Git_exercises from Yingji Zheng with student iD s2512460

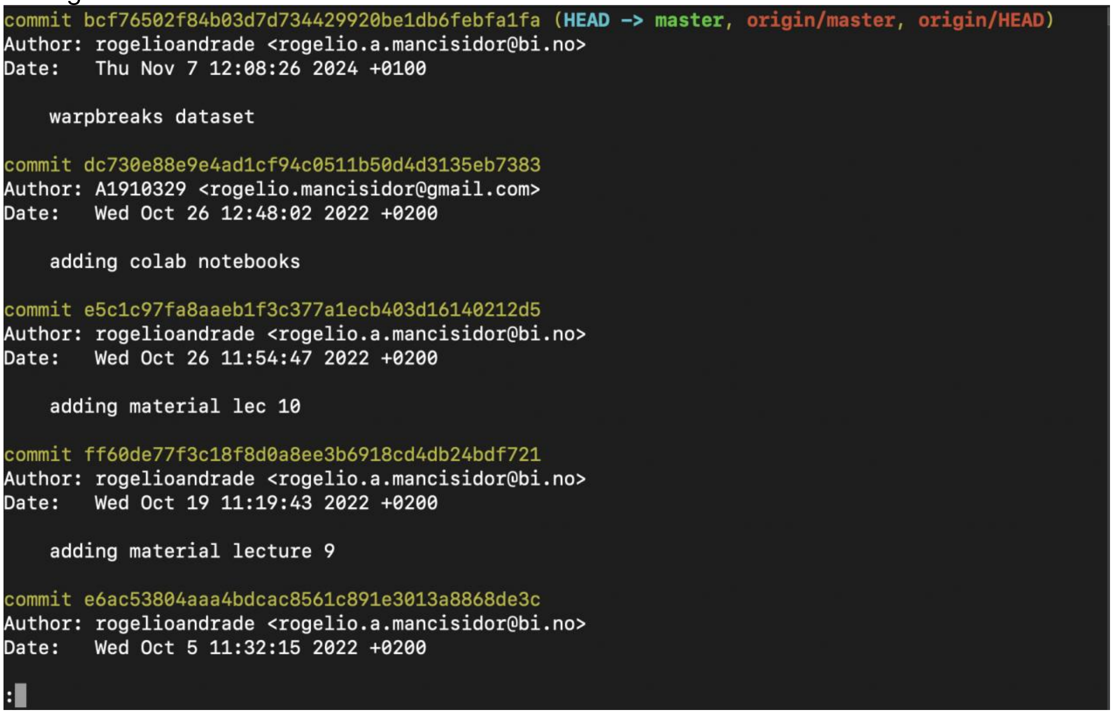
1.

a)

