Git is a version control system designed to handle everything from small to very large projects. Every time you commit, or save the state of your project in Git, it basically takes a picture of what all your files look like at that moment and stores a reference to that snapshot. Git projects are stored in so called repositories, or “repo” for short, a digital directory or storage space where you can access your project, its files, and all the versions of its files that Git saves.

GitHub is a Git repository hosting service, but it adds many of its own features. While Git is a command line tool, GitHub provides a Web-based graphical interface. It also provides access control and several collaboration features, such as a wikis and basic task management tools for every project.

Getting started – Installing Git

Linux (CS50 appliance)

Git comes pre-installed on the CS50 appliance. If you use another Linux-based operating system, depending on the distribution, you can download and install Git as follows:

Debian based distribution, like Ubuntu:

```
$ sudo apt-get install git
```

For more options, there are instructions for installing on several different Unix flavors on the Git website, at http://git-scm.com/download/linux.

Mac OS

There are several ways to install Git on a Mac. The easiest is probably to install the Xcode Command Line Tools. On Mavericks (10.9) or above you can do this simply by trying to run Git from the Terminal the very first time. If you don’t have it installed already, it will prompt you to install it. Click “Install” to download and install Xcode Command Line Tools.

```
$ git
```

Alternatively, you can use a command to install Xcode Command Line Tools. It will produce a similar alert box. Note the double hyphen:

```
$ xcode-select --install
```

You can also install it as part of the GitHub for Mac install. Their GUI Git tool has an option to install command line tools as well. You can download that tool from the GitHub for Mac website, at http://mac.github.com.

Windows

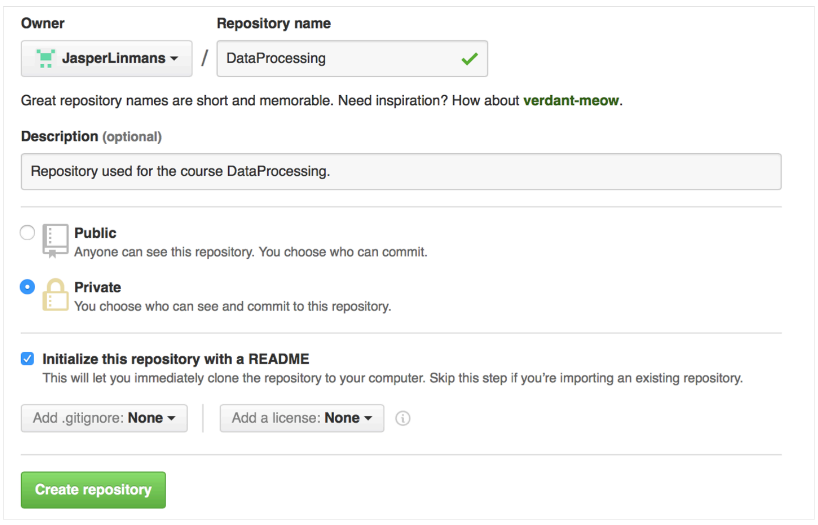
There are also a few ways to install Git on Windows. The easiest way to get Git installed is by installing GitHub for Windows. The installer includes a command line version of Git as well as the GUI. You can download this from the GitHub for Windows website, at http://windows.github.com.

Getting started – Creating a repository

In order to do anything in Git, you have to have a Git repository. This is where Git stores the data for the snapshots you are saving.

There are two main ways to get a Git repository. One way is to simply create a new one from an existing directory. The second way is to clone a public Git repository, if someone else already created a repo that you want to use.

Since we will be using Github to synchronise our repositories, the easiest way to create a new repository is to make one on the Github website. In the upper right corner of the screen, next to your profile picture, you have the option to create a new repository. Make sure you initialize your new repository with a README file.


Owner: JasperLinmans / Repository name: DataProcessing

Great repository names are short and memorable. Need inspiration? How about **verdant-meow**.

Description (optional): Repository used for the course DataProcessing.

Public — Anyone can see this repository. You choose who can commit.

[✓ Private] — You choose who can see and commit to this repository.

