Sipna College of Engineering & Technology, Amravati.
Department of Computer Science & Engineering
Session 2022-2023

Branch :- Computer Sci. & Engg. **Class :- Final Year**
Subject :-Block Chain Fundamentals Lab manual **Sem :- VII**

Teacher Manual

PRACTICAL NO 6

AIM: To Create Your Own Cryptocurrency Using Python

S/W REQUIRED: Phython

Cryptocurrency

In computer science, a cryptocurrency, crypto-currency, or crypto is a digital currency that does not rely on any central authority to uphold or maintain it. Instead, transaction and ownership data is stored in a digital ledger using distributed ledger technology, typically a blockchain.

A Cryptocurrency is a system that meets six conditions:

1. The system does not require a central authority; its state is maintained through distributed consensus.
2. The system keeps an overview of cryptocurrency units and their ownership.
3. The system defines whether new cryptocurrency units can be created. If new cryptocurrency units can be created, the system defines the circumstances of their origin and how to determine the ownership of these new units.
4. Ownership of cryptocurrency units can be proved exclusively cryptographically.
5. The system allows transactions to be performed in which ownership of the cryptographic units is changed. A transaction statement can only be issued by an entity proving the current ownership of these units.
6. If two different instructions for changing the ownership of the same cryptographic units are simultaneously entered, the system performs at most one of them.

Implementation:

```
import hashlib
import time


class Block:

    def __init__(self, index, proof_no, prev_hash, data, timestamp=None):
        self.index = index
        self.proof_no = proof_no
        self.prev_hash = prev_hash
        self.data = data
        self.timestamp = timestamp or time.time()

    @property
    def calculate_hash(self):
        block_of_string = "{}{}{}{}{}".format(self.index, self.proof_no,
                                              self.prev_hash, self.data,
                                              self.timestamp)
        return hashlib.sha256(block_of_string.encode()).hexdigest()

    def __repr__(self):
        return "{} - {} - {} - {} - {}".format(self.index, self.proof_no,
                                               self.prev_hash, self.data,
                                               self.timestamp)


class BlockChain:

    def __init__(self):
        self.chain = []
        self.current_data = []
        self.nodes = set()
        self.construct_genesis()

    def construct_genesis(self):
        self.construct_block(proof_no=0, prev_hash=0)

    def construct_block(self, proof_no, prev_hash):
        block = Block(
            index=len(self.chain),
            proof_no=proof_no,
            prev_hash=prev_hash,
            data=self.current_data)
        self.current_data = []

        self.chain.append(block)
        return block

    @staticmethod
    def check_validity(block, prev_block):
        if prev_block.index + 1 != block.index:
            return False

        elif prev_block.calculate_hash != block.prev_hash:
            return False

        elif not BlockChain.verifying_proof(block.proof_no,
                                            prev_block.proof_no):
            return False

        elif block.timestamp <= prev_block.timestamp:
            return False

        return True

    def new_data(self, sender, recipient, quantity):
        self.current_data.append({
            'sender': sender,
            'recipient': recipient,
            'quantity': quantity
        })
        return True

    @staticmethod
    def proof_of_work(last_proof):
        '''this simple algorithm identifies a number f' such that hash(ff') contain 4 leading zeroes
         f is the previous f'
         f' is the new proof
        '''
        proof_no = 0
        while BlockChain.verifying_proof(proof_no, last_proof) is False:
            proof_no += 1

        return proof_no

    @staticmethod
    def verifying_proof(last_proof, proof):
        #verifying the proof: does hash(last_proof, proof) contain 4 leading zeroes?

        guess = f'{last_proof}{proof}'.encode()
        guess_hash = hashlib.sha256(guess).hexdigest()
        return guess_hash[:4] == "0000"

    @property
    def latest_block(self):
        return self.chain[-1]

    def block_mining(self, details_miner):

        self.new_data(
            sender="Sipna COET",  #it implies that this node has created a new block
            receiver=details_miner,
            quantity=
            1,  #creating a new block (or identifying the proof number) is awarded with 1
        )

        last_block = self.latest_block

        last_proof_no = last_block.proof_no
        proof_no = self.proof_of_work(last_proof_no)

        last_hash = last_block.calculate_hash
        block = self.construct_block(proof_no, last_hash)

        return vars(block)

    def create_node(self, address):
        self.nodes.add(address)
        return True

    @staticmethod
    def obtain_block_object(block_data):
        #obtains block object from the block data
        return Block(
            block_data['index'],
            block_data['proof_no'],
            block_data['prev_hash'],
            block_data['data'],
            timestamp=block_data['timestamp'])

blockchain = BlockChain()

print("***Mining fccCoin about to start***")
print(blockchain.chain)

last_block = blockchain.latest_block
last_proof_no = last_block.proof_no
proof_no = blockchain.proof_of_work(last_proof_no)

blockchain.new_data(
    sender="Sipna COET",  #it implies that this node has created a new block
    recipient="SIPNA CSE Department",  #let's send Sipna CSE some coins!
    quantity=
    1,  #creating a new block (or identifying the proof number) is awarded with 1
)

last_hash = last_block.calculate_hash
block = blockchain.construct_block(proof_no, last_hash)

print("***Mining fccCoin has been successful***")
print(blockchain.chain)
```

Output:

```
PS C:\Users\PC-1059\hello>  & 'C:\Python310\python.exe' 'c:\Users\PC-1059\.vscode\extensions\ms-python.python-2022.14.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '51936' '--' 'c:\Users\PC-1059\hello\cryptocurrency.py'
***Mining fccCoin about to start***
[0 - 0 - 0 - [] - 1662359736.2395442]
***Mining fccCoin has been successful***
[0 - 0 - 0 - [] - 1662359736.2395442, 1 - 88914 - cf01d26b936e6e87a464b53979bbd9ce51e6e5fd50e6ba3b96cd6d4ae780dd80 - [{'sender': 'Sipna COET', 'recipient': 'SIPNA CSE Department', 'quantity': 1}] - 1662359736.6053078]
PS C:\Users\PC-1059\hello>
```

CONCLUSION: Thus we have studied and created our own Cryptocurrency Using Python.