Git clone https://github.com/BI-DS/GRA-4152.git

cd GRA-4152

Git log

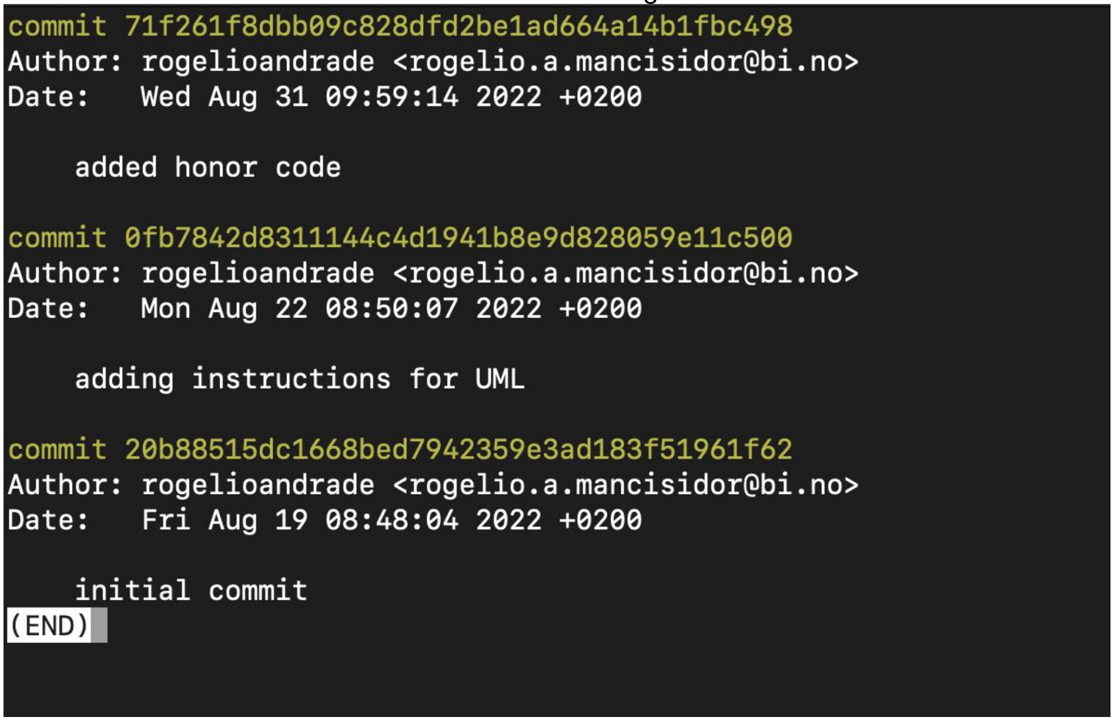


```
commit bcf76502f84b03d7d734429920be1db6febfa1fa (HEAD -> master, origin/master, origin/HEAD)
Author: rogelioandrade <rogelio.a.mancisidor@bi.no>
Date:   Thu Nov 7 12:08:26 2024 +0100

    warpbreaks dataset

commit dc730e88e9e4ad1cf94c0511b50d4d3135eb7383
Author: A1910329 <rogelio.mancisidor@gmail.com>
Date:   Wed Oct 26 12:48:02 2022 +0200

    adding colab notebooks

commit e5c1c97fa8aaeb1f3c377a1ecb403d16140212d5
Author: rogelioandrade <rogelio.a.mancisidor@bi.no>
Date:   Wed Oct 26 11:54:47 2022 +0200

    adding material lec 10

commit ff60de77f3c18f8d0a8ee3b6918cd4db24bdf721
Author: rogelioandrade <rogelio.a.mancisidor@bi.no>
Date:   Wed Oct 19 11:19:43 2022 +0200

    adding material lecture 9

commit e6ac53804aaa4bdcac8561c891e3013a8868de3c
Author: rogelioandrade <rogelio.a.mancisidor@bi.no>
Date:   Wed Oct 5 11:32:15 2022 +0200

:
```

b)

git log -- README.md

The last time README.md was modified was Wed Aug 31 09:59:14 2022



```
commit 71f261f8dbb09c828dfd2be1ad664a14b1fbc498
Author: rogelioandrade <rogelio.a.mancisidor@bi.no>
Date:   Wed Aug 31 09:59:14 2022 +0200

    added honor code

commit 0fb7842d8311144c4d1941b8e9d828059e11c500
Author: rogelioandrade <rogelio.a.mancisidor@bi.no>
Date:   Mon Aug 22 08:50:07 2022 +0200

    adding instructions for UML

commit 20b88515dc1668bed7942359e3ad183f51961f62
Author: rogelioandrade <rogelio.a.mancisidor@bi.no>
Date:   Fri Aug 19 08:48:04 2022 +0200

    initial commit
(END)
```

