

**وظيفة مقرر برمجة الشبكات الثانية**

**إعداد الطالب:**

**حمزة عبد الله – علي عيسى**

**إشراف:**

**د.م. مهند عيسى**

**الجمهورية العربية السورية**

**وزارة التعليم العالي والبحث العلمي**

**جامعة تشرين**

**كلية الهندسة الميكانيكية والكهربائية**

**هندسة الاتصالات والالكترونيات**

**السنة الخامسة – مقرر برمجة الشبكات**

Second Network Programming Homework

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

كود السيرفر: حتى نجعل السيرفر يخدم عدد كبير من المستخدمين بنفس الوقت يجب الاستفادة من المودل threading، تم تعيين IP السيرفر على 0.0.0.0 من أجل نخديم أي عنوان بالشبكة، ورقم المنفذ على 8000. خزنت الحسابات في dictionary له الاسم accounts، اعتمدت على رقم الحساب في التخزين بحيث جعلت رقم الحساب هو المفتاح والمبلغ المالي هو القيمة المقابلة.

بتعرف التابع handle\_client(conn, addr) أتعامل مع اتصالات العملاء بحيث مررت له سوكيت العميل وهو البارمتر conn وعنوان العميل addr. استقبل معلومات العميل باستخدام:

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

وثم عن طريق تعريف المتحولات رقم الحساب ونوع العملية المرادة وإجمالي القيمة المضافة أو المسحوبة أستطيع فصل هذه البيانات باستخدام data.split().

من ثم حسب العملية التي يرد القيام بها العميل أستطيع تنفيذ ما يريد كما يلي:

try:

            amount = float(amount)

        except ValueError:

            conn.sendall("Invalid amount".encode())

            continue

        if operation == "check\_balance":

            balance = accounts[account\_number]

            conn.sendall(f"Your balance is: {balance}".encode())

        elif operation == "deposit":

            accounts[account\_number] += amount

            conn.sendall(f"Deposit successful. New balance: {accounts[account\_number]}".encode())

        elif operation == "withdraw":

            if accounts[account\_number] < amount:

                conn.sendall("Insufficient funds".encode())

            else:

                accounts[account\_number] -= amount

                conn.sendall(f"Withdrawal successful. New balance: {accounts[account\_number]}".encode())

        else:

            conn.sendall("Invalid operation".encode())

    conn.close()

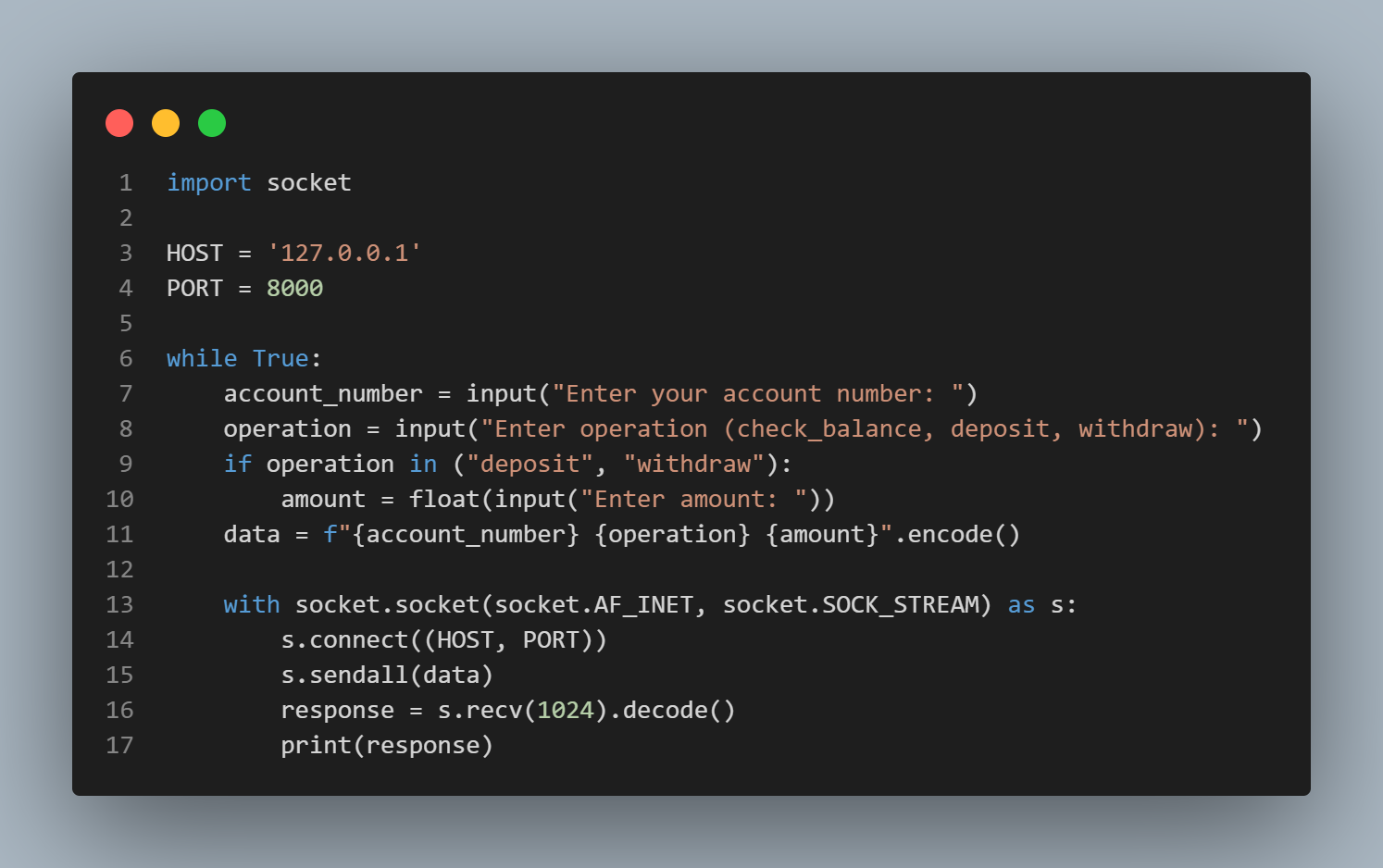
    print(f"Client {addr} disconnected")

