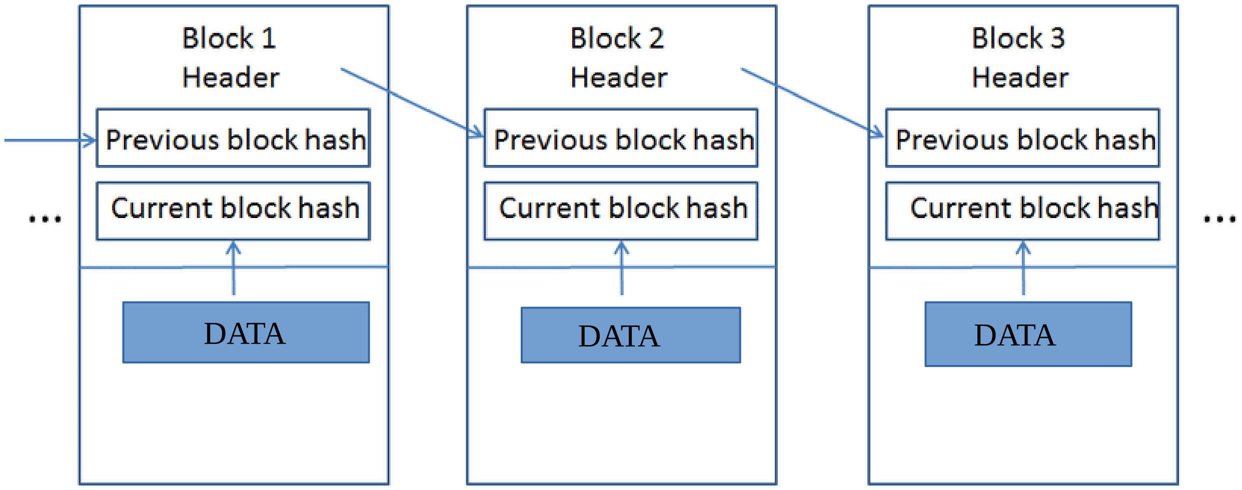
BLOCKCHAIN IMPLEMENTATION USING PYTHON

ROHINI R (2017103579)
ANDREA SHARON S (2017103056)

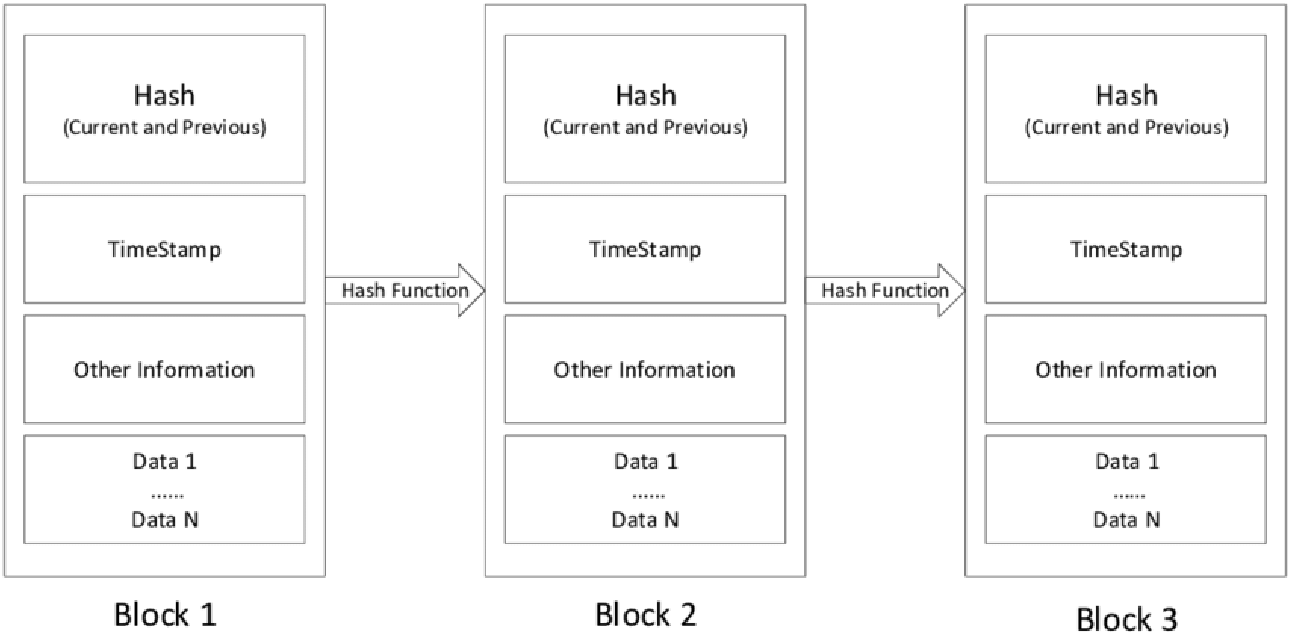
AIM:

The goal of this project is to implement a basic blockchain structure. It is focused exclusively in the hashed ledger feature (Ledger Database— A NoSQL database that provides an immutable, transparent log. Here we summarize all the results obtained with suitable diagrammatic representations.

