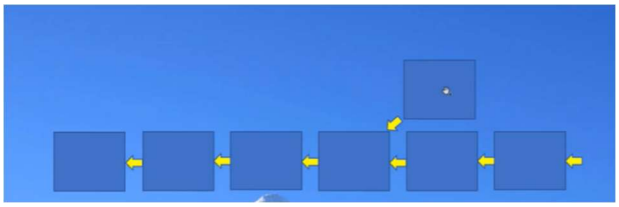
Vid 74 – Assignment 4: Blockchain branching

Clean up tasks → joining to a non-head node, which is necessary because we have miners that are receiving new blocks at different places

The wallet is storing a Blockchain. Let's say then a wallet receives (from a miner) the block on top, which doesn't fit on the end of the head node. Need to be able to add these because it could be that this will soon become the longest



Stopping the miner and then create a brand-new block that will attach NOT to this head node but to a block further back in the chain (0:00 – 1:10)

1:50 - 6:00 → Failing to Add Sister Block

```
new block has 0 txs.
Finding Nonce...
new block has 0 txs.
Finding Nonce...
new block has 0 txs.
Finding Nonce...
new block has 0 txs.
Finding Nonce...
new block has 0 txs.
Finding Nonce...
new block has 0 txs.
Finding Nonce...
new block has 0 txs.
Finding Nonce...
Good nonce found
Sending to localhost:5006
new block has 0 txs.Rec'd block

Finding Nonce...
new block has 0 txs.
Finding Nonce...
new block has 0 txs.
Finding Nonce...
-4.000000000000002
Error! Wrong balance for pu1
Error! Wrong balance for pu2
Error! Wrong balance for pu3
Rec'd block
Saving 0 txs to Txs.dat
Error! Failed to add sister block
Exit successful.
```

Wallet.py - C:\Users\Russ\Python36\Wallet.py (3.6.4)

File Edit Format Run Options Window Help

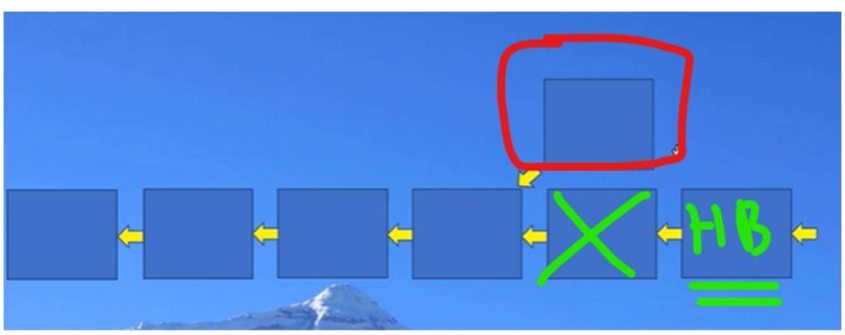
```
    Miner.StopAll()

    num_heads = len(head_blocks)
    sister = TxBlock.TxBlock(head_blocks[0].previousBlock.previousBlock)
    sister.previousBlock = None
    SocketUtils.sendObj('localhost',sister,5006)
    time.sleep(10)
    if (len(head_blocks) == num_heads + 1):
        print ("Success! New head_block created")
    else:
        print("Error! Failed to add sister block")

    StopAll()

    t1.join()
    t2.join()
    t3.join()

    print ("Exit successful.")
```

7:48 → Creating Genesis.dat making a copy of WalletBlocks.dat and renaming that copy to Genesis.dat

