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:_علي نواف حمود , Number: 2053 , Submitted To GitHub:YES

Second Network Programming Homework

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

في البداية، يتم تعريف قائمة الأسئلة والإجابات في الخادم كمتغير عام. يتم تخزين الأسئلة والإجابات في قاموس (Dictionary) اسمه quiz حيث تكون الأسئلة هي المفاتيح Keys والإجابات هي القيم values. هذا القاموس يُمكن الخادم من استدعاء الأسئلة والإجابات بسهولة خلال فترة الاختبار.

بعد ذلك، يتم تعريف تابع لمعالجة اتصالات العملاءhandle_client .يتم تنفيذ هذا التابع في كل مرة يتصل فيها عميل جديد بالخادم. يبدأ التابع بإرسال الأسئلة إلى العميل واستقبال الإجابات التي يرسلها.

بالنسبة للتعامل مع عدة عملاء في نفس الوقت، يتم استخدام المسارات (Threads) في الخادم، بحيث يتم إنشاء مسار منفصل لكل عميل يتصل بالخادم. يتم تنفيذ التابع handle_client في كل مسار عندما يتصل عميل جديد بالخادم، ويعمل على تشغيل الاختبار على العميل وتحديث النتائج.

يتم تخزين نتائج العملاء في قاموس (Dictionary) ايضاً اسمه scores. يتم استخدام هذا القاموس لتتبع نتائج كل عميل خلال فترة الاختبار.

الأسئلة والإجابات هي من نوع (String). يتم تحويل الرسائل إلى بايت (Bytes) باستخدام التابع encode قبل إرسالها عبر الشبكة. وعند استلام الرسائل من الشبكة، يتم تحويلها مرة أخرى إلى نص باستخدام التابع decode.

بالنسبة للعميل، يتم استقبال الأسئلة من الخادم باستخدام التابع recv وإرسال الإجابات باستخدام التابع sendall. وعندما يتم الانتهاء من الاختبار، يتم استلام النتائج النهائية باستخدام تابع recv أخر.

يتم إغلاق الاتصال بعد الانتهاء من الاختبار باستخدام التابع close.

كود السيرفر:

```
import socket
import threading

# Define the quiz questions and answers
quiz = {
    "1+1": "2",
    "2+1": "3",
    "3+2": "5",
    "2+3": "5",
}

# Define a function to handle client connections
def handle_client(conn, addr, scores):
    print(f"New connection from {addr}")

    # Send the quiz questions to the client
    for question in quiz.keys():
        conn.sendall(question.encode())
        answer = conn.recv(1024).decode()

        # Check the answer and update the score
```

```
            if answer == quiz[question]:
                scores[addr] += 1

        # Send the final score to the client
        final_score = f"Your final score is: {scores[addr]}/{len(quiz)}"
        conn.sendall(final_score.encode())
        conn.close()
        print(f"Connection from {addr} closed")


    # Define the main function
    def main():
        # Create a TCP socket and bind it to a port
        host = "127.0.0.1"
        port = 9999
        server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        server.bind((host, port))
        server.listen()
        print(f"Server listening on {host}:{port}")

        # Keep track of the scores for each client
        scores = {}

        # Accept client connections and start a new thread for each
    client
        while True:
            conn, addr = server.accept()
            scores[addr] = 0
            thread = threading.Thread(target=handle_client, args=(conn,
    addr, scores))
            thread.start()


    if __name__ == "__main__":
        main()
```