DIAGRAMMATIC REPRESENTATION:



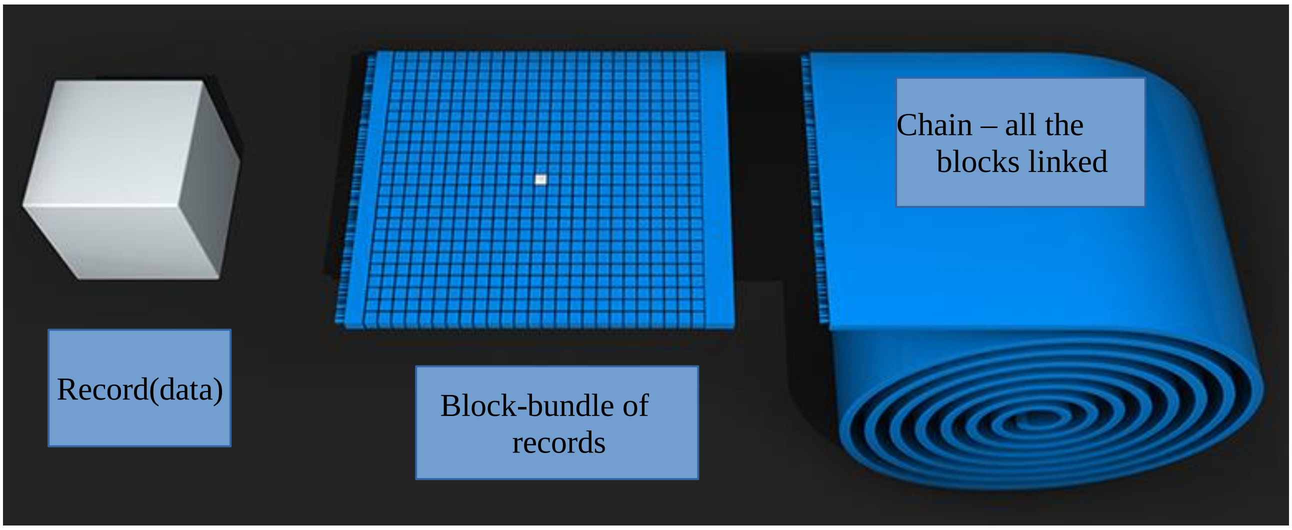
FLOWCHART:



EXPLANATION---> CREATING A BLOCKCHAIN STRUCTURE INCLUDES:

STEP: 1

The record lists the details, including a digital signature from each party.



STEP : 2

The record is checked by the network. The computers in the network, called 'nodes', check the details of the trade to make sure it is valid.

STEP : 3

The records that the network accepted are added to a block. Each block contains a unique code called a hash. It also contains the hash of the previous block in the chain.

STEP : 4

The block is added to the blockchain. The hash codes connect the blocks together in a specific order.

These are the steps followed to create a blockchain structure.

LIBRARIES INCLUDED:

1) datetime
2) hashlib
3) time

IMPLEMENTATION:

Here the implemented blockchain is composed of 3 classes.

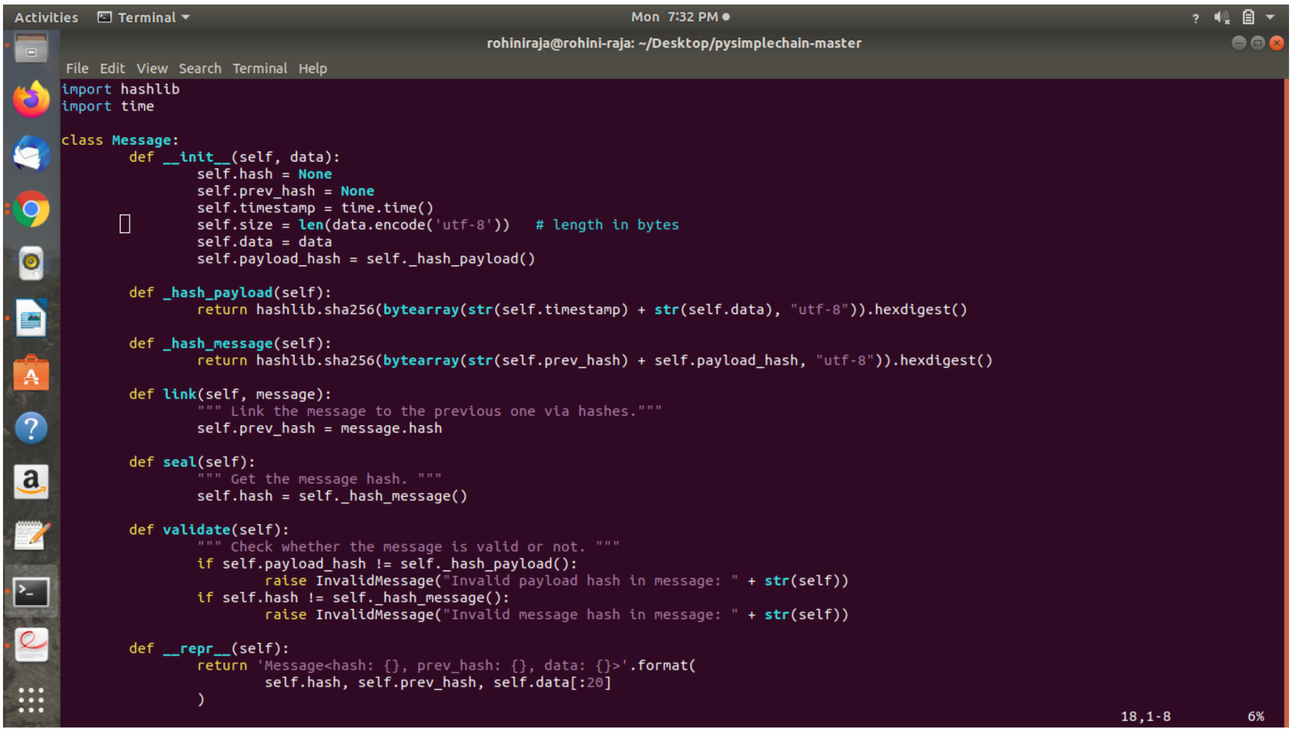
1) The Message() class,
2) The Block() class,
3) The Chain() class.

