



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

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
[Import a repository.](#)

Owner * /

Great repository names are short and memorable. Need inspiration? How about [animated-dollop](#)?

Description (optional)

Public
Anyone on the internet can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

Add a README file
This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

Donner le nom du repositories

Donner une description du projet

Mettre votre repositories en public / privée

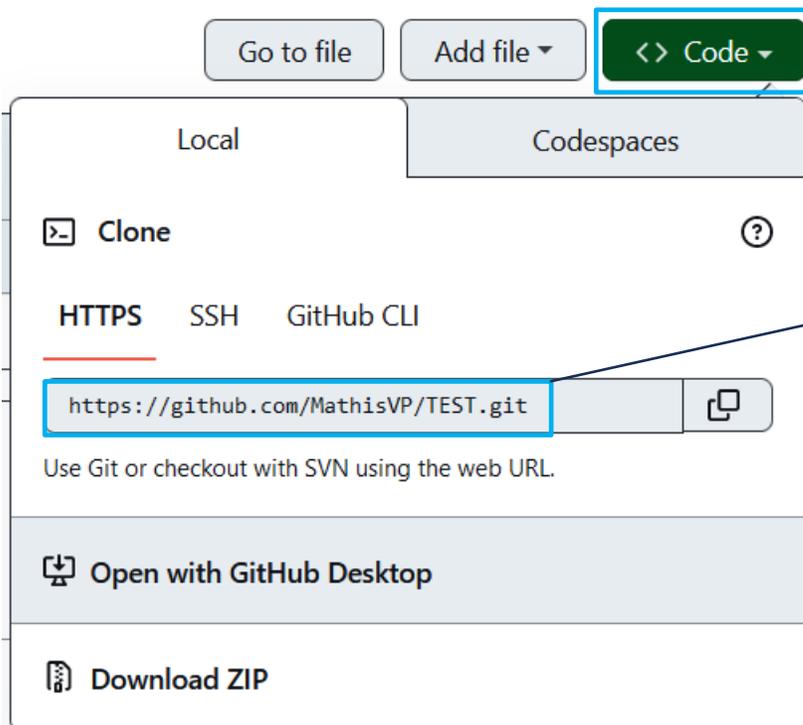
Ajouter un readme, (ça permet d'expliquer à quoi sert votre code par exemple)

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

```
PROBLÈMES  SORTIE  CONSOLE DE DÉBOGAGE  TERMINAL  GITLENS
PS C:\Users\mathi> cd desktop
PS C:\Users\mathi\desktop> git clone https://github.com/MathisVP/TEST.git
Cloning into 'TEST'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
PS C:\Users\mathi\desktop>
```

