

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.....	9
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.....	10
1.4.1.1	Prerequisites.....	10
1.4.1.2	Install VS Code and Required Extensions.....	10
1.4.1.3	Create a Python Project.....	10
1.4.1.4	Initialize a Git Repository.....	13
1.4.1.5	Create and Link a GitHub Repository.....	13
1.4.1.6	Set Up the Python Environment.....	14
1.4.1.7	Configure VS Code.....	17
1.4.1.8	Push Changes to GitHub.....	18
1.4.1.9	Work with GitHub in VS Code.....	18
1.4.2	Comment désactiver la subbrillance des caractères accentués:.....	18
1.4.3	Comment lire directement les .pdf dans VSC.....	19
1.4.4	Comment lire directement les docx et .odf (fichier libreoffice writer) dans VSC.....	19
1.4.5	raccourci utiles.....	20
1.4.6	Profile vsc.....	21

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>

pour nous https://github.com/NicolasPontoizeau/Utt-Py-S4-metadata_analyzer/tree/main

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

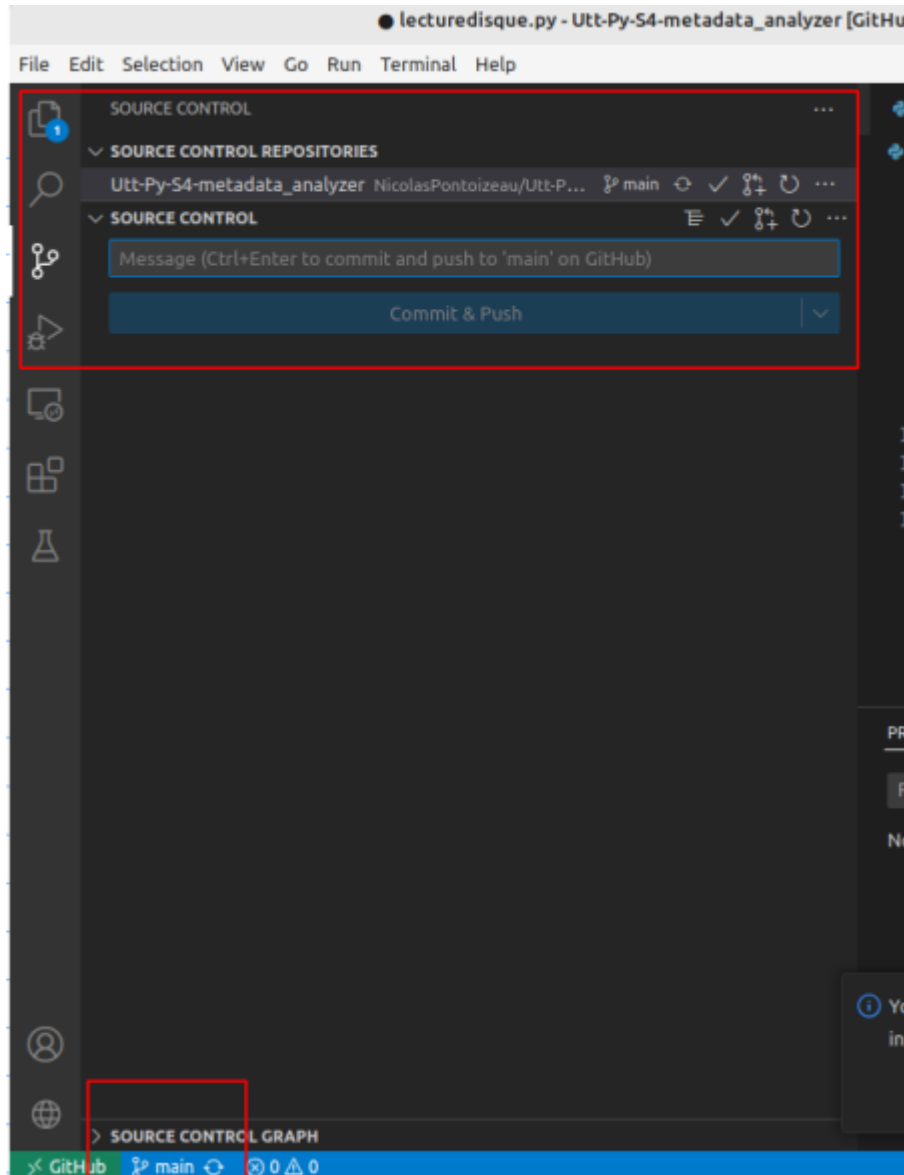
points clés de la vidéo :

[0:35](#) Initialize repository

pour refléter son repertoire en local avec son repository git => remarque en bas à gauche on travaille sur la 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 cliquer sur + pour “stage” et la lettre en face devient A puis commit pour ajouter le nouveau fichier 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

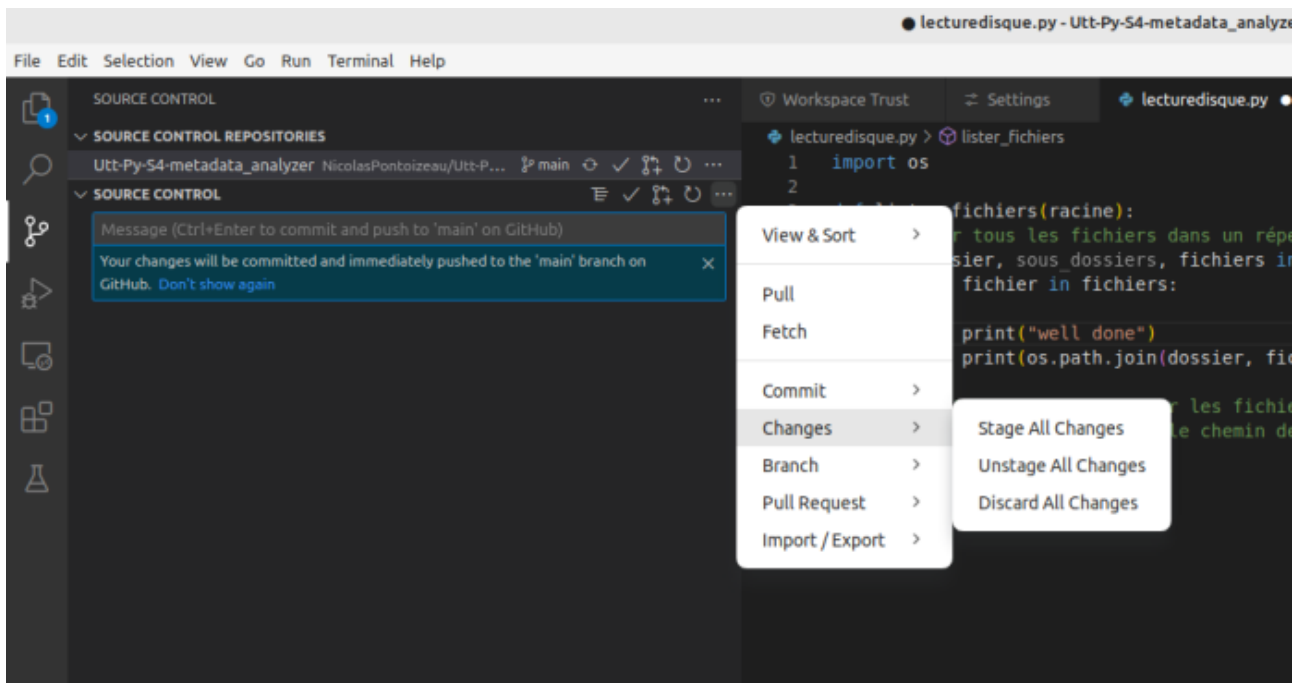
Quand on ajoute des lignes de code a un fichier elles apparaissent en vert dans la marge (gutter area)

Quand on modifier une ligne existante elles apparaissent en shaded 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 décrit 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 committée 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 mûres il faut les merger (fusionner) à la branche main.

