UTILISATION DE GITHUB

PLATEFORME QUI PERMET DE VERSIONNER SON PROGRAMME ET DE TRAVAILLER EN ÉQUIPE.

GitHub est une plateforme de développement de logiciels qui permet aux développeurs de collaborer et de gérer des projets en utilisant Git, un système de contrôle de version. Les utilisateurs peuvent héberger des dépôts de code, suivre les modifications apportées à leur code et proposer des modifications à d'autres dépôts. GitHub offre également des outils pour la gestion de projets, la documentation, la création de pages Web... Il est utilisé par de nombreuses entreprises, organisations et individus pour le développement de logiciels open source et propriétaires.

INITIALISATION DE GITHUB

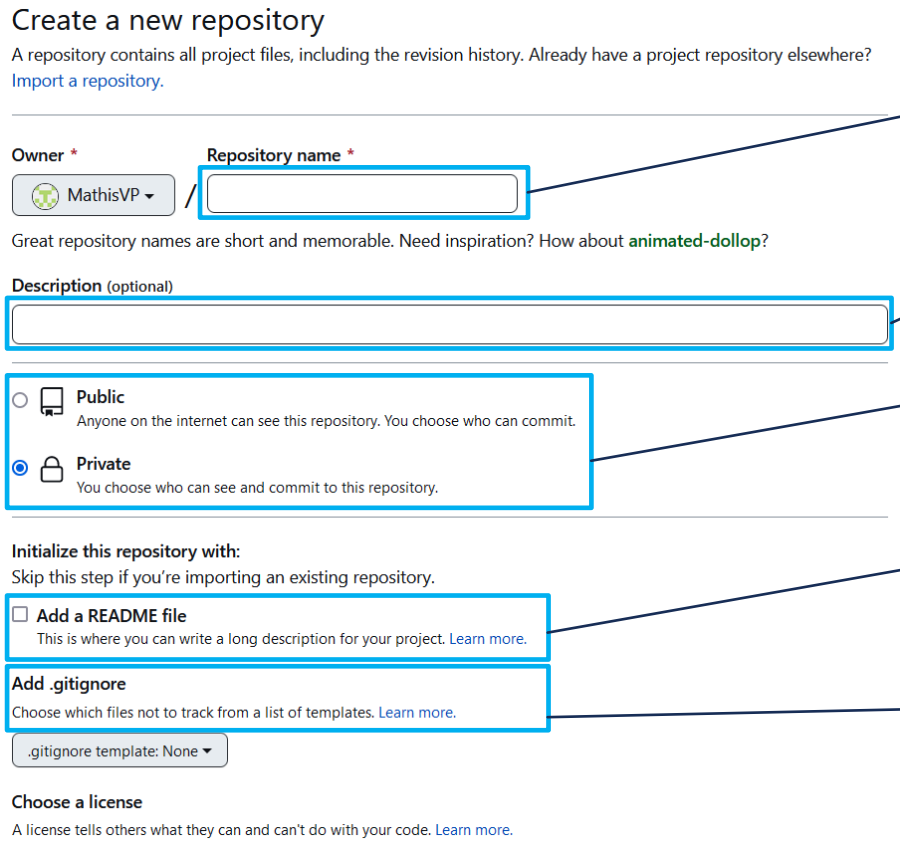
SETUP

Configuring user information used across all local repositories

`git config --global user.name "[firstname lastname]"`
set a name that is identifiable for credit when review version history
`git config --global user.email "[valid-email]"`
set an email address that will be associated with each history marker
`git config --global color.ui auto`
set automatic command line coloring for Git for easy reviewing

Conseil : Utiliser VS code comme IDE pour programmer.

CRÉATION D'UN REPOSITORIES



Donner le nom du repositories

Donner une description du projet

Mettre votre repositories en public / privée

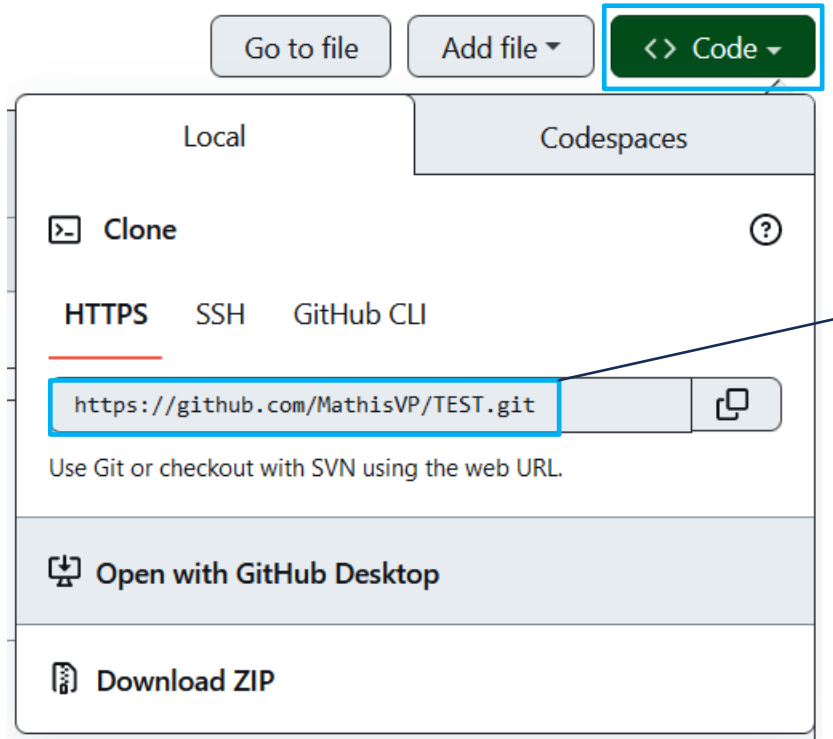
Ajouter un readme, (ça permet d'expliquer à quoi sert votre code par exemple)

Fichier qui permet de ne pas commit des fichiers qui nous intéresse pas. Exemple pour le langage python le fichier « __pycache__/ »

MISE DU REPOSITORIES EN LOCAL

`git clone [url]`

retrieve an entire repository from a hosted location via URL

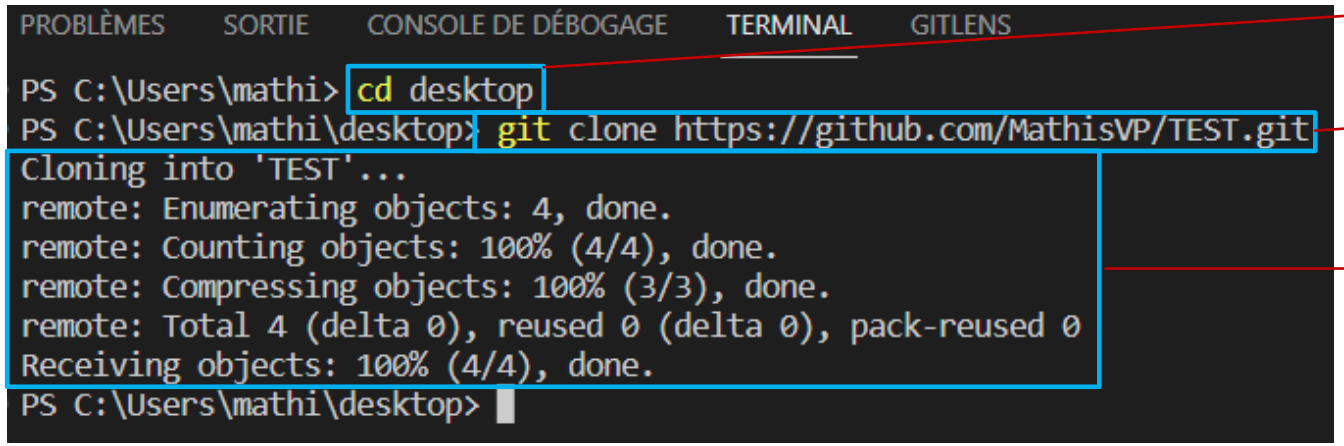


url du repositories

Commandes utiles pour naviguer dans le terminal

pwd	savoir dans quel dossier je suis
mkdir "dossier"	créer un dossier (Make Directory)
touch fichier.txt	créer fichier
ls	liste le dossier courant
ls -la	liste tout plus précisément que ls
cd dossier	aller dans le dossier (Change Directory)
cd ..	Remonter d'un dossier

