BRIEF INTRODUCTION TO REVISION CONTROL USING GIT

OVERVIEW

- Revision control using git (git-scm.com)
- Local repository for code development
- Remote repositories (for backup and for distributing source)
- github, gitlab for ease

REVISION CONTROL

Revision or version control provides a **management of (commits of) changes** made to files over time. Allows you to **keep track of versions, revert** to earlier stages so you can try out ideas without worrying about reverting the changes if the ideas don't work out. And it allows for the **merging of changes** made by your collaborators on a joint project. Finally, git can act as a **remote backup** of your work, and a method for **synchronising source** between various computers (e.g. your personal computer and the super computer used in executing the program).

Download git at http://git-scm.com

SIMPLE WORKFLOW WITH GIT

1. Create a git repository of the current directory by issuing the command

```
git init
```

2. Edit project files, e.g. “SolveProblem.py”. Satisfied that it works, add the current status to the staging area for the next commit:

```
git add SolveProblem.py
```

3. Commit the current status to the local repository

```
git commit –m “First attempt”
```

SIMPLE WORKFLOW WITH GIT

4. Work some more, then realise this is a mistake, and revert file to the last commit

```
git checkout -- SolveProblem.py
```

5. Implement a new feature, but want also to be able to fix bugs in the current version while the new feature is implemented (may take some time to implement): Create a new branch called NewFeature for the work:

```
git checkout -b NewFeature
```

This also changes the working directory to be on that branch

SIMPLE WORKFLOW WITH GIT

- The name for the starting branch differ between setups. You can check the name relevant for your setup by issuing the command

  `git status`

  This will report something like:

  ```
  On branch master
  No commits yet
  nothing to commit
  ```

  meaning the branch name is `master`. `Git status` is useful!

SIMPLE WORKFLOW WITH GIT

6. Implement start of feature, add, commit:

```
git add SolveProblem.py
git commit –m “Start of new feature”
```

7. Change back to original branch to correct a bug there

```
git checkout master
```

8. Import merge the features from NewFeature into original:

```
git merge NewFeature
```

SIMPLE WORKFLOW WITH GIT

Before committing changes, it is good practice to check exactly what has changed, and that all of these changes were intended:

```
git diff SolveProblem.py
```

Each commit has a unique checksum. One can use this to revert to earlier versions

```
git log
```

```
git checkout <checksum>
```

REMOTE REPOSITORIES

Instead of storing the repository on the computer used for development, one can use remote repositories. One can then check out a copy on several machines, e.g. check out a copy on the machine used for development, and check out one on the machine used for running the code. Check out a repository from a remote server:

```
git clone [url]
```

This clones all the repository to the local computer, and you can work on this as detailed so far. The local commits can be pushed to the remote server by

```
git push
```

REMOTE REPOSITORIES

If someone else is committing to the remote repository too, you may want to pull in changes committed to the remote repository

```
git pull
```

This is best done if your local directory is in a clean state (i.e. has no changes compared to a commit). One way round:

```
git stash
git pull
git stash pop
```

GITHUB AND GITLAB

Remote repositories can be created on **github.com** or **gitlab.com**, which also offers nice **graphs of the branch histories** and good facilities for checking differences between commits. And loads of other advanced features for project management.

EXERCISE

Create a local repository in a directory. Add a file with the content ‘first attempt’ and commit this to the repository. Create a branch and add a line with the content ‘change’ then commit this to the branch. Change back to the original branch and verify the file now contains just the one line. Add this to the first line ‘ might work’.

Then merge the new branch into the original. This will fail because of the change made. Use the stash commands to merge the changes in the new branch into the working directory. Check the content of the file.

GIT CHEAT SHEETS

- Comprehensive: https://www.atlassian.com/git/tutorials/atlassian-git-cheatsheet
- Illustrative: https://education.github.com/git-cheat-sheet-education.pdf

  https://about.gitlab.com/images/press/git-cheat-sheet.pdf