Problemset: Git and Github (I)

1 Git and GitHub installation

- Download Git from https://git-scm.com and install. You will be prompted for many aspects. I recommend the following installation settings (for other settings choose the defaults): (1) Default editor: Visual Studio Code, (2) Initial branch name: "main", (3) Credential helper: Git Credential Manager
- To configure git, run the following commands on the command line:

```
git config --global user.name "Your Name"
git config --global user.email "your@email.address"
```

- Create a GitHub account, using the email address that you specified in the git configuration.

2 Basic Git Workflow

- **Initialize Git**: First, create a new folder and open it in VS Code. Then initialize git for this directory (`git init`). Run `git status` to see the current state of your git repository.
- **Basic Git Workflow**: (1) Create a new file `readme.md` and save it. (2) `add` the file to the staging area. 3. `commit` your changes to your local git repository using a commmit message that describes your change (`git commit -m "your message"`).
- **Repeat**: Carry out some standard file operations (create a new file; edit a file; delete a file), and repeat the above git workflow multiple times. (1) Use `git status` after each git operation to learn about the current status of your repository. (2) Use `git log` after each commit to see the history of your commits. (3) Use `git diff` to compare different versions of your repository.
- **VS Code Source Control GUI**: Again repeat the basic git workflow multiple times using VS Code's Source Control GUI. Install the VS Code extension **Gitlens** or **Git Graph** to see a visualization of your commit history.

3 Workflow including Github

- **Publish your repo**: Publish your repo to GitHub using VS Code's Source Control GUI. Then open the repo on Github and look at how the information is presented there.
- **Push**: Again create/edit/delete files and repeat the basic git workflow. But this time finalize the worflow by pushing your commits to Github (`git push`).
- **Pull**: Now edit one of the files directy on Github and commit your change there. Then use `git pull` to incorporate your remote changes into your working directory.
- **Fetch and merge**: Run a variant of the previous operation: Edit a file directly on Github and commit. Then `git fetch` your remote changes. Can you already see the changes in your working directory? What does `git fetch` do? Use `git status` to check the current state of your repository. Then use `git merge origin/main` to merge into your working directory.

4 Undo changes

- **Revert**: (1) Create a "bad" commit by deleting some contents from some file. (2) Use `git diff` and `git log` to verify which commit is the last "good" one. (3) Revert the bad commit using `git revert <badcommit>`. (4) Check how your commit history has changed.
- **Recover file using revert**: Run a variant of the previous operation: Delete some important file and commit ("bad commit"). Then recover the file using the same procedure as before.
- **Reset**: Now undo changes using `git reset --hard` instead of `git revert`. Use `git log` and `git status` to analyze how resetting is different from reverting. In what sense is `reset --hard` a risky operation? Then try out how `git reset --mixed` differs from `git reset --hard`.

5 Clone Github repository

- So far, you have followed a local-first approach: you first created a local git repository, and then published it to Github. Now, use a Github-first approach: create a new Github repository using the Github website. Then copy the HTTPS url and clone it to your local system (`git clone <url>`). Make sure that you first switch to the location on your system where you want to get the new repository cloned to.
- You can also use `git clone` to clone some public Github repo of other authors: Search for the Github repo of the Python Pandas package and clone it to your system.