**Introduction to branches**

**INTERMEDIATE GIT**

# George Boorman

Curriculum Manager, DataCamp

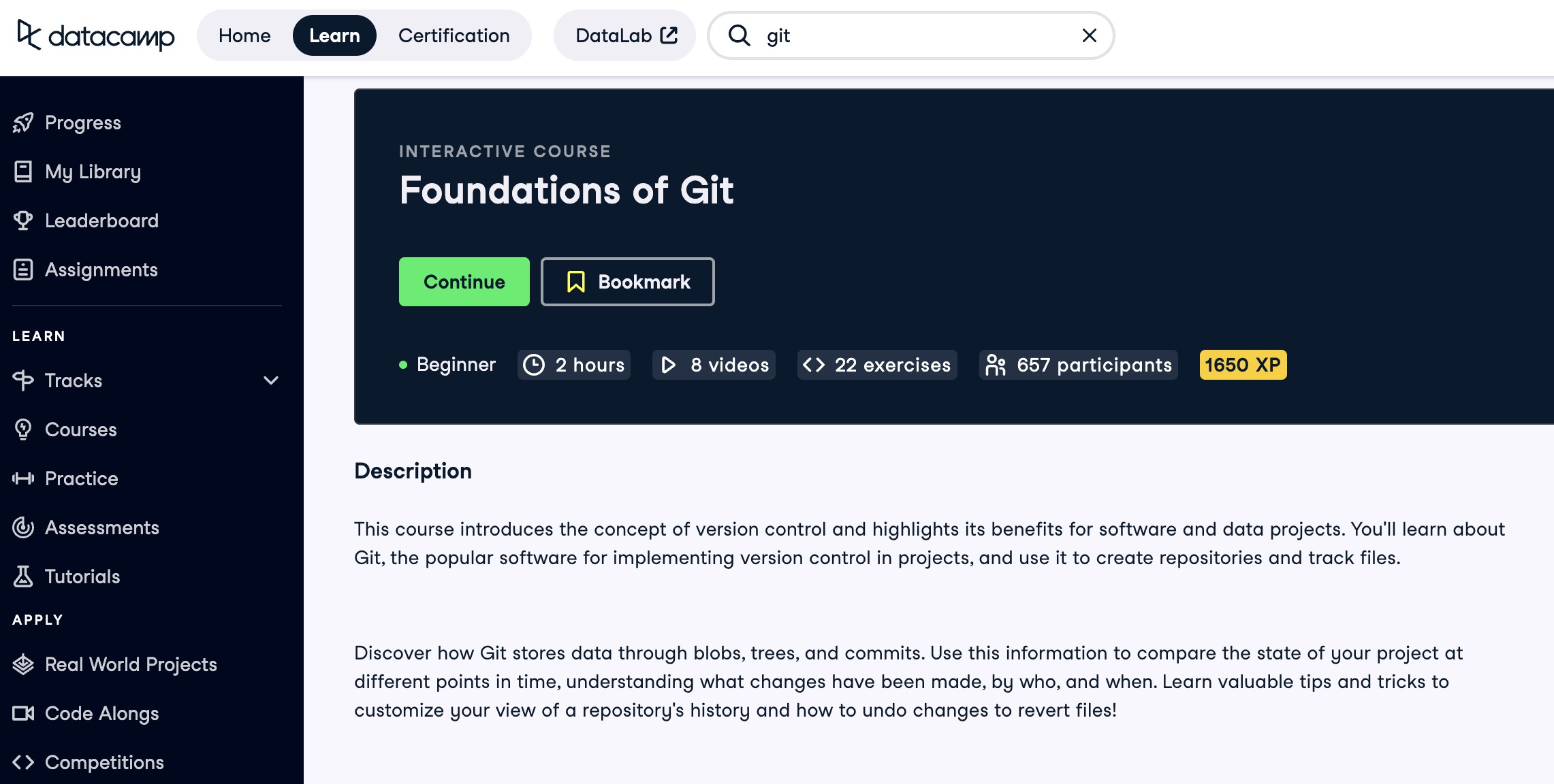
# What we will cover

 Branches

Remotes

Conflicts

# What you should know

 How Git stores data

How to create repos

How to make commits

