جامـــــــعة تشـــــــرين
كلـــية الهندسة الميكانيكية والكهربائية
قســـــم الاتـــصالات
الســـــنة الخامســـــة

Second Network Programming Homework

إشراف الدكتور : مهند عيسى

تقديم الطلاب:
أحمد بشار عبد الرحمن 2014
إيلين عبد الله غصة 2258

Question 1:

Bank ATM Application with TCP Server/Client and Multi-threading
Project Description:
Build a TCP server and client Bank ATM application using Python. The server should handle multiple client connections simultaneously using multi-threading. The application should allow clients to connect, perform banking operations (such as check balance, deposit, and withdraw), and receive their updated account status upon completion.

الحل:

server: كود

```
import socket

def main():
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(('127.0.0.1', 2014))

account_number = input("Enter your account number: ")
client.send(account_number.encode())

print(client.recv(1024).decode())

while True:
print("\nOptions:")
print("1. Check Balance")
print("2. Deposit")
print("3. Withdraw")
print("4. Exit")
option = input("Enter option: ")

if option == '4':
break

client.send(option.encode())

if option == '1':
print(client.recv(1024).decode())
elif option == '2' or option == '3':
amount = input("Enter amount: ")
client.send(amount.encode())
print(client.recv(1024).decode())

client.close()
if name == "main":
main()
```

شرح الكود :

هذا الكود يُمثل جانب العميل (client) في تطبيق عميل خادم (client-server application) باستخدام (socket programming) في. هدفه توفير طريقة للتواصل مع خادم عبر شبكة باستخدام مقابس TCP.

1. استيراد وحدة المقابس **(socket):**

import socket

يبدأ الكود بإدراج وحدة

socket

، وهي واجهة برمجة التطبيقات (API) المستخدمة في بايثون لتمكين الاتصالات عبر الشبكة.

2. تعريف الدالة **main():**

def main()

يُدرج بعدها تعريف

main()

، وهي الدالة الرئيسة التي تحتوي على منطق تشغيل العميل.

3. إنشاء كائن المقبس:

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

يتم إنشاء كائن المقبس باستدعاء

socket.socket()

وتحديد

socket.AF_INET

لاستخدام بروتوكول الإنترنت النسخة الرابعة (IPv4) و

socket.SOCK_STREAM

لاستخدام بروتوكول التحكم في النقل (TCP) للاتصالات الموثوقة والمتسلسلة.

4. الاتصال بالخادم:

client.connect(('127.0.0.1', 2014))

يتصل العميل بالخادم على العنوان

127.0.0.1

(العنوان المحلي loopback) على المنفذ

2014

. هذا يعني أنه يتوقع تشغيل خادم على نفس الجهاز.

5. إرسال واستقبال البيانات:

يطلب من المستخدم إدخال رقم حسابه، يُكود هذا الإدخال إلى صيغة بايتس باستخدام

.encode()

ويُرسل إلى الخادم. ثم ينتظر رداً من الخادم ويطبعه على الشاشة بعد فك تكويد الرسالة الواردة من صيغة بايتس إلى نص باستخدام

.decode()

.

6.الدوران في قائمة الخيارات:

تُعرض للمستخدم قائمة خيارات للتحقق من الرصيد، الإيداع، السحب، أو الخروج. يُقرأ خيار المستخدم ويُرسل إلى الخادم. بناءً على الخيار، إذا كان القصد الاستفسار عن الرصيد، يُطبع الرد مباشرة. لعمليات الإيداع والسحب، يُطلب من المستخدم إدخال المبلغ ويُرسل هذا المبلغ إلى الخادم للمعالجة ويُطبع الرد.

7.إغلاق الاتصال:

client.close()

بمجرد الانتهاء من جميع العمليات واختيار الخروج من الدورة، يتم إغلاق اتصال المقبس لتحرير الموارد.