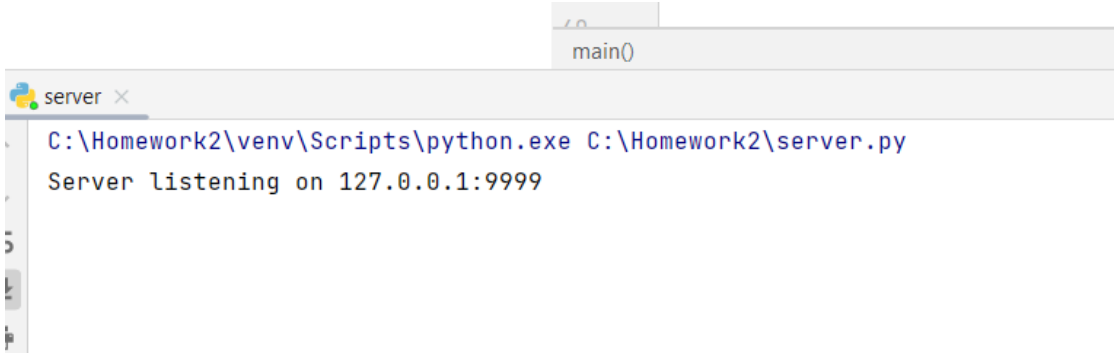
كود العميل:

```
import socket

# Define the server address and port
host = "127.0.0.1"
port = 9999

# Create a TCP socket and connect to the server
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect((host, port))
print(f"Connected to {host}:{port}")

# Receive the quiz questions from the server and send answers
while True:
    question = client.recv(1024).decode()
    if not question:
        break
    print(question)
    answer = input()
    client.sendall(answer.encode())
```

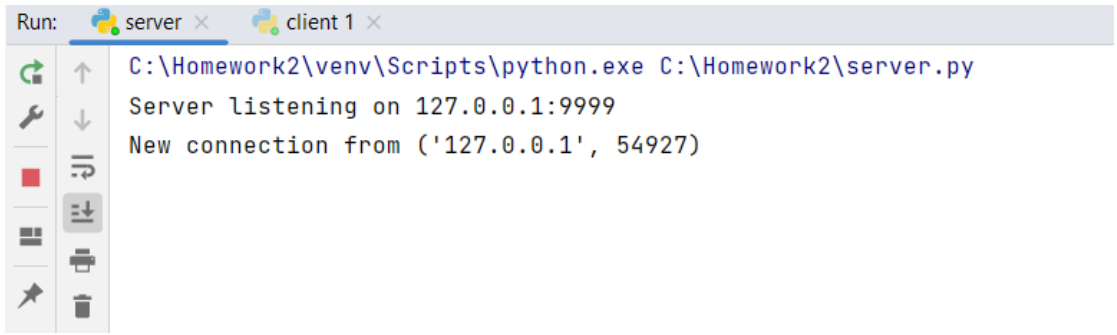
```
# Receive the final score from the server and close the connection
final_score = client.recv(1024).decode()
print(final_score)
client.close()
```

التشغيل:

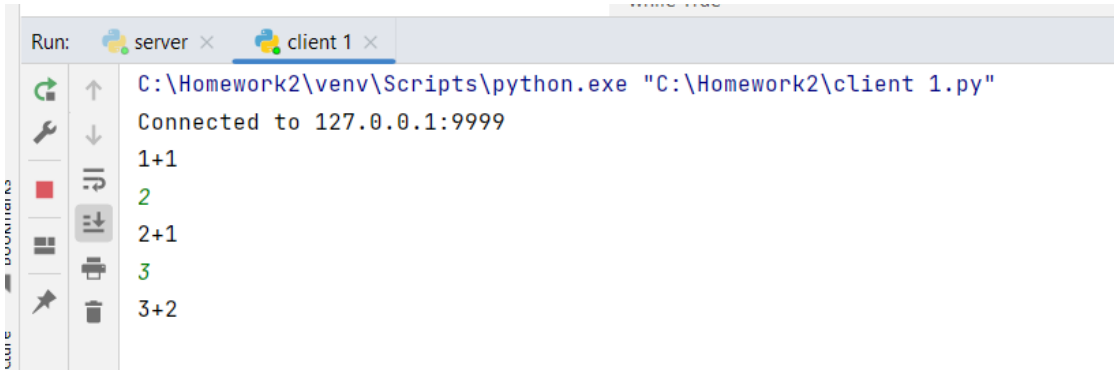
في البداية نقوم بتشغيل الخادم فيظهر لدينا الشكل التالي:



