**BITCOIN**

Bitcoin is a decentralized digital currency that you can buy, sell and exchange directly, without an intermediary like a bank. Bitcoin’s creator, Satoshi Nakamoto, originally described the need for “an electronic payment system based on cryptographic proof instead of trust.”

Each and every Bitcoin transaction that’s ever been made exists on a public ledger accessible to everyone, making transactions hard to reverse and difficult to fake. That’s by design: Core to their decentralized nature, Bitcoins aren’t backed by the government or any issuing institution, and there’s nothing to guarantee their value besides the proof baked in the heart of the system.

Bitcoin is built on a distributed digital record called a blockchain. As the name implies, blockchain is a linked body of data, made up of units called blocks that contain information about each and every transaction, including date and time, total value, buyer and seller, and a unique identifying code for each exchange. Entries are strung together in chronological order, creating a digital chain of blocks.

“Once a block is added to the blockchain, it becomes accessible to anyone who wishes to view it, acting as a public ledger of cryptocurrency transactions,” says Stacey Harris, consultant for Pelicoin, a network of cryptocurrency ATMs.

Blockchain is decentralized, which means it’s not controlled by any one organization. “It’s like a Google Doc that anyone can work on,” says Buchi Okoro, CEO and co-founder of African cryptocurrency exchange Quidax. “Nobody owns it, but anyone who has a link can contribute to it. And as different people update it, your copy also gets updated.”

While the idea that anyone can edit the blockchain might sound risky, it’s actually what makes Bitcoin trustworthy and secure. In order for a transaction block to be added to the Bitcoin blockchain, it must be verified by the majority of all Bitcoin holders, and the unique codes used to recognize users’ wallets and transactions must conform to the right encryption pattern.

These codes are long, random numbers, making them incredibly difficult to fraudulently produce. In fact, a fraudster guessing the key code to your Bitcoin wallet has roughly the same odds as someone winning a Powerball lottery nine times in a row, according to Bryan Lotti of Crypto Aquarium. This level of statistical randomness blockchain verification codes, which are needed for every transaction, greatly reduces the risk anyone can make fraudulent Bitcoin transactions.

**BLOCK (BITCOIN BLOCK)**

Blocks are files where data pertaining to the Bitcoin network are permanently recorded. A block records some or all of the most recent Bitcoin transactions that have not yet entered any prior blocks. Thus, a block is like a page of a ledger or record book. Each time a block is ‘completed’, it gives way to the next block in the blockchain. A block is thus a permanent store of records which, once written, cannot be altered or removed.

**BLOCKCHAIN**

A blockchain is a distributed database that is shared among the nodes of a computer network. As a database, a blockchain stores information electronically in digital format. Blockchains are best known for their crucial role in cryptocurrency systems, such as Bitcoin, for maintaining a secure and decentralized record of transactions. The innovation with a blockchain is that it guarantees the fidelity and security of a record of data and generates trust without the need for a trusted third party.

One key difference between a typical database and a blockchain is the way the data is structured. A blockchain collects information together in groups, known as "blocks" that hold sets of information. Blocks have certain storage capacities and, when filled, are closed and linked to the previously filled block, forming a chain of data known as the “blockchain.” All new information that follows that freshly added block is compiled into a newly formed block that will then also be added to the chain once filled.

A database usually structures its data into tables whereas a blockchain, like its name implies, structures its data into chunks (blocks) that are strung together. This data structure inherently makes an irreversible timeline of data when implemented in a decentralized nature. When a block is filled it is set in stone and becomes a part of this timeline. Each block in the chain is given an exact timestamp when it is added to the chain.

**Program:**

**Implement the creation of Bitcoin Block/Blockchain (Genesis Block)**

#defining the list/chain

blockchain = []

#getting the last value/transaction

def get\_last\_value():

return(blockchain[-1])

#adding transaction

#now we have sender, recipient and amount

def add\_value(sender, recipient, amount=1.0):

transaction = {'sender': sender,

'recipient': recipient,

'amount': amount}

blockchain.append(transaction)

#getting the details of transaction by entering to the prompt

def get\_transaction\_value():

tx\_sender = input('Enter the sender: ')

tx\_recipient = input('Enter the recipient of the transaction: ')

tx\_amount = float(input('Enter your transaction amount: '))

return tx\_sender, tx\_recipient, tx\_amount

#printing the blockchain

def print\_block():

for block in blockchain:

print("Here is your block")

print(block)

#the code will keep repeating for more transaction

#until the user answer is no

again = True

while again == True:

tx = get\_transaction\_value()

s, r, a = tx

add\_value(s, r, a)

print(blockchain)

more = input("add more block (Y/N)? ")

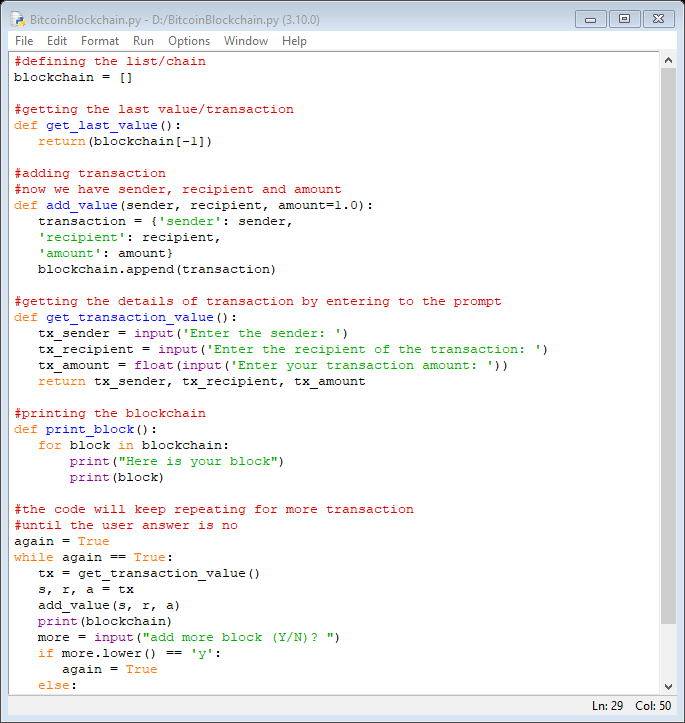
if more.lower() == 'y':

