

git

- git仓库master分支
 - `git@github.com:QinHsiu/imporve_git_practice.git`
- ToDo
 - 克隆仓库到本地，并进入到目录中

```
1 git clone git@github.com:QinHsiu/imporve_git_practice.git
2 cd <repository_directory>
```

- 创建branch分支

```
1 git checkout -b <your_branch_name>
```

- 修改`load_dataset(self, id)`函数，添加一个参数，例如添加下列代码

```
1 def load_dataset(self, id, add=None):
2     print(add)
```

- 拉取最新分支

```
1 git fetch origin
2 git checkout main # 切换到主分支（如果你要合并的是主分支）
3 git pull origin main # 拉取最新的更改
```

- 合并最新主分支到自己的分支

```
1 git checkout <your_branch_name> # 返回到你的分支
2 git merge main
```

- 会产生冲突，并且标记他们
- ？这里可以使用rebase？
 - 基础重置
 - 这个命令会把当前分支的所有提交移到 `<branch>` 分支的最新提交之后。这是最常见的用法，特别是在处理特性分支时。

```
1 git rebase <branch>
```

- 交互式重置
 - 使用 `-i` 或 `--interactive` 可以进行交互式重置，这样你可以在编辑器中选择、修改、合并或删除提交。

```
1 git rebase -i <commit>
```

- 案例：这会打开一个文本编辑器，让你对最近的三个提交进行操作。

```
1 git rebase -i HEAD~ 3
```

- 当需要依据主分支更新自己的分支

```
1 git checkout feature-branch
2 git rebase main
```

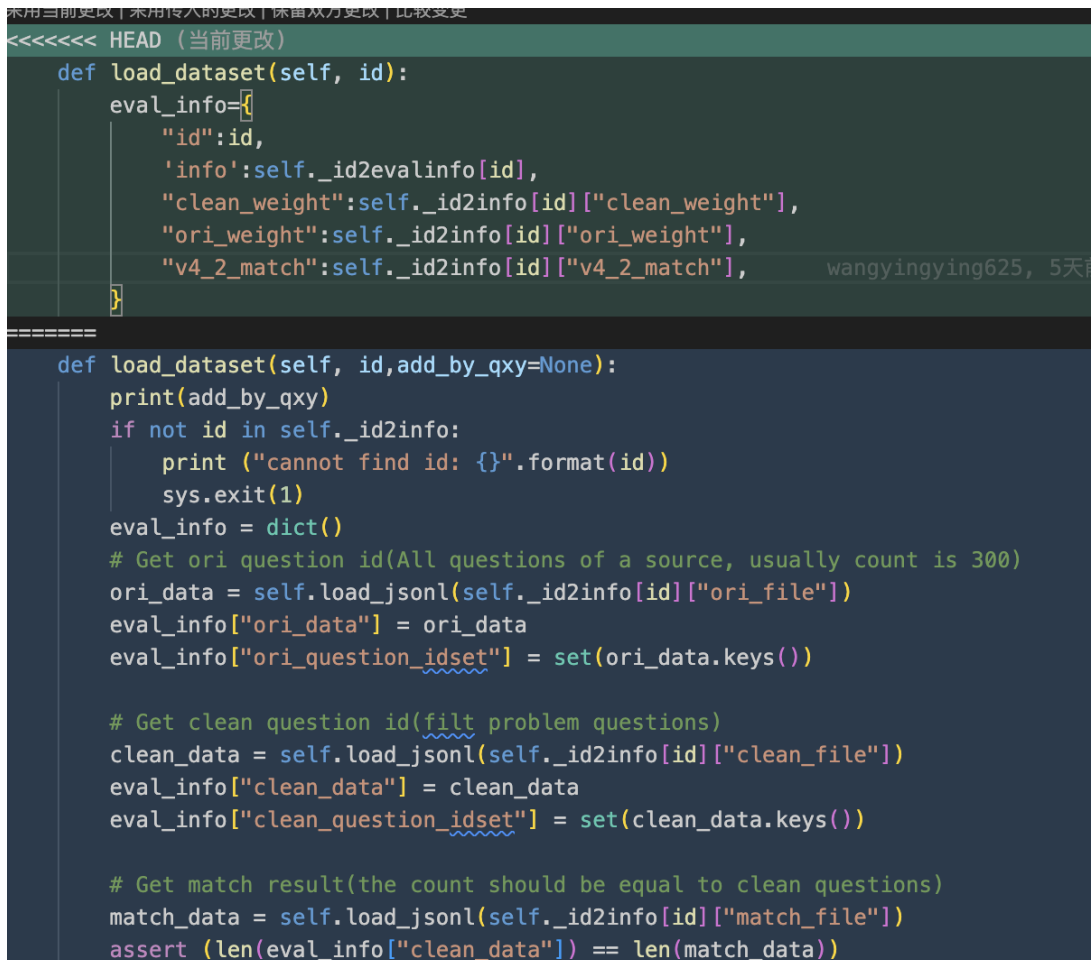
□ 出现合并的错误

```
1 CONFLICT (content): Merge conflict in eval.py
2 error: could not apply d1da36b... modify load_dataset function to add ad
3 Resolve all conflicts manually, mark them as resolved with
4 "git add/rm <conflicted_files>", then run "git rebase --continue".
5 You can instead skip this commit: run "git rebase --skip".
6 To abort and get back to the state before "git rebase", run "git rebase
7 Could not apply d1da36b... modify load_dataset function to add add_by_qx
```

■ 使用git status检查错误

```
1 interactive rebase in progress; onto c6dafc1
2 Last command done (1 command done):
3   pick d1da36b modify load_dataset function to add add_by_qxy
4 No commands remaining.
5 You are currently rebasing branch 'qxy' on 'c6dafc1'.
6   (fix conflicts and then run "git rebase --continue")
7   (use "git rebase --skip" to skip this patch)
8   (use "git rebase --abort" to check out the original branch)
9 Unmerged paths:
10   (use "git restore --staged <file>..." to unstage)
11   (use "git add <file>..." to mark resolution)
12       both modified:   eval.py
13 no changes added to commit (use "git add" and/or "git commit -a")
```

■ 打开文件找到冲突的部分



```
采用当前更改 | 采用传入的更改 | 保留双方更改 | 比较变更
<<<<<<< HEAD (当前更改)
def load_dataset(self, id):
    eval_info={
        "id":id,
        'info':self._id2evalinfo[id],
        "clean_weight":self._id2info[id]["clean_weight"],
        "ori_weight":self._id2info[id]["ori_weight"],
        "v4_2_match":self._id2info[id]["v4_2_match"],
    }

=====
def load_dataset(self, id, add_by_qxy=None):
    print(add_by_qxy)
    if not id in self._id2info:
        print ("cannot find id: {}".format(id))
        sys.exit(1)
    eval_info = dict()
    # Get ori question id(All questions of a source, usually count is 300)
    ori_data = self.load_jsonl(self._id2info[id]["ori_file"])
    eval_info["ori_data"] = ori_data
    eval_info["ori_question_idset"] = set(ori_data.keys())

    # Get clean question id(filt problem questions)
    clean_data = self.load_jsonl(self._id2info[id]["clean_file"])
    eval_info["clean_data"] = clean_data
    eval_info["clean_question_idset"] = set(clean_data.keys())

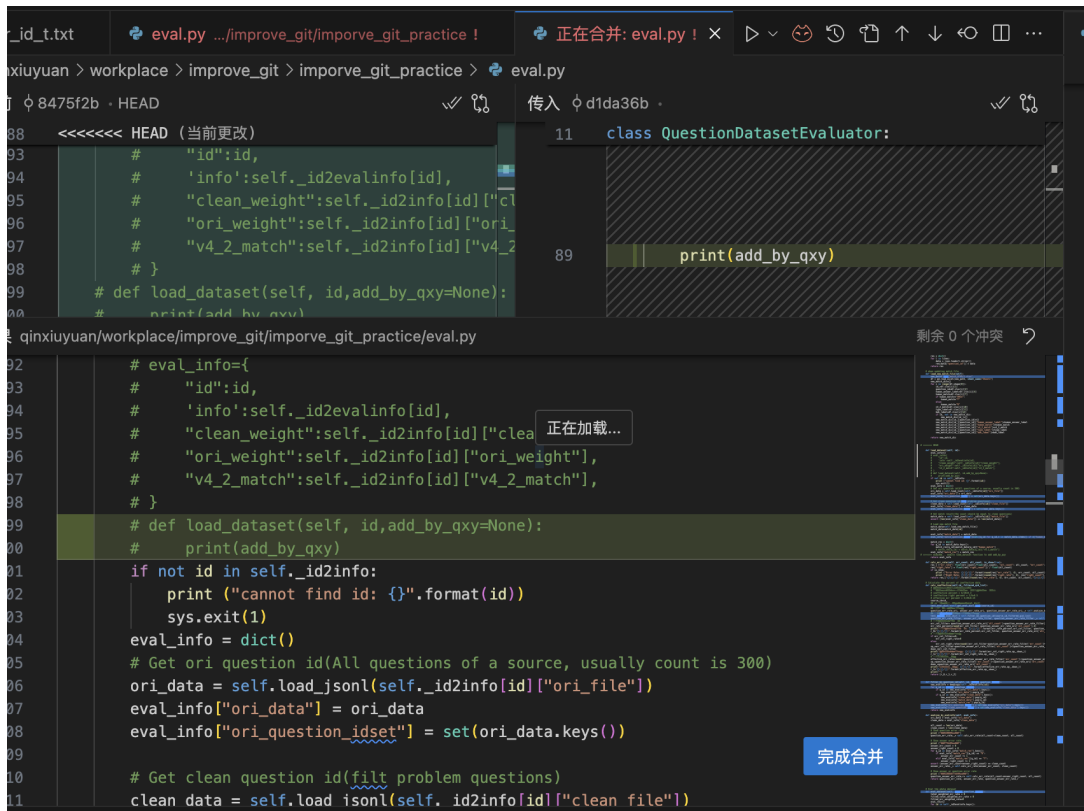
    # Get match result(the count should be equal to clean questions)
    match_data = self.load_jsonl(self._id2info[id]["match_file"])
    assert (len(eval_info["clean_data"]) == len(match_data))
```

■ 依据分支A修改分支B

```
1 git checkout featureB
2 git rebase featureA
```

- 处理冲突

- 在重置过程中，如果有冲突，Git 会暂停并让你手动解决冲突。



- 解决完冲突后，你需要执行以下命令继续重置过程：

```
1 git add <resolved-files>
2 git rebase --continue
```

- 放弃重置

```
1 git rebase --abort
```

- 修改提交信息

```
1 git commit --amend
2 git rebase --continue
```

- 一直添加文件，直到没有冲突为止

```
1 # 编辑冲突文件，解决冲突
2 git add <conflicted_file>
```

- 完成合并并提交

```
1 git commit -m "Resolved merge conflicts"
```

- 推送自己的分支到远程仓库

```
1 git push origin <your_branch_name>
```

- 备注

- 避免在公共分支上重置：因为重置会改变提交历史，所以最好只在私有分支上使用，以免影响其他开发者。

- **备份重要数据：**在执行复杂的重置操作前，建议先备份代码库，防止意外丢失数据