MISE DU REPOSITORIES EN LOCAL



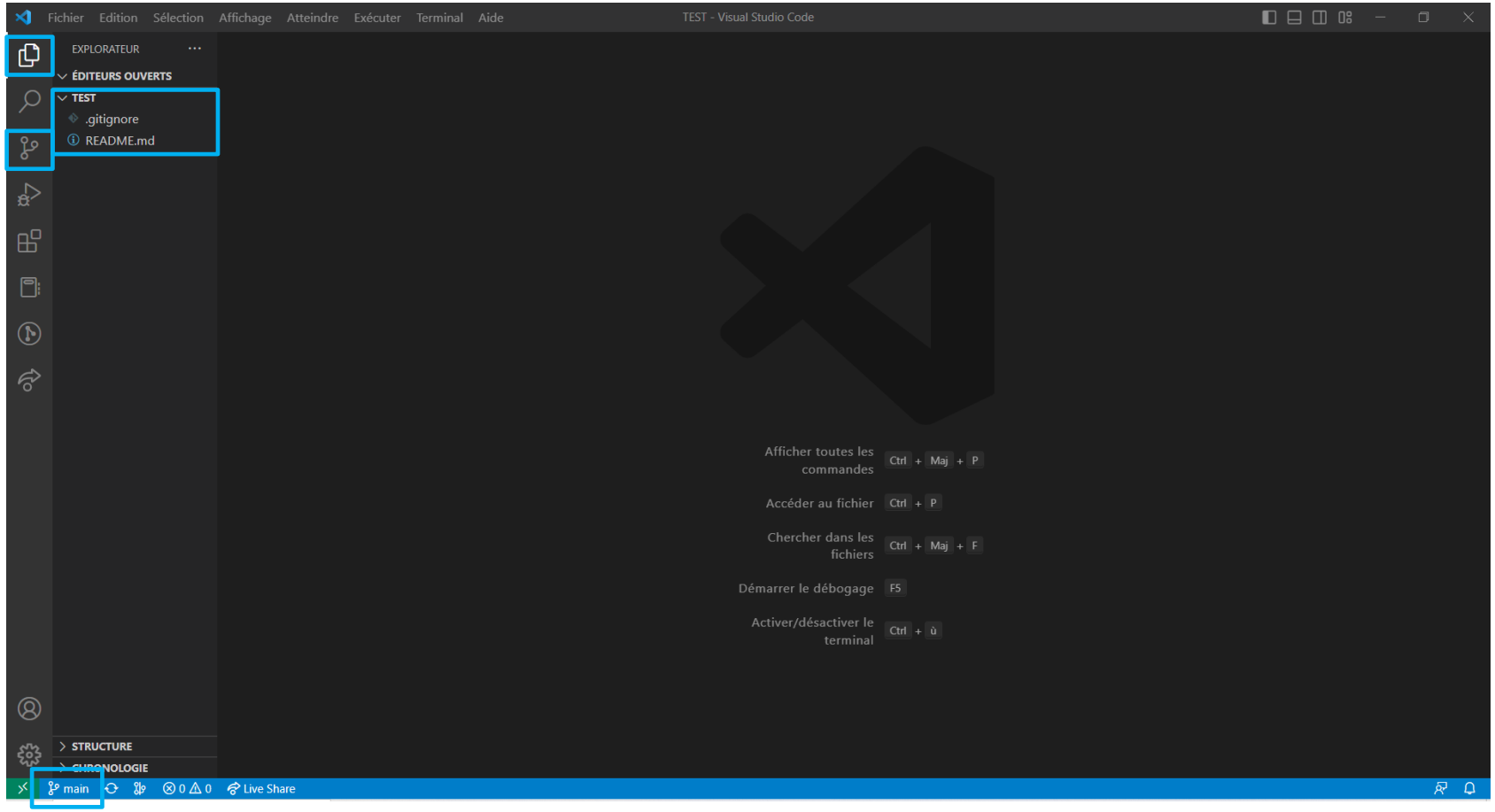
Changement de dossier

Cloner le dossier depuis le dépôt Git

Le dossier est bien cloné (ici sur le bureau / desktop)

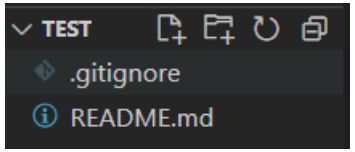
Ouvrir ensuite le dossier dans VS code

OUVRIR LE DOSSIER DANS VS CODE



Explorateur de fichier

Contrôle de code source

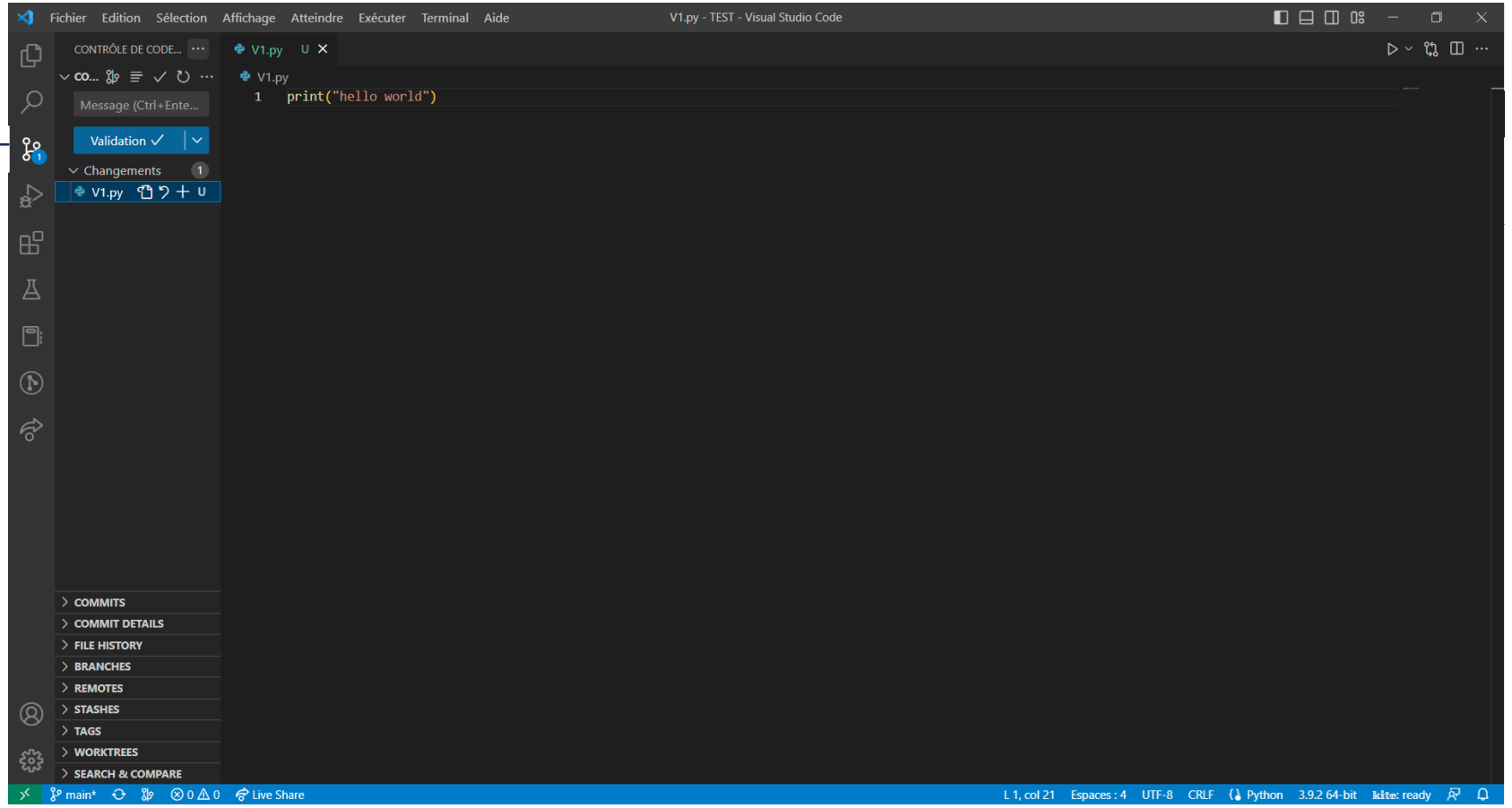


Fichiers présent dans le repositories

Branche de travail

TRAVAILLER SUR LA BRANCHE « MAIN »

Le 1 signifie qu'il y a un fichier qui a été modifié / ajouté
Ce fichier a été ajouté / modifié en **local** (sur le repositories git il n'apparaîtra pas). La branche « main » a été modifiée (nous reviendrons plus tard sur les branches).



Pour que les modifications soient ajoutées sur le repositories GitHub il faut l'ajouter, le commit et le push. Nous allons détailler ces étapes.

➢ Ajout :

Deux méthodes :

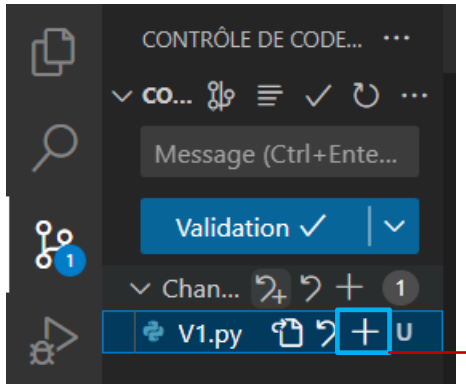
1er méthode avec les commandes :

git add .	ajoute toutes les modifications (le . symbolise tout)

git add [file]

add a file as it looks now to your next commit (stage)

2ème méthode avec VS code :



Le « + » permet d'ajouter le fichier

➢ Commit :

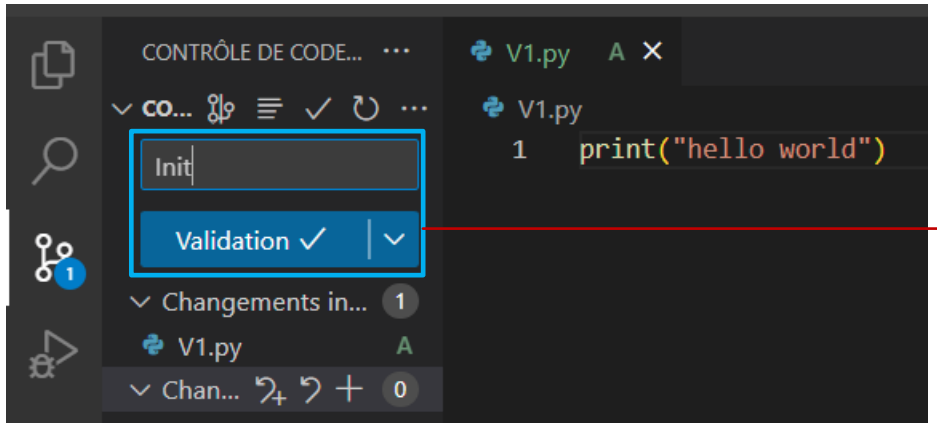
Deux méthodes :

1er méthode avec les commandes :

```
git commit -m "[descriptive message]"
```

commit your staged content as a new commit snapshot

2ème méthode avec VS code :



Mettre le message et appuyer sur « validation »

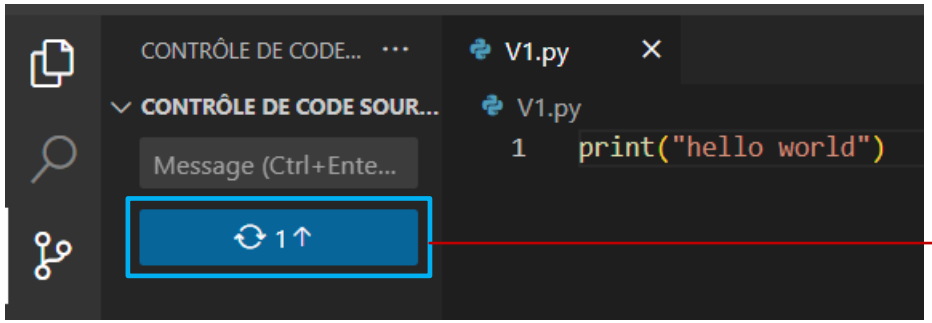
➢ Push :

Deux méthodes :

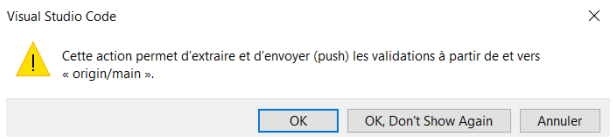
1er méthode avec les commandes :

```
git push [alias] [branch]

Transmit local branch commits to the remote repository branch
```

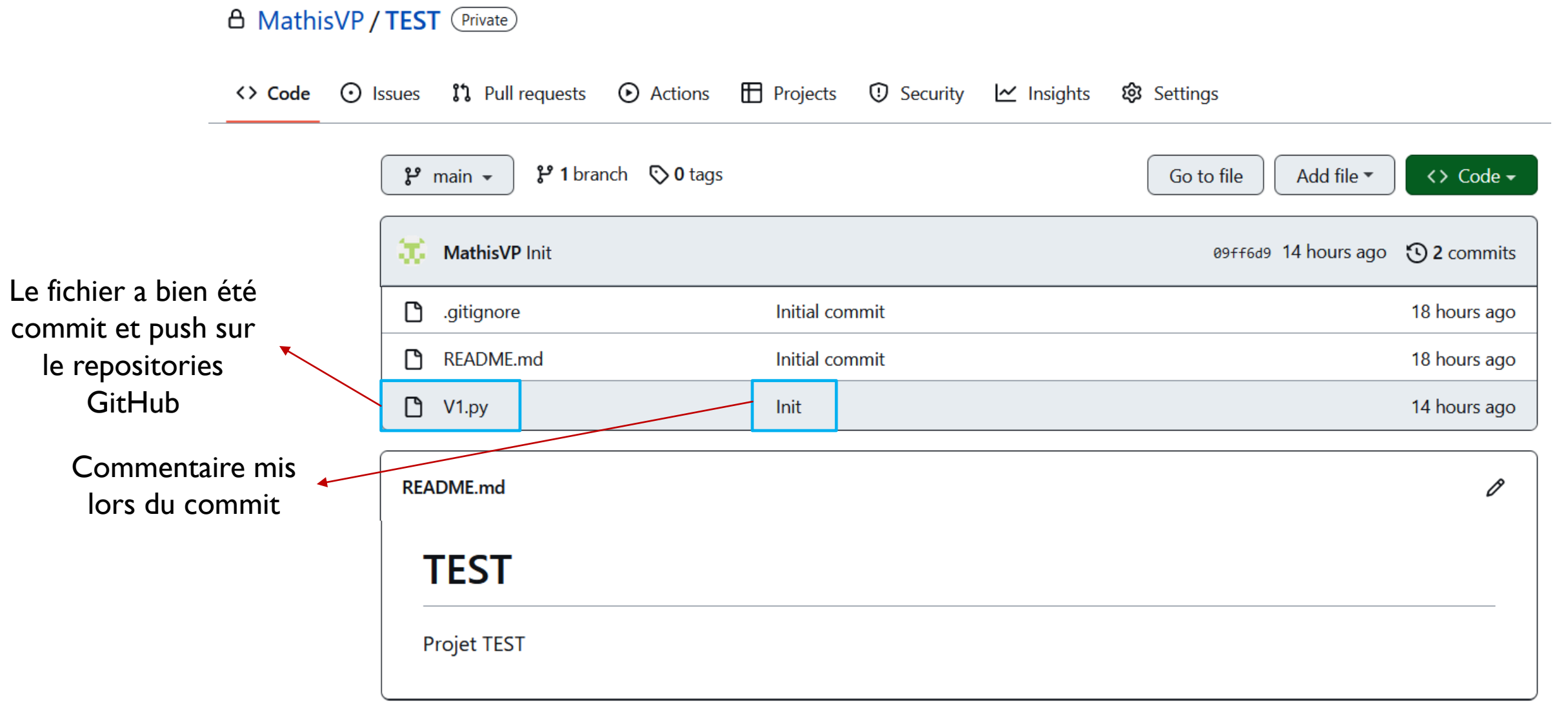
2ème méthode avec VS code :

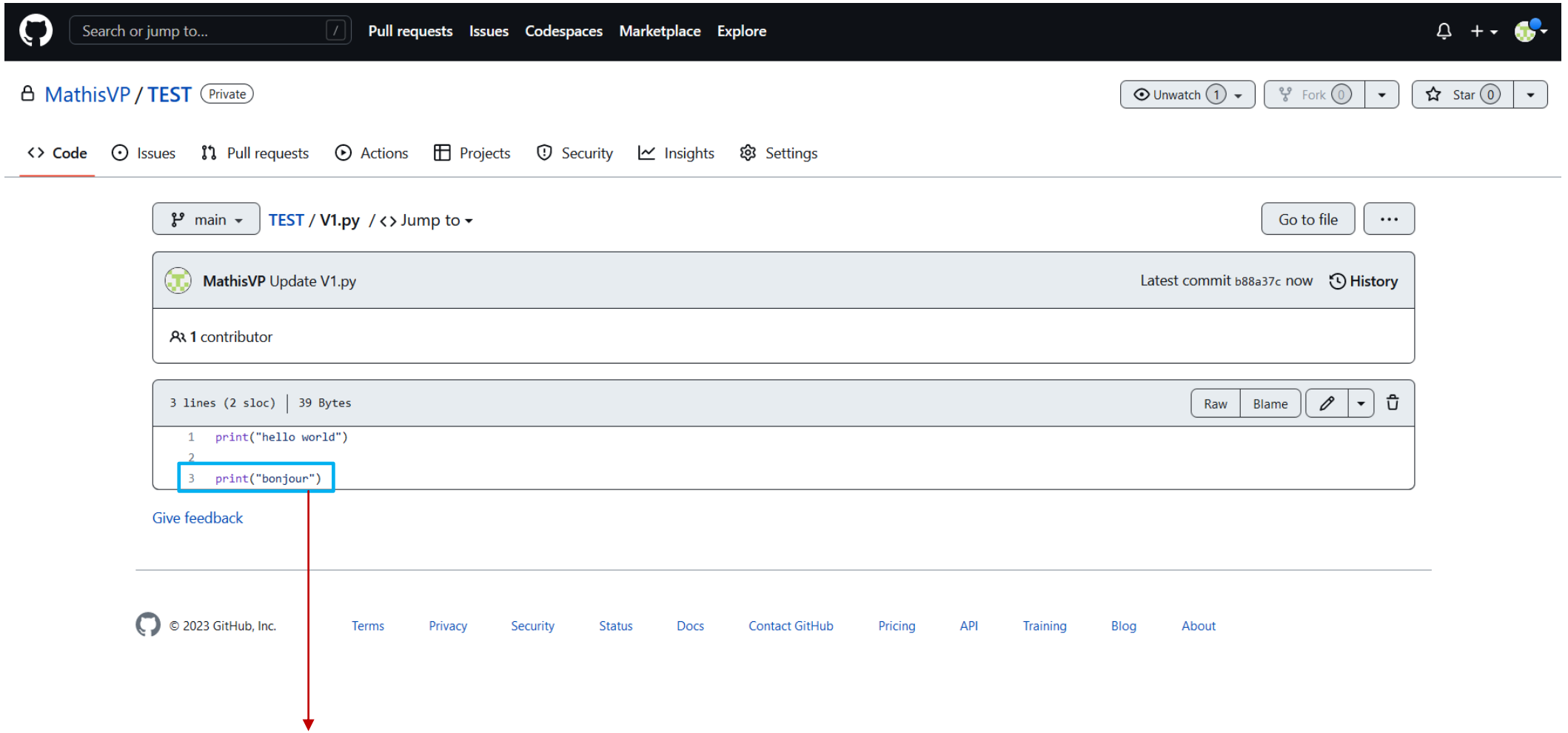


Appuyer sur ce bouton pour push

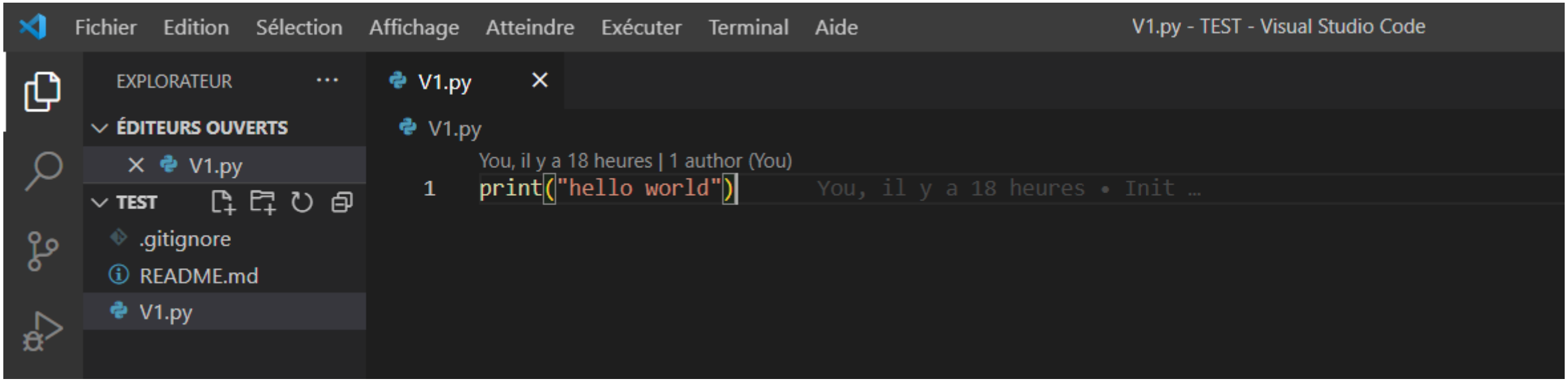


Appuyer sur « ok »


MathisVP / TEST
Private
Code
Issues
Pull requests
Actions
Projects
Security
Insights
Settings
main
1 branch
0 tags
Go to file
Add file
Code
MathisVP Init
09ff6d9 14 hours ago
2 commits
.gitignore
Initial commit
18 hours ago
README.md
Initial commit
18 hours ago
V1.py
Init
14 hours ago
README.md
TEST
Projet TEST
Le fichier a bien été commit et push sur le repositories GitHub
Commentaire mis lors du commit



Modification apportée au fichier



Les modifications ne sont toujours pas arrivées en local

Pour que les modifications soient ajoutées en local, il faut pull.

