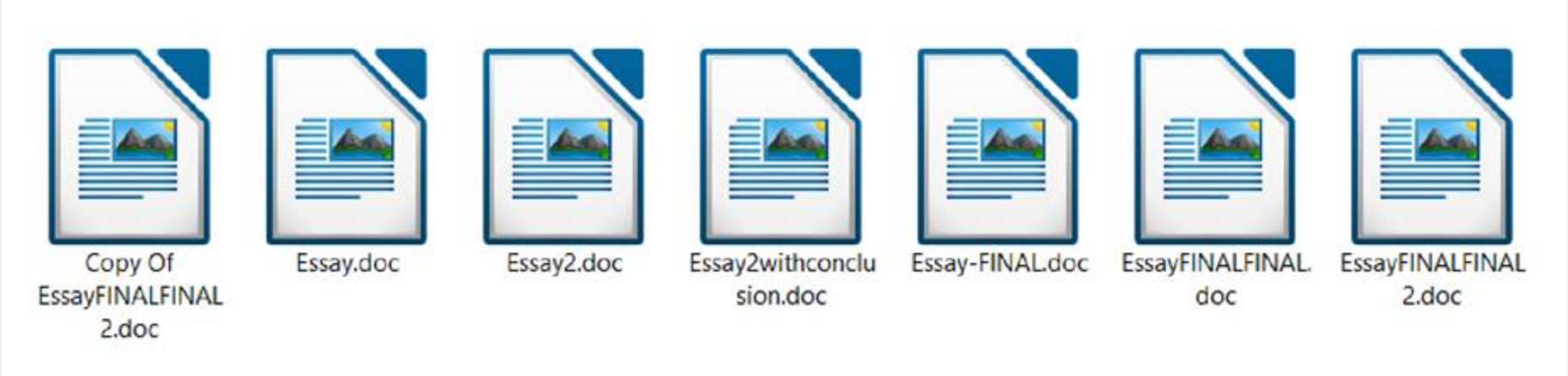
git good

Learn to use git for version control

University of Warwick
Computing Society

Has this ever happened to you?



Why is this bad?

- Multiple copies of nearly the same thing
  - Need to remember which one is the latest one
- Gets worse with multiple files
- Gets *even* worse with multiple people

So why do we do it?

- We might care about the history of a file
  - Especially important when working in a group
- We might want to experiment with changes
  - (and get back to the old state if it didn't work)

Is there a better way?

Version control

- Software that tracks changes made to files
  - Most often used for source code, but can be anything
- Lots of different ones
  - CVS, SVN, Bazaar, Perforce (), Mercurial
  - git

Why git?

- It's really popular!
  - ~90% market share
  - Many open source projects and most (tech) companies use it
- It's ergonomic and has powerful features
  - Cheap local branching, distributed model, …
  - Easier to use/learn than competitors
  - https://z.github.io/whygitisbetter/

About git

- Created by Linus Torvalds for use developing the Linux kernel
- It's free, and open source!
- Over 18 years old
  - First released in July 2005
  - Older than some of you…?

Aim of this talk

- Give an introduction to *basic* git
- Convince you that git is worth using

The command line

- Who's used the command line before?
- Git uses subcommands
  - `git <subcommand> <flags> <arguments>`
- Can easily find documentation with
  - `man git <subcommand>`

Repositories

- Repositories are just folders managed by git
  - Can be thought of as a project
  - Tracks changes over time
  - Also called a “repo”

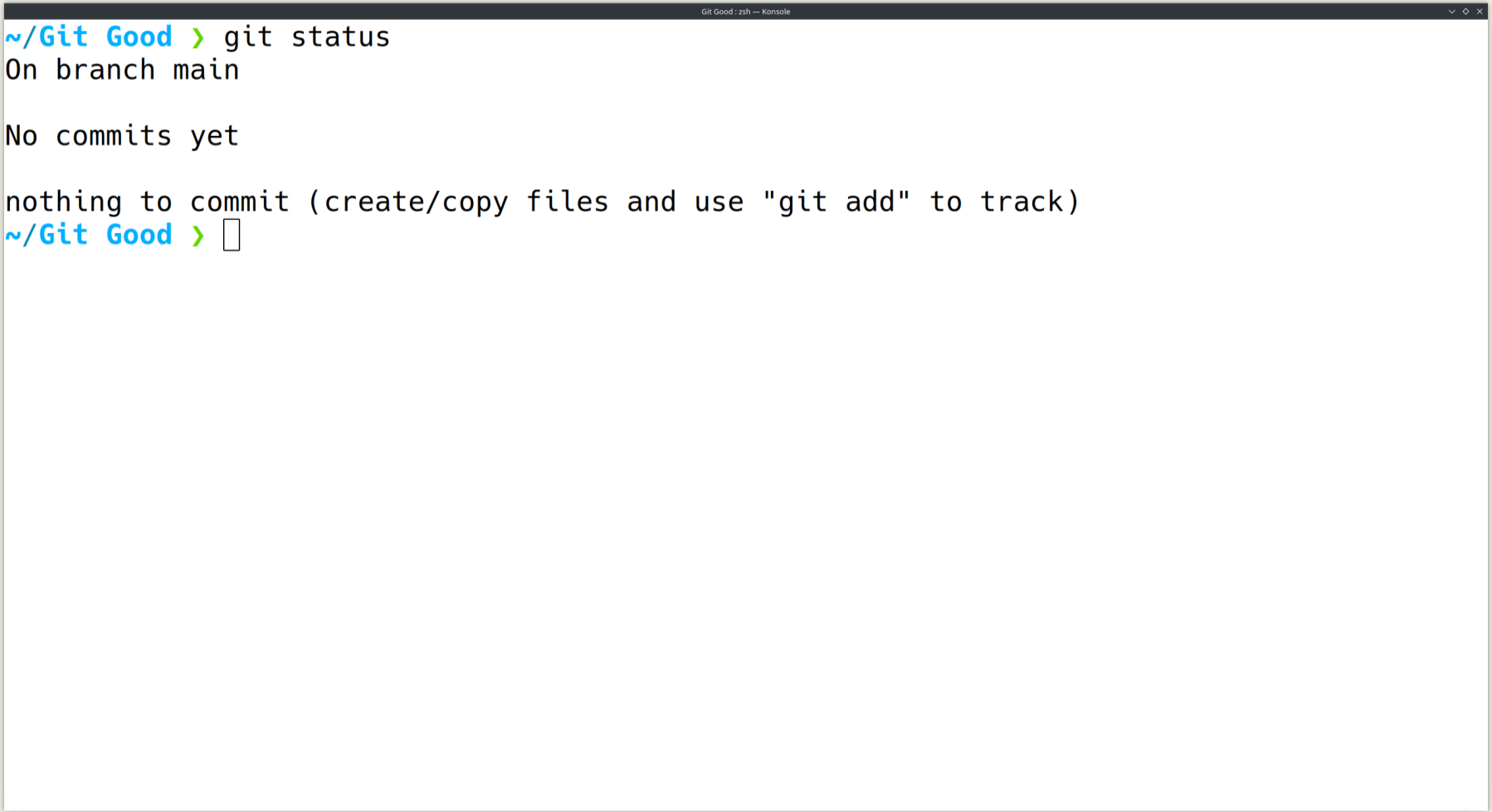
- `git init` initialises a repository in the current folder

- Internal workings stored in the `.git` folder
  - *Don’t touch this!*


Git Good : zsh — Konsole
~/Git Good ❯ ls -a
. ..
~/Git Good ❯ git init
Initialized empty Git repository in /home/edjg/Git Good/.git/
~/Git Good ❯ ls -a
. .. .git
~/Git Good ❯

Working directory

- The "state" of your project
  - What you see when you type `ls` (excluding `.git` folder)
  - If there are no changes since the last commit, we say it is "clean"
- We make changes in the working directory as we develop our code
- We can see what has changed in the working directory with the `git status` command

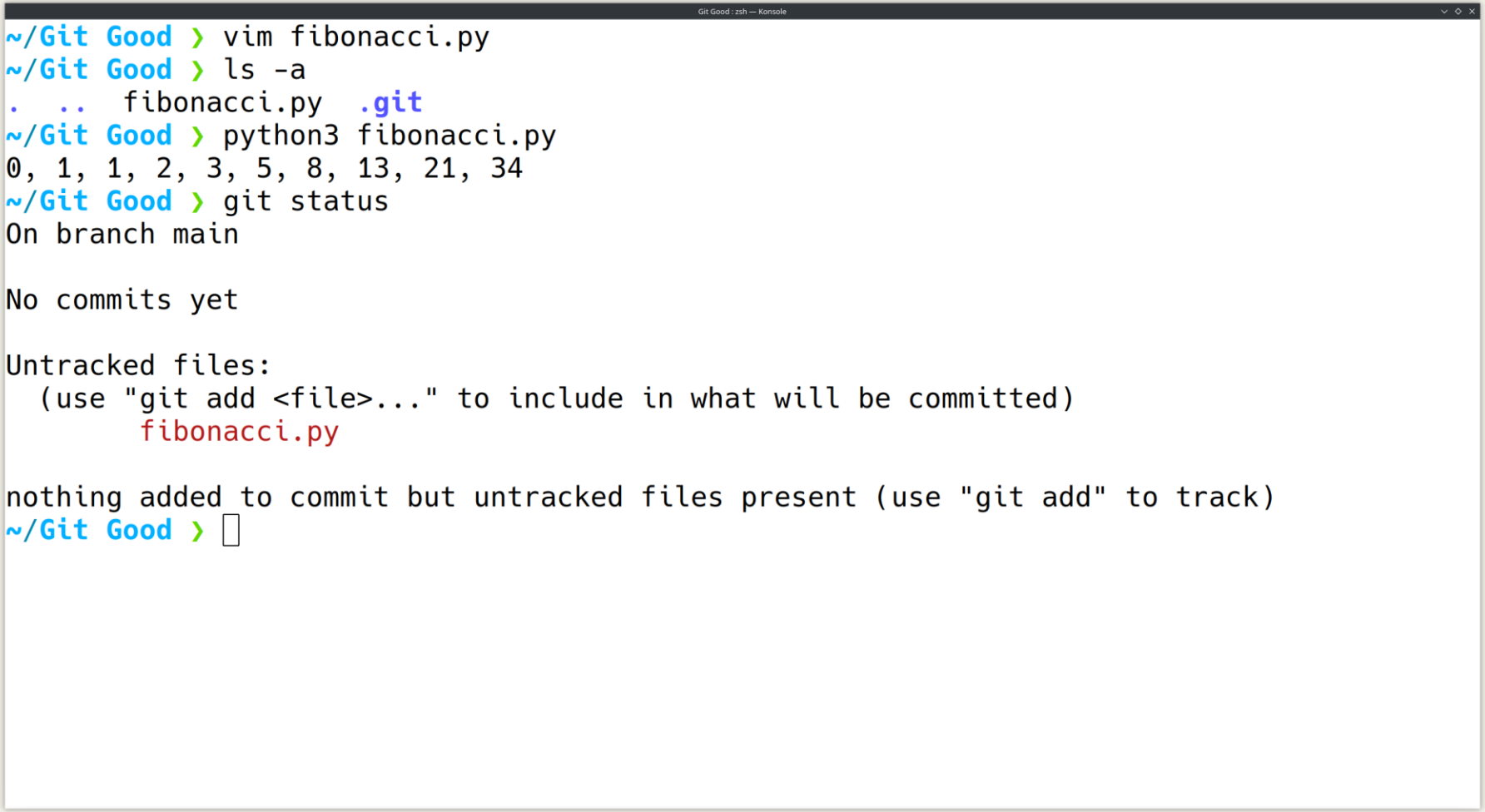

Git Good : zsh — Konsole
~/Git Good ❯ git status
On branch main
No commits yet
nothing to commit (create/copy files and use "git add" to track)
~/Git Good ❯

What is a commit?

- Commits are snapshots in history of the repository
- "Named" by hashes of their content
  - Commits can be referred to by that hash

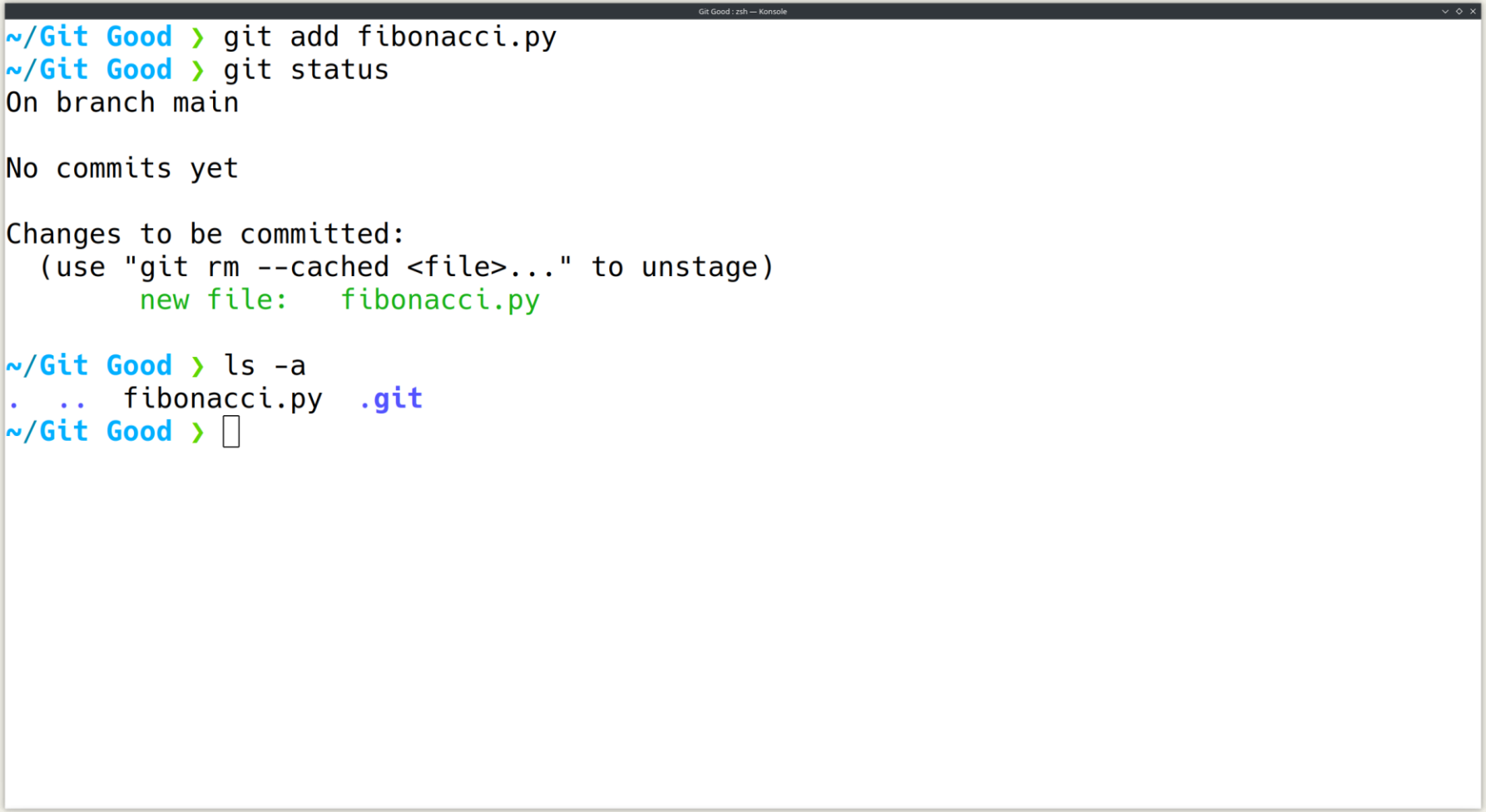
Git Good : vim — Konsole

```
a = 0
b = 1

output = f"{a}, {b}"

for i in range(3, 11):
    a, b = b, a + b
    output += f", {b}"

print(output)

~
~
~
~
~
~
~
~
~
~
~
:wq
```


