```python
import datetime              #Time stamp
import hashlib               #hash the blocks
from os import system


# inside working

#  Class for block
class Block:
    # constructor containing definition of parameters (sender name, transaction amount, receiver name, message and chain)
    def __init__(self, sender, transaction, receiver, message = ""):
       self.sender = sender
       self.transaction = transaction
       self.receiver = receiver
       self.message = message
       self.chain = []

    # function to create a block of transaction, data type used: Dictionary(key/value pair)
    def create_block(self, p_hash = 0):
        b = { 'index': len(self.chain) + 1,                  #block number, for genesis block index = 0
              'timestamp': str(datetime.datetime.now()),     #time of generation of block
              'transaction': self.transaction,               #transaction data
              'p_hash': p_hash,                              #hash of previous block, for genesis block, p_hash = 0
              'hash': self.generate_hash(p_hash)             #hash of current block
        }
        #append block to the chain to create a link
        self.chain.append(b)
        return b

    #access previous block of the chain
    def get_prev_block(self):
        return self.chain[-1]

    # function to generate hash using SHA256, hash algorithm created using transaction, sender, receiver data
    def generate_hash(self, p_hash):
        hash = hashlib.sha256((self.sender + str(self.transaction) + self.receiver + self.message + str(p_hash)).encode()).hexdigest()
        return hash


# function to retrieve the chain formed after transaction and mining of block
```

```python
#datatype dictionary, contains the chain of block and the length of chain
def get_chain():
    response = { 'chain': S1.chain,
                 'length': len(S1.chain)}
    return response

# mining block, generating chain of blocks linked via previous hash datatype d
ictionary, contains the next block with following vaues
#asigning hash of prev block to p_hash of current block for linking
def mine_block():
    p_block = S1.get_prev_block()
    p_hash = p_block['hash']
    block = S1.create_block(p_hash)
    response = { 'index': block['index'],
                 'timestamp': block['timestamp'],
                 'p_hash':block['p_hash'],
                 'hash': block['hash']
                 }

    return response




# Main interface

# loop for interface
while True:

    # inputing data of transaction
    S = input('enter sender name: ')
    T = input('enter amount of coins to send: ')
    R = input('enter receiver name: ')
    print('** optional **')
    M = input('enter message: ')
    if M == "":
        z = input('send without any message ? Y/N ')
        if z == 'Y' or 'y':
            S1 = Block(S, T, R)
        else:
            M = input('enter message: ')

    #  object creation of class Block, object name S1
    S1 = Block(S, T, R, M)
    # creating block, access create block function of class Block
    S1.create_block()

    print('*****************************************************************
**************')
```

```python
    # to input choice to view screen
    print('input choice: ')

    # loop for choice
    while '1' or '2' or '3':
        choice = input('1. Sender\n2. Receiver\n3. Transaction\n4. Exit  ')
        if choice == '1':
            sender = {  'sender': S,
                        'Coins sent': T,
                        'cash in acc': 100 - int(T) }
            print('-------------------------------------------------------------------------')
            print(sender)
            print('-------------------------------------------------------------------------')

        elif choice == '2':
            receiver = {  'receiver': R,
                        'Coins received': T,
                        'cash in acc': 100 + int(T) }
            print('-------------------------------------------------------------------------')
            print(receiver)
            print('-------------------------------------------------------------------------')

        elif choice == '3':
            print('-------------------------------------------------------------------------')
            print('Transaction: ')
            # function call for mining the block
            for i in range(0,5):        #creating a chain of 5 blocks.
                mine_block()

            print(get_chain())
            print('-------------------------------------------------------------------------')

        else:
            break
    # choice loop ends here

    # to input choice for continuing transactions
    x = input('Do you wish to make more transactions?\nY/N  ')
    try:
        if x == 'Y':
            print('Starting new transaction....')
            print('clearing screen.......')
            system('cls')
            True
        elif x == 'N':
            break
        else:
            print('Invalid input')
    except:
        print('please turn on caps lock')

# loop for interface ends here
print('********************************************************************************
***************')
print('Thank you for using blockchain tech')
```