[✓] Initialize this repository with a README — This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None | Add a license: None

You now have a Git repository on Github. To make changes, save and edit new or existing files, you need to clone this repository locally. To clone the repository with Git commands, you need the Github repository URL. It is basically the URL of the Github repository with a *.git* suffix. You can find it on the right side of the repositories page in the box titled: "**HTTPS** clone URL".

Once you have the .git URL you are ready to clone the repository locally with the *git clone* command. If it is you first time connecting, Git will prompt you for your password. This is the password used to create your login on Github.com.

```
$ git clone https://github.com/JasperLinmans/DataProce
ssing.git
$ ls
DataProcessing
$ cd DataProcessing
$ ls
README
```

The three step process - Add, Commit and Push

Now that you have created a Git folder, synchronised with Github, you are ready to edit files and keep track of changes using Git. In a nutshell, you will use *git add* to start tracking new files and also to stage changes to already tracked files, then *git commit* to record your snapshot into your history and finally *git push* to synchronise the changes to the remote repository on Github. This will be the basic workflow that you use most of the time.

Git add

This command does not add new files to your repository. Instead, it brings new files to Git's attention. After you add files, they're included in Git's "snapshot" of the repository. In other words: you run *git add* on a file when you want to include whatever changes you've made to it in your next commit snapshot.

For the sake of this tutorial I will show you how to create a new text file and add it using *git add*, however this works on every file present in your Git folder.

```
$ echo “This is a new text file” >> test.txt
$ git add test.txt
```

To check if "test.txt" is ready to be committed you can use *git status*. This will show you all changes ready to be committed, all untracked files and all unstaged changes. Play around with this for a bit, try and create new files or edit them and see what *git status* gives as output. To undo a *git add <file_name>* action, you can use *git reset <file_name>*. To follow along with this tutorial, make sure that only test.txt is in the "changes to be committed" list if you execute *git status*. Just use *git reset* on the other files you just made and remove them from the directory.

Tip: you can add more than one file in one *git add* command by appending all filenames to the command, or simply add every current change to the files in your Git folder by pointing to ".".

```
$ git add test.txt test2.txt test3.txt
```

```
$ git add .
```

Git commit

Git's most important command. After you make any sort of change, you input this in order to take a "snapshot" of the repository. In other words: After you have staged the content you want to snapshot with the *git add* command, you run *git commit* to actually record the snapshot. Usually it goes *git commit -m “Message here.”* The –m parameter is used to add the commit message.

In general, it's very important to write a good commit message, especially since Github is all about sharing code! On Github, you can easily read the history of the project by getting the list of all commit messages.

After modifying a file and setting it up to be committed with *git add* you can commit all changes with *git commit* like so:

```
$ git commit –m ‘added an empty file to the
repository’
[master 3b51c65] added an empty file to the repository
 1 file changed, 1 insertion(+)
```

Notice that you don’t need to specify the filename that has to be committed. *Git commit* simply commits everything that was set up by *git add*. Naturally, removing files works in the same way: removing a file from the repository, adding the change to *git add*, checking it with *git status* and committing the change with *git commit*.

Git push

Git commit is useful since you can now keep track of changes in the repository. However, this information is now only present on your computer. For this course it is necessary to synchronise your Git repository with your Github account so that the staff can grade your results.

Since you cloned your Github repository with *git clone*, other Git commands involving connection to the remote repository already know where to synchronise all files to. So it is as simple as:

```
$ git push
```

If you are using the same repository on more then a single computer, or if you are collaborating using the Github repository, the opposite of push is just as important:

```
$ git pull
```

Git pull synchronises all files found in the remote repository to your local repository.

In case you did not clone the repository to your computer, you can still connect your local Git folder with a remote repository on Github using the following commands:

```
$ git remote add origin <github URL to a
repository.git>
$ git push
```

Questions

1. Install Git
2. Create a Github account and immediately request an educational discount at: https://education.github.com
3. Create a private repository on the Github website, do not forget to include a README. This will be the repository that will contain all DataProcessing programming assignments. Name the repository: DataProcessing.
4. Create a folder inside your repository called week-1. Create a txt file in the “week-1” folder and use it to save the answer to question 5.
5. What is a so called “branch” in Git terms and why would you use it.
6. Make sure to (only) add the staff to your Dataprocessing repository’s collaborators list.
7. Ask help if you need it!