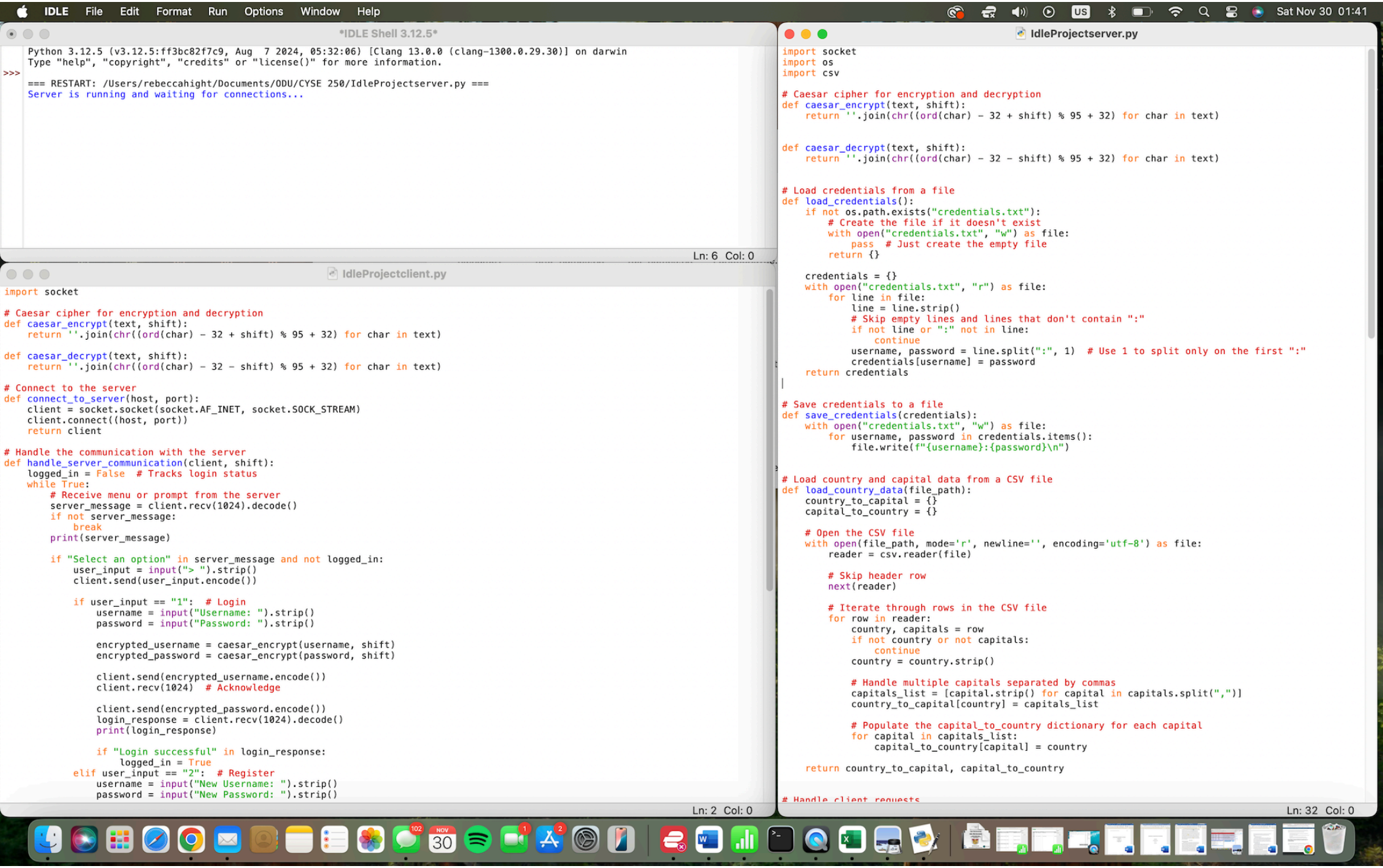
IDLE File Edit Format Run Options Window Help
Sat Nov 30 01:41
IDLE Shell 3.12.5
Python 3.12.5 (v3.12.5:ff3bc82f7c9, Aug 7 2024, 05:32:06) [Clang 13.0.0 (clang-1300.0.29.30)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=== RESTART: /Users/rebeccahight/Documents/ODU/CYSE 250/IdleProjectserver.py ===
Server is running and waiting for connections...
Ln: 6 Col: 0
IdleProjectclient.py
import socket
# Caesar cipher for encryption and decryption
def caesar_encrypt(text, shift):
return ''.join(chr((ord(char) - 32 + shift) % 95 + 32) for char in text)
def caesar_decrypt(text, shift):
return ''.join(chr((ord(char) - 32 - shift) % 95 + 32) for char in text)
# Connect to the server
def connect_to_server(host, port):
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect((host, port))
return client
# Handle the communication with the server
def handle_server_communication(client, shift):
logged_in = False # Tracks login status
while True:
# Receive menu or prompt from the server
server_message = client.recv(1024).decode()
if not server_message:
break
print(server_message)
if "Select an option" in server_message and not logged_in:
user_input = input("> ").strip()
client.send(user_input.encode())
if user_input == "1": # Login
username = input("Username: ").strip()
password = input("Password: ").strip()
encrypted_username = caesar_encrypt(username, shift)
encrypted_password = caesar_encrypt(password, shift)
client.send(encrypted_username.encode())
client.recv(1024) # Acknowledge
client.send(encrypted_password.encode())
login_response = client.recv(1024).decode()
print(login_response)
if "Login successful" in login_response:
logged_in = True
elif user_input == "2": # Register
username = input("New Username: ").strip()
password = input("New Password: ").strip()
Ln: 2 Col: 0
IdleProjectserver.py
import socket
import os
import csv
# Caesar cipher for encryption and decryption
def caesar_encrypt(text, shift):
return ''.join(chr((ord(char) - 32 + shift) % 95 + 32) for char in text)
def caesar_decrypt(text, shift):
return ''.join(chr((ord(char) - 32 - shift) % 95 + 32) for char in text)
# Load credentials from a file
def load_credentials():
if not os.path.exists("credentials.txt"):
# Create the file if it doesn't exist
with open("credentials.txt", "w") as file:
pass # Just create the empty file
return {}
credentials = {}
with open("credentials.txt", "r") as file:
for line in file:
line = line.strip()
# Skip empty lines and lines that don't contain ":"
if not line or ":" not in line:
continue
username, password = line.split(":", 1) # Use 1 to split only on the first ":"
credentials[username] = password
return credentials
# Save credentials to a file
def save_credentials(credentials):
with open("credentials.txt", "w") as file:
for username, password in credentials.items():
file.write(f"{username}:{password}\n")
# Load country and capital data from a CSV file
def load_country_data(file_path):
country_to_capital = {}
capital_to_country = {}
# Open the CSV file
with open(file_path, mode='r', newline='', encoding='utf-8') as file:
reader = csv.reader(file)
# Skip header row
next(reader)
# Iterate through rows in the CSV file
for row in reader:
country, capitals = row
if not country or not capitals:
continue
country = country.strip()
# Handle multiple capitals separated by commas
capitals_list = [capital.strip() for capital in capitals.split(",")]
country_to_capital[country] = capitals_list
# Populate the capital_to_country dictionary for each capital
for capital in capitals_list:
capital_to_country[capital] = country
return country_to_capital, capital_to_country
Handle client requests
Ln: 32 Col: 0