Syrian Arab Republic

Lattakia - Tishreen University

Department of Communication and electrical engineering

5th , Network Programming : Homework No2

الجمهورية العربية السورية

اللاذقية -جامعــة تشريـــن

كلية الهندسة الكهربائية والميكانيكية

قسم هندسة الاتصالات والالكترونيات

السنة الخامسة: وظيفة2 برمجة شبكات

Name:_ Dina Demyati_____, Number:_ 2290,_____, Submitted To GitHub:_ @DinaDemyati_____

Second Network Programming Homework

Question 1: TCP Server/Client Quiz App with Multi-threading?

As an improvement to previous first homework, build a TCP server and client quiz application using Python. The server should handle multiple client connections simultaneously using multi-threading. The application should allow clients to connect, participate in a quiz, and receive their quiz scores upon completion.

Requirements:

A. The server should be able to handle multiple client connections concurrently.
B. The quiz should consist of a set of pre-defined questions stored on the server.
C. Each client should connect to the server and receive the quiz questions.
D. Clients should send their answers to the server.
E. The server should keep track of the scores for each client.
F. At the end of the quiz, the server should send the final scores to each client.

Guidelines:

- Use Python's socket module “don’t use 3thd-party packages”.
- Implement multi-threading to handle multiple client connections concurrently.
- Store the quiz questions and correct answers on the server side.

Notes:

- Write brief report describing the design choices you made and any challenges faced during implementation.
- You can make a TCP Server/Client of your choice, such as Bank ATM, Chat application, or any other appropriate application that fulfil all requirements.

server.py client.py

```python
import socket
import threading

q = {
    "What is the currency of the United States? :\na.Dollar\nb.Euro\nThe answer": "a",
    "What is the currency of Japan? :\na.Dollar\nb.Yen\nThe answer": "b",
    "What is the currency of Germany? :\na.Pound\nb.Euro\nThe answer": "b",
    "What is the currency of Australia? :\na.Dollar\nb.Yen\nThe answer": "a",
    "What is the currency of Brazil? :\na.Dollar\nb.Real\nThe answer": "b",
    "What is the currency of India? :\na.Rupee\nb.Euro\nThe answer": "a",
    "What is the currency of Canada? :\na.Dollar\nb.Euro\nThe answer": "a",
    "What is the currency of France? :\na.Dollar\nb.Euro\nThe answer": "b",
    "What is the currency of China? :\na.Ruble\nb.Yuan\nThe answer": "b",
    "What is the currency of Russia? :\na.Ruble\nb.Euro\nThe answer": "a",
    "What is the currency of South Africa? :\na.Rand\nb.Euro\nThe answer": "a",
    "What is the currency of Saudi Arabia? :\na.Riyal\nb.Dollar\nThe answer": "a",
    "What is the currency of Mexico? :\na.Dollar\nb.Peso\nThe answer": "b",
    "What is the currency of Italy? :\na.Euro\nb.Dollar\nThe answer": "a",
    "What is the currency of Egypt? :\na.Pound\nb.Dollar\nThe answer": "a",
    "What is the currency of Argentina? :\na.Euro\nb.Peso\nThe answer": "b",
    "What is the currency of United Kingdom? :\na.Pound\nb.Dollar\nThe answer": "a",
    "What is the currency of Spain? :\na.Euro\nb.Dollar\nThe answer": "a",
    "What is the currency of South Korea? :\na.Won\nb.Dollar\nThe answer": "a",
    "What is the currency of Nigeria? :\na.Naira\nb.Dollar\nThe answer": "a"
}

client_scores = {}

def handleClient(client_soc, client_address):
    try:
        client_soc.send(str(len(q)).encode())

        for question in q:
            client_soc.send(question.encode())

            client_ans = client_soc.recv(1024).decode().strip()

            if client_ans.lower() == q[question].lower():
                client_scores[client_address] = client_scores.get(client_address, 0) + 1
```

Name:_ Dina Demyati_____, Number:_ 2290,_____, Submitted To GitHub:_ @DinaDemyati_____

```
            score = client_scores.get(client_address, 0)
            client_soc.send(f"Score: {score}/{len(q)}\n".encode())

        except ConnectionAbortedError:
            print(f"Connection the client: {client_address}")

        client_soc.close()
        print(f"Disconnected client: {client_address}")

    def RUN_server():
        server_soc = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

        server_address = ('localhost', 3333)
        server_soc.bind(server_address)

        server_soc.listen(5)
        print("Server started.....")

        while True:
            client_socket, client_address = server_soc.accept()
            print(f"Connected to {client_address}")

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

    if __name__ == '__main__':
        RUN_server()
```

يتم استيراد وحدات socket و threading لإنشاء سيرفر متعدد الخيوط.