MESSAGE( ) CLASS:

A message is the basic data container. It is sealed when added to a block and has 2 hashes that identify it:

->the payload hash and
->the block hash.

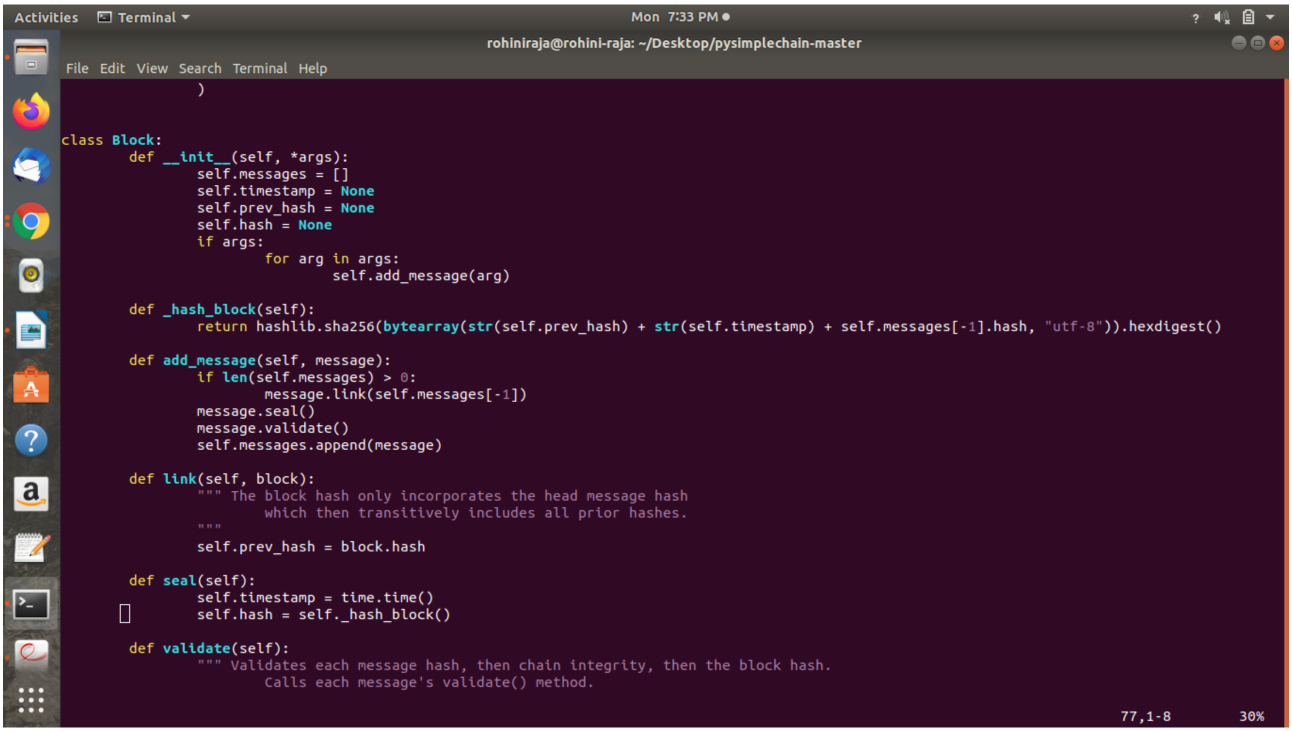
Each message will be linked to the previous message via hash pointers (the prev_hash attribute). The validate message method will ensure the integrity of each message, but will not check if the hash pointers are correct. This is left to the validate method in the Block() class.


Activities
Terminal
Mon 7:32 PM
rohiniraja@rohini-raja: ~/Desktop/pysimplechain-master
File Edit View Search Terminal Help

```
import hashlib
import time

class Message:
        def __init__(self, data):
                self.hash = None
                self.prev_hash = None
                self.timestamp = time.time()
                self.size = len(data.encode('utf-8'))   # length in bytes
                self.data = data
                self.payload_hash = self._hash_payload()

        def _hash_payload(self):
                return hashlib.sha256(bytearray(str(self.timestamp) + str(self.data), "utf-8")).hexdigest()

        def _hash_message(self):
                return hashlib.sha256(bytearray(str(self.prev_hash) + self.payload_hash, "utf-8")).hexdigest()

        def link(self, message):
                """ Link the message to the previous one via hashes."""
                self.prev_hash = message.hash

        def seal(self):
                """ Get the message hash. """
                self.hash = self._hash_message()

        def validate(self):
                """ Check whether the message is valid or not. """
                if self.payload_hash != self._hash_payload():
                        raise InvalidMessage("Invalid payload hash in message: " + str(self))
                if self.hash != self._hash_message():
                        raise InvalidMessage("Invalid message hash in message: " + str(self))

        def __repr__(self):
                return 'Message<hash: {}, prev_hash: {}, data: {}>'.format(
                        self.hash, self.prev_hash, self.data[:20]
                )
                                                                                18,1-8          6%
```