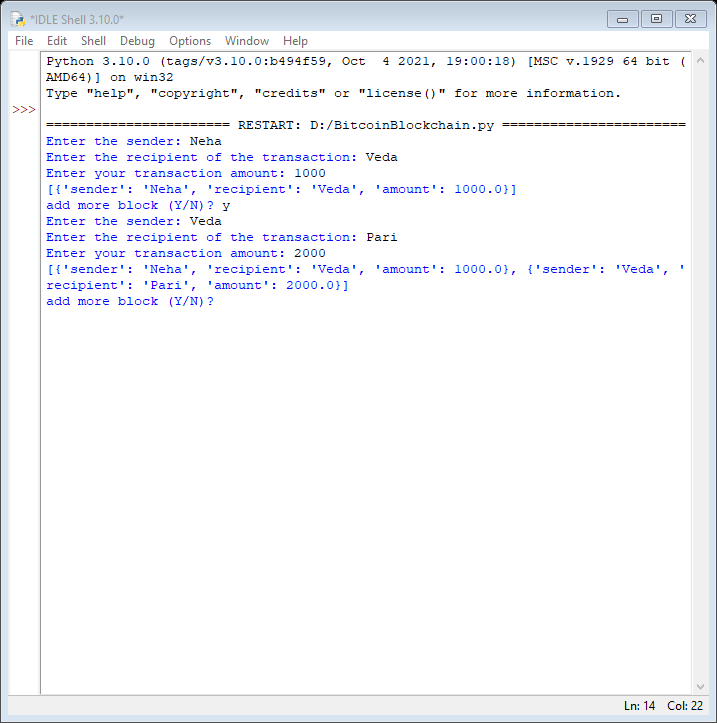
again = True

else:

again = False

**Output:**

****

****

**Program:**

**Implement the creation of a Blockchain (Adding the blocks to the chain and validating)**import hashlib as hasher

import datetime as date

# Define what a Snakecoin block is

class Block:

def \_\_init\_\_(self, index, timestamp, data, previous\_hash):

self.index = index

self.timestamp = timestamp

self.data = data

self.previous\_hash = previous\_hash

self.hash = self.hash\_block()

def \_\_repr\_\_(self):

return " index %04d: \n Time %s, \n Data %s : \n Previous hash %s" % (self.index,str(self.timestamp),str(self.data),str(self.previous\_hash))

def hash\_block(self):

sha = hasher.sha256()

sha.update(repr(self).encode('ascii'))

return sha.hexdigest()

# Generate genesis block

def create\_genesis\_block():

# Manually construct a block with

# index zero and arbitrary previous hash

return Block(0, date.datetime.now(), "Genesis Block", "0")

# Create the blockchain and add the genesis block

blockchain = [create\_genesis\_block()]

previous\_block = blockchain[0]

# Show the blockchain

blockchain

# Generate all later blocks in the blockchain

def next\_block(last\_block):

this\_index = last\_block.index + 1

this\_timestamp = date.datetime.now()

this\_data = "Hey! I'm block " + str(this\_index)

this\_hash = last\_block.hash

return Block(this\_index, this\_timestamp, this\_data, this\_hash)

# How many blocks should we add to the chain

# after the genesis block

num\_of\_blocks\_to\_add = 5

# Add blocks to the chain

for i in range(0, num\_of\_blocks\_to\_add):

block\_to\_add = next\_block(previous\_block)

blockchain.append(block\_to\_add)

previous\_block = block\_to\_add

# Tell everyone about it!

print(repr(block\_to\_add))

print("-------------------------")

#index,time,data,previous has

from warnings import warn

def validate\_blockchain(in\_blockchain):

for current\_position in range(1, len(in\_blockchain)):

previous\_position = current\_position - 1

if in\_blockchain[previous\_position].hash\_block() == in\_blockchain[current\_position].previous\_hash:

print('Block %d is valid' % current\_position)

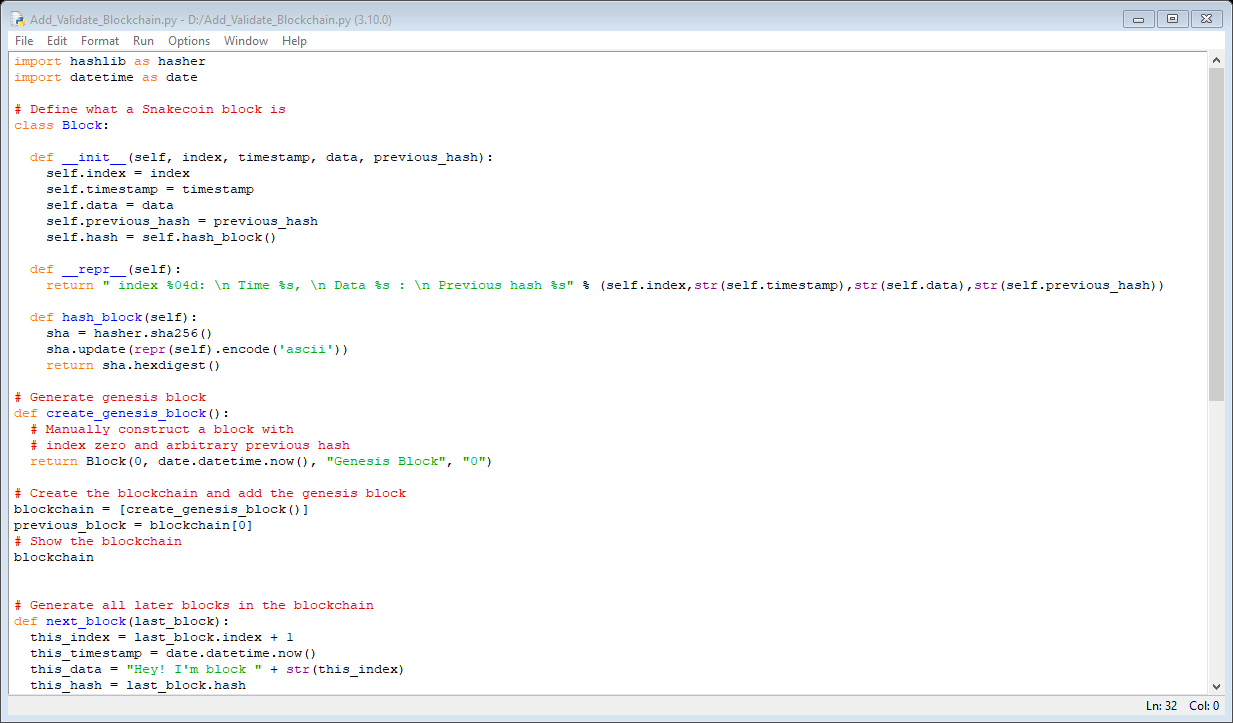
else:

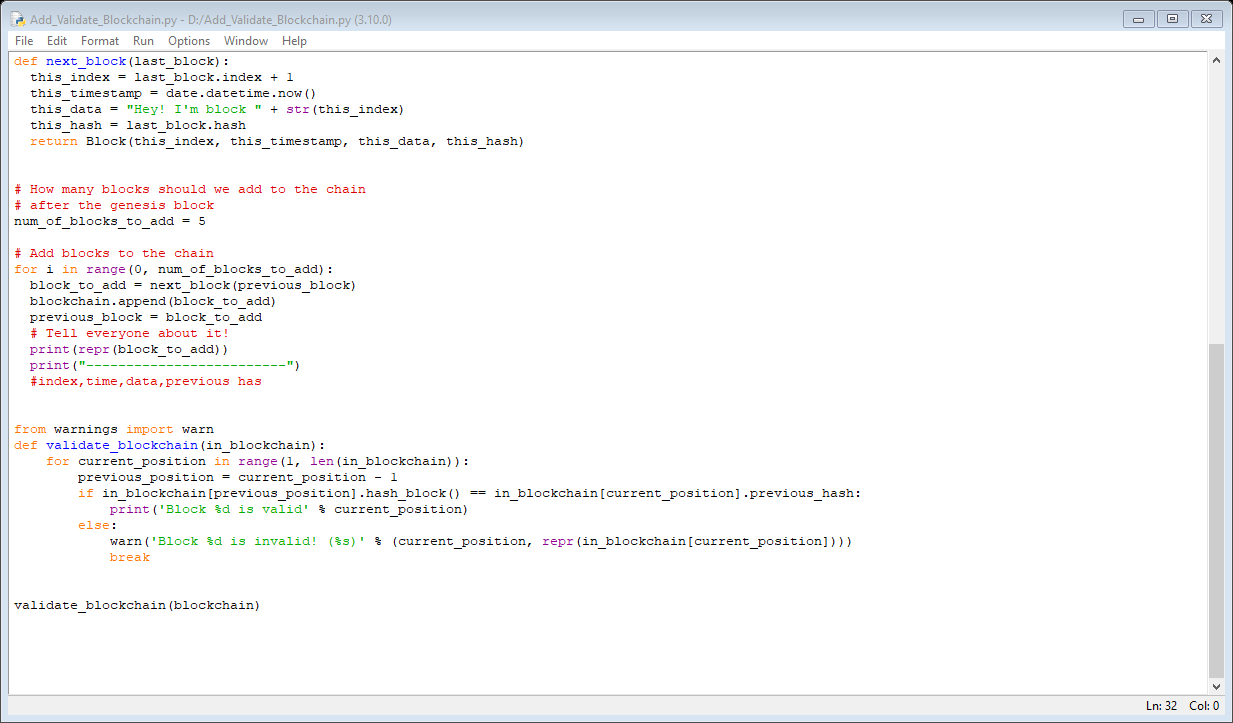
warn('Block %d is invalid! (%s)' % (current\_position, repr(in\_blockchain[current\_position])))

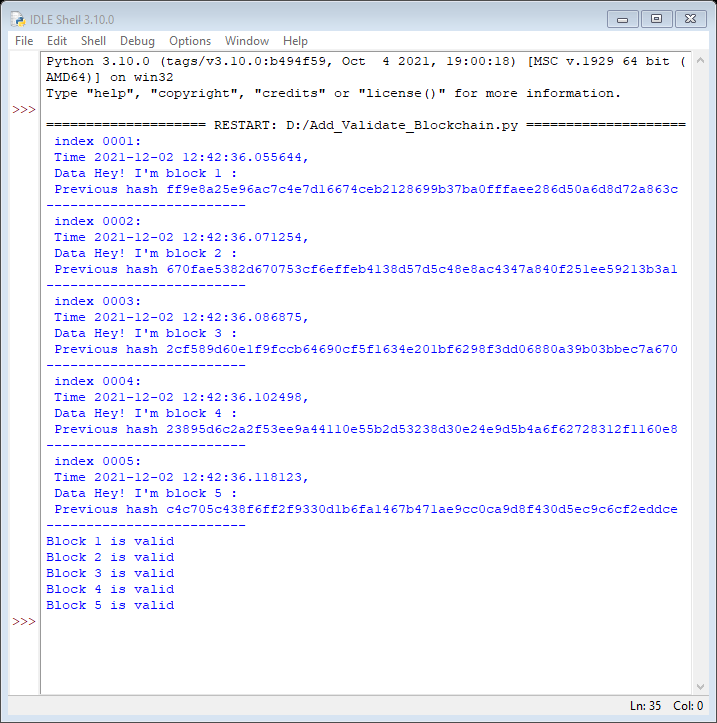
break

validate\_blockchain(blockchain)

**Output:**

****

****

****

**PRIVATE BLOCKCHAIN USING GETH**

**(Implement the creation of a public/private Blockchain)**

Ethereum node is any device that is running the Ethereum protocol (blockchain).

When we connect to the Ethereum protocol we are on the Ethereum blockchain network.

By running an Ethereum node we can connect to other nodes in the network, have direct access to the blockchain, and even do things like mine blocks, send transactions, and deploy smart contracts.

**Step 1**

Download and Install NodeJs

<https://nodejs.org/en/download/>

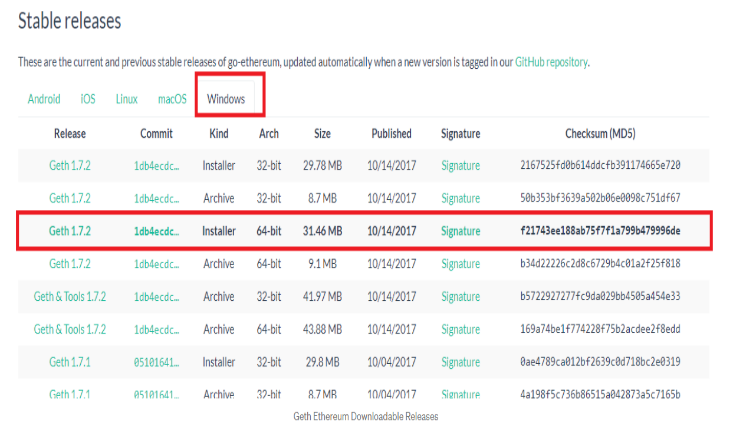
**Step 2**

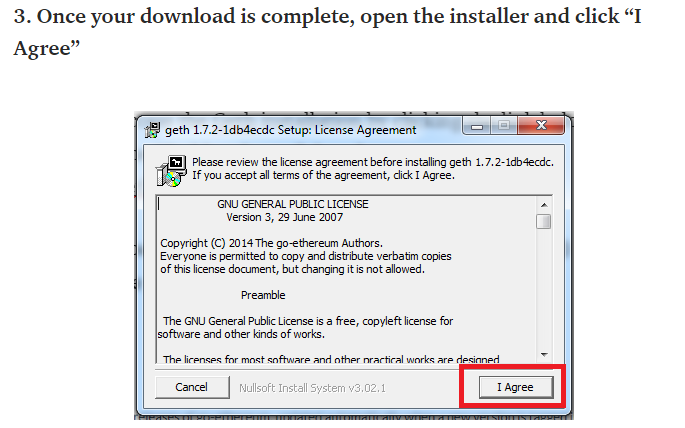
* **Installation:**

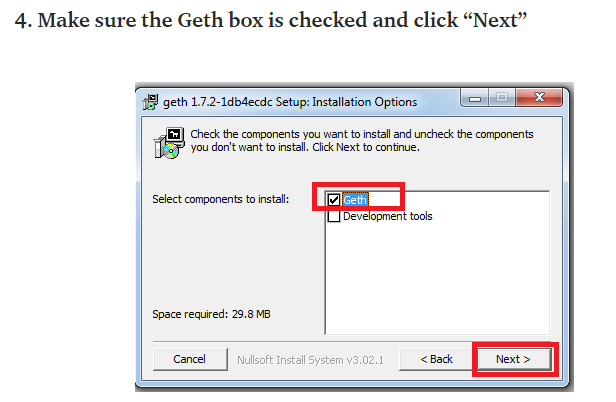
1. Visit the Go Ethereum website and install Geth

