2N6 Programmation 2

python™

GIT

Système de gestion de code source

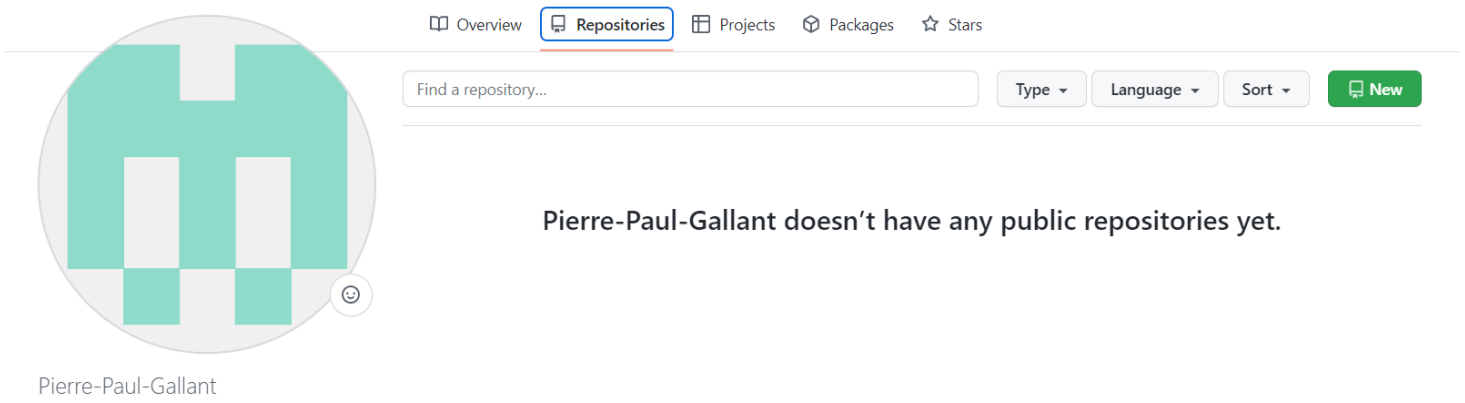
https://info.cegepmontpetit.ca/git

Système de gestion de code source

- Un système de gestion de versions permet de garder une trace de toutes les modifications faites dans un fichier, avec une date et une explication de ce qui a été fait.
- Même lorsqu'on travaille seul, c'est très utile car en sauvegardant votre code à un certain intervalle de temps, vous ne risquez pas de perdre votre travail.
- Cela évitera aussi d'avoir à enregistrer et déposer votre travail dans Teams à la fin du cours car en ajoutant votre professeur comme collaborateur, il pourra voir votre travail au besoin.
- Facilite le travail d'équipe. Le système gestion de versions permet d'éviter la suppression accidentelle de code et de résoudre les conflits facilement.

Système de gestion de code source

- https://education.github.com/pack/join
- Utilisez le compte email de l'école
  - matricule@cegepmontpetit.ca
- Vous devez fournir une preuve de votre statut étudiant
  - (une capture d'écran de votre grille de cheminement dans LÉA)
- Créez un nouveau repo pour cette rencontre



