# Source Code

Kindly refer the python files named Block\_Ledger.py and Block\_Miner.py for the full source code.

# Documentations

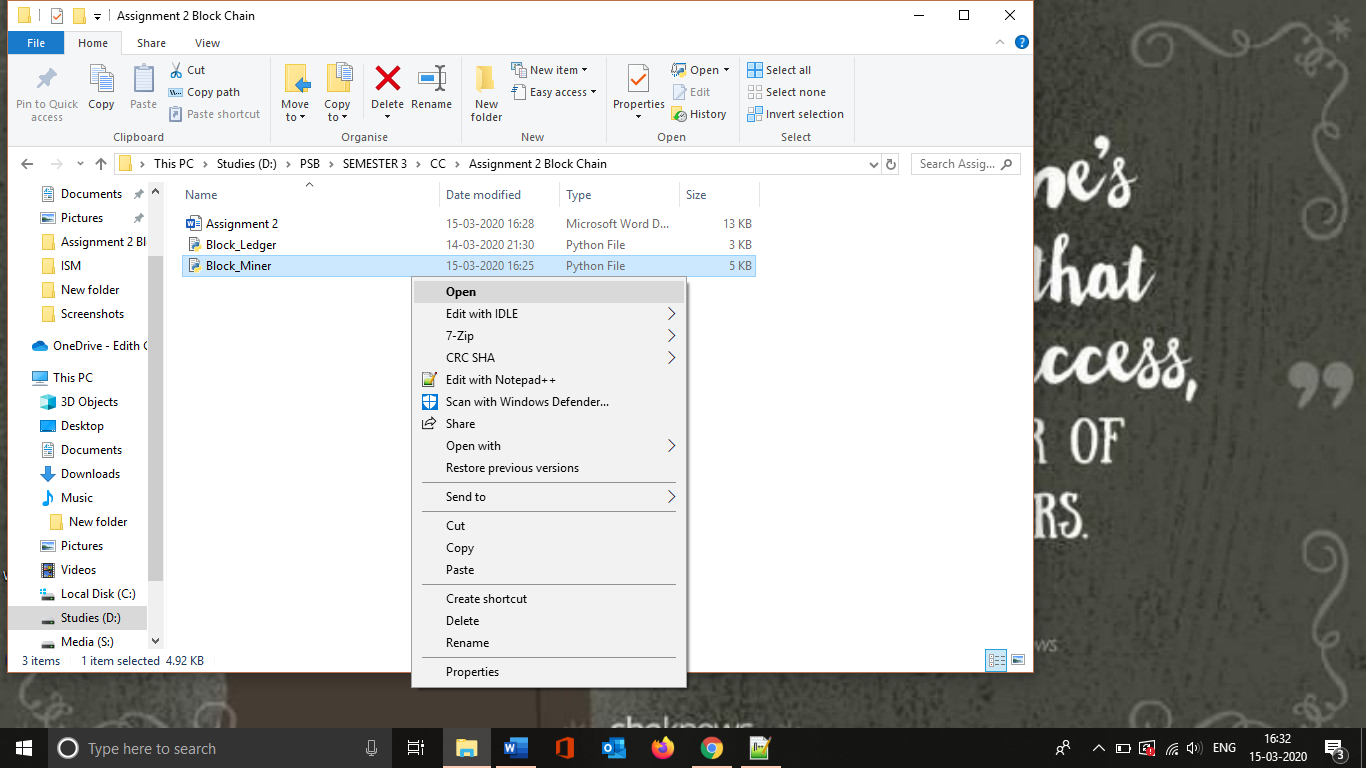
To run the program the user should fulfil the following requirements,

* Python 3.6 or higher
* Write permission to program folder
* OS – Windows 10 (Prepared)