Dans la side bar du source control cliquer sur les ... (ellipses) puis branch puis merge branch puis sélectionner la branch à merger sur le main

[5:00](#) Publish to Github

Dans la side bar du source control cliquer sur « Publish Branch » (rectangle 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 palette puis git : clone from git hub puis coller l'url puis sélectionner un repertoire en local où stocker le repo chargé)

[6:20](#) Summary

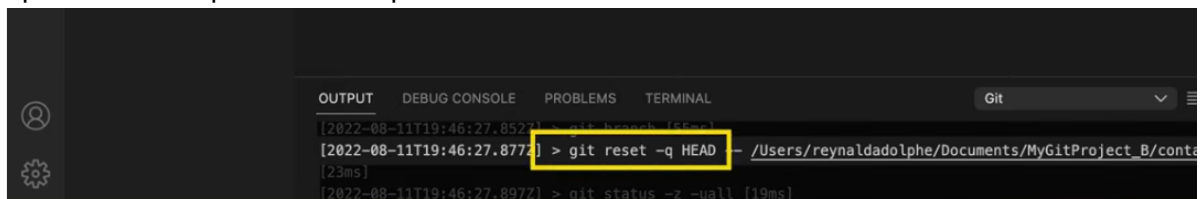
plus de details:

<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 commandes 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

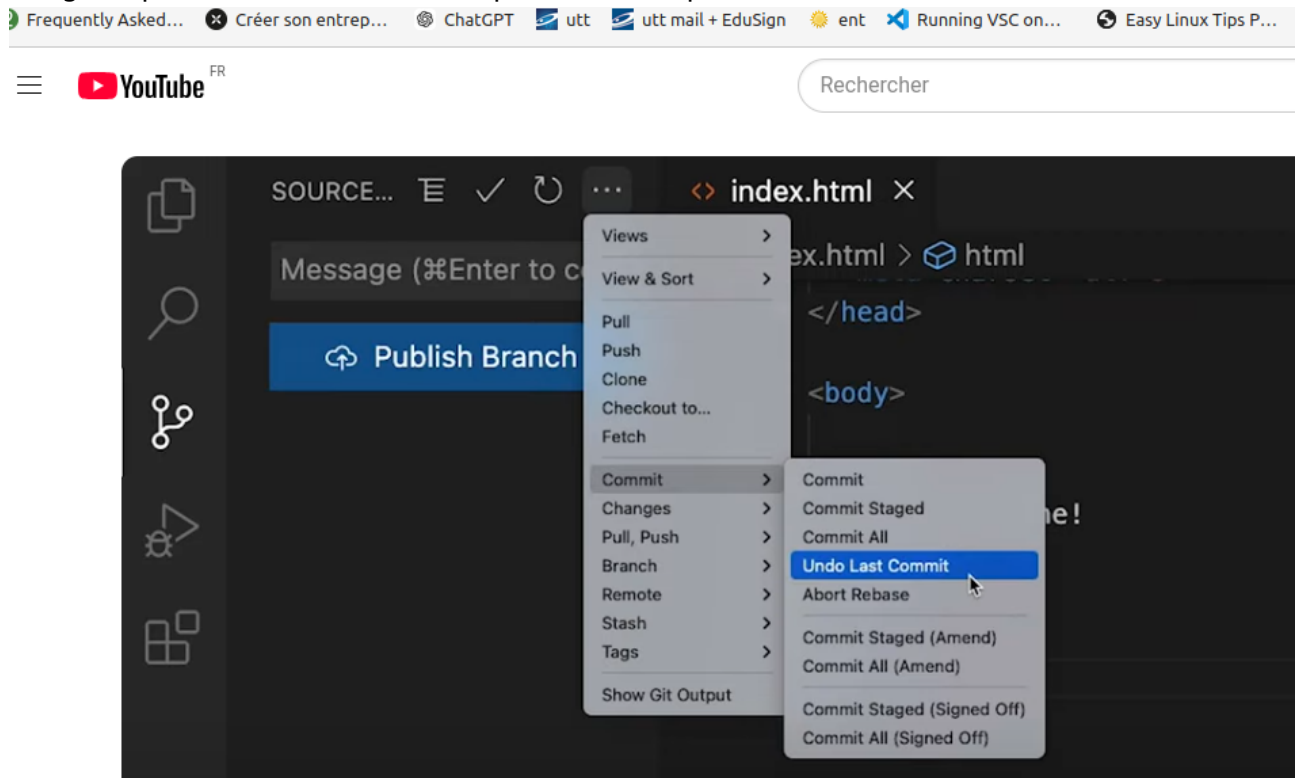
git reset = supprimer un fichier en stage et le remettre dans la section changes => dans VSC activity bar = source control / side bar = changes puis faire - en face du fichier

[1:11](#) How to do git commit --amend in VS Code

imagine que l'on a fait un commit incomplet on peut le modifier pour ajouter la modification manquante

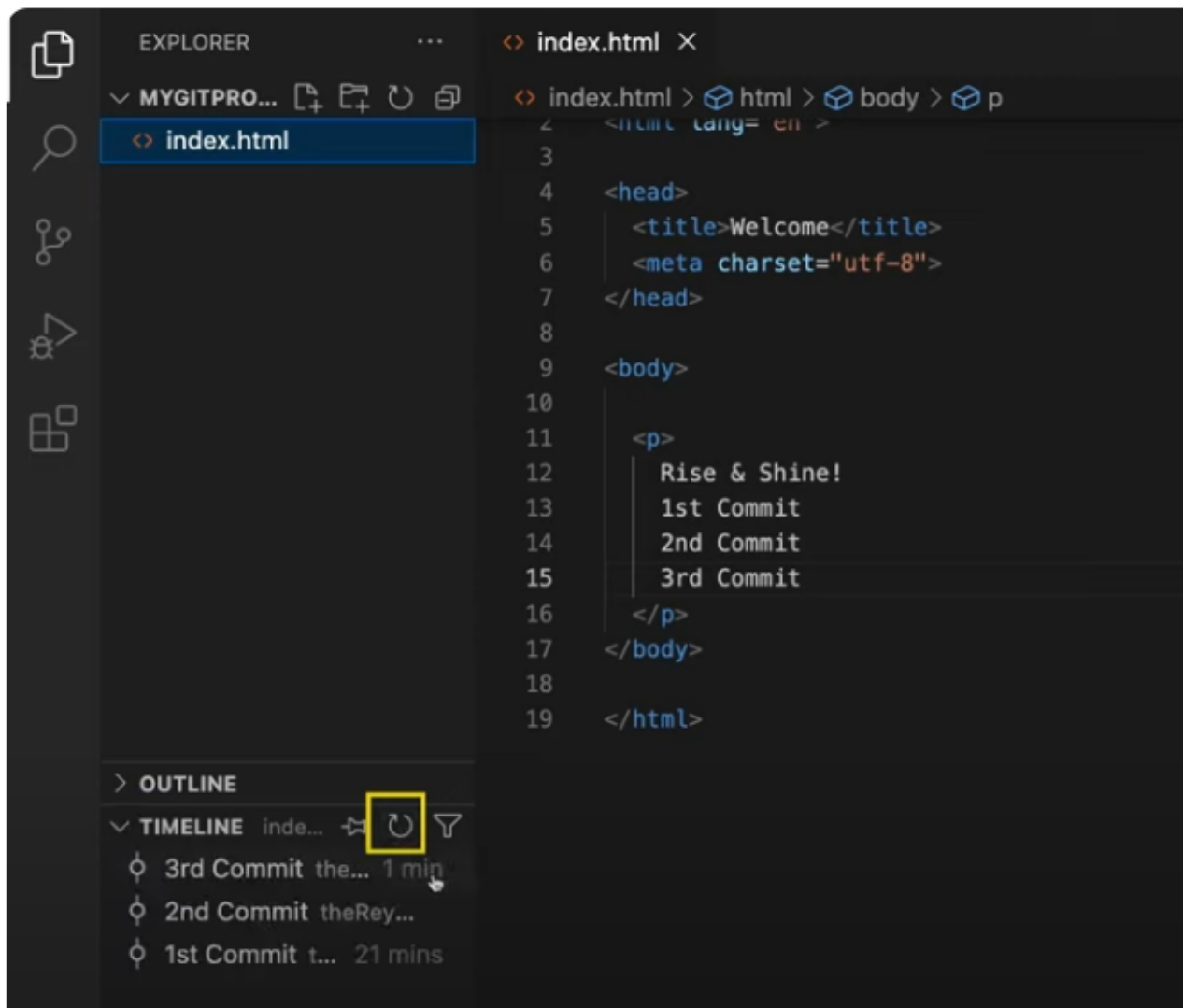
[2:37](#) How to undo last commit (git update-ref)

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 arriere. 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 commiter, 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 commiter 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 POO in Python

1.3.1 Classes and Instances

[Python OOP Tutorial 1: Classes and In...](#)

1.3.2 Class Variables

[Python OOP Tutorial 2: Class Variables](#)

1.3.3 Classmethods and Staticmethods

[Python OOP Tutorial 3: classmethods a...](#)

1.3.4 Inheritance

[Python OOP Tutorial 4: Inheritance - ...](#)

1.3.5 Special (Magic/Dunder) Methods

[Python OOP Tutorial 5: Special \(Magic...](#)

1.3.6 Property Decorators

[Python OOP Tutorial 6: Property Decor...](#)

1.4 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 ! Une fois le virtual environnement activé, les packages qui seront installés (par ex discord) seront isolés des autres environnements

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>

1.5 Bonus : taking notes (markdown for documentation...)

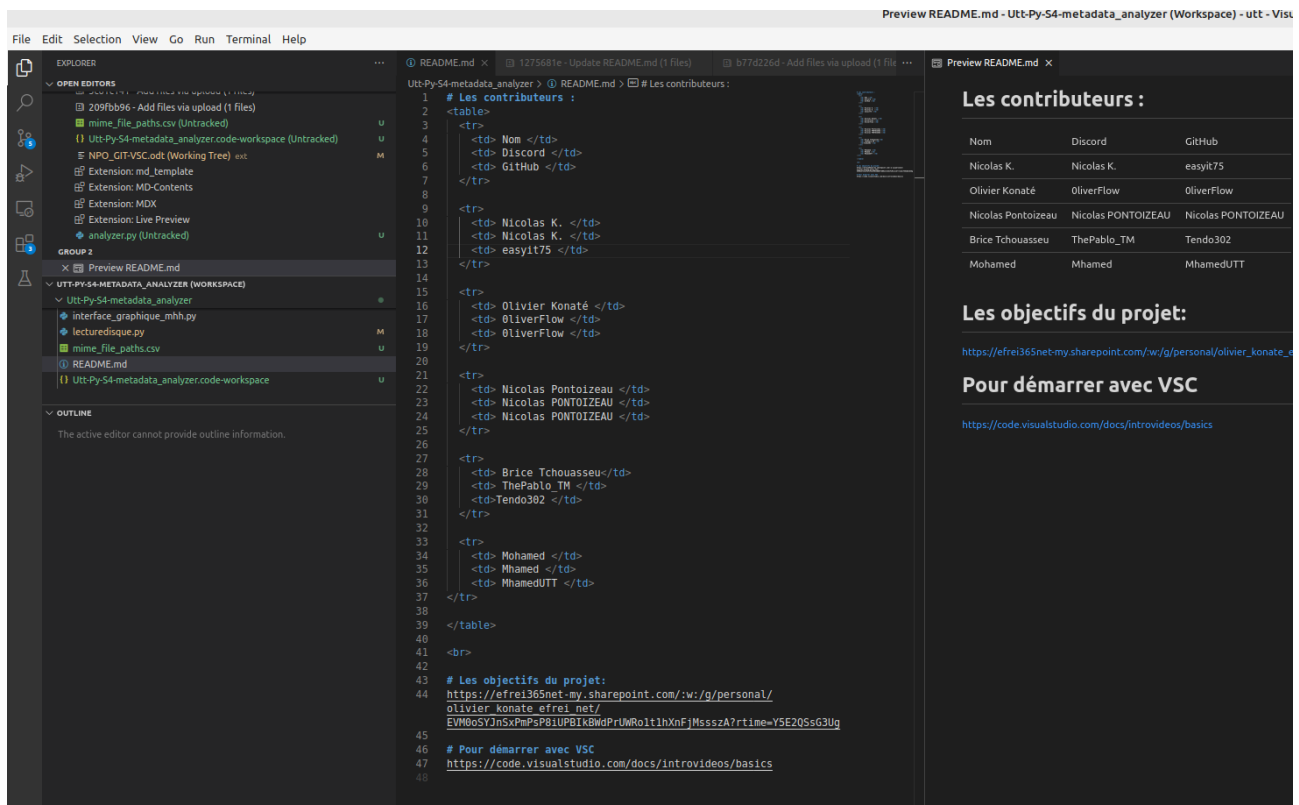
[https://www.youtube.com/watch?](https://www.youtube.com/watch?v=Hgucu1ch3mo&list=PLj6YeMhvp2S5UgiQnBfvD7XgOMKs3O_G6&index=4)

[v=Hgucu1ch3mo&list=PLj6YeMhvp2S5UgiQnBfvD7XgOMKs3O_G6&index=4](https://www.youtube.com/watch?v=Hgucu1ch3mo&list=PLj6YeMhvp2S5UgiQnBfvD7XgOMKs3O_G6&index=4)

[0:00](#) intro to documentation in VS Code

[0:22](#) markdown overview

CTRL K v faire apparaitre la preview du fichier md



[1:05](#) creating lists

[2:05](#) drag and drop images

[2:30](#) adding code

[3:00](#) adding links

[4:45](#) publishing documentation

[5:15](#) GitDoc for documentation

[6:50](#) vscode.dev for documentation

[7:35](#) summary

1.6 Bonus : resolve merge conflicts

https://www.youtube.com/watch?v=HosPml1qkrg&list=PLj6YeMhvp2S5UgiQnBfvD7XgOMKs3O_G6&index=9

[00:00](#) - Intro

[00:30](#) - What is a merge conflict?

[01:00](#) - "Base", explained

[02:18](#) - Simple merge conflict

[03:00](#) - When merge conflicts get complicated
[03:34](#) - Introducing the new Merge Editor
[04:19](#) - Showing the base
[04:30](#) - Comparing changes with the base
[05:15](#) - Resolving the conflicts
[07:10](#) - Getting back to the default merge experience
[07:35](#) - A 3 way merge

1.7 Bonus : Time saver shortcuts

https://www.youtube.com/watch?v=MLvLiWueESY&list=PLj6YeMhvp2S5UgiQnBfvD7XgOMKs3O_G6&index=7

[00:00](#) Serious Timesavers in VS Code
[00:03](#) Word Wrap Code
[00:18](#) Joining Lines Hack
[00:32](#) AutoSave - More Than One Way
[01:14](#) The Magic Button - Stop rage-quitting
[01:50](#) Most Underutilized Keyboard Shortcut

1.8 Bonus : using multi cursors

https://www.youtube.com/watch?v=1V4Lj8Eqdvc&list=PLj6YeMhvp2S5UgiQnBfvD7XgOMKs3O_G6&index=13

1.9 Bonus : debugging

https://www.youtube.com/watch?v=3HiLLByBWkg&list=PLj6YeMhvp2S5UgiQnBfvD7XgOMKs3O_G6&index=12

Chapters:

[00:00](#) GitHub Insider Newsletter
[00:22](#) Debugging in VS Code
[01:11](#) Variables
[01:21](#) Watch
[01:30](#) Call Stack

[01:38](#) Loaded Scripts

[01:49](#) Debug Tool Bar

[02:29](#) Polymorphism Demo

[06:10](#) CRUD App Demo

[09:12](#) Edit mode

[10:35](#) Debug in Microsoft Edge

[12:09](#) Learn more

[12:30](#) Want more on Debugging? Drop a "Part 2" in the comments!

doc sur le debugging:

<https://code.visualstudio.com/docs/editor/debugging>

1.10 Annexe

1.10.1 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:

1.10.1.1 Prerequisites

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

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

Create a GitHub account.

1.10.1.2 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).

1.10.1.3 *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

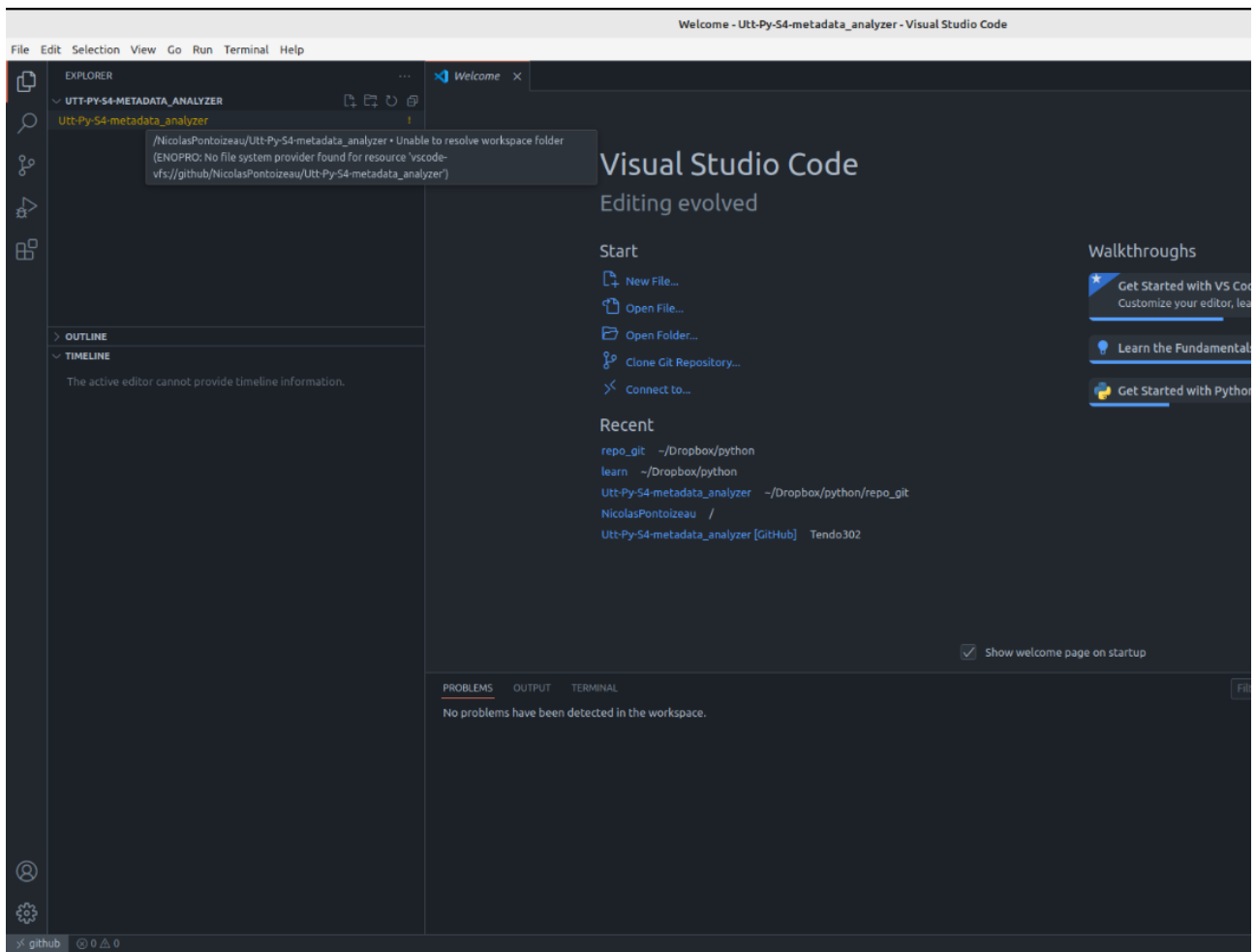
moi j'en ai plusieurs:

1) ~/Dropbox/python/learn

=> pour les petits scripts hors projet

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

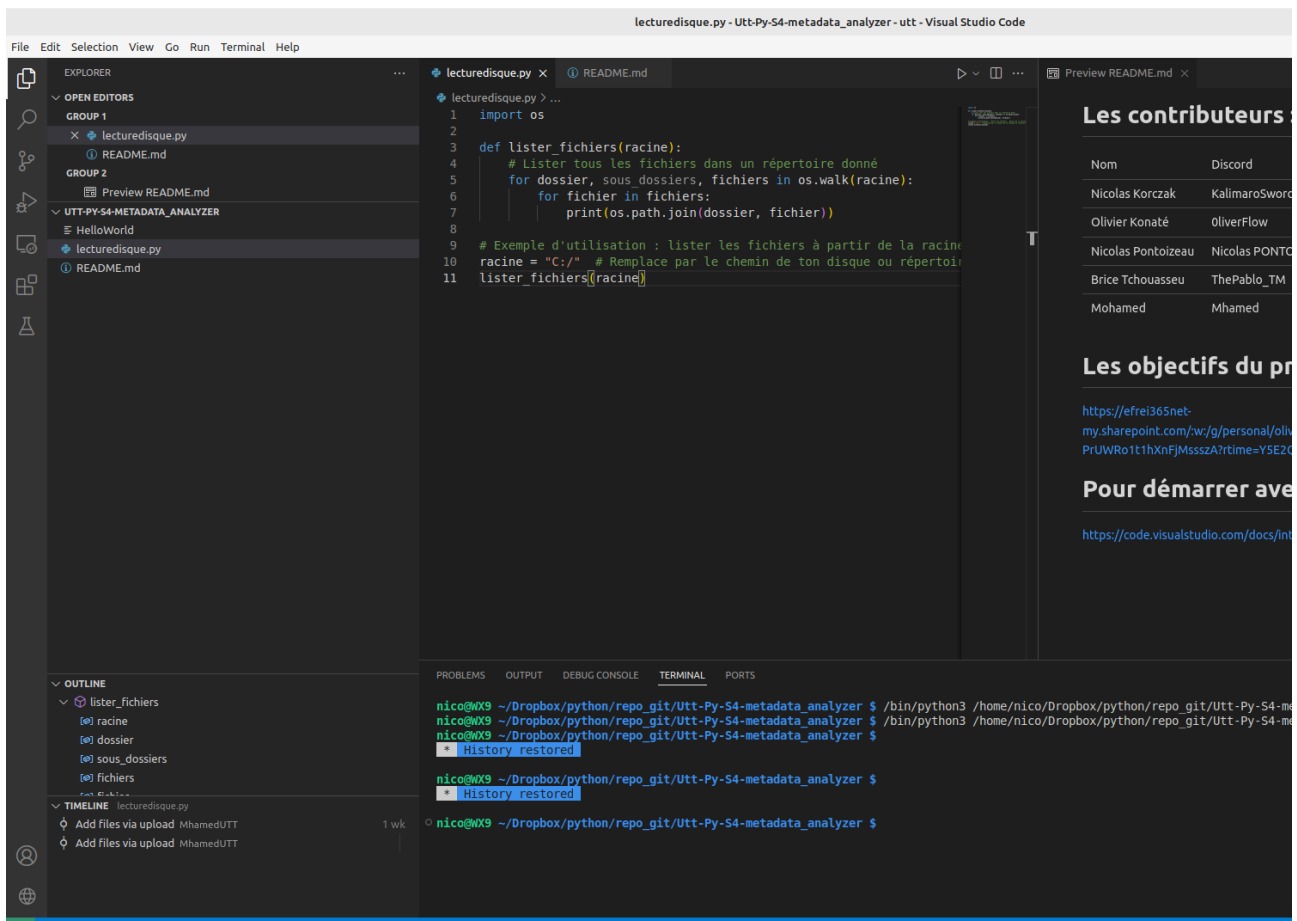
mais il déconne car j'ai supprimé le repertoire de mon disque



et

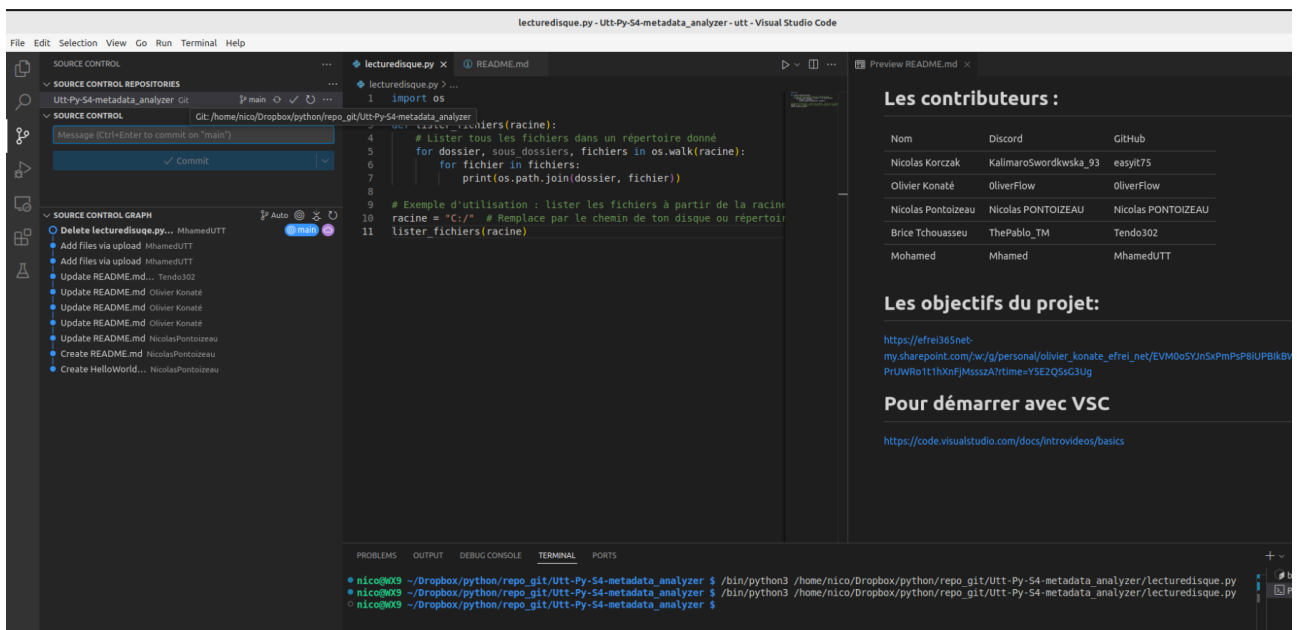
3) ~/Dropbox/python/repo_git/Utt-Py-S4-metadata_analyzer

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).

On a pas encore de main.py mais il en faudra un!

1.10.1.4 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"
```

1.10.1.5 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
```

1.10.1.6 Set Up the Python Environment

On va créer un environnement isolé dédié à notre projet et qui garantit que les bibliothèques installées avec `pip` n'affectent pas le Python système global.

lire la video : <https://www.youtube.com/watch?v=GZbeL5AcTgw>

Install a virtual environment:

```
win python -m venv env
```

```
mac/linux:python3 -m venv .venv
```

/!\ pour moi il fallait installer le module `venv` car il n'était pas installé avec ma distrib python3

voici la commande pour installer `venv`:

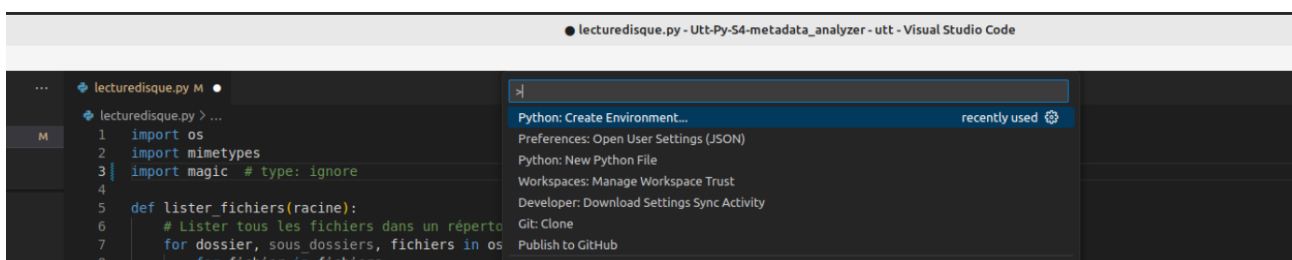
```
sudo apt install python3.10-venv
```

Activate the environment:

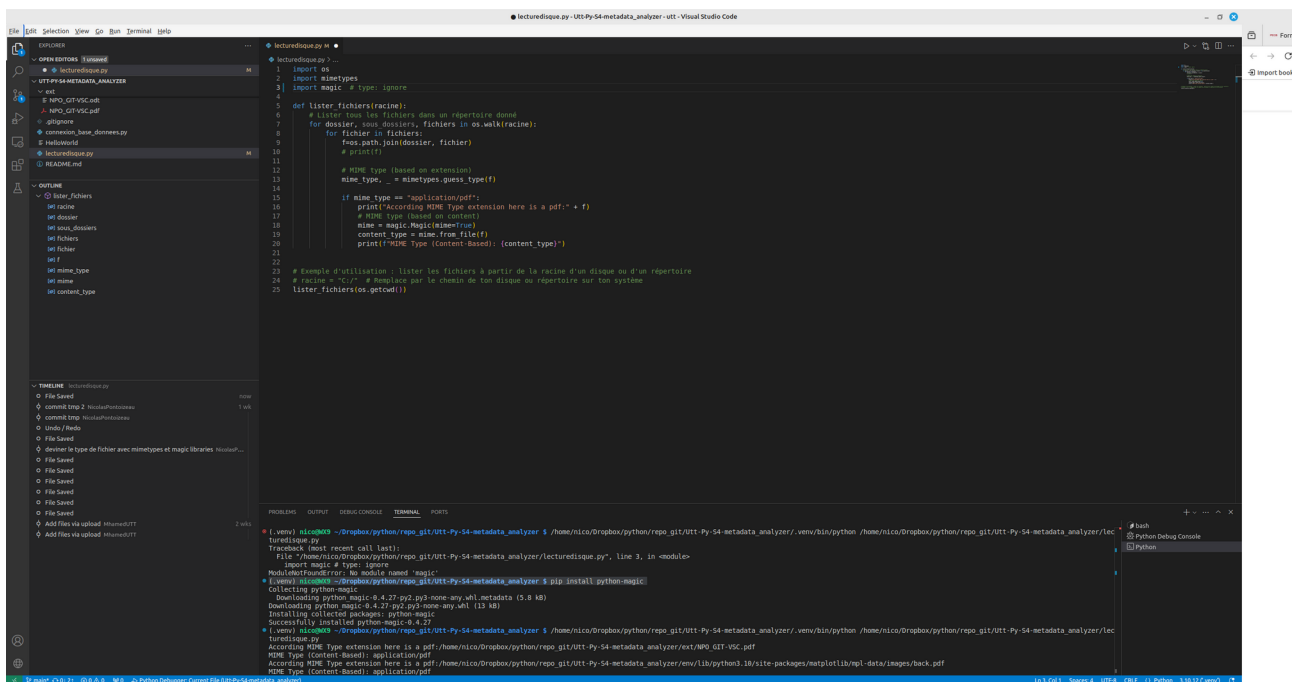
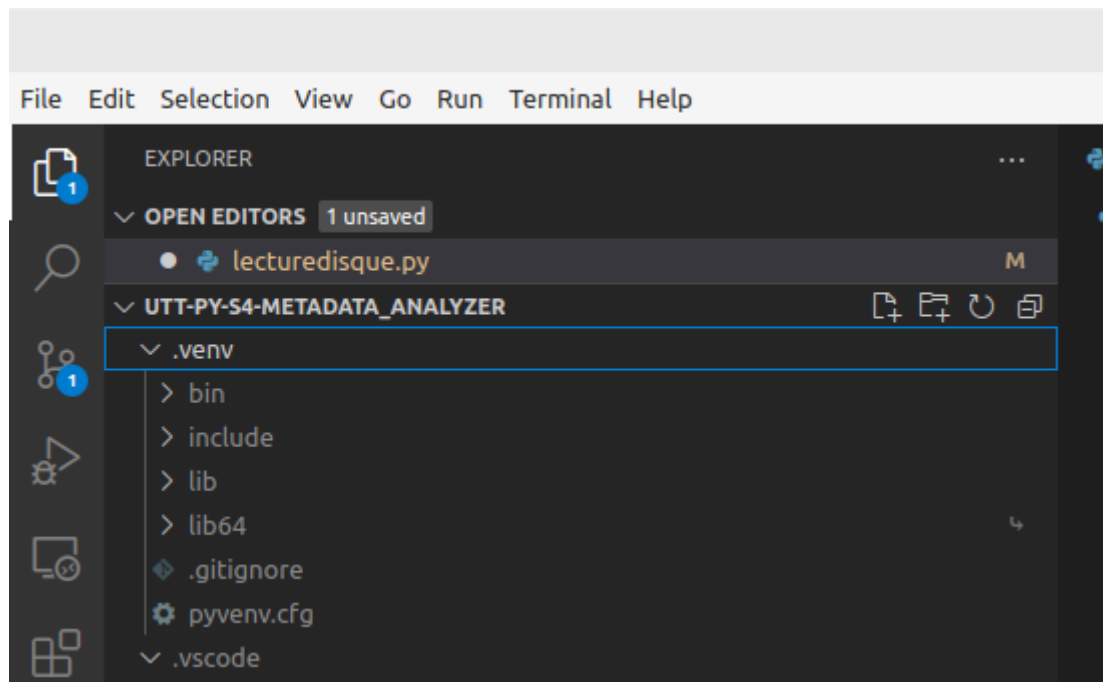
Linux/Mac: `source .venv/bin/activate`

