneline --decorate --graph
--all
* 36d88d6 (main) Added user-facing explanation
* 4afb4df (HEAD) First go at Fibonacci program
~/fibonacci ➠ 
```



```
~/fibonacci ➠ ls
fibonacci.py
~/fibonacci ➠ cat fibonacci.py
a = 1
b = 1

output = "{}, {}".format(a, b)

for i in range(3, 10 + 1):
    b = a + b
    a = b - a
    output += ", {}".format(b)

print(output)
~/fibonacci ➠ 
```



```
~/fibonacci ➠ git checkout main
Previous HEAD position was 4afb4df First go at Fibonacci pro
gram
Switched to branch 'main'
~/fibonacci ➠ ls
fibonacci.py  README.md
~/fibonacci ➠ tail -n 1 fibonacci.py
print("The first 10 numbers are: " + output)
~/fibonacci ➠ 
```

Branches I: What

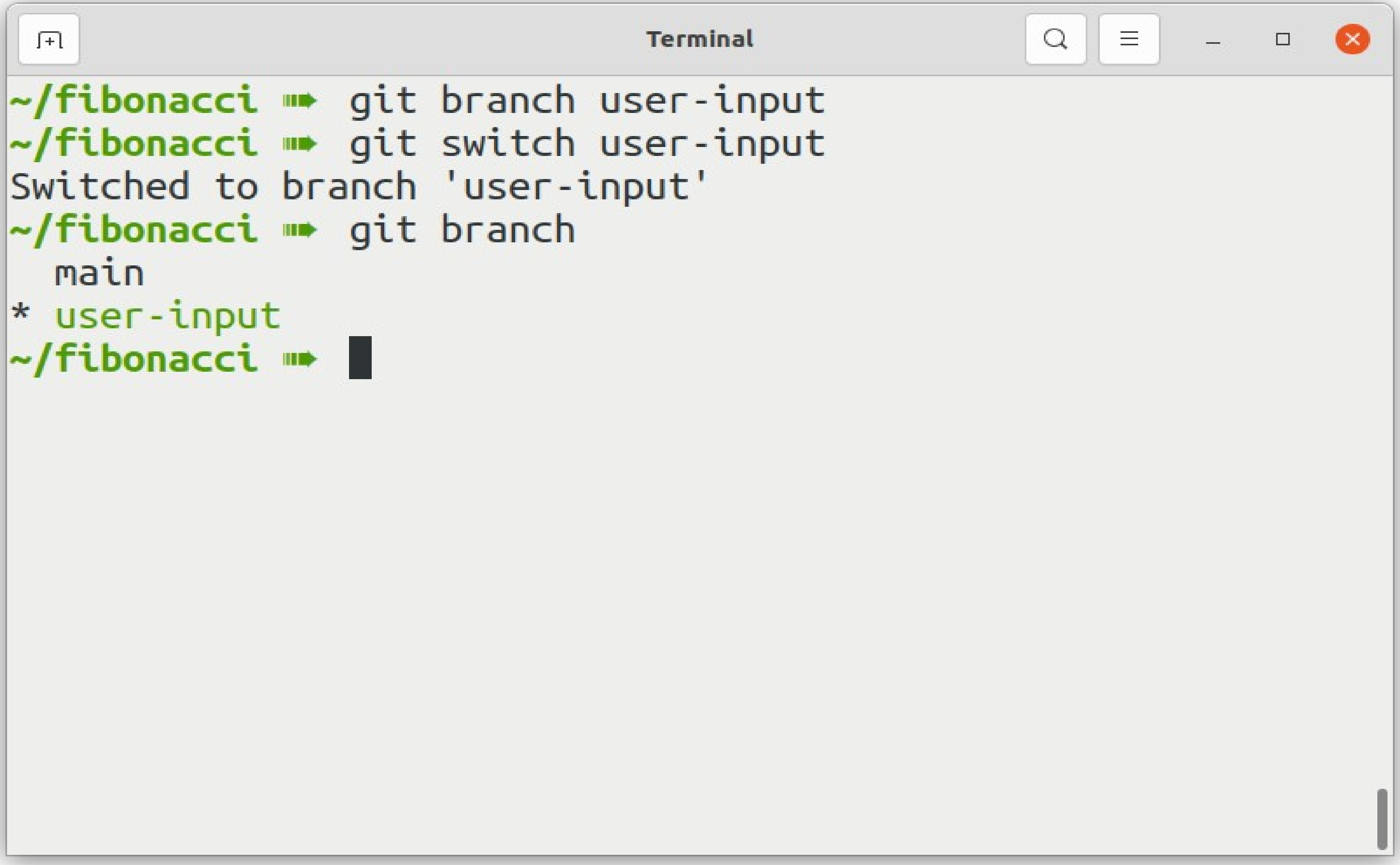
- We can make “alternative timelines”!
- You can think of a branch as a series of commits
- Default branch is called “main” (or “master”)
  - This is the branch that you start on
  - Generally used as the “up-to-date” and “stable” version of a project

Branches II: Why

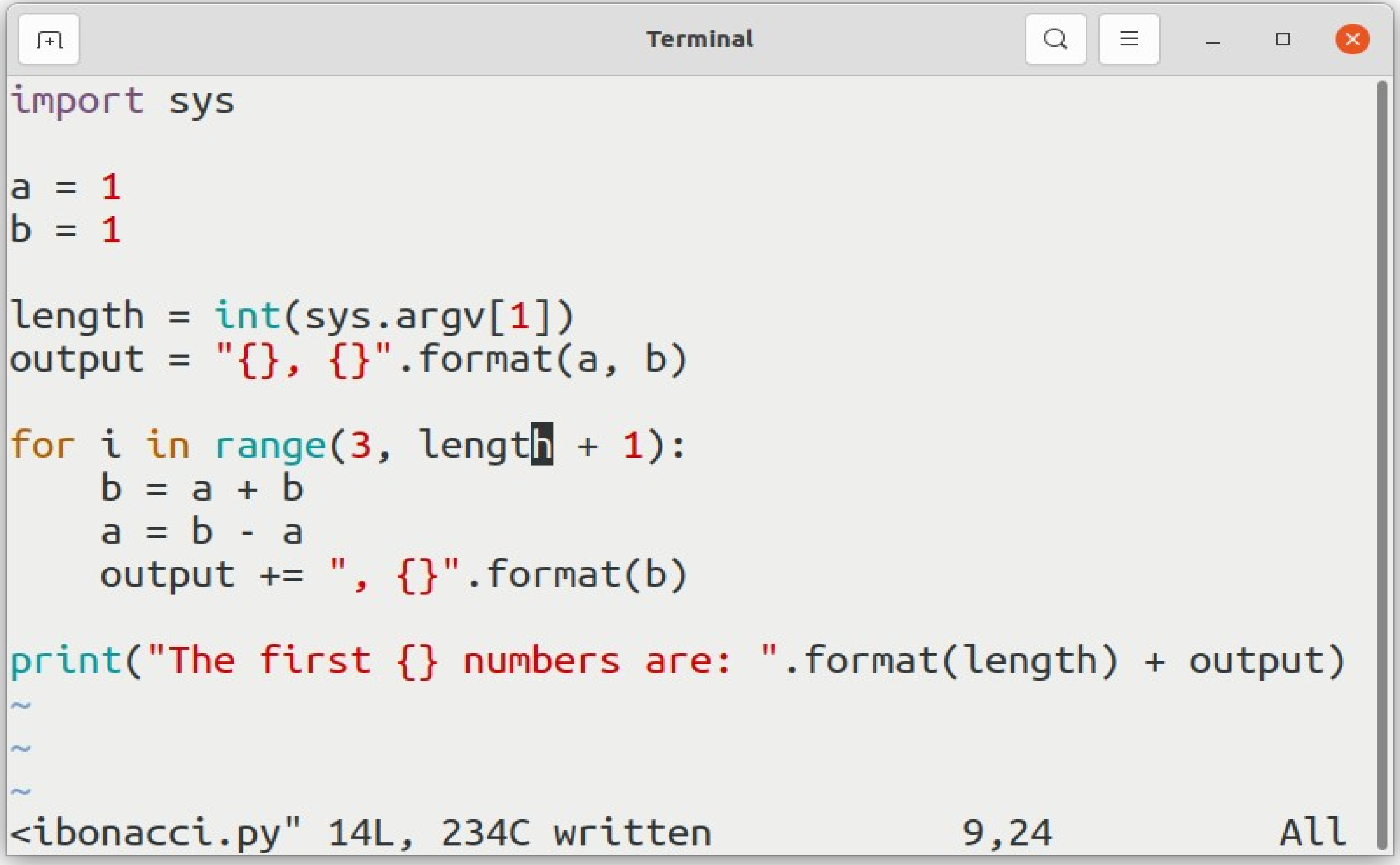
- Sometimes we want to experiment
- Helps isolate feature development
  - Easier to choose not to incorporate a feature
- Collaborative work (we'll get to this later)

Branches III: How

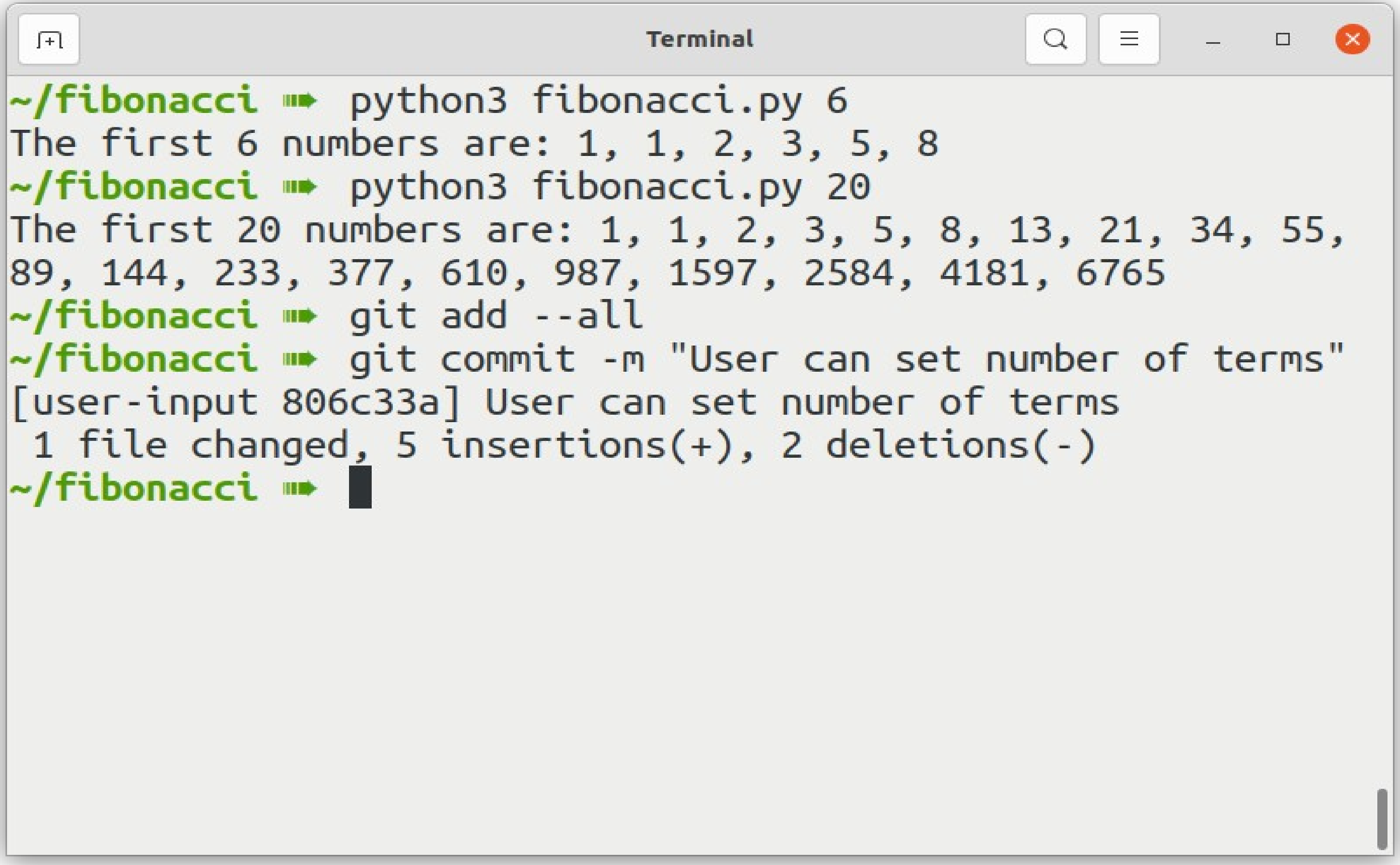
- To make a branch, use the **branch** subcommand
- You'll create a branch on the commit that you're currently on
- If you want to commit to the new branch, check it out!
  - You can now also use git **switch** to switch branches



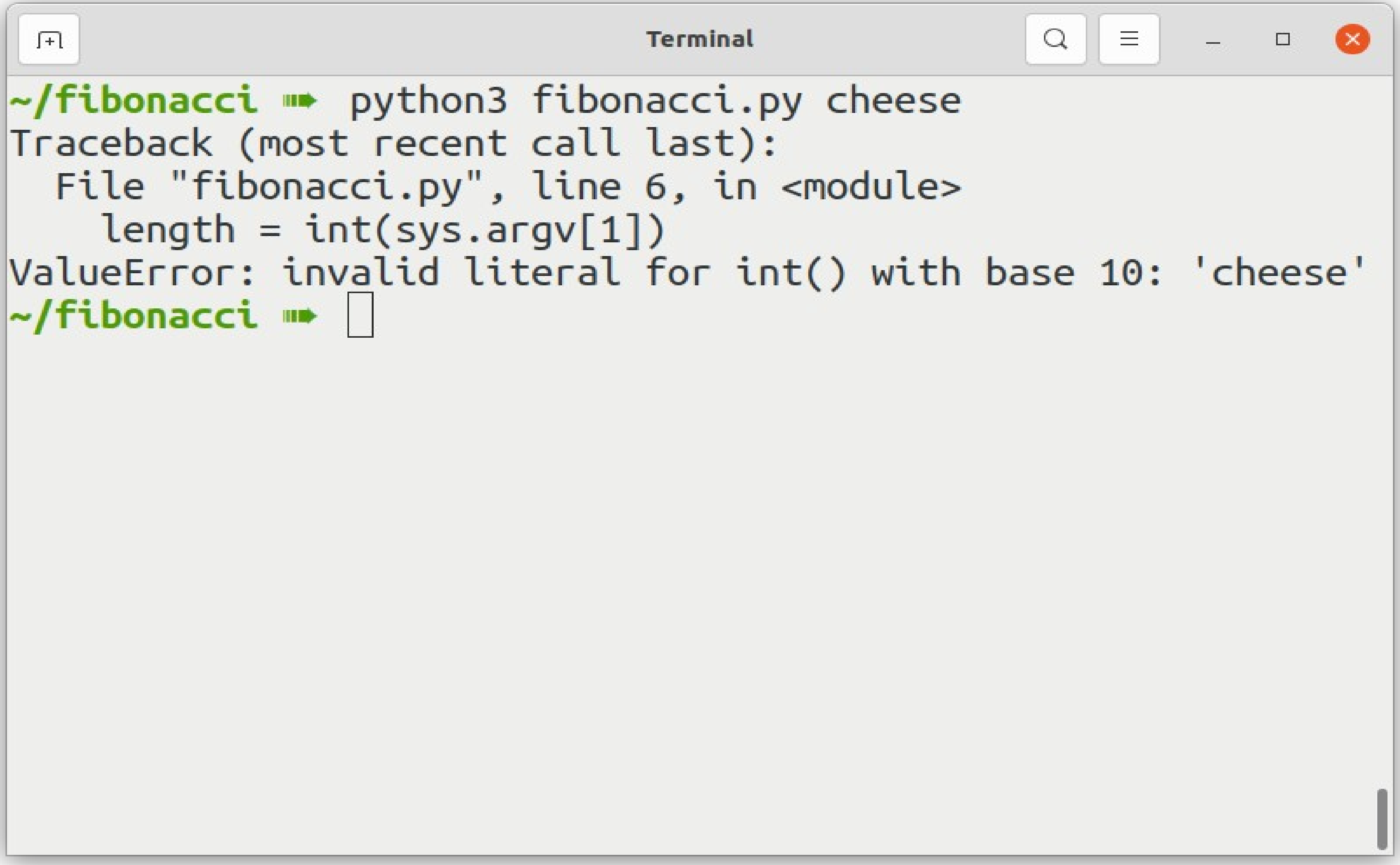
```
~/fibonacci ➠ git branch user-input
~/fibonacci ➠ git switch user-input
Switched to branch 'user-input'
~/fibonacci ➠ git branch
  main
* user-input
~/fibonacci ➠ 
```



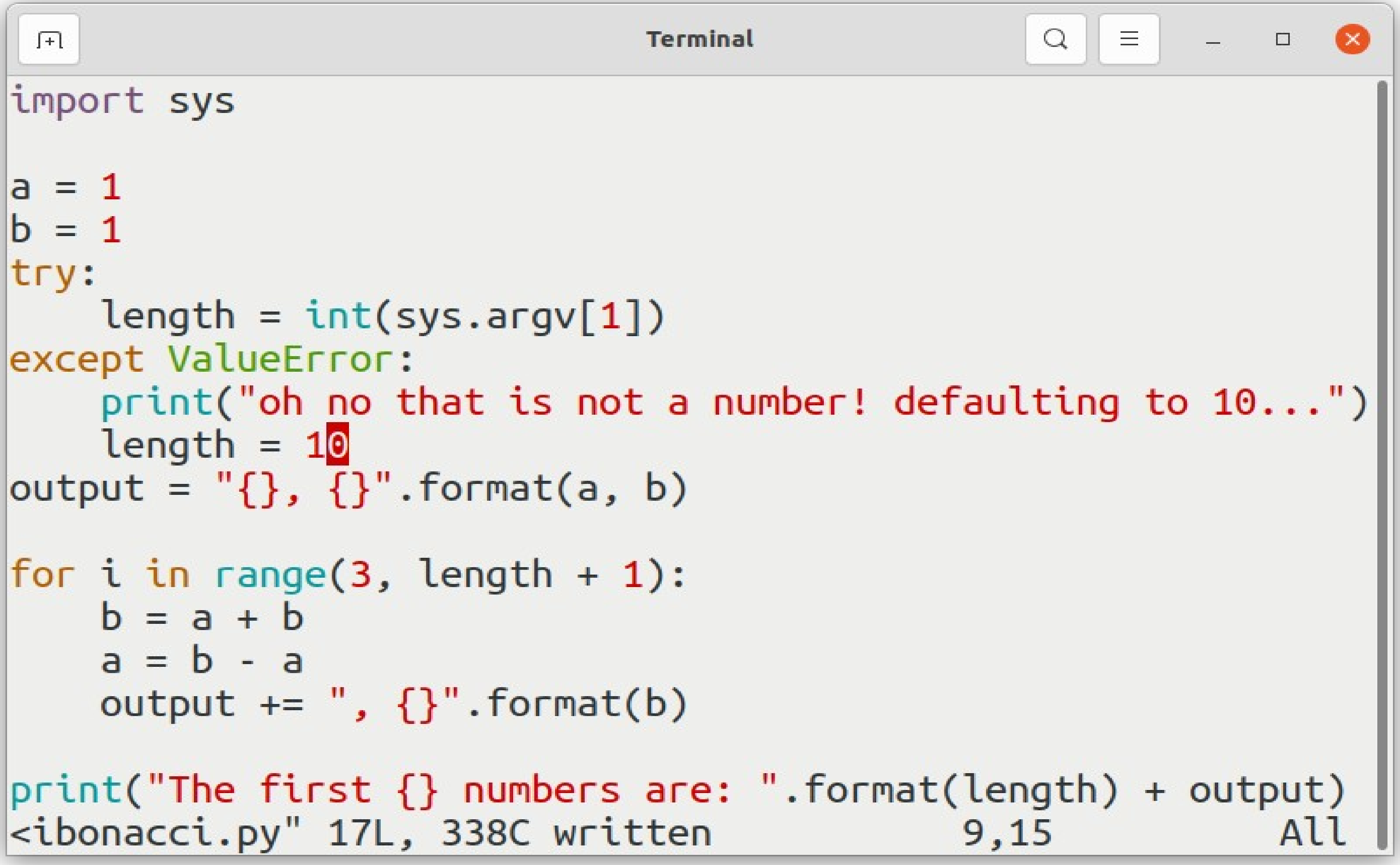
```
import sys

