git_report

Clone your repo to the local directory.

```
git clone <https://github.com/Balan666/ITMO_ScientificPython_2024.git>
```

To set the origin I used command

```
git remote add origin
<https://[access_token]@github.com/Balan666/ITMO_ScientificPython_2024>
```

We need a token because of double authentification, if you try to push without doing this, it won't be able to authentificate despite of login&password being right

I started again with deleting the branches:

```
(base) bananna@bananna-victus:~/Downloads/SciPyITMO2024/ITMO_Sci
* main
  remotes/origin/HW1
  remotes/origin/main
  remotes/origin/testing
```

Create a branch: `git branch HW1`

Switch to a branch: `git checkout HW1`

```
touch README.md
mkdir HW1
cd HW1
touch hw1.txt test_revert.txt test_revert_merge.txt
nano hw1.txt #write the needed text, do this to all the files
git add . # add all the files to git
git commit -m 'added three needed text files to the HW1 director
git push -u origin HW1 #pushed new branch to the repo
```

Now create branch "testing" but do not checkout it. `git branch testing`

we're still in the branch HW1 and we change hw1.txt:

```
nano hw1.txt #write a new string
git add hw1.txt
git commit -m 'modified   hw1.txt added the new line'
git push origin HW1:HW1
```

Now checkout "testing" branch and change "test_revert.txt" (all the files are there bc testing was made out of HW1)
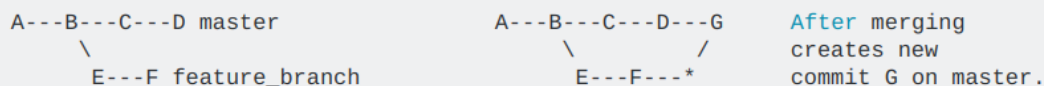
```
git checkout testing
nano test_revert.txt
git add test_revert.txt
git commit -m 'added string to test_revert.txt'
git push -u origin testing
```

Now let's merge testing into HW1: `git merge testing`

▼ git branch -a

　*HW1
　main
　testing

　remotes/origin/HW1
　remotes/origin/main
　remotes/origin/testing

```
A---B---C---D master          A---B---C---D---G      After merging
     \                             \         /       creates new
      E---F feature_branch          E---F---*        commit G on master.
```

In our casse it looks like this, bc we have less commits:

A—B—C HW1
 \   /
  E—*   testing

the changes from the merged branch are combined into the branch you merged them into. Git also creates a new commit to record the merge.

```

Both branches still exist, but the history will show that they have been merged. Each branch will have its own commit history, and you can continue development on either branch after the merge.

Undo the merge:

> 1. Identify the parent commit of the merge commit by running the following command:
>
> `git log`
>
> 2. Copy the hash of the parent commit from the output of the log. It should be the commit before the merge commit you want to revert.
>
> 3. Revert the merge commit indicating the parent commit to keep:
>
> `git revert -m 1 <parent_commit_hash>`
>
> Replace `<parent_commit_hash>` with the actual hash of the parent commit identified in step2.

Now, we go to test_revert_change.txt, that always looked like one string in both of the branches

```
git checkout testing
nano test_revert_merge.txt
git add test_revert_merge.txt
git commit -m 'added a line to test_revert_merge.txt'
git push origin testing:testing
```

```
git checkout HW1
git merge testing
```

Merge made by the 'ort' strategy.
HW1/test_revert_merge.txt | 1 +

1 file changed, 1 insertion(+)

No conflict arose, but I expected to see two strings in all of the files from HW1 branch. However, it looks like that:

```
git merge testing
#Merge made by the 'ort' strategy.
 #HW1/test_revert_merge.txt | 1 +
 #1 file changed, 1 insertion(+)


cat hw1.txt
#This is an initial file.
#This is a change.

cat test_revert.txt
#This is a file for testing revert.

cat test_revert_merge.txt
#This is a file for testing revert and merge.
#This is a change.

git checkout testing

#Switched to branch 'testing'
#Your branch is up to date with 'origin/testing'.

cat hw1.txt
#This is an initial file.

cat test_revert.txt
#This is a file for testing revert.
#This is a change.

cat test_revert_merge.txt
```

```
#This is a file for testing revert and merge.
#This is a change.
```

It could be resolved manually, but it's a bad solution

`git push origin HW1:HW1`

Delete the branch

`git branch -d touch`