تعريف قاموس q يحتوي على أسئلة الاختبار وإجاباتها الصحيحة.

تعريف قاموس client_scores لتخزين نتائج العملاء.

تعريف الدالة handleClient التي تقوم بمعالجة اتصال العميل.

داخل الدالة، يتم إرسال عدد الأسئلة إلى العميل ومن ثم يتم تنفيذ الاختبار.

بعد الاختبار، يتم إرسال نتيجة العميل إليه.

تعريف الدالة RUN_server التي تقوم بإعداد وتشغيل السيرفر.

يتم ربط السيرفر بعنوان ومنفذ معين.(localhost: 3333)

يتم استماع السيرفر للاقترانات وتشغيل حلقة لا نهائية لقبول العملاء الجدد.

عند توصيل العميل، يتم إنشاء خيط جديد للتعامل معه.

يتم تشغيل السيرفر باستخدام الدالة RUN_server() عند تشغيل البرنامج.

عند تشغيل الكود، ستظهر رسالة"Server started..." ، والسيرفر سيكون قيد الاستماع على المنفذ 3333 على جهاز localhost.
عند توصيل العملاء، سيتم إرسال اختبار الأسئلة إليهم واستلام إجاباتهم، ثم يتم إرسال النتيجة إلى العميل.

Name:_ Dina Demyati_____, Number:_ 2290,_____, Submitted To GitHub:_ @DinaDemyati____

server.py client.py

client.py > ...

```python
import socket

def start_client():
    server_host = 'localhost'
    server_port = 3333

    client_soc = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    try:
        client_soc.connect((server_host, server_port))
        print(f"Connected to server {server_host}:{server_port}")

        num_questions = int(client_soc.recv(1024).decode())

        for _ in range(num_questions):
            question = client_soc.recv(1024).decode()

            answer = input(f"{question}: ")

            client_soc.sendall(answer.encode())

        final_score = client_soc.recv(1024).decode()
        print(f" score: {final_score}")

    except ConnectionRefusedError:
        print("Failed .")
    finally:
        client_soc.close()

if __name__ == '__main__':
    start_client()

```

يتم استيراد وحدة socket.

تعريف الدالة start_client التي تقوم بتنفيذ عملية الاتصال بالسيرفر.

داخل الدالة، يتم إعداد عنوان السيرفر ومنفذه.

يتم إنشاء مأخذ العميل باستخدام socket.socket وتهيئته للاتصال بالسيرفر.

يتم محاولة الاتصال بالسيرفر باستخدام client_soc.connect((server_host, server_port)).

إذا تم الاتصال بنجاح، سيتم استلام عدد الأسئلة من السيرفر باستخدام client_soc.recv(1024).decode() وتحويله إلى عدد صحيح.

يتم تنفيذ حلقة تكرار للحصول على الأسئلة من السيرفر وإرسال إجاباتها.

يتم استقبال النتيجة النهائية من السيرفر باستخدام client_soc.recv(1024).decode() وطباعتها.

إذا فشل الاتصال بالسيرفر، سيتم طباعة رسالة "Failed.".

أخيرًا، يتم إغلاق مأخذ العميل باستخدام client_soc.close().

عند تشغيل البرنامج، سيتم إعداد العميل للاتصال بسيرفر على جهاز localhost على المنفذ ٣٣٣٣. سيتم استلام الأسئلة من السيرفر وإرسال إجاباتها. ثم سيتم استقبال النتيجة النهائية من السيرفر وطباعتها. إذا فشل الاتصال بالسيرفر، ستظهر رسالة "Failed.".

Name:_ Dina Demyati_____, Number:_ 2290,_____, Submitted To GitHub:_ @DinaDemyati_____

```
Server listening on localhost:8888
Connected client: ('127.0.0.1', 51798)
Connected client: ('127.0.0.1', 51799)
Disconnected client: ('127.0.0.1', 51798)
Disconnected client: ('127.0.0.1', 51799)
```

```
The answer: a
What is the currency of Egypt? :
a.Pound
b.Dollar
The answer: a
What is the currency of Argentina? :
a.Euro
b.Peso
The answer: a
What is the currency of United Kingdom? :
a.Pound
b.Dollar
The answer: a
What is the currency of Spain? :
a.Euro
b.Dollar
The answer: a
What is the currency of South Korea? :
a.Won
b.Dollar
The answer: a
What is the currency of Nigeria? :
a.Naira
b.Dollar
The answer: a
Your final score: 13
```

```
The answer: b
What is the currency of Egypt? :
a.Pound
b.Dollar
The answer: b
What is the currency of Argentina? :
a.Euro
b.Peso
The answer: b
What is the currency of United Kingdom? :
a.Pound
b.Dollar
The answer: b
What is the currency of Spain? :
a.Euro
b.Dollar
The answer: b
What is the currency of South Korea? :
a.Won
b.Dollar
The answer: b
What is the currency of Nigeria? :
a.Naira
b.Dollar
The answer: b
Your final score: 7
```

Name:_ Dina Demyati_____, Number:_ 2290,_____, Submitted To GitHub:_ @DinaDemyati_____

Question 2: Simple Website with Python Flask Framework

Create a simple website with multiple pages using Flask, HTML, CSS, and Bootstrap. The website should demonstrate your understanding of web design principles.