Windows: `env\Scripts\activate`

ou avec la command palette `>Python: Create Environment`

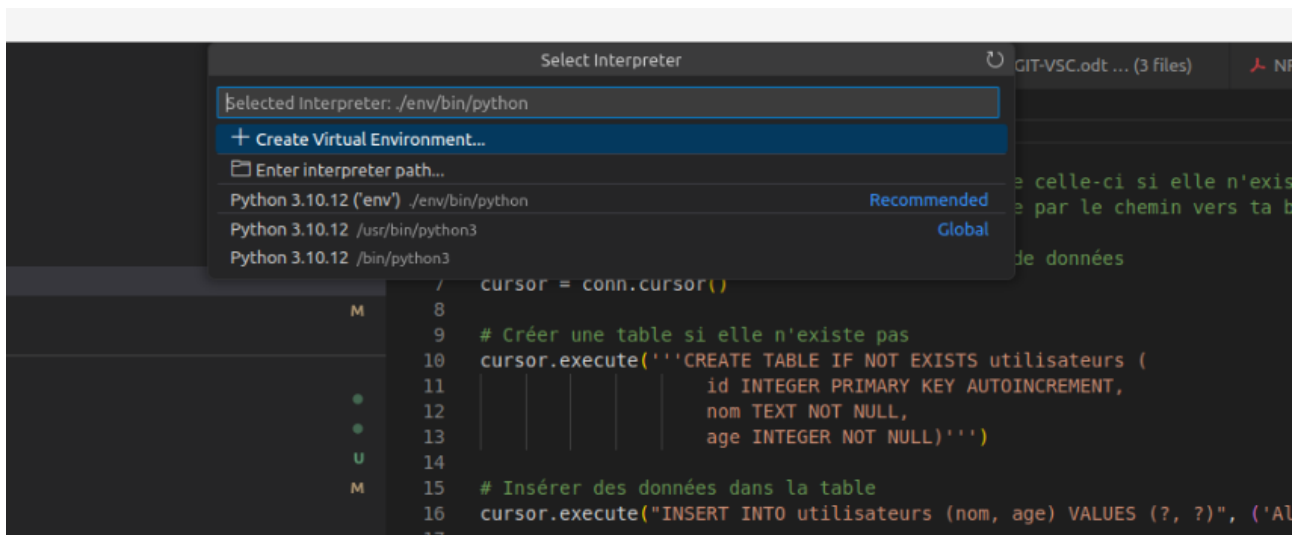


ceci va créer un répertoire `.venv` contenant plusieurs répertoires:

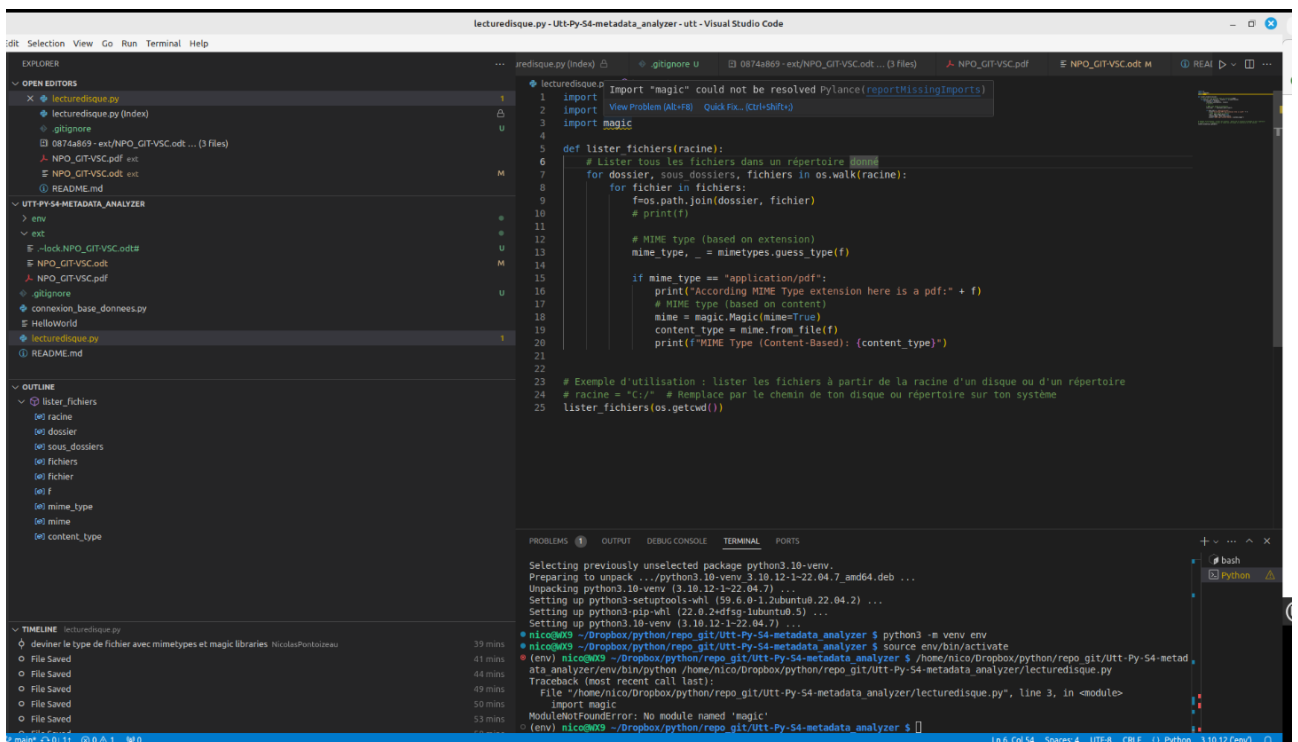


on voit en bas que je suis en Python 3.10.12 'env'

en cliquant on en bas à droit on peut changer la version de python3



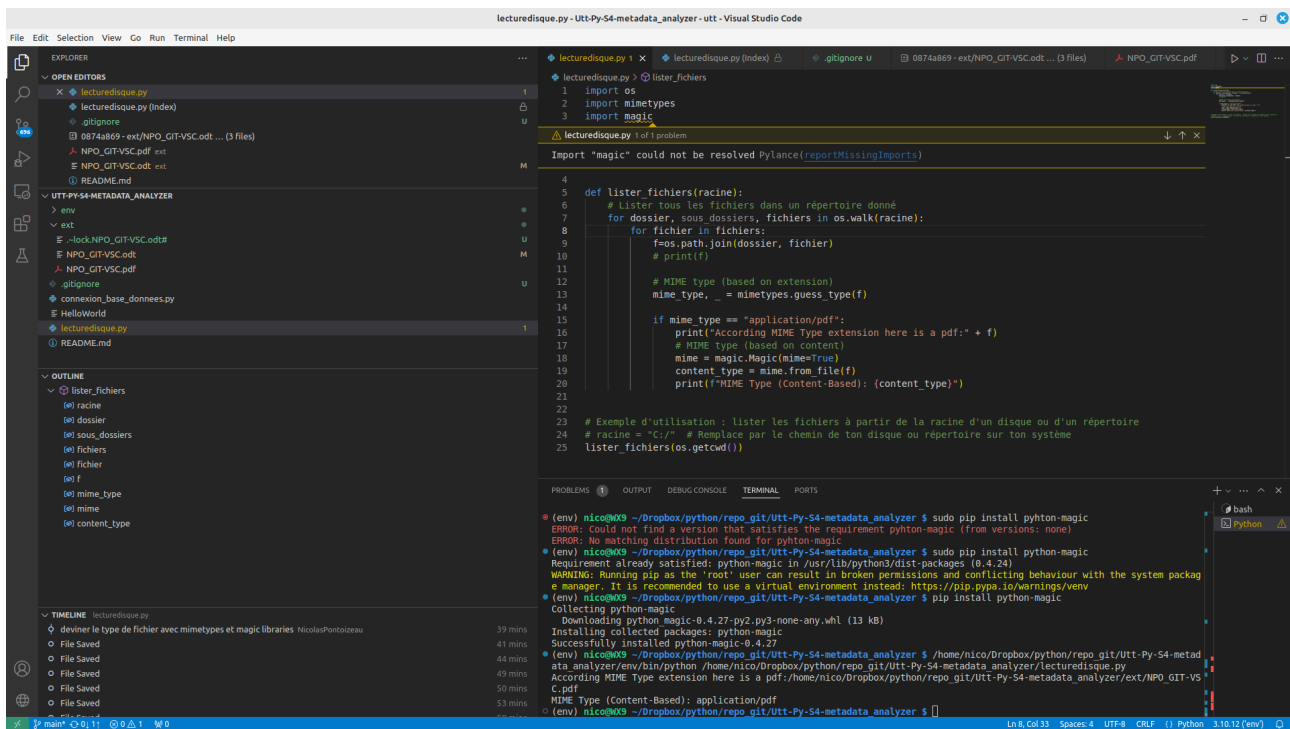
suite à la creation du venv vsc ne trouve plus la librairie magic. C'est normal



Pour corriger le pb avec l'import magic il faut réinstaller la librairie python-magic dans son virtual env (ne pas le faire avec sudo comme moi!)

dans le terminal taper

(.venv) nico@WX9 ~/Dropbox/python/repo_git/Utt-Py-S4-metadata_analyzer \$ pip install python-magic



et ajouter le commentaire `import magic # type: ignore` dans `lecturedisque.py` ce qui ignore le warning pour vs

Install dependencies (if any):

`pip install <package-name>`

Save dependencies in a `requirements.txt`:

`pip freeze > requirements.txt`

pour installer les packages listés dans `requirements.txt`

`pip install -r requirements.txt`

pour aller plus loin:

<https://python-guide-pt-br.readthedocs.io/fr/latest/dev/virtualenvs.html>

1.10.1.7 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`).