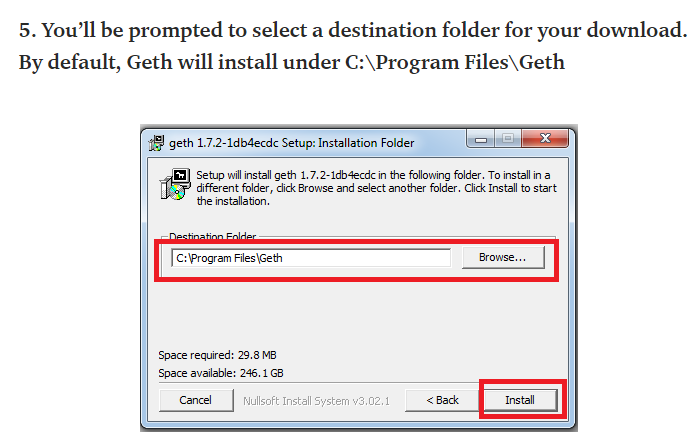
Visit here: <https://geth.ethereum.org/downloads/>

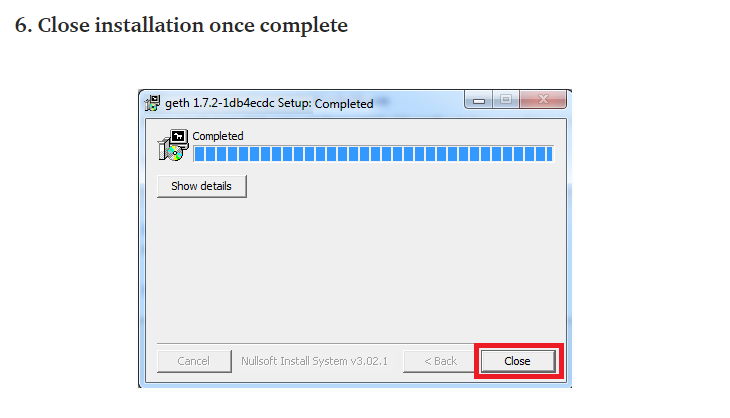
1. Download the latest release of Geth for Windows, make sure you download the 64-bit version











**Step 3**

**Establishing Our Own Private Ethereum Network**

1. Create a new folder on your desktop called “Private-chain”
2. Open command prompt in this folder and create a data directory folder for our chaindata by typing “mkdir chaindata
3. Next, we need to create and save our genesis.json block in our Private-chain folder, as the genesis block will be used to initialize our private network and store data in the data directory folder “chaindata” .Initially “chaindata” folder is NULL. Once all steps done, observe the structure of “chaindata” folder.

Open up notepad, copy & paste the code below into a new file called “genesis.json” and save this file in our Private-chain folder.

{

"config": {

"chainId": 4777,

"homesteadBlock": 0,

"eip150Block": 0,

"eip155Block": 0,

"eip158Block": 0

},

"alloc" : {},

"difficulty" : "0x400",

"extraData" : "",

"gasLimit" : "0x7A1200",

"parentHash" :

"0x0000000000000000000000000000000000000000000000000000000000000000",

"timestamp" : "0x00"

}

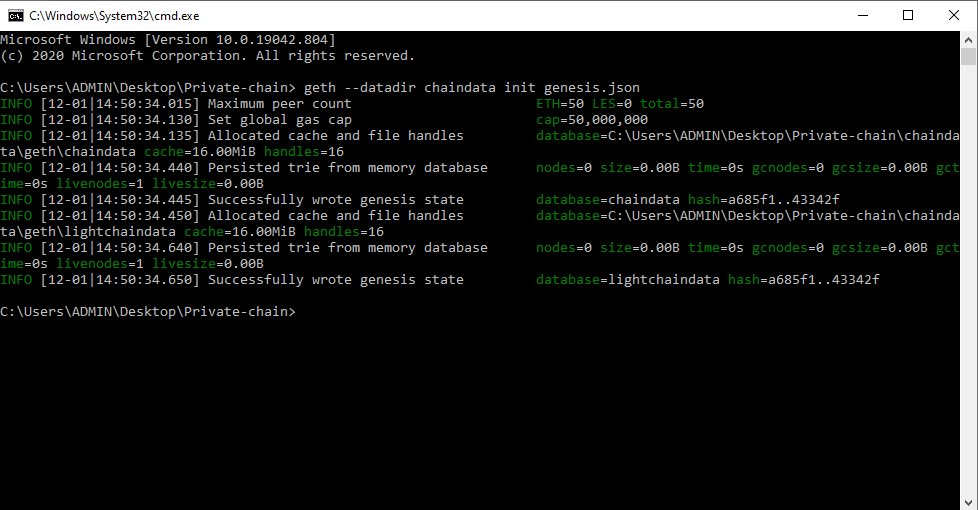
**Step 4**

**Start the Ethereum peer node (Start the Blockchain)**

Run the command:

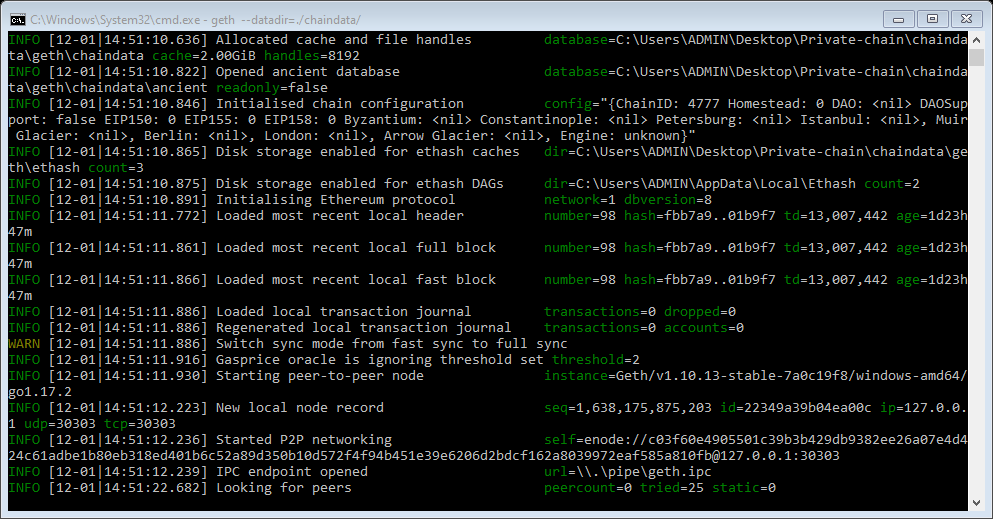
geth --datadir chaindata init genesis.json

