Report - Assignment 1B
Submitted By - Paritosh Goel

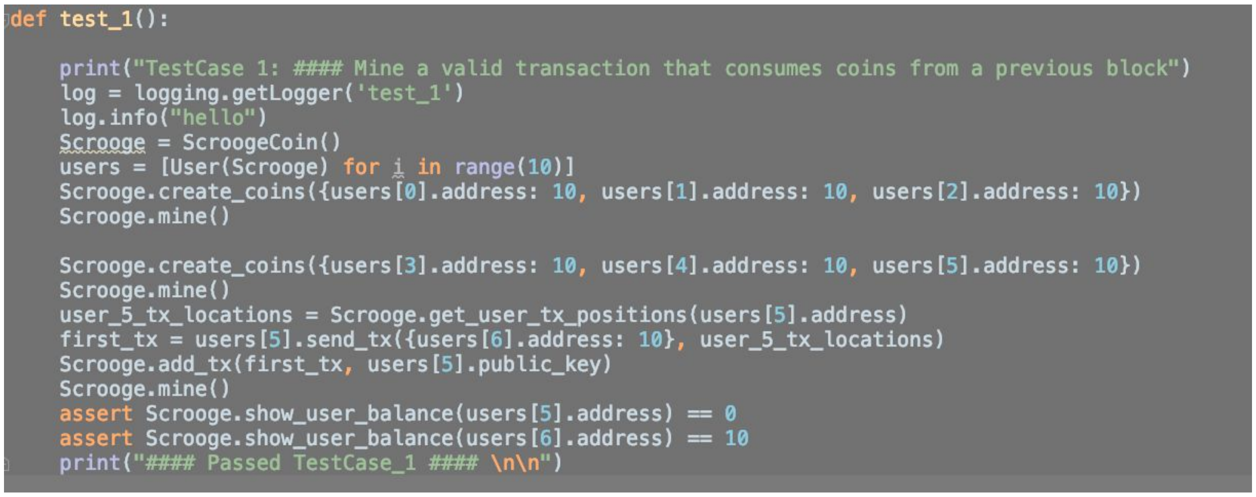
Scrooge Coin

Description - All functionalities are completed and working fine. Below are the screenshots. A user is able to successfully add transactions by sending it to Scrooge. The blockchain looks good with all the information.
I have used python assert library for asserting in the test cases.

