**Program Documentation**

**User interaction program (transaction\_recording program)**

Importing python library json

When the user runs the program, it will ask whether he/she wants to write a transaction. If the user entered “yes” then he/she can write the transaction after each transaction, the program will ask the user whether he/she wants to continue or exit. If the user prompts “no” then the program will end.

**Functions**

**A screenshot of a cell phone

Description automatically generated**

Top of the program, I have created three functions. Such as inputFloat function and savedata function. inputFloat function will go through a loop until the user prompts a float value. Savedata function will open a file name data.txt, and you can dump any data into it and then close it.

**Program**

**A screenshot of a cell phone

Description automatically generated**

Trying to open the data.txt file and load its data to a variable named data. If the data.txt file does not exist, then create an empty list variable name data. Afterwards, inside a while loop, the program will save each transaction of users to a dictionary variable called “name” (to check, if the user’s input is an integer and a float we use the inputFloat function) and then the dictionary will be appended to the list variable named data. Afterwards, the data list will be saved into data.txt using the savedata function.

After each transaction, the program will ask from the user whether he wants to continue or exit (yes/no) the program, if the user selects the exit option, the program will end.

**Block\_mining program**

Importing python libraries json, datetime, hashlib

The user has to run the program, and if transactions exist, it will ask the user whether he/she wants to create the blockchain for those transactions.

If new transactions do not exist, then the program will print no new transactions available, and the program will end

If transaction file does not exist, then the program will print an error message saying the transaction file does not exist and the program will end.

**Functions**

****

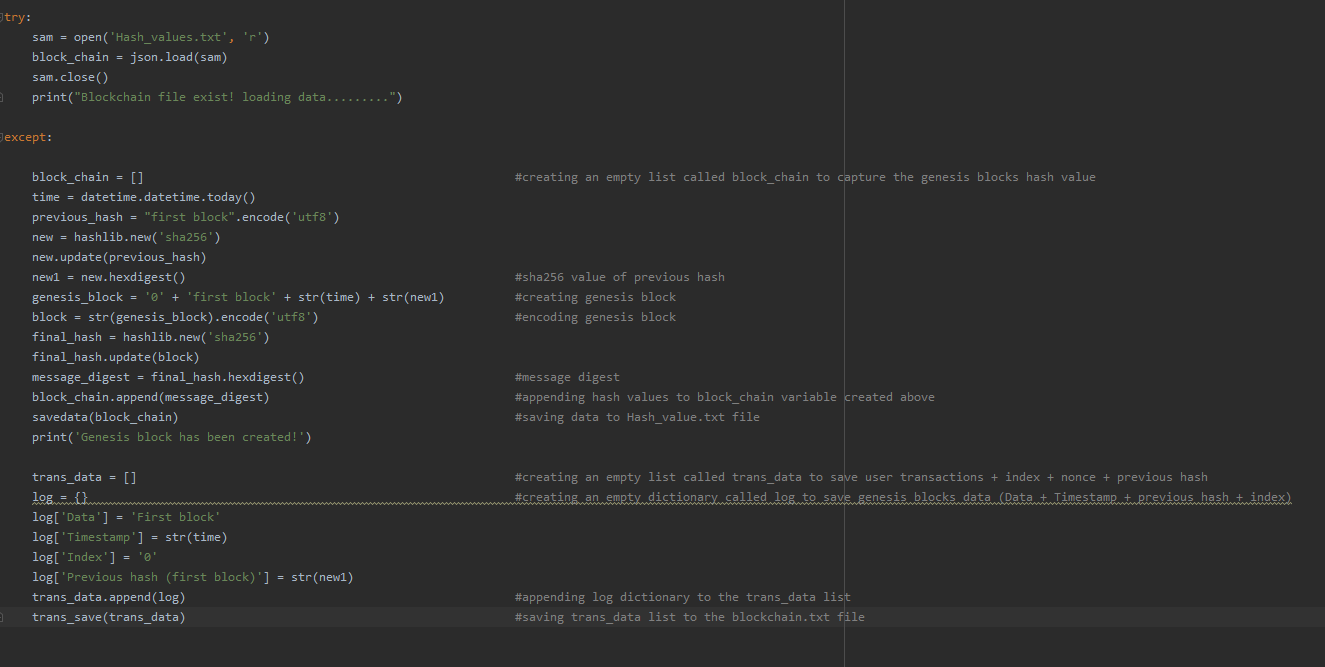
Top of the program, I have created three functions. Such as savedata function, save function, and trans\_save function. Savedata function will open a Hash\_values.txt file in write mode and dump any data into it and close it. Save function also works same as savedata function, but it will open a new file as data.txt, and save\_trans function also works same as savedata function, it will open Blockchain\_values.txt file in write mode.

**Program**

**A screenshot of a cell phone

Description automatically generated**

The program will try to open data.txt file in read mode and load its data to a variable name data and then close the file and print “transaction file exists”. If the data.txt file does not exist (if no transactions are available) it will print “no transaction file available”, and the program will end.



If the data.txt file exists, then the program will try to open Hash\_values.txt file in read mode and load its data into the block\_chain variable and then close the file and print “blockchain exists! Loading data.” If the Hash\_values.txt file does not exist, then the program will create an empty list variable named block\_chain. Afterwards, the program will create the genesis block (first block of hash).

**Genesis block**

**To create the first genesis block**

Index=0

Previous hash = “first block” hash value

Timestamp= current datetime

Data = “first block”

Created a variable named genesis\_block and added all the variable created above together in the genesis\_block variable (data + index + timestamp + previous hash (hash value using hashlib.new(‘sha256’)). After creating the genesis block, had to encode it using (‘utf8’) to turn the value into hash. Then the final hash value is set to a variable named message\_digest and appended that variable into the block\_chain list and saved the block\_chain data into the Hash\_values.txt file using savedata function. Finally, it will print out “Genesis block has been created”.

And then an empty list called trans\_data will be created to store the transaction data. Afterwards, a dictionary called log will be created to store the data of the genesis block (index + data + previous hash + timestamp) and then the dictionary will be appended into the trans\_data list and saving trans\_data list to Blockchain.txt file using trans\_data function.

**Using pseudocode to explain the final steps of the program**



The program checks the length of the data variable created above and if its higher than or equal to 1 the following commands will continue

If the length of block\_chain is equal to 1

create a variable named counter and set it to 0

create a variable named counter2 and set it to 1

Else

Create a variable named counter and set it to the length of the bock\_chain - 1

Create a variable named counter2 and set it to the length of block\_chain

Continuous loop

Print out a message asking “Do you want to start the blockchain process: (yes/no)

Place users input into a variable named dam

If the dam variable is equal to “yes.”

For each transaction data in the data variable

Each transaction data will be added together and placed in a variable named first (sender, receiver, amount, timestamp and counter2 (which is the index number)).

The last hash value of block\_chain will be added to the first1 variable and concatenate it with the first variable created above

Create a dictionary name log\_data to store all the user transactions + index + previous hash

Try

Try to open blockchain.txt file and load its data to trans\_data variable (user transaction data file) and then close the file

Except

Create an empty list called trans\_data

Increment counter by one (to get the previous hash value)

Increment counter2 by one (to keep getting the correct index)

Create a variable named nonce and set it to 0

A screenshot of a cell phone

Description automatically generated

Continuous loop

The nonce variable value will be added to the first1 variable and placed the value in a variable named second

The second variable will be turned into a hash (first encode using utf8 and then hash it) and save it in a variable named message\_digest

The count variable will count the zeroes in message\_digest

if the count variable equal to 14 (checking whether 14 zeroes exist in the message digest)

Then message\_digest data will be appended into block\_chain variable created above

Block \_chain variable data will be saved into Hash\_values.txt file using save data function

The nonce value will be added to the log\_data dictionary, and then it will append to the trans\_data list and save it to the Blockchain.txt file using trans\_save function

Break the loop

if the nonce variable is higher than 5000

Then message\_digest data will be appended into block\_chain variable created above

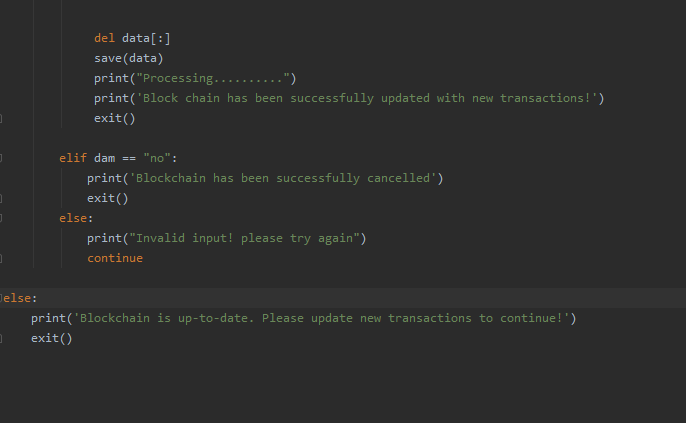
Block \_chain variable data will be saved into Hash\_values.txt file using save data function

The nonce value will be added to the log\_data dictionary, and then it will append to the trans\_data list and save it to the Blockchain.txt file using trans\_save function

Break the loop

else

Increment nonce variable by one and continue the loop



Afterwards, Delete all the contents in the data variable and save the changes of data variable into data.txt using save function

If dam variable is equal to “no” then

Then print “blockchain has been successfully cancelled” and exit the program

If the user prompted anything else other than yes or no

Then the program will print “invalid input! Please try again”, and the loop will continue until the user enter a valid input

If the length of data variable is lower than 1

Then print “blockchain is up to date” and exit the program

**Transaction\_recording and block\_mining programs will generate three files**

* Data.txt (will capture the transactions and save those transactions into Blockchain.txt file and at the end of the program data.txt fill be an empty file)
* Blockchain.txt (will contain all the transaction data + index + nonce + previous hash)
* Hash\_values.txt (will contain all the final hash values of transactions and the genesis block)