



Name: __ باسمال نزيه طرموش__ Number 2034__

Second Network Programming Homework

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

عند تشغيل السيرفر، يتم إنشاء socket باستخدام تابع socket في المكتبة socket وتعيينها على AF_INET و SOCK_STREAM كما يستخدم setsockopt لتعيين SO_REUSEADDR إلى 1 لتجنب مشكلات الاتصال المستقبلية.

ثم يتم استدعاء تابع listen للتنصت على الاتصالات الواردة للسيرفر حيث عندما يتم تلقي اتصال جديد يتم استدعاء تابع accept لتكوين socket جديد لكل اتصال ويتم إسنادها إلى csock و caddr على التوالي.

يتم إنشاء thread جديد باستخدام threading.Thread عند تلقي اتصال جديد لكل عميل ويتم تمرير csock و caddr إلى التابع handle_client كبارامترات.

تابع handle_client يستخدم حلقة for تحوي على تعليمات لإرسال الأسئلة للعميل و تلقي الإجابات منه وتتم مقارنة كل إجابة مع الإجابة الصحيحة الموجودة في القائمة ANSWERS وإذا كانت الإجابة صحيحة يتم زيادة متغير a بمقدار واحد وفي النهاية يتم إرسال إجمالي الإجابات الصحيحة إلى العميل ويتم إغلاق الاتصال.

يتم تشغيل السيرفر باستخدام التابع start_server ويتم تمرير ip السيرفر ورقم المنفذ كبارامترات.

```
import socket
import threading
```

```
# الأسئلة قائمة
```

```
QUESTIONS = [
    "What is the result of 5 + 3?",
    "What is the result of 7 - 2?",
    "What is the result of 4 * 6?",
    "What is the result of 10 / 2?",
    "What is the result of 9 % 4?",
    "What is the square root of 25?",
    "What is the cube root of 27?",
    "What is the result of (2 + 3) * 4?",
    "What is the result of 2^3?",
    "What is the result of (6 * 4) / (2 + 1)?"
]
```

```
# الصحيحة الإجابات قائمة
```

```
ANSWERS = [
    "8",
    "5",
    "24",
    "5",
    "1",

```

```

"5",
"3",
"20",
"8",
"8"
]

# العميل اتصال لمعالجة تابع
def handle_client(csock, caddr):
    # الصحيحة الإجابات عدد لتخزين متغير
    a = 0
    # الإجابات وتلقي الأسئلة لإرسال حلقة
    for i in range(len(QUESTIONS)):
        # العميل إلى السؤال إرسال
        csock.sendall(QUESTIONS[i].encode())
        # الفارغة الأحرف وإزالة العميل من الإجابة تلقي
        answer = csock.recv(1024).decode().strip()
        # الصحة حالة في العدد وزيادة الإجابة صحة من التحقق
        if answer == ANSWERS[i]:
            a += 1
    # العميل إلى الصحيحة الإجابات عدد إرسال
    csock.sendall(f"Your final result is {a}".encode())
    # الاتصال إغلاق
    csock.close()
    # الاتصال إغلاق من للتحقق رسالة طباعة
    print(caddr, "closed connection")

# السيرفر لبدء تابع
def start_server(host, port):
    # والمنفذ بالمضيف وربطه socket إنشاء
    ssock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    ssock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR,
1)
    ssock.bind((host, port))
    # الواردة الاتصالات على الاستماع
    ssock.listen()
    # السيرفر بدء من للتحقق رسالة طباعة
    print(f"Server listening on {host}:{port}")
    # الجديدة الاتصالات لقبول حلقة
    while True:
        # عميل لكل جديد socket وإنشاء الاتصال قبول
        csock, caddr = ssock.accept()
        # الاتصال قبول من للتحقق رسالة طباعة
        print(f"Accepted connection from
{caddr[0]}:{caddr[1]}")
        # التابع على وتشغيله عميل لكل جديد thread إنشاء
        handle_client()
        threading.Thread(target=handle_client, args=(csock,
caddr)).start()

# مستقل كبرنامج البرنامج تنفيذ عند السيرفر تشغيل في الشروع
if __name__ == '__main__':
    start_server('localhost', 8888)

