Shravani Anil Patil
D20A-40

Experiment :02
Blockchain Lab

Aim: Create a Blockchain using Python
Theory:

1. What is Blockchain?

Blockchain is a technology where multiple parties involved in communication can perform different transactions without third-party intervention. Special nodes verify and validate these transactions.

Features of Blockchain

i. Decentralization: In centralized transaction systems, each transaction must be validated in the central trusted agency (e.g., the central bank), naturally resulting in cost and performance jams at the central servers. In contrast to the centralized mode, a third party is not needed in the blockchain. Consensus algorithms in blockchain maintain data stability in a decentralized network.

ii. Persistence: Transactions can be validated quickly and invalid transactions would not be admitted by persons or miners who mining the crypto. It is not possible to delete or roll back transactions once they are included in the blockchain network. Invalid transactions do not carry forward further.

iii.. Anonymity: Each user can interact with the blockchain with a generated address, which does not disclose the real identity
of the miner. Note that blockchain cannot guarantee perfect privacy preservation due to the permanent thing.

iv. Auditability: Blockchain stores data of users based on the Unspent Transaction Output (UTXO) model.

Every transaction has to refer to some previous unspent transactions. Once the current transaction is recorded into the
blockchain, the position of those referred unspent transactions switches from unspent to spent. Due to this process, the transactions can be easily tracked and not harmed between transactions.

v. Transparency: The transparency of blockchain is like cryptocurrency, in Bitcoin for tracking every transaction is done by the address. For security, it hides the person's identity between and after the transaction. All the transactions are made by the owner of the block associated with the address, this process is transparent and there is no loss for anyone who is involved in this transaction.

vi. Cryptography: The blockchain concept is fully based on security and for that, all the blocks on the blockchain network want to be secure. For security, it implements cryptography and secures the data using the cipher text and ciphers.

2. Core Components of Blockchain

i. Node: Nodes are network participants and their devices permit them to keep track of the distributed ledger and serve as communication hubs in various network tasks. A block broadcasts all the network nodes when a miner looks to add a new block in transactions to the blockchain.

ii. Transactions: A transaction refers to a contract or agreement and transfers of assets between parties. The asset is typically cash or property. The network of computers in blockchain stores the transactional data as a copy with the storage typically referred to as a digital ledger.

iii. Block: A block in a blockchain network is similar to a link in a chain. In the field of cryptocurrency, blocks are like records that store transactions like a record book, and those are encrypted into a hash tree. There are a huge number of transactions occurring every day in the world. The users need to keep track of those transactions, and they do it with the help of a block structure. The block structure of the blockchain is mentioned in the very first diagram in this article.

iv. Chain: Chain is the concept where all the blocks are connected with the help of a chain in the whole blockchain structure in the world. And those blocks are connected with the help of the previous block hash and it indicates a chaining structure.

v. Miners: Blockchain mining is a process that validates every step in the transactions while operating all cryptocurrencies. People involved in this mining they called miners. Blockchain mining is a process to validate each step in the transactions while operating cryptocurrencies.

vi. Consensus: A consensus is a fault-tolerant mechanism that is used in computer and blockchain systems to achieve the necessary agreement on a single state of the network among

distributed processes or multi-agent systems, such as with cryptocurrencies. It is useful in record keeping and other things.

Data Storage and Management

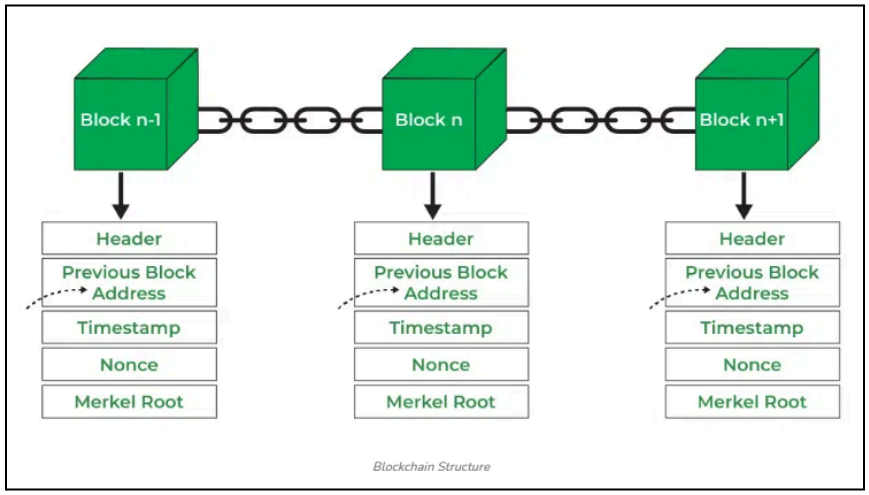
1. Header: It is used to identify the particular block in the entire blockchain. It handles all blocks in the blockchain. A block header is hashed periodically by miners by changing the nonce value as part of normal mining activity, also Three sets of block metadata are contained in the block header.