supposed to ensure that the wallet and miner can always get back in sync

Vid 75 – Assignment 4: Solution

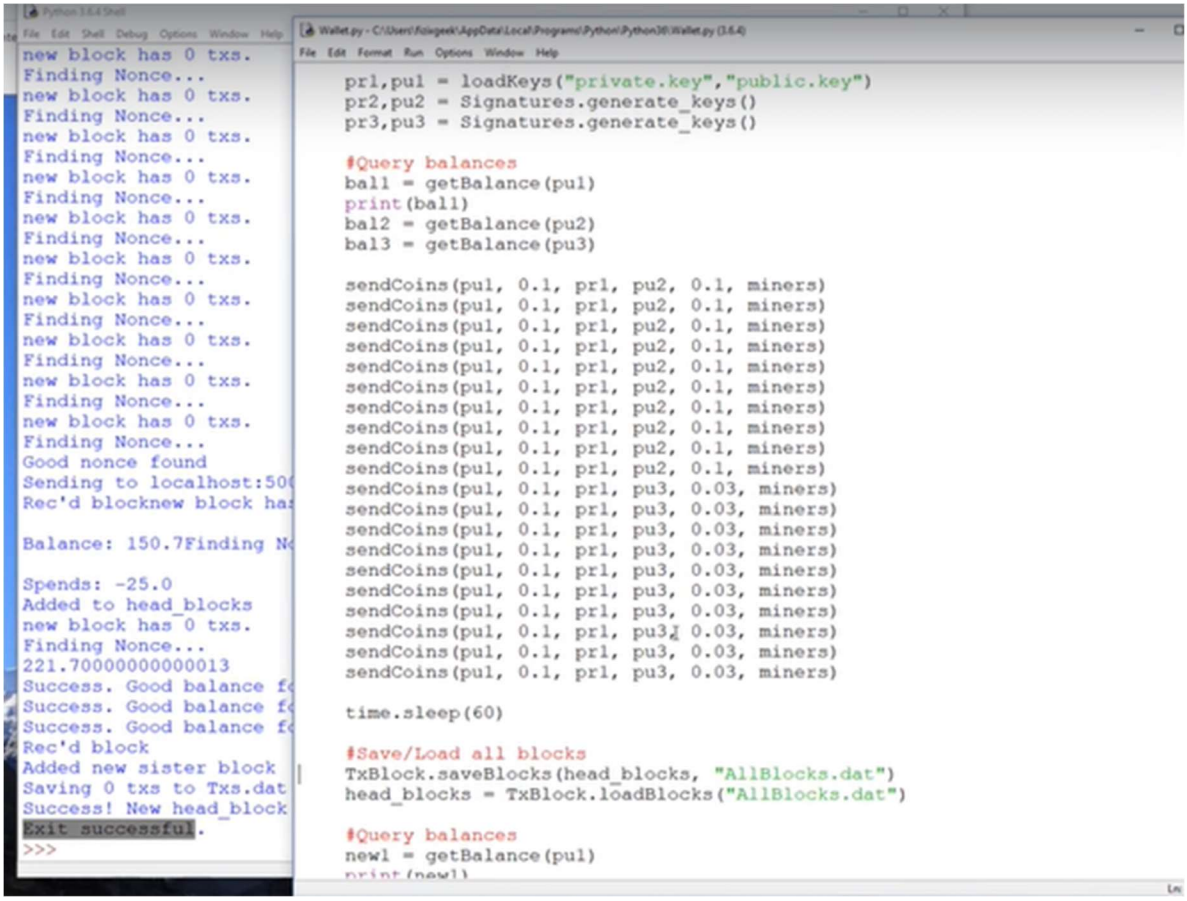
Reviewing walletServer function → and how the new block added to the longest chain becomes the new head block and the previousBlock is removed as the head block (0:00 - 1:45)



This code just adds sister blocks over and over

```
else:
    this_block = b
    while this_block != None:
        if newBlock.previousHash == this_block.previousHash:
            newBlock.previousBlock = this_block.previousBlock
            head_blocks.append(newBlock)
            if verbose: print("Added new sister block")


        this_block = this_block.previousBlock
```

Correct output for the wallet → last time I had correct balances was before started adding to a branch chain!!