```

كود العميل:

```
import socket

# الاختبار لإجراء تابع
def take_quiz(sock):
    # الإجابات وإرسال الأسئلة لتلقي حلقة
    while True:
        # السيرفر من السؤال تلقي
        question = sock.recv(1024).decode()
        # فارغة رسالة وجود عدم من التأكد
        if not question:
            break
        # المستخدم من الإجابة طلب
        answer = input(question + '\n')
        # السيرفر إلى الإجابة إرسال
        sock.sendall(answer.encode())
        # وطباعتها السيرفر من النتيجة تلقي
        score = sock.recv(1024).decode()
        print(score)

# العميل لبدء تابع
def start_client(host, port):
    # المحددين والمنفذ بالمضيف به والاتصال عميل socket إنشاء
    csock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    csock.connect((host, port))
    # take_quiz التابع باستخدام الاختبار إجراء
    take_quiz(csock)
    # الاتصال إغلاق
    csock.close()

# مستقل كبرنامج البرنامج تنفيذ عند العميل تشغيل في الشروع
if __name__ == '__main__':
    start_client('localhost', 8888)
```

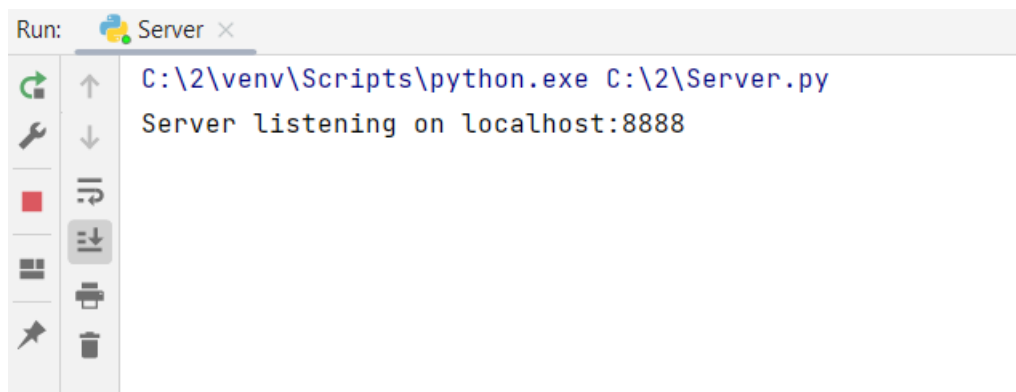
تم تعريف تابعين:

take_quiz(sock) هو التابع الذي يتعامل مع السيرفر من خلال socket client ويقوم بتلقي الأسئلة من السيرفر وإرسال الإجابات من المستخدم إلى السيرفر وتلقي النتيجة وطباعتها.

start_client(host, port) هو التابع الرئيسي الذي يقوم بإنشاء socket client والاتصال به بالمضيف والمنفذ المحددين ومن ثم يستخدم التابع take_quiz(sock) لإجراء الاختبار.

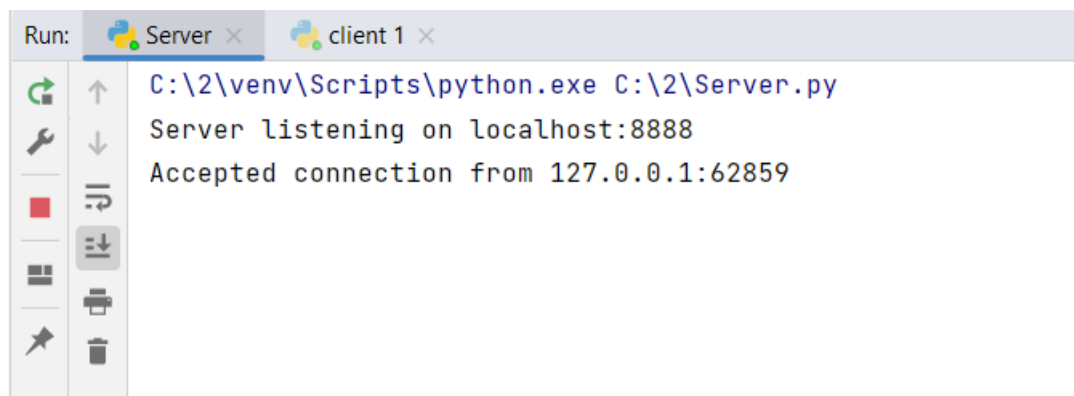
يتم تشغيل كود العميل عن طريق استدعاء التابع start_client('localhost', 8888) والذي يمرر إليه المضيف والمنفذ.