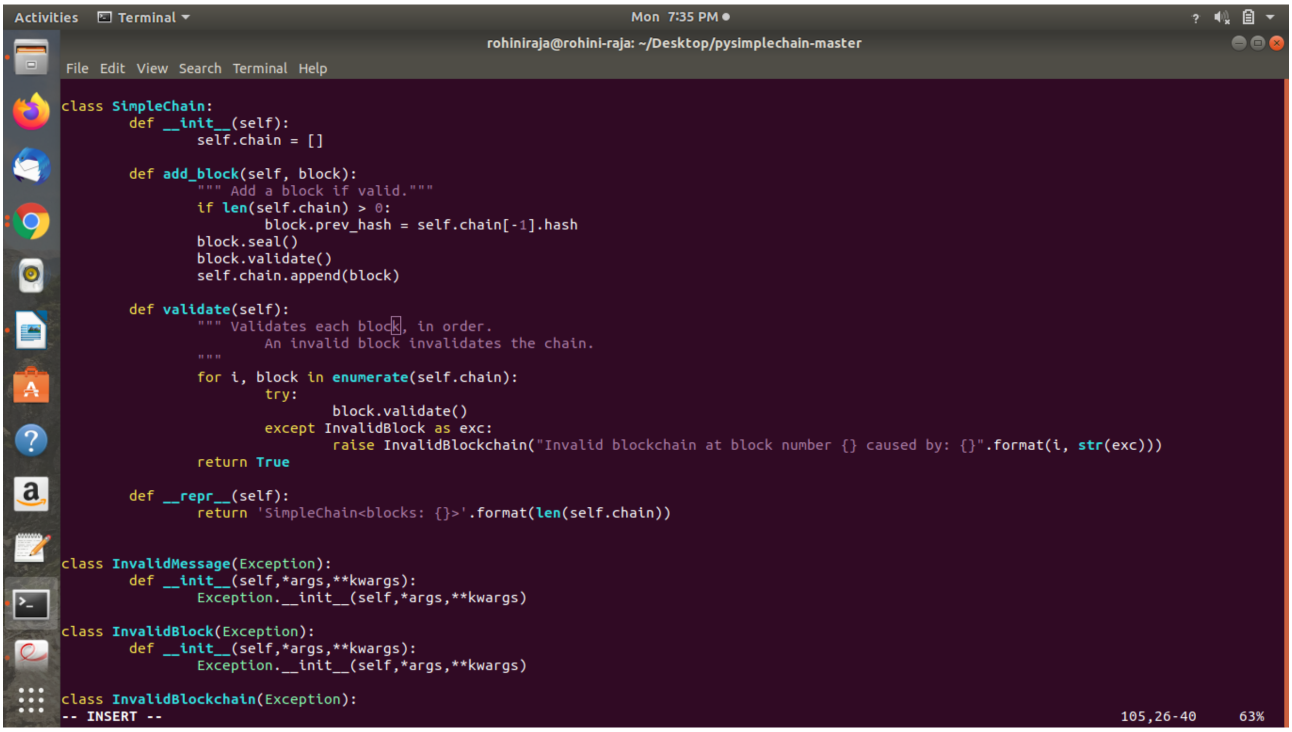
BLOCK( ) CLASS:

A block can contain 1,...,n messages that are linked sequentially one after the other. When a block is added to the chain, it's sealed and validated to ensure that the messages are correctly ordered and the hash pointers match. Once the block is sealed and hashed, it is validated by checking the expected vs the actual.


Activities
Terminal
Mon 7:33 PM
rohiniraja@rohini-raja: ~/Desktop/pysimplechain-master
File Edit View Search Terminal Help

```
                )

class Block:
        def __init__(self, *args):
                self.messages = []
                self.timestamp = None
                self.prev_hash = None
                self.hash = None
                if args:
                        for arg in args:
                                self.add_message(arg)

        def _hash_block(self):
                return hashlib.sha256(bytearray(str(self.prev_hash) + str(self.timestamp) + self.messages[-1].hash, "utf-8")).hexdigest()

        def add_message(self, message):
                if len(self.messages) > 0:
                        message.link(self.messages[-1])
                message.seal()
                message.validate()
                self.messages.append(message)

        def link(self, block):
                """ The block hash only incorporates the head message hash
                        which then transitively includes all prior hashes.
                """
                self.prev_hash = block.hash

        def seal(self):
                self.timestamp = time.time()
                self.hash = self._hash_block()

        def validate(self):
                """ Validates each message hash, then chain integrity, then the block hash.
                        Calls each message's validate() method.
                                                                                        77,1-8          30%
```

CHAIN( ) CLASS:


Activities
Terminal
Mon 7:35 PM
rohiniraja@rohini-raja: ~/Desktop/pysimplechain-master
File Edit View Search Terminal Help

```
class SimpleChain:
        def __init__(self):
                self.chain = []

        def add_block(self, block):
                """ Add a block if valid."""
                if len(self.chain) > 0:
                        block.prev_hash = self.chain[-1].hash
                block.seal()
                block.validate()
                self.chain.append(block)

        def validate(self):
                """ Validates each block, in order.
                        An invalid block invalidates the chain.
                """
                for i, block in enumerate(self.chain):
                        try:
                                block.validate()
                        except InvalidBlock as exc:
                                raise InvalidBlockchain("Invalid blockchain at block number {} caused by: {}".format(i, str(exc)))
                return True

        def __repr__(self):
                return 'SimpleChain<blocks: {}>'.format(len(self.chain))

class InvalidMessage(Exception):
        def __init__(self,*args,**kwargs):
                Exception.__init__(self,*args,**kwargs)

class InvalidBlock(Exception):
        def __init__(self,*args,**kwargs):
                Exception.__init__(self,*args,**kwargs)

class InvalidBlockchain(Exception):
-- INSERT --                                                                            105,26-40       63%
```

A chain can contain 1,...,m blocks that are linked sequentially one after another. The chain integrity can be validated at any time calling the validate method, which will call each block's validate method and will raise an InvalidBlockchain exception.

