Overview

Each function has been implemented, as per the project specification. Here is a command by command breakdown.

login - Login is one of two commands that the client can use before logged in. It takes one command, ID, and passes that to the server to login. If the server is able to validate the ID, it sends back a confirmation and the client is allowed to continue running. If the server is unable to validate the ID, the user is prompted to try logging in again. Upon a successful login, the user's ID will be displayed next to each further prompt and login will be disabled until the user logs out.

help - This command prints the help menu out to the client. It includes the command list from the README, and is shown below.

```
login <#> - Takes one argument, your user ID. Will log you into the server to access the forum. You must login before
you can access any of the other commands. Your user ID must be only alphanumerics.
help - Displays this help menu.
ag [<#>] - Has one optional argument. Returns a list of all existing discussion groups, N groups at a time. If the
argument is not provided, a default value of 5 will be used. When in ag mode,  the following subcommands are available.
    s - Subscribe to groups. Takes between 1 and N arguments. Subscribes to all groups listed in the argument.
    u - Unsubscribe to groups. Same functionality as s, but instead unsubscribes.
    n - Lists the next N discussion groups. If there are no more to display, exits ag mode.
    q - Exits from ag mode.
sg [<#>] - Has one optional argument. Returns a list of all subscribed groups, N groups at a time. If the argument is
not provided, then a default value of 5 will be used.
    u - Unsubscribe to groups. See ag.
    n - Lists the next N discussion groups. If there are no more to display, exits sg mode.
    q - Exits from sg mode.
rg <gname> [<#>] - Takes one mandatory argument and one optional argument. It displays the top N posts in a given
discussion group. The argument, 'gname' determines which group to display. If the optional argument is not provided,
then a default value of 5 will be used. After entering this command, the application will enter 'rg' mode, which uses
the following subcommands.
    [#] - The post number to display. Entering this mode will give even more subcommands.
        n - Display, at most, n more lines of content.
        q - Quit this read mode to return to normal rg mode
    r [#] - Marks the given post as read. If a single number is specified, then that single post will be marked as read.
    You can use the format, 'n-m' to mark posts n through m as read.
    n - Lists the next n posts. If there are no more posts to display, exits rg mode.
    p - Post to the group. This subcommand will enter post mode.
        The application will request a title for the post. Then, the application will allow you to write the body of the
        post. It will accept input until you enter a blank line, followed by a period, followed by another blank line.
        After this command sequence is accepted, the post will be submitted and you will be returned to rg mode.
logout - Logs you out from the server, subsequently closing the application.
```

ag - This command takes in one optional argument, N. It uses a default value if N is not provided. ag will list all groups available, N at a time. This command switches the user to ag mode and stays in ag mode until either 1) the user uses the q command or 2) the user uses the n command when there are no more groups to show. If a user is subscribed to a group, there will be an "s" next to the given group. In ag mode, the user can use the following commands:

      n - Displays the next N groups, or the rest of the groups if N groups are not available

s <#> - Subscribes the user to the given number group. The user can input multiple groups to subscribe to them many at a time. Example: "s 1 2 5" subscribes the user to groups 1, 2, and 5.

u <#> - Similar functionality to s, except unsubscribes the user from those given groups.

q - Exits the user from ag mode

sg - This command takes one optional argument, N. It runs very similar to ag except it only displays the groups the user is subscribed to. It switches the user to sg mode. It will list the groups the user is subscribed to, 1 through N. If the user is not subscribed to any groups, then the command will exit immediately. The number between the number of the group and the name of the group is the number of new posts in the group with a blank if the user has read every post in that group. The command borrows the n, u, and q commands from ag.

rg - This command takes in one mandatory argument, GNAME, and one optional argument, N. If N is not provided, it will use a default value. It moves the user into rg mode. It lists all the posts in the given group GNAME, 1-N. It will list the posts in descending order of date. It lists the post's date of publication, title, and status of new or not. rg mode also has the following commands:

<#> - Read the given post number. This will put the user into read mode. It will list the group, subject, date, and author of the post, followed by the post content. The user is then returned to rg mode.

