import hashlib

import json

class Block():

def \_\_init\_\_(self,nonce,tstamp,transaction,prevhash=''):

self.nonce=nonce

self.tstamp=tstamp

self.transaction=transaction

self.prevhash=prevhash

self.hash=self.calcHash()

def calcHash(self): block\_string=json.dumps({"nonce":self.nonce,"tstamp":self.tstamp,"transaciton":self.transaction,"prevhash":self.prevhash},sort\_keys=True).encode()

return hashlib.sha256(block\_string).hexdigest()

def mineBlock(self,diffic):

while(self.hash[:diffic] != str('').zfill(diffic)):

self.nonce += 1

self.hash=self.calcHash() print("Block mined ", self.hash) def \_\_str\_\_(self): string="nonce: " + str(self.nonce) + "\n" string+= "tstamp: " + str(self.tstamp)+ "\n" string += "transaction: " + str(self.transaction)+ "\n" string += "prevhas: " + str (self.prevhash)+ "\n" string += "hash: " + str (self.hash)+ "\n" return string class BlockChain():

def \_\_init\_\_(self):

self.chain=[self.generateGenesisBlock(),] self.difficulty=3

def generateGenesisBlock(self):

return Block(0,'01/01/2017','Gensis Block')

def getLastBlock(self):

return self.chain[-1]

def addBlock(self,newBlock): newBlock.prevhash=self.getLastBlock().hash newBlock.mineBlock(self.difficulty) self.chain.append(newBlock)

def isChainValid(self):

for i in range(1,len(self.chain)):

prevb=self.chain[i-1] currb=self.chain[i] if(currb.hash != currb.calcHash()):

print("invalid block")

return False if(currb.prevhash != prevb.hash):

print("invalid chain")

return False

return True

osaCoin=BlockChain()

print("adding the first block")osaCoin.addBlock(Block(1,'20/05/2017',100))

print("adding the second block")osaCoin.addBlock(Block(2,'21/05/2017',20))