ALGORITHMS USED:

Proof-Of-Work Algorithm:

This algorithm is used to confirm transactions and produce new blocks to the chain.

INTERACTIONS PERFORMED:

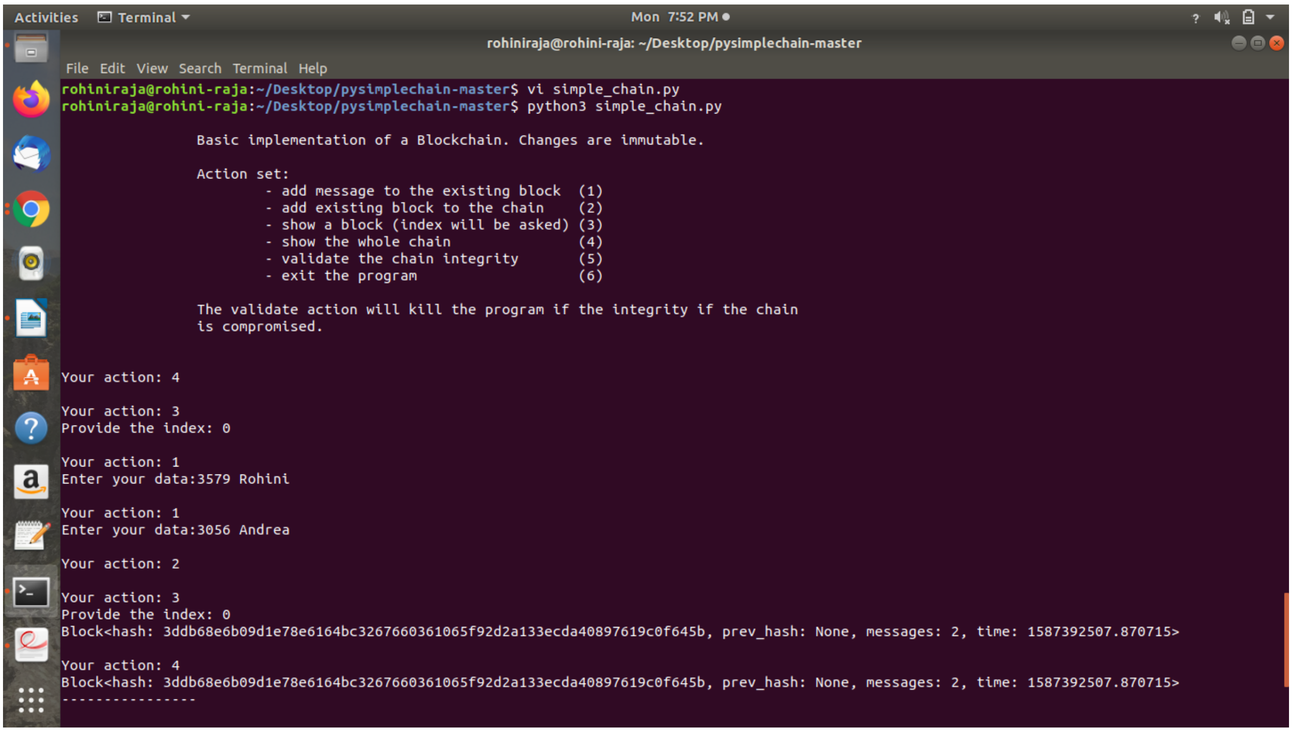
The interactions performed are as follows:

A function named manager() is provided which is used to interact with the blockchain via the Terminal/Console.