n - List the following N posts, or the remaining posts if there are more than N posts remaining.

q - Quit rg mode.

p - Switches the user to post mode. The user will be prompted for a title. After the user submits their title, they will be prompted for their post content. The program will take in input until the user enters in the escape sequence which is designated as follows: "Enter, followed by a period, followed by another enter." After the command sequence is found, the user will be shown their message and asked if that is adequeate. They will be prompted to answer "Y" or "N" to determine whether or not to submit the message. If the user submits, "Y", the message is sent to the server and published. If the user submits, "N", the post is discarded. The user can submit "/help" at any time to see a printout of the help menu. This input is not included in their message. An example of a post is as follows:

```
Login successful! Welcome user TestUser
TestUser>>> rg comp.lang.c
comp.lang.c found!
1.  N   2016-12-03 00:01:31 CSE 320 Help?
2.  N   2016-12-03 00:01:23 Welcome to comp.lang.c!
TestUser(RG Mode)>>> p
TestUser(Enter title)>>> This is a test post
TestUser(Compose Post)>>> Here is an example of how you can compose a post.
TestUser(Compose Post)>>> Hitting enter does not stop composition mode.
TestUser(Compose Post)>>> You can use this to enter in many lines until the stop sequence is found.
TestUser(Compose Post)>>> The stop sequence is as follows:
TestUser(Compose Post)>>>
TestUser(Compose Post)>>> .
TestUser(Compose Post)>>>
Title: This is a test post
Content:
Here is an example of how you can compose a post.
Hitting enter does not stop composition mode.
You can use this to enter in many lines until the stop sequence is found.
The stop sequence is as follows:
Submit this message?
TestUser(Y or N)>>>
```

logout - This command will safely log the user out from the server and then close the application.

User documentation

Both the server and the client are written in Python 3. As such it *will not* work in Python 2. You would run both scripts just as you would run any other Python 3 script on your machine.

Server Specifics

The Server.py file requires no arguments to launch.

The only file that the server absolutely needs to run is "Server.py", but for there to be any functionality with the server, a "groups.txt" files must also be included. If the Server.py file is run without the groups.txt file, the server will still startup but will issue a warning that it's basically useless. The server creates the "groups" subfolder from the given groups.txt file. The server will also create a "users.txt" file to keep track of the clients that connect.

The server will bind to the IP address of the computer on which it is hosted but listens to hardcoded port 9966.

The server starts immediately once the script is ran. There are several helpful print statements to see that the server is responding to users. For example, when a new user connects,

it will print out statements detailing the thread created for that user and show the connection status of that user. When that same user disconnects, statements will be printed out to show that that user's thread has been shut down.

Each command has their own corresponding print statement. Here is an example printout. In this example, a user has connected, logged in, used the ag, sg, and rg commands, and then logged out.

```
User file found! Loading users...
Groups file found! Loading groups...
Beginning server.
Listening on port 9966
Login Thread running
Connection received from 127.0.0.1:52749
Running thread 0 for 127.0.0.1:52749
Login received from Thread0@127.0.0.1:52749
Valid login from Thread0@127.0.0.1:52749(User: TestUser)
Sent all groups to Thread0@127.0.0.1:52749(User: TestUser)
Sent all groups to Thread0@127.0.0.1:52749(User: TestUser) for use with sg.
Group 'comp.lang.python' requested from Thread0@127.0.0.1:52749(User: TestUser)
Client from Thread0@127.0.0.1:52749(User: TestUser) has disconnected.
Closing Thread0@127.0.0.1:52749(User: TestUser)
```

There are also print out statements for behavior such as an invalid login is received or when the client disconnects unexpectedly. Here is an example of that.

```
User file found! Loading users...
Groups file found! Loading groups...
Beginning server.
Listening on port 9966
Login Thread running
Connection received from 127.0.0.1:52782
Running thread 0 for 127.0.0.1:52782
Login received from Thread0@127.0.0.1:52782
Invalid login from Thread0@127.0.0.1:52782
Login received from Thread0@127.0.0.1:52782
Valid login from Thread0@127.0.0.1:52782(User: TestUser)
Client from Thread0@127.0.0.1:52782(User: TestUser) has disconnected unexpectedly.
Closing Thread0@127.0.0.1:52782(User: TestUser)
```

Client Specifics

The Client.py has two mandatory arguments, address and port. The script will abort if more or fewer than two arguments are found.

