حل الوظيفة الثانية من برمجة الشبكات

الطالبة : ألاء جمال طويل ٢٩٧٢

الطالب : محمد بشار مرتكوش ٢٩٨٢

حل السؤال الأول :

كود السيرفر

```python
import socket
import threading

bank_accounts = {
    "123456": {"pin": "1234", "balance": 7700},
    "654321": {"pin": "4321", "balance": 1783}
}


def handle_client_connection(client_socket):
    try:
        client_socket.send("Enter your account number: ".encode())
        account_number = client_socket.recv(1024).decode().strip()

        if account_number in bank_accounts:
            client_socket.send("Enter your PIN: ".encode())
            pin = client_socket.recv(1024).decode().strip()

            if pin == bank_accounts[account_number]["pin"]:
                client_socket.send("Authentication successful!\n".encode())
                while True:
                    client_socket.send("Choose an option: 1. Check Balance 2. Deposit 3. Withdraw 4. Exit\n".encode())
                    option = client_socket.recv(1024).decode().strip()

                    if option == '1':
                        balance = bank_accounts[account_number]["balance"]
                        client_socket.send(f"Your balance is: ${balance}\n".encode())
                    elif option == '2':
                        client_socket.send("Enter amount to deposit: ".encode())
                        try:
                            amount = int(client_socket.recv(1024).decode().strip())
                            bank_accounts[account_number]["balance"] += amount
                            client_socket.send("Deposit successful!\n".encode())
                        except ValueError:
                            client_socket.send("Invalid amount entered. Try again.\n".encode())
                    elif option == '3':
                        client_socket.send("Enter amount to withdraw: ".encode())
                        try:
                            amount = int(client_socket.recv(1024).decode().strip())
                            if amount <= bank_accounts[account_number]["balance"]:
                                bank_accounts[account_number]["balance"] -= amount
                                client_socket.send("Withdrawal successful!\n".encode())
                            else:
                                client_socket.send("Insufficient funds!\n".encode())
                        except ValueError:
                            client_socket.send("Invalid amount entered. Try again.\n".encode())
                    elif option == '4':
                        final_balance = bank_accounts[account_number]["balance"]
                        client_socket.send(f"Your final balance is: ${final_balance}\n".encode())
                        break
                    else:
                        client_socket.send("Invalid option. Please choose again.\n".encode())
            else:
                client_socket.send("Authentication failed. Incorrect PIN.\n".encode())
        else:
            client_socket.send("Authentication failed. Account not found.\n".encode())
    finally:
        client_socket.close()


def main():
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind(("0.0.0.0", 9999))
    server.listen(5)
    print("Server is running on port 9999...")

    while True:
        client_socket, addr = server.accept()
        print(f"New connection from {addr}")
        client_handler = threading.Thread(target=handle_client_connection, args=(client_socket,))
        client_handler.start()


if __name__ == "__main__":
    main()

```

شرح كود السيرفر :

استيراد المكتبات اللازمة

socket: تستخدم لإنشاء اتصالات الشبكة.

threading: تستخدم حتى يمكن للخادم معالجة عدة اتصالات للعملاء في نفس الوقت.

تم تعريف قاموس يحوي على حسابات بنكية

دالة لمعالجة اتصال العميل

client_socket.send: يرسل رسالة للعميل لطلب إدخال رقم الحساب.

client_socket.recv: يستقبل رقم الحساب المدخل من قبل العميل.

strip(): يزيل المسافات البيضاء الزائدة من بداية ونهاية النص.

التحقق من صحة رقم الحساب ورقم التعريف الشخصي (PIN)

if account_number in bank_accounts: التحقق مما إذا كان رقم الحساب موجودًا في القاموس.

if pin == bank_accounts[account_number]["pin"]: التحقق مما إذا كان رقم التعريف الشخصي المدخل صحيحًا.

تنفيذ عمليات الحساب المختلفة

while True: حلقة لعرض الخيارات وتنفيذ العمليات المتاحة (التحقق من الرصيد، الإيداع، السحب، الخروج).

option == '1': التحقق من الرصيد.

option == '2': الإيداع (مع التحقق من صحة المبلغ المدخل).

option == '3': السحب (مع التحقق من صحة المبلغ المدخل والتأكد من كفاية الرصيد).

option == '4': الخروج من النظام.

ValueError: معالجة الخطأ عند إدخال مبلغ غير صحيح.

إغلاق الاتصال بالعميل في النهاية

finally: تأكيد إغلاق الاتصال بالعميل بعد انتهاء المعالجة.

