Syrian Arab Republic

Attakai - Screen University

Department of Communication and electrical engineering

5th , Network Programming : Homework No2

الجمهورية العربية السورية

اللاذقية -جامعـــة تشريـــن

كلية الهندسة الكهربائية والميكانيكية

قسم هندسة الاتصالات والالكترونيات

السنة الخامسة: وظيفة2 برمجة شبكات

وظيفة برمجة الشبكات الثانية

إعداد الطالب:

علي أسعد

2365

إعادة عملي

إشراف الدكتور:

مهند عيسى

Question 1: TCP Server/Client Quiz App with Multi-threading?

As an improvement to previous first homework, build a TCP server and client quiz application using Python. The server should handle multiple client connections simultaneously using multi-threading. The application should allow clients to connect, participate in a quiz, and receive their quiz scores upon completion.

Requirements:

A. The server should be able to handle multiple client connections concurrently.
B. The quiz should consist of a set of pre-defined questions stored on the server.
C. Each client should connect to the server and receive the quiz questions.
D. Clients should send their answers to the server.
E. The server should keep track of the scores for each client.
F. At the end of the quiz, the server should send the final scores to each client.

Guidelines:

- Use Python's socket module “don’t use 3thd-party packages”.
- Implement multi-threading to handle multiple client connections concurrently.
- Store the quiz questions and correct answers on the server side.

هذا الكود هو ببساطة تطبيق لخادم أسئلة وأجوبة. يقوم الخادم بقراءة ملف يسمى "questions.txt" ويحتوي على الأسئلة والإجابات المتوقعة لهذه الأسئلة، ثم يبدأ في الاستماع على المنفذ 7474 باستخدام مكتبة socket ويقبل الاتصالات الواردة من العملاء.

عندما يتم توصيل عميل، يتم إنشاء موضوع منفصل لمعالجة الاتصال باستخدام threading. في وقت التشغيل، يتم إرسال الأسئلة إلى العميل وتلقي الإجابات المتوقعة، ثم يتم التحقق من صحة هذه الإجابات وحساب النتيجة. يتم إرسال الدرجة النهائية إلى العميل ويتم إغلاق الاتصال.

– يتم استيراد مكتبة socket وthreading. مكتبة socket تسمح للخادم بالتواصل مع العملاء باستخدام بروتوكول TCP/IP، بينما تسمح مكتبة threading بإنشاء موضوعات جديدة لمعالجة الاتصالات.

– يتم قراءة الملف "questions.txt" باستخدام دالة open ويتم تعيين مصفوفة questions لقائمة الأسئلة والإجابات الموجودة في الملف.

– يتم تعيين متغيرات host وport للخادم.

– تعرف دالة server () التي تنشئ خادمًا باستخدام socket، يربط الخادم على المنفذ المحدد ويستمع على الاتصالات الواردة. عندما يتم توصيل عميل، يتم إنشاء موضوع منفصل لمعالجة الاتصال باستخدام handle quiz ().

– دالة handle quiz () تتلقى اتصال العميل وترسل الأسئلة إلى العميل وتتوقع الإجابات منه. يتم التحقق من صحة الإجابات وحساب النتيجة، ثم يتم إرسال الدرجة النهائية إلى العميل.