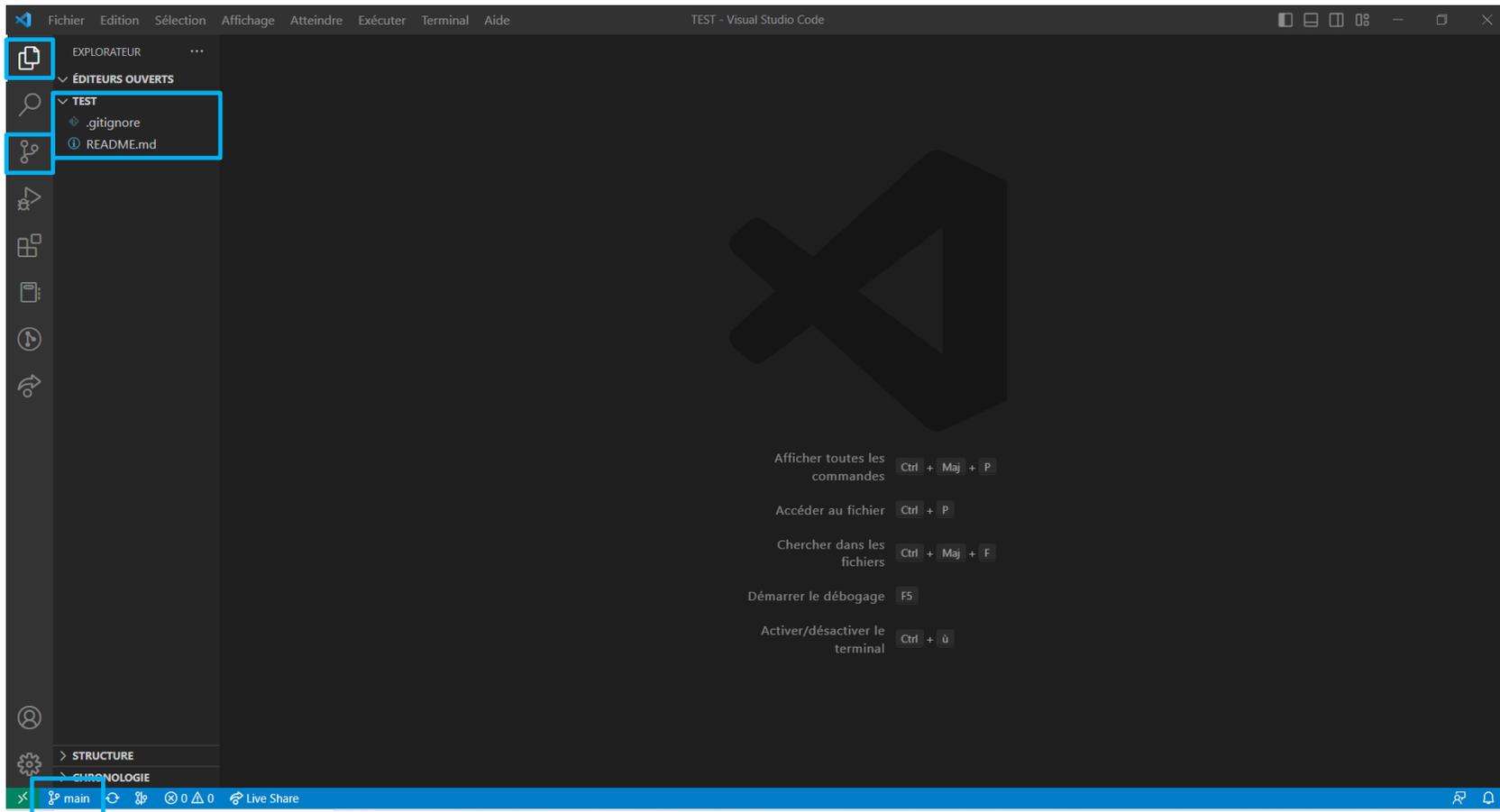
→ Changement de dossier

→ Cloner le dossier depuis le dépôt Git

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

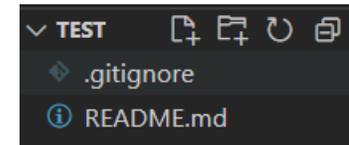
Ouvrir ensuite le dossier dans VS code

OUVRIR LE DOSSIER DANS VS CODE



 Explorateur de fichier

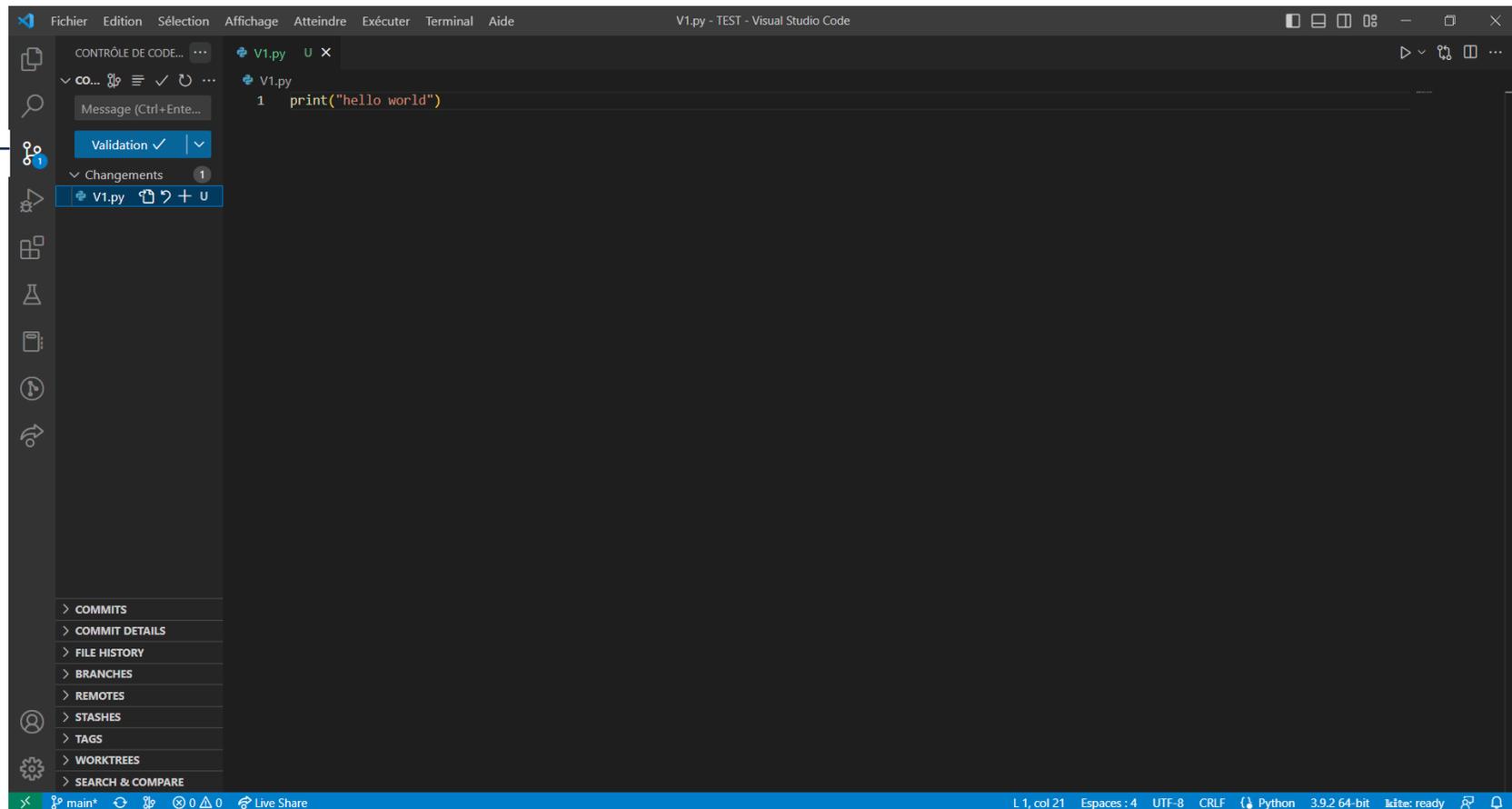
 Contrôle de code source



Fichiers présent
dans le repositories

 main Branche de travail

TRAVAILLER SUR LA BRANCHE « MAIN »



Le I 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 :

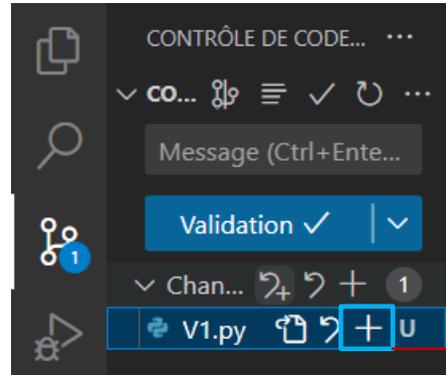
1^{er} méthode avec les commandes :