Git Good : zsh — Konsole
~/Git Good ❯ vim fibonacci.py
~/Git Good ❯ ls -a
. .. fibonacci.py .git
~/Git Good ❯ python3 fibonacci.py
0, 1, 1, 2, 3, 5, 8, 13, 21, 34
~/Git Good ❯ git status
On branch main
No commits yet
Untracked files:
(use "git add <file>..." to include in what will be committed)
fibonacci.py
nothing added to commit but untracked files present (use "git add" to track)
~/Git Good ❯

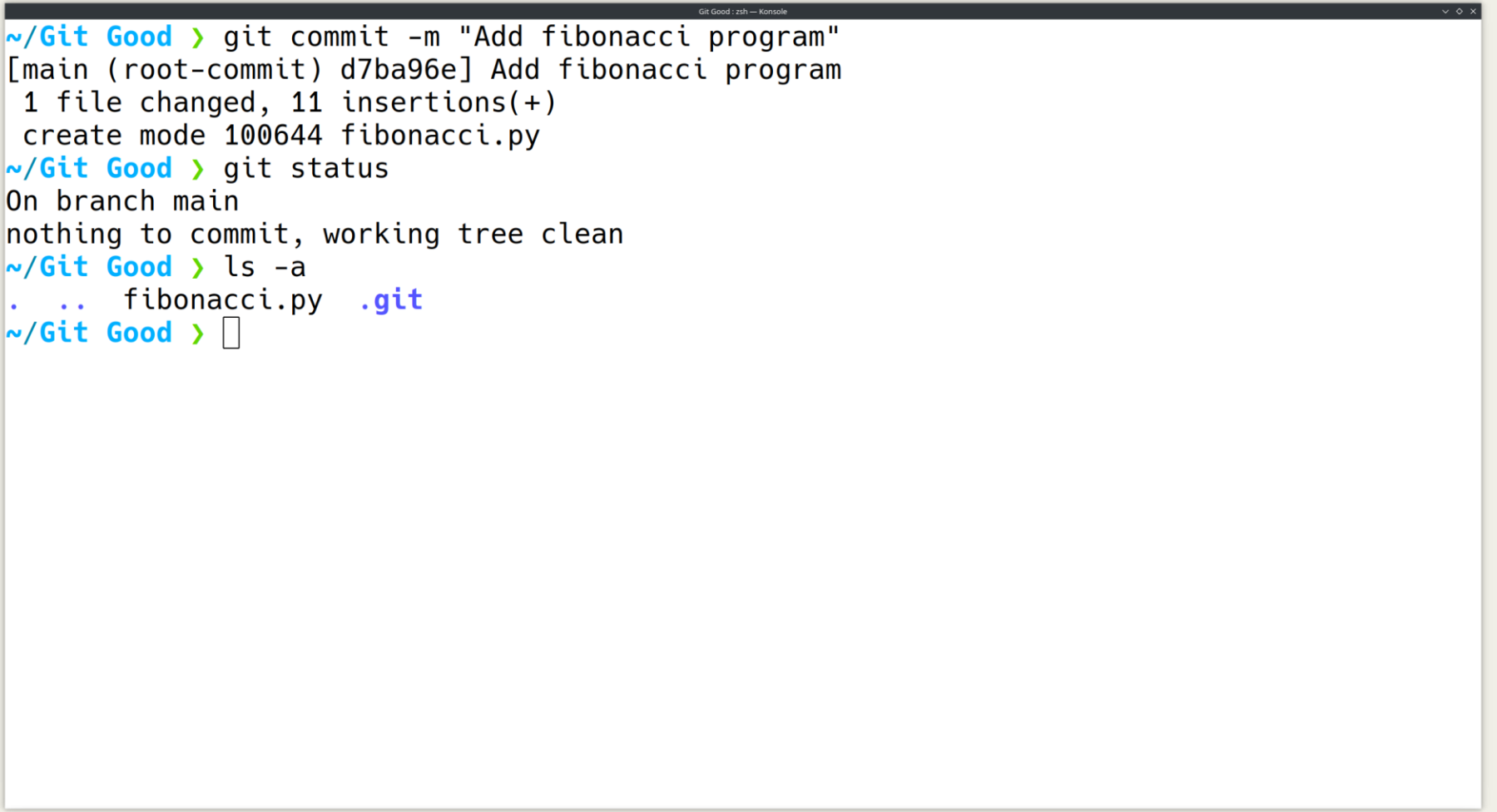
Staging area

- The changes we want to include in the next commit
  - Sometimes we only want to pick some of the changes!
- We can add things to the staging are with the `git add` command
- These will also show up when we re-run `git status`

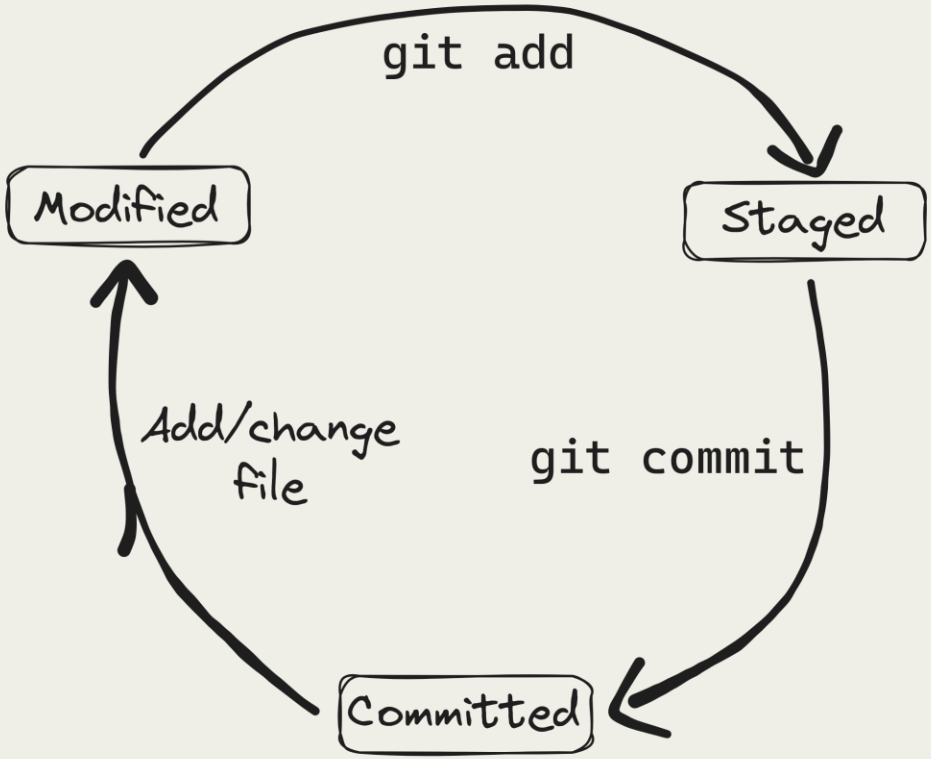

Git Good : zsh — Konsole
~/Git Good ❯ git add fibonacci.py
~/Git Good ❯ git status
On branch main
No commits yet
Changes to be committed:
(use "git rm --cached <file>..." to unstage)
new file: fibonacci.py
~/Git Good ❯ ls -a
. .. fibonacci.py .git
~/Git Good ❯

Making a commit

- Taking a snapshot of the changes we picked in our staging area
- Use the `git commit` command to make one
  - Use the `-m` flag to give them short messages
    - Should be an imperative phrase


Git Good : zsh — Konsole
~/Git Good ❯ git commit -m "Add fibonacci program"
[main (root-commit) d7ba96e] Add fibonacci program
1 file changed, 11 insertions(+)
create mode 100644 fibonacci.py
~/Git Good ❯ git status
On branch main
nothing to commit, working tree clean
~/Git Good ❯ ls -a
. .. fibonacci.py .git
~/Git Good ❯

The three stages of a file

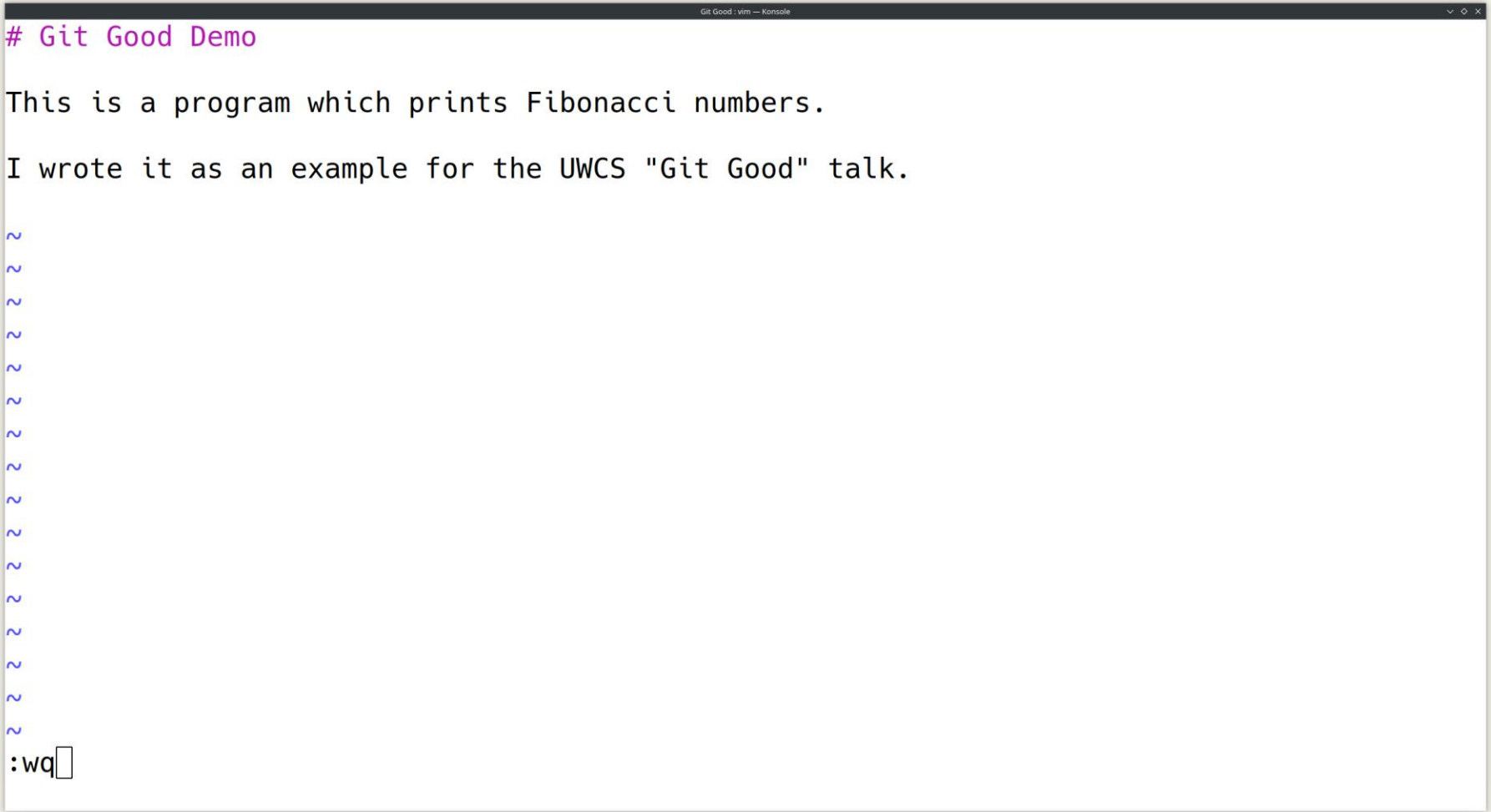


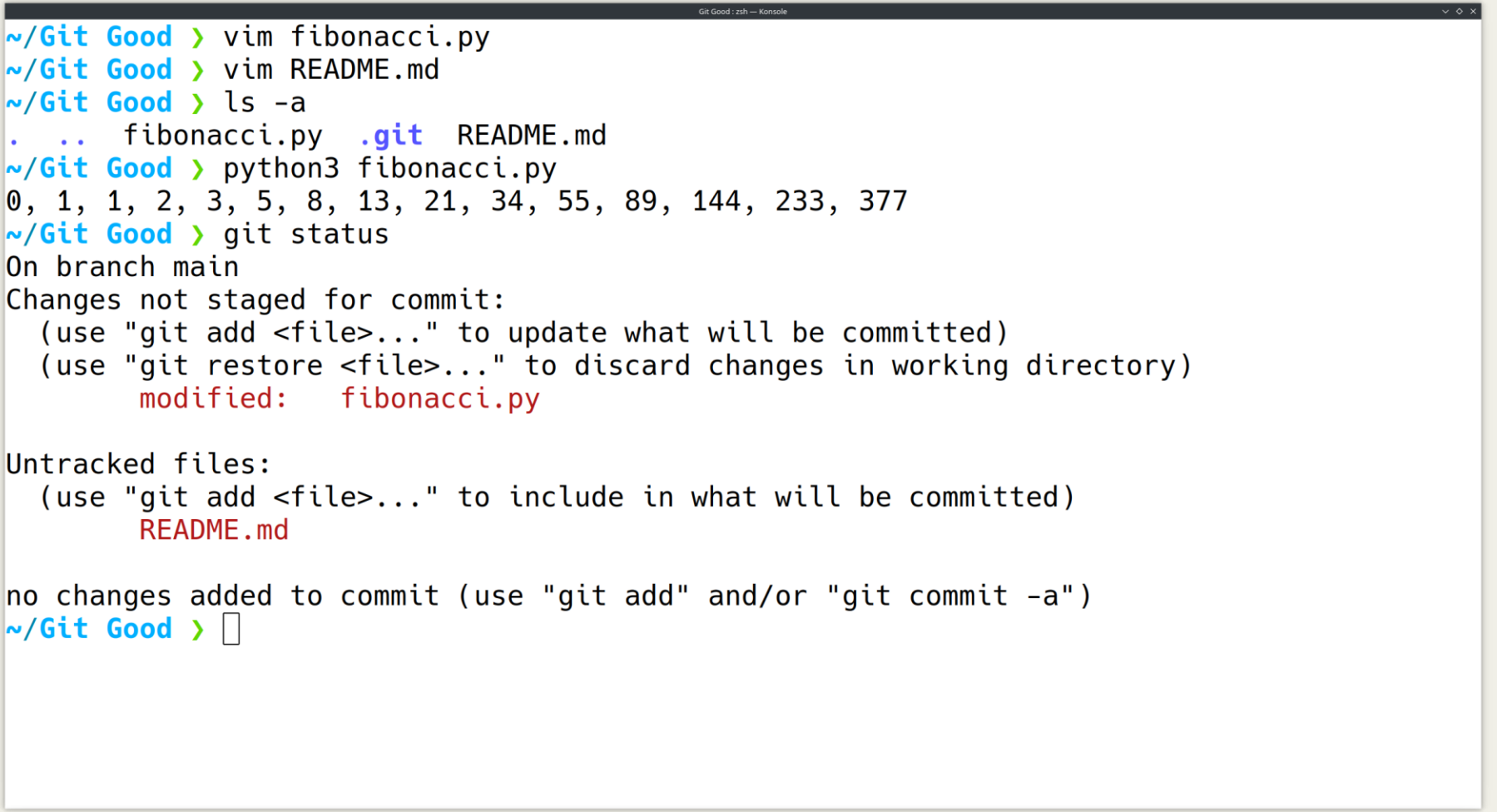
Git Good : vim — Konsole

```
a = 0
b = 1

output = f"{a}, {b}"

for i in range(3, 16):
    a, b = b, a + b
    output += f", {b}"

print(output)
```

```
~
~
~
~
~
~
~
~
~
~
~
```