كود client:

```
import socket
import threading

# Bank account details
accounts = {
 '2014': 15000,
 '2258': 90000,
 '2204': 52500,
 }


def handle_client(client_socket):
account_number = client_socket.recv(1024).decode()
if account_number in accounts:
client_socket.send(b"Welcome! You have connected to the bank server.")
else:
client_socket.send(b"Invalid account number. Connection terminated.")
client_socket.close()
return

while True:
option = client_socket.recv(1024).decode()

if option == '1':
balance = accounts[account_number]
client_socket.send(f"Your current balance is: {balance}".encode())
elif option == '2':
amount = int(client_socket.recv(1024).decode())
accounts[account_number] += amount
client_socket.send(f"Deposit successful. Your new balance is:
{accounts[account_number]}".encode())
elif option == '3':
amount = int(client_socket.recv(1024).decode())
if accounts[account_number] >= amount:
```

```
                accounts[account_number] -= amount
                client_socket.send(f"Withdrawal successful. Your new balance is: {accounts[account_number]}".encode())
            else:
                client_socket.send("Insufficient funds. Withdrawal failed.".encode())
        else:
            break

    client_socket.close()


def start_server():
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind(('127.0.0.1', 2014))
    server.listen(2)
    print("Server listening on port 2014...")

    while True:
        client_socket, address = server.accept()
        print(f"Connection from {address} established.")
        client_thread = threading.Thread(target=handle_client, args=(client_socket,))
        client_thread.start()


start_server()
```

شرح الكود

هذا الكود يُنشئ خادمًا بسيطًا لبنك باستخدام السوكت (socket) ، يمكن من خلاله إدارة حسابات العملاء بشكل أساسي من خلال الإيداع والسحب والاستعلام عن الرصيد.

1. المكتبات المستوردة:

- socket : لتمكين الاتصالات الشبكية بين العميل والخادم.
- threading : لتمكين معالجة عدة طلبات عملاء في الوقت نفسه، حيث يُنشئ الخادم خيطًا جديدًا لكل عميل.

2. تفاصيل الحساب البنكي:

يتم تعريف قاموس accounts، حيث أن كل مفتاح هو رقم حساب وكل قيمة هي الرصيد المتاح في الحساب.

3. دالة handle_client:

هذه الدالة تتعامل مع كل عميل يتصل بالخادم. تُنفذ الخطوات التالية:

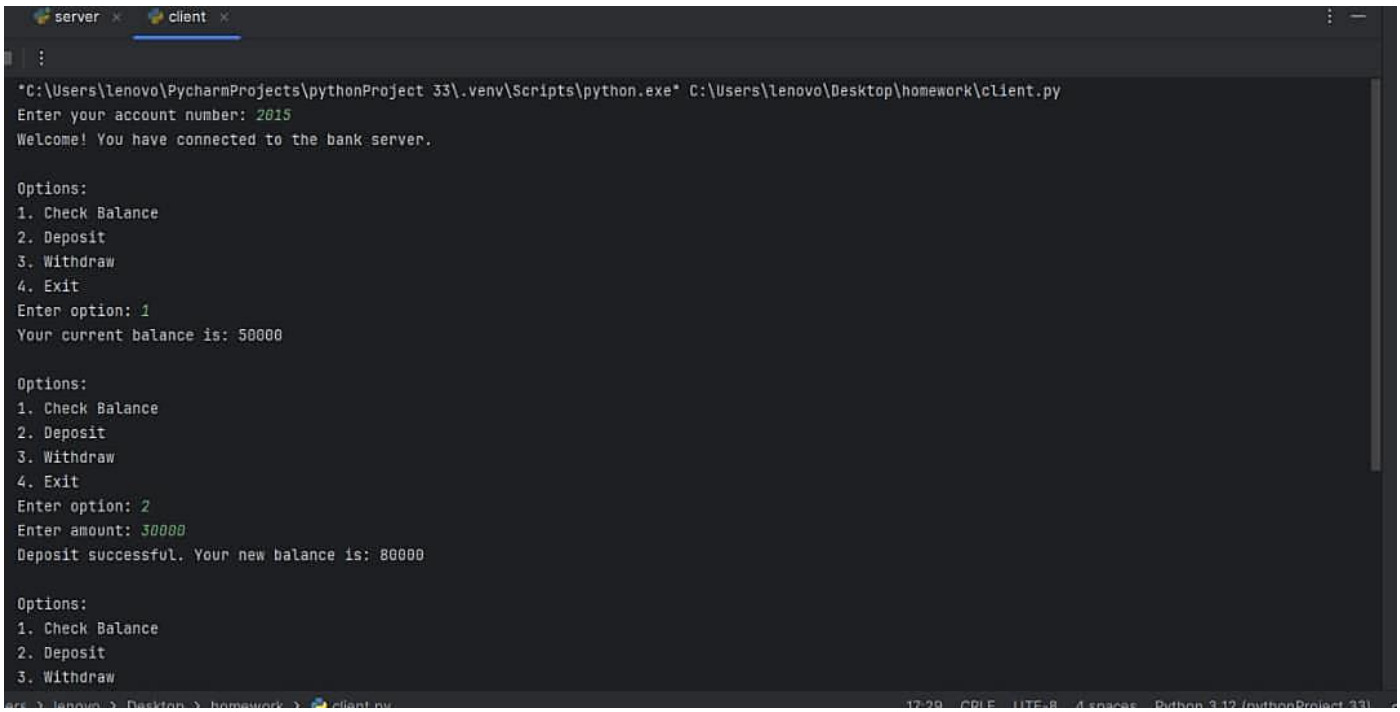
- تستقبل رقم الحساب من العميل وتتحقق مما إذا كان الرقم صحيحًا. إذا لم يكن كذلك، يتم إرسال رسالة خطأ ويُنهى الاتصال.
- تدخل في حلقة لا نهاية لها تستمع خلالها لطلب العميل:
- الاستعلام عن الرصيد
- الإيداع