a = 1
b = 1

length = int(sys.argv[1])
output = "{}, {}".format(a, b)

for i in range(3, length + 1):
    b = a + b
    a = b - a
    output += ", {}".format(b)

print("The first {} numbers are: ".format(length) + output)
~
~
~
<ibonacci.py" 14L, 234C written                    9,24          All
```



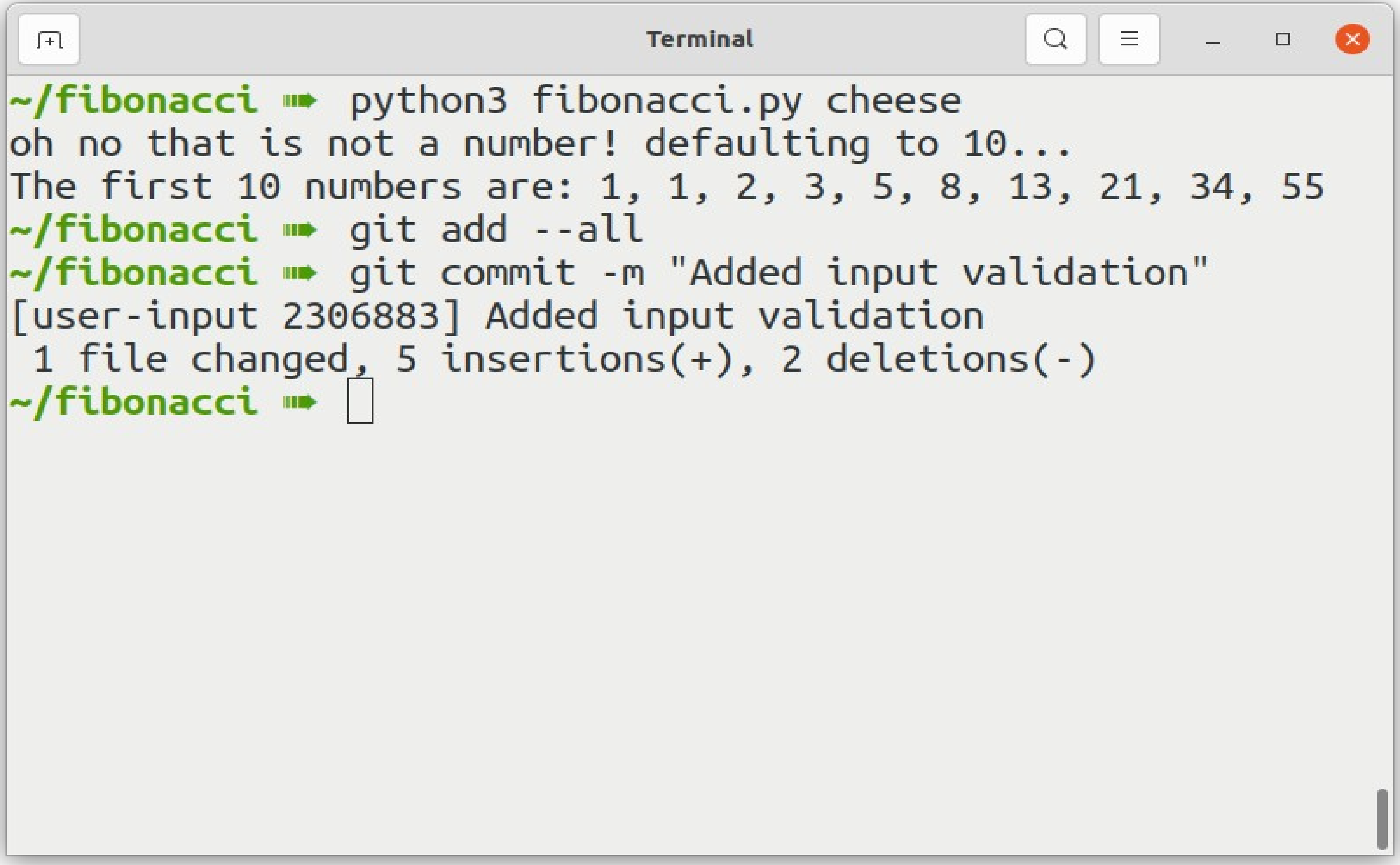
```
~/fibonacci ➠ python3 fibonacci.py 6
The first 6 numbers are: 1, 1, 2, 3, 5, 8
~/fibonacci ➠ python3 fibonacci.py 20
The first 20 numbers are: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55,
89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765
~/fibonacci ➠ git add --all
~/fibonacci ➠ git commit -m "User can set number of terms"
[user-input 806c33a] User can set number of terms
 1 file changed, 5 insertions(+), 2 deletions(-)
~/fibonacci ➠ 
```



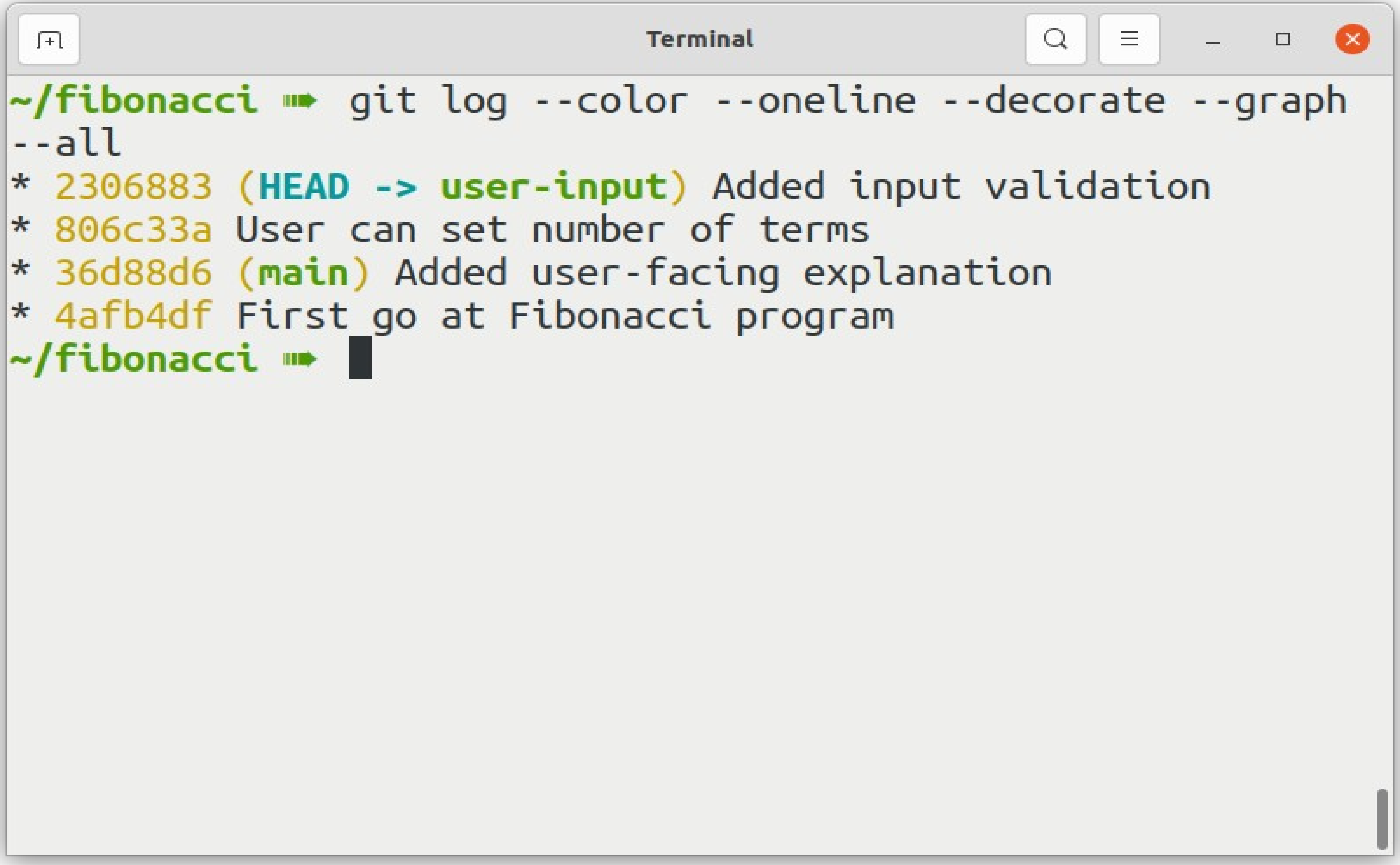
```
~/fibonacci ➠ python3 fibonacci.py cheese
Traceback (most recent call last):
  File "fibonacci.py", line 6, in <module>
    length = int(sys.argv[1])
ValueError: invalid literal for int() with base 10: 'cheese'
~/fibonacci ➠ 
```


Terminal

```
import sys

a = 1
b = 1
try:
    length = int(sys.argv[1])
except ValueError:
    print("oh no that is not a number! defaulting to 10...")
    length = 10
output = "{}, {}".format(a, b)

for i in range(3, length + 1):
    b = a + b
    a = b - a
    output += ", {}".format(b)

print("The first {} numbers are: ".format(length) + output)
<ibonacci.py" 17L, 338C written                    9,15          All
```



```
~/fibonacci ➠ python3 fibonacci.py cheese
oh no that is not a number! defaulting to 10...
The first 10 numbers are: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55
~/fibonacci ➠ git add --all
~/fibonacci ➠ git commit -m "Added input validation"
[user-input 2306883] Added input validation
 1 file changed, 5 insertions(+), 2 deletions(-)
~/fibonacci ➠ 
```



