وظيفة برمجة الشبكات الثانية

Second Network

Programming Homework

إشراف الدكتور المهندس: مهند عيسى

إعداد الطالبتين:

هديل غياث إسماعيل 2827 حلا جابر جعبري 2886

السنة الخامسة

هندسة الاتصالات والالكترونيات

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

كود السيرفر :

bank-server.py > handle_client

```python
import socket
import threading

accounts = {
    "Hadeel": 1000,
    "Hala": 1500,
    "Hadeel2": 2000,
    "Hala2": 2500
}

def handle_client(client_socket, client_address):
    print("Accepted connection from {}".format(client_address))

    client_socket.sendall(b"Welcome to the Bank ATM. Please enter your account name: ")
    account_name = client_socket.recv(1024).decode().strip()

    if account_name not in accounts:
        client_socket.sendall(b"Account not found. Connection closed.")
        client_socket.close()
        return

    client_socket.sendall("Hello, {}. Please enter your PIN: ".format(account_name).encode())
    pin = client_socket.recv(1024).decode().strip()
```

```python
while True:
        client_socket.sendall(b"Available operations:\n1. Check
Balance\n2. Deposit\n3. Withdraw\n4. Exit\nEnter your choice: ")
        choice = client_socket.recv(1024).decode().strip()

        if choice == "1":
            client_socket.sendall("Your current balance is:
{}".format(accounts[account_name]).encode())
        elif choice == "2":
            amount = float(client_socket.recv(1024).decode().strip())
            accounts[account_name] += amount
            client_socket.sendall("Deposit successful. Your updated
balance is: {}".format(accounts[account_name]).encode())
        elif choice == "3":
            amount = float(client_socket.recv(1024).decode().strip())
            if amount > accounts[account_name]:
                client_socket.sendall(b"Insufficient funds.")
            else:
                accounts[account_name] -= amount
                client_socket.sendall("Withdrawal successful. Your updated
balance is: {}".format(accounts[account_name]).encode())
        elif choice == "4":
```

```
            break

    print("Closing connection with {}".format(client_address))
    client_socket.close()

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(("0.0.0.0", 5555))
server.listen(5)
print("Server listening on port 0.0.0.0")

while True:
    client_socket, client_address = server.accept()
    client_thread = threading.Thread(target=handle_client,
args=(client_socket, client_address))
    client_thread.start()
```

شرح الكود:

تم استيراد المكتبتين socket وthreading لإنشاء المقابس وجعل السيرفر يخدم العملاء بنفس الوقت. باستخدام المتغير accounts تم بناء الحسابات بحيث يكون اسم صاحب الحساب هو المفتاح Key للقاموس والمبلغ المالي في الحساب هو القيمة Value المقابلة للمفتاح.

جعلنا السيرفر يعمل على أي عنوان IP في التطبيق باستخدام server.bind(("0.0.0.0", 5555)) ومع ملاحظة أننا جعلنا رقم المنفذ للسيرفر هو: 5555.

باستخدام التابع handle_client(client_socket, client_address) تم التعامل مع اتصالات العملاء جميعها كما يلي: تم تمرير البارمترين client_socket وclient_address اللذان يمثلان مقبس العميل وعنوان العميل وذلك عند قبول الاتصال معه، وعند قبول الاتصال نقوم بإرسال رسالة ترحيبية للمستخدم نطلب فيها إدخال اسمه (اسم صاحب الحساب) ونخزن الاسم في المتحول account_name ثم نتحقق من كون اسم صاحب الحساب موجود فعلاً في الحسابات في البنك أم لا ونطبع في حال لم يكن موجودا الرسالة: .Account not found. Connection closed.

إن كان اسم المستخدم موجود فعلاً أي لديه حساب نطلب منه إدخال رمز أمان له PIN، ومن ثم نعرض عليه جميع الخيارات الممكنة: Available operations:\n1. Check Balance\n2. Deposit\n3. Withdraw\n4. Exit

نتابع حسب رقم الخيار الذي يختاره العميل: بالعملية المرادة.

في النهاية حتى نجعل كود السيرفر يتعامل مع العملاء بنفس الوقت نقوم بكتابة التعليمة:

client_thread = threading.Thread(target=handle_client, args=(client_socket, client_address))

المتمثلة بإنشاء غرض من الصنف Thread.

