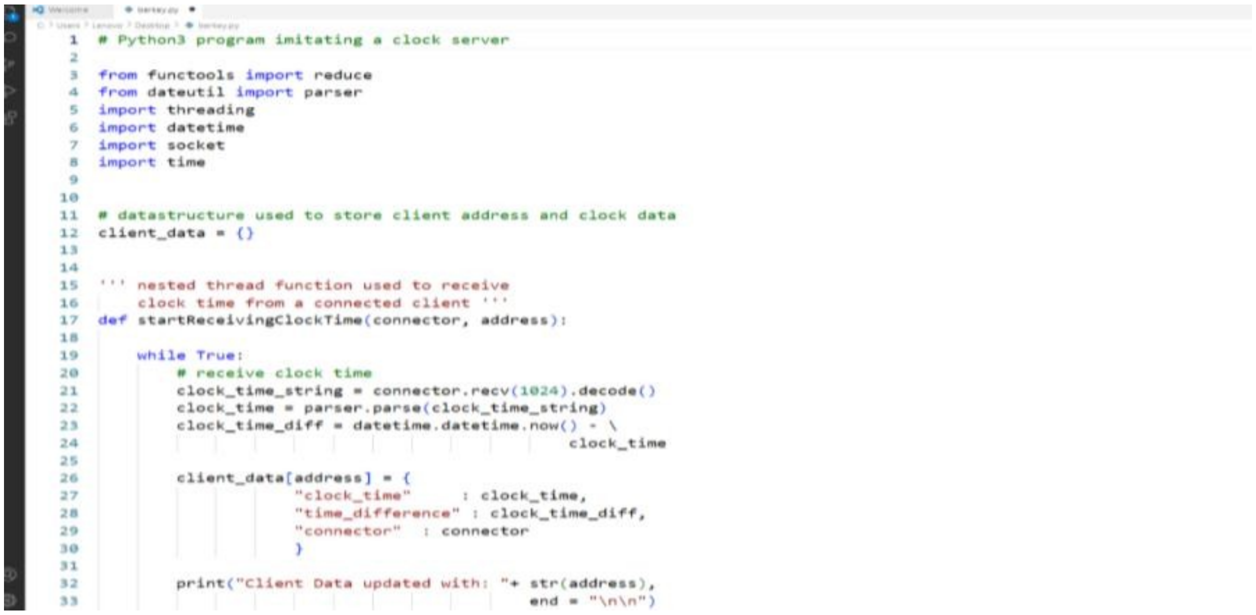
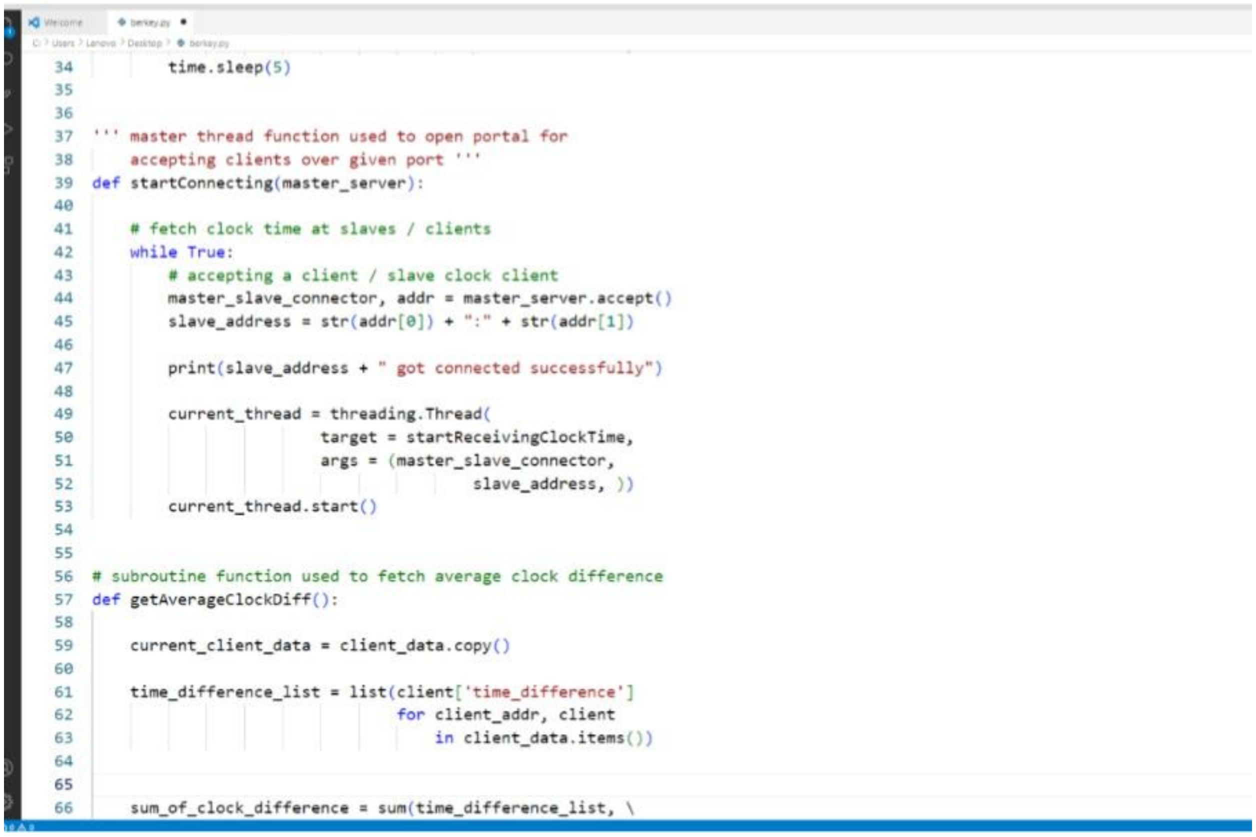
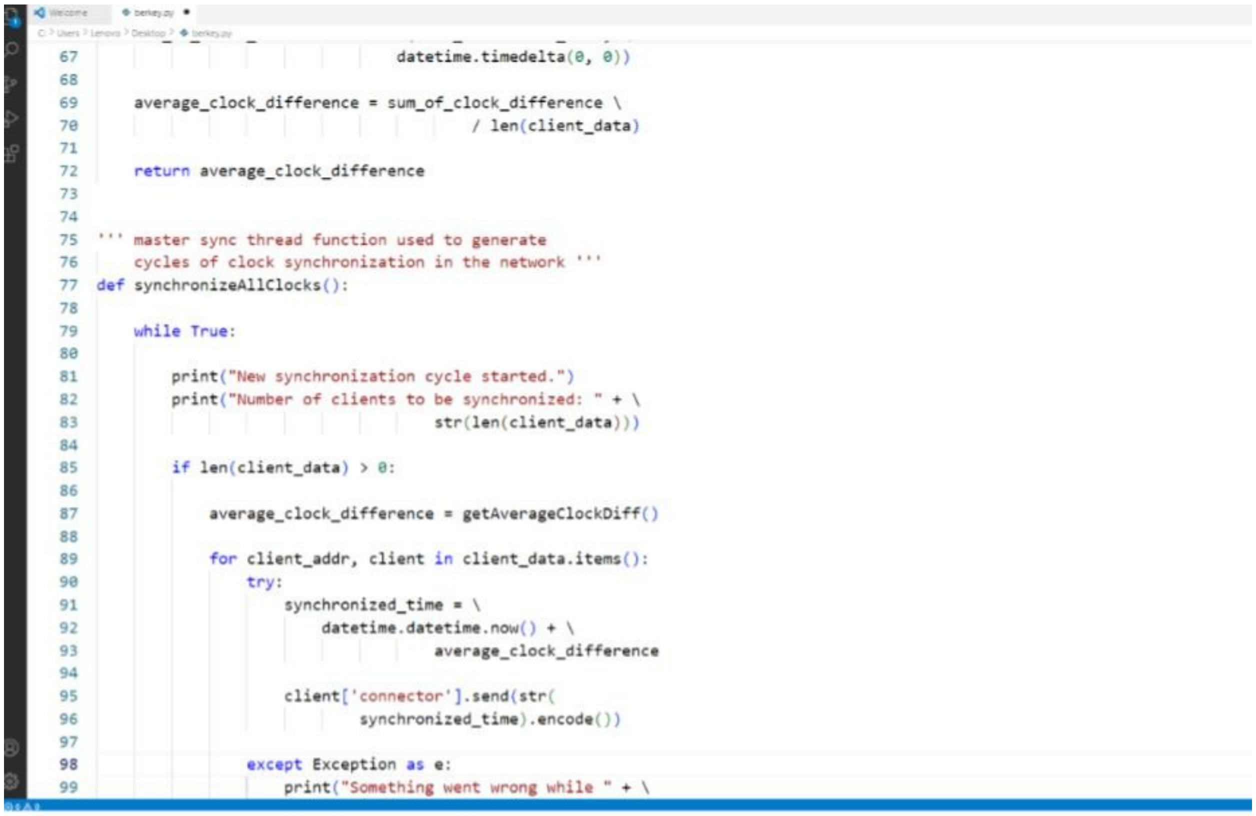


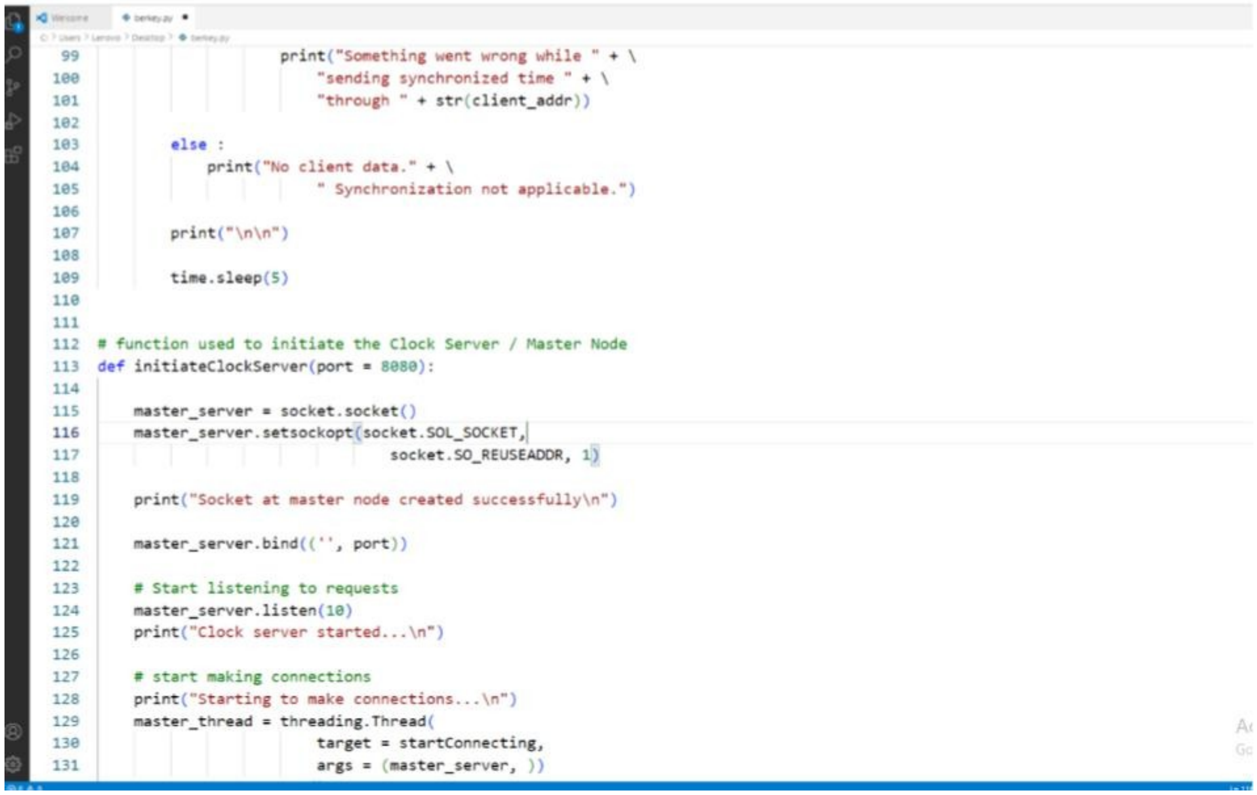
```python
# Python3 program imitating a clock server

from functools import reduce
from dateutil import parser
import threading
import datetime
import socket
import time


# datastructure used to store client address and clock data
client_data = {}


''' nested thread function used to receive
    clock time from a connected client '''
def startReceivingClockTime(connector, address):