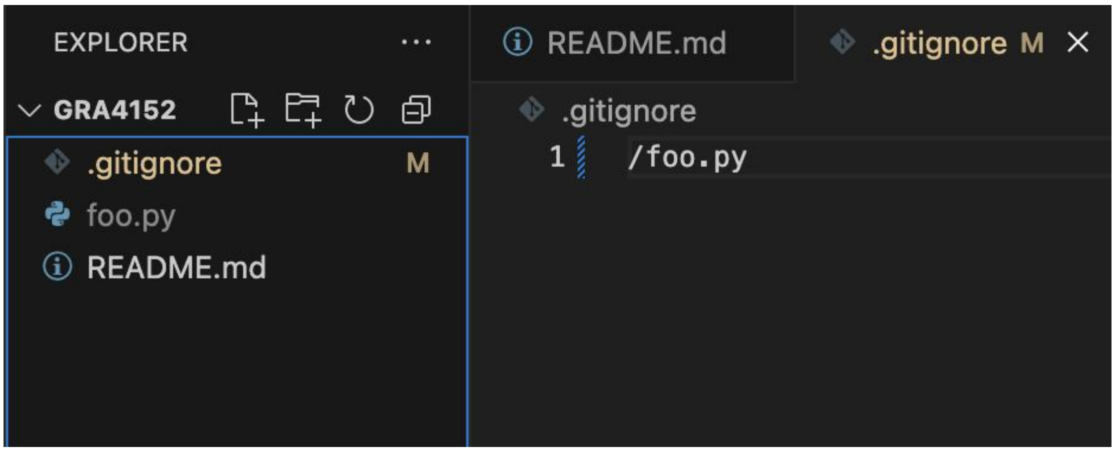
c)

git blame -- README.md

the commit message associated with the last modification to the

README.md is: You are free to form study groups and may discuss homework in groups. However, each student must write down the solutions and code from scratch independently and must understand the solution well enough. It is a honor code violation to copy, refer to, or look at written or code solutions from a previous year or solutions posted online (inspired by the Stanford Honor Code).

2.

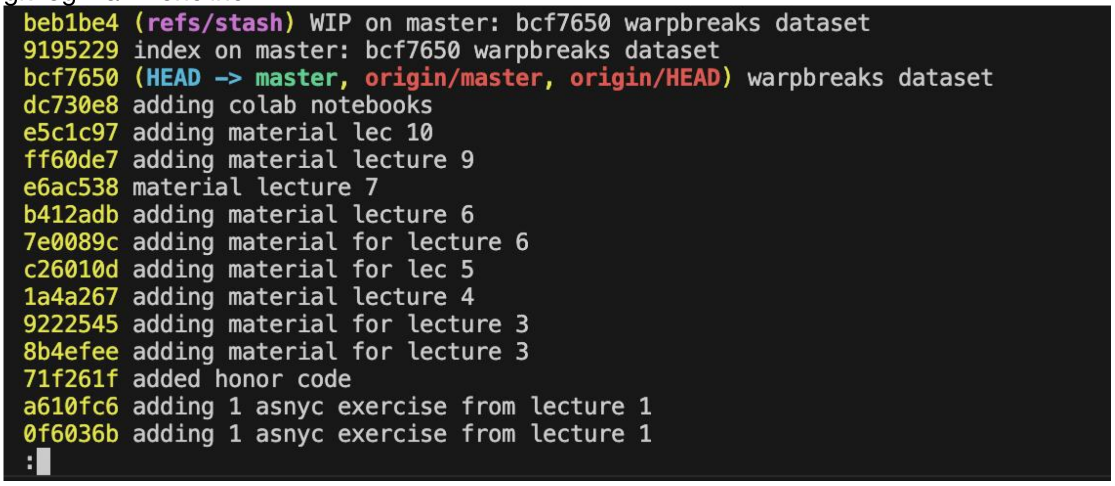


```
.gitignore
/foo.py
```

3.
a)
git stash
Saved working directory and index state WIP on master: bcf7650 warpbreaks dataset

My uncommitted changes are saved into a stash stack.

b)
git log --all --oneline



```
beb1be4 (refs/stash) WIP on master: bcf7650 warpbreaks dataset
9195229 index on master: bcf7650 warpbreaks dataset
bcf7650 (HEAD -> master, origin/master, origin/HEAD) warpbreaks dataset
dc730e8 adding colab notebooks
e5c1c97 adding material lec 10
ff60de7 adding material lecture 9
e6ac538 material lecture 7
b412adb adding material lecture 6
7e0089c adding material for lecture 6
c26010d adding material for lec 5
1a4a267 adding material lecture 4
9222545 adding material for lecture 3
8b4efee adding material for lecture 3
71f261f added honor code
a610fc6 adding 1 asnyc exercise from lecture 1
0f6036b adding 1 asnyc exercise from lecture 1
:
```

c)
git stash pop
On branch master
Your branch is up to date with 'origin/master'.