Requirements:

A. Set up a local web server using XAMPP, IIS, or Python's built-in server (using Flask).
B. Apply CSS and Bootstrap to style the website and make it visually appealing.
C. Ensure that the website is responsive and displays correctly on different screen sizes.
D. Implement basic server-side functionality using Flask to handle website features

server.py | client.py | index.html

web > templates > index.html > html > body > div.container.mt-5 > table.table.table-striped.table-bordered > tbody

```
<!DOCTYPE html>
<html>
<head>
  <title>Currency Rate Website</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='css/bootstrap.min.css') }}">
    <link rel="stylesheet" href="static/css/style.css">
</head>
<body>
    <header>
        <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('contact') }}">Contact</a></li>
            </ul>
        </nav>
    </header>

  <div class="container mt-5">
    <h1 class="text-primary">Welcome to the Currency Rate Website!</h1>

    <h2 class="text-primary">Currency Rates:</h2>
    <table class="table table-striped table-bordered">
      <thead class="thead-dark">
        <tr>
          <th>Currency</th>
          <th>Rate (in USD)</th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <td>USD</td>
          <td>1.00</td>
        </tr>
        <tr>
          <td>EUR</td>
          <td>0.90</td>
        </tr>
        <tr>
          <td>GBP</td>
```

Name:_ Dina Demyati_____, Number:_ 2290,_____, Submitted To GitHub:_ @DinaDemyati____

server.py client.py index.html

web > templates > index.html > html > body > div.container.mt-5

```
            <tr>
              <td>GBP</td>
              <td>0.80</td>
            </tr>
            <tr>
              <td>JPY</td>
              <td>0.009</td>
            </tr>
            <tr>
              <td>AUD</td>
              <td>0.70</td>
            </tr>
            <tr>
              <td>CAD</td>
              <td>0.75</td>
            </tr>
            <tr>
              <td>CHF</td>
              <td>0.92</td>
            </tr>
            <tr>
              <td>CNY</td>
              <td>0.15</td>
            </tr>
            <tr>
              <td>INR</td>
              <td>0.014</td>
            </tr>
            <tr>
              <td>AED</td>
              <td>0.27</td>
            </tr>
          </tbody>
        </table>
      </div>
    </body>
    </html>
```

server.py client.py index.html about.html

web > templates > about.html > ...

```
<!DOCTYPE html>
<html>
<head>
  <title>Currency Rate Website - About</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='css/bootstrap.min.css') }}">
    <link rel="stylesheet" href="static/css/style.css">
</head>
<body>
    <header>
        <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('contact') }}">Contact</a></li>
            </ul>
        </nav>
    </header>

  <div class="container mt-5">
    <h1 class="text-primary">About Currency Rate Website</h1>
    <p class="lead">This website provides currency rates from various sources.</p>
    <p class="lead">Our goal is to help users easily access and compare currency exchange rates. We collect rates from reliable sources and preser
    <p class="lead">Whether you are a traveler planning your next trip, a business owner dealing with international transactions, or simply curio
    <p class="lead">Feel free to explore our website and check the latest currency rates. We strive to keep the information up to date and accurat
  </div>


</body>
</html>
```

Name:_ Dina Demyati_____, Number:_ 2290,_____, Submitted To GitHub:_ @DinaDemyati_____

server.py | client.py | index.html | contact.html

web > templates > contact.html > ...

```
<!DOCTYPE html>
<html>
<head>
  <title>Currency Rate Website - Contact</title>
  <link rel="stylesheet" href="{{ url_for('static', filename='css/bootstrap.min.css') }}">
  <link rel="stylesheet" href="static/css/style.css">
</head>
<body>
  <header>
    <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('contact') }}">Contact</a></li>
      </ul>
    </nav>
  </header>

  <div class="container mt-5">
    <h1 class="text-primary">Contact Currency Rate Website</h1>
    <p class="lead">You can contact us using the following information:</p>
    <ul class="lead">
      <li>Email: contact@currencyrates.com</li>
      <li>Phone: 922-111-123-321</li>
    </ul>
  </div>


</body>
</html>

```

server.py | client.py | index.html | app.py

web > app.py > ...

```
from flask import Flask, render_template

app = Flask(__name__)

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

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

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

if __name__ == '__main__':
    app.run(debug=True , port=8888)

```

Name:_ Dina Demyati_____, Number:_ 2290,_____, Submitted To GitHub:_ @DinaDemyati_____

يتم استيراد الوحدة Flask من إطار العمل.Flask

يتم إنشاء تطبيق من الكلاس Flask وتعيينه للمتغير .app يتم تمرير __name__ كمعامل لتحديد اسم التطبيق.