إعداد الخادم لقبول الاتصالات

socket.socket(socket.AF_INET, socket.SOCK_STREAM): إنشاء مقبس TCP/IP.

server.bind(("٠,٠,٠,٠", ٩٩٩٩)): ربط المقبس على جميع الواجهات المحلية على المنفذ ٩٩٩٩.

server.listen(٥): الاستماع للاتصالات الواردة، مع تحديد الحد الأقصى لطابور الاتصالات المعلقة بـ ٥.

server.accept(): قبول الاتصالات الواردة وإنشاء مقبس جديد لكل اتصال.

threading.Thread: إنشاء خيط جديد لمعالجة كل اتصال على حدة.

تشغيل الخادم

if __name__ == "__main__":: تأكد من أن الكود يتم تشغيله كبرنامج رئيسي وليس كمكتبة مستوردة.

main(): استدعاء دالة main لبدء تشغيل الخادم.

كود العميل :

```python
import socket


def main():
    try:
        client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        client.connect(("127.0.0.1", 9999))

        while True:
            response = client.recv(4096).decode()
            print(response)

            if "Enter" in response or "Choose" in response or "Try" in response:
                client_input = input()
                client.send(client_input.encode())

            if "Your final balance is" in response:
                break

    except socket.error as e:
        print(f"An error occurred: {e}")
    finally:
        client.close()
        print("Connection closed.")


if __name__ == "__main__":
    main()

```

شرح الكود :

يستورد مكتبة socket التي تتيح لك إنشاء اتصالات الشبكة بين العميل والخادم.

دالة main

تنشئ دالة main وفي داخلها:

يحاول الكود إنشاء كائن socket باستخدام socket.AF_INET (لإشارة إلى بروتوكول الإنترنت) وsocket.SOCK_STREAM (لإشارة إلى استخدام بروتوكول TCP).

يتصل الخادم (server) بالعنوان IP 127.0.0.1 والمنفذ ٩٩٩٩. العنوان ١٢٧,٠,٠,١ هو عنوان الـ loopback الذي يستخدم لاتصال الجهاز بنفسه.

يبدأ الكود حلقة غير منتهية while True لاستقبال الرسائل من الخادم.

يستخدم client.recv(4096).decode() لاستقبال بيانات من الخادم. الدالة recv تأخذ عدد البايتات التي سيتم استقبالها كمعلمة، وهنا تم تعيينها إلى ٤٠٩٦ بايت. البيانات المستقبلة يتم تحويلها من بايتات إلى نص باستخدام decode().

يتم طباعة الرسالة المستقبلة من الخادم باستخدام print(response).

إذا كانت الرسالة المستقبلة تحتوي على كلمات مثل ""Enter، Choose""، أو "Try"، فهذا يعني أن الخادم يطلب مدخلاً من المستخدم.

يتم استقبال المدخل من المستخدم باستخدام input().

يتم إرسال المدخل إلى الخادم باستخدام client.send(client_input.encode()). البيانات يتم تحويلها إلى بايتات باستخدام encode().

إذا كانت الرسالة المستقبلة تحتوي على النص "Your final balance is"، فهذا يعني أن العملية قد انتهت (إشارة من الخادم) ويجب إنهاء الحلقة.

إذا حدث خطأ في الاتصال، يتم القبض على الاستثناء socket.error وطباعة رسالة الخطأ.

أخيرًا، سواء حدث خطأ أم لا، يتم إغلاق الاتصال بالخادم باستخدام client.close() ويتم طباعة "Connection closed".

الجزء الرئيسي

يضمن هذا الجزء أن الدالة main يتم استدعاؤها فقط عندما يتم تشغيل هذا الملف كبرنامج رئيسي وليس عندما يتم استيراده كجزء من برنامج آخر.

الشرح المفصل للخطوات

إنشاء عميل (Client)

يتم إنشاء كائن socket لتمثيل اتصال العميل.

يتم استخدام connect لربط العميل بالخادم عند العنوان ١٢٧,٠,٠,١ والمنفذ ٩٩٩٩.

استقبال الرسائل

يدخل العميل في حلقة لاستقبال الرسائل من الخادم باستخدام recv.

يتم تحويل الرسائل المستقبلة إلى نص باستخدام decode ويتم طباعتها.

إرسال المدخلات

يتم التحقق من الرسالة المستقبلة لمعرفة ما إذا كانت الخادم يطلب مدخلاً من المستخدم.

إذا كان الأمر كذلك، يتم استقبال المدخل من المستخدم باستخدام input وإرساله إلى الخادم باستخدام send.

