Démarrer avec VSC et GIT

Table des matières

1 Prérequis....2
1.1 Creation compte personnel Git....2
1.2 Prise en main VSC et Git....2
1.2.1 overview....2
1.2.2 video: Using git with vsc....2
1.2.3 video: commits in visual studio code....5
1.2.4 Video Git: branches in Visual Studio Code....9
1.3 Bonus: video Getting Started with Python in VS Code (Official Video)....9
1.4 Annexe....10
1.4.1 Comment désactiver la subrillance des caractères accentués:....10
1.4.2 Comment lire directement les .pdf dans VSC....10
1.4.3 Profile vsc....11

1 Prérequis

1.1 Creation compte personnel Git

On suppose qu’un compte github a été créé

https://docs.github.com/fr/get-started/start-your-journey/creating-an-account-on-github

1.2 Prise en main VSC et Git

1.2.1 overview

https://code.visualstudio.com/docs/sourcecontrol/overview

1.2.2 video: Using git with vsc

https://www.youtube.com/watch?v=i_23KUAEtUM

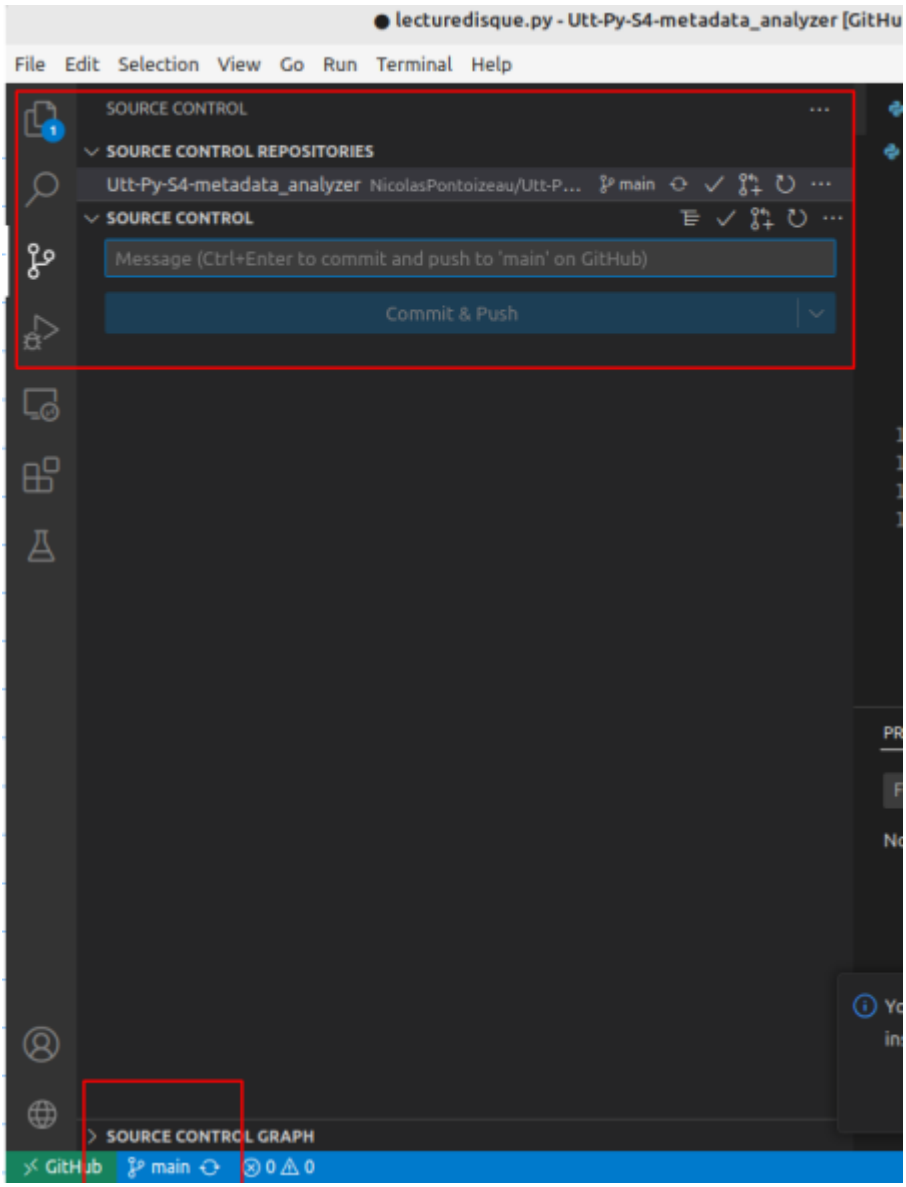
points clés de la vidéo :

0:35 Initialize repository

pour refleter son repertoire en local avec son repository git => remarque en bas à gauche on travaille sur l branche "main" avec une étoile

concretement vsc passe la commande git init

cette commande va créer un nouveau dépôt Git dans un répertoire en local sur la machine prêt à recevoir des fichiers et des commits.



0:55 Rename branch

1:25 Staging files

untracked files = a file that is new or changed but has not been added to the repository => (lettre U en face . Sous changes clicker sur + pour “stage” et la lettre en face devient A puis commit pour ajouter le nouveau ficher dans le repo.

(Si c’est un fichier existant qui a été modifié la lettre M apparait en face)

2:00 Committing files

2:10 Create new branch

on cree une branche pour ajouter des features à l’application

CTRL +P Git : create branch <nom_de_la_feature>

NB En bas à gauche le nom de la branche a changé en « nom_de_la_feature »

NB **les commits suivants se feront dans la branch <nom_de_la_feature> et n'affecteront pas main**

2:40 Gutter overview

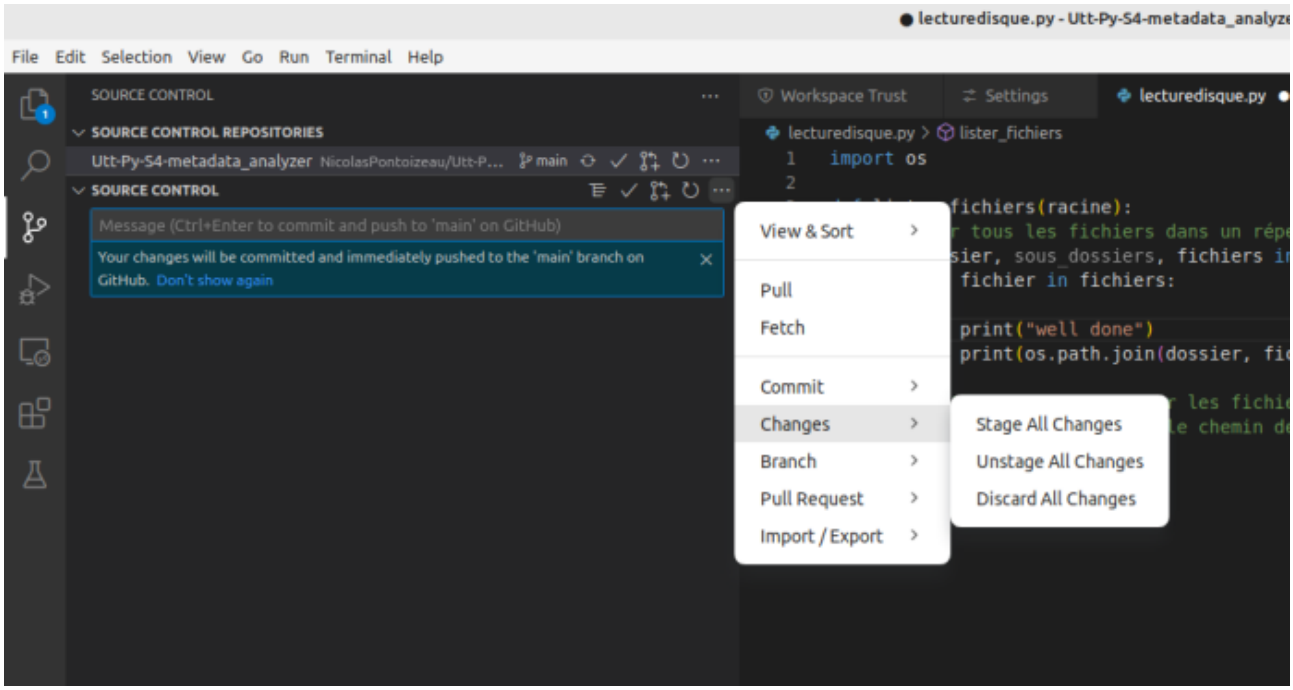
Quand on ajoute des lignes de code a un fichier elles apparaissent en vert dans la marge (gutter area)

Quand on modifier une ligne existante elles apparaissent en shadded blue dans la marge

Quand on supprime des lignes elles apparaissent avec une flèche rouge

On peut traiter les changements en masse :

- discard all my changes (activity bar sous « changes » fleche retour en arriere pour rejeter tous les changements)
- stage all my changes tous (activity bar sous « changes » puis signe + remarque que les fichiers sont maintenant dans la section « stages » mais ne sont plus dans la sections « changes »)



- commit des changement : au dessus de la section stages **mettre un message qui decrit les modifications pour informer les autres de ce contient le commit puis click sur la check mark en form de v**

3:30 Comparing files / inline view

- Selectionner un fichier qui été modifié dans la section « changes » pour comparer les fichiers 2 à deux entre la version commitée sur le repo et la version en local
- la vue inline consolide les changements dans le fichier. La vue inline en haut à droite click sur les … (ellipsis) puis « inline view »

4:30 Merging branches

Les nouvelles features sont faites sur des branches dédiées en dehors de main. Une fois matures il faut les merger (fusionner) à la branche main.

Dans la side bar du source control clicker sur les …(ellipsis) puis branch puis merge branch puis selectionner la branch à merger sur le main

5:00 Publish to Github

Dans la side bar du source control clicker sur « Publish Branch » (rectange bleu) pour publier la branch dans le repo distant sur git hub

NB la 1ere fois qu'on associe compte git hub avec son repo local il faut s'authentifier et autoriser l'extension VSC Git hub à ouvrir notre url de compte git hub à s'enregistrer à son compte git hub. En retour on autorise git hub à ouvrir VSC)

6:00 Clone repository

pour récupérer un repo distant sur git hub (ctrl palettepuis git : clone from git hub puis coller l'url puis selectionner un repertoire en local où stocker le repo chargé)

6:20 Summary

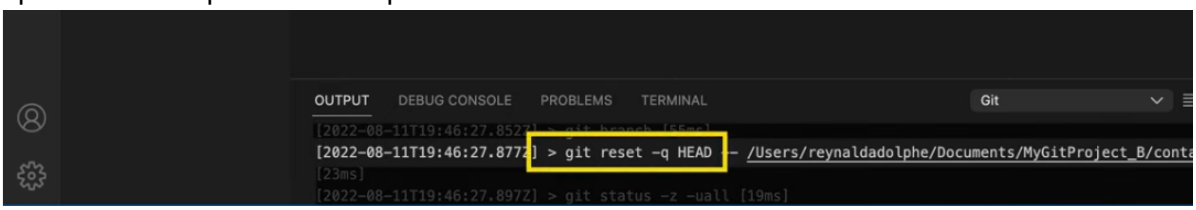
plus de details:

https://code.visualstudio.com/docs/sourcecontrol/overview

1.2.3 video: commits in visual studio code

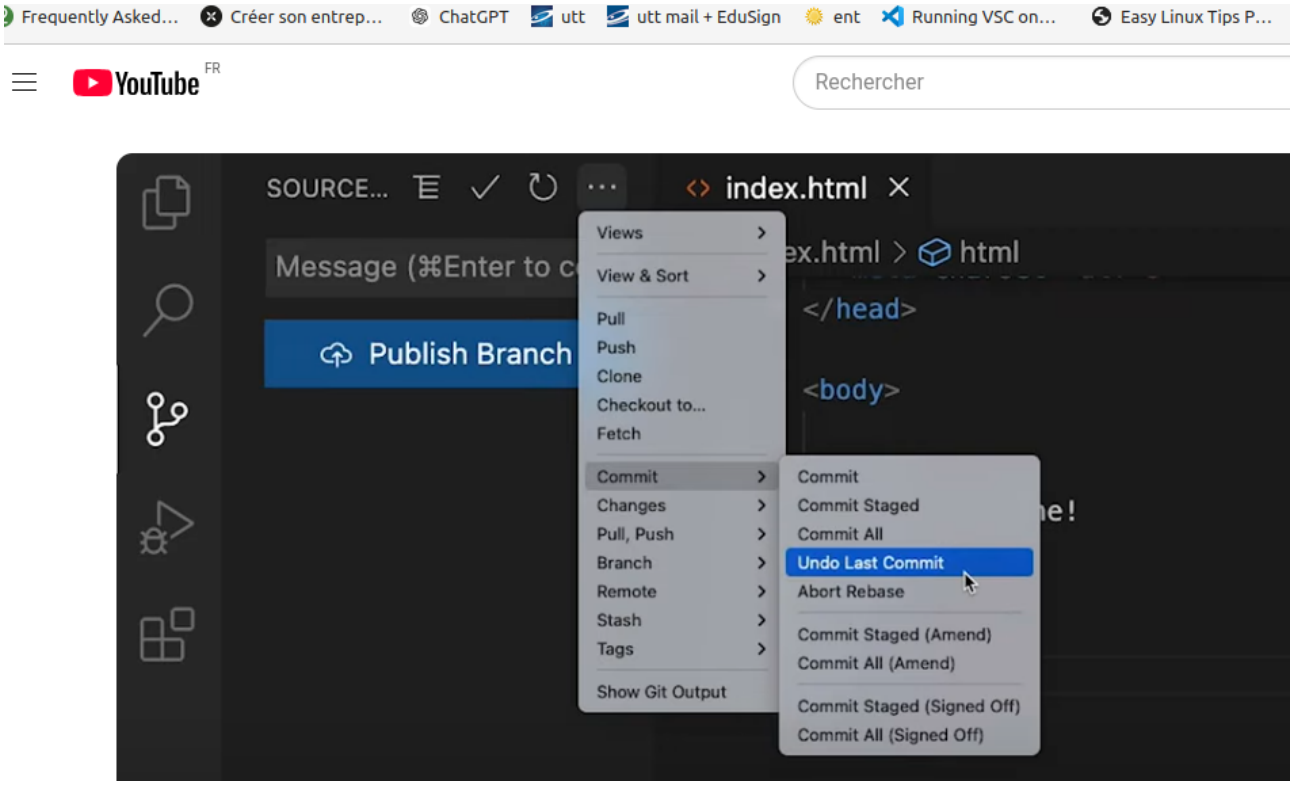
https://www.youtube.com/watch?v=E6ADS2k8oNQ

faire le lien entre les commande git et les features git dans vsc. On voit les commandes git dans le panel en bas puis tab "output"

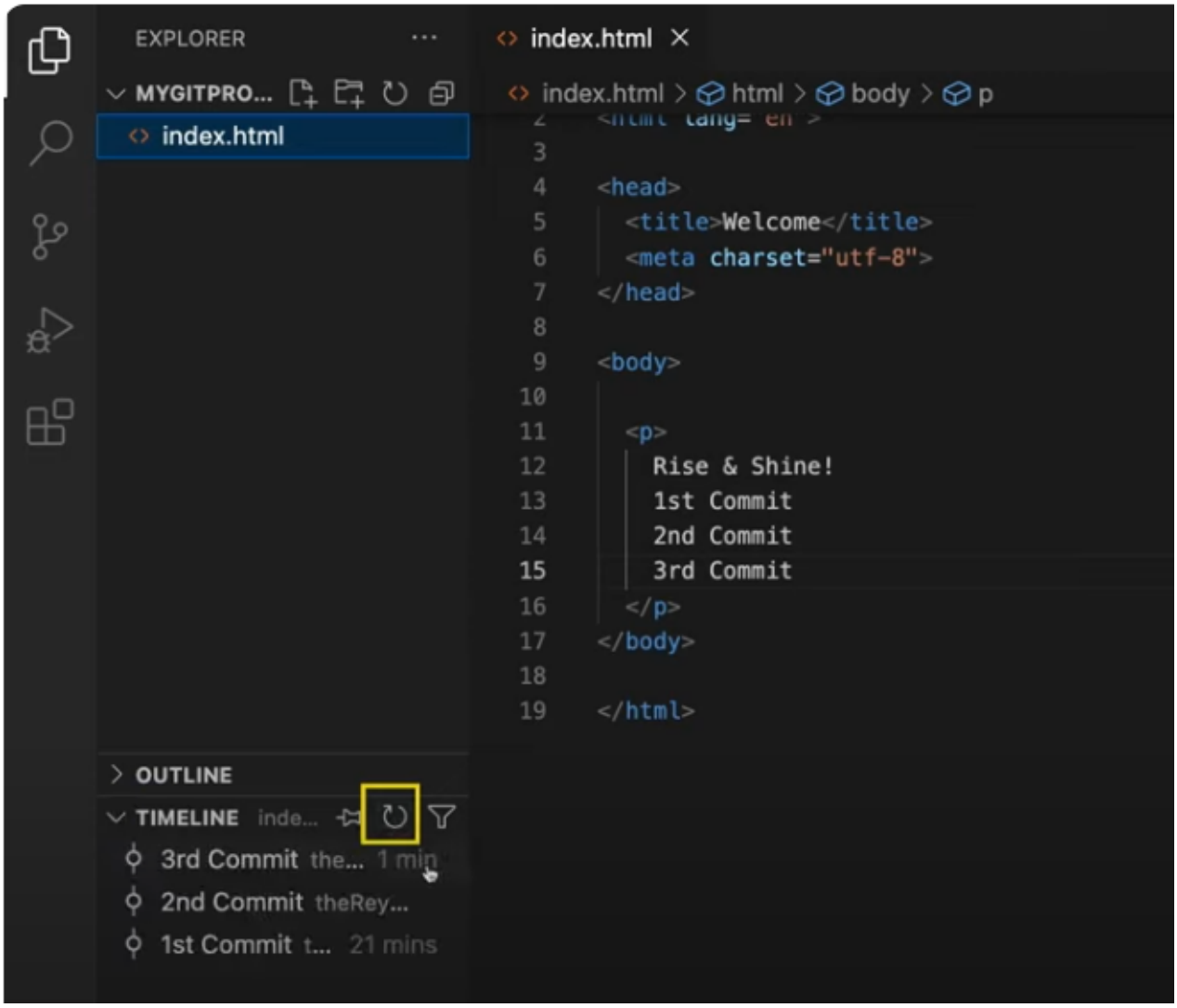


0:00 Intro to Git in VS Code
0:18 How to do git add in VS Code
git add = stage file=> dans VSC activity bar=source control /side bar = changes puis faire + en face du fichier
0:37 How to do git reset in VS Code
git reset = supprimer un fichier en stage et le remettre dans la section changes => dans VSC activity bar=source control /side bar = changes puis faire - en face du fichier
1:11 How to do git commit --amend in VS Code
imagine que l'on a fait un commit imcomplet on peut le remodifier pour ajouter la modification manquante
2:37 How to undo last commit (git update-ref)

imagine que l’on a fait un commit par erreur on peut revenir en arriere:



3:33 How to rollback to previous commit (git update-ref)
Si on a fait plusieurs commit par erreur on peut aussi revenir en arrière. Par exemple ce fichier a subit 3 commits (Explorer puis timeline puis refresh (la fleche qui tourne)



on fait undo last commit autant de fois que nécessaire
une fois qu'on est remonté à la bonne version
unstage (click sur -) puis discard changes et supprimer le commentaire de commit
=> verifier dans la timeline du fichier que les commits non désirés ont disparu
5:18 How to do git stash in VS Code
stash enregistrer des modifications que l'on voudra annuler (= pop stash) ou restaurer (apply stash) plus tard
Dans Git, la commande stash est utilisée pour sauvegarder temporairement les modifications en cours dans votre espace de travail (working directory) sans les committer, afin de pouvoir travailler sur autre chose ou changer de branche sans perdre ces modifications.

Le stash est pratique pour éviter d'annuler ou de committer des modifications non terminées.

Cas d'usage :

Vous travaillez sur un fichier mais devez basculer rapidement sur une autre branche ou corriger un bug.

Vous voulez sauvegarder vos modifications incomplètes pour revenir à un état propre.

Fonctionnement :

1. git stash : Enregistre les changements dans un stockage temporaire (le stash) et restaure l'état propre de la branche.

2. git stash list : Montre la liste des stashes sauvegardés.

3. git stash apply : Réapplique le dernier stash sans le supprimer.

4. git stash pop : Réapplique le dernier stash et le supprime de la liste.

7:28 How to do git clean command

Enleve de maniere recursive les fichiers dans le “working tree” et qui ne sont pas versionnés dans Git

3 Utilisations courantes :

1. Liste des fichiers non suivis :

bash

Copy code

git clean -n

Affiche les fichiers/répertoires qui seraient supprimés sans les supprimer réellement (mode simulation).

2. Supprimer les fichiers non suivis :

bash

Copy code

git clean -f

Supprime les fichiers non suivis. Le flag -f (force) est nécessaire pour confirmer la suppression.

3. Supprimer aussi les répertoires non suivis :

bash

Copy code

git clean -fd

Supprime les fichiers et répertoires non suivis.

4. Supprimer uniquement les fichiers ignorés (définis dans .gitignore) :

bash

Copy code

git clean -Xf

5. Supprimer tous les fichiers non suivis (ignorés et non ignorés) :

bash

Copy code

git clean -xdf

Précautions :

• Action irréversible : Une fois les fichiers supprimés, ils ne peuvent pas être récupérés via Git.

• Toujours tester avec -n avant d'utiliser -f pour éviter les pertes accidentelles.

1.2.4 Video Git: branches in Visual Studio Code

https://www.youtube.com/watch?v=b9LTz6joMf8

Time Stamps:

0:23 creating branches in visual studio code

1:40 listing branches in visual studio code

1:55 switching branches in visual studio code

2:54 rename a branch in visual studio code

3:19 deleting branches in visual studio code

1.3 Bonus: video Getting Started with Python in VS Code (Official Video)

https://www.youtube.com/watch?v=D2cwvpJSBX4

creation du virtual environnement

Chapters:

00:00 Getting started with Python in VS Code

00:23 Install Python

01:31 Install Python extension

02:29 Virtual Environment

IMPORTANT !

04:50 Executing Python file options

05:25 Using the Python REPL for quick tests

06:20 Code navigation and debugging

08:27 Debugging

09:21 Documentation

09:48 Let us know what you want to see next. Comment below!

Pour aller plus loin
https://code.visualstudio.com/docs/python/python-tutorial

1.4 Annexe

1.4.1 D’après chat GPT step-by-step guide to set up a Python project in Visual Studio Code (VS Code) with GitHub

Here’s a step-by-step guide to set up a Python project in Visual Studio Code (VS Code) with GitHub version control:

1. Prerequisites

Install Python on your system (check with python --version or python3 --version).

Install Git and configure it (git --version).

Create a GitHub account.

2. Install VS Code and Required Extensions

Download and install VS Code if not already installed.

Install these extensions from the Extensions Marketplace (Ctrl+Shift+X):

Python (by Microsoft).

Pylance (optional for IntelliSense).

GitHub Pull Requests and Issues (for GitHub integration).

GitLens (optional for advanced Git tools).

3. Create a Python Project

Open VS Code.

Create a new folder for your project:

Go to File > Open Folder and choose or create a folder.

Tout commence avec un folder

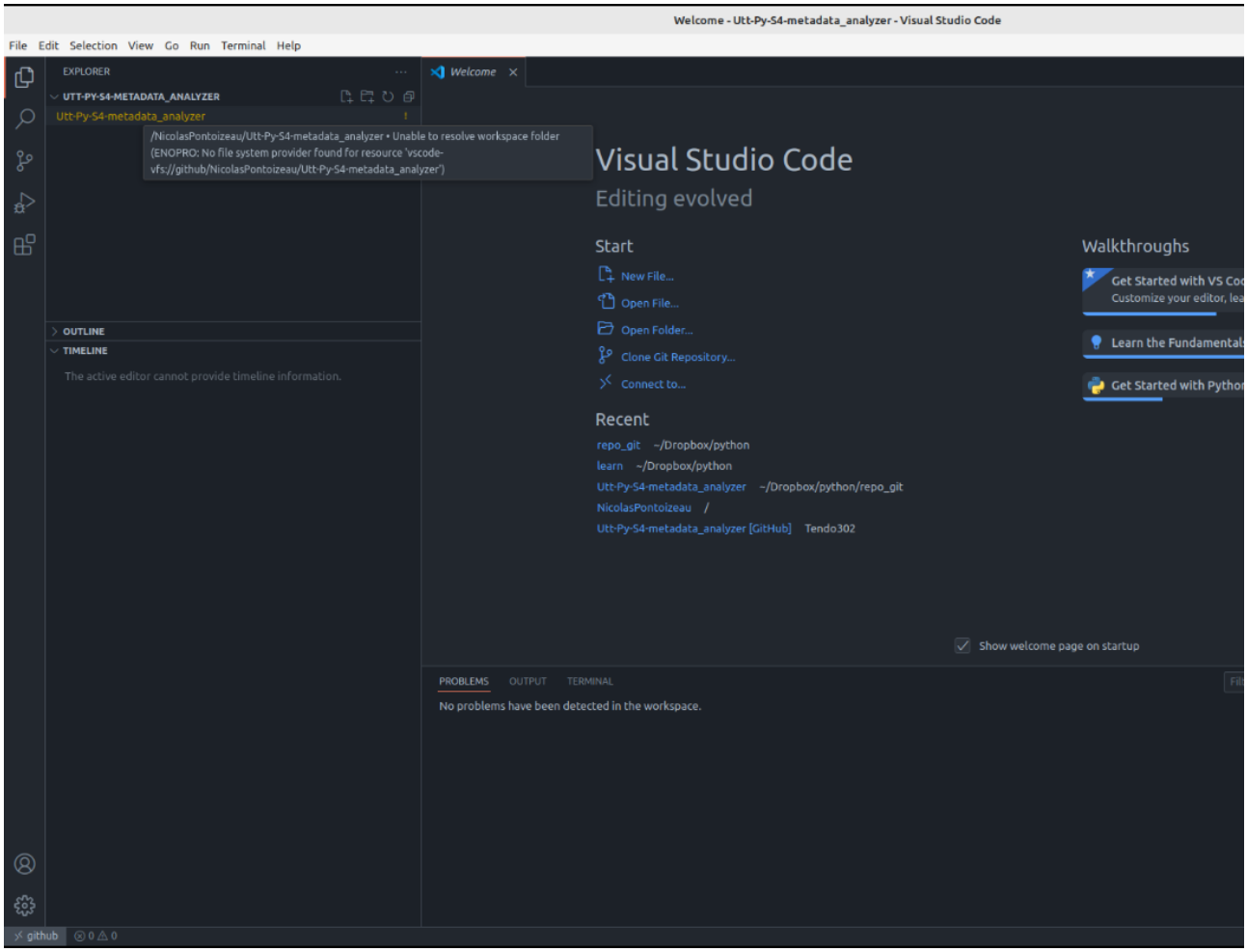
moi j’en ai plusieurs:

1) ~/Dropbox/python/learn

=> pour les petits scripts hors projet

2) /NicolasPontoizeau/Utt-Py-S4-metadata_analyzer

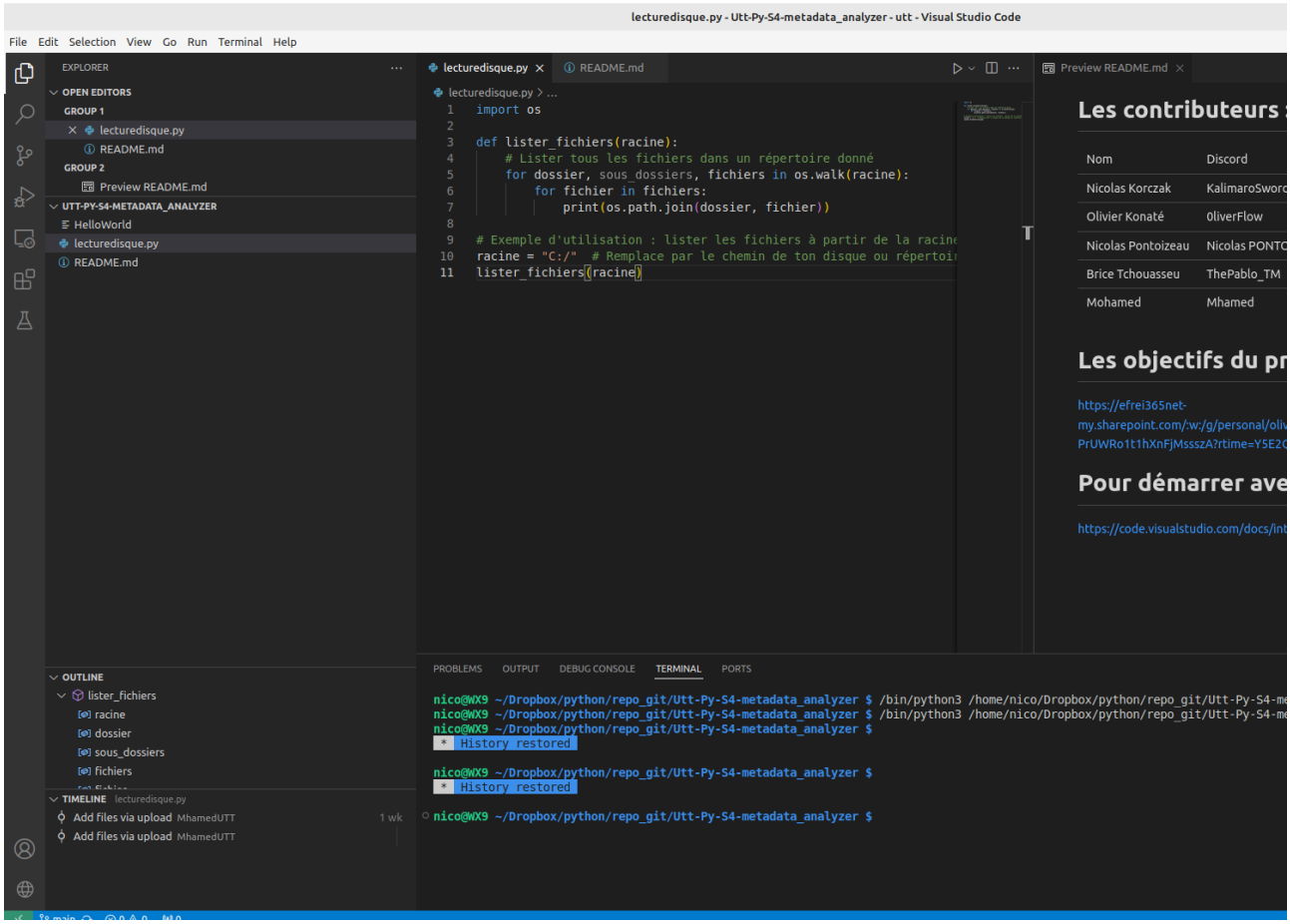
mais il déconne car j’ai supprimé le repertoire de mon disque



et

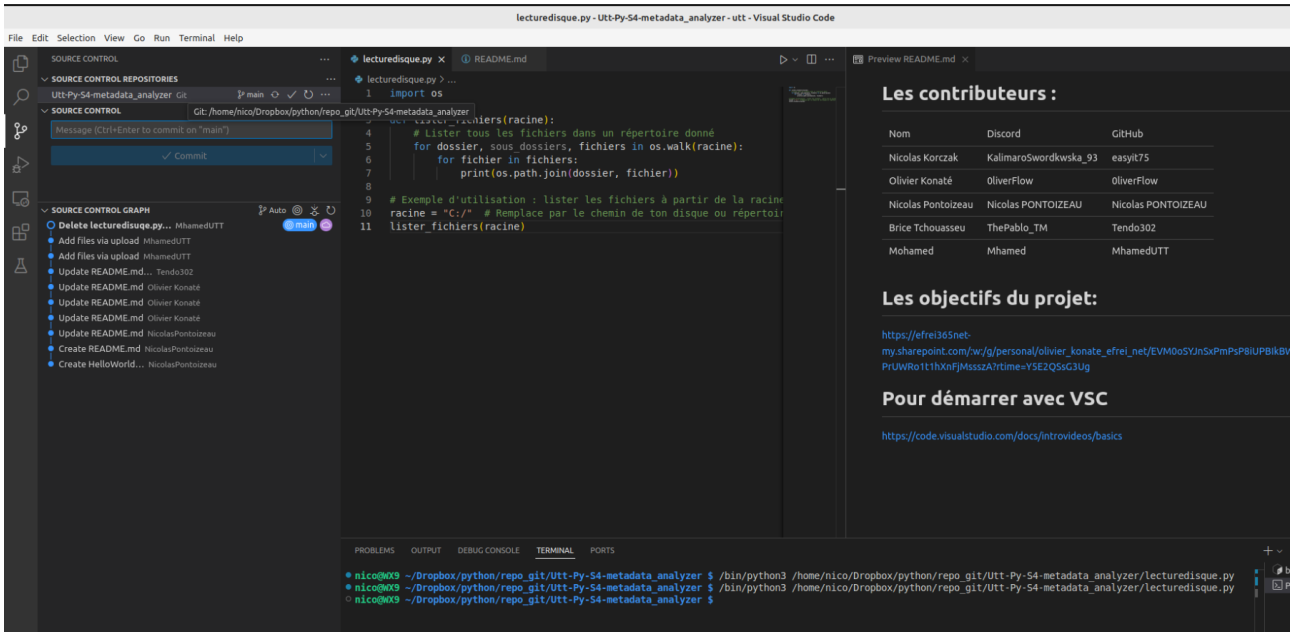
3) ~/Dropbox/python/repo_git/Utt-Py-S4-metadata_analyzer

ce Folder fonctionne mieux



le repo git local initialisé est dans

home/nico/Dropbox/python/repogit/Utt-Py-S4-metadata_analyzer



Add a new Python file:

Click on the Explorer icon (Ctrl+Shift+E).

Right-click the folder → New File → name it (e.g., main.py).

4. Initialize a Git Repository

Open the terminal in VS Code (Ctrl+`).

Run the following commands:

bash

Copy code

git init

Add a .gitignore file to exclude unnecessary files:

Right-click in the folder → New File → name it .gitignore.

Add common Python exclusions:

bash

Copy code

__pycache__/

*.pyc

.vscode/

env/

*.log

Stage and commit the changes:

bash

Copy code

git add .

git commit -m "Initial commit"

5. Create and Link a GitHub Repository

Go to GitHub → New Repository.

Set repository name (e.g., my-python-project) and create it.

Link it to your local repository:

bash

Copy code

git remote add origin https://github.com/your-username/my-python-project.git

git branch -M main

git push -u origin main

6. Set Up the Python Environment

Install a virtual environment:

bash

Copy code

python -m venv env

Activate the environment:

Linux/Mac: source env/bin/activate

Windows: env\Scripts\activate

Install dependencies (if any):

bash

Copy code

pip install <package-name>

Save dependencies in a requirements.txt:

bash

Copy code

pip freeze > requirements.txt

7. Configure VS Code

Select the Python interpreter:

Press Ctrl+Shift+P → type Python: Select Interpreter → select your virtual environment.

Enable auto-formatting:

Go to Settings → Search for Python Formatting → Choose your formatter (e.g., black).

Create a launch.json for debugging:

Go to Run > Add Configuration → Select Python → Customize the settings.

8. Push Changes to GitHub

Stage and commit your changes:

bash

Copy code

git add .

git commit -m "Add initial Python code"

Push to GitHub:

bash

Copy code

git push

9. Work with GitHub in VS Code

View Git changes in the Source Control tab (Ctrl+Shift+G).

Pull, push, or clone repositories directly in VS Code using the Git integration.

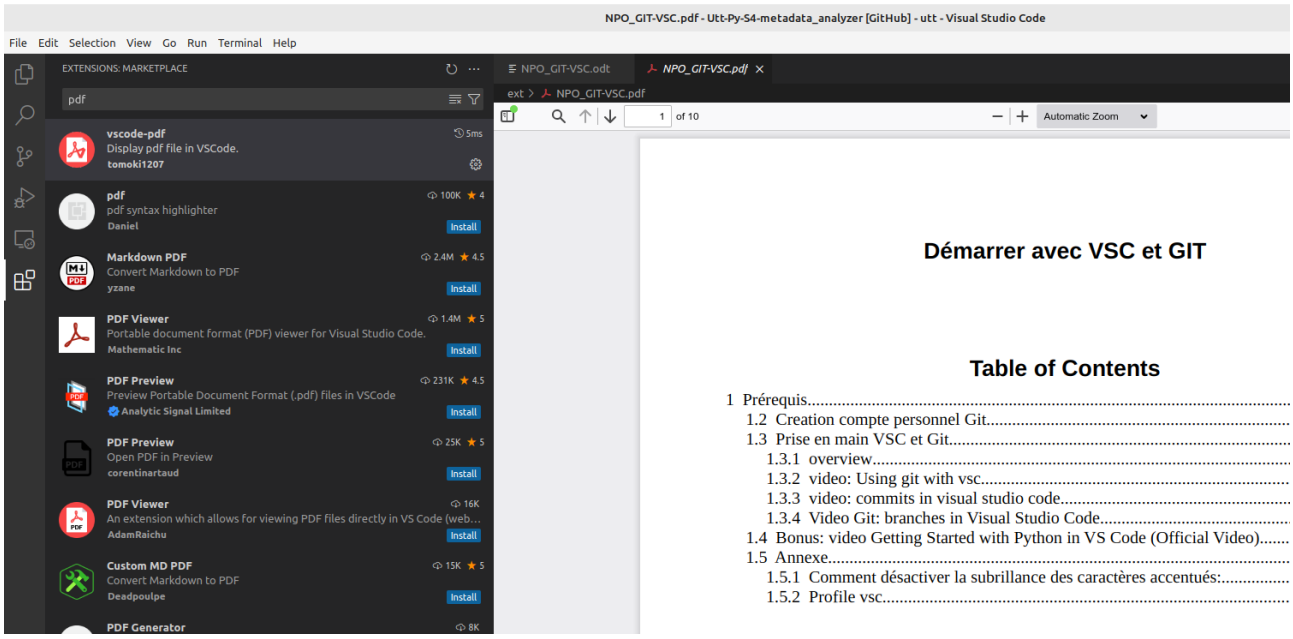
Use GitHub Pull Requests and Issues for advanced collaboration.

1.4.2 Comment désactiver la subrillance des caractères accentués:

```c
// na função main/outra  ** (não copiar este comentário) **
    pthread_t tids[NUM_THREADS];
    thread_params_t thread_params[NUM_THREADS];


    // Inicialização das estruturas - para cada thread
    for (int i = 0; i < NUM_THREADS; i++){
        thread_params[i].id = i + 1;

    }

    // Criação das threads + passagem de parâmetro
    for (int i = 0; i < NUM_THREADS; i++){
        if ((errno = pthread_create(&tids[i], NULL, task, &thread_params[i])) != 0)
            ERROR(10, "Erro no pthread_create()!");
    }
```

il faut truster le répertoire (ctrl palette puis workspaces :manage workspace trust) pour que les caractères apparaissent normalement

1.4.3 Comment lire directement les .pdf dans VSC

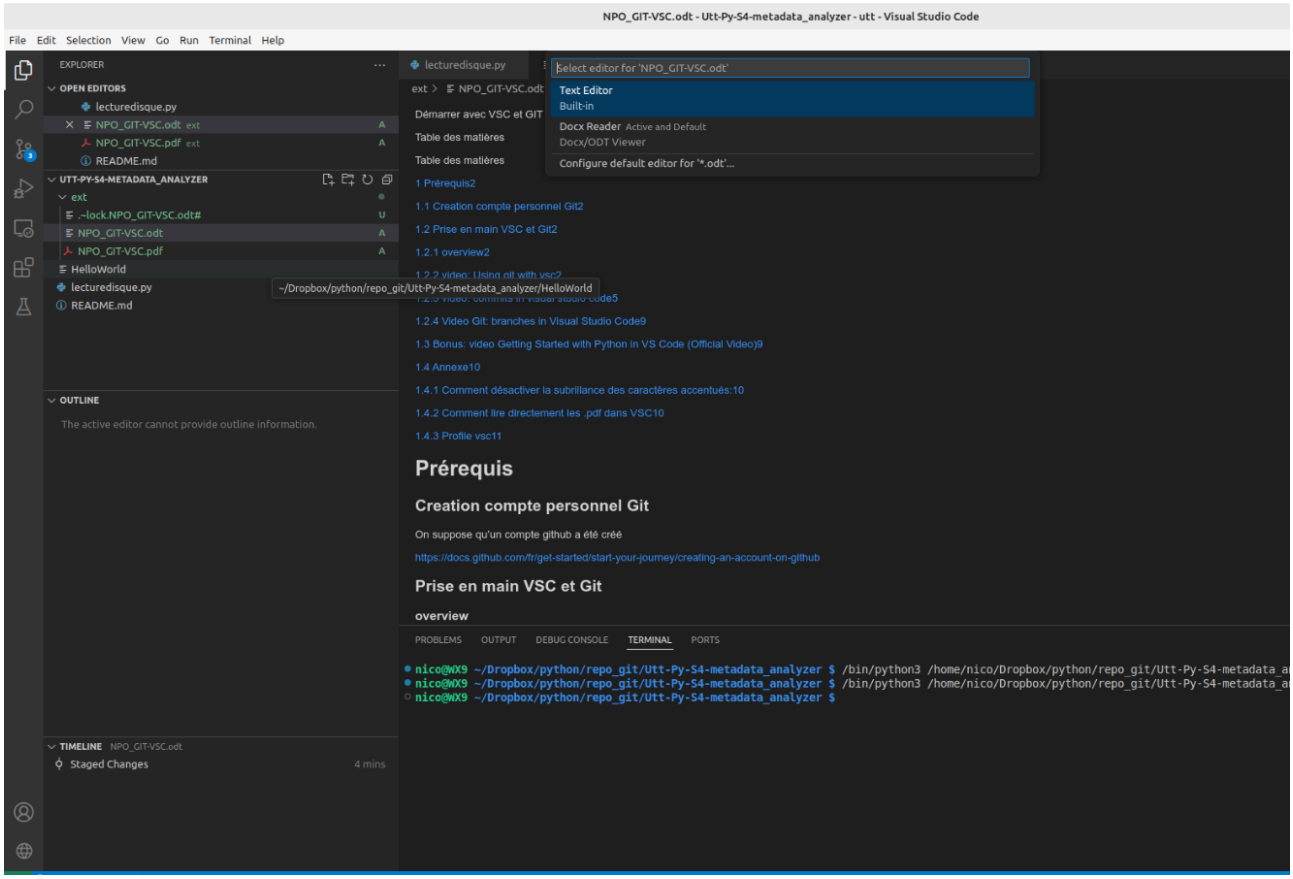


CTRL palette puis installer extension vscode-pdf
puis configurer l'ouverture des *.pdf avec le reader pdf

1.4.4 Comment lire directement les docx et .odf (fichier libreoffice writter) dans VSC

Installer l'extension docx/ODT viewer

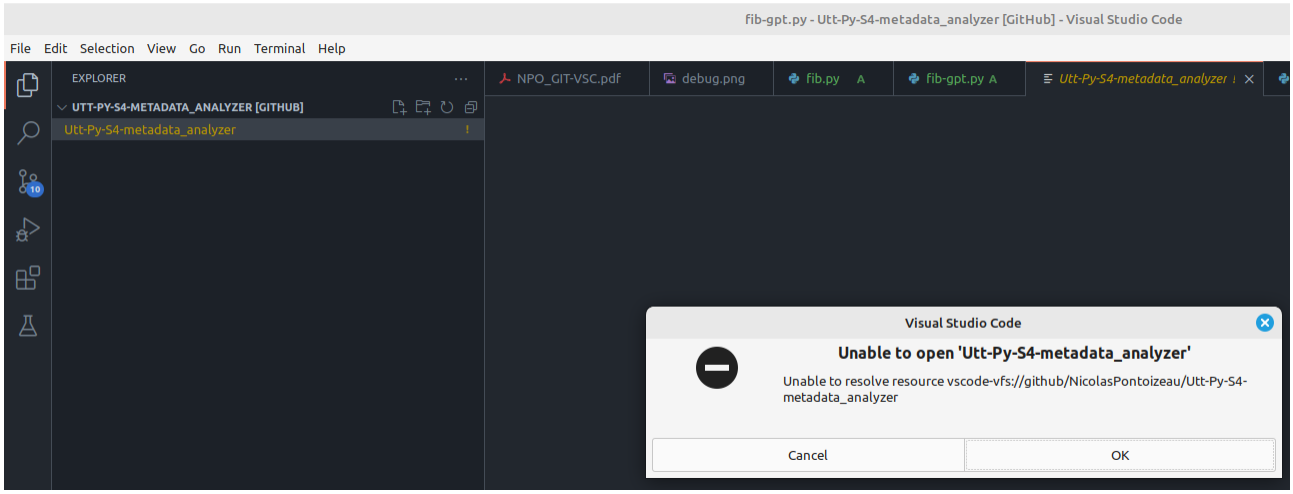
puis dans l'explorer click droit sur le fichier odt ou docx « reopen editor with » dans la palette en haut apparaît : (il faut choisir le 2eme docx reader)



le viewer ne permet pas de modifier le fichier. Pour ce faire il faut dans l'explorer click droit sur le fichier puis Open Containing Folder puis ouvrir le fichier avec Word ou libreOffice...

1.4.5 erreur "unable to resolve resource vscode-vfs ..."?

J'ai supprimé mon



1.4.6 Profile vsc

En bas à gauche de l'activity barre clic sur sur le globe

! par défaut l'icone du profile est une roue crantée :

On peut changer l'icone pour différencier ses projets personnels des projets pour l'utt

définir un workspace local (ex moi /home/nico/Dropbox/python/repo_git/Utt-Py-S4-metadata_analyzer