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

```
git add [file]
```

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

2^{ème} méthode avec VS code :



Le « + » permet d'ajouter le fichier

➤ Commit :

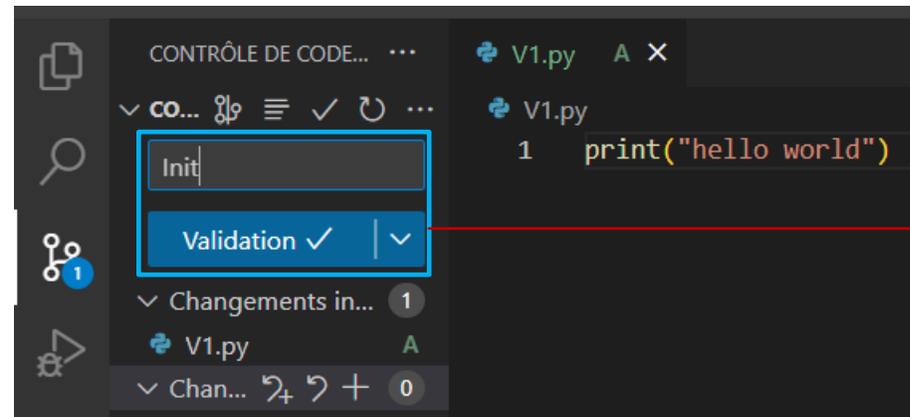
Deux méthodes :

1^{er} méthode avec les commandes :

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

commit your staged content as a new commit snapshot

2^{ème} méthode avec VS code :



Mettre le message et appuyer sur
« validation »

➤ Push :

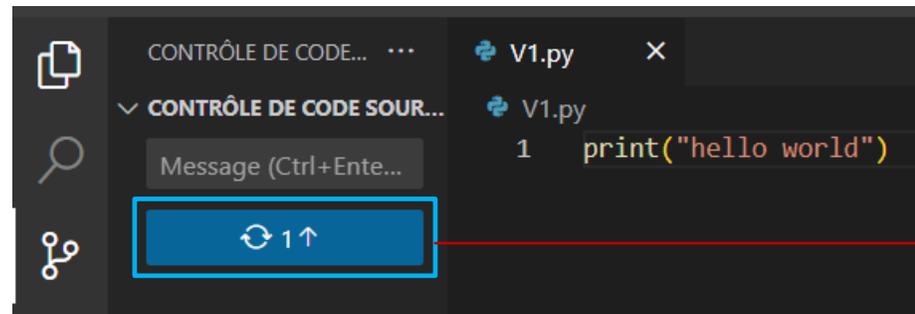
Deux méthodes :

1^{er} méthode avec les commandes :

```
git push [alias] [branch]
```

Transmit local branch commits to the remote repository branch

2^{ème} méthode avec VS code :



Appuyer sur ce bouton pour push

Visual Studio Code



Cette action permet d'extraire et d'envoyer (push) les validations à partir de et vers « origin/main ».

OK

OK, Don't Show Again

Annuler

Appuyer sur « ok »

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

Le fichier a bien été
commit et push sur
le repositories
GitHub

Commentaire mis
lors du commit

```
README.md
TEST
Projet TEST
```



Search or jump to...

[Pull requests](#) [Issues](#) [Codespaces](#) [Marketplace](#) [Explore](#)



[MathisVP / TEST](#) Private

[Unwatch](#) 1 [Fork](#) 0 [Star](#) 0

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Security](#) [Insights](#) [Settings](#)

[main](#) [TEST / V1.py](#) / [Jump to](#)

[Go to file](#) [...](#)

MathisVP Update V1.py Latest commit b88a37c now [History](#)

[1 contributor](#)

3 lines (2 sloc) | 39 Bytes

[Raw](#) [Blame](#) [Edit](#) [Delete](#)

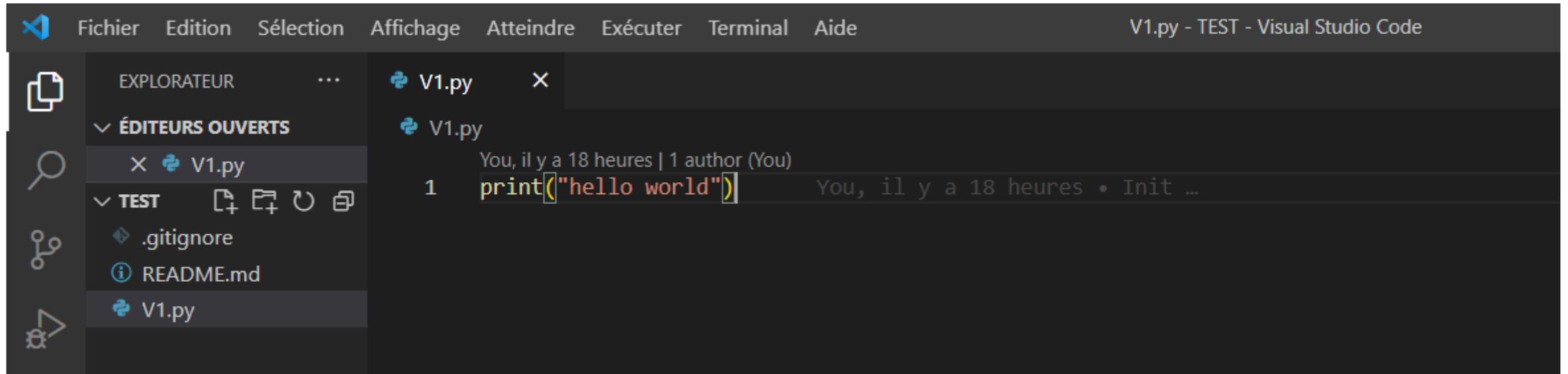
```
1 print("hello world")
2
3 print("bonjour")
```

[Give feedback](#)

© 2023 GitHub, Inc.

