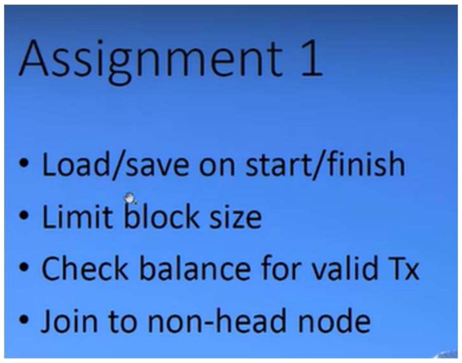
Vid 67 – Potpourri



Assignment 1

- Load/save on start/finish
- Limit block size
- Check balance for valid Tx
- Join to non-head node

Load and save → both transaction lists, blockchains, and private and public keys as we start and finish our wallet and miner.

Limit the block size → lots of practical reasons to limit the number of transactions and specifically the number of bytes that our block can contain

Check balance for valid Tx → Ensure the Balance is available when we make a transaction, before we declare a transaction is valid (eg if your going to send me 5 coins I need to make sure you have at least 5 coins

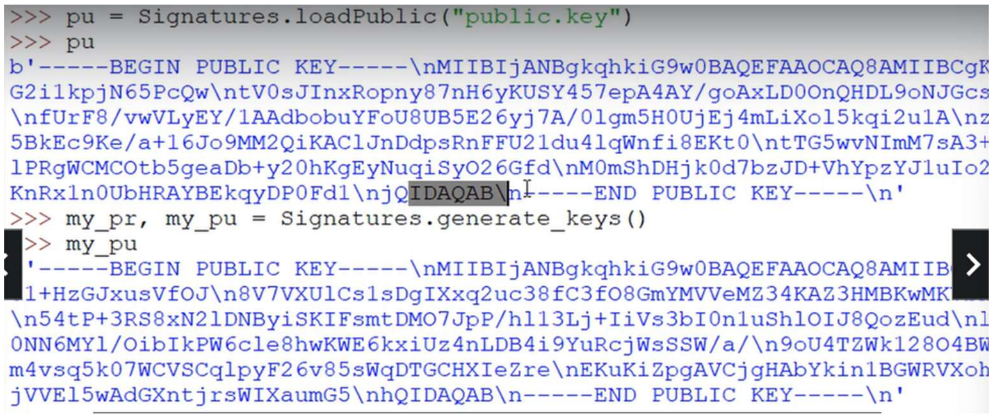
Join to non-head node → This can happen when you have blocks circulating thru your network at different rates. We don't want to give a preference to those blocks that are better connected. (1:45-

Vid 68 – Assignment 1: load and save states

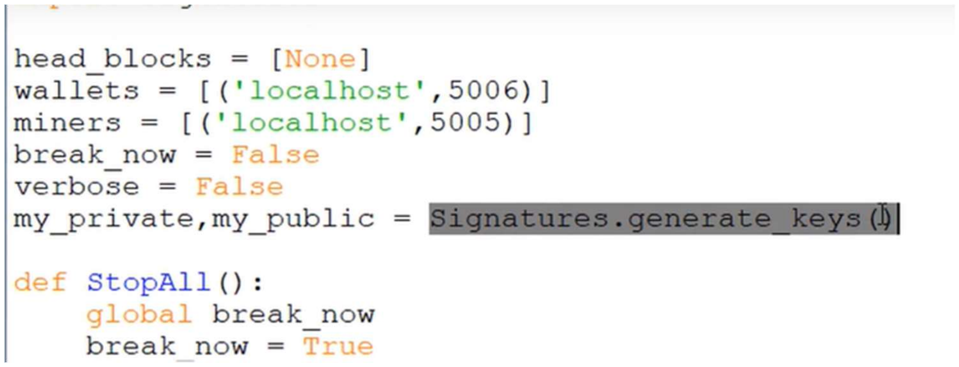
EZCoin is going to be what we expect users to load and call when they want to fire up our coin

Going to create a miner and a wallet

Don't look at the beginning and end of the public keys to see if they're different → discrepancies are more likely to be in the middle!!!!

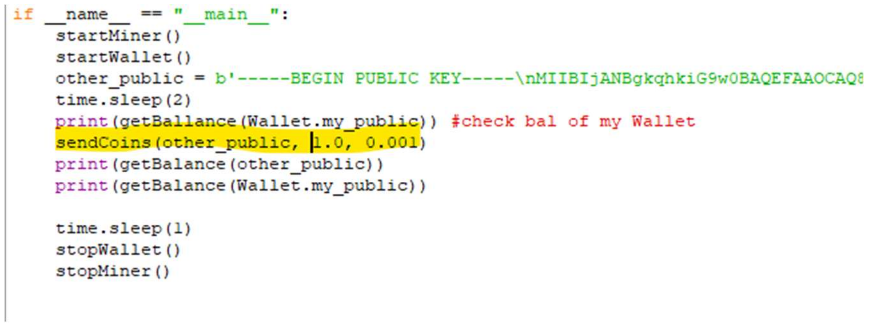


```
>>> pu = Signatures.loadPublic("public.key")
>>> pu
b'-----BEGIN PUBLIC KEY-----\nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgK
G2i1kpjN65PcQw\ntV0sJInxRopny87nH6yKUSY457epA4AY/goAxLD0OnQHDL9oNJGcs
\nfUrF8/vwVLyEY/1AAdbobuYFoU8UB5E26yj7A/01gm5H0UjEj4mLiXol5kqi2u1A\nz
5BkEc9Ke/a+16Jo9MM2QiKAClJnDdpsRnFFU21du41qWnfi8EKt0\ntTG5wvNImM7sA3+
lPRgWCMCOtb5geaDb+y20hKgEyNuqiSyO26Gfd\nM0mShDHjk0d7bzJD+VhYpzYJ1uIo2
KnRx1n0UbHRAYBEkqyDP0Fd1\njQIDAQAB\n-----END PUBLIC KEY-----\n'
>>> my_pr, my_pu = Signatures.generate_keys()
>>> my_pu
'-----BEGIN PUBLIC KEY-----\nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIB
1+HzGJxusVfOJ\n8V7VXUlCs1sDgIXxq2uc38fC3fO8GmYMVVeMZ34KAZ3HMBKwMK
\n54tP+3RS8xN2lDNByiSKIFsmtDMO7JpP/hl13Lj+IiVs3bI0n1uShlOIJ8QozEud\nl
0NN6MYl/OibIkPW6cle8hwKWE6kxiUz4nLDB4i9YuRcjWsSSW/a/\n9oU4TZWk128O4BW
m4vsq5k07WCVSCqlpyF26v85sWqDTGCHXIeZre\nEKuKiZpgAVCjgHAbYkin1BGWRVXoh
jVVEl5wAdGXntjrsWIXaumG5\nhQIDAQAB\n-----END PUBLIC KEY-----\n'
```



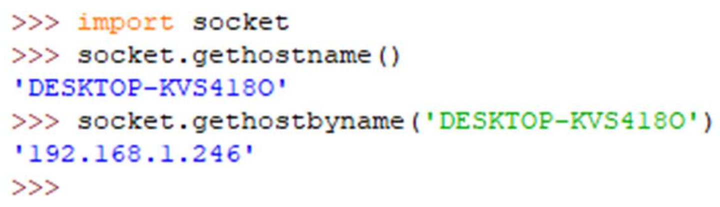
```
head_blocks = [None]
wallets = [('localhost',5006)]
miners = [('localhost',5005)]
break_now = False
verbose = False
my_private,my_public = Signatures.generate_keys()

def StopAll():
    global break_now
    break_now = True
```

Problem: we're generating new keys every time, but we want our wallet to load keys if it can



```
if __name__ == "__main__":
    startMiner()
    startWallet()
    other_public = b'-----BEGIN PUBLIC KEY-----\nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8
    time.sleep(2)
    print(getBallance(Wallet.my_public)) #check bal of my Wallet
    sendCoins(other_public, 1.0, 0.001)
    print(getBalance(other_public))
    print(getBalance(Wallet.my_public))

    time.sleep(1)
    stopWallet()
    stopMiner()
```

What you want your user to be able to do is say, "you wallet worry about what my_public and my_private keys are. All I want to do is give you a **place to send it, tell you how much, and maybe the tx fee that I'm going to offer to entice the other miners to do it**

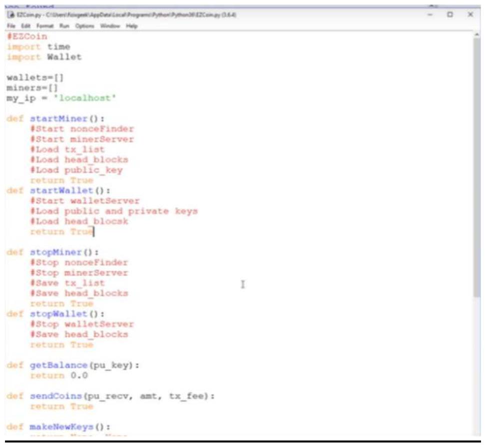


```
>>> import socket
>>> socket.gethostname()
'DESKTOP-KVS418O'
>>> socket.gethostbyname('DESKTOP-KVS418O')
'192.168.1.246'
>>>
```

- This tells me the name of this computer
- Also the local IP address

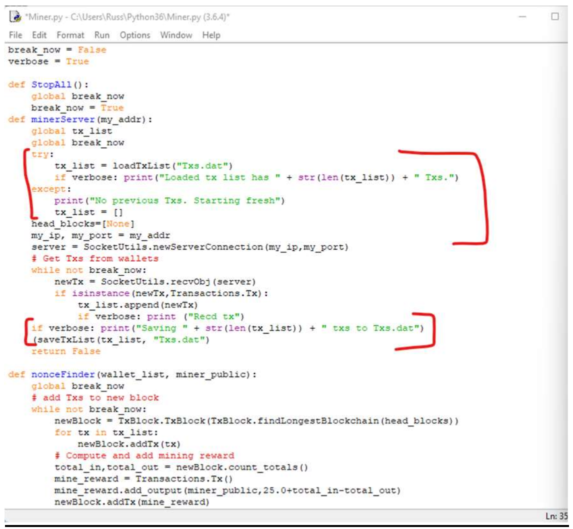
Vid 69 – Assignment 1: Solution

TODO's for the assignment

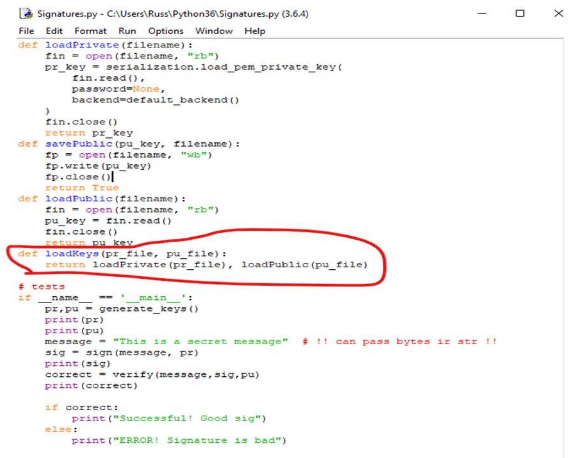


```
#EZCoin
import time
import Wallet

wallets=[]
miners=[]
my_ip = 'localhost'

def startMiner():
    #Start nonceFinder
    #Start minerServer
    #Load tx_list
    #Load head_blocks
    #Load public_key
    return True
def startWallet():
    #Start walletServer
    #Load public and private keys
    #Load head_blocsk
    return True

def stopMiner():
    #Stop nonceFinder
    #Stop minerServer
    #Save tx_list
    #Save head_blocks
    return True
def stopWallet():
    #Stop walletServer
    #Save head_blocks
    return True

def getBalance(pu_key):
    return 0.0

def sendCoins(pu_recv, amt, tx_fee):
    return True

def makeNewKeys():
```

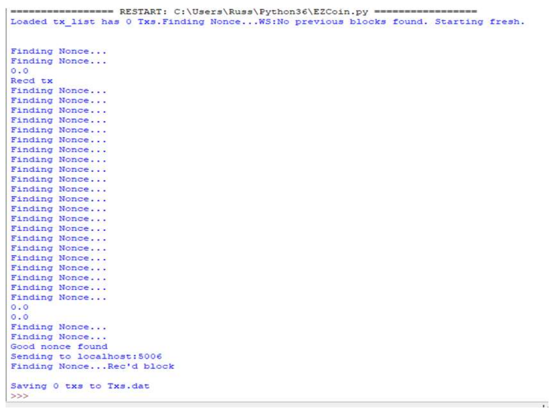
Changes to Miner


Miner.py - C:\Users\Russ\Python36\Miner.py (3.6.4)
File Edit Format Run Options Window Help
Ln: 35

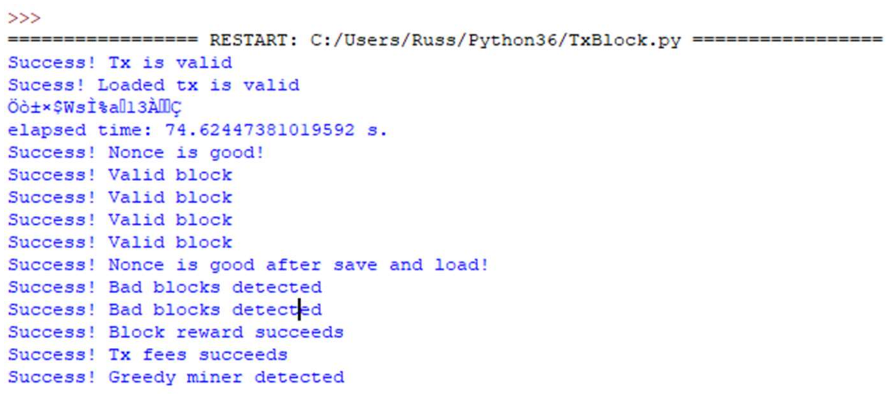
```python
break_now = False
verbose = True

def StopAll():
    global break_now
    break_now = True
def minerServer(my_addr):
    global tx_list
    global break_now
    try:
        tx_list = loadTxList("Txs.dat")
        if verbose: print("Loaded tx list has " + str(len(tx_list)) + " Txs.")
    except:
        print("No previous Txs. Starting fresh")
        tx_list = []
    head_blocks=[None]
    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):
            tx_list.append(newTx)
            if verbose: print ("Recd tx")
    if verbose: print("Saving " + str(len(tx_list)) + " txs to Txs.dat")
    (saveTxList(tx_list, "Txs.dat")
    return False

def nonceFinder(wallet_list, miner_public):
    global break_now
    # add Txs to new block
    while not break_now:
        newBlock = TxBlock.TxBlock(TxBlock.findLongestBlockchain(head_blocks))
        for tx in tx_list:
            newBlock.addTx(tx)
        # Compute and add mining reward
        total_in,total_out = newBlock.count_totals()
        mine_reward = Transactions.Tx()
        mine_reward.add_output(miner_public,25.0+total_in-total_out)
        newBlock.addTx(mine_reward)
```


Signatures.py - C:\Users\Russ\Python36\Signatures.py (3.6.4)
File Edit Format Run Options Window Help

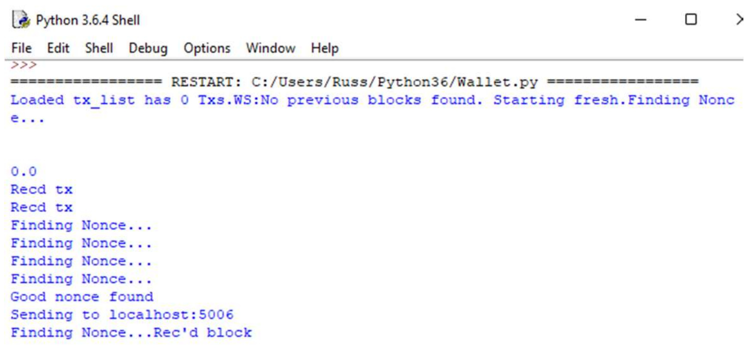
```python
def loadPrivate(filename):
    fin = open(filename, "rb")
    pr_key = serialization.load_pem_private_key(
        fin.read(),
        password=None,
        backend=default_backend()
    )
    fin.close()
    return pr_key
def savePublic(pu_key, filename):
    fp = open(filename, "wb")
    fp.write(pu_key)
    fp.close()
    return True
def loadPublic(filename):
    fin = open(filename, "rb")
    pu_key = fin.read()
    fin.close()
    return pu_key
def loadKeys(pr_file, pu_file):
    return loadPrivate(pr_file), loadPublic(pu_file)

# tests
if __name__ == '__main__':
    pr,pu = generate_keys()
    print(pr)
    print(pu)
    message = "This is a secret message"  # !! can pass bytes ir str !!
    sig = sign(message, pr)
    print(sig)
    correct = verify(message,sig,pu)
    print(correct)

    if correct:
        print("Successful! Good sig")
    else:
        print("ERROR! Signature is bad")
```



```
================= RESTART: C:\Users\Russ\Python36\EZCoin.py =================
Loaded tx_list has 0 Txs.Finding Nonce...WS:No previous blocks found. Starting fresh.


Finding Nonce...
Finding Nonce...
0.0
Recd tx
Finding Nonce...
Finding Nonce...
Finding Nonce...
Finding Nonce...
Finding Nonce...
Finding Nonce...
Finding Nonce...
Finding Nonce...
Finding Nonce...
Finding Nonce...
Finding Nonce...
Finding Nonce...
Finding Nonce...
Finding Nonce...
Finding Nonce...
Finding Nonce...
Finding Nonce...
Finding Nonce...
Finding Nonce...
Finding Nonce...
Finding Nonce...
Finding Nonce...
0.0
0.0
Finding Nonce...
Finding Nonce...
Good nonce found
Sending to localhost:5006
Finding Nonce...Rec'd block

Saving 0 txs to Txs.dat
>>>
```



```
>>>
================= RESTART: C:/Users/Russ/Python36/TxBlock.py =================
Success! Tx is valid
Sucess! Loaded tx is valid
Öò±×$WsÌ%a13ÀÇ
elapsed time: 74.62447381019592 s.
Success! Nonce is good!
Success! Valid block
Success! Valid block
Success! Valid block
Success! Valid block
Success! Nonce is good after save and load!
Success! Bad blocks detected
Success! Bad blocks detected
Success! Block reward succeeds
Success! Tx fees succeeds
Success! Greedy miner detected
```



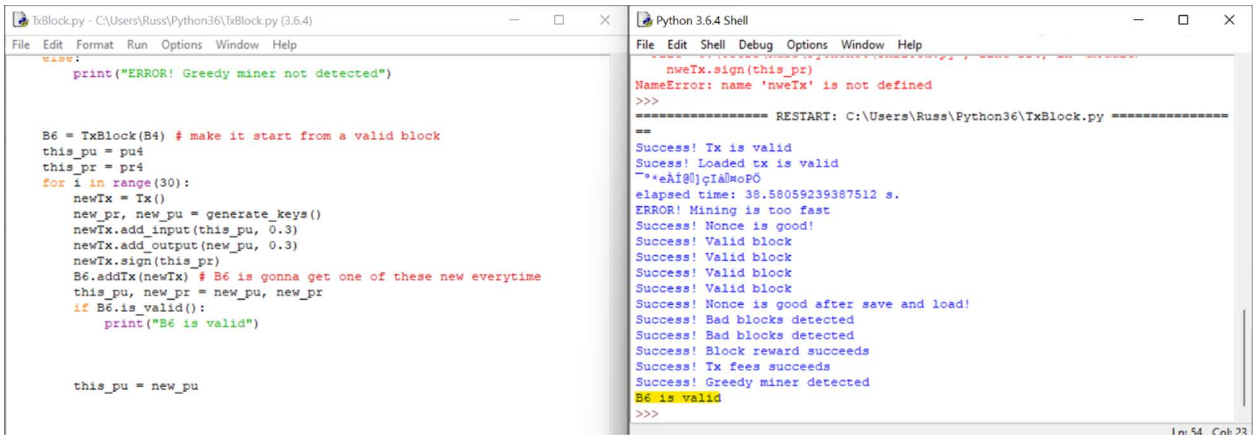
```
>>>
================= RESTART: C:/Users/Russ/Python36/Wallet.py =================
Loaded tx_list has 0 Txs.WS:No previous blocks found. Starting fresh.Finding Nonc
e...


0.0
Recd tx
Recd tx
Finding Nonce...
Finding Nonce...
Finding Nonce...
Finding Nonce...
Good nonce found
Sending to localhost:5006
Finding Nonce...Rec'd block
Finding Nonce...
Finding Nonce...
Finding Nonce...
Finding Nonce...
0.0
Error! Wrong balance for pu1
Error! Wrong balance for pu2
Error! Wrong balance for pu3
Saving 0 txs to Txs.dat
Exit successful.
>>>
```

Wrong Wallet balances

Vid 70 - Assignment 2: Limit block size

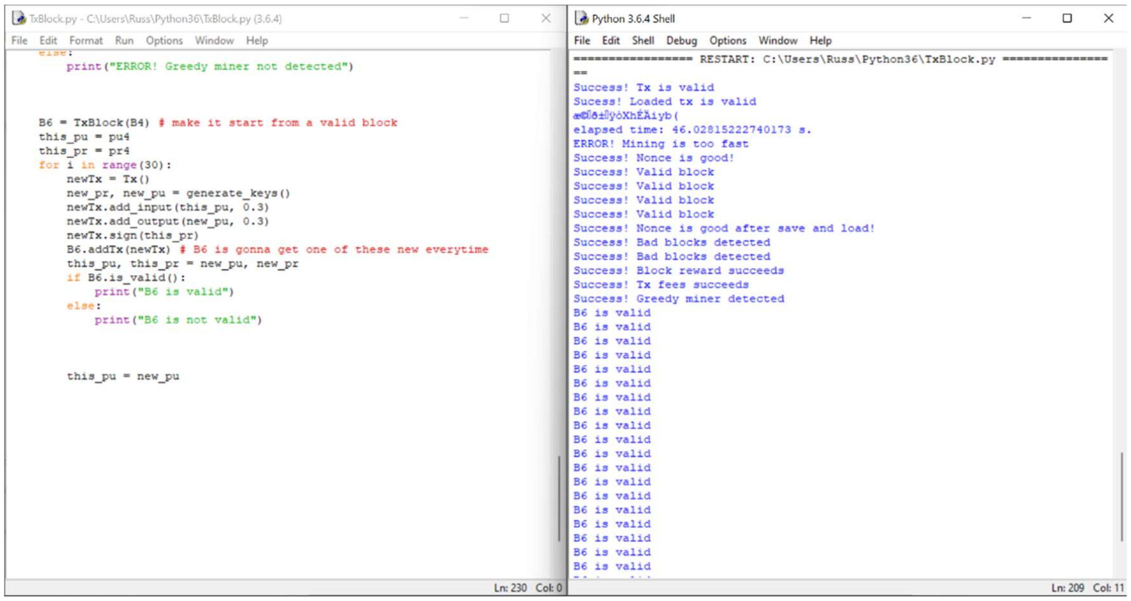
Changes to TxBlock

Need to create a big block of things fast → let pu4 send a bunch of transactions to a bunch of random addresses.


TxBlock.py - C:\Users\Russ\Python36\TxBlock.py (3.6.4)
File Edit Format Run Options Window Help
Python 3.6.4 Shell
File Edit Shell Debug Options Window Help
Ln: 54 Col: 23

```python
    else:
        print("ERROR! Greedy miner not detected")


B6 = TxBlock(B4) # make it start from a valid block
this_pu = pu4
this_pr = pr4
for i in range(30):
    newTx = Tx()
    new_pr, new_pu = generate_keys()
    newTx.add_input(this_pu, 0.3)
    newTx.add_output(new_pu, 0.3)
    newTx.sign(this_pr)
    B6.addTx(newTx) # B6 is gonna get one of these new everytime
    this_pu, new_pr = new_pu, new_pr
    if B6.is_valid():
        print("B6 is valid")



    this_pu = new_pu
```

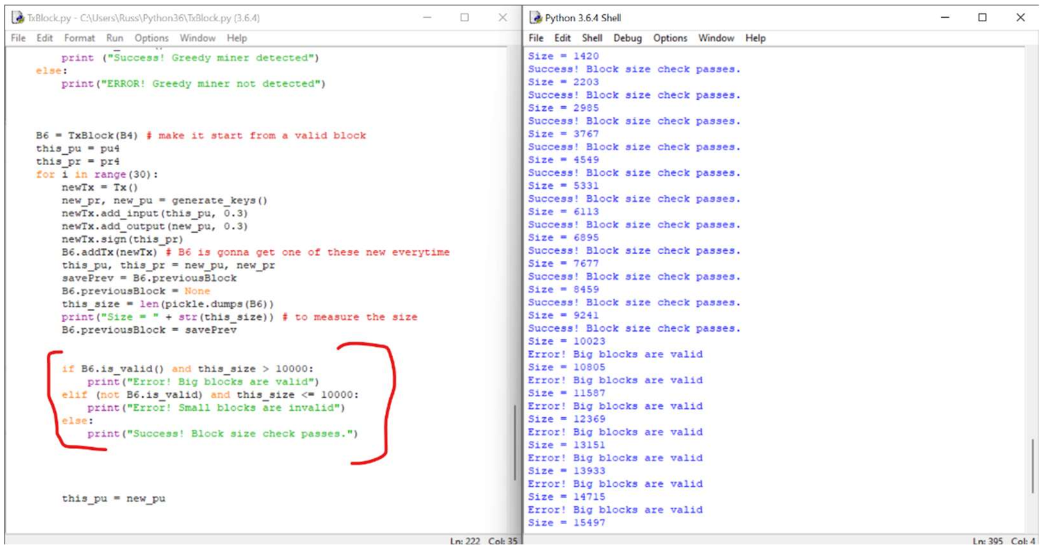
```
    nweTx.sign(this_pr)
NameError: name 'nweTx' is not defined
>>>
================= RESTART: C:\Users\Russ\Python36\TxBlock.py ===============
==
Success! Tx is valid
Sucess! Loaded tx is valid
¯°*eÀÍ@]çIà¤oPŐ
elapsed time: 38.58059239387512 s.
ERROR! Mining is too fast
Success! Nonce is good!
Success! Valid block
Success! Valid block
Success! Valid block
Success! Valid block
Success! Nonce is good after save and load!
Success! Bad blocks detected
Success! Bad blocks detected
Success! Block reward succeeds
Success! Tx fees succeeds
Success! Greedy miner detected
B6 is valid
>>>
```


TxBlock.py - C:\Users\Russ\Python36\TxBlock.py (3.6.4)
File Edit Format Run Options Window Help
Ln: 230 Col: 0
Python 3.6.4 Shell
File Edit Shell Debug Options Window Help
Ln: 209 Col: 11

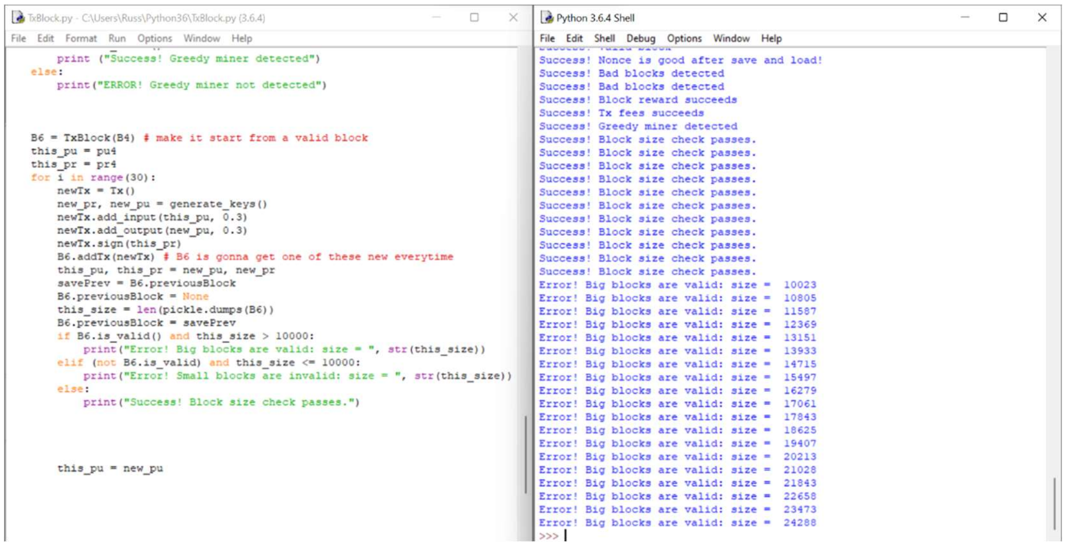
```python
    else:
        print("ERROR! Greedy miner not detected")


B6 = TxBlock(B4) # make it start from a valid block
this_pu = pu4
this_pr = pr4
for i in range(30):
    newTx = Tx()
    new_pr, new_pu = generate_keys()
    newTx.add_input(this_pu, 0.3)
    newTx.add_output(new_pu, 0.3)
    newTx.sign(this_pr)
    B6.addTx(newTx) # B6 is gonna get one of these new everytime
    this_pu, this_pr = new_pu, new_pr
    if B6.is_valid():
        print("B6 is valid")
    else:
        print("B6 is not valid")



    this_pu = new_pu
```

```
================= RESTART: C:\Users\Russ\Python36\TxBlock.py ===============
==
Success! Tx is valid
Sucess! Loaded tx is valid
æ©ð±ÿòXhÉÄiyb(
elapsed time: 46.02815222740173 s.
ERROR! Mining is too fast
Success! Nonce is good!
Success! Valid block
Success! Valid block
Success! Valid block
Success! Valid block
Success! Nonce is good after save and load!
Success! Bad blocks detected
Success! Bad blocks detected
Success! Block reward succeeds
Success! Tx fees succeeds
Success! Greedy miner detected
B6 is valid
B6 is valid
B6 is valid
B6 is valid
B6 is valid
B6 is valid
B6 is valid
B6 is valid
B6 is valid
B6 is valid
B6 is valid
B6 is valid
B6 is valid
B6 is valid
B6 is valid
B6 is valid
B6 is valid
B6 is valid
B6 is valid
```

Then well restore it → So well pull it out of there and that's when we'll pickle it and dump it → **size increasing, once we get passed the 10,000 size threshold we see that the ERROR is passes.**


```
    print ("Success! Greedy miner detected")
else:
    print("ERROR! Greedy miner not detected")



B6 = TxBlock(B4) # make it start from a valid block
this_pu = pu4
this_pr = pr4
for i in range(30):
    newTx = Tx()
    new_pr, new_pu = generate_keys()
    newTx.add_input(this_pu, 0.3)
    newTx.add_output(new_pu, 0.3)
    newTx.sign(this_pr)
    B6.addTx(newTx) # B6 is gonna get one of these new everytime
    this_pu, this_pr = new_pu, new_pr
    savePrev = B6.previousBlock
    B6.previousBlock = None
    this_size = len(pickle.dumps(B6))
    print("Size = " + str(this_size)) # to measure the size
    B6.previousBlock = savePrev


    if B6.is_valid() and this_size > 10000:
        print("Error! Big blocks are valid")
    elif (not B6.is_valid) and this_size <= 10000:
        print("Error! Small blocks are invalid")
    else:
        print("Success! Block size check passes.")




    this_pu = new_pu
```

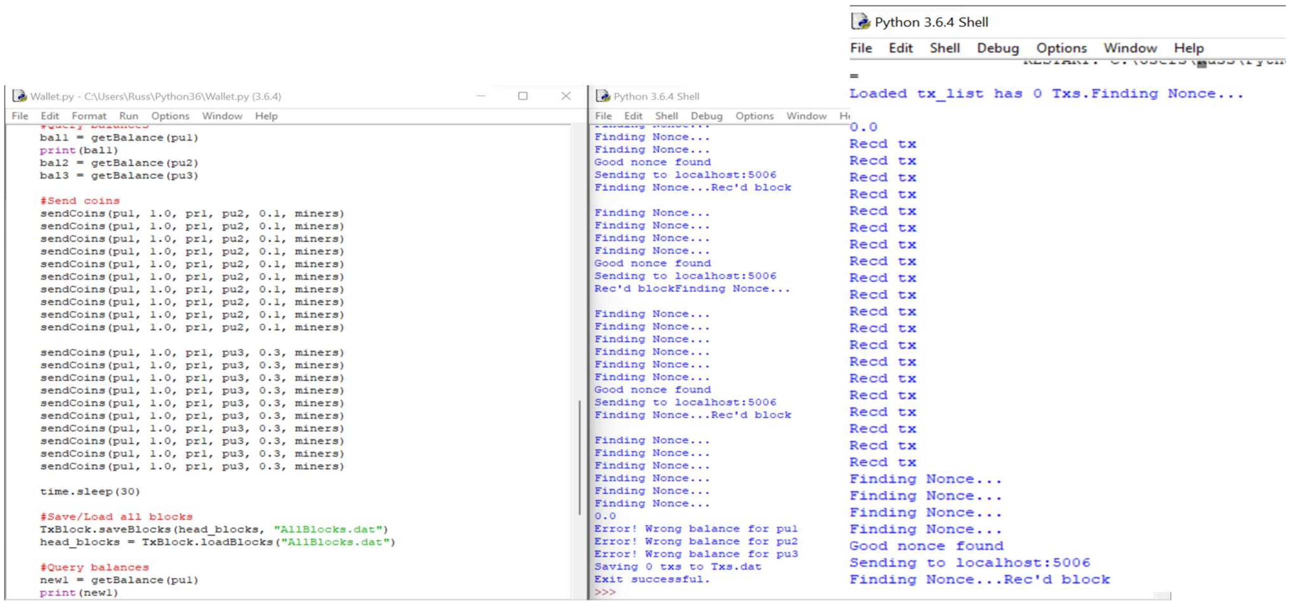
```
Size = 1420
Success! Block size check passes.
Size = 2203
Success! Block size check passes.
Size = 2985
Success! Block size check passes.
Size = 3767
Success! Block size check passes.
Size = 4549
Success! Block size check passes.
Size = 5331
Success! Block size check passes.
Size = 6113
Success! Block size check passes.
Size = 6895
Success! Block size check passes.
Size = 7677
Success! Block size check passes.
Size = 8459
Success! Block size check passes.
Size = 9241
Success! Block size check passes.
Size = 10023
Error! Big blocks are valid
Size = 10805
Error! Big blocks are valid
Size = 11587
Error! Big blocks are valid
Size = 12369
Error! Big blocks are valid
Size = 13151
Error! Big blocks are valid
Size = 13933
Error! Big blocks are valid
Size = 14715
Error! Big blocks are valid
Size = 15497
```


```
    print ("Success! Greedy miner detected")
else:
    print("ERROR! Greedy miner not detected")



B6 = TxBlock(B4) # make it start from a valid block
this_pu = pu4
this_pr = pr4
for i in range(30):
    newTx = Tx()
    new_pr, new_pu = generate_keys()
    newTx.add_input(this_pu, 0.3)
    newTx.add_output(new_pu, 0.3)
    newTx.sign(this_pr)
    B6.addTx(newTx) # B6 is gonna get one of these new everytime
    this_pu, this_pr = new_pu, new_pr
    savePrev = B6.previousBlock
    B6.previousBlock = None
    this_size = len(pickle.dumps(B6))
    B6.previousBlock = savePrev
    if B6.is_valid() and this_size > 10000:
        print("Error! Big blocks are valid: size = ", str(this_size))
    elif (not B6.is_valid) and this_size <= 10000:
        print("Error! Small blocks are invalid: size = ", str(this_size))
    else:
        print("Success! Block size check passes.")




    this_pu = new_pu
```

```
Success! Nonce is good after save and load!
Success! Bad blocks detected
Success! Bad blocks detected
Success! Block reward succeeds
Success! Tx fees succeeds
Success! Greedy miner detected
Success! Block size check passes.
Success! Block size check passes.
Success! Block size check passes.
Success! Block size check passes.
Success! Block size check passes.
Success! Block size check passes.
Success! Block size check passes.
Success! Block size check passes.
Success! Block size check passes.
Success! Block size check passes.
Success! Block size check passes.
Error! Big blocks are valid: size =  10023
Error! Big blocks are valid: size =  10805
Error! Big blocks are valid: size =  11587
Error! Big blocks are valid: size =  12369
Error! Big blocks are valid: size =  13151
Error! Big blocks are valid: size =  13933
Error! Big blocks are valid: size =  14715
Error! Big blocks are valid: size =  15497
Error! Big blocks are valid: size =  16279
Error! Big blocks are valid: size =  17061
Error! Big blocks are valid: size =  17843
Error! Big blocks are valid: size =  18625
Error! Big blocks are valid: size =  19407
Error! Big blocks are valid: size =  20213
Error! Big blocks are valid: size =  21028
Error! Big blocks are valid: size =  21843
Error! Big blocks are valid: size =  22658
Error! Big blocks are valid: size =  23473
Error! Big blocks are valid: size =  24288
>>>
```

Changes to Miner (13:00)

Make sure that our miner isn't creating blocks that are too big


```
    bal1 = getBalance(pu1)
    print(bal1)
    bal2 = getBalance(pu2)
    bal3 = getBalance(pu3)

    #Send coins
    sendCoins(pu1, 1.0, pr1, pu2, 0.1, miners)
    sendCoins(pu1, 1.0, pr1, pu2, 0.1, miners)
    sendCoins(pu1, 1.0, pr1, pu2, 0.1, miners)
    sendCoins(pu1, 1.0, pr1, pu2, 0.1, miners)
    sendCoins(pu1, 1.0, pr1, pu2, 0.1, miners)
    sendCoins(pu1, 1.0, pr1, pu2, 0.1, miners)
    sendCoins(pu1, 1.0, pr1, pu2, 0.1, miners)
    sendCoins(pu1, 1.0, pr1, pu2, 0.1, miners)
    sendCoins(pu1, 1.0, pr1, pu2, 0.1, miners)
    sendCoins(pu1, 1.0, pr1, pu2, 0.1, miners)

    sendCoins(pu1, 1.0, pr1, pu3, 0.3, miners)
    sendCoins(pu1, 1.0, pr1, pu3, 0.3, miners)
    sendCoins(pu1, 1.0, pr1, pu3, 0.3, miners)
    sendCoins(pu1, 1.0, pr1, pu3, 0.3, miners)
    sendCoins(pu1, 1.0, pr1, pu3, 0.3, miners)
    sendCoins(pu1, 1.0, pr1, pu3, 0.3, miners)
    sendCoins(pu1, 1.0, pr1, pu3, 0.3, miners)
    sendCoins(pu1, 1.0, pr1, pu3, 0.3, miners)
    sendCoins(pu1, 1.0, pr1, pu3, 0.3, miners)
    sendCoins(pu1, 1.0, pr1, pu3, 0.3, miners)

    time.sleep(30)

    #Save/Load all blocks
    TxBlock.saveBlocks(head_blocks, "AllBlocks.dat")
    head_blocks = TxBlock.loadBlocks("AllBlocks.dat")

    #Query balances
    new1 = getBalance(pu1)
    print(new1)
```

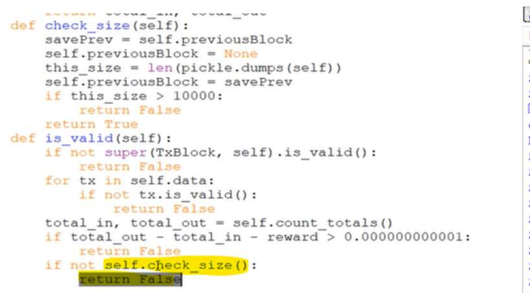
```
Finding Nonce...
Finding Nonce...
Good nonce found
Sending to localhost:5006
Finding Nonce...Rec'd block

Finding Nonce...
Finding Nonce...
Finding Nonce...
Finding Nonce...
Good nonce found
Sending to localhost:5006
Rec'd blockFinding Nonce...

Finding Nonce...
Finding Nonce...
Finding Nonce...
Finding Nonce...
Finding Nonce...
Finding Nonce...
Good nonce found
Sending to localhost:5006
Finding Nonce...Rec'd block

Finding Nonce...
Finding Nonce...
Finding Nonce...
Finding Nonce...
Finding Nonce...
Finding Nonce...
0.0
Error! Wrong balance for pu1
Error! Wrong balance for pu2
Error! Wrong balance for pu3
Saving 0 txs to Txs.dat
Exit successful.
>>>
```

```
=
Loaded tx_list has 0 Txs.Finding Nonce...
0.0
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Finding Nonce...
Finding Nonce...
Finding Nonce...
Finding Nonce...
Good nonce found
Sending to localhost:5006
Finding Nonce...Rec'd block
```

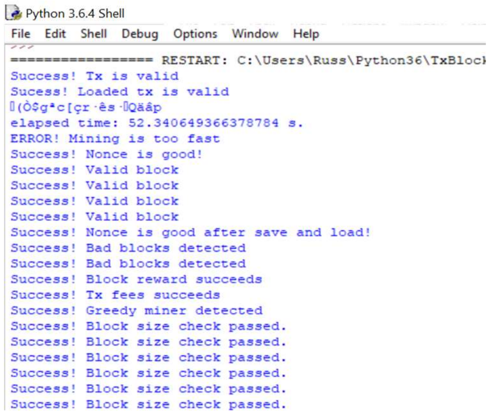
Vid 71 – Assignment 2: Solution-

If check size comes back true, that means it passed → then not self.check_size is going to be False. So we won't return false

Check_size tells us that the size is ok, if not the size is ok then return false

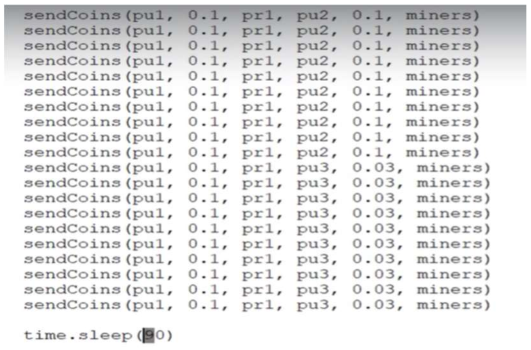


```
def check_size(self):
    savePrev = self.previousBlock
    self.previousBlock = None
    this_size = len(pickle.dumps(self))
    self.previousBlock = savePrev
    if this_size > 10000:
        return False
    return True
def is_valid(self):
    if not super(TxBlock, self).is_valid():
        return False
    for tx in self.data:
        if not tx.is_valid():
            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
```

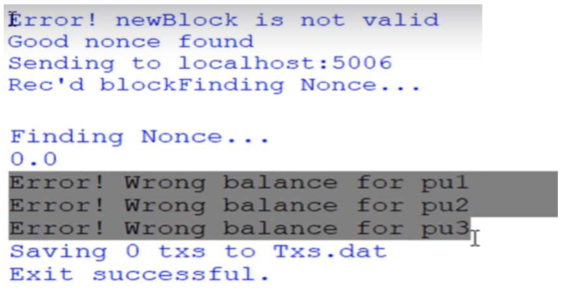


```
================= RESTART: C:\Users\Russ\Python36\TxBlock
Success! Tx is valid
Sucess! Loaded tx is valid
(Ò$gªc[çr·ês·Qäâp
elapsed time: 52.340649366378784 s.
ERROR! Mining is too fast
Success! Nonce is good!
Success! Valid block
Success! Valid block
Success! Valid block
Success! Valid block
Success! Nonce is good after save and load!
Success! Bad blocks detected
Success! Bad blocks detected
Success! Block reward succeeds
Success! Tx fees succeeds
Success! Greedy miner detected
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
```

Expected to fail → it going to be too big of a block, and the miner won't be "paying attention" its just throwing all of the transactions it sees.



```
    sendCoins(pu1, 0.1, pr1, pu2, 0.1, miners)
    sendCoins(pu1, 0.1, pr1, pu2, 0.1, miners)
    sendCoins(pu1, 0.1, pr1, pu2, 0.1, miners)
    sendCoins(pu1, 0.1, pr1, pu2, 0.1, miners)
    sendCoins(pu1, 0.1, pr1, pu2, 0.1, miners)
    sendCoins(pu1, 0.1, pr1, pu2, 0.1, miners)
    sendCoins(pu1, 0.1, pr1, pu2, 0.1, miners)
    sendCoins(pu1, 0.1, pr1, pu2, 0.1, miners)
    sendCoins(pu1, 0.1, pr1, pu2, 0.1, miners)
    sendCoins(pu1, 0.1, pr1, pu2, 0.1, miners)
    sendCoins(pu1, 0.1, pr1, pu3, 0.03, miners)
    sendCoins(pu1, 0.1, pr1, pu3, 0.03, miners)
    sendCoins(pu1, 0.1, pr1, pu3, 0.03, miners)
    sendCoins(pu1, 0.1, pr1, pu3, 0.03, miners)
    sendCoins(pu1, 0.1, pr1, pu3, 0.03, miners)
    sendCoins(pu1, 0.1, pr1, pu3, 0.03, miners)
    sendCoins(pu1, 0.1, pr1, pu3, 0.03, miners)
    sendCoins(pu1, 0.1, pr1, pu3, 0.03, miners)
    sendCoins(pu1, 0.1, pr1, pu3, 0.03, miners)
    sendCoins(pu1, 0.1, pr1, pu3, 0.03, miners)

    time.sleep(90)
```



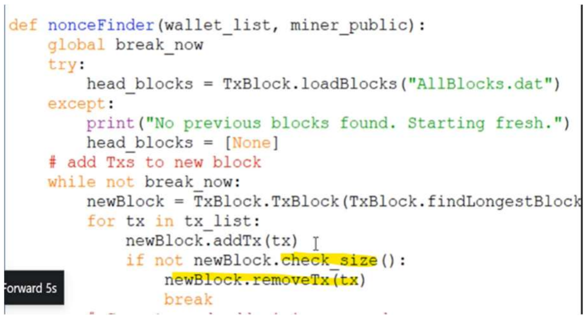
```
Error! newBlock is not valid
Good nonce found
Sending to localhost:5006
Rec'd blockFinding Nonce...

Finding Nonce...
0.0
Error! Wrong balance for pu1
Error! Wrong balance for pu2
Error! Wrong balance for pu3
Saving 0 txs to Txs.dat
Exit successful.
```

Because the block wasn't valid, it didn't get added to the blockchain that the wallet is keeping → hence why we get the wrong balances

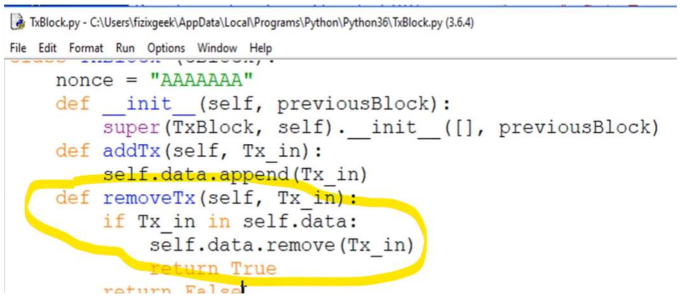
How to Fix this

Right now, Miner is recklessly adding all of the transactions that its sees (takes the whole tx_list and says add list for every single one.

- Adding to Miner: check_size and then also remove the last transaction (bc we added one that got too big)
- TxBlock: adding removeTx functionality



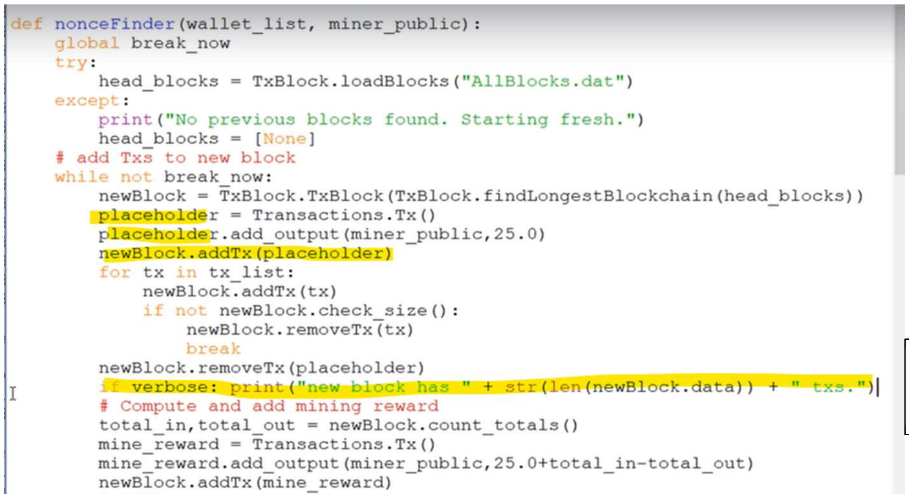
```
def nonceFinder(wallet_list, miner_public):
    global break_now
    try:
        head_blocks = TxBlock.loadBlocks("AllBlocks.dat")
    except:
        print("No previous blocks found. Starting fresh.")
        head_blocks = [None]
    # add Txs to new block
    while not break_now:
        newBlock = TxBlock.TxBlock(TxBlock.findLongestBlock
        for tx in tx_list:
            newBlock.addTx(tx)
            if not newBlock.check_size():
                newBlock.removeTx(tx)
                break
```



TxBlock.py - C:\Users\fizixgeek\AppData\Local\Programs\Python\Python36\TxBlock.py (3.6.4)
File Edit Format Run Options Window Help

```
    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
```

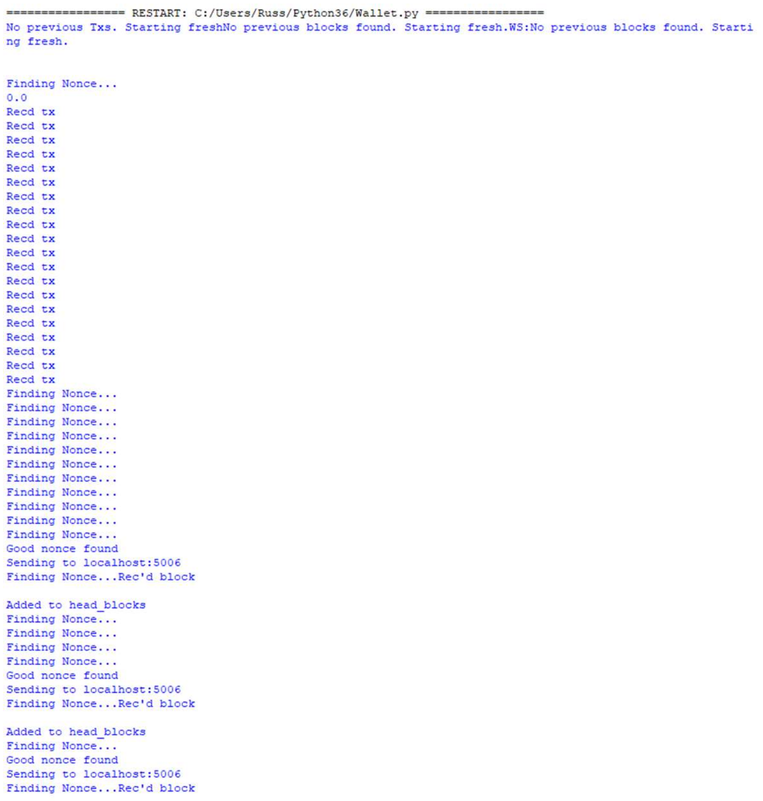
➔ Mine_reward gets added after we've done the size check and that can make the tx too large



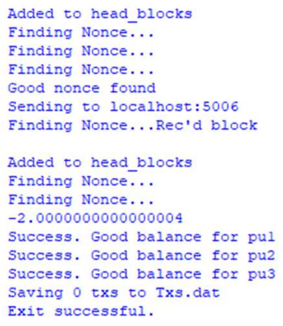
```
def nonceFinder(wallet_list, miner_public):
    global break_now
    try:
        head_blocks = TxBlock.loadBlocks("AllBlocks.dat")
    except:
        print("No previous blocks found. Starting fresh.")
        head_blocks = [None]
    # add Txs to new block
    while not break_now:
        newBlock = TxBlock.TxBlock(TxBlock.findLongestBlockchain(head_blocks))
        placeholder = Transactions.Tx()
        placeholder.add_output(miner_public,25.0)
        newBlock.addTx(placeholder)
        for tx in tx_list:
            newBlock.addTx(tx)
            if not newBlock.check_size():
                newBlock.removeTx(tx)
                break
        newBlock.removeTx(placeholder)
        if verbose: print("new block has " + str(len(newBlock.data)) + " txs.")
        # Compute and add mining reward
        total_in,total_out = newBlock.count_totals()
        mine_reward = Transactions.Tx()
        mine_reward.add_output(miner_public,25.0+total_in-total_out)
        newBlock.addTx(mine_reward)
```

Prints # of transactions so that we can watch them go by

Good Balances (matches prof – 9:48)



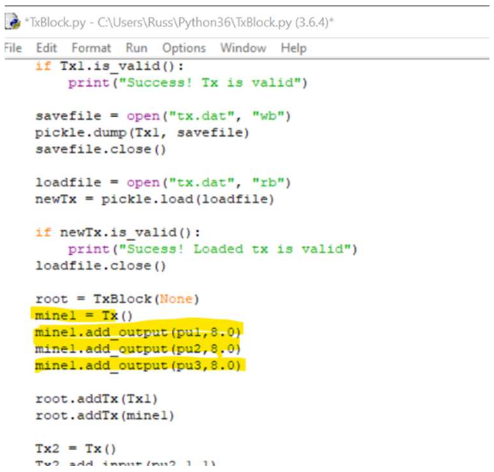
```
================= RESTART: C:/Users/Russ/Python36/Wallet.py =================
No previous Txs. Starting freshNo previous blocks found. Starting fresh.WS:No previous blocks found. Starti
ng fresh.


Finding Nonce...
0.0
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Finding Nonce...
Finding Nonce...
Finding Nonce...
Finding Nonce...
Finding Nonce...
Finding Nonce...
Finding Nonce...
Finding Nonce...
Finding Nonce...
Finding Nonce...
Finding Nonce...
Good nonce found
Sending to localhost:5006
Finding Nonce...Rec'd block

Added to head_blocks
Finding Nonce...
Finding Nonce...
Finding Nonce...
Finding Nonce...
Good nonce found
Sending to localhost:5006
Finding Nonce...Rec'd block

Added to head_blocks
Finding Nonce...
Good nonce found
Sending to localhost:5006
Finding Nonce...Rec'd block
```



```
Added to head_blocks
Finding Nonce...
Finding Nonce...
Finding Nonce...
Good nonce found
Sending to localhost:5006
Finding Nonce...Rec'd block

Added to head_blocks
Finding Nonce...
Finding Nonce...
-2.0000000000000004
Success. Good balance for pu1
Success. Good balance for pu2
Success. Good balance for pu3
Saving 0 txs to Txs.dat
Exit successful.
```

Vid 72 - Assignment 3: Checking Account Balances

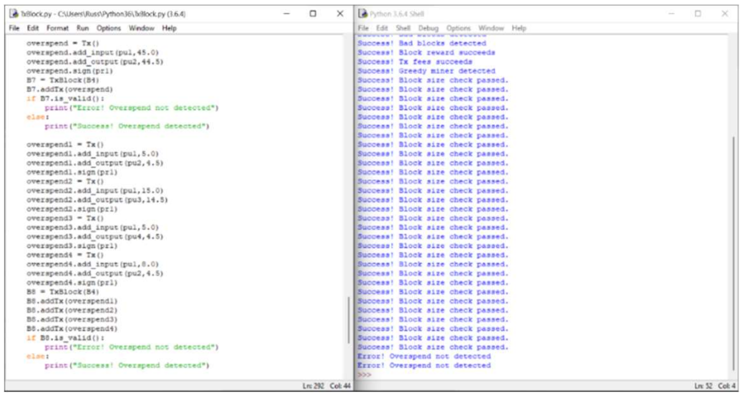
Task → Check the balances for every transaction. Transaction shouldn't be valid if the user doesn't have enough coins to spend. Can't send coins if you don't have any!!



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

```
    if Tx1.is_valid():
        print("Success! Tx is valid")

    savefile = open("tx.dat", "wb")
    pickle.dump(Tx1, savefile)
    savefile.close()

    loadfile = open("tx.dat", "rb")
    newTx = pickle.load(loadfile)

    if newTx.is_valid():
        print("Sucess! Loaded tx is valid")
    loadfile.close()

    root = TxBlock(None)
    mine1 = Tx()
    mine1.add_output(pu1,8.0)
    mine1.add_output(pu2,8.0)
    mine1.add_output(pu3,8.0)

    root.addTx(Tx1)
    root.addTx(mine1)

    Tx2 = Tx()
```

Overspend not detected



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

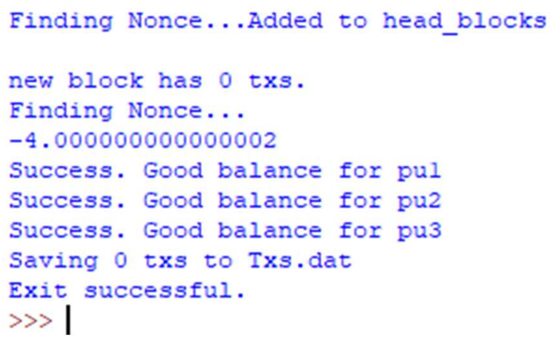
```
    overspend = Tx()
    overspend.add_input(pu1,45.0)
    overspend.add_output(pu2,44.5)
    overspend.sign(pr1)
    B7 = TxBlock(B4)
    B7.addTx(overspend)
    if B7.is_valid():
        print("Error! Overspend not detected")
    else:
        print("Success! Overspend detected")

    overspend1 = Tx()
    overspend1.add_input(pu1,5.0)
    overspend1.add_output(pu2,4.5)
    overspend1.sign(pr1)
    overspend2 = Tx()
    overspend2.add_input(pu1,15.0)
    overspend2.add_output(pu3,14.5)
    overspend2.sign(pr1)
    overspend3 = Tx()
    overspend3.add_input(pu1,5.0)
    overspend3.add_output(pu4,4.5)
    overspend3.sign(pr1)
    overspend4 = Tx()
    overspend4.add_input(pu1,8.0)
    overspend4.add_output(pu2,4.5)
    overspend4.sign(pr1)
    B8 = TxBlock(B4)
    B8.addTx(overspend1)
    B8.addTx(overspend2)
    B8.addTx(overspend3)
    B8.addTx(overspend4)
    if B8.is_valid():
        print("Error! Overspend not detected")
    else:
        print("Success! Overspend detected")
```

Python 3.6.4 Shell

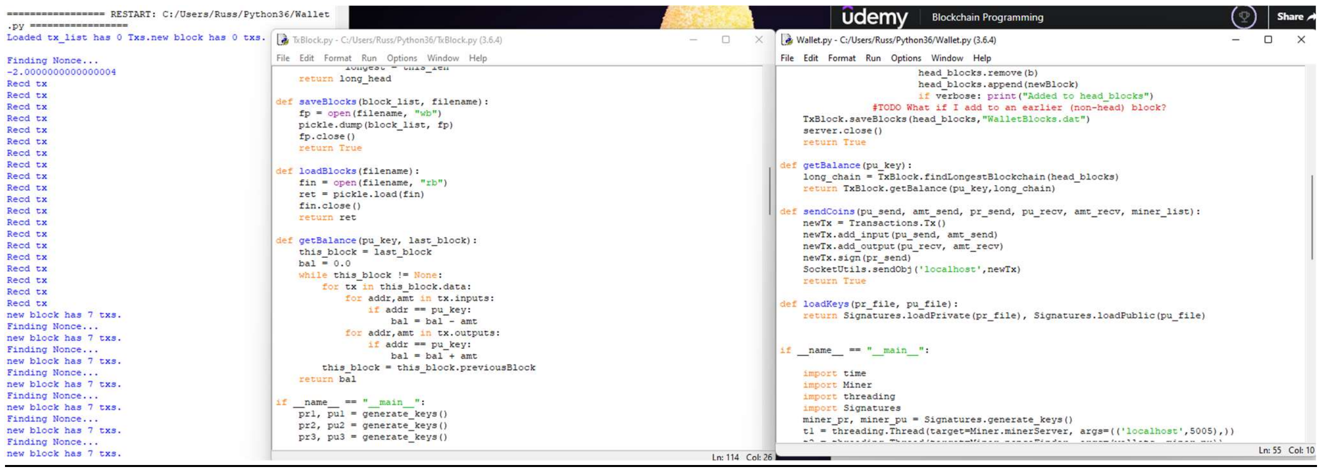
```
Success! Bad blocks detected
Success! Block reward succeeds
Success! Tx fees succeeds
Success! Greedy miner detected
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Error! Overspend not detected
Error! Overspend not detected
>>>
```

Vid 73 – Assignment 3: Solution

Transferred getBalance in wallet to TxBlock → Wallet still running well with good balances but, did not find many good nonces found



```
Finding Nonce...Added to head_blocks

new block has 0 txs.
Finding Nonce...
-4.000000000000002
Success. Good balance for pu1
Success. Good balance for pu2
Success. Good balance for pu3
Saving 0 txs to Txs.dat
Exit successful.
>>>
```



```
================= RESTART: C:/Users/Russ/Python36/Wallet
.py =================
Loaded tx_list has 0 Txs.new block has 0 txs.

Finding Nonce...
-2.0000000000000004
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
Recd tx
new block has 7 txs.
Finding Nonce...
new block has 7 txs.
Finding Nonce...
new block has 7 txs.
Finding Nonce...
new block has 7 txs.
Finding Nonce...
new block has 7 txs.
Finding Nonce...
new block has 7 txs.
Finding Nonce...
new block has 7 txs.
```

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

```python
    return long_head

def saveBlocks(block_list, filename):
    fp = open(filename, "wb")
    pickle.dump(block_list, fp)
    fp.close()
    return True

def loadBlocks(filename):
    fin = open(filename, "rb")
    ret = pickle.load(fin)
    fin.close()
    return ret

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 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

if __name__ == "__main__":
    pr1, pu1 = generate_keys()
    pr2, pu2 = generate_keys()
    pr3, pu3 = generate_keys()
```

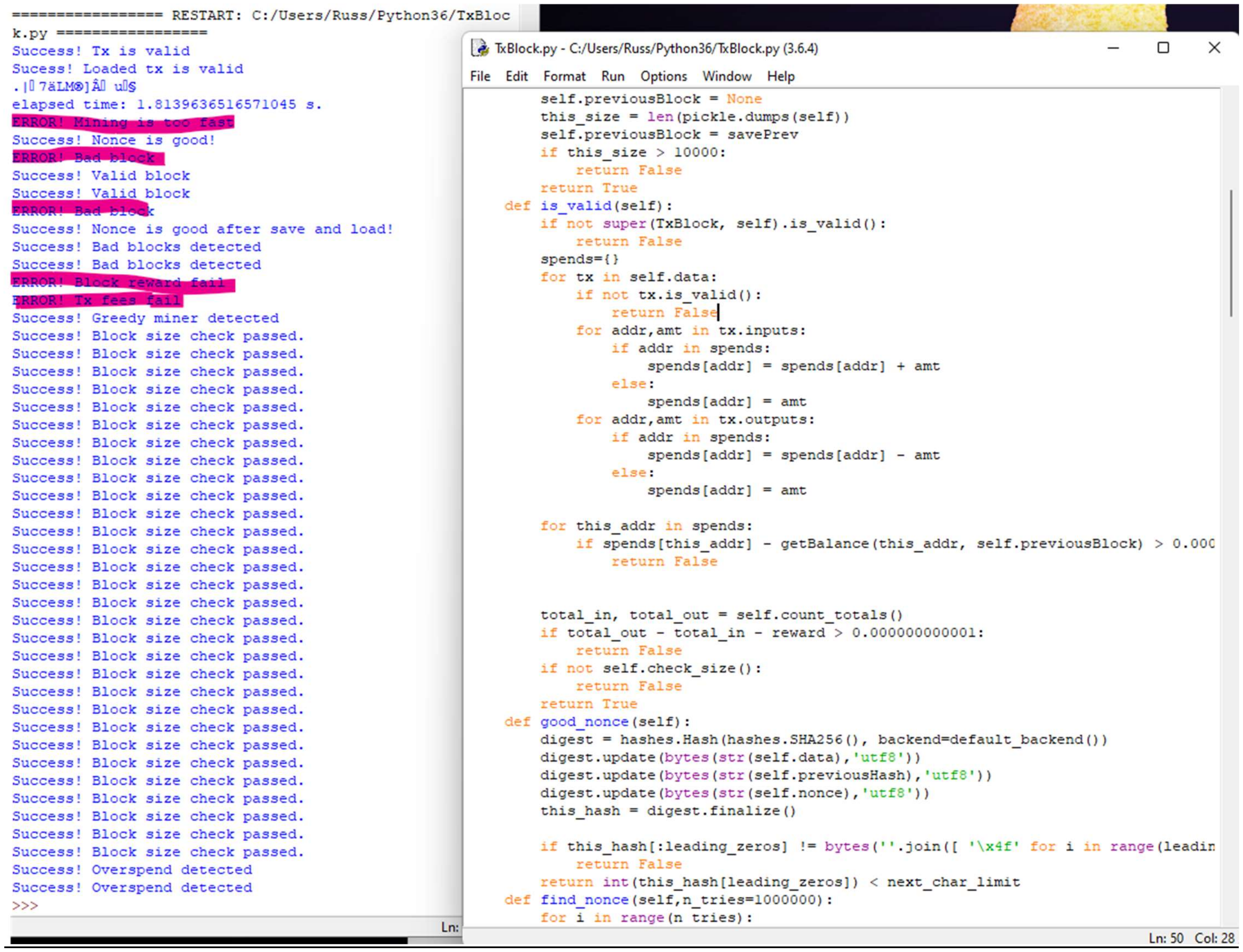
Ln: 114 Col: 26

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

```python
                        head_blocks.remove(b)
                        head_blocks.append(newBlock)
                        if verbose: print("Added to head_blocks")
                #TODO What if I add to an earlier (non-head) block?
    TxBlock.saveBlocks(head_blocks,"WalletBlocks.dat")
    server.close()
    return True

def getBalance(pu_key):
    long_chain = TxBlock.findLongestBlockchain(head_blocks)
    return TxBlock.getBalance(pu_key,long_chain)

def sendCoins(pu_send, amt_send, pr_send, pu_recv, amt_recv, miner_list):
    newTx = Transactions.Tx()
    newTx.add_input(pu_send, amt_send)
    newTx.add_output(pu_recv, amt_recv)
    newTx.sign(pr_send)
    SocketUtils.sendObj('localhost',newTx)
    return True

def loadKeys(pr_file, pu_file):
    return Signatures.loadPrivate(pr_file), Signatures.loadPublic(pu_file)


if __name__ == "__main__":

    import time
    import Miner
    import threading
    import Signatures
    miner_pr, miner_pu = Signatures.generate_keys()
    t1 = threading.Thread(target=Miner.minerServer, args=(('localhost',5005),))
```

Ln: 55 Col: 10

Overspend detected



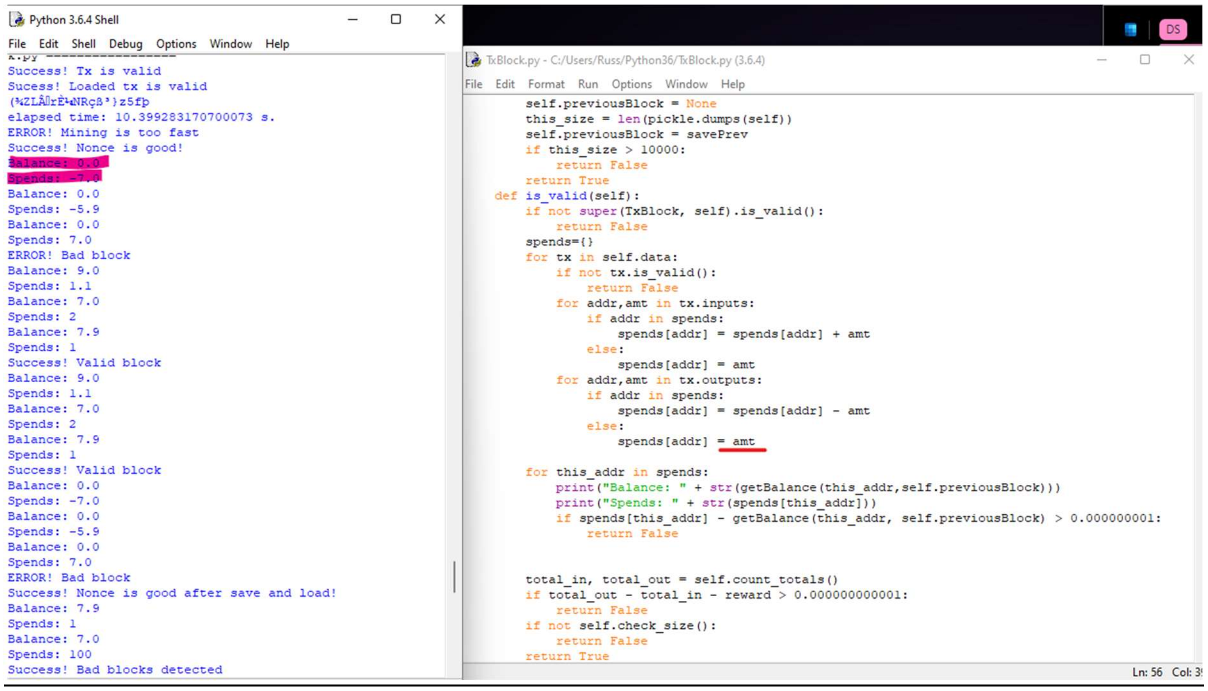
```
================= RESTART: C:/Users/Russ/Python36/TxBloc
k.py =================
Success! Tx is valid
Sucess! Loaded tx is valid
.|] 7äLM®)Â] uß
elapsed time: 1.8139636516571045 s.
ERROR! Mining is too fast
Success! Nonce is good!
ERROR! Bad block
Success! Valid block
Success! Valid block
ERROR! Bad block
Success! Nonce is good after save and load!
Success! Bad blocks detected
Success! Bad blocks detected
ERROR! Block reward fail
ERROR! Tx fees fail
Success! Greedy miner detected
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Overspend detected
Success! Overspend detected
>>>
```

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

```python
        self.previousBlock = None
        this_size = len(pickle.dumps(self))
        self.previousBlock = savePrev
        if this_size > 10000:
            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 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 spends[this_addr] - getBalance(this_addr, self.previousBlock) > 0.000
                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
    def good_nonce(self):
        digest = hashes.Hash(hashes.SHA256(), backend=default_backend())
        digest.update(bytes(str(self.data),'utf8'))
        digest.update(bytes(str(self.previousHash),'utf8'))
        digest.update(bytes(str(self.nonce),'utf8'))
        this_hash = digest.finalize()

        if this_hash[:leading_zeros] != bytes(''.join([ '\x4f' for i in range(leadin
            return False
        return int(this_hash[leading_zeros]) < next_char_limit
    def find_nonce(self,n_tries=1000000):
        for i in range(n_tries):
```

Ln: 50 Col: 28

Having a 0 balance and spending 7 coins because were subtracting in tx.outputs but spends[addr] should be equal to -amt

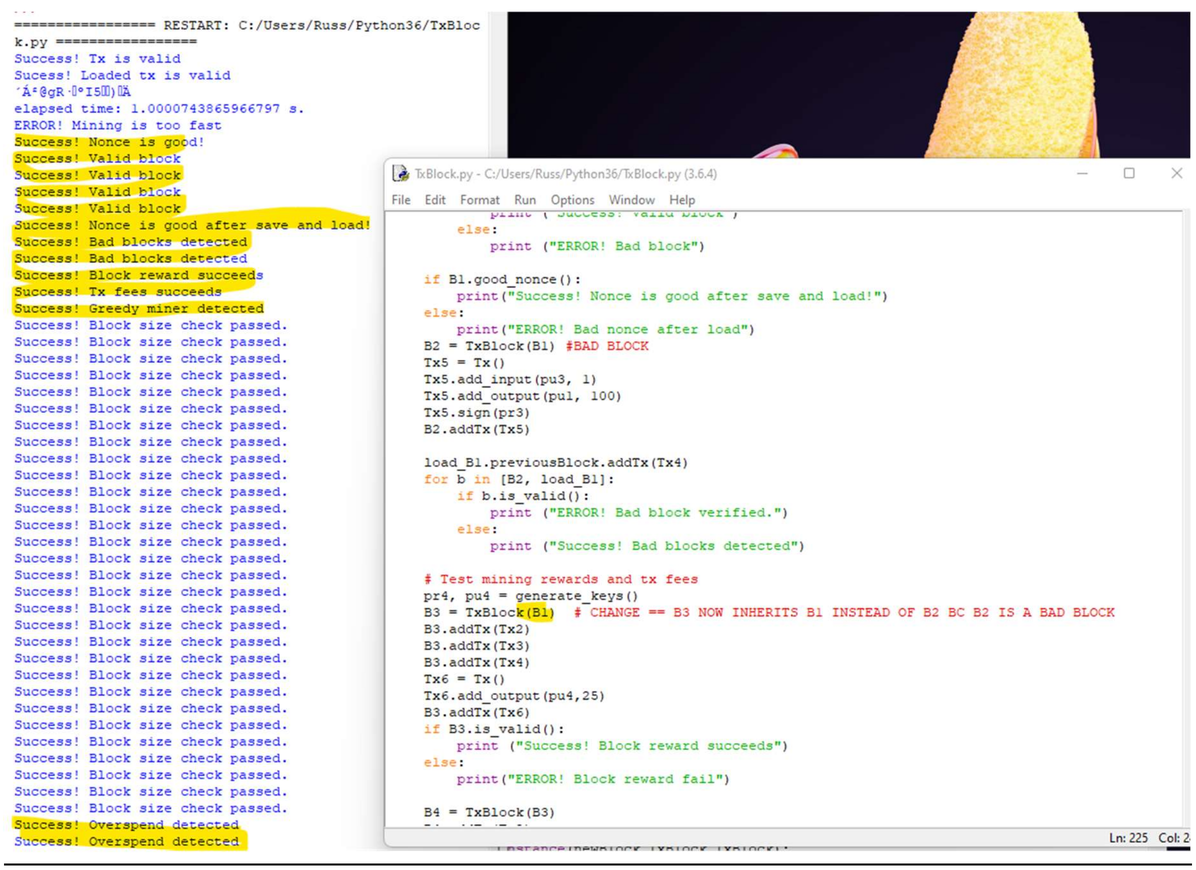


```
Python 3.6.4 Shell
k.py ==================
Success! Tx is valid
Sucess! Loaded tx is valid
(¾ZLÂrÈ¼NRçß³}z5fþ
elapsed time: 10.399283170700073 s.
ERROR! Mining is too fast
Success! Nonce is good!
Balance: 0.0
Spends: -7.0
Balance: 0.0
Spends: -5.9
Balance: 0.0
Spends: 7.0
ERROR! Bad block
Balance: 9.0
Spends: 1.1
Balance: 7.0
Spends: 2
Balance: 7.9
Spends: 1
Success! Valid block
Balance: 9.0
Spends: 1.1
Balance: 7.0
Spends: 2
Balance: 7.9
Spends: 1
Success! Valid block
Balance: 0.0
Spends: -7.0
Balance: 0.0
Spends: -5.9
Balance: 0.0
Spends: 7.0
ERROR! Bad block
Success! Nonce is good after save and load!
Balance: 7.9
Spends: 1
Balance: 7.0
Spends: 100
Success! Bad blocks detected
```

```python
TxBlock.py - C:/Users/Russ/Python36/TxBlock.py (3.6.4)
        self.previousBlock = None
        this_size = len(pickle.dumps(self))
        self.previousBlock = savePrev
        if this_size > 10000:
            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 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:
            print("Balance: " + str(getBalance(this_addr,self.previousBlock)))
            print("Spends: " + str(spends[this_addr]))
            if spends[this_addr] - getBalance(this_addr, self.previousBlock) > 0.000000001:
                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
```

Ln: 56 Col: 3

After adding – everything passes but overspend was not detected (see 12:19 for troubleshooting)



```
...
================== RESTART: C:/Users/Russ/Python36/TxBloc
k.py ==================
Success! Tx is valid
Sucess! Loaded tx is valid
´Á²@gR·°I5)Ä
elapsed time: 1.0000743865966797 s.
ERROR! Mining is too fast
Success! Nonce is good!
Success! Valid block
Success! Valid block
Success! Valid block
Success! Valid block
Success! Nonce is good after save and load!
Success! Bad blocks detected
Success! Bad blocks detected
Success! Block reward succeeds
Success! Tx fees succeeds
Success! Greedy miner detected
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Block size check passed.
Success! Overspend detected
Success! Overspend detected
```

```python
TxBlock.py - C:/Users/Russ/Python36/TxBlock.py (3.6.4)
            print ("Success! Valid block")
        else:
            print ("ERROR! Bad block")

    if B1.good_nonce():
        print("Success! Nonce is good after save and load!")
    else:
        print("ERROR! Bad nonce after load")
    B2 = TxBlock(B1) #BAD BLOCK
    Tx5 = Tx()
    Tx5.add_input(pu3, 1)
    Tx5.add_output(pu1, 100)
    Tx5.sign(pr3)
    B2.addTx(Tx5)

    load_B1.previousBlock.addTx(Tx4)
    for b in [B2, load_B1]:
        if b.is_valid():
            print ("ERROR! Bad block verified.")
        else:
            print ("Success! Bad blocks detected")

    # Test mining rewards and tx fees
    pr4, pu4 = generate_keys()
    B3 = TxBlock(B1)  # CHANGE == B3 NOW INHERITS B1 INSTEAD OF B2 BC B2 IS A BAD BLOCK
    B3.addTx(Tx2)
    B3.addTx(Tx3)
    B3.addTx(Tx4)
    Tx6 = Tx()
    Tx6.add_output(pu4,25)
    B3.addTx(Tx6)
    if B3.is_valid():
        print ("Success! Block reward succeeds")
    else:
        print("ERROR! Block reward fail")

    B4 = TxBlock(B3)
```

Ln: 225 Col: 2