Explications :

Git est un outil de versionnage, c'est-à-dire que lorsque Git est **attaché** à un dossier contenant tous les fichiers sur lesquels on travaille, ce dossier est appelé « **Repository** ». Dans ce Repository, Git enregistre tous les changements qui ont été réalisés au cours du temps.

Git permet également de synchroniser son Repository sur son ordinateur avec un Repository sur internet. Cela fonctionne comme un miroir, c'est-à-dire que tous les fichiers mis dans le Repository sur internet vont apparaître également dans son Repository sur son ordinateur.

Durant notre premier cours, nous avons pu récupérer tout le Repository sur internet stocké sur un serveur d'un site web appelé GitHub pour les mettre dans ton Repository sur le chemin suivant : *Documents/godotproject/.*

Comment s'en servir

Pour utiliser l'outil Git, il faut écrire des **lignes de commandes** dans un **Terminal.**

Un Terminal est un outil très puissant propre à chaque système d'exploitation comme Windows, Linux ou MacOS par exemple. Avec un Terminal, on peut par exemple parcourir des dossiers, en créer de nouveaux, naviguer sur internet. Mais alors pourquoi utiliser un Terminal si on peut également faire ces mêmes choses avec l'**interface graphique** ? Tout simplement parce que le Terminal fait beaucoup plus de choses que ce que l'on peut faire habituellement avec notre interface graphique.

Figure 1 : Exemple d'interface graphique Windows 10

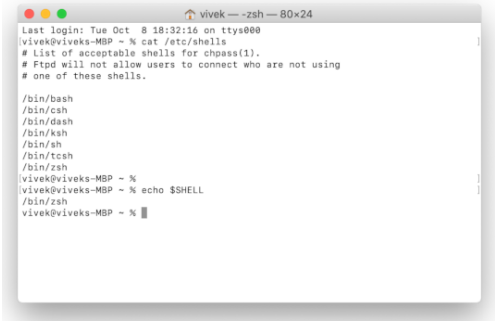

Figure 2: Exemple de Terminal sous MacOS

Commandes du Terminal

Avant d'utiliser Git, voici les quelques commandes qu'il faut savoir pour maitriser un minimum le Terminal :

- *ls* : c'est un terme raccourci pour du mot *list*. Permet de voir la liste de fichiers et de dossiers à l'endroit où l'on est.

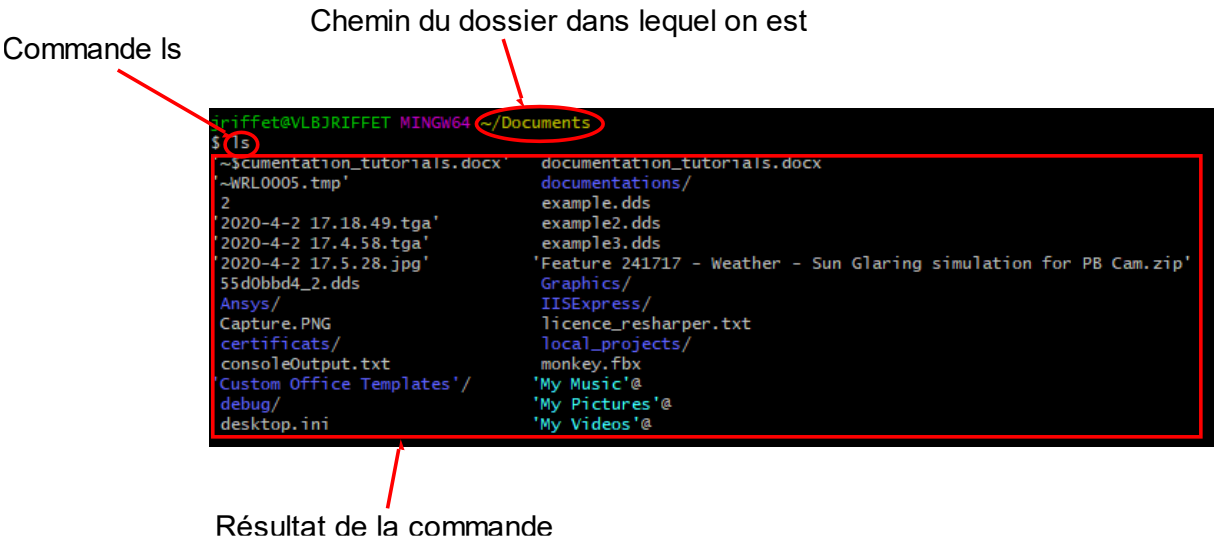

Figure 3: Voici l'exécution de la commande ls. On peut voir que le résultat de cette commande est une liste de tous les fichiers et dossiers contenus dans Documents.

- *cd* : c'est un terme raccourci pour du mot *Change Directory*. Il permet de se déplacer de dossier en dossier. C'est l'équivalent de double cliquer sur un dossier pour rentrer dedans. Pour aller vers un dossier spécifique comme par exemple le dossier *cours_programmation*, on fera alors la commande *cd cours_programmation*. Pour aller vers un dossier parent comme par exemple pour revenir au dossier *Perso* depuis le dossier *cours_programmation,* on pourra écrire *cd ..* Les *..* indiquent le chemin du dossier parent.