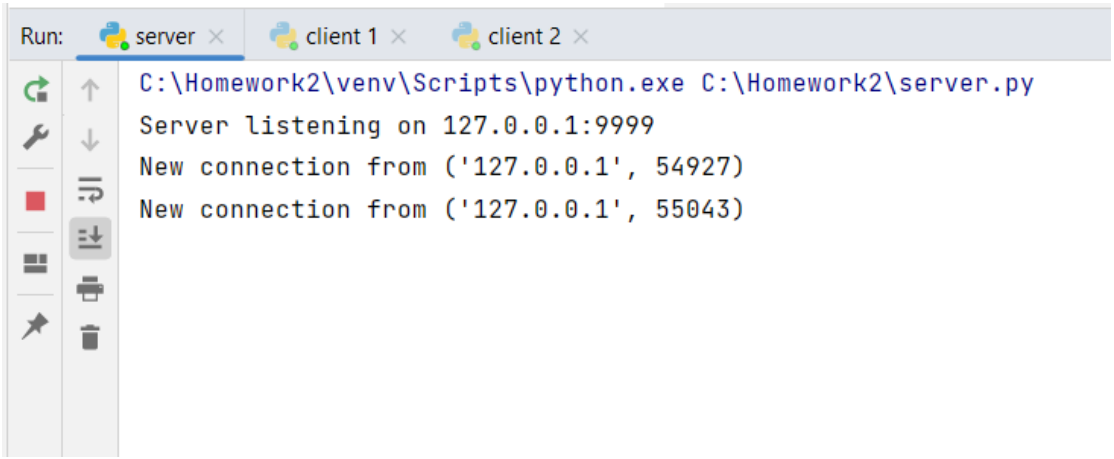
يقوم العميل الأول بالاتصال فيظهر لنا في خرج السيرفر عنوان العميل الاول:

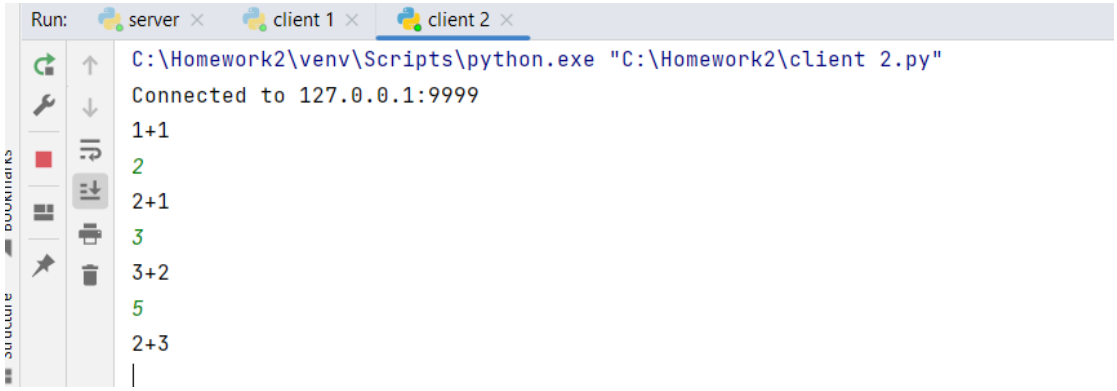


تبدأ الأسئلة تظهر على خرج العميل الأول ويبدأ بالاجابة:

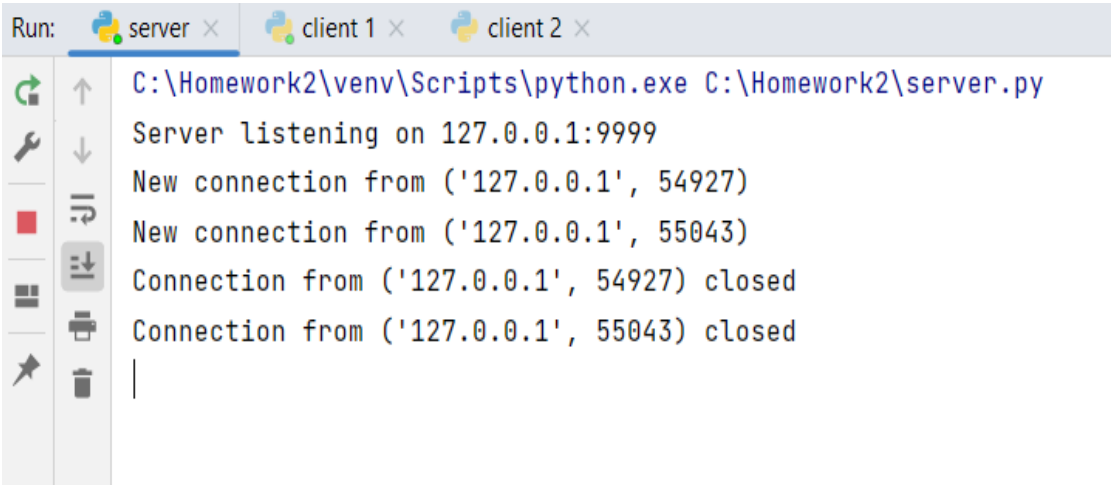


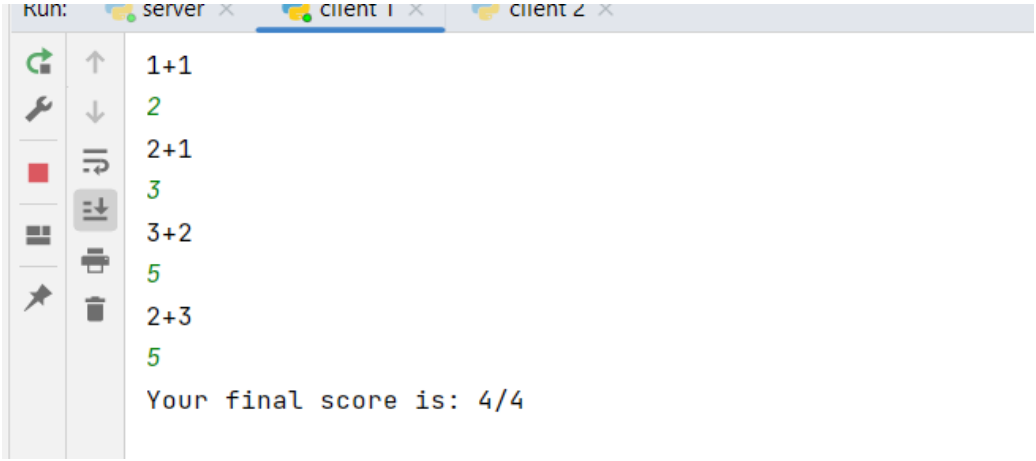
يقوم العميل الثاني بالاتصال في نفس الوقت:

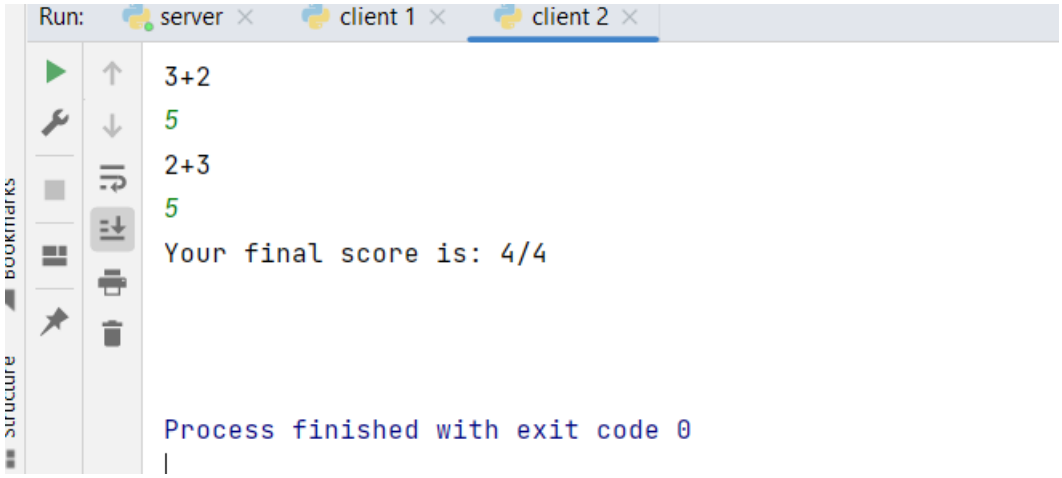




عند الانتهاء من الأسئلة يتم غلق الاتصال وتظهر النتيجة على خرج العميل:







Question 2: Simple Website with Python Flask Framework

يتم استدعاء الحزم اللازمة منFlask ، حيث يتم استدعاء الحزمة Flask و .render_template يتم إنشاء تطبيق Flask جديد باستخدام البيانات باستخدام السطر التالي:

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

نعرف ثلاثة توابع يتم استدعاء حسب الرابط الذي يتم التقاطه باستخدام app.route@ الذي يتم استخدمه للربط بين عنوان URL محدد تابع Python.

