

## Mini-Blockchain en Python

**Nom complet :** ADDAOUI Bouchra

**Groupe :** 01

## **1. Introduction :**

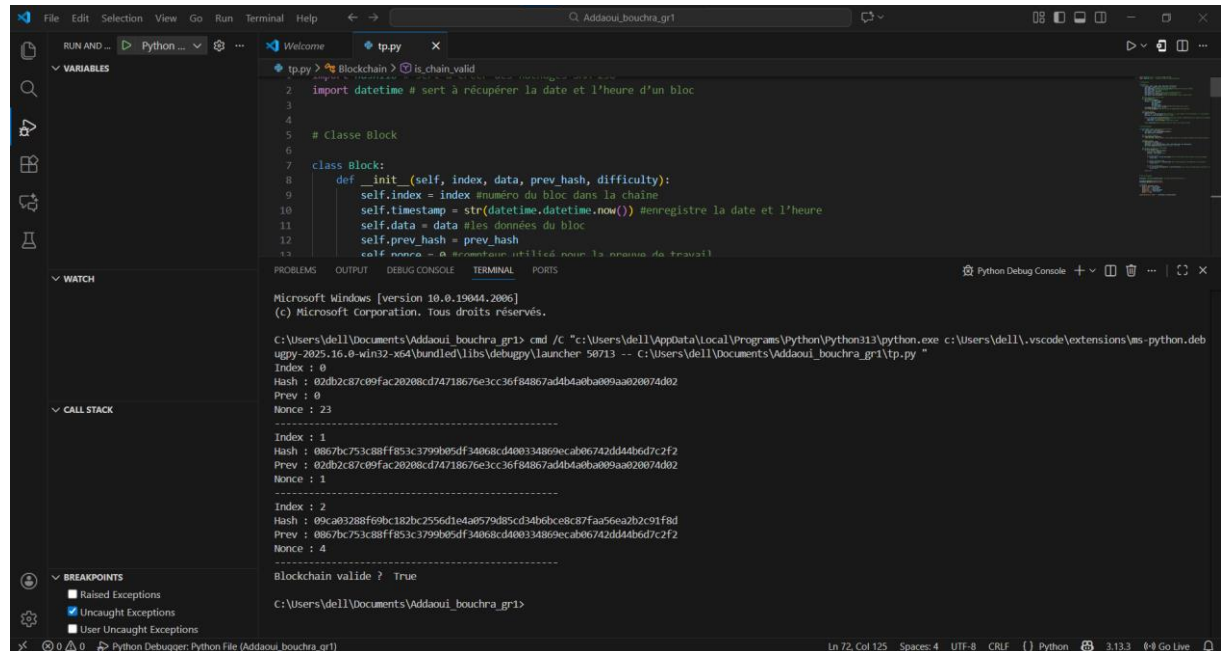
Une blockchain est une chaîne de blocs sécurisée grâce à la cryptographie. Le but de ce projet est de créer une mini-blockchain fonctionnelle en Python afin de comprendre les concepts de hachage, de minage et de validation.

## **2. Méthodologie :**

- Création de la classe Block qui contient les informations essentielles d'un bloc : index, timestamp, données, hash du bloc précédent, nonce et le hash du bloc.
- Implémentation de la fonction de hachage avec l'algorithme SHA-256.
- Mise en place de la preuve de travail (minage) qui consiste à trouver un hash commençant par un certain nombre de zéros.
- Construction de la classe Blockchain qui contient la liste des blocs, crée le bloc de genèse et permet l'ajout de nouveaux blocs.
- Validation de la chaîne à l'aide d'une fonction vérifiant les haches et la cohérence de la preuve de travail.

## **3. Résultats :**

Le programme permet d'ajouter plusieurs blocs à la chaîne, d'afficher leurs contenus détaillés, et de vérifier que la chaîne est valide.



```
tp.py > Blockchain ? is_chain_valid
2 import datetime # sert à récupérer la date et l'heure d'un bloc
3
4
5 # classe Block
6
7 class Block:
8     def __init__(self, index, data, prev_hash, difficulty):
9         self.index = index #numéro du bloc dans la chaîne
10        self.timestamp = str(datetime.datetime.now()) #enregistre la date et l'heure
11        self.data = data #les données du bloc
12        self.prev_hash = prev_hash
13        self.nonce = 0 #nombre utilisé pour la preuve de travail

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python Debug Console

Microsoft Windows [version 10.0.19044.2006]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\del1\Documents\Addaoui_bouchra_gr1> cmd /c "c:\Users\del1\AppData\Local\Programs\Python\Python313\python.exe c:\Users\del1\.vscode\extensions\ms-python.de
ugpy-2025.16.0-win32-x64\handed\libs\debugpy\launcher 58713 -- c:\Users\del1\Documents\Addaoui_bouchra_gr1\tp.py"

Index : 0
Hash : 02db2c87c09fac20208cd74718676e3cc36f84867ad4b4a0ba009aa020074d02
Prev : 0
Nonce : 23
-----
Index : 1
Hash : 0867bc753c88ff853c3799b05df34068cd400334869ecab06742dd44b6d7c2f2
Prev : 02db2c87c09fac20208cd74718676e3cc36f84867ad4b4a0ba009aa020074d02
Nonce : 1
-----
Index : 2
Hash : 09ca03288f69bc182bc2556d1e4a0579d85cd34b6bce8b7faa56ea2b2c91fad
Prev : 0867bc753c88ff853c3799b05df34068cd400334869ecab06742dd44b6d7c2f2
Nonce : 4
-----
Blockchain valide ? True

C:\Users\del1\Documents\Addaoui_bouchra_gr1>
```

- Le code source montre la classe Block, où chaque bloc est défini par ses attributs (index, timestamp, données, hash précédent, nonce, hash).
- Dans le terminal, on voit l'extraction et l'affichage de plusieurs blocs ajoutés à la blockchain :
  - ❖ Chaque bloc affiche son index, son hash, le hash du bloc précédent et la valeur du nonce.
- La dernière ligne « Blockchain valide ? True » indique que la chaîne est correcte et intègre, car tous les blocs respectent la preuve de travail et la structure attendue.

**Discussion :**

Le hachage garantit l'intégrité des données car toute modification d'un bloc change son hash, ce qui invalide la chaîne.

La preuve de travail sécurise la blockchain en rendant le minage coûteux en calcul. La validation empêche toute falsification du contenu puisque le hash de chaque bloc doit correspondre et respecter la difficulté.

### **Conclusion :**

Ce projet a permis de comprendre les concepts fondamentaux d'une blockchain simple.

Des améliorations sont possibles, comme augmenter la difficulté du minage ou implémenter un réseau décentralisé.