## Undoing things

# 1- Undo a non-staged change in one of the files *echo* "#some changes" *>> cleaner.py* # modify *git* checkout cleaner.py # undo

echo "concordia" >> cleaner.py

git checkout cleaner.py

# 2- Undo a staged change in one of the files *echo* "#some changes" *>> cleaner.py* # modify *git* a dd cleaner.py # stage *git* r eset HEAD # undo (stage) gi

*git* c heckout . # undo (modify)

echo "continue education" >> cleaner.py

git add cleaner.py

git commit -m "assignment2 commit"

git reset --hard HEAD~1

HEAD is now at fb2d4cc commit02

# 3- Undo a committed change in one of the files *echo* "#some changes" *>> cleaner.py* # modify *git* a dd cleaner.py # stage *git* c ommit -m "make changes staged" # commit *git* r eset --hard HEAD~1 # undo

echo "canadian" >> cleaner.py

git add cleaner.py

git commit -m "practice commit"

git reset --hard HEAD~1

HEAD is now at fb2d4cc commit02

# 4- Reset your master branch to an older commit (your changes will be lost, add a new fake commit if you don’t want to lose it!) *touch* f akefile *git* a dd fakefile

*git* c ommit -m "adding fake file" *git* l og --pretty=oneline older\_commit= **$** *(git log --pretty=oneline | grep "Adding remaining files" | cut -d' ' -f1) git* r eset --hard **$** {older\_commit} # undo

echo > fake.txt

ls

HOMEWORK cleaner.py fake.txt processor.py

README.md data.csv practice.py submit.py

git add fake.txt

git commit -m "adding fake.txt"

git log -- pretty=oneline older\_commit= $(git log --pretty=oneline | grep "Adding remaining files" | cut -d' ' -f1)

git reset -- hard $ {older\_commit}

## Create a branch to split your development

# 1- Create a branch called new-data and add a commit to it with a new file called data2.csv.

*git* c heckout -b new-data master *echo* "# data2.csv content" *> data2.csv git* a dd data2.csv *git* c ommit -m "adding data2.csv"

git checkout -b new-data master

Switched to a new branch 'new-data'

echo "data2.csv new data file" > data2.csv

git add data2.csv

# 2- Checkout your master and check the files that you see.

*git* c heckout master *ls*

git checkout master

ls

HOMEWORK cleaner.py fake.txt processor.py

README.md data.csv practice.py submit.py

# 3- Create another branch from master and check the files that you see. Add two commits to it in any files. *git* c heckout -b another-branch master *echo* "#some changes" *>> cleaner.py* # modify

created new branch named –practice-git

git checkout -b practice-git master

Switched to a new branch 'practice-git'

echo "practice git" >> cleaner.py

cat cleaner.py

# cleaner.py content

Hello Git

hello sombel nice to see you

practice git



*git* a dd cleaner.py *git* c ommit -m "first commit" *echo* "#some changes" *>> processor.py* # modify *git* a dd processor.py *git* c ommit -m "second commit"

git add cleaner.py

git commit -m "practice commit"

echo "learning git" >> processor.py

cat processor.py

# processor.py content

learning git

git add processor.py

git commit -m "processor commit"

# 4- Checkout your master again and checking the files at each step, merge both branches into it. *git* c heckout master *ls*

*git* m erge --no-edit new-data *ls git* m erge --no-edit another-branch *ls*

git checkout master

Switched to branch 'master'

git merge -- new-data practice-git

Fast-forwarding to: new-data

Trying simple merge with practice-git

Merge made by the 'octopus' strategy.

cleaner.py | 1 +

data2.csv | 1 +

processor.py | 1 +

3 files changed, 3 insertions(+)

# 5- Visualize with git log --graph --oneline --decorate --all what happened.

*git* l og --graph --oneline --decorate --all

log --graph --oneline --decorate --all

\* eb1019d (**HEAD -> master**) Merge branches 'new-data' and 'practice-git' "Merging Commit"

|\

| \* a6e3dfc (**practice-git**) processor commit

| \* 4e033ec practice commit

\* | 2d0313d (**new-data**) data2 commit

|/

\* 17e4704 adding fake.txt

\* fb2d4cc (**origin/master**, **origin/HEAD**) commit02

\* 8696f0a modifid commit

\* 6fc3a11 files commited

\* d74cade Initial commit

   