6,20 All


Git Good : vim — Konsole
Git Good Demo
This is a program which prints Fibonacci numbers.
I wrote it as an example for the UWCS "Git Good" talk.
:wq


Git Good : zsh — Konsole
~/Git Good ❯ vim fibonacci.py
~/Git Good ❯ vim README.md
~/Git Good ❯ ls -a
. .. fibonacci.py .git README.md
~/Git Good ❯ python3 fibonacci.py
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377
~/Git Good ❯ git status
On branch 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: fibonacci.py
Untracked files:
(use "git add <file>..." to include in what will be committed)
README.md
no changes added to commit (use "git add" and/or "git commit -a")
~/Git Good ❯

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

~/Git Good ❯ ls -a
.  ..  fibonacci.py  .git  README.md
~/Git Good ❯
```

```
~/Git Good ❯ git commit -m "Update range and add README"
[main b1eb1e6] Update range and add README
 2 files changed, 7 insertions(+), 1 deletion(-)
 create mode 100644 README.md
~/Git Good ❯ git status
On branch main
nothing to commit, working tree clean
~/Git Good ❯ ls -a
.  ..  fibonacci.py  .git  README.md
~/Git Good ❯
```

Recap so far

- Making repositories with `git init`
- Looking at their state with `git status`
- Adding to the staging area with `git add`
- Taking snapshots of history with `git commit`

Questions?

Looking at histories with log

- The `git log` command lets us look back on our commit history
- We can use some flags to make it look prettier
  - `--color --oneline --graph –decorate --all`
  - Will use these in all examples going forward

Git Good : git — Konsole
commit b1eb1e6e54df8b4c34d4d54fa6c20a89e7a4f3e7 (HEAD -> main)
Author: EdmundGoodman <egoodman3141@gmail.com>
Date: Tue Sep 26 00:13:35 2023 +0100
Update range and add README
commit d7ba96eb41ad0c936ffe24eabb1b7fd58c49de10
Author: EdmundGoodman <egoodman3141@gmail.com>
Date: Tue Sep 26 00:07:15 2023 +0100
Add fibonacci program
~
~
~
~
~
~
~
~
~
~
~
(END)

Git Good : git — Konsole
* b1eb1e6 (HEAD -> main) Update range and add README
* d7ba96e Add fibonacci program
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
(END)

Time travelling with checkout

- The `git checkout` command lets us travel around the history of our repo
- `git checkout <name>` lets us visit commits
  - Working directory must be clean, otherwise we'd lose our changes!
- `HEAD` is a synonym for the current location history
  - Don't need to remember long hashes
- `HEAD~n` means the nth previous commit

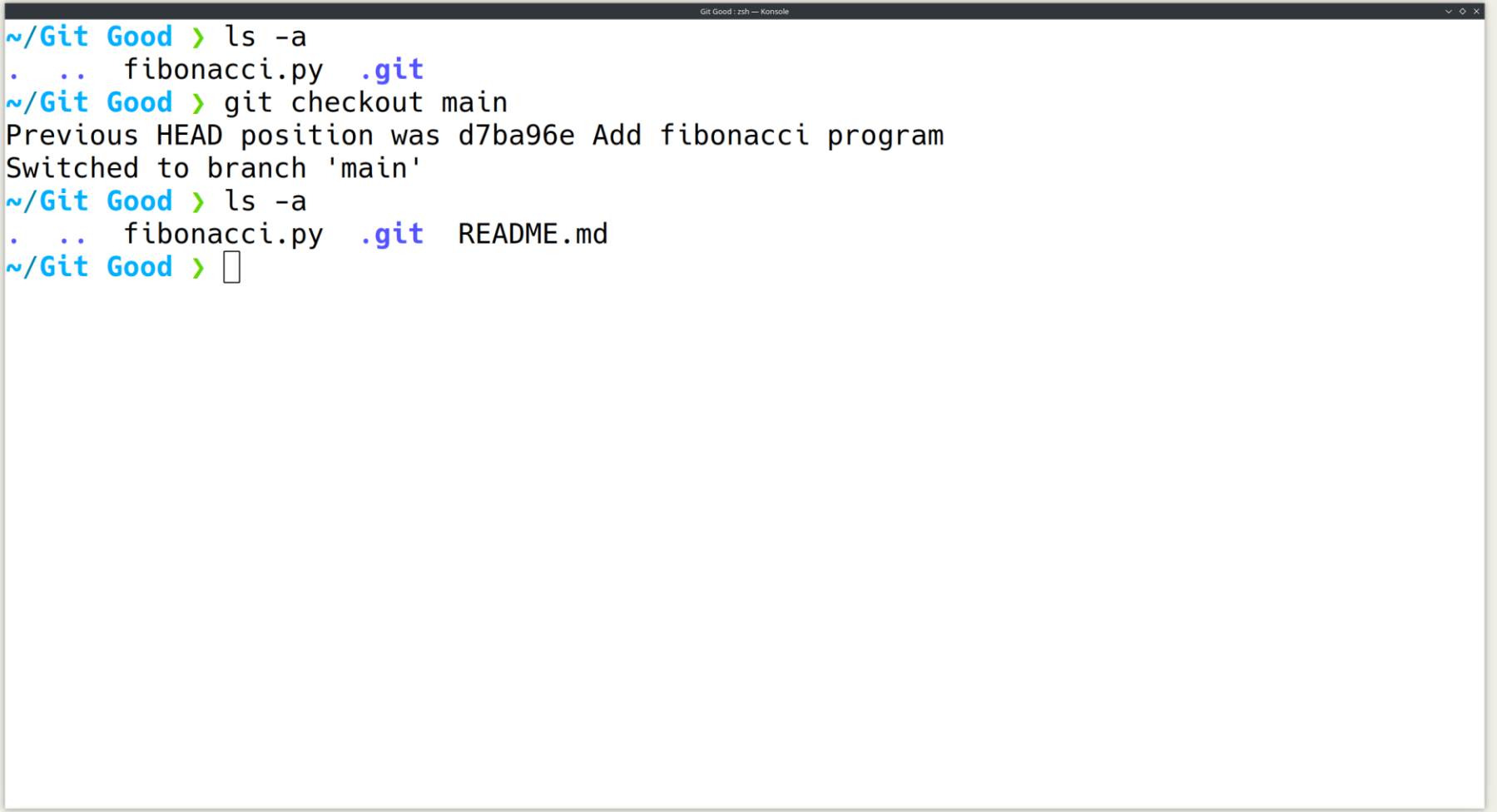

Git Good : zsh — Konsole
~/Git Good ❯ ls -a
. .. fibonacci.py .git README.md
~/Git Good ❯ git checkout HEAD
~/Git Good ❯ ls -a
. .. fibonacci.py .git README.md
~/Git Good ❯

```
~/Git Good ❯ ls -a
.  ..  fibonacci.py  .git  README.md
~/Git Good ❯ 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 create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at d7ba96e Add fibonacci program
~/Git Good ❯ ls -a
.  ..  fibonacci.py  .git
```

Git Good : git — Konsole
* b1eb1e6 (main) Update range and add README
* d7ba96e (HEAD) Add fibonacci program
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
(END)

Git Good : zsh — Konsole

```
~/Git Good ❯ git log --color --oneline --graph --decorate --all
~/Git Good ❯ ls -a
.  ..  fibonacci.py  .git
~/Git Good ❯ cat fibonacci.py
a = 0
b = 1

output = f"{a}, {b}"

for i in range(3, 11):
    a, b = b, a + b
    output += f", {b}"

print(output)

~/Git Good ❯
```


Git Good : zsh — Konsole
~/Git Good ❯ ls -a
. .. fibonacci.py .git
~/Git Good ❯ git checkout main
Previous HEAD position was d7ba96e Add fibonacci program
Switched to branch 'main'
~/Git Good ❯ ls -a
. .. fibonacci.py .git README.md
~/Git Good ❯

Branches: into the multiverse

- We can make "alternative universes"
- You can think of a branch as just a series of commits
- We've used branches already!
  - "main" (sometimes "master") is the default branch we started on
  - Generally kept both up-to-date and not broken...

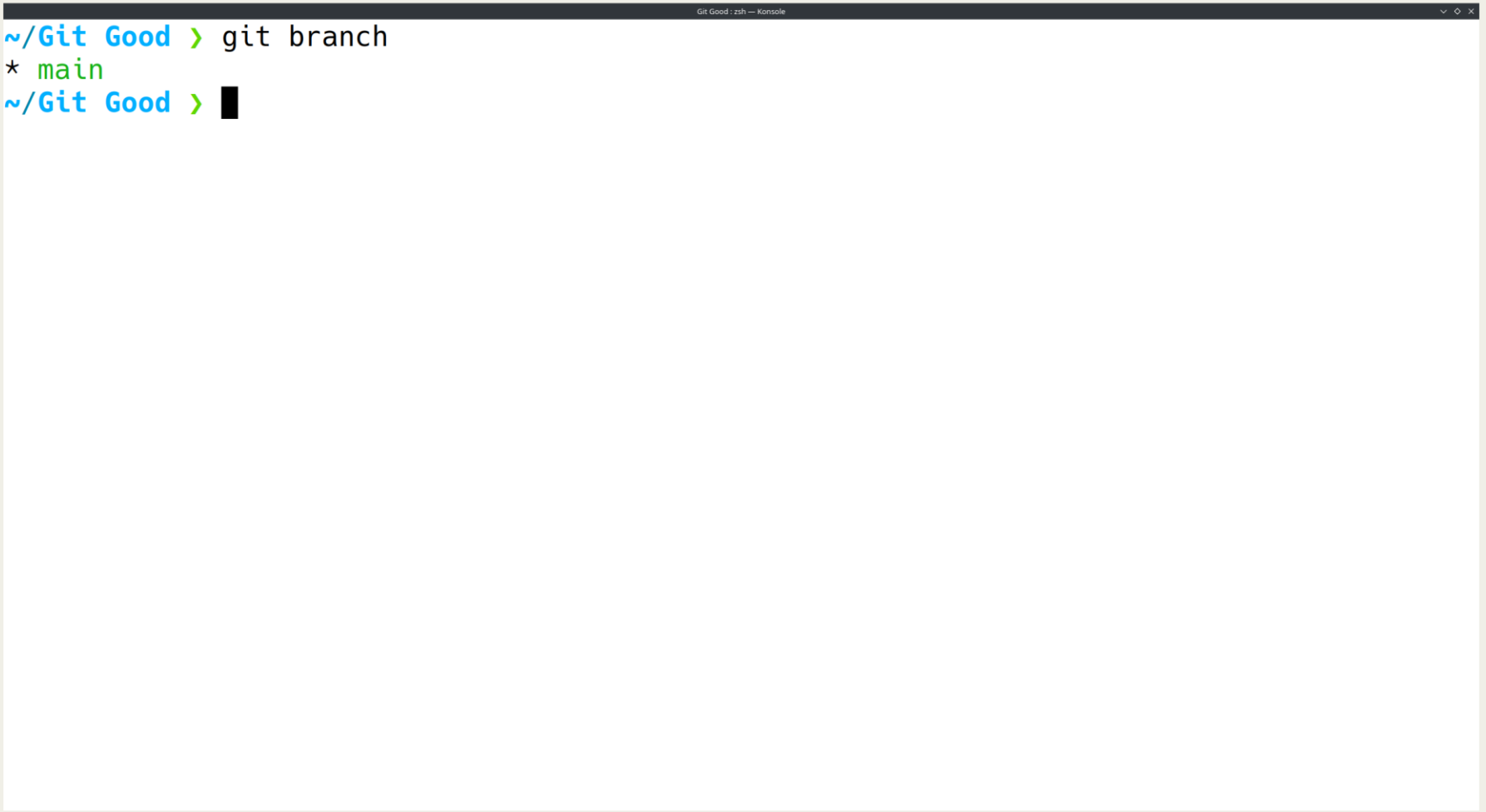
Why branches?

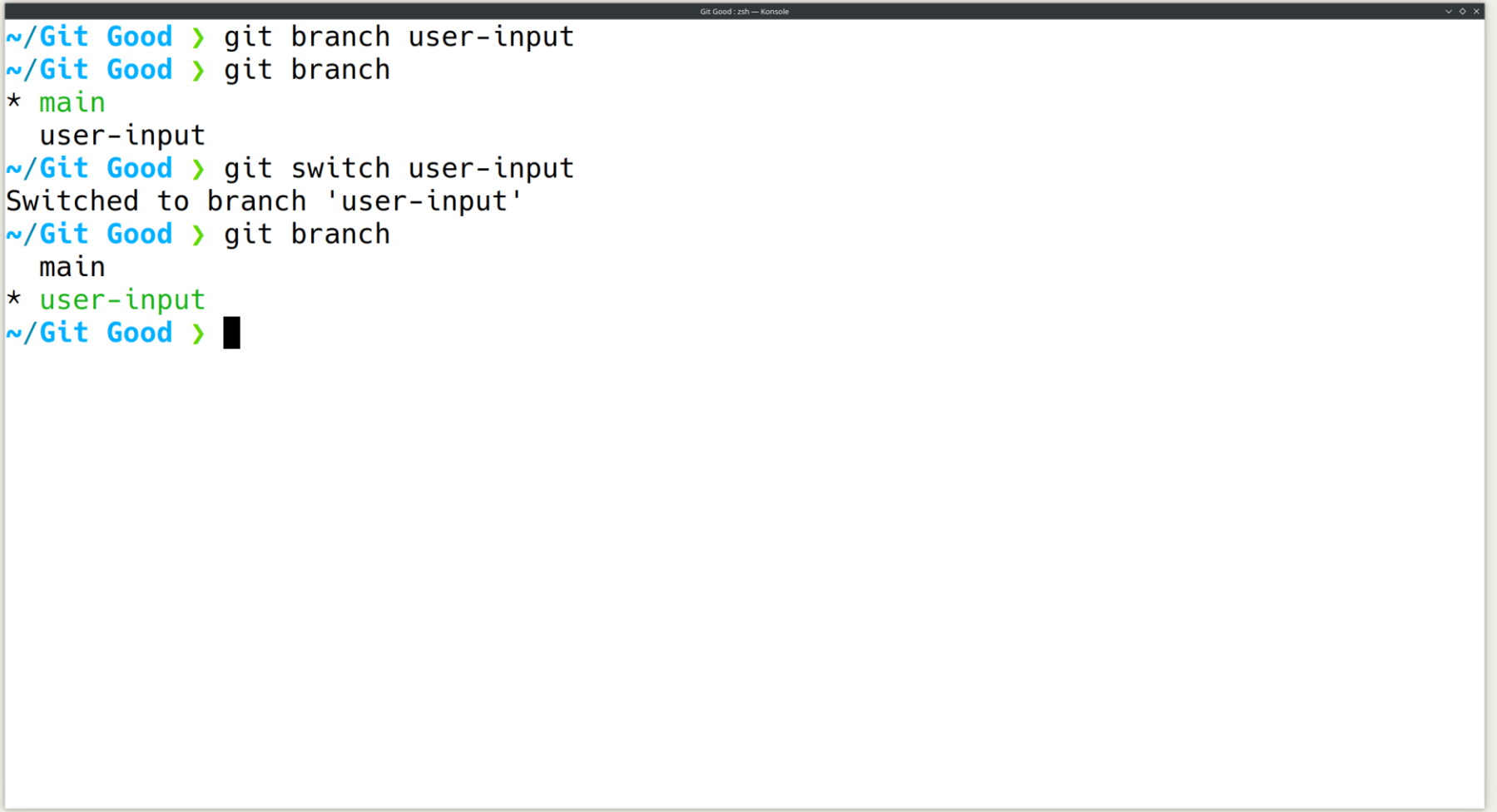
- Sometimes we want to experiment
  - If it doesn't work out, just discard the branch
- Helps isolate feature development
- Makes collaborative work easier (we'll discuss this more later)

How branches?

- The `git branch` command creates a new branch starting at the commit you're currently on

- To commit to the new branch, check it out!
- Modern version is `git switch <branch-name>`

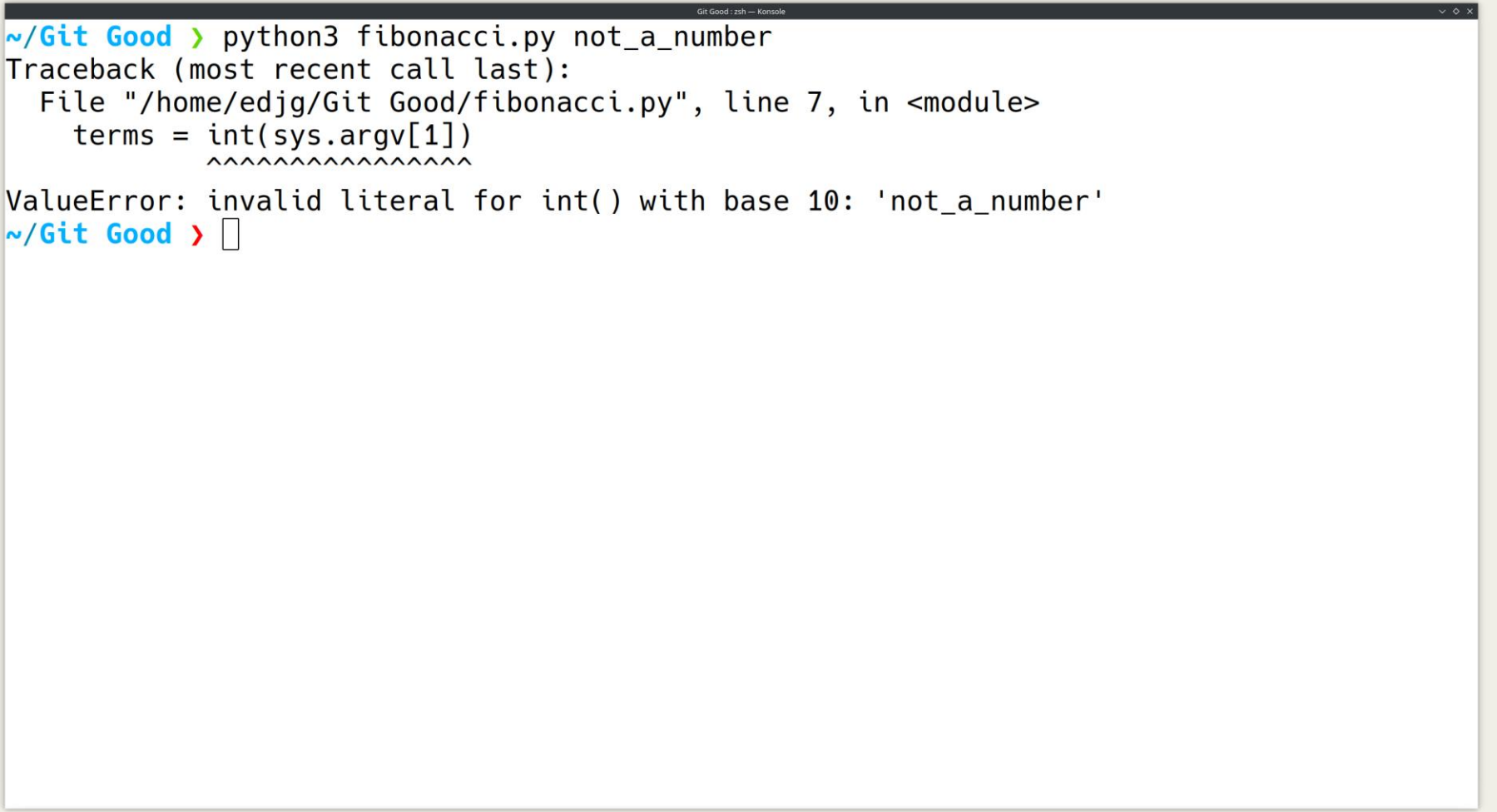

Git Good : zsh — Konsole
~/Git Good ❯ git branch
* main
~/Git Good ❯


Git Good : zsh — Konsole
~/Git Good ❯ git branch user-input
~/Git Good ❯ git branch
* main
  user-input
~/Git Good ❯ git switch user-input
Switched to branch 'user-input'
~/Git Good ❯ git branch
  main
* user-input
~/Git Good ❯

Git Good : vim — Konsole

