



بتول علي حسن 2885 + مايا محمد حسن 2955

Q1:

يتم تعريف قاموس يحتوي على الحسابات المبدئية. كل حساب يحتوي على رقم الحساب (account number) كالمفتاح وبيانات الحساب رمز PIN والرصيد كقيم.

التابع `handle_client` يتعامل مع العميل المتصل:

- يطلب من العميل إدخال رقم الحساب ورمز PIN للتحقق.
- إذا كان الحساب صحيحاً، يسمح للعميل باختيار عملية (التحقق من الرصيد، الإيداع، السحب، أو إنهاء الجلسة).
- بناءً على العملية المختارة، يتم تنفيذها وإرسال النتيجة للعميل.
- يتم إغلاق الاتصال في النهاية.

التابع `start_server`:

- يقوم بإنشاء سوكت الخادم باستخدام `socket.AF_INET` لإنشاء اتصال إنترنت و `socket.SOCK_STREAM` لإنشاء اتصال TCP.
- يقوم بربط الخادم على جميع الواجهات (0.0.0.0) وعلى المنفذ 9999.
- يبدأ الاستماع للاتصالات الواردة بحد أقصى 5 اتصالات في رتل الانتظار.
- في حلقة لا نهائية، يقبل اتصالات العملاء وينشئ ثريد جديد لكل عميل باستخدام التابع `handle_client`.

```
import socket
import threading
```

```
# أرصدها مع المبدئية البنك حسابات تعريف
accounts = {
    '123456': {'pin': '1234', 'balance': 1000.0},
    '654321': {'pin': '4321', 'balance': 1500.0}
}
```

```
# متصل عميل كل مع للتعامل تابع
def handle_client(client_socket):
```

```
    try:
        # العميل من PIN ورمز الحساب بيانات استقبال
        client_socket.send(b'Enter account number: ')
        account_number = client_socket.recv(1024).decode().strip()
```

```

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

# الحساب بيانات صحة من التحقق
if account_number in accounts and accounts[account_number]['pin'] ==
pin:
    client_socket.send(b'Authenticated\n')
    while True:
        # للعميل المتاحة الخيارات إرسال
        client_socket.send(b'Choose operation: 1. Check Balance 2.
Deposit 3. Withdraw 4. Exit\n')
        operation = client_socket.recv(1024).decode().strip()

        if operation == '1':
            # الرصيد من تحقق
            balance = accounts[account_number]['balance']
            client_socket.send(f'Your balance is
{balance}\n'.encode())

            elif operation == '2':
                # المال إيداع
                client_socket.send(b'Enter amount to deposit: ')
                amount = float(client_socket.recv(1024).decode().strip())
                accounts[account_number]['balance'] += amount
                client_socket.send(b'Deposit successful\n')

            elif operation == '3':
                # المال سحب
                client_socket.send(b'Enter amount to withdraw: ')
                amount = float(client_socket.recv(1024).decode().strip())
                if accounts[account_number]['balance'] >= amount:
                    accounts[account_number]['balance'] -= amount
                    client_socket.send(b'Withdrawal successful\n')
                else:
                    client_socket.send(b'Insufficient balance\n')

            elif operation == '4':
                # الجلسة إنهاء
                balance = accounts[account_number]['balance']
                client_socket.send(f'Your final balance is
{balance}\n'.encode())
                break
            else:
                client_socket.send(b'Invalid operation\n')

        else:
            client_socket.send(b'Authentication failed\n')
    finally:
        client_socket.close()

# الخادم بدء تابع
def start_server():
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind(('0.0.0.0', 9999))
    server.listen(5)

```

```

print("Server started and listening on port 9999")

while True:
    client_socket, addr = server.accept()
    print(f'Accepted connection from {addr}')
    client_handler = threading.Thread(target=handle_client,
args=(client_socket,))
    client_handler.start()

if __name__ == "__main__":
    start_server()

```

بالنسبة للعميل:

إنشاء **سوكيت**: تم إنشاء سوكيت باستخدام `socket.socket()` لتكوين اتصال TCP.

الاتصال **بالخادم**: باستخدام `client.connect()` للاتصال بالخادم الذي يعمل على 127.0.0.1 على المنفذ 9999.

حلقة الاستقبال والإرسال:

- يستقبل العميل البيانات من الخادم باستخدام `client.recv(1024)`، ثم يقوم بفك تشفير البيانات باستخدام `decode()` ويطبّعها.
- إذا استلم العميل رسالة تحتوي على "Your final balance is"، يتم إغلاق الاتصال باستخدام `client.close()` وكسر الحلقة باستخدام `break`.
- يتم استقبال إدخال المستخدم باستخدام `input()` ثم إرسال البيانات إلى الخادم باستخدام `client.send()` بعد ترميزها باستخدام `encode()`.