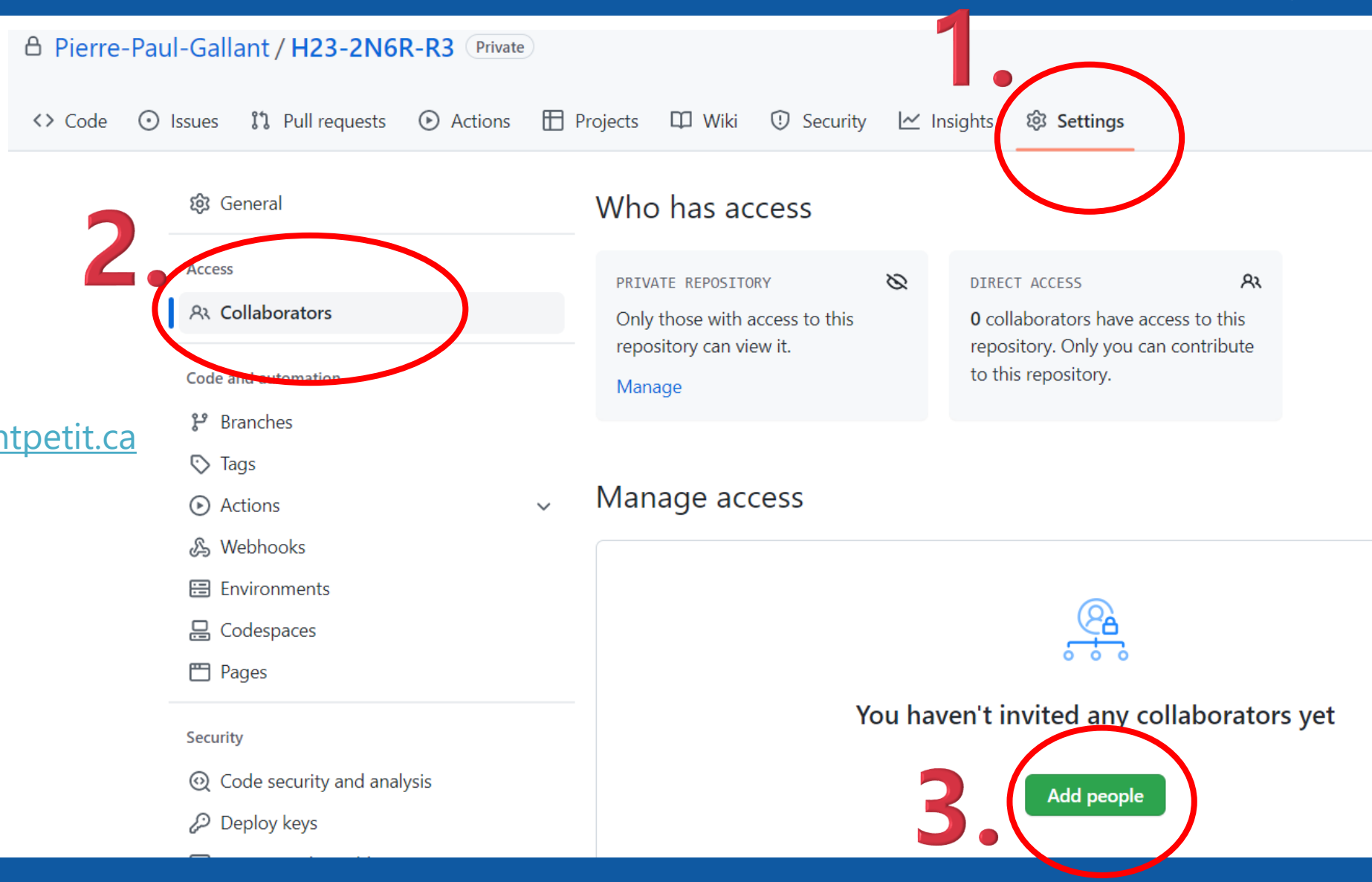
Système de gestion de code source

On ajoute les collaborateurs (le professeur dans ce cas-ci)