عند تشغيل كود السيرفر:

Server listening on port 0.0.0.0

برامج العميل:

```
import socket

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(("127.0.0.1", 5555))

response = client.recv(1024).decode()
print(response)
account_name = input()
client.sendall(account_name.encode())

response = client.recv(1024).decode()
print(response)
pin = input()
client.sendall(pin.encode())

while True:
    response = client.recv(1024).decode()
    print(response)
    choice = input()
    client.sendall(choice.encode())

    if choice == "4":
        break

    if choice in ["2", "3"]:
        amount = input("Enter amount: ")
        client.sendall(amount.encode())
```

```
    operation_result = client.recv(1024).decode()
    print(operation_result)

client.close()
```

```
import socket

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(("127.0.0.2", 5555))

response = client.recv(1024).decode()
print(response)
account_name = input()
client.sendall(account_name.encode())

response = client.recv(1024).decode()
print(response)
pin = input()
client.sendall(pin.encode())

while True:
    response = client.recv(1024).decode()
    print(response)
    choice = input()
    client.sendall(choice.encode())

    if choice == "4":
        break

    if choice in ["2", "3"]:
        amount = input("Enter amount: ")
        client.sendall(amount.encode())

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

client.close()
```

تشغيل البرامج:

Welcome to the Bank ATM. Please enter your account name:

Hadeel

Hello, Hadeel. Please enter your PIN:

963

Available operations:

1. Check Balance

2. Deposit

3. Withdraw

4. Exit

Enter your choice:

1

Your current balance is: 1000

Welcome to the Bank ATM. Please enter your account name:

Hala

Hello, Hala. Please enter your PIN:

123

Available operations:

1. Check Balance

2. Deposit

3. Withdraw

4. Exit

Enter your choice:

2

Enter amount: 1020

Deposit successful. Your updated balance is: 2520.0

Available operations:

1. Check Balance

2. Deposit

3. Withdraw

4. Exit

Enter your choice:

1

Your current balance is: 2520.0

Question 2: generate passwords Program

```
import random
import string

def generate_password(length):

    characters = string.ascii_letters + string.digits + string.punctuation

    password = ''.join(random.choice(characters) for _ in range(length))

    if not any(char.isdigit() for char in password):
        password = password[:random.randint(0, length - 1)] +
random.choice(string.digits) + password[random.randint(0, length - 1):]

    return password

length = int(input("Enter the desired password length: "))

password = generate_password(length)

print(f"Your password is: {password}")
```

الغرض من الكود إنشاء مولد كلمات مرور قوية مكونة من محارف وأرقام، وذلك بجعل المستخدم يحدد طول الكلمة التي يحتاجها وعلى أساسها نولد السلسلة.

1. import random
2. import string

يتم استيراد المودل random لتوليد المحارف والأرقام العشوائية والمودل string للتعامل مع سلاسل النصوص.

3. def generate_password(length)

تعريف التابع الذي المستخدم لتوليد كلمات المرور، والذي يأخذ بارامتر وحيد هو طول السلسلة المراد توليدها.

4. characters = string.ascii_letters + string.digits + string.punctuation

يتم إنشاء متغير characters الذي يحتوي على جميع الأحرف الأبجدية (كبيرة وصغيرة)، والأرقام، والرموز.

5. password = ''.join(random.choice(characters) for _ in range(length))

تم استخدام دالة random.choice لاختيار عشوائي لكل حرف في كلمة المرور من بين الأحرف الموجودة في المتغير characters.

6. if not any(char.isdigit() for char in password):
7. password = password[:random.randint(0, length - 1)] + random.choice(string.digits) + password[random.randint(0, length - 1):]

يتحقق الشرط هنا مما إذا كانت كلمة المرور لا تحتوي على أي أرقام، إذا كانت كلمة المرور لا تحتوي على أي أرقام، يتم إضافة رقم عشوائي إلى كلمة المرور عن طريق قطعها إلى نصفين وإضافة الرقم العشوائي في الوسط.

return password

8. length = int(input("Enter the desired password length: "))

إعادة كلمة المرور المولدة.

9. password = generate_password(length)

10. print(f"Your password is: {password}")