- تتحقق الشرطية if __name__ == '__main__': من أن البرنامج يعمل كملف مستقل وتشغيل الخادم.

```
import socket
import threading

questions = []

try:
    # Read the questions file
    with open('questions.txt', 'r') as file:
        for line in file:
            question, answer = line.strip().split(':')
            questions.append((question.strip(), answer.strip()))
except FileNotFoundError:
    print('Error: Questions file not found')
    exit()

# Define a function to start the server
host = '0.0.0.0'
port = 7474
def server():
    try:
        server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        server_socket.bind((host, port))
        server_socket.listen()
        print(f'Server listening on {host}:{port}')

        while True:
            client_socket, client_address = server_socket.accept()
            c_thread = threading.Thread(target=handle_quiz,
args=(client_socket, client_address))
            c_thread.start()
    except socket.error:
        print('Error: Could not start the server')
    finally:
        server_socket.close()


def handle_quiz(client_socket, client_address):
    result = 0
    try:
        # Send the quiz questions to the client
        for question, answer in questions:
```

```
            client_socket.send(question.encode())
            client_answer = client_socket.recv(1024).decode().strip()

            if client_answer == answer:
                result += 1
        client_socket.send(f'Your score is {result} out of
{len(questions)}.'.encode())
    except socket.error:
        print(f'Error: Could not handle quiz for client
{client_address[0]}:{client_address[1]}')
    finally:
        client_socket.close()

if __name__ == '__main__':
    server()
```

Server listening on 0.0.0.0:7474

```
import socket

try:
    client_socket = socket.socket()
    client_socket.connect(('127.0.0.1', 7474))

    while True:
        try:
            question = client_socket.recv(1024).decode()
            if not question:
                break
            # Send the answer to the server
            answer = input(question + ' ')
            client_socket.send(answer.encode())
        except socket.error:
            print('Error')
            break

    # Receive the final score from the server
    try:
        score = client_socket.recv(1024).decode()
        print(score)
    except socket.error:
        print('Error: Could not receive the final score from the server')
except ConnectionRefusedError:
    print('Error: Could not connect to the server. Please check that it is
running and try again.')
```

```
finally:
    # Close the client socket
    client_socket.close()
```
your city? lattakia

what is your name? ali

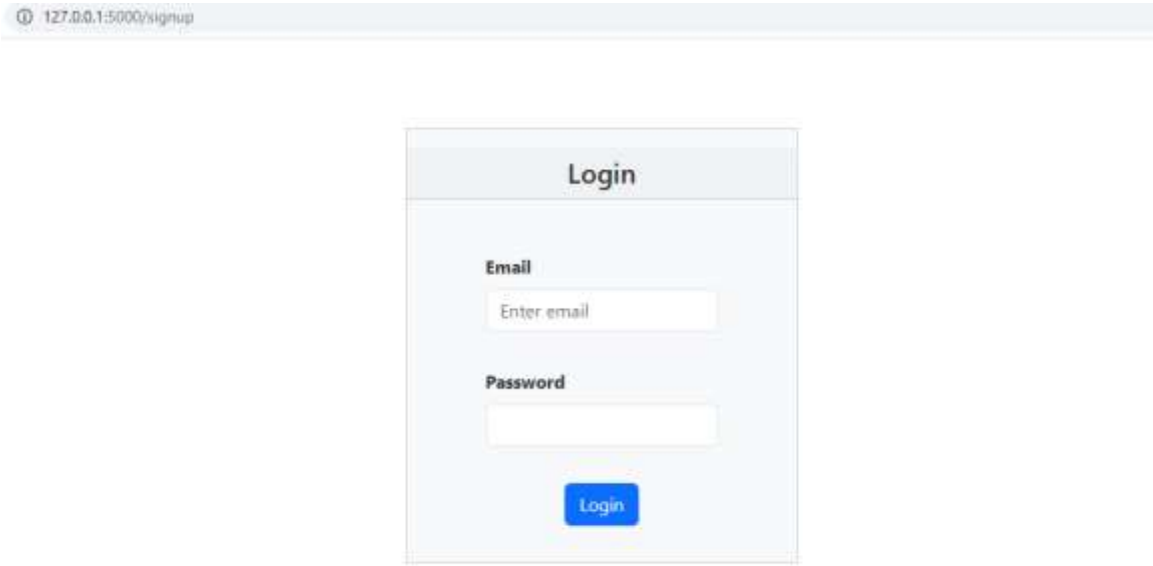
12+32? 44

152-150? 2

170/170? 1

12+85? 97

Your score is 6 out of 6.




127.0.0.1:5000/about
about me
Ali