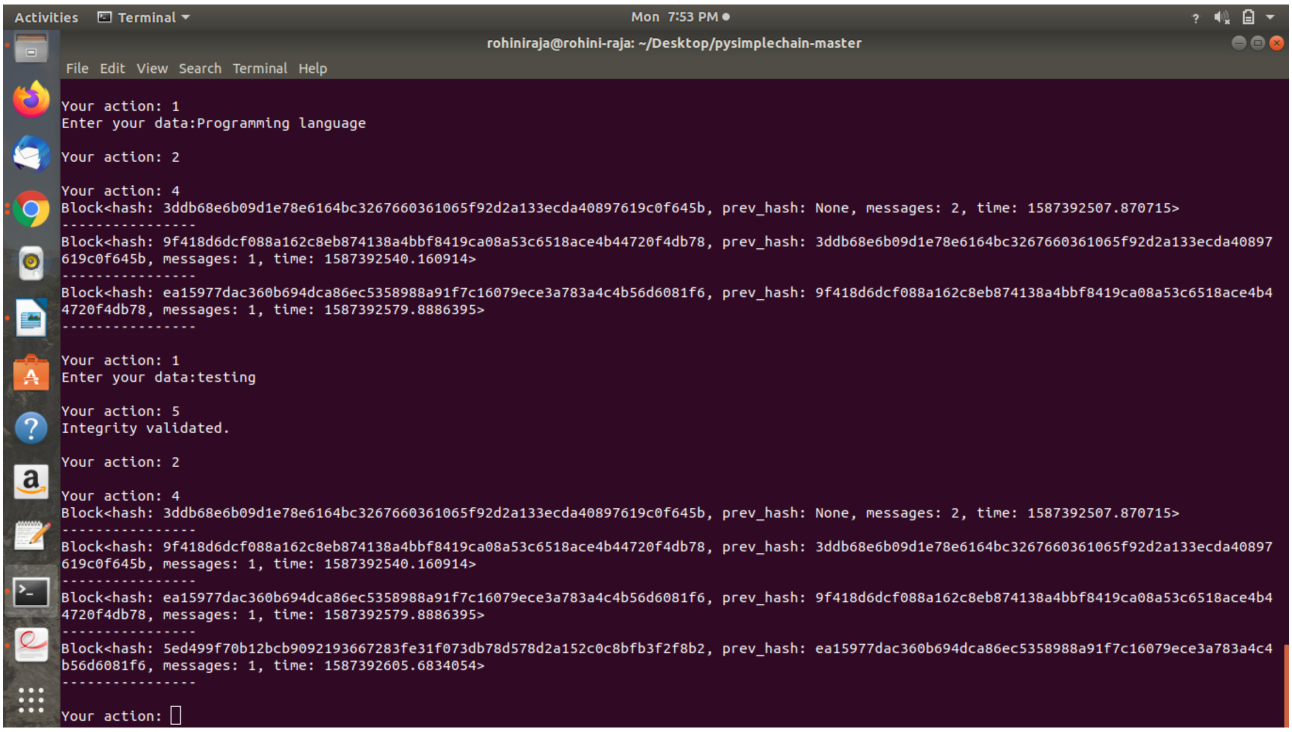
Basic Actions include:

- **To Add Message to Block:** Allows to add a data/message to the current block.
- **To Add Block to Chain:** Allows to add the current block to the existing chain if it is not empty.
- **Show the Block:** This asks for an index and if exists a block with that index, returns the attributes of the block.
- **Show Chain:** This returns some of the block attributes for each block in the chain.
- **Validating Integrity:** This will return True if the integrity has been validated otherwise terminates the program raising the appropiate exception.
- **Exit:** This will delete the blockchain and terminates the program.

OUTPUT SCREENSHOTS:

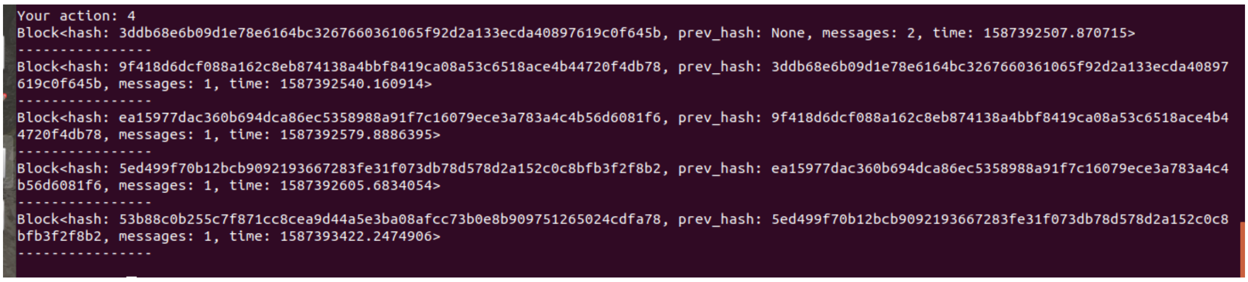

Activities
Terminal
Mon 7:52 PM
rohiniraja@rohini-raja: ~/Desktop/pysimplechain-master
File Edit View Search Terminal Help

```
rohiniraja@rohini-raja:~/Desktop/pysimplechain-master$ vi simple_chain.py
rohiniraja@rohini-raja:~/Desktop/pysimplechain-master$ python3 simple_chain.py

                Basic implementation of a Blockchain. Changes are immutable.

                Action set:
                        - add message to the existing block  (1)
                        - add existing block to the chain    (2)
                        - show a block (index will be asked) (3)
                        - show the whole chain               (4)
                        - validate the chain integrity       (5)
                        - exit the program                   (6)

                The validate action will kill the program if the integrity if the chain
                is compromised.


Your action: 4

Your action: 3
Provide the index: 0

Your action: 1
Enter your data:3579 Rohini

Your action: 1
Enter your data:3056 Andrea

Your action: 2

Your action: 3
Provide the index: 0
Block<hash: 3ddb68e6b09d1e78e6164bc3267660361065f92d2a133ecda40897619c0f645b, prev_hash: None, messages: 2, time: 1587392507.870715>

Your action: 4
Block<hash: 3ddb68e6b09d1e78e6164bc3267660361065f92d2a133ecda40897619c0f645b, prev_hash: None, messages: 2, time: 1587392507.870715>
----------------
```