# Instructions

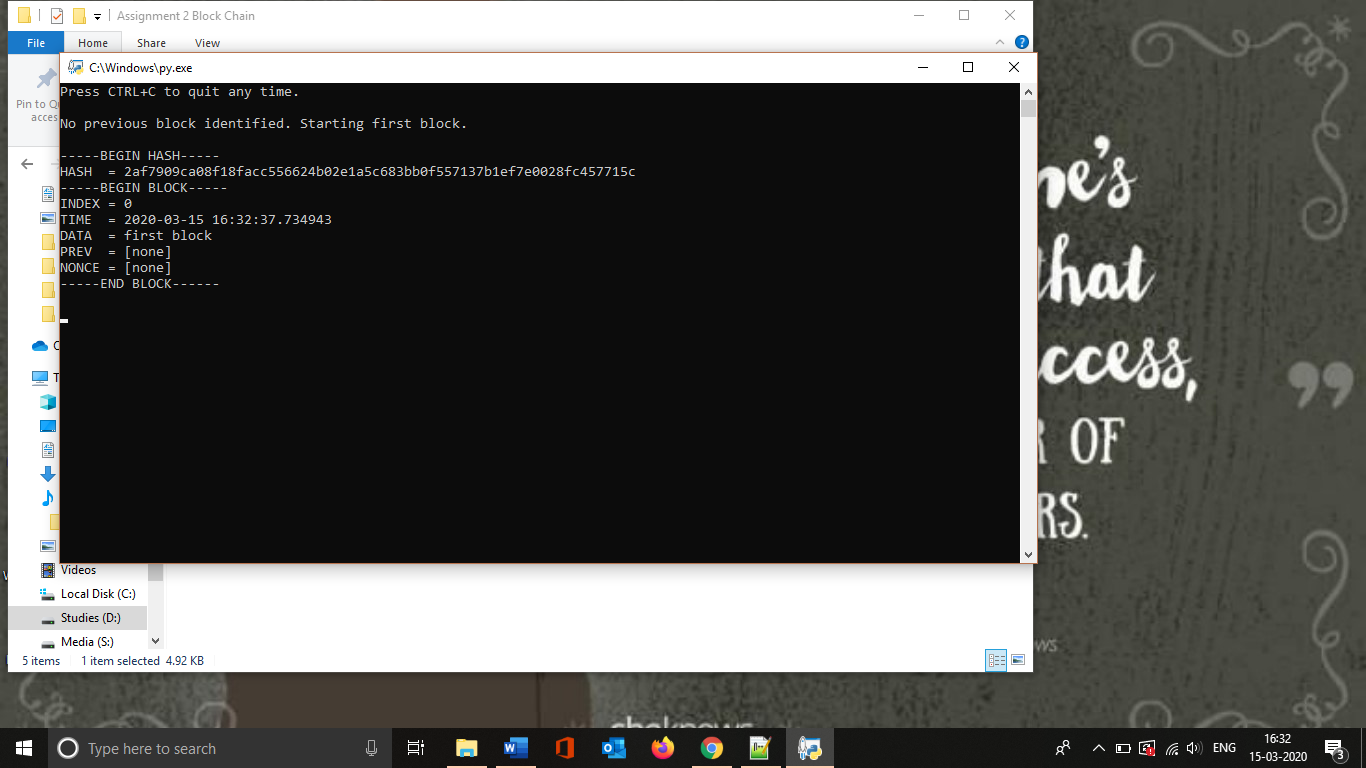
(It is assumed that the given program contain write access to the folder it is stores and the files blockchain.txt and ledger.txt are created and modified only through the program and not by user)