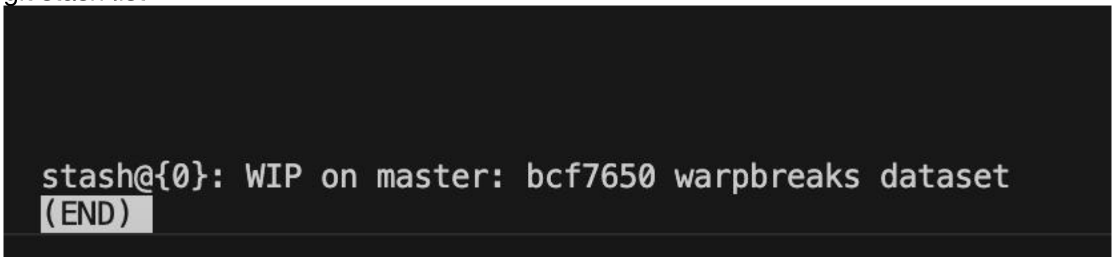
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)

```
	modified:   cubes.py

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (beb1be4e1dc42949be2ac17be3cdea3459ec4ee0)
```

I saw my changes reappear in my files.

Scenario:
When I need to handle something urgent but don't want to commit my recent changes yet, I can use git stash. It saves my changes into a stash stack and gives me a clean working directory so I can switch branches or make other commits. Later, I can restore my changes with git stash pop. It is very useful in OOP.

d)
git stash list



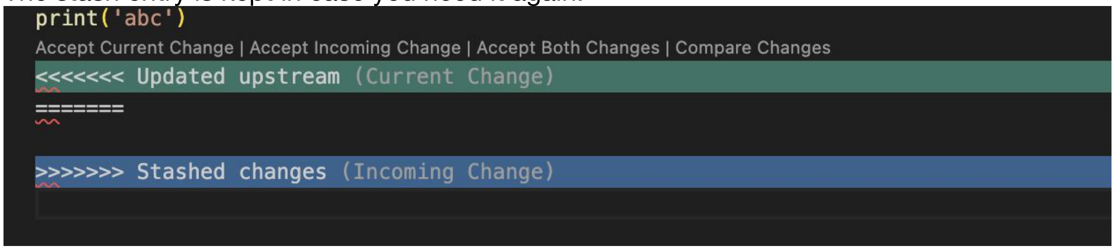
```
stash@{0}: WIP on master: bcf7650 warpbreaks dataset
(END)
```

git stash drop stash@{0}

Dropped stash@{0} (702c14a831e52938ea487d913706e4d5e4daf45a)

a)

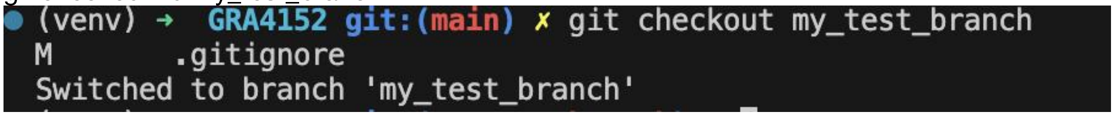
```
git stash pop
Auto-merging Lecture-1/cubes.py
CONFLICT (content): Merge conflict in Lecture-1/cubes.py
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

Unmerged paths:
  (use "git restore --staged <file>..." to unstage)
  (use "git add <file>..." to mark resolution)
	both modified:   cubes.py

no changes added to commit (use "git add" and/or "git commit -a")
The stash entry is kept in case you need it again.
```



```
print('abc')
<<<<<<< Updated upstream (Current Change)
=======

>>>>>>> Stashed changes (Incoming Change)
```