إغلاق الاتصال: بعد كسر الحلقة أو الانتهاء من العمليات، يتم إغلاق الاتصال باستخدام `client.close()`.

```

import socket

def start_client():
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect(('127.0.0.1', 9999))

    while True:
        data = client.recv(1024)
        print(data.decode(), end='')

        if 'Your final balance is' in data.decode():
            client.close()
            break

    user_input = input()

```

```

        client.send(user_input.encode())

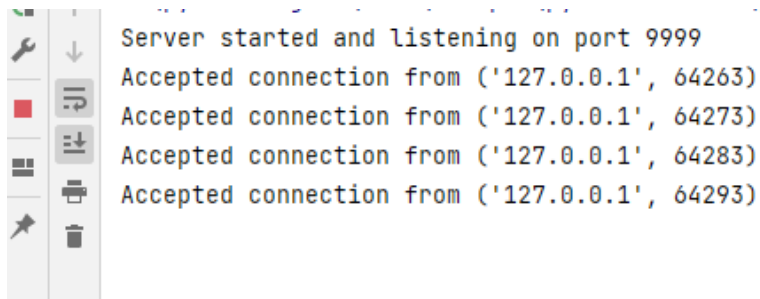
    client.close()

if __name__ == "__main__":
    start_client()

```

الخرج

الخادم يعمل ومتصل به 4 عملاء بنفس الوقت



```

Server started and listening on port 9999
Accepted connection from ('127.0.0.1', 64263)
Accepted connection from ('127.0.0.1', 64273)
Accepted connection from ('127.0.0.1', 64283)
Accepted connection from ('127.0.0.1', 64293)

```

عمليات العميل الأول

```

Enter account number: 123456
Enter PIN: 1234
Authenticated

Choose operation: 1. Check Balance 2. Deposit 3. Withdraw 4. Exit
1
Your balance is 1000.0

Choose operation: 1. Check Balance 2. Deposit 3. Withdraw 4. Exit
2
Enter amount to deposit: 150
Deposit successful

Choose operation: 1. Check Balance 2. Deposit 3. Withdraw 4. Exit
4
Your final balance is 1150.0

Process finished with exit code 0

```

```
Enter account number: 654321
Enter PIN: 4321
Authenticated

Choose operation: 1. Check Balance 2. Deposit 3. Withdraw 4. Exit
3
Enter amount to withdraw: 650
Withdrawal successful

Choose operation: 1. Check Balance 2. Deposit 3. Withdraw 4. Exit
1
Your balance is 850.0

Choose operation: 1. Check Balance 2. Deposit 3. Withdraw 4. Exit
4
Your final balance is 850.0

Process finished with exit code 0
|
```

Q2:

نقوم بإنشاء تطبيق ويب باستخدام Flask يحتوي على عدة صفحات. في البداية يتم استيراد مكتبة Flask والتابع `render_template` الذي يستخدم لتحميل ملفات HTML

```
from flask import Flask, render_template
```

ثم نقوم بإنشاء السيرفر باستخدام الكلاس `Flask`:

```
app = Flask(__name__)
```

نقوم بإنشاء توابيع العرض لكل صفحة بالاستناد إلى عناوين URL محددة بواسطة المنسق `@app.route` عندما يتم طلب عنوان URL محدد يتم استدعاء التابع المرتبط بهذا العنوان.

على سبيل المثال: التابع `index` لعرض الصفحة الرئيسية عندما يتم الوصول إلى عنوان URL الرئيسي `/`.

```
@app.route('/')
def index():
    return render_template('index.html')
```

```
@app.route('/about')
def about():
    return render_template('about.html')
```

```
@app.route('/services')
def services():
```

```

        return render_template('services.html')

@app.route('/contact')
def contact():
    return render_template('contact.html')

```

يتم تشغيل التطبيق باستخدام app.run ويتم تمكين وضع التصحيح (debug mode) باستخدام debug=True لتسهيل عملية تطوير التطبيق وإظهار أي أخطاء تحدث أثناء التطوير.

```

if __name__ == '__main__':
    app.run(debug=True)

```

الصفحة الرئيسية index.html:

```

<!DOCTYPE html>
<html dir="rtl">
<head>
    <style>
    p {
        display: block;
        margin-bottom: 10px;
    }
    </style>
    <meta charset="utf-8">
    <title>question 2</title>
    <link rel="stylesheet" type="text/css" href="{{ url_for('static',
filename='style.css') }}">
</head>
<body>
    <nav>
        <ul>
            <li><a href="{{ url_for('index') }}">Homepage</a></li>
            <li><a href="{{ url_for('about') }}">About</a></li>
            <li><a href="{{ url_for('services') }}">Services</a></li>
            <li><a href="{{ url_for('contact') }}">Contact</a></li>
        </ul>
    </nav>
    <main>
        <div>
            <h1>شركات برمجة - الثانية الوظيفة</h1>

            <p></p>
            <p>الشركات برمجة مادة في لنا ويب تطبيق اول هذا</p>
            <p><a href="{{ url_for('about') }}">هنا</a> انقر عنا اكثر لمعلومات</p>

```

```

        <p> انقر المقرر محتويات لمعرفة <a href="{{ url_for('services')
}}">هنا</a></p>
        <p> انقر الوظيفة ايميل لمعرفة <a href="{{ url_for('contact')
}}">هنا</a></p>
    </div>
</main>
</body>
</html>

```

يتم تحديد اتجاه النص باللغة العربية باستخدام السمة `dir="rtl"` في العنصر `<html>`

يتم تضمين ملف CSS خارجي باستخدام العنصر `<link>` والسمة `href` التي تحدد مسار الملف الخارجي ويتم استخدام `url_for()` لإنشاء روابط.

