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
- Installer Git (laisser les options par défaut)

[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"
```

Et Ensuite vérifiez votre configuration avec **git config --list**.

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

- ❖ Créer un compte sur github.com
- ❖ Créer un nouveau dépôt
- ❖ Lier le projet local à GitHub

Étape 1 : Créer un compte GitHub

- Rendez-vous sur https://github.com
- Cliquez sur "Sign up"
- Renseignez vos informations (email, mot de passe, nom d'utilisateur)
- Validez votre compte via l'email de confirmation

Étape 2 : Créer un dépôt GitHub

- Cliquez sur "New repository"
- Donnez-lui un nom, par exemple ToDoList
- Laissez les options par défaut (ne pas cocher README pour éviter les conflits)
- Cliquez sur "Create repository"

Étape 3 : Lier le dépôt local à GitHub

Un dépôt distant est une version de ton projet stockée sur un serveur, généralement sur GitHub.

Cela permet :

❑ de sauvegarder son code en ligne

❑ de travailler sur plusieurs appareils

❑ de collaborer avec d'autres développeurs

Depuis votre terminal dans le dossier du projet :

bash

```
git init
git add .
git commit -m "Initial commit"
git branch -M main
git remote add origin https://github.com/VotreNom/ToDoList.git
git push -u origin main
```

Authentification : si nécessaire, utilisez un token GitHub (voir : https://github.com/settings/tokens)

6. Branches et collaboration

Création des Branches

Une branche en Git est une version parallèle du code dans un projet. Elle permet de travailler sur de nouvelles fonctionnalités ou corrections sans affecter le code principal. La branche par défaut dans Git quand vous créez un dépôt s'appelle

main.

Pourquoi des branches ?

- Travailler sur des fonctionnalités sans affecter main.
- Pouvoir tester différentes implémentations d'une même fonctionnalité de manière indépendant
- Permet de tester et d'éviter les conflits

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

Cloner un Dépôt GitHub

Vous pouvez récupérer un projet existant depuis GitHub grâce à la commande

git clone https://github.com/VotrePseudo/mon_projet.git

Et Pour mettre à jour vos dépôts local

git pull origin le nom de la branche à modifier

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.

Ressources complémentaires

- Documentation officielle Git : https://git-scm.com/doc
- Tutoriel vidéo : Git et GitHub pour les débutants (Grafikart)
- Formation interactive : https://learngitbranching.js.org
- Fiche mémo Git : https://education.github.com/git-cheat-sheet-education.pdf
- OpenClassrooms – Gérer son code avec Git et GitHub
- Tutoriel GitHub complet : https://guides.github.com
- Conventions de documentation PEP 257 : https://peps.python.org/pep-0257/
- Vidéo : https://youtu.be/X3KCX99I2pQ?si=aL6uZvx_EKsCKxSU