- **Pull :**

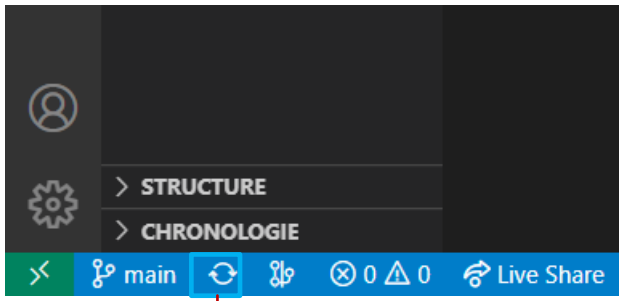
Deux méthodes :

1er méthode avec les commandes :

```
git pull
```

fetch and merge any commits from the tracking remote branch

2ème méthode avec VS code :

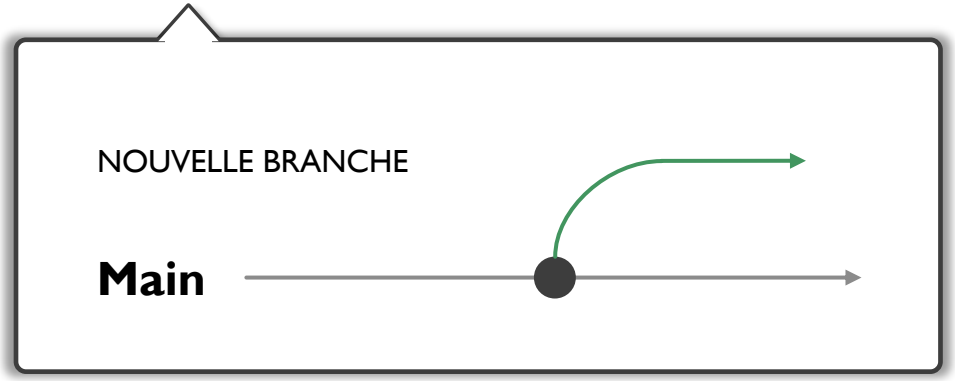


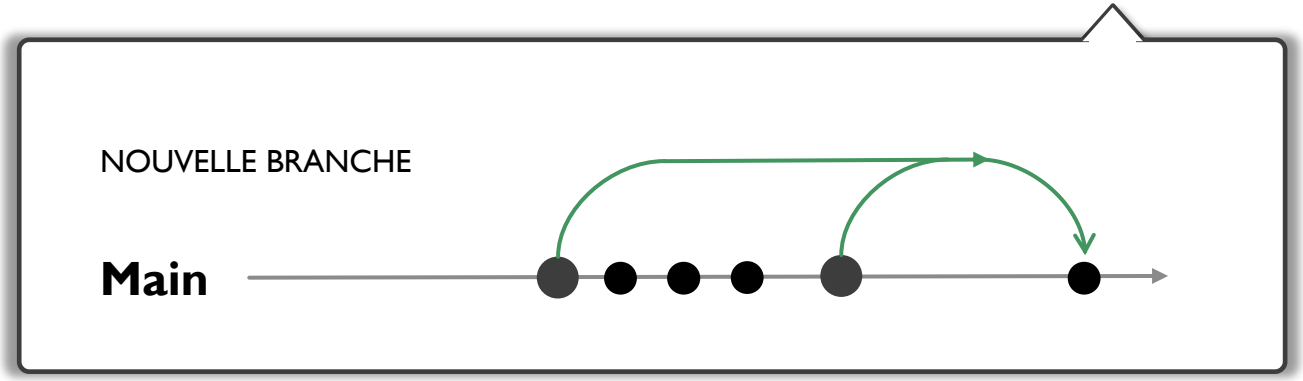
Appuyer sur ce bouton pour pull
(puis sur « ok »)

GESTION DE BRANCHES

La gestion des branches est primordiale en programmation agile. Les branches servent à ne pas casser le code principal (sur la branche main). Personne ne doit coder sur cette branche. Aucun conflit ne doit être sur le main et elle (la branche) doit toujours être fonctionnelle.

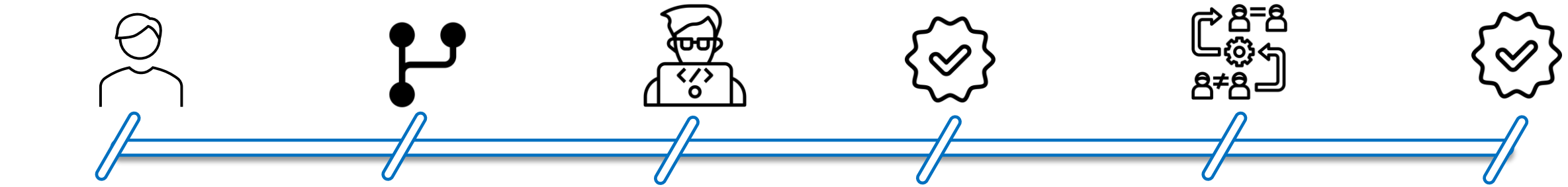
Utilité de la création de branche.





Sur le deuxième schéma, on constate que la personne 1 a créée une branche et qu'il y entre temps la personne 2 qui à push sur le main. La personne 1 va commit et push sur sa branche (faire cette action régulièrement avec des messages explicites lors du commit). Par la suite, la personne 1 va devoir pull avant de push sur le main (pour récupérer ce qui a été modifier sur le main). Des conflits vont apparaître. Il va devoir gérer ces conflits sur sa branche (explication de comment faire par la suite) et lorsque les conflits sont gérés alors il va pouvoir push sur le main.

PROGRAMMATION AGILE



CRÉATION D'UNE BRANCHE

Deux méthodes :

1er méthode avec les commandes :

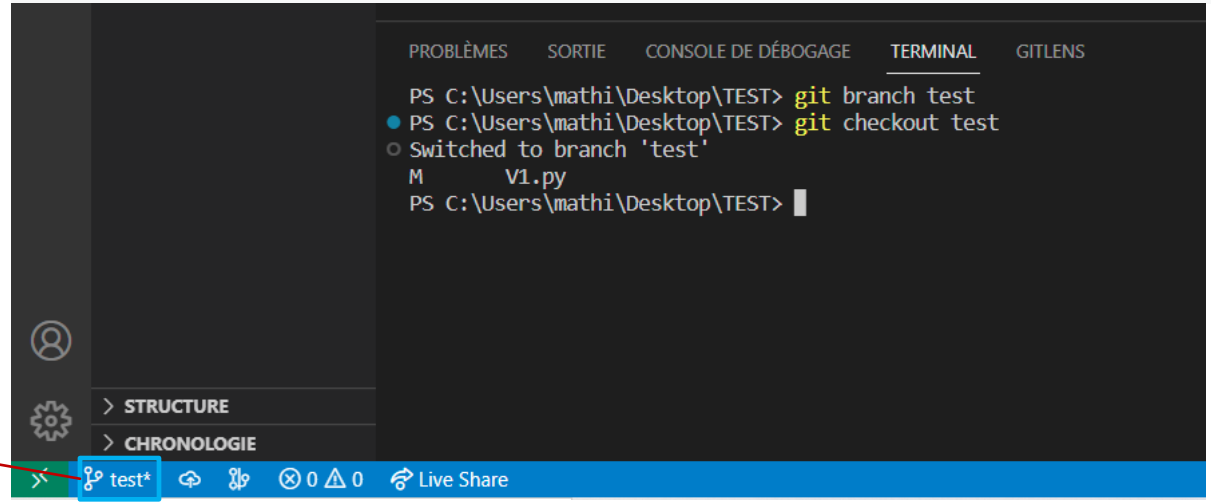
```
git branch [branch-name]
```

create a new branch at the current commit

Naviguer à travers les branches :

