**README**

**Components**

**1. Block**

* Class: **createBlock**
* Description: Represents a single block in the blockchain.
  + Properties:
    - **index**: Index of the block in the chain.
    - **timestamp**: Timestamp of when the block was created.
    - **data**: Data stored in the block (e.g., transactions).
    - **previous\_hash**: Hash of the previous block.
    - **nonce**: Nonce used in PoW.
    - **hash**: Hash of the block's data.

**2. Blockchain**

* Class: **Blockchain**
* Description: Manages the chain of blocks.
  + Methods:
    - **create\_genesis\_block()**: Creates the initial block (genesis block).
    - **get\_latest\_block()**: Retrieves the latest block in the chain.
    - **add\_block(new\_block)**: Adds a new block to the chain.

**3. Miner**

* Class: **Miner**
* Description: Simulates mining by finding valid blocks.
  + Methods:
    - **mineBlock(data)**: Mines a new block with the given data.
    - **proof\_of\_work(block)**: Implements the PoW algorithm to find a valid hash.

**4. Transaction**

* Classes: **Course**, **Student**, **Educator**
* Description: Represents transactions between students and educators.
  + **Course**: Details about a specific course.
  + **Student**: Information about a student and their enrolled courses.
  + **Educator**: Information about an educator and the courses they teach.
  + Methods:
    - **add\_course(...)**: Adds a course to a student's or educator's list.

**5. Verification**

* Class: **verifyTransaction**
* Description: Verifies transactions using HMAC.
  + Methods:
    - **generate\_bit()**: Generates a random bit (0 or 1).
    - **create\_response(challenge, bit)**: Creates HMAC response using challenge and secret key.
    - **get\_hmac(data, user, transaction)**: Computes HMAC for a given data and transaction.
    - **verify\_transaction(transaction, student, educator)**: Verifies a transaction's authenticity using HMAC.

**Usage**

* Creates instances of students, educators, and transactions.
* Verifies transactions and mines blocks when a certain number of valid transactions are accumulated.

**How to Use**

1. Instantiate blockchain: **blockchain = Blockchain()**
2. Create a miner: **miner = Miner(blockchain)**
3. Define students and educators.
4. Add courses for students and educators.
5. Conduct transactions and verify them.

**Important Concepts**

* **Proof of Work (PoW)**: Used by miners to find valid blocks by solving cryptographic puzzles.
* **Transactions**: Represent interactions between students and educators, stored in blocks.
* **HMAC Verification**: Ensures the authenticity of transactions using HMAC.

**Example**

* Simulates a scenario where students enroll in courses taught by educators, conduct transactions, and validate them using Proof of Work and HMAC.

**Conclusion**

This code provides a basic implementation of a blockchain system with Proof of Work, transactions, and verification mechanisms. It can be extended and customized for various applications, such as academic record management, financial transactions, etc.

Group members

* Nischith P 2021A7PS3220H
* Aayush Chitkara 2021A7PS2852H
* S Ananth 2021A7PS2809H