Experiment 3

Aim: Create a Blockchain using Python

Theory:

Blockchain is a distributed and decentralized digital ledger technology used to record transactions in a secure, transparent, and tamper-proof manner. Instead of storing data in a single central server, blockchain maintains a continuously growing list of records called blocks, which are linked together using cryptographic hashes.

Each block contains:

- Transaction or data details
- A timestamp
- A cryptographic hash of the previous block
- A proof value (nonce) used for mining

This linking of blocks forms a chain, known as the blockchain. Once data is stored in a block and added to the chain, it becomes extremely difficult to modify because changing one block would require recalculating the hash of all subsequent blocks, making the system secure and tamper-resistant.

Key Concepts

1. **Block Structure:** A block is the fundamental unit of a blockchain. Each block contains:
   - **Index** – Position of the block in the chain
   - **Timestamp** – Time when the block was created
   - **Proof (Nonce)** – Number used to satisfy Proof-of-Work
   - **Previous Hash** – Hash of the previous block

2. **Hashing (SHA-256):** Hashing is a cryptographic technique that converts input data into a fixed-length string.
   Properties of hashing:
   - One-way function (cannot be reversed)
   - Small change in data produces a completely different hash
   - Ensures data integrity and immutability
   - If any block data is modified, its hash changes, breaking the blockchain chain and making tampering detectable.

3. **Proof-of-Work (PoW) :**Proof-of-Work is a consensus mechanism used to validate and add new blocks to the blockchain.Miners must solve a computational puzzle to find a valid proof (nonce) such that the hash begins with a specific number of leading zeros.This ensures: Security, Resistance to tampering, Prevention of fake block creation.

4. **Chain Validation :** Blockchain validity is verified by:
   - Checking whether each block's previous_hash matches the hash of the previous block
   - Re-computing Proof-of-Work for each block
     This ensures:
     1. The chain has not been altered
     2. All blocks follow PoW rules
     3. Blockchain integrity is maintained

Working of the System

1. Genesis block is created
2. User requests mining
3. Proof-of-Work is computed
4. Block is created with previous hash
5. Block is added to chain
6. Chain validity is verified

Implementation:

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.

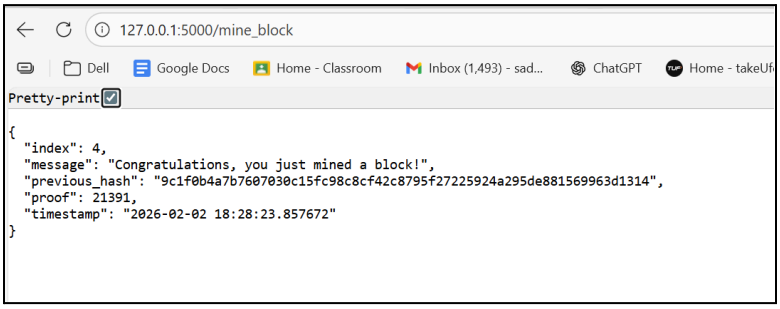
Code:

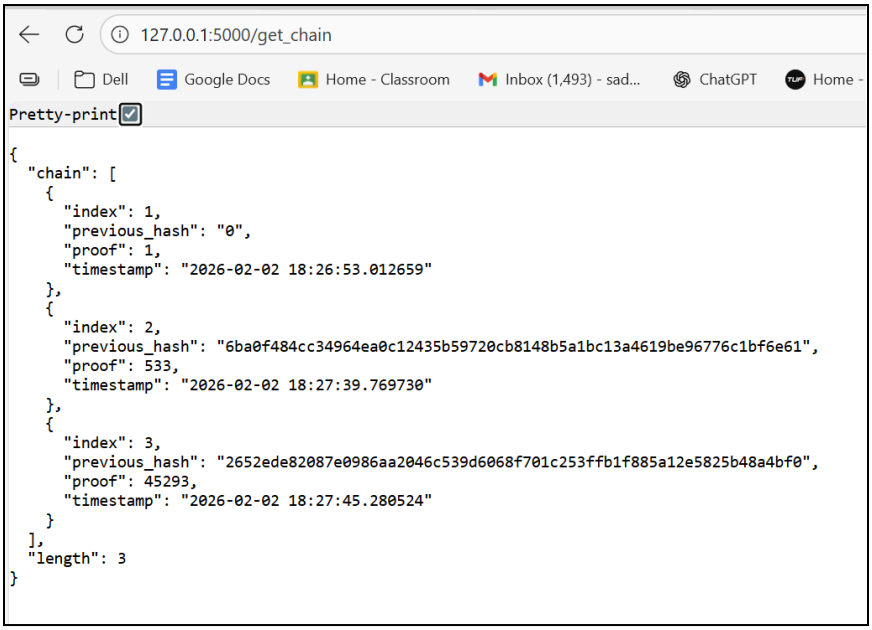
```
import datetime
import hashlib
import json
from flask import Flask, jsonify

# Part 1 - Building a Blockchain
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}
        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 check_proof is False:
            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
```

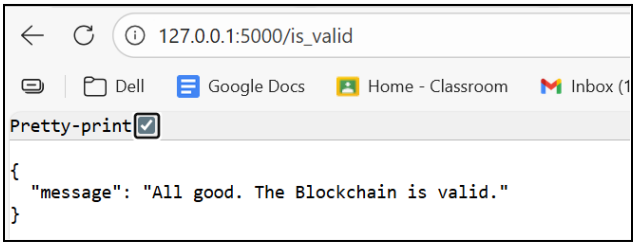
```
        return new_proof

    def hash(self, block):
        encoded_block = json.dumps(block, 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]
            if block['previous_hash'] != self.hash(previous_block):
                return False
            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
            previous_block = block
            block_index += 1
        return True

# Part 2 - Mining our Blockchain
# Creating a Web App
app = Flask(__name__)
# Creating a Blockchain
blockchain = Blockchain()
# Mining a new block
@app.route('/mine_block', methods = ['GET'])
def mine_block():
    previous_block = blockchain.get_previous_block()
    previous_proof = previous_block['proof']
    proof = blockchain.proof_of_work(previous_proof)
    previous_hash = blockchain.hash(previous_block)
    block = blockchain.create_block(proof, previous_hash)
    response = {'message': 'Congratulations, you just mined a block!',
                'index': block['index'],
                'timestamp': block['timestamp'],
                'proof': block['proof'],
                'previous_hash': block['previous_hash']}
    return jsonify(response), 200

# Getting the full Blockchain
@app.route('/get_chain', methods = ['GET'])
def get_chain():
    response = {'chain': blockchain.chain,
                'length': len(blockchain.chain)}
    return jsonify(response), 200

# Checking if the Blockchain is valid
@app.route('/is_valid', methods = ['GET'])
def is_valid():
    is_valid = blockchain.is_chain_valid(blockchain.chain)
    if is_valid:
        response = {'message': 'All good. The Blockchain is valid.'}
    else:
```

```
    response = {'message': 'Houston, we have a problem. The Blockchain is not valid.'}
return jsonify(response), 200

app.run(host = '0.0.0.0', port = 5000)
```

Output:







Conclusion:

We successfully created a basic blockchain using Python and Flask that supports block mining, chain retrieval, and validity checks. Proof-of-Work ensures security by making block creation computationally intensive, while cryptographic hashing guarantees immutability. The implementation demonstrates core blockchain principles decentralization, immutability, and consensus in a simplified environment, providing a strong foundation for building more advanced blockchain systems.