1. Execute the Program Block\_Miner.py either by using right click and selecting open or by using double click on it.



Screenshot 1 Running the Program

2. Once the user executed the program it should display the following terminal

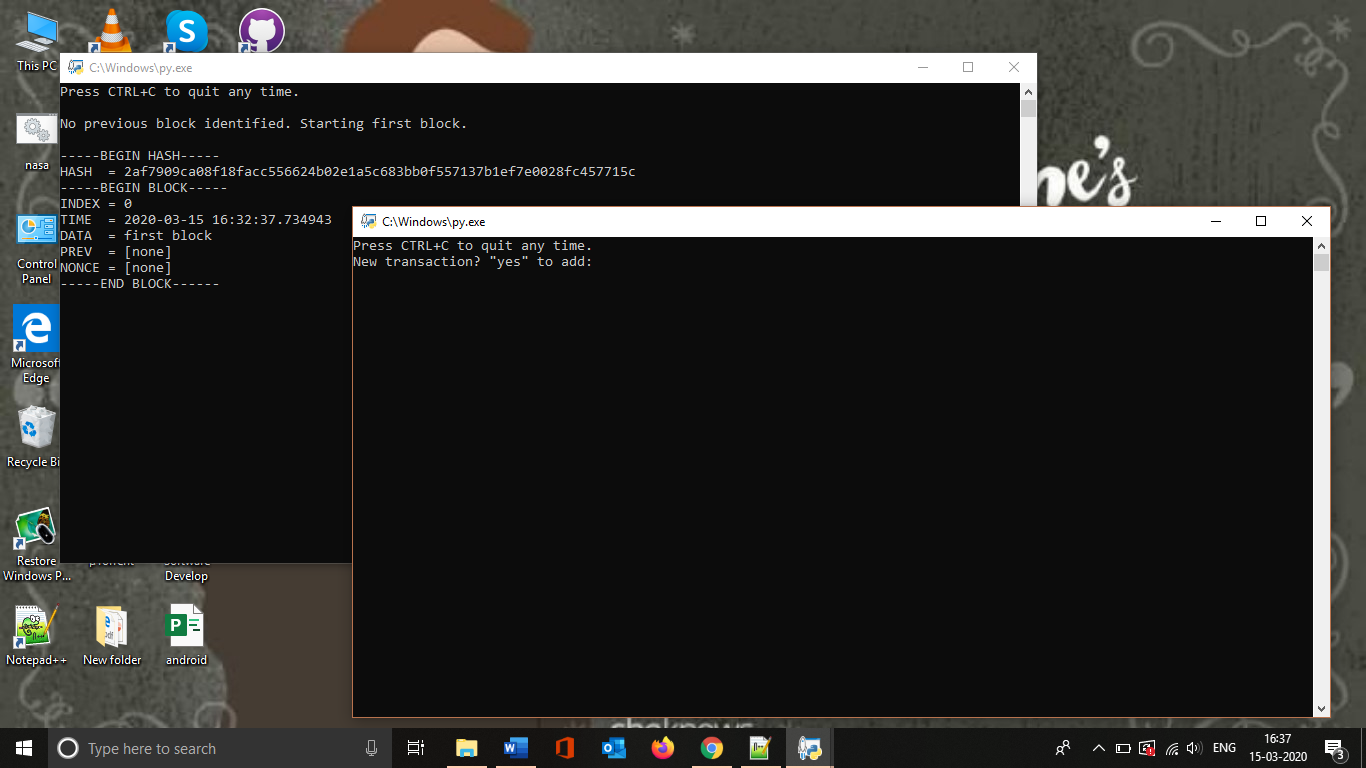


Screenshot 2 Running the program (cont....)