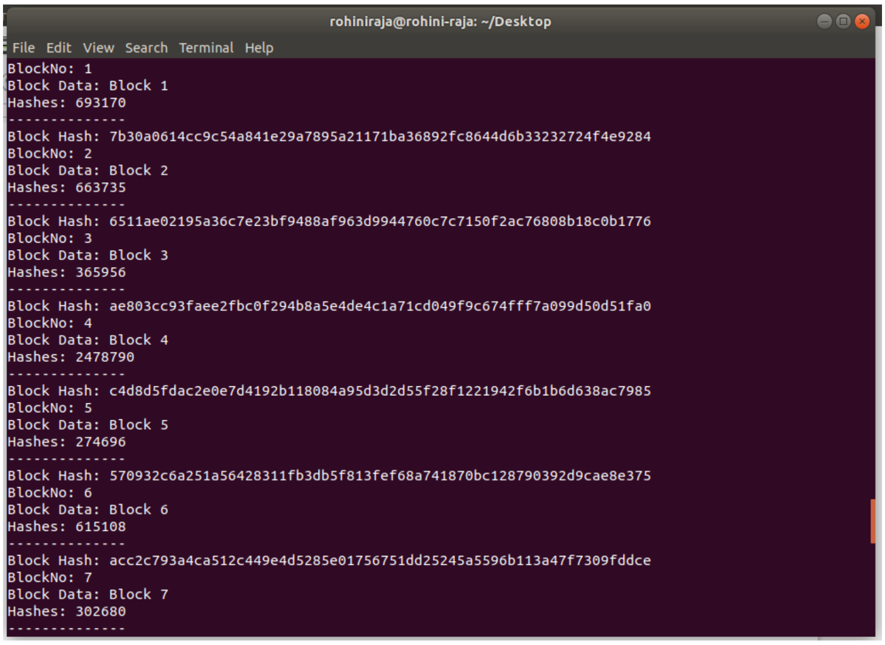

Activities
Terminal
Mon 7:53 PM
rohiniraja@rohini-raja: ~/Desktop/pysimplechain-master
File Edit View Search Terminal Help

```
Your action: 1
Enter your data:Programming language

Your action: 2

Your action: 4
Block<hash: 3ddb68e6b09d1e78e6164bc3267660361065f92d2a133ecda40897619c0f645b, prev_hash: None, messages: 2, time: 1587392507.870715>
----------------
Block<hash: 9f418d6dcf088a162c8eb874138a4bbf8419ca08a53c6518ace4b44720f4db78, prev_hash: 3ddb68e6b09d1e78e6164bc3267660361065f92d2a133ecda40897
619c0f645b, messages: 1, time: 1587392540.160914>
----------------
Block<hash: ea15977dac360b694dca86ec5358988a91f7c16079ece3a783a4c4b56d6081f6, prev_hash: 9f418d6dcf088a162c8eb874138a4bbf8419ca08a53c6518ace4b4
4720f4db78, messages: 1, time: 1587392579.8886395>
----------------

Your action: 1
Enter your data:testing

Your action: 5
Integrity validated.

Your action: 2

Your action: 4
Block<hash: 3ddb68e6b09d1e78e6164bc3267660361065f92d2a133ecda40897619c0f645b, prev_hash: None, messages: 2, time: 1587392507.870715>
----------------
Block<hash: 9f418d6dcf088a162c8eb874138a4bbf8419ca08a53c6518ace4b44720f4db78, prev_hash: 3ddb68e6b09d1e78e6164bc3267660361065f92d2a133ecda40897
619c0f645b, messages: 1, time: 1587392540.160914>
----------------
Block<hash: ea15977dac360b694dca86ec5358988a91f7c16079ece3a783a4c4b56d6081f6, prev_hash: 9f418d6dcf088a162c8eb874138a4bbf8419ca08a53c6518ace4b4
4720f4db78, messages: 1, time: 1587392579.8886395>
----------------
Block<hash: 5ed499f70b12bcb9092193667283fe31f073db78d578d2a152c0c8bfb3f2f8b2, prev_hash: ea15977dac360b694dca86ec5358988a91f7c16079ece3a783a4c4
b56d6081f6, messages: 1, time: 1587392605.6834054>
----------------

Your action: 
```

This lists all the blocks and its attributes such as timestamp that has been created.

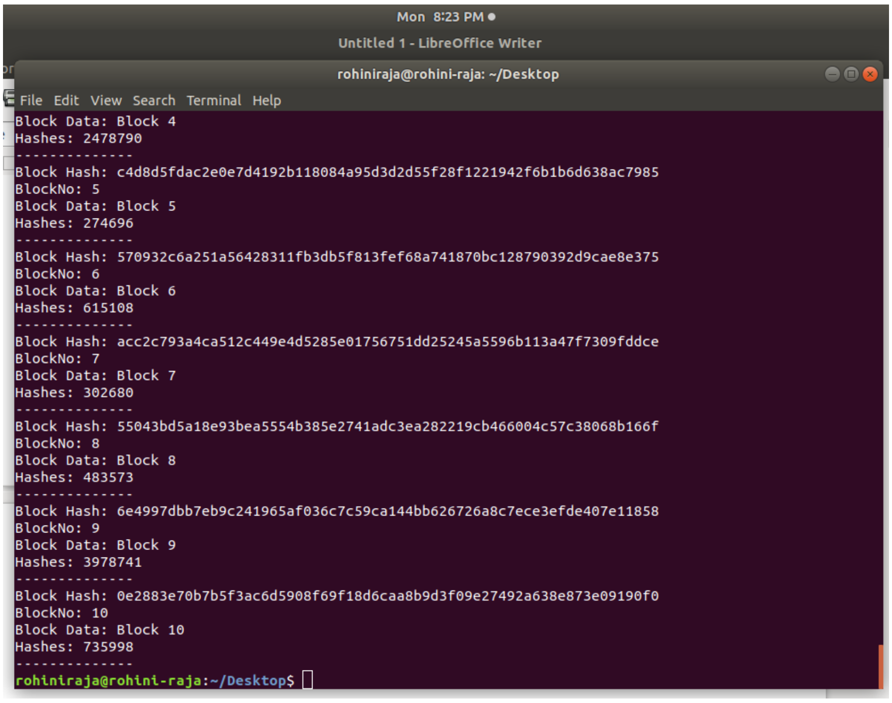


```
Your action: 4
Block<hash: 3ddb68e6b09d1e78e6164bc3267660361065f92d2a133ecda40897619c0f645b, prev_hash: None, messages: 2, time: 1587392507.870715>
----------------
Block<hash: 9f418d6dcf088a162c8eb874138a4bbf8419ca08a53c6518ace4b44720f4db78, prev_hash: 3ddb68e6b09d1e78e6164bc3267660361065f92d2a133ecda40897
619c0f645b, messages: 1, time: 1587392540.160914>
----------------
Block<hash: ea15977dac360b694dca86ec5358988a91f7c16079ece3a783a4c4b56d6081f6, prev_hash: 9f418d6dcf088a162c8eb874138a4bbf8419ca08a53c6518ace4b4
4720f4db78, messages: 1, time: 1587392579.8886395>
----------------
Block<hash: 5ed499f70b12bcb9092193667283fe31f073db78d578d2a152c0c8bfb3f2f8b2, prev_hash: ea15977dac360b694dca86ec5358988a91f7c16079ece3a783a4c4
b56d6081f6, messages: 1, time: 1587392605.6834054>
----------------
Block<hash: 53b88c0b255c7f871cc8cea9d44a5e3ba08afcc73b0e8b909751265024cdfa78, prev_hash: 5ed499f70b12bcb9092193667283fe31f073db78d578d2a152c0c8
bfb3f2f8b2, messages: 1, time: 1587393422.2474906>
----------------
```



rohiniraja@rohini-raja: ~/Desktop
File Edit View Search Terminal Help

```
BlockNo: 1
Block Data: Block 1
Hashes: 693170
--------------
Block Hash: 7b30a0614cc9c54a841e29a7895a21171ba36892fc8644d6b33232724f4e9284
BlockNo: 2
Block Data: Block 2
Hashes: 663735
--------------
Block Hash: 6511ae02195a36c7e23bf9488af963d9944760c7c7150f2ac76808b18c0b1776
BlockNo: 3
Block Data: Block 3
Hashes: 365956
--------------
Block Hash: ae803cc93faee2fbc0f294b8a5e4de4c1a71cd049f9c674fff7a099d50d51fa0
BlockNo: 4
Block Data: Block 4
Hashes: 2478790
--------------
Block Hash: c4d8d5fdac2e0e7d4192b118084a95d3d2d55f28f1221942f6b1b6d638ac7985
BlockNo: 5
Block Data: Block 5
Hashes: 274696
--------------
Block Hash: 570932c6a251a56428311fb3db5f813fef68a741870bc128790392d9cae8e375
BlockNo: 6
Block Data: Block 6
Hashes: 615108
--------------
Block Hash: acc2c793a4ca512c449e4d5285e01756751dd25245a5596b113a47f7309fddce
BlockNo: 7
Block Data: Block 7
Hashes: 302680
--------------
```



Mon 8:23 PM ●
Untitled 1 - LibreOffice Writer
rohiniraja@rohini-raja: ~/Desktop
File Edit View Search Terminal Help

```
Block Data: Block 4
Hashes: 2478790
--------------
Block Hash: c4d8d5fdac2e0e7d4192b118084a95d3d2d55f28f1221942f6b1b6d638ac7985
BlockNo: 5
Block Data: Block 5
Hashes: 274696
--------------
Block Hash: 570932c6a251a56428311fb3db5f813fef68a741870bc128790392d9cae8e375
BlockNo: 6
Block Data: Block 6
Hashes: 615108
--------------
Block Hash: acc2c793a4ca512c449e4d5285e01756751dd25245a5596b113a47f7309fddce
BlockNo: 7
Block Data: Block 7
Hashes: 302680
--------------
Block Hash: 55043bd5a18e93bea5554b385e2741adc3ea282219cb466004c57c38068b166f
BlockNo: 8
Block Data: Block 8
Hashes: 483573
--------------
Block Hash: 6e4997dbb7eb9c241965af036c7c59ca144bb626726a8c7ece3efde407e11858
BlockNo: 9
Block Data: Block 9
Hashes: 3978741
--------------
Block Hash: 0e2883e70b7b5f3ac6d5908f69f18d6caa8b9d3f09e27492a638e873e09190f0
BlockNo: 10
Block Data: Block 10
Hashes: 735998
--------------
rohiniraja@rohini-raja:~/Desktop$
```

COMPARATIVE ANALSIS AND INFERENCE:

1) Enhanced Security-Blockchain technology solves the major key issues such as Security and trust in a network.
2) Data Durability-The data which is stored is durable and it is immutable.
3) Better Transparency – With blockchain, we can go for a complete decentralized network and there will be no necessity for a centralized authority – improving the transparency.
4) Reduced costs -There is less interaction needed when it comes to validating a transaction in blockchain.
5) Improved Speed and Highly Efficient - The streamlining and automation of processes in blockchain mean that everything becomes highly efficient and fast.

These features all together implies the importance of blockchain Technology and hereby, a simple blockchain has been implemented using python and all the results are summarized with appropriate screenshots.