Names: بشرى اسعد عباس + جميل سليمان طوشان

Homework Number: Homework 2

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

برنامج السيرفر

1. تعريف الحسابات:
   - يحتوي المتغير من نوع dictionary: accountsعلى معلومات الحسابات البنكية، حيث يكون رقم الحساب هو المفتاح. يحتوي كل حساب على رقم التعريف الشخصي (PIN) والرصيد.
2. تابع handle_client
   - يتعامل هذا التابع مع الاتصالات الواردة من العملاء.
   - يرسل رسالة ترحيبية ويطلب من العميل إدخال رقم الحساب.
   - يستلم رقم الحساب ويقوم بالتحقق من صحته. إذا كان غير صالح، يغلق الاتصال.
   - يطلب رقم التعريف الشخصي (PIN) ويقوم بالتحقق من صحته. إذا كان غير صحيح، يغلق الاتصال.
   - يخطر العميل بنجاح التحقق وينتقل إلى التابع handle_transactionsللتعامل مع المعاملات.
   - يغلق الاتصال في حالة حدوث خطأ أو عند انتهاء المعاملات.
3. تابع handle_transactions
   - يتعامل هذا التابع مع المعاملات البنكية المختلفة للعميل.
   - يعرض الخيارات المتاحة ويستلم الخيار من العميل.
   - ينفذ العمليات بناءً على الخيار المحدد:
     - التحقق من الرصيد: يعرض الرصيد الحالي للعميل.
     - الإيداع: يطلب مبلغ الإيداع ويضيفه إلى رصيد الحساب.
     - السحب: يطلب مبلغ السحب ويتحقق من توفر الرصيد الكافي قبل الخصم.
     - الخروج: يعرض الرصيد النهائي ويغلق الاتصال.
     - في حالة خيار غير صالح: يطلب من العميل المحاولة مرة أخرى بخيار صحيح.
4. تابع client_handler
   - ينشئ ثريد جديد لكل عميل ويتصل بالتابع handle_client.
5. إعداد وتشغيل السيرفر:
   - إنشاء سوكيت وتعيين عنوان IP ومنفذ للاستماع للاتصالات الواردة.
   - تعيين السيرفر للاستماع للاتصالات الواردة على المنفذ المحدد.

- في حلقة لا نهائية، يقبل الاتصالات الجديدة ويستخدم التابع client_handler لمعالجة كل اتصال في ثريد جديد.

```
import socket
import threading

accounts = {
    '1234': {'pin': '1234', 'balance': 1000.0},
    '12345': {'pin': '12345', 'balance': 2000.0},
}

def handle_client(client_socket, accounts):
    try:
        client_socket.sendall(b'Welcome to the Bank ATM!\n')
        client_socket.sendall(b'Enter account number: ')
        account_number = client_socket.recv(1024).decode().strip()

        if account_number not in accounts:
            client_socket.sendall(b'Invalid account number.\n')
            client_socket.close()
            return

        client_socket.sendall(b'Enter PIN: ')
        pin = client_socket.recv(1024).decode().strip()

        if accounts[account_number]['pin'] != pin:
            client_socket.sendall(b'Invalid PIN.\n')
            client_socket.close()
            return

        client_socket.sendall(b'Authenticated successfully.\n')
        handle_transactions(client_socket, account_number, accounts)
    except Exception as e:
        print(f"Error: {e}")
    finally:
        client_socket.close()

def handle_transactions(client_socket, account_number, accounts):
    while True:
        client_socket.sendall(b'\nChoose an option:\n1. Check Balance\n2.
Deposit\n3. Withdraw\n4. Exit\n')
        option = client_socket.recv(1024).decode().strip()

        if option == '1':  # Check balance
            balance = accounts[account_number]['balance']
            client_socket.sendall(f'Your balance is: ${balance}\n'.encode())

        elif option == '2':  # Deposit
            client_socket.sendall(b'Enter amount to deposit: ')
            amount = float(client_socket.recv(1024).decode().strip())
            accounts[account_number]['balance'] += amount
            client_socket.sendall(f'Successfully deposited ${amount}. Your
new balance is ${accounts[account_number]["balance"]}\n'.encode())

        elif option == '3':  # Withdraw
            client_socket.sendall(b'Enter amount to withdraw: ')
```

```
            amount = float(client_socket.recv(1024).decode().strip())
            if amount > accounts[account_number]['balance']:
                client_socket.sendall(b'Insufficient funds.\n')
            else:
                accounts[account_number]['balance'] -= amount
                client_socket.sendall(f'Successfully withdrew ${amount}. Your
new balance is ${accounts[account_number]["balance"]}\n'.encode())

        elif option == '4':  # Exit
            client_socket.sendall(f'Your final balance is
${accounts[account_number]["balance"]}\n'.encode())
            break

        else:
            client_socket.sendall(b'Invalid option. Please try again.\n')

def client_handler(client_socket, accounts):
    thread = threading.Thread(target=handle_client, args=(client_socket,
accounts))
    thread.start()

server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind(('0.0.0.0', 12345))
server_socket.listen(5)
print('Server is listening on port 12345...')

while True:
    client_socket, addr = server_socket.accept()
    print(f'Connection from {addr}')
    client_handler(client_socket, accounts)
```