3. Leave the miner program as it is to monitor the changes occurs in the ledger.txt document.

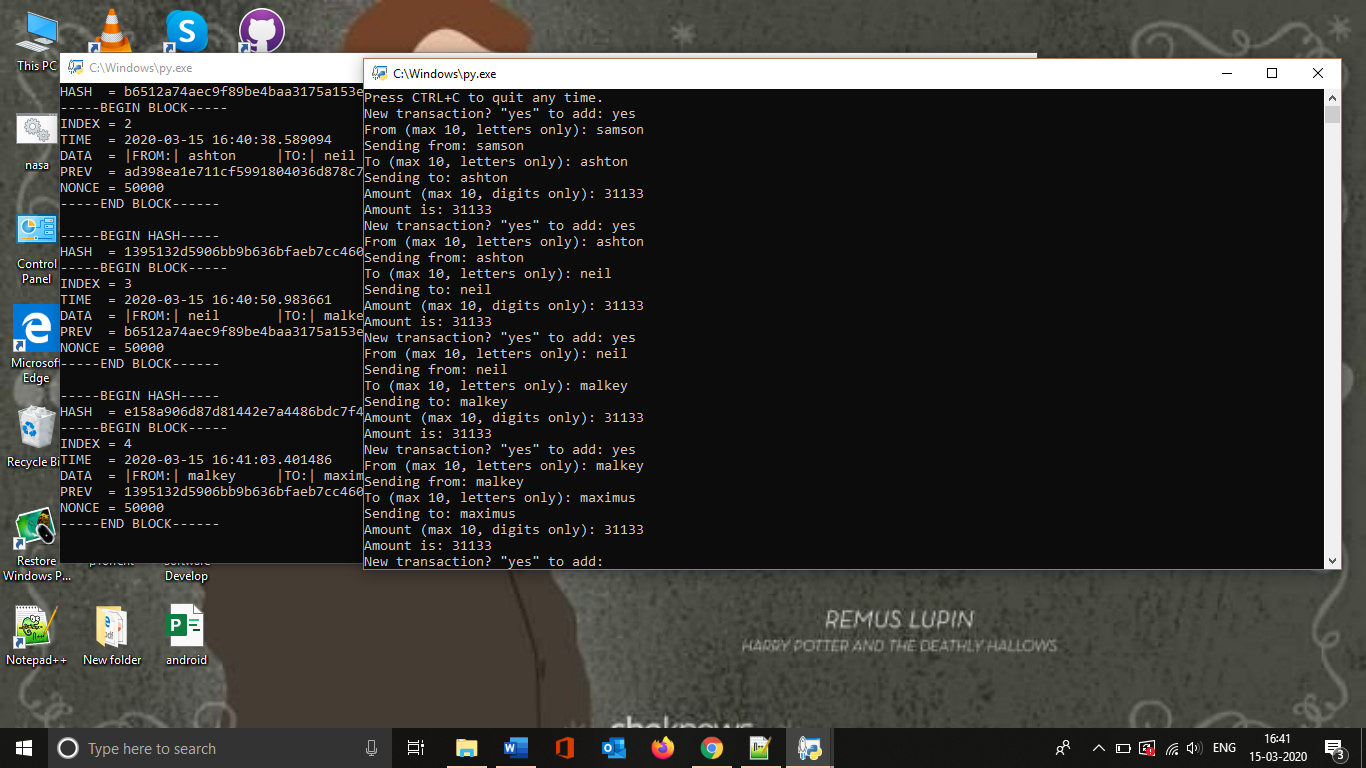
4. Now execute the Block\_ledger.py.