إنهاء الاتصال

إذا كانت الرسالة المستقبلة تشير إلى أن العملية انتهت (مثلاً: تحتوي على " Your final balance is")، يتم إنهاء الحلقة.

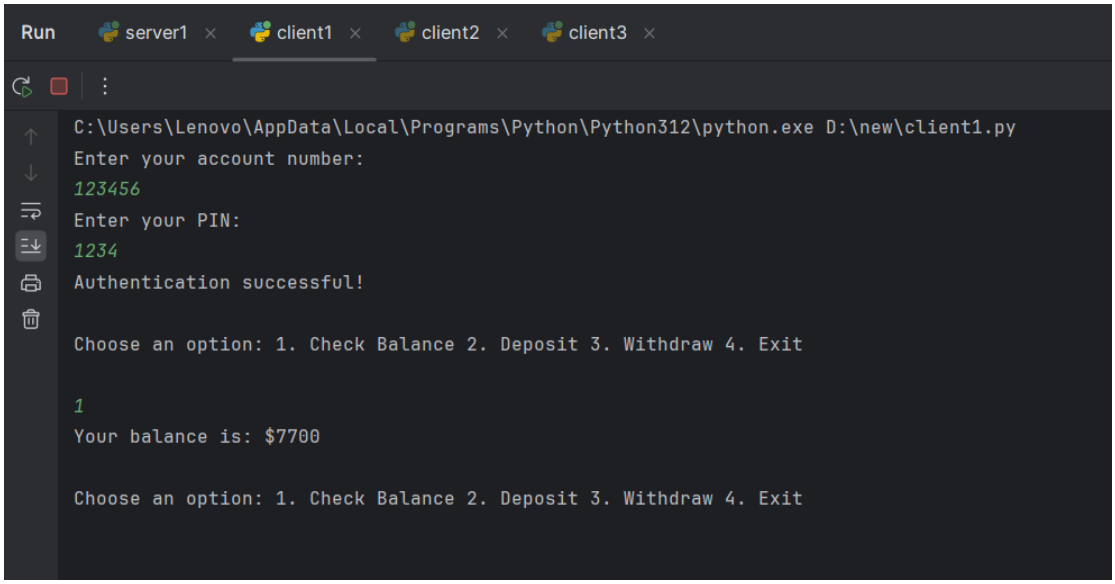
في حالة حدوث خطأ في الاتصال، يتم طباعة رسالة الخطأ وإغلاق الاتصال.

يتم التأكد من إغلاق الاتصال دائمًا عن طريق كتلة finally.

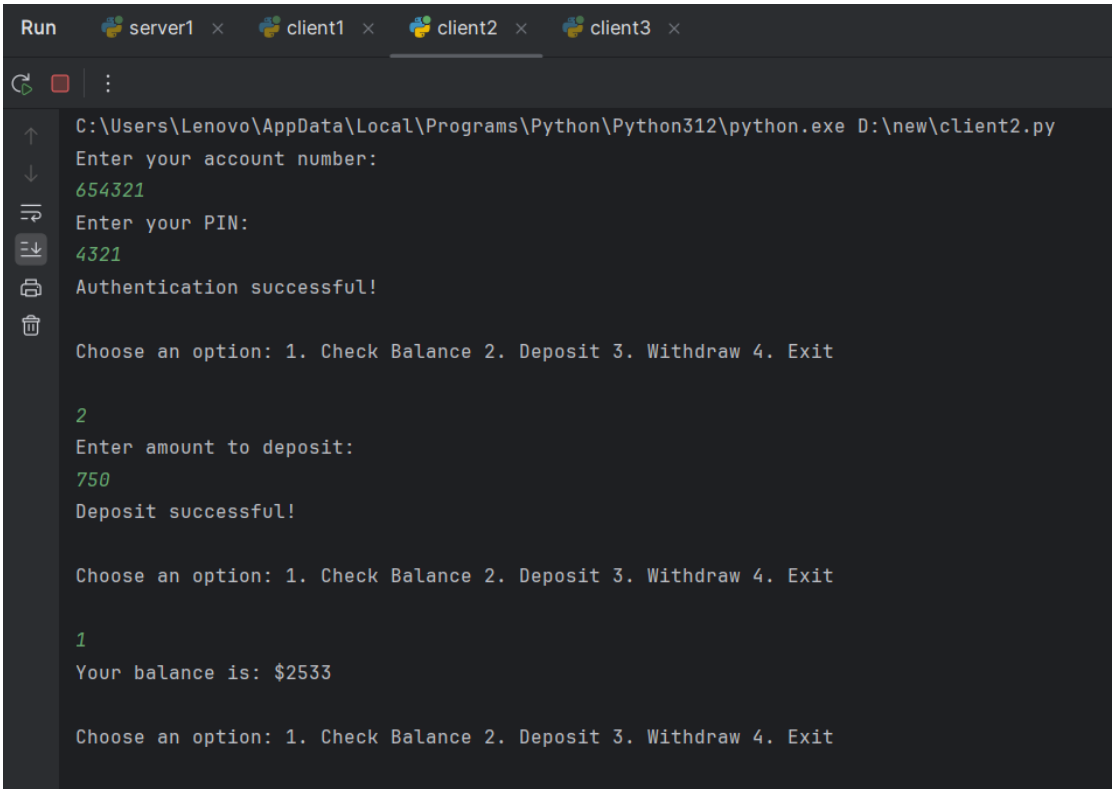
بهذه الطريقة، يتمكن العميل من التواصل مع الخادم وفقاً لبروتوكول محدد، معالجة المدخلات، والتفاعل مع النظام بشكل صحيح وآمن.