2. **Previous Block Address/ Hash:** It is used to connect the i+1th block to the ith block using the hash. In short, it is a reference to the hash of the previous (parent) block in the chain.

3. Timestamp: It is a system that verifies the data into the block and assigns a time or date of creation for digital documents. The timestamp is a string of characters that uniquely identifies the document or event and indicates when it was created.

4. Nonce: A nonce number which used only once. It is a central part of the proof of work in the block. It is compared to the live target if it is smaller or equal to the current target. People who mine, test, and eliminate many Nonce per second until they find that Valuable Nonce is valid.

5. Merkel Root: It is a type of data structure frame of different blocks of data. A Merkle Tree stores all the transactions in a block by producing a digital fingerprint of the entire transaction. It allows the users to verify whether a transaction can be included in a block or not.



Blockchain Structure

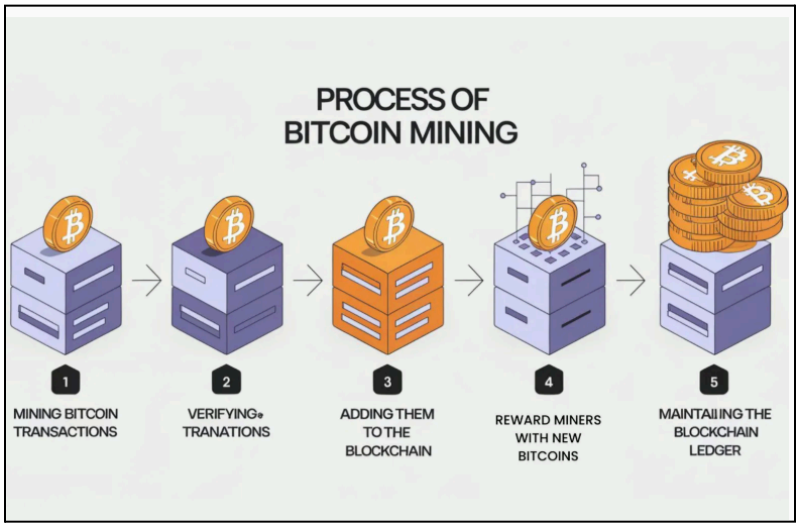
3. Process of Mining

Bitcoin Mining is the process of verifying bitcoin transactions and storing them in a blockchain(ledger). It is a process similar to gold mining but instead, it is a computer process that creates new bitcoin in addition to tracking Bitcoin transactions. Let's further study bitcoin and the various concepts related to it.

Bitcoin mining is a computation-intensive process that uses complicated computer code to generate a secure cryptographic system. The bitcoin miner is the person who solves mathematical puzzles(also called proof of work) to validate the transaction. Anyone with mining hardware and computing power can take part in this. Numerous miners take part simultaneously to solve the complex mathematical puzzle, the one who solves it first, wins 6.25 bitcoin as a part of the reward. Miner verifies the transactions(after solving the puzzle) and then adds the block to the blockchain when confirmed. The blockchain contains the history of every transaction that has taken place in the blockchain network. Once the miner adds the block to the blockchain, bitcoins are then transferred which are associated with the transaction.

For the miners to earn rewards from verifying the bitcoin Transactions, two things must be ensured:

1. The miners must verify the one-megabyte size of the transaction.
2. For the addition of a new block of transaction in the blockchain, miners must have the ability to solve complex computational maths problems called proof for work by finding a 64-bit hexadecimal hash value.



4. How to check validity of blocks in blockchain ?

i. The validity of a blockchain begins by verifying the connection between consecutive blocks. Each block stores the hash of the previous block, and this stored value is compared with the newly calculated hash of the previous block. If both hashes match, the link between the blocks is considered valid; otherwise, the chain is marked invalid.

ii. The proof-of-work of each block is then checked to ensure that the block was mined correctly. The proof value is recalculated using the same mathematical formula, and the resulting hash must satisfy the predefined difficulty condition, such as having leading zeros. This confirms that sufficient computational effort was applied during mining.

iii. The integrity of the block data is verified by recalculating the hash of the entire block. Any modification in transaction details, timestamp, or proof value will change the hash, immediately revealing tampering.

iv. The validation process is performed sequentially from the genesis block to the most recent block. Each block must pass the previous hash check and proof-of-work verification to maintain the continuity of the blockchain.