```
import sys

a = 0
b = 1

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

for i in range(3, terms + 1):
    a, b = b, a + b
    output += f", {b}"

print(output)

~
~
~
~
~
~
~
~
:wq
```

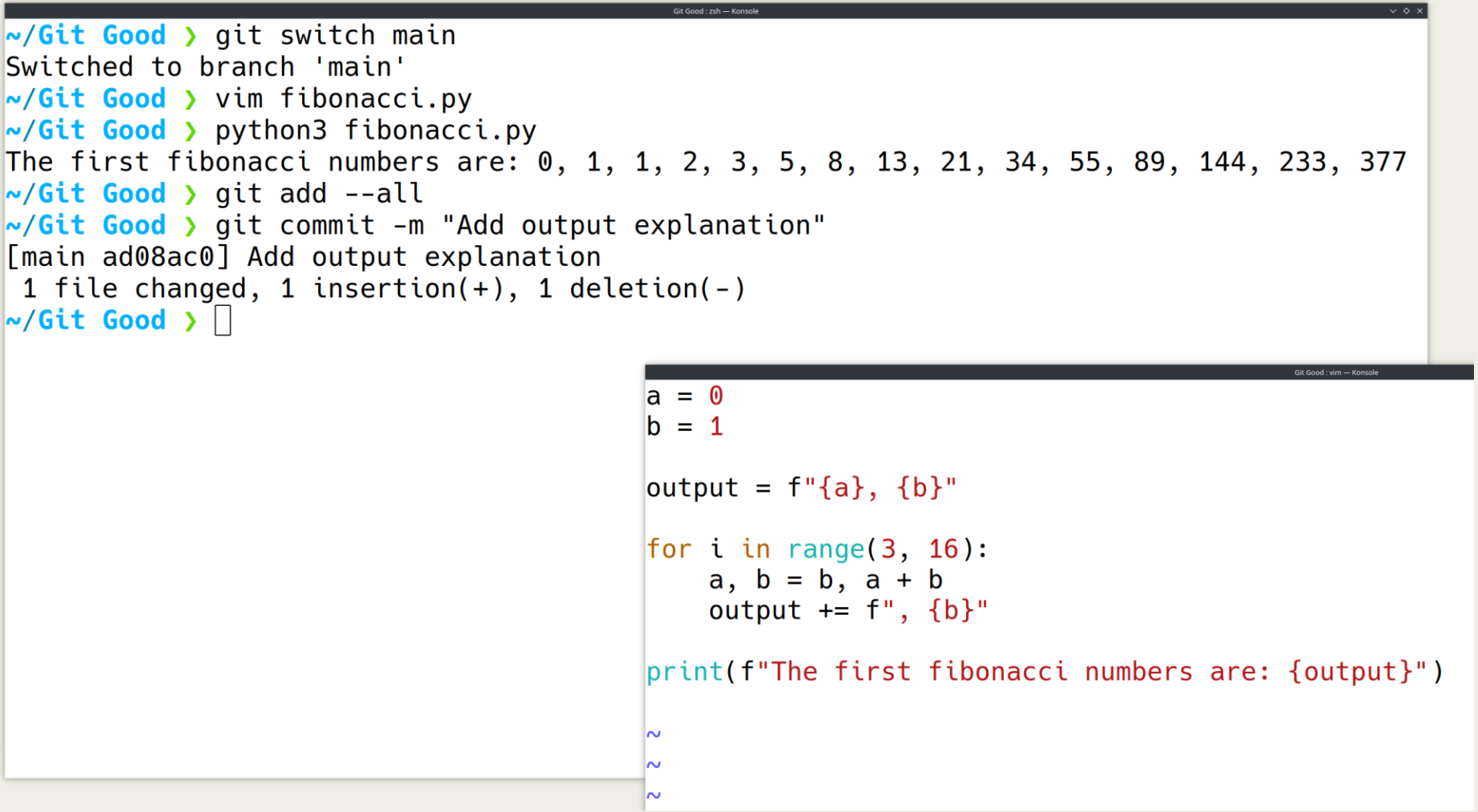
```
~/Git Good ❯ vim fibonacci.py
~/Git Good ❯ python3 fibonacci.py 5
0, 1, 1, 2, 3
~/Git Good ❯ python3 fibonacci.py 10
0, 1, 1, 2, 3, 5, 8, 13, 21, 34
~/Git Good ❯ git add --all
~/Git Good ❯ git commit -m "Add user input for number of terms"
[user-input 12a17d7] Add user input for number of terms
 1 file changed, 4 insertions(+), 1 deletion(-)
~/Git Good ❯
```

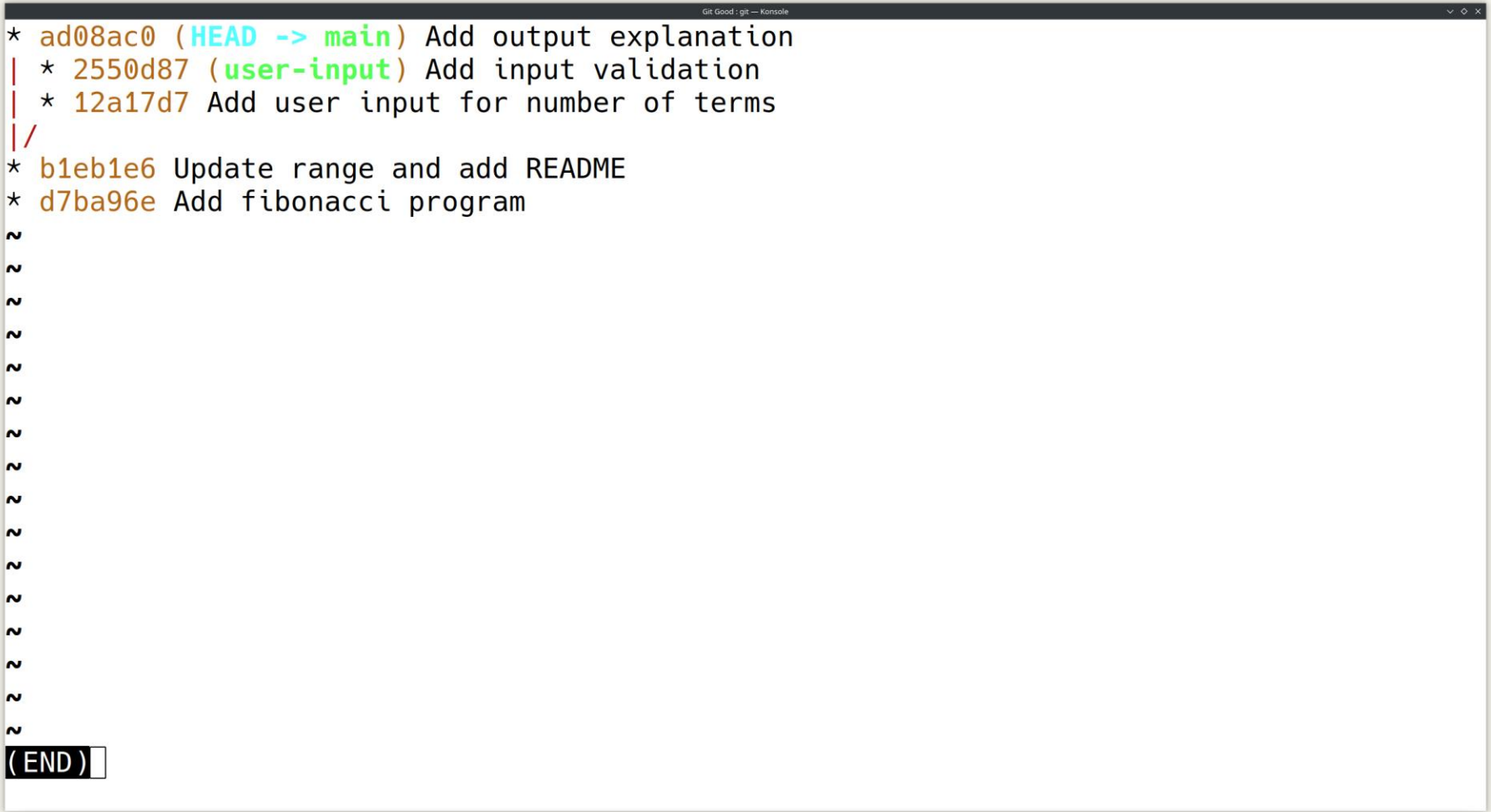

Git Good : zsh — Konsole
~/Git Good ❯ python3 fibonacci.py not_a_number
Traceback (most recent call last):
File "/home/edjg/Git Good/fibonacci.py", line 7, in <module>
terms = int(sys.argv[1])
^^^^^^^^^^^^^^^^
ValueError: invalid literal for int() with base 10: 'not_a_number'
~/Git Good ❯

Git Good : vim — Konsole

```
import sys

a = 0
b = 1

output = f"{a}, {b}"

try:
    terms = int(sys.argv[1])
except:
    print("That isn't a number! Defaulting to 10 terms")
    terms = 10

for i in range(3, terms + 1):
    a, b = b, a + b
    output += f", {b}"

print(output)