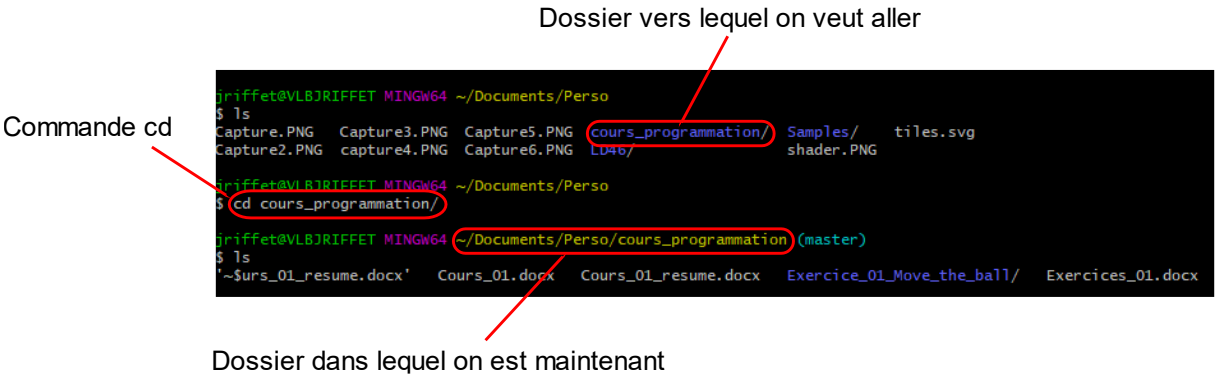

Figure 4: Exemple pour la commande cd. On veut se déplacer vers le dossier "cours_programmation"

Voici les deux premières commandes avec lesquelles on peut aller partout dans les fichiers de l'ordinateur sans passer par l'interface graphique. Nous verrons plus tard quelles autres commandes peuvent nous être utiles par la suite.

Commandes Git

Nous allons voir ici, seulement une commande, c'est la commande : *git pull*. Cette commande permet tout simplement de mettre à jour tous les fichiers qui sont dans le Repository. Lorsque de nouveaux exercices, cours, … sont mis dans le Repository en ligne, alors il faudra synchroniser son Repository local avec cette commande.

```
jriffet@VLBJRIFFET MINGW64 ~/Documents/Perso/cours_programmation (master)
$ git pull
Already up to date.
```

Figure 5 : Exemple de commande avec git pull. Dans ce cas-là, le Repository est déjà à jour. C'est à dire que rien n'a besoin d'être téléchargé du Repository distant (sur GitHub)

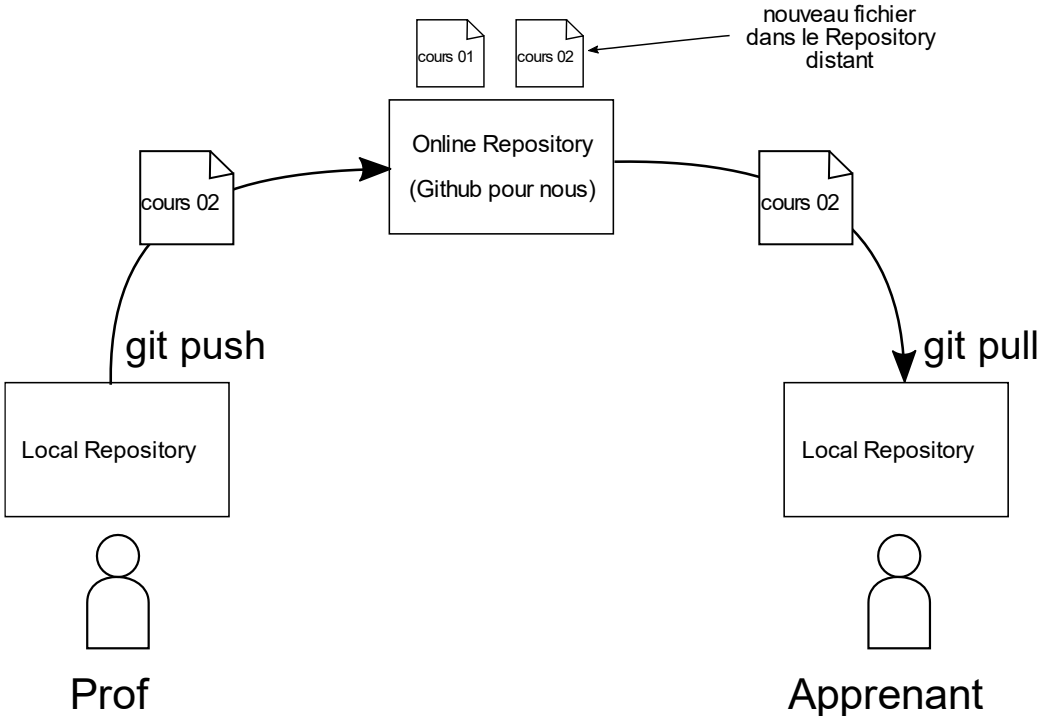

Figure 6: Exemple de commande avec git. Quand le prof a fait des cours supplémentaires comme le cours 02. Il va envoyer le fichier cours_02 sur le Repository distant avec la commande git push (nous parlerons de cette commande plus tard), Le repository distant est donc mis à jour avec les nouveaux fichiers. L'apprenant n'a plus qu'à récupérer les nouveaux fichiers sur son ordinateur en faisant la commande git pull.