نعرف التابع home الذي يقوم بإرجاع الصفحة الرئيسية home.html عند الاتصال بالعنوان URL الأساسي "/". والتابع page1 يقوم بإرجاع الصفحة 1.html عند الاتصال بالعنوان URL "1/" والتابع page2 يقوم بإرجاع الصفحة2.html عند الاتصال بالعنوان URL "2/".

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

@app.route('/1')
def page1():
    return render_template('1.html')

@app.route('/2')
def page2():
    return render_template('2.html')
```

يتم تشغيل سيرفر Flask **باستخدام السطر التالي:**

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

نقوم بإنشاء مجلد templates **خاص بملفات ال**html **ومجلد** static **خاص بملفات** js **و** css.

ملفات html**:**

Home.html

```
<!DOCTYPE html>
<html>
<head>
<style>
#a {
  display: flex;
  justify-content: center;
  align-items: center;
color: green;
}
#b{
display: flex;
justify-content: center;
color: red;
}
</style>
<meta name="viewport" content="width=device-width, initial-scale=1">
   <title>My Website</title>
</head>
<body>
<div id="b">
      <a href="/">Home</a>
      <a href="/1">Page 1</a>
      <a href="/2">Page 2</a>
```

```
</div>
   <div id="a">
      <h1>السؤال الثاني</h1>
</div>
<div id="a">
      <p>علي نواف  حمود 2053 </p>
</div>
<div id="a">
      <img src="{{ url_for('static', filename='1.jpg') }}" alt="My Image">
   </div>
</body>
</html>
```

العنصر <!DOCTYPE html>: يعرف نوع الملف كـ HTML5.

العنصر <html>: يشير إلى بداية عنصر HTML وينهي بعنصر </html>.

توجد أقسام مختلفة داخل العنصر <head> العنصر <style> يستخدم لتعريف أنماط CSS داخل العنصر <head> حيث تم استخدام هذه الأنماط لتعريف العناصر a# وb# و تم تعيين display:flex لكلا العنصرين a# و b# لتمكين التحكم في توزيع العناصر داخلهما. تم تحديد justify-content:center و align-items:center للعنصر a# لتوسيط العناصر الموجودة داخله. تم تحديد color لكلا العنصرين a# و b# لتحديد لون النص.

تم استخدام <meta> لتحديد عرض الشاشة والمقياس الأولي لتصفح الصفحة لجعل الصفحة responsive.

العنصر <body>: يشير إلى بداية الجسم أو المحتوى الأساسي للصفحة.

قمنا بإنشاء عناصر مختلفة داخل العنصر <body> وتم استخدام العنصر لإنشاء عناصر فرعية مختلفة. تم تحديد معرف id مختلف لكل عنصر فرعي #a و #b. تم استخدام id للتحكم في التنسيق والتصميم الخاص بكل عنصر فرعي. تم استخدام العنصر <a> لإنشاء روابط لصفحات HTML الأخرى. تم استخدام العنصر <h1> و <p> و <img> لإضافة عناصر نصية وصورة إلى الصفحة.

1.html

```
<!DOCTYPE html>
<html>
<head>
   <title>Page 1</title>
</head>
```

```
<body>
        <a href="/">Home</a>
        <a href="/1">Page 1</a>
        <a href="/2">Page 2</a>
    <div>
        <h1>الصفحة الاولى</h1>
    </div>
</body>
</html>
```

2.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Page 2</title>
</head>
<body>
        <a href="/">Home</a>
        <a href="/1">Page 1</a>
        <a href="/2">Page 2</a>
    <div>
        <h1>الصفحة الثانية</h1>
    </div>
</body>
</html>
```

مرحلة التشغيل:

يقوم السيرفر بالعمل على المضيف المحلي والمنفذ 5000 أي الرابط http://127.0.0.1:5000

بعد تشغيل السيرفر وطلب الصفحة الرئيسية يكون خرج السيرفر كالتالي:

وشكل الصفحة:

الصفحة الثانية:

Home Page 1 Page 2

الصفحة الاولى

الصفحة الثالثة:

Home Page 1 Page 2

الصفحة الثانية

خرج السيرفر: