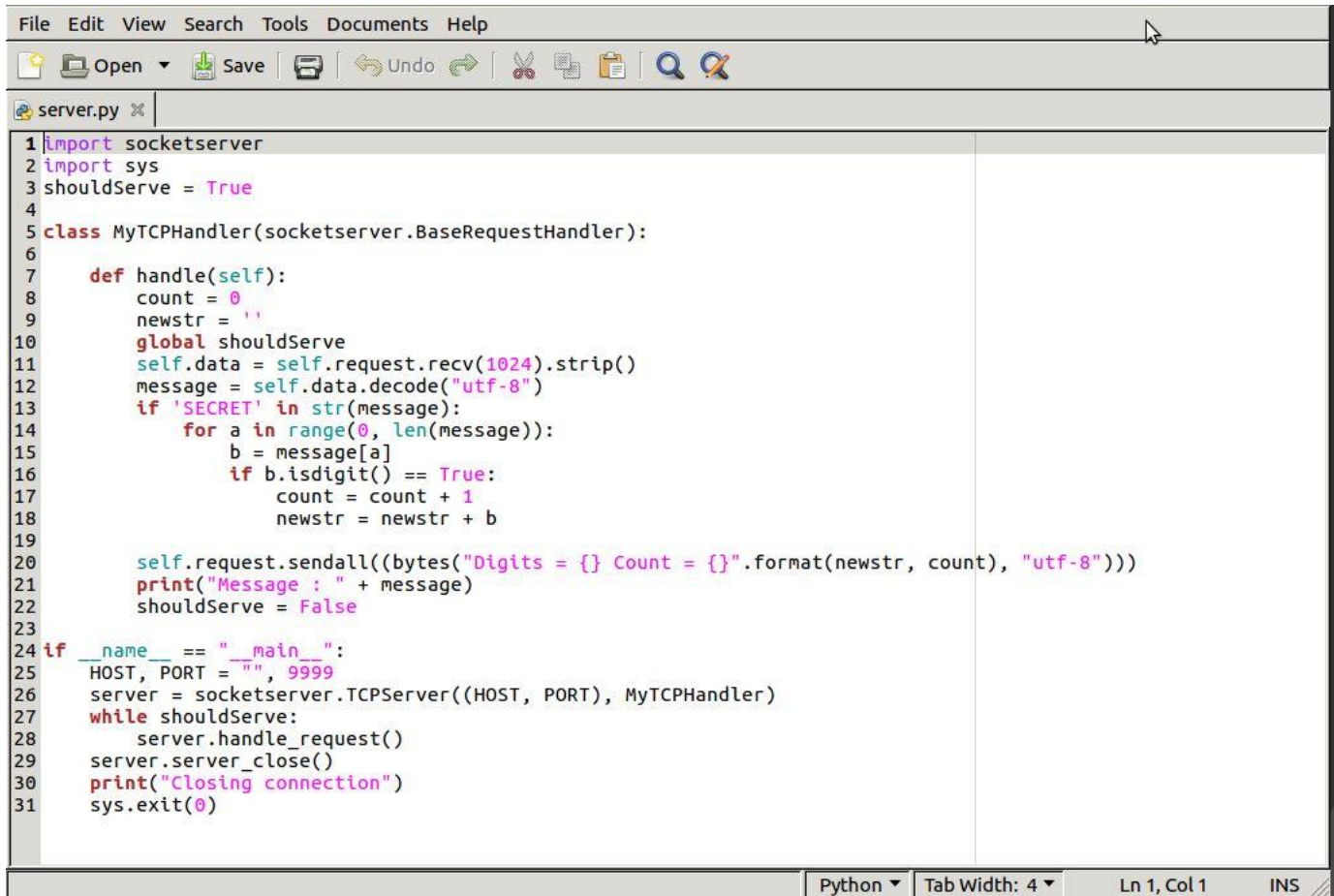


Assignment : Socket Programming
Computer Networking
Tanmay Dureja (td1391)

- Write an Echo Server/Client code using **socketserver** framework

Echo Server:

- a) should receive a string from Client
- b) If the string contains secret code "SECRET", then server should return all the digits in the string and count of digits
- c) close the connection with client



```

1 import socketserver
2 import sys
3 shouldServe = True
4
5 class MyTCPHandler(socketserver.BaseRequestHandler):
6
7     def handle(self):
8         count = 0
9         newstr = ''
10        global shouldServe
11        self.data = self.request.recv(1024).strip()
12        message = self.data.decode("utf-8")
13        if 'SECRET' in str(message):
14            for a in range(0, len(message)):
15                b = message[a]
16                if b.isdigit() == True:
17                    count = count + 1
18                newstr = newstr + b
19
20        self.request.sendall((bytes("Digits = {} Count = {}".format(newstr, count), "utf-8")))
21        print("Message : " + message)
22        shouldServe = False
23
24 if __name__ == "__main__":
25     HOST, PORT = "", 9999
26     server = socketserver.TCPServer((HOST, PORT), MyTCPHandler)
27     while shouldServe:
28         server.handle_request()
29     server.server_close()
30     print("Closing connection")
31     sys.exit(0)

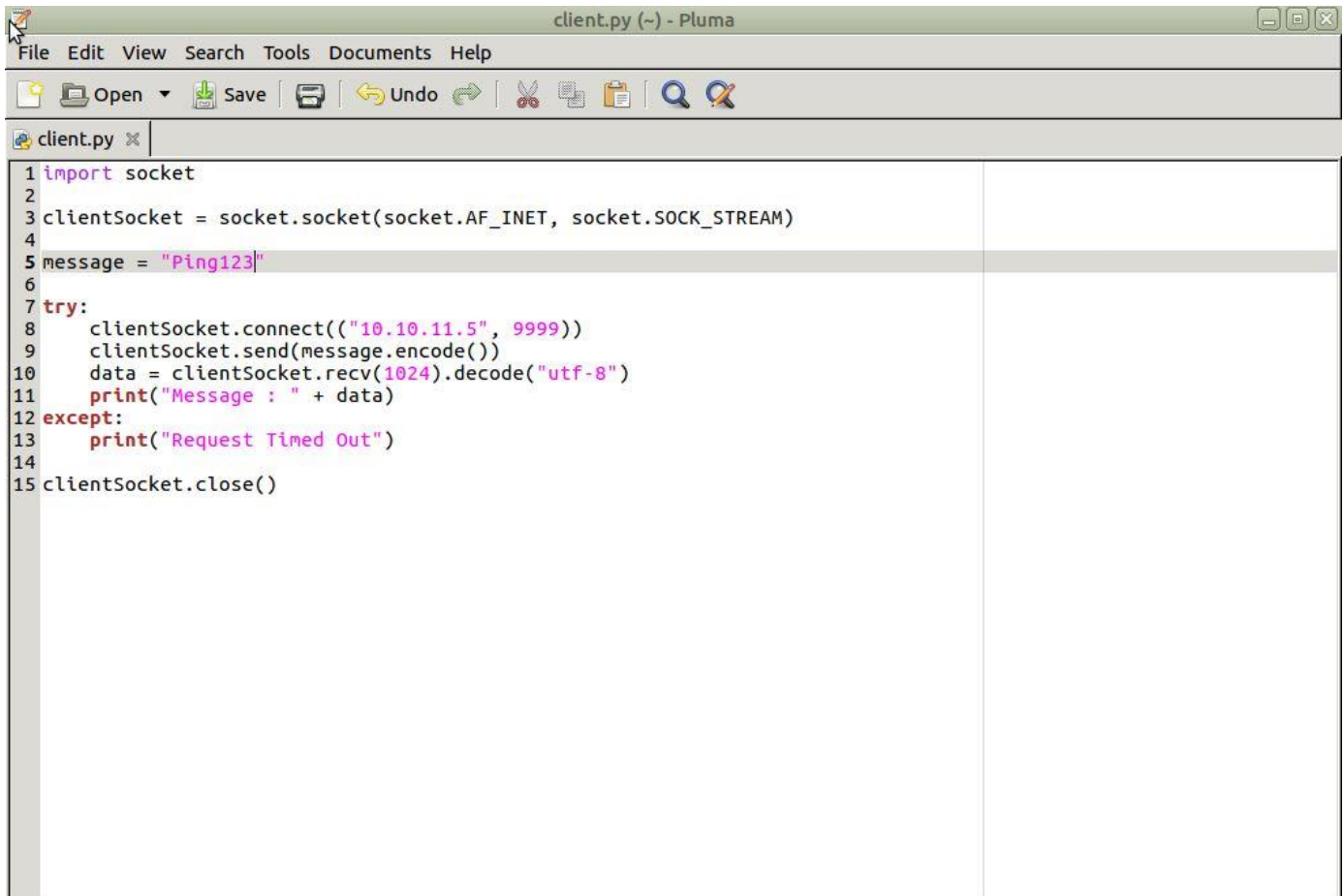
```

Capture 1 Server Program to count the number of digits if message contains the keyword 'SECRET'

1. Open SocketServer, check for shouldServe flag to receive data and parse.
2. Line 8 and 9 in the program are used to initialize the values of variables 'count' and 'newstr' representing the counter and the newstring generated for the numbers respectively.
3. In Line 12 the message from the client is received by the server and decoded as string.
4. In Line 13 the keyword 'SECRET' is checked if in string received, if true the string is parsed character by character and compared if is a digit.
If the character is a digit, the counter is incremented, and the digit appended to newstring.
If the keyword 'SECRET' is not in the string received, an empty message with the default values for 'count' and 'newstr' are sent as the string is not parsed.
5. The newstring and counter data are sent over to the client.
6. Connection is closed.

Echo Client:

- a) Should send a string to the server
- b) Should receive the output

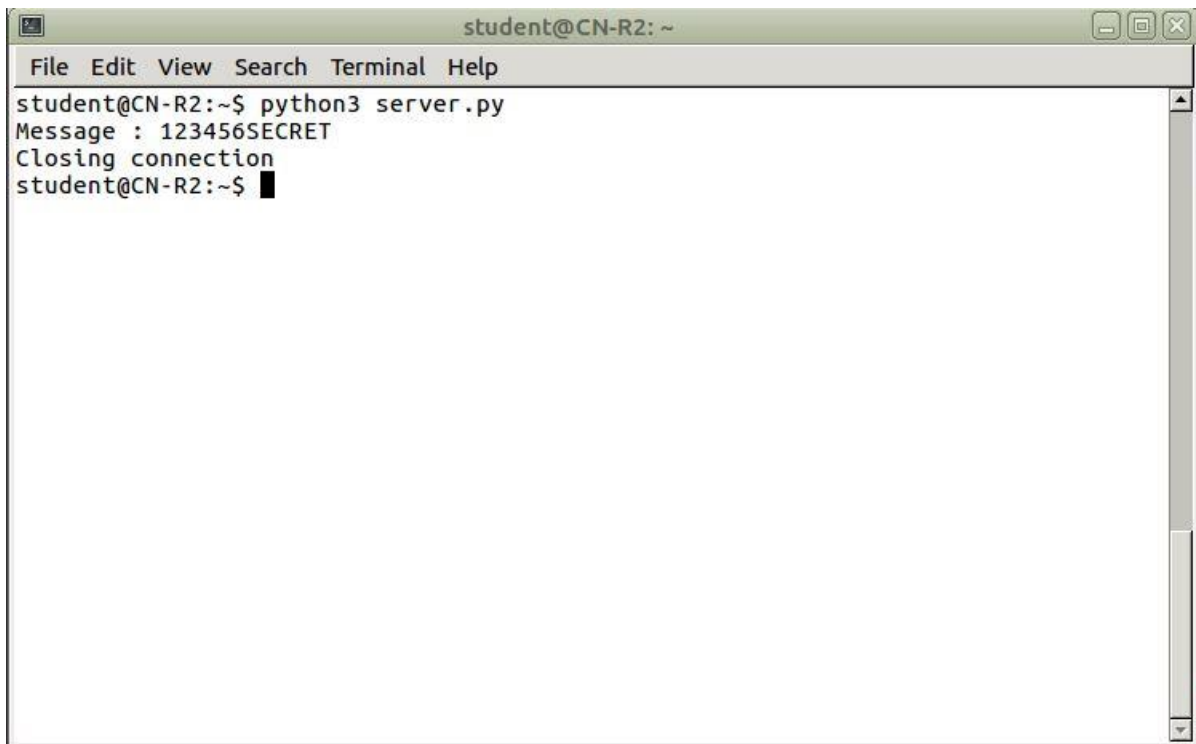


The screenshot shows a text editor window titled "client.py (~) - Pluma". The window has a menu bar with "File", "Edit", "View", "Search", "Tools", "Documents", and "Help". Below the menu bar is a toolbar with icons for "Open", "Save", "Undo", "Redo", "Cut", "Copy", "Paste", "Find", and "Replace". The editor area contains the following Python code:

```
1 import socket
2
3 clientSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
4
5 message = "Ping123"
6
7 try:
8     clientSocket.connect(("10.10.11.5", 9999))
9     clientSocket.send(message.encode())
10    data = clientSocket.recv(1024).decode("utf-8")
11    print("Message : " + data)
12 except:
13     print("Request Timed Out")
14
15 clientSocket.close()
```

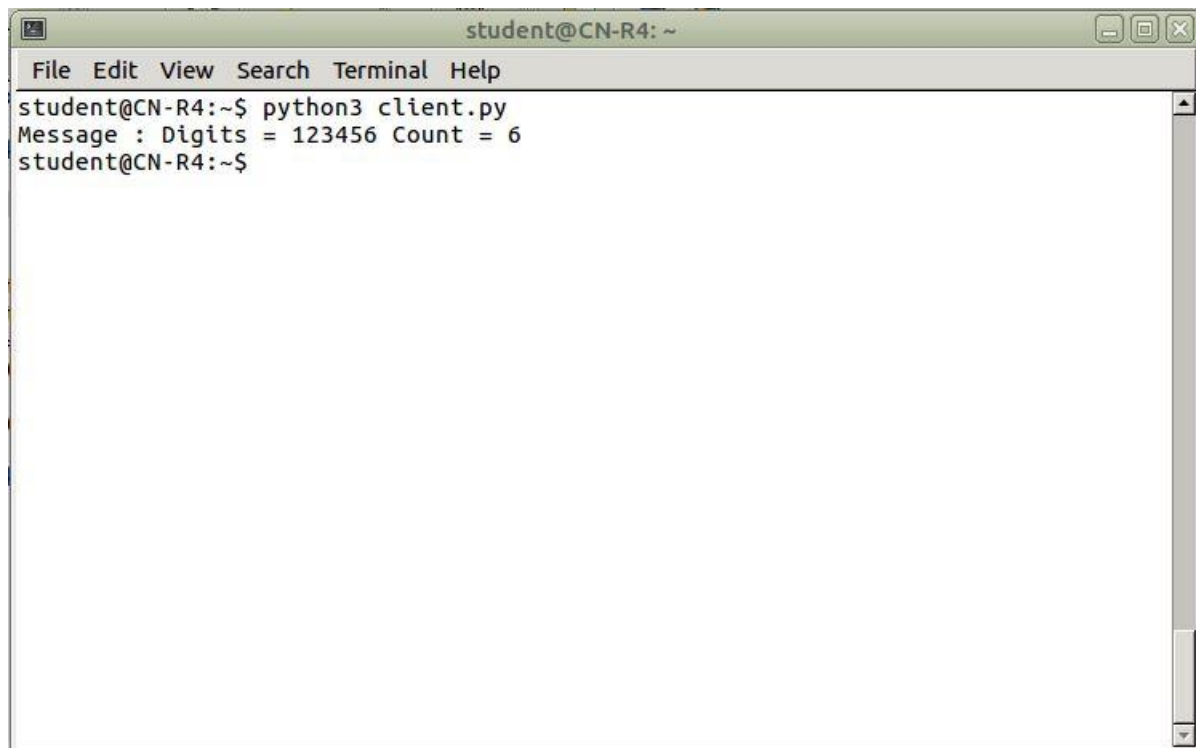
Capture 2 Client Program to send and receive data

1. Socket is created.
2. The message to be sent over to the server is defined in Line 5.
3. Now in Line 8-11 the client sends the message to the server and receives the parsed message and prints the data.
4. Socket is closed.

A terminal window titled 'student@CN-R2: ~' with a menu bar (File, Edit, View, Search, Terminal, Help). The output shows the execution of 'python3 server.py', which prints 'Message : 123456SECRET' and 'Closing connection' before returning to the prompt.

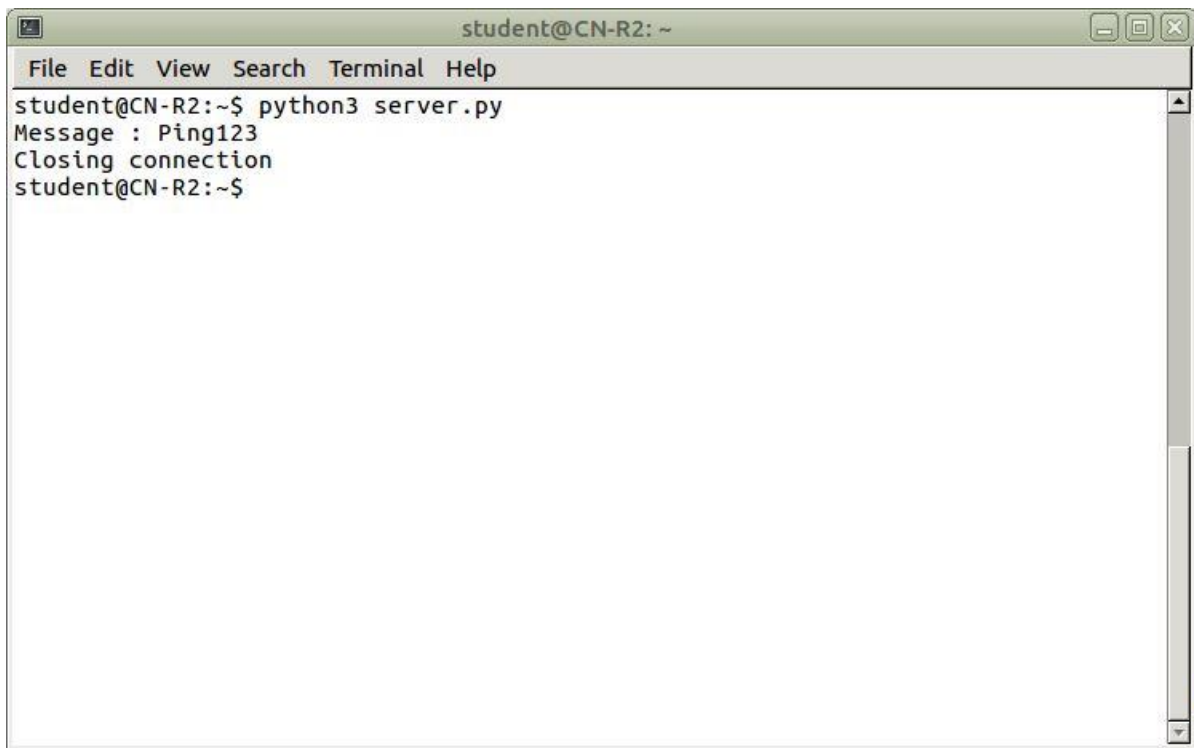
```
student@CN-R2: ~  
File Edit View Search Terminal Help  
student@CN-R2:~$ python3 server.py  
Message : 123456SECRET  
Closing connection  
student@CN-R2:~$
```

Capture 3 Server Program Example 1

A terminal window titled 'student@CN-R4: ~' with a menu bar (File, Edit, View, Search, Terminal, Help). The output shows the execution of 'python3 client.py', which prints 'Message : Digits = 123456 Count = 6' before returning to the prompt.

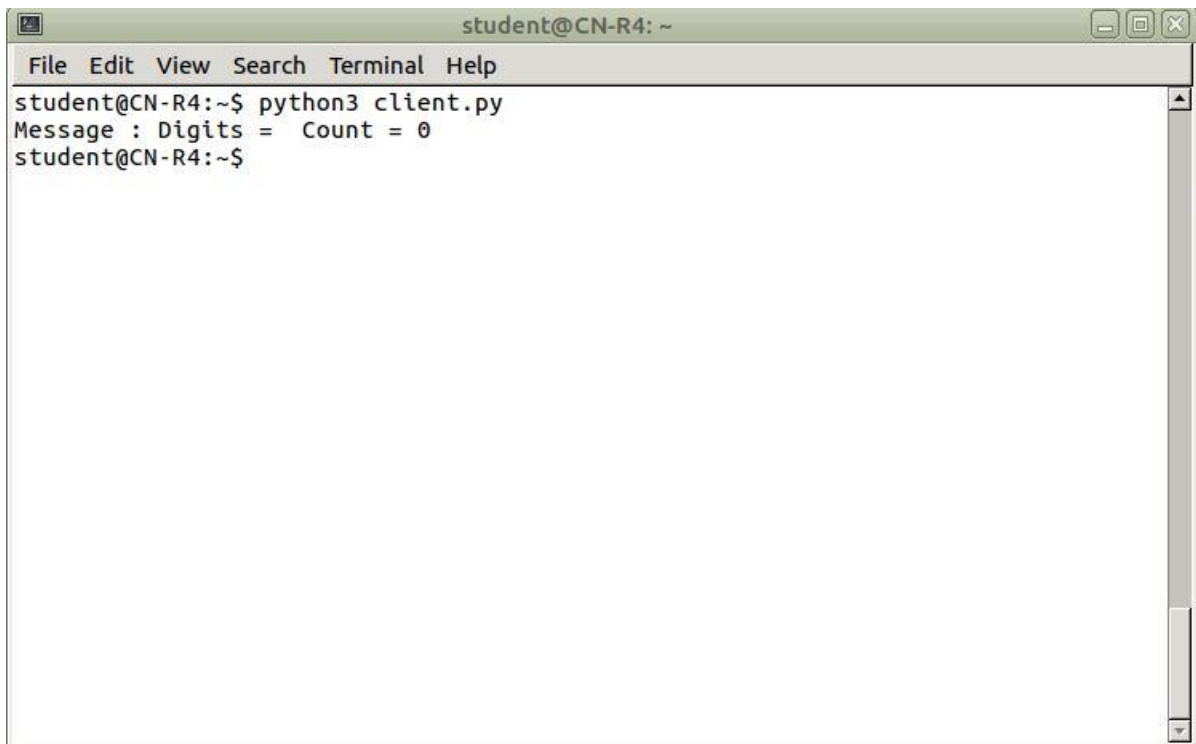
```
student@CN-R4: ~  
File Edit View Search Terminal Help  
student@CN-R4:~$ python3 client.py  
Message : Digits = 123456 Count = 6  
student@CN-R4:~$
```

Capture 4 Client Program Example 1

A terminal window titled 'student@CN-R2: ~' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the execution of 'python3 server.py', which outputs 'Message : Ping123' and 'Closing connection' before returning to the prompt.

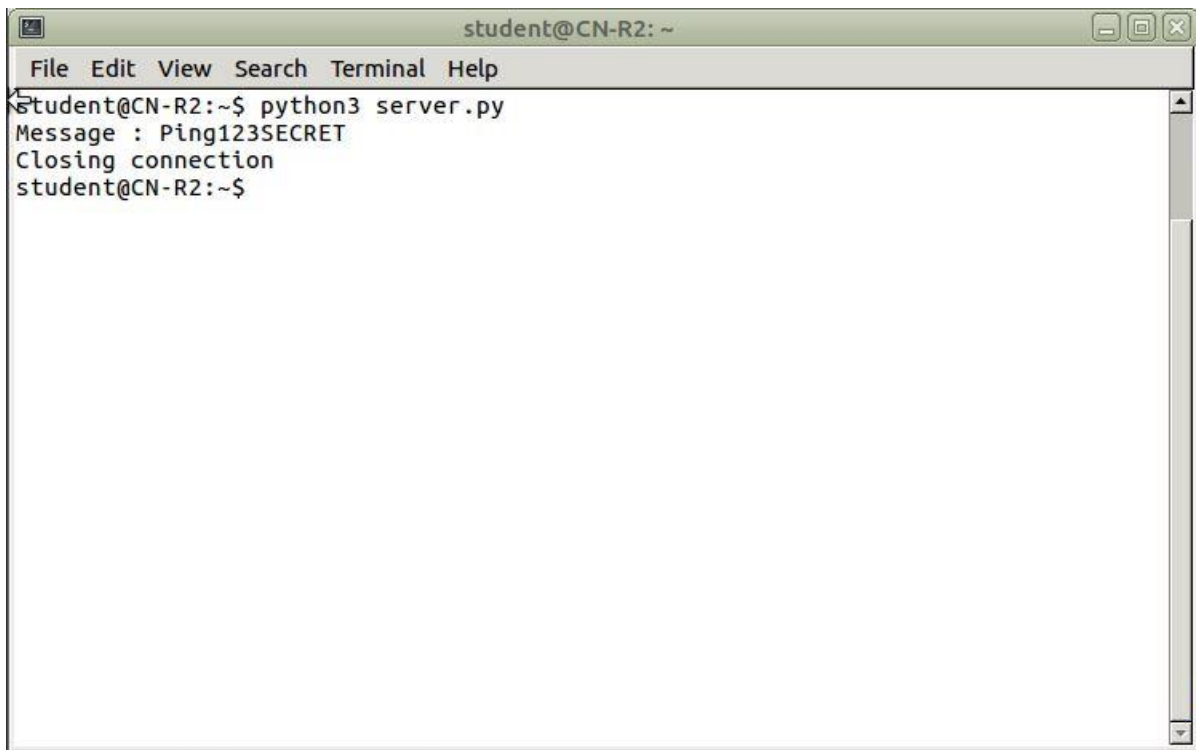
```
student@CN-R2: ~  
File Edit View Search Terminal Help  
student@CN-R2:~$ python3 server.py  
Message : Ping123  
Closing connection  
student@CN-R2:~$
```

Capture 5 Server Program Example 2

A terminal window titled 'student@CN-R4: ~' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the execution of 'python3 client.py', which outputs 'Message : Digits = Count = 0' before returning to the prompt.

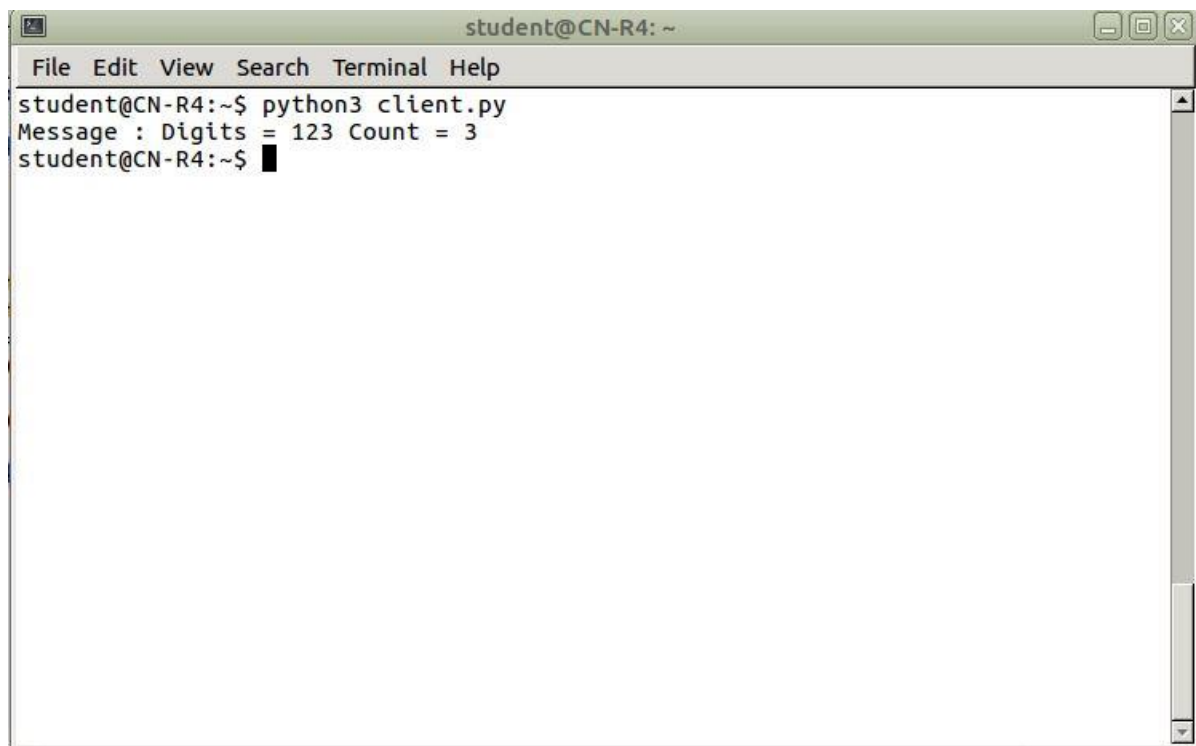
```
student@CN-R4: ~  
File Edit View Search Terminal Help  
student@CN-R4:~$ python3 client.py  
Message : Digits = Count = 0  
student@CN-R4:~$
```

Capture 6 Client Program Example 2

A terminal window titled 'student@CN-R2: ~' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the execution of 'python3 server.py', which outputs 'Message : Ping123SECRET' and 'Closing connection' before returning to the prompt.

```
student@CN-R2: ~  
File Edit View Search Terminal Help  
student@CN-R2:~$ python3 server.py  
Message : Ping123SECRET  
Closing connection  
student@CN-R2:~$
```

Capture 7 Server Program Example 3

A terminal window titled 'student@CN-R4: ~' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the execution of 'python3 client.py', which outputs 'Message : Digits = 123 Count = 3' before returning to the prompt.

