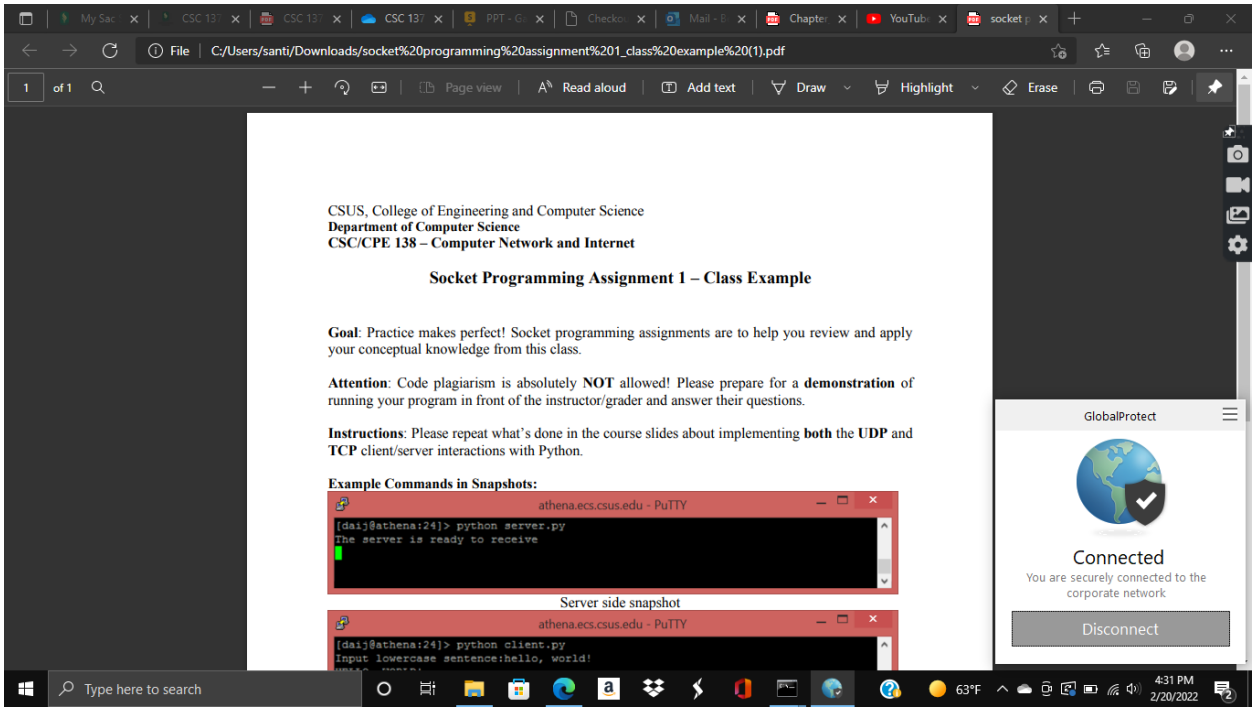
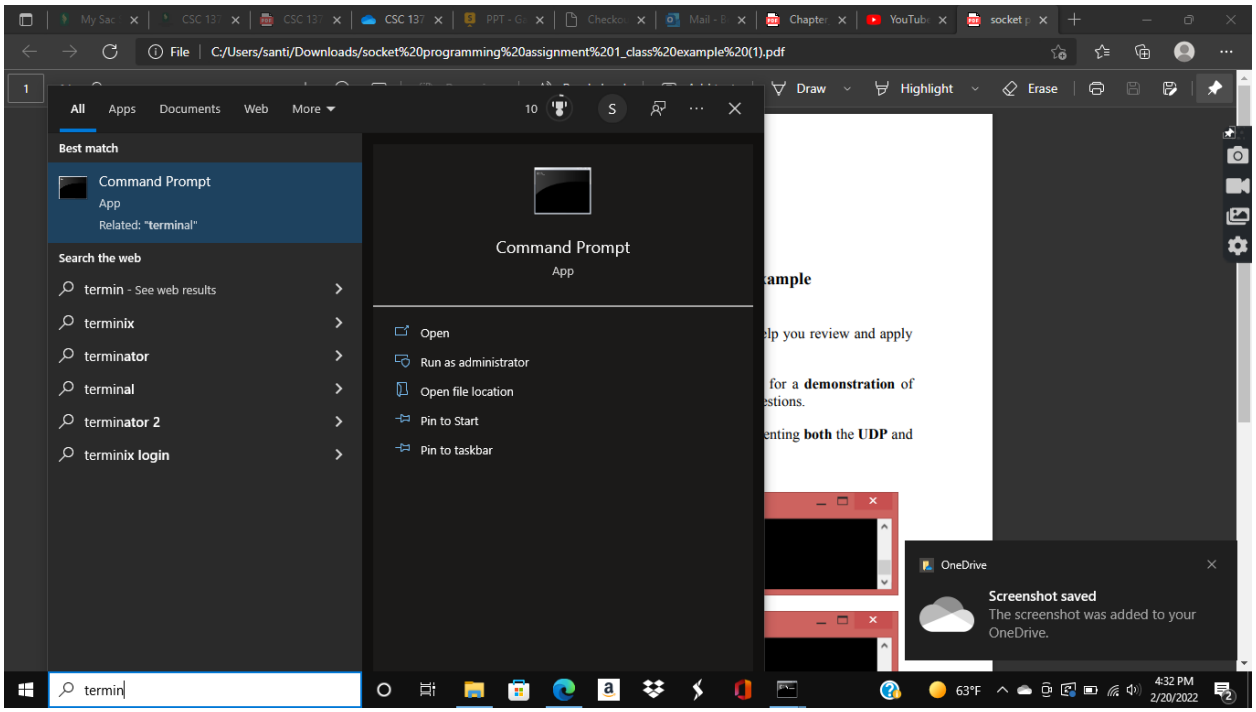
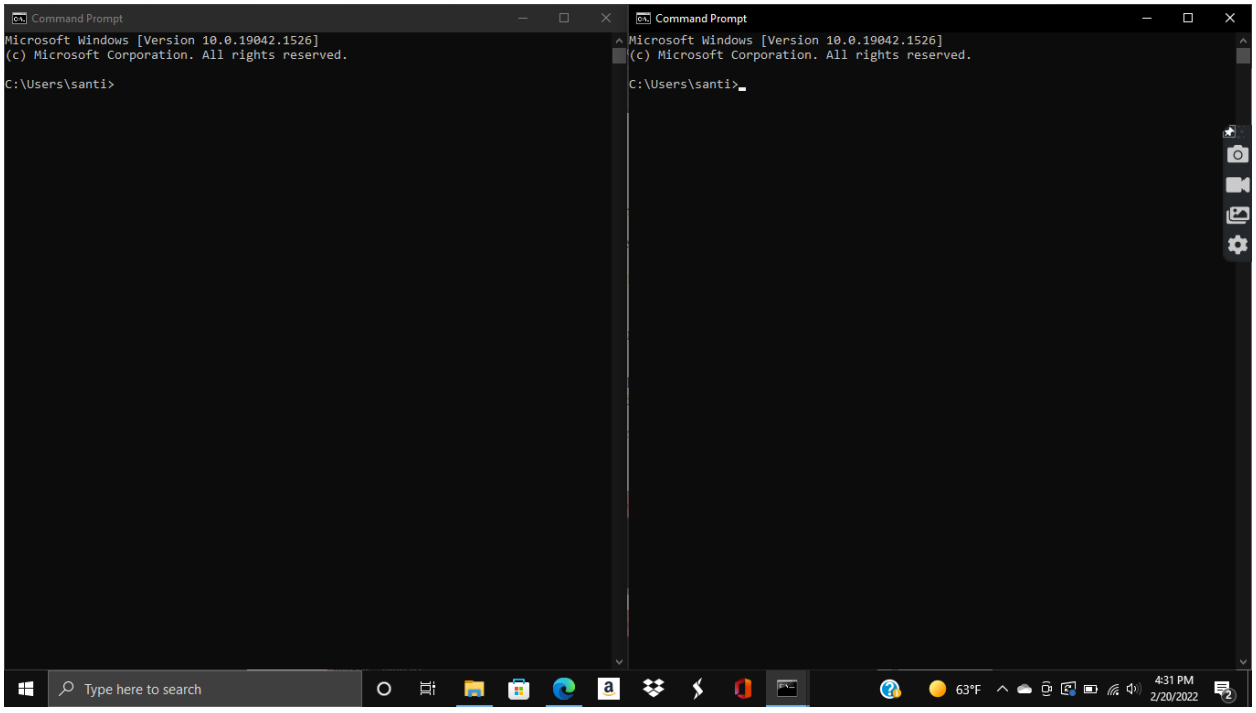
For starters, you will want to make sure that you have GlobalProtectVPN installed and connected. As you can see in the lower right of the screenshot posted below, I already have GlobalProtectVPN installed and connected.