It will Intialize geth into chaindata



**Now we can start Geth and connect to our own private chain**

geth --datadir=./chaindata/



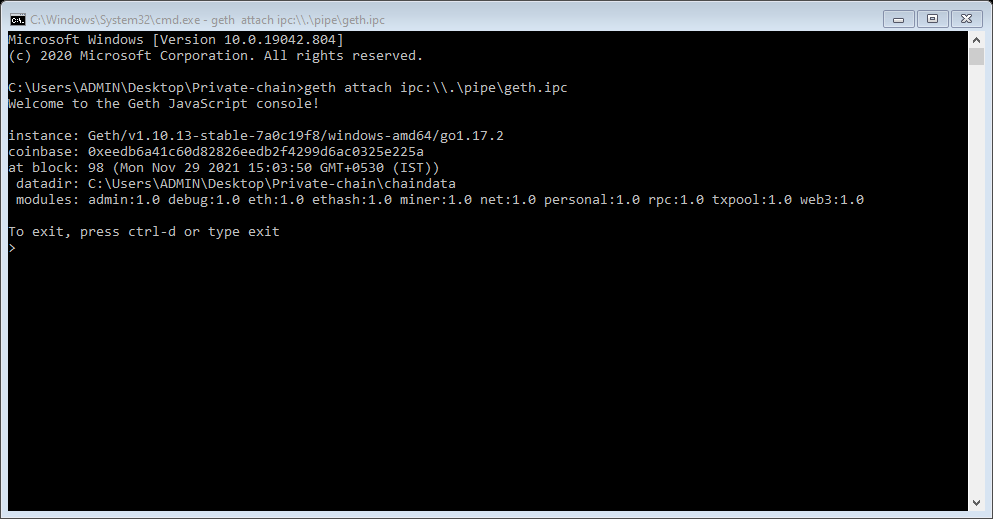
**Step 5**

**Minimize the terminal and open a new terminal**

**IPC to interact with Geth:**

**geth attach ipc:\\.\pipe\geth.ipc**

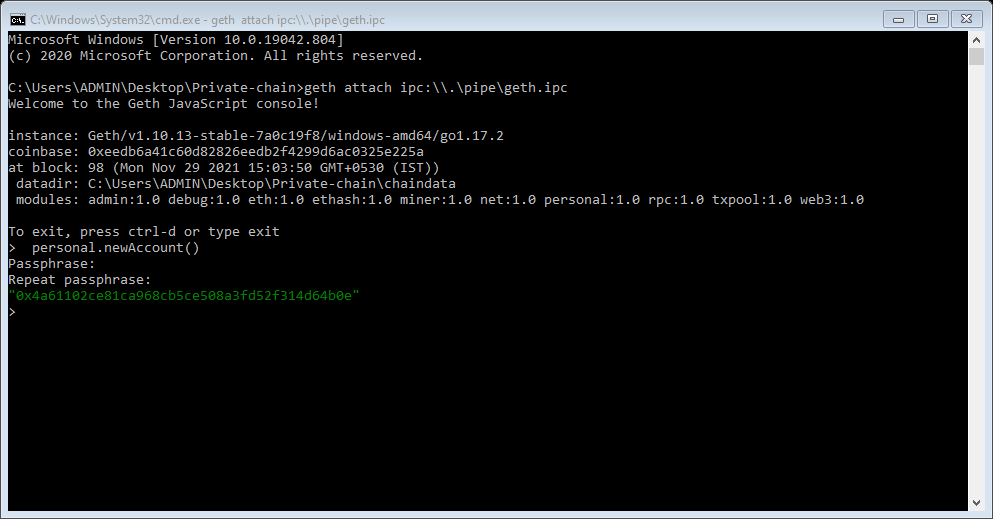
(Run this command on new terminal)



**Create Account**

personal.newAccount()

(Run this command on terminal)

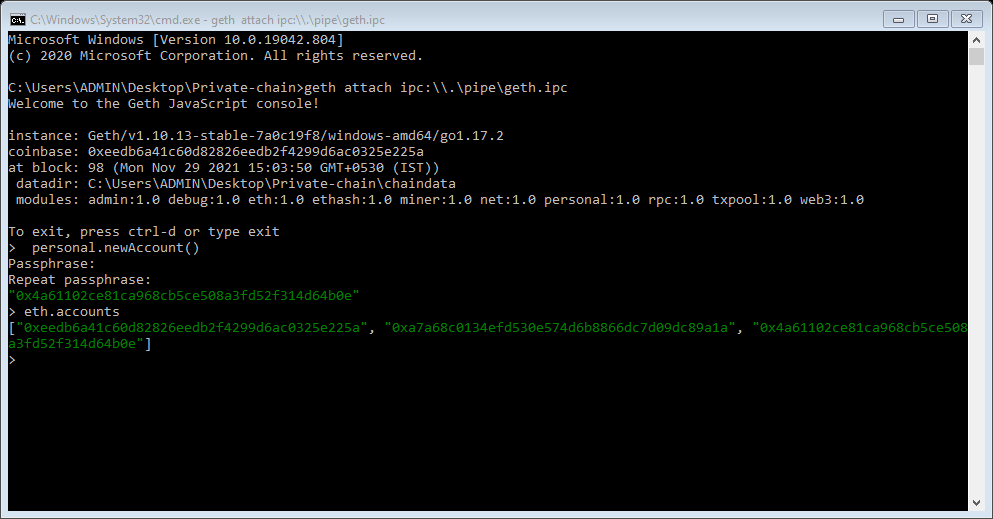


(you can create multiple accounts. It will ask password. Note the password. This password required while you unlock account)

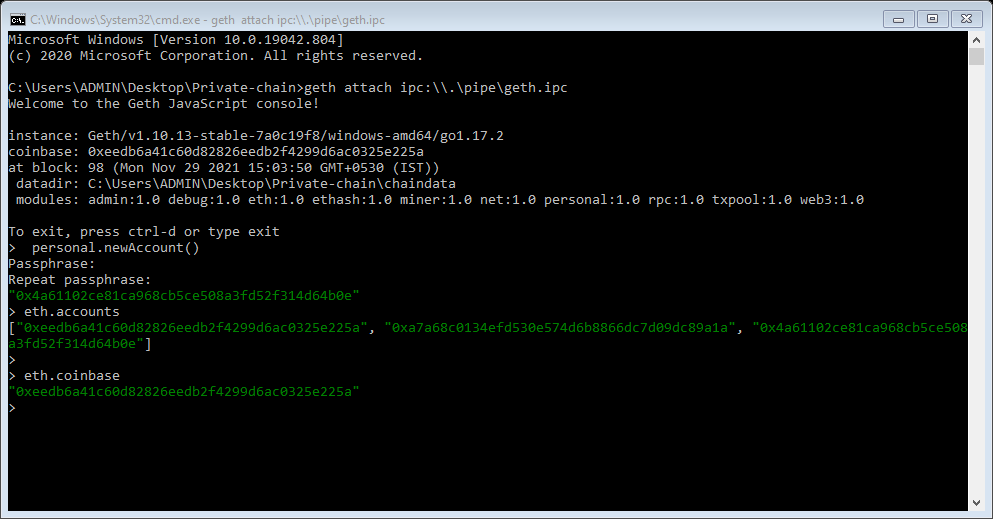
eth.accounts

(Run this command on terminal)

