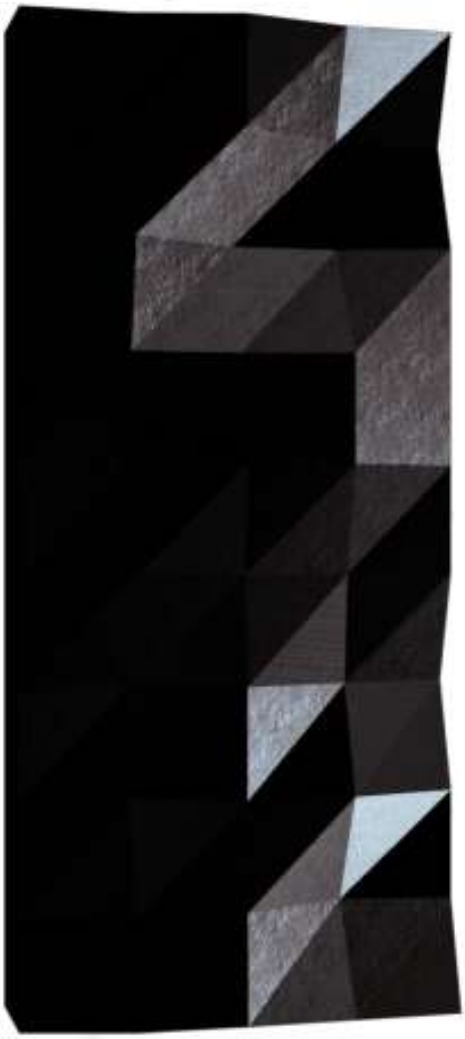

GIT - MODULE 00

Working directory, staging area, local repository, remote repository

Summary:

This document contains the exercises of module 00 of git piscine.

Version 1.1

Contents

I	**Introduction**	**2**
II	**General Rules**	**3**
III	**Instructions**	**4**
IV	**Exercise 00: making acquaintance**	**5**
V	**Exercise 01: .gitignore**	**6**
VI	**Exercise 02: Retreat!**	**7**
VII	**Exercise 03: Slicing commits**	**8**
VIII	**Exercise 04: Make it shine!**	**9**

Chapter I

Introduction

When Linus Torvalds got bored of dealing with his emails to patch new content on Linux he looked for other ways of doing it. Existing version control software was insufficient so he made his own.

These modules aim at making you a better git user. However, git being complex and dense software, we highly encourage you to go beyond these modules.

You are about to start a new journey on git.

Chapter II

General Rules

- This project will be evaluated by humans.
- No norm
- No compiler
- You do not have to turn in a `Makefile`
- The exercises are ordered from the easiest to the hardest.
- The exercise directories will be named this way: `ex00`, `ex01`, ... , `exn`
- You will respect the files to turn in and include an empty `notes` file if necessary.
- Your commits will always contain at least a title (one line).
- Your commits will always contain at least a description (several lines).

Chapter III

Instructions

- Fully read this document before starting working on the project.
- During the evaluation you will be asked to redo and explain what you did, so take notes.
- You are not allowed to use any shortcuts or aliases provided by your shell or shell theme.
- You are not allowed to use an existing repository you have to start from scratch.
- You are highly encouraged to try and make mistakes with git now rather than later.

Chapter IV

Exercise 00: getting acquainted

	Exercise: 00
Getting acquainted	
Turn-in directory: ex00/	
Files to turn in: script.sh, notes, *	
Forbidden functions: all shell functions	

Make sure that `git` knows who you are:

Write a script in file `script.sh` containing only git commands to list your name, email and your default git editor.

Then set your name to your login and your email to *yourlogin*@student.42.fr

Can you tell the difference between system, global and local scopes?

Chapter V

Exercise 01: .gitignore

	Exercise: 01
.gitignore	
Turn-in directory: ex01/	
Files to turn in: notes ../.gitignore	
Forbidden functions: none	

Someone working with you on an imaginary project keeps creating test files starting with the name of the current directory and followed by random generated characters.

Create the following files in your turn-in-directory to simulate your mate's behaviour:

- `ex01thisisanexample`
- `ex01aweijawek`
- `ex01244466666`
- `ex01dQw4w9WgXcQ`

You obviously do not want any of these files showing up in your repository.

Listing those files one by one would be impossible given the fact that the only rule your mate seems to be following is to create a file starting with the name of your directory.

You will therefore create a rule that automatically prevents all files starting with `ex01` to be added in your staging area.

Are you sure your new rule does not cause any other trouble?

Chapter VI

Exercise 02: Retreat!

	Exercise: 02
Retreat!	
Turn-in directory: ex02/	
Files to turn in: notes	
Forbidden functions: all shell functions	

Now that your `.gitignore` is properly set up, you may get a bit too used to it and abuse the "`git add .` " command. Therefore you may end up adding wrong files by mistake.

Create a `git unstage` command. In effect, "`git unstage filename`" removes the file `filename` from the staging area. Except being removed from the staging area, no changes are made to the file whatsoever.

Chapter VII

Exercise 03: Slicing commits

	Exercise: 03
Slicing commits	
Turn-in directory: ex03/	
Files to turn in: notes,	
Forbidden functions: all shell functions	

Familiarize yourself familiar with the staging area, find out why the following is possible.

```
On branch master
Your branch and 'origin/master' have diverged,
and have 1 and 1 different commits each, respectively.
  (use "git pull" to merge the remote branch into yours)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   toto

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:   toto
```

It is a good practice to commit regularly but sometimes you forget about it and get carried on. Let's say you modified line 15 to 20 on a file for feature one, while changing line 42 to fix an issue, your *42header* keeps track of the last update.

Without modifying the file in between, using git only:

- commit changes line 15 to 20 without line 42, also commit the changes done to the header.
- commit changes line 42 the header should not change.

Chapter VIII

Exercise 04: Make it shine!

	Exercise: 04
Make it shine!	
Turn-in directory: ex04/	
Files to turn in: notes	
Forbidden functions: all shell functions	

- Create a `git hist` function.
- Figure out a way for the output to look like this
  - the date must be the commit date.
  - the login must be the commit author's login
  - the colours may vary, however references cannot be one solid colour.

```
* 9b695f0 18h59 03/09/2022 aweaver  (HEAD -> master, origin/master) ADD EX04:
* b538e25 18h46 03/09/2022 aweaver  ADD EX03:
* 0dc6d9e 18h54 03/09/2022 aweaver  UPDATE EX02:
* f239fb7 18h46 03/09/2022 aweaver  ADD EX02:
* dbaa082 14h16 03/09/2022 aweaver  ADD EX01:
* 0fca18d 14h09 03/09/2022 aweaver  ADD EX00:
```

What exactly are those characters in the first column?