#### Test Cases ####

All Test Cases Passed

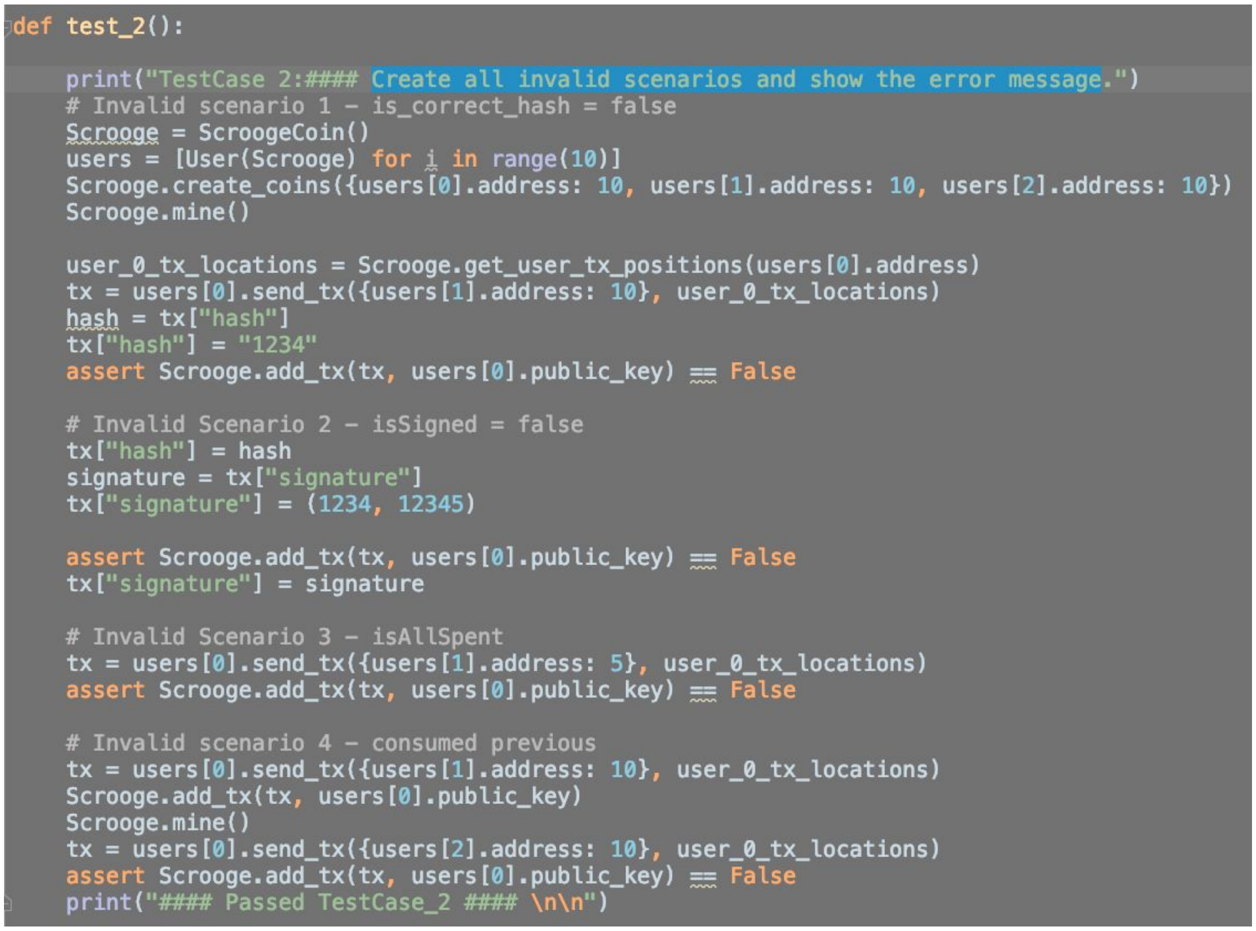
Test Case 1: Mine a valid transaction that consumes coins from a previous block



```python
def test_1():

    print("TestCase 1: #### Mine a valid transaction that consumes coins from a previous block")
    log = logging.getLogger('test_1')
    log.info("hello")
    Scrooge = ScroogeCoin()
    users = [User(Scrooge) for i in range(10)]
    Scrooge.create_coins({users[0].address: 10, users[1].address: 10, users[2].address: 10})
    Scrooge.mine()

    Scrooge.create_coins({users[3].address: 10, users[4].address: 10, users[5].address: 10})
    Scrooge.mine()
    user_5_tx_locations = Scrooge.get_user_tx_positions(users[5].address)
    first_tx = users[5].send_tx({users[6].address: 10}, user_5_tx_locations)
    Scrooge.add_tx(first_tx, users[5].public_key)
    Scrooge.mine()
    assert Scrooge.show_user_balance(users[5].address) == 0
    assert Scrooge.show_user_balance(users[6].address) == 10
    print("#### Passed TestCase_1 #### \n\n")
```