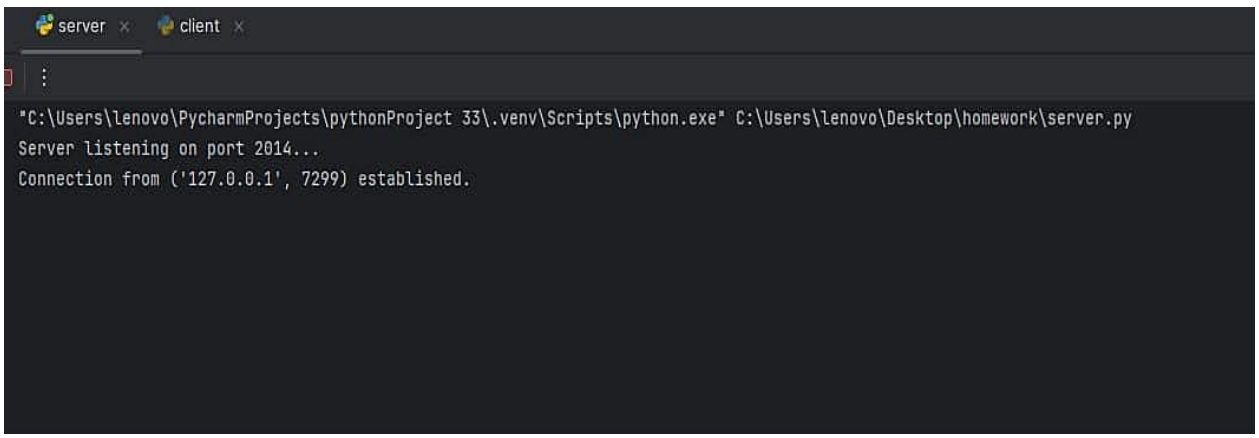
- السحب
- انتهاء الجلسة إذا تم إرسال خيار غير معرف.

4. دالة start_server:

هذه الدالة تعمل على إعداد وبدء تشغيل الخادم. تُنفذ الخطوات التالية:
- تنشئ كائن سوكت socket يعمل على البروتوكول 4IPv ونوع الاتصال (TCP SOCK_STREAM).
- تربط السوكت بعنوان IP (127.0.0.1 - الجهاز المحلي) والمنفذ 2014.
- يبدأ الاستماع على المنفذ المحدد، مع تحديد عدد الاتصالات القادمة التي يمكن قبولها بالوقت نفسه.
- داخل حلقة لا نهاية لها، يقبل الخادم الاتصالات الواردة ويُنشئ خيطًا (thread) جديدًا لكل اتصال باستخدام دالة handle_client.

بشكل إجمالي، يُظهر هذا الكود كيفية إنشاء خادم بسيط لمحاكاة وظائف البنك الأساسية مع استخدام ميزة الخيوط لتمكين التعامل مع عدة عملاء في الوقت نفسه





Question 2:

Simple Website Project with Python Flask Framework (you have choice to use Django or any Other Deferent Useful Python Project “from provide Project Links”)
Create a simple website with multiple pages using Flask, HTML, CSS, and Bootstrap. The website should demonstrate your understanding of web design principles.

الحل:

```
from datetime import datetime

def calculate_age(birthdate):
today = datetime.today()
age = today.year - birthdate.year - ((today.month, today.day) < (birthdate.month,
birthdate.day))

months = today.month - birthdate.month - (today.day < birthdate.day)
if months < 0:
months += 12

days = today.day - birthdate.day
if days < 0:
# حساب عدد الأيام في شهر الميلاد
days_in_birth_month = (birthdate.replace(month=birthdate.month % 12 + 1, day=1) -
birthdate).days
days += days_in_birth_month

return age, months, days

year = int(input("أدخل سنة ميلادك (YYYY): "))
month = int(input("أدخل شهر ميلادك (MM): "))
day = int(input("أدخل يوم ميلادك (DD): "))
birthdate = datetime(year, month, day)

age, months, days = calculate_age(birthdate)

print(f"عمرك هو: {age} سنوات و {months} شهور و {days} أيام.")
```

شرح الكود :

هذا الكود يقوم بحساب العمر بالسنوات والشهور والأيام بناءً على تاريخ الميلاد المدخل من قبل المستخدم

1. استيراد المكتبة:

```
from datetime import datetime
```

- هنا، يتم استيراد الفئة datetime من مكتبة datetime. هذه الفئة تتيح لنا العمل بسهولة مع التواريخ والأوقات.

