



الجمهورية العربية السورية
وزارة التعليم العالي والبحث العلمي

جامعة تشرين

كلية الهندسة الميكانيكية والكهربائية

قسم هندسة الالكترونيات والاتصالات

السنة الخامسة: وظيفة 2 برمجة شبكات

Name:

زين العابدين وائل عثمان 2884

ناره هيثم الغرير 2906

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.

Requirements:

A. The server should be able to handle multiple client connections concurrently.

- B .The server should maintain a set of pre-defined bank accounts with balances
- C .Each client should connect to the server and authenticate with their account details.
- D. Clients should be able to perform banking operations:
.check balance, deposit money, and withdraw money
- E. The server should keep track of the account balances for each client.
- F. At the end of the session, the server should send the final account balance to each client.

Guidelines:

.Use Python's socket module without third-party packages

-

Implement multi-threading to handle multiple client connections concurrently

-

.Store the account details and balances on the server side

Solution:

في البداية سنقوم بتعريف حسابات البنك:

قمنا بتعريف قاموس accounts لتخزين تفاصيل الحسابات المصرفية المسبقة التعريف. و كل حساب له اسم مستخدم، كلمة مرور، ورصيد.

ثم نتقوم بإنشاء ملقم TCP:

قمنا بإنشاء مقبس TCP باستخدام الدالة `socket.socket()`

تم ربط المقبس بعنوان IP المحلي والمنفذ ٨٠٠٠ باستخدام

`server_socket.bind()`

ثم تم البدء في الاستماع على المقبس للاتصالات الواردة باستخدام

`server_socket.listen(5)`

واستخدمنا دالة `:handle_client()`

هذه الدالة مسؤولة عن التعامل مع كل اتصال من العميل.

وعند استقبال اتصال جديد، يتم إنشاء خيط جديد لهذا العميل باستخدام

`.threading.Thread()`

وفي داخل الدالة `:handle_client()`

يتم أولاً طلب اسم المستخدم وكلمة المرور من العميل للمصادقة.

إذا كانت بيانات تسجيل الدخول صحيحة، يمكن للعميل إجراء العمليات المصرفية (التحقق من الرصيد، الإيداع، السحب).

ويقوم الملقم بتحديث رصيد الحساب وإرسال الرصيد النهائي إلى العميل قبل إغلاق الاتصال.

بالإضافة لدالة `:start_server()`

هذه الدالة هي نقطة الدخول الرئيسية للبرنامج.

تقوم بطباعة رسالة لإعلام المستخدم بأن الملقم قيد التشغيل.

تدخل في حلقة بينما `True`، والتي تستمر في الاستماع والقبول على اتصالات العملاء الواردة.

عند استقبال اتصال جديد، يتم إنشاء خيط جديد باستخدام

`threading.Thread()` وتمرير دالة `handle_client()` كوظيفة

الهدف.

يتم بدء تشغيل الخيط الجديد لمعالجة اتصال العميل.