[Terms](#) [Privacy](#) [Security](#) [Status](#) [Docs](#) [Contact GitHub](#) [Pricing](#) [API](#) [Training](#) [Blog](#) [About](#)

**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 :

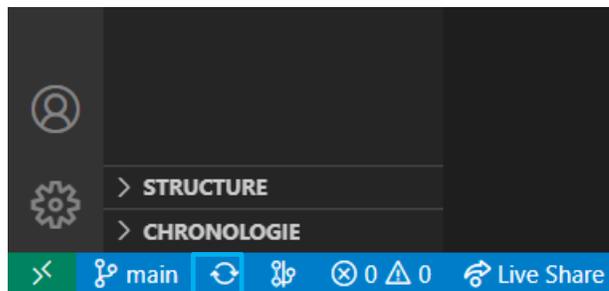
Deux méthodes :

1^{er} méthode avec les commandes :

```
git pull
```

fetch and merge any commits from the tracking remote branch

2^{ème} méthode avec VS code :

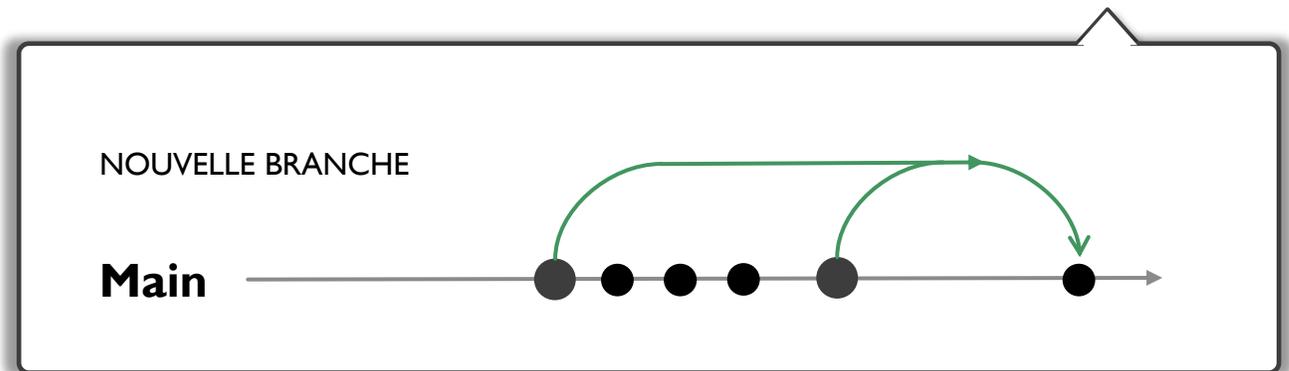
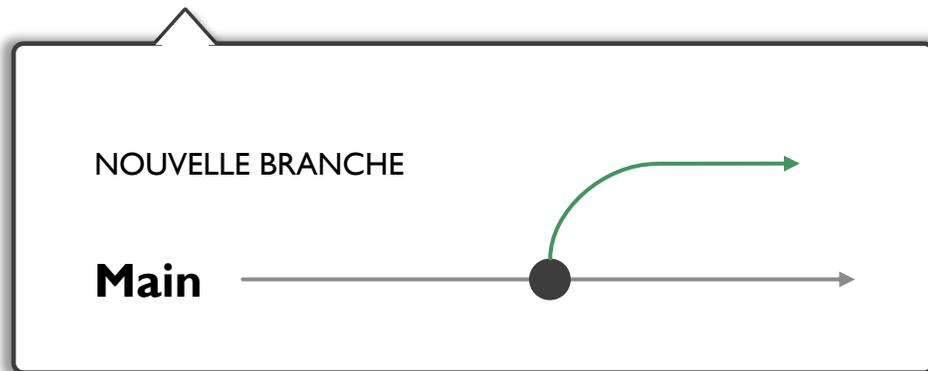


Appuyer sur ce bouton pour pull
(puis sur « ok »)

GESTION DE BRANCHES

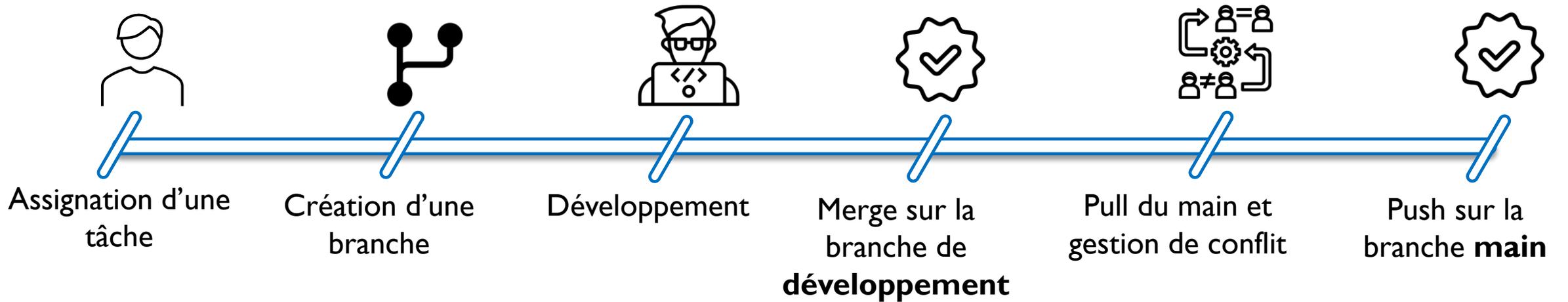
La gestion des branches est primordiale en programmation agile. Les branches servent à ne pas casser le code principal (sur la branche main). Personne ne doit coder sur cette branche. Aucun conflit ne doit être sur le main et elle (la branche) doit toujours être fonctionnelle.

