

وظيفة برمجة الشبكات الثانية

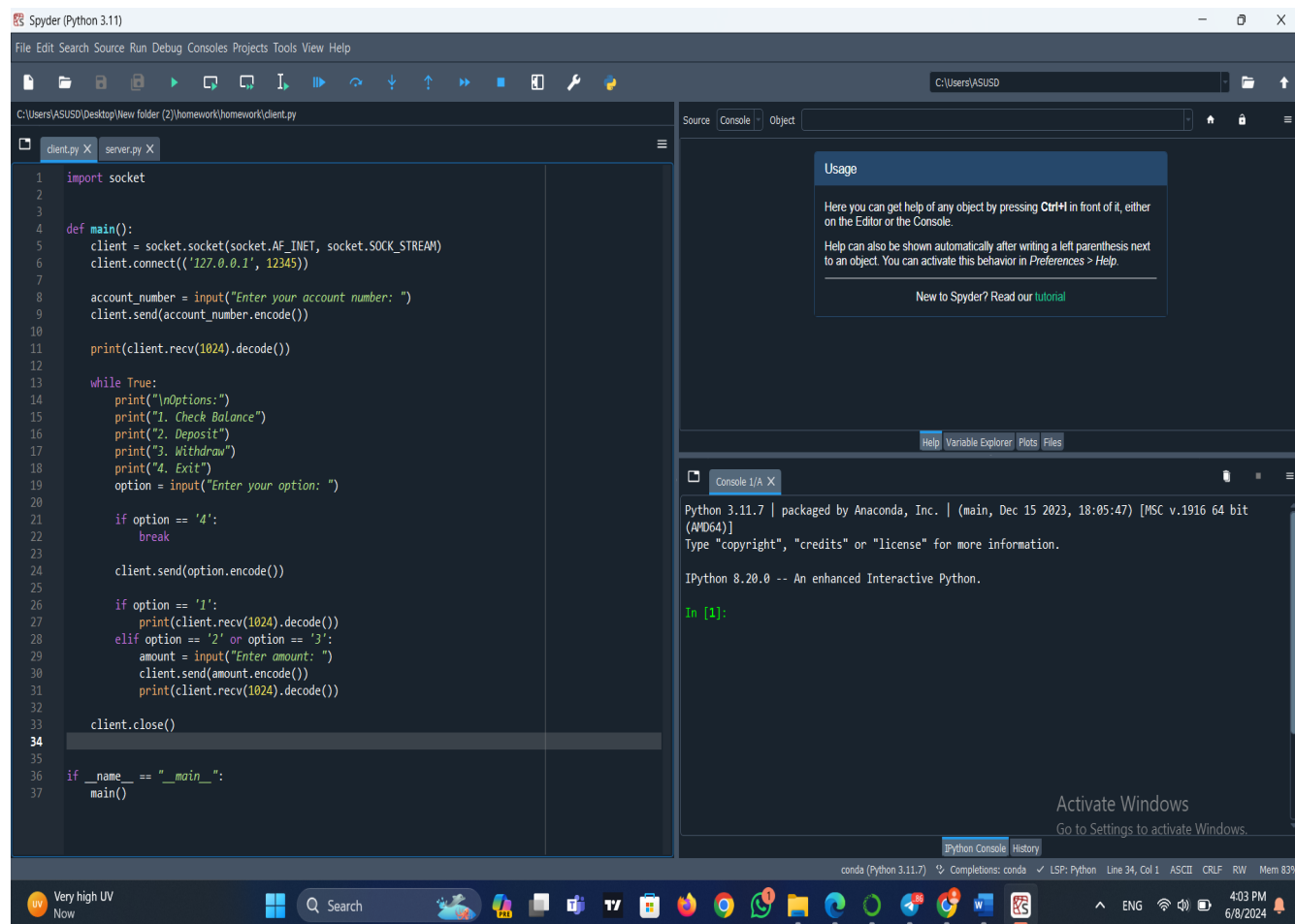
إعداد الطالب :

يوسف غسان خليل 2094

محمد أحمد قدور 2535

Q1 :

كود الزبون :



