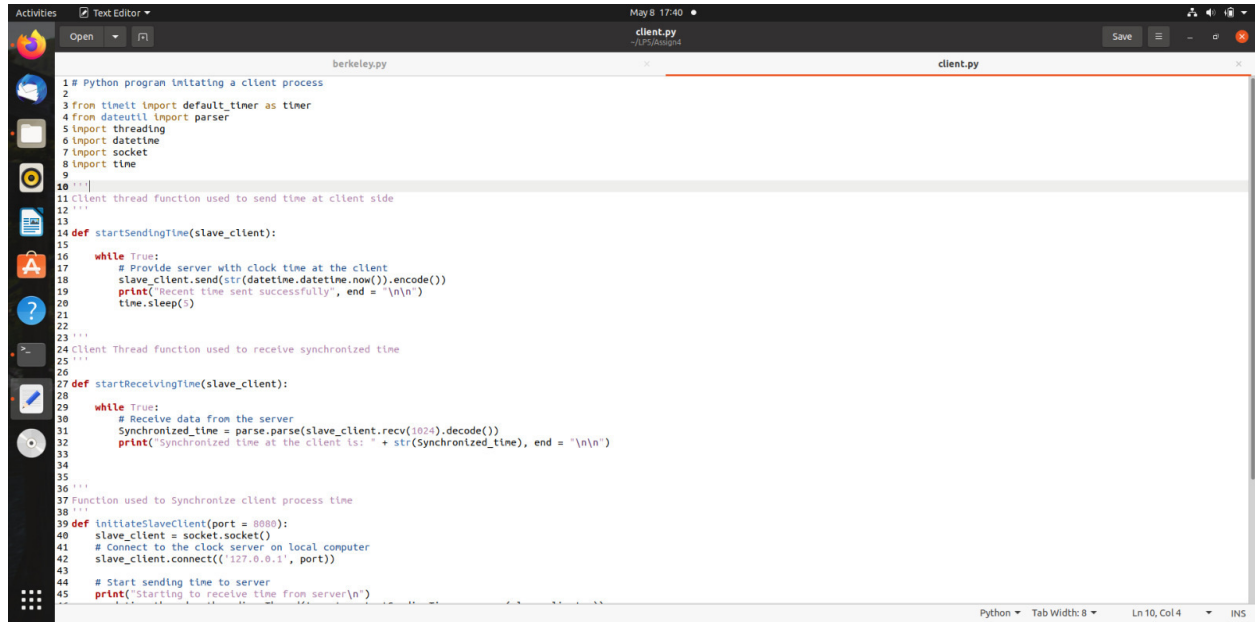


Name- Vaibhav Bichave

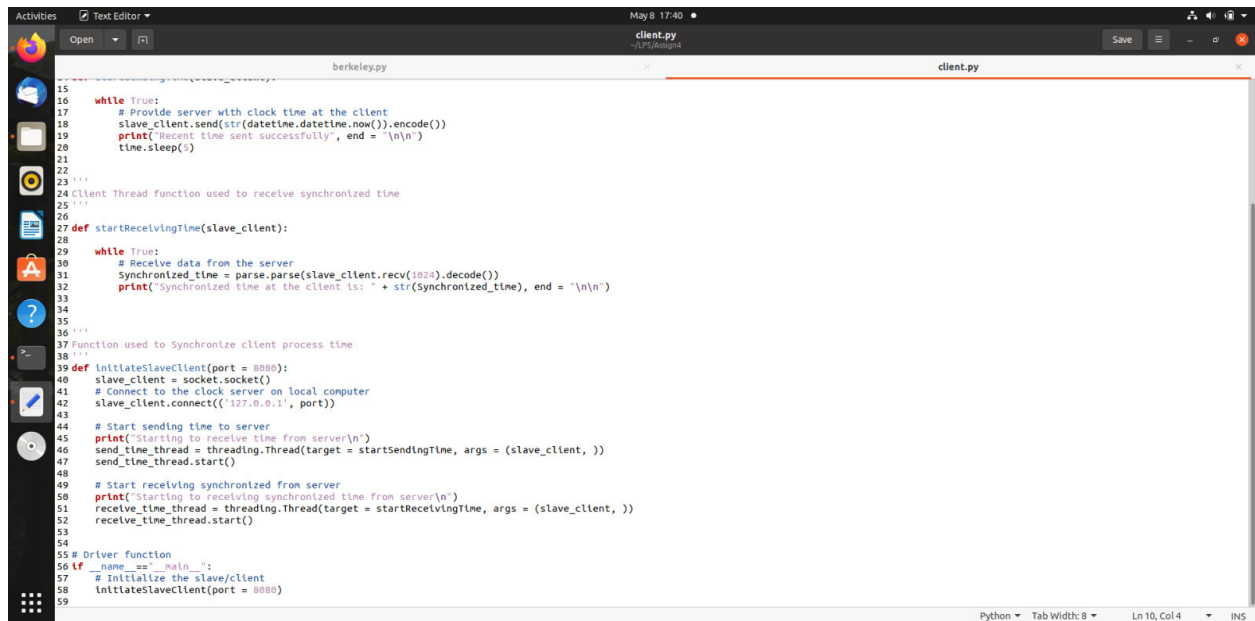
Roll no.-43209

CODE IMPLEMENTATION :



The screenshot shows a text editor window titled 'client.py' with a dark theme. The code is for a Python program that initiates a client process. It includes imports for time, threading, datetime, and socket. The code defines two functions: 'startSendingTime' and 'startReceivingTime', both using while loops to send and receive time data. The main function 'initiateSlaveClient' sets up a socket, connects to a server at 127.0.0.1 on port 8080, and starts the sending and receiving threads. The code is as follows:

```
1 # Python program initiating a client process
2
3 from timeit import default_timer as timer
4 from datetime import parser
5 import threading
6 import datetime
7 import socket
8 import time
9
10 ...
11 Client thread function used to send time at client side
12 ...
13
14 def startSendingTime(slave_client):
15
16     while True:
17         # Provide server with clock time at the client
18         slave_client.send(str(datetime.datetime.now()).encode())
19         print("Recent time sent successfully", end = "\n\n")
20         time.sleep(5)
21
22 ...
23
24 Client Thread function used to receive synchronized time
25 ...
26
27 def startReceivingTime(slave_client):
28
29     while True:
30         # Receive data from the server
31         Synchronized_time = parse.parse(slave_client.recv(1024).decode())
32         print("Synchronized time at the client is: " + str(Synchronized_time), end = "\n\n")
33
34 ...
35
36 ...
37 Function used to Synchronize client process time
38 ...
39
40 def initiateSlaveClient(port = 8080):
41     slave_client = socket.socket()
42     # Connect to the clock server on local computer
43     slave_client.connect(('127.0.0.1', port))
44
45     # Start sending time to server
46     print("Starting to receive time from server\n")
47     send_time_thread = threading.Thread(target = startSendingTime, args = (slave_client, ))
48     send_time_thread.start()
49
50     # Start receiving synchronized from server
51     print("Starting to receiving synchronized time from server\n")
52     receive_time_thread = threading.Thread(target = startReceivingTime, args = (slave_client, ))
53     receive_time_thread.start()
54
55 # Driver function
56 if __name__ == "__main__":
57     # Initialize the slave/client
58     initiateSlaveClient(port = 8080)
59
```