The client requires no additional files to launch properly. All files that are associated with the client program can be created after launch. Upon a successful login, the program will create a folder titled as the ID the user used to login. For example, if "TestUser" logged in, a folder titled,

"TestUser" would be created in the same folder as the Client.py script. During the course of normal use, several folders and files will be made inside that folder. Once a user subscribes to a group, a "subscriptions.txt" file will be created inside that folder. That file holds the name of each subscribed group for that user. When a user subscribes to a group, a folder with the name of that group will also be created. When a user reads a post from that group or marks a post from that group as read, some information about the post, such as date, author, and title, is saved inside that folder.

Here is a printout of the output of the help command. It details the usage of each command that the client can use. A copy is also in the README.

```
login <#> - Takes one argument, your user ID. Will log you into the server to access the forum. You must login before
you can access any of the other commands. Your user ID must be only alphanumerics.
help - Displays this help menu.
ag [<#>] - Has one optional argument. Returns a list of all existing discussion groups, N groups at a time. If the
argument is not provided, a default value of 5 will be used. When in ag mode,  the following subcommands are available.
    s - Subscribe to groups. Takes between 1 and N arguments. Subscribes to all groups listed in the argument.
    u - Unsubscribe to groups. Same functionality as s, but instead unsubscribes.
    n - Lists the next N discussion groups. If there are no more to display, exits ag mode.
    q - Exits from ag mode.
sg [<#>] - Has one optional argument. Returns a list of all subscribed groups, N groups at a time. If the argument is
not provided, then a default value of 5 will be used.
    u - Unsubscribe to groups. See ag.
    n - Lists the next N discussion groups. If there are no more to display, exits sg mode.
    q - Exits from sg mode.
rg <gname> [<#>] - Takes one mandatory argument and one optional argument. It displays the top N posts in a given
discussion group. The argument, 'gname' determines which group to display. If the optional argument is not provided,
then a default value of 5 will be used. After entering this command, the application will enter 'rg' mode, which uses
the following subcommands.
    [#] - The post number to display. Entering this mode will give even more subcommands.
        n - Display, at most, n more lines of content.
        q - Quit this read mode to return to normal rg mode
    r [#] - Marks the given post as read. If a single number is specified, then that single post will be marked as read.
    You can use the format, 'n-m' to mark posts n through m as read.
    n - Lists the next n posts. If there are no more posts to display, exits rg mode.
    p - Post to the group. This subcommand will enter post mode.
        The application will request a title for the post. Then, the application will allow you to write the body of the
        post. It will accept input until you enter a blank line, followed by a period, followed by another blank line.
        After this command sequence is accepted, the post will be submitted and you will be returned to rg mode.
logout - Logs you out from the server, subsequently closing the application.
```

System documentation

Here is an excerpt from the README file pertaining to the many aspects of the system.

Protocol:

The server and client communicate through various keywords. Here are all included keywords:
LOGIN
AG
SG
RG
LOGOUT

ERROR
SD
POST

The client and server communicate as follows:

1. A keyword is sent, followed by a space. On both ends, the messages are parsed by space, ignoring spaces between ' '.

2. If the keyword has optional information sent along with it, then all of that information goes in the next few spaces. Different arguments should be separated by spaces.

3. Both the client and server receive data until an End Of Message (EOM) is found. The EOM is determined by the following sequence: '\r\n\r\n'. This includes the quotation marks. The quotation marks are necessary because of the way the parser functions.

LOGIN sent from the client to the server must include the userID as a second argument.