2. تعريف الدالة calculate_age:

```
def calculate_age(birthdate):
```

- تعريف دالة تُدعى calculate_age تقوم بحساب العمر، وهي تأخذ معامل واحد هو birthdate الذي يمثل تاريخ الميلاد.

3. حساب العمر:

```
today = datetime.today()
age = today.year - birthdate.year - ((today.month, today.day) < (birthdate.month,
birthdate.day))
```

- يتم حساب العمر بالسنين عن طريق طرح سنة الولادة من السنة الحالية. الجزء ((today.month, today.day) < (birthdate.month, birthdate.day))) هو شرط يُطبق لمعرفة إذا مر تاريخ الميلاد في العام الحالي أم لا. إذا لم يتم بعد، يُخصم سنة واحدة من العمر.

4. حساب الشهور والأيام:

- للشهور:

```
months = today.month - birthdate.month - (today.day < birthdate.day)
if months < 0:
    months += 12
```

هنا يتم حساب عدد الشهور بنفس طريقة السنوات تقريباً. إذا كانت النتيجة سالبة، يعني أننا لم نصل بعد لشهر الميلاد في السنة الحالية، فنضيف 12 شهراً إلى النتيجة.

- للأيام:

```
days = today.day - birthdate.day
if days < 0:
    days_in_birth_month = (birthdate.replace(month=birthdate.month % 12 + 1, day=1) -
birthdate).days
    days += days_in_birth_month
```

تُحسب عدد الأيام المتبقية حتى تاريخ الميلاد. إذا كانت النتيجة سالبة، يُحسب عدد الأيام في شهر الميلاد لتصحيح النتيجة.

5. قراءة تاريخ الميلاد من المستخدم:

```
year = int(input("أدخل سنة ميلادك (YYYY): "))
month = int(input("أدخل شهر ميلادك (MM): "))
day = int(input("أدخل يوم ميلادك (DD): "))
```

birthdate = datetime(year, month, day)

- يُطلب من المستخدم إدخال تاريخ ميلاده على شكل سنة، شهر، ويوم. ثم يتم إنشاء كائن datetime باستخدام هذه المدخلات.

6. طباعة العمر:

age, months, days = calculate_age(birthdate)
print(f"عمرك هو: {age} سنوات و {months} شهور و {days} أيام.")

- يتم استدعاء الدالة calculate_age باستخدام تاريخ الميلاد المدخل وتُخزن القيم الناتجة (السنين، الشهور، والأيام) في المتغيرات age, months, days. ثم تُطبع النتيجة بشكل واضح للمستخدم.

Q2.py

```python
from datetime import datetime
1 usage
def calculate_age(birthdate):
 today = datetime.today()
 age = today.year - birthdate.year - ((today.month, today.day) < (birthdate.month, birthdate.day))
 months = today.month - birthdate.month - (today.day < birthdate.day)
 if months < 0:
    months += 12
 days = today.day - birthdate.day
 if days < 0:
 # حساب عدد الأيام في شهر الميلاد
   days_in_birth_month = (birthdate.replace(month=birthdate.month % 12 + 1, day=1) - birthdate).days1998
   days += days_in_birth_month
 return age, months, days
year = int(input("أدخل سنة ميلادك (YYYY): "))
month = int(input("أدخل شهر ميلادك (MM): "))
day = int(input("أدخل يوم ميلادك (DD): "))
birthdate = datetime(year, month, day)

age, months, days = calculate_age(birthdate)

print(f"عمرك هو: {age} سنوات و {months} شهور و {days} أيام.")

```

```
C:\Users\lenovo\PycharmProjects\pythonProject\.venv\Scripts\python.exe C:\Users\lenovo\Desktop\Q2.py
أدخل سنة ميلادك (YYYY): 1998
أدخل شهر ميلادك (MM): 12
أدخل يوم ميلادك (DD): 8
عمرك هو: 25 سنوات و 6 شهور و 1 أيام.

Process finished with exit code 0
```