EXPERIMENT NO: 02

Aim :

Create a Blockchain using Python

Theory :

Blockchain is a distributed and immutable ledger that records data in a sequence of linked blocks. Each block contains a timestamp, proof (from Proof-of-Work), and the hash of the previous block, ensuring integrity and security.

In this program:

1. **Block Creation** – A block is generated using `create_block()` with a proof and previous hash.
2. **Proof-of-Work (PoW)** – Implemented to mine a block by solving a cryptographic puzzle (finding a hash with leading zeros).
3. **Hashing** – SHA-256 algorithm ensures data immutability.
4. **Validation** – `is_chain_valid()` checks that every block correctly references the previous hash and satisfies PoW conditions.
5. **Flask Web API** – Routes `/mine_block`, `/get_chain`, and `/is_valid` allow mining, fetching the chain, and verifying blockchain integrity through a browser or API client.

This program runs on a local server and simulates mining a blockchain without a network of nodes, making it an ideal introductory model to understand blockchain basics.

1. What is a Blockchain?

A Blockchain is a distributed, decentralized, and immutable digital ledger used to record data in the form of blocks. Each block contains a set of records, a timestamp, a cryptographic hash of the previous block, and a proof generated using a consensus algorithm.

The blocks are linked together using cryptographic hashes, forming a secure chain. This structure ensures that once data is recorded, it cannot be modified without altering all subsequent blocks, making the system highly secure and tamper-proof.

Key Features of Blockchain:

- Decentralization: No central authority controls the data.
- Immutability: Once recorded, data cannot be altered.
- Transparency: All transactions are visible and verifiable.
- Security: Cryptographic hashing ensures data integrity.

2. Process of Mining

Mining is the process of adding a new block to the blockchain by solving a complex mathematical puzzle. This puzzle is part of a Proof-of-Work (PoW) consensus algorithm, which ensures that blocks are added securely and prevents malicious users from modifying the blockchain.

Mining Steps:

1. The miner retrieves the previous block from the blockchain.
2. The miner extracts the previous block’s proof value.
3. A new proof is calculated by repeatedly hashing a mathematical expression until the hash satisfies a difficulty condition (for example, starting with four leading zeros).
4. Once a valid proof is found, a new block is created using:
    - Index
    - Timestamp
    - Proof value
    - Previous block’s hash
5. The new block is added to the blockchain.

Purpose of Mining:

- Ensures data security.
- Prevents tampering.
- Maintains the integrity of the blockchain.
- Controls the rate of block generation.

3. How to Check the Validity of Blocks in a Blockchain

Blockchain validation is the process of verifying the integrity and correctness of all blocks in the chain. It ensures that no block has been altered or tampered with.

Validation Steps:

1. **Verify Hash Link:**
   Each block contains the hash of the previous block. The system recalculates the hash of the previous block and compares it with the stored value. If they differ, the blockchain is invalid.