The screenshot shows the Spyder Python IDE interface. The left pane displays the code for 'client.py'. The code imports the 'socket' module and defines a 'main()' function. It creates a client socket, connects to '127.0.0.1' on port '12345', and sends the account number. It then enters a loop where it prints menu options (1. Check Balance, 2. Deposit, 3. Withdraw, 4. Exit) and processes user input based on the selected option. The right pane shows the 'Console' tab with the IPython prompt and the output of the program, including the menu options and the user's input '1'.

```
1 import socket
2
3
4 def main():
5     client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
6     client.connect(('127.0.0.1', 12345))
7
8     account_number = input("Enter your account number: ")
9     client.send(account_number.encode())
10
11     print(client.recv(1024).decode())
12
13     while True:
14         print("\nOptions:")
15         print("1. Check Balance")
16         print("2. Deposit")
17         print("3. Withdraw")
18         print("4. Exit")
19         option = input("Enter your option: ")
20
21         if option == '4':
22             break
23
24         client.send(option.encode())
25
26         if option == '1':
27             print(client.recv(1024).decode())
28         elif option == '2' or option == '3':
29             amount = input("Enter amount: ")
30             client.send(amount.encode())
31             print(client.recv(1024).decode())
32
33     client.close()
34
35
36 if __name__ == "__main__":
37     main()
```

Python 3.11.7 | packaged by Anaconda, Inc. | (main, Dec 15 2023, 18:05:47) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license()" for more information.
IPython 8.20.0 -- An enhanced Interactive Python.
In [1]:

شرح الكود :

- 1- يتم إنشاء كائن مأخذ توصيل لاتصالات TCP/IP.
- 2- يتصل العميل بخادم يعمل على 127.0.0.1 (مضيف محلي) على المنفذ 12345.
- 3- يُطلب من المستخدم إدخال رقم حسابه، والذي يتم إرساله بعد ذلك إلى الخادم.

4- يرسل الخادم الرد، والذي يتم طباعته.

5- يتم عرض قائمة بها خيارات التحقق من الرصيد أو الإيداع أو السحب أو الخروج.

6- يقوم المستخدم بتحديد خيار، وبناءً على هذا التحديد، يتم اتخاذ إجراءات مختلفة:

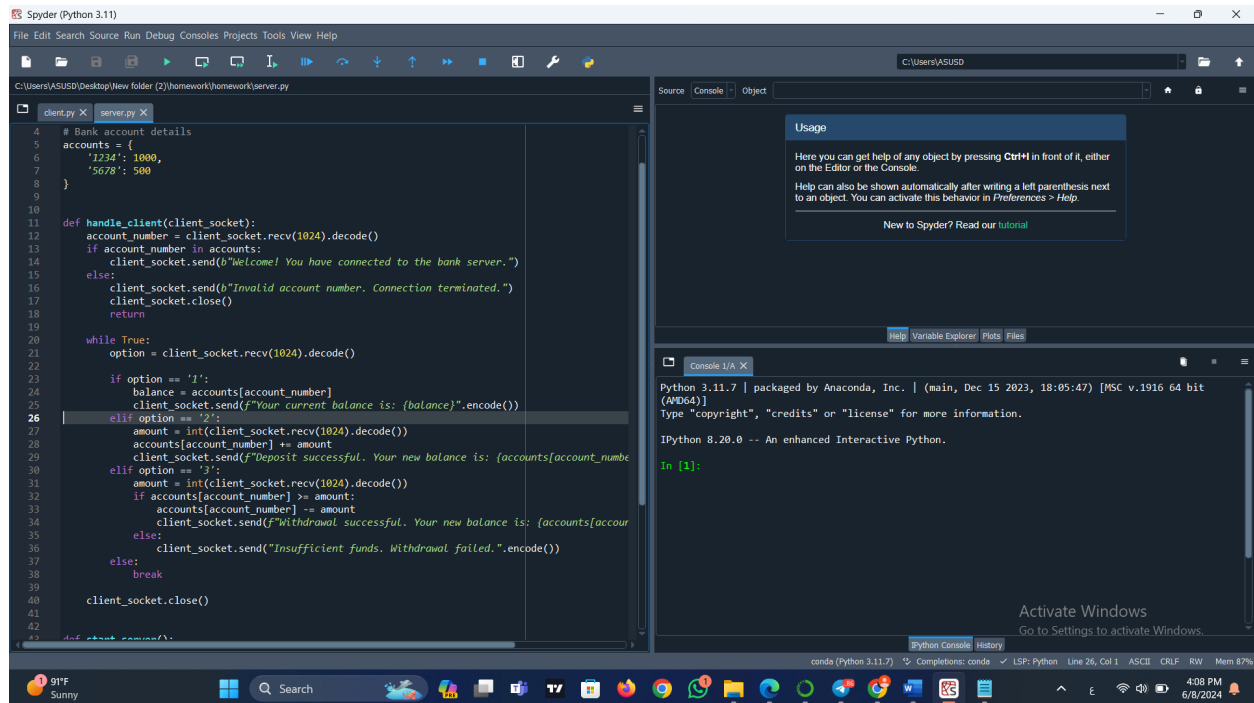
إذا تم تحديد "1"، فإنه يتحقق من الرصيد.

إذا تم تحديد "2" أو "3"، فإنه يطالب بمبلغ ثم يقوم بإجراء إيداع أو سحب.

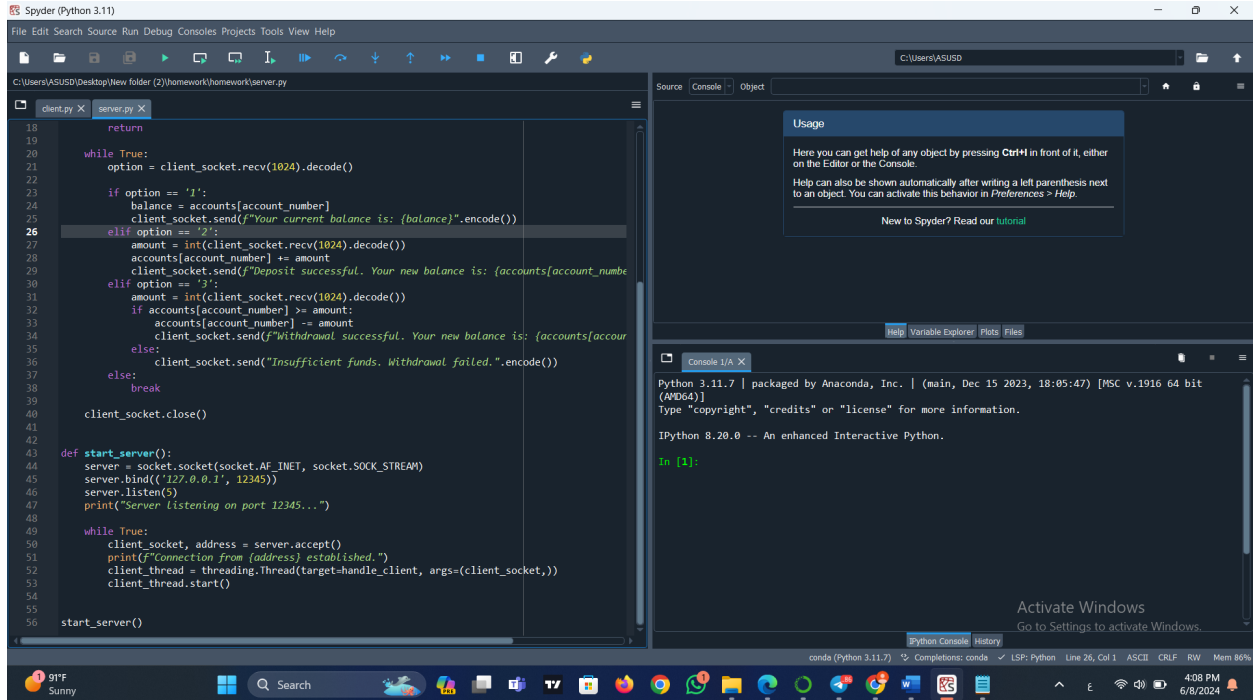
7- تستمر هذه الحلقة حتى يختار المستخدم "4" للخروج.

8- وأخيراً تم إغلاق الاتصال بالخادم.

كود السيرفر :



```
1 # Bank account details
2 accounts = {
3     '1234': 1000,
4     '5678': 500
5 }
6
7 def handle_client(client_socket):
8     account_number = client_socket.recv(1024).decode()
9     if account_number in accounts:
10         client_socket.send(b"Welcome! You have connected to the bank server.")
11     else:
12         client_socket.send(b"Invalid account number. Connection terminated.")
13         client_socket.close()
14         return
15
16 while True:
17     option = client_socket.recv(1024).decode()
18
19     if option == '1':
20         balance = accounts[account_number]
21         client_socket.send(f"Your current balance is: {balance}".encode())
22     elif option == '2':
23         amount = int(client_socket.recv(1024).decode())
24         accounts[account_number] += amount
25         client_socket.send(f"Deposit successful. Your new balance is: {accounts[account_number]}".encode())
26     elif option == '3':
27         amount = int(client_socket.recv(1024).decode())
28         if accounts[account_number] >= amount:
29             accounts[account_number] -= amount
30             client_socket.send(f"Withdrawal successful. Your new balance is: {accounts[account_number]}".encode())
31         else:
32             client_socket.send("Insufficient funds. Withdrawal failed.".encode())
33     else:
34         break
35
36 client_socket.close()
37
38 def start_server():
39     server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
40     server_socket.bind(('0.0.0.0', 1024))
41     server_socket.listen(5)
42     while True:
43         client_socket, address = server_socket.accept()
44         handle_client(client_socket)
```



