

**Second Network Programming Homework**

**إعداد الطالبتين:**

**راما احمد حلوم (1528) - حنان ماهر طيبة (1890)**

**إشراف الدكتور:**

**مهند عيسى**

**الجمهورية العربية السورية**

**وزارة التعليم العالي والبحث العلمي**

**جامعة تشرين- اللاذقية**

**كلية الهندسة الميكانيكية والكهربائية**

**قسم هندسة الاتصالات والالكترونيات**

**السنة الخامسة**

**برمجة الشبكات**

**Question 1:** Bank ATM Application with TCP Server/Client and Multi-threading

import socket

import threading

HOST = '127.0.0.1'

PORT = 6666

accounts = {

    'Hanan': {'balance': 5000},

    'Rama': {'balance': 1500}}

def handle\_client(client\_socket, address):

    """Handles individual client connections."""

    print(f"New connection from {address}")

    while True:

        try:

            data = client\_socket.recv(1024).decode()

            if not data:

                break

            request = data.split('|')

            username = request[0]

            operation = request[1]

            amount = int(request[2]) if len(request) > 2 else 0

            if username not in accounts:

                client\_socket.sendall("Invalid username".encode())

            else:

                account = accounts[username]

                if operation == 'balance':

                    client\_socket.sendall(f"Your balance is: {account['balance']}".encode())

                elif operation == 'deposit':

                    account['balance'] += amount

                    client\_socket.sendall(f"Deposit successful. New balance: {account['balance']}".encode())

                elif operation == 'withdraw':

                    if amount > account['balance']:

                        client\_socket.sendall("Insufficient funds".encode())

                    else:

                        account['balance'] -= amount

                        client\_socket.sendall(f"Withdrawal successful. New balance: {account['balance']}".encode())

                else:

                    client\_socket.sendall("Invalid operation".encode())

        except Exception as e:

            print(f"Error handling client: {e}")

            break

    client\_socket.close()

    print(f"Connection closed with {address}")

with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as s:

    s.bind((HOST, PORT))

    s.listen()

    print(f"Server started on {HOST}:{PORT}")

    while True:

        conn, addr = s.accept()

        thread = threading.Thread(target=handle\_client, args=(conn, addr))

        thread.start()

شرح البرنامج:

تم بناء السيرفر وتعيين عنوان الـ IP له 127.0.0.1، وربطه على المنفذ 6666. باستدعاء المكتبتين socket وthreading، تم تخزين الحسابات في السيرفر على شكل قاموس dictionary له الاسم accounts بحيث نتعامل مع اسم صاحب الحساب كمفتاح للقاموس والقيمة المقابلة لها هي المبلغ المالي الإجمالي في الحساب. باستخدام التابع handle\_client نقوم بمعالجة طلبات الزبائن بعد قبول اتصالاتهم. نخزن الأوامر (البيانات) القادمة من الـ Clients في المتغير data، نفصل بين البيانات القادمة باستخدام |، في القائمة request العنصر الأول يمثل اسم المستخدم (صاحب الحساب في البنك) والعنصر الثاني هو نوع العملية المراد القيام بها، العنصر الثالث هو القيمة المراد سحبها أو إضافتها للحساب.

يتيح السيرفر العمليات التالية: كشف المبلغ المالي الكلي في الحساب، إيداع قيمة، سحب قيمة.

import socket

HOST = '127.0.0.1'

PORT = 6666

def main():

    with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as s:

        s.connect((HOST, PORT))

        while True:

            username = input("Enter your username: ")

            operation = input("Enter operation (balance, deposit, withdraw): ").lower()

            if operation == 'deposit' or operation == 'withdraw':

                amount = int(input("Enter amount: "))

            else:

                amount = 0

            request = f"{username}|{operation}|{amount}"

            s.sendall(request.encode())

            data = s.recv(1024).decode()

            print(data)

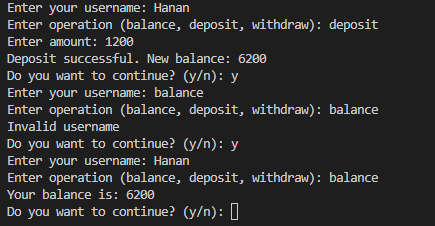
            if input("Do you want to continue? (y/n): ").lower() != 'y':

                break

        s.close()

if \_\_name\_\_ == '\_\_main\_\_':

    main()

يقوم الزبون باستخدام اسمه (اسم صاحب الحساب) ونوع العملية التي يريد القيام بها، إذا كانت العملية إيداع أو سحب يجب عليه تحديد المبلغ المالي المراد سحبه أو إيداعه.



**Question 2: guess the number**

import random

def guess\_the\_number():

  min\_number = 1

  max\_number = 100

  secret\_number = random.randint(min\_number, max\_number)

  num\_guesses = 0

  print(f"I'm thinking of a number between {min\_number} and {max\_number}. Try to guess it!")

  while True:

    try:

      guess = int(input("Take a guess: "))

      num\_guesses += 1

      if guess < secret\_number:

        print("Too low. Guess again.")

      elif guess > secret\_number:

        print("Too high. Guess again.")

      else:

        print(f"You guessed it! The number was {secret\_number} in {num\_guesses} guesses.")

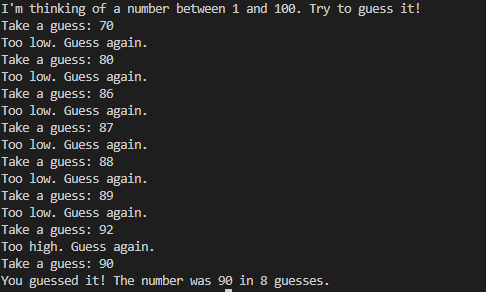
        break

    except ValueError:

      print("Invalid input. Please enter a number.")

if \_\_name\_\_ == "\_\_main\_\_":

  guess\_the\_number()



**نتيجة التنفيذ:**

1. import random:

تعتبر تعليمة في Python لاستيراد مكتبة random التي تتيح استخدام مختلف الوظائف المتعلقة بالأرقام العشوائية.

2. guess\_the\_number():

هذه هي دالة تقوم بتنفيذ لعبة تخمين الرقم. يتم فيها اختيار رقم عشوائي بين 1 و 100 باستخدام random.randint(min\_number, max\_number).

3. min\_number = 1 و max\_number = 100:

تعيين الحد الأدنى والحد الأقصى للرقم الذي يمكن تخمينه في اللعبة.

4. num\_guesses = 0:

عدد المرات التي يقوم فيها اللاعب بتخمين الرقم.

5. "I'm thinking of a number between {minnumber} and {maxnumber}. Try to guess it!":

رسالة تظهر للاعب لإعلامه بأن البرنامج يفكر في رقم بين الحدود التي حددت مسبقًا.

1. guess = int(input("Take a guess: "))

بينما يتم تنفيذ اللعبة، يُطلب من اللاعب إدخال تخمين للرقم.

1. الحلقة while True تمثل دورة لمساعدة اللاعب في تخمين الرقم بشكل متكرر حتى يتم تخمين الرقم الصحيح.
2. اختبار التخمين، حيث إذا كان الرقم أقل من الرقم السري secret\_number سيتم طباعة "Too low. Guess again."، وإذا كان أكبر فسيطبع "Too high. Guess again."، وإذا تم تخمين الرقم بشكل صحيح، يتم طباعة "You guessed it! The number was {secretnumber} in {numguesses} guesses.".
3. استثناء ValueError يُمسك بالأخطاء عندما يقوم المستخدم بإدخال سواء قيمة غير قابلة للتحويل إلى رقم صحيح.
4. الجزء الأخير if name == "main": guess\_the\_number() هو للتحقق مما إذا كان البرنامج يتم تشغيله كبرنامج رئيسي. إذا كان كذلك، يتم تنفيذ دالة guess\_the\_number() التي تعمل على تنفيذ لعبة تخمين الأرقام.