It will show number of accounts created



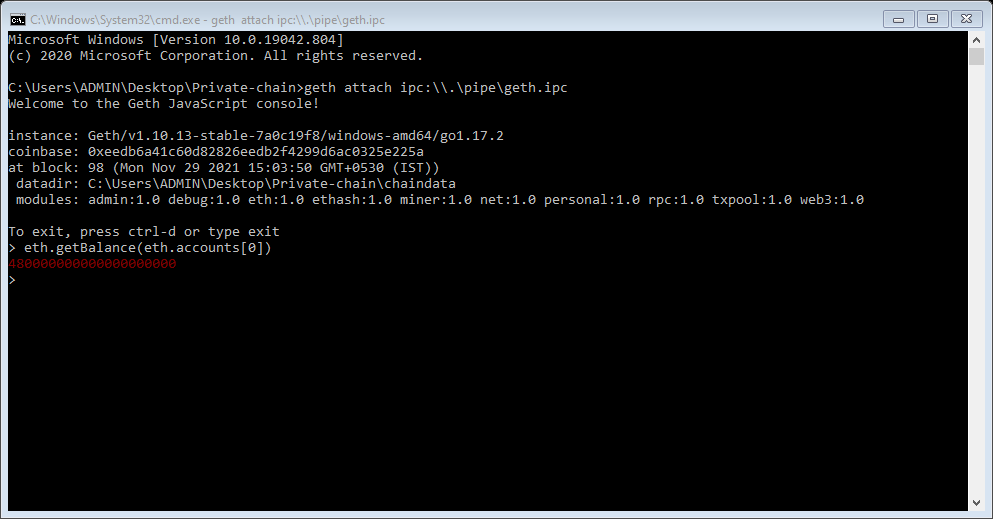
eth.coinbase



eth.getBalance(eth.accounts[0])

(Run this command on terminal)

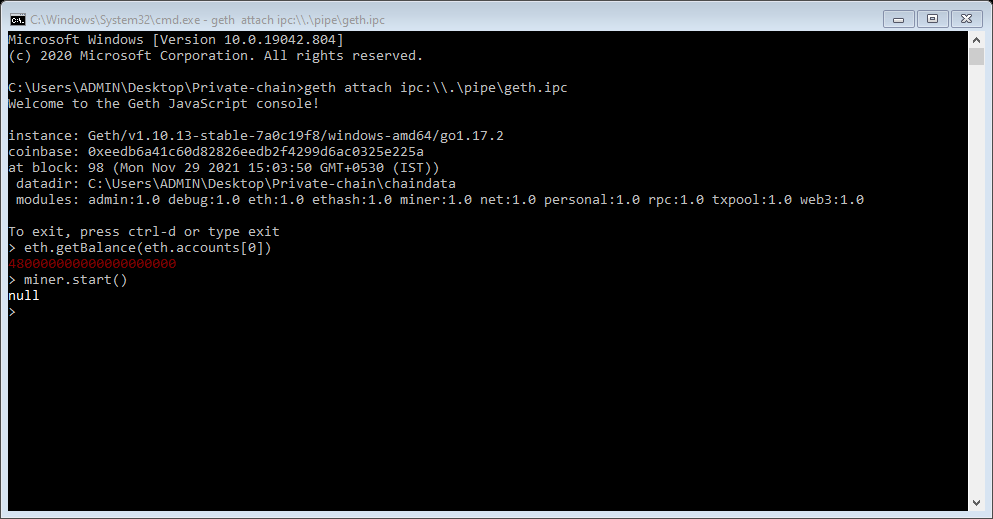
It will show balance of account.



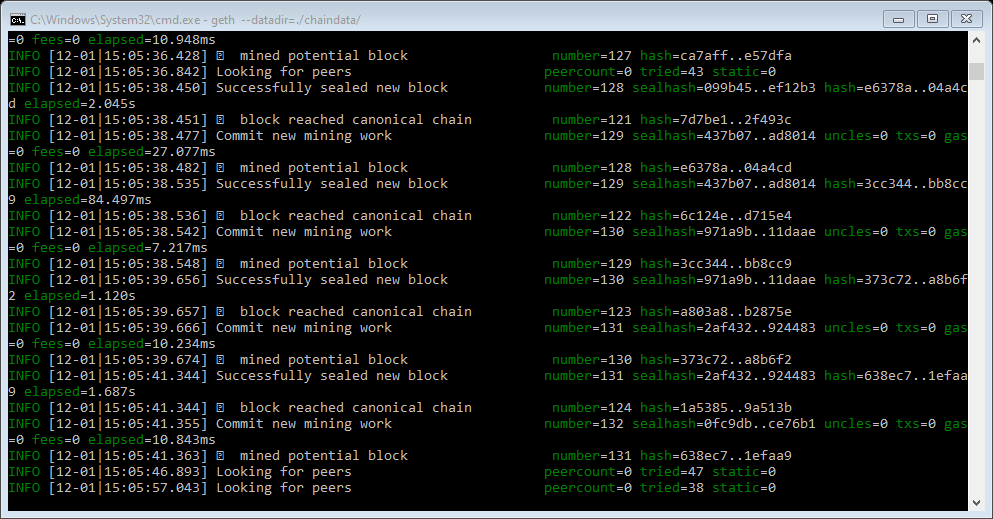
miner.start()

(Run this command on terminal)

Mining is the process of creating a block of transactions to be added to the Ethereum blockchain.

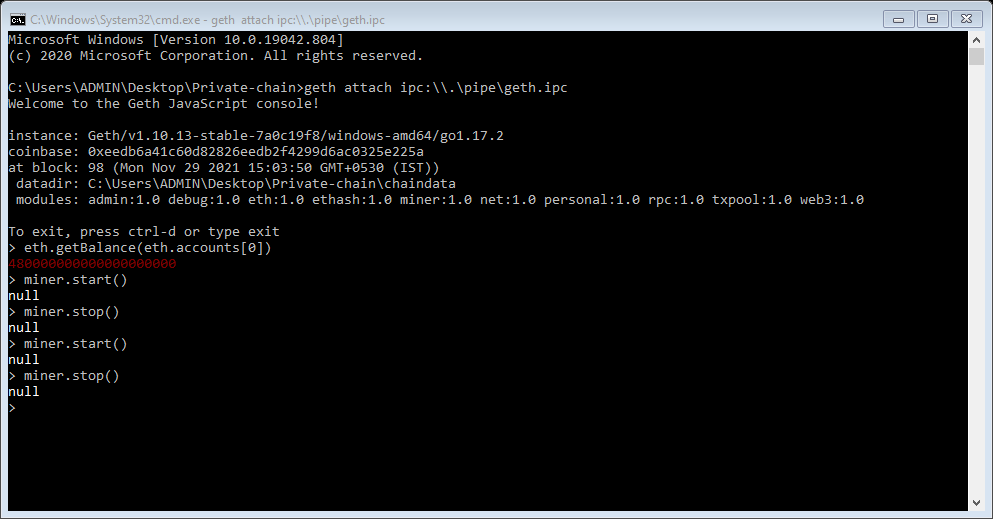


(Now observe the terminal which we have minimized.