5. Once it get executed the user has to enter “yes” to add an transaction



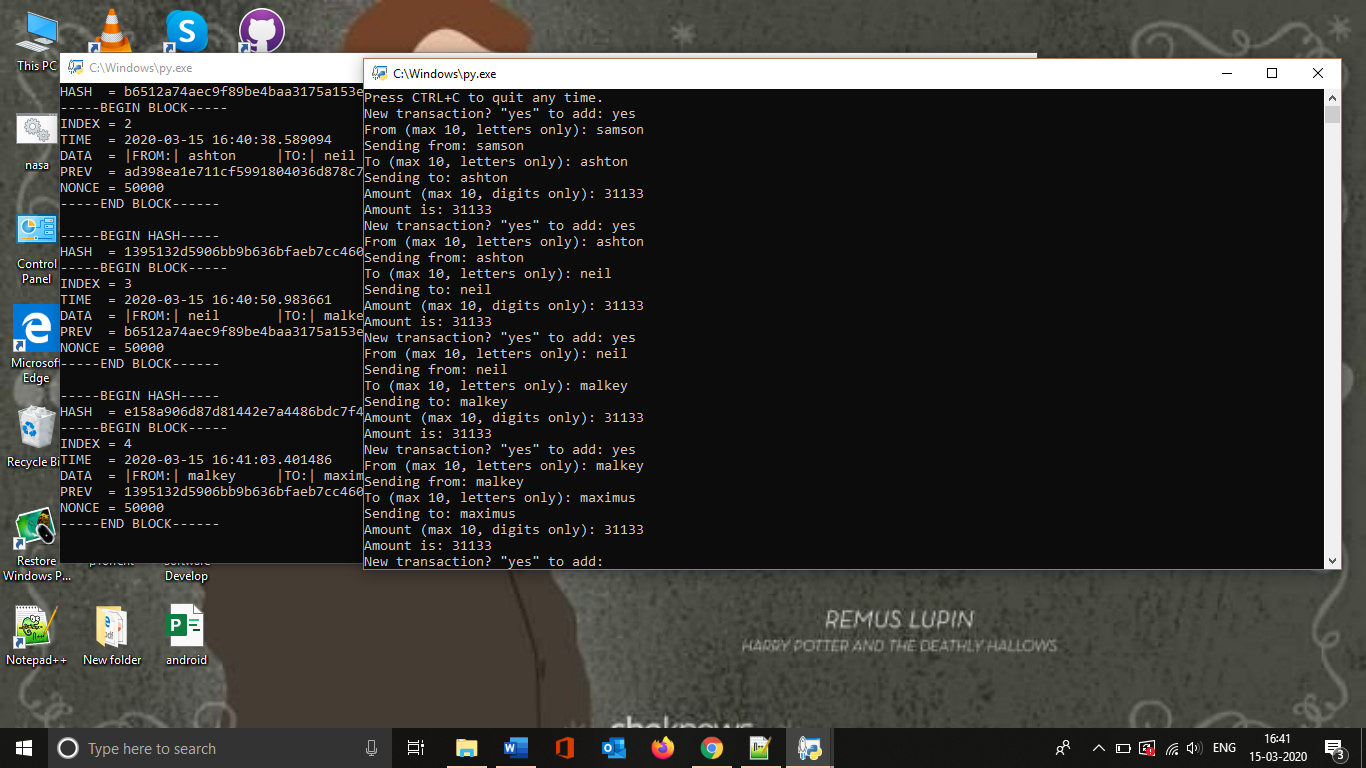
Screenshot 3 Running the program (cont....)

6. Once the user entered yes then the sender name should be entered. The name should contains only the letters, cannot exceed 10 letters and if the user leave it as empty then a random name would get generated.