Utilité de la création de branche.



Sur le deuxième schéma, on constate que la personne 1 a créé une branche et qu'il y a eu entre temps la personne 2 qui a 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 modifié 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 :

1^{er} méthode avec les commandes :

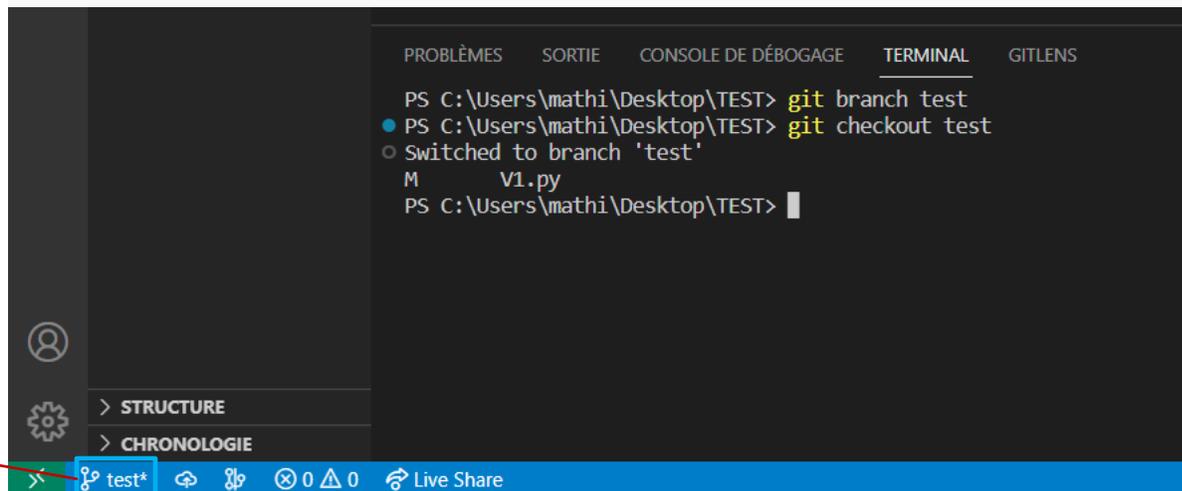
```
git branch [branch-name]
```

create a new branch at the current commit

Naviguer à travers les branches :

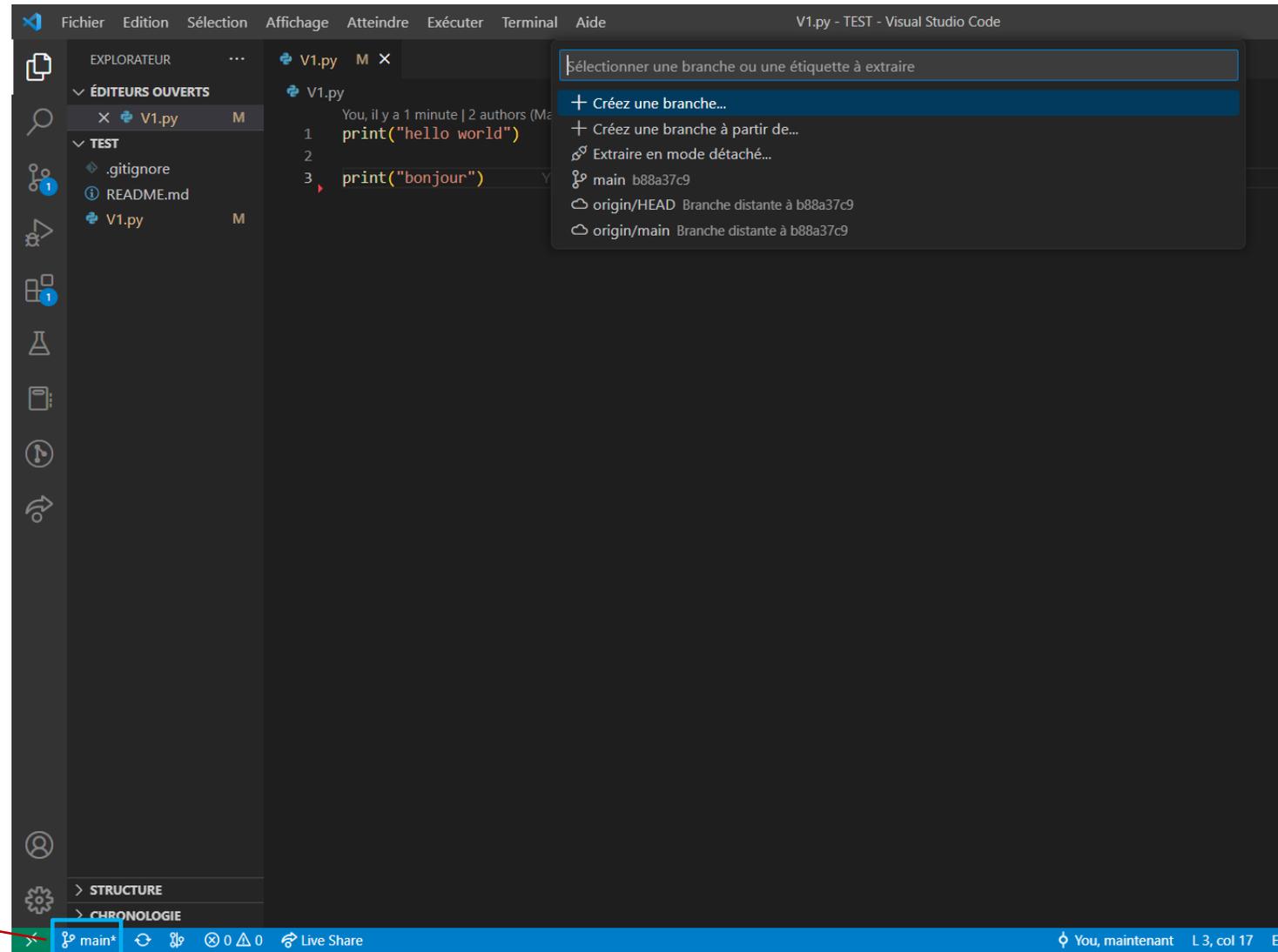
```
git checkout
```

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



J'ai bien changé de branche

2^{ème} 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.