~
~
~
:wq
```

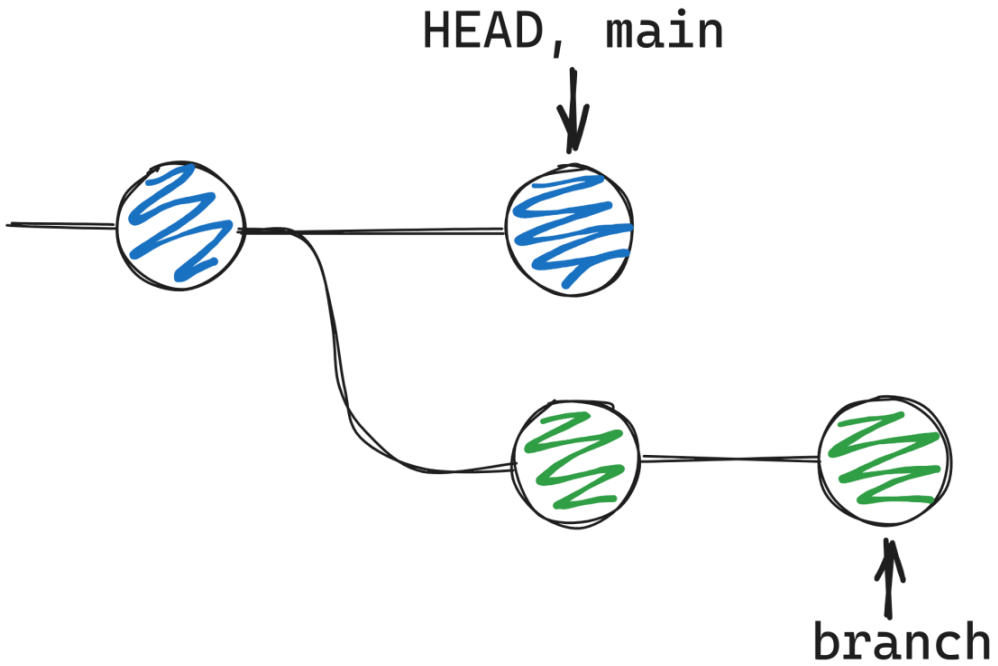
```
~/Git Good ❯ vim fibonacci.py
~/Git Good ❯ python3 fibonacci.py not_a_number
That isn't a number! Defaulting to 10 terms
0, 1, 1, 2, 3, 5, 8, 13, 21, 34
~/Git Good ❯ git add --all
~/Git Good ❯ git commit -m "Add input validation"
[user-input 2550d87] Add input validation
 1 file changed, 6 insertions(+), 1 deletion(-)
~/Git Good ❯
```


Git Good : zsh — Konsole
~/Git Good ❯ git switch main
Switched to branch 'main'
~/Git Good ❯ vim fibonacci.py
~/Git Good ❯ python3 fibonacci.py
The first fibonacci numbers are: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377
~/Git Good ❯ git add --all
~/Git Good ❯ git commit -m "Add output explanation"
[main ad08ac0] Add output explanation
1 file changed, 1 insertion(+), 1 deletion(-)
~/Git Good ❯
Git Good : vim — Konsole
a = 0
b = 1
output = f"{a}, {b}"
for i in range(3, 16):
a, b = b, a + b
output += f", {b}"
print(f"The first fibonacci numbers are: {output}")
~
~
~


Git Good : git — Konsole
* ad08ac0 (HEAD -> main) Add output explanation
| * 2550d87 (user-input) Add input validation
| * 12a17d7 Add user input for number of terms
|/
* b1eb1e6 Update range and add README
* d7ba96e Add fibonacci program
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
(END)

Git Good : git — Konsole

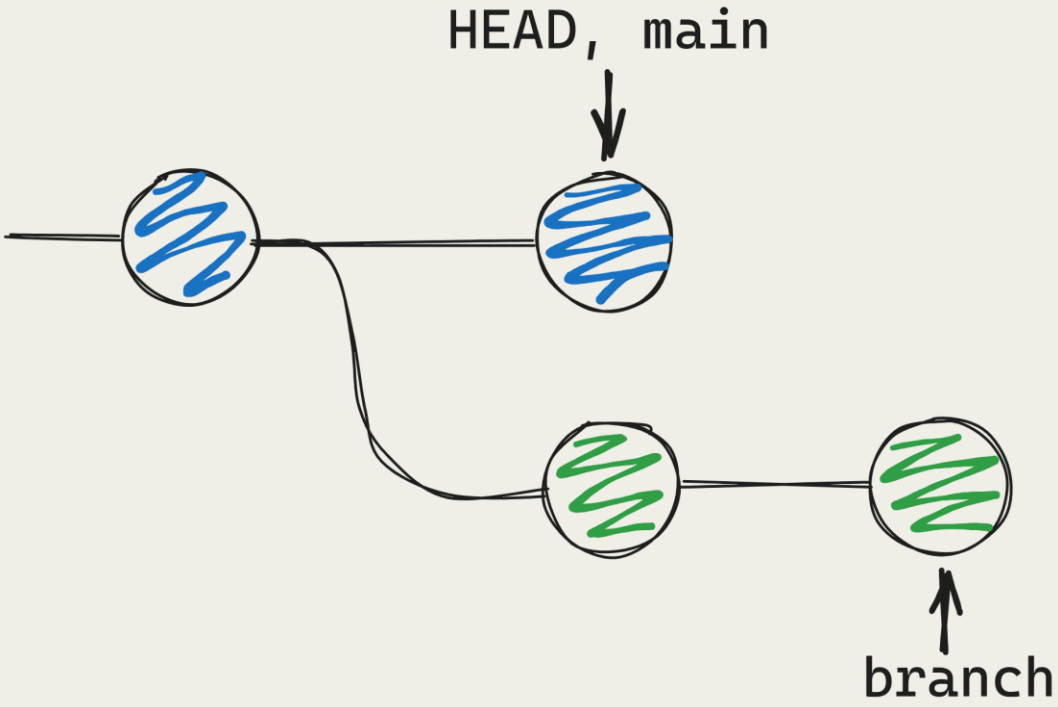
```
* ad08ac0 (HEAD -> main) Add output explanation
| * 2550d87 (user-input) Add input validation
| * 12a17d7 Add user input for number of terms
|/
* b1eb1e6 Update range and add README
* d7ba96e Add fibonacci program
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
(END)
```

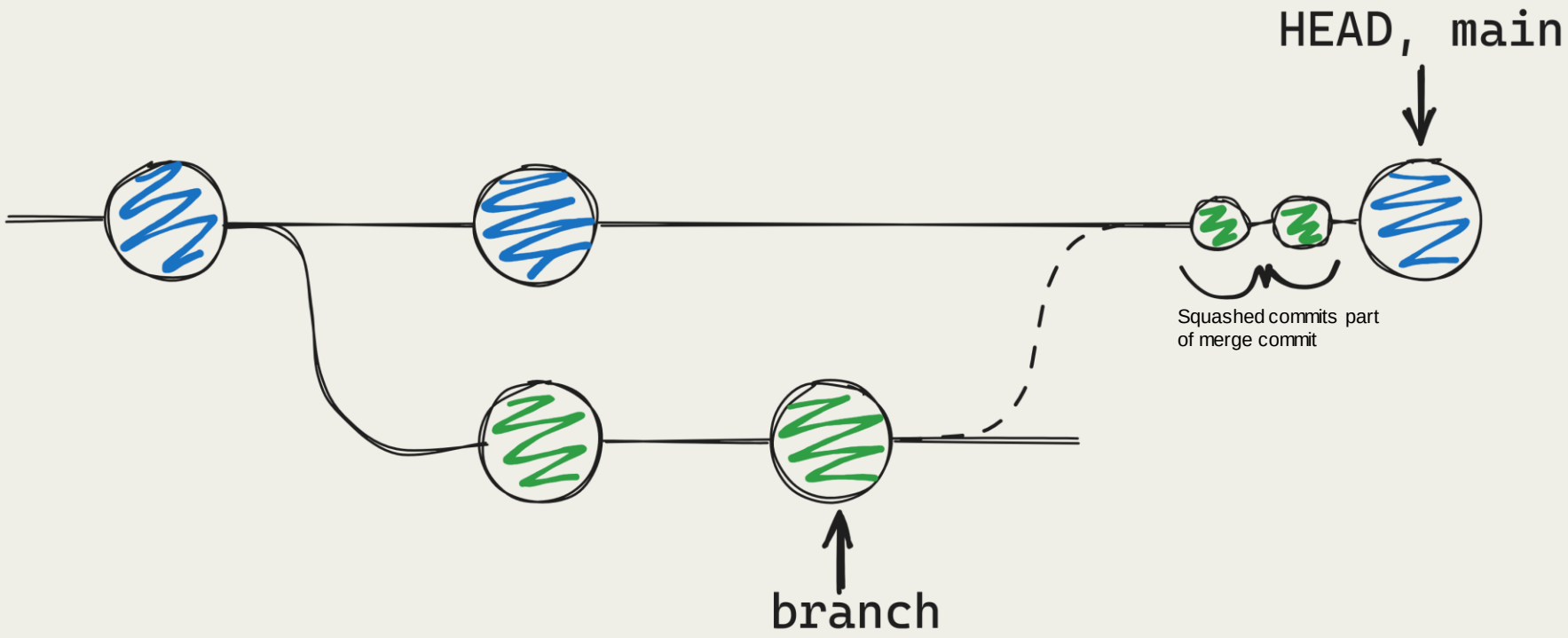


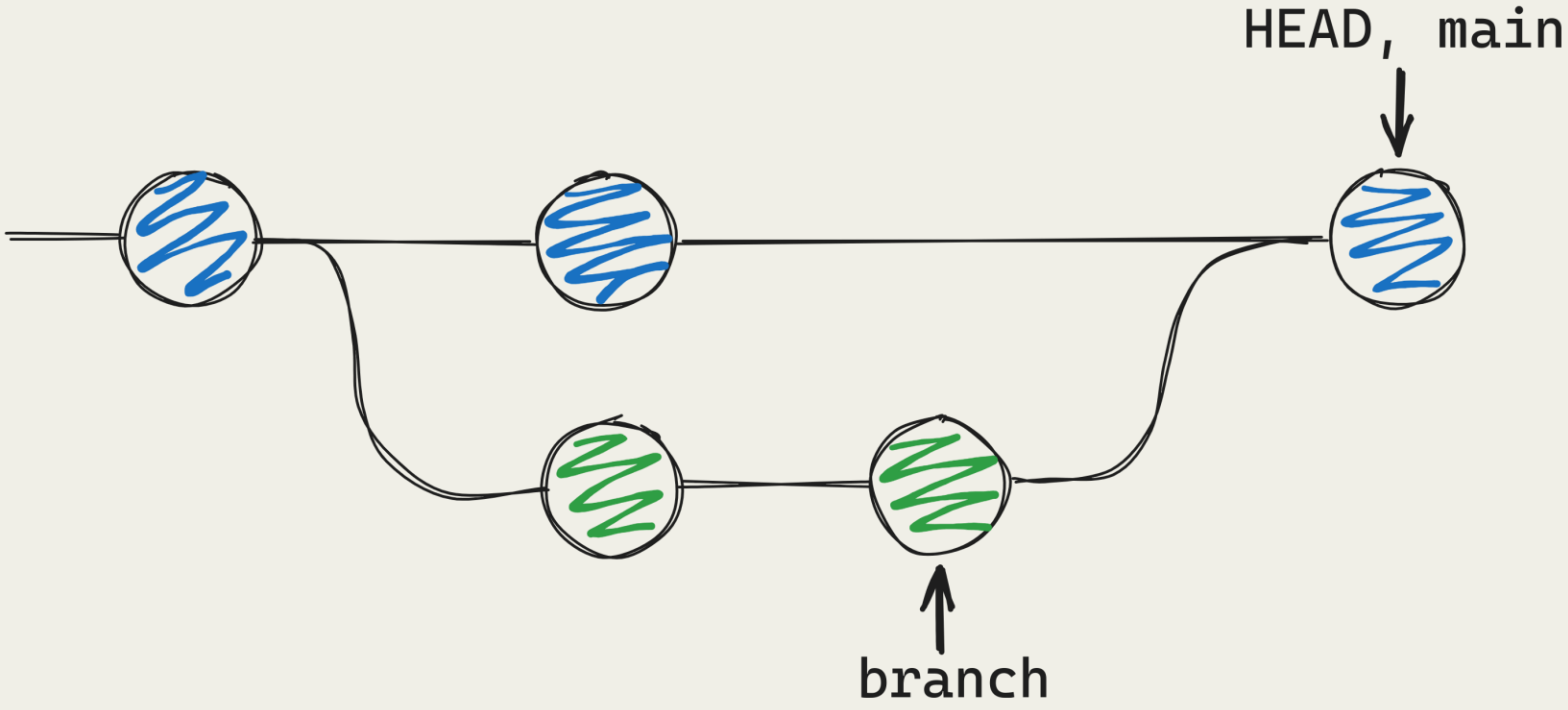
Questions?

What is merging?

- Sometimes we want to have changes from more than one branch
  - For example, if we developed a feature in a branch, and want to include it in our main branch
- We can "merge" branch B into branch A to give branch A the changes from branch B


HEAD, main
branch


HEAD, main
Squashed commits part
of merge commit
branch


HEAD, main
branch

How to merge

- Switch to the branch you want to merge ***into***
- Use the `git merge <other>` command to merge the other branch into it
  - Creates a new commit containing the changes from the other branch on the current branch
  - Does not modify the other branch


Git Good : zsh — Konsole
~/Git Good ❯ git switch main
Already on 'main'
~/Git Good ❯ git merge user-input

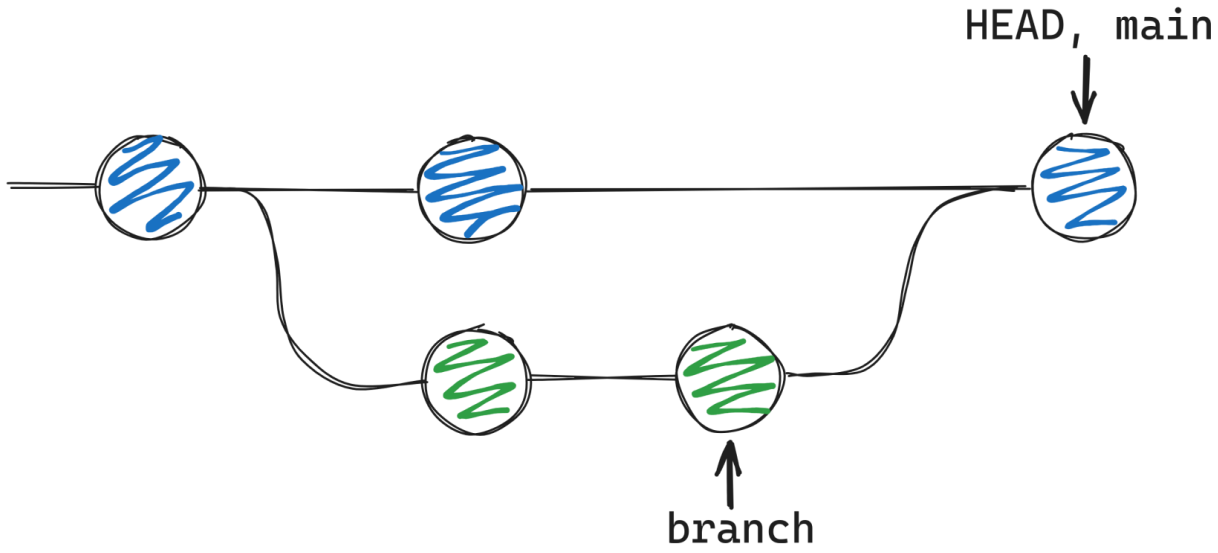
```
Merge branch 'user-input'
# Please enter a commit message to explain why this merge is necessary,
# especially if it merges an updated upstream into a topic branch.
#
# Lines starting with '#' will be ignored, and an empty message aborts
# the commit.
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
:wq
```


Git Good : zsh — Konsole
~/Git Good ❯ git switch main
Already on 'main'
~/Git Good ❯ git merge user-input
Auto-merging fibonacci.py
Merge made by the 'ort' strategy.
fibonacci.py | 10 +++++++++-
1 file changed, 9 insertions(+), 1 deletion(-)
~/Git Good ❯