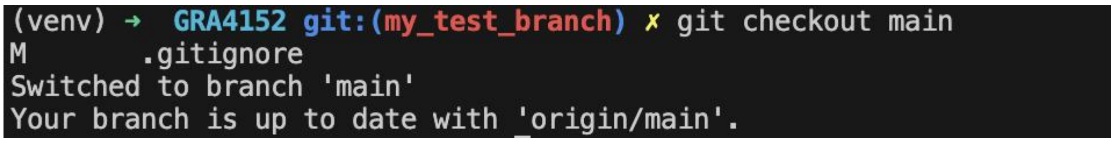
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes

4.
a)
git checkout -b my_test_branch



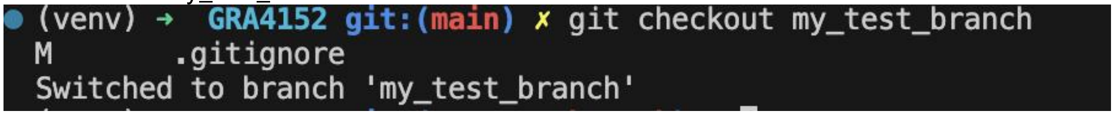
```
● (venv) ➜  GRA4152 git:(main) ✗ git checkout my_test_branch
  M       .gitignore
  Switched to branch 'my_test_branch'
```

Git checkout main



```
(venv) ➜  GRA4152 git:(my_test_branch) ✗ git checkout main
M       .gitignore
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```

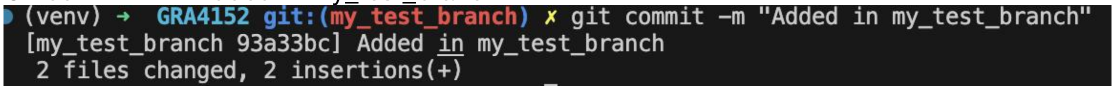
Git checkout my_test_branch



```
● (venv) ➜  GRA4152 git:(main) ✗ git checkout my_test_branch
  M       .gitignore
  Switched to branch 'my_test_branch'
```

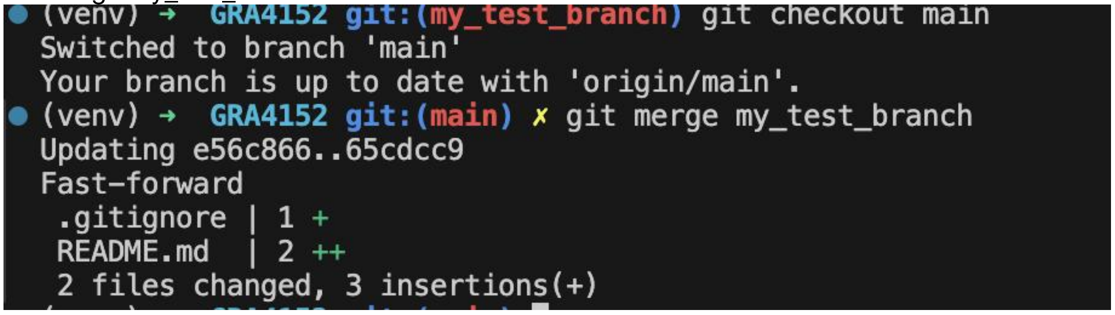
b)
Git add .

Git commit -m "Added in my_test_branch"



```
● (venv) ➜  GRA4152 git:(my_test_branch) ✗ git commit -m "Added in my_test_branch"
  [my_test_branch 93a33bc] Added in my_test_branch
   2 files changed, 2 insertions(+)
```

c)
Git checkout main
Git merge my_test_branch



```
● (venv) ➜  GRA4152 git:(my_test_branch) git checkout main
  Switched to branch 'main'
  Your branch is up to date with 'origin/main'.
● (venv) ➜  GRA4152 git:(main) ✗ git merge my_test_branch
  Updating e56c866..65cdcc9
  Fast-forward
   .gitignore | 1 +
   README.md  | 2 ++
   2 files changed, 3 insertions(+)
```