يتم إنشاء شريط الاختصارات باستخدام العنصر `<nav>` وقائمة غير مرتبة `` وقائمة فرعية `` وروابط `<a>`

المحتوى الرئيسي للصفحة يتم وضعه داخل العنصر `<main>` ويتضمن عنوان `<h1>` وعدة فقرات `<p>`

تم استخدام العنصر `<a>` مع السمة `href` لإنشاء روابط للصفحات الأخرى في الموقع.

تم تعريف ملف CSS داخل العنصر `<style>` ويتضمن خاصية `display: block` و `margin-bottom: 10px` للفقرات `<p>`.

ملف `:style.css`

```

body {
    background-color: pink;
}

main {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
}

nav {
    display: flex;
    justify-content: top;
    align-items: top;
}

```

استخدمنا الخاصية `display: flex` لإنشاء عناصر مرنة داخل العناصر `<main>` و `<nav>` واستخدمنا `justify-content: center` لتوسيط العناصر عمودياً و `align-items: center` لتوسيطهم أفقياً داخل العنصر `<main>`

أيضاً `justify-content: top` و `align-items: top` لتحديد موقع العناصر داخل العنصر `<nav>` بالجزء العلوي منه.

شكل الصفحة الرئيسية في المتصفح:

الوظيفة الثانية - برمجة شبكات

هذا اول تطبيق ويب لنا في مادة برمجة الشبكات

لمعلومات اكثر عنا انقر [هنا](#)

لمعرفة محتويات المقرر انقر [هنا](#)

لمعرفة ايميل الوظيفة انقر [هنا](#)

صفحة about:

```
<!DOCTYPE html>
<html dir="rtl">
<head>
  <meta charset="utf-8">

  <title>معلومات</title>
  <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
  <nav>
    <ul>
      <li><a href="{{ url_for('index') }}">Homepage</a></li>
      <li><a href="{{ url_for('about') }}">About</a></li>
      <li><a href="{{ url_for('services') }}">Services</a></li>
      <li><a href="{{ url_for('contact') }}">Contact</a></li>
    </ul>
  </nav>
  <main>
    <h1>2955 حسن محمد مايا + 2885 حسن علي بتول</h1>
  </main>
</body>
</html>
```


بتول علي حسن 2885 + مايا محمد حسن 2955

صفحة :services

```
<!DOCTYPE html>
<html dir="rtl">
<head>
  <meta charset="utf-8">

  <title>المقرر</title>
  <link rel="stylesheet" href="{{ url_for('static', filename='style.css')
}}">
</head>
<body>
  <nav>
    <ul>
      <li><a href="{{ url_for('index') }}">Home</a></li>
      <li><a href="{{ url_for('about') }}">About</a></li>
      <li><a href="{{ url_for('services') }}">Services</a></li>
      <li><a href="{{ url_for('contact') }}">Contact</a></li>
    </ul>
  </nav>
  <main>
    <h1>يحيوي المقرر</h1>
    <ul>
      <li>البايثون أساسيات</li>
      <li>وتطبيقاته السوكيت مفهوم</li>
      <li>الويب برمجة</li>
    </ul>
  </main>
</body>
</html>
```

المقرر يحوي

- أساسيات البايثون
- مفهوم السوكيت وتطبيقاته
- برمجة الويب

صفحة :contact

```
<!DOCTYPE html>
<html dir="rtl">
<head>
  <meta charset="utf-8">

  <title>التواصل</title>
  <link rel="stylesheet" href="{{ url_for('static', filename='style.css')
  }}">
</head>
<body>
  <nav>
    <ul>
      <li><a href="{{ url_for('index') }}">Home</a></li>
      <li><a href="{{ url_for('about') }}">About</a></li>
      <li><a href="{{ url_for('services') }}">Services</a></li>
      <li><a href="{{ url_for('contact') }}">Contact</a></li>
    </ul>
  </nav>
  <main>
    <h1>الوظيفة ايميل</h1>
    <p>promail0101@gmail.com</p>
  </main>
</body>
</html>
```

- [Home](#) •
- [About](#) •
- [Services](#) •
- [Contact](#) •

ایمیل الوظيفة promail0101@gmail.com