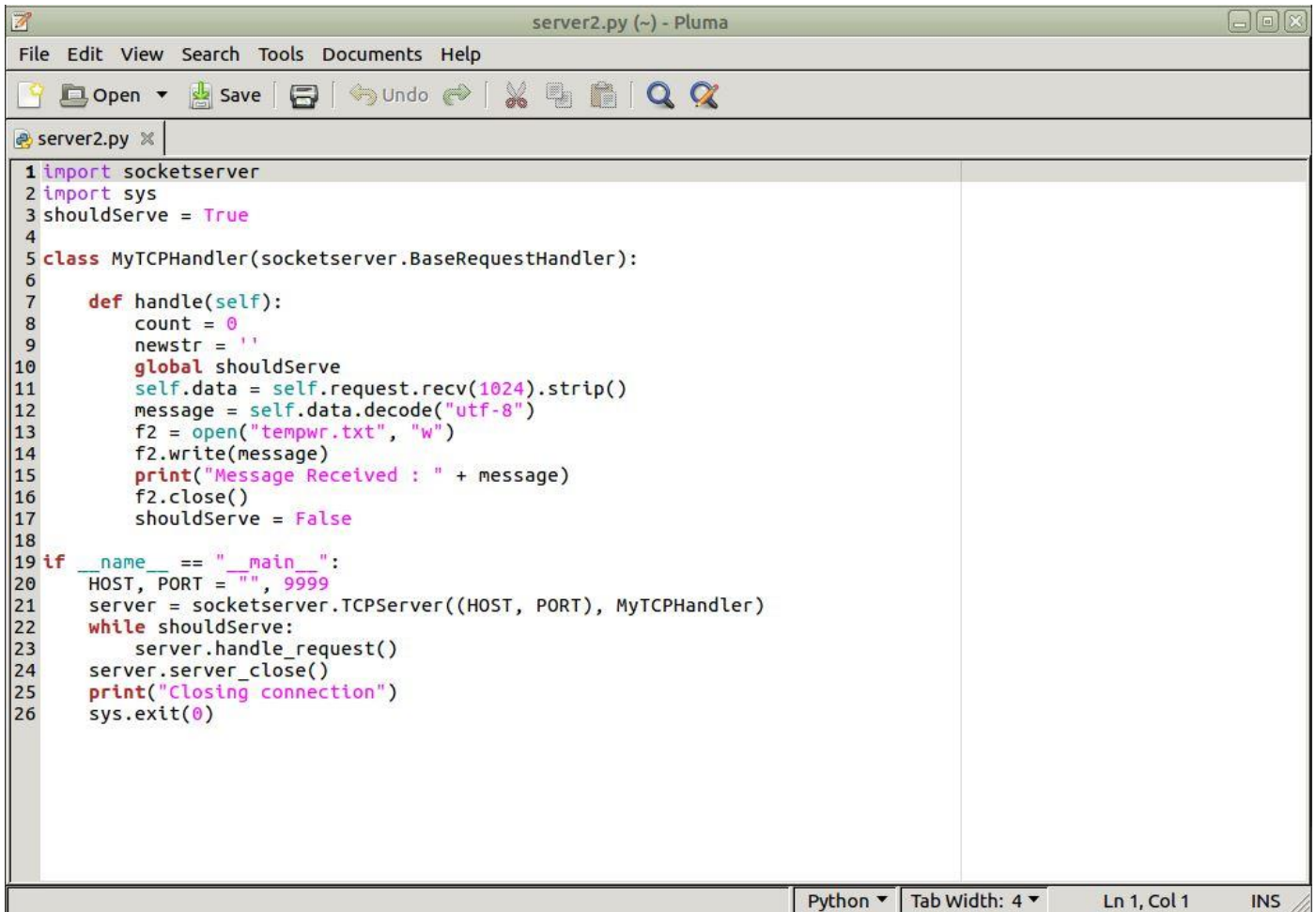
```
student@CN-R4: ~  
File Edit View Search Terminal Help  
student@CN-R4:~$ python3 client.py  
Message : Digits = 123 Count = 3  
student@CN-R4:~$
```

Capture 8 Client Program Example 3

- File Transfer over network using sockets.

Server:

- a) should receive data from Client (create a text file)
- b) write the received data in a file
- c) close the connection

A screenshot of a code editor window titled 'server2.py (~) - Pluma'. The editor has a menu bar (File, Edit, View, Search, Tools, Documents, Help) and a toolbar with icons for Open, Save, Undo, Redo, Cut, Copy, Paste, Find, and Replace. The code is as follows:

```
1 import socketserver
2 import sys
3 shouldServe = True
4
5 class MyTCPHandler(socketserver.BaseRequestHandler):
6
7     def handle(self):
8         count = 0
9         newstr = ''
10        global shouldServe
11        self.data = self.request.recv(1024).strip()
12        message = self.data.decode("utf-8")
13        f2 = open("tempwr.txt", "w")
14        f2.write(message)
15        print("Message Received : " + message)
16        f2.close()
17        shouldServe = False
18
19 if __name__ == "__main__":
20     HOST, PORT = "", 9999
21     server = socketserver.TCPServer((HOST, PORT), MyTCPHandler)
22     while shouldServe:
23         server.handle_request()
24     server.server_close()
25     print("Closing connection")
26     sys.exit(0)
```