النتائج حيث تم اختبار اتصال ٣ عملاء في وقت واحد

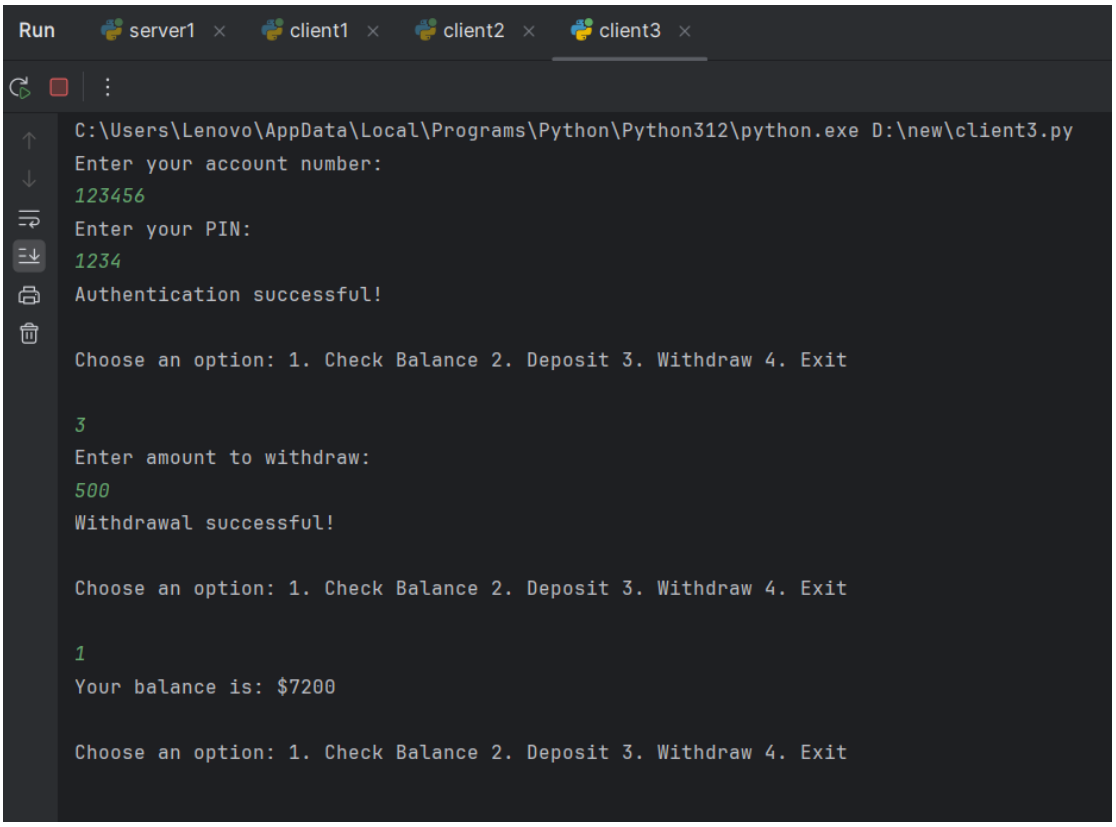
١- اتصال العميل الأول وطلب التحقق من القيمة



٢-اتصال العميل الثاني وطلب ادخال قيمة ثم عرضها



٣-العميل الثالث طلب سحب مبلغ وعرض قيمة الحساب



حل السؤال الثاني : تم تصميم لعبة حجرة ورقة مقص

```python
import random

choices = ['rock', 'paper', 'scissors']


def get_user_choice():
    while True:
        user_choice = input("Enter your choice (rock, paper, scissors): ").lower()
        if user_choice in choices:
            return user_choice
        else:
            print("Invalid choice. Please choose again.")


def get_computer_choice():
    return random.choice(choices)


def determine_winner(choice1, choice2):
    if choice1 == choice2:
        return "It's a tie!"
    elif (choice1 == 'rock' and choice2 == 'scissors') or \
            (choice1 == 'scissors' and choice2 == 'paper') or \
            (choice1 == 'paper' and choice2 == 'rock'):
        return "Player 1 wins!"
    else:
        return "Player 2 wins!"


def play_game():
    print("Welcome to Rock, Paper, Scissors!")
    mode = input("Choose mode (1: Player vs Computer, 2: Player vs Player): ")

    player1_score = 0
    player2_score = 0
    rounds_played = 0

    if mode == '1':
        while True:
            user_choice = get_user_choice()
            computer_choice = get_computer_choice()
            print(f"Computer chose: {computer_choice}")
            result = determine_winner(user_choice, computer_choice)
            print(result)

            rounds_played += 1
            if "Player 1" in result:
                player1_score += 1
            elif "Player 2" in result:
                player2_score += 1

            print(f"Score after round {rounds_played}: Player 1: {player1_score}, Computer: {player2_score}")

            play_again = input("Do you want to play again? (yes/no): ").lower()
            if play_again != 'yes':
                break

    elif mode == '2':
        while True:
            print("Player 1:")
            player1_choice = get_user_choice()
            print("Player 2:")
            player2_choice = get_user_choice()
            result = determine_winner(player1_choice, player2_choice)
            print(result)

            rounds_played += 1
            if "Player 1" in result:
                player1_score += 1
            elif "Player 2" in result:
                player2_score += 1

            print(f"Score after round {rounds_played}: Player 1: {player1_score}, Player 2: {player2_score}")

            play_again = input("Do you want to play again? (yes/no): ").lower()
            if play_again != 'yes':
                break

    else:
        print("Invalid mode selected. Please restart the game and choose a valid mode.")
        return

    print(f"\nGame over! You played {rounds_played} rounds.")
    print(f"Player 1 score: {player1_score}")
    if mode == '1':
        print(f"Computer score: {player2_score}")
    else:
        print(f"Player 2 score: {player2_score}")


if __name__ == "__main__":
    play_game()

```

شرح الكود :

يتم استيراد مكتبة random لاستخدامها في اختيار العشوائي للكمبيوتر.

تعريف قائمة الخيارات الممكنة للعبة: rock، paper، وscissors.

دالة get_user_choice:

هذه الدالة تطلب من المستخدم إدخال اختياره.

يتم التحقق من أن الاختيار صحيح (أي أنه واحد من الخيارات الثلاثة المحددة) وإلا يُطلب من المستخدم إعادة المحاولة.

دالة get_computer_choice:

هذه الدالة تقوم باختيار عشوائي لإحدى الخيارات الثلاثة للكمبيوتر باستخدام random.choice.

دالة determine_winner:

هذه الدالة تحدد الفائز بناءً على الخيارات المدخلة.

يتم تحديد حالة التعادل أو الفوز لأحد اللاعبين بناءً على قواعد اللعبة.

دالة play_game:

تطبع ترحيباً باللاعبين وتطلب منهم اختيار نمط اللعبة: لاعب ضد الكمبيوتر أو لاعب ضد لاعب.

يتم تهيئة المتغيرات لتتبع النقاط وعدد الجولات.

التعامل مع وضع اللعب:

إذا تم اختيار نمط "لاعب ضد الكمبيوتر"، يتم تكرار الخطوات التالية في حلقة:

الحصول على اختيار المستخدم واختيار الكمبيوتر.

تحديد الفائز وطباعة النتيجة.

تحديث عدد الجولات والنقاط.

سؤال المستخدم إذا كان يريد اللعب مجددًا.

إذا تم اختيار نمط "لاعب ضد لاعب"، يتم تكرار نفس الخطوات مع اختلاف أن كلا اللاعبين هما مستخدمين.

التعامل مع اختيار نمط غير صالح:

إذا أدخل المستخدم نمط غير صالح، يتم طباعة رسالة خطأ ويخرج من الدالة.

عرض النتيجة النهائية:

بعد انتهاء اللعبة، يتم عرض عدد الجولات التي تم لعبها والنقاط النهائية لكل لاعب.

تشغيل اللعبة:

يضمن أن دالة play_game تُستدعى فقط عندما يُشغّل هذا الملف كبرنامج رئيسي وليس عند استيراده كجزء من برنامج آخر

نتائج اختبار الكود :