RG Protocol
RG, since it has the most server functionality, has its own protocol for sending discussion posts to the client. When the client sends an RG request, the server will respond with the RG keyword, followed by the number of total posts for the group. The server will then send a message as follows: "RG_KEYWORD N FILE.TXT DATE AUTHOR TITLE CONTENTS EOM" where N is the number of the post starting at 1 and going to the max number of discussion posts, DATE is the date the file was created, AUTHOR is the creator of the post, TITLE is the title, and FILE.TXT is the name of the file. CONTENTS is the content of the file, which can be sent across several messages based on the size of the post.

If the ERROR keyword is sent from the server to the client it will be accompanied by an error code number. Here is a list of all error numbers and what they mean:
0 - Command not found
1 - Login invalid
10 - When using the RG command, group not found

PERSISTENCE AND I/O

All persistent information is stored in txt files. If a given file is not found, then it is created before it is first used. Here is a list of all files used by the program:
Server files:
        users.txt
        groups.txt
Client files:
        Client files are stored in a folder that is named after their userId. This is why userIds can only be named with numbers and characters.
Files that are stored in the user folder:
        subscription.txt

Testing documentation

1. Test Case 1: A new user logs into the server, views all groups (8 at a time), subscribes to one group, and then subscribes to 2 groups with one command, before exiting.

```
C:\Windows\system32\cmd.exe - python Server.py

D:\Coding\PyCharm\CSE310Project>python Server.py
User file found! Loading users...
Groups file found! Loading groups...
Traceback (most recent call last):
  File "Server.py", line 349, in <module>
    serverSocket.bind(("", serverPort))
OSError: [WinError 10048] Only one usage of each socket address (protocol/networ
k address/port) is normally permitted

D:\Coding\PyCharm\CSE310Project>python Server.py
User file found! Loading users...
Groups file found! Loading groups...
Beginning server.
Listening on port 9966
Login Thread running
Connection received from 127.0.0.1:54370
Running thread 0 for 127.0.0.1:54370
Login received from Thread0@127.0.0.1:54370
New user found. Added user NewUser
Valid login from Thread0@127.0.0.1:54370(User: NewUser)
Sent all groups to Thread0@127.0.0.1:54370(User: NewUser)
Client from Thread0@127.0.0.1:54370(User: NewUser) has disconnected.
Closing Thread0@127.0.0.1:54370(User: NewUser)
```

```
C:\Windows\system32\cmd.exe

Attempting to connect to localhost:9966
Connection successful!
>>> login NewUser
NewUser
Login successful! Welcome user NewUser
NewUser>>> ag 8
1. ( ) comp.programming
2. ( ) comp.os.threads
3. ( ) comp.lang.c
4. ( ) comp.lang.python
5. ( ) comp.lang.javascript
6. ( ) sci.bio
7. ( ) sci.chem
8. ( ) sci.physics
NewUser(AG Mode)>>> n
9. ( ) sci.geology
10. ( ) sci.astronomy
11. ( ) art.drawing
12. ( ) art.writing
13. ( ) art.music
14. ( ) art.acting
15. ( ) art.painting
NewUser(AG Mode)>>> s 7
Subscribed to sci.chem
NewUser(AG Mode)>>> s 10 9
Subscribed to sci.astronomy
Subscribed to sci.geology
NewUser(AG Mode)>>> n
Exiting AG Mode
NewUser>>> logout
Logging out from the server...
Received from server: Logging you out from the server.

D:\Coding\PyCharm\CSE310Project>
```