برنامج العميل

```
import socket

client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect(('127.0.0.1', 12345))

try:
    while True:
        data = client_socket.recv(1024).decode()
        print(data)

        if 'Choose an option' in data:
            option = input('Your choice: ')
            client_socket.send(option.encode())
        elif 'Enter' in data:
            value = input()
            client_socket.send(value.encode())
        elif 'Invalid' in data or 'Authenticated' in data or 'Your balance'
```

```
in data or 'Successfully' in data or 'Insufficient' in data:
            # Print the message received and continue
            continue
        elif 'final balance' in data:
            break
except Exception as e:
    print(f"Error: {e}")
finally:
    client_socket.close()
```

شرح الكود:

- الاتصال بالسيرفر:
  - client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM): إنشاء سوكيت جديد.
  - client_socket.connect(('127.0.0.1', 12345)): الاتصال بالسيرفر على العنوان 127.0.0.1 والمنفذ 12345.
- تلقي وإرسال البيانات:
  - data = client_socket.recv(1024).decode(): تلقي البيانات من السيرفر بحد أقصى 1024 بايت.
  - print(data): طباعة البيانات المستلمة من السيرفر.
- التفاعل مع السيرفر بناءً على البيانات المستلمة:
  - إذا كانت البيانات تحتوي على "Choose an option" ، ينتظر المدخلات من المستخدم ويرسل الخيار المحدد للسيرفر.
  - إذا كانت البيانات تحتوي على "Enter" ، ينتظر المدخلات من المستخدم ويرسل القيمة المدخلة للسيرفر.
  - إذا كانت البيانات تحتوي على أي من الرسائل "Invalid" ، "Authenticated"، "Your balance"، "Successfully"، أو "Insufficient" ، يعرض الرسالة ويستمر في الحلقة.
  - إذا كانت البيانات تحتوي على "final balance" ، ينهي الاتصال ويخرج من الحلقة.
- معالجة الاستثناءات:
  - إذا حدث أي خطأ أثناء الاتصال، يتم طباعته ويتم إغلاق السوكيت في النهاية.

الخرج:

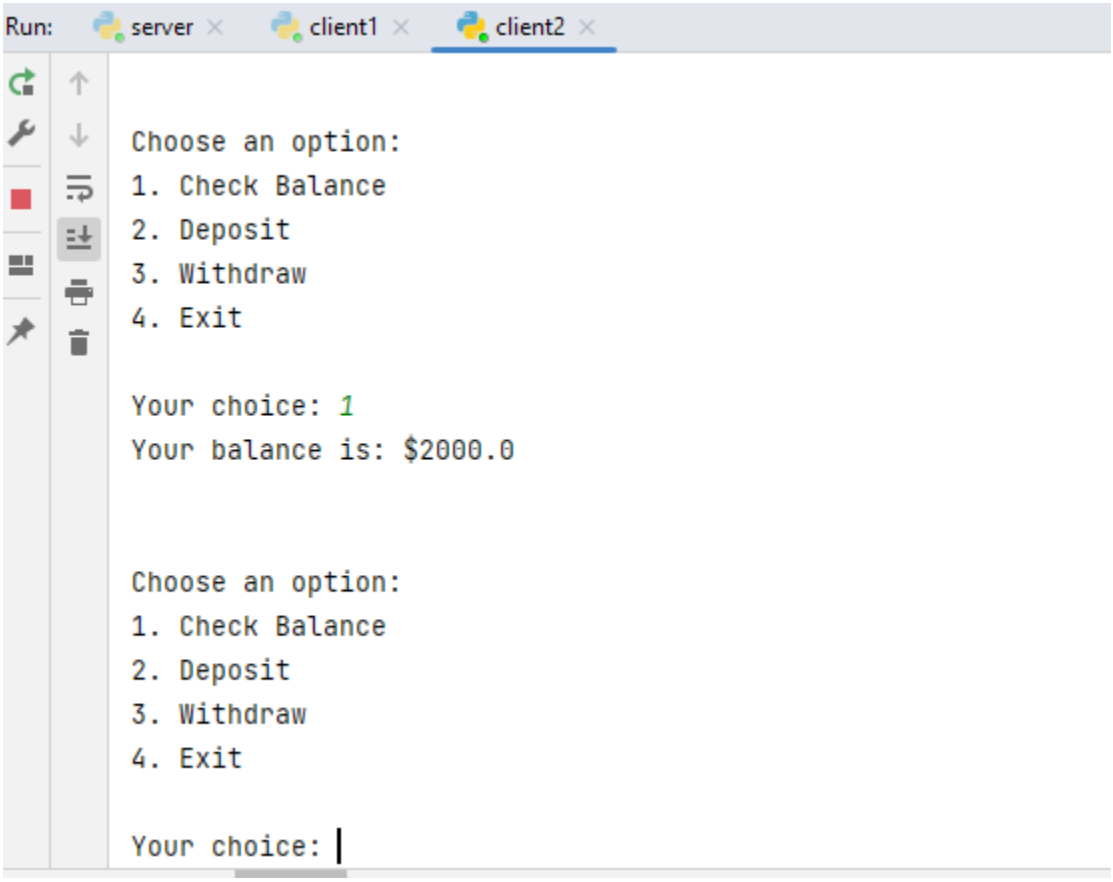
السيرفر مع اتصال عميلين بنفس الوقت

```
Server is listening on port 12345...
Connection from ('127.0.0.1', 52331)
Connection from ('127.0.0.1', 52367)
```

العميل الأول

```
Run: server  client1  client2

Choose an option:
1. Check Balance
2. Deposit
3. Withdraw
4. Exit

Your choice: 1
Your balance is: $1000.0


Choose an option:
1. Check Balance
2. Deposit
3. Withdraw
4. Exit

Your choice:
```

العميل الثاني



Question 2: Simple Website Project with Python Flask Framework

```python
from flask import Flask, render_template
webapp = Flask(__name__)
@webapp.route('/')
def index():
    return render_template('index.html')
@webapp.route('/about')
def about():
    return render_template('about.html')

if __name__ == '__main__':
    webapp.run(debug=True, port=8000)
```

1. يتم استيراد **Flask** و **render_template** من مكتبة **Flask. Flask** هي مكتبة **Python** التي تستخدم لإنشاء تطبيقات الويب، و **render_template**هي تابع يستخدم لعرض صفحات**HTML**
2. يتم إنشاء تطبيق **Flask** باستخدام **Flask(__name__)**يتم تعيين هذا التطبيق إلى المتغير **webapp**
3. يتم تعيين التابع **index()**إلى العنوان **'/'**باستخدام المنسق **@webapp.route('/')**يقوم هذا المنسق بتنفيذ التابع الذي يليه عند ورود العنوان المحدد.
4. يتم استدعاء **render_template('index.html')**في التابع **index()**، الذي يقوم بإرجاع صفحة **HTML** المسماة **index.html**الموجودة في مجلد **tempalates**
5. يتم تعيين التابع **about()**إلى العنوان **'/about'**باستخدام المنسق **@webapp.route('/about')**
6. يتم استدعاء **render_template('about.html')**في التابع **about()**، الذي يقوم بإرجاع صفحة **HTML** المسماة **about.html**الموجودة في مجلد **tempalates**
7. يتم تشغيل التطبيق إذا تم تنفيذ هذا الملف مباشرة **(__name__ == '__main__')** باستخدام **webapp.run()**، والذي يقوم بتشغيل التطبيق على السيرفر المحلي باستخدام **debug mode** و**port 8000**

الصفحة **index.html**

```html
<!DOCTYPE html>
<html lang="ar">
  <head>
    <meta charset="utf-8">
    <title>الصفحة الرئيسية</title>
```

```
    <link href="{{ url_for('static', filename='bootstrap.min.css') }}"
rel="stylesheet">
    <link href="{{ url_for('static', filename='style.css') }}"
rel="stylesheet">
  </head>
  <body>
    <div class="container mt-5">
      <div class="jumbotron">
        <h1 class="display-4">الرئيسية صفحة في بك مرحباً!</h1>
        <p class="lead">طوشان سليمان جميل و عباس اسعد بشرى تقديم</p>
        <hr class="my-4">
        <p>برمجية كورسات يقدم موقعنا</p>
        <a class="btn btn-primary btn-lg" href="{{ url_for('about') }}"
role="button">الموقع عن اكثر لمعلومات</a>
      </div>
      <div class="row">
        <div class="col-sm-6">
          <div class="card">
            <img src="{{ url_for('static', filename='download.jpeg') }}"
class="card-img-top">
            <div class="card-body">
              <h5 class="card-title">طوشان سليمان جميل</h5>
              <p class="card-text">الجامعي الرقم 1393</p>
              <a href="{{ url_for('about') }}" class="btn btn-primary">نبذة
عنا</a>
            </div>
          </div>
        </div>
        <div class="col-sm-6">
          <div class="list-group">
            <a href="#" class="list-group-item list-group-item-action">الكورس
الأول</a>
            <a href="#" class="list-group-item list-group-item-action">الكورس
الثاني</a>
          </div>
        </div>
      </div>
    </div>
  </body>
</html>
```

<!DOCTYPE html>يحدد نوع المستند كـ**HTML.**

<html lang="ar">بداية عنصر **HTML** الذي يحتوي على لغة الصفحة الرئيسية المستخدمة هنا هي اللغة العربية**.**

<head>يحتوي على المعلومات الأساسية للصفحة، مثل عنوان الصفحة والرابط إلى ملفات**CSS.**

<meta charset="utf-8">يحدد ترميز الحروف المستخدم في الصفحة**.**

<title>الصفحة الرئيسية **</title>**يعرض عنوان الصفحة في شريط العنوان في المتصفح**.**

<link href="{{ url_for('static', filename='bootstrap.min.css') }}" rel="stylesheet">يتم استدعاء ملف CSS لـ Bootstrap الموجود في المجلد static

<link href="{{ url_for('static', filename='style.css') }}" rel="stylesheet">يتم استدعاء ملف CSS خاص بالصفحة.

<body>يحتوي على محتوى الصفحة الفعلي.

يعرض الصفحة في عنصر div مع تحديد الفئة container وتغيير المسافة العلوية بواسطة الفئة mt-5

يعرض المحتوى الرئيسي للصفحة في عنصر jumbotron

<h1 class="display-4">مرحباً بك في صفحة الرئيسية </h1>يعرض عنوان المحتوى الرئيسي في الصفحة.

 يعرض المحتوى الفرعي في الصفحة.

<hr class="my-4">يعرض خط فاصل بين المحتويات.

موقعنا يقدم كورسات برمجية يعرض نص فرعي آخر في الصفحة.

<a class="btn btn-primary btn-lg" href="{{ url_for('about') }}" role="button">لمعلومات اكثر عن الموقع </a>يعرض زر لربط الصفحة بصفحة أخرى.

يعرض صفوف لـgrid system

يعرض عناصر في الصفحة على الجانب الأيمن والأيسر.

يعرض عناصر في الصفحة في عنصر بطاقة.

<img src="{{ url_for('static', filename='download.jpeg') }}" class="card-img-top">يعرض صورة في العنصر البطاقة.

<h5 class="card-title"> </h5>يعرض عنوان عنصر البطاقة.

 يعرض نص فرعي آخر في العنصر البطاقة.

<a href="{{ url_for('about') }}" class="btn btn-primary">نبذة عنا :</a>يعرض زر لربط الصفحة بصفحة أخرى.

** **يعرض عناصر في الصفحة في عنصر **list-group**

<a href="#" class="list-group-item list-group-item-action">الكورس الأول **</a>**يعرض رابطاً لعنصر في **list-group**

<a href="#" class="list-group-item list-group-item-action">الكورس الثاني **</a>**يعرض رابطاً لعنصر آخر في **list-group**

ملف **Style.css** يتم وضعه في مجلد **static**.

```
body {
  font-family: "Helvetica Neue", Helvetica, Arial, sans-serif;
}
.btn {
  display: block;
}
.card {
  background-color: #f8f9fa;
}

.card-title, .card-text {
  color: #212529;
}

.list-group-item.active {
  background-color: #007bff;
  border-color: #007bff;
}

.list-group-item.active a {
  color: #fff;
}

.list-group-item-action {
  color: #007bff;
}

h1, h2, h3, h4, h5, h6 {
  color: #007bff;
}
```

body: تحديد الخط الأساسي لنص الصفحة إلى **Helvetica Neue**، **Helvetica**، **Arial**، **sans-serif**.

.btn: تغيير العرض الافتراضي للزر إلى **block**.

.card: تغيير لون خلفية البطاقة إلى **#f8f9fa**.

.**card-title, .card-text**: تحديد لون النص داخل البطاقة إلى #212529.

.**list-group-item.active**: تحديد الحافة الداخلية للعنصر النشط داخل القائمة إلى #**bff**007 وتحديد لون الحدود إلى #**bff**007.

.**list-group-item.active a**: تحديد لون النص في العنصر النشط داخل القائمة إلى #**fff**.

.**list-group-item-action**: تحديد لون الخط الرئيسي للعناصر في القائمة إلى #**bff**007.

h1, h2, h3, h4, h5, h6: تحديد لون الخط الأساسي للعناصر في العناوين إلى #**bff**007.