pierre-paul.gallant@cegepmontpetit.ca

Ou

Pierre-Paul-Gallant

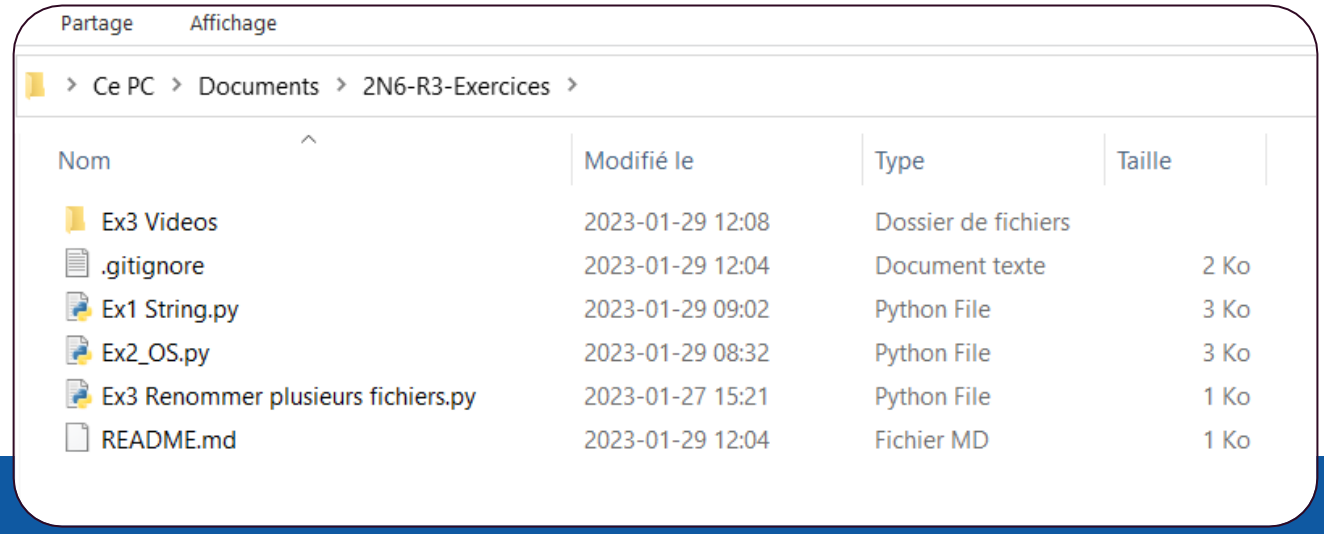


Système de gestion de code source

- Ouvrez le répertoire documents dans le terminal cmd ou PS
- Utilisez la commande :
  git clone ADDRESSE_DU_GIT nom_du_répertoire_local

```
t\Documents> git clone https://github.com/Pierre-Paul-Gallant/H23-2N6R-R3 2N6-R3-Exercices
```

- Téléchargez les fichiers de départ depuis Teams et ajoutez-les au répertoire ainsi créé.



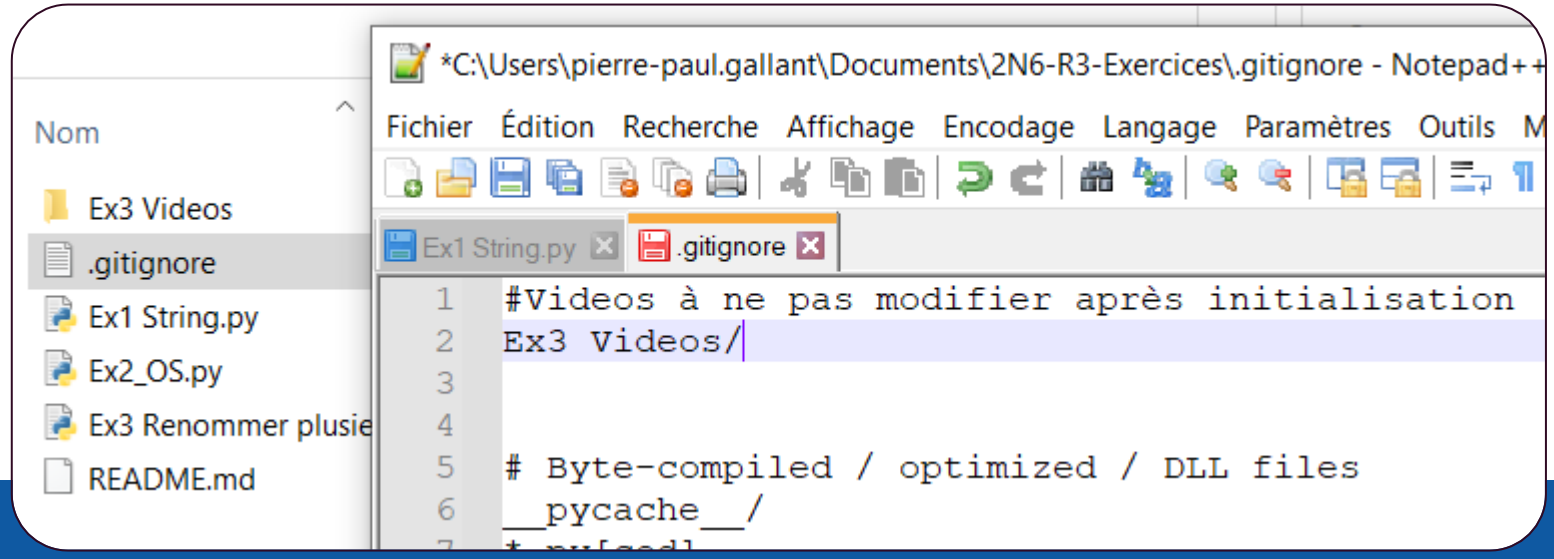
Système de gestion de code source

- On ouvre un terminal dans le répertoire.
- En ligne de commande :

```
\Documents\2N6-R3-Exercices> git add *
```

  - Afin d'ajouter les fichiers au contrôle de code source.