Vid 76 – Assignment 5: Replay Attacks

Avoiding Duplicate Transactions (separates ETH from BTC)

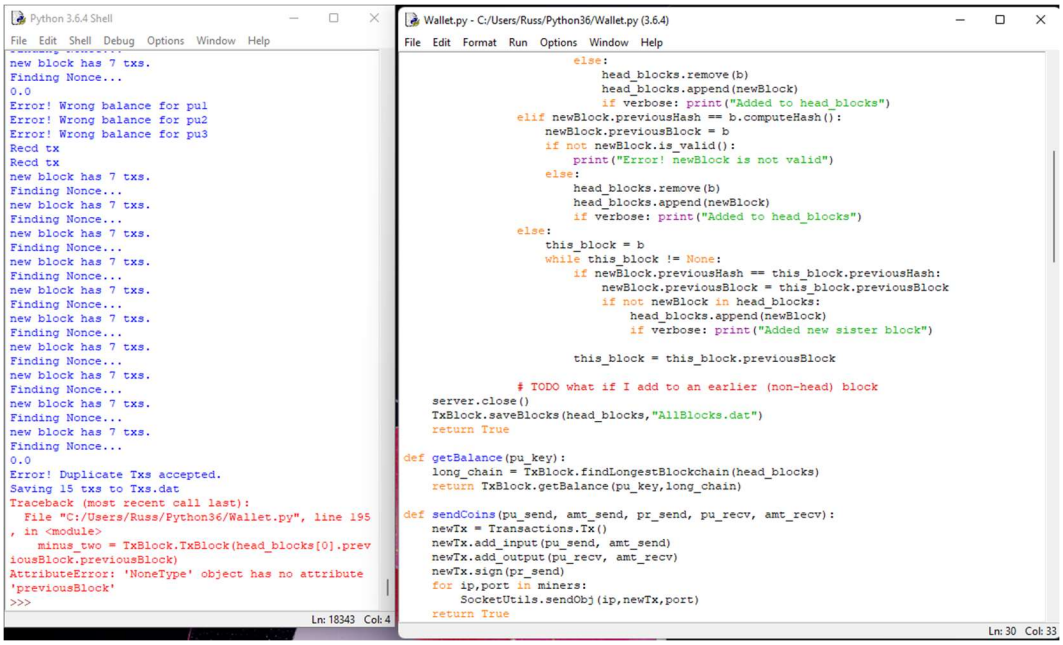
We're playing the role of the hacker, one of the ways to do this is by creating duplicate txs

Miner opens the server connection, received txs from wallets, If it finds an actual tx it appends it to the tx list

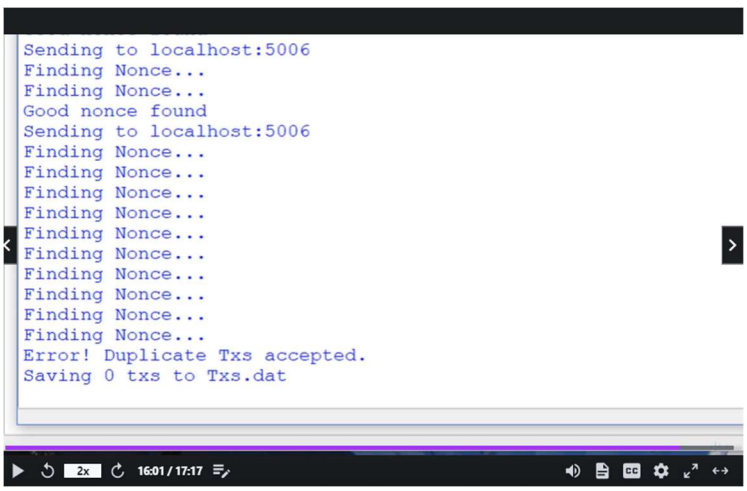
```
if __name__ == "__main__":

    import time
    import Miner
    import threading
    import Signatures
    # recieve and open a connection, if its an actual tx then its going to open
    def Thief(my_addr):
        my_ip, my_port = my_addr
        server = SocketUtils.newServerConnection(my_ip,my_port)
        # Get Txs from wallets
        while not break_now:
            newTx = SocketUtils.recvObj(server)
            if isinstance(newTx,Transactions.Tx):
                for ip,port in miners:
                    if not (ip == my_ip and port == my_port):
                        pass
                    else:
                        SocketUtils.sendObj(ip,newTx,port)

    Miner.saveTxList([], "Txs.dat")
```

10:50 → when this is true (when its my own address) then not this is gonna be false. When this is NOT true (if its not me) then send it to the port.