The screenshot shows a GitHub repository interface. At the top, there are buttons for 'Go to file', 'Add file', and 'Code'. Below this is a commit history table with three entries: 'Initial commit' (4 days ago), 'Initial commit' (4 days ago), and 'Update V1.py' (13 minutes ago). The commit hash '724483e' and '5 commits' are also visible. Below the commit history is a file viewer for 'README.md' containing the text 'TEST' and 'Projet TEST'. A 'Switch branches/tags' dialog is open, showing a search bar and a list of branches. The 'main' branch is selected and marked as 'default'. A red arrow points from the text 'La branche test n'est pas mise sur GitHub.' to the 'main' branch in the dialog.

Commit Hash	Time	Commits
724483e	13 minutes ago	5 commits
Initial commit	4 days ago	
Initial commit	4 days ago	
Update V1.py	13 minutes ago	

Switch branches/tags

Find or create a branch...

Branches Tags

✓ main (default)

View all branches

The screenshot shows the Visual Studio Code interface. The top-left sidebar contains icons for Explorer, Search, Run and Debug, and Source Control. The Source Control view shows a commit message: "Message (Ctrl+Ente...)" and a validation status of "Validation ✓". The main editor displays a Python file named "V1.py" with the following code:

```
1 You, il y a 45 secondes | 2 authors (MathisVP and others)
2 print("hello world")
3 print("bonjour")
```

The bottom terminal window shows the output of a Git command:

```
PS C:\Users\mathi\Desktop\TEST> git push origin test
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 12 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (10/10), 2.41 KiB | 2.41 MiB/s, done.
Total 10 (delta 2), reused 3 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), done.
remote:
remote: Create a pull request for 'test' on GitHub by visiting:
remote:   https://github.com/MathisVP/TEST/pull/new/test
remote:
To https://github.com/MathisVP/TEST.git
 * [new branch]   test -> test
PS C:\Users\mathi\Desktop\TEST>
```

The screenshot shows a GitHub repository page for "TEST". The top navigation bar includes "main", "2 branches", and "0 tags". A "Switch branches/tags" dropdown menu is open, showing a search bar "Find or create a branch..." and a list of branches: "main" (marked as "default") and "test". Below the dropdown, the commit history is visible:

Commit Hash	Time	Commits
724483e	16 minutes ago	5 commits
Initial commit	4 days ago	
Initial commit	4 days ago	
Update V1.py	16 minutes ago	

The main content area shows the file "README.md" with the text "TEST" and "Projet TEST".

La branche a bien été ajoutée au repositories GitHub.

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

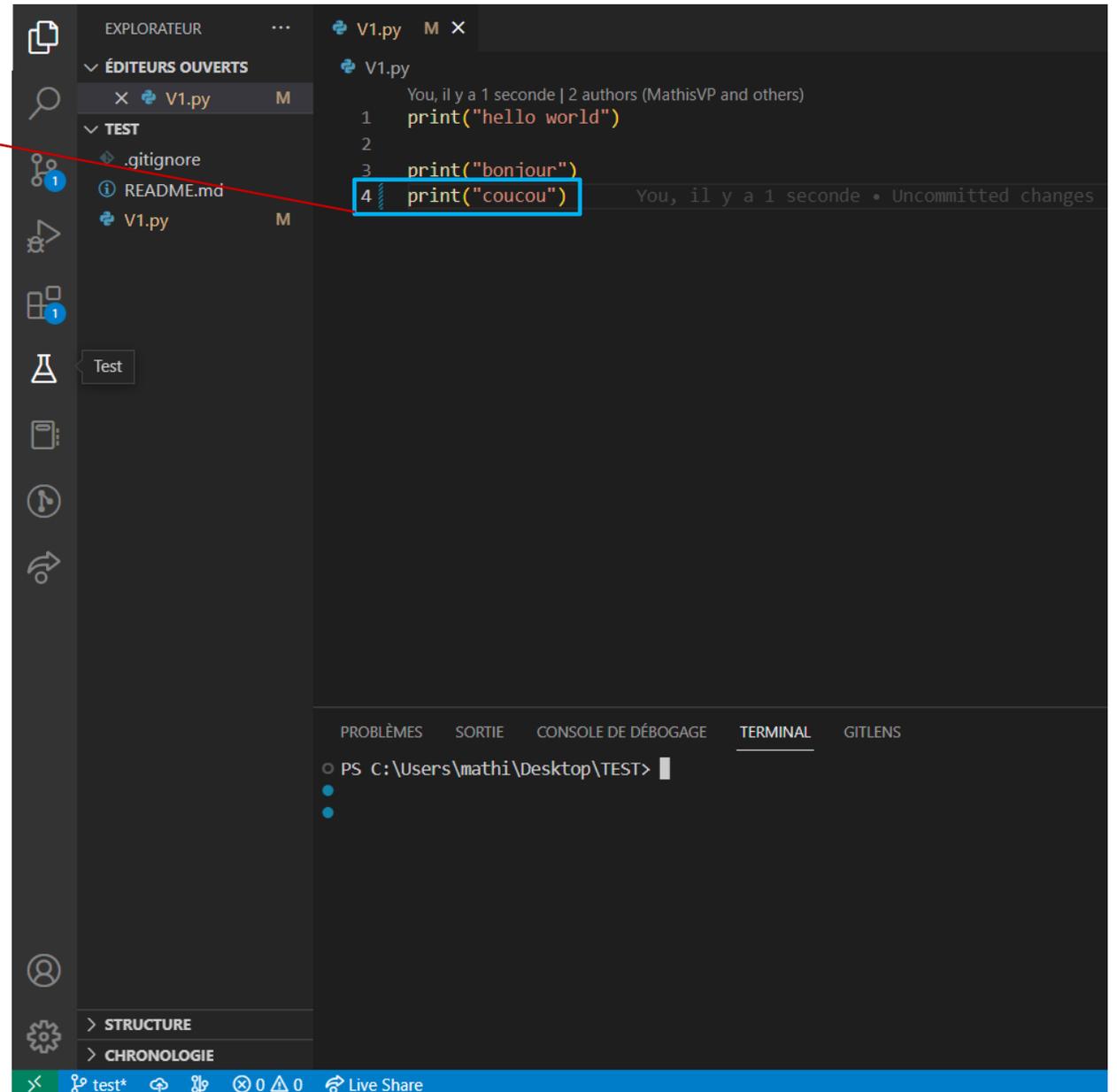
Il faut ajouter, commit et push le code sur notre branche. (C'est ce que fais le code ci-dessous)

```
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

<> Code Issues Pull requests Actions Projects Security Insights Settings

TEST / V1.py in main Cancel changes

<> Edit file Preview changes Spaces 2 No wrap