يتم استخدام مزخرف @ لتعريف مسارات الويب المختلفة.

تعريف الدالة index التي تعيد نموذج HTML مستخدماً render_template للصفحة الرئيسية.

تعريف الدالة about التي تعيد نموذج HTML مستخدماً render_template لصفحة "."about

تعريف الدالة contact التي تعيد نموذج HTML مستخدماً render_template لصفحة ". contact "

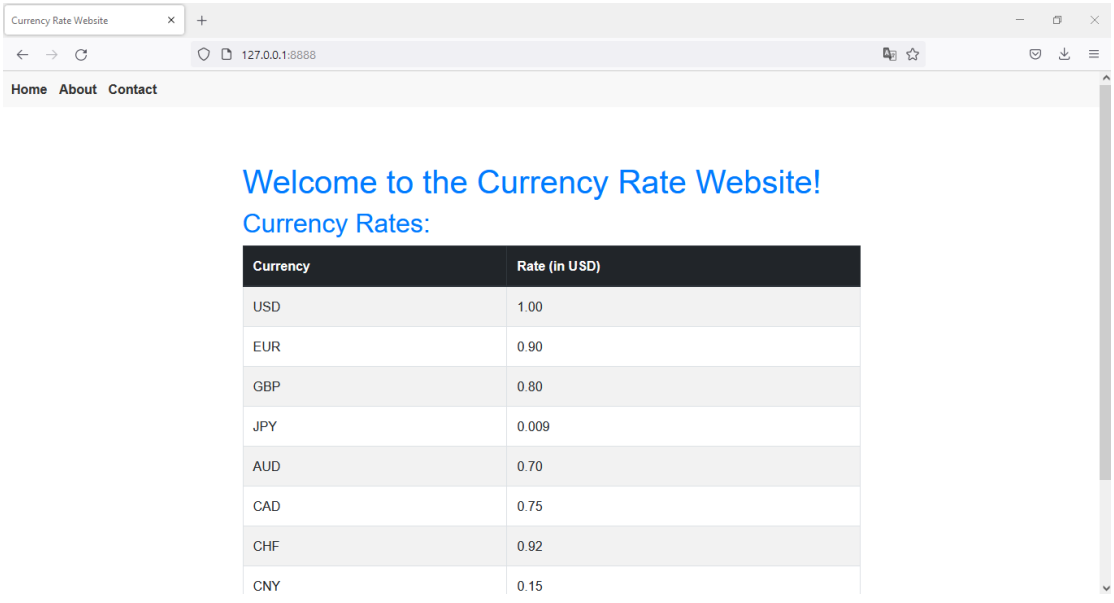
يتم استدعاء app.run() لتشغيل التطبيق على الخادم المحلي. يتم تفعيل وضع التصحيح (debug mode) بواسطة debug=True وتحديد رقم المنفذ.port=8888

عند تشغيل البرنامج، سيتم تشغيل التطبيق على الخادم المحلي ويكون لديك ثلاثة مسارات ويب متاحة:

المسار الرئيسي '/' الذي يستدعي دالة index ويعرض نموذج HTML للصفحة الرئيسية.

المسار 'about' الذي يستدعي دالة about ويعرض نموذج HTML لصفحة "." about

المسار 'contact' الذي يستدعي دالة contact ويعرض نموذج HTML لصفحة "'contact



Name:_ Dina Demyati_____, Number:_ 2290,_____, Submitted To GitHub:_ @DinaDemyati_____

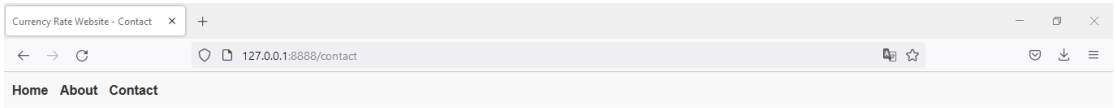
About Currency Rate Website

This website provides currency rates from various sources.

Our goal is to help users easily access and compare currency exchange rates. We collect rates from reliable sources and present them in a user-friendly manner.

Whether you are a traveler planning your next trip, a business owner dealing with international transactions, or simply curious about currency rates, our website provides the information you need.

Feel free to explore our website and check the latest currency rates. We strive to keep the information up to date and accurate.



Contact Currency Rate Website

You can contact us using the following information:

- Email: contact@currencyrates.com
- Phone: 922-111-123-321
- Address: 123 Main St, City, Country