```
*   240024d (HEAD -> main) Merge branch 'user-input'
|\
| * 2550d87 (user-input) Add input validation
| * 12a17d7 Add user input for number of terms
* | ad08ac0 Add output explanation
|/
* b1eb1e6 Update range and add README
* d7ba96e Add fibonacci program
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
(END)
```

Git Good : git — Konsole

```
*   240024d (HEAD -> main) Merge branch 'user-input'
|\
| * 2550d87 (user-input) Add input validation
| * 12a17d7 Add user input for number of terms
* | ad08ac0 Add output explanation
|/
* b1eb1e6 Update range and add README
* d7ba96e Add fibonacci program
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
(END)
```



Git Good : vim — Konsole

```
import sys

a = 0
b = 1

output = f"{a}, {b}"

try:
    terms = int(sys.argv[1])
except:
    print("That isn't a number! Defaulting to 10 terms")
    terms = 10

for i in range(3, terms + 1):
    a, b = b, a + b
    output += f", {b}"

print(f"The first fibonacci numbers are: {output}")

~
~
~
"fibonacci.py" 19L, 289B                                    1,1           All
```

Questions?

Something broke my merge!

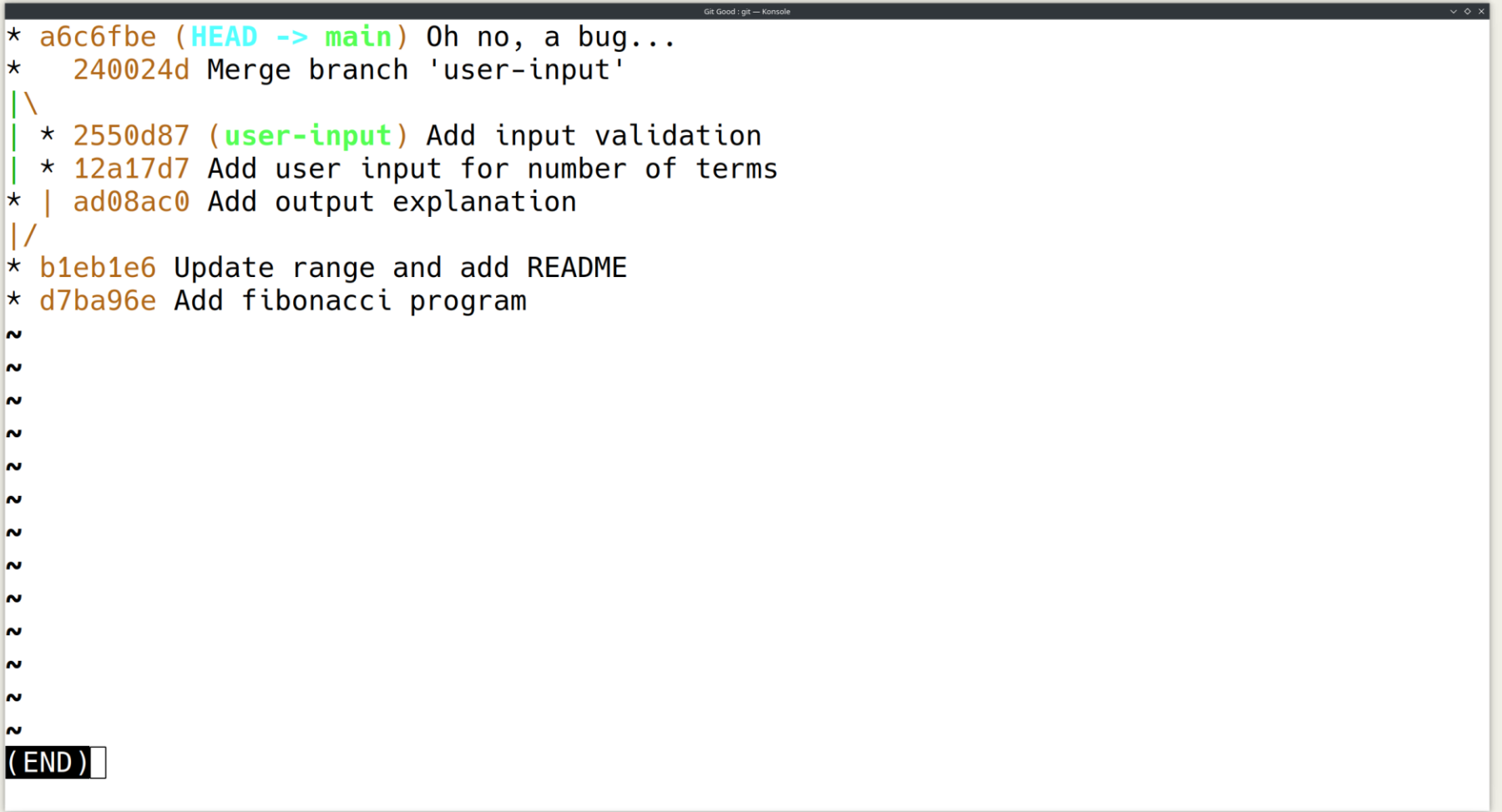
- Sometimes, git is unable to merge automatically
  - For example, a line is changed in both branches, and it doesn't know which one to pick
- This is called a merge conflict
- You have to resolve it manually
  - Outside the scope of this course, but lots of online tutorials
- **You should be careful when doing this**

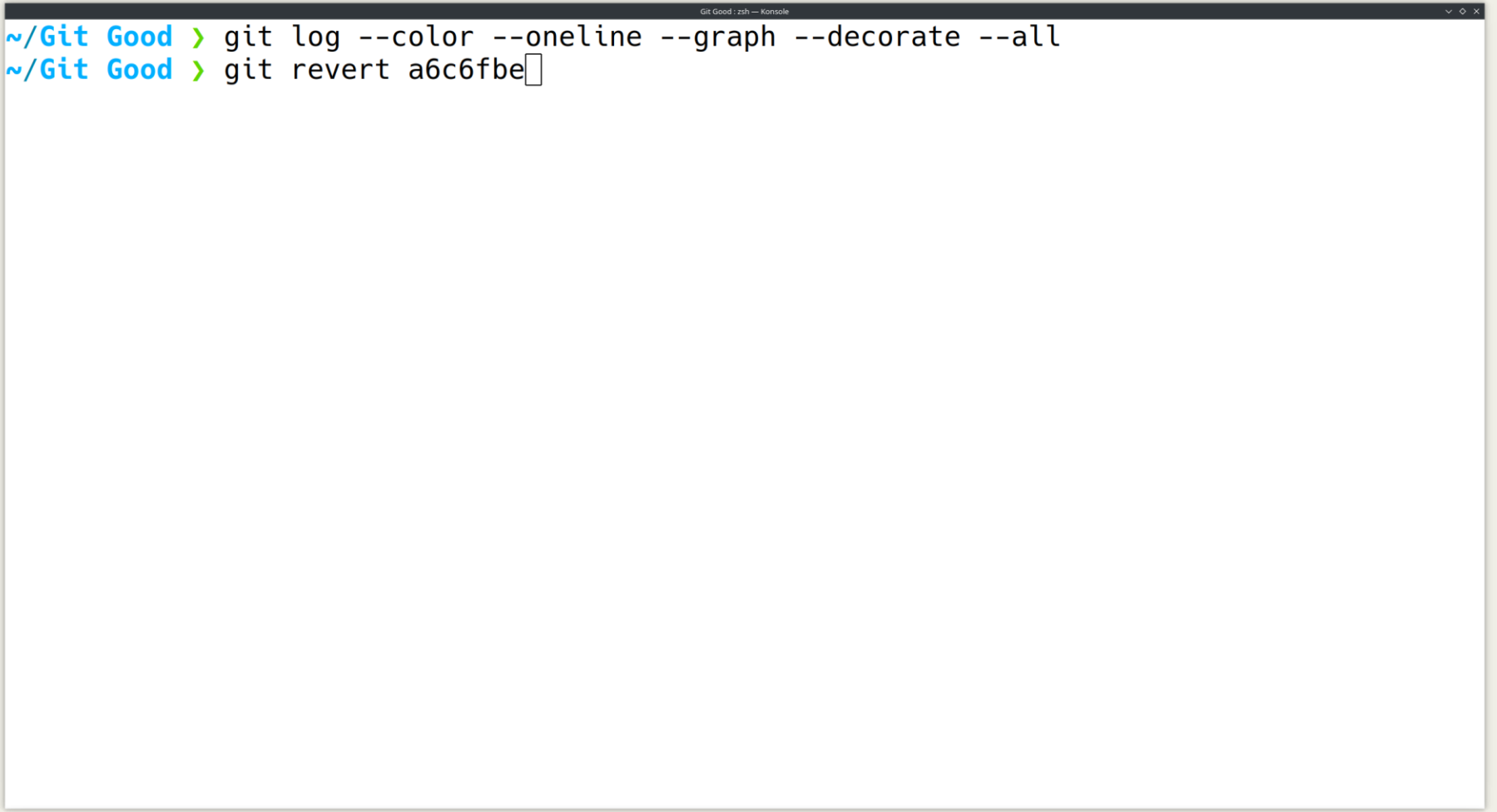
Changing history

- One of the benefits of version control is easily fixing mistakes!
- The `git revert` command undoes a single commit
  - Creates a new commit doing the undoing the old one
- The `git reset` command is more dangerous
  - Won't discuss now, look it up if you need it
  - Soft/mixed doesn't affect working directory, only HEAD
  - Hard **discards everything** back to a specified point

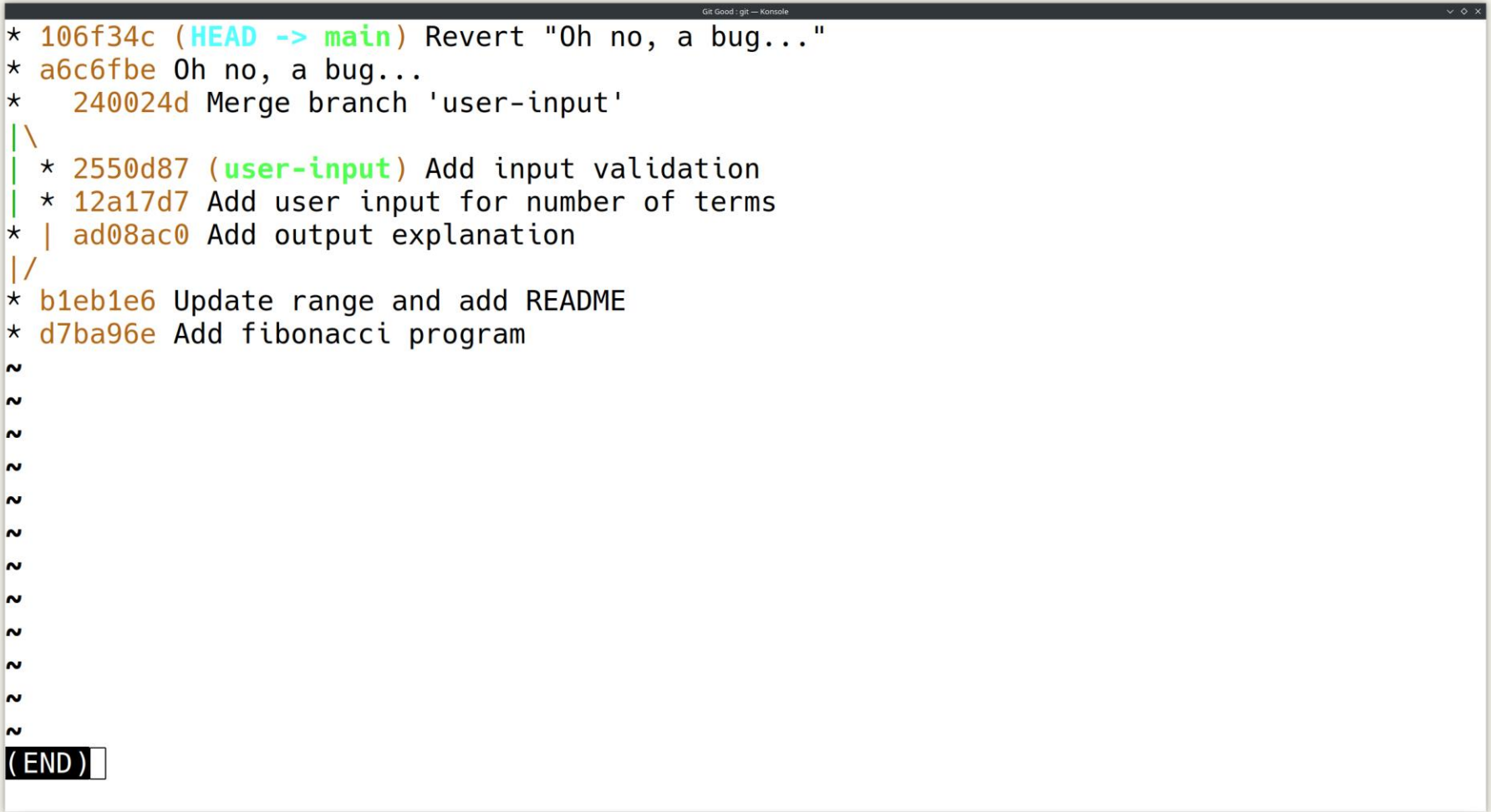
Git Good : vim — Konsole

```
import sys

a = 0
b = 1

print("Garden tiger moth")

output = f"{a}, {b}"

try:
    terms = int(sys.argv[1])
except:
    print("That isn't a number! Defaulting to 10 terms")
    terms = 10

for i in range(3, terms + 1):
    a, b = b, a + b
    output += f", {b}"

print(f"The first fibonacci numbers are: {output}")

~
-- VISUAL LINE --                                    6,26          All
```


Git Good : git — Konsole
* a6c6fbe (HEAD -> main) Oh no, a bug...
*   240024d Merge branch 'user-input'
|\
| * 2550d87 (user-input) Add input validation
| * 12a17d7 Add user input for number of terms
* | ad08ac0 Add output explanation
|/
* b1eb1e6 Update range and add README
* d7ba96e Add fibonacci program
~
~
~
~
~
~
~
~
~
~
~
~
~
~
(END)


Git Good : zsh — Konsole
~/Git Good ❯ git log --color --oneline --graph --decorate --all
~/Git Good ❯ git revert a6c6fbe

Git Good : git — Konsole
Revert "Oh no, a bug..."
This reverts commit a6c6fbe7a5409b4a672de82fd16c6bce33c247d1.
Please enter the commit message for your changes. Lines starting
with '#' will be ignored, and an empty message aborts the commit.
#
On branch main
Changes to be committed:
#	modified: fibonacci.py
#
~
~
~
~
~
~
~
~
~
~
~
"~/Git Good/.git/COMMIT_EDITMSG" 11L, 299B
1,1
All


Git Good : git — Konsole
* 106f34c (HEAD -> main) Revert "Oh no, a bug..."
* a6c6fbe Oh no, a bug...
*   240024d Merge branch 'user-input'
|\
| * 2550d87 (user-input) Add input validation
| * 12a17d7 Add user input for number of terms
* | ad08ac0 Add output explanation
|/
* b1eb1e6 Update range and add README
* d7ba96e Add fibonacci program
~
~
~
~
~
~
~
~
~
~
~
~
~
(END)

Remote work

- Remote repos are versions of a repo that live online
- The `git remote` command lets you manage them
- This allows us to collaborate!
- GitHub, GitLab, and others offer remote repo hosting
  - You can also do it yourself for a challenge!