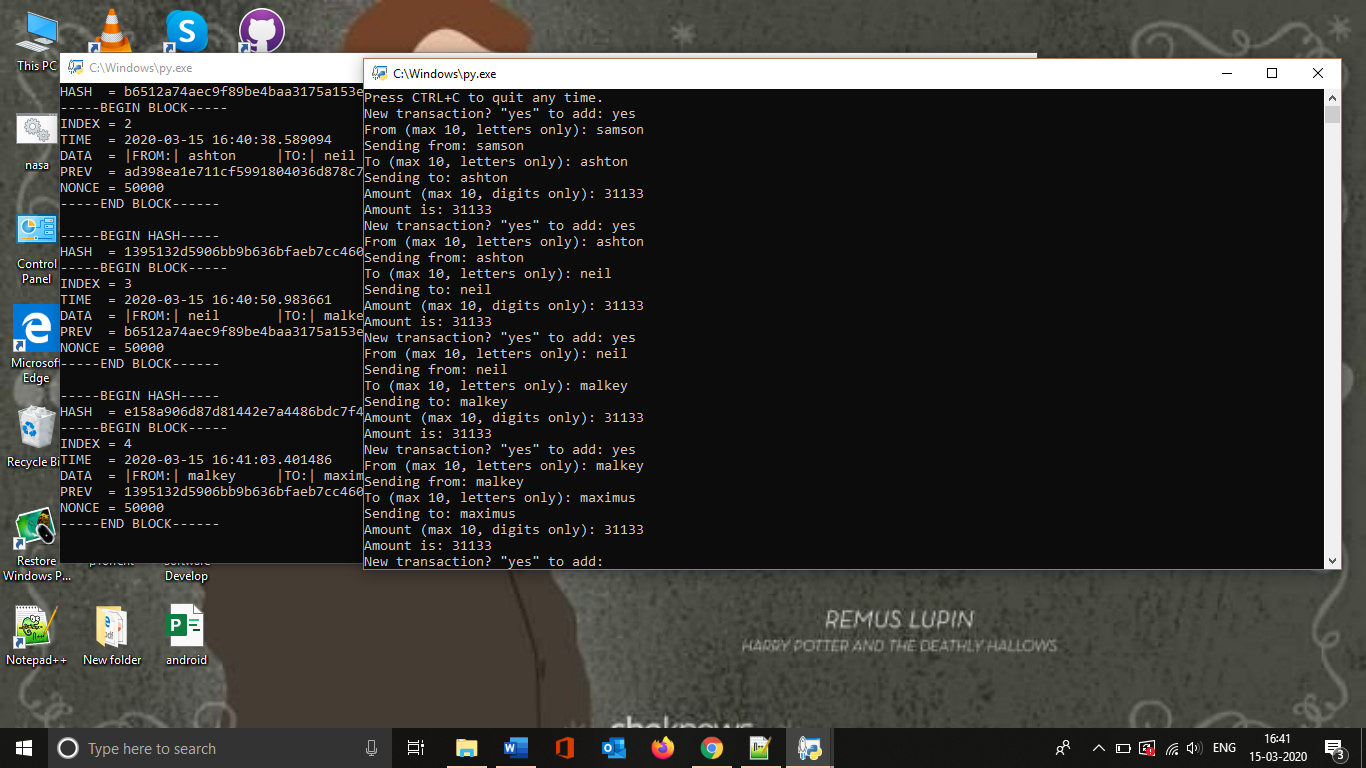
Screenshot 4 Running the program (cont....)

7. Once the sender name entered then the receiver name should get entered. Same policies of sender name should be followed.



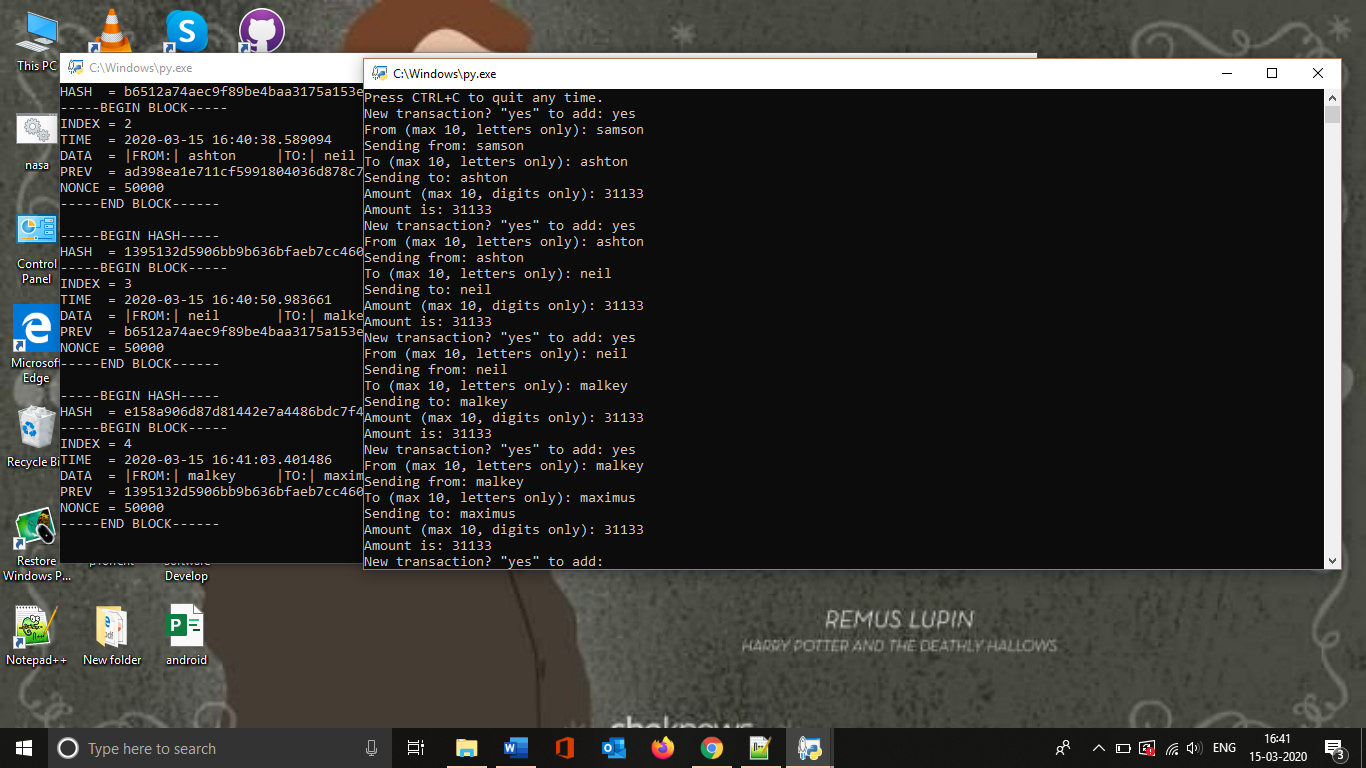
Screenshot 5 Running the program (cont....)

8. Once the receiver name entered press enter to enter the amount. And the number should contain only number characters and cannot exceed 10 digits. And if the user left it blank then a random 10 digit number would get generated.



Screenshot 6 Running the program (cont....)