```
~/fibonacci ➠ git log --color --oneline --decorate --graph
--all
* 2306883 (HEAD -> user-input) Added input validation
* 806c33a User can set number of terms
* 36d88d6 (main) Added user-facing explanation
* 4afb4df First go at Fibonacci program
~/fibonacci ➠ 
```

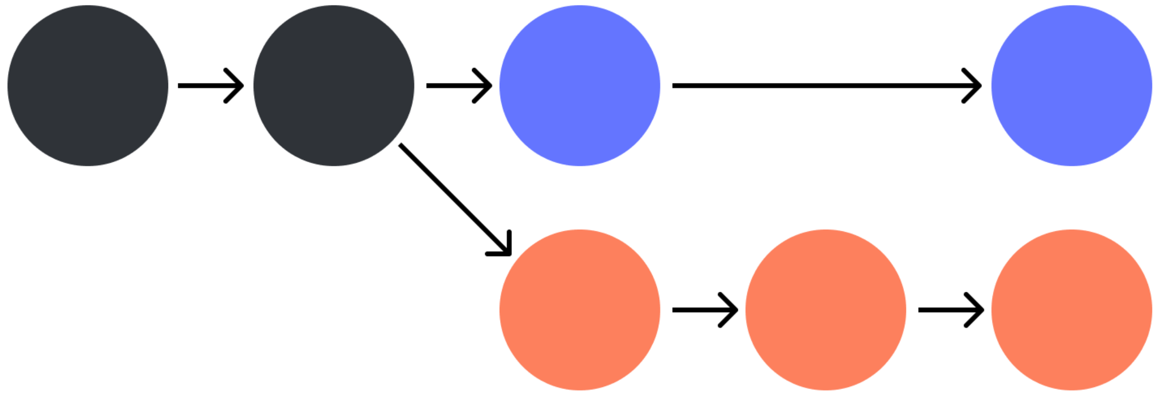
Merging I: What

- Sometimes we want the changes from more than one branch
  - Could be bug fixes, features, etc.
- So we can merge branch B into branch A to give branch A the changes from branch B

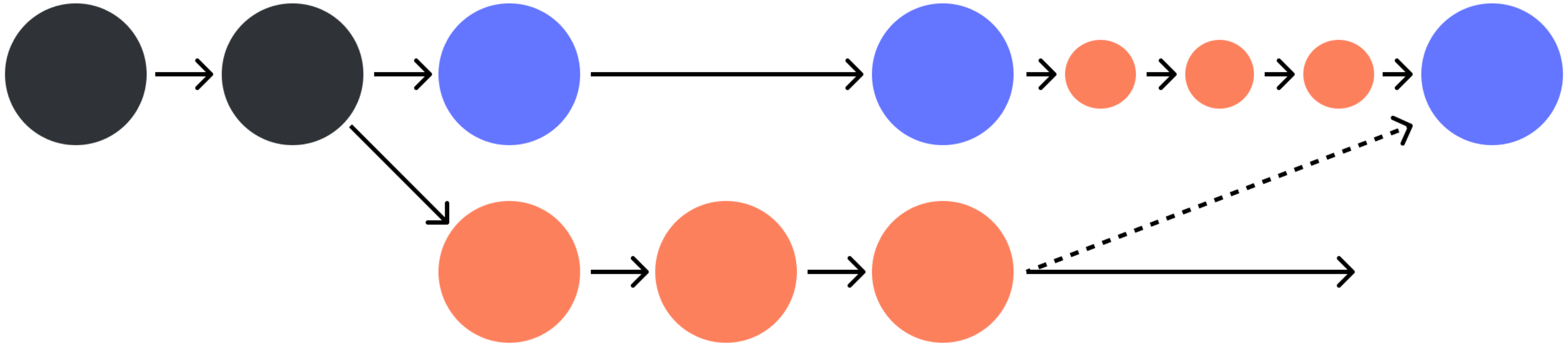
Merging II: How

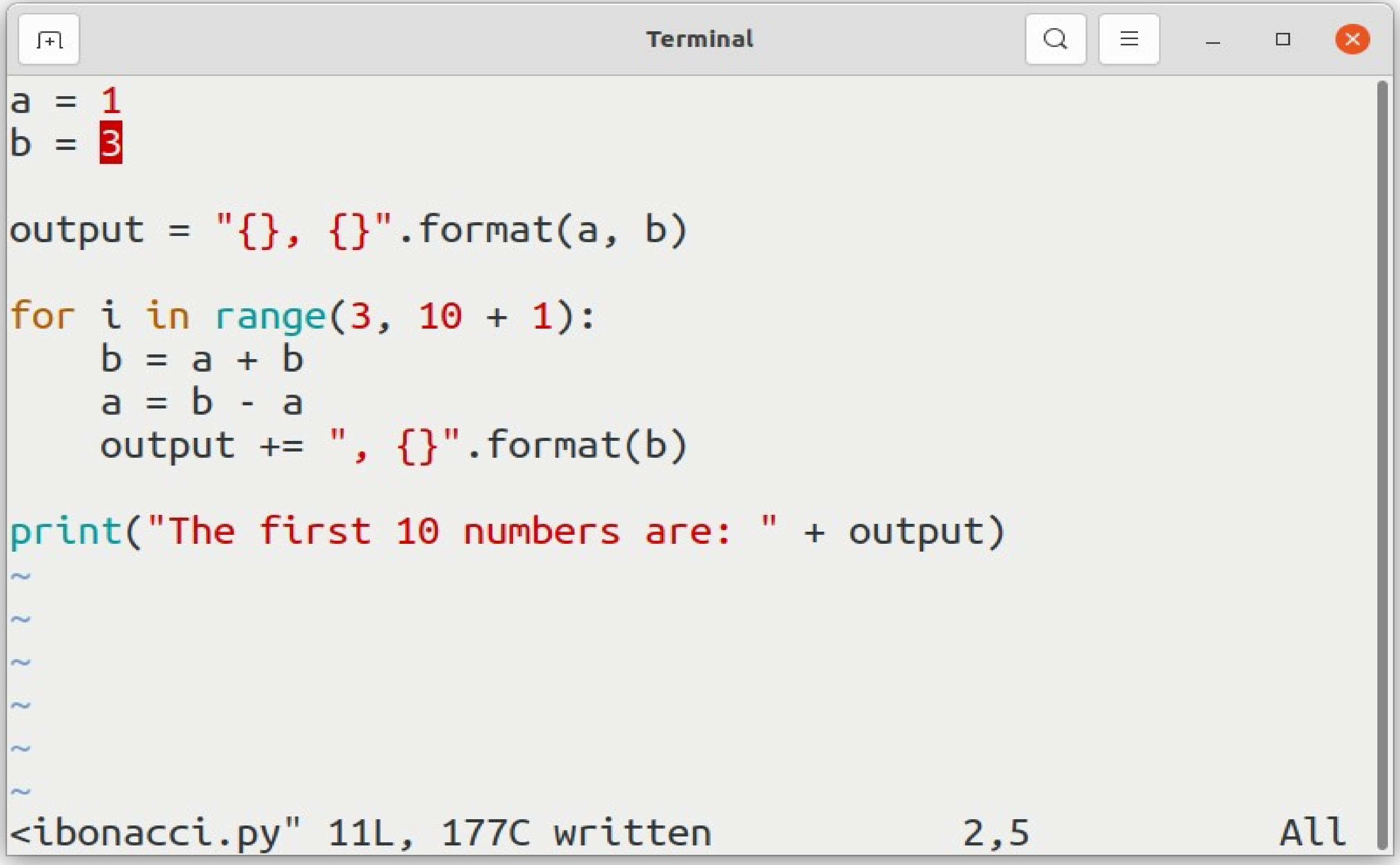
- Switch to the branch you want to merge **into**
- Use the **merge** subcommand to merge another branch into it
  - Creates a new commit
  - The branch that has been merged in does not change

Merging: In Pictures

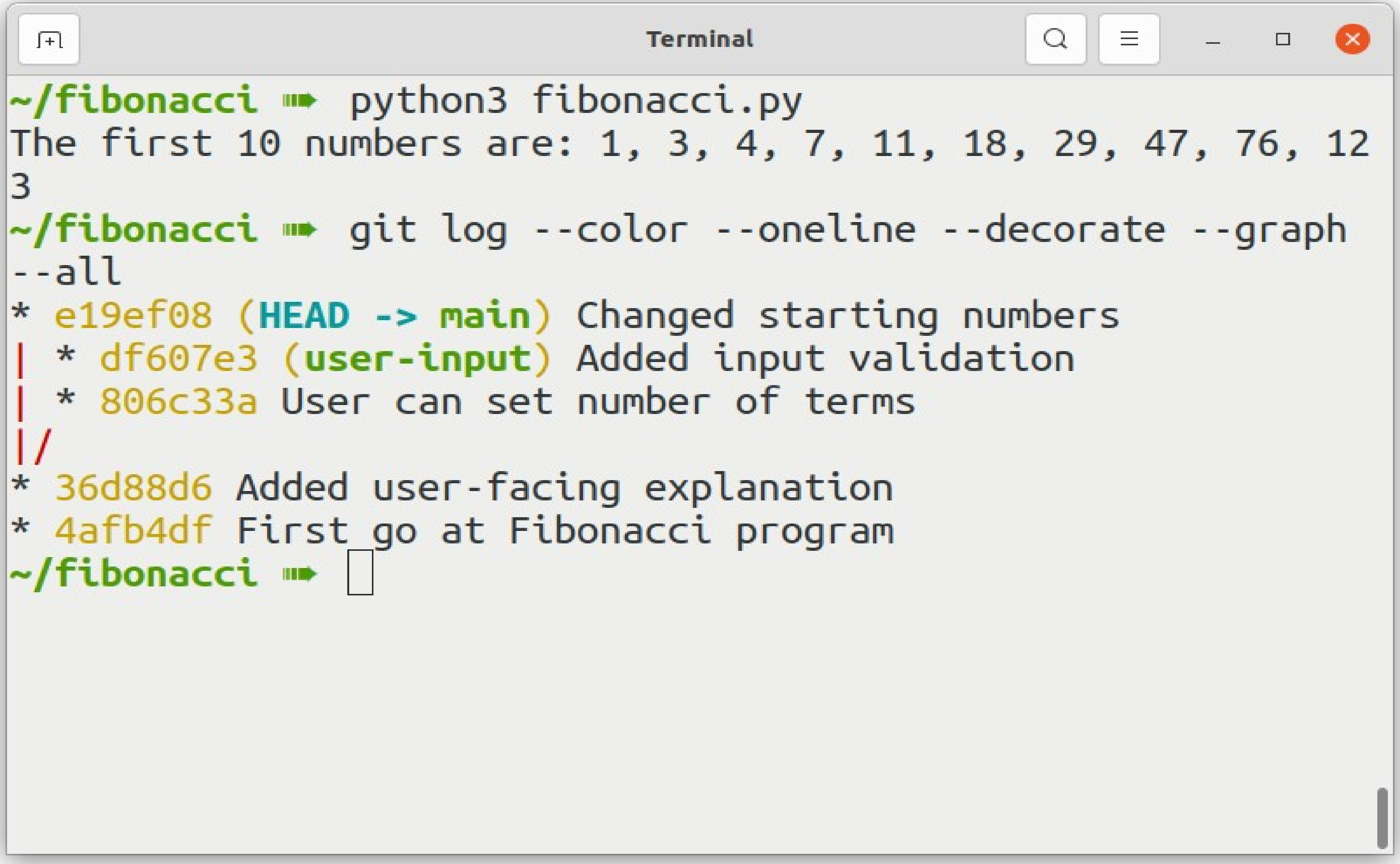


Merging: In Pictures

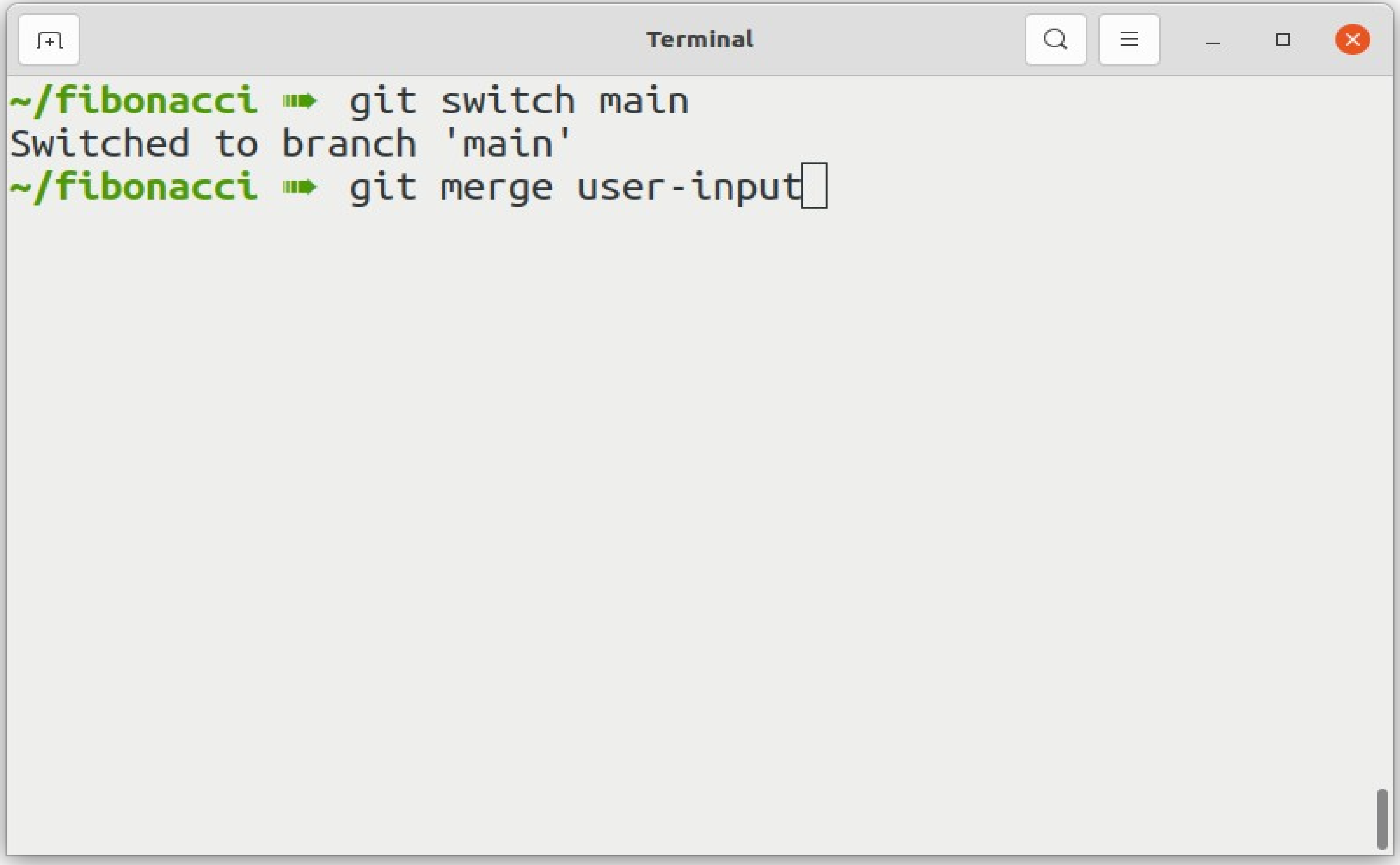




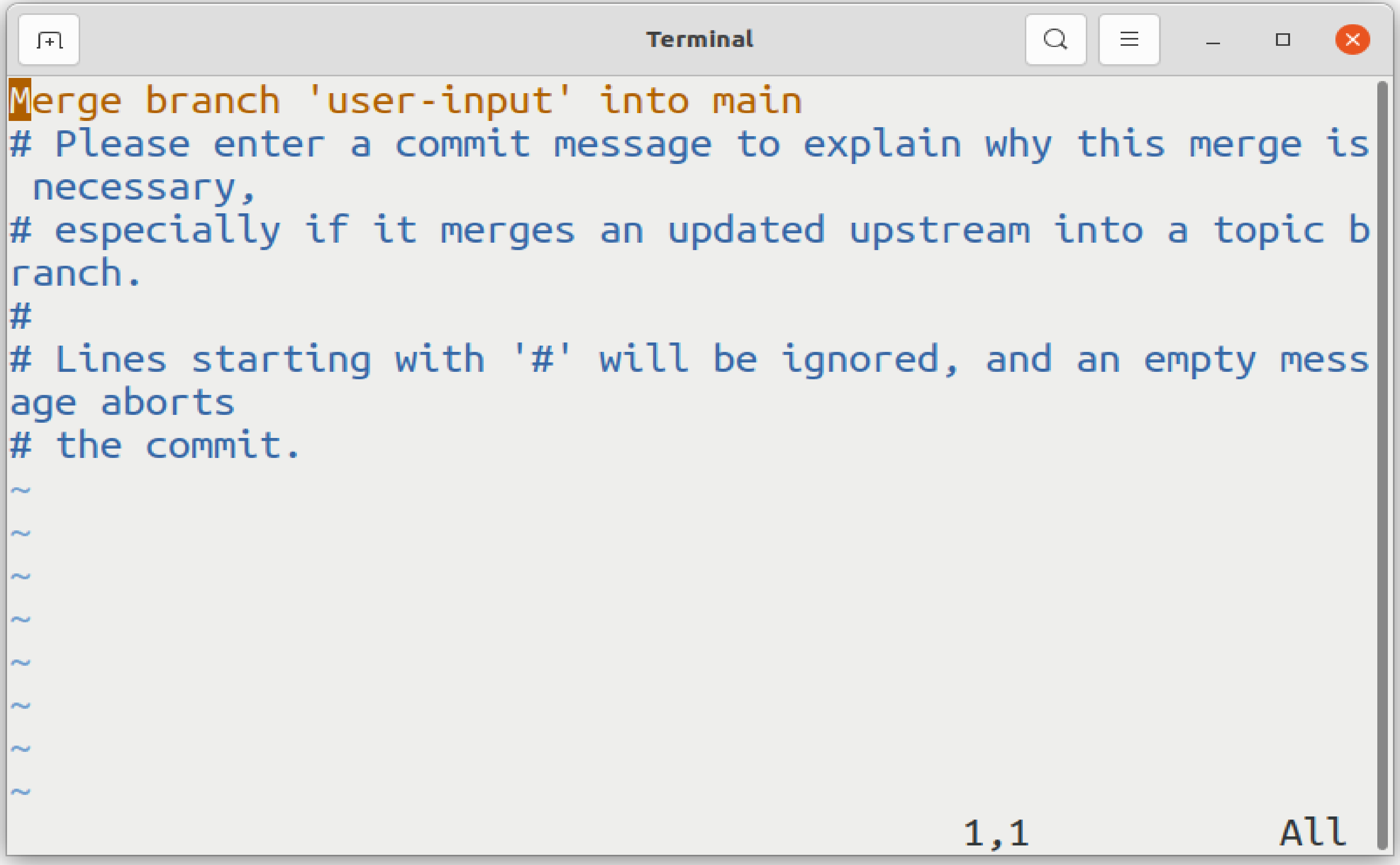
```
a = 1
b = 3

