



الجمهورية العربية السورية

وزارة التعليم العالي والبحث العلمي

جامعة تشرين

كلية الهندسة الميكانيكية والكهربائية

هندسة الاتصالات والالكترونيات

السنة الخامسة - مقرر برمجة الشبكات

وظيفة مقرر برمجة الشبكات الثانية

إعداد الطالب:

حمزة عبد الله - علي عيسى

إشراف:

د.م. مهند عيسى

# Second Network Programming Homework

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

```
1  import socket
2  import threading
3
4  HOST = '0.0.0.0'
5  PORT = 8080
6
7  accounts = {
8      "1A": 1000,
9      "2A": 500,
10     "3A": 1500,
11     "1B": 2000,
12     "2B": 1500,
13     "3B": 2500
14 }
15
16
17 def handle_client(conn, addr):
18     print(f"Connected by {addr}")
19     while True:
20         data = conn.recv(1024).decode()
21         if not data:
22             break
23
24         account_number, operation, amount = data.split()
25         if account_number not in accounts:
26             conn.sendall("Invalid account number".encode())
27             continue
28
29         try:
30             amount = float(amount)
31         except ValueError:
32             conn.sendall("Invalid amount".encode())
33             continue
34
35         if operation == "check_balance":
36             balance = accounts[account_number]
37             conn.sendall(f"Your balance is: {balance}".encode())
38         elif operation == "deposit":
39             accounts[account_number] += amount
40             conn.sendall(f"Deposit successful. New balance: {accounts[account_number]}".encode())
41         elif operation == "withdraw":
42             if accounts[account_number] < amount:
43                 conn.sendall("Insufficient funds".encode())
44             else:
45                 accounts[account_number] -= amount
46                 conn.sendall(f"Withdrawal successful. New balance: {accounts[account_number]}".encode())
47         else:
48             conn.sendall("Invalid operation".encode())
49
50     conn.close()
51     print(f"Client {addr} disconnected")
52
53 with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
54     s.bind((HOST, PORT))
55     s.listen()
56     print(f"Server listening on {HOST}:{PORT}")
57     while True:
58         conn, addr = s.accept()
59         thread = threading.Thread(target=handle_client, args=(conn, addr))
60         thread.start()
61
```

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")

```

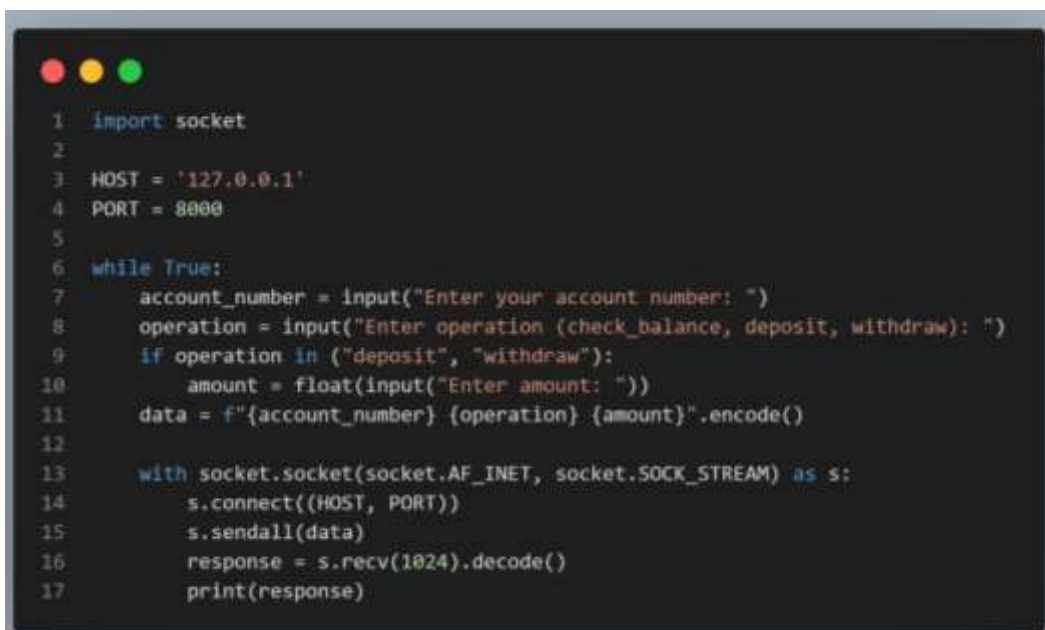
بإنشاء الأوبجيكت (الغرض) thread أستطيع جعل السيرفر يتعامل مع المستخدمين بنفس الوقت بإمرار التابع target=handle\_client ومن ثم .args=(conn, addr)

تشغيل السيرفر:

python tcp-bank-server.py

Server listening on 0.0.0.0:8000

برنامج العميل الأول:



```

1  import socket
2
3  HOST = '127.0.0.1'
4  PORT = 8000
5
6  while True:
7      account_number = input("Enter your account number: ")
8      operation = input("Enter operation (check_balance, deposit, withdraw): ")
9      if operation in ("deposit", "withdraw"):
10         amount = float(input("Enter amount: "))
11         data = f"{account_number} {operation} {amount}".encode()
12
13         with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
14             s.connect((HOST, PORT))
15             s.sendall(data)
16             response = s.recv(1024).decode()
17             print(response)

```

عند تشغيله:

python tcp-bank-client1.py

Enter your account number: 1A

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

Enter amount: 2000

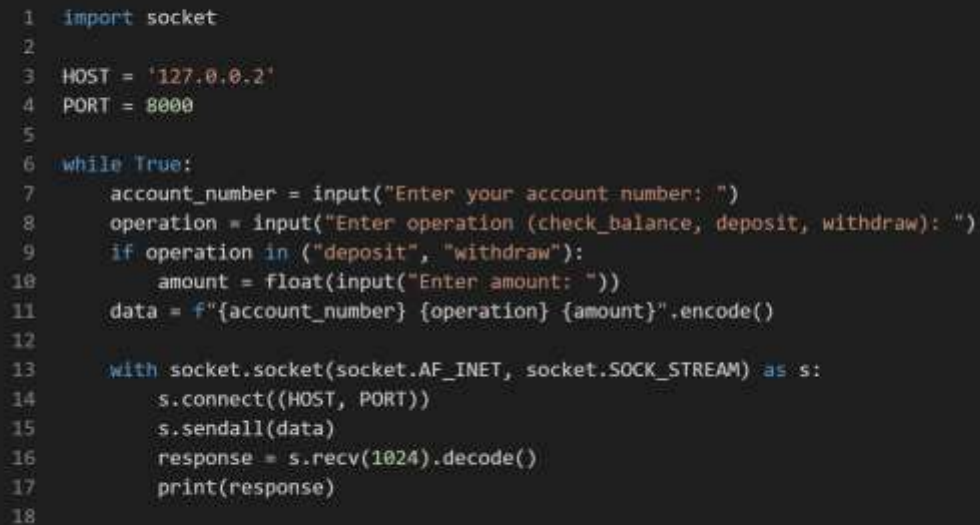
Deposit successful. New balance: 3000.0

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

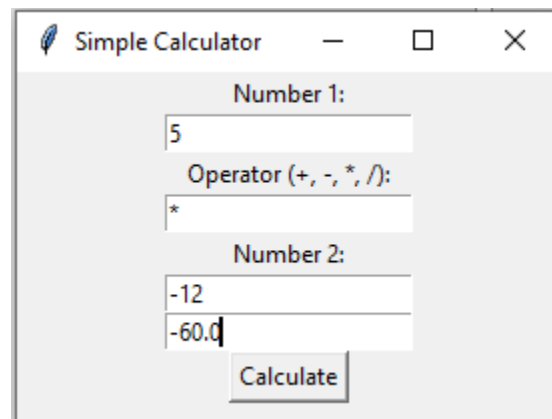
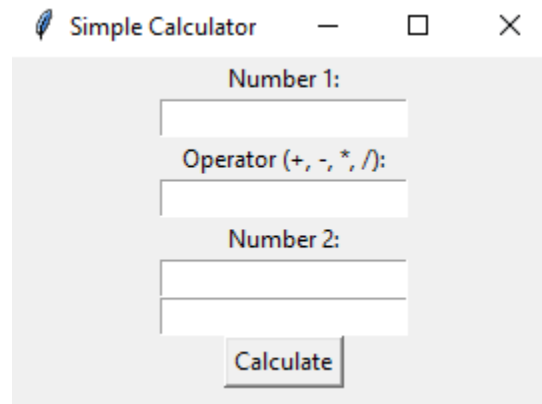


```
1 import socket
2
3 HOST = '127.0.0.2'
4 PORT = 8000
5
6 while True:
7     account_number = input("Enter your account number: ")
8     operation = input("Enter operation (check_balance, deposit, withdraw): ")
9     if operation in ("deposit", "withdraw"):
10         amount = float(input("Enter amount: "))
11         data = f"{account_number} {operation} {amount}".encode()
12
13     with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
14         s.connect((HOST, PORT))
15         s.sendall(data)
16         response = s.recv(1024).decode()
17         print(response)
18
```

## Question 2: Calculator App Using Tkinter

```
1 import tkinter as tk
2
3 def calculate():
4     try:
5         num1 = float(entry1.get())
6         num2 = float(entry2.get())
7         operator = entry_operator.get()
8
9         if operator == '+':
10             result = num1 + num2
11         elif operator == '-':
12             result = num1 - num2
13         elif operator == '*':
14             result = num1 * num2
15         elif operator == '/':
16             if num2 == 0:
17                 result = 'Error! Division by zero.'
18             else:
19                 result = num1 / num2
20         else:
21             result = 'Invalid operator'
22
23         entry_result.delete(0, tk.END)
24         entry_result.insert(0, result)
25     except ValueError:
26         entry_result.delete(0, tk.END)
27         entry_result.insert(0, "Error! Invalid input")
28
29 root = tk.Tk()
30 root.title("Simple Calculator")
31
32 label1 = tk.Label(root, text="Number 1:")
33 label1.pack()
34 entry1 = tk.Entry(root)
35 entry1.pack()
36
37 label_operator = tk.Label(root, text="Operator (+, -, *, /):")
38 label_operator.pack()
39 entry_operator = tk.Entry(root)
40 entry_operator.pack()
41
42 label2 = tk.Label(root, text="Number 2:")
43 label2.pack()
44 entry2 = tk.Entry(root)
45 entry2.pack()
46
47 entry_result = tk.Entry(root)
48 entry_result.pack()
49
50 calculate_button = tk.Button(root, text="Calculate", command=calculate)
51 calculate_button.pack()
52
53 root.mainloop()
```

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 عددين 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 (+, -, *, /):"):`

نشأ 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.