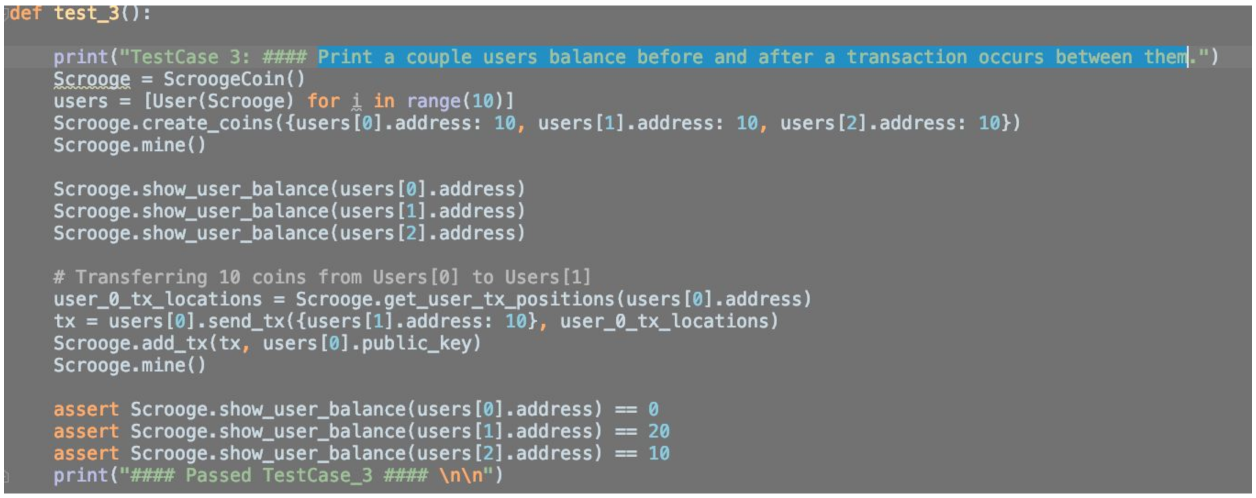
```
Logs: TestCase 1: #### Executing Test1: Mine a valid transaction that consumes coins from a previous block
0
10
#### Passed TestCase_1 ####
```

Test Case 2: Create all invalid scenarios and show the error message


```
def test_2():

    print("TestCase 2:#### Create all invalid scenarios and show the error message.")
    # Invalid scenario 1 - is_correct_hash = false
    Scrooge = ScroogeCoin()
    users = [User(Scrooge) for i in range(10)]
    Scrooge.create_coins({users[0].address: 10, users[1].address: 10, users[2].address: 10})
    Scrooge.mine()

    user_0_tx_locations = Scrooge.get_user_tx_positions(users[0].address)
    tx = users[0].send_tx({users[1].address: 10}, user_0_tx_locations)
    hash = tx["hash"]
    tx["hash"] = "1234"
    assert Scrooge.add_tx(tx, users[0].public_key) == False

    # Invalid Scenario 2 - isSigned = false
    tx["hash"] = hash
    signature = tx["signature"]
    tx["signature"] = (1234, 12345)

    assert Scrooge.add_tx(tx, users[0].public_key) == False
    tx["signature"] = signature

    # Invalid Scenario 3 - isAllSpent
    tx = users[0].send_tx({users[1].address: 5}, user_0_tx_locations)
    assert Scrooge.add_tx(tx, users[0].public_key) == False

    # Invalid scenario 4 - consumed previous
    tx = users[0].send_tx({users[1].address: 10}, user_0_tx_locations)
    Scrooge.add_tx(tx, users[0].public_key)
    Scrooge.mine()
    tx = users[0].send_tx({users[2].address: 10}, user_0_tx_locations)
    assert Scrooge.add_tx(tx, users[0].public_key) == False
    print("#### Passed TestCase_2 #### \n\n")
```

Logs: TestCase 2:#### Executing Test2: Create all invalid scenarios and show the error message.
Passed TestCase_2