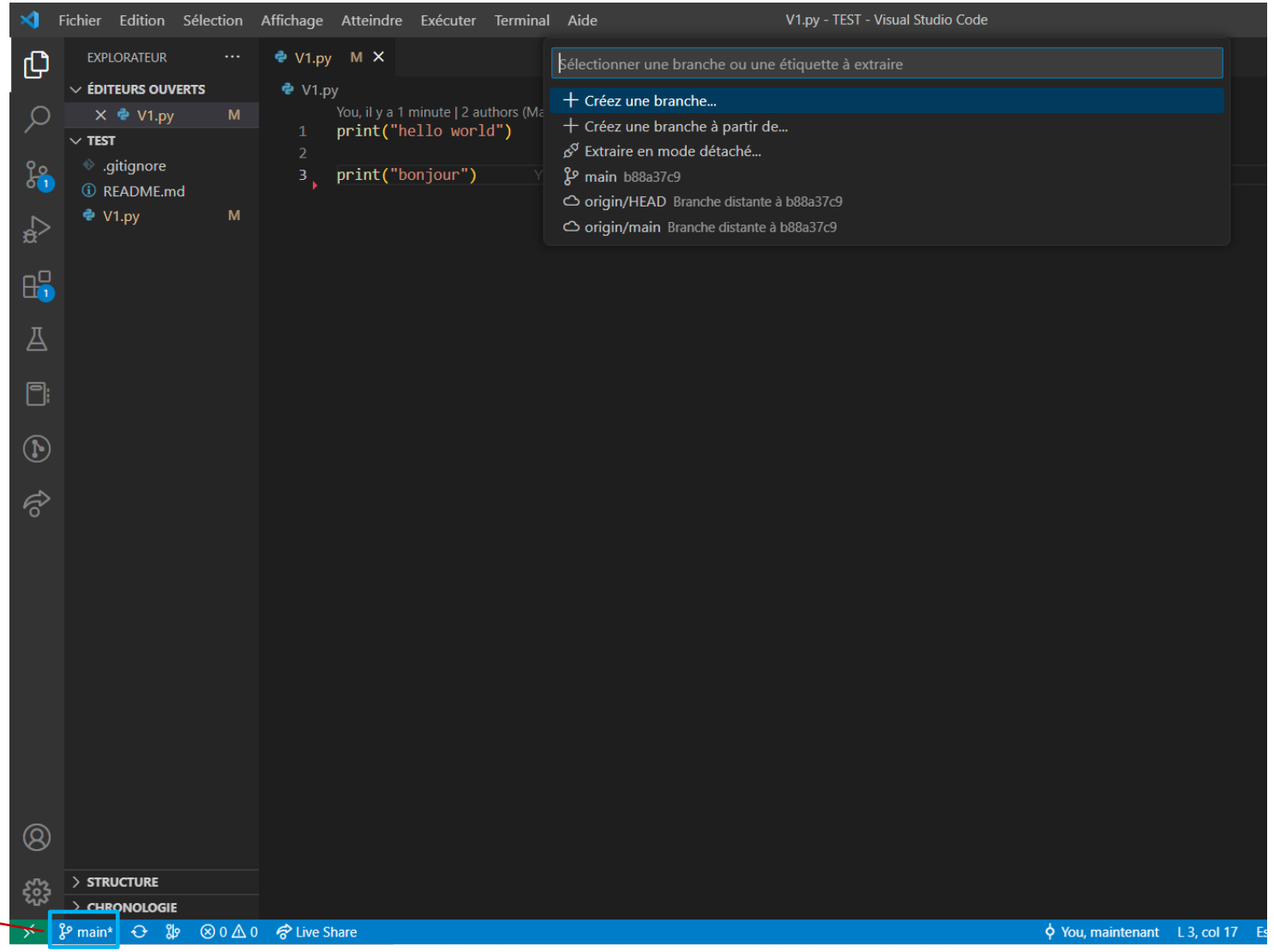
```
git checkout
```

switch to another branch and check it out into your working directory



J'ai bien changé de branche

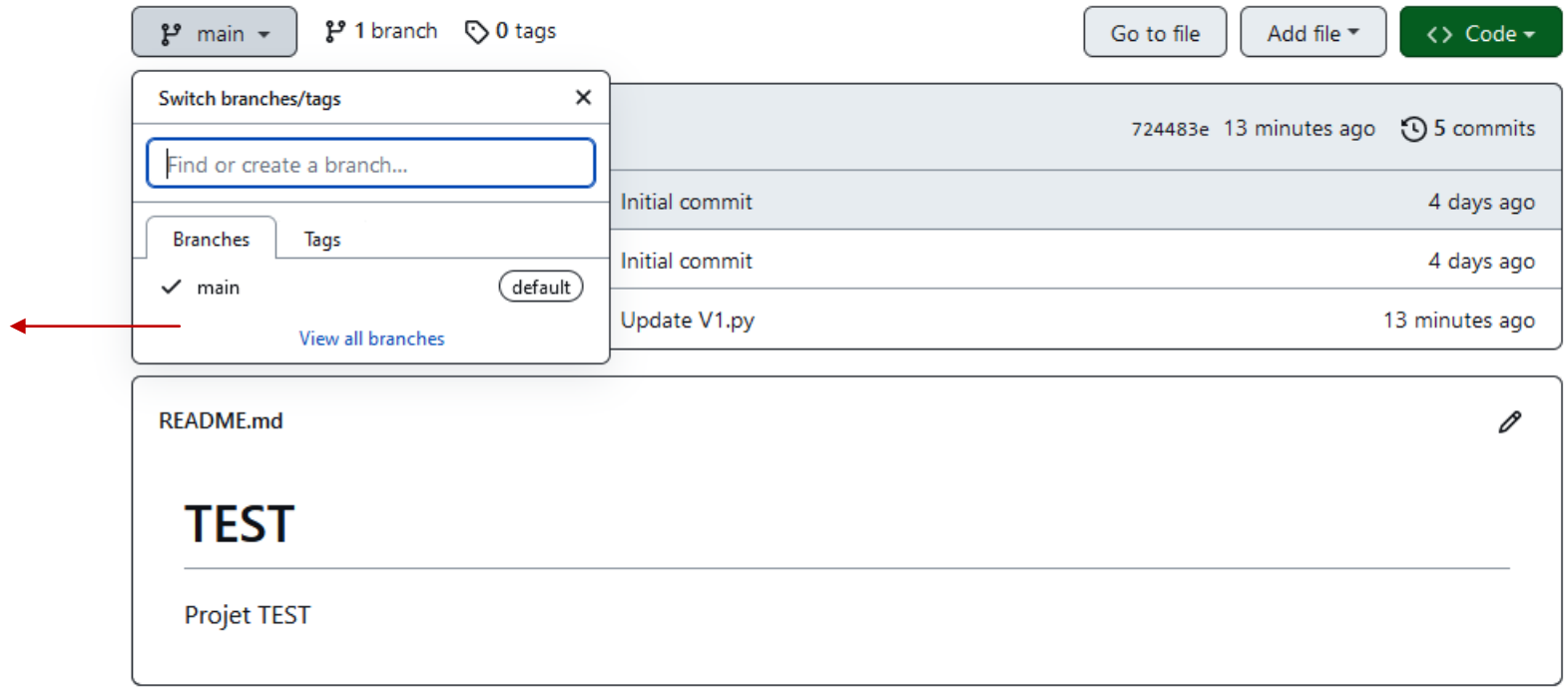
2ème méthode avec VS code :