How to compare versions

How to revert versions

# Branches

 Branch = an individual **version** of a repo

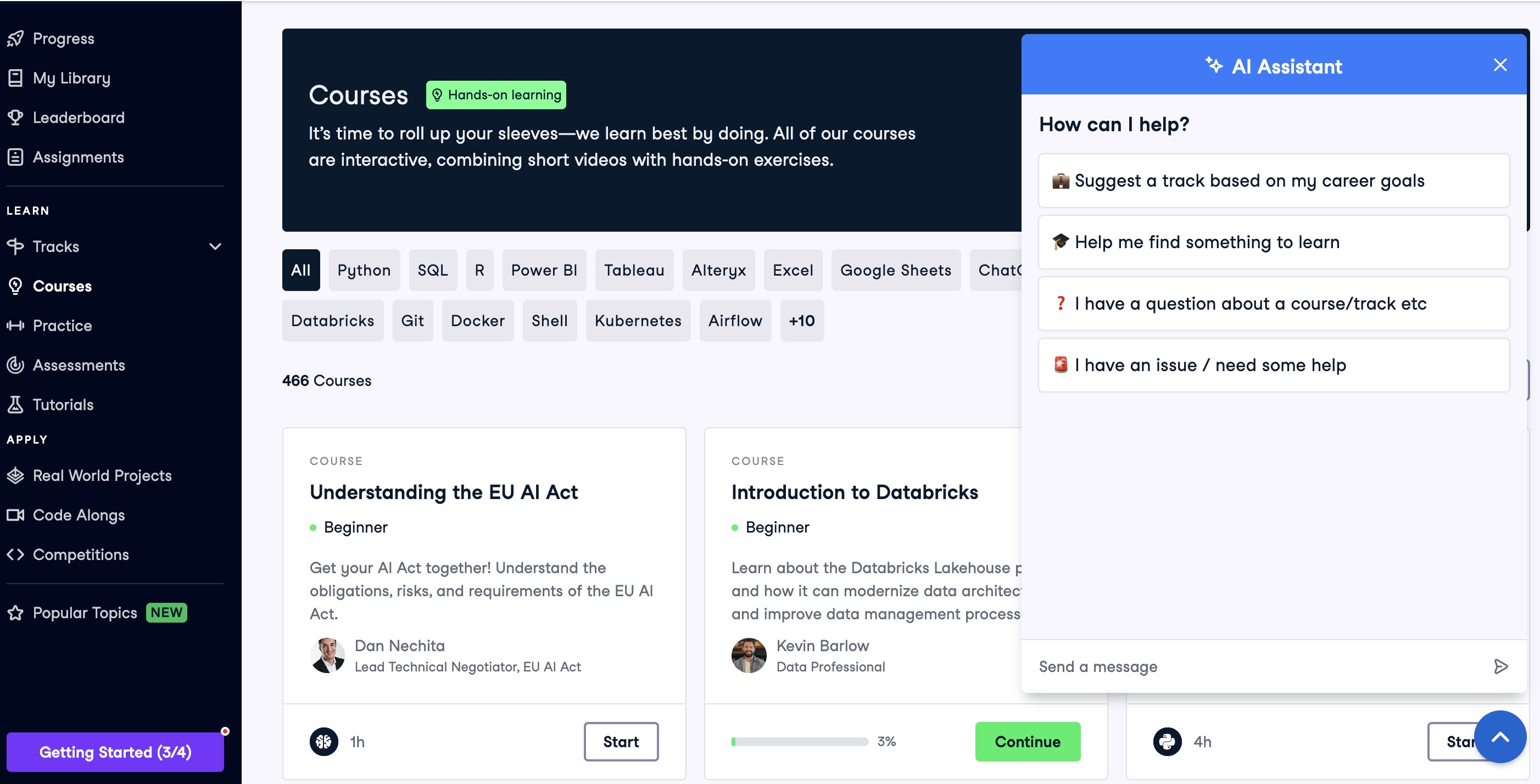
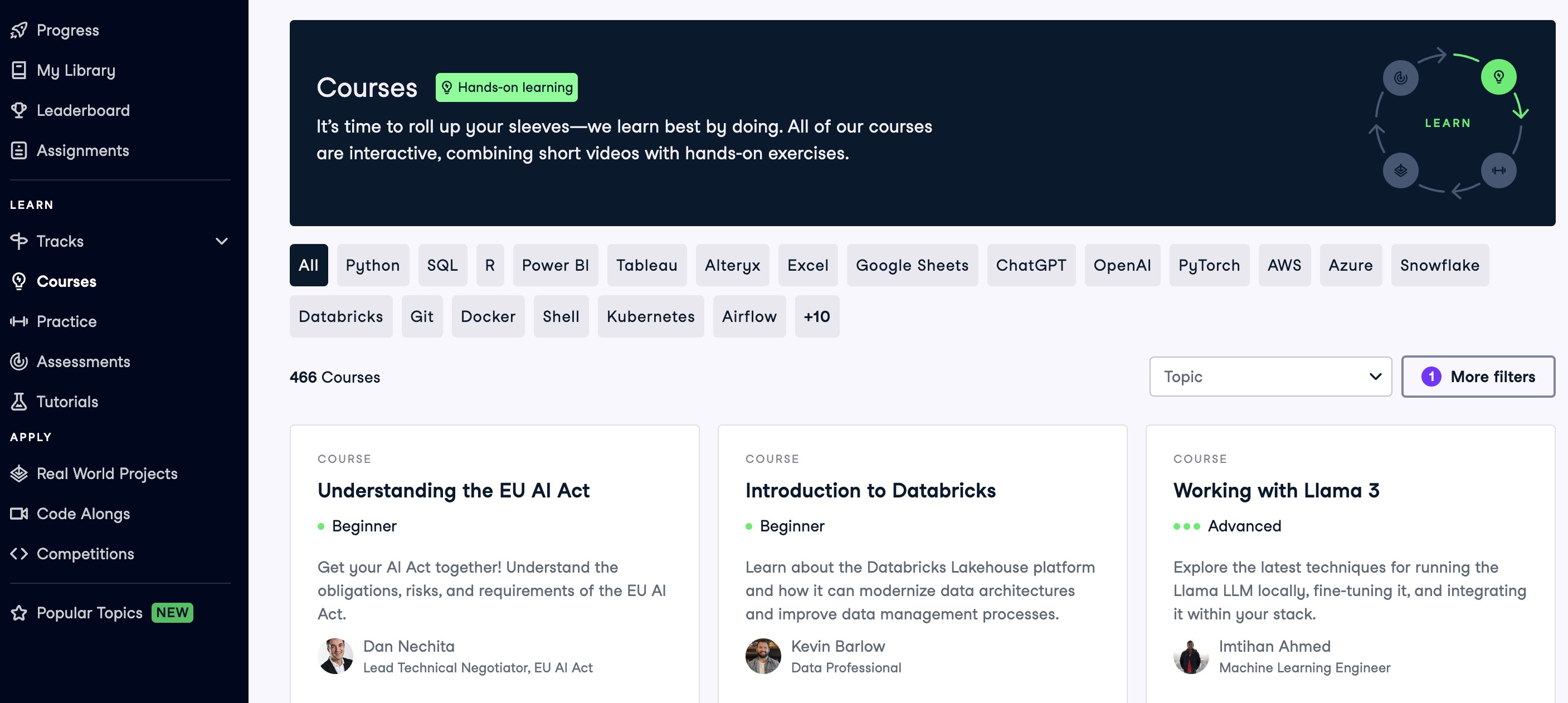
Git uses **branches** to systematically track multiple versions of files