```
Git Good : zsh — Konsole
~/Git Good ❯ git remote add origin https://github.com/EdmundGoodman/git-good-demo.git
~/Git Good ❯ git push origin main
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), 2.35 KiB | 401.00 KiB/s, done.
Total 23 (delta 7), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (7/7), done.
To https://github.com/EdmundGoodman/git-good-demo.git
 * [new branch]      main -> main
~/Git Good ❯
```

git-good-demo

Public

Pin | Unwatch 1 | Fork 0 | Star 0

main | 1 branch | 0 tags | Go to file | Add file | Code

EdmundGoodman Revert "Oh no, a bug..." ... `106f34c` 8 minutes ago **8** commits

README.md	Update range and add README	36 minutes ago
fibonacci.py	Revert "Oh no, a bug..."	8 minutes ago

README.md

Git Good Demo

This is a program which prints Fibonacci numbers.

I wrote it as an example for the UWCS "Git Good" talk.

About

The demo for the "Git Good" UWCS talk

Readme
Activity
0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

Python 100.0%

Remotes and cloning

- You can get a local copy of a remote repo by "cloning" it
- The `git clone` subcommand does this
- Sometimes software is distributed by cloning the repo, then building/running it yourself


Git Good : zsh — Konsole
~/Git Good ❯ ls -a
. ..
~/Git Good ❯ git clone https://github.com/EdmundGoodman/git-good-demo
Cloning into 'git-good-demo'...
remote: Enumerating objects: 23, done.
remote: Counting objects: 100% (23/23), done.
remote: Compressing objects: 100% (15/15), done.
remote: Total 23 (delta 7), reused 23 (delta 7), pack-reused 0
Receiving objects: 100% (23/23), done.
Resolving deltas: 100% (7/7), done.
~/Git Good ❯ ls -a
. .. git-good-demo
~/Git Good ❯ ls -a git-good-demo
. .. fibonacci.py .git README.md
~/Git Good ❯

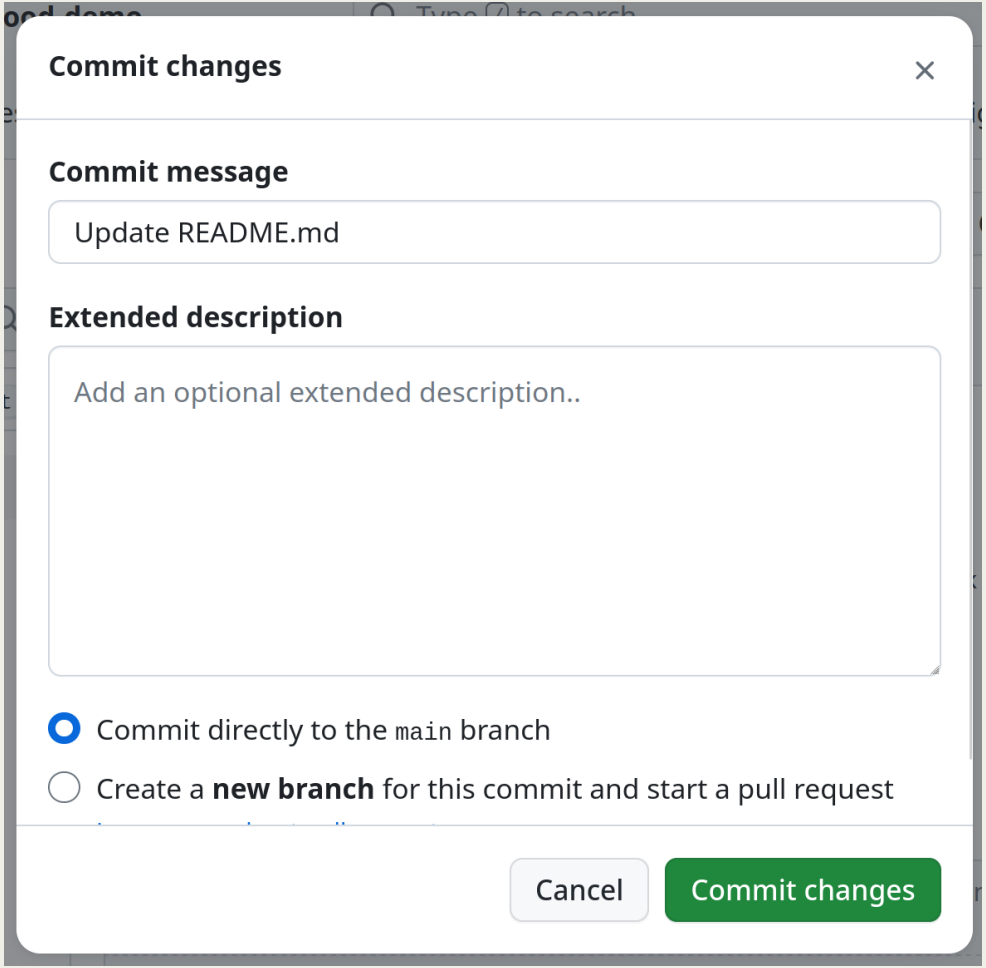
Remotes and branches

- Local branches can correspond to remote branches
  - The remote copy is called `<remote>/<branch>`, for example `origin/main`
  - You can have local branches which aren't on the remote (and vice versa)

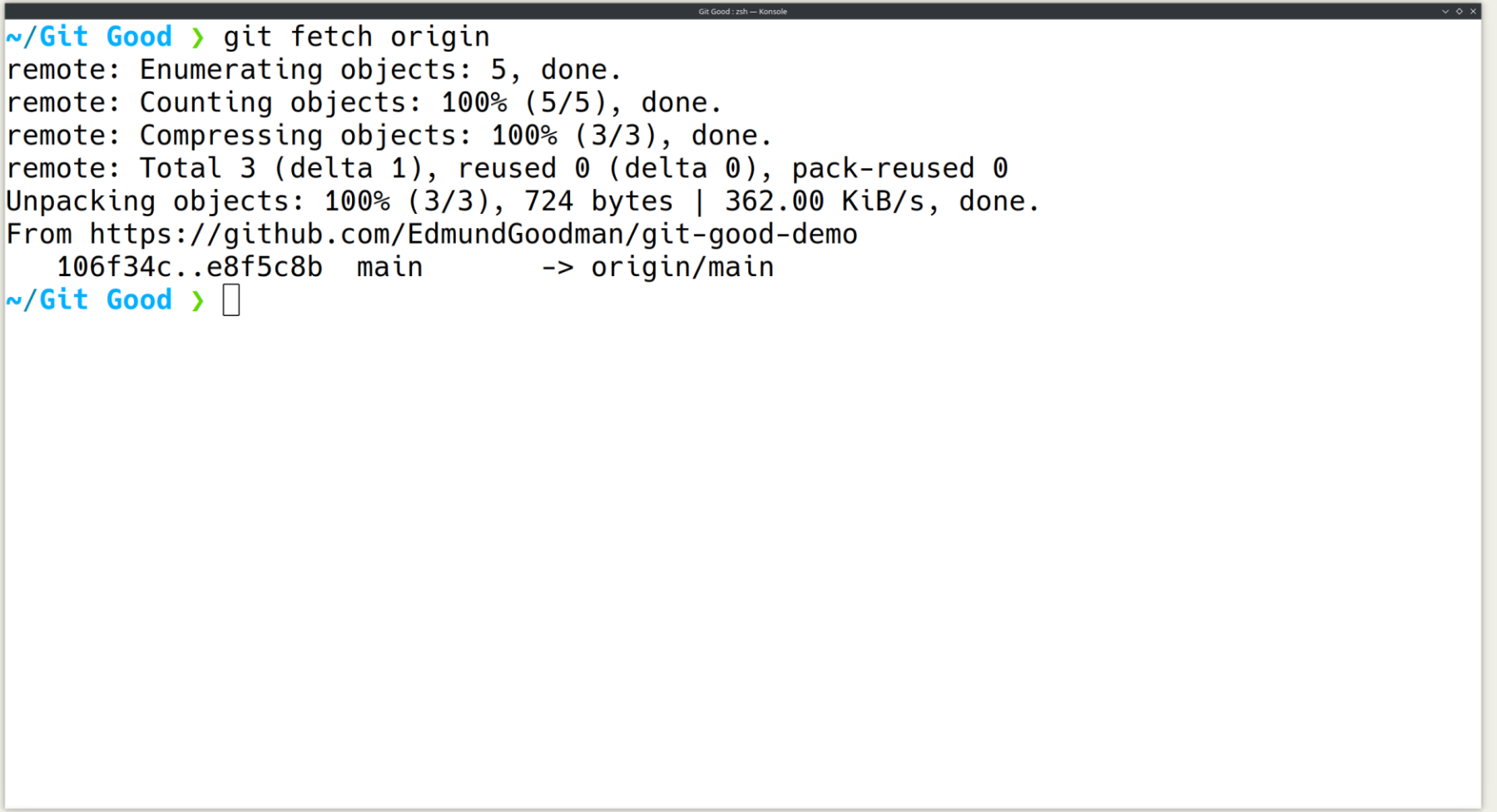
Fetch, Push, and Pull

- `git fetch` updates what the local repo knows about the remote repo
- `git push` updates the remote branch from the local branch
- `git pull` updates the local branch from the remote branch
  - This is like a `git fetch` followed by a `git merge <remote>/<branch>`


git-good-demo / README.md in main
Cancel changes
Commit changes...
Edit
Preview
Code 55% faster with GitHub Copilot
Spaces
2
Soft wrap
1 # Git Good Demo
2
3 This is a program which prints Fibonacci numbers.
4
5 I wrote it as an example for the UWCS "Git Good" talk.
6
7 I edited this in the GitHub web UI!
8

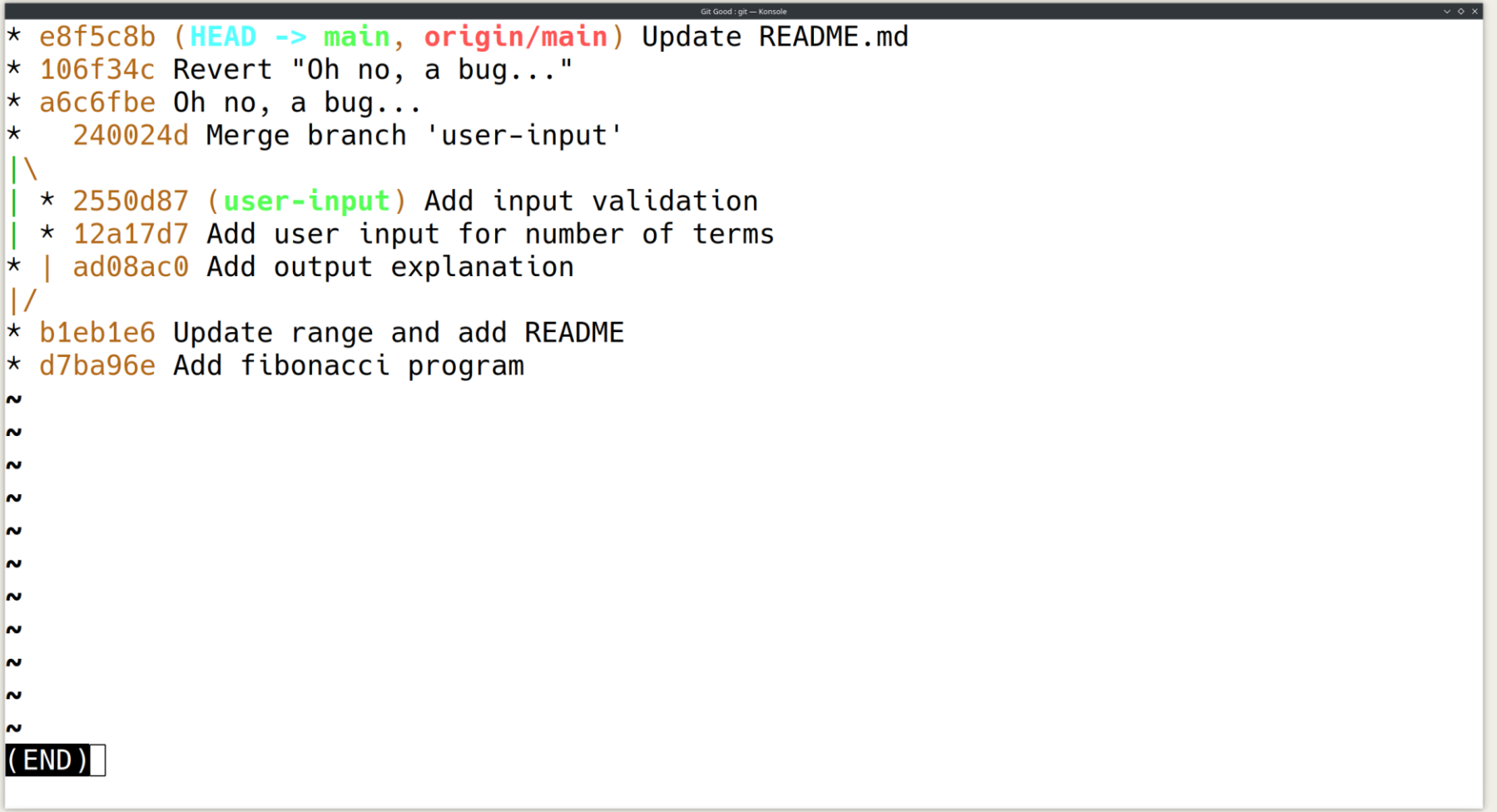

Commit changes
Commit message
Update README.md
Extended description
Add an optional extended description..
Commit directly to the main branch
Create a new branch for this commit and start a pull request
Cancel
Commit changes

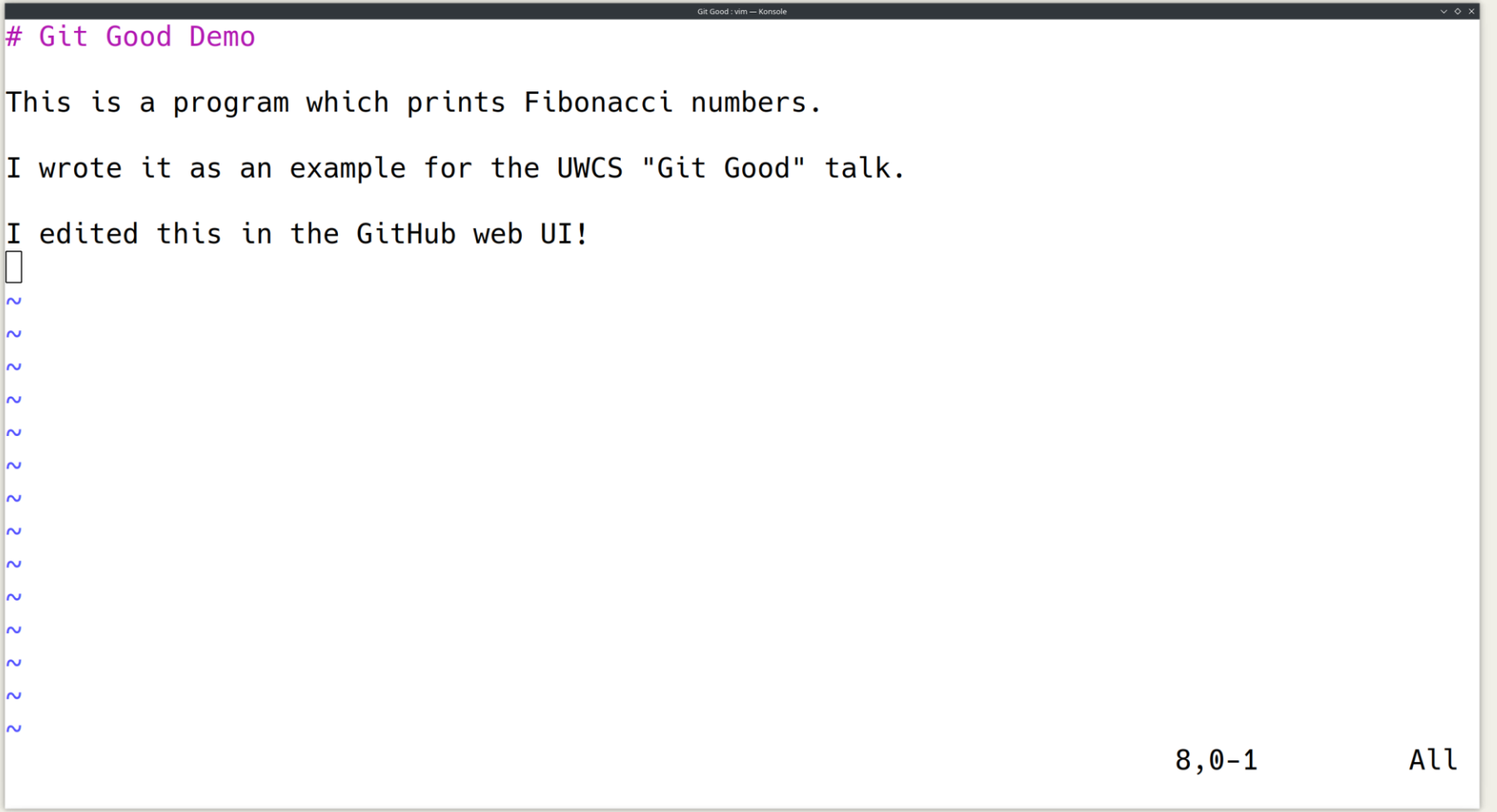
Git Good : git — Konsole
* 106f34c (HEAD -> main, origin/main) Revert "Oh no, a bug..."
* a6c6fbe Oh no, a bug...
*   240024d Merge branch 'user-input'
|\
| * 2550d87 (user-input) Add input validation
| * 12a17d7 Add user input for number of terms
* | ad08ac0 Add output explanation
|/
* b1eb1e6 Update range and add README
* d7ba96e Add fibonacci program
~
~
~
~
~
~
~
~
~
~
~
~
~
(END)

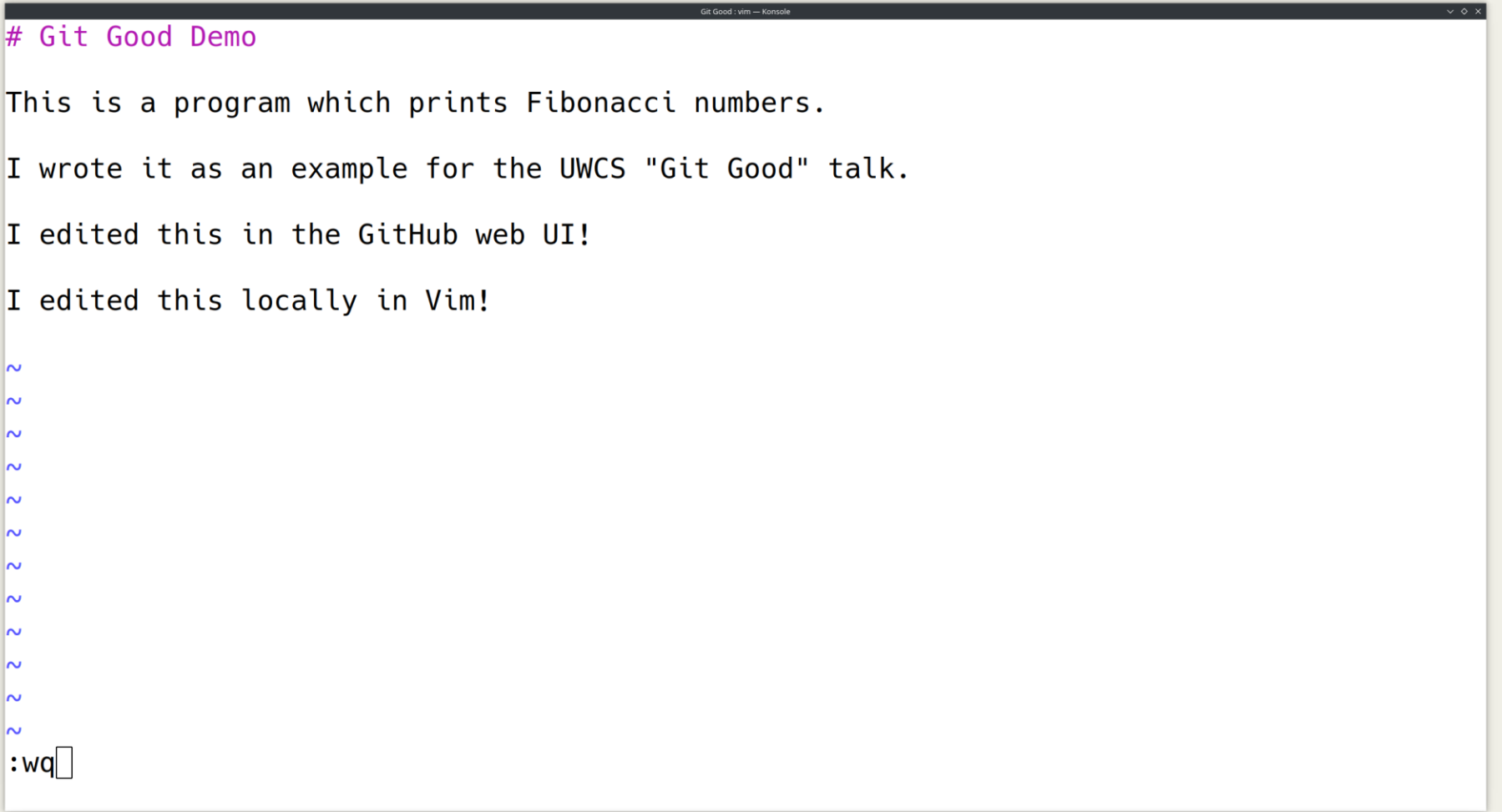

Git Good : zsh — Konsole
~/Git Good ❯ git fetch origin
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 724 bytes | 362.00 KiB/s, done.
From https://github.com/EdmundGoodman/git-good-demo
106f34c..e8f5c8b main -> origin/main
~/Git Good ❯

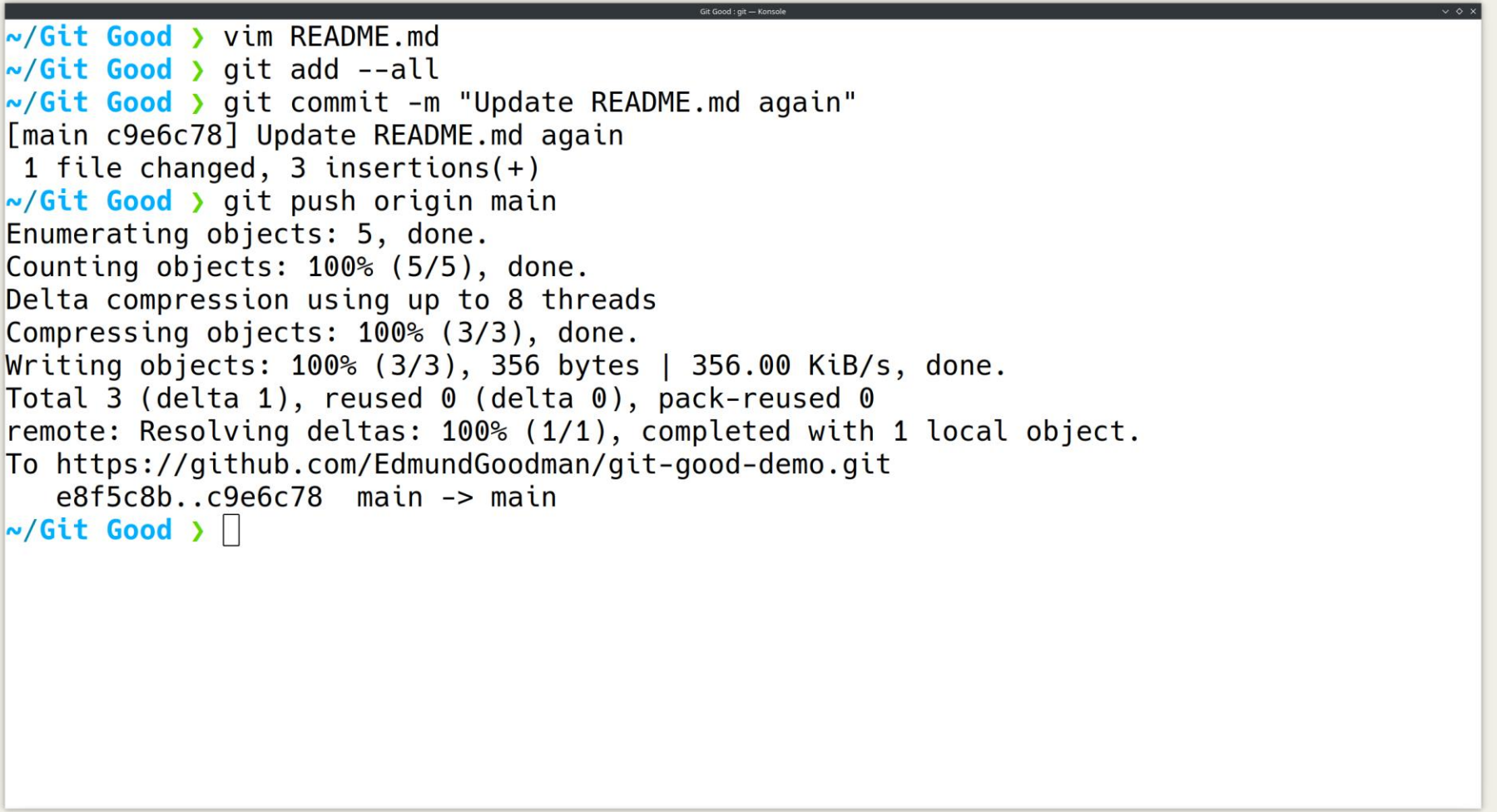
```
* e8f5c8b (origin/main) Update README.md
* 106f34c (HEAD -> main) Revert "Oh no, a bug..."
* a6c6fbe Oh no, a bug...
*   240024d Merge branch 'user-input'
|\
| * 2550d87 (user-input) Add input validation
| * 12a17d7 Add user input for number of terms
* | ad08ac0 Add output explanation
|/
* b1eb1e6 Update range and add README
* d7ba96e Add fibonacci program
~
~
~
~
~
~
~
~
~
~
~
(END)
```


Git Good : zsh — Konsole
~/Git Good ❯ git pull origin main
From https://github.com/EdmundGoodman/git-good-demo
 * branch   main   -> FETCH_HEAD
Updating 106f34c..e8f5c8b
Fast-forward
 README.md | 1 +
 1 file changed, 1 insertion(+)
~/Git Good ❯