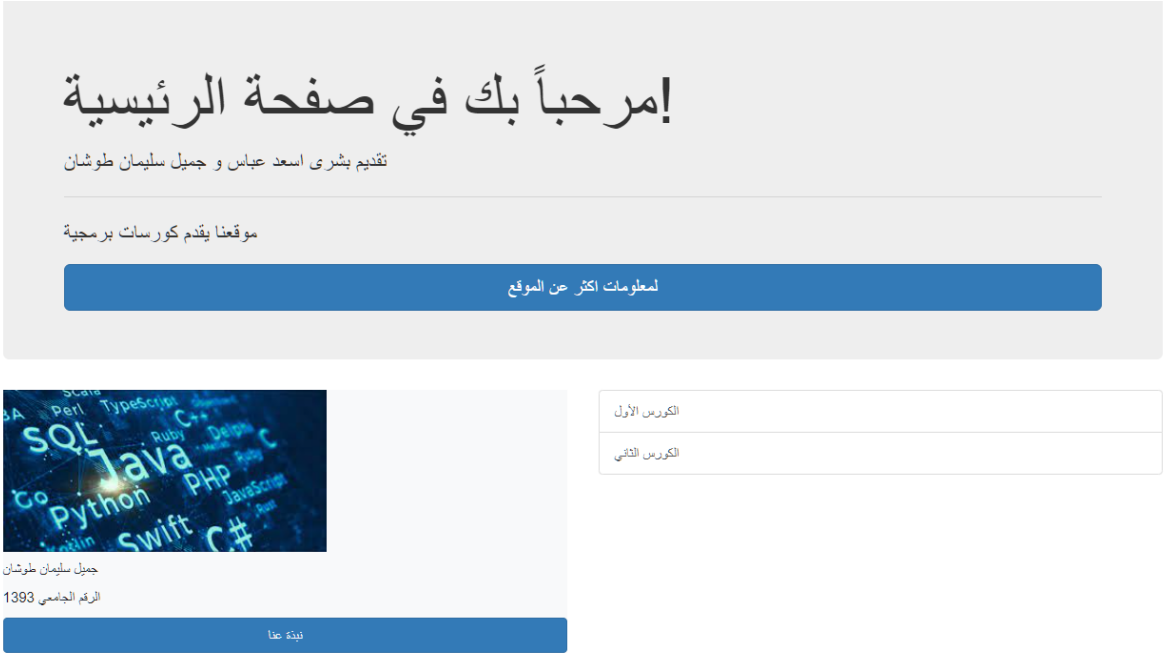
الصفحة **about.html**

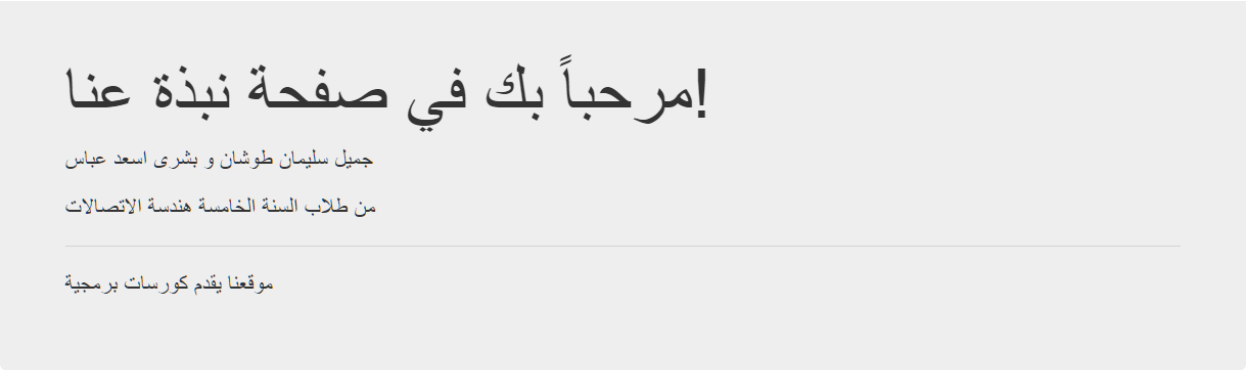
```
<!DOCTYPE html>
<html lang="ar">
  <head>
    <meta charset="utf-8">
    <title>عنا نبذة</title>
    <link href="{{ url_for('static', filename='bootstrap.min.css') }}"
rel="stylesheet">
    <link href="{{ url_for('static', filename='style.css') }}"
rel="stylesheet">
  </head>
  <body>
    <div class="container mt-5">
      <div class="jumbotron">
        <h1 class="display-4">عنا نبذة صفحة في بك مرحباً!</h1>
        <p class="lead">  عباس اسعد بشرى و طوشان سليمان جميل  </p>
        <p class="lead">  الاتصالات هندسة الخامسة السنة طلاب من  </p>
        <hr class="my-4">
        <p>برمجية كورسات يقدم موقعنا</p>

        </div>
      </div>
  </body>
</html>
```

الصفحات في المتصفح:

Index

الرابط: http://127.0.0.1:8000 /



About

الرابط: http://127.0.0.1:8000/about