    while True:
        # receive clock time
        clock_time_string = connector.recv(1024).decode()
        clock_time = parser.parse(clock_time_string)
        clock_time_diff = datetime.datetime.now() - \
                                                    clock_time

        client_data[address] = {
                    "clock_time"      : clock_time,
                    "time_difference" : clock_time_diff,
                    "connector"   : connector
                    }

        print("Client Data updated with: "+ str(address),
                                              end = "\n\n")
```



```python
        time.sleep(5)


''' master thread function used to open portal for
    accepting clients over given port '''
def startConnecting(master_server):

    # fetch clock time at slaves / clients
    while True:
        # accepting a client / slave clock client
        master_slave_connector, addr = master_server.accept()
        slave_address = str(addr[0]) + ":" + str(addr[1])

        print(slave_address + " got connected successfully")

        current_thread = threading.Thread(
                        target = startReceivingClockTime,
                        args = (master_slave_connector,
                                        slave_address, ))
        current_thread.start()


# subroutine function used to fetch average clock difference
def getAverageClockDiff():

    current_client_data = client_data.copy()

    time_difference_list = list(client['time_difference']
                            for client_addr, client
                                in client_data.items())


    sum_of_clock_difference = sum(time_difference_list, \
```

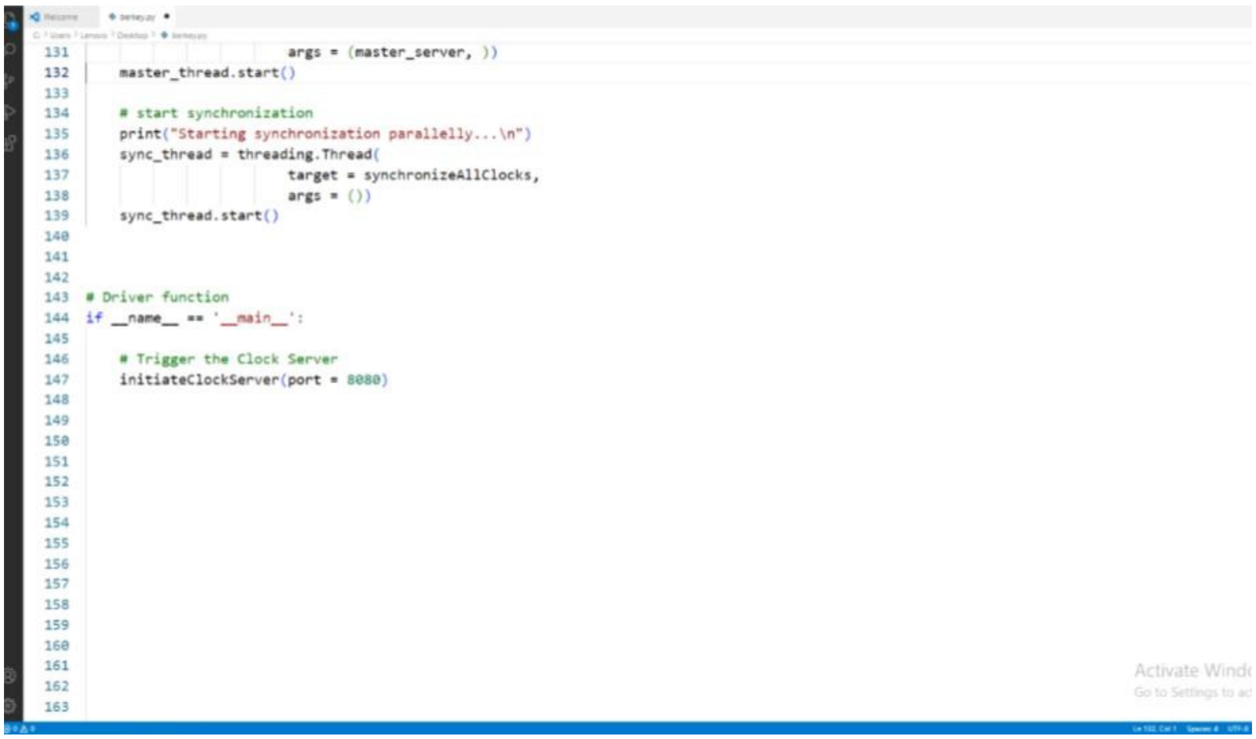


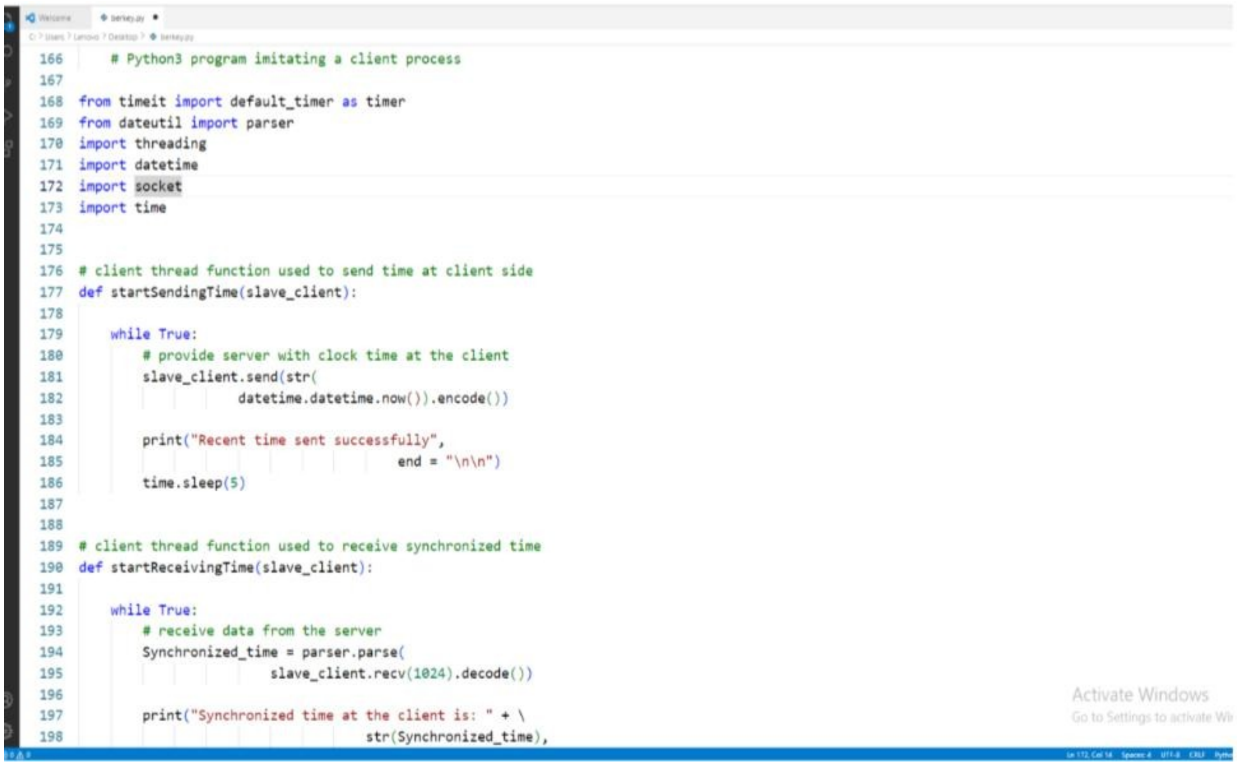
```
                                   datetime.timedelta(0, 0))

    average_clock_difference = sum_of_clock_difference \
                                         / len(client_data)

    return average_clock_difference


''' master sync thread function used to generate
    cycles of clock synchronization in the network '''
def synchronizeAllClocks():

    while True:

        print("New synchronization cycle started.")
