

Exercise 1: Clone the repository for the class <https://github.com/BI-DS/GRA-4152>

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

a. Explore the version history by visualizing it as a graph

```
cd ./GRA-4152  
  
git log --graph
```

```
* commit 71f261f8dbb09c828dfd2be1ad664a14b1fbc498 (HEAD -> master, origin/master, origin/HEAD)  
| Author: rogelioandrade <rogelio.a.mancisidor@bi.no>  
| Date:   Wed Aug 31 09:59:14 2022 +0200  
|  
|     added honor code  
|  
* commit a610fc6f9cdf3dd3e84e42e8e6ac168a73ab3a28  
| Author: rogelioandrade <rogelio.a.mancisidor@bi.no>  
| Date:   Mon Aug 29 09:33:29 2022 +0200  
|  
|     adding 1 async exercise from lecture 1  
|  
* commit 0f6036be424d789df67656a3be50cec8848f2d4f  
| Author: rogelioandrade <rogelio.a.mancisidor@bi.no>  
| Date:   Mon Aug 29 09:31:42 2022 +0200  
|  
|     adding 1 async exercise from lecture 1  
|  
* commit ae45ae7891f40dab569153b9decaaedc71601ff5  
| Author: rogelioandrade <rogelio.a.mancisidor@bi.no>  
| Date:   Mon Aug 29 08:51:21 2022 +0200  
|  
|     removing git_exercise.py file  
|  
* commit 84ed53d35856ac4c6a59a8de7853e2d723b44a39  
| Author: rogelioandrade <rogelio.a.mancisidor@bi.no>  
| Date:   Fri Aug 26 08:56:46 2022 +0200  
|  
|     adding file for git exercise in lecture 2  
| ...skipping...
```

Just highlighting the latest commits.

b. When was the last time README.md was modified? (Hint: use git log with an argument)

```
git log README.md
```

```
commit 71f261f8dbb09c828dfd2be1ad664a14b1fbc498 (HEAD -> master, origin/master, origin/HEAD)  
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
```

Was last modified 31. August 09:59:14 2022 +0200 by rogelioandrade.

- c. What was the commit message associated with the last modification to the README.md? (Hint: use git blame and git show)

```
git blame README.md
```

```
^20b8851 (rogelioandrade 2022-08-19 08:48:04 +0200 1) # GRA-4152
0fb7842d (rogelioandrade 2022-08-22 08:50:07 +0200 2) This repository contains different materials used throughout the
course, e.g. examples shown in lectures, suggested solutions for homework, problems discussed in tutorial sessions, etc.
You should follow this repository frequently, as different materials will become available as we cover th
0fb7842d (rogelioandrade 2022-08-22 08:50:07 +0200 3)
0fb7842d (rogelioandrade 2022-08-22 08:50:07 +0200 4) ## Packages
0fb7842d (rogelioandrade 2022-08-22 08:50:07 +0200 5) Unified Modeling Language (UML) is a tool to visualize the design
, or architecture, of (complex) software systems. Just like classes in OOP. We can generate UML diagrams for 'Python' cl
asses using the library 'pylint', which uses 'graphviz' to generate 'png' or 'pdf' files showing the architecture of a g
iven class.
0fb7842d (rogelioandrade 2022-08-22 08:50:07 +0200 6) ``bash
0fb7842d (rogelioandrade 2022-08-22 08:50:07 +0200 7) pip install pylint
0fb7842d (rogelioandrade 2022-08-22 08:50:07 +0200 8) sudo apt install graphviz
0fb7842d (rogelioandrade 2022-08-22 08:50:07 +0200 9) pyreverse -o png <your code>.py
0fb7842d (rogelioandrade 2022-08-22 08:50:07 +0200 10) ``
71f261f8 (rogelioandrade 2022-08-31 09:59:14 +0200 11)
71f261f8 (rogelioandrade 2022-08-31 09:59:14 +0200 12) ## Honor Code
71f261f8 (rogelioandrade 2022-08-31 09:59:14 +0200 13) You are free to form study groups and may discuss homework in gro
ups. However, each student must write down the solutions and code from scratch independently and must understand the sol
ution 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).
71f261f8 (rogelioandrade 2022-08-31 09:59:14 +0200 14)
```