output = "{}, {}".format(a, b)

for i in range(3, 10 + 1):
    b = a + b
    a = b - a
    output += ", {}".format(b)

print("The first 10 numbers are: " + output)
~
~
~
~
~
~
<ibonacci.py" 11L, 177C written                 2,5           All
```



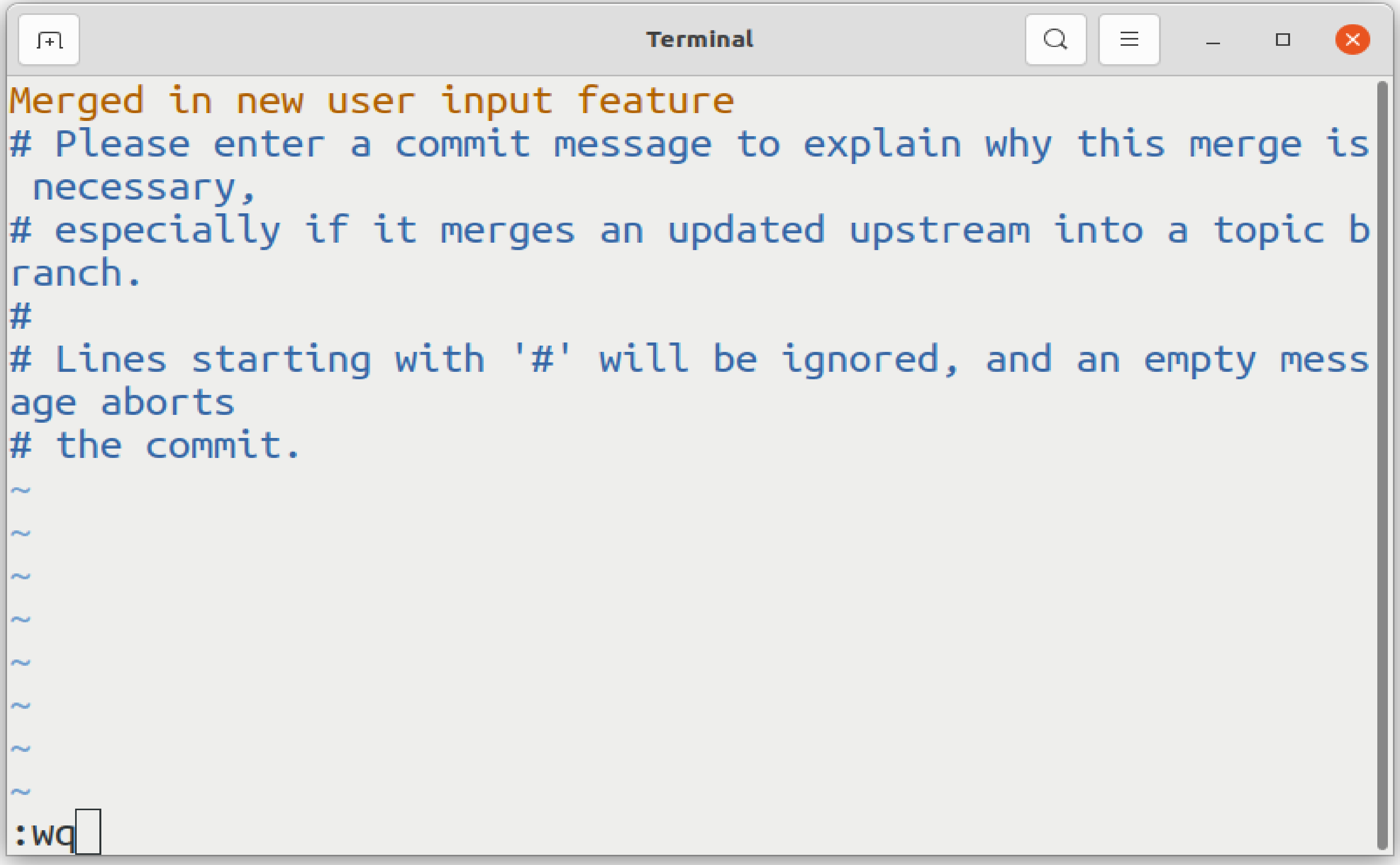
```
~/fibonacci ➠ python3 fibonacci.py
The first 10 numbers are: 1, 3, 4, 7, 11, 18, 29, 47, 76, 12
3
~/fibonacci ➠ git log --color --oneline --decorate --graph
--all
* e19ef08 (HEAD -> main) Changed starting numbers
| * df607e3 (user-input) Added input validation
| * 806c33a User can set number of terms
|/
* 36d88d6 Added user-facing explanation
* 4afb4df First go at Fibonacci program
~/fibonacci ➠ 
```



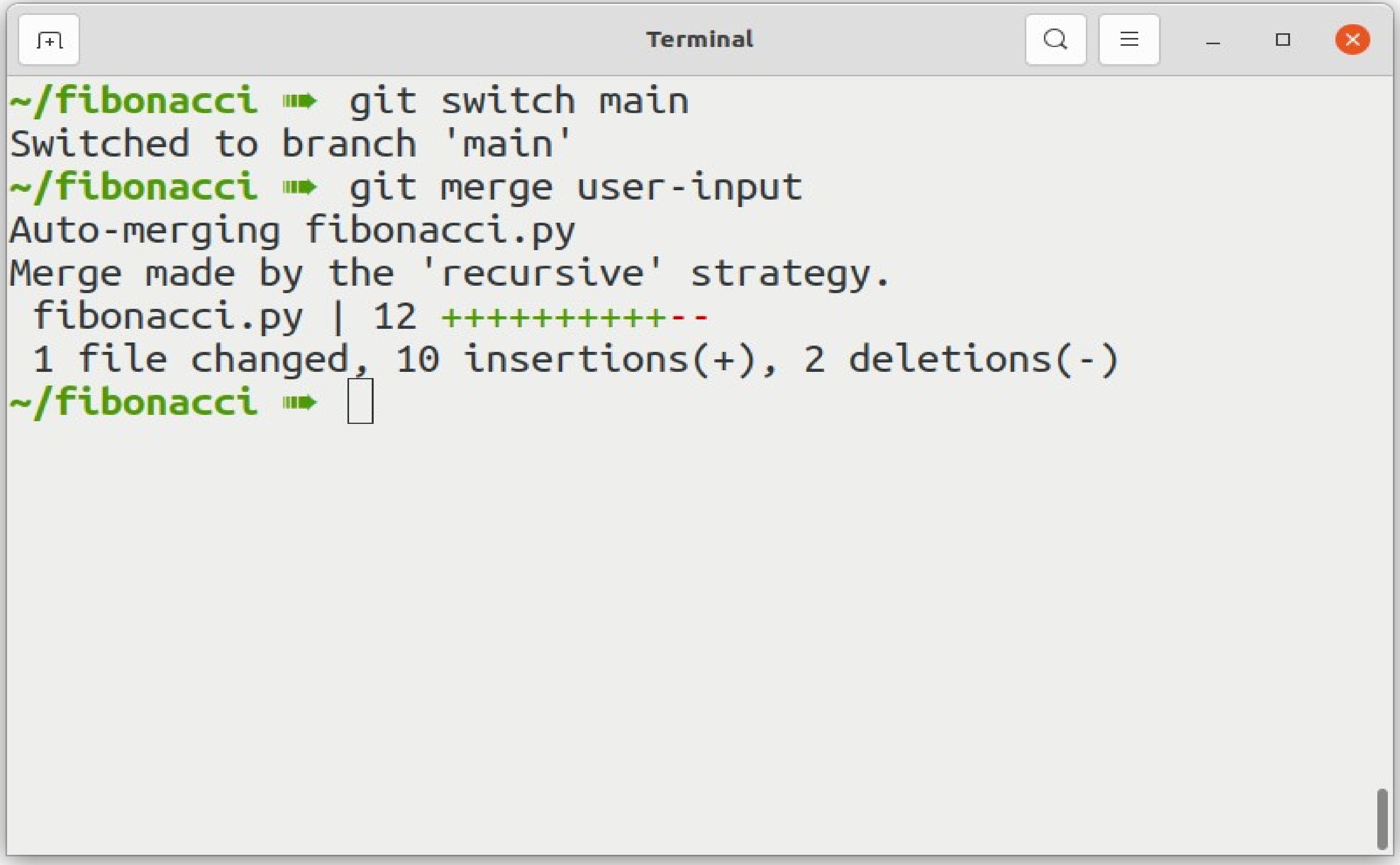
```
~/fibonacci ➠ git switch main
Switched to branch 'main'
~/fibonacci ➠ git merge user-input
```



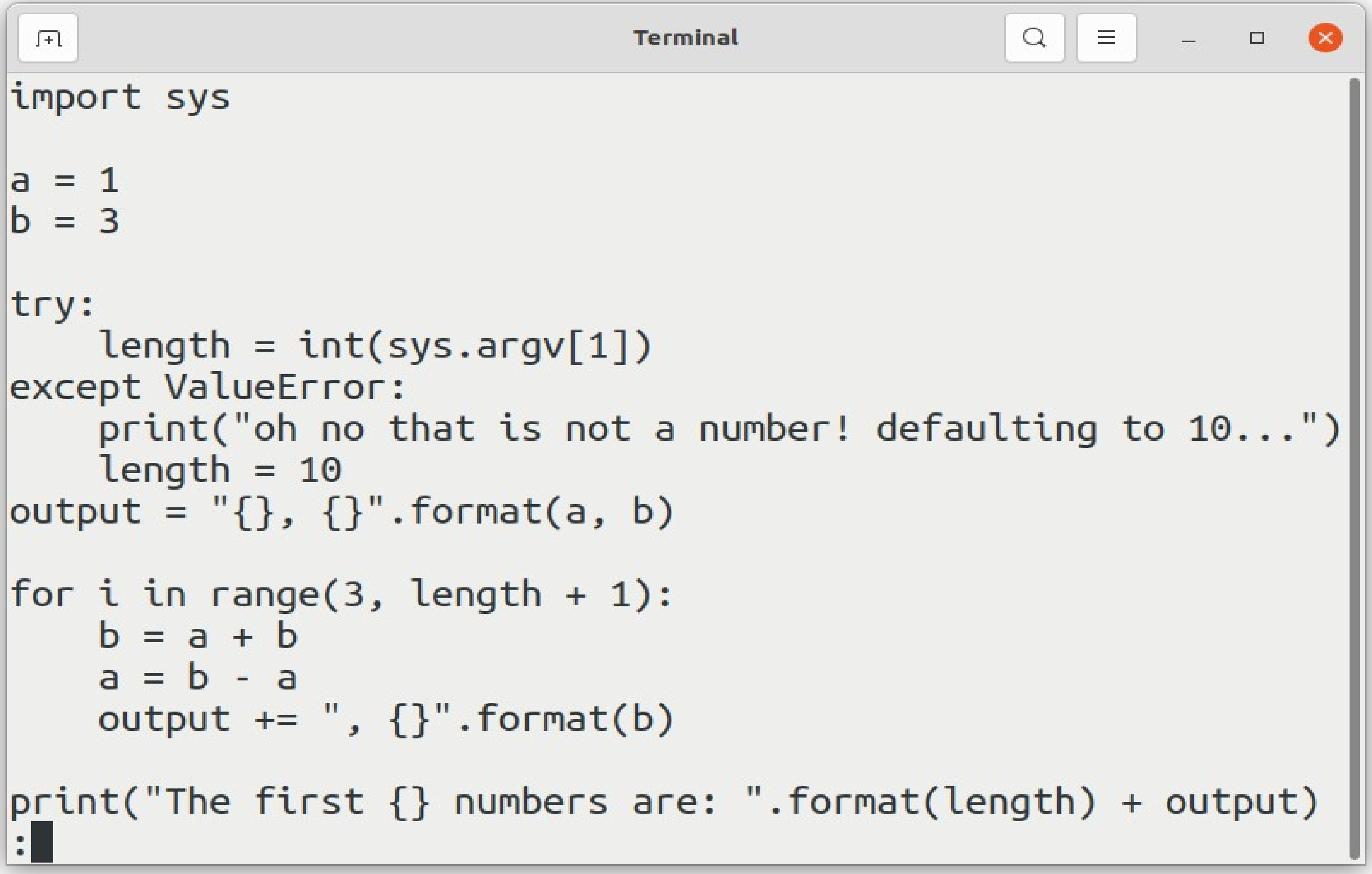
```
Merge branch 'user-input' into main
# Please enter a commit message to explain why this merge is
 necessary,
# especially if it merges an updated upstream into a topic b
ranch.
#
# Lines starting with '#' will be ignored, and an empty mess
age aborts
# the commit.
~
~
~
~
~
~
~
~
                                        1,1           All
```



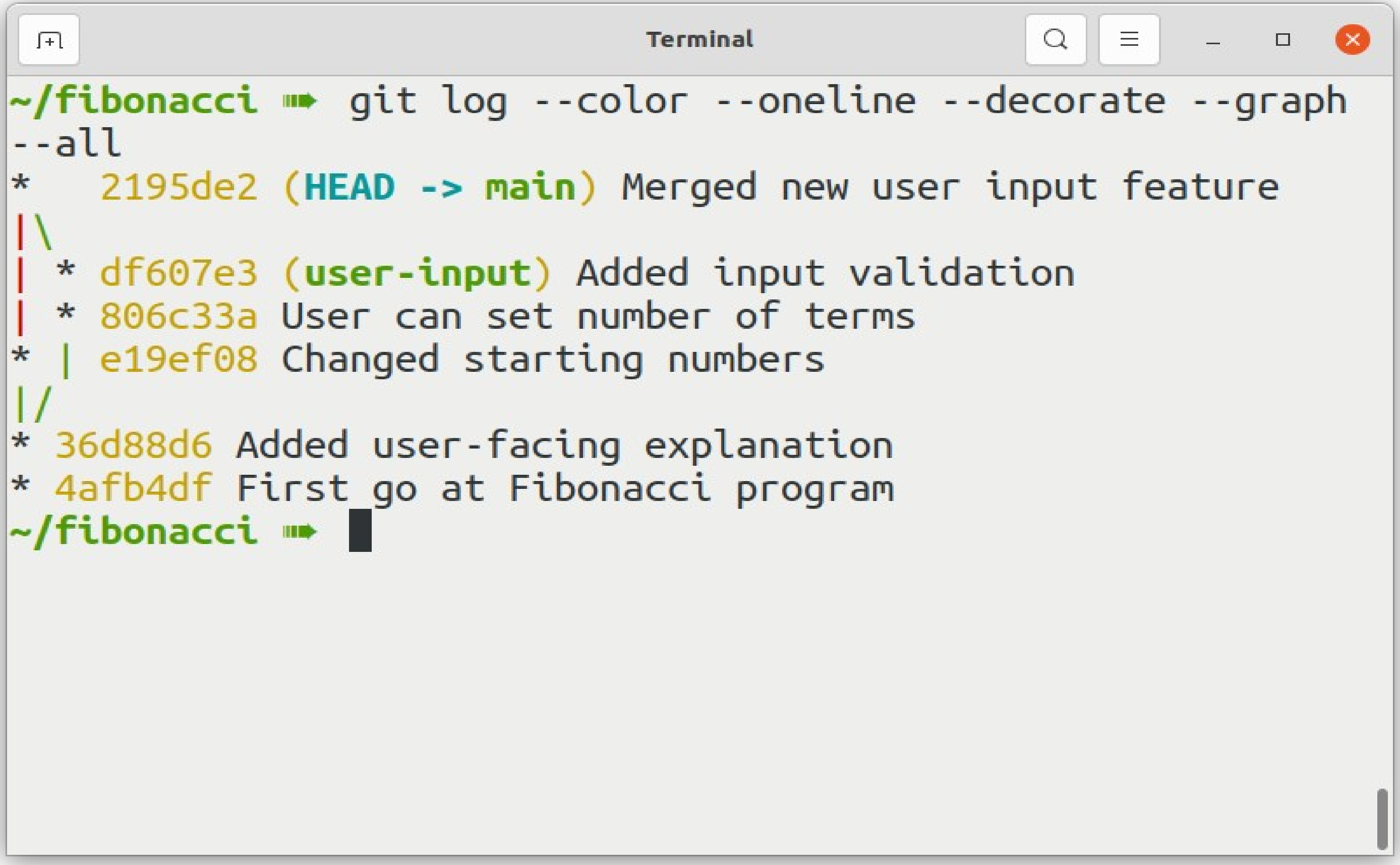
```
Merged in new user input feature
# Please enter a commit message to explain why this merge is
 necessary,
# especially if it merges an updated upstream into a topic b
ranch.
#
# Lines starting with '#' will be ignored, and an empty mess
age aborts
# the commit.
~
~
~
~
~
~
~
~
:wq
```