Wait till you get msg on terminal that successfully sealed new block. Then run following command)

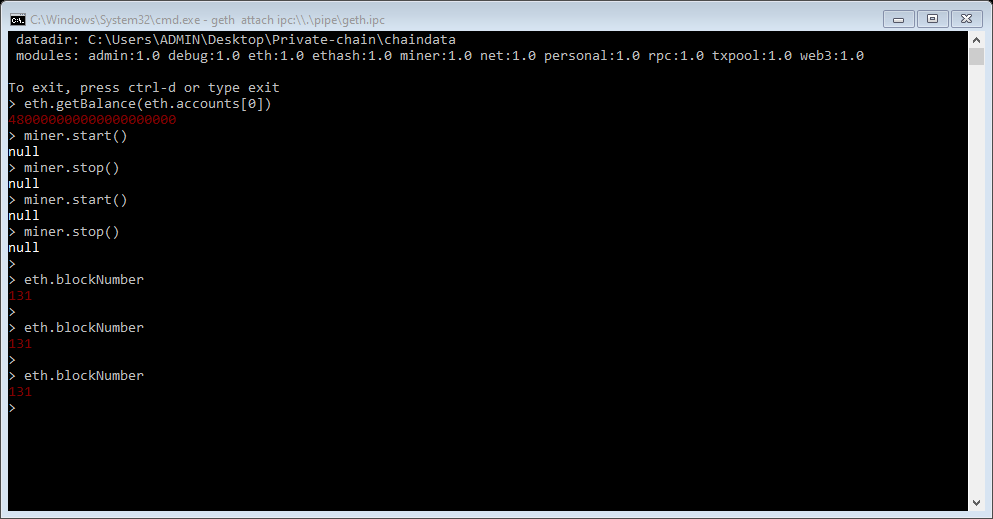
miner.stop()



eth.blockNumber

(Run this command on terminal)

It will show you blocknumber.



**Now we will perform transaction**

To perform transaction, we are having one account. Now we will create another account.

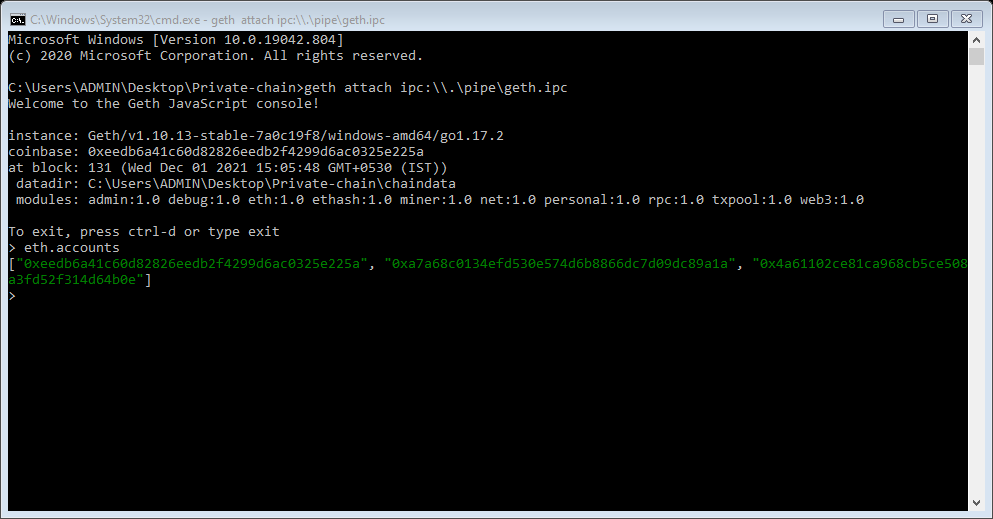
personal.newAccount()

(Run this command on terminal it will create new account)

eth.accounts

(Run this command on terminal)

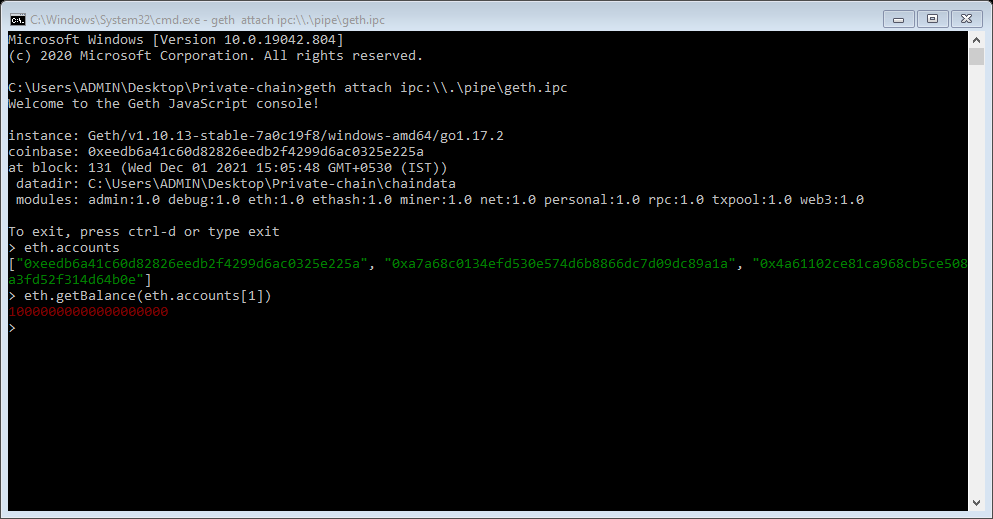
It will show number of accounts created



eth.getBalance(eth.accounts[1])

(Run this command on terminal)

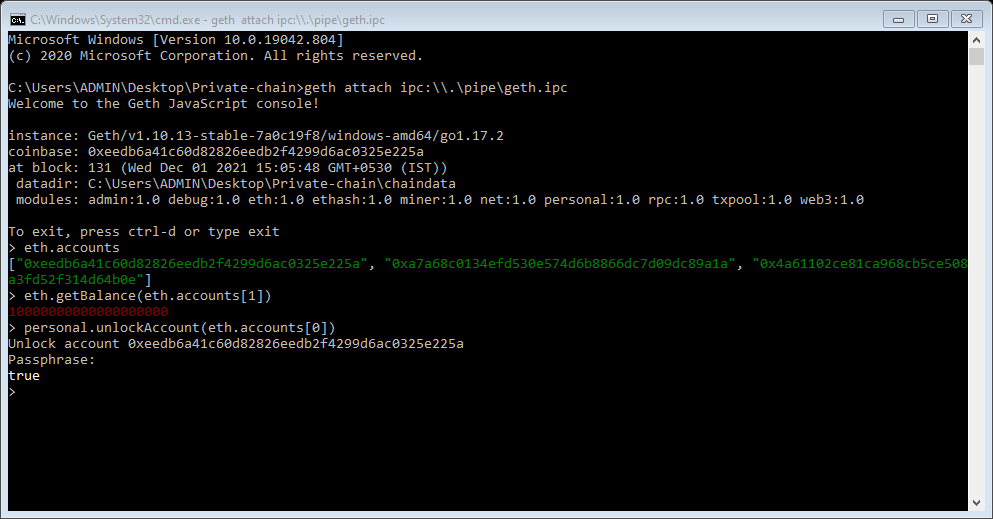
It will show balance of account.



personal.unlockAccount(eth.accounts[0])

