Développement du projet - Partie 1

Objectifs pédagogiques

- Comprendre l'importance de Git et GitHub dans la gestion de projets collaboratifs
- Savoir initialiser un dépôt, commiter ses fichiers, utiliser les branches
- Être capable de documenter le code (selon PEP257)
- Appliquer les bonnes pratiques de versionnement sur leur projet "ToDo List"

1. Qu'est-ce que Git ?

Git est un **système de gestion de versions distribué**. Il permet de :

- Suivre l'évolution de vos fichiers dans le temps
- Travailler à plusieurs sans conflits
- Revenir en arrière facilement
- Travailler en parallèle grâce aux **branches**

2. Qu'est-ce que GitHub ?

GitHub est une **plateforme d'hébergement de projets Git** :

- Dépôts accessibles en ligne
- Collaboration facilitée (issues, pull requests, commentaires)
- Sauvegarde dans le cloud

Principales commandes :

bash[1]

```
git init                    # Initialiser un dépôt
git add .                   # Ajouter tous les fichiers modifiés
git commit -m "message"     # Enregistrer un snapshot avec un message
git status                  # Voir l'état du dépôt
git log                     # Historique des commits
```

3. Installation de Git

- Télécharger Git depuis [git-scm.com](git-scm.com)

[1] Le BASH est un langage programmation shell intégré à Linux

- Configurer Git :

bash

```
git config --global user.name "VotreNom"

git config --global user.email "VotreEmail@example.com"
```

4. Commandes essentielles de Git

Commande	Description
`git init`	Initialise un dépôt Git local
`git add .`	Suit tous les fichiers dans le projet
`git commit -m "msg"`	Enregistre les changements avec un message
`git status`	Affiche les fichiers modifiés ou en attente
`git log`	Affiche l'historique des commits

5. Travailler avec GitHub

1. Créer un compte sur github.com
2. Créer un nouveau dépôt
3. Lier le projet local à GitHub :

bash

```
git remote add origin https://github.com/VotreNom/nom-du-repo.git

git push -u origin main
```

6. Branches et collaboration

Commande	Action
`git branch ma-branche`	Crée une branche
`git checkout ma-branche`	Bascule vers cette branche

`git merge ma-branche`	Fusionne avec la branche principale
`git push origin ma-branche`	Envoie la branche sur GitHub
`git pull origin main`	Met à jour son dépôt local
`git clone https://...`	Cloner un dépôt existant

7. Documentation du code

La documentation permet à d'autres (ou vous-même plus tard) de comprendre rapidement le fonctionnement du programme.

- **Docstring Python** (convention PEP 257)

```
class Task:
    """
    Représente une tâche dans la todo list.

    Attributs :
        title (str) : Titre de la tâche
        done (bool) : Statut de la tâche (terminée ou non)
    """

    def __init__(self, title):
        """
        Initialise une nouvelle tâche avec un titre donné.
        """
        self.title = title
        self.done = False
```

🛠 Travaux Pratiques

Exercice 1 : Initialisation du dépôt Git

1. Créez un dossier `todolist`.
2. Initialisez un dépôt Git avec git `init`.
3. Créez les fichiers de base : `main.py, task.py, README.md`
4. Ajoutez-les et faites un premier commit.

```
git init
touch main.py task.py README.md
git add .
git commit -m "Initialisation du projet ToDo List"
```

Exercice 2 : Documentation du code

1. Créez une classe `Task` dans `task.py`
2. Ajoutez une documentation en docstring pour la classe et ses méthodes.
3. Faites un commit des modifications.

- Documentation officielle Git
- Vidéo – Git et GitHub pour les débutants (Grafikart)
- Formation interactive Git sur LearnGitBranching
- Cheat Sheet Git GitHub PDF
- OpenClassrooms – Gérer son code avec Git et GitHub
- Créer son compte et son premier dépôt GitHub
- GitHub et Git – Tutoriel complet
- PEP 257 – Docstring Conventions (en anglais)
- Comment bien documenter son code Python
- https://youtu.be/X3KCX99I2pQ?si=aL6uZvx_EKsCKxSU