```
~/fibonacci ➠ git switch main
Switched to branch 'main'
~/fibonacci ➠ git merge user-input
Auto-merging fibonacci.py
Merge made by the 'recursive' strategy.
 fibonacci.py | 12 ++++++++++--
 1 file changed, 10 insertions(+), 2 deletions(-)
~/fibonacci ➠ 
```



```
import sys

a = 1
b = 3

try:
    length = int(sys.argv[1])
except ValueError:
    print("oh no that is not a number! defaulting to 10...")
    length = 10
output = "{}, {}".format(a, b)

for i in range(3, length + 1):
    b = a + b
    a = b - a
    output += ", {}".format(b)

print("The first {} numbers are: ".format(length) + output)
:
```



Terminal

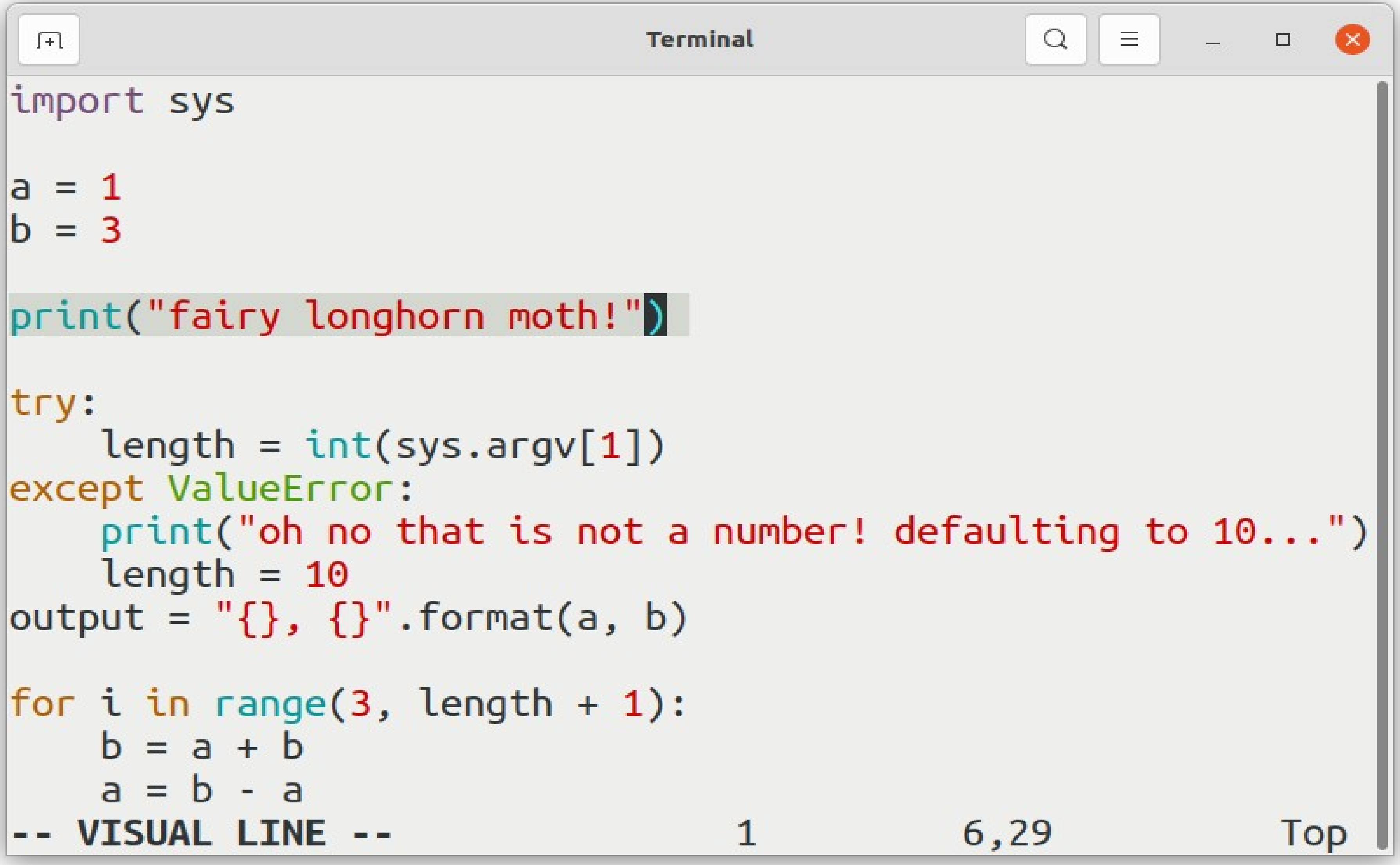
```
~/fibonacci ➠ git log --color --oneline --decorate --graph
--all
*   2195de2 (HEAD -> main) Merged new user input feature
|\
| * df607e3 (user-input) Added input validation
| * 806c33a User can set number of terms
* | e19ef08 Changed starting numbers
|/
* 36d88d6 Added user-facing explanation
* 4afb4df First go at Fibonacci program
~/fibonacci ➠ 
```

Merging III: oh no

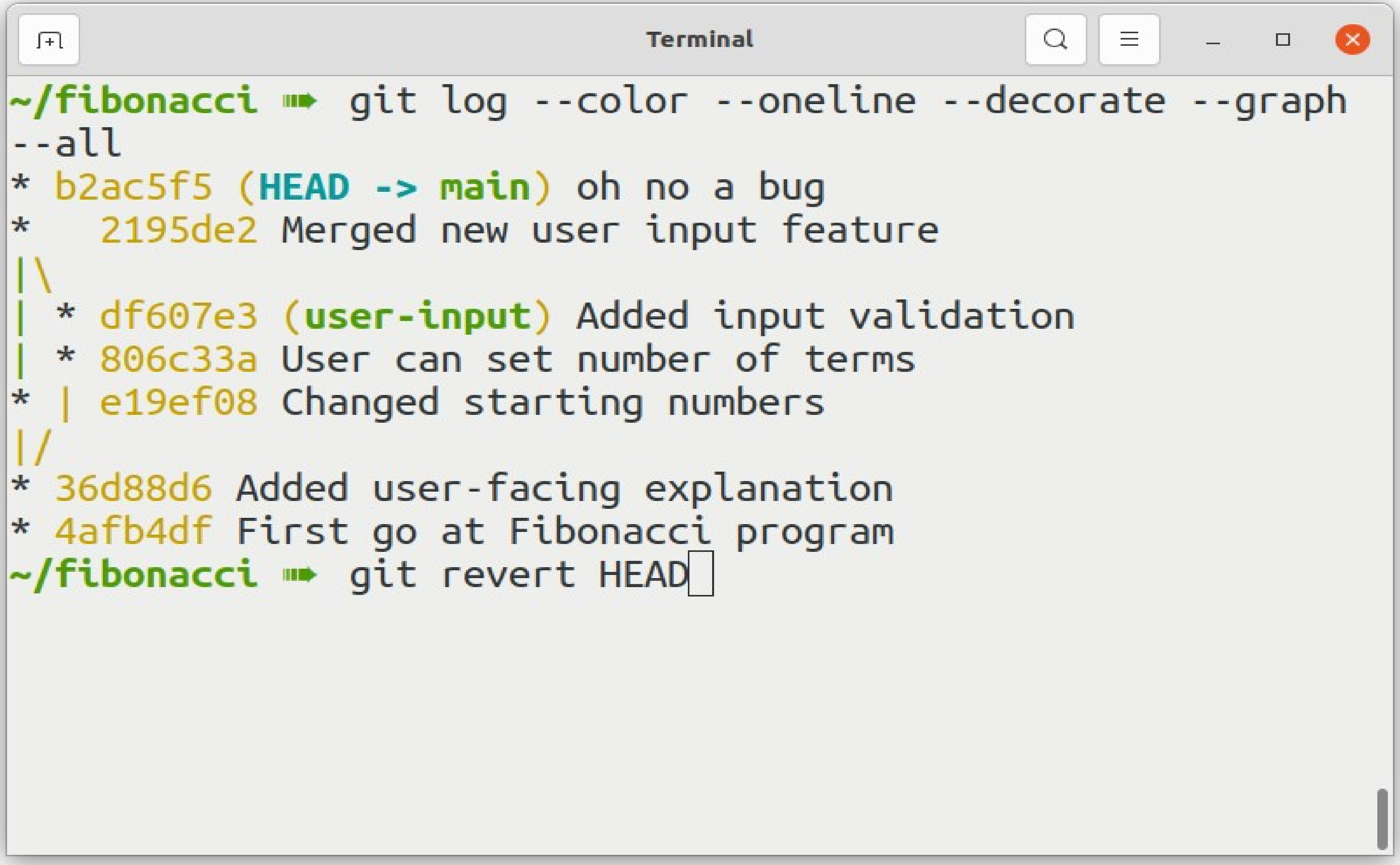
- Sometimes, git can’t merge cleanly
- This is called a merge conflict
- You have to resolve it manually
  - Choose which version of the conflict is “correct”
  - It could be a bit of both branches that you want to keep
- You should take care when doing this – look it up online if you’re unsure

Changing History

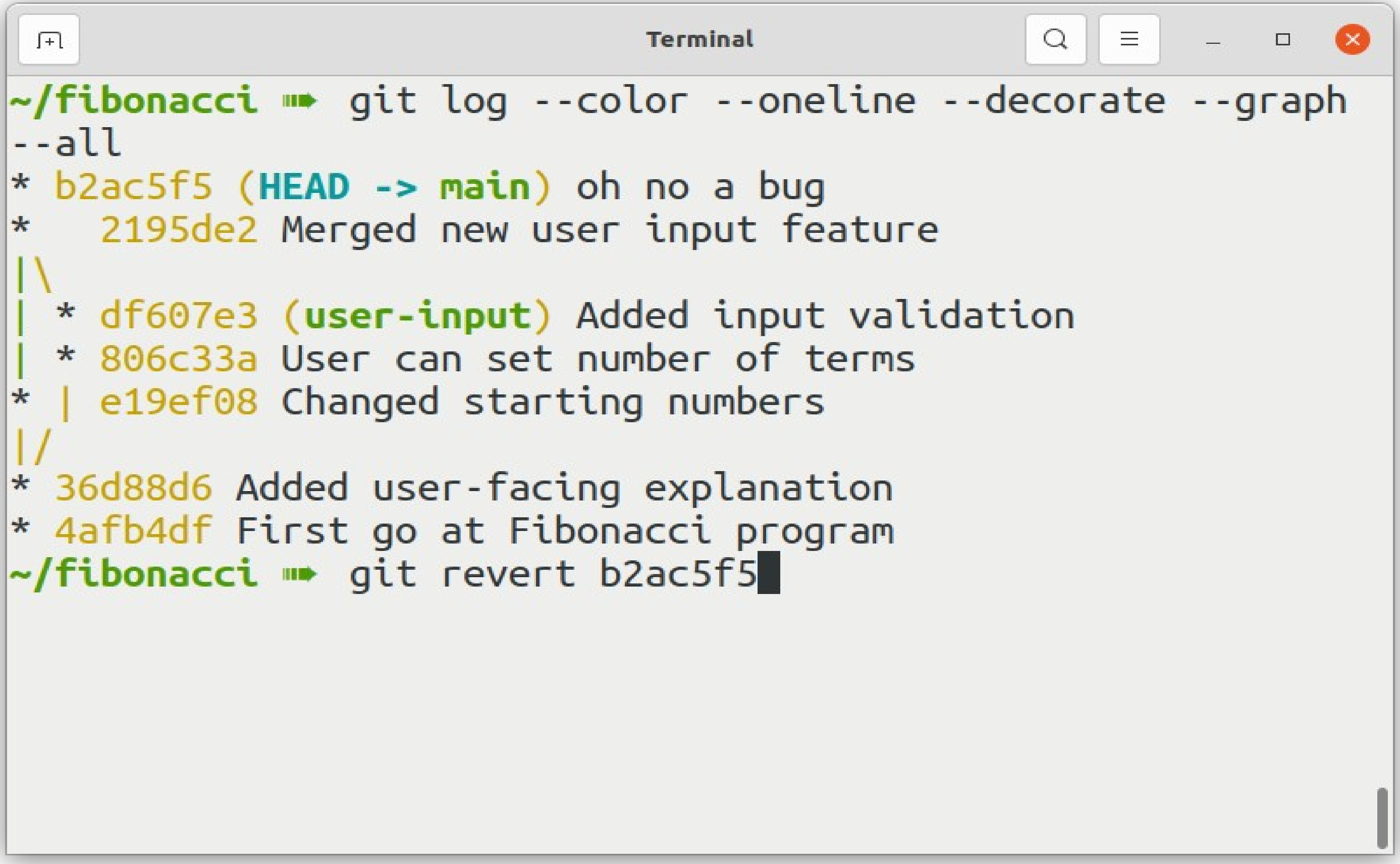
- Part of the benefit of version control is fixing mistakes!
- Use the **revert** subcommand to undo a commit
  - This will make a new commit, whose sole purpose is undoing the bad commit
- The other option: the **reset** subcommand (a little more dangerous)
  - Soft/Mixed: Doesn’t change your working directory, moves HEAD back
  - Hard: Discards all changes back to a point



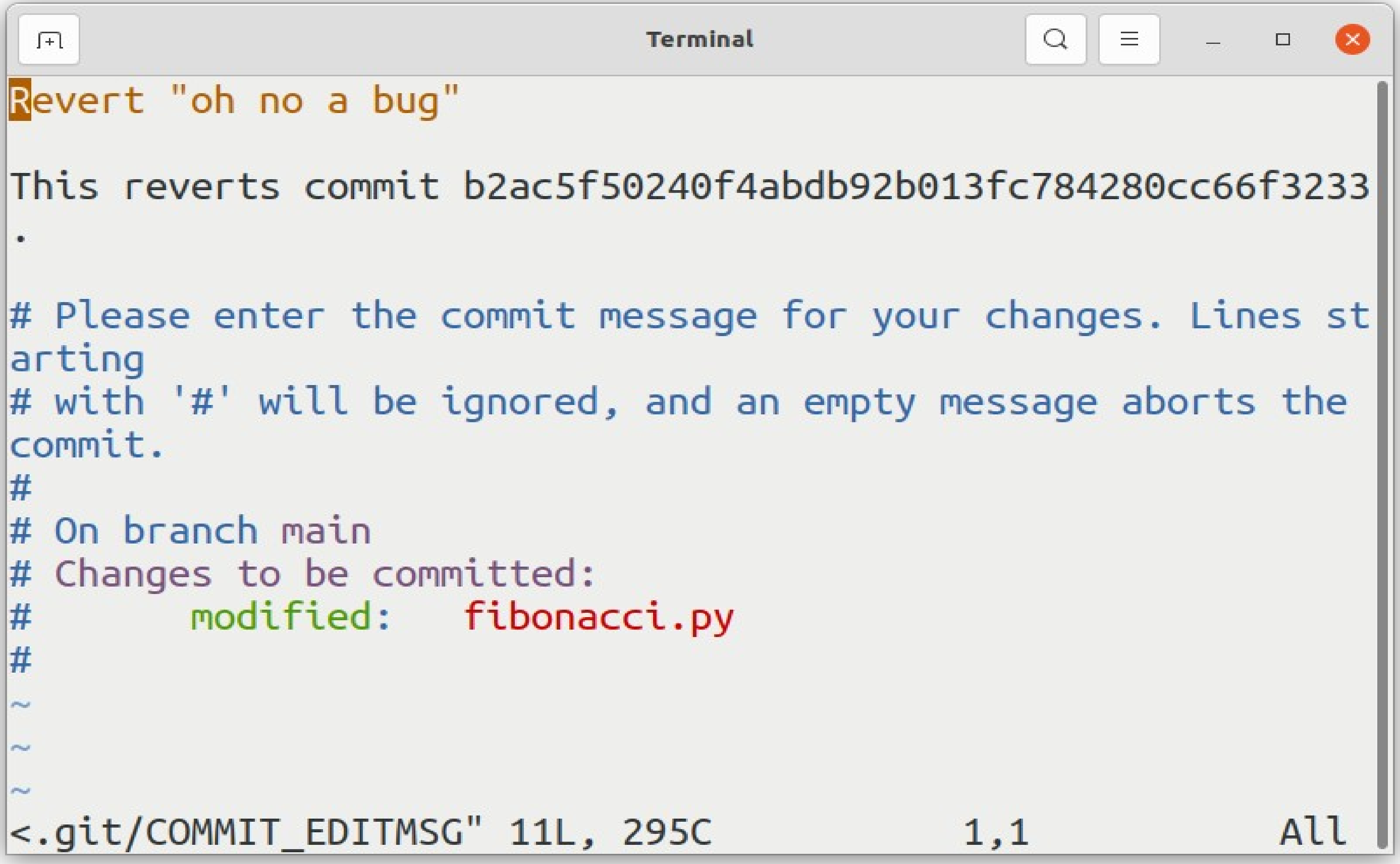
```
import sys

a = 1
b = 3

print("fairy longhorn moth!")