        print("Number of clients to be synchronized: " + \
                                   str(len(client_data)))

        if len(client_data) > 0:

            average_clock_difference = getAverageClockDiff()

            for client_addr, client in client_data.items():
                try:
                    synchronized_time = \
                        datetime.datetime.now() + \
                                   average_clock_difference

                    client['connector'].send(str(
                            synchronized_time).encode())

                except Exception as e:
                    print("Something went wrong while " + \
```

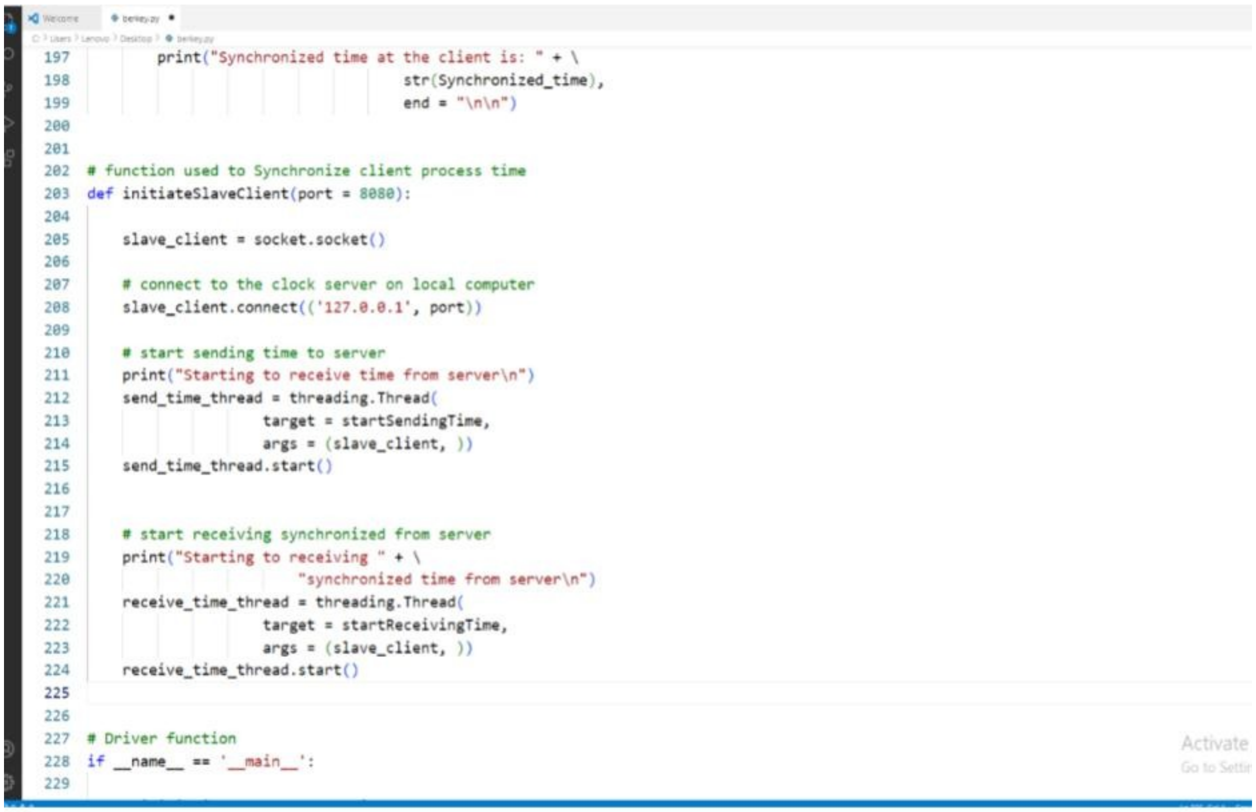


```
                    print("Something went wrong while " + \
```

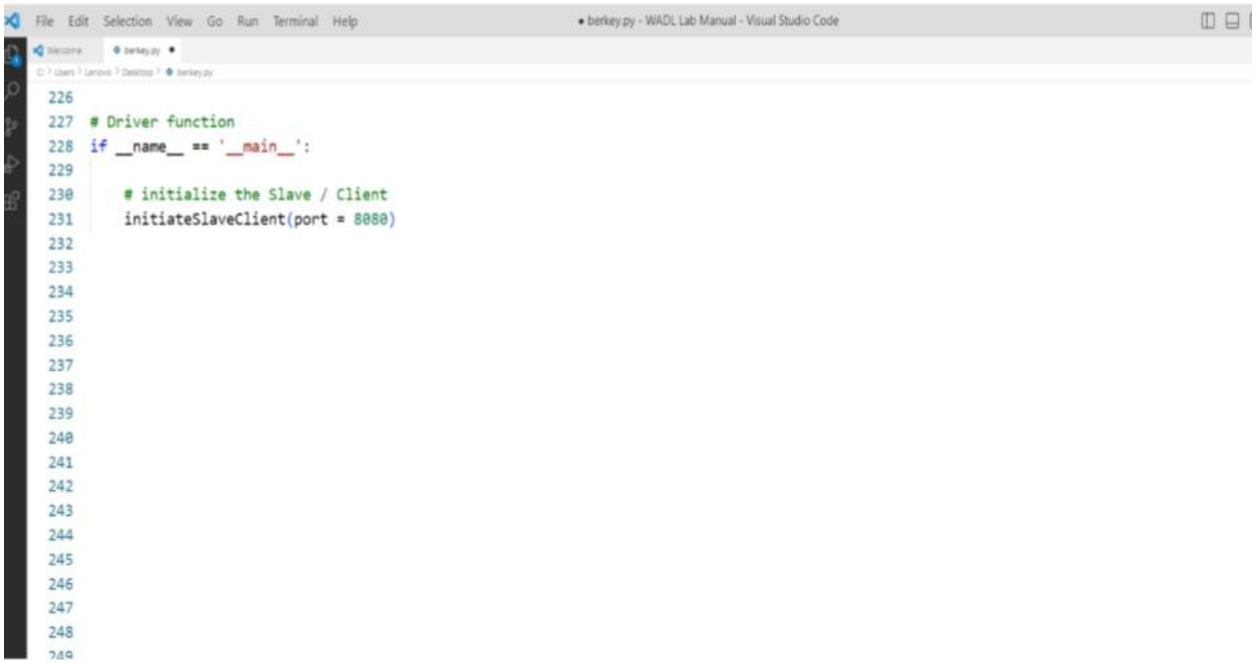
```
                        "sending synchronized time " + \
                        "through " + str(client_addr))

        else :
            print("No client data." + \
                        " Synchronization not applicable.")

        print("\n\n")

        time.sleep(5)


# function used to initiate the Clock Server / Master Node
def initiateClockServer(port = 8080):

    master_server = socket.socket()
    master_server.setsockopt(socket.SOL_SOCKET,
                                 socket.SO_REUSEADDR, 1)

    print("Socket at master node created successfully\n")

    master_server.bind(('', port))

    # Start listening to requests
    master_server.listen(10)
    print("Clock server started...\n")

    # start making connections
    print("Starting to make connections...\n")
    master_thread = threading.Thread(
                        target = startConnecting,
                        args = (master_server, ))
```


```python
                                args = (master_server, ))
