Cycle de vie d'un projet informatique

TP1

Objectifs du TP

Appréhender différents outils liés à la gestion d'un projet informatique que nous avons vu en cours..

---

1 Tableaux Kanban - Trello

Question 1 : Ouverture de compte et création d'un tableau

Si vous n'avez pas déjà un compte Trello, créez-en un et créez ensuite un nouveau tableau

Question 2 : Ajout de listes

Si elles n'existent pas, créez trois listes intitulées *À faire*, *En cours* et *Terminé*.

Question 3 : Ajout de cartes

Ajouter dans la liste *À faire*, toutes les tâches qui vous semblent nécessaires pour un logiciel de gestion de prêt dans une bibliothèque communale

Question 4 : Personnalisation des cartes

Pour chaque carte, ajoutez une description détaillée de la tâche, une date d'échéance et des étiquettes de couleur pour indiquer la priorité.

Question 5 : Invitation des membres de l'équipe

Invitez deux camarades de votre équipe à votre tableau Trello.

Question 6 : Attribution des tâches

Attribuez chaque carte à un ou plusieurs membres de l'équipe

Question 7 : Ajout de commentaires

Utilisez la fonction de commentaires pour communiquer avec les membres de l'équipe sur chaque carte

Question 8 : Utilisation des checklists

Créez des checklists dans les cartes pour décomposer les tâches en sous-tâches plus gérables

Question 9 : Mouvement des cartes

Afin de simuler un avancement du projet, déplacez quelques cartes dans les listes *En cours* et *Terminé* cartes vers les listes appropriées pour indiquer leur état actuel

Question 10 : Création de règles

Créez une règles pour déplacer automatiquement une carte dans la liste *En cours* lorsque sa date d'échéance est dans 48 heures.

Question 11 : Création de boutons

Créez un bouton *Done* qui permettent d'envoyer une carte dans la liste *Terminé*

Question 12 : En mode SCRUM

Renommez la liste *A faire* en *Product Backlog* et ajoutez une liste *Sprint Backlog*

Question 13 : Utilisation des labels pour la priorité

Utilisez des étiquettes de couleur pour indiquer la priorité de chaque élément dans le backlog.

Question 14 : Création de listes pour les fonctionnalités et les bugs

Créez des listes séparées pour les nouvelles fonctionnalités demandées et les bugs à résoudre.

2 Environnement - Conda

Question 1 : Installation

Installer Conda depuis le site https://conda.io/projects/conda/en/latest/user-guide/install/index.html

Question 2 : Vérification

Une fois installé, ouvrez votre terminal ou invite de commande et tapez `conda --version` pour vérifier que Conda est correctement installé.

Question 3 : Installation de paquet

Utilisez Conda pour installer le paquet numpy : `conda install numpy`

Question 4 : Vérification de paquet

Vérifiez la version installée de numpy avec la commande `conda list numpy`

Question 5 : Mise à jour de paquet

Mettez à jour le paquet à la dernière version disponible avec `conda update numpy`

Question 6 : Désinstallation de paquet

Désinstallez numpy avec la commande `conda remove numpy`

Question 7 : Création d'environnement

Créez un nouvel environnement Conda appelé `my_env` avec Python 3.9 : `conda create --name my_env python`

Question 8 : Activation d'environnement

Activez l'environnement avec `conda activate my_env`

Question 9 :

Pendant que `my_env` est activé, installez les paquets *numpy*, *pandas*, et *matplotlib*

Question 10 : Export d'un environnement

Exportez cet environnement dans un fichier *environment.yml* avec la commande `conda env export > enviro`
Regardez le contenu du fichier *environment.yml*

Question 11 : Désactivation d'environnement

Désactivez l'environnement avec `conda deactivate`

Question 12 : Suppression d'environnement

Supprimez l'environnement *my_env* avec `conda remove --name my_env --all`

Question 13 : Import d'un environnement

Recréez *my_env* en utilisant le fichier *environment.yml* avec `conda env create -f environment.yml`

3 IDE - Pycharm

Question 1 : Installation

Téléchargez et installez PyCharm depuis le site officiel. En tant qu'étudiant vous pouvez avoir un compte "pro" JetBrains gratuitement pour un an.

Question 2 : Créer un nouveau projet

Créez un nouveau projet Python en choisissant un interpréteur Python installé sur votre système. Choisissez celui de *conda*.

Question 3 : Découverte de PyCharm

Familiarisez-vous avec l'interface utilisateur de PyCharm. Identifiez les différentes parties de l'IDE telles que l'éditeur, l'explorateur de projets, etc.

Question 4 : Premier fichier python

Dans votre nouveau projet, créez un nouveau fichier Python (extension .py). Écrivez un script Python simple, par exemple, un script qui imprime "Hello world!" à la console

Question 5 : Exécution

Exécutez votre script en cliquant sur le bouton "Run" dans la barre d'outils

Question 6 : Création d'un environnement virtuel

Apprenez à créer un environnement virtuel dans PyCharm avec conda

Question 7 : Ajouter un package dans un environnement

Installez un package Python externe dans votre environnement virtuel conda à l'aide de l'outil PyCharm pour gérer les packages, par exemple numpy.

Question 8 : Utilisez le package d'un environnement

Utilisez le package installé dans un script Python d'une dizaine de lignes

Question 9 : Ajouter un breakpoint

Ajoutez une pause (breakpoint) à une ligne de votre script Python

Question 10 : Exécution en mode debug

Exécutez le script en mode débogage (Debug) et observez comment PyCharm s'arrête à la pause

Question 11 : Inspection des variables

Utilisez les outils de débogage pour inspecter les valeurs des variables et avancer dans le script étape par étape

Question 12 : Refactoring

Utilisez les outils de refactoring de PyCharm pour améliorer le code. Par exemple, vous pourriez renommer des variables pour qu'elles aient des noms plus descriptifs.

4 Gestion de version - Git(hub)

Question 1 : Installation

Installez Git si vous ne l'avez pas déjà fait. Vous pouvez suivre les instructions sur le site officiel de Git.

Question 2 : Configurer nom/email

Configurez votre nom d'utilisateur et votre adresse e-mail à l'aide des commandes suivantes :
— `git config --global user.name "Votre nom"`
— `git config --global user.email "votre@email.com"`

Question 3 : Création d'un dépot

Créez un nouveau répertoire pour votre projet et naviguez-y à partir du terminal puis initialisez un nouveau dépôt git avec la commande `git init`.
Si vous avez un dépot existant sur un repository distant, vous pouvez y associer ce nouveau dépot avec la commande `git remote add origin <URL_de_votre_repo>`.

Question 4 : Clonage d'un dépôt

Si vous avez un dépôt en ligne, clonez-le en utilisant la commande `git clone <URL_de_votre_depot>` (équivalent au `git init` + `git remote add origin`)

Question 5 : Ajoutez un fichier

Créez un nouveau fichier dans votre répertoire de projet (par exemple, `readme.md`) et ajoutez un peu de contenu à l'intérieur. Utilisez `git add readme.md` pour l'ajouter à la gestion de version.

Question 6 : Faire un commit

Utilisez `git commit -m "Ajout du fichier readme"` pour commettre le changement avec un message de commit.

Question 7 : Création d'une branche

Créez une nouvelle branche avec la commande `git branch nom_de_la_branche`.

Question 8 : Changement de branche

Basculez vers votre nouvelle branche avec `git checkout nom_de_la_branche`. Apportez quelques modifications à votre fichier, puis faîtes un commit.

Question 9 : Fusion de branches

Revenez à la branche principale avec `git checkout main` (ou `master` si votre branche principale est nommée "master"). Fusionnez votre branche secondaire dans la branche principale avec `git merge nom_de_la_b`
Résolvez tous les conflits de fusion, si nécessaire, puis faîtes un commit.

Question 10 : Envoyez les modifications vers le dépot distant

Poussez vos changements vers le dépôt distant avec `git push -u origin main` (remplacez `main` par le nom de votre branche si nécessaire)

Question 11 : Mise à jour du dépôt local

Simulez une collaboration en effectuant des modifications directement sur votre dépôt en ligne (par exemple, via l'interface GitHub). Sur votre machine locale, utilisez `git pull` pour récupérer les derniers

changements du dépôt distant.

Question 12 : Voir l'historique

Utilisez `git log` pour voir l'historique des commits de votre dépôt

Question 13 : Annuler des changements

Utilisez la commande `git reset HASH_DU_COMMIT` pour réinitialiser l'historique de votre branche à un état précédent (remplacez `HASH_DU_COMMIT` par le hash réel du commit vers lequel vous souhaitez revenir)

Question 14 : Tag de version

Faites un commit qui représente une version stable de votre projet. Utilisez `git tag -a v1.0 -m "Ma première` pour taguer ce commit comme une version stable. Poussez ensuite vos tags vers le dépôt distant avec `git push origin --tags`

Pour aller plus loin :

Vous pouvez regarder ce que font les commandes `git rebase` et `git bisect`

Hors de la ligne de commande :

Il existe des applications permettant de gérer graphiquement vos dépôts comme GitHub Desktop pour les dépots sur GitHub. Et il est également possible de les intégrer directement dans un IDE, regardez comment intégrer la gestion de git directement dans PyCharm (par exemple dans la documentation de PyCharm).

5 Tests unitaires - PyTest

Pour intégrer PyTest dans PyCharm, je vous invite à lire la documentation de PyCharm et plus précisément cette page https://www.jetbrains.com/help/pycharm/pytest.html. Créez quelques tests.

6 Documentation - Sphinx

Rien de tel pour comprendre comment faire une documentation que de lire la documentation du système de documentation : https://www.sphinx-doc.org/fr/master/. Créez un projet de documentation.