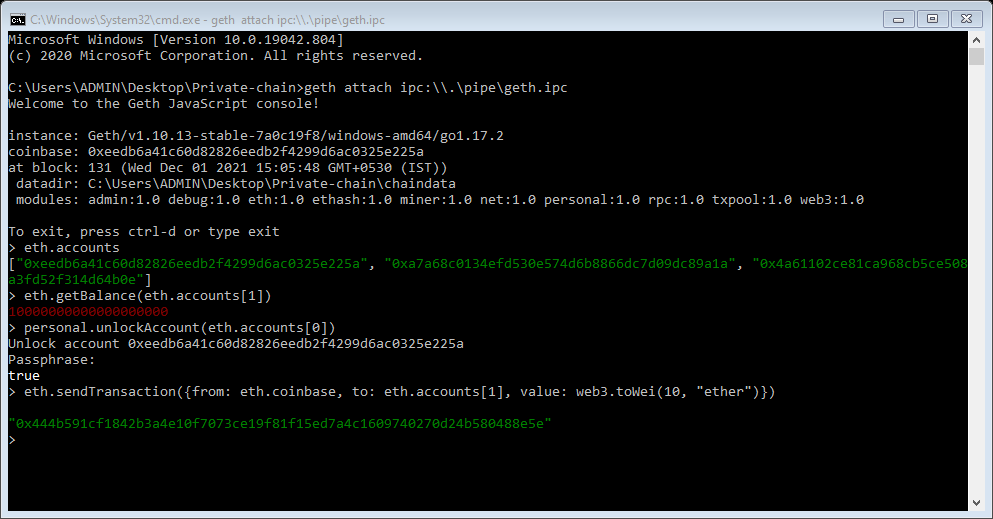
(Run this command on terminal)

It will unlock the block which was sealed previously, it requires password)



**Transaction:**

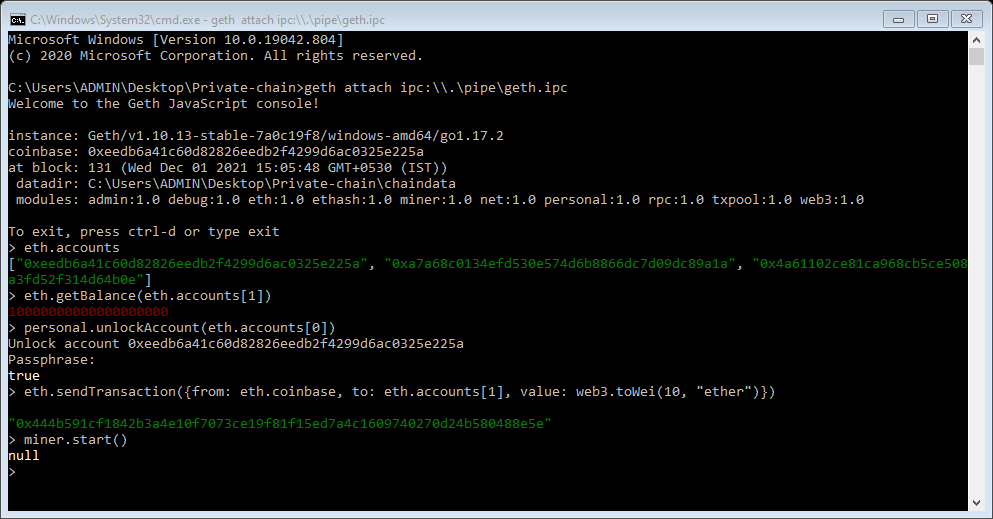
eth.sendTransaction({from: eth.coinbase, to: eth.accounts[1], value: web3.toWei(10, "ether")})



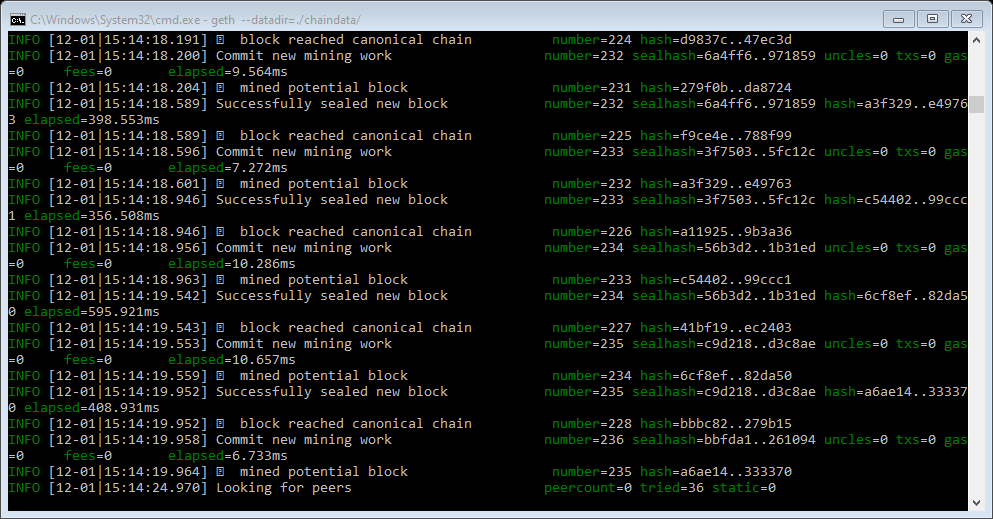
(start mining and stop)

miner.start()

(Run this command on terminal)

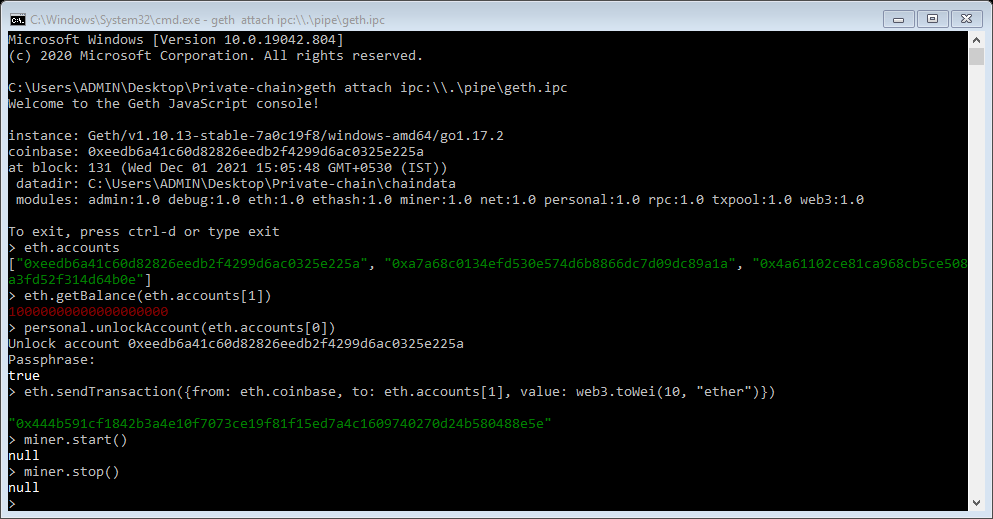


(Now observe the terminal which we have minimized.



Wait till you get msg on terminal that successfully sealed. Then run following command)

miner.stop()



eth.getBalance(eth.accounts[1])

(Run this command on terminal)

It will show balance of account.