- On ne veut pas modifier les fichiers videos, on va donc ouvrir le fichier .gitignore et y ajouter le répertoire contenant les videos.



Système de gestion de code source

- On veut maintenant ajouter les fichiers au dépôt en ligne.
- On fait un " git commit " avec un message informatif.

```
s\2N6-R3-Exercices> git add *
s\2N6-R3-Exercices> git commit -m "Initialisation"
```

- Suivi d'un " git push " pour envoyer le tout au dépôt en ligne

```
ts\2N6-R3-Exercices> git push
```

Utilisation de git

- Si on travaille avec le même dépôt en ligne à partir d'un autre ordinateur et ou d'une autre session de travail.

```
t\Documents> git clone https://github.com/Pierre-Paul-Gallant/H23-2N6R-R3 2N6-R3-Exercices
```

- **SI** on travaille à plusieurs, on commence par faire un "git pull" même si le répertoire existe déjà. (afin d'avoir les dernières modifications de nos collègues)

```
PS C:\Users\pierre-paul.gallant\Documents\2N6-R3-Exaette> git pull
Already up to date.
PS C:\Users\pierre-paul.gallant\Documents\2N6-R3-Exaette>
```

Utilisation de git

- Lorsqu'on fait des modifications, on doit ajouter les fichiers que l'on veut "commit"
  - Pour un seul fichier :

```
s\2N6-R3-Exercices> git add '.\Ex1 String.py'
```

  - Pour tous les fichiers modifiés:

```
\2N6-R3-Exercices> git add -A
```

- Puis on fait un git commit avec un message court et informatif :

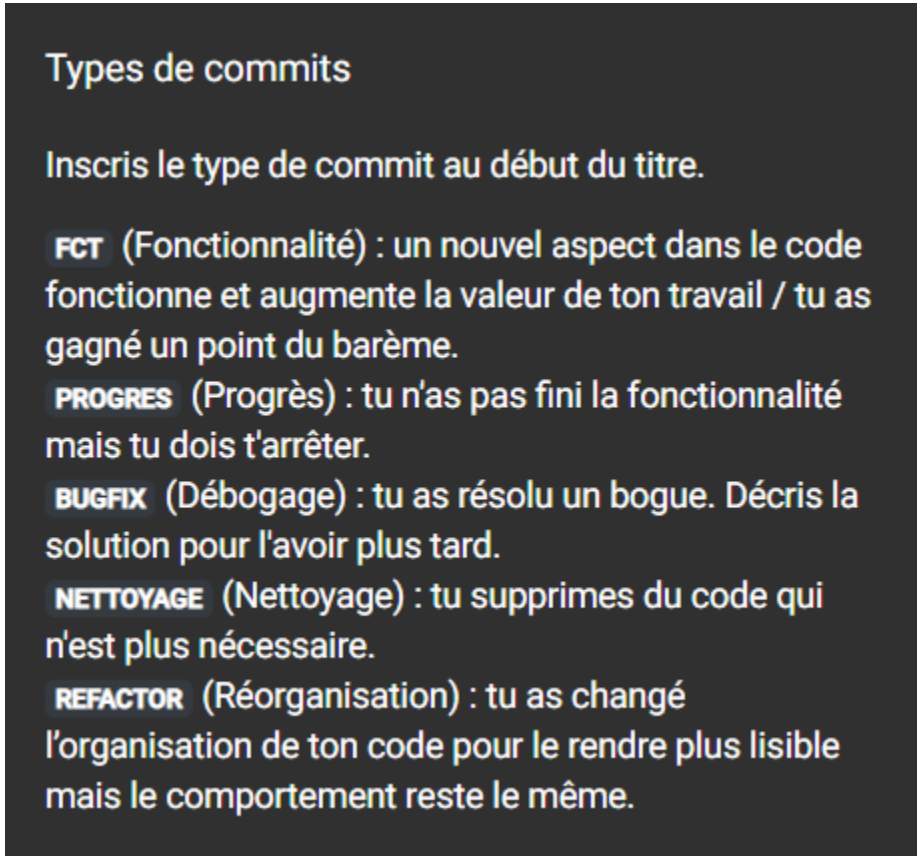
```
\2N6-R3-Exercices> git commit -m "Q1 terminé"
```

- Enfin, on fait un git push pour envoyer le commit sur le dépôt en ligne :

```
s\2N6-R3-Exercices> git push
```

GIT

- Dans ce cours :
- On fait un commit / push pour **CHAQUE** exercice terminé avec le message commençant par FCT
- S'il y plus de 30 minutes de travail depuis votre dernier commit… faites un commit / push avec le message commençant par PROGRES



https://info.cegepmontpetit.ca/git

Approfondissement des strings

Ex1 – string.py

Les strings

- Les objets « str » peuvent être considérés comme des listes de caractères.
- Mais ils sont immuables (ne peuvent pas être changés). Les sous strings générés sont de nouveaux objets.
- Toutes les opérations de « slicing » peuvent être effectuées sur des « strings ».

```
>>> liste1 = "Bonjour le monde"
>>> print(liste1)
Bonjour le monde
```

```
>>> print(liste1[0:5])
Bonjo
>>> print(liste1[:5])
Bonjo
```

```
>>> print(liste1[2:])
njour le monde
```

```
>>> print(liste1[-1])
e
>>> print(liste1[-5:])
monde
```

Les méthodes des strings

- encode() change l'encodage des caractères du string

```
>>> cours = "réseau 1"
>>> cours.encode("utf-8")
```

- find() et index() retournent l'index du premier caractère de la séquence trouvée

```
>>> print(cours.find("eau"))
3
```

- Les méthodes is... retournent une valeur booléenne indiquant si le string correspond à ce que la méthode vérifie.

```
>>> "3".isdigit()
True
```

Les méthodes des strings

- split() / rsplit() retournent les portions de strings en séparant le string initial selon un caractère passé en paramètre.
  - Peut-être capturé dans une seule variable sous forme de liste.
  - Peut aussi être capturé dans plusieurs variables distinctes

```
>>> cours = "réseau1/prog1/prog2/prog3"
>>> liste_cours = cours.split("/")
>>> print(liste_cours)
['réseau1', 'prog1', 'prog2', 'prog3']
>>>
```

```
>>> cours = "réseau1/prog1"
>>> cours1, cours2 = cours.split("/")
>>> print(cours1)
réseau1
>>> print(cours2)
prog1
>>>
```

Les méthodes des strings

- lstrip() retire les espaces au début du string (left-strip)
- rstrip() retire les espaces à la fin du string (right-strip)
- strip() retire les espaces au début et à la fin

```
>>> x = ("  foo  ")
>>> x.rstrip()
'  foo'
>>> x.lstrip()
'foo  '
>>> x.strip()
'foo'
```

Les méthodes des strings

- zfill() ajoute des zéros à gauche d'un string pour obtenir un string d'une taille prédéfinie

```
>>> y = ["1-foo","2-bar","10-byr"]
>>> print(y.sort())
None
>>> y = ["1-foo","2-bar","10-byr"]
>>> y.sort()
>>> print(y)
['1-foo', '10-byr', '2-bar']
```

```
>>> for i in y : x.append(i.zfill(6))
...
>>> print(x)
['01-foo', '10-byr', '02-bar']
>>> x.sort()
>>> print(x)
['01-foo', '02-bar', '10-byr']
>>>
```

Introduction aux modules

Ex-2 os.py

Le module os

- os.chdir() : change le répertoire dans lequel l'interprétateur « agit »
- os.getcwd() : affiche le répertoire courant
- os.listdir() :liste les réteroires et fichiers
- os.mkdir() : crée un répertoire
- os.makedirs() : crée répertoire(s) et sous-répertoire(s)
- os.rmdir() : supprime un répertoire VIDE

Le module os

- os.environ : dictionnaire des variables de système
  - getenv() et putenv() pour obtenir ou modifier ces variables
- os.remove() : supprime un fichier
- os.rename() : renomme un fichier ou répertoire
- os.path() : sous-module pour travailler avec les paths
- os.syteme() : exécute des commandes dans le shell
- os.stat() : donne des informations sur le fichier / répertoire passé en argument