HW 4: Blockchain Construction

In this assignment you will extend and implement a class framework to create a simple but functional blockchain that combines the ideas of proof-of-work, transactions, blocks, and blockchains.

You may create new member functions, but DO NOT MODIFY any existing APIs.

These are the interface into your blockchain. The test functions will call these APIs!

This blockchain has the following consensus rules (it is a little different from Bitcoin to make testing easier):

Blockchain

1. There are no consensus rules pertaining to the minimum proof-of-work of any blocks. That is it has no "difficulty adjustment algorithm". Instead, your code will be expected to properly place blocks of different difficulty into the correct place in the blockchain and discover the most-work tip.

2. A block with no transactions (no coinbase) is valid (this will help us isolate tests).

3. If a block as > 0 transactions, the first transaction MUST be the coinbase transaction.

Merkle Tree of Transactions

1. You must use sha256 hash
2. You must use 0 if additional items are needed to pad odd merkle levels (more specific information is included below)

Transactions

1. A transaction with inputs==None is a valid mint (coinbase) transaction. For these transactions, the coins created must not exceed the per-block "minting" maximum.

2. If the transaction is not a coinbase transaction, coins cannot be created. In other words, coins spent (inputs) must be >= coins sent (outputs).

3. Constraint scripts (permission to spend) are implemented via python lambda expressions (anonymous functions). These constraint scripts must accept a list of parameters, and return True if permission to spend is granted. If execution of the constraint script throws an exception or returns anything except True do not allow spending!

Course Differences

461: You may assume that every submitted transaction is correct.

This means that you should just make the Transaction validate() function return True You do not need to worry about tracking the UTXO (unspent transaction outputs) set.

661: You need to verify transactions, their constraint and satisfier scripts, and track the UTXO set.

What to submit

Submit a .zip file with a single file named blockchain.py, or blockchain.py directly.

Do not rename 'blockchain.py', and implement everything in this one file.

Some useful library functions

Hashlib

Read about hashlib.sha256() to do sha256 hashing in python.

Byte to integer conversion

Convert the sha256 array of bytes to a big endian integer via: int.from_bytes(bunchOfBytes,"big")

Serialization

Read about the "dill" library to serialize objects automatically (dill.dumps()). "Dill" is like "pickle", but it can serialize python lambda functions.

You need to install it via "pip3 install dill".

You'll probably need this when calculating a transaction id.