• If all blocks satisfy these conditions, the blockchain is declared valid. If any block fails the verification process, the entire blockchain is considered compromised, ensuring security and immutability of the data.

```
C:\Users\INFT506-34>python --version
Python 3.14.2

C:\Users\INFT506-34>pip install  flask
Collecting flask
  Downloading flask-3.1.2-py3-none-any.whl.metadata (3.2 kB)
Collecting blinker>=1.9.0 (from flask)
  Downloading blinker-1.9.0-py3-none-any.whl.metadata (1.6 kB)
Collecting click>=8.1.3 (from flask)
  Downloading click-8.3.1-py3-none-any.whl.metadata (2.6 kB)
Collecting itsdangerous>=2.2.0 (from flask)
  Downloading itsdangerous-2.2.0-py3-none-any.whl.metadata (1.9 kB)
Collecting jinja2>=3.1.2 (from flask)
  Downloading jinja2-3.1.6-py3-none-any.whl.metadata (2.9 kB)
Collecting markupsafe>=2.1.1 (from flask)
  Downloading markupsafe-3.0.3-cp314-cp314-win_amd64.whl.metadata (2.8 kB)
Collecting werkzeug>=3.1.0 (from flask)
  Downloading werkzeug-3.1.5-py3-none-any.whl.metadata (4.0 kB)
Collecting colorama (from click>=8.1.3->flask)
  Downloading colorama-0.4.6-py2.py3-none-any.whl.metadata (17 kB)
Downloading flask-3.1.2-py3-none-any.whl (103 kB)
Downloading blinker-1.9.0-py3-none-any.whl (8.5 kB)
Downloading click-8.3.1-py3-none-any.whl (108 kB)
Downloading itsdangerous-2.2.0-py3-none-any.whl (16 kB)
Downloading jinja2-3.1.6-py3-none-any.whl (134 kB)
Downloading markupsafe-3.0.3-cp314-cp314-win_amd64.whl (15 kB)
Downloading werkzeug-3.1.5-py3-none-any.whl (225 kB)
Downloading colorama-0.4.6-py2.py3-none-any.whl (25 kB)
```

```
Downloading jinja2-3.1.6-py3-none-any.whl (134 kB)
Downloading markupsafe-3.0.3-cp314-cp314-win_amd64.whl (15 kB)
Downloading werkzeug-3.1.5-py3-none-any.whl (225 kB)
Downloading colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Installing collected packages: markupsafe, itsdangerous, colorama, blinker, werkzeug, jinja2, click, flask
Successfully installed blinker-1.9.0 click-8.3.1 colorama-0.4.6 flask-3.1.2 itsdangerous-2.2.0 jinja2-3.1.6 markupsafe-3
.0.3 werkzeug-3.1.5
```

block.py

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

class Blockchain:
    def __init__(self):
        self.chain = []
        self.transactions = []
        # Create genesis block with mining hash
```

```
    self.create_block(proof=1, previous_hash='0', mining_hash='0'*64)

def create_block(self, proof, previous_hash, mining_hash):
    block = {
        'index': len(self.chain) + 1,
        'timestamp': str(datetime.datetime.now()),
        'proof': proof,
        'previous_hash': previous_hash,
        'mining_hash': mining_hash,   # <-- Added mining hash
        'transactions': self.transactions
    }
    self.transactions = []
    self.chain.append(block)
    return block

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

# Proof of Work (4 leading zeroes)
def proof_of_work(self, previous_proof):
    new_proof = 1
    while True:
        hash_operation = hashlib.sha256(
            str(new_proof**2 - previous_proof**2).encode()
        ).hexdigest()

        if hash_operation[:4] == "0000":
            return new_proof, hash_operation  # <-- return the hash too
        new_proof += 1

# Hash a block
def hash(self, block):
    encoded_block = json.dumps(block, sort_keys=True).encode()
    return hashlib.sha256(encoded_block).hexdigest()

# Add a transaction
def add_transaction(self, sender, receiver, amount):
    self.transactions.append({
        'sender': sender,
        'receiver': receiver,
```

```
            'amount': amount
        })
        return self.get_previous_block()['index'] + 1

    # Check if blockchain is valid
    def is_chain_valid(self, chain):
        previous_block = chain[0]
        index = 1

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

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

            # Check 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

            previous_block = block
            index += 1

        return True
```

app.py

```
app = Flask(__name__)
blockchain = Blockchain()

# Mine a new block
@app.route('/mine_block', methods=['GET'])
def mine_block():
    previous_block = blockchain.get_previous_block()
```

```
    proof, mining_hash = blockchain.proof_of_work(previous_block['proof'])
    previous_hash = blockchain.hash(previous_block)
    block = blockchain.create_block(proof, previous_hash, mining_hash)

    return jsonify({
        'message': 'Block mined successfully',
        'block': block
    }), 200

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

# Check if blockchain is valid
@app.route('/is_valid', methods=['GET'])
def is_valid():
    return jsonify({
        'valid': blockchain.is_chain_valid(blockchain.chain)
    }), 200

# Add a transaction
@app.route('/add_transaction', methods=['POST'])
def add_transaction():
    data = request.get_json()

    if not all(k in data for k in ('sender', 'receiver', 'amount')):
        return jsonify({'error': 'Missing data'}), 400

    index = blockchain.add_transaction(
        data['sender'],
        data['receiver'],
        data['amount']
    )

    return jsonify({
        'message': f'Transaction will be added to block {index}'
```

```
    }), 201

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

```
127.0.0.1 - - [22/Jan/2026 12:08:08] "GET /mine_block HTTP/1.1" 200 -
MINING HASH: 00000f1a0dbcb6f6e74d8b2e15153e0320119ed7cb7519683bc116851d6d4fc3
127.0.0.1 - - [22/Jan/2026 12:08:10] "GET /mine_block HTTP/1.1" 200 -
 * Detected change in 'C:\\Users\\INFT506-34\\Desktop\\blockchain\\block.py', reloading
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 105-211-773
127.0.0.1 - - [22/Jan/2026 12:09:06] "GET /mine_block HTTP/1.1" 200 -
127.0.0.1 - - [22/Jan/2026 12:09:09] "GET /mine_block HTTP/1.1" 200 -
127.0.0.1 - - [22/Jan/2026 12:09:10] "GET /mine_block HTTP/1.1" 200 -
127.0.0.1 - - [22/Jan/2026 12:09:41] "GET /get_chain HTTP/1.1" 200 -
```

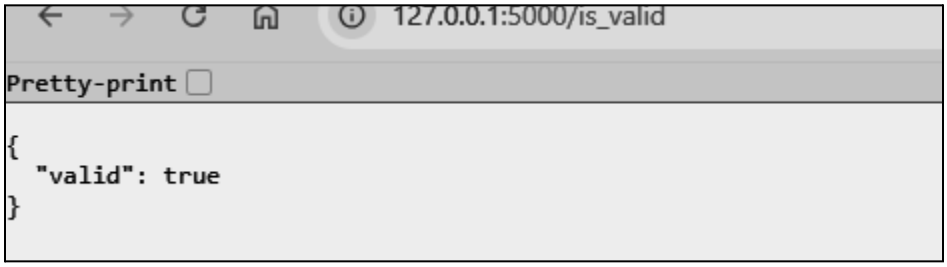
http://127.0.0.1:5000/get_chain

Pretty-print


```
{
  "chain": [
    {
      "index": 1,
      "mining_hash": "0000000000000000000000000000000000000000000000000000000000000000",
      "previous_hash": "0",
      "proof": 1,
      "timestamp": "2026-01-22 12:08:59.435754",
      "transactions": []
    },
    {
      "index": 2,
      "mining_hash": "0000c00870f23a23ae80377298491b091db400d575be0efbde5b310f2f763ed1",
      "previous_hash": "024d22c9512e10f992e64ef49ea39b600150ec46fc2e3b7bb853e6cd44f0cdd9",
      "proof": 533,
      "timestamp": "2026-01-22 12:09:06.684401",
      "transactions": []
    },
    {
      "index": 3,
      "mining_hash": "0000b8318b7ca4e2bc9be063d7dfcb0d0e70f1b27133001f714f11c058c31e99",
      "previous_hash": "849521ebed68399b1d5e78d35e4bcd67879a1cc9c758b81eb44f26389900cee9",
      "proof": 45293,
      "timestamp": "2026-01-22 12:09:09.954630",
      "transactions": []
    },
    {
      "index": 4,
      "mining_hash": "000012c643d9610a15e350cb92ec6386b2b15e9d4170a0f06a4fd8525e440207",
      "previous_hash": "aaa71448b78546eb1c356108de526a11390fc4ea6d1168ef550f9d90c8886673",
      "proof": 21391,
      "timestamp": "2026-01-22 12:09:10.823606",
      "transactions": []
    }
  ],
  "length": 4
}
```

http://127.0.0.1:5000/mine_block

http://127.0.0.1:5000/is_valid



Conclusion:

In this experiment, a basic blockchain was successfully created and implemented using Python and the Flask framework. The blockchain system demonstrated the fundamental concepts of blockchain technology such as block creation, hashing, proof of work, transaction handling, and chain validation. A genesis block was generated, and subsequent blocks were mined using a proof-of-work mechanism, ensuring data integrity and security.

The implementation verified how blocks are linked using cryptographic hashes and how transactions are stored in a decentralized ledger structure. The validity of the blockchain was checked to ensure immutability and resistance to tampering. Through this experiment, the working principles of blockchain, mining, and consensus mechanisms were clearly understood, thereby achieving the objective of creating a simple blockchain using Python.