بإنشاء الأوبجيكت (الغرض) thread أستطيع جعل السيرفر يتعامل مع المستخدمين بنفس الوقت بإمرار التابع target=handle\_client ومن ثم args=(conn, addr).

تشغيل السيرفر:

python tcp-bank-server.py

Server listening on 0.0.0.0:8000

برنامج العميل الأول:

عند تشغليه:

python tcp-bank-client1.py

Enter your account number: 1A

Enter operation (check\_balance, deposit, withdraw): deposit

Enter amount: 2000

Deposit successful. New balance: 3000.0

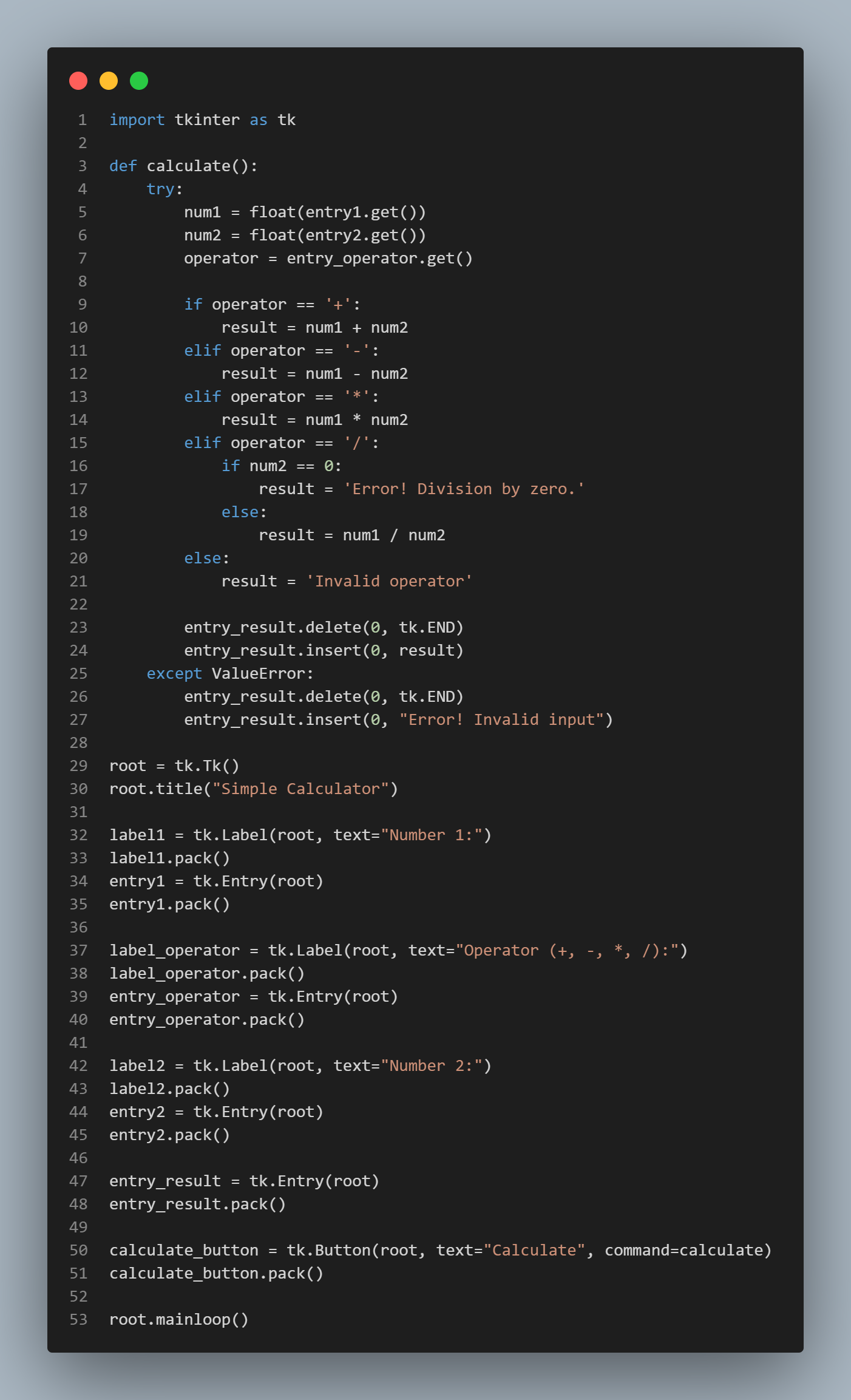
ما يظهر في واجهة السيرفر:

Server listening on 0.0.0.0:8000

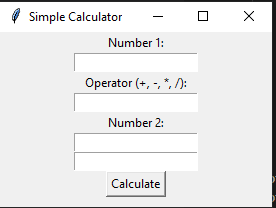
Connected by ('127.0.0.1', 7777)

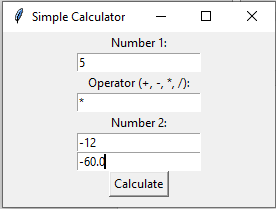
Client ('127.0.0.1', 7777) disconnected



**Question 2: Calculator App Using Tkinter**

عند تشغيل الكود:





قمت بالاستفادة من المودل tkinter لبناء واجهة بسيطة تظهر الآلة الحاسبة، تقوم الآلة الحاسبة هذه بتنفيذ العمليات الأساسية من جمع وضرب وطرح وقسمة.

يقوم التابع def calculate() باستقبال عدديين من المستخدم وتحديد المحرف عن طريق المتغير operator ثم اختبار operator لتحديد نوع العملية.

تعليمات tkinter المستخدمة:

root = tk.Tk():

هذا السطر يقوم بإنشاء نافذة رئيسية للتطبيق باستخدام الميثود TK().

root.title("Simple Calculator"):

هذا السطر يقوم بتعيين عنوان للنافذة الرئيسية، وفي هذه الحالة العنوان هو "Simple Calculator".

label1 = tk.Label(root, text="Number 1:"):

إنشاء (مكون واجهة مستخدم يعرض النص) داخل النافذة الرئيسية (root) ويحدد نص هذه العلامة بـ "Number 1:".

label1.pack():

تقوم بوضع العلامة التي تم إنشاؤها في النافذة الرئيسية وترتيبها حسب التصميم الافتراضي، لأنني لم أعدل القياسات للواجهة أو تصميمها.

entry1 = tk.Entry(root):

إنشاء حقل إدخال نصي في النافذة الرئيسية حيث يمكن للمستخدم إدخال الرقم الأول.

entry1.pack():

تضع هذه العلامة حقل الإدخال في النافذة الرئيسية.

label\_operator = tk.Label(root, text="Operator (+, -, \*, /):"):

نشاء علامة أخرى تحمل نص "Operator (+, -, \*, /):" لتوجيه المستخدم لإدخال العملية الحسابية التي يريد إجراؤها.

calculate\_button = tk.Button(root, text="Calculate", command=calculate):

ينشئ زر بالنص "Calculate" ويتم تعيين الدالة calculate ليتم تنفيذها عند الضغط على الزر.

root.mainloop():

هذا السطر يبدأ حلقة الحدث الرئيسية للنافذة، مما يجعل النافذة تظهر وتكون قادرة على الاستجابة للأحداث مثل النقرات على الأزرار وإدخال النص.