Exercise 4

Goals

- Exercise the typical Git workflow

Structure

In this exercise we won't learn any new git commands. We rather repeat all commands and workflows learned in the previous exercises. To do so we work on the *partyplan.txt* document and simulate a possible workflow. Because it is all repetition, there is less explanation provided.

The raw outline of this workflow:

- Add food and people
- Ben does not bring juice
- Add schedule to document
- Ben changes his mind again: brings food as well
- "Accidentially" delete all files, recover it
- Change schedule in another branch
- Create two additional branches for different music options
- Merge new schedule into master
- Delete merged branch
- Merge your preferred music branch into master and delete merged branch

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 us 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
- **init_empty_folder:** create the directory `party_planning` and the empty file `partyplan.txt` in it.

Reminder: *all text enclosed with* `<>` *denotes a placeholder to be replaced by a specific string appropriate in your context.*

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

Exercise typical Git workflow

In [ ]:
```
# this line will create the directory `party_planning` and the empty file `
init_directory_with_empty_file
```

In [ ]:
```
# init git repository
```

In [ ]:
```
# execute "ls -a" to also see the hidden git-folder
```

Add participants (Julia, Ben and Lea).
Julia brings chips, Ben brings juice and Lea brings wine.

To edit *partyplan.txt*, follow the instructions below:

- Go to folder *work* and enter *party_planning*
- Open *partyplan.txt*
- Change file
- Save changes by clicking on *File -> Save*

Please us this way of editing files throughout the exercise.

In [ ]:
```
# commit your changes
```

Ben does not brings any food to the party!

Adapt your document accordingly.

In [ ]:
```
# commit the adapted "partyplan.txt"
```

The party needs a rough schedule of what is happening:

- Door opening
- Start Music
- Happy Hour

Implement these points into your document.

In [ ]:
```
# commit your changes
```

Check if all of your changes are really tracked by Git.

In [ ]:
```
# git status

# git log
```

In case you don't have any untracked changes in your repository move on to the next part of this exercise.

It is already late at night, our concentration is not very high anymore...

By *accident* we delete all of our existing files...

In [ ]:
```
# CAUTION: executing this panel deletes all files in the current directory
rm *
```

Thanks to Git we can easily restore files, even if they are deleted.

In [ ]:
```
# restore deleted files
```

As a next step create a new branch and edit the schedule there, because we are not sure about the expected delay yet:

- Postpone the door opening for 2 hours

In [ ]:
```
# create new branch and adjust schedule
```

In [ ]:
```
# commit changes
```

Switch to branch *master* and checkout another branch on top of it. Implement the following changes to the document:

- The music played will be *Classical Music*
- The music will be for two hours

In [ ]:
```
# create new branch and adapt music played
```

In [ ]:
```
# commit changes
```

Switch to branch *master* and checkout another branch on top of it. Implement the following changes to the document:

- The music played will be *Dj-Set Techno*
- The music will be for 6 hours

In [ ]:
```
# create new branch and adapt music played
```

In [ ]:
```
# commit changes
```

It's time to get an overview of what we just did. Our repository has currently 4 branches (names may be different for your case:

- master (base version of all subsequent branches)
- schedule (later door opening)
- classic_music (classical music for 2 hours)
- dj_set (Dj-Set with Techno for 6 hours)

Ensure your repository contains the same amount of different branches.

In [ ]:
```
# display all branches of the Git repository
```

The delay of the door-opening is confirmed. Therefore merge the branch *schedule* into master.

Remark: Make sure you are in the master branch before starting the merge, because Git always merges a branch **into** your current branch.

In [ ]:
```
# merge schedule branch into master
```

Inspect the document and make sure the changes from the merged branch are present.

If everything is ok, make a commit

In [ ]:
```
# commit merge
```

In [ ]:
```
# follow good practice and delete merged branch
```

The last step of the planning is to choose one of the music options. Decide what music you prefer.

Merge the corresponding branch into master.

In [ ]:
```
# merge your preferred music into master
```

In [ ]:
```
# delete merged branch
```

There is one unused branch left in our repository. We want to keep the repository clean and nice. So please delete the unmerged branch as well.

Remark: Since we did not merge the remaining branch, a different option is needed to *git branch*

In [ ]:
```
# delete unused branch
```

Congrats

You already new the most essential Git commands for the local use of Git.

Further exercises will focus more on typical workflows with remote servers like GitHub or Gitlab.