try:
    length = int(sys.argv[1])
except ValueError:
    print("oh no that is not a number! defaulting to 10...")
    length = 10
output = "{}, {}".format(a, b)

for i in range(3, length + 1):
    b = a + b
    a = b - a
-- VISUAL LINE --                    1          6,29          Top
```



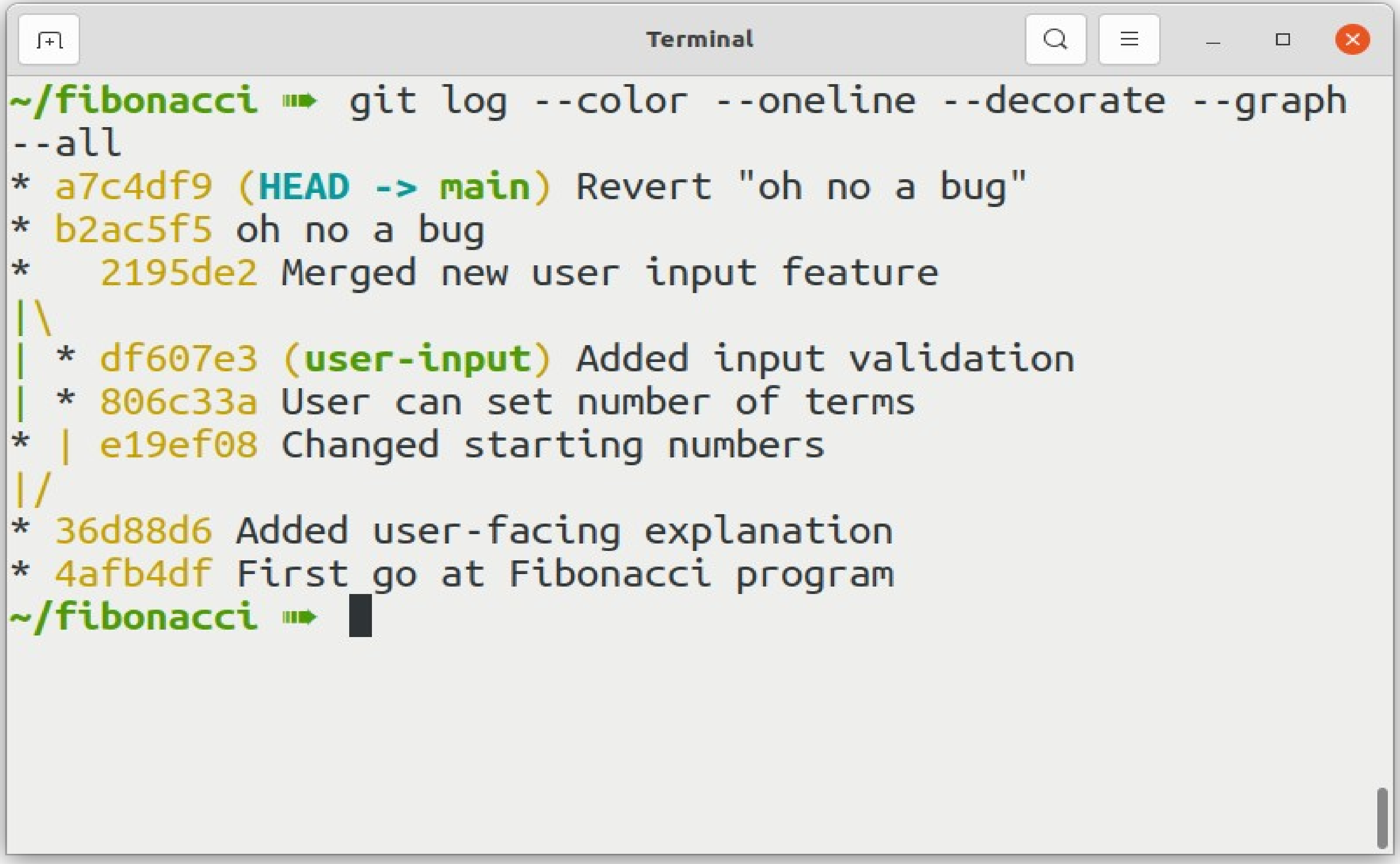
```
~/fibonacci ➠ git log --color --oneline --decorate --graph
--all
* b2ac5f5 (HEAD -> main) oh no a bug
*   2195de2 Merged new user input feature
|\
| * df607e3 (user-input) Added input validation
| * 806c33a User can set number of terms
* | e19ef08 Changed starting numbers
|/
* 36d88d6 Added user-facing explanation
* 4afb4df First go at Fibonacci program
~/fibonacci ➠ git revert HEAD
```



```
~/fibonacci ➠ git log --color --oneline --decorate --graph
--all
* b2ac5f5 (HEAD -> main) oh no a bug
*   2195de2 Merged new user input feature
|\
| * df607e3 (user-input) Added input validation
| * 806c33a User can set number of terms
* | e19ef08 Changed starting numbers
|/
* 36d88d6 Added user-facing explanation
* 4afb4df First go at Fibonacci program
~/fibonacci ➠ git revert b2ac5f5
```



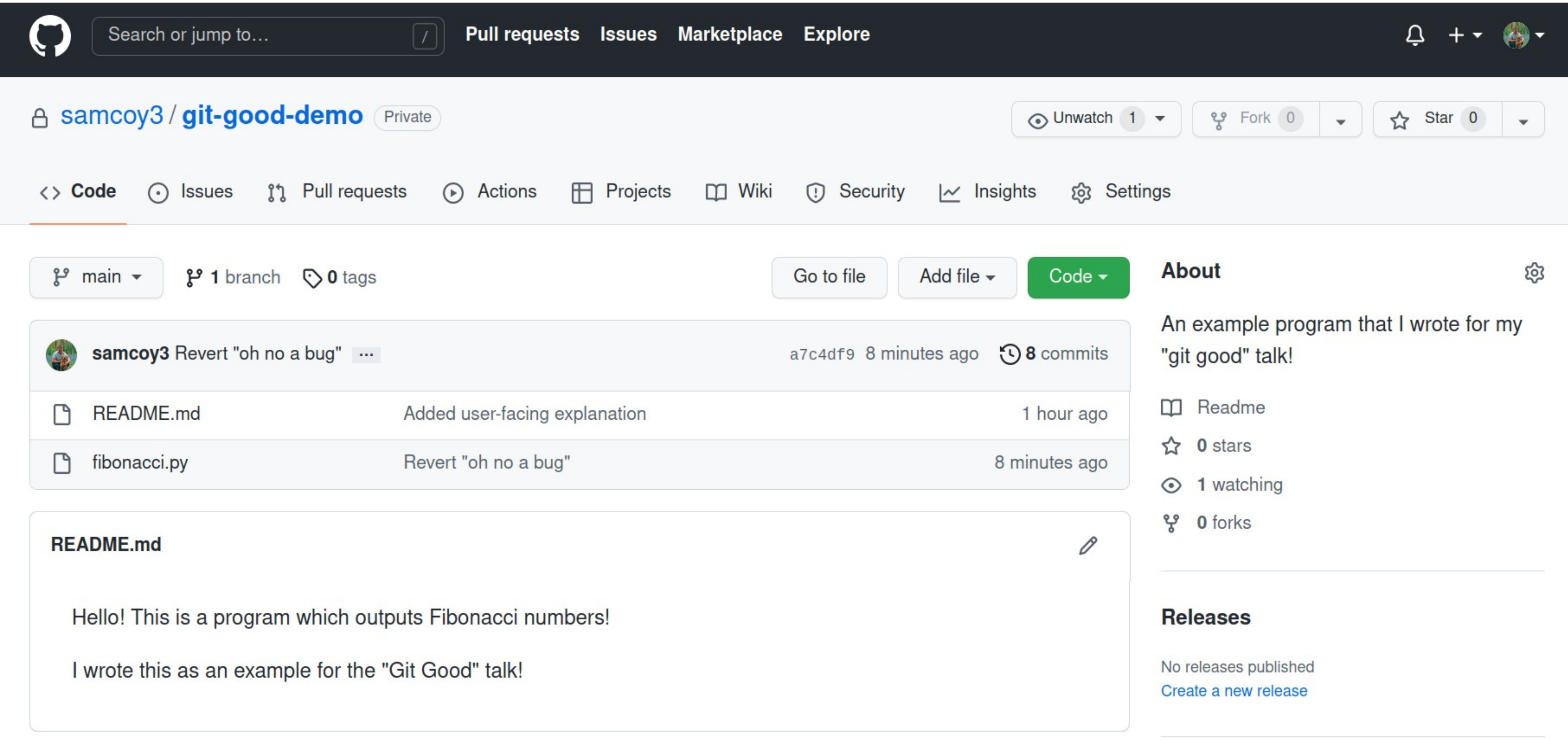
```
Revert "oh no a bug"

This reverts commit b2ac5f50240f4abdb92b013fc784280cc66f3233
.

# Please enter the commit message for your changes. Lines st
arting
# with '#' will be ignored, and an empty message aborts the
commit.
#
# On branch main
# Changes to be committed:
#       modified:   fibonacci.py
#
~
~
~
<.git/COMMIT_EDITMSG" 11L, 295C                1,1          All
```



```
~/fibonacci ➠ git log --color --oneline --decorate --graph
--all
* a7c4df9 (HEAD -> main) Revert "oh no a bug"
* b2ac5f5 oh no a bug
*   2195de2 Merged new user input feature
|\
| * df607e3 (user-input) Added input validation
| * 806c33a User can set number of terms
* | e19ef08 Changed starting numbers
|/
* 36d88d6 Added user-facing explanation
* 4afb4df First go at Fibonacci program
~/fibonacci ➠ 
```

Remote Work

- Remote repos are versions of a repo that live online
- The **remote** subcommand lets you manage them
- This is how we collaborate!
- GitHub, GitLab offer remote repo hosting
  - Can also do it yourself



samcoy3 / git-good-demo Private

1 branch 0 tags

samcoy3 Revert "oh no a bug" … a7c4df9 8 minutes ago 8 commits

README.md	Added user-facing explanation	1 hour ago
fibonacci.py	Revert "oh no a bug"	8 minutes ago

README.md

Hello! This is a program which outputs Fibonacci numbers!

I wrote this as an example for the "Git Good" talk!

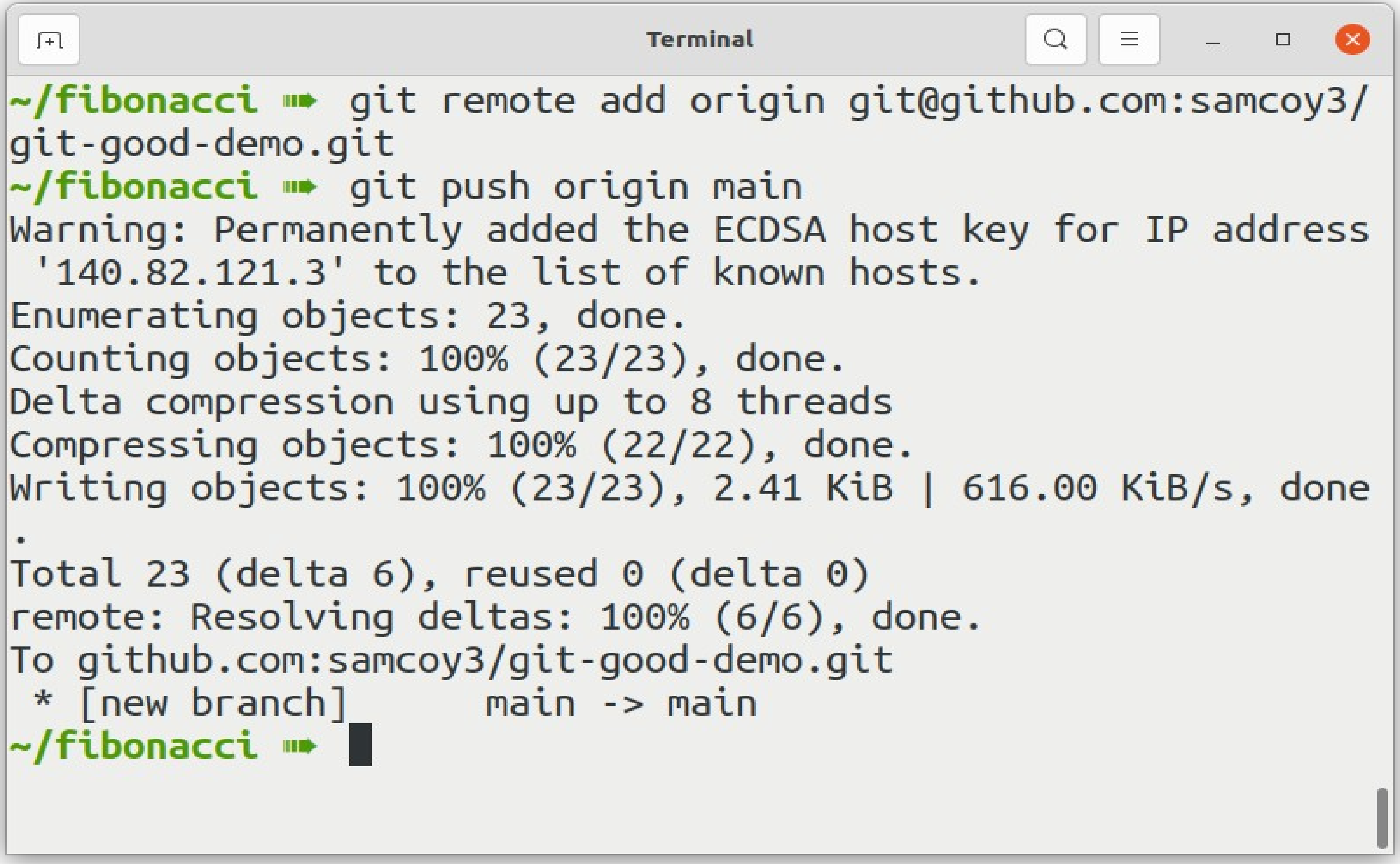
About

An example program that I wrote for my "git good" talk!

Readme
0 stars
1 watching
0 forks

Releases

No releases published
Create a new release