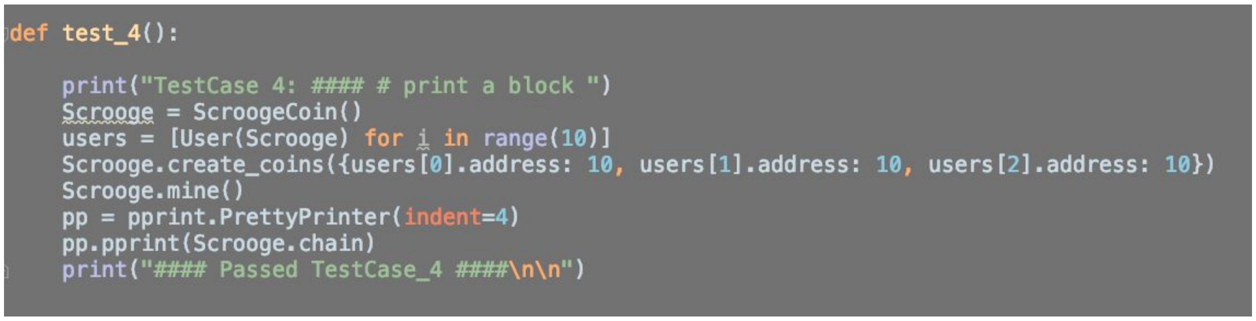
Test_Case 3: Print a couple users balance before and after a transaction occurs between them



```python
def test_3():

    print("TestCase 3: #### Print a couple users balance before and after a transaction occurs between them.")
    Scrooge = ScroogeCoin()
    users = [User(Scrooge) for i in range(10)]
    Scrooge.create_coins({users[0].address: 10, users[1].address: 10, users[2].address: 10})
    Scrooge.mine()

    Scrooge.show_user_balance(users[0].address)
    Scrooge.show_user_balance(users[1].address)
    Scrooge.show_user_balance(users[2].address)

    # Transferring 10 coins from Users[0] to Users[1]
    user_0_tx_locations = Scrooge.get_user_tx_positions(users[0].address)
    tx = users[0].send_tx({users[1].address: 10}, user_0_tx_locations)
    Scrooge.add_tx(tx, users[0].public_key)
    Scrooge.mine()

    assert Scrooge.show_user_balance(users[0].address) == 0
    assert Scrooge.show_user_balance(users[1].address) == 20
    assert Scrooge.show_user_balance(users[2].address) == 10
    print("#### Passed TestCase_3 #### \n\n")
```

```
TestCase 3: #### Print a couple users balance before and after a transaction occurs between them.
10
10
10
0
20
10
#### Passed TestCase_3 ####
```

Test_Case_4: print a block



```
def test_4():

    print("TestCase 4: #### # print a block ")
    Scrooge = ScroogeCoin()
    users = [User(Scrooge) for i in range(10)]
    Scrooge.create_coins({users[0].address: 10, users[1].address: 10, users[2].address: 10})
    Scrooge.mine()
    pp = pprint.PrettyPrinter(indent=4)
    pp.pprint(Scrooge.chain)
    print("#### Passed TestCase_4 ####\n\n")
```

TestCase 4: #### # print a block

```
[   {   'hash': 'ec326c7e77732497335efd95e70e7665dbc6a2b8261d7ffb0eecf8306eb30ef3',
        'index': 0,
        'previous_hash': None,
        'signature': (
23313559572102596979784229415530541844343313474521192267748523512131872017024,

96909862837340034458784933325944490377356218289979842742429676700530307066286),
        'transactions': [   {   'hash':
'8e25b1cef8e7f68badbc56f878add68be88237febcab7c7ec1ca06b12b3da827',
                                'location': {'block': -1, 'tx': -1},
                                'receivers': {
'8b79d33919a584075df4d8158f46090e8be3047767029d193bf746951a5ae920': 10,

'94a219495eb1475bbc0715655c3011cce07a5a7ee24a957777a21c664dfe4bf7': 10,

'd35331746727c4fba61c18a1af6f373537e35e8a4f03caa1fa06c0b701bd43e3': 10},
                                'sender':
'94a568b6094109bdfaa3bd1fc539a8679eacd81142217dc5860c2a00cecad230',
                                'signature': (
89808409879106700551932548500938539010181591820791239091984694933280271429026,

83421271347782155627820472797005043494773316339829938273207704117447653260441)}]}]
```

Passed TestCase_4