=> Installer le black formatter de microsoft. Maintenant le raccourci CTRL MAJ I permet de formater tout le fichier

Create a launch.json for debugging:

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

d'après

<https://www.youtube.com/watch?v=mpk4Q5feWaw>

l'extension ruff permet l'auto format on save ([18:09](#) Auto Formatting) A tester...

1.10.1.8 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

1.10.1.9 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.

1.10.2 Comment désactiver la subbrillance des caractères accentués:

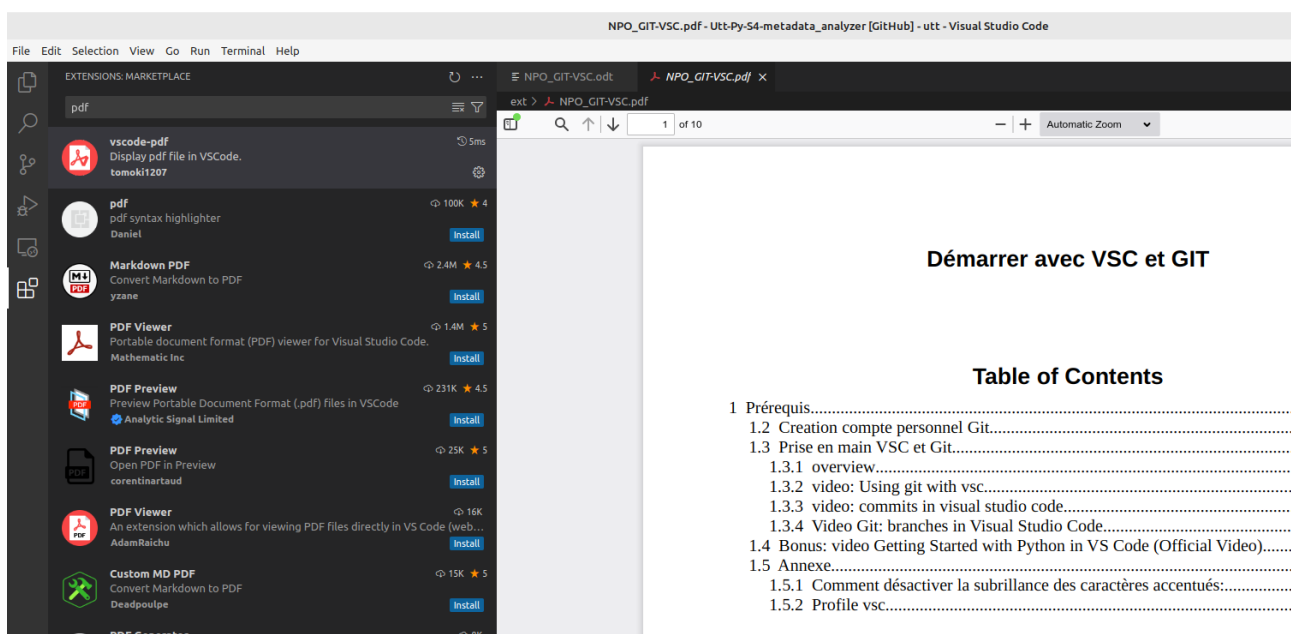
```
// na função main/outra ** (não copiar este comentário) **
pthread_t tids[NUM_THREADS];
thread_params_t thread_params[NUM_THREADS];

// 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

1.10.3 Comment lire directement les .pdf dans VSC



The screenshot shows the Visual Studio Code interface. On the left, the 'EXTENSIONS: MARKETPLACE' sidebar is open, displaying a list of PDF-related extensions. The 'vscode-pdf' extension by tomoki1207 is highlighted. The main editor area shows a PDF document titled 'NPO_GIT-VSC.pdf' from the repository 'NPO_GIT-VSC.pdf - Utt-Py-S4-metadata_analyzer [GitHub] - utt - Visual Studio Code'. The document content includes a 'Table of Contents' section with the following items:

- 1 Prérequis.....
- 1.2 Creation compte personnel Git.....
- 1.3 Prise en main VSC et Git.....
 - 1.3.1 overview.....
 - 1.3.2 video: Using git with vsc.....
 - 1.3.3 video: commits in visual studio code.....
 - 1.3.4 Video Git: branches in Visual Studio Code.....
- 1.4 Bonus: video Getting Started with Python in VS Code (Official Video).....
- 1.5 Annexe.....
 - 1.5.1 Comment désactiver la subrillance des caractères accentués.....
 - 1.5.2 Profile vsc.....

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

1.10.4 Comment lire directement les docx et .odf (fichier libreoffice writer) dans VSC

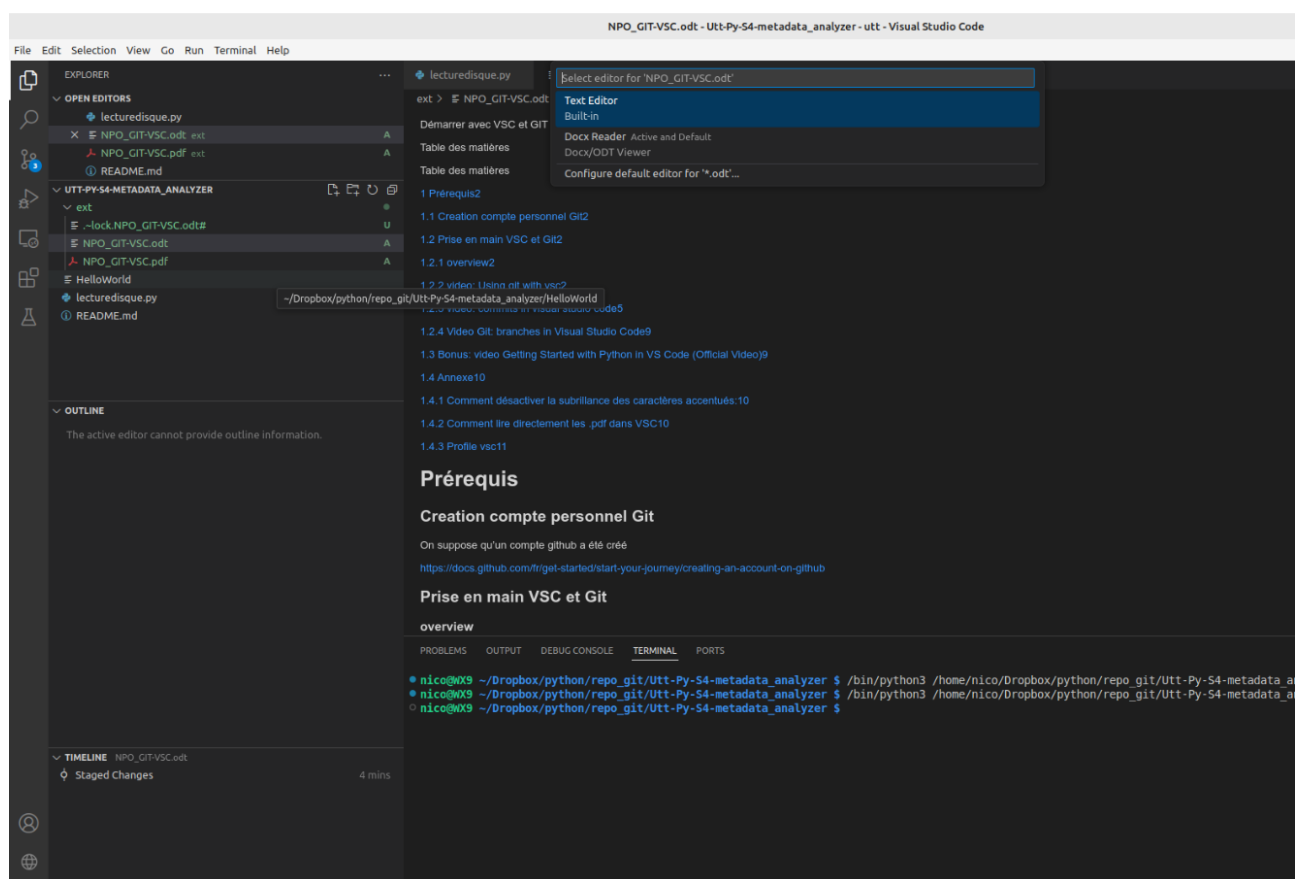
Installer l'extension docx/ODT viewer

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...

1.10.5 Comment installer une version python qui n'est pas celle de son systeme par défaut

Précautions : Modifier la version par défaut de Python sur votre système peut affecter des scripts ou des applications qui dépendent d'une version spécifique. Assurez-vous de tester vos environnements après de telles modifications.



Gérer les Versions de Python avec Pyenv

<https://blog.stephane-robert.info/docs/developper/programmation/python/pyenv/> pyenv install 3.12.0

installation des dépendences nécessaires à pyenc

sudo apt install -y make build-essential libssl-dev zlib1g-dev \

libbz2-dev libreadline-dev libsqlite3-dev wget curl llvm \

libncurses5-dev libncursesw5-dev xz-utils tk-dev libffi-dev \
liblzma-dev python3-openssl

Télécharge le script