In each branch: Some files might be the same

Others might be different Some may not exist at all

**Why use branches?**

**Live system Feature development**



Works as expected Might encounter issues during development

Default branch = main and testing

Doesn't affect the live system

**Why use branches?**

Multiple developers can work on a project **simultaneously**

**Compare the state** of a repo **between branches**

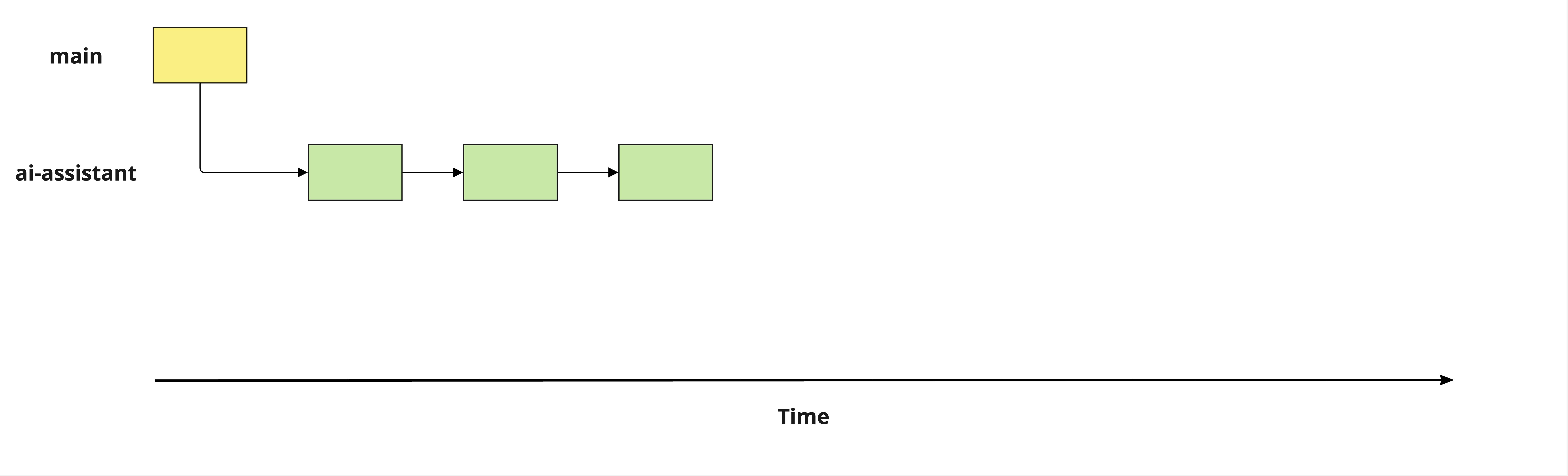
**Combine contents**, pushing new features to a live system

Each branch should have a **specific purpose**

# Visualizing branches



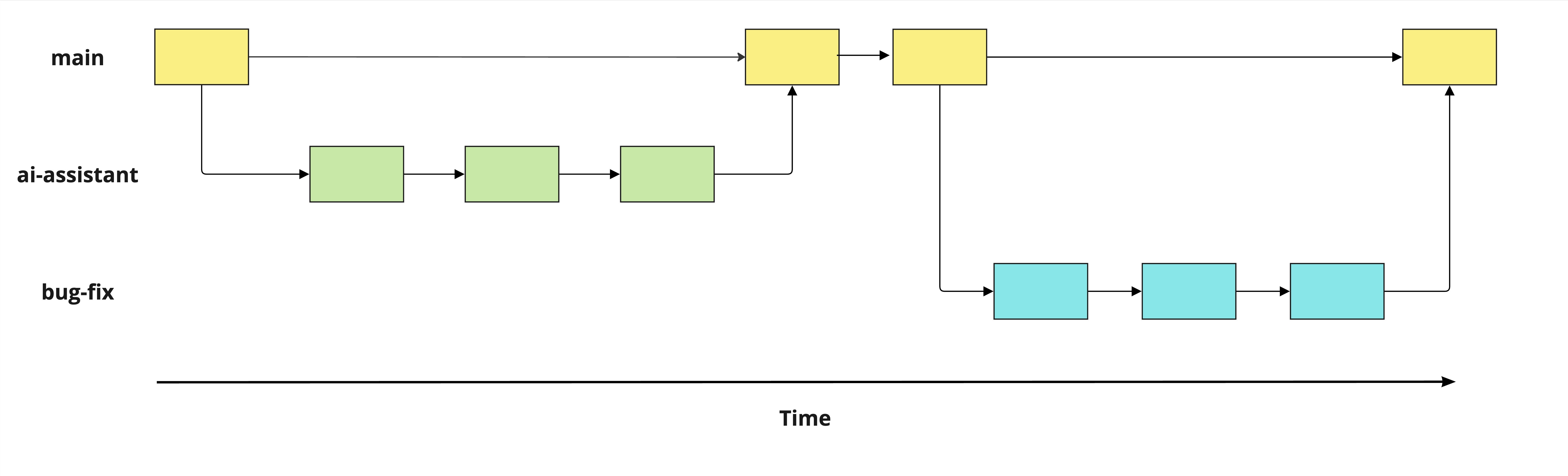
# Branching off



# Merging back into main



# Fixing a bug



# Identifying branches

Listing all branches

git branch

main

\* ai-assistant

\*

= current branch

# Switching between branches

git switch main

Switched to branch 'main'

# Creating a new branch

Create a new branch called

speed-test

git branch speed-test

Move to the

speed-test

branch

git switch speed-test

Switched to branch 'speed-test'

Create a new branch called

speed-test

**and**

switch to it

git switch -c speed-test

Switched to a new branch 'speed-test'

# Terminology

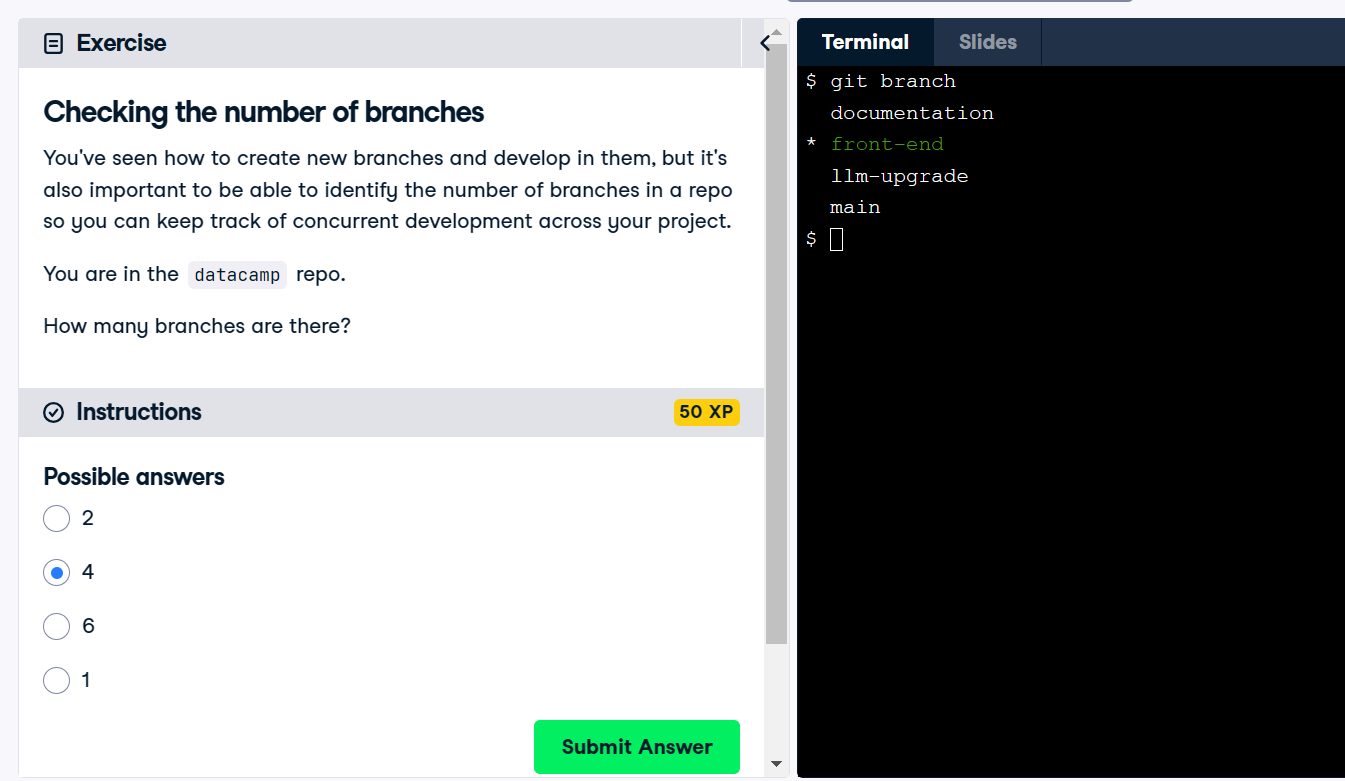
Creating a new branch = "branching off"

Creating speed-test from main = "branching off main "

**Let's practice!**

**INTERMEDIATE GIT**





**Modifying and comparing branches**

**INTERMEDIATE GIT**

## George Boorman

Curriculum Manager, DataCamp

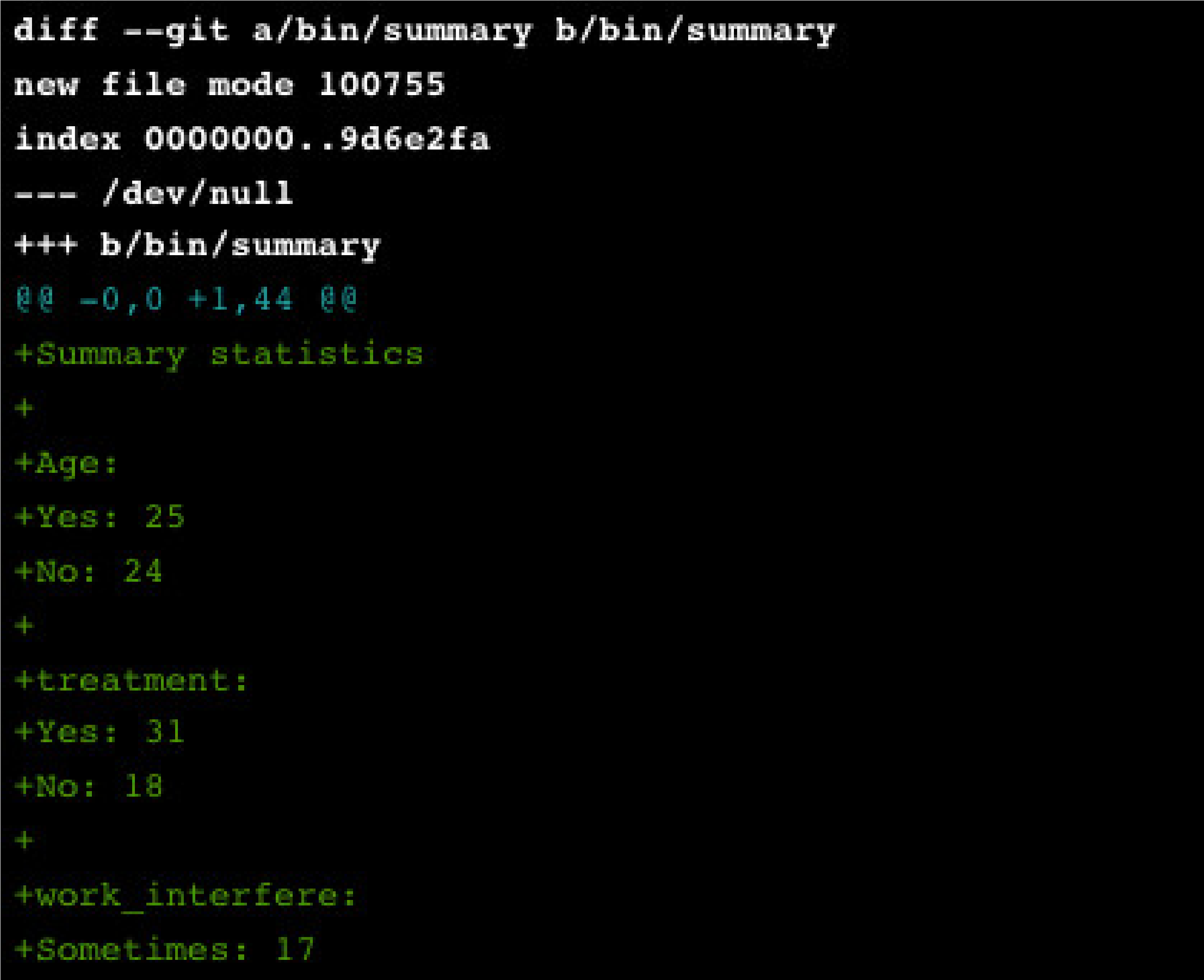
# Diff recap

|  |  |
| --- | --- |
| **Command** | **Function** |
| git diff | Show changes between all unstaged files and the latest commit |
| git diff report.md | Show changes between an unstaged file and the latest commit |
| git diff --staged | Show changes between all staged files and the latest commit |
| git diff --staged report.md | Show changes between a staged file and the latest commit |
| git diff 35f4b4d 186398f | Show changes between two commits using hashes |
| git diff HEAD~1 HEAD~2 | Show changes between two commits using HEAD instead of commit hashes |

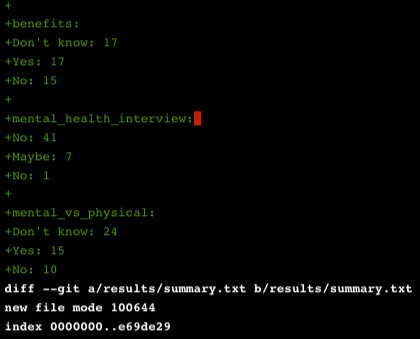
# Comparing branches

git diff main summary-statistics

# git diff output



# git diff output



# Navigating large git outputs

Can produce large outputs!

Press space to progress through and q to exit

# Modifying branches

git branch

main

\* feature\_dev

feature\_dev

Need another branch for a second new feature being developed

Solution - rename feature\_dev

Renaming a branch

git branch -m

# Renaming a branch

git branch

main

\* feature\_dev

feature\_dev

Need another branch for a second new feature being developed

Solution - rename feature\_dev

Renaming a branch

git branch -m feature\_dev chatbot

# Checking our branches

git branch

main

\* chatbot

# Deleting a branch

Large projects can have many branches

Delete branches once we are finished with them

Delete the chatbot branch with -d flag

git branch -d chatbot

Deleted branch chatbot (was 3edb989).

# Deleting a branch that hasn't been merged

If chatbot hasn't been merged to main , git branch -d chatbot will produce an error

error: The branch 'chatbot' is not fully merged.

If you are sure you want to delete it, run 'git branch -D chatbot'.

Delete with

-

D

flag

git branch -D chatbot

Deleted branch chatbot (was 3edb989).

Difficult, but not impossible, to recover deleted branches

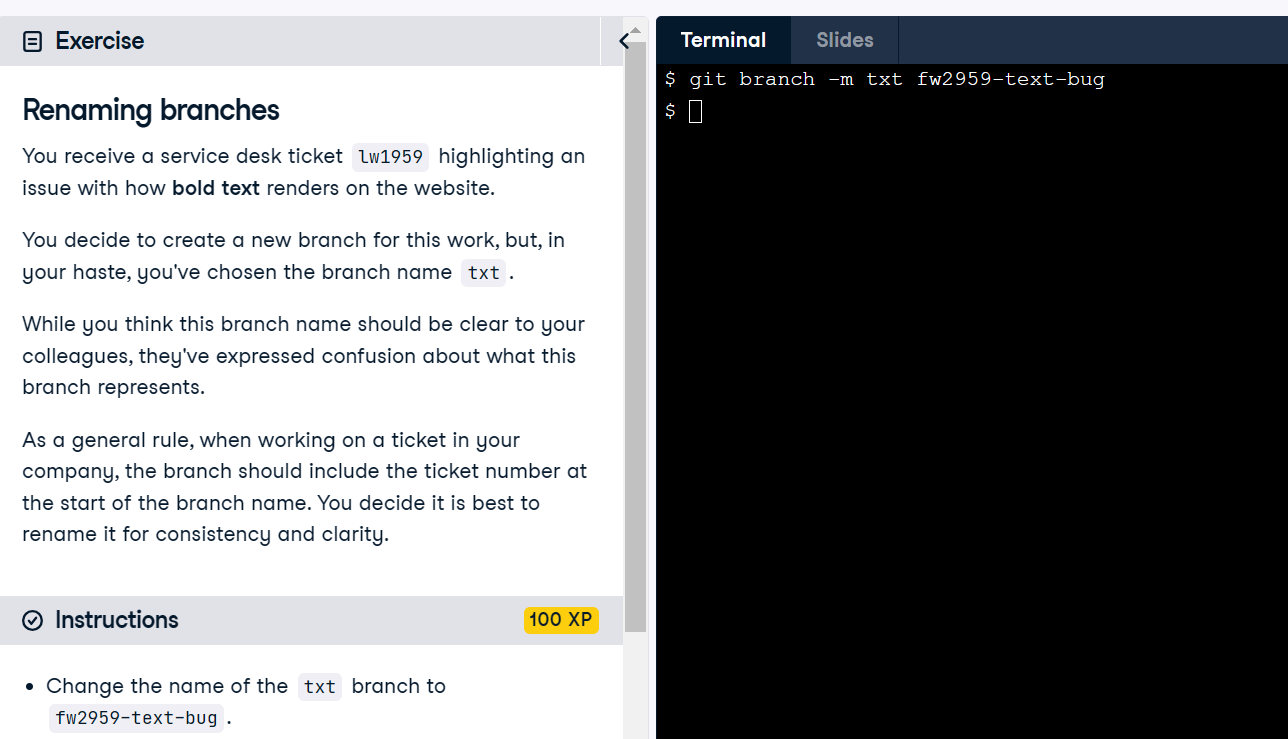
Be sure we don't need the branch any more before deleting!

# Summary

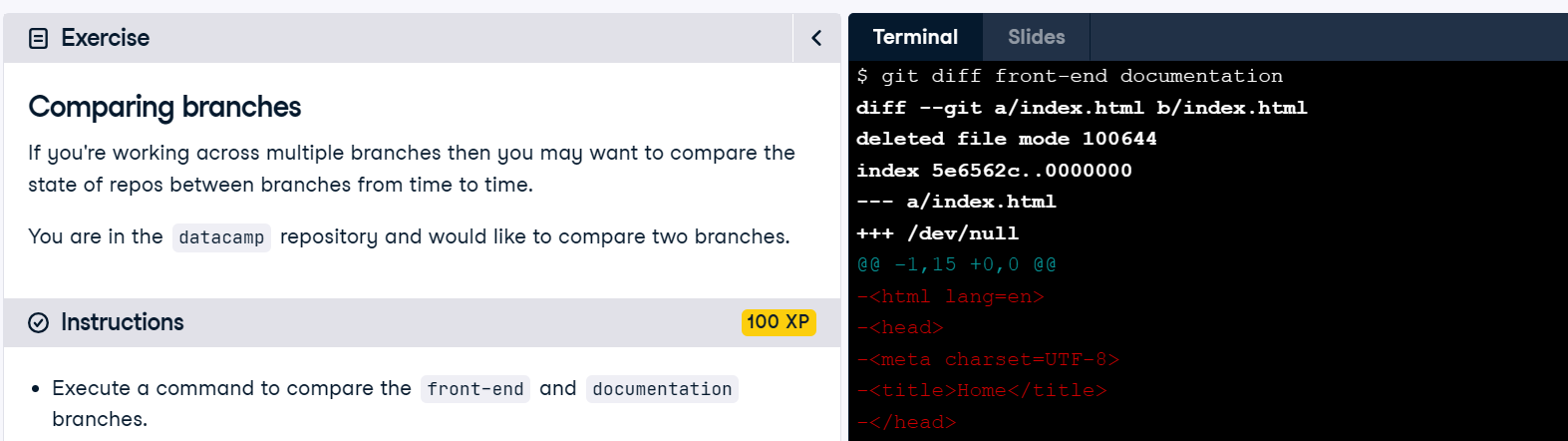
|  |  |
| --- | --- |
| **Command** | **Function** |
| git diff main chatbot | Compare the state of the main and chatbot branches |
| git branch | List all branches |
| git branch -m old\_name new\_name | Rename branch called old\_name to new\_name |
| git branch -d chatbot | Delete chatbot branch, which **has** been merged |
| git branch -D chatbot | Delete chatbot branch, which **has not** been merged |

**Let's practice!**

**INTERMEDIATE GIT**







**Merging branches**

**INTERMEDIATE GIT**

## George Boorman

Curriculum Manager, DataCamp

# The purpose of branches

Each branch should have a particular purpose Developing a new feature

Debugging an error

Once the task is complete, we incorporate the changes into production Typically the main branch - "ground truth"

# Source and destination

When merging two branches:

the last commits from each branch are called **parent commits**

source —the branch we want to merge **from** destination —the branch we want to merge **into**

When merging ai-assistant into main :

ai-assistant = source main = destination

# Merging branches

From another branch:

git merge source destination

git merge ai-assistant main

Move to the destination branch:

git switch main

git merge source

From

main

, to merge

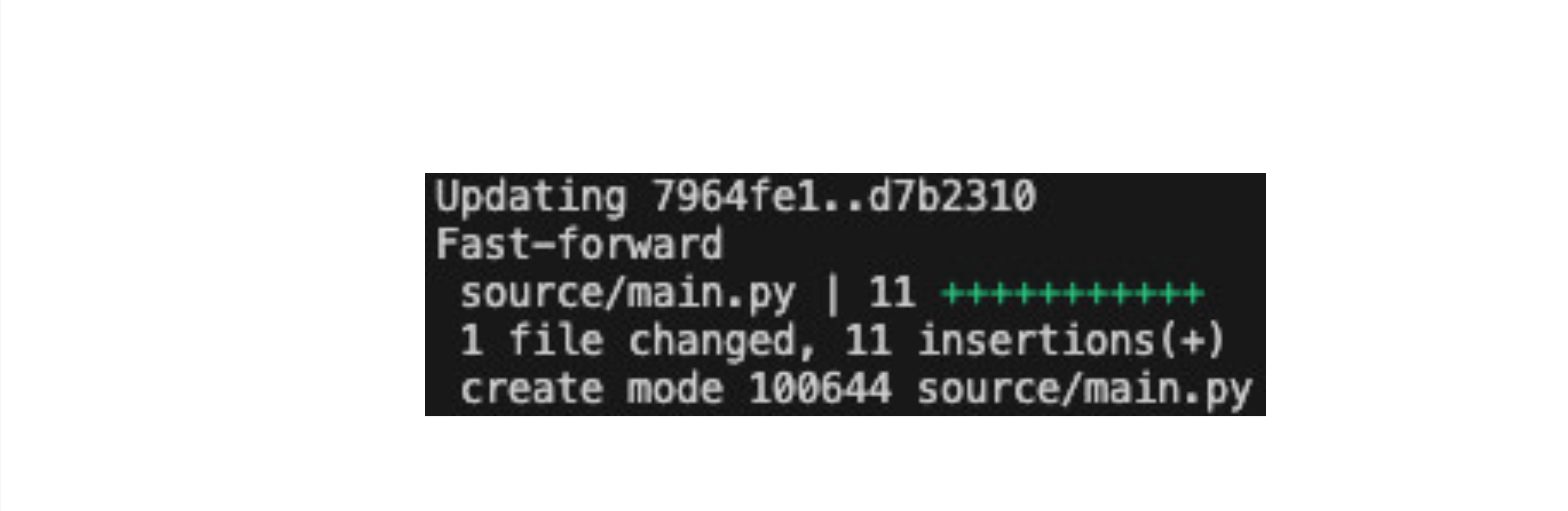
ai-assistant

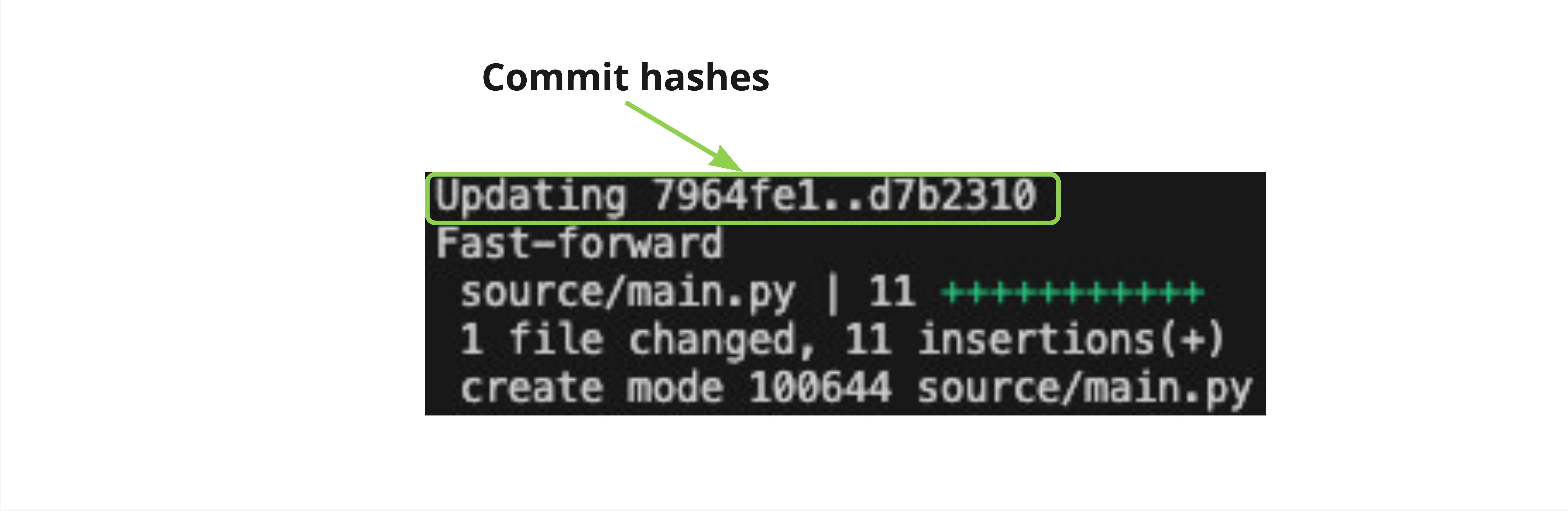
into

main

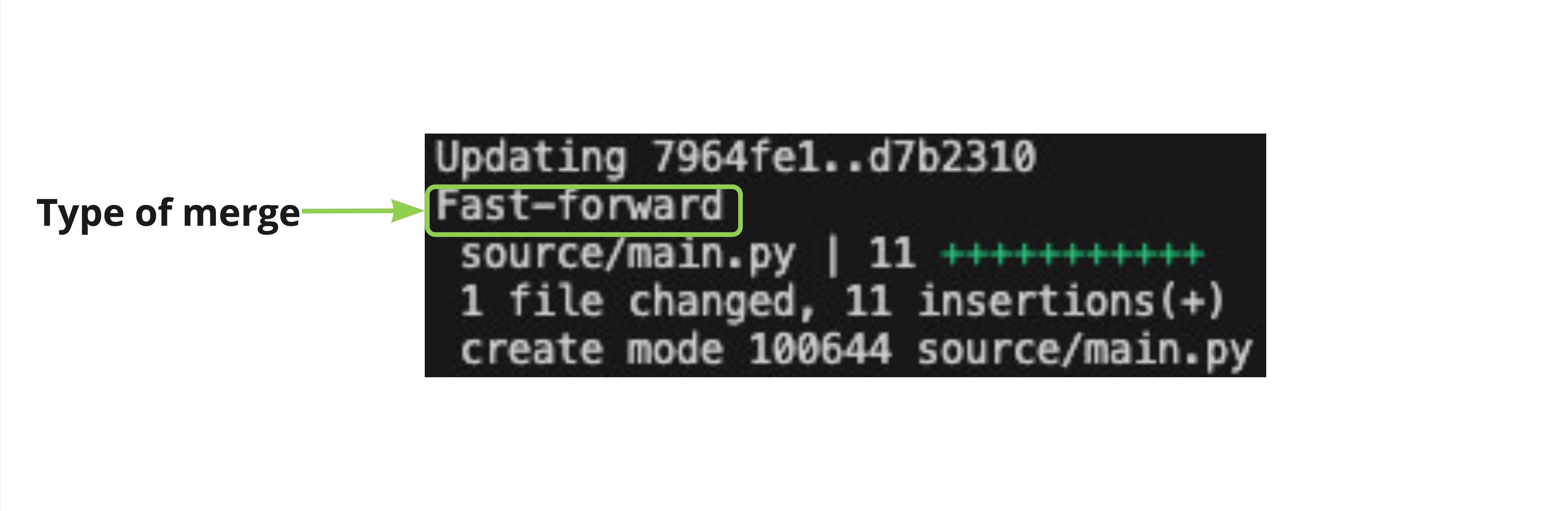
:

git merge ai-assistant

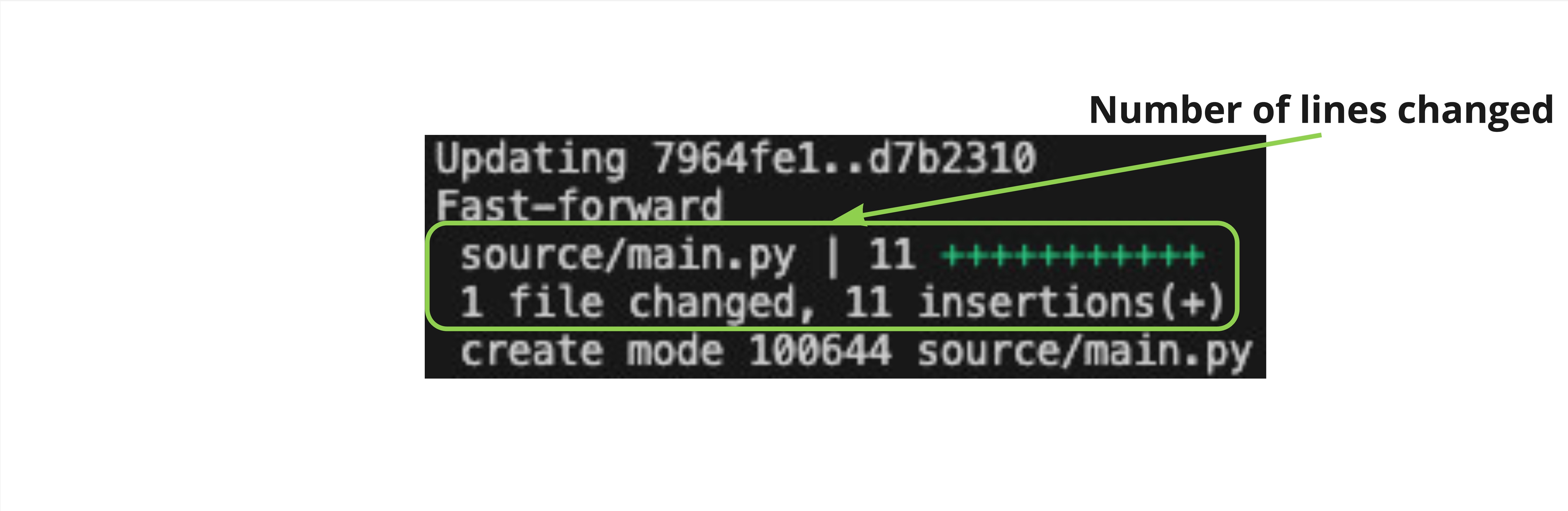


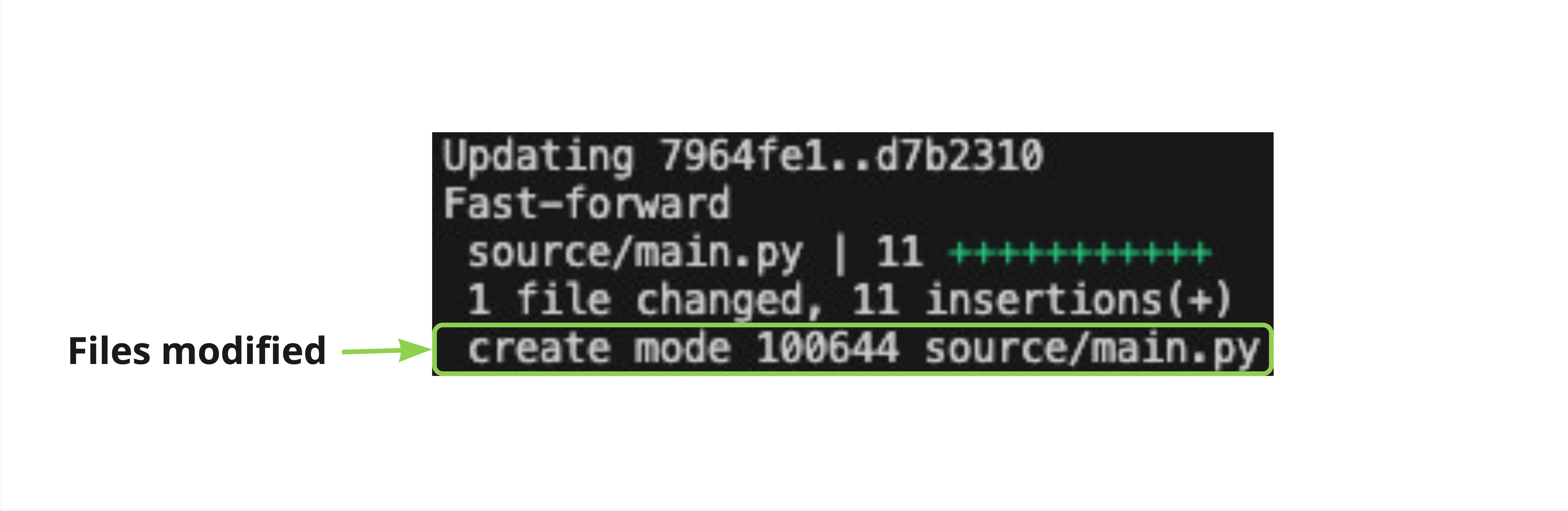


Parent commits Linear commit history: branched off main to create ai-assistant



Fast-forward: point main to the last commit in the ai-assistant branch





**Let's practice!**

**INTERMEDIATE GIT**