السيرفر:



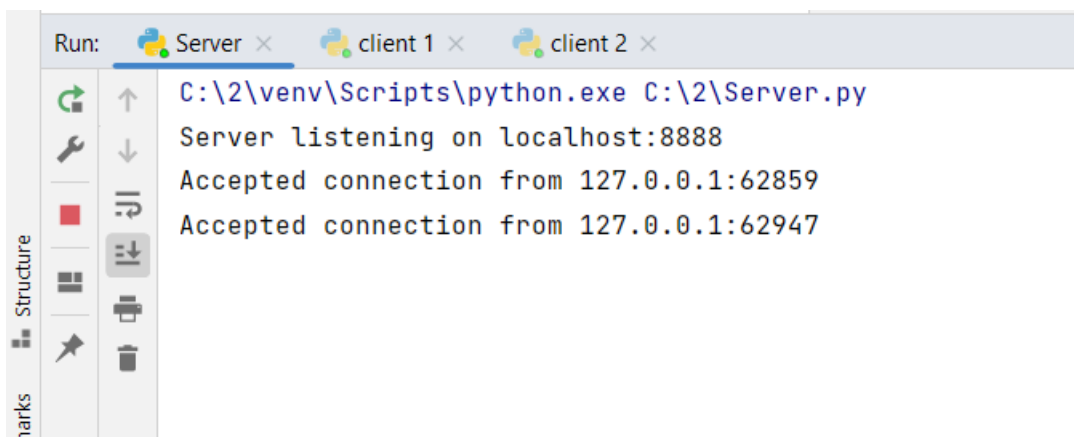
```
Run: Server ×  
C:\2\venv\Scripts\python.exe C:\2\Server.py  
Server listening on localhost:8888
```

بعد اتصال واحد:



```
Run: Server × client 1 ×  
C:\2\venv\Scripts\python.exe C:\2\Server.py  
Server listening on localhost:8888  
Accepted connection from 127.0.0.1:62859
```

بعد اتصال ثاني مع الأول:



```
Run: Server × client 1 × client 2 ×  
C:\2\venv\Scripts\python.exe C:\2\Server.py  
Server listening on localhost:8888  
Accepted connection from 127.0.0.1:62859  
Accepted connection from 127.0.0.1:62947
```

بعد انتهاء الاختبار للاحد العملاء:

```
Run: Server x client 1 x client 2 x
C:\2\venv\Scripts\python.exe C:\2\Server.py
Server listening on localhost:8888
Accepted connection from 127.0.0.1:62859
Accepted connection from 127.0.0.1:62947
('127.0.0.1', 62859) closed connection
|
```

العميل:

```
Run: Server x client 1 x client 2 x
29
What is the result of 2^3?
8
What is the result of (6 * 4) / (2 + 1)?
8
Your final result is 9

Process finished with exit code 0
|
```

Question 2: Simple Website with Python Flask Framework

يتم إنشاء التطبيق باستخدام Flask ويتم تعريف التطبيق باستخدام الكود التالي:

```
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)

```

حيث:

يتم اشتقاق غرض من الكلاس Flask باستخدام الأمر التالي:

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

ويتم تعريف المسارات (routes) بواسطة الكود التالي:

```

@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')

```

يتم تعريف التابع render_template الذي يستخدم لعرض صفحات HTML حيث يتم استدعاء هذا التابع في التتابع المعرفة للمسارات على سبيل المثال في التابع index تم استدعاء التابع render_template لعرض صفحة HTML الخاصة بالصفحة الرئيسية حيث يتم تنفيذ التابع index عند ورود الرابط الأساسي من المتصفح.

يتم تشغيل التطبيق باستخدام التابع run حيث يتم تنفيذه إذا كان التشغيل مباشرة من هذا الكود:

```

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

```

الخطوة 2: إنشاء الصفحات HTML و CSS و Bootstrap

تم إنشاء ثلاث صفحات بسيطة: الصفحة الرئيسية (index.html) و صفحة "حول" (about.html) و صفحة "اتصل بنا" (contact.html).
نضع الصفحات في مجلد templates

تم إنشاء ملف css بسيط نضعه في مجلد static في المجلد css.

• ملف الـ CSS:

```
body {
  font-family: Arial, sans-serif;
  background-color: #f7f7f7;
  color: #333;
}

.container {
  margin-top: 50px;
  text-align: center;
}

.btn {
  margin-right: 10px;
}
```

يتم تعريف خصائص الجسم (body) بواسطة الكود التالي:

```
body {
  font-family: Arial, sans-serif;
  background-color: #f7f7f7;
  color: #333;
}
```

يتم تعريف خصائص عنصر الحاوية (container) بواسطة الكود التالي:

```
.container {
  margin-top: 50px;
  text-align: center;
}
```

يتم تعريف خصائص الزر (button) بواسطة الكود التالي:

```
.btn {
  margin-right: 10px;
}
```

• الصفحة الرئيسية (index.html)

```
<!DOCTYPE html>
<html>
<head>
  <title>Flask Website</title>
  <link rel="stylesheet" href="{ url_for('static',
filename='css/style.css') }}">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.cs
s">
</head>
<body>
  <div class="container">
```

```

<h1>Welcome to Flask Website!</h1>
<p>This is a simple website built using Flask.</p>
<a href="{{ url_for('about') }}" class="btn btn-primary">About</a>
<a href="{{ url_for('contact') }}" class="btn btn-
primary">Contact</a>
</div>
</body>
</html>

```

تم استدعاء ملف CSS الخاص بالصفحة بواسطة الكود التالي:

```
<link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
```

تم استدعاء Bootstrap بواسطة الكود التالي:

```

<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

```

تم إضافة روابط لصفحات حول و اتصل بنا باستخدام الكود التالي:

```

<a href="{{ url_for('about') }}" class="btn btn-primary">About</a>
<a href="{{ url_for('contact') }}" class="btn btn-primary">Contact</a>

```

الشكل التالي هو شكل الصفحة في المتصفح

Welcome to Flask Website!

This is a simple website built using Flask.

About

Contact

• صفحة حول (about.html)

```

<!DOCTYPE html>
<html>
<head>
  <title>About | Flask Website</title>
  <link rel="stylesheet" href="{{ url_for('static',
filename='css/style.css') }}">
  <link rel="stylesheet"

```



```

href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
</head>
<body>
  <div class="container">
    <h1>About</h1>
    <p>This website was built using Flask.</p>
    <a href="{{ url_for('index') }}" class="btn btn-
primary">Home</a>
    <a href="{{ url_for('contact') }}" class="btn btn-
primary">Contact</a>
  </div>
</body>
</html>

```

تم إضافة روابط لصفحة الرئيسية وصفحة اتصل بنا

الشكل التالي هو شكل الصفحة في المتصفح

About

This website was built using Flask.

Home

Contact

• صفحة اتصل بنا (contact.html)

```

<!DOCTYPE html>
<html>
<head>
  <title>Contact | Flask Website</title>
  <link rel="stylesheet" href="{{ url_for('static',
filename='css/style.css') }}">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.cs
s">
</head>
<body>
  <div class="container">
    <h1>Contact</h1>
    <form action="" method="post">
      <div class="form-group">
        <label for="name">Name:</label>

```

```

        <input type="text" name="name" id="name" class="form-
control" required>
    </div>
    <div class="form-group">
        <label for="email">Email:</label>
        <input type="email" name="email" id="email" class="form-
control" required>
    </div>
    <div class="form-group">
        <label for="message">Message:</label>
        <textarea name="message" id="message" class="form-control"
required></textarea>
    </div>
    <button type="submit" class="btn btn-primary">Send</button>
</form>
<a href="{{ url_for('index') }}" class="btn btn-primary">Home</a>
<a href="{{ url_for('about') }}" class="btn btn-primary">About</a>
</div>
</body>
</html>

```

تم إنشاء نموذج اتصال يتضمن حقول الاسم والبريد الإلكتروني والرسالة باستخدام الكود التالي:

```

<form action="" method="post">
    <div class="form-group">
        <label for="name">Name:</label>
        <input type="text" name="name" id="name" class="form-control" required>
    </div>
    <div class="form-group">
        <label for="email">Email:</label>
        <input type="email" name="email" id="email" class="form-control" required>
    </div>
    <div class="form-group">
        <label for="message">Message:</label>
        <textarea name="message" id="message" class="form-control"
required></textarea>
    </div>
    <button type="submit" class="btn btn-primary">Send</button>
</form>

```

الشكل التالي هو شكل الصفحة في المتصفح

Contact

Name:

Email:

Message:

Send

Home

About