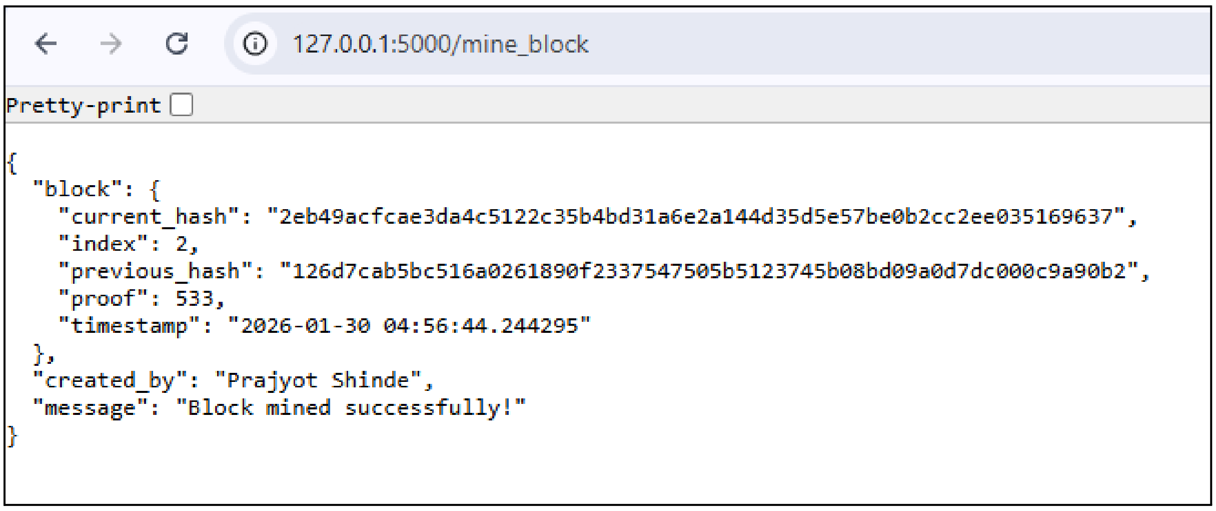
2. **Verify Proof-of-Work:**
   The proof value of each block is verified to ensure that it satisfies the mining difficulty condition (e.g., hash starts with four zeros).

3. **Verify Current Hash:**
   The current block's hash is recalculated and compared with the stored hash to detect any data modification.

Code

```
import datetime
import hashlib
import json
from flask import Flask, jsonify


# ----------------------
# Blockchain Class
# ----------------------

class Blockchain:
    def __init__(self):
        self.chain = []
        self.create_block(proof=1, previous_hash='0')

    def create_block(self, proof, previous_hash):
        block = {
            'index': len(self.chain) + 1,
            'timestamp': str(datetime.datetime.now()),
            'proof': proof,
            'previous_hash': previous_hash,
        }

        # Calculate current block hash
        block['current_hash'] = self.hash(block)

        self.chain.append(block)
        return block

    def get_previous_block(self):
        return self.chain[-1]

    def proof_of_work(self, previous_proof):
        new_proof = 1
        check_proof = False

        while not check_proof:
            hash_operation = hashlib.sha256(
                str(new_proof**2 - previous_proof**2).encode()
            ).hexdigest()

            if hash_operation[:4] == '0000':
                check_proof = True
            else:
                new_proof += 1

        return new_proof

    def hash(self, block):
        # Remove current_hash field before hashing
        block_copy = block.copy()
        block_copy.pop('current_hash', None)

        encoded_block = json.dumps(block_copy, sort_keys=True).encode()
        return hashlib.sha256(encoded_block).hexdigest()
    def is_chain_valid(self, chain):
        previous_block = chain[0]
        block_index = 1

        while block_index < len(chain):
            block = chain[block_index]

            # Validate previous hash link
            if block['previous_hash'] != previous_block['current_hash']:
                return False

            # Validate proof of work
            previous_proof = previous_block['proof']
            proof = block['proof']

            hash_operation = hashlib.sha256(
                str(proof**2 - previous_proof**2).encode()
            ).hexdigest()

            if hash_operation[:4] != '0000':
                return False

            # Validate current hash integrity
            if block['current_hash'] != self.hash(block):
                return False

            previous_block = block
            block_index += 1

        return True


# ----------------------
# Flask Web App
# ----------------------

app = Flask(__name__)
blockchain = Blockchain()


@app.route('/mine_block', methods=['GET'])
def mine_block():

    previous_block = blockchain.get_previous_block()
    proof = blockchain.proof_of_work(previous_block['proof'])

    block = blockchain.create_block(
        proof, previous_block['current_hash']
    )

    return jsonify({
        'message': 'Block mined successfully!',
        'created_by': 'Prajyot Shinde',
        'block': block
    }), 200


@app.route('/get_chain', methods=['GET'])
def get_chain():
    return jsonify({
        'chain': blockchain.chain,
        'length': len(blockchain.chain),
        'created_by': 'Prajyot Shinde',

    }), 200


@app.route('/is_valid', methods=['GET'])
def is_valid():
    return jsonify({
        'is_valid': blockchain.is_chain_valid(blockchain.chain),
        'created_by': 'Prajyot Shinde',
    }), 200

@app.route('/tamper', methods=['GET'])
def tamper():
    blockchain.chain[1]['proof'] = 9999
    return jsonify({'message': 'Block 2 has been tampered!' , 'created_by': 'Prajyot Shinde',}),
200



if __name__ == '__main__':
    app.run(debug=True)
```