The screenshot shows a text editor window titled 'client.py' with a dark theme. The code is for a Python program that initiates a client process. It includes imports for time, threading, datetime, and socket. The code defines two functions: 'startSendingTime' and 'startReceivingTime', both using while loops to send and receive time data. The main function 'initiateSlaveClient' sets up a socket, connects to a server at 127.0.0.1 on port 8080, and starts the sending and receiving threads. The code is as follows:

```
15
16 while True:
17     # Provide server with clock time at the client
18     slave_client.send(str(datetime.datetime.now()).encode())
19     print("Recent time sent successfully", end = "\n\n")
20     time.sleep(5)
21
22 ...
23
24 Client Thread function used to receive synchronized time
25 ...
26
27 def startReceivingTime(slave_client):
28
29     while True:
30         # Receive data from the server
31         Synchronized_time = parse.parse(slave_client.recv(1024).decode())
32         print("Synchronized time at the client is: " + str(Synchronized_time), end = "\n\n")
33
34 ...
35
36 ...
37 Function used to Synchronize client process time
38 ...
39
40 def initiateSlaveClient(port = 8080):
41     slave_client = socket.socket()
42     # Connect to the clock server on local computer
43     slave_client.connect(('127.0.0.1', port))
44
45     # Start sending time to server
46     print("Starting to receive time from server\n")
47     send_time_thread = threading.Thread(target = startSendingTime, args = (slave_client, ))
48     send_time_thread.start()
49
50     # Start receiving synchronized from server
51     print("Starting to receiving synchronized time from server\n")
52     receive_time_thread = threading.Thread(target = startReceivingTime, args = (slave_client, ))
53     receive_time_thread.start()
54
55 # Driver function
56 if __name__ == "__main__":
57     # Initialize the slave/client
58     initiateSlaveClient(port = 8080)
59
```

```
Activities Text Editor May 8 17:41
Open Berkeley.py - /LP5/Assignm Save
berkeley.py client.py
1 # Importing necessary libraries
2
3 from functools import reduce
4 from datetime import parser
5 import threading
6 import datetime
7 import socket
8 import time
9
10 # datastructure used to store client address and clock data
11 client_data = {}
12
13 '''
14 Nested thread function used to receive clock time
15 from a connected client
16 '''
17
18 def startReceivingClockTime(connector, address):
19
20     while True:
21         # Receive clock time
22         clock_time_string = connector.recv(1024).decode()
23         clock_time = parser.parse(clock_time_string)
24         clock_time_diff = datetime.datetime.now() - clock_time
25
26         client_data[address] = {
27             "clock_time": clock_time,
28             "time_difference": clock_time_diff,
29             "connector": connector
30         }
31
32         print("client data updated with : " + str(address), end = "\n\n")
33         time.sleep(5)
34
35 '''
36 Master thread function used to open portal for accepting clients over given port
37 '''
38
39 def startConnecting(master_server):
40
41     # Fetch clock time at slaves/clients
42     while True:
43         # Accepting a client/slave clock client
44         master_slave_connector, addr = master_server.accept()
45         slave_address = str(addr[0]) + ":" + str(addr[1])
```

```
Activities Text Editor May 8 17:41
Open Berkeley.py - /LP5/Assignm Save
berkeley.py client.py
45 slave_address = str(addr[0]) + ":" + str(addr[1])
46
47 print(slave_address + " got connected successfully")
48
49 current_thread = threading.Thread(target = startReceivingClockTime, args = (master_slave_connector, slave_address))
50
51 current_thread.start()
52
53 '''
54 Subroutine function used to fetch average clock difference
55 '''
56
57 def getAverageClockDiff():
58
59     current_client_data = client_data.copy()
60
61     time_difference_list = list(client['time_difference'] for client_addr, client in client_data.items())
62
63     sum_of_clock_difference = sum(time_difference_list, datetime.timedelta(0, 0))
64
65     average_clock_difference = sum_of_clock_difference / len(client_data)
66
67     return average_clock_difference
68
69
70
71 '''
72 Master sync thread function used to generate cycles of clock synchronization in the network
73 '''
74 def synchronizeAllClocks():
75
76     while True:
77
78         print("New synchronization cycle started")
79         print("Number of clients to be synchronized: " + str(len(client_data)))
80
81         if len(client_data) > 0:
82             average_clock_difference = getAverageClockDiff()
83             for client_addr, client in client_data.items():
84                 try:
85                     synchronized_time = datetime.datetime.now() + average_clock_difference
86
87                     client['connector'].send(str(synchronized_time).encode())
88
89             except Exception as e:
```