```
~/fibonacci ➠ git remote add origin git@github.com:samcoy3/
git-good-demo.git
~/fibonacci ➠ git push origin main
Warning: Permanently added the ECDSA host key for IP address
 '140.82.121.3' to the list of known hosts.
Enumerating objects: 23, done.
Counting objects: 100% (23/23), done.
Delta compression using up to 8 threads
Compressing objects: 100% (22/22), done.
Writing objects: 100% (23/23), 2.41 KiB | 616.00 KiB/s, done
.
Total 23 (delta 6), reused 0 (delta 0)
remote: Resolving deltas: 100% (6/6), done.
To github.com:samcoy3/git-good-demo.git
 * [new branch]      main -> main
~/fibonacci ➠ 
```

Remotes and Cloning

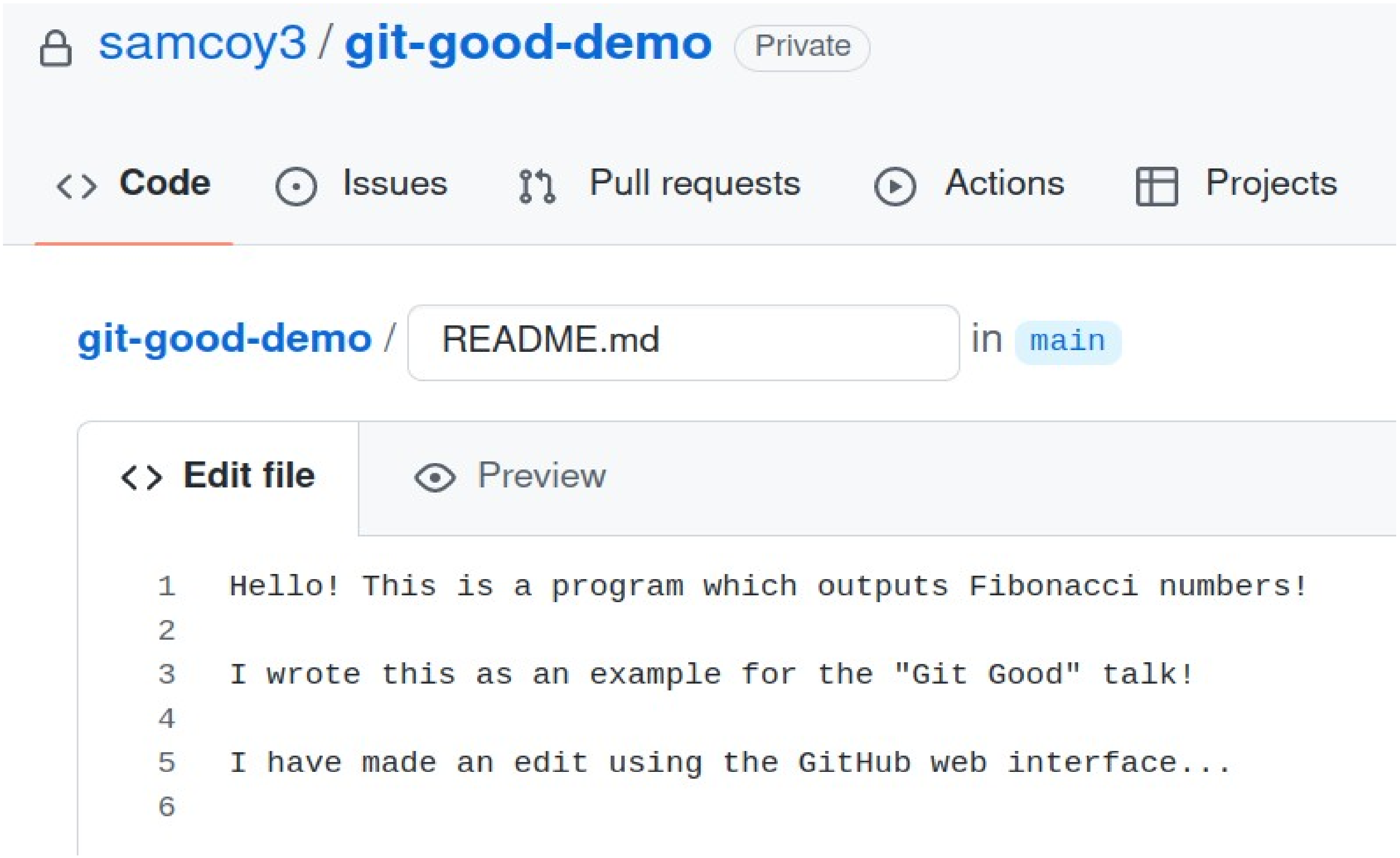
- You can get a local copy of a remote repo by cloning it
  - Use the **clone** subcommand
- Depending on permissions, you can then contribute to the repo
- Some software is distributed using remote git repos

Remotes and Branches

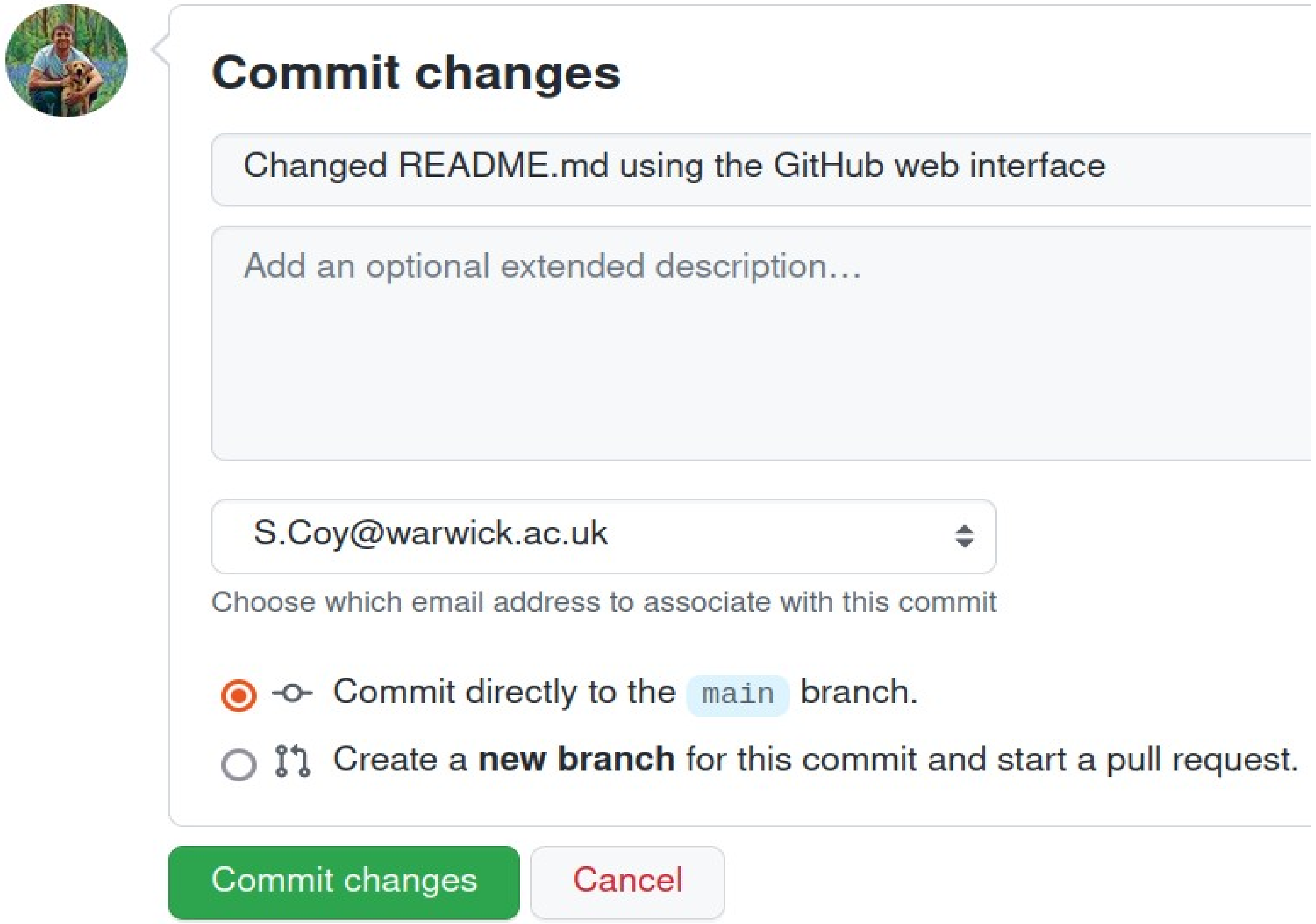
- Branches locally can correspond to branches in a remote
  - We call them <remote>/<branch>, e.g. origin/main
- You can also have branches locally that aren’t on the remote (and vice versa)

Fetch, Push, and Pull

- Fetching updates what the local repo knows about the remote repo
- Pushing updates the remote branch with updates from the local branch
- Pulling a branch updates the local branch with updates from the remote branch
  - Basically fetch + merge



```
Hello! This is a program which outputs Fibonacci numbers!

I wrote this as an example for the "Git Good" talk!

