NAME : Palak Chanchlani
ROLL NO: 05
CLASS : D20A
BATCH : C

BLOCKCHAIN EXP 2

Aim: Create a Blockchain using Python

1. What is blockchain?
Blockchain is a digital record-keeping system where information is stored in connected blocks. Each block is linked to the previous one using cryptographic hashes, forming a secure chain.

- It works like a distributed ledger, shared across many computers (nodes).
- Once data is added, it becomes permanent and cannot be changed without breaking the chain.
- Every participant in the network has a copy of the blockchain, ensuring transparency.
- Consensus mechanisms (like Proof of Work) make sure only valid data is added.
- Because of decentralization, no single authority controls the blockchain, making it trustworthy and tamper-resistant.

2. What is a block?
A block is the basic unit of a blockchain. It acts like a container that holds verified data.

Main parts of a block:

- **Index:** Position of the block in the chain.
- **Timestamp:** Exact time when the block was created.
- **Proof:** Result of solving the cryptographic puzzle.
- **Hash:** Unique digital fingerprint of the block.
- **Previous Hash:** Connects the block to the one before it, ensuring continuity.

In simple terms, a block is like a “page” in a digital notebook, where each page is permanently linked to the previous one.

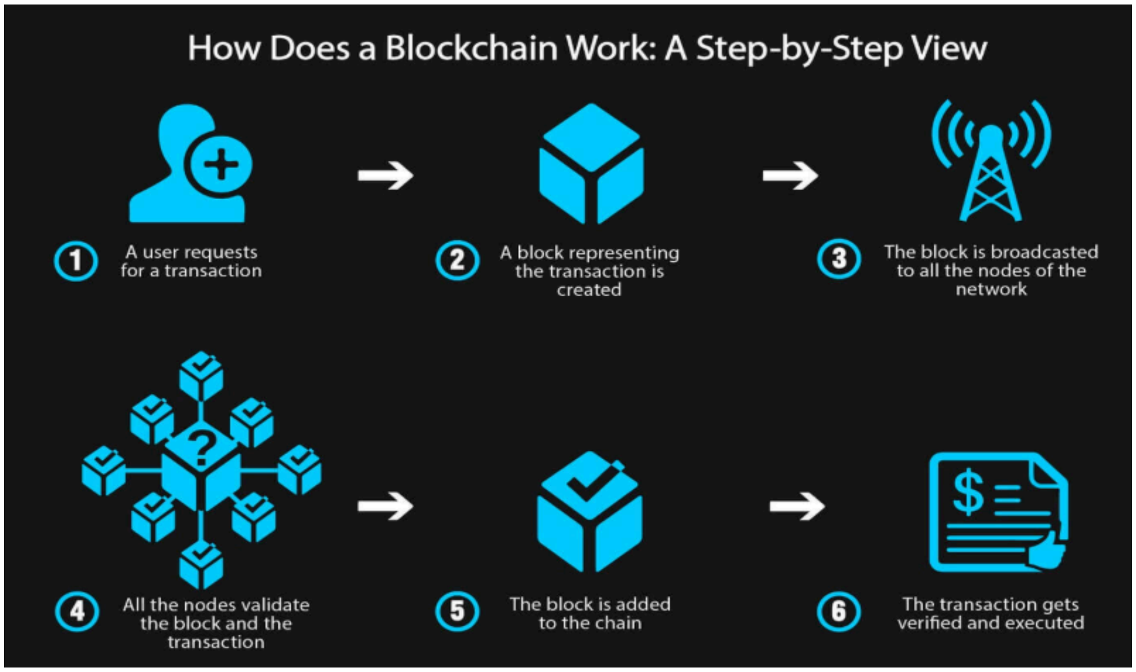
3. Process of mining
Mining is the method of adding new blocks to the blockchain by solving a puzzle.

Steps in mining:

1. **Transaction broadcast:** Users send transactions to the network.
2. **Block formation:** Miners collect these transactions into a candidate block.

3. **Proof of Work:** Miners try different numbers (nonce) until the hash of the block starts with a fixed number of zeros.
4. **Block validation:** The first miner to solve the puzzle shares the solution, and other nodes verify it.
5. **Block addition:** Once verified, the block is added permanently to the chain.
6. **Reward:** The miner receives incentives (like cryptocurrency or recognition).

Mining ensures that adding blocks requires effort, which protects the blockchain from tampering.


How Does a Blockchain Work: A Step-by-Step View
1 A user requests for a transaction
2 A block representing the transaction is created
3 The block is broadcasted to all the nodes of the network
4 All the nodes validate the block and the transaction
5 The block is added to the chain
6 The transaction gets verified and executed

4. How to check validity of block in blockchain
To confirm that a blockchain is valid, each block must pass certain checks:

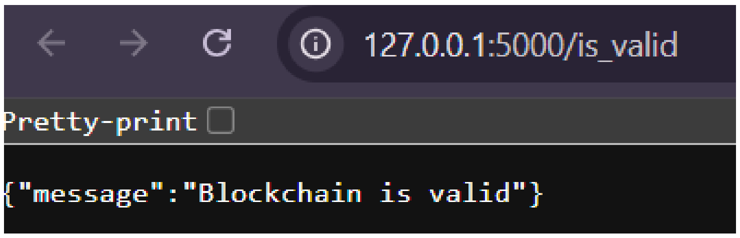
- **Hash linkage:** The previous_hash of a block must match the actual hash of the block before it.
- **Proof of Work check:** The block's proof must satisfy the condition (hash starting with zeros).
- **Transaction integrity:** All transactions inside the block must follow the rules of the network.
- **Chain consistency:** If any block is altered, its hash changes, breaking the chain and making it invalid.

In short, validity ensures that the blockchain has not been tampered with and all blocks are correctly linked.

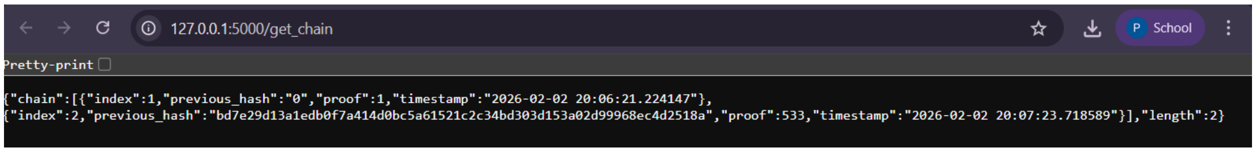
Code :

```
# ---------------------------------------------------------
# Simple Blockchain Implementation using Python & Flask
# ---------------------------------------------------------

# Required installation:
# pip install flask==2.2.5
import datetime
import hashlib
import json
from flask import Flask, jsonify


# ---------------------------------------------------------
# Blockchain Class Definition
# ---------------------------------------------------------
class Blockchain:
    def __init__(self):
        self.chain = []
        # Create the first block (Genesis Block)
        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
        while True:
            hash_value = hashlib.sha256(
                str(new_proof**2 - previous_proof**2).encode()
            ).hexdigest()
            if hash_value[:4] == '0000':
                return new_proof
            new_proof += 1
    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]
        index = 1

        while index < len(chain):
            current_block = chain[index]

            # Check if previous hash matches
            if current_block['previous_hash'] != self.hash(previous_block):
                return False

            # Check Proof of Work validity
            previous_proof = previous_block['proof']
            proof = current_block['proof']
            hash_value = hashlib.sha256(
                str(proof**2 - previous_proof**2).encode()
            ).hexdigest()
            if hash_value[:4] != '0000':
                return False
            previous_block = current_block
            index += 1
        return True
# -----------------------------------------------------------
# Flask Application Setup
# -----------------------------------------------------------
app = Flask(__name__)
blockchain = Blockchain()

# Home route (to avoid 404 at root URL)
@app.route('/', methods=['GET'])
def home():
    return "Welcome to Blockchain Lab! Use /mine_block, /get_chain, /is_valid"

@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': 'New block has been successfully mined',
        'index': block['index'],
        'timestamp': block['timestamp'],
        'proof': block['proof'],
        'previous_hash': block['previous_hash']
    }
    return jsonify(response), 200

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

@app.route('/is_valid', methods=['GET'])
def is_valid():
    result = blockchain.is_chain_valid(blockchain.chain)
    if result:
        return jsonify({'message': 'Blockchain is valid'}), 200
    else:
        return jsonify({'message': 'Blockchain is invalid'}), 200
# ------------------------------------------------------------
# Run the server
# ------------------------------------------------------------
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

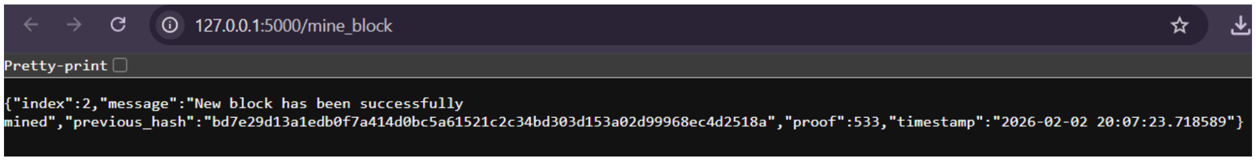
Output :



```
{"message":"Blockchain is valid"}
```



```
{"chain":[{"index":1,"previous_hash":"0","proof":1,"timestamp":"2026-02-02 20:06:21.224147"},
{"index":2,"previous_hash":"bd7e29d13a1edb0f7a414d0bc5a61521c2c34bd303d153a02d99968ec4d2518a","proof":533,"timestamp":"2026-02-02 20:07:23.718589"}],"length":2}
```


127.0.0.1:5000/mine_block

Pretty-print

```
{"index":2,"message":"New block has been successfully
mined","previous_hash":"bd7e29d13a1edb0f7a414d0bc5a61521c2c34bd303d153a02d99968ec4d2518a","proof":533,"timestamp":"2026-02-02 20:07:23.718589"}
```

Conclusion:
In this experiment, we created a simple blockchain using Python and Flask. Each block was linked using cryptographic hashing, and Proof of Work was used to secure the chain. By checking validity, we ensured that the blockchain remains consistent and trustworthy. This experiment demonstrates the core principles of blockchain — immutability, decentralization, and security — in a simplified environment.