Syrian Arab Republic

Lattakia - Tishreen University

Department of Communication and electrical engineering

5th , Network Programming : Homework No1

الجمهورية العربية السورية

اللاذقية - جامعـــة تشريـــن

كلية الهندسة الكهربائية والميكانيكية

قسم هندسة الاتصالات والالكترونيات

السنة الخامسة: وظيفة1 برمجة شبكات

Second Network Programming Homework

الاسم	الرقم
يوسف حسن سالم	2498
حسن محمد محمد	2447
حسين سلمان سليمان	2568

بإشراف الدكتور : د. مهند عيسى

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

server code :

```
import socket
import threading

# قائمة الحسابات المصرفية
accounts = [
    {"account_number": "111111", "balance": 1000},
    {"account_number": "222222", "balance": 2000},
    {"account_number": "333333", "balance": 1500}
]


# دالة لمعالجة طلبات العميل
def handle_client(client_socket):
    try:
        client_socket.send(b"Welcome to the ATM. Please enter your account
number:")
        account_number = client_socket.recv(1024).decode()
        account = next((acc for acc in accounts if acc["account_number"] ==
account_number), None)

        if not account:
            client_socket.send(b"Account not found. Disconnecting.")
            client_socket.close()
            return

        client_socket.send(b"Account authenticated. Enter your command
(balance/deposit/withdraw/exit):")
        while True:
            command = client_socket.recv(1024).decode()

            if command == "balance":
                client_socket.send(f"Your balance is
{account['balance']}".encode())
            elif command.startswith("deposit"):
                amount = int(command.split()[1])
                account['balance'] += amount
                client_socket.send(f"Deposited {amount}. New balance is
{account['balance']}".encode())
            elif command.startswith("withdraw"):
                amount = int(command.split()[1])
                if amount > account['balance']:
                    client_socket.send(b"Insufficient funds.")
                else:
                    account['balance'] -= amount
                    client_socket.send(f"Withdrew {amount}. New balance is
{account['balance']}".encode())
            elif command == "exit":
                client_socket.send(f"Final balance is {account['balance']}.
Disconnecting.".encode())
                break
            else:
                client_socket.send(b"Invalid command.")
    except Exception as e:
        print(f"Error: {e}")
```

```
    finally:
        client_socket.close()


# إعداد الخادم
def start_server():
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind(("0.0.0.0", 9999))
    server.listen(5)
    print("Server started and listening on port 9999")

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


start_server()
```

شرح الكود :

هذا الكود يقوم بإنشاء خادم بسيط يشبه أجهزة الصراف الآلي (ATM) باستخدام مكتبات socket و threading في لغة البرمجة بايثون. دعونا نشرح كل جزء من الكود:

1. استيراد المكتبات:

يتم استيراد مكتبتين:

- socket: لإنشاء الاتصال بالشبكة والتعامل مع الاتصالات بين الخادم والعميل.
- threading: لإنشاء خيوط (threads) تمكن من التعامل مع طلبات متعددة من العملاء في نفس الوقت.

2. قائمة الحسابات المصرفية :

يتم تعريف قائمة تحتوي على حسابات مصرفية لكل حساب رقم حساب ورصيد.

3 دالة لمعالجة طلبات العميل:

هذه الدالة تقوم بالتالي:

- إرسال رسالة ترحيبية للعميل وطلب رقم الحساب.
- استقبال رقم الحساب من العميل والتحقق من وجوده في قائمة الحسابات.
- إذا لم يتم العثور على الحساب، يتم إرسال رسالة إلى العميل ويتم قطع الاتصال.

- إذا تم العثور على الحساب، يتم إعلام العميل بأنه قد تم التحقق من الحساب، ويُطلب منه إدخال الأوامر.

- الدالة تدخل في حلقة تستقبل الأوامر (balance, deposit, withdraw, exit) وتنفذ العمليات المطلوبة.

- التعامل مع استثناءات الطوارئ وإغلاق الاتصال في النهاية

4.إعداد الخادم:

هذه الدالة تقوم بالتالي:

- إنشاء مقبس خادم باستخدام.socket.socket

- ربط الخادم بجميع عناوين IP المحلية على المنفذ 9999.

- وضع الخادم في وضع الاستماع للاتصالات الواردة مع تحديد الحد الأقصى لطابور الاتصالات إلى 5.

- عند قبول اتصال من عميل، يتم إنشاء خيط جديد للتعامل مع هذا العميل باستخدام دالة.handle_client

- الخادم يستمر في الاستماع للاتصالات الجديدة في حلقة لا نهائية.

5.بدء الخادم :

استدعاء دالة start_server لتشغيل الخادم.

Client code :

```
import socket
def start_client():
    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 "Disconnecting" in response:
            break
        command = input("Enter command: ")
        client.send(command.encode())
start_client()
```

شرح الكود :

هذا الكود ينشئ عميل بسيط يتصل بخادم عبر الشبكة باستخدام مكتبة socket في بايثون. دعونا نشرح كل جزء من الكود:

1. استيراد مكتبة socket:

يتم استيراد مكتبة socket، وهي مكتبة قياسية في بايثون تتيح التعامل مع الاتصالات الشبكية.

2. دالة start_client:

هذه الدالة تقوم بالتالي:

إنشاء مقبس (socket) جديد يستخدم بروتوكول IPv4 (AF_INET) وبروتوكول TCP (SOCK_STREAM).

- الاتصال بالخادم:

client.connect(("127.0.0.1", 9999))

يتصل العميل بالخادم على العنوان المحلي (127.0.0.1) والمنفذ (9999). العنوان 127.0.0.1 يشير إلى الجهاز المحلي (localhost).

- حلقة الاتصال

يستقبل العميل رسائل من الخادم باستخدام recv بحجم 4096 بايت، ويقوم بفك ترميز الرسالة من صيغة البايت إلى نص باستخدام decode، ثم يطبع الرسالة.

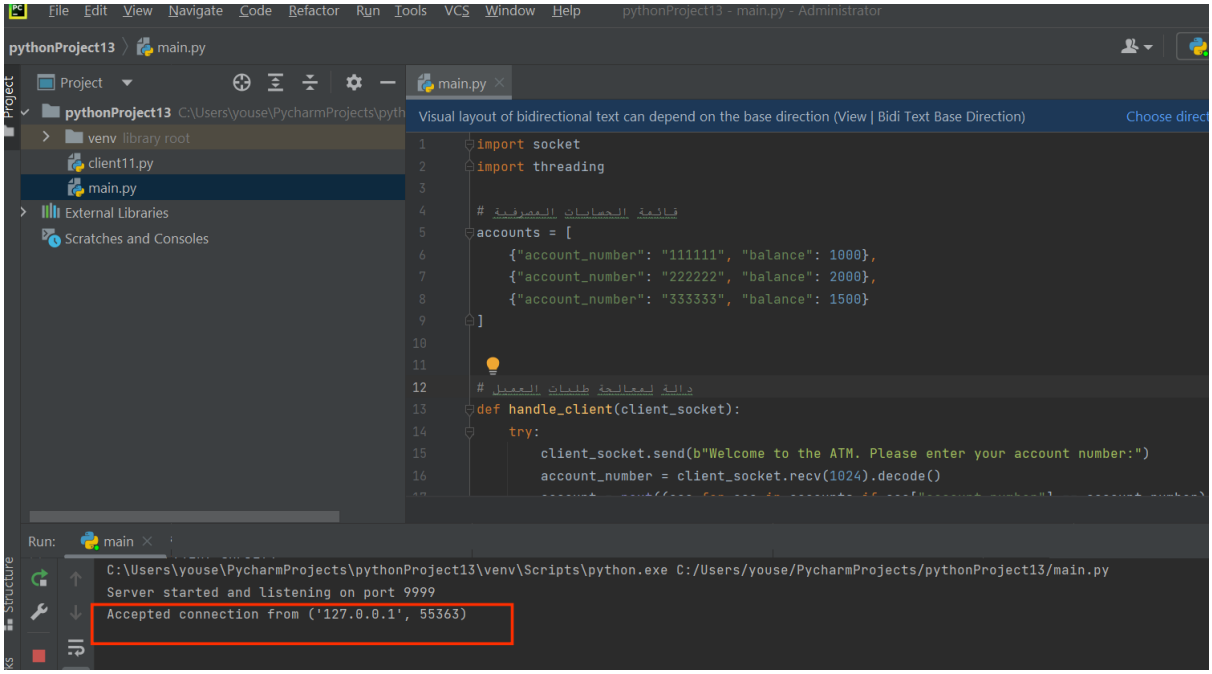
- التحقق من رسالة الانفصال:

إذا كانت الرسالة المستقبلة تحتوي على كلمة "Disconnecting"، فهذا يعني أن الخادم ينهي الاتصال، ويكسر الحلقة ويخرج من الدالة.

- إرسال الأوامر للخادم:

إذا لم تتضمن الرسالة كلمة "Disconnecting"، يطلب العميل من المستخدم إدخال أمر (مثل balance, deposit, withdraw)، ثم يرسل هذا الأمر إلى الخادم بعد تحويله إلى صيغة البايت باستخدام encode.

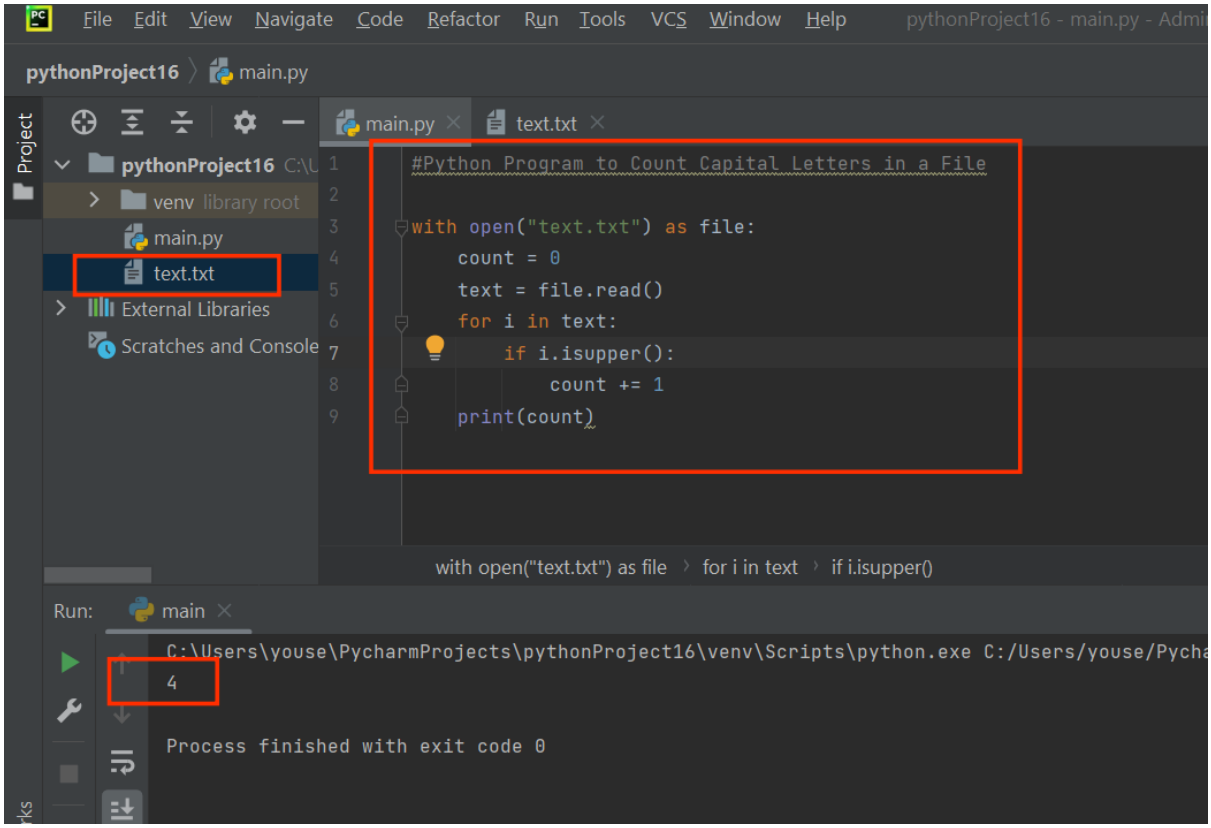
3. بدء العميل: باستدعاء دالة start_client لتشغيل العميل والبدء في الاتصال بالخادم وتنفيذ الأوامر.



توضيح عملية السحب والإيداع وتفقد الرصيد :

```
C:\Users\youse\PycharmProjects\pythonProject14\venv\Scripts\python.exe C:/Users/youse/PycharmProjects/pythonProject14/main.py
Welcome to the ATM. Please enter your account number:
Enter command: 111111
Account authenticated. Enter your command (balance/deposit/withdraw/exit):
Enter command: balance
Your balance is 1000
Enter command: deposit 100
Deposited 100. New balance is 1100
Enter command: withdraw 100
Withdrew 100. New balance is 1000
Enter command:
```

Question 2 : Python Program to Count Capital Letters in a File



أولاً نفتح ملف نصي تم حفظه بالفعل على جهاز الكمبيوتر

ثم تقديم متغير باسم count، والذي يستخدم هنا لتخزين عدد الأحرف الكبيرة.

في البداية، أعلنا أن قيمته هي 0، وفي السطر التالي، قراءة الملف النصي

ثم أستخدمنا حلقة for فوق محتوى الملف النصي وأستخدم عبارة if داخل حلقة for والتي ستستمر في إضافة 1 إلى متغير العدد حتى تستمر في العثور على أحرف كبيرة في الملف باستخدام وظيفة isupper() في بايثون.