    master_thread.start()

    # start synchronization
    print("Starting synchronization parallelly...\n")
    sync_thread = threading.Thread(
                                target = synchronizeAllClocks,
                                args = ())
    sync_thread.start()


# Driver function
if __name__ == '__main__':

    # Trigger the Clock Server
    initiateClockServer(port = 8080)
```


```python
    # Python3 program imitating a client process

from timeit import default_timer as timer
from dateutil import parser
import threading
import datetime
import socket
import time


# client thread function used to send time at client side
def startSendingTime(slave_client):

    while True:
        # provide server with clock time at the client
        slave_client.send(str(
                    datetime.datetime.now()).encode())

        print("Recent time sent successfully",
                                            end = "\n\n")
        time.sleep(5)


# client thread function used to receive synchronized time
def startReceivingTime(slave_client):

    while True:
        # receive data from the server
        Synchronized_time = parser.parse(
                        slave_client.recv(1024).decode())

        print("Synchronized time at the client is: " + \
                                        str(Synchronized_time),
```

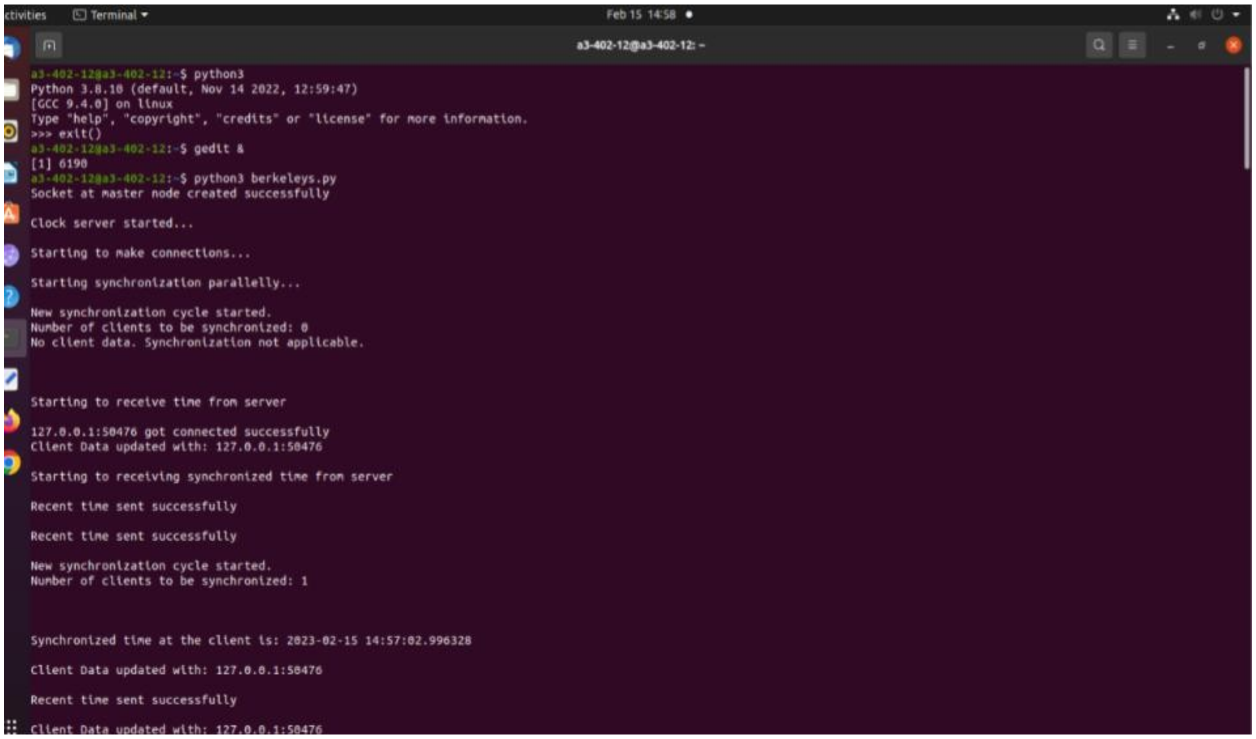


```
        print("Synchronized time at the client is: " + \
                                str(Synchronized_time),
                                end = "\n\n")


# function used to Synchronize client process time
def initiateSlaveClient(port = 8080):

    slave_client = socket.socket()

    # connect to the clock server on local computer
    slave_client.connect(('127.0.0.1', port))

    # start sending time to server
