1. Understanding of how blockchain technology functions

Blockchain technology is a decentralized and distributed ledger system that underlies cryptocurrencies like Bitcoin, but its applications extend far beyond digital currencies. It is designed to provide a secure and transparent way to record transactions and data across a network of computers. Below, I'll explain the principles and working mechanisms of blockchain technology:

1. Distributed Ledger: At its core, a blockchain is a digital ledger that is distributed across a network of computers (nodes). This distributed nature ensures that no single entity has complete control over the data, making it resistant to tampering and censorship.

2. Blocks: Transactions and data are grouped together into blocks. Each block contains a list of transactions and a reference to the previous block, forming a chain of blocks, hence the term "blockchain."

3. Cryptographic Hashing: Blocks are linked together using cryptographic hashes. A hash is a fixed-length string of characters that is generated from the data within a block. Changing any data within a block would alter its hash, which would, in turn, require changing the hash of all subsequent blocks, making tampering highly impractical.

4. Consensus Mechanism: To add a new block to the blockchain, a consensus mechanism is used to validate and agree upon the transactions. The most well-known consensus mechanism is Proof of Work (PoW), which requires nodes (miners) to solve complex mathematical puzzles to validate transactions. More recently, Proof of Stake (PoS) and other mechanisms have emerged, which require validators to hold a certain amount of cryptocurrency and put it at stake to confirm transactions.

5. Decentralization: Blockchains are decentralized, meaning there is no central authority or intermediary controlling the network. Instead, consensus among nodes is achieved through the chosen consensus mechanism. This decentralization enhances security and reduces the risk of a single point of failure.

6. Transparency: The blockchain ledger is transparent and public. Anyone can view the entire transaction history and verify the integrity of the data. However, the identities of participants in transactions are typically pseudonymous, represented by cryptographic addresses.

7. Immutability: Once a block is added to the blockchain, it is extremely difficult to alter or delete the data within it due to the cryptographic hashing and consensus mechanisms. This immutability makes blockchains trustworthy for recording important information.

8. Smart Contracts: Some blockchains, like Ethereum, support smart contracts. Smart contracts are self-executing contracts with the terms of the agreement directly written into code. They automatically execute when predefined conditions are met, eliminating the need for intermediaries in many types of transactions and agreements.

9. Security: Blockchain networks are considered highly secure because of their decentralized and cryptographic nature. Hacking a blockchain would require tremendous computational power and consensus of the majority of nodes, making it economically infeasible.

10. Use Cases: Blockchain technology has a wide range of applications, including cryptocurrency, supply chain management, voting systems, healthcare data management, identity verification, and more. It is particularly valuable in situations where trust, transparency, and security are paramount.

In summary, blockchain technology functions by creating a decentralized, transparent, and secure ledger of transactions and data through the use of cryptographic techniques, consensus mechanisms, and distributed computing. Its potential applications are vast and continue to evolve as the technology matures.

2. Comprehensive review of the code

3. **Blockchain Class:**
    a. So, the heart of this code is the **Blockchain** class. It's responsible for managing the chain of blocks, transactions, and some basic consensus logic.
    b. We kick things off by initializing an empty chain, an empty list for pending transactions, and a set of nodes representing network participants. The first block, the "genesis block," is created right away with predefined values.
    c. To add transactions to the blockchain, we have the **new_transaction** method. It adds transactions to the **current_transactions** list until we're ready to mine a new block.
    d. When we're ready to mine, the **new_block** method comes into play. It takes a proof of work and an optional previous hash. This method finalizes the current transactions, adds the block to the chain, and returns it.
    e. We calculate the hash of a block using the **hash** method, which utilizes SHA-256.
    f. The **proof_of_work** function is where we perform a simple proof-of-work algorithm to find a valid nonce (proof) that satisfies certain criteria.
    g. **valid_proof** checks if a given proof meets the required conditions, typically having a specific prefix of zeros.
    h. The **resolve_conflicts** method is meant to resolve conflicts in the blockchain among different nodes, but it's currently incomplete.
    i. **valid_chain** checks if a given chain is valid, considering hash linkage and proof of work.
    j. Lastly, **consensus** is also unfinished and might be a placeholder for a more advanced consensus algorithm like Proof of Work.
4. **MerkleTree Class:**
    a. The **MerkleTree** class represents a Merkle tree, which is commonly used for transaction validation.

b. You can add transactions to the Merkle tree using the **add_tx** method, and it recalculates the Merkle root hash whenever a new transaction is added.

c. The **update_root_hash** method performs the calculation of the Merkle root hash based on the added transactions.

5. **Driver Code:**

a. In the driver code, I demonstrate how to use the **Blockchain** and **MerkleTree** classes.

b. Two sample transactions are added to the blockchain, and their hashes are added to the Merkle tree.

c. We then mine a new block and obtain the Merkle root hash.

d. Finally, the newly mined block is printed.