The commit message associated with the latest modification contains the hash 71f261f8, and we can use this to read the message:

```
git show 71f261f8
```

```
commit 71f261f8dbb09c828dfd2be1ad664a14b1fbc498 (HEAD -> master, origin/master, origin/HEAD)
Author: rogelioandrade <rogelio.a.mancisidor@bi.no>
Date:   Wed Aug 31 09:59:14 2022 +0200

    added honor code

diff --git a/README.md b/README.md
index a7359ae..a404da3 100644
--- a/README.md
+++ b/README.md
@@ -8,3 +8,7 @@ pip install pylint
 sudo apt install graphviz
 pyreverse -o png <your code>.py
 ``
+
+## Honor Code
+You are free to form study groups and may discuss homework in groups. However, each student must write down the solutio
ns and code from scratch independently and must understand the solution well enough. It is a honor code violation to cop
y, refer to, or look at written or code solutions from a previous year or solutions posted online (inspired by the Stanf
ord Honor Code).
```

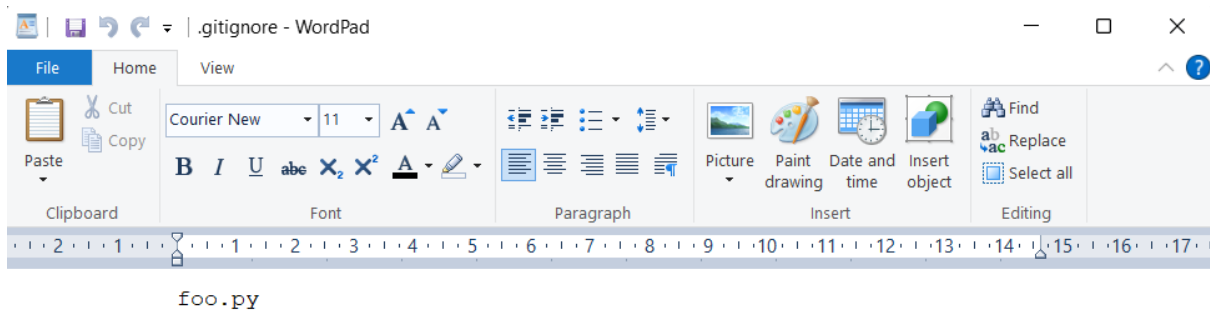
Exercise 2: One common mistake when learning git is to commit large files that should not be managed by Git or adding sensitive information. Add a .gitignore file to your portfolio code repository (<https://github.com/S1813163/GRA4152>) and exclude files and/or folders. You might need to create a foo.py file to be excluded.

```
echo "# GRA4152" >> foo.py  
  
git status
```

```
Untracked files:  
(use "git add <file>..." to include in what will be committed)  
foo.py
```

Now we create a ".gitignore" file and include "foo.py" in it, then we "add it" in the terminal using

```
git add .gitignore
```



Testing with different files, such as creating an "test.exe" file, and then adding "*.exe" into .gitignore, we see that everything works correctly.

Exercise 3: Clone some repository from GitHub and modify one of its existing files. What happens when you type `git stash`? What do you see when running `git log --all --oneline`? Run `git stash pop` to undo what you did with `git stash`. In what scenario might this be useful? List your current stashes and delete them with `git stash drop <stash_id>`

Now, modify a file and stash changes. Make a new modification to the same file, but this time commit those changes. What happens if you type `git stash pop` and open the file that you have modified? What do you see in the file?