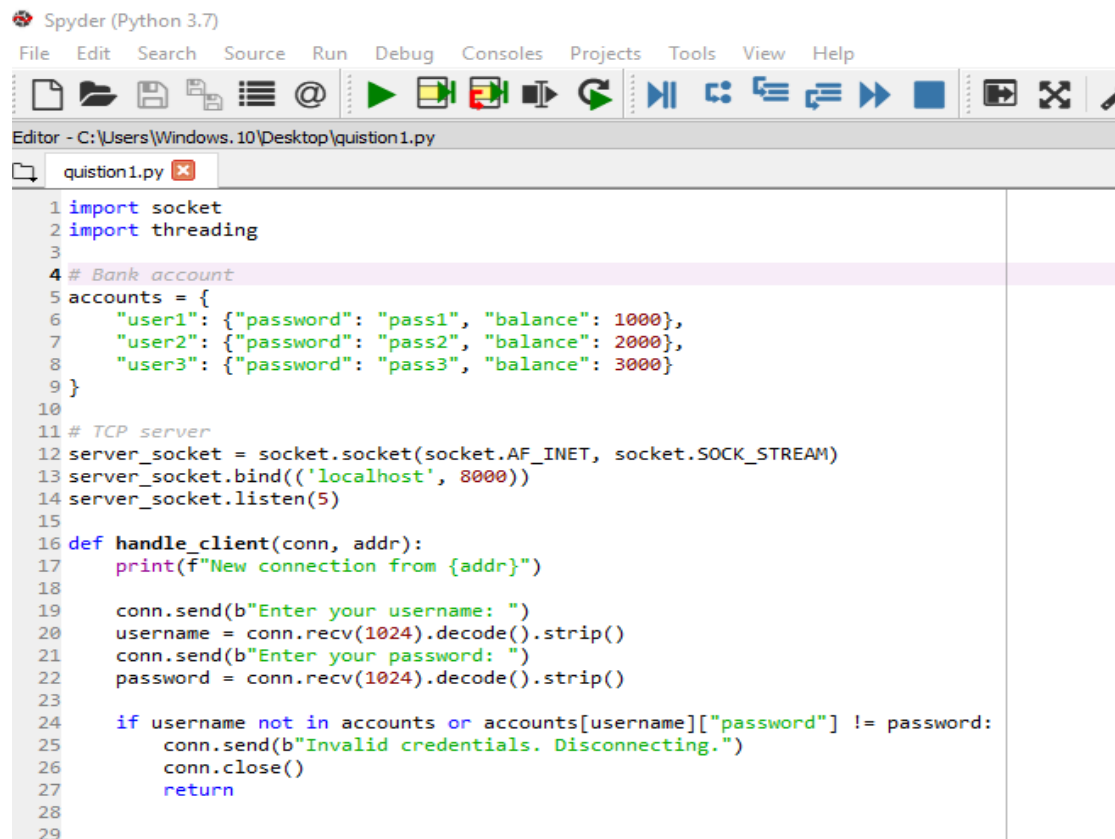
استخدام المكتبة `:socket`

تم استخدام مكتبة Python socket للبرمجة شبكية دون الحاجة إلى أي حزم طرف ثالث.

استخدام التعدد التخاطبي:

تم تنفيذ التعدد التخاطبي باستخدام مكتبة threading لمعالجة اتصالات العملاء المتعددة بشكل متزامن.

إجمالاً، هذا الكود ينفذ تطبيق ATM البنكي باستخدام ملقم TCP وعملاء متعددين. يتعامل الملقم مع كل اتصال عميل في خيط منفصل، ويسمح للعملاء بالمصادقة والقيام بالعمليات المصرفية مع الحفاظ على رصيد الحساب. في نهاية كل جلسة، يتم إرسال الرصيد النهائي للعميل.



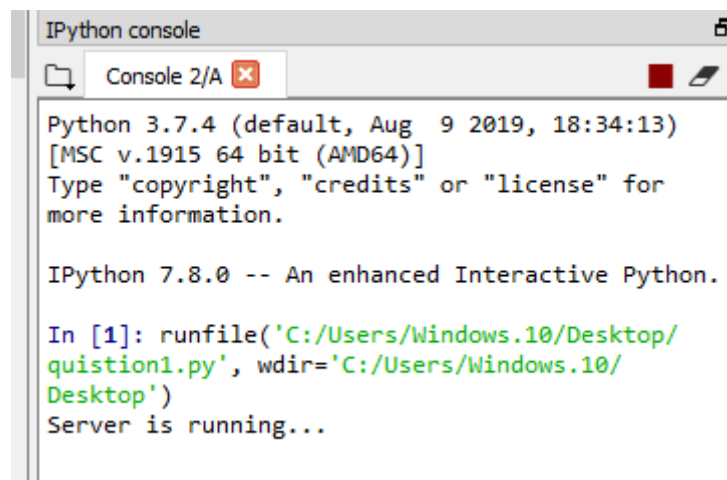
```
1 import socket
2 import threading
3
4 # Bank account
5 accounts = {
6     "user1": {"password": "pass1", "balance": 1000},
7     "user2": {"password": "pass2", "balance": 2000},
8     "user3": {"password": "pass3", "balance": 3000}
9 }
10
11 # TCP server
12 server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
13 server_socket.bind(('localhost', 8000))
14 server_socket.listen(5)
15
16 def handle_client(conn, addr):
17     print(f"New connection from {addr}")
18
19     conn.send(b"Enter your username: ")
20     username = conn.recv(1024).decode().strip()
21     conn.send(b"Enter your password: ")
22     password = conn.recv(1024).decode().strip()
23
24     if username not in accounts or accounts[username]["password"] != password:
25         conn.send(b"Invalid credentials. Disconnecting.")
26         conn.close()
27         return
28
29
```

```

29
30 while True:
31     conn.send(b"Enter your choice (balance/deposit/withdraw/exit): ")
32     choice = conn.recv(1024).decode().strip()
33
34     if choice == "balance":
35         balance = accounts[username]["balance"]
36         conn.send(f"Your current balance is: {balance}".encode())
37     elif choice == "deposit":
38         conn.send(b"Enter the amount to deposit: ")
39         amount = int(conn.recv(1024).decode().strip())
40         accounts[username]["balance"] += amount
41         conn.send(f"Deposit successful. New balance: {accounts[username]['balance']}".encode())
42     elif choice == "withdraw":
43         conn.send(b"Enter the amount to withdraw: ")
44         amount = int(conn.recv(1024).decode().strip())
45         if amount > accounts[username]["balance"]:
46             conn.send(b"Insufficient funds.")
47         else:
48             accounts[username]["balance"] -= amount
49             conn.send(f"Withdrawal successful. New balance: {accounts[username]['balance']}".encode())
50     elif choice == "exit":
51         conn.send(f"Final balance: {accounts[username]['balance']}".encode())
52         conn.close()
53         print(f"Connection with {addr} closed")
54         return
55     else:
56         conn.send(b"Invalid choice. Please try again.")
57
58 def start_server():
59     print("Server is running...")
60     while True:
61         conn, addr = server_socket.accept()
62         thread = threading.Thread(target=handle_client, args=(conn, addr))
63         thread.start()
64
65 start_server()

```

تشغل الكود:



```

IPython console
Console 2/A
Python 3.7.4 (default, Aug 9 2019, 18:34:13)
[MSC v.1915 64 bit (AMD64)]
Type "copyright", "credits" or "license" for
more information.

IPython 7.8.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/Windows.10/Desktop/
quistion1.py', wdir='C:/Users/Windows.10/
Desktop')
Server is running...

```

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

Requirements:

G. Set up a local web server using XAMPP, IIS, or Python's built-in server (using Flask).

H. Apply CSS and Bootstrap to style the website and make it visually appealing.

I. Ensure that the website is responsive and . displays correctly on different screen sizes.

j. Implement basic server-side functionality using Flask to handle website features.

Solution:

app.py

الاستيرادات:

يبدأ الكود بتحميل الوحدات الضرورية من إطار عمل Flask: `render_template`، و `request`.

إنشاء تطبيق Flask:

السطر `Flask(__name__)` ينشئ مثيل جديد لتطبيق Flask، مع إرسال اسم الوحدة الحالية (`__name__`) كحجة.

ثم تعريفات المسارات:

يُعرف الكود ثلاثة مسارات باستخدام المزين `@app.route()`:

`"/`: هو مسار الصفحة الرئيسية، والذي يقوم بتقديم قالب `index.html`.

`/about`: هو المسار يقدم قالب `about.html`.

`/contact`: هو المسار يتعامل مع الطلبات `GET` و `POST`. عندما يتم إجراء طلب `GET`، يقدم قالب `contact.html`. وعندما يتم إجراء طلب `POST`، يتعامل مع إرسال النموذج ثم إعادة توجيه المستخدم إلى قالب `contact.html` مع رسالة نجاح.

معالجة النماذج:

في المسار `/contact`، يتحقق الكود من طريقة الطلب باستخدام `request.method`. إذا كانت الطريقة `'POST'`، فإنه يستخرج بيانات النموذج (الاسم والبريد الإلكتروني والرسالة) من قاموس `request.form` ويقوم بإجراء بعض الإجراءات مع البيانات (على سبيل المثال، إرسال بريد إلكتروني). ثم، يقدم قالب `contact.html` مع تعيين متغير `success` إلى `True`، مما يشير إلى نجاح إرسال النموذج.

تشغيل التطبيق:

السطر النهائي `__if __name__ == '__main__':` يضمن أن خادم التطوير سيتم تشغيله فقط عندما يتم تشغيل السكريبت مباشرة (وليس عندما يتم استيراده كوحدة).