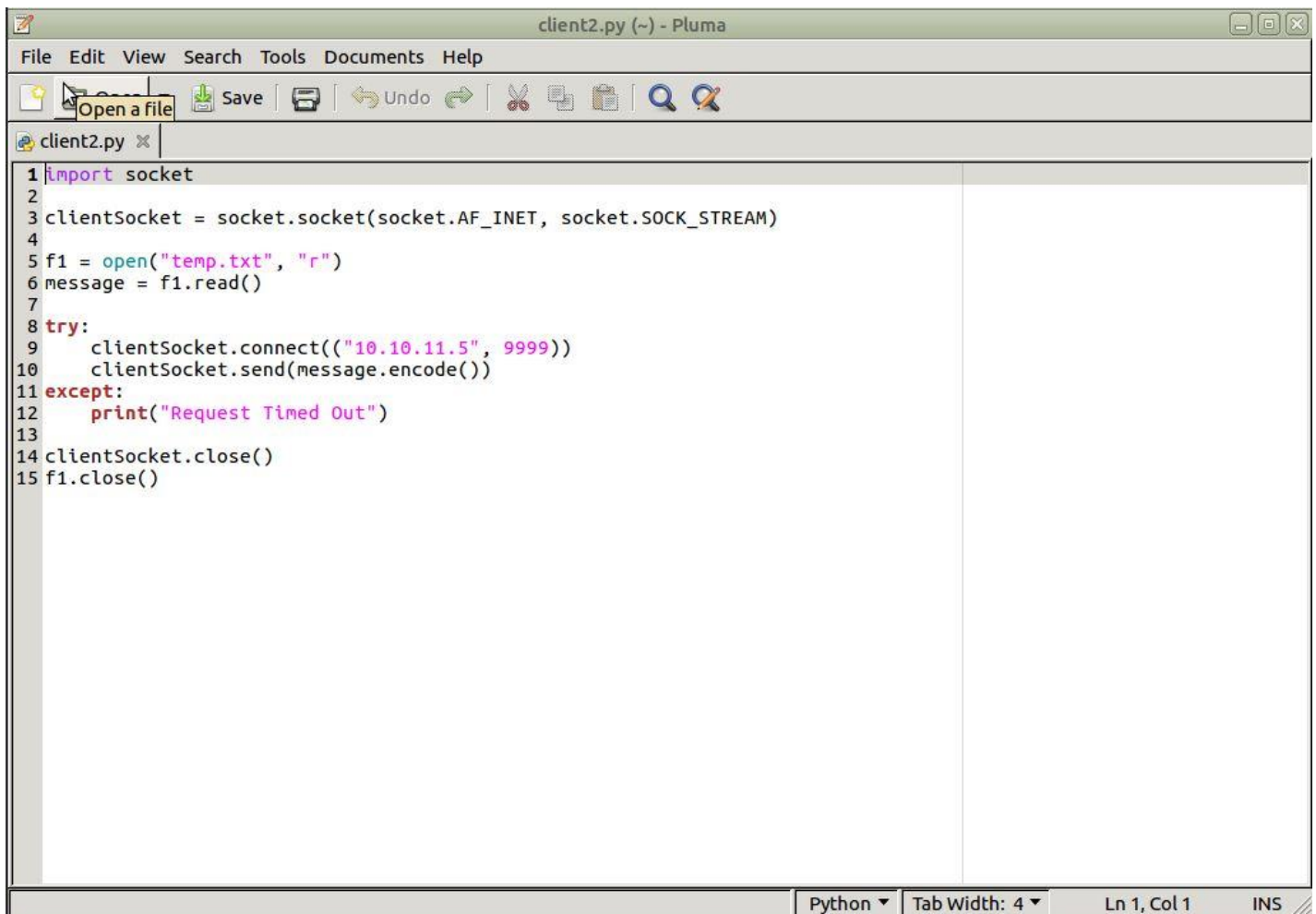
The status bar at the bottom shows 'Python', 'Tab Width: 4', 'Ln 1, Col 1', and 'INS'.

Capture 9 Server Program to write data to file

1. Create socketserver and check for flag if true.
2. In Line 11-12, the server receives the data and decodes it.
3. The server then opens the file, if not created then a new file is created and writes the received data on it.
4. The data received is also logged for the user.
5. The connection is then closed.

Client:

- a) Connect with the server
- b) Send the file
- c) close the connection



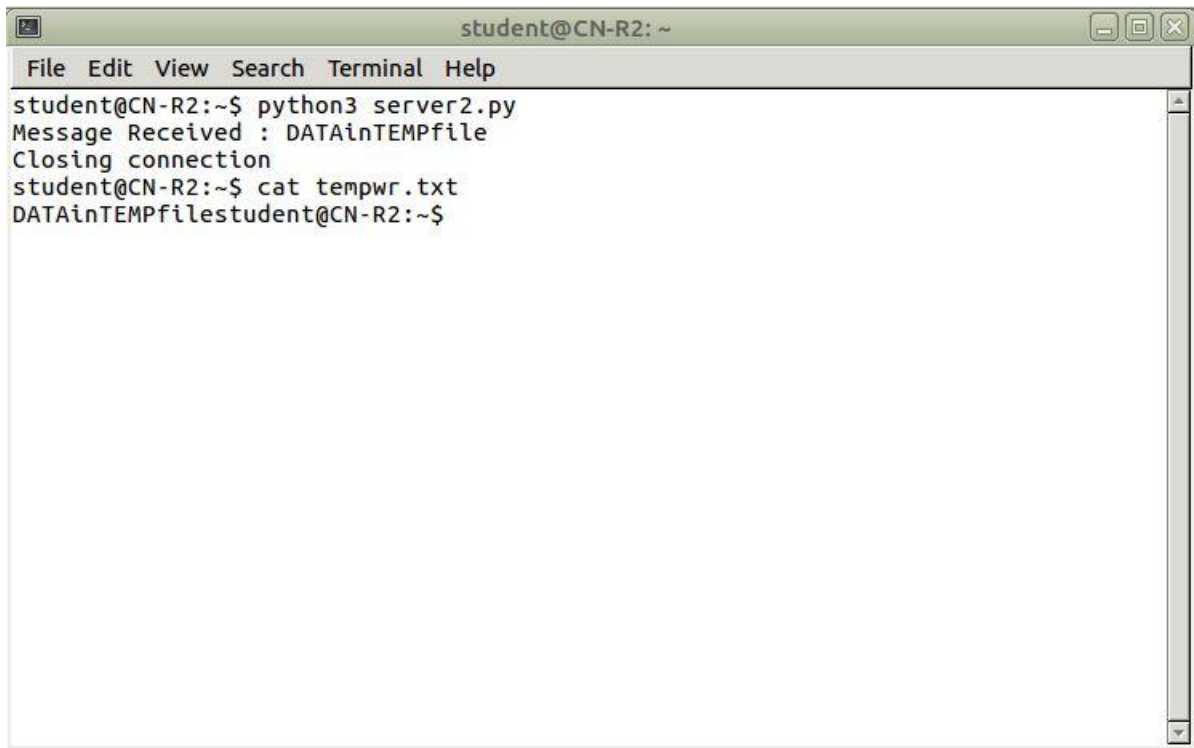
The screenshot shows a text editor window titled "client2.py (~) - Pluma". The window has a menu bar (File, Edit, View, Search, Tools, Documents, Help) and a toolbar with icons for opening files, saving, undo, redo, cut, copy, paste, find, and replace. The editor contains the following Python code:

```
1 import socket
2
3 clientSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
4
5 f1 = open("temp.txt", "r")
6 message = f1.read()
7
8 try:
9     clientSocket.connect(("10.10.11.5", 9999))
10    clientSocket.send(message.encode())
11 except:
12     print("Request Timed Out")
13
14 clientSocket.close()
15 f1.close()
```