The repository of choice is <https://github.com/maxlamberti/time-series-momentum>.

```
git clone https://github.com/maxlamberti/time-series-momentum.git
cd ./time-series-momentum
```

We modify the `.gitignore` file associated with the cloned repository and check its status:

```
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:   .gitignore

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

```
git stash
```

```
Saved working directory and index state WIP on master: 602369f Update README.md
```

When “stashing” we temporarily store changes without committing them. Running `git log --all --oneline` we can see changes with their associated hashes.

```
558a2e6 (refs/stash) WIP on master: 602369f Update README.md
9b73cde index on master: 602369f Update README.md
602369f (HEAD -> master, origin/master, origin/HEAD) Update README.md
c675023 Update README.md
23c58ae Updates read me with links to report and summary image
4f60c5a Merge pull request #2 from maxlamberti/refactor-clean-up
a2713ff (origin/refactor-clean-up) adds report write up to repo
a86dfcf syncs with local
```

The hash associated with our stash is “602369f” and can be read on the third line. Running `git stash pop` to undo what we did with `git stash`:

```
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:   .gitignore

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (558a2e6a7d0e26b4786502ed3d086f5524a485f2)
```

The command “git stash” is very convenient if one wants to work on another task without committing the changes that are unfinished.

```
git stash list
```

```
stash@{0}: WIP on master: 602369f Update README.md
```

To delete our current stashes we remove them in accordance to their stash ID:

```
git stash drop stash@`{0}`
```

```
Dropped stash@{0} (6387208b486249fb0fddc0b8946b57dc09ef95d6)
```

Now we modify the same file again, stash these changes, make a new modification and then commit. Running git stash pop gives us an merge conflict:

```
Auto-merging .gitignore
CONFLICT (content): Merge conflict in .gitignore
On branch master
```

In the file we can observe the following changes which are made in regards to the stashes:

```
<<<<<<< Updated upstream
# stash change 2
d.exe
=====
# Stash changes
a.exe
b.exe
>>>>>>> Stashed changes
```

Exercise 4: Create a new branch in your class repository (<https://github.com/S1813163/GRA4152>) and call it `my_test_branch`. Explore both branches, by switching back and forth. Add a comment line in any file in the branch `my_test_branch`, add and commit your changes. Finally, merge `my_test_branch` into `master`.

First we create a branch called “`my_test_branch`”:

```
git branch my_test_branch
```

Now we can switch between the branches using `git checkout` (`master/my_test_branch`):

```
git checkout my_test_branch    “or git checkout master”
```

```
Switched to branch 'my_test_branch'
```

```
Switched to branch 'master'
```

Lastly, we switch to the test branch and add a line in the `.gitignore` file.