2. Test Case 2: The same user from Test Case 1 logs back in, views subscribed groups, unsubscribes from one group, then views all groups.

```
C:\Windows\system32\cmd.exe - python Server.py
      serverSocket.bind(("", serverPort))
OSError: [WinError 10048] Only one usage of each socket address (protocol/networ
k address/port) is normally permitted

D:\Coding\PyCharm\CSE310Project>python Server.py
User file found! Loading users...
Groups file found! Loading groups...
Beginning server.
Listening on port 9966
Login Thread running
Connection received from 127.0.0.1:54370
Running thread 0 for 127.0.0.1:54370
Login received from Thread0@127.0.0.1:54370
New user found. Added user NewUser
Valid login from Thread0@127.0.0.1:54370(User: NewUser)
Sent all groups to Thread0@127.0.0.1:54370(User: NewUser)
Client from Thread0@127.0.0.1:54370(User: NewUser) has disconnected.
Closing Thread0@127.0.0.1:54370(User: NewUser)
Connection received from 127.0.0.1:54426
Running thread 1 for 127.0.0.1:54426
Login received from Thread1@127.0.0.1:54426
Valid login from Thread1@127.0.0.1:54426(User: NewUser)
Sent all groups to Thread1@127.0.0.1:54426(User: NewUser) for use with sg.
Sent all groups to Thread1@127.0.0.1:54426(User: NewUser)
```

```
C:\Windows\system32\cmd.exe - python Client.py localhost 9966
Subscribed to sci.geology
NewUser(AG Mode)>>> n
Exiting AG Mode
NewUser>>> logout
Logging out from the server...
Received from server: Logging you out from the server.

D:\Coding\PyCharm\CSE310Project>python Client.py localhost 9966
Attempting to connect to localhost:9966
Connection successful!
>>> login NewUser
NewUser
Login successful! Welcome user NewUser
NewUser>>> sg
1.      1       sci.chem
2.      1       sci.astronomy
3.      1       sci.geology
NewUser(SG Mode)>>> u 1
Unsubscribed to sci.chem
NewUser(SG Mode)>>> q
Exiting SG Mode
NewUser>>> ag 15
1. ( ) comp.programming
2. ( ) comp.os.threads
3. ( ) comp.lang.c
4. ( ) comp.lang.python
5. ( ) comp.lang.javascript
6. ( ) sci.bio
7. ( ) sci.chem
8. ( ) sci.physics
9. (s) sci.geology
10. (s) sci.astronomy
11. ( ) art.drawing
12. ( ) art.writing
13. ( ) art.music
14. ( ) art.acting
15. ( ) art.painting
NewUser(AG Mode)>>>
```

3. Test Case 3: The same user from Test Case 1 and 2, still logged in, views a post, then makes a post and submits it, in a subscribed group. Then, marks it as read.

Server window:

```
C:\Windows\system32\cmd.exe - python  Server.py

D:\Coding\PyCharm\CSE310Project>python Server.py
User file found! Loading users...
Groups file found! Loading groups...
Beginning server.
Listening on port 9966
Login Thread running
Connection received from 127.0.0.1:54370
Running thread 0 for 127.0.0.1:54370
Login received from Thread0@127.0.0.1:54370
New user found. Added user NewUser
Valid login from Thread0@127.0.0.1:54370(User: NewUser)
Sent all groups to Thread0@127.0.0.1:54370(User: NewUser)
Client from Thread0@127.0.0.1:54370(User: NewUser) has disconnected.
Closing Thread0@127.0.0.1:54370(User: NewUser)
Connection received from 127.0.0.1:54426
Running thread 1 for 127.0.0.1:54426
Login received from Thread1@127.0.0.1:54426
Valid login from Thread1@127.0.0.1:54426(User: NewUser)
Sent all groups to Thread1@127.0.0.1:54426(User: NewUser) for use with sg.
Sent all groups to Thread1@127.0.0.1:54426(User: NewUser)
Group 'sci.astronomy' requested from Thread1@127.0.0.1:54426(User: NewUser)
New post received from Thread1@127.0.0.1:54426(User: NewUser)
Group 'sci.astronomy' requested from Thread1@127.0.0.1:54426(User: NewUser)
Group 'sci.astronomy' requested from Thread1@127.0.0.1:54426(User: NewUser)
```

Client window:

```
C:\Windows\system32\cmd.exe - python  Client.py localhost 9966

NewUser(AG Mode)>>>> q
Exiting AG Mode
NewUser>>> rg sci.astronomy
sci.astronomy found!
1.       N        2016-12-03 00:14:12        Welcome to sci.astronomy!
NewUser(RG Mode)>>>> 1
Group: sci.astronomy
Subject: Welcome to sci.astronomy!
Author: Admin
Date: 2016-12-03 00:14:12
Welcome to the sci.astronomy forum! Discuss things pertaining to astronomy.
NewUser(RG Mode)>>>> p
NewUser(Enter title)>>>> I really like space
NewUser(Compose Post)>>>> Hey guys. I really enjoy reading about NASA's recent mi
ssions.
NewUser(Compose Post)>>>> I was wondering if anyone could link me some websites s
o I could read up on the history of the Apollo missions.
NewUser(Compose Post)>>>> Thank you in advance!
NewUser(Compose Post)>>>>
NewUser(Compose Post)>>>> .
NewUser(Compose Post)>>>>
Title: I really like space
Content:
Hey guys. I really enjoy reading about NASA's recent missions.
I was wondering if anyone could link me some websites so I could read up on the
history of the Apollo missions.
Thank you in advance!
Submit this message?
NewUser(Y or N)>>>> Y
Submitting message...
NewUser(RG Mode)>>>> q
Exiting RG Mode
NewUser>>> rg sci.astronomy
sci.astronomy found!
1.       N        2016-12-10 22:58:24        I really like space
2.                2016-12-03 00:14:12        Welcome to sci.astronomy!
NewUser(RG Mode)>>>> r 1
NewUser(RG Mode)>>>> q
Exiting RG Mode
NewUser>>> rg sci.astronomy
sci.astronomy found!
1.                2016-12-10 22:58:24        I really like space
2.                2016-12-03 00:14:12        Welcome to sci.astronomy!
NewUser(RG Mode)>>>> q
Exiting RG Mode
NewUser>>> _
```

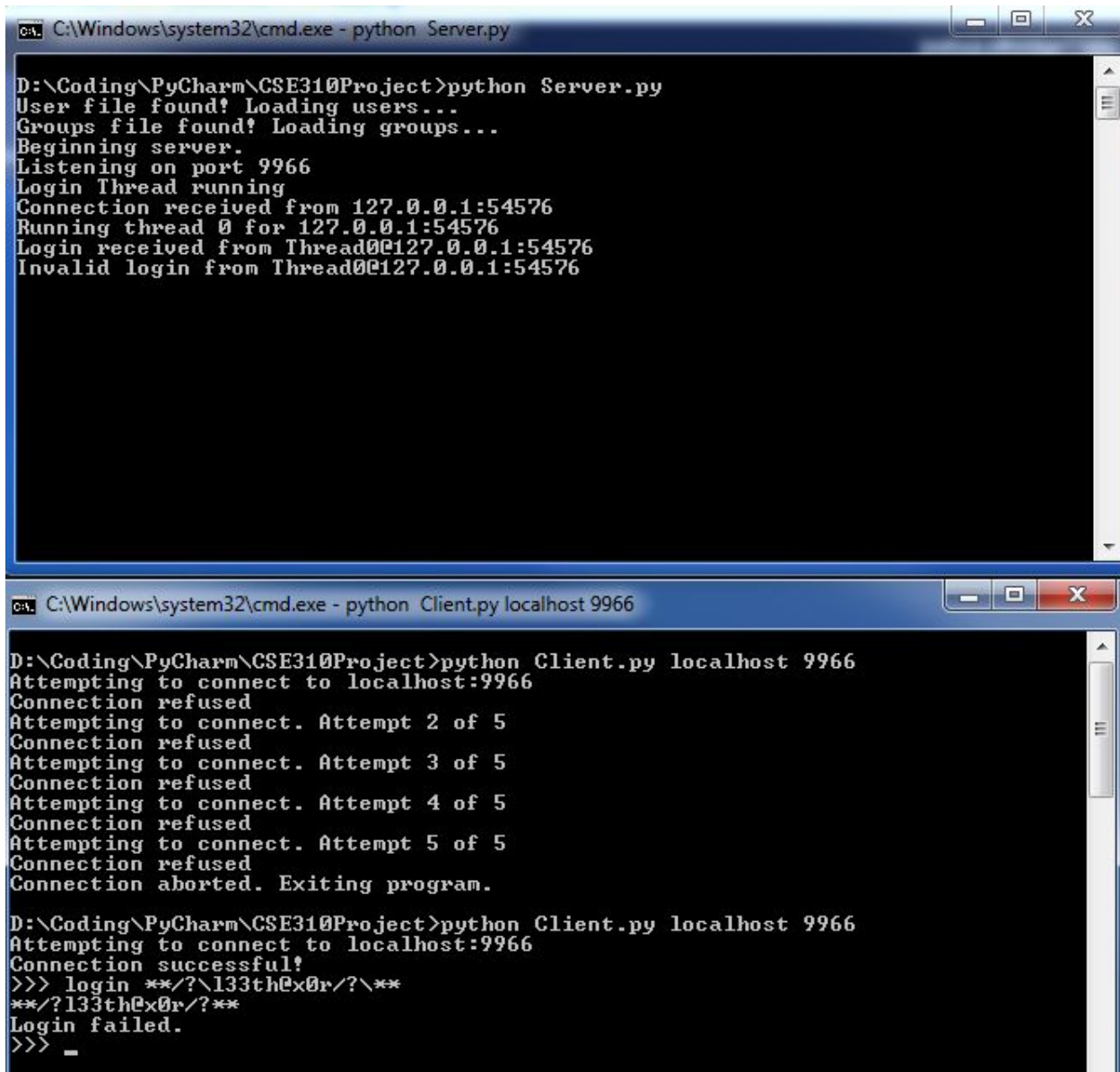4. Test Case 4: A user attempts to connect when the server is not running.

```
C:\Windows\system32\cmd.exe
```

```
D:\Coding\PyCharm\CSE310Project>
```

```
C:\Windows\system32\cmd.exe
```

```
D:\Coding\PyCharm\CSE310Project>python Client.py localhost 9966
Attempting to connect to localhost:9966
Connection refused
Attempting to connect. Attempt 2 of 5
Connection refused
Attempting to connect. Attempt 3 of 5
Connection refused
Attempting to connect. Attempt 4 of 5
Connection refused
Attempting to connect. Attempt 5 of 5
Connection refused
Connection aborted. Exiting program.

D:\Coding\PyCharm\CSE310Project>
```