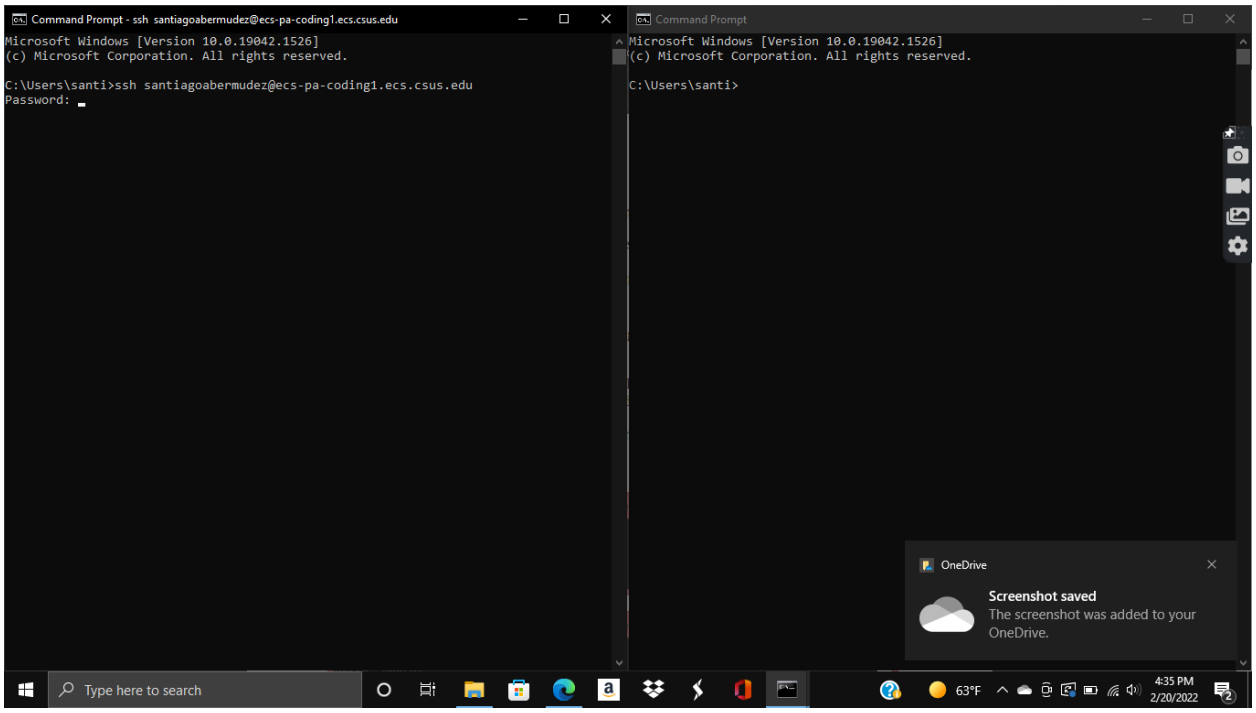
Next, you will want to have two Command Prompt or Terminal windows opened. On Windows 10, you can simply use the search bar below and search for either 'terminal' or 'command prompt'.



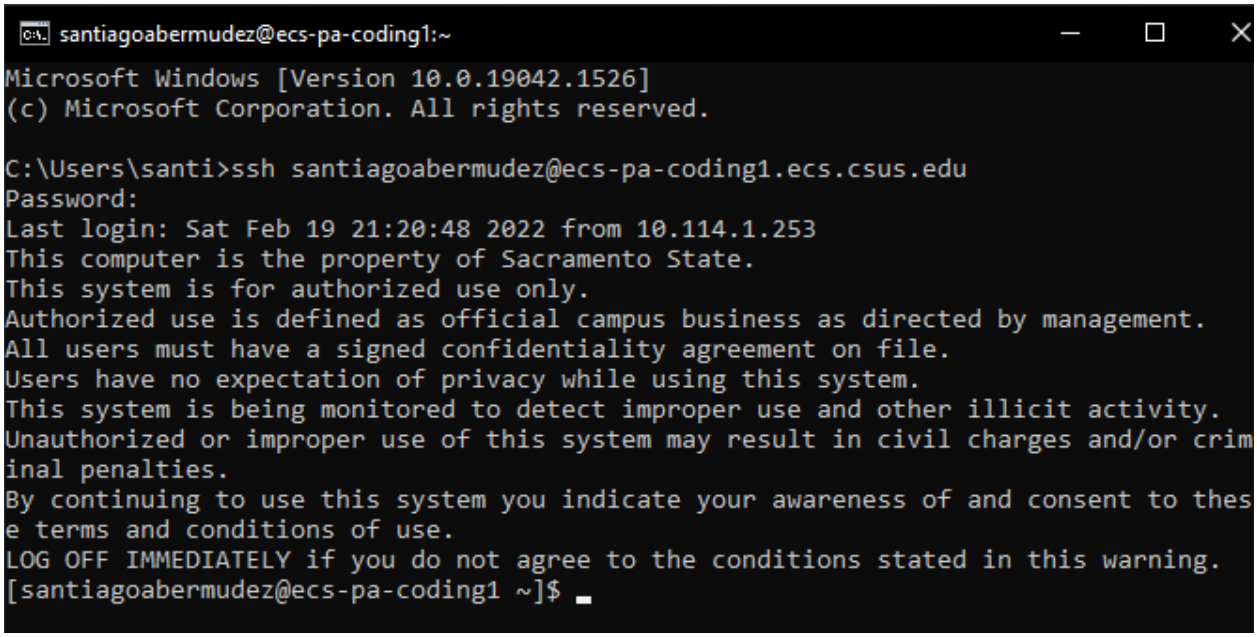
From there, you will want to open two terminal windows. One window will run the client code and one will run the server code. More details on this later.



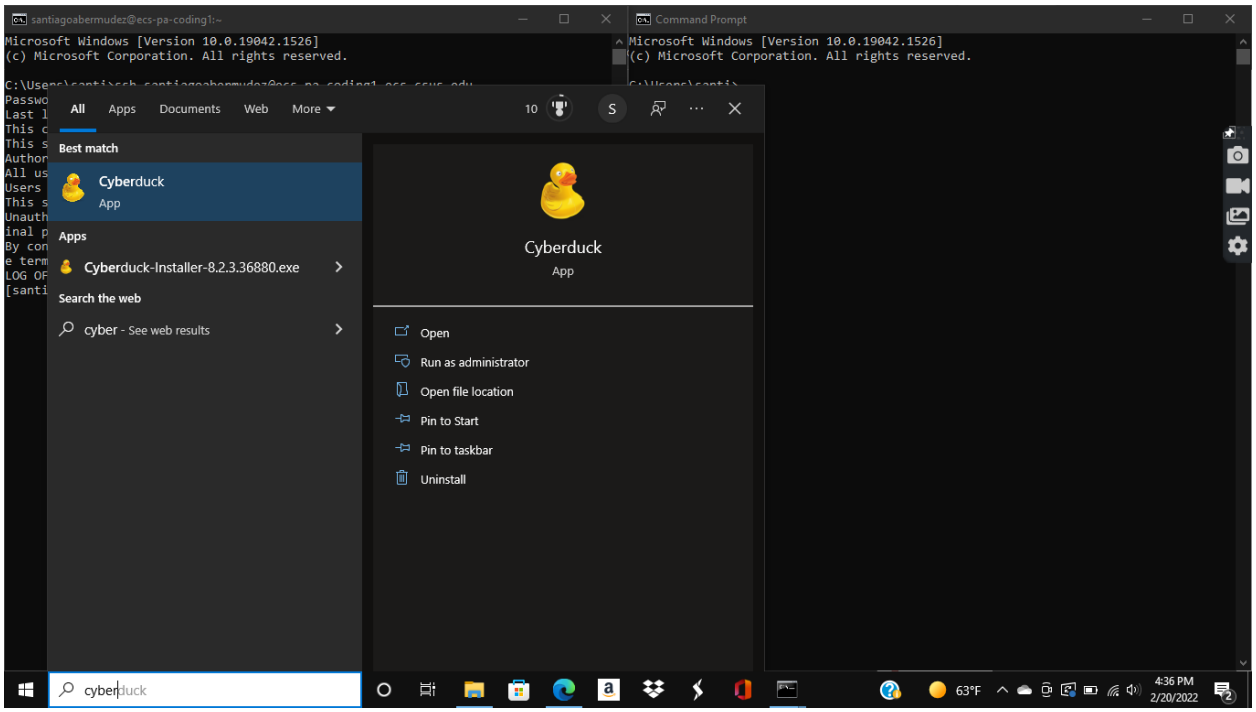
After, you will want to start signing in to the ECS server using the following command: 'ssh yourSacStateUsername@ecs-pa-coding1.ecs.csus.edu'. If successful, you will be asked for a password, which should be the same as what you normally use for the student center, logging in, etc.

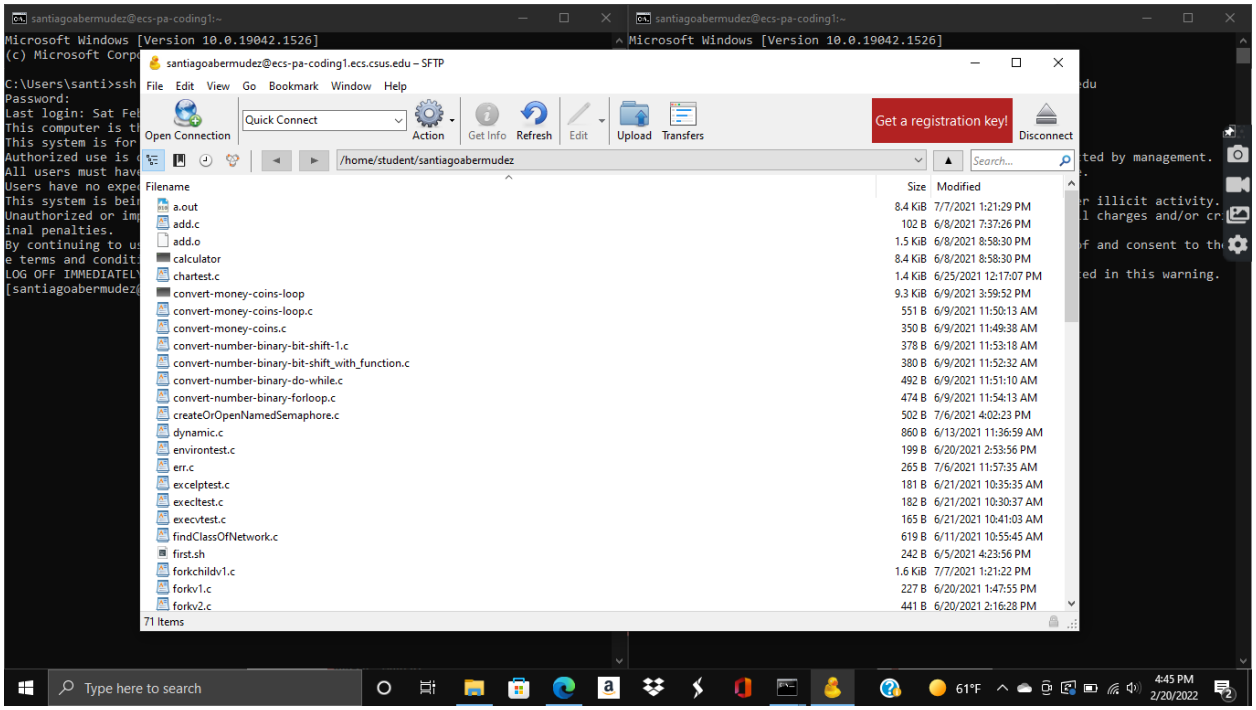


Once you have successfully signed in, you should see output as shown below. Make sure you sign in for both terminals.

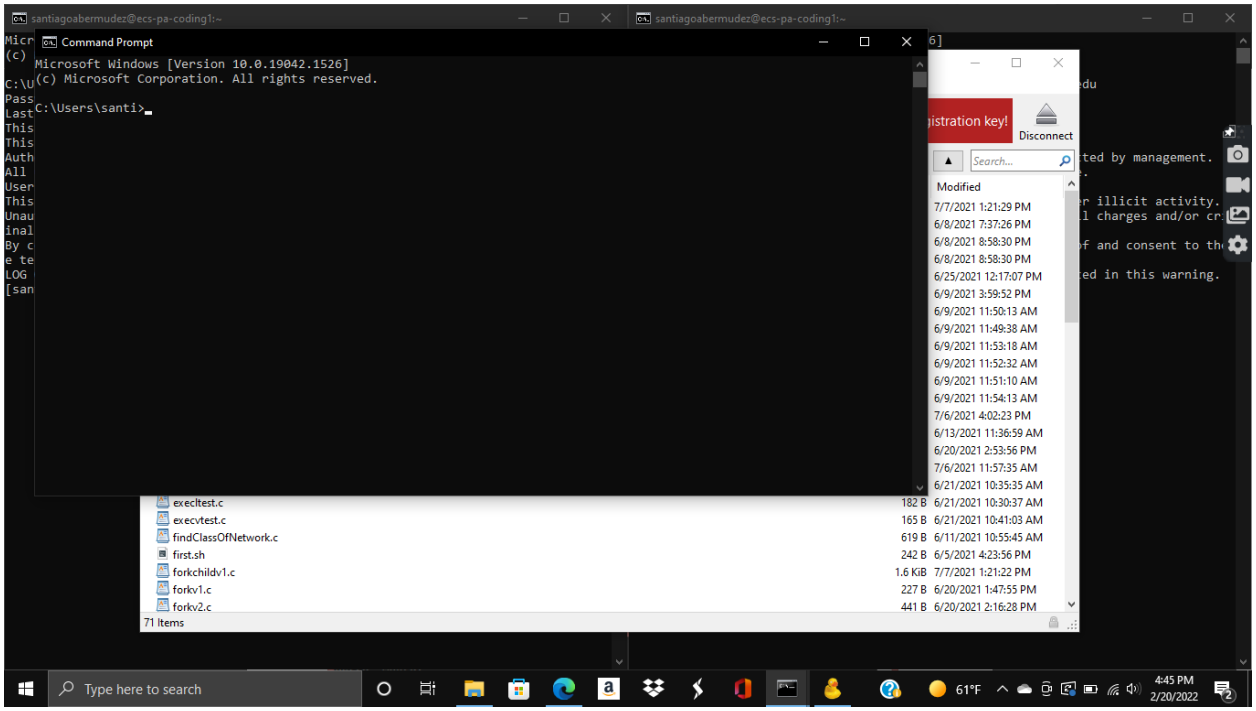


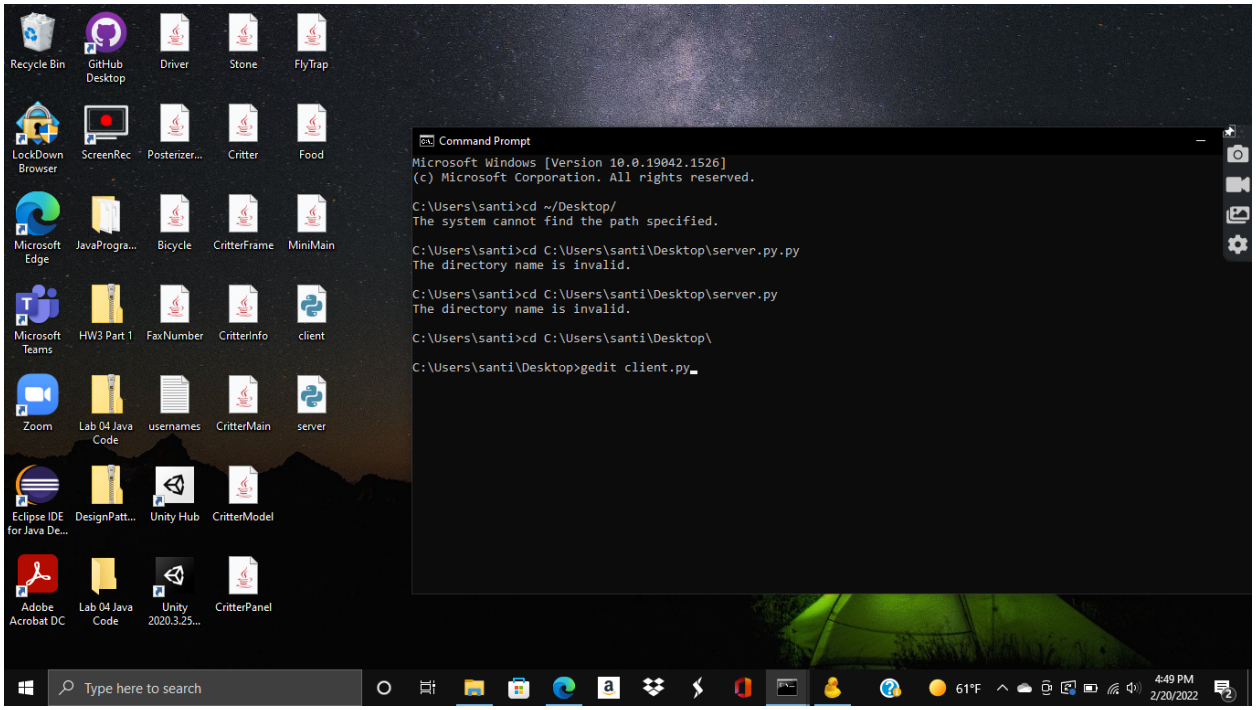
Then you will want to search for Cyberduck and open it. You may expect to see an empty filename list upon opening Cyberduck. In my case, I see a list filled with a bunch of C programs, but that is just because I have used the ECS server for my CSC 60 class.



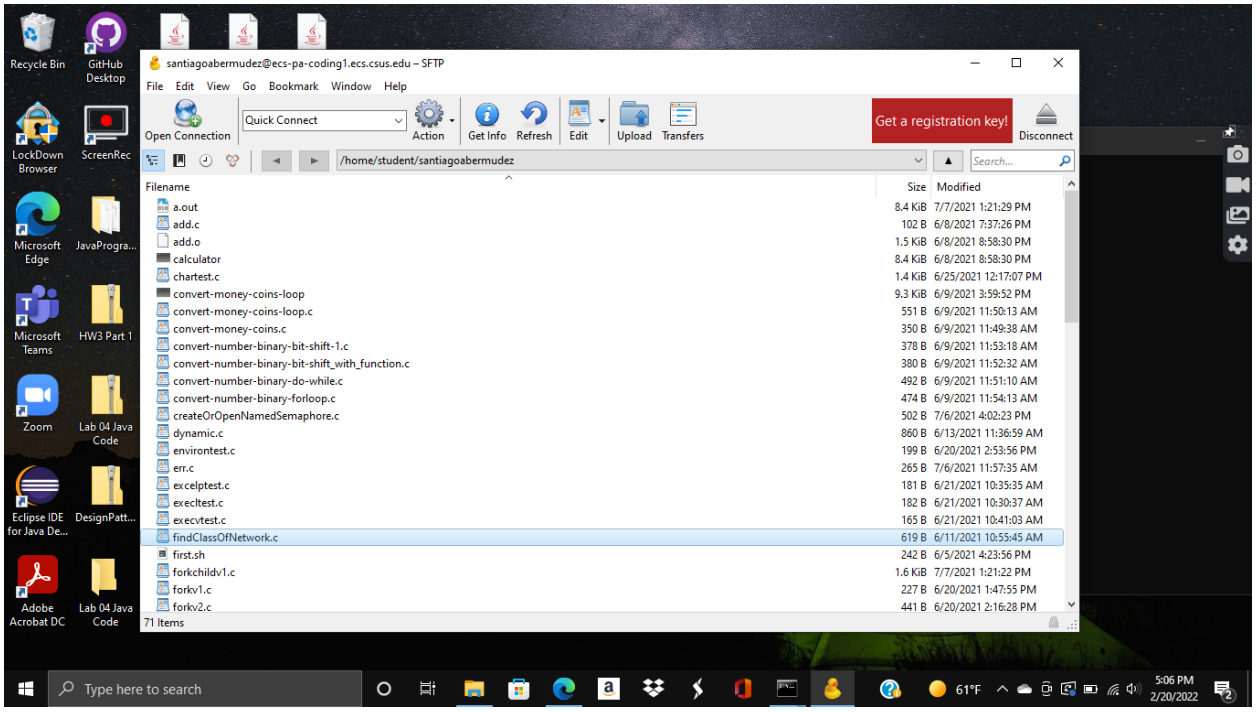


After, you may want to open a third terminal without signing in. In this terminal, you will want to change the directory to desktop. It took me a few tries to figure things out, but I eventually got it. This is really just if you want to perform ‘gedit’ on a python file and shouldn’t prevent you from doing the rest of the assignment.

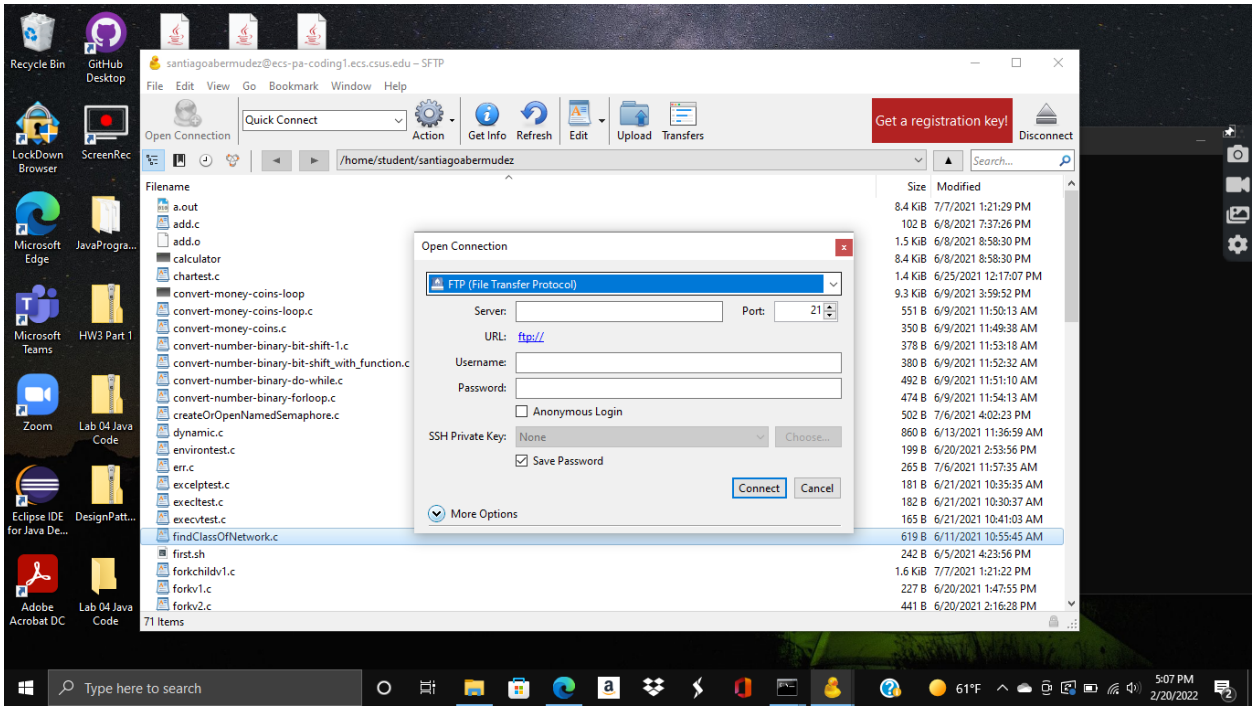


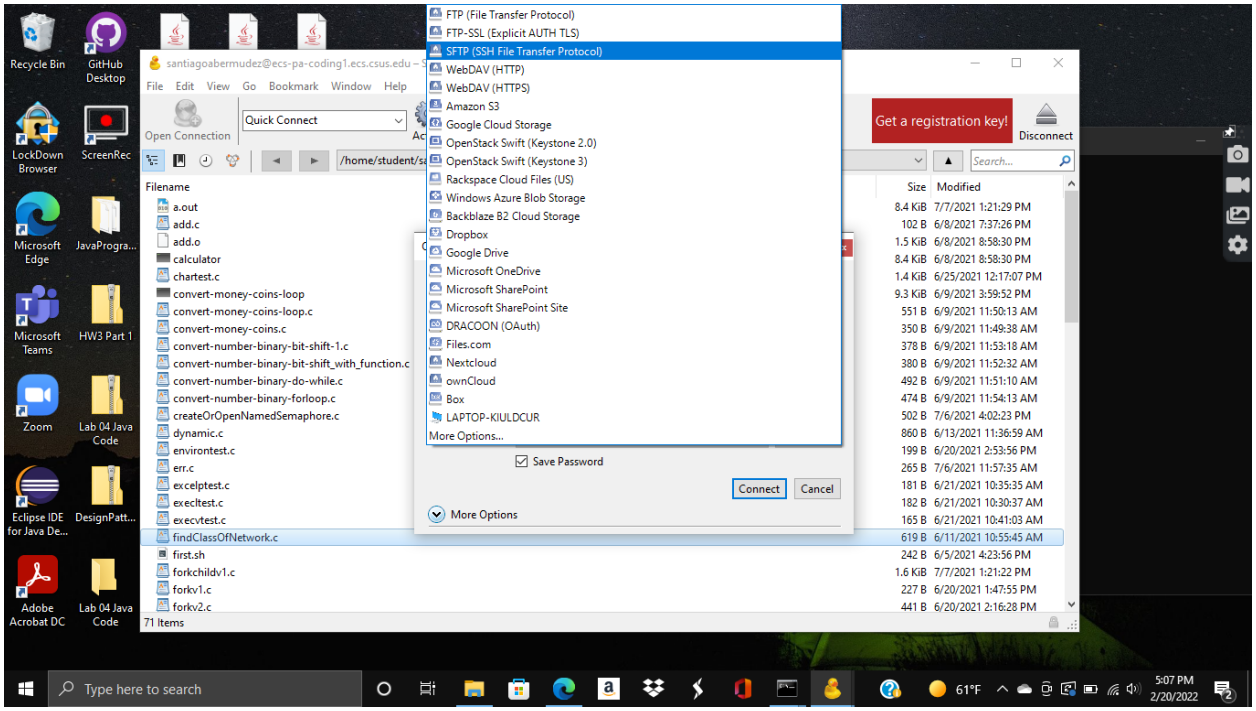


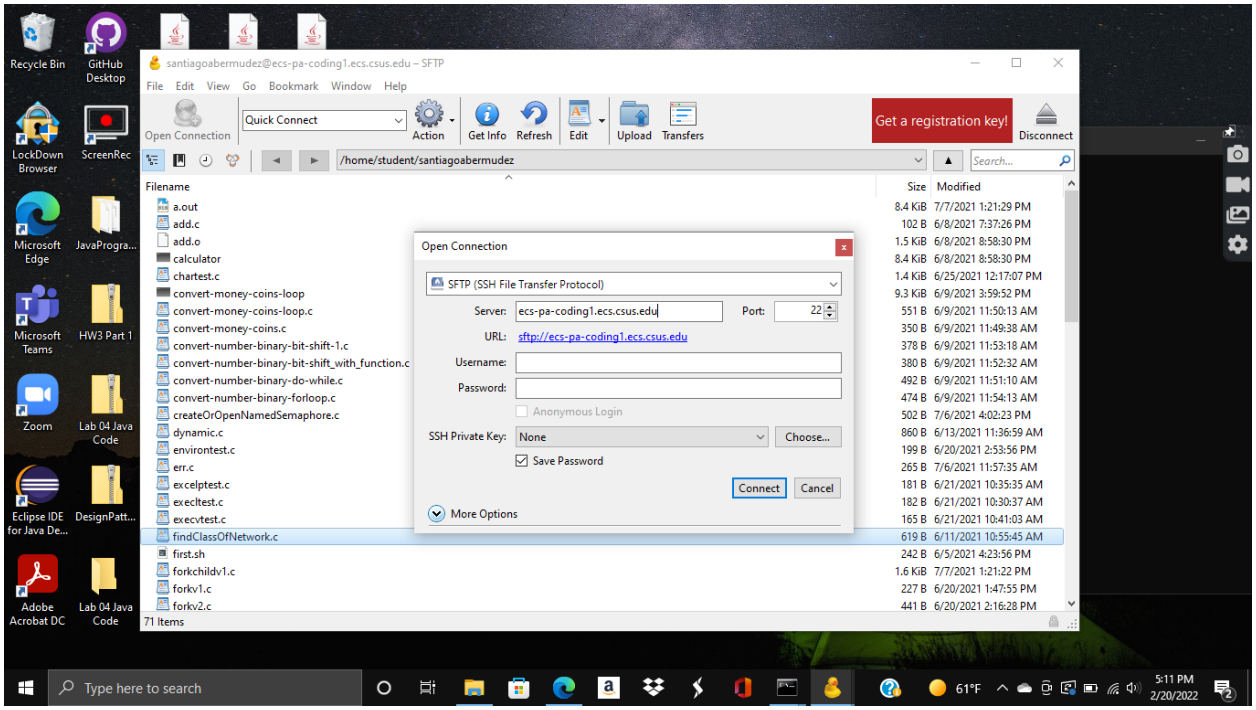
In Cyberduck, click on ‘Open Connection’.

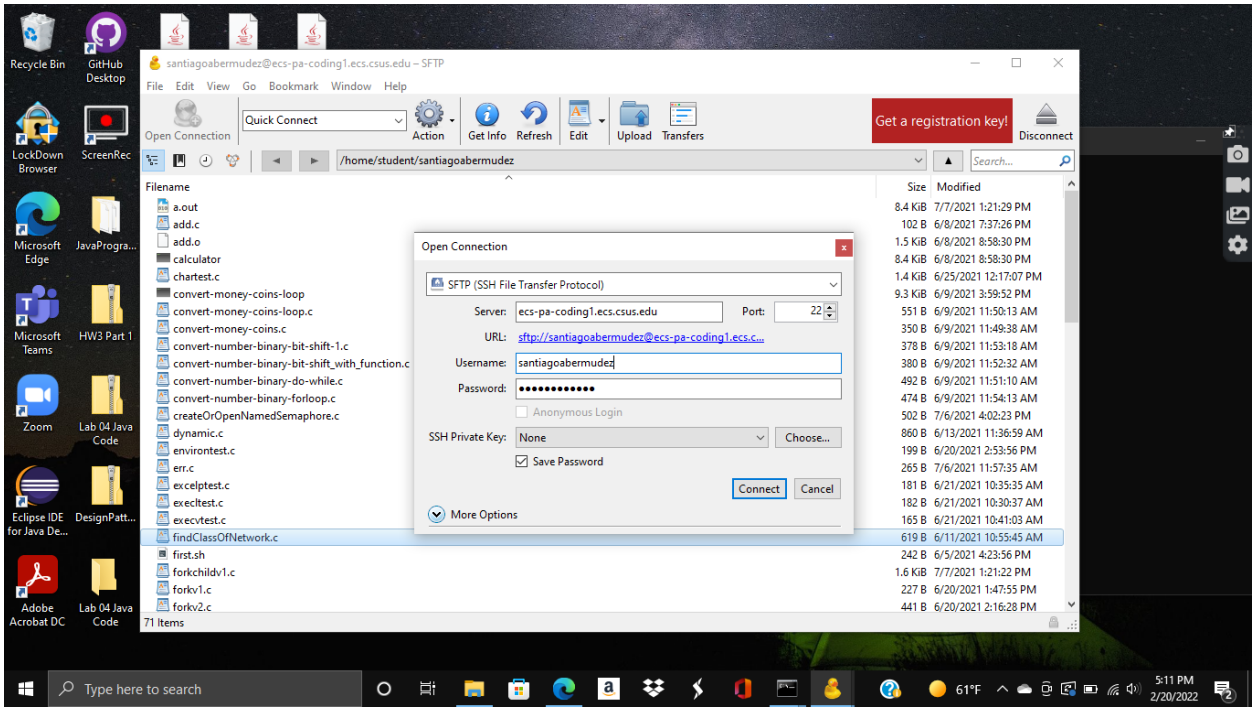


After opening a connection, you should see a window open up. In that window you will want to change FTP (File Transfer Protocol) to SFTP (SSH File Transfer Protocol), enter the ECS server name, your username and password, then click connect.

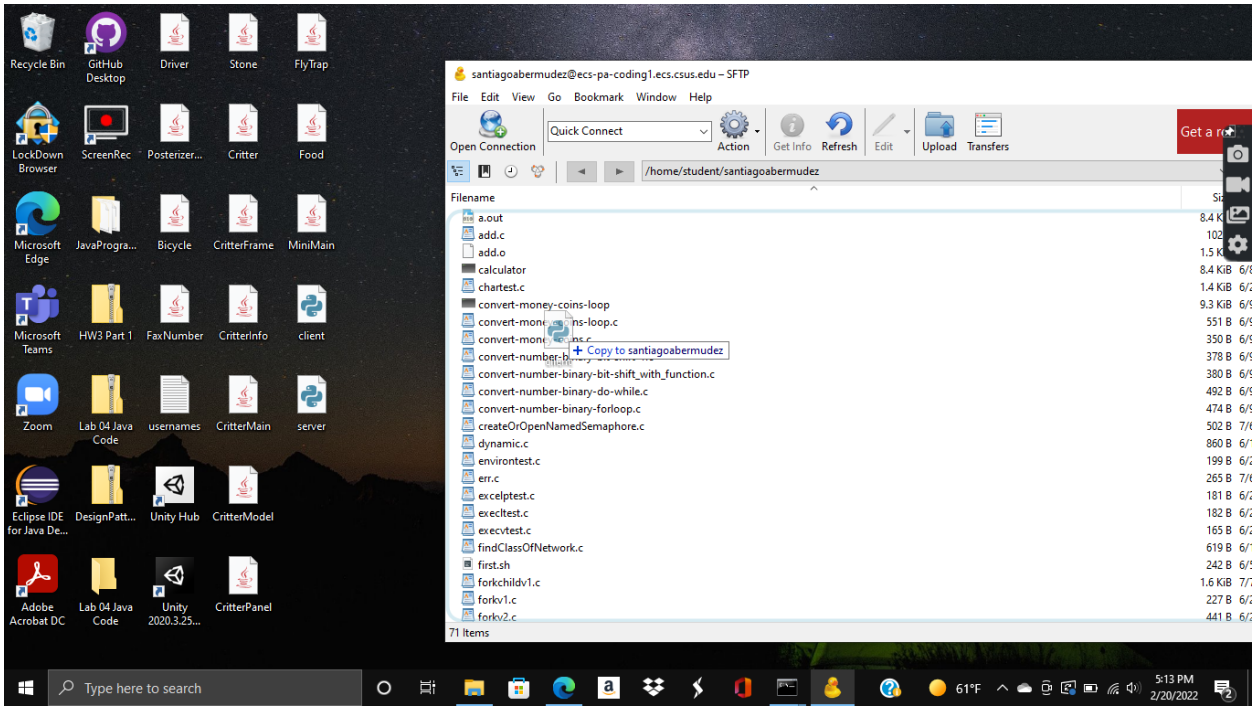

santiagoabermudez@ecs-pa-coding1.ecs.csus.edu – SFTP
File Edit View Go Bookmark Window Help
Open Connection
Quick Connect
Action
Get Info
Refresh
Edit
Upload
Transfers
Get a registration key!
Disconnect
/home/student/santiagoabermudez
Filename
Size
Modified
a.out 8.4 KiB 7/7/2021 1:21:29 PM
add.c 102 B 6/8/2021 7:37:26 PM
add.o 1.5 KiB 6/8/2021 8:58:30 PM
calculator 8.4 KiB 6/8/2021 8:58:30 PM
chartest.c 1.4 KiB 6/25/2021 12:17:07 PM
convert-money-coins-loop 9.3 KiB 6/9/2021 3:59:52 PM
convert-money-coins-loop.c 551 B 6/9/2021 11:50:13 AM
convert-money-coins.c 350 B 6/9/2021 11:49:38 AM
convert-number-binary-bit-shift-1.c 378 B 6/9/2021 11:53:18 AM
convert-number-binary-bit-shift_with_function.c 380 B 6/9/2021 11:52:32 AM
convert-number-binary-do-while.c 492 B 6/9/2021 11:51:10 AM
convert-number-binary-forloop.c 474 B 6/9/2021 11:54:13 AM
createOrOpenNamedSemaphore.c 502 B 7/6/2021 4:02:23 PM
dynamic.c 860 B 6/13/2021 11:36:59 AM
environtest.c 199 B 6/20/2021 2:53:56 PM
err.c 265 B 7/6/2021 11:57:35 AM
excelptest.c 181 B 6/21/2021 10:35:35 AM
execltest.c 182 B 6/21/2021 10:30:37 AM
execvtest.c 165 B 6/21/2021 10:41:03 AM
findClassOfNetwork.c 619 B 6/11/2021 10:55:45 AM
first.sh 242 B 6/5/2021 4:23:56 PM
forkchildv1.c 1.6 KiB 7/7/2021 1:21:22 PM
forkv1.c 227 B 6/20/2021 1:47:55 PM
forkv2.c 441 B 6/20/2021 2:16:28 PM
71 Items
Open Connection
FTP (File Transfer Protocol)
Server:
Port: 21
URL: ftp://
Username:
Password:
Anonymous Login
SSH Private Key: None
Choose...
Save Password
Connect
Cancel
More Options
Recycle Bin
GitHub Desktop
LockDown Browser
ScreenRec
Microsoft Edge
JavaProgra...
Microsoft Teams
HW3 Part 1
Zoom
Lab 04 Java Code
Eclipse IDE for Java De...
DesignPatt...
Adobe Acrobat DC
Lab 04 Java Code
Type here to search
61°F
5:07 PM
2/20/2022

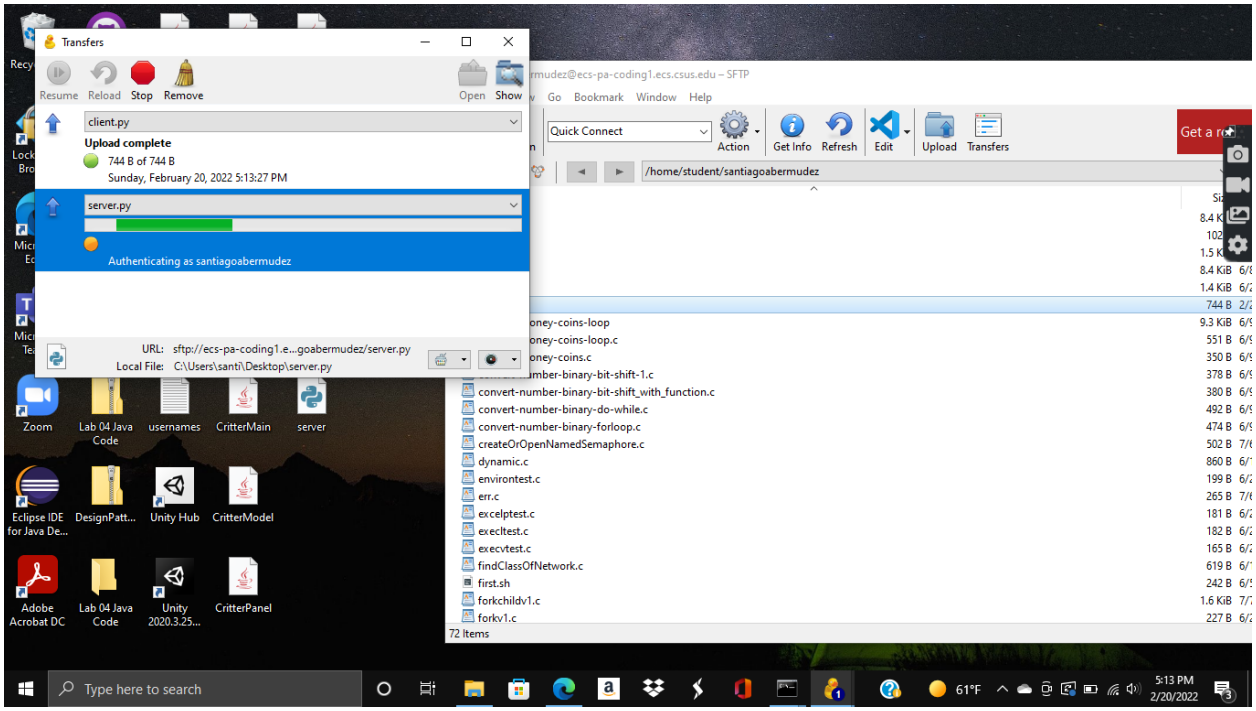

FTP (File Transfer Protocol)
FTP-SSL (Explicit AUTH TLS)
SFTP (SSH File Transfer Protocol)
WebDAV (HTTP)
WebDAV (HTTPS)
Amazon S3
Google Cloud Storage
OpenStack Swift (Keystone 2.0)
OpenStack Swift (Keystone 3)
Rackspace Cloud Files (US)
Windows Azure Blob Storage
Backblaze B2 Cloud Storage
Dropbox
Google Drive
Microsoft OneDrive
Microsoft SharePoint
Microsoft SharePoint Site
DRACOON (OAuth)
Files.com
Nextcloud
ownCloud
Box
LAPTOP-KIULDCUR
More Options...
Save Password
Connect
Cancel
More Options
Type here to search
61°F
5:07 PM
2/20/2022



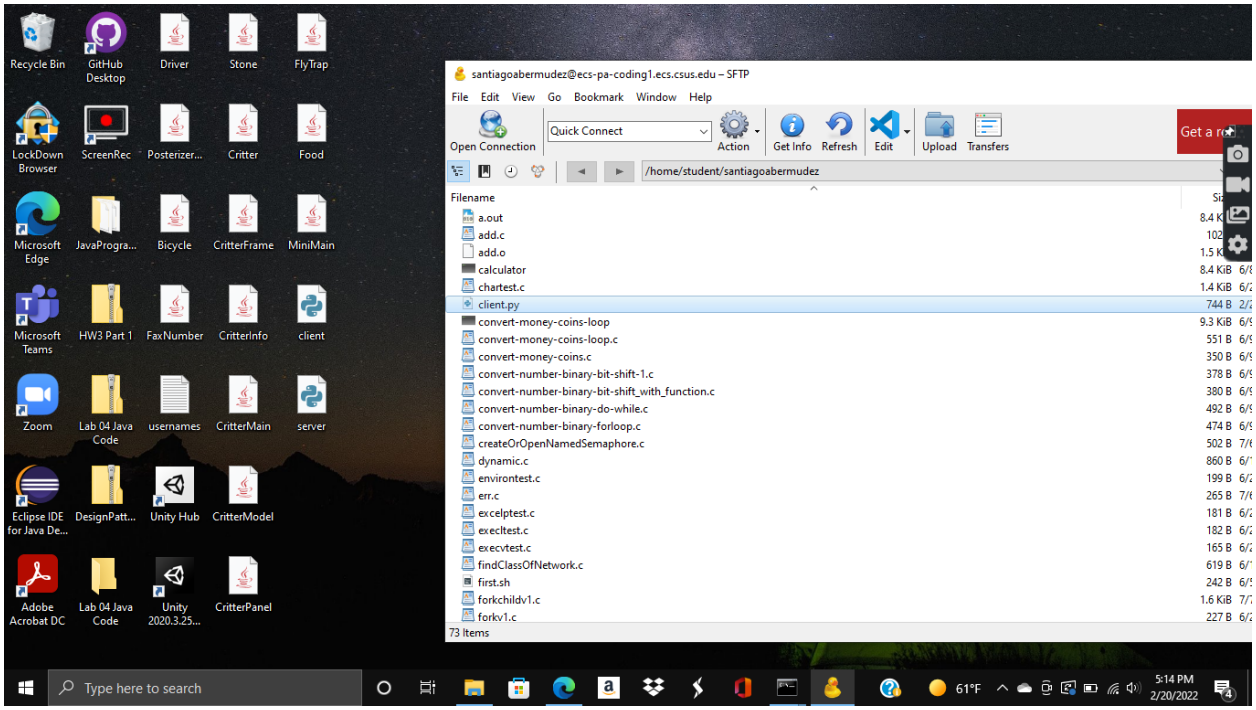


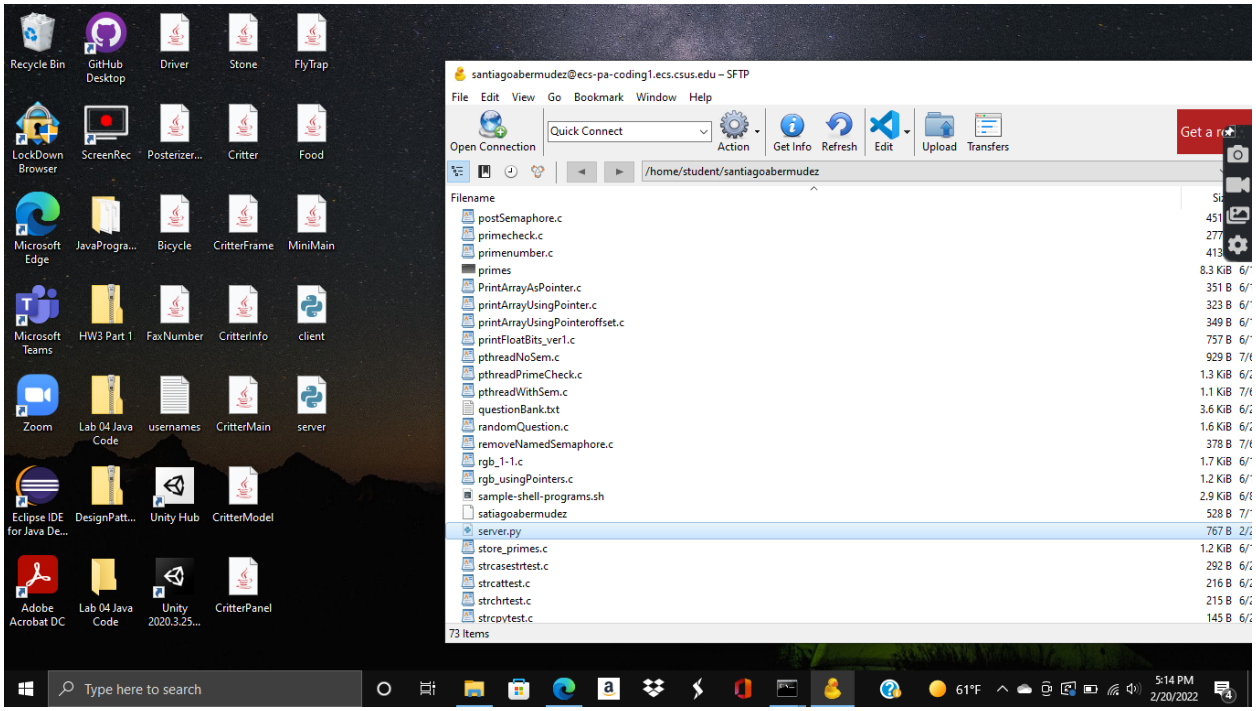
Once you are done, you should have two python programs (“client.py” and “server.py”) on your desktop. You simply click and drag them into the Cyberduck file list. A window should open showing the download progress.