I have made an edit using the GitHub web interface...

```



Commit changes

Changed README.md using the GitHub web interface

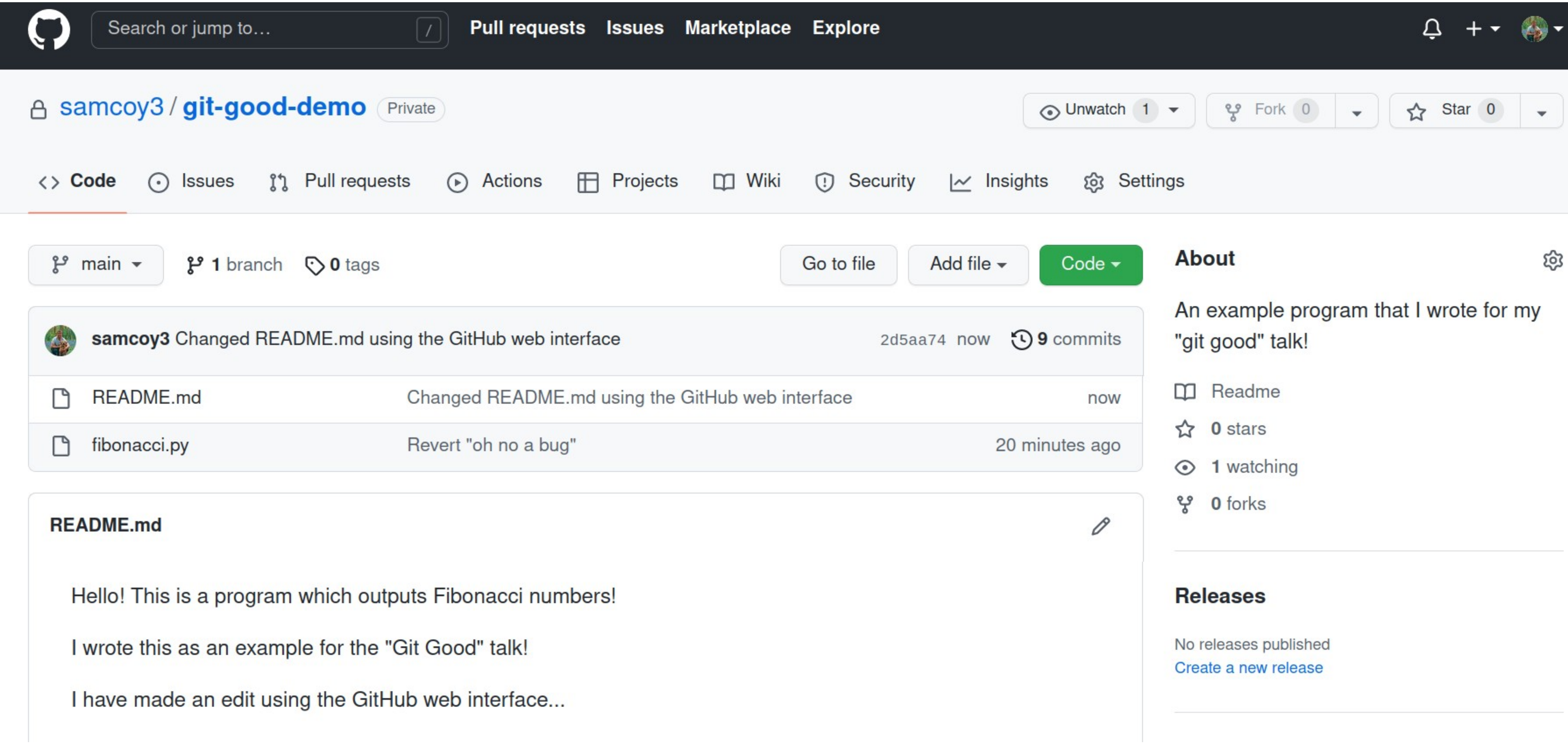
Add an optional extended description…

S.Coy@warwick.ac.uk

Choose which email address to associate with this commit

[✓ Commit directly to the `main` branch.]

Create a **new branch** for this commit and start a pull request.



samcoy3 Changed README.md using the GitHub web interface 2d5aa74 now 9 commits

README.md	Changed README.md using the GitHub web interface	now
fibonacci.py	Revert "oh no a bug"	20 minutes ago

README.md

Hello! This is a program which outputs Fibonacci numbers!

I wrote this as an example for the "Git Good" talk!

I have made an edit using the GitHub web interface...

About

An example program that I wrote for my "git good" talk!

Readme

0 stars

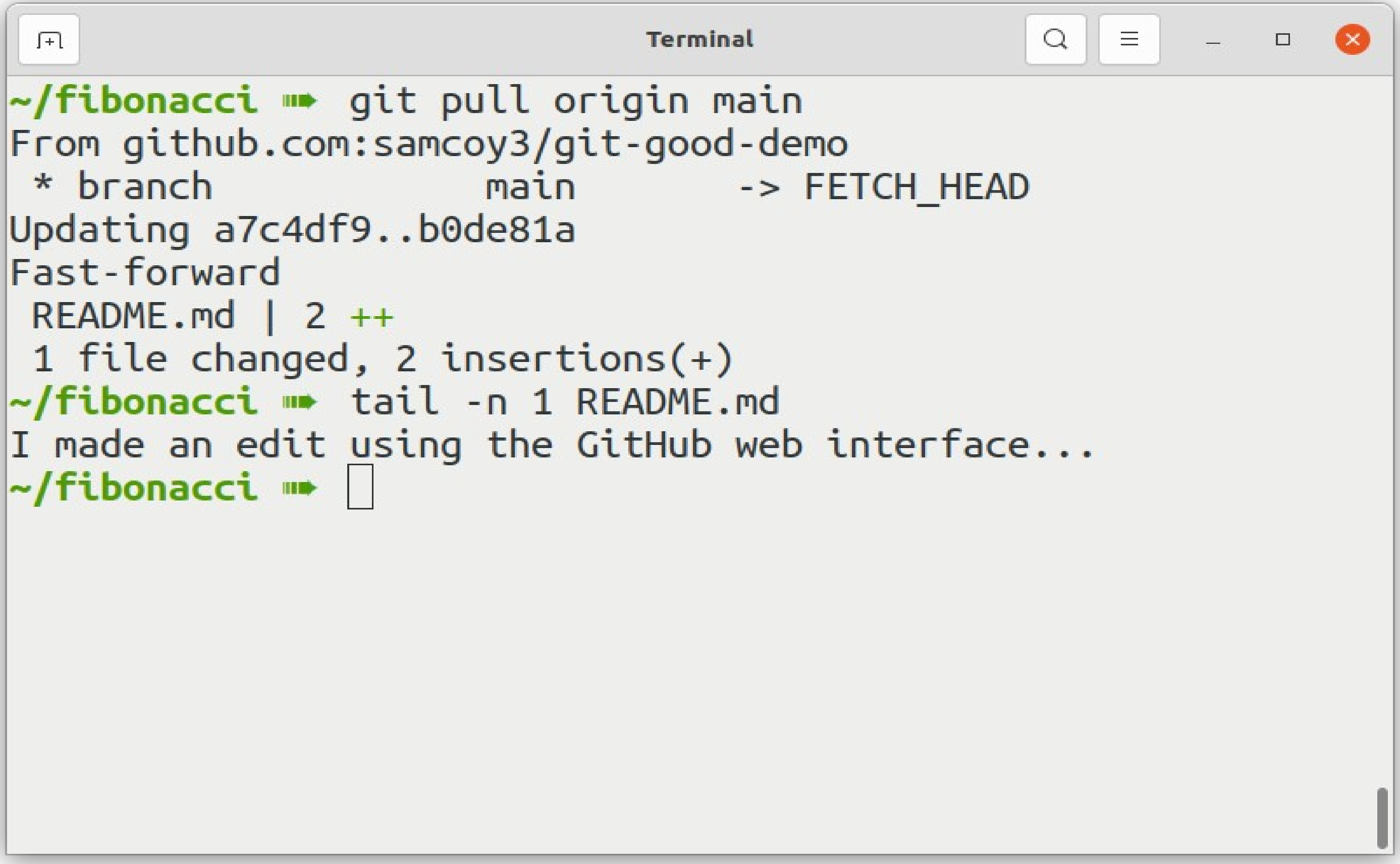
1 watching

0 forks

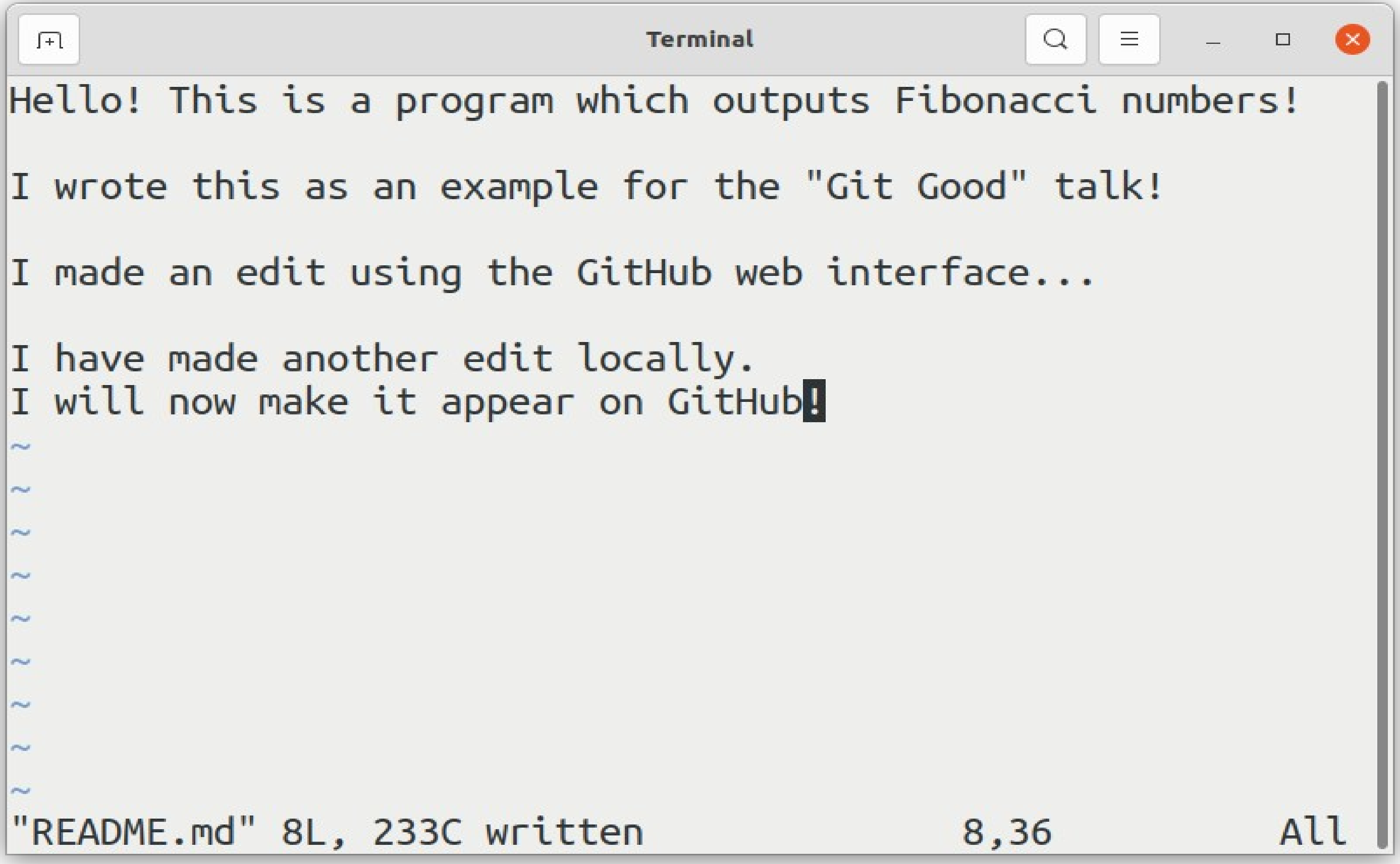
Releases

No releases published

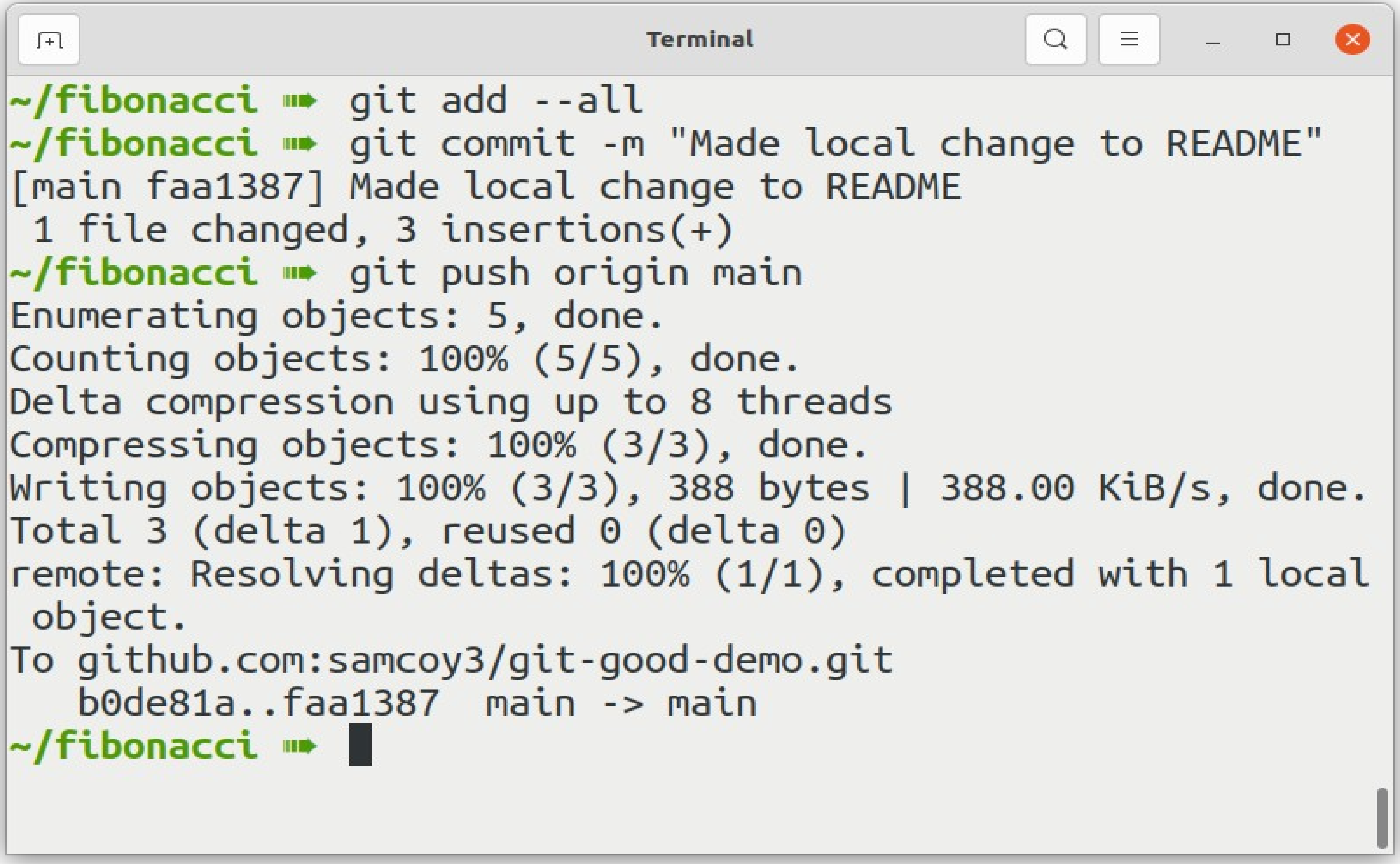
Create a new release



```
~/fibonacci ➠ git pull origin main
From github.com:samcoy3/git-good-demo
 * branch            main       -> FETCH_HEAD
Updating a7c4df9..b0de81a
Fast-forward
 README.md | 2 ++
 1 file changed, 2 insertions(+)
~/fibonacci ➠ tail -n 1 README.md
I made an edit using the GitHub web interface...
~/fibonacci ➠ 
```

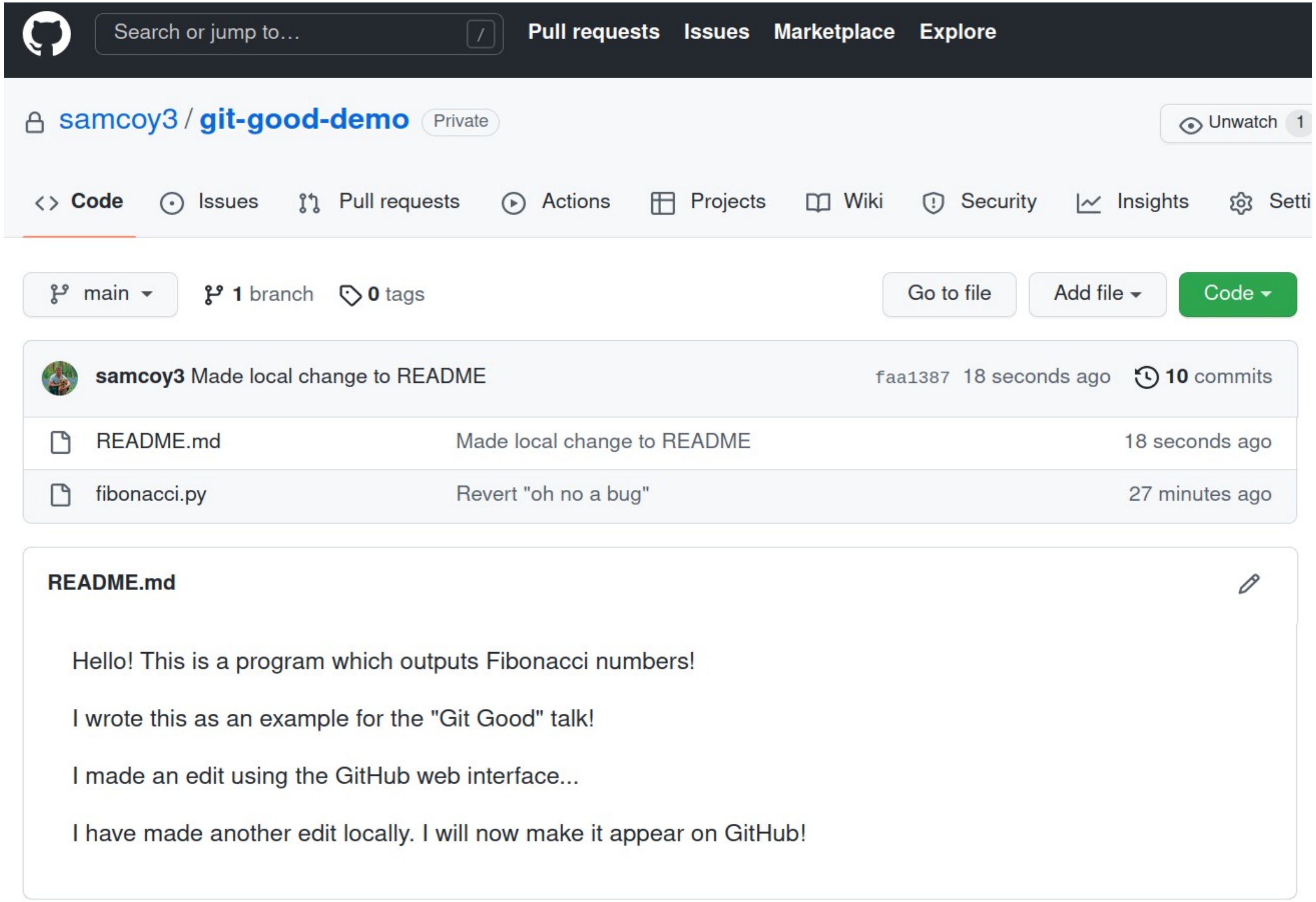


```
Hello! This is a program which outputs Fibonacci numbers!

I wrote this as an example for the "Git Good" talk!

I made an edit using the GitHub web interface...

I have made another edit locally.
I will now make it appear on GitHub!
~
~
~
~
~
~
~
~
~
"README.md" 8L, 233C written                    8,36           All
```



Terminal

```
~/fibonacci ➠ git add --all
~/fibonacci ➠ git commit -m "Made local change to README"
[main faa1387] Made local change to README
 1 file changed, 3 insertions(+)
~/fibonacci ➠ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 388 bytes | 388.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local
 object.
To github.com:samcoy3/git-good-demo.git
   b0de81a..faa1387  main -> main
~/fibonacci ➠ 
```



samcoy3 Made local change to README faa1387 18 seconds ago 10 commits

README.md	Made local change to README	18 seconds ago
fibonacci.py	Revert "oh no a bug"	27 minutes ago

README.md

Hello! This is a program which outputs Fibonacci numbers!

I wrote this as an example for the "Git Good" talk!

I made an edit using the GitHub web interface...

I have made another edit locally. I will now make it appear on GitHub!

“Pull Requests”

- A request for the remote repo owner to make a merge
- When you’ve made changes on your own branch, and want them incorporated
- Stupid name
  - Has nothing to do with “pulling” really
  - GitLab (correctly) call them merge requests

Git In Practice: Top Tips

- DO NOT COMMIT SECRETS
- Commit little and often
  - Less painful to revert smaller changes
- Make small branches and merge them in
- Clean up dead branches (use **branch -d**)
- DO NOT COMMIT SECRETS

Hate the Command Line?

- Lots of software exists for managing git repos
  - Git GUI for Windows
  - SourceTree for Mac
- Virtually all modern IDEs have git plugins
    - ...including VSCode!

Installing

- Windows: **https://git-scm.com/download/win**
- Mac: **https://git-scm.com/download/mac**
- Both come with options to just use the command line or to download a GUI program as well
- For Linux, use the package manager of your choice
  - But on Linux it's almost definitely pre-installed

Ignoring Files

- Sometimes you don’t want to keep track of certain files
  - .jar files, __pycache__, databases
- .gitignore contains a list of regexes for git to ignore
  - You might have to make this file yourself!

Config

- Git is very configurable, and a lot can be changed
  - Default editor, commit template, global gitignore, merge tool, aliases, handling of whitespace, default login credentials….
- Use the **config** subcommand to edit them
  - Can do this on a project, user, or system level

EVERYTHING IS ON FIRE HELP

- Especially when inexperienced, easy to mess up
- Someone has messed up exactly how you have before
- **ohshitgit.com** to fix many common mistakes

So Many More Features

- We’ve barely scratched the surface of what’s available
- But you can get value out of only what’s presented here
- Some other things to check out:
  - Stashes, tags, rebasing, reflog, cherry-picking, bisect, blame, hooks…

How to learn more?

- Git Reference
  - http://git.github.io/git-reference/index.html
- Pro Git
  - https://book.git-scm.com/book/en/v2
- Learn Git Branching
  - https://learngitbranching.js.org/
- GitHub have useful help pages on many topics

Questions?