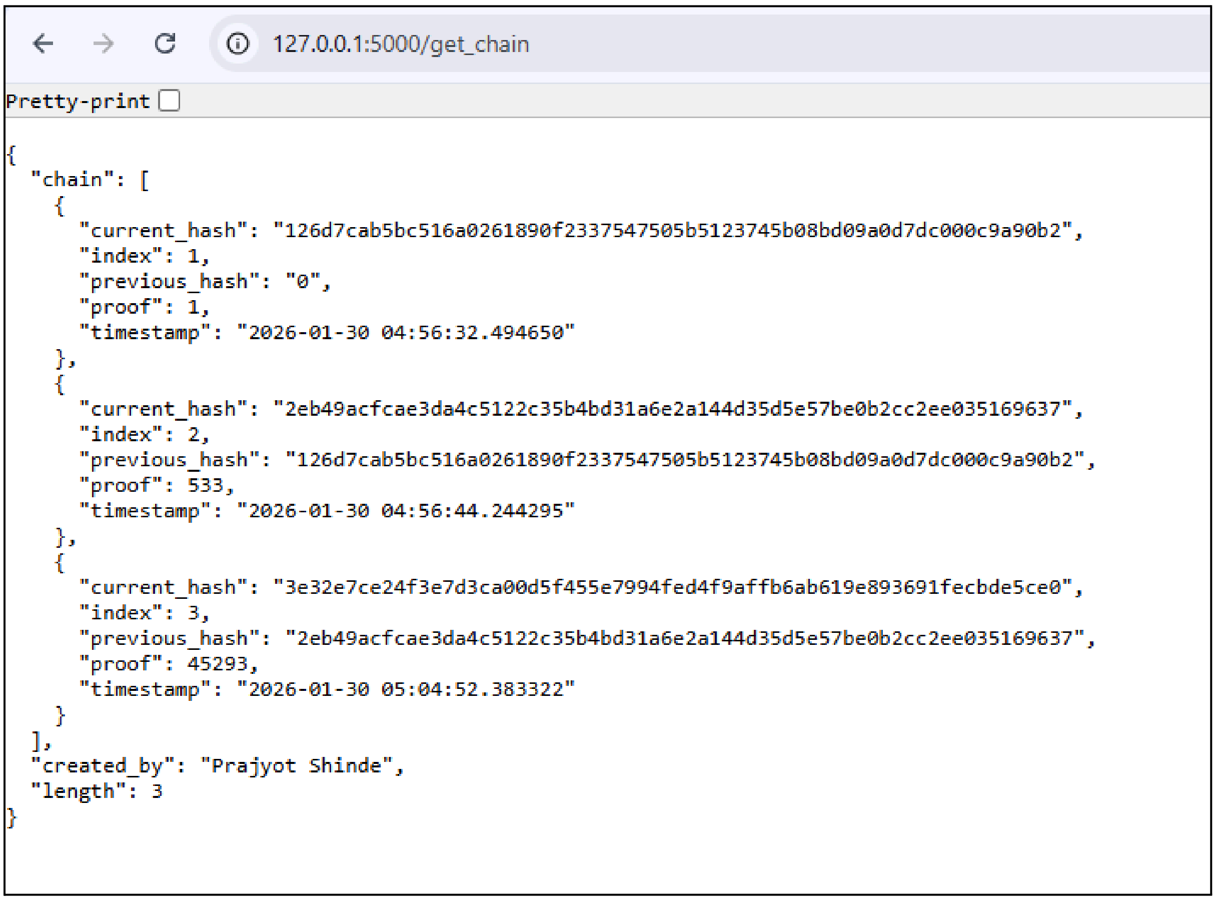
Output



127.0.0.1:5000/mine_block

Pretty-print

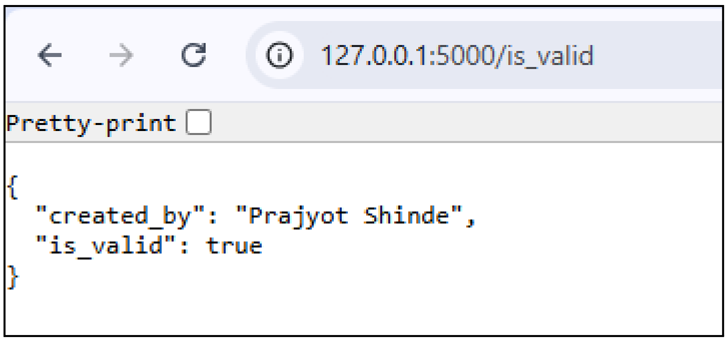
```
{
  "block": {
    "current_hash": "2eb49acfcae3da4c5122c35b4bd31a6e2a144d35d5e57be0b2cc2ee035169637",
    "index": 2,
    "previous_hash": "126d7cab5bc516a0261890f2337547505b5123745b08bd09a0d7dc000c9a90b2",
    "proof": 533,
    "timestamp": "2026-01-30 04:56:44.244295"
  },
  "created_by": "Prajyot Shinde",
  "message": "Block mined successfully!"
}
```



127.0.0.1:5000/get_chain

Pretty-print

```
{
  "chain": [
    {
      "current_hash": "126d7cab5bc516a0261890f2337547505b5123745b08bd09a0d7dc000c9a90b2",
      "index": 1,
      "previous_hash": "0",
      "proof": 1,
      "timestamp": "2026-01-30 04:56:32.494650"
    },
    {
      "current_hash": "2eb49acfcae3da4c5122c35b4bd31a6e2a144d35d5e57be0b2cc2ee035169637",
      "index": 2,
      "previous_hash": "126d7cab5bc516a0261890f2337547505b5123745b08bd09a0d7dc000c9a90b2",
      "proof": 533,
      "timestamp": "2026-01-30 04:56:44.244295"
    },
    {
      "current_hash": "3e32e7ce24f3e7d3ca00d5f455e7994fed4f9affb6ab619e893691fecbde5ce0",
      "index": 3,
      "previous_hash": "2eb49acfcae3da4c5122c35b4bd31a6e2a144d35d5e57be0b2cc2ee035169637",
      "proof": 45293,
      "timestamp": "2026-01-30 05:04:52.383322"
    }
  ],
  "created_by": "Prajyot Shinde",
  "length": 3
}
```


127.0.0.1:5000/is_valid

Pretty-print

```
{
  "created_by": "Prajyot Shinde",
  "is_valid": true
}
```

Conclusion

In this experiment, a simple blockchain system was successfully designed and implemented using Python and Flask. The blockchain was constructed using a sequence of cryptographically linked blocks, where each block contains a timestamp, proof-of-work value, and hash of the previous block.

The mining process was implemented using the Proof-of-Work (PoW) algorithm, which ensures security by requiring computational effort to add new blocks. Cryptographic hashing using the SHA-256 algorithm provided data integrity and immutability. Blockchain validation was also implemented to verify block linkage, proof-of-work correctness, and detect any tampering attempts.

Through this experiment, the fundamental concepts of blockchain technology such as decentralization, immutability, security, and transparency were clearly understood. This implementation provides a strong foundation for learning advanced blockchain concepts such as transaction management, wallet systems, distributed networks, and smart contracts.

Thus, the experiment successfully demonstrates the working principles of blockchain technology in a simple and effective manner.