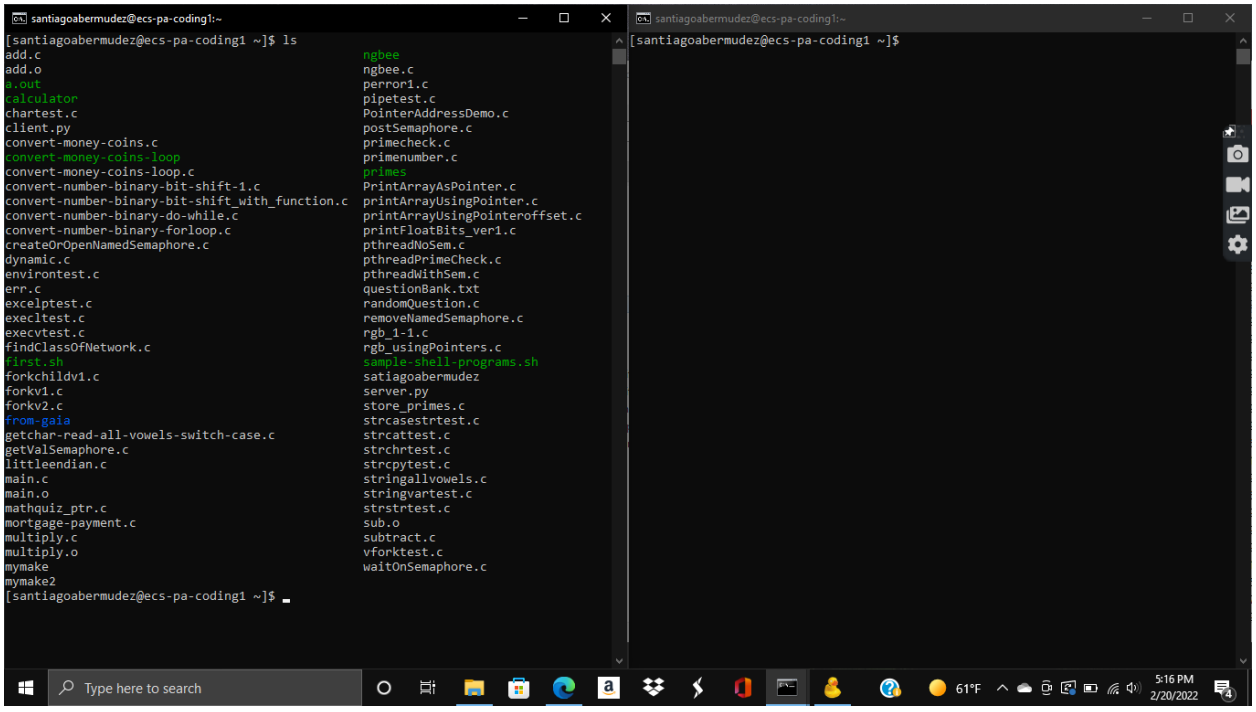


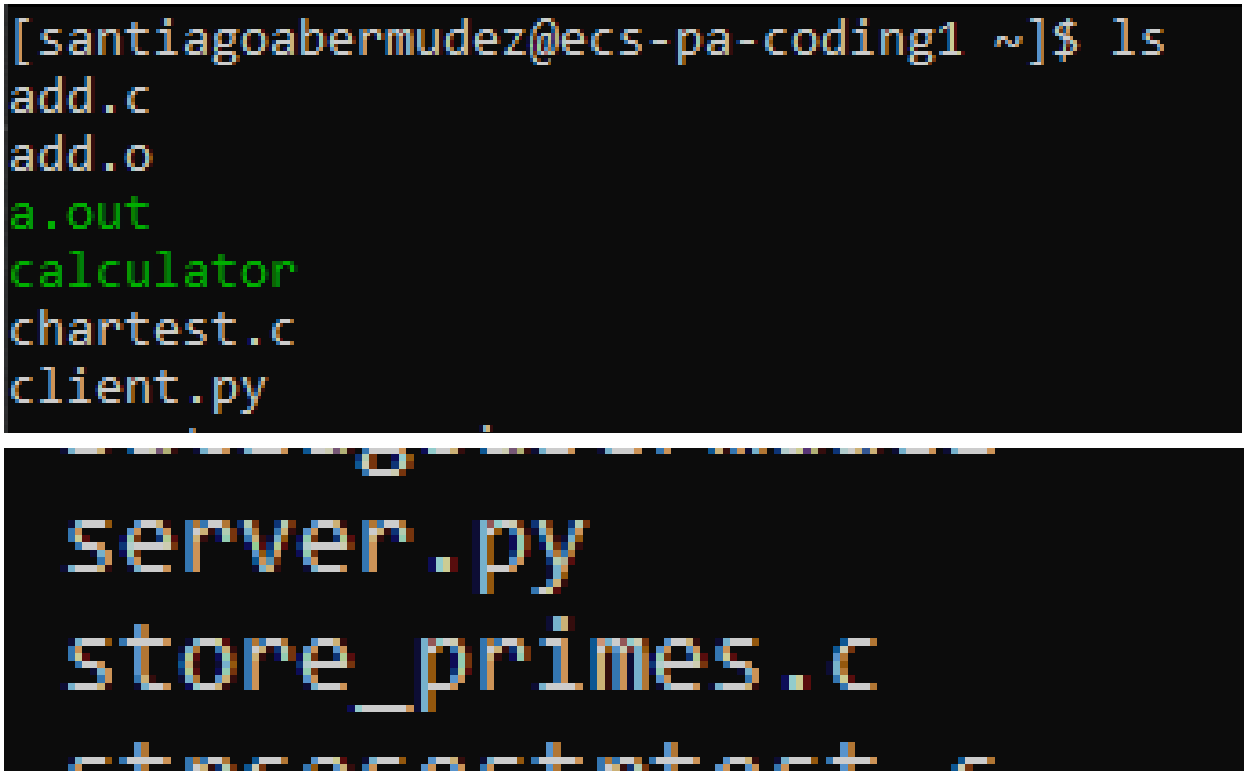
Once the download is complete, you should see both programs listed in Cyberduck. Again, my filename list is full of programs, but you should see both highlighted in the screenshots below.





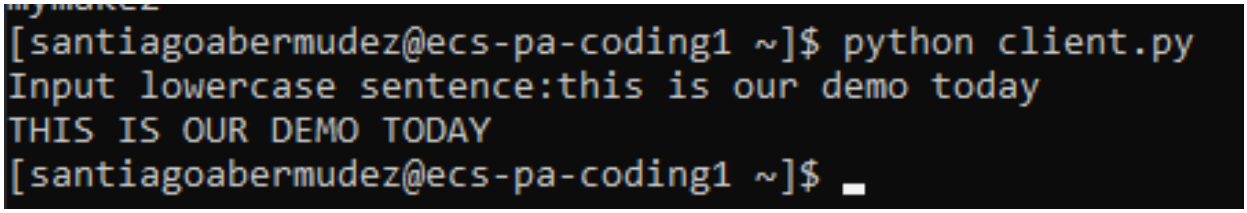
Then you will want to go back to the terminals and type the ‘ls’ command to list out the programs in the server for both terminals. You should be able to see ‘client.py’ and ‘server.py’ listed somewhere underneath.





On the right terminal is where you will execute the server program, so you will type ‘python server.py’ on the right and enter as shown below. You should see a response like “The server is ready to receive”.

On the left terminal is where you will want to execute the client program. After doing so, you will see a prompt to enter a lowercase sentence. Entering a lowercase sentence will simply return an uppercase version of that sentence as output. That should conclude the lab!



Source code below:

*Code for UDP:

```
#UDP Client
from socket import * #Python's socket library!
serverName = '127.0.0.1'
serverPort = 12061
clientSocket = socket(AF_INET, SOCK_DGRAM)  #Creates a UDP socket for the server.
#^Creates a datagram with server IP and port = x;
message = raw_input('Input lowercase sentence:') #Gets user input from the keyboard.
clientSocket.sendto(message,(serverName, serverPort))
#^Attaches the server name and port to the message and then sends it into the socket.
modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
#^Simply reads the reply characters from the socket into string.
print modifiedMessage #Prints out the received string.
clientSocket.close() #Closes the socket.

#UDP Server
from socket import *
serverPort = 12061
serverSocket = socket(AF_INET, SOCK_DGRAM) #Creates the UDP socket.
serverSocket.bind(('', serverPort)) #Binds the socket to a local port number (*12061 in my case).
print 'The server is ready to receive'
while 1: #This is simply an eternal loop.
        message, clientAddress = serverSocket.recvfrom(2048)
#^Reads from UDP socket into the message, getting the client's address (*client IP and port).
        modifiedMessage = message.upper()
        serverSocket.sendto(modifiedMessage, clientAddress)
#^Sends an uppercase string back to the client.
```

*Output for UDP code below!

*Code for TCP:

```
#TCP Client
from socket import *
serverName = '127.0.0.1'
serverPort = 12061
clientSocket = socket(AF_INET, SOCK_STREAM) #Creates a TCP socket for the server.
#^Remote port 12061.
clientSocket.connect((serverName,serverPort))
sentence = raw_input('Input lowercase sentence:')
clientSocket.send(sentence) #No need to attach server name, port in this case.
modifiedSentence = clientSocket.recv(1024)
print 'From Server:', modifiedSentence
clientSocket.close()

#TCP Server
from socket import *
serverPort = 12061
serverSocket = socket(AF_INET,SOCK_STREAM) #Creates a TCP welcoming socket.
serverSocket.bind(('',serverPort))
serverSocket.listen(1) #Server begins listening for incoming TCP requests.
print 'The server is ready to receive'
while 1:  #This is simply an eternal loop.
        connectionSocket, addr = serverSocket.accept()
        #^The server waits on accept() for incoming requests. A new socket created on return.
        sentence = connectionSocket.recv(1024)
        capitalizedSentence = sentence.upper()
        #^Reads bytes from socket (but not the client's address as in UDP).
        connectionSocket.send(capitalizedSentence)
connectionSocket.close() #Close connection to this client.
```

*Output for TCP code below!