```
1 print("hello world")
2
3 print("bonjour")
4 print("salut")
```

test TEST / V1.py /<> Jump to Go to file ...

MathisVP nouveau print Latest commit 6e1ed68 11 minutes ago History

1 contributor

4 lines (3 sloc) | 54 Bytes Raw Blame ✎ ⌵ 🗑️

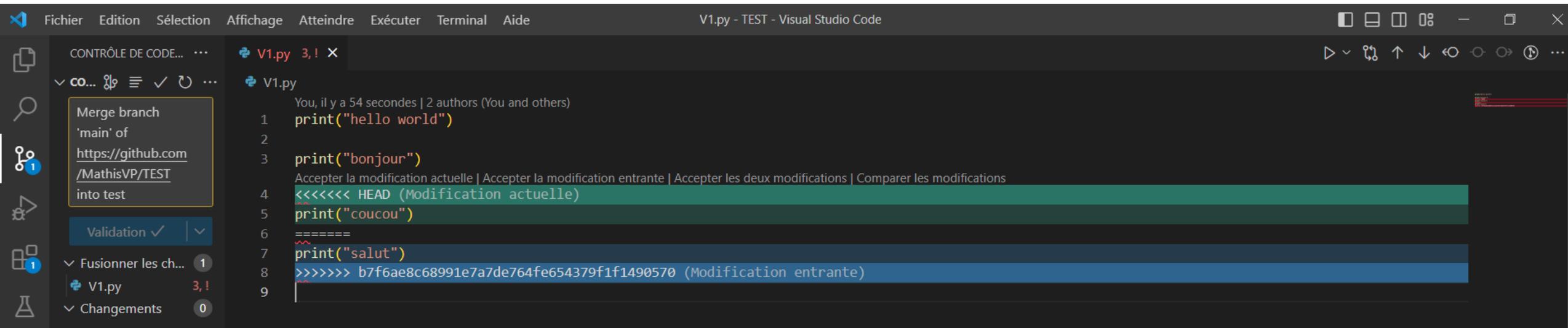
```
1 print("hello world")
2
3 print("bonjour")
4 print("coucou")
```

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

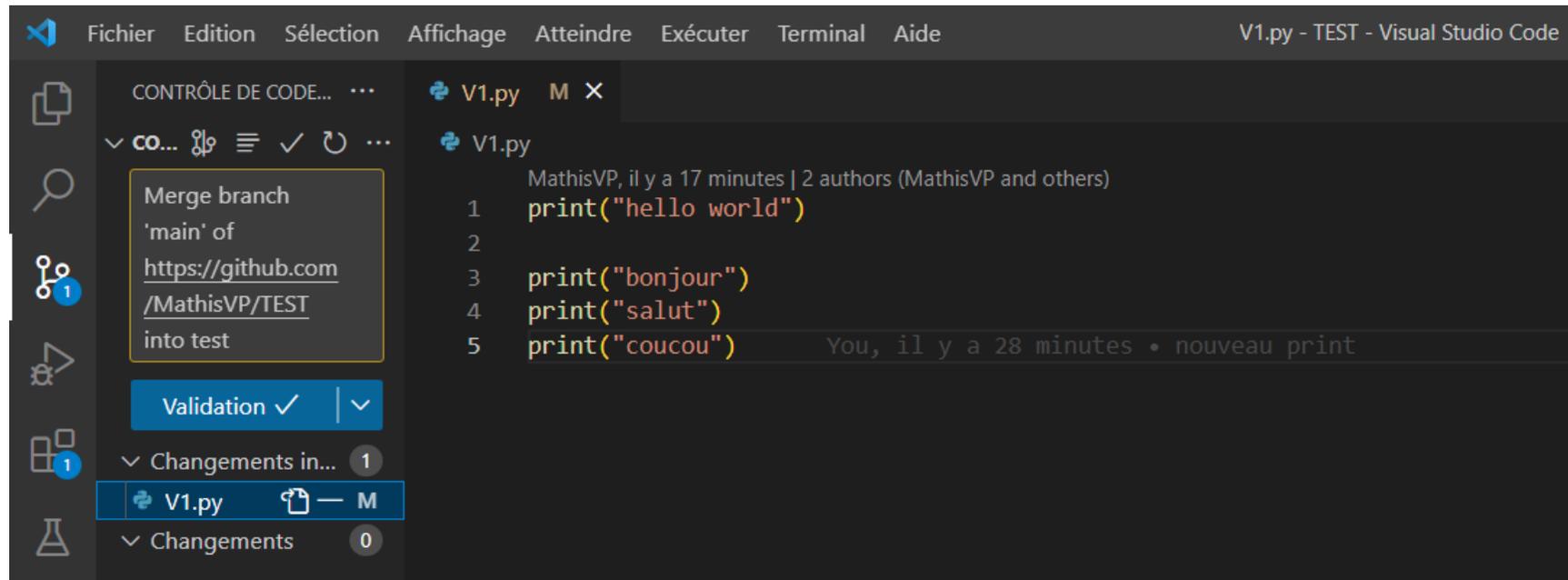
Résoudre dans l'éditeur de fusion

The screenshot displays the Visual Studio Code interface during a merge conflict resolution. The top menu bar includes 'Fichier', 'Edition', 'Sélection', 'Affichage', 'Atteindre', 'Exécuter', 'Terminal', and 'Aide'. The title bar indicates 'Fusion en cours : V1.py - TEST - Visual Studio Code'. The left sidebar shows the 'CONTRÔLE DE CODE SOUR...' panel with a merge menu and a 'Fusionner les ch...' section. The main editor area is split into two panes: 'Entrant' (left) and 'Actuelle' (right). The 'Entrant' pane shows the original code with a conflict on line 4, and the 'Actuelle' pane shows the current code with a conflict on line 4. A blue box highlights the 'Accepter la combinaison' button in the conflict resolution menu. The bottom panel shows the 'Résultat V1.py' with the merged code and a message 'Aucune modification acceptée' on line 4. A '1 conflit restant' indicator is visible in the bottom right corner.

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

Terminer la fusion



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 :

The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar shows a project named 'TEST' with files: '.gitignore', 'README.md', and 'V1.py'. The Editor window shows the content of 'V1.py':

```
1 print("hello world")
2
3 print("bonjour")
4 print("salut")
5 print("coucou")
```

The Terminal window at the bottom shows the following commands and output:

```
PS C:\Users\mathi\Desktop\TEST> git checkout main
Switched to branch 'main'
Your branch is behind 'origin/main' by 3 commits, and can be fast-forwarded.
(use "git pull" to update your local branch)
PS C:\Users\mathi\Desktop\TEST> git pull origin test
From https://github.com/MathisVP/TEST
 * branch          test          -> FETCH_HEAD
Updating b88a37c..cb4c32e
Fast-forward
 1 file changed, 2 insertions(+)
PS C:\Users\mathi\Desktop\TEST>
```

Red arrows point from the text on the left to the terminal commands: 'git checkout main' and 'git pull origin test'.

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.

main ▾ TEST / V1.py / <> Jump to ▾ Go to file ...

MathisVP mise du code de la branche test sur le main Latest commit cb4c32e 15 hours ago History

1 contributor

5 lines (4 sloc) | 69 Bytes Raw Blame

```
1 print("hello world")
2
3 print("bonjour")
4 print("salut")
5 print("coucou")
```

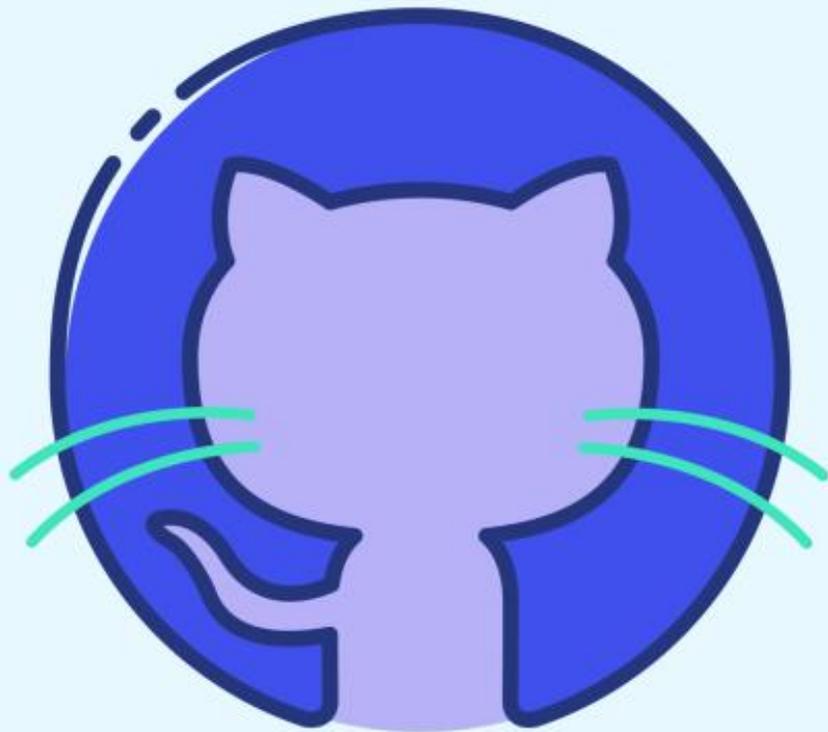
Autres commandes utiles :

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](https://education.github.com/git-cheat-sheet-education.pdf)
(FICHE QUI REGROUPE TOUTES LES COMMANDES GITHUB)
- [HTTPS://DESKTOP.GITHUB.COM/](https://desktop.github.com/)
(INSTALLER GITHUB)
- [HTTPS://CODE.VISUALSTUDIO.COM/](https://code.visualstudio.com/)
(INSTALLER VS CODE)
- EXTENSIONS VS CODE UTILES POUR GITHUB:
 - GITLENS (GITKRAKEN)
 - LIVE SHARE (MICROSOFT)