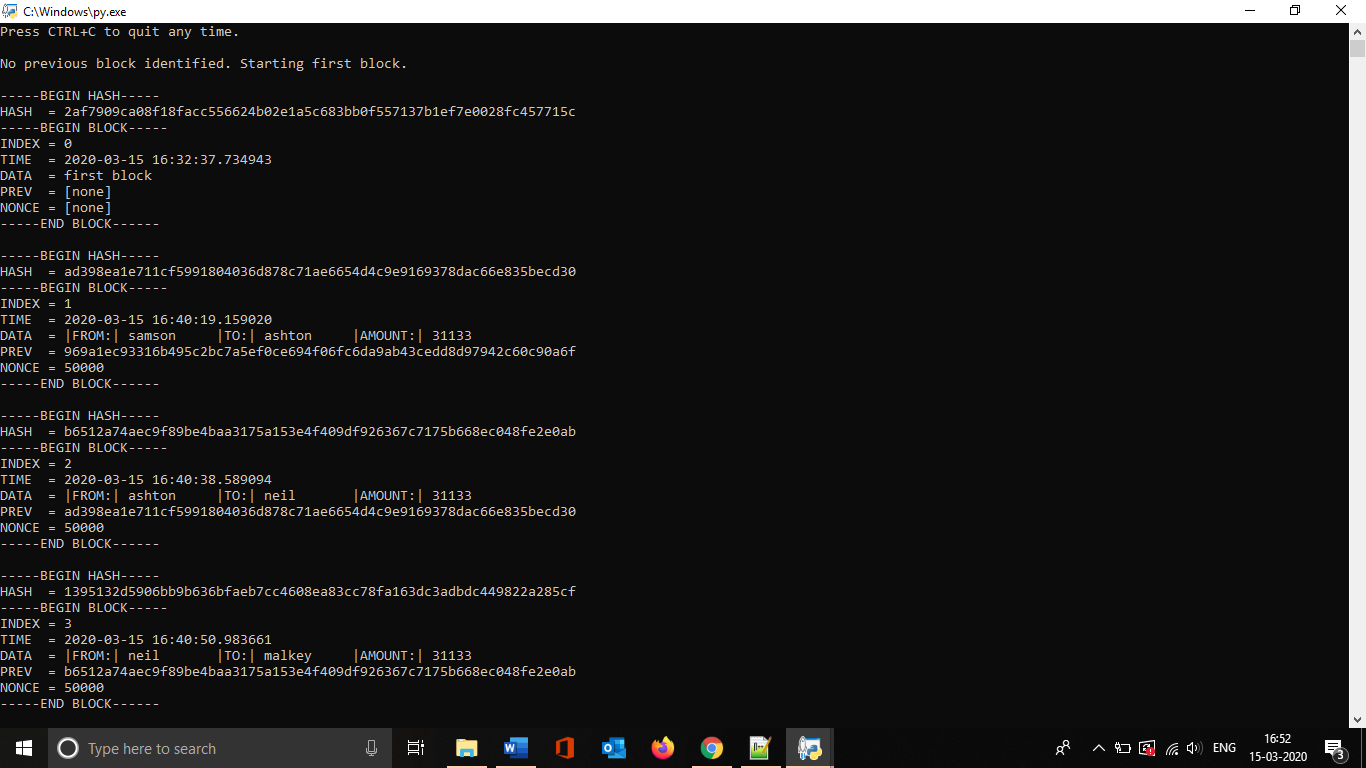
9. Once it done the user can repeat the 5-8 to add more transactions.



Screenshot 7 Running the program (cont....)

10. Block details can be viewed either by opening the text file blockchain.txt or the terminal processing Block\_Miner.py.

11. User can exit from the program at any time by using **CTRL+C**



Screenshot 8 Running the program (cont....)



Screenshot 9 Running the program (cont....)

Screenshots 1 -9 are shows the demonstration with required data.

# Flow

**Block\_Ledger.py**

Sender (from) Details



* Letters only
* No numbers
* Convers to lower case
* If empty, generates random

Yes to perform transaction

Sender (from) Details

* Letters only
* No numbers
* Convers to lower case
* If empty, generates random

Start

CTRL + C to Exit

Amount

* Only number
* If empty, generates random

randomvalue()

Generates random values if user failed to enter any

transadd()

Gets the details from user

Writes the details in ledger.txt

**Block\_Miner.py**

If any ledger change detected and

Start

look for previous block

if not

|  |
| --- |
| blocknew()   * Increment index * Get timestamp * Read new data * Hash previous block |

|  |
| --- |
| **timestamp()**  python library for date time function  **blockwrite()**  print output in terminal  **emptyblock()**  opens blockchain.txt to look for first hash  **Hashledger()**  Convert the legder.txt data to SHA 256 contents  **prevhashblock()**  Reads the lash block from the blockchain.txt file  **blocknew()**  calls all the above stated programs |

References

Gerald, N. (2017). Let’s Build the Tiniest Blockchain. Retrieved from <https://medium.com/crypto-currently/lets-build-the-tiniest-blockchain-e70965a248b>

Python Software Foundation. (2020). hashlib — Secure hashes and message digests — Python 3.8.2 documentation. Retrieved from <https://docs.python.org/3/library/hashlib.html>

Satwik, K. (2020). Develop a blockchain application from scratch in Python. Retrieved from <https://developer.ibm.com/technologies/blockchain/tutorials/develop-a-blockchain-application-from-scratch-in-python/>