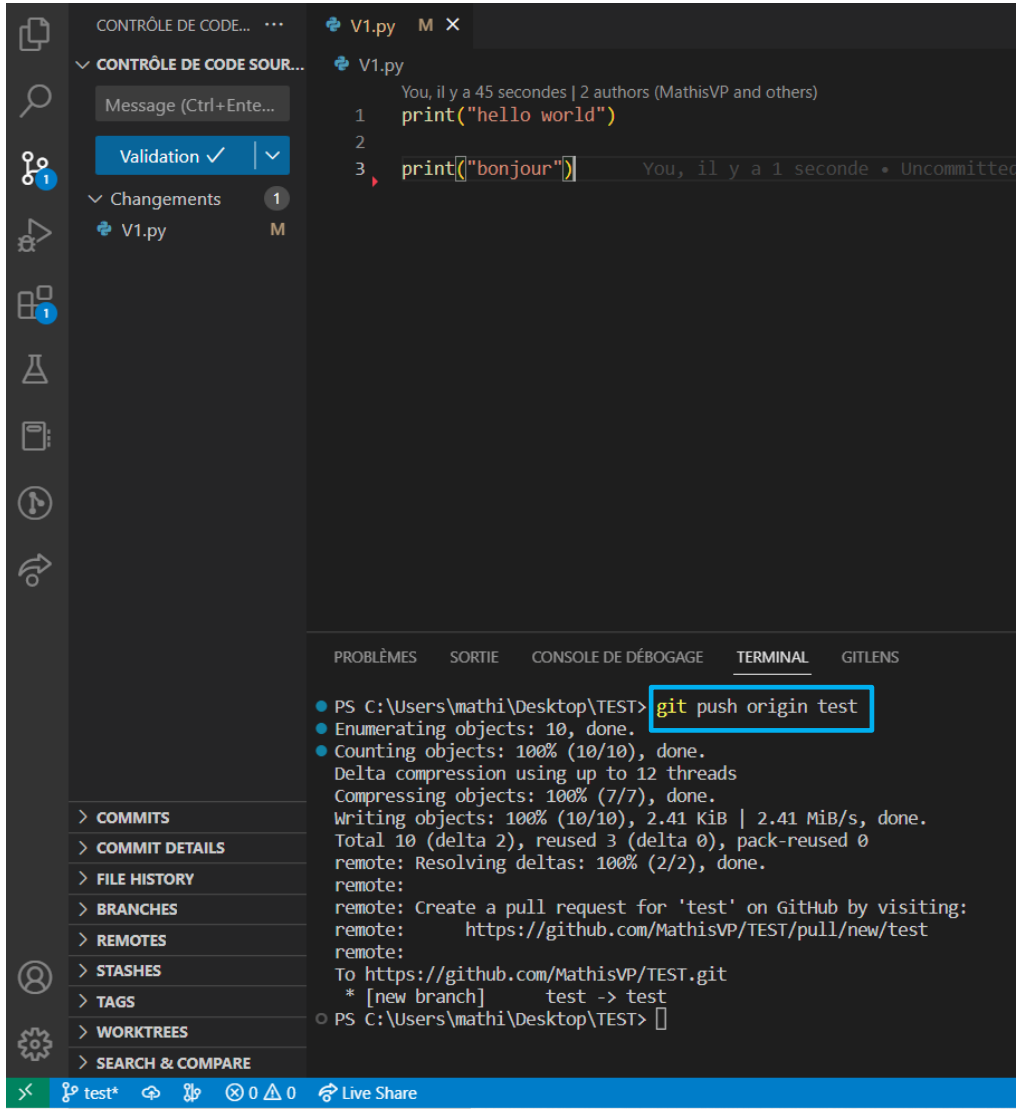


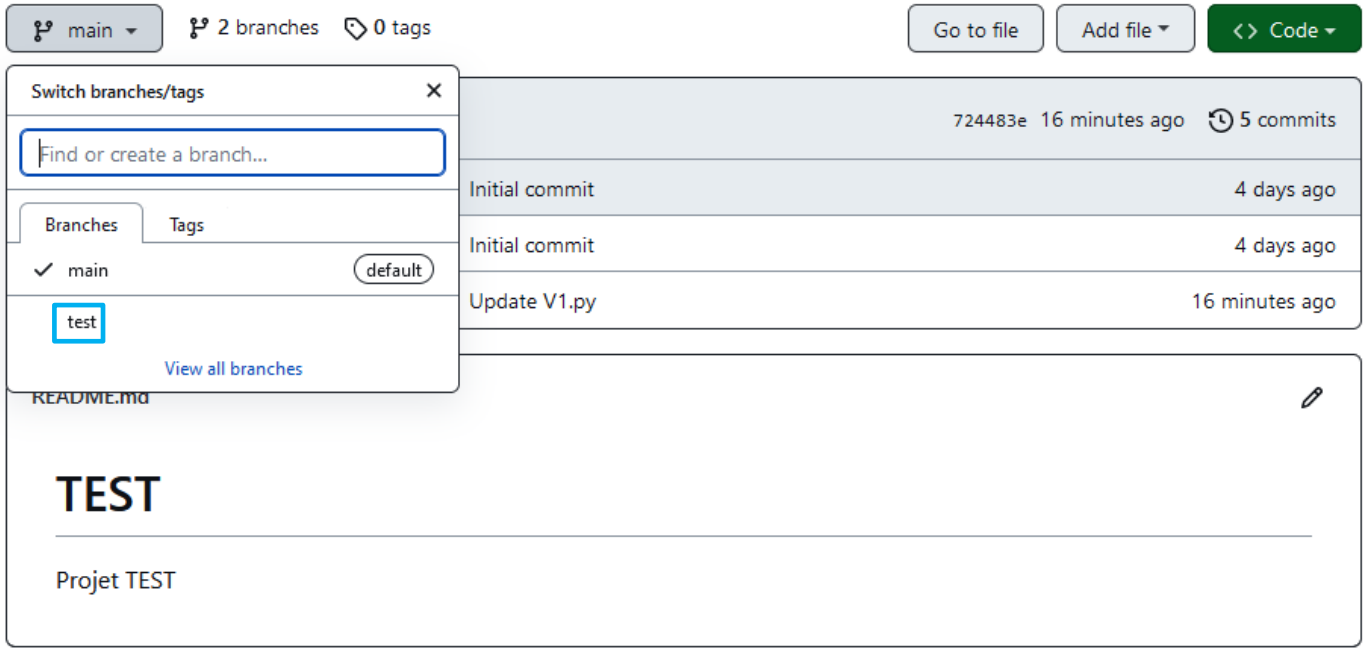
Cliqué ici et créez une nouvelle branche

Mettre la nouvelle branche local (test) sur le repositories GitHub.

La branche test n’est pas mise sur GitHub.







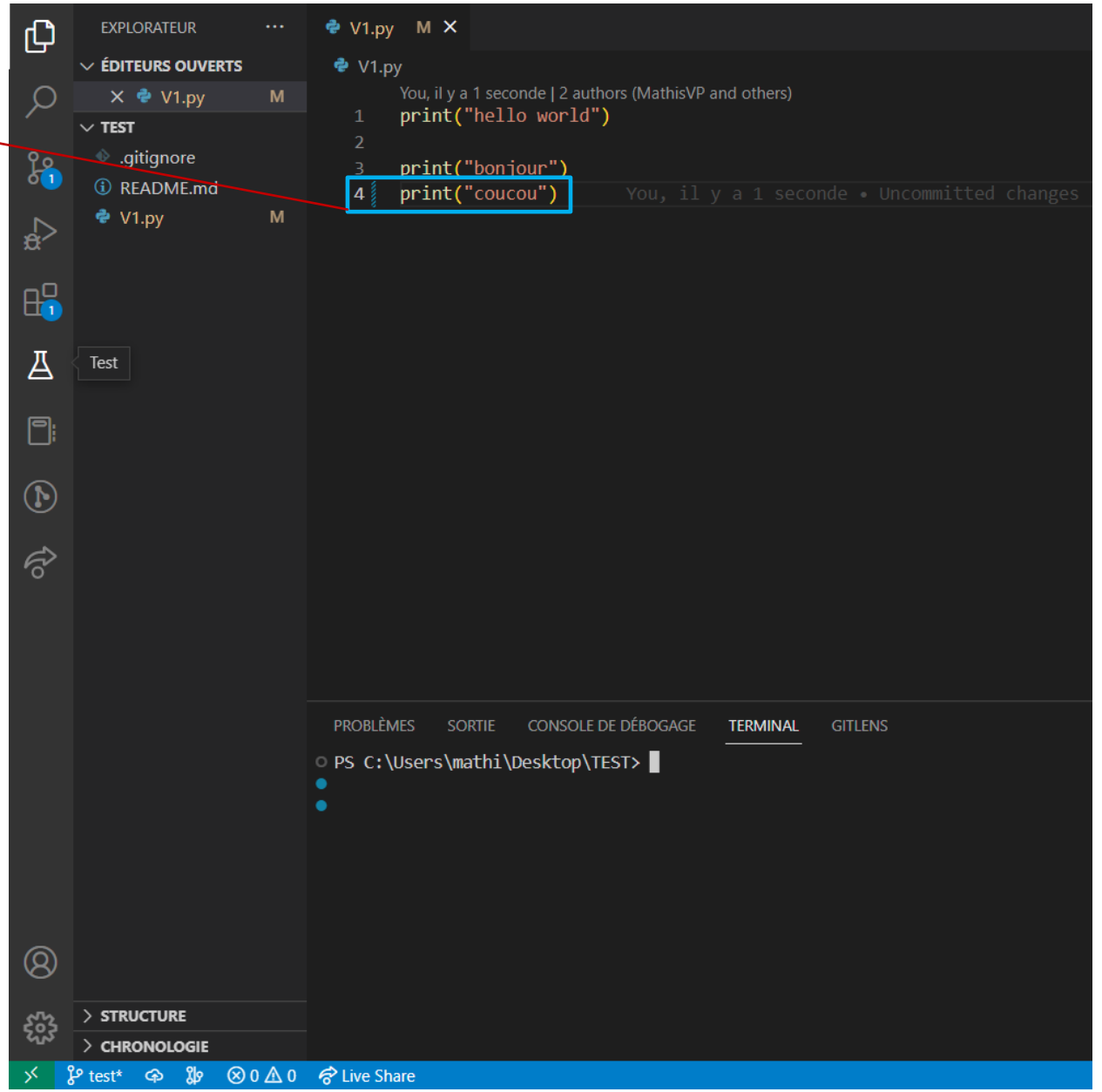
La branche a bien été ajoutée au repositories GitHub.

Modification du code (en local) sur la branche « test »

Il faut ajouter, commit et push le code sur notre branche. (C'est ce que fais le code ci-dessous)

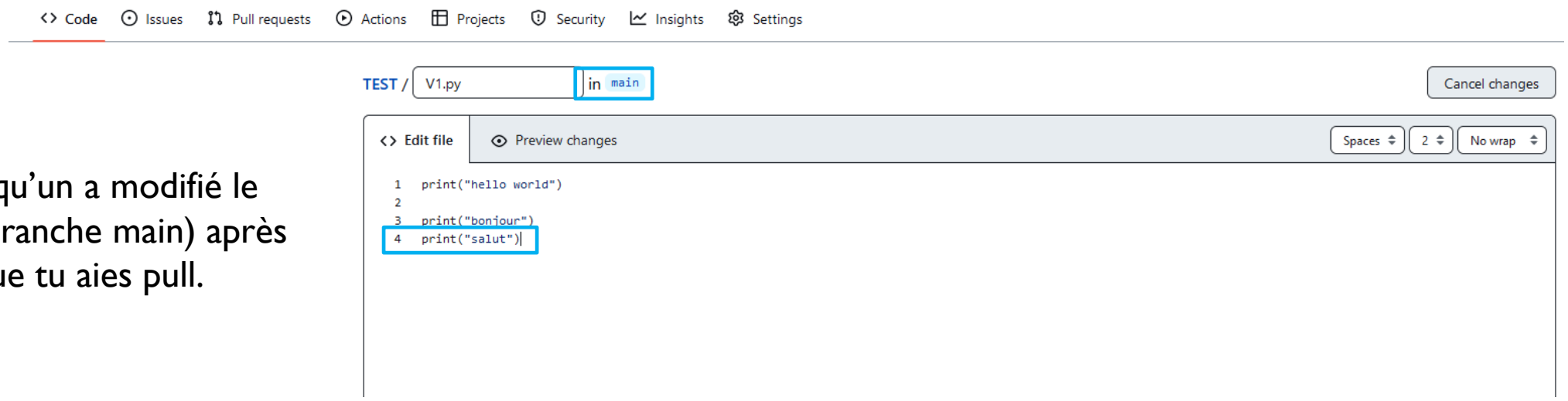
```
PS C:\Users\mathi\Desktop\TEST> git add .
PS C:\Users\mathi\Desktop\TEST> git commit -m "nouveau print"
[test 6e1ed68] nouveau print
 1 file changed, 1 insertion(+)
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 291 bytes | 291.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/MathisVP/TEST.git
   b88a37c..6e1ed68  test -> test
```

```
PS C:\Users\mathi\Desktop\TEST> git push
fatal: The current branch test has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin test

PS C:\Users\mathi\Desktop\TEST> git push --set-upstream origin test
Everything up-to-date
Branch 'test' set up to track remote branch 'test' from 'origin'.
PS C:\Users\mathi\Desktop\TEST>
```

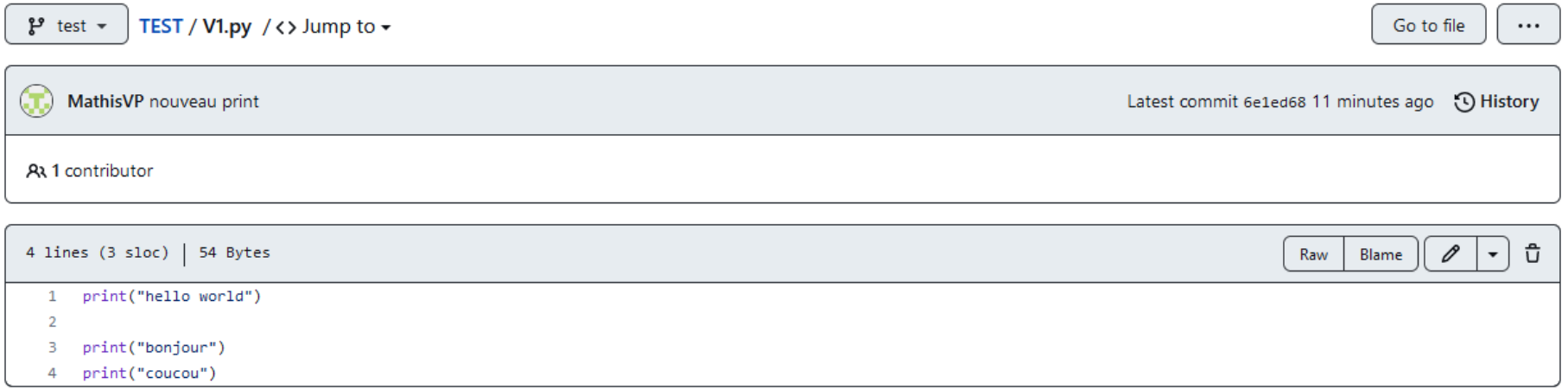


GESTION DES CONFLITS

Quelqu'un a modifié le code (branche main) après que tu aies pull.

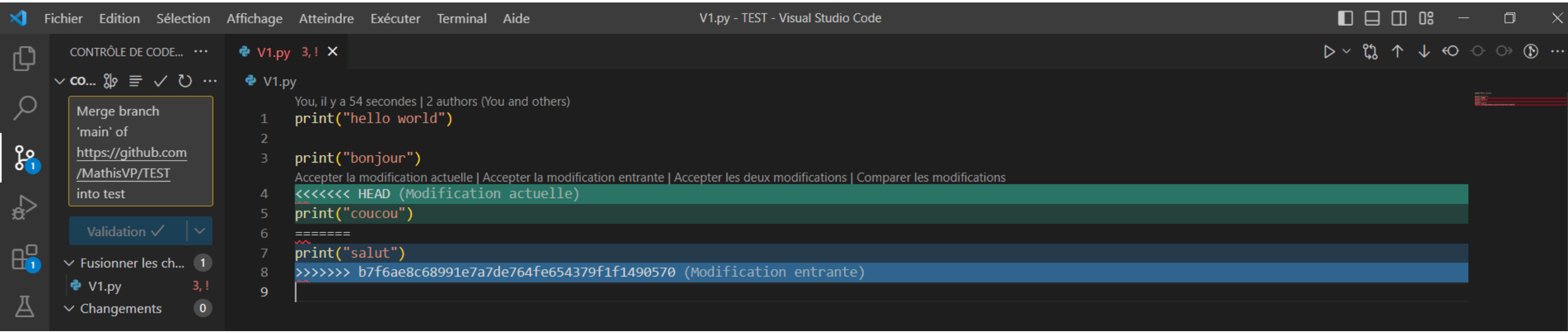


Sur ta branche, le code n'est pas le même et au moment où tu vas vouloir push sur le main il va y avoir des conflits.



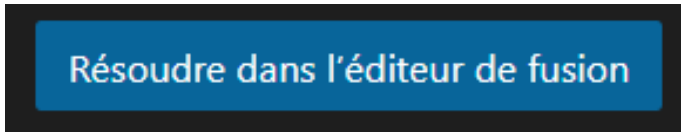
On pull se qui a été fait entre temps sur la branche main, et nous constatons une erreur « merge failed » nous devons donc régler des conflits.

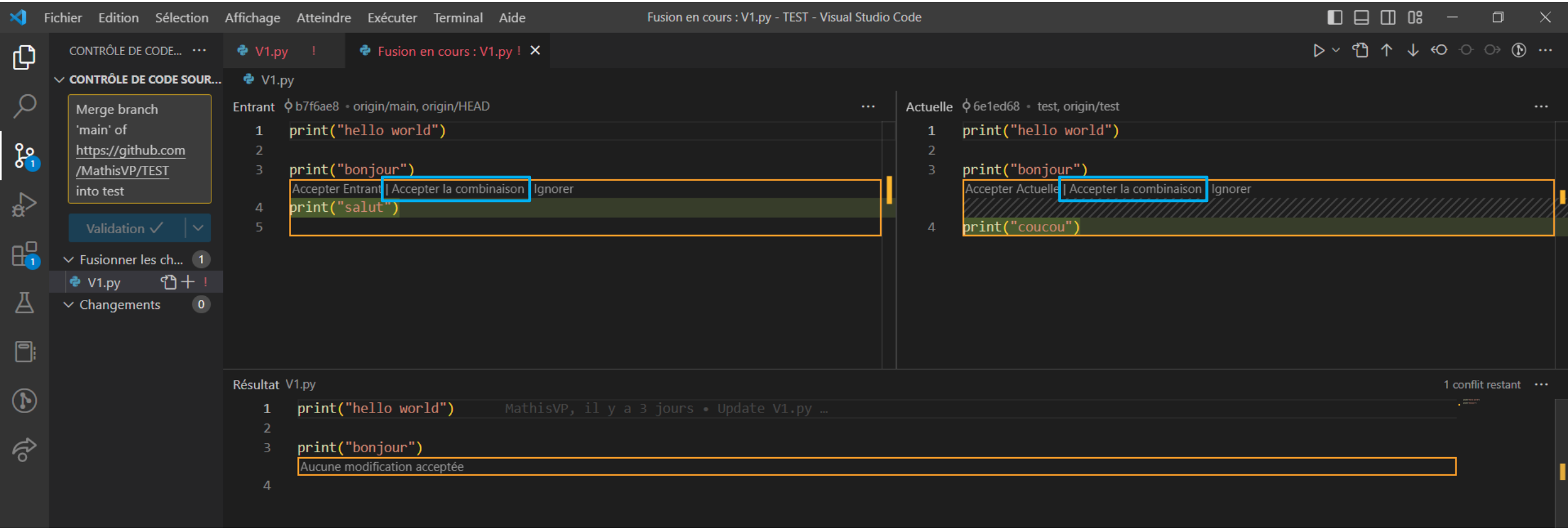
```
PS C:\Users\mathi\Desktop\TEST> git pull origin main
From https://github.com/MathisVP/TEST
 * branch            main       -> FETCH_HEAD
Auto-merging V1.py
CONFLICT (content): Merge conflict in V1.py
Automatic merge failed; fix conflicts and then commit the result.
PS C:\Users\mathi\Desktop\TEST>
```



VS Code nous aide en nous montrant les conflits. En vert la modification qui est sur ma branche et que je souhaite mettre sur le main et en bleu les modifications qu'il y a eu entre temps sur le main.

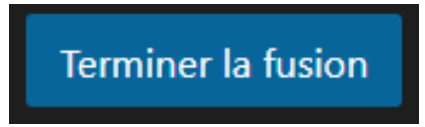
Appuyer sur ce bouton pour traiter les conflits

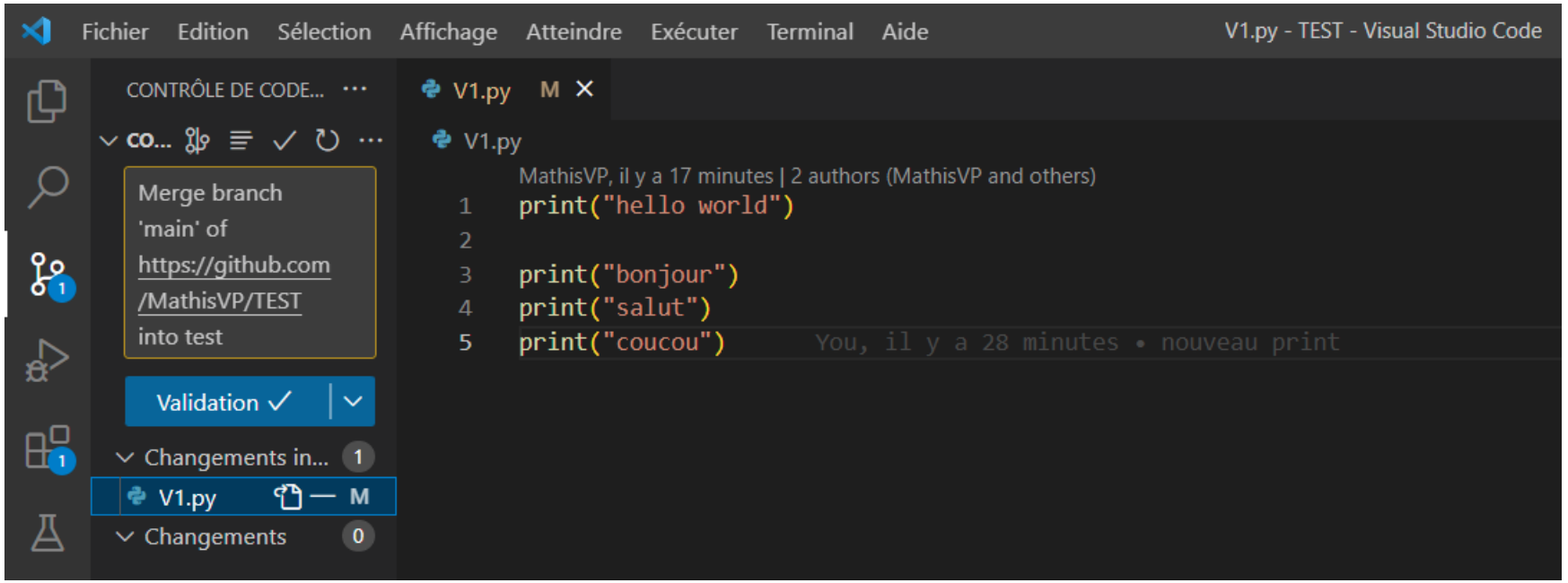




Cette fenêtre va s'ouvrir et vous pourrez traiter les conflits. Si vous voulez accepter les deux modifications alors cliquer sur « Accepter la combinaison ».

Cliquer sur ce bouton pour terminer



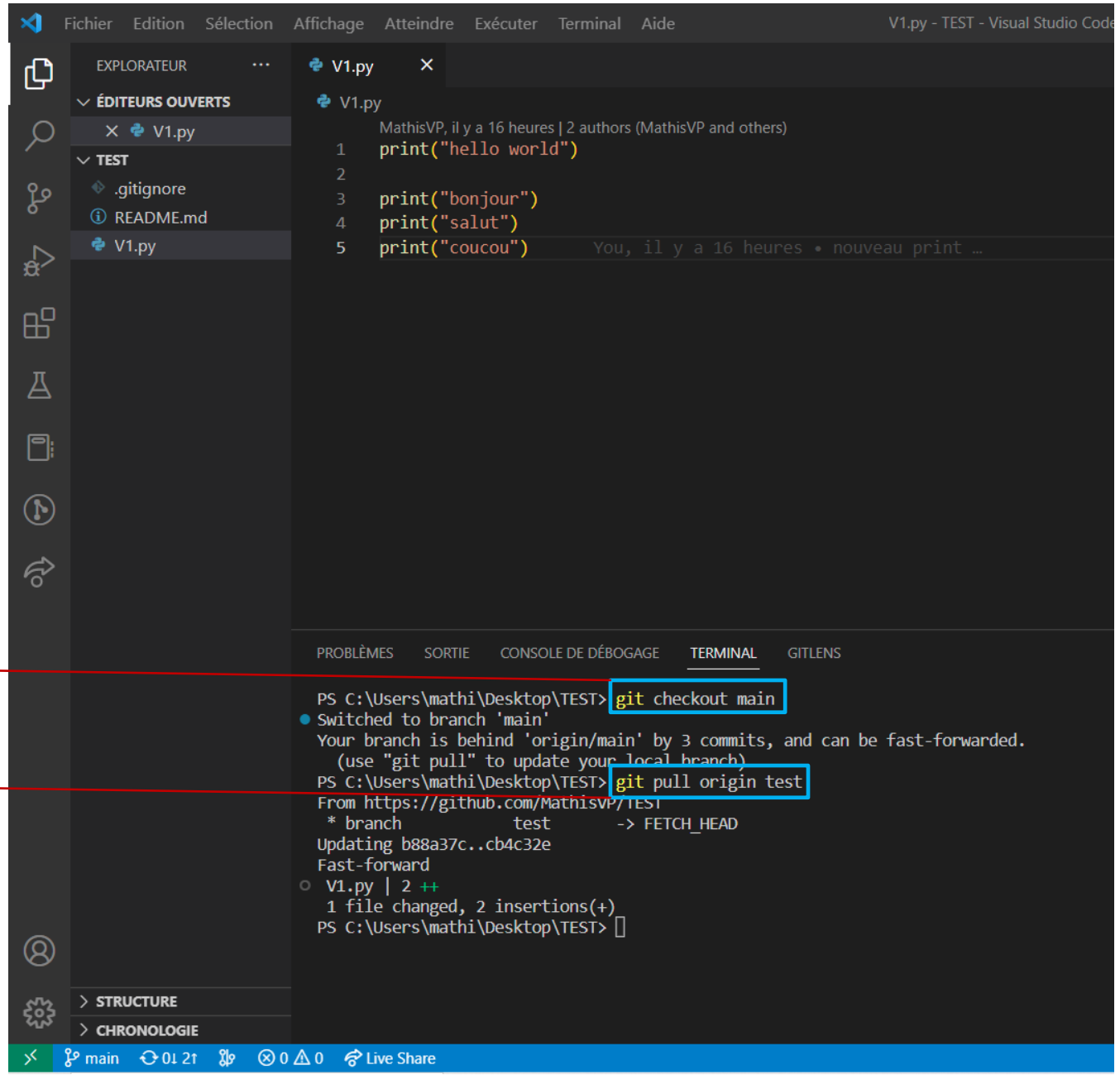


Voilà ce que vous aurez à la fin. Par la suite, il faut le commit et le push sur la branche test.

Pour mettre le code de la branche test sur la branche main :

Il faut changer de branche et passer sur le main

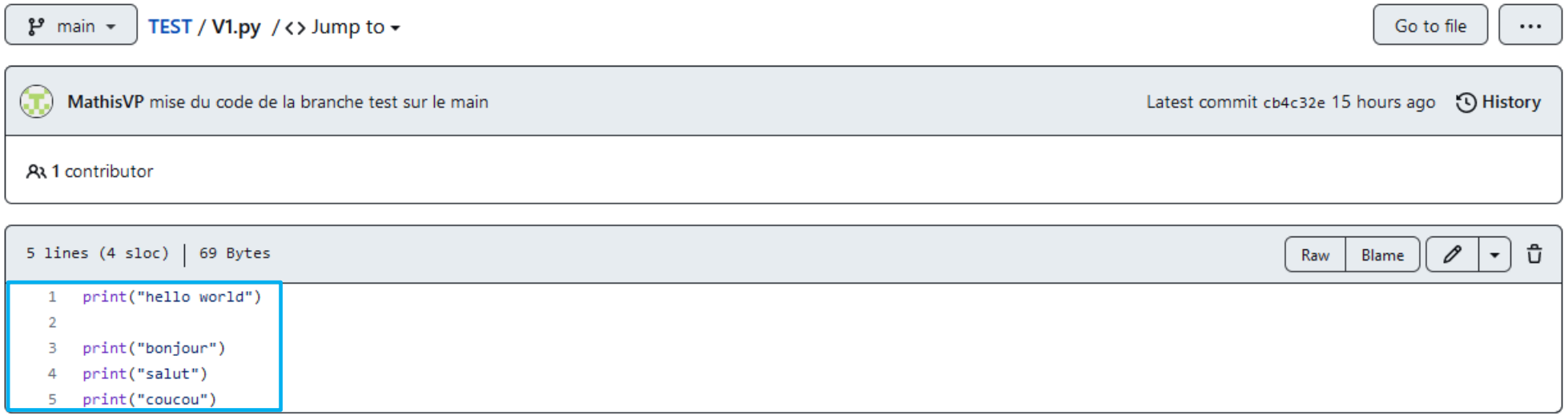
Il faut ensuite pull la branche test



Il faut ensuite push le main sur le repositories GitHub.

```
PS C:\Users\mathi\Desktop\TEST> git push
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/MathisVP/TEST.git
   b7f6ae8..cb4c32e  main -> main
```

Les modifications ont bien été prises en compte.



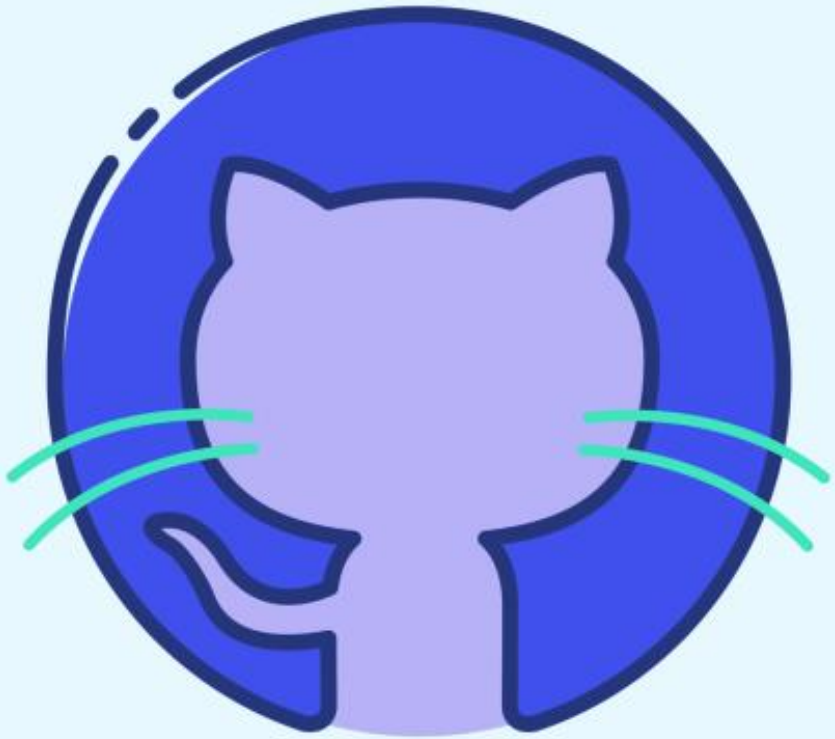
Autres commandes utiles :

```
git status
```

show modified files in working directory, staged for your next commit

```
git log
```

show all commits in the current branch's history



AUTEUR
MATHIS VEYRAT-PARISIEN
(PROMO IPSA 2025)

REMERCIEMENT
NICOLAS HEUDE
(PROMO EPITECH 2025)

LIENS UTILES :

- HTTPS://EDUCATION.GITHUB.COM/GIT-CHEAT-SHEET-EDUCATION.PDF

(FICHE QUI REGROUPE TOUTES LES COMMANDES GITHUB)

- HTTPS://DESKTOP.GITHUB.COM/

(INSTALLER GITHUB)

- HTTPS://CODE.VISUALSTUDIO.COM/

(INSTALLER VS CODE)

- **EXTENSIONS VS CODE UTILES POUR GITHUB:**
  - **GITLENS (GITKRAKEN)**
  - **LIVE SHARE (MICROSOFT)**