- 1- يقوم قاموس يسمى الحسابات بتخزين أرقام الحسابات المصرفية كمفاتيح والأرصدة المقابلة لها كقيم.
- 2- تم تعريف تابع `Handle_client` لإدارة الاتصال مع عميل واحد. يتلقى رقم حساب، ويتحقق من وجوده، ثم يدخل في حلقة للتعامل مع أوامر مختلفة مثل التحقق من الرصيد، أو الإيداع، أو السحب.
- 3- تقوم وظيفة `start_server` بإنشاء مقبس خادم يستمع للاتصالات الواردة على `127.0.0.1` (المضيف المحلي) في المنفذ `12345`.
- 4- عند قبول الاتصال، يتم إنشاء مؤشر ترابط جديد للتعامل مع هذا الاتصال باستخدام وظيفة `Handle_client`.
- 5- الخادم يعمل لأجل غير مسمى، ويقبل الاتصالات الجديدة ويتعامل معها في سلاسل منفصلة.

يسمح هذا لعدة عملاء بالاتصال بالخادم في نفس الوقت، ولكل منهم سلسلة التنفيذ الخاصة به، مما يتيح المعالجة المتزامنة لطلبات العملاء المتعددة.

Q2:

شرح الكود :

لإنشاء تطبيق ويب Flask ويستخدم إطار العمل Python هذا الكود مكتوب بلغة بسيط. فيما يلي شرح الكود بالكامل باللغة العربية

```
python
from flask import Flask, render_template,
request

app = Flask(__name__)
```

- **import Flask, render_template, request:** نستورد الكائن Flask من مكتبة request وrender_template والدوال Flask.
- **app = Flask(name):** Flask ننشئ مثيلاً لتطبيق.

```
pytome
def check_guess(guess, answer):
if guess.lower() == answer.lower():
    return True
return False
```

- **check_guess(guess, answer):** دالة تقوم بفحص إذا كانت الإجابة بعد تحويلهما إلى (answer) تطابق الإجابة الصحيحة (guess) المقدمة إذا لم تكونا False إذا كانتا متطابقتين و True حروف صغيرة. تعيد الدالة كذلك.

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

- **@app.route('/'):** ('/') الرئيسية للصفحة مسارًا ينشئ مسارًا لهذا المسار، يتم استدعاء الدالة **home** عند زيارة المستخدم لهذا المسار، يتم استدعاء الدالة **home.html** دالة تقوم بعرض صفحة **render_template**.

```
python
@app.route('/quiz', methods=['GET',
                             'POST'])
def quiz():
    score = None
    if request.method == 'POST':
        guess1 = request.form.get('guess1')
        guess2 = request.form.get('guess2')
        guess3 = request.form.get('guess3')

        score = 0
        if check_guess(guess1, "polar bear"):
            score += 1
        if check_guess(guess2, "cheetah"):
            score += 1
        if check_guess(guess3, "blue whale"):
            score += 1

    return render_template('quiz.html',
                           score=score)
```

- **@app.route('/quiz', methods=['GET', 'POST']):** هذا ويقبل الطلبات بنوعيهما ('/quiz') ديكتاتور ينشئ مسارًا لصفحة الاختبار GET و POST.

دالة تعالج منطق الاختبار: **quiz()**

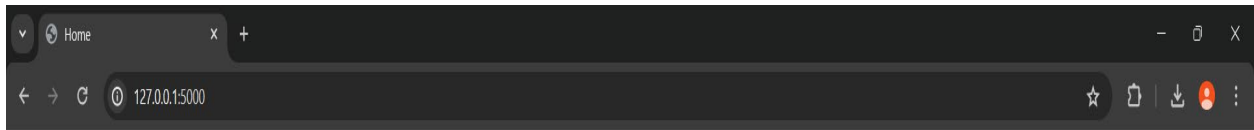
- **score:** None تمثل درجة الاختبار وتكون قيمتها مبدئيًا.
- **request.method == 'POST':** تفحص إذا كان نوع الطلب هو POST، مما يعني أن المستخدم أرسل النموذج POST يحصل على قيمة الحقل: **request.form.get('guess1')** من النموذج guess1.
- **score = 0:** يتم تعيين الدرجة إلى 0. ثم يتم فحص كل إجابة وزيادة الدرجة إذا كانت الإجابة **check_guess** باستخدام دالة صحيحة.
- **render_template('quiz.html', score=score):** تعرض مع تمرير الدرجة الحالية إلى القالب quiz.html صفحة

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

- **if name == 'main':** يضمن أن التطبيق يعمل فقط إذا تم تشغيل هذا الملف مباشرة، وليس استيراده كجزء من وحدة أخرى.
- **app.run(debug=True):** مع تمكين وضع التصحيح، Flask يبدأ خادم مما يوفر رسائل خطأ تفصيلية وإعادة تحميل تلقائي عند تغيير الكود.

مع صفحتين Flask هذا الