Git Advanced 2
[SWE2021]

JinYeong Bak
jy.bak@skku.edu
Human Language Intelligence Lab, SKKU

Slide credit: HyunJin Kim / khyunjin1993@g.skku.edu

Objective

By the end of this class, you will understand…

- Basic branching and merging
- Managing branch
- Basic merge conflict
- Branching workflow

Basic branching and merging

Imagine a workflow that looks like the following …

- Do some work on a repository **“main”** or **“master”**
- Create a branch **“even_list”** for developing a new function
- Do some work in the **“even_list”** branch

At this stage, you have encountered an issue that need to be fixed at **“main”** branch

Basic branching and merging

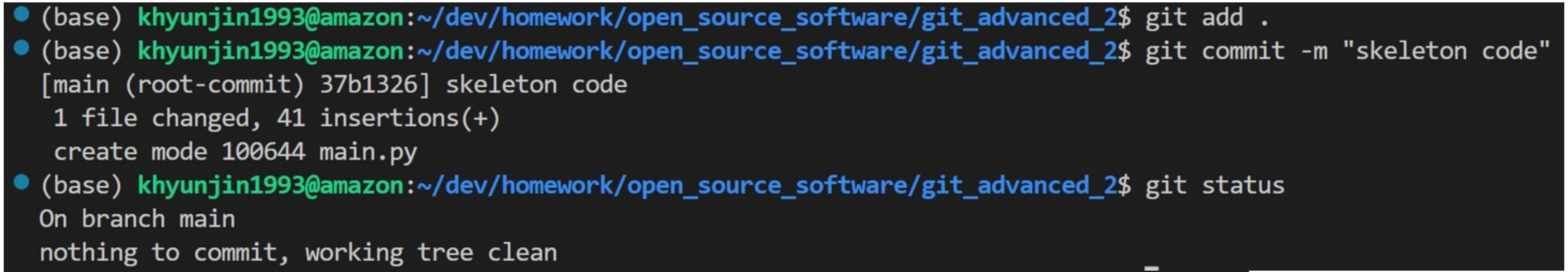
Since the issue should be fixed, you need a hotfix (i.e., small patch)

- Switch your branch to **“main”**
- Create a branch **“hotfix”** to add the hotfix
- Fix codes in the **“hotfix”** branch and commit
- Merge branch **“main”** with **“hotfix”**
- Merge branch **“main”** with **“even_list”** branch

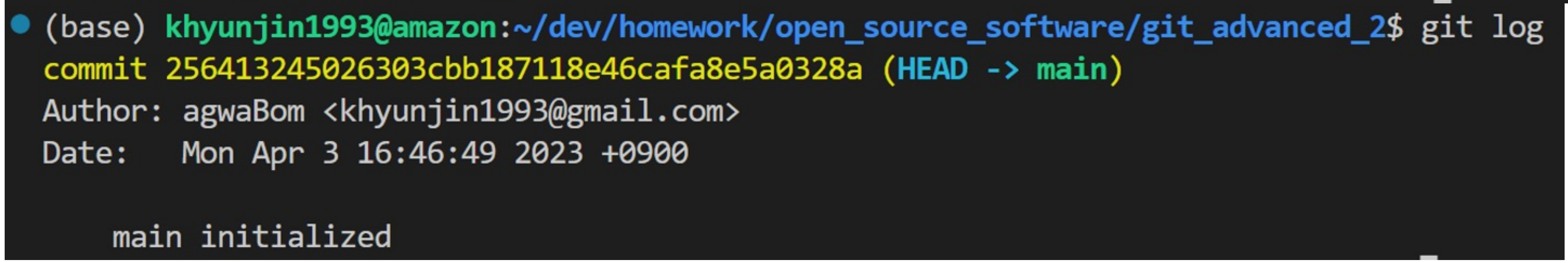
Basic branching and merging

Do some work on a repository **"main"** or **"master"**

- Create a new repository "git_advanced_2" and create a "main.py" with skeleton code in HERE
- Add and commit in the **"main"** branch



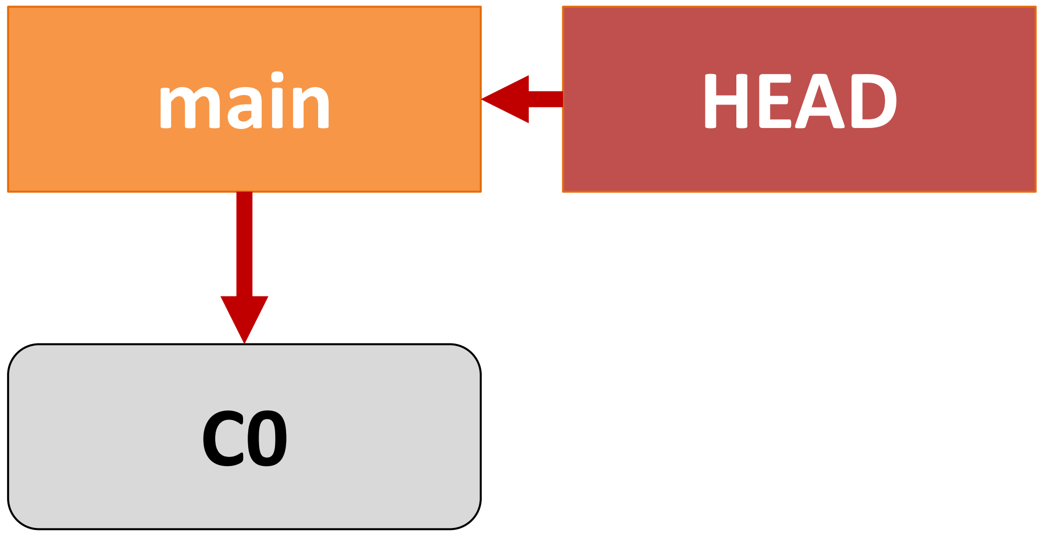
```
● (base) khyunjin1993@amazon:~/dev/homework/open_source_software/git_advanced_2$ git add .
● (base) khyunjin1993@amazon:~/dev/homework/open_source_software/git_advanced_2$ git commit -m "skeleton code"
  [main (root-commit) 37b1326] skeleton code
   1 file changed, 41 insertions(+)
   create mode 100644 main.py
● (base) khyunjin1993@amazon:~/dev/homework/open_source_software/git_advanced_2$ git status
  On branch main
  nothing to commit, working tree clean
```



```
● (base) khyunjin1993@amazon:~/dev/homework/open_source_software/git_advanced_2$ git log
  commit 256413245026303cbb187118e46cafa8e5a0328a (HEAD -> main)
  Author: agwaBom <khyunjin1993@gmail.com>
  Date:   Mon Apr 3 16:46:49 2023 +0900

      main initialized
```

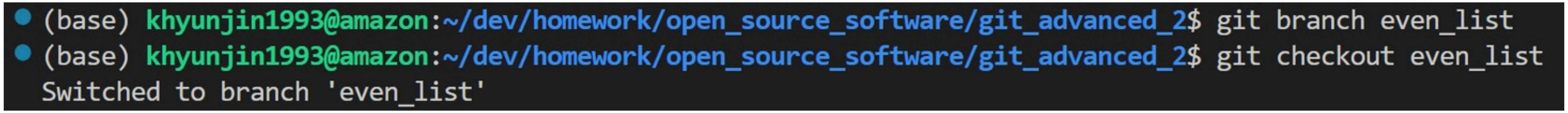
Basic branching and merging

Do some work on a repository **“main”** or **“master”**

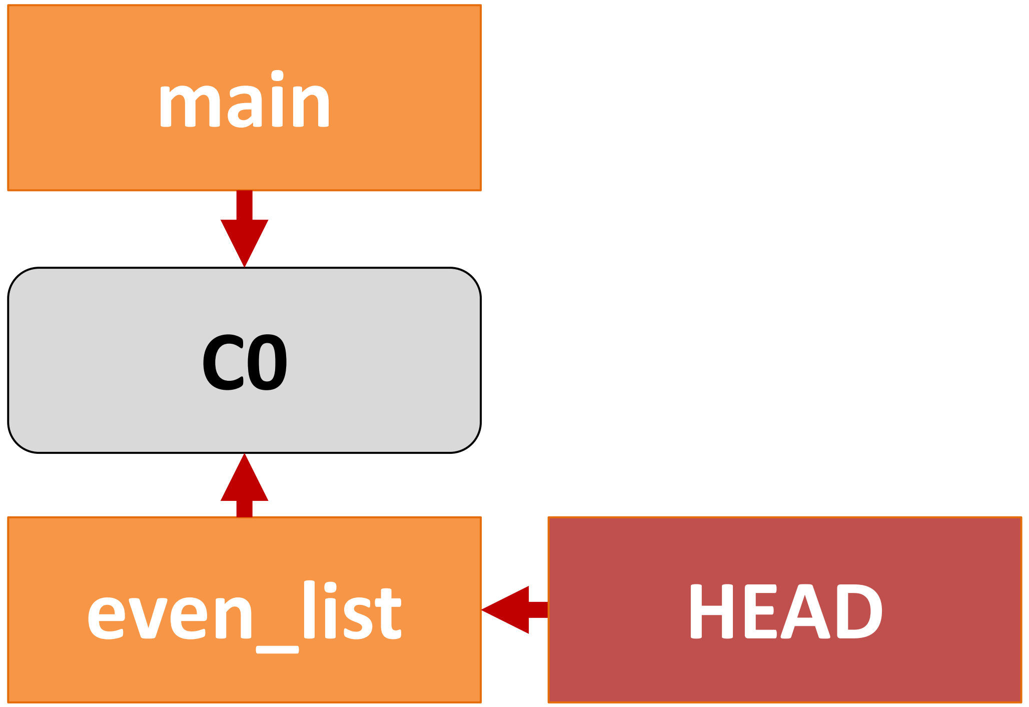


Basic branching and merging

Create a branch **"even_list"** for a new function you're working on

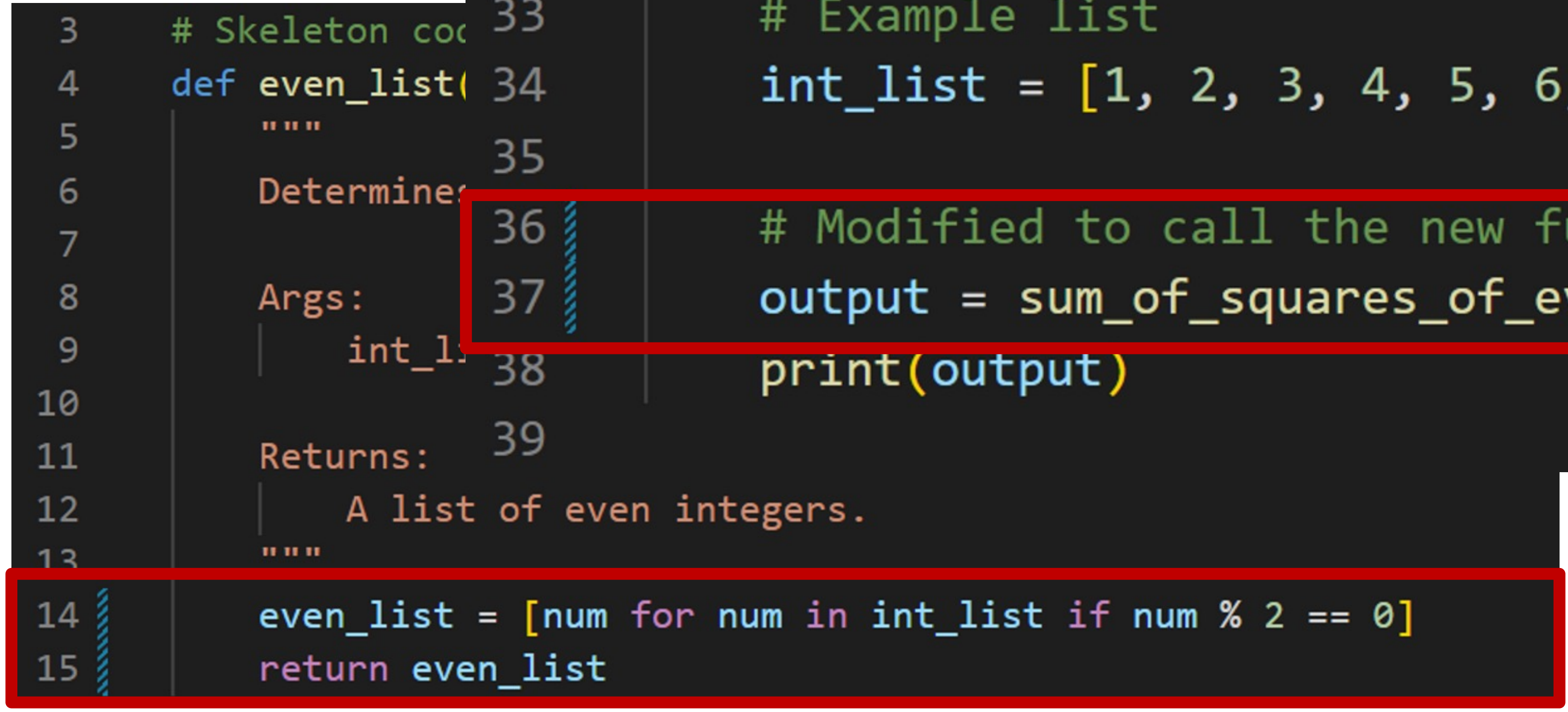


```
● (base) khyunjin1993@amazon:~/dev/homework/open_source_software/git_advanced_2$ git branch even_list
● (base) khyunjin1993@amazon:~/dev/homework/open_source_software/git_advanced_2$ git checkout even_list
  Switched to branch 'even_list'
```

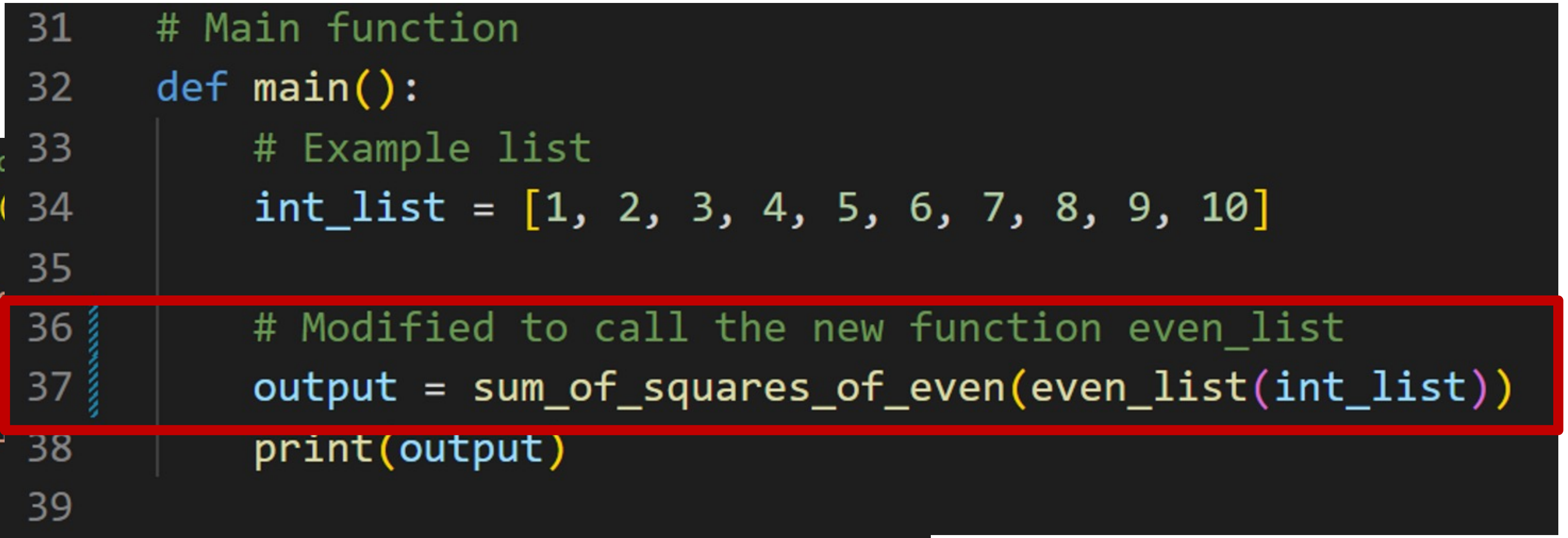


Basic branching and merging

- Do some work in the **“even_list”** branch
- Click HERE to refer changes



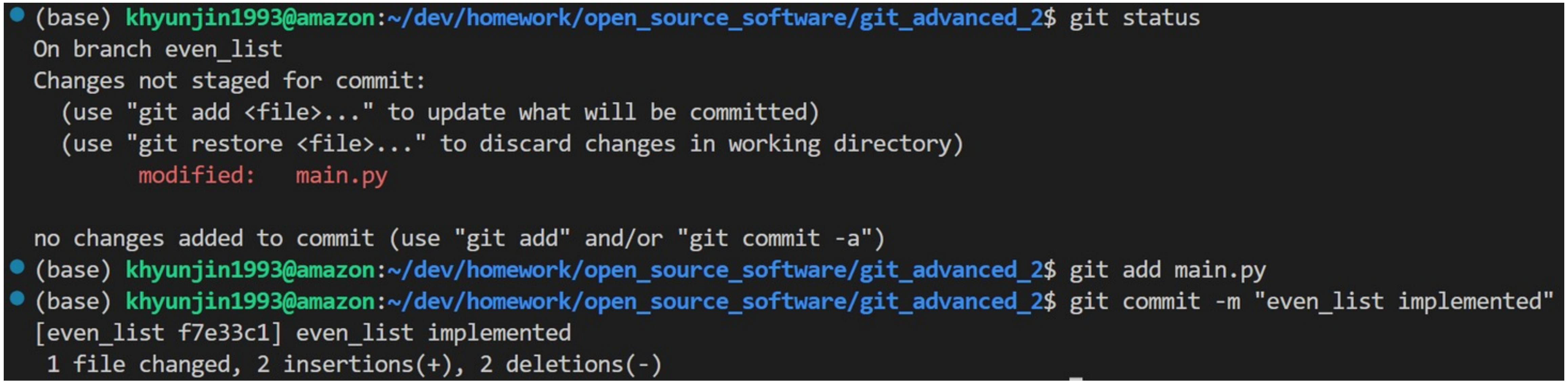
```
# Skeleton coc
def even_list(
    """
    Determine
    
    Args:
        int_li

    Returns:
        A list of even integers.
    """
    even_list = [num for num in int_list if num % 2 == 0]
    return even_list
```



```
# Main function
def main():
    # Example list
    int_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

    # Modified to call the new function even_list
    output = sum_of_squares_of_even(even_list(int_list))
    print(output)

```

Basic branching and merging

- Do some work in the **"even_list"** branch
- Check **"main.py"** has been modified and commit

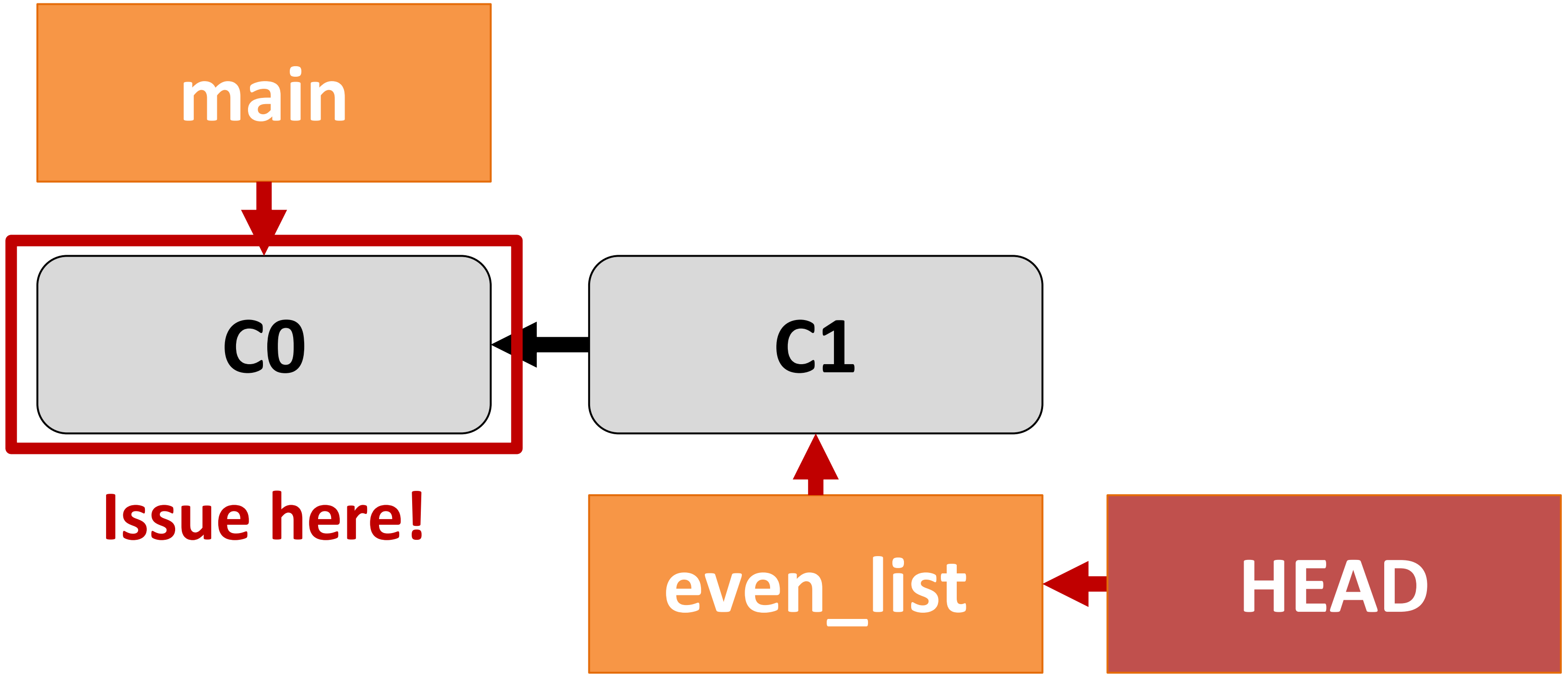


```
(base) khyunjin1993@amazon:~/dev/homework/open_source_software/git_advanced_2$ git status
On branch even_list
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   main.py

no changes added to commit (use "git add" and/or "git commit -a")
(base) khyunjin1993@amazon:~/dev/homework/open_source_software/git_advanced_2$ git add main.py
(base) khyunjin1993@amazon:~/dev/homework/open_source_software/git_advanced_2$ git commit -m "even_list implemented"
[even_list f7e33c1] even_list implemented
 1 file changed, 2 insertions(+), 2 deletions(-)
```

Basic branching and merging

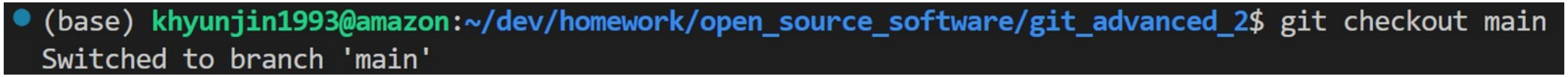
Do some work in the **"even_list"** branch

- Assume, we've encountered an issue at this stage



Basic branching and merging

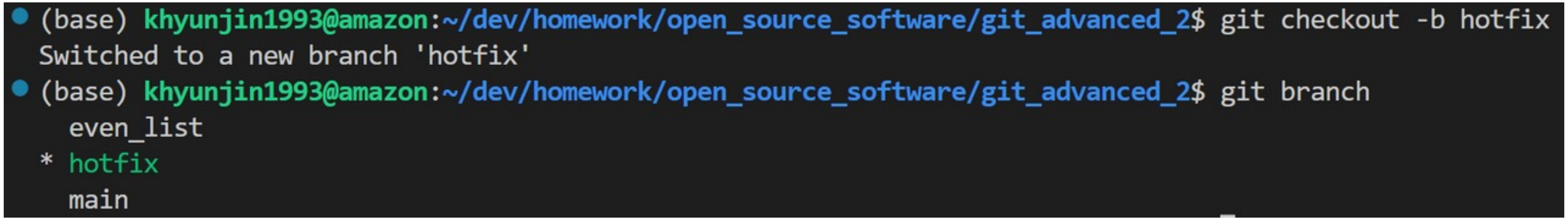
- Switch to your production branch **"main"** or **"master"**



```
(base) khyunjin1993@amazon:~/dev/homework/open_source_software/git_advanced_2$ git checkout main
Switched to branch 'main'
```

- Create a branch **"hotfix"** to add the hotfix

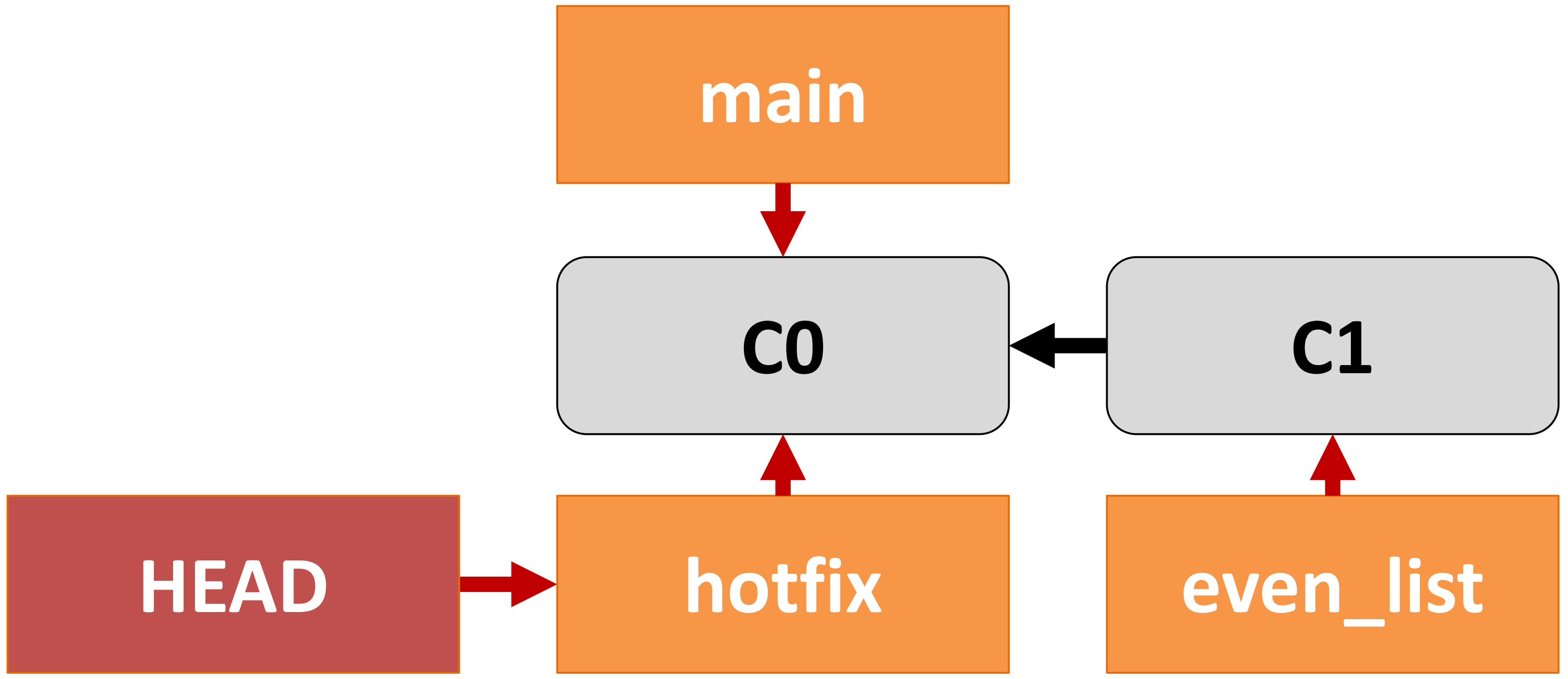
  "-b" flag enables to automatically make and change into a newly made branch



```
(base) khyunjin1993@amazon:~/dev/homework/open_source_software/git_advanced_2$ git checkout -b hotfix
Switched to a new branch 'hotfix'
(base) khyunjin1993@amazon:~/dev/homework/open_source_software/git_advanced_2$ git branch
  even_list
* hotfix
  main
```

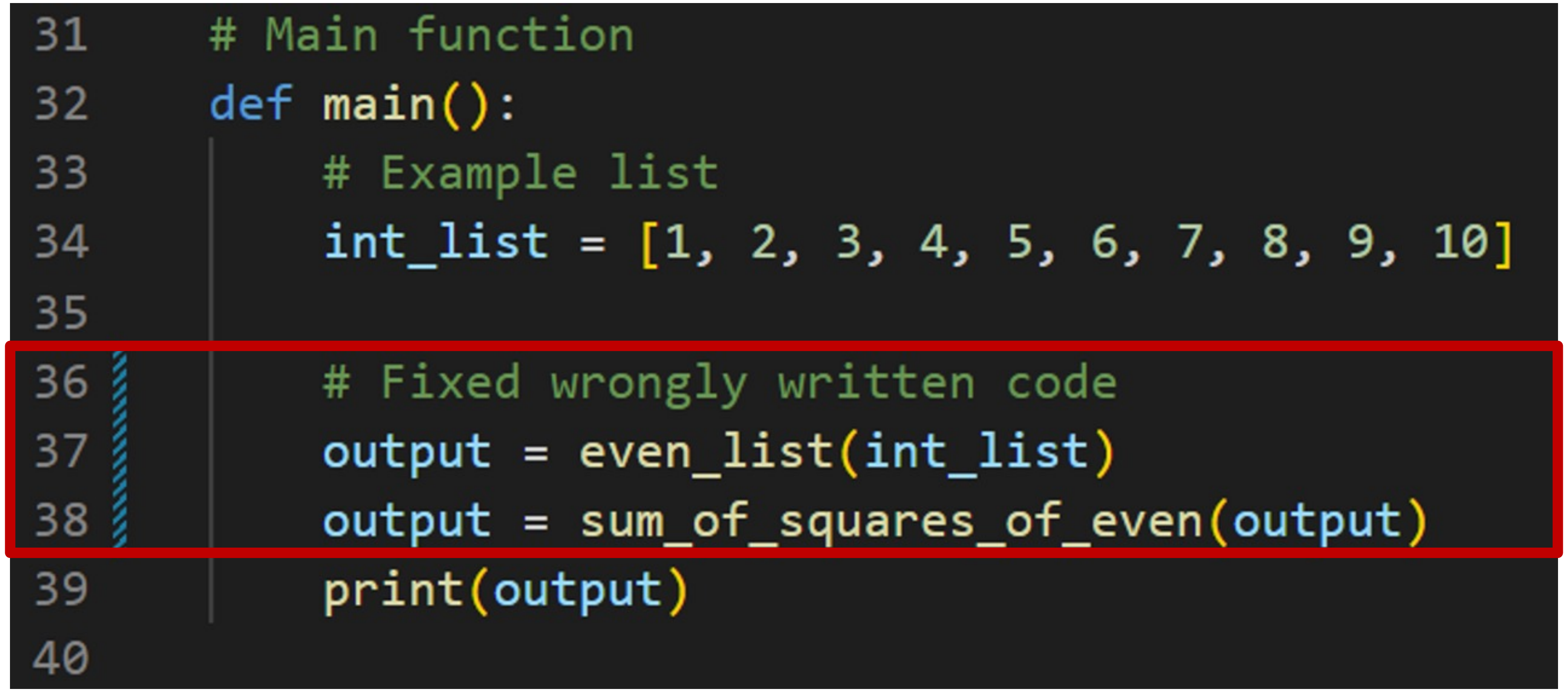
Basic branching and merging

- Switch to your production branch **“main”** or **“master”**
- Create a branch **“hotfix”** to add the hotfix



Basic branching and merging

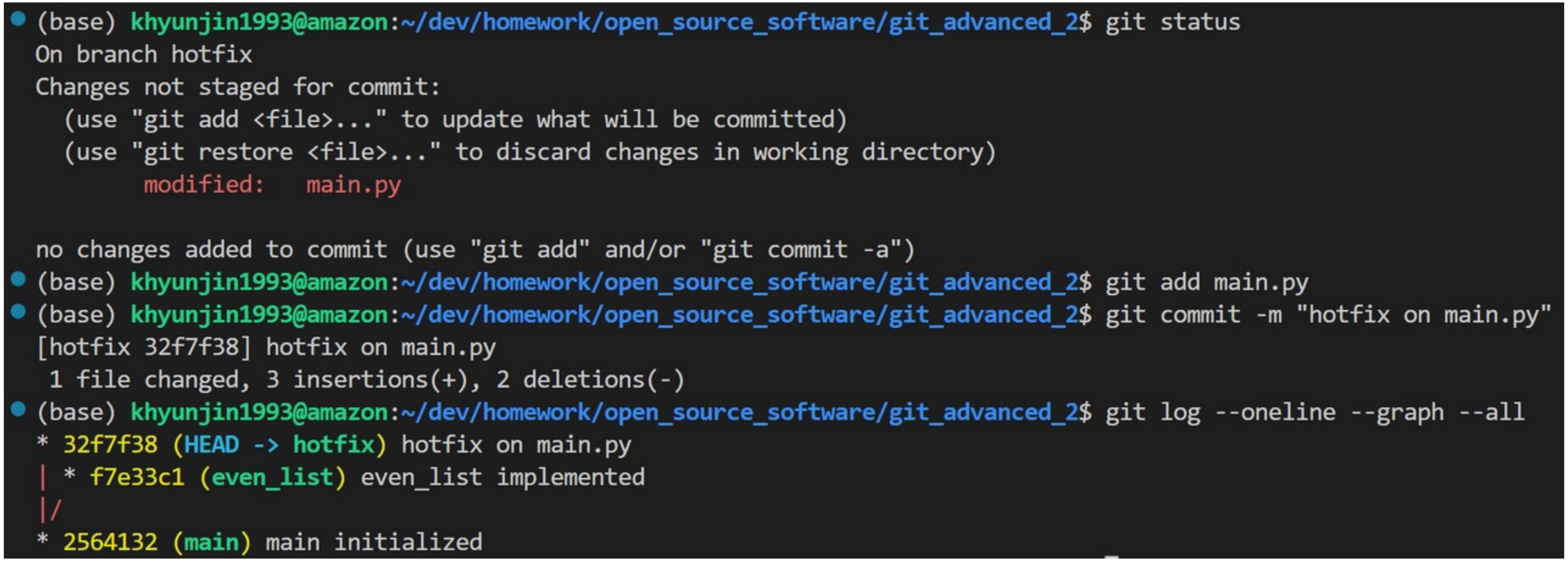
- Fix codes in the **"hotfix"** branch, and commit
- Details on changes can be found HERE



```
# Main function
def main():
    # Example list
    int_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

    # Fixed wrongly written code
    output = even_list(int_list)
    output = sum_of_squares_of_even(output)
    print(output)

```

Basic branching and merging

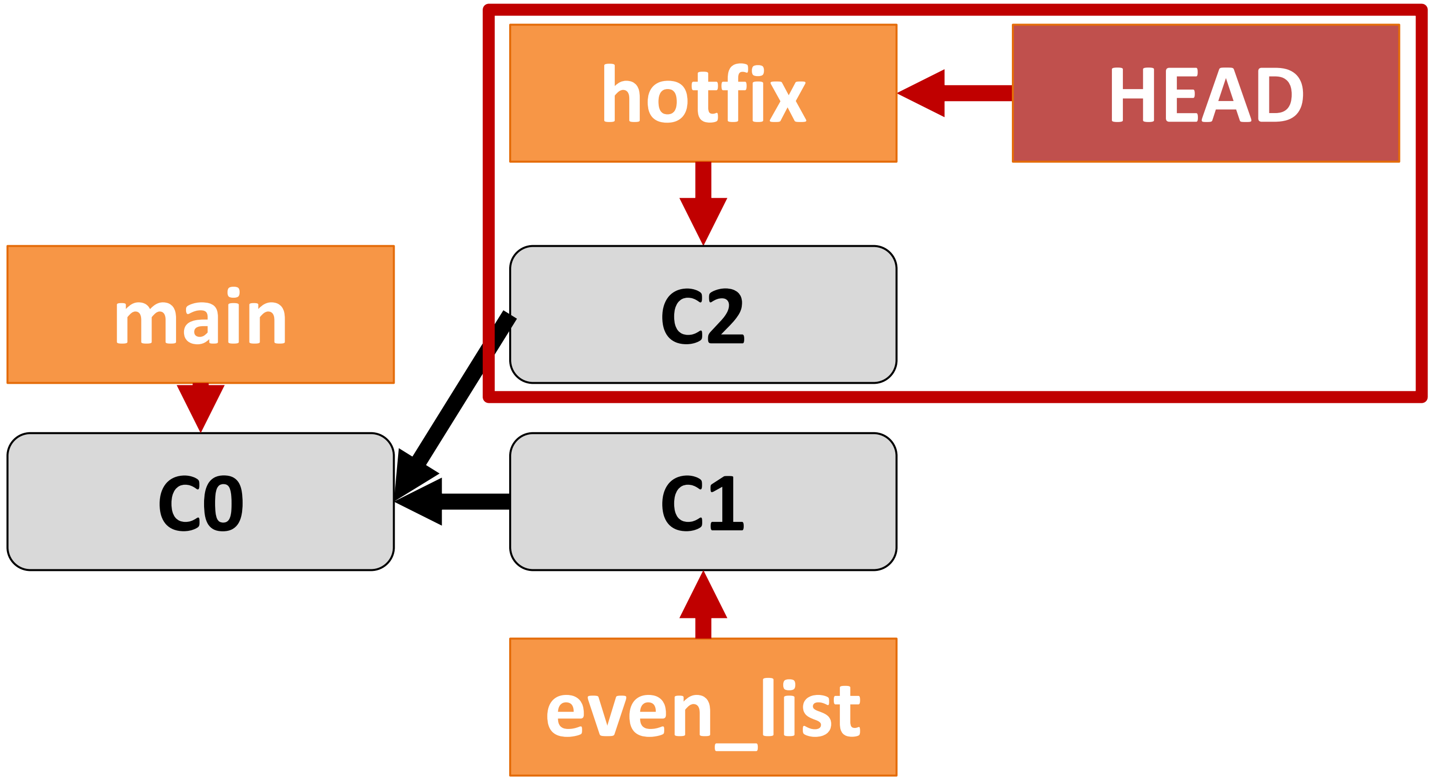
- Fix codes in the **“hotfix”** branch, and commit
- Details on changes can be found HERE



```
● (base) khyunjin1993@amazon:~/dev/homework/open_source_software/git_advanced_2$ git status
  On branch hotfix
  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:   main.py

  no changes added to commit (use "git add" and/or "git commit -a")
● (base) khyunjin1993@amazon:~/dev/homework/open_source_software/git_advanced_2$ git add main.py
● (base) khyunjin1993@amazon:~/dev/homework/open_source_software/git_advanced_2$ git commit -m "hotfix on main.py"
  [hotfix 32f7f38] hotfix on main.py
   1 file changed, 3 insertions(+), 2 deletions(-)
● (base) khyunjin1993@amazon:~/dev/homework/open_source_software/git_advanced_2$ git log --oneline --graph --all
  * 32f7f38 (HEAD -> hotfix) hotfix on main.py
  | * f7e33c1 (even_list) even_list implemented
  |/
  * 2564132 (main) main initialized
```

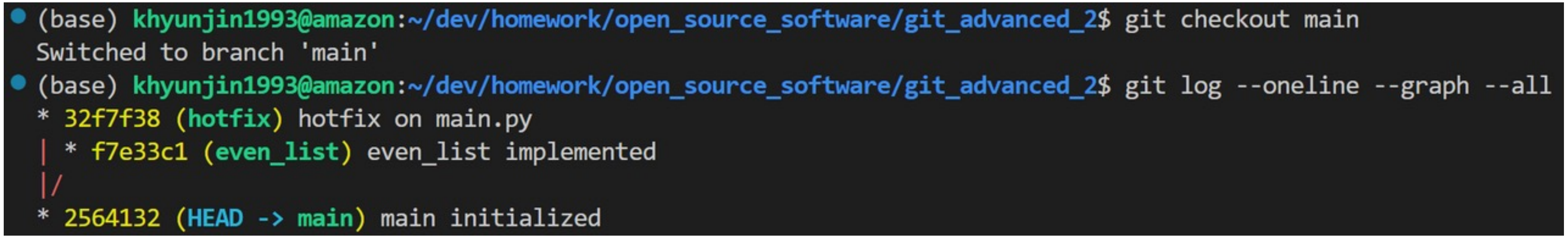
Basic branching and merging

- Fix codes in the **“hotfix”** branch, and commit
- Details on changes can be found HERE

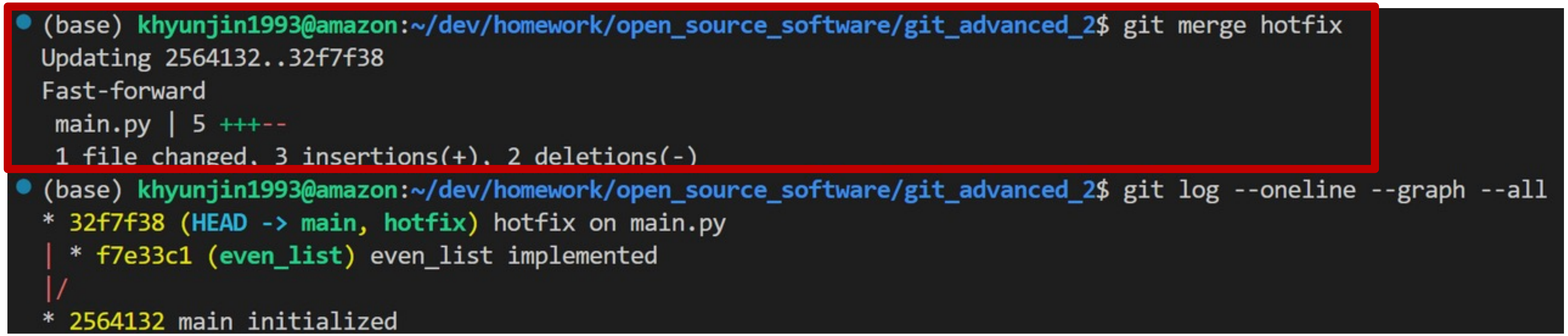


Basic branching and merging

Merge branch **“main”** with **“hotfix”** (using **“git merge <branch_name>”**)



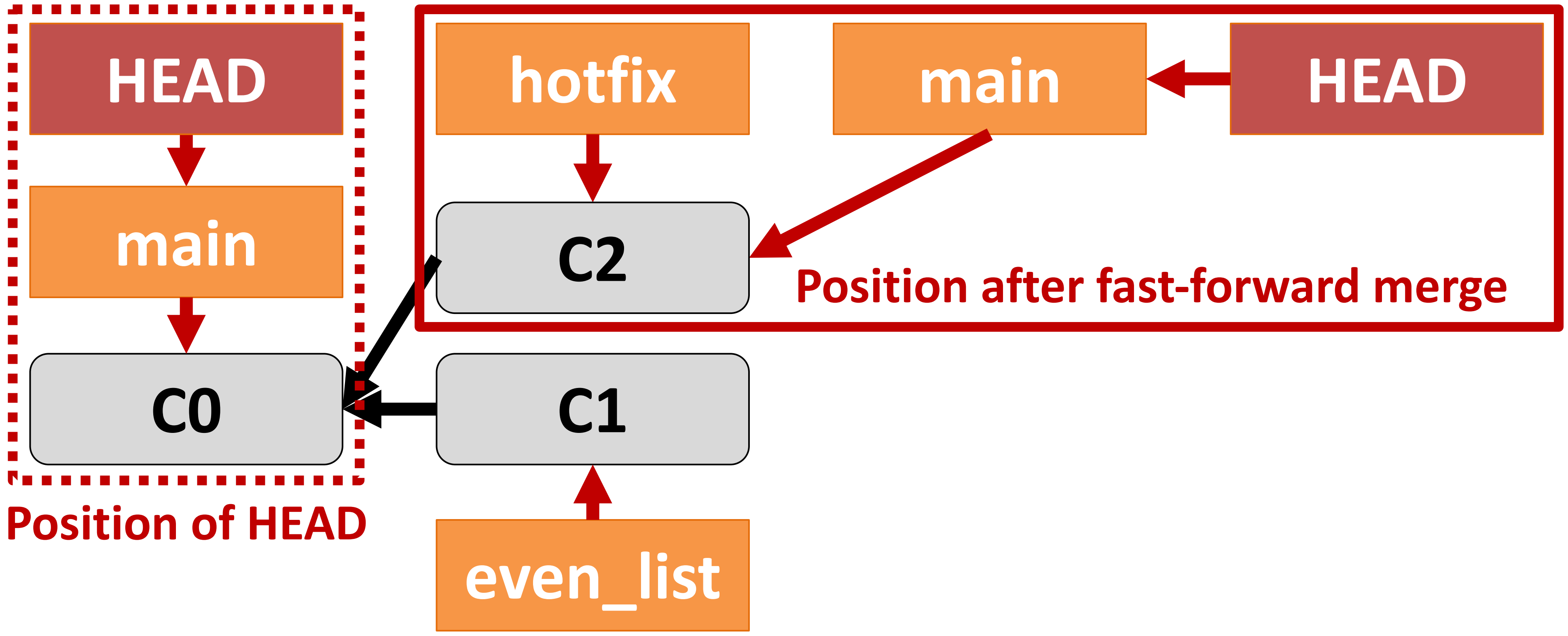
```
● (base) khyunjin1993@amazon:~/dev/homework/open_source_software/git_advanced_2$ git checkout main
  Switched to branch 'main'
● (base) khyunjin1993@amazon:~/dev/homework/open_source_software/git_advanced_2$ git log --oneline --graph --all
  * 32f7f38 (hotfix) hotfix on main.py
  | * f7e33c1 (even_list) even_list implemented
  |/
  * 2564132 (HEAD -> main) main initialized
```



```
● (base) khyunjin1993@amazon:~/dev/homework/open_source_software/git_advanced_2$ git merge hotfix
  Updating 2564132..32f7f38
  Fast-forward
   main.py | 5 +++--
   1 file changed, 3 insertions(+), 2 deletions(-)
● (base) khyunjin1993@amazon:~/dev/homework/open_source_software/git_advanced_2$ git log --oneline --graph --all
  * 32f7f38 (HEAD -> main, hotfix) hotfix on main.py
  | * f7e33c1 (even_list) even_list implemented
  |/
  * 2564132 main initialized
```

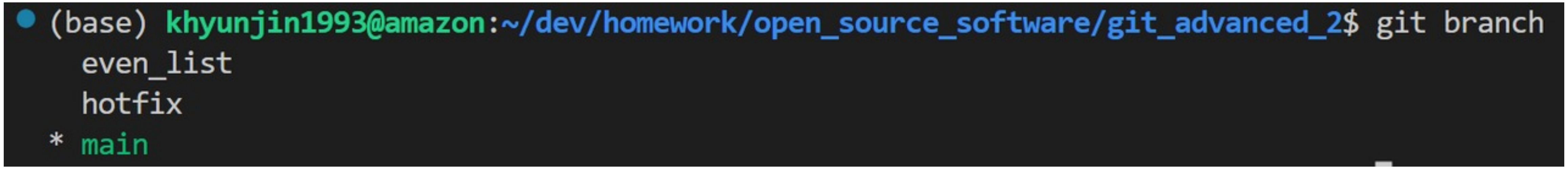
Basic branching and merging

- Merge branch **“main”** with **“hotfix”** (using **“git merge <branch_name>”**)
- **“fast-forward”**: simply move the pointer forward that is directly ahead of current branch (i.e., position of HEAD)



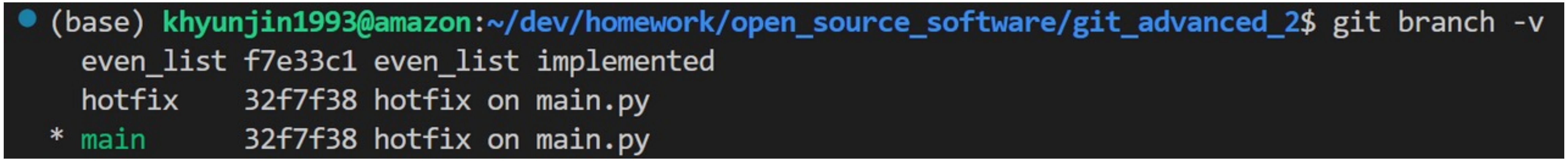
Branch management

- We know that there are a total of three branches using the **"git branch"** command



```
● (base) khyunjin1993@amazon:~/dev/homework/open_source_software/git_advanced_2$ git branch
    even_list
    hotfix
  * main
```

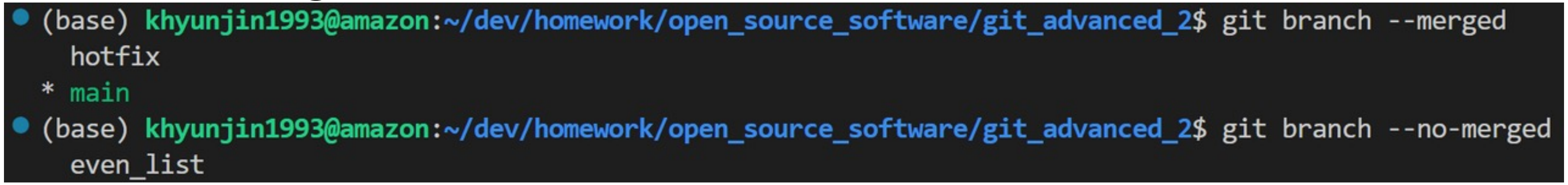
- We can see the latest commit on each branch by using **"git branch -v"**



```
● (base) khyunjin1993@amazon:~/dev/homework/open_source_software/git_advanced_2$ git branch -v
    even_list f7e33c1 even_list implemented
    hotfix    32f7f38 hotfix on main.py
  * main      32f7f38 hotfix on main.py
```

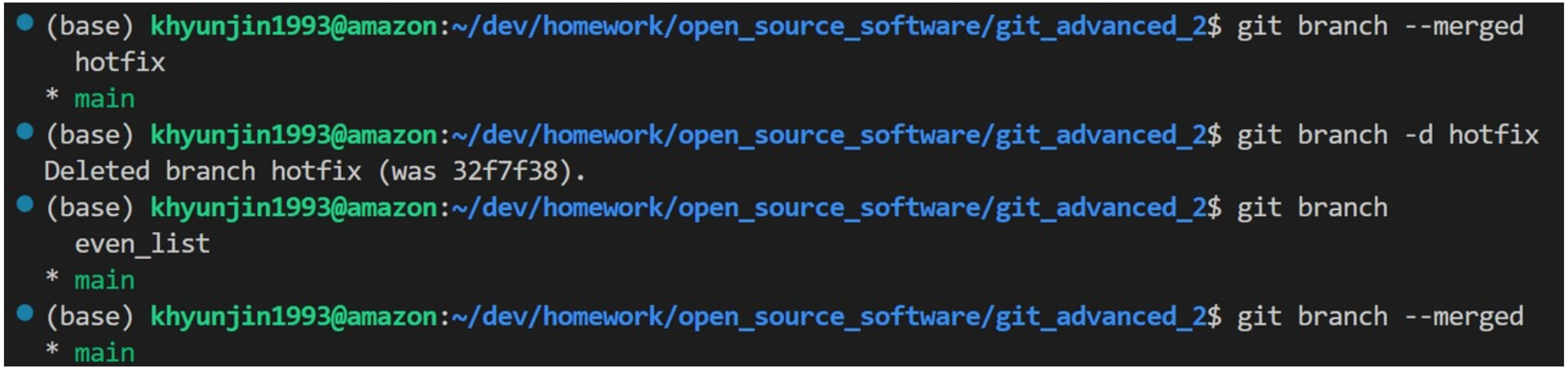
Branch management

- The **“--merged”** and **“--no-merged”** option can filter the list of branches whether merged of not



```
● (base) khyunjin1993@amazon:~/dev/homework/open_source_software/git_advanced_2$ git branch --merged
    hotfix
  * main
● (base) khyunjin1993@amazon:~/dev/homework/open_source_software/git_advanced_2$ git branch --no-merged
    even_list
```

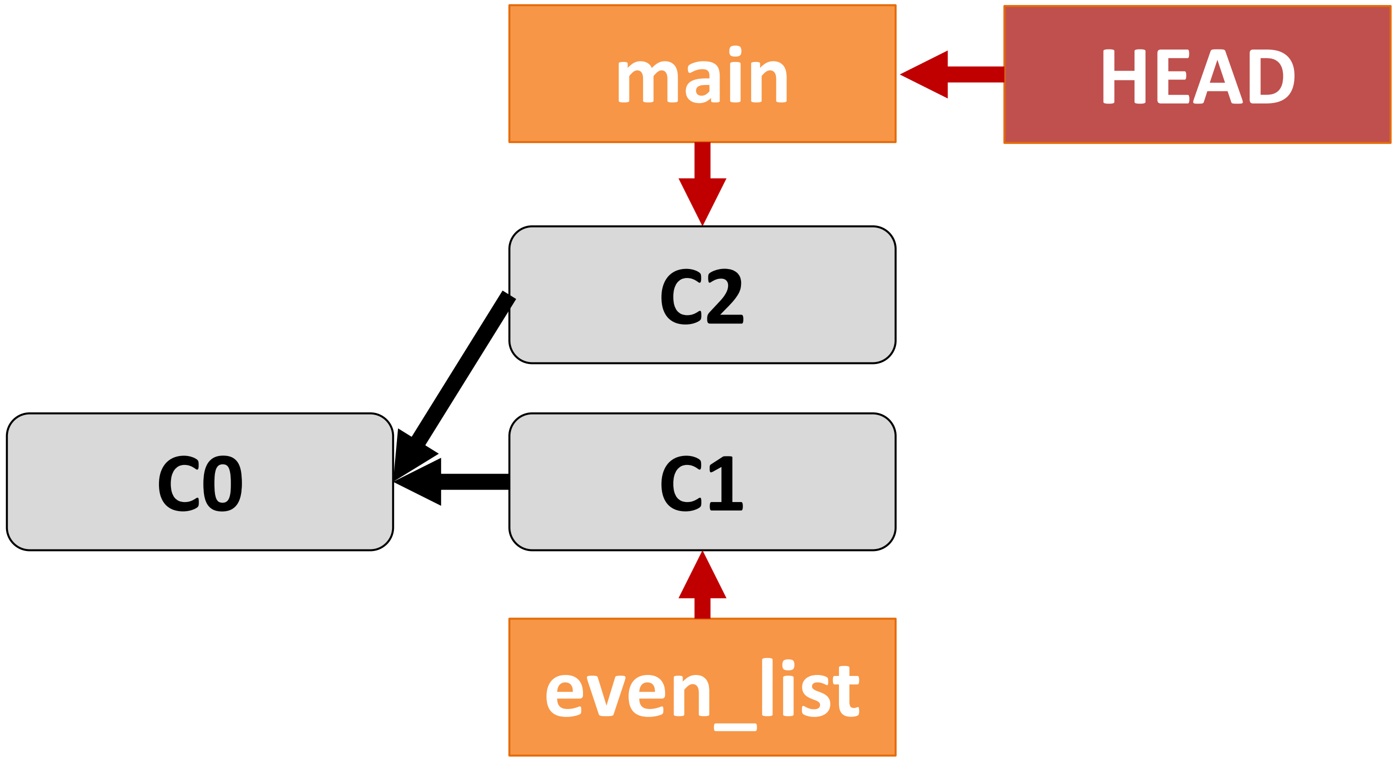
- We can delete merged branches by **“git branch -d <branch_name>”**



```
● (base) khyunjin1993@amazon:~/dev/homework/open_source_software/git_advanced_2$ git branch --merged
    hotfix
  * main
● (base) khyunjin1993@amazon:~/dev/homework/open_source_software/git_advanced_2$ git branch -d hotfix
  Deleted branch hotfix (was 32f7f38).
● (base) khyunjin1993@amazon:~/dev/homework/open_source_software/git_advanced_2$ git branch
    even_list
  * main
● (base) khyunjin1993@amazon:~/dev/homework/open_source_software/git_advanced_2$ git branch --merged
  * main
```

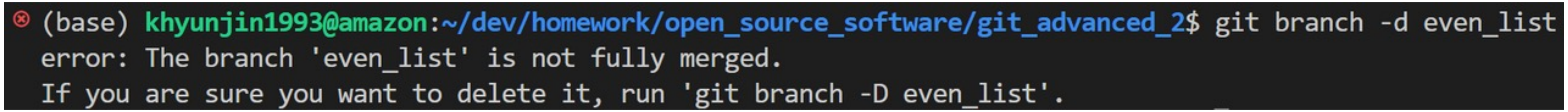
Branch management

- We can delete merged branches by **“git branch -d <branch_name>”**
- Resulting in following commit graph



Branch management

- Deleting not merged branch will cause the error



```
(base) khyunjin1993@amazon:~/dev/homework/open_source_software/git_advanced_2$ git branch -d even_list
error: The branch 'even_list' is not fully merged.
If you are sure you want to delete it, run 'git branch -D even_list'.
```

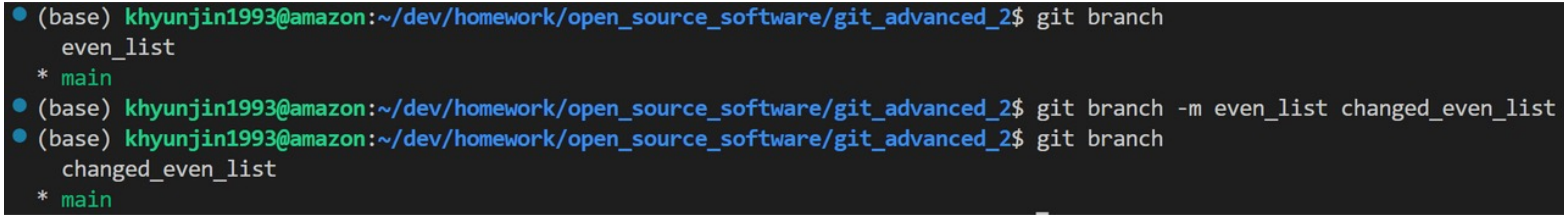
- If you want to really delete the branch, run

  “git branch –D <branch_name>” (Do not run in class today)

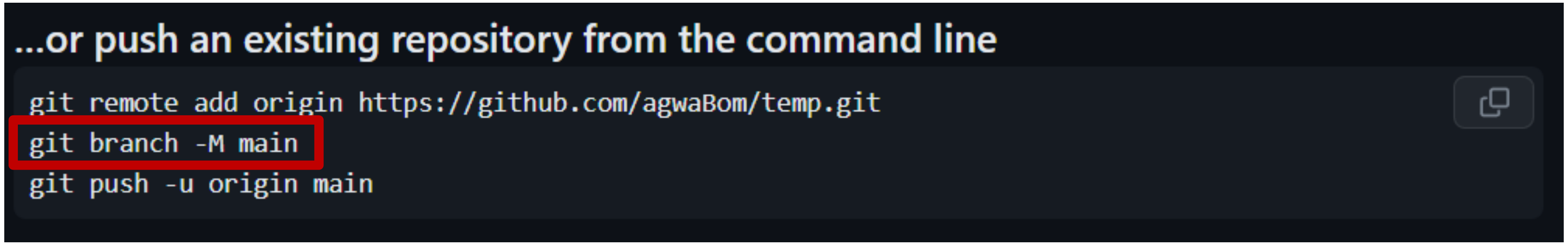
Branch management

We can also change the branch name by using

- **“git branch --move <old_branch_name> <new_branch_name>”**
- **“git branch -m <old_branch_name> <new_branch_name>”**



```
(base) khyunjin1993@amazon:~/dev/homework/open_source_software/git_advanced_2$ git branch
  even_list
* main
(base) khyunjin1993@amazon:~/dev/homework/open_source_software/git_advanced_2$ git branch -m even_list changed_even_list
(base) khyunjin1993@amazon:~/dev/homework/open_source_software/git_advanced_2$ git branch
  changed_even_list
* main
```

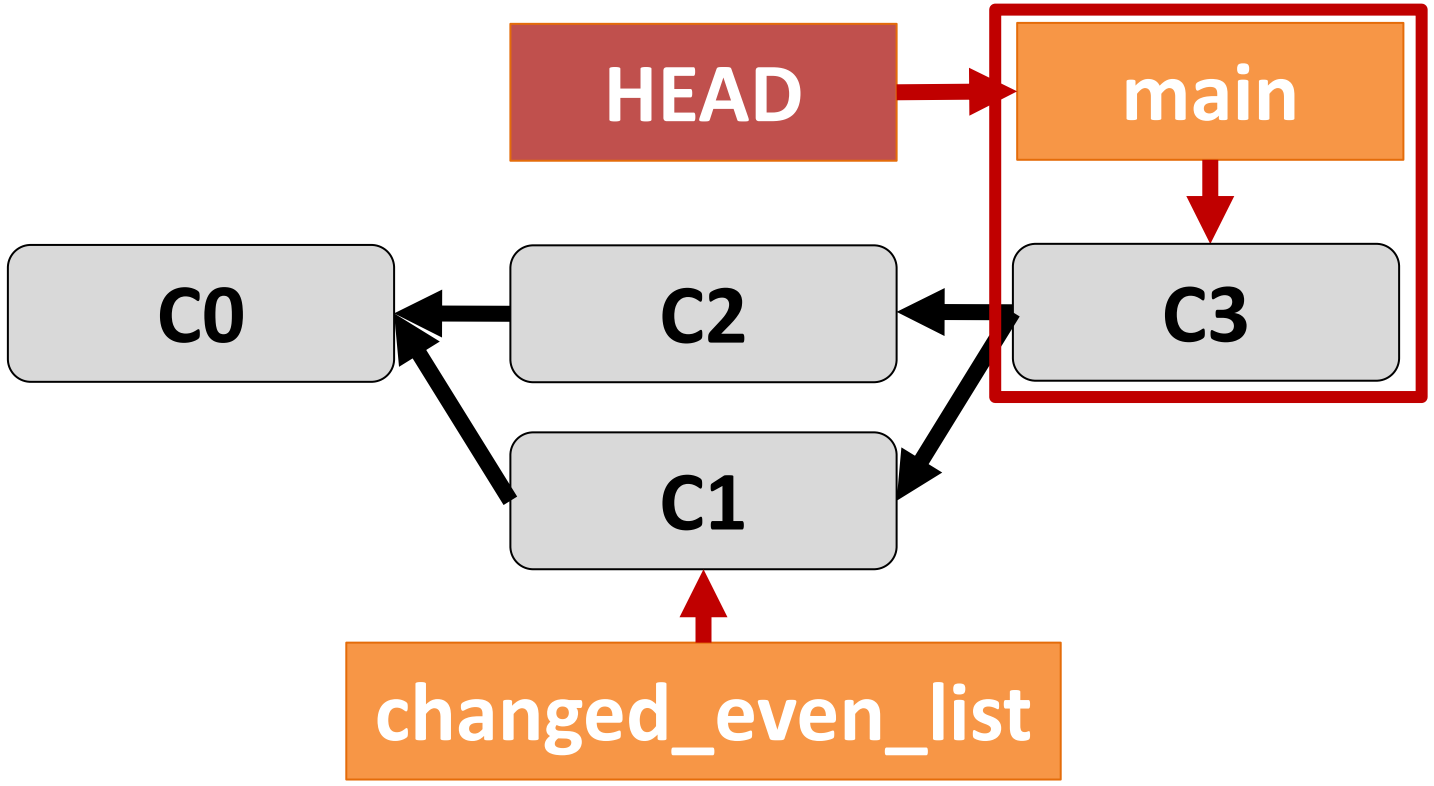


...or push an existing repository from the command line

```
git remote add origin https://github.com/agwaBom/temp.git
git branch -M main
git push -u origin main
```

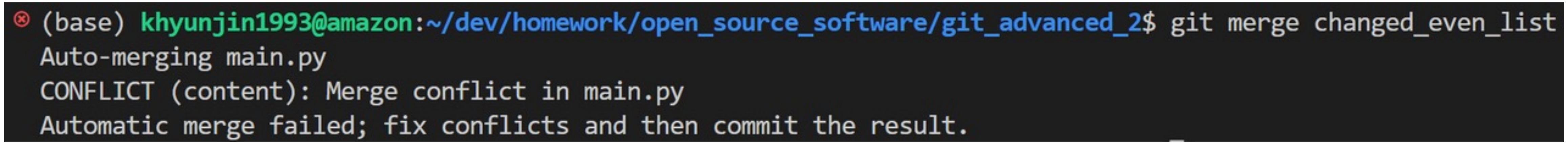
Basic merging

Suppose you've decided that your work (**"even_list"**) is complete and ready to merge the **"even_list"** branch into your **"main"** branch



Basic merging

If you try to merge **“main”** with **“changed_even_list”** branch, merge conflict will occur

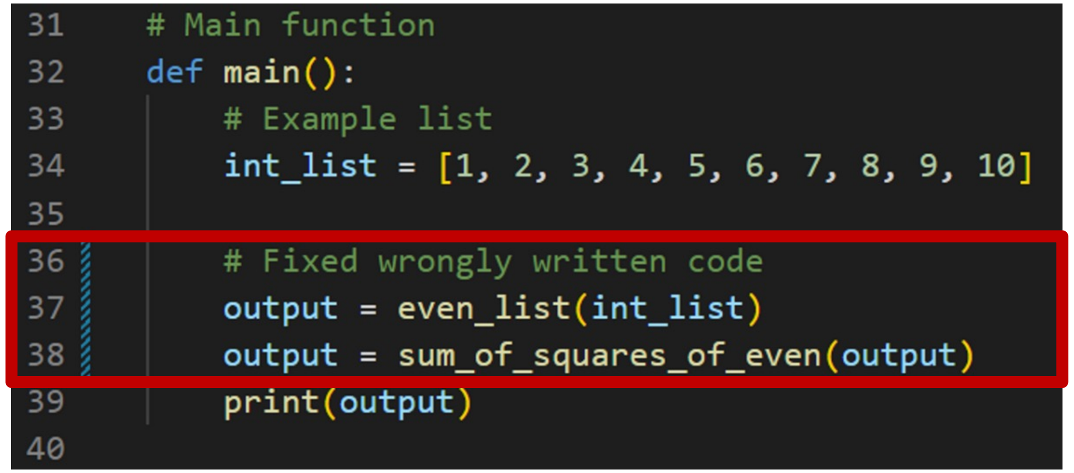


```
(base) khyunjin1993@amazon:~/dev/homework/open_source_software/git_advanced_2$ git merge changed_even_list
Auto-merging main.py
CONFLICT (content): Merge conflict in main.py
Automatic merge failed; fix conflicts and then commit the result.
```

Basic merging

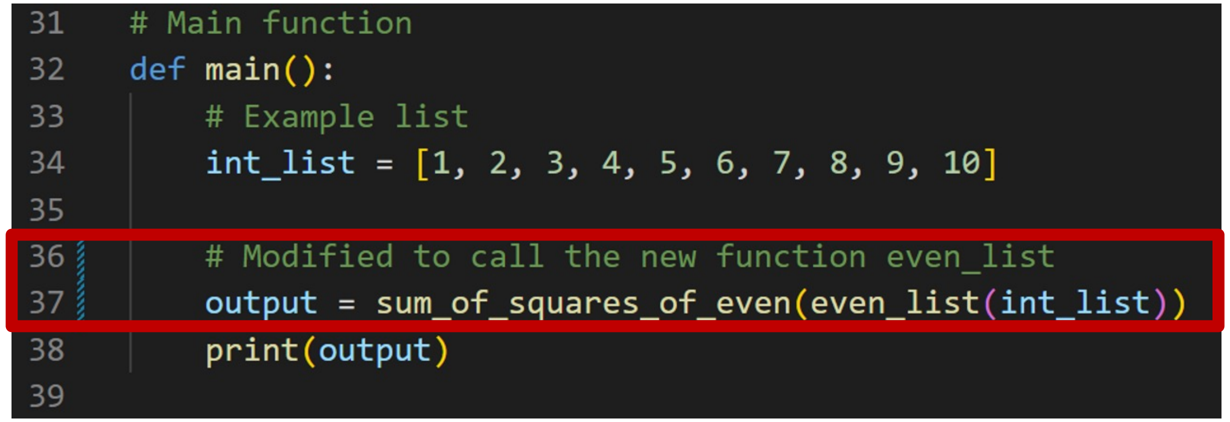
- If you try to merge **"main"** with **"changed_even_list"** branch, a merge conflict will occur
- Because **you changed the same part of the file differently in the two branches** you're merging, git won't be able to merge them

"main" branch



```python
# Main function
def main():
    # Example list
    int_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

    # Fixed wrongly written code
    output = even_list(int_list)
    output = sum_of_squares_of_even(output)
    print(output)

```

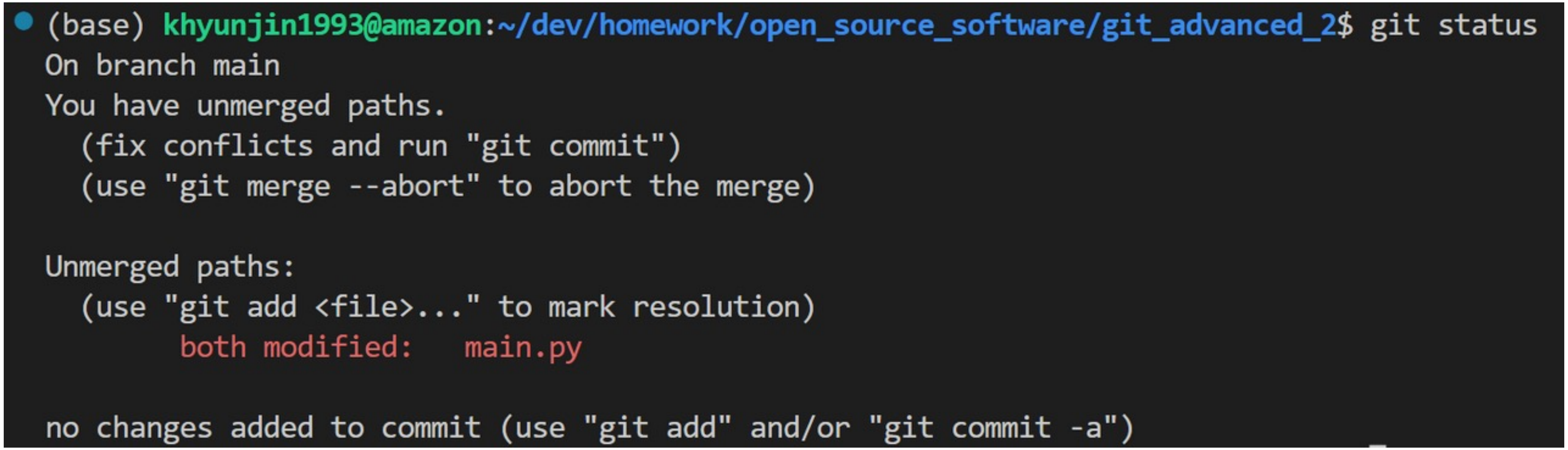
"changed_even_list" branch



```python
# Main function
def main():
    # Example list
    int_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

    # Modified to call the new function even_list
    output = sum_of_squares_of_even(even_list(int_list))
    print(output)

```

Basic merging

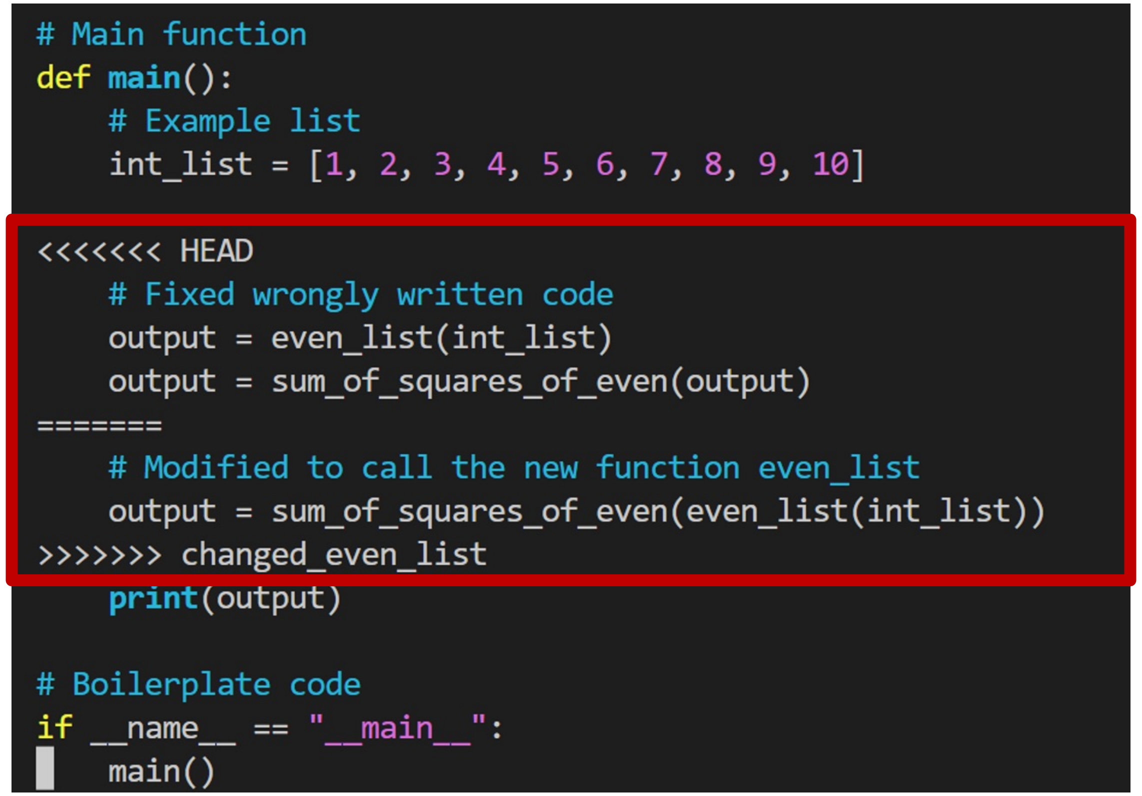
After **“git status”** you will see that **“main.py”** is both modified



```
● (base) khyunjin1993@amazon:~/dev/homework/open_source_software/git_advanced_2$ git status
  On branch main
  You have unmerged paths.
    (fix conflicts and run "git commit")
    (use "git merge --abort" to abort the merge)

  Unmerged paths:
    (use "git add <file>..." to mark resolution)
          both modified:   main.py

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

Basic merging

- You can open them manually with your favorite editor **“main.py”** and resolve those conflicts
- Your file contains a section that looks something like this:



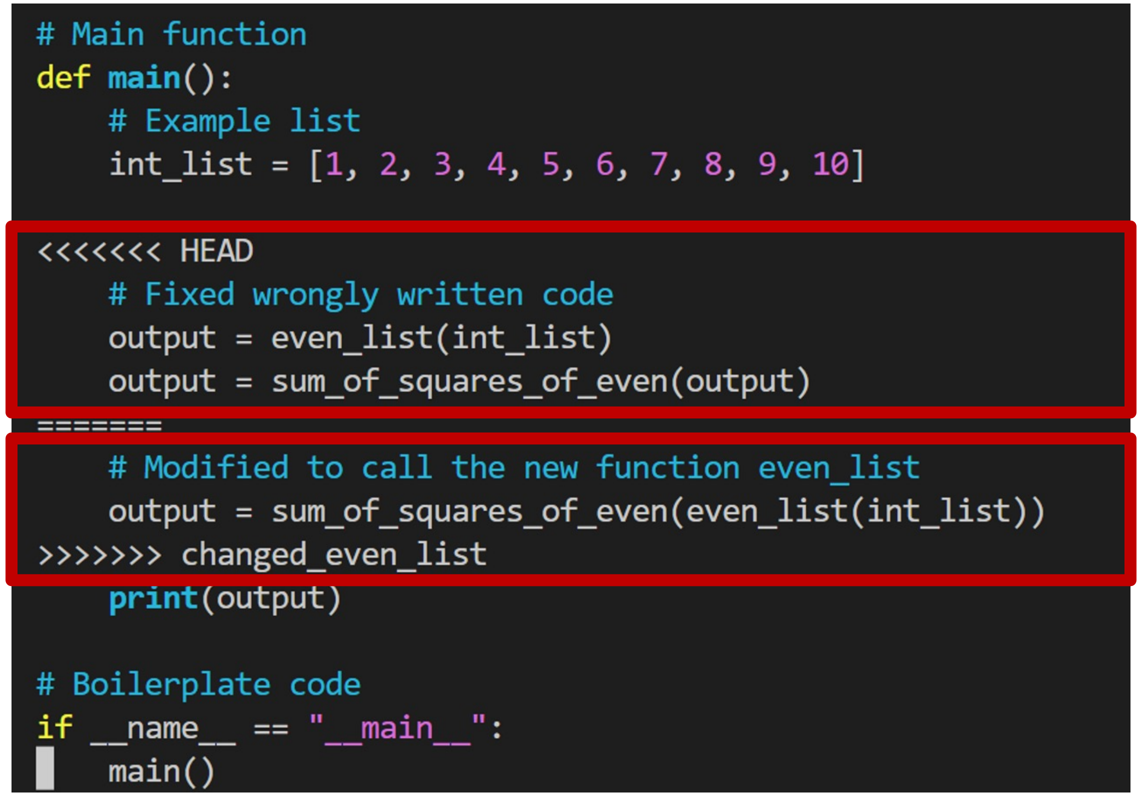
```
# Main function
def main():
    # Example list
    int_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

<<<<<<< HEAD
    # Fixed wrongly written code
    output = even_list(int_list)
    output = sum_of_squares_of_even(output)
=======
    # Modified to call the new function even_list
    output = sum_of_squares_of_even(even_list(int_list))
>>>>>>> changed_even_list
    print(output)

# Boilerplate code
if __name__ == "__main__":
    main()
```

Basic merging

HEAD is in the **“main”** branch, Upper Redbox will be the main branch’s modification and the bottom Redbox will be the **“changed_even_list”** branch

We can either choose a code between

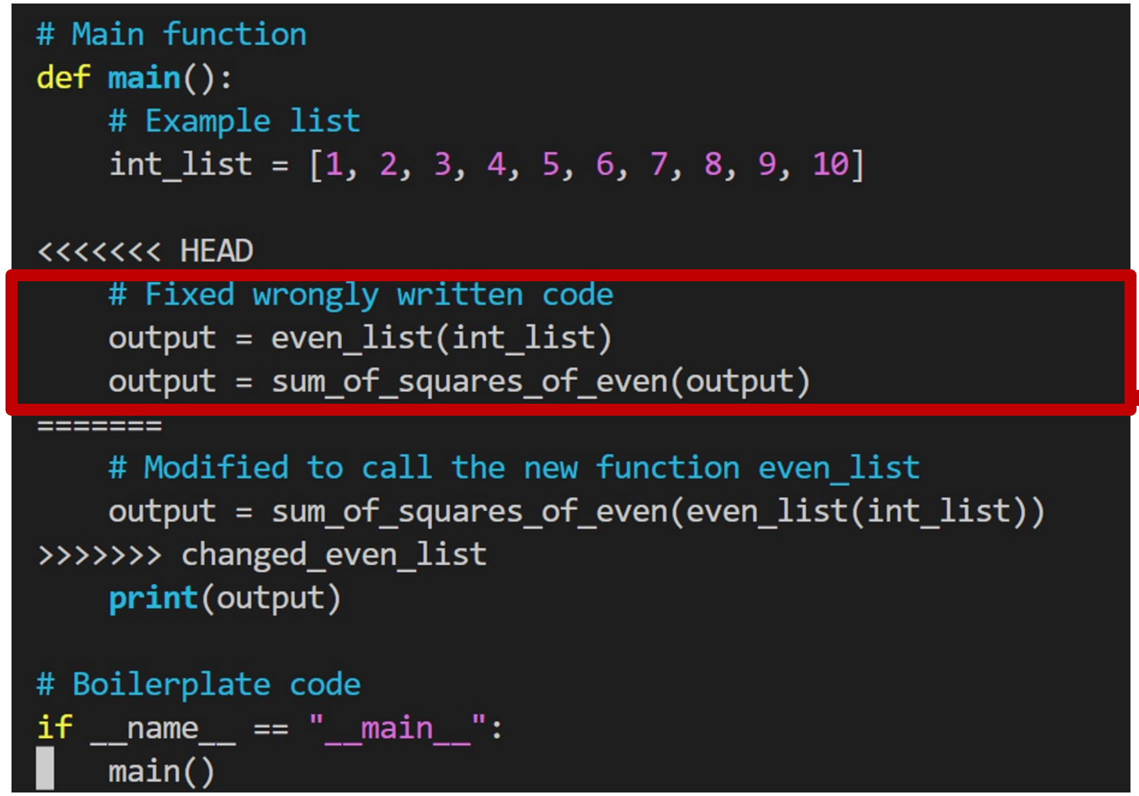
- <<<<<<< and =======
- ====== and >>>>>>>



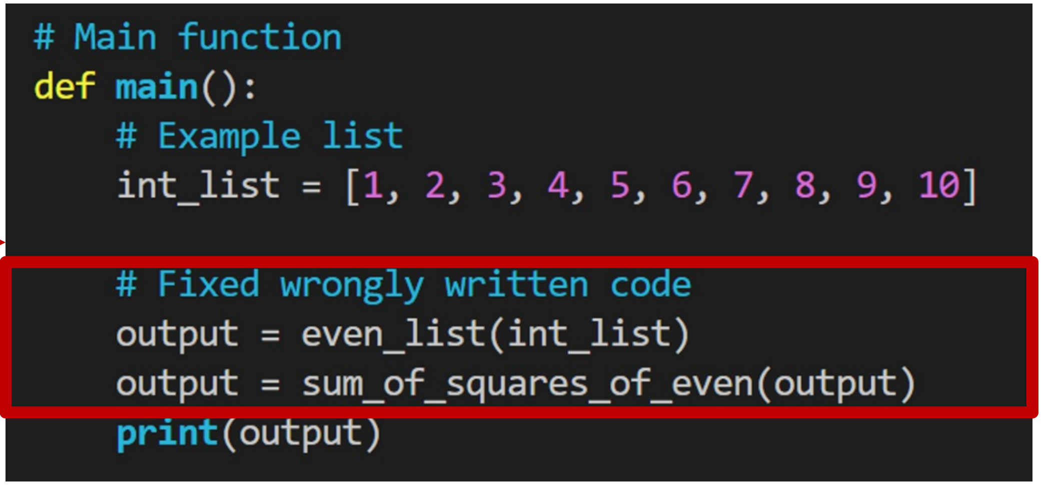
```
# Main function
def main():
    # Example list
    int_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

<<<<<<< HEAD
    # Fixed wrongly written code
    output = even_list(int_list)
    output = sum_of_squares_of_even(output)
=======
    # Modified to call the new function even_list
    output = sum_of_squares_of_even(even_list(int_list))
>>>>>>> changed_even_list
    print(output)

# Boilerplate code
if __name__ == "__main__":
    main()
```

Basic merging

In my case, I choose the HEAD section (i.e., **“main”** branch)



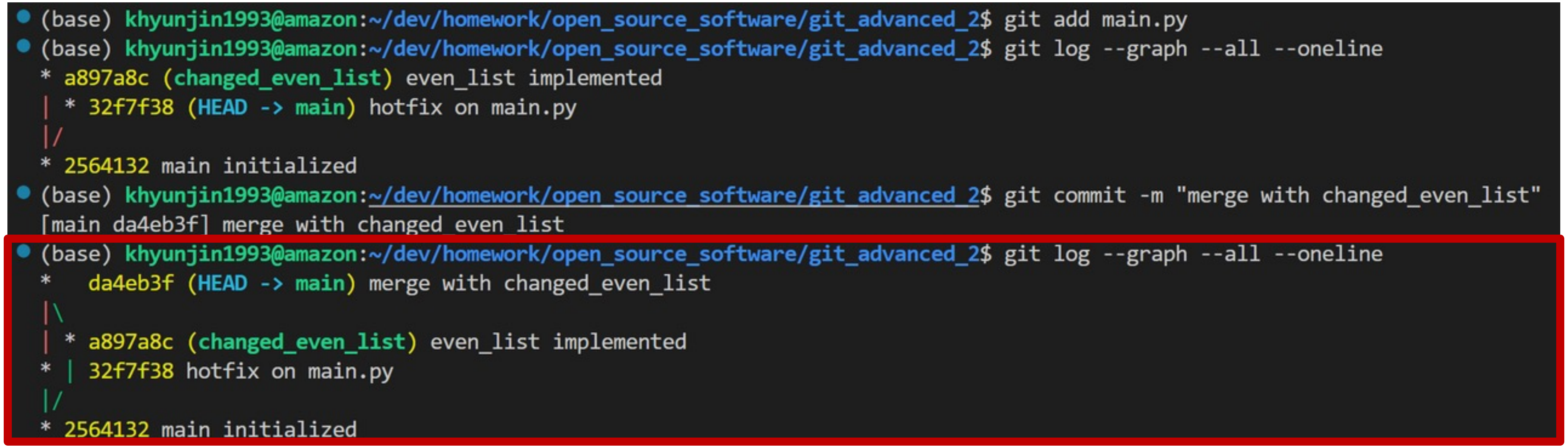
```
# Main function
def main():
    # Example list
    int_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

<<<<<<< HEAD
    # Fixed wrongly written code
    output = even_list(int_list)
    output = sum_of_squares_of_even(output)
=======
    # Modified to call the new function even_list
    output = sum_of_squares_of_even(even_list(int_list))
>>>>>>> changed_even_list
    print(output)

# Boilerplate code
if __name__ == "__main__":
    main()
```



```
# Main function
def main():
    # Example list
    int_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

    # Fixed wrongly written code
    output = even_list(int_list)
    output = sum_of_squares_of_even(output)
    print(output)
```

Basic merging

After modification, commit, and merge



```
(base) khyunjin1993@amazon:~/dev/homework/open_source_software/git_advanced_2$ git add main.py
(base) khyunjin1993@amazon:~/dev/homework/open_source_software/git_advanced_2$ git log --graph --all --oneline
* a897a8c (changed_even_list) even_list implemented
| * 32f7f38 (HEAD -> main) hotfix on main.py
|/
* 2564132 main initialized
(base) khyunjin1993@amazon:~/dev/homework/open_source_software/git_advanced_2$ git commit -m "merge with changed_even_list"
[main da4eb3f] merge with changed_even_list
(base) khyunjin1993@amazon:~/dev/homework/open_source_software/git_advanced_2$ git log --graph --all --oneline
*   da4eb3f (HEAD -> main) merge with changed_even_list
|\
| * a897a8c (changed_even_list) even_list implemented
* | 32f7f38 hotfix on main.py
|/
* 2564132 main initialized
```

You can see that it is now successfully merged :)

Branching workflow

Two approaches in the workflow

- Long-Running Branches
- Topic Branches

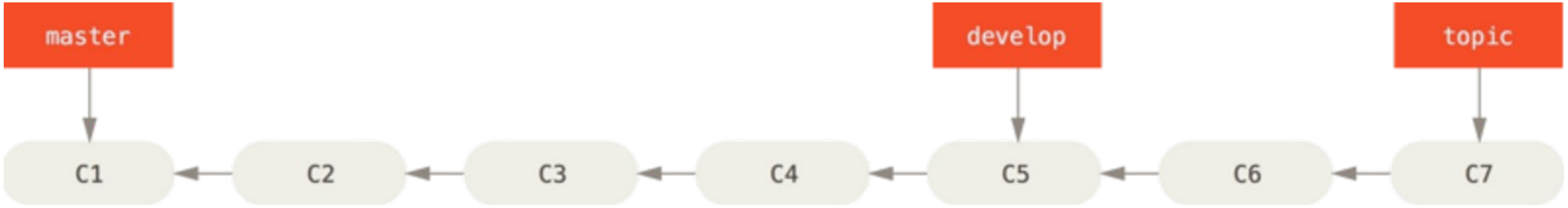
Branching workflow (Long-running branches)

Many git developers have a workflow that embraces this approach

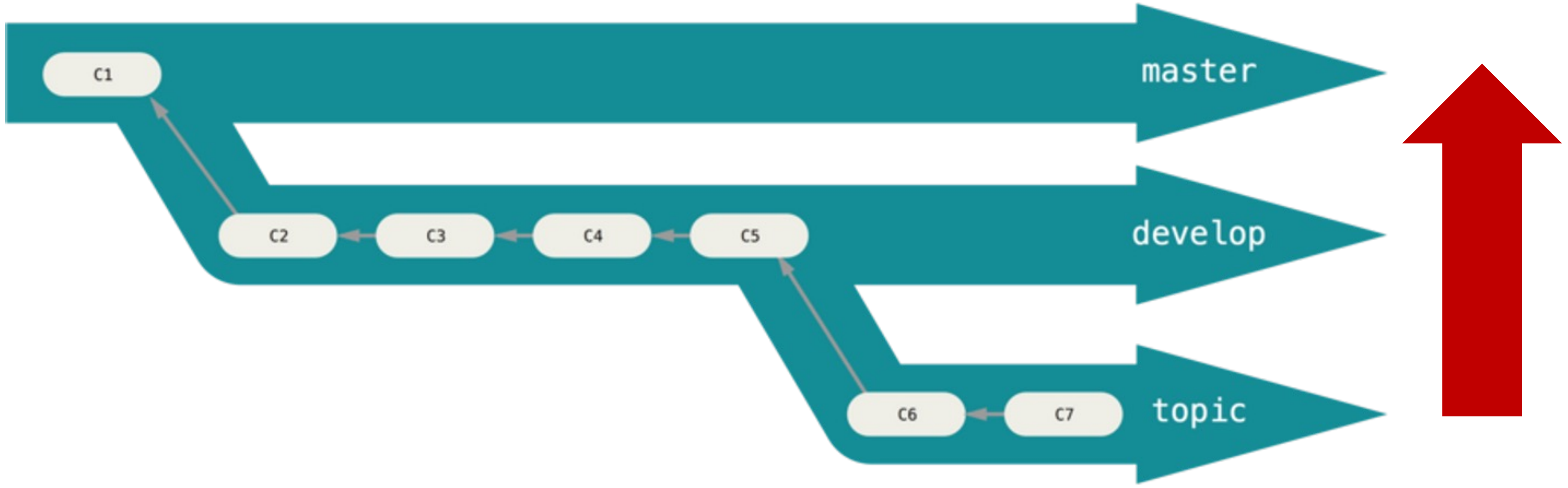
- Code that is entirely stable in their **“main”** branch (only code that has been or will be released)
- They have another parallel branch named **“develop”** that they work from or use to **test stability**
- **“topic”** branches for a new feature

Branching workflow (Long-running branches)

A linear view of long-running branching (progressive-stability branching)



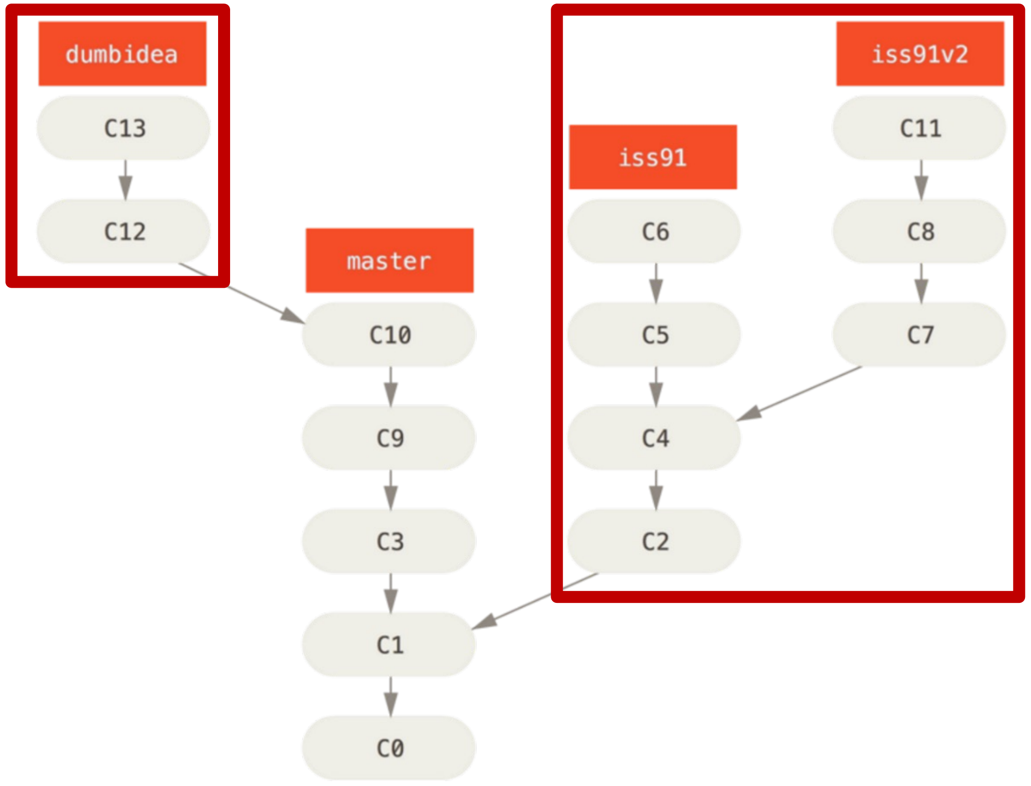
Generally easier to think about them as work **"silos"** (i.e., work individually)



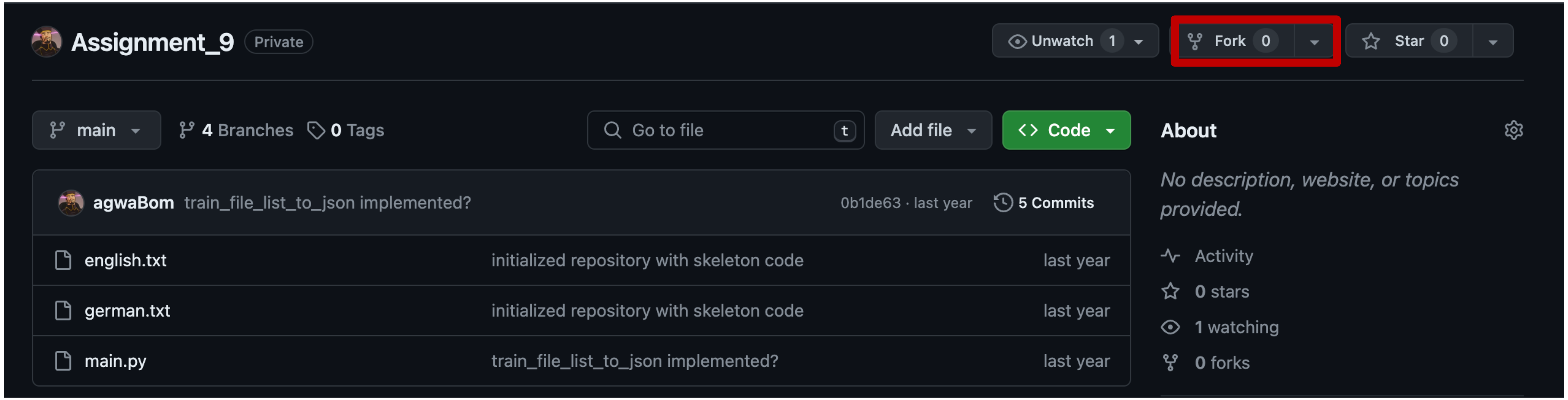
Higher stability as it is closer to main branch

Branching workflow (Topic branches)

- A topic branch is a **short-lived branch that you create and use for a single particular feature**
- This technique allows you to context-switch quickly and completely



Fork

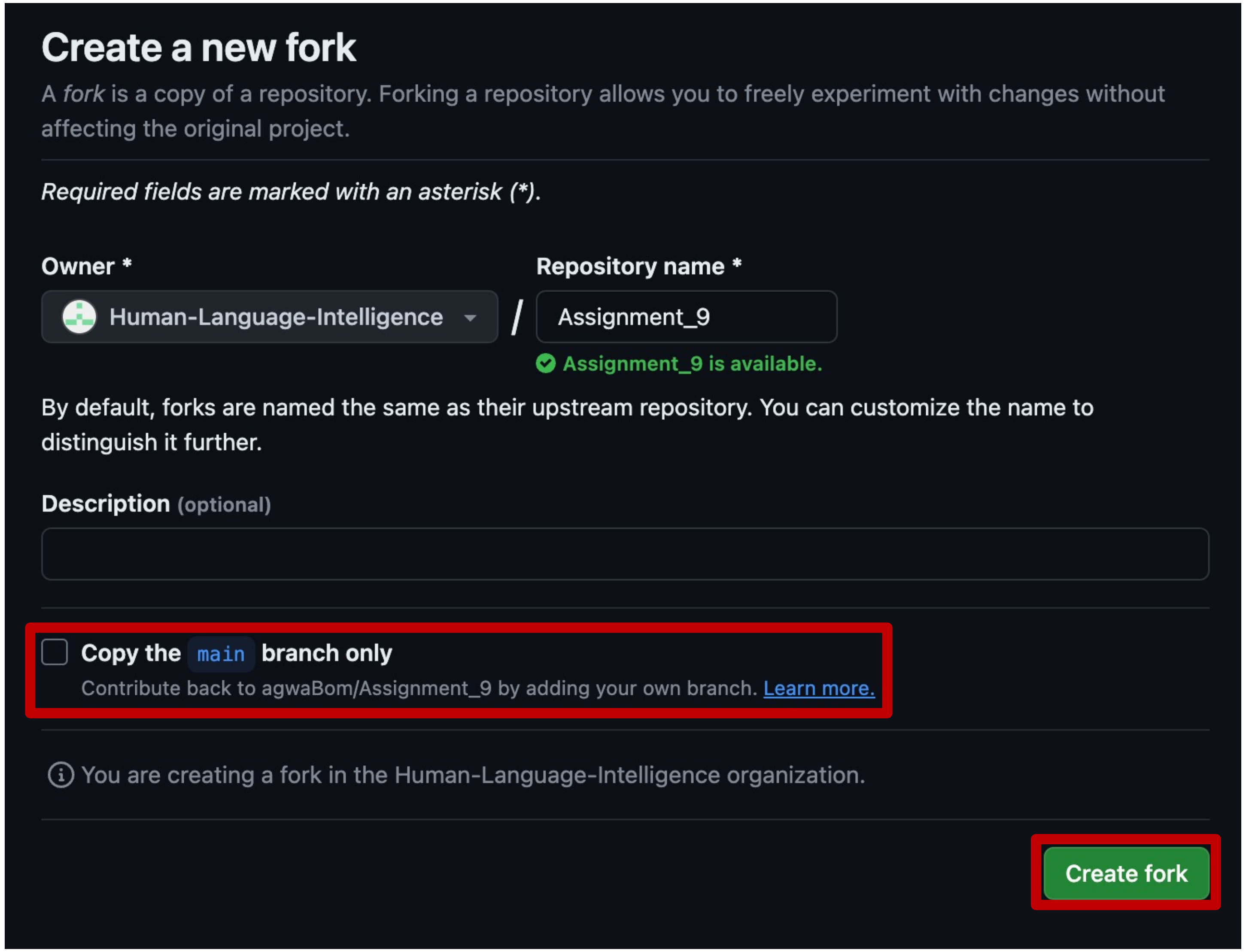


No description, website, or topics provided.

agwaBom train_file_list_to_json implemented? 0b1de63 · last year 5 Commits

english.txt	initialized repository with skeleton code	last year
german.txt	initialized repository with skeleton code	last year
main.py	train_file_list_to_json implemented?	last year

Fork



Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.

Required fields are marked with an asterisk (*).

Owner * : Human-Language-Intelligence / Repository name * : Assignment_9

Assignment_9 is available.

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional)

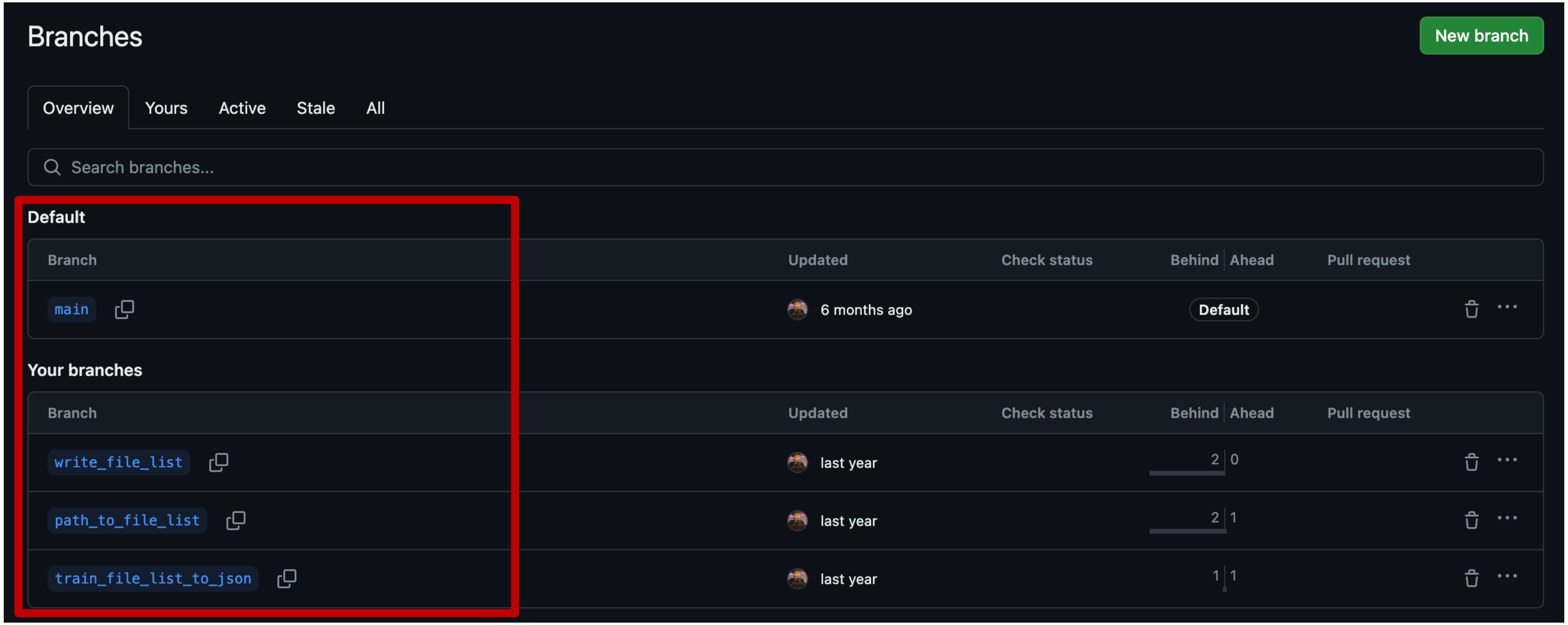
[ ] Copy the `main` branch only

Contribute back to agwaBom/Assignment_9 by adding your own branch. Learn more.

You are creating a fork in the Human-Language-Intelligence organization.

Uncheck!

Fork



Branches

New branch

Overview Yours Active Stale All

Search branches...

Default

Branch	Updated	Check status	Behind | Ahead	Pull request
main	6 months ago		Default	

Your branches

Branch	Updated	Check status	Behind | Ahead	Pull request
write_file_list	last year		2 | 0	
path_to_file_list	last year		2 | 1	
train_file_list_to_json	last year		1 | 1	