Git Good : git — Konsole
* e8f5c8b (HEAD -> main, origin/main) Update README.md
* 106f34c Revert "Oh no, a bug..."
* a6c6fbe Oh no, a bug...
*   240024d Merge branch 'user-input'
|\
| * 2550d87 (user-input) Add input validation
| * 12a17d7 Add user input for number of terms
* | ad08ac0 Add output explanation
|/
* b1eb1e6 Update range and add README
* d7ba96e Add fibonacci program
~
~
~
~
~
~
~
~
~
~
~
~
(END)


Git Good : vim — Konsole
Git Good Demo
This is a program which prints Fibonacci numbers.
I wrote it as an example for the UWCS "Git Good" talk.
I edited this in the GitHub web UI!
8,0-1 All


Git Good : vim — Konsole
Git Good Demo
This is a program which prints Fibonacci numbers.
I wrote it as an example for the UWCS "Git Good" talk.
I edited this in the GitHub web UI!
I edited this locally in Vim!
:wq


Git Good : git — Konsole
~/Git Good ❯ vim README.md
~/Git Good ❯ git add --all
~/Git Good ❯ git commit -m "Update README.md again"
[main c9e6c78] Update README.md again
1 file changed, 3 insertions(+)
~/Git Good ❯ 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), 356 bytes | 356.00 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/EdmundGoodman/git-good-demo.git
e8f5c8b..c9e6c78 main -> main
~/Git Good ❯

git-good-demo / README.md

EdmundGoodman Update README.md again c9e6c78 · 1 minute ago History

Preview Code Blame 10 lines (5 loc) · 192 Bytes Code 55% faster with GitHub Copilot Raw

Git Good Demo

This is a program which prints Fibonacci numbers.

I wrote it as an example for the UWCS "Git Good" talk.

I edited this in the GitHub web UI!

I edited this locally in Vim!

Recap so far

- Looking at histories with `git log`
- Travelling in time with `git checkout`
- Working with branches with `git branch/merge`
- Undoing mistakes with `git revert`
- Working remotely with `git remote/...`

Top tips

- DO NOT COMMIT SECRETS
- Commit little and often
- Give commits meaningful names
- Make small branches and merge regularly
- Clean up dead branches
  - Can be done with `git branch -d`

✨ *Do not commit secrets!!!* ✨

Installation

- 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, it is almost always pre-installed (otherwise use a package manager of your choice)

Hate the command line?

- Lots of software exists to help manage git repos graphically
  - Git GUI for windows
  - SourceTree for Mac
- Almost all modern IDEs also have git plugins
  - This includes VSCode!

Ignoring files

- Sometimes you don’t want to keep track of certain files
  - Generated files (`.jar`, `__pycache__`), databases, etc.
  - Put secrets in an ignored .env file
- Create a file called `.gitignore` in the repository
  - This can contain a list of globs of filenames to ignore

Configuration

- Git is very configurable!
- Many things can be changed, including
  - Default editor, commit template, global gitignore, merge tool, aliases, handling of whitespace, default login credentials…

- Use the `git config` command to do this
  - Can do this on a project, user, or system level

EVERYTHING IS ON FIRE HELP

- Especially when inexperienced, it can be easy to mess up
- Someone has messed up exactly how you have before
- https://ohshitgit.com to fix many common mistakes

This was just an introduction

- We have barely scratched the surface of what git can do
  - Hopefully enough to get started/convince you git is useful

https://twitter.com/zadow28/status/787555337343799297

I want 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 and Atlassian both have helpful pages on many topics

When will I ever use this???

- Good for programming course-work (e.g. 118)
- Eases collaboration in group projects
- When you get a job, it will probably use git in some way

I want to practice using git!

- Luckily for you, we are running a workshop!
  - Put everything we've covered today into practice
  - Get help if you get stuck

- During Comp Café next week
  - 5-6pm, Tuesday 3rd October
  - CS0.06 (the lab on the right as you come into DCS)
  - Free food!
  - Be there or be  !

University of Warwick
Computing Society

I am an (un-)paid shill...

- Hopefully you found this talk interesting!
  - If you did, we do loads more academic events throughout the year
  - If you are currently bored to death/asleep, we do other things too!

University of Warwick
Computing Society

Questions?