Exercise 1

Goals

- Create Git repository from scratch
- Track changes in files using `git add`, `git status` and `git commit`
- Know state of Git repo using `git diff` and `git log`

Structure

This exercise consists of short descriptions about a specific git command, followed by a practical part where you can execute appropriate git commands. In order to allow a smooth exercise, there are some functions written by C2SM in the file *helpers.sh* that are **NOT** part of Git. For this exercise we use the following functions from that file:

- **init_exercise:** It will create the *work* directory and navigate into it
- **reset:** It will delete the *work* folder and enable you a clean restart of the exercise in case you completely mess it up.

In the following (and all further exercises), all text enclosed with `<>` denotes a placeholder to be replaced by a specific string appropriate in **your** context.

Start exercises in correct folder

This exercise (and all further exercises) assume that the shell is already located in the folder of the exercise notebooks. It could happen, for some reason, that the notebook does not by default switch to the folder of the notebook. In that case you need to manually change the directory first in order to do the exercises.

If the command

`pwd`
returns something similar to
`/home/juckerj/git-course/Exercise_1`
everything is fine.

In case it returns only
`/home/juckerj`
please go the right directory.

Initialization

In [ ]:
```
# check current directory with "pwd"
pwd
# go to folder of this exercise using "cd"
```

In [ ]:

```
# execute this code at the very beginning to initialize the exercise proper
source ../helpers.sh
init_exercise
```

Optional: clear notebook and restart

In case you mess up your notebook completely,
execute *reset* in the following cell. This will restore a clean environment!

In [ ]:

```
## only execute in case of (serious) trouble ##
## it will delete your entire work directory ##
reset
```

Exercise

Create Git-Repository from scratch

In [ ]:

```
# create new folder (e.g. <git_repo>) and enter it
```

In [ ]:

```
# use the command "git init" to initiate your first Git-repository
```

You should now get an output similar to that:

```
Initialized empty Git repository in /Users/juckerj/Desktop
/git_course/notebook/work/git_repo/.git/
```

Track changes in files using git add and git commit

In a next step you will add some files to you repository.
To do so, we make use of the *echo* command in combination with the ">" operator to direct its output to a file.

In [ ]:

```
# to create a new file use the echo-command
# echo "<my text for file>" > first_file
```

In [ ]:

```
# use echo and ">" again to create a second text file
```

In [ ]:

```
# check the status of your Git-repository with "git status"
```

You should now get an output similar to:

```
On branch master

No commits yet
```

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

nothing added to commit but untracked files present (use "git
add" to track)
```

Git recognized these two new files, but the files are not yet included in the repository.

In [ ]:

```
# add the files using "git add"

# check your actions with "git status" again
```

Your output should look as follows:

```
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   first_file
    new file:   second_file
```

The last thing to do is to commit these files.

Prior to that, Git may require us to tell who we are. To do so, execute the lines below with your credentials:

```
git config --global user.name ”<John Doe>"
git config --global user.email ”<my_name@some.domain>"
```

Note: The email must be identical to the one that is used for your GitHub account.

In [ ]:

```
# tell git who you are
```

In [ ]:

```
# use 'git commit -m "<meaningful message>"'
```

Congrats!
Your files are included in the Git-repository.

Know state of Git-repo using git diff and git log

For now we have two files in our Git-repository.
Let's see what happens when we modify them. We therefore use the ">>" operator to append a new line of text to our files.

In [ ]:
```
# append a new line of text with "echo" and ">>"
```

In [ ]:
```
# check state of you repository with "git status"
```

Your output should look similar to:

```
On branch 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:   first_file

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

At some point you forget about the additional line we just added.
Git provides a command to check, what new changes are contained in a file:
git diff

In [ ]:
```
# see local changes of modified file with "git diff your_filename"
```

In the output

```
diff --git a/first_file b/first_file
index 3829ab8..a32d2f3 100644
--- a/first_file
+++ b/first_file
@@ -1 +1,2 @@
 myfirstline
+mysecondline
```

we see a lot of information. But all we care at the moment is the last line:
The + indicates that we have a new line in our file.

Because Git is so easy, we modify the second file as well.

In [ ]:
```
# add a new line in the second file as well
```

The next lecture is starting soon, so let's add and commit our changes for safety reasons in Git.

In [ ]:
```
# add you two modified files with "git add"


# git status to check if your action was successful
```

In [ ]:
```
# use 'git commit -m "<meaningful message>"' to commit your files
```

Congrats!

But how many commits do I already have in this repository? Git does all this tracking for us!

`git log` allows us to look back in time and explore what commits are contained in our repository.

In [ ]:

```
# type git log go get an overview of the (very short)
# life of your repository
```

Below you see an example how your log could look like:

```
commit 26c65dd070e995db55ac46d76cdb5052da03f5cb (HEAD ->
master)
Author: juckerj <jonas.jucker@env.ethz.ch>
Date:   Tue Feb 23 17:16:03 2021 +0100

    second commit

commit 495eb9387e4407f3accb57a8f29d7362eead85bb
Author: juckerj <jonas.jucker@env.ethz.ch>
Date:   Tue Feb 23 16:09:54 2021 +0100

    test
```

We see the unique hash of each commit, its author as well as the date of the commit. Those are all very useful things as we will see later in this course.