```
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:   .gitignore
```

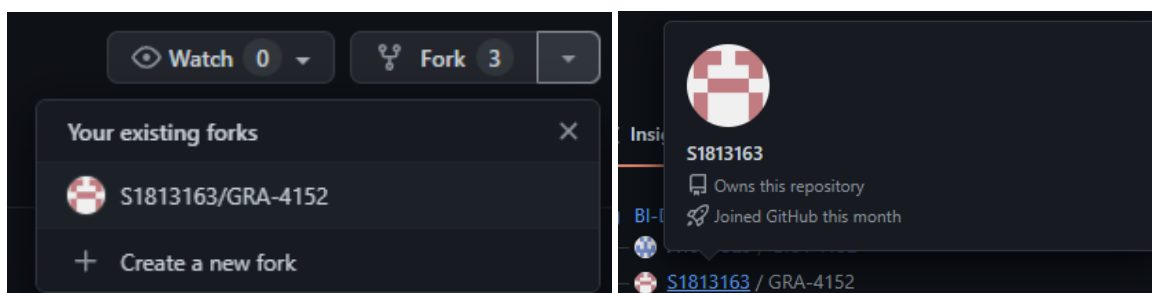
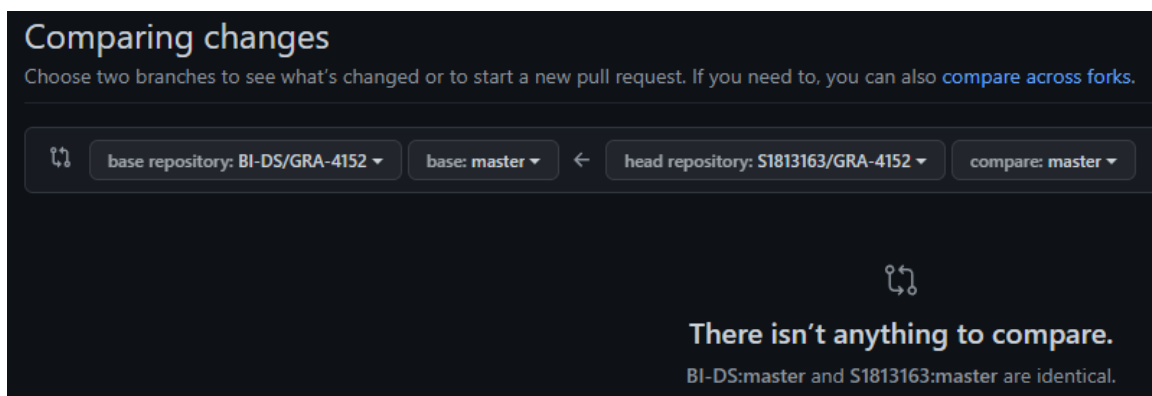
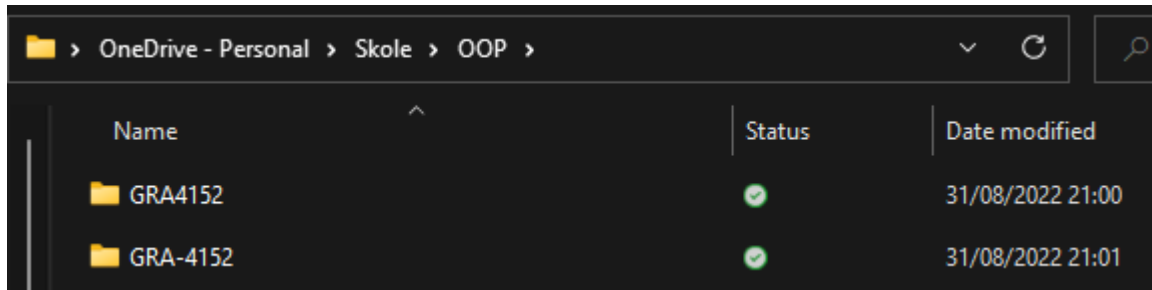
```
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   .gitignore
```

```
git add .gitignore
git commit -m "added line to .gitignore"
git checkout master
git merge my_test_branch
```

```
Updating 0d3bf03..d2a1f32
Fast-forward
 .gitignore | 3 ++-
```

Exercise 5: Fork the class repository (<https://github.com/BI-DS/GRA-4152>) and clone it, so you can contribute to its development. Send me a pull request to add a text file with your student id, i.e. S1813163.txt


We fork the class repository as shown in the pictures below and we clone the class repository using the same method as in exercise 1.





```
git checkout -b request master
git remote add upstream https://github.com/BI-DS/GRA-4152
git add S1813163.txt
git commit -m "Pull request to add student ID"
git push -u origin master
```


```
[request 69a3bd2] Pull request to add student ID
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 S1813163.txt
```


Pull request to add student ID - S1813163 #15

 Open S1813163 wants to merge 1 commit into `BI-DS:master` from `S1813163:master` 

 Conversation **0**

 Commits **1**

 Checks **0**

 Files changed **1**



S1813163 commented now



No description provided.



Pull request to add student ID

82f2c51

Add more commits by pushing to the `master` branch on `S1813163/GRA-4152`.



This branch has no conflicts with the base branch

Only those with [write access](#) to this repository can merge pull requests.