    print("Starting to receive time from server\n")
    send_time_thread = threading.Thread(
                    target = startSendingTime,
                    args = (slave_client, ))
    send_time_thread.start()


    # start receiving synchronized from server
    print("Starting to receiving " + \
                        "synchronized time from server\n")
    receive_time_thread = threading.Thread(
                    target = startReceivingTime,
                    args = (slave_client, ))
    receive_time_thread.start()


# Driver function
if __name__ == '__main__':
```



File Edit Selection View Go Run Terminal Help
• berkey.py - WADL Lab Manual - Visual Studio Code

```

# Driver function
if __name__ == '__main__':

    # initialize the Slave / Client
    initiateSlaveClient(port = 8080)
```



```
a3-402-12@a3-402-12:~$ python3
Python 3.8.10 (default, Nov 14 2022, 12:59:47)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()
a3-402-12@a3-402-12:~$ gedit &
[1] 6190
a3-402-12@a3-402-12:~$ python3 berkeleys.py
Socket at master node created successfully

Clock server started...

Starting to make connections...

Starting synchronization parallelly...

New synchronization cycle started.
Number of clients to be synchronized: 0
No client data. Synchronization not applicable.



Starting to receive time from server

127.0.0.1:50476 got connected successfully
Client Data updated with: 127.0.0.1:50476

Starting to receiving synchronized time from server

Recent time sent successfully

Recent time sent successfully

New synchronization cycle started.
Number of clients to be synchronized: 1



Synchronized time at the client is: 2023-02-15 14:57:02.996328

Client Data updated with: 127.0.0.1:50476

Recent time sent successfully

Client Data updated with: 127.0.0.1:50476
```