15:10 → Output



Professors Output



Vid 77 – Assignment 5: Solution

Ethereum calls this the tx nonce, which is just the index of the tx

```
def sendCoins(pu_send, amt_send, pr_send, pu_recv, amt_recv):
    newTx = Transactions.Tx()
    newTx.add_input(pu_send, amt_send, tx_index)
    newTx.add_output(pu_recv, amt_recv)
    newTx.sign(pr_send)
    for ip,port in miners:
        SocketUtils.sendObj(ip,newTx,port)
    return True
```

Our wallet should be keeping track of these tx_inx

```
#Wallet
import SocketUtils
import Transactions
import TxBlock
import pickle
import Signatures

head_blocks = [None]
wallets = [('localhost',5006)]
miners = [('localhost',5005)]
break_now = False
verbose = False
my_private,my_public = Signatures.generate_keys()
tx_index = 0
```

Correct Output for Transactions

Python 3.6.4 Shell

File Edit Shell Debug Options Window Help

```
AttributeError: 'NoneType' object has no attribute
'previousBlock'
>>>
============== RESTART: C:\Users\Russ\Python36\Tran
sactions.py ==============
Success! Tx is valid
Success! Tx is valid
Success! Tx is valid
Success! Tx is valid
Success! Bad Tx is invalid
Success! Bad Tx is invalid
Success! Bad Tx is invalid
ERROR! Bad Tx is valid
Success! Bad Tx is invalid
Success! Bad Tx is invalid
```

Transactions.py - C:\Users\Russ\Python36\Transactions.py (3.6.4)

File Edit Format Run Options Window Help

```python
#Transaction.py
import Signatures
#Signatures.sign
#Signatures.verify

class Tx:
    inputs = None
    outputs =None
    sigs = None
    reqd = None
    def __init__(self):
        self.inputs = []
        self.outputs = []
        self.sigs = []
        self.reqd = []
    def add_input(self, from_addr, amount, index=0):
        self.inputs.append((from_addr, amount, index))
    def add_output(self, to_addr, amount):
        self.outputs.append((to_addr, amount))
    def add_reqd(self, addr):
        self.reqd.append(addr)
    def sign(self, private):
        message = self.__gather()
        newsig = Signatures.sign(message, private)
        self.sigs.append(newsig)
    def is_valid(self):
        total_in = 0
        total_out = 0
        message = self.__gather()
        for addr,amount,inx in self.inputs:
            found = False
            for s in self.sigs:
                if Signatures.verify(message, s, addr) :
                    found = True
            if not found:
                #print ("No good sig found for " + str(message))
                return False
            if amount < 0:
                return False
            total_in = total_in + amount
```

TxBlock.py - C:/Users/Russ/Python36/TxBlock.py (3.6.4)

File Edit Format Run Options Window Help

```python
#TxBlock
from BlockChain import CBlock
from Signatures import generate_keys, sign, verify
from Transactions import Tx
import pickle
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.backends import default_backend
import random
from cryptography.hazmat.primitives import hashes
import time

reward = 25.0
leading_zeros = 2
next_char_limit = 100
verbose = True

class TxBlock (CBlock):
    nonce = "AAAAAAA"
    def __init__(self, previousBlock):
        super(TxBlock, self).__init__([], previousBlock)
    def addTx(self, Tx_in):
        self.data.append(Tx_in)
    def removeTx(self, Tx_in):
        if Tx_in in self.data:
            self.data.remove(Tx_in)
            return True
        return False
    def count_totals(self):
        total_in = 0
        total_out = 0
        for tx in self.data:
            for addr,amt,inx in tx.inputs:
                total_in = total_in + amt
            for addr,amt in tx.outputs:
                total_out = total_out + amt
        return total_in, total_out
    def check_size(self):
        savePrev = self.previousBlock
        self.previousBlock = None
        this_size = len(pickle.dumps(self))
        self.previousBlock = savePrev
        if this_size > 10000:
            return False
        return True
```

```python
def is_valid(self):
    if not super(TxBlock, self).is_valid():
        return False
    spends={}
    for tx in self.data:
        if not tx.is_valid():
            return False
        for addr,amt,inx in tx.inputs:
            if addr in spends:
                spends[addr] = spends[addr] + amt
            else:
                spends[addr] = amt
        for addr,amt in tx.outputs:
            if addr in spends:
                spends[addr] = spends[addr] - amt
            else:
                spends[addr] = -amt
    for this_addr in spends:
        if verbose: print ("Balance: " + str(getBalance(this_addr,self.previ
        if verbose: print ("Spends: " + str(spends[this_addr]))
        if spends[this_addr] - getBalance(this_addr,self.previousBlock) > 0.
            return False

    total_in, total_out = self.count_totals()
    if total_out - total_in - reward > 0.000000000001:
        return False
    if not self.check_size():
        return False
    return True
```

```python
def getBalance(pu_key,last_block):
    this_block = last_block
    bal = 0.0
    while this_block != None:
        for tx in this_block.data:
            for addr,amt,inx in tx.inputs:
                if addr == pu_key:
                    bal = bal - amt
            for addr,amt in tx.outputs:
                if addr == pu_key:
                    bal = bal + amt
        this_block = this_block.previousBlock
    return bal
```

Transactions.py - C:\Users\Russ\Python36\Transactions.py (3.6.4)

File Edit Format Run Options Window Help

```python
Tx5.add_reqd(pu4)
Tx5.sign(pr3)

# Two input addrs, signed by one
Tx6 = Tx()
Tx6.add_input(pu3, 1)
Tx6.add_input(pu4, 0.1)
Tx6.add_output(pu1, 1.1)
Tx6.sign(pr3)

# Outputs exceed inputs
Tx7 = Tx()
Tx7.add_input(pu4, 1.2)
Tx7.add_output(pu1, 1)
Tx7.add_output(pu2, 2)
Tx7.sign(pr4)

# Negative values
Tx8 = Tx()
Tx8.add_input(pu2, -1)
Tx8.add_output(pu1, -1)
Tx8.sign(pr2)

# Modified Tx
Tx9 = Tx()
Tx9.add_input(pu1, 1)
Tx9.add_output(pu2, 1)
Tx9.sign(pr1)
# outputs = [(pu2,1)]
# change to [(pu3,1)]
Tx9.outputs[0] = (pu3,1)


for t in [Tx4, Tx5, Tx6, Tx7, Tx8, Tx9]:
    if t.is_valid():
        print("ERROR! Bad Tx is valid")
    else:
        print("Success! Bad Tx is invalid")
```

```python
def __repr__(self):
    reprstr = "INPUTS:\n"
    for addr, amt, inx in self.inputs:
        reprstr = reprstr + str(amt) + " from " + str(addr) + " inx = " + str(inx) + "\n"
    reprstr = reprstr + "OUTPUTS:\n"
    for addr, amt in self.outputs:
        reprstr = reprstr + str(amt) + " to " + str(addr) + "\n"
    reprstr = reprstr + "REQD:\n"
    for r in self.reqd:
        reprstr = reprstr + str(r) + "\n"
    reprstr = reprstr + "SIGS:\n"
    for s in self.sigs:
        reprstr = reprstr + str(s) + "\n"
    reprstr = reprstr + "END\n"
    return reprstr
```

Getting rid of Tx7 B/c the outputs can exceed inputs, this is actually the responsibility of the miner (before 8:13)

Only TxBlock needs to worry about these indices actually being passes in the right order

9:05 – 9:45 → adding to is_valid in TxBlock

getLastTxIndex → function that looks back thru the blockchain and finds the last tx that includes this addr as one of the inputs

TxBlock.py - C:/Users/Russ/Python36/TxBlock.py (3.6.4)

File Edit Format Run Options Window Help

```python
        return False
        return True
    def is_valid(self):
        if not super(TxBlock, self).is_valid():
            return False
        spends={}
        for tx in self.data:
            if not tx.is_valid():
                return False
            for addr,amt,inx in tx.inputs:
                if not addr in spends:
                    spends[addr] = amt
                else:
                    spends[addr] = spends[addr] + amt
                if not inx = getLastTxIndex(addr) + 1:
                    return False
            for addr,amt in tx.outputs:
                if not addr in spends:
                    spends[addr] = - amt
                else:
                    spends[addr] = spends[addr] - amt
        for addr in spends:
            if self.previousBlock == None:
                if spends[addr] > 0:
                    return False
            if spends[addr] - getBalance(addr,self.previousBlock) > 0.00000001:
                return False
        total_in, total_out = self.count_totals()
        if total_out - total_in - reward > 0.000000000001:
            return False
        if not self.check_size():
            return False
        return True
```

10:30 → Creating function getLastTxIndex under balance

Vid 78 – Approaches to Replay Attacks

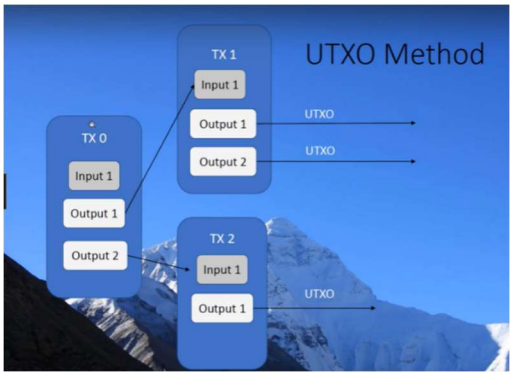
How do Blockchains avoid these duplicate transactions?

- Account Balance method (ETH)
  - Involves additional transaction index

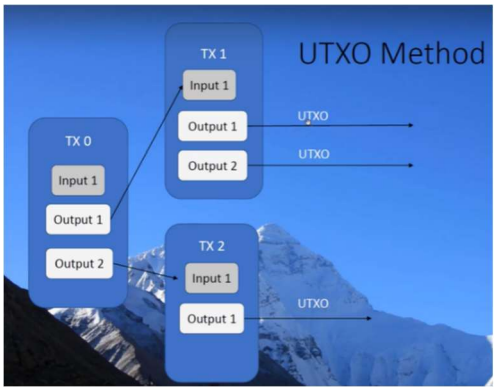
```
Tx.add_inputs(pu_send, amt, tx_num)
```

- UTXO – Unspent Transaction Output method (BTC)
  - Rather than keeping track of the balances of specific addresses it keeps track of unspent transaction outputs (which means they know the amt in them and know what public key that controls them)

1:00 – 3:30 → explanation of the UTXO method



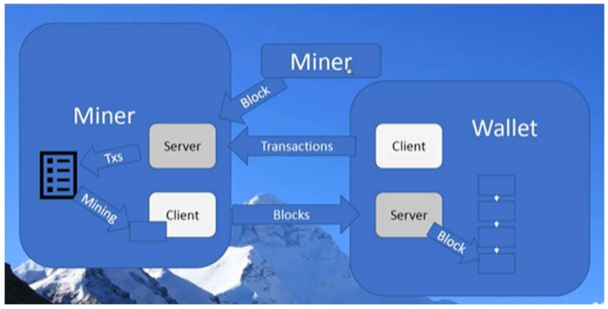
3:30 – 4:20 → UTXO w/ 3 transactions



4:29 – End → Advantages and Disadvantages

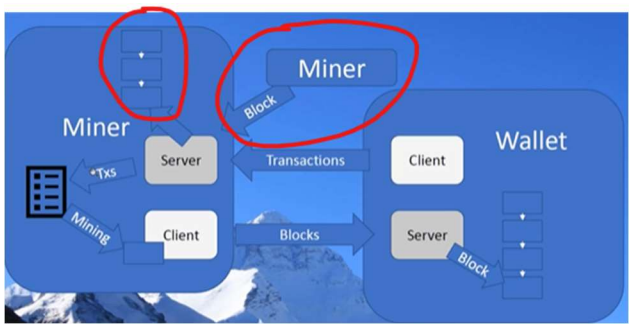
Vid 79 – Assignment 6: Multiple Miners

- Task: keep the miner blockchain up to date
- Must consider the blocks that are mined by other miners
    - Need to have a way for those blocks to also get into the blockchain that is maintained inside our miner



Many miners can send blocks to our miner and we need to take those blocks and add them to our internal blockchain

- Remove from the tx_list → any txs that were included in this miner
- Remove duplicate indices that I see are already in other txs



Task 1 → we're going to receive some blocks from another miner and were going to have to remove the transactions in this block from our transaction list so we don't duplicate

1:30 → Starting Assignment creating the test case