```
curl https://pyenv.run | bash
```

mise à jour du .bashrc:

```
# Load pyenv automatically by appending  
# the following to  
# ~/.bash_profile if it exists, otherwise ~/.profile (for login shells)  
# and ~/.bashrc (for interactive shells) :
```

```
export PYENV_ROOT="$HOME/.pyenv"  
[[ -d $PYENV_ROOT/bin ]] && export PATH="$PYENV_ROOT/bin:$PATH"  
eval "$(pyenv init - bash)"
```

```
# Restart your shell for the changes to take effect.
```

```
# Load pyenv-virtualenv automatically by adding  
# the following to ~/.bashrc:
```

```
eval "$(pyenv virtualenv-init -)"
```

installation de python 3.12 sur le systeme:

```
nico@WX9 ~ $ pyenv install 3.12.0
```

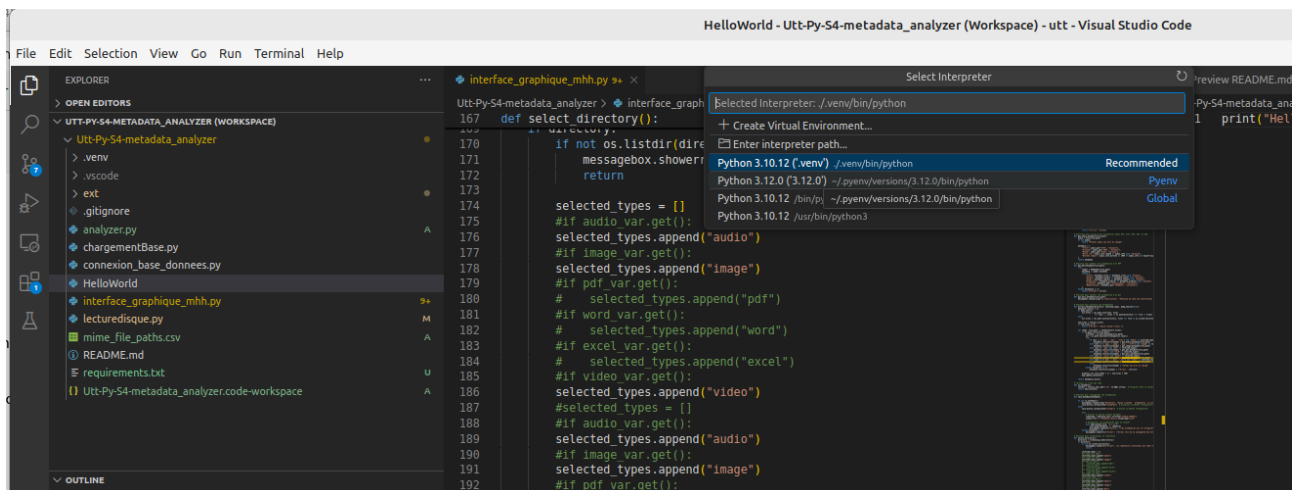
```
Downloading Python-3.12.0.tar.xz...
```

```
-> https://www.python.org/ftp/python/3.12.0/Python-3.12.0.tar.xz
```

```
Installing Python-3.12.0...
```

```
Installed Python-3.12.0 to /home/nico/.pyenv/versions/3.12.0
```

selection de la version via command palette



je veux une version 3.12 dans ce repertoire

(.venv) nico@WX9 ~/Dropbox/python/repo_git/Utt-Py-S4-metadata_analyzer \$

poetry

<https://www.youtube.com/watch?v=V5rKVrVhEh8>

uv semble prometteur

<https://www.youtube.com/watch?v=k4qh83m1jg0>

nico@WX9 ~ \$ python3 --version

Python 3.10.12

mon projet a besoin de Python 3.12

gerer les versions de python globale et locales pyenv

<https://www.youtube.com/watch?v=1Zgo8M9yUtM>

pour recompiler python3.12 d'après les sources

<https://www.youtube.com/watch?v=X0Oh8weERmI>

1.10.6 raccourci utiles

CTRL k MAJ P command palette (super pratique)

CTRL + - augmenter / diminuer la taille de la police

CTRL k MAJ t theme

activity bar :

CTRL k MAJ o open file

CTRL k MAJ F find/search

CTRL k MAJ G Git (source control)

CTRL k MAJ d Debug

CTRL k MAJ x extensions

CTRL k MAJ S shortcut preferences

CTRL J Panel avec terminal...

CTRL , Settings

CTRL MAJ ² create a new integrated terminal

CTRL MAJ C create a new external terminal

CTRL MAJ /

CTRL MAJ I format entire active file (install blackformatter)

CTRL k MAJ t theme

1.10.7 Profile vsc

En bas à gauche de l'activity barre clic 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

1.10.8 comment fixer l'erreur ModuleNotFoundError: No module named 'exceptions'

chatGPT=> The error occurs because the `exceptions` module is not a standard module in Python 3.x. It existed in Python 2.x but was removed in Python 3. If a library or code is still trying to import `exceptions`, it's likely not compatible with Python 3, especially with newer versions like 3.12.

```
(3.12.0) nico@WX9 ~/Dropbox/python/repo_git/Utt-Py-S4-metadata_analyzer $ pip install docx
```

```
Requirement already satisfied: docx in /home/nico/.pyenv/versions/3.12.0/lib/python3.12/site-packages (0.2.4)
```

```
Requirement already satisfied: lxml in /home/nico/.pyenv/versions/3.12.0/lib/python3.12/site-packages (from docx) (5.3.0)
```

```
Requirement already satisfied: Pillow>=2.0 in /home/nico/.pyenv/versions/3.12.0/lib/python3.12/site-packages (from docx) (11.1.0)
```

```
(3.12.0) nico@WX9 ~/Dropbox/python/repo_git/Utt-Py-S4-metadata_analyzer $ ^C
```

```
(3.12.0) nico@WX9 ~/Dropbox/python/repo_git/Utt-Py-S4-metadata_analyzer $ pip install --upgrade python-docx
```

```
Collecting python-docx
```

```
Using cached python_docx-1.1.2-py3-none-any.whl.metadata (2.0 kB)
```

```
Requirement already satisfied: lxml>=3.1.0 in /home/nico/.pyenv/versions/3.12.0/lib/python3.12/site-packages (from python-docx) (5.3.0)
```

```
Collecting typing-extensions>=4.9.0 (from python-docx)
```

```
Using cached typing_extensions-4.12.2-py3-none-any.whl.metadata (3.0 kB)
```

```
Using cached python_docx-1.1.2-py3-none-any.whl (244 kB)
```

```
Using cached typing_extensions-4.12.2-py3-none-any.whl (37 kB)
```

```
Installing collected packages: typing-extensions, python-docx
```

```
Successfully installed python-docx-1.1.2 typing-extensions-4.12.2
```