السطر `app.run(debug=True)` يبدأ تشغيل خادم Flask للتطوير في وضع التصحيح، مما يوفر رسائل خطأ مفيدة وإعادة تحميل تلقائية للتطبيق عند إجراء تغييرات.

قوالب HTML

تتحمل قوالب HTML في دليل `templates` مسؤولية عرض الصفحات المختلفة للموقع.

:base.html

هو القالب الأساسي الذي يحدد البنية الكلية والتخطيط للموقع. يتضمن قسم `<HTML>` `<head>`، والذي يقوم بتحميل ملف CSS الخاص بـ Bootstrap والملف `styles.css` المخصص. كما يتضمن رأس مع قائمة تنقل وتذييل.

ويُعد القسم `{% block content %}` المكان الذي سيتم وضع محتوى كل صفحة فيه.

:index.html

هذه هي الصفحة الرئيسية للموقع.

تمتد هذا القالب من `base.html` باستخدام التوجيه `{% extends 'base.html' %}`.

ويتم وضع محتوى الصفحة الرئيسية داخل القسم { % block content % }.

:about.html

هي صفحة "حول" للموقع.

مشابهة لـ index.html، فهي تمتد من قالب base.html وتضيف محتوى صفحة "حول".

:contact.html

هي صفحة الاتصال للموقع.

تمتد من قالب base.html وتتضمن نموذج اتصال.

وإذا تم تعيين المتغير success إلى True (عند إرسال النموذج بنجاح)، فإنه يعرض رسالة نجاح.

Bootstrap و CSS

:styles.css

الملف يحتوي على أنماط CSS المخصصة للموقع.

يتضمن أنماطًا للتخطيط والطباعة والألوان وأي عناصر تصميم مخصصة أخرى.

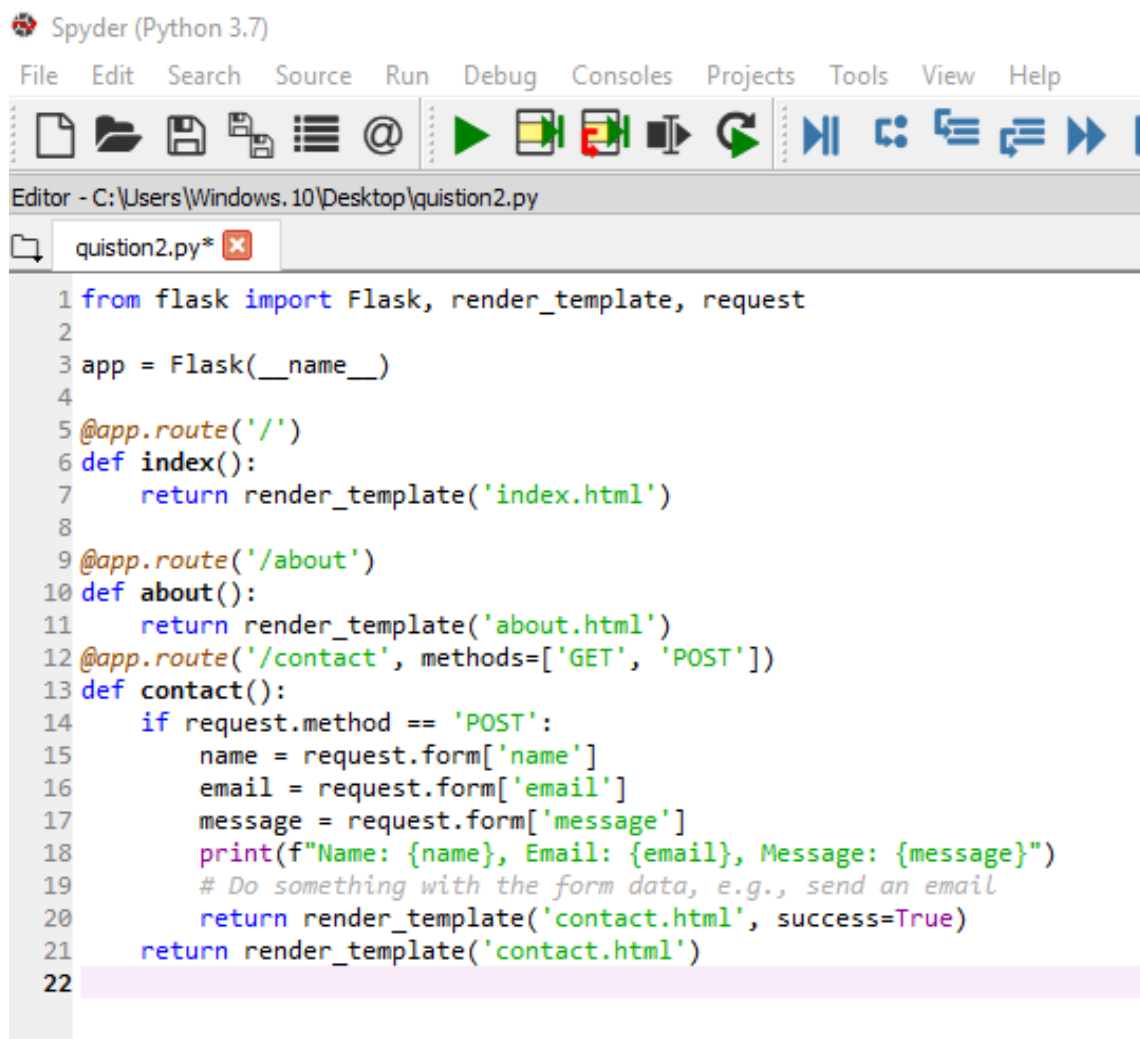
:Bootstrap

يستخدم الموقع أيضًا إطار عمل CSS Bootstrap للتصميم المستجيب وعناصر البناء المجهزة مسبقًا.

يتضمن قالب base.html ملف CSS الخاص بـ Bootstrap، والذي يتم تحميله من CDN.

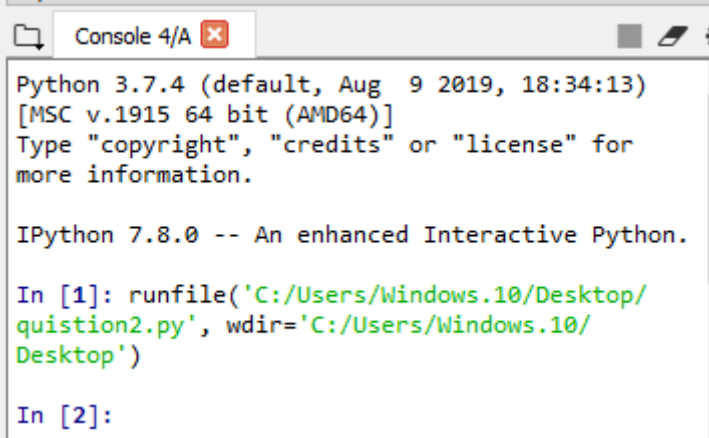
تستخدم قوالب HTML فئات وعناصر Bootstrap، مثل نظام الشبكة والتنقل والعناصر النموذجية.

يسمح مزيج تطبيق Flask وقوالب HTML وملفات CSS وBootstrap بإنشاء موقع بسيط ومستجيب وجذاب بصريًا.



```
1 from flask import Flask, render_template, request
2
3 app = Flask(__name__)
4
5 @app.route('/')
6 def index():
7     return render_template('index.html')
8
9 @app.route('/about')
10 def about():
11     return render_template('about.html')
12 @app.route('/contact', methods=['GET', 'POST'])
13 def contact():
14     if request.method == 'POST':
15         name = request.form['name']
16         email = request.form['email']
17         message = request.form['message']
18         print(f"Name: {name}, Email: {email}, Message: {message}")
19         # Do something with the form data, e.g., send an email
20         return render_template('contact.html', success=True)
21     return render_template('contact.html')
22
```

تنفيذ الكود:



```
Python 3.7.4 (default, Aug  9 2019, 18:34:13)
[MSC v.1915 64 bit (AMD64)]
Type "copyright", "credits" or "license" for
more information.

IPython 7.8.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/Windows.10/Desktop/
quistion2.py', wdir='C:/Users/Windows.10/
Desktop')

In [2]:
```