# Программа на Python для создания блокчейна  
# Для временной метки  
import datetime  
# Вычисление хэша для добавления цифровой подписи к блокам  
import hashlib  
# Для хранения данных в блокчейне  
import json  
# Flask предназначен для создания веб-приложения, а jsonify - для  
# отображения блокчейнаn  
from flask import Flask, jsonify  
# подключение к бд  
import psycopg2  
import pandas as pd  
class Blockchain:  
 def \_\_init\_\_(self):  
 self.chain = []  
 self.create\_block(proof=1, hashed\_data='0')  
  
# Импорт базы данных  
def connect\_db():  
 conn = psycopg2.connect(dbname='postgres', user='postgres', password='1111', host='localhost')  
 cursor = conn.cursor()  
 cursor.execute('SELECT \* FROM "users"')  
 df = cursor.fetchall()  
 df = pd.DataFrame(df)  
  
 return df  
database = connect\_db()  
database[0].iloc[0] = 'Belashov Anton'  
database[2].iloc[0] = '89001234567'  
database[0].iloc[1] = 'Ivanov Ivan'  
database[2].iloc[1] = '89012345678'  
database[0].iloc[2] = 'Petrov Petr'  
database[2].iloc[2] = '89023456789'  
database[0].iloc[3] = 'Halpin Pavel'  
database[2].iloc[3] = '89034567890'  
database[0].iloc[4] = 'Kuryanov Denis'  
database[2].iloc[4] = '89045678901'  
print(database)  
class Blockchain:  
# Эта функция ниже создана для создания самого первого блока и установки его хэша равным "0"  
 def \_\_init\_\_(self):  
 self.chain = []  
 self.create\_block(proof=1, hashed\_data='0')  
# Эта функция ниже создана для добавления дополнительных блоков в цепочку  
 def create\_block(self, proof, hashed\_data):  
 block = {  
 'index': len(self.chain) + 1,  
 'name': str(database[0].iloc[len(self.chain)]),  
 'email': str(database[1].iloc[len(self.chain)]),  
 'phone': str(database[2].iloc[len(self.chain)]),  
 'hashed\_data': str(database[3].iloc[len(self.chain)]),  
 'adress': str(database[4].iloc[len(self.chain)]),  
 'proof': proof # добавлено  
 }  
  
 self.chain.append(block)  
 return block  
# Эта функция ниже создана для отображения предыдущего блока  
 def print\_previous\_block(self):  
 return self.chain[-1]  
# Это функция для проверки работы и используется для успешного майнинга блока  
 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[:5] == '00000':  
 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 chain\_valid(self, chain):  
 previous\_block = chain[0]  
 block\_index = 1  
 while block\_index < len(chain):  
 block = chain[block\_index]  
 if block['hashed\_data'] != 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[:5] != '00000':  
 return False  
 previous\_block = block  
 block\_index += 1  
 return True  
  
# Создание веб-приложения с использованием flask  
app = Flask(\_\_name\_\_)  
# Создаем объект класса blockchain  
blockchain = Blockchain()  
# Страница с подсказками  
@app.route('/')  
def index():  
 return "Майнинг нового блока: /mine\_block " \  
 "Отобразить блокчейн в формате json: /display\_chain " \  
 "Проверка валидности блокчейна: /valid "  
  
# Майнинг нового блока  
@app.route('/mine\_block', methods=['GET'])  
def mine\_block():  
 previous\_block = blockchain.print\_previous\_block()  
 previous\_proof = previous\_block['proof']  
 proof = blockchain.proof\_of\_work(previous\_block['proof']) # изменено  
 previous\_hash = blockchain.hash(previous\_block)  
 block = blockchain.create\_block(proof, previous\_hash)  
 response = {'message': 'A block is MINED',  
 'index': block['index'],  
 'name': block['name'],  
 'email': block['email'],  
 'phone': block['phone'],  
 'hashed\_data': block['hashed\_data'],  
 'adress': block['adress']}  
 return jsonify(response), 200  
# Отобразить блокчейн в формате json  
@app.route('/display\_chain', methods=['GET'])  
def display\_chain():  
 chain = []  
 for block in blockchain.chain:  
 data = {  
 'index': block['index'],  
 'name': block['name'],  
 'email': block['email'],  
 'phone': block['phone'],  
 'hashed\_data': block['hashed\_data'],  
 'adress': block['adress']  
 }  
 chain.append(data)  
 response = {'chain': chain, 'length': len(chain)}  
 return jsonify(response), 200  
# Проверка валидности блокчейна  
@app.route('/valid', methods=['GET'])  
def valid():  
 valid = blockchain.chain\_valid(blockchain.chain)  
 if valid:  
 response = {'message': 'The Blockchain is valid.'}  
 else:  
 response = {'message': 'The Blockchain is not valid.'}  
 return jsonify(response), 200  
# Запустите сервер flask локально  
app.run(debug = True)