The status bar at the bottom indicates "Python", "Tab Width: 4", "Ln 1, Col 1", and "INS".

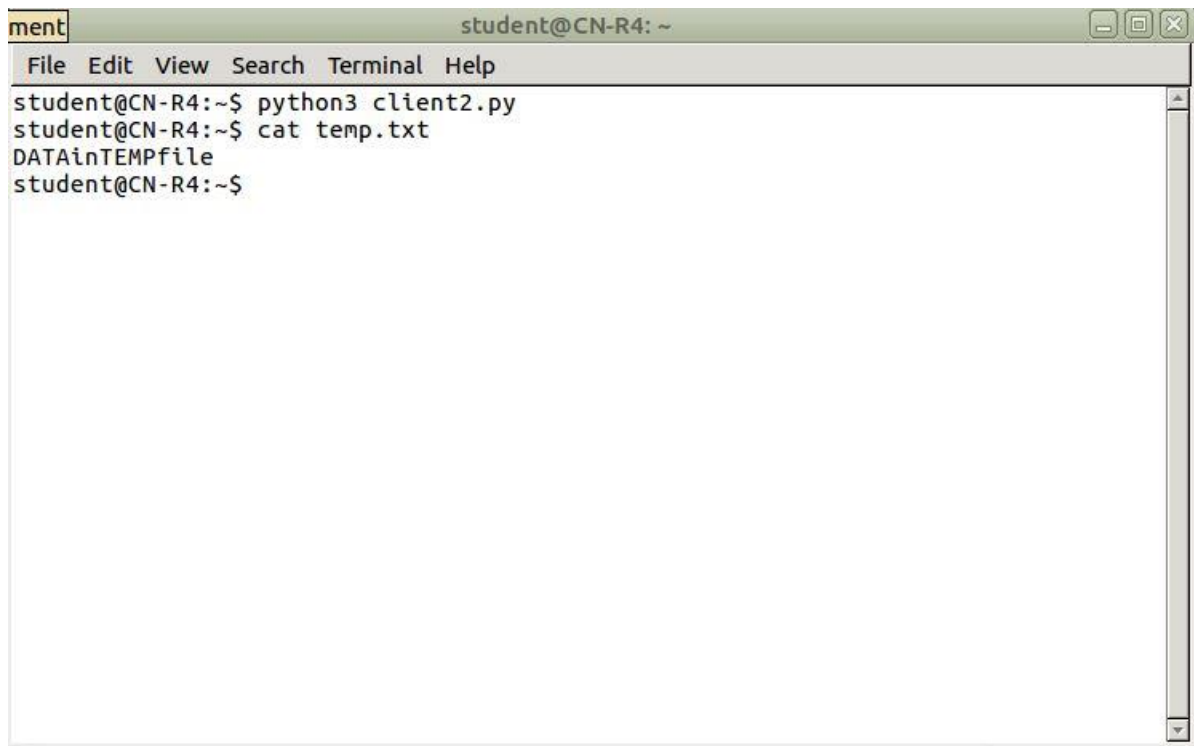
Capture 10 Client Program to read data from file and send to the server

1. Socket created.
2. In Line 5-6, the client opens and reads a file.
3. The client then sends the read data over to the server where the server stores the data in a file.
4. Socket closed.

A terminal window titled 'student@CN-R2: ~' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the execution of a Python server program. The user runs 'python3 server2.py', which outputs 'Message Received : DATAinTEMPfile' and 'Closing connection'. Then, the user runs 'cat tempwr.txt', which outputs 'DATAinTEMPfilestudent@CN-R2:~\$'.

```
student@CN-R2:~$ python3 server2.py
Message Received : DATAinTEMPfile
Closing connection
student@CN-R2:~$ cat tempwr.txt
DATAinTEMPfilestudent@CN-R2:~$
```

Capture 11 Server Program Example

A terminal window titled 'student@CN-R4: ~' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the execution of a Python client program. The user runs 'python3 client2.py', which outputs 'DATAinTEMPfile'. Then, the user runs 'cat temp.txt', which also outputs 'DATAinTEMPfile'. The terminal ends with the prompt 'student@CN-R4:~\$'.

```
student@CN-R4:~$ python3 client2.py
DATAinTEMPfile
student@CN-R4:~$ cat temp.txt
DATAinTEMPfile
student@CN-R4:~$
```

Capture 12 Client Program Example