5. Test Case 5: A user attempts to login with an invalid name.


```
C:\Windows\system32\cmd.exe - python Server.py

D:\Coding\PyCharm\CSE310Project>python Server.py
User file found! Loading users...
Groups file found! Loading groups...
Beginning server.
Listening on port 9966
Login Thread running
Connection received from 127.0.0.1:54576
Running thread 0 for 127.0.0.1:54576
Login received from Thread0@127.0.0.1:54576
Invalid login from Thread0@127.0.0.1:54576
```

```
C:\Windows\system32\cmd.exe - python Client.py localhost 9966

D:\Coding\PyCharm\CSE310Project>python Client.py localhost 9966
Attempting to connect to localhost:9966
Connection refused
Attempting to connect. Attempt 2 of 5
Connection refused
Attempting to connect. Attempt 3 of 5
Connection refused
Attempting to connect. Attempt 4 of 5
Connection refused
Attempting to connect. Attempt 5 of 5
Connection refused
Connection aborted. Exiting program.

D:\Coding\PyCharm\CSE310Project>python Client.py localhost 9966
Attempting to connect to localhost:9966
Connection successful!
>>> login **/?\l33th@x0r/?\**
**/?l33th@x0r/?**
Login failed.
>>> _
```

6. Test Case 6: A client's window is closed while still connected to the server.

```
C:\Windows\system32\cmd.exe - python Server.py

D:\Coding\PyCharm\CSE310Project>python Server.py
User file found! Loading users...
Groups file found! Loading groups...
Beginning server.
Listening on port 9966
Login Thread running
Connection received from 127.0.0.1:54576
Running thread 0 for 127.0.0.1:54576
Login received from Thread0@127.0.0.1:54576
Invalid login from Thread0@127.0.0.1:54576
Client from Thread0@127.0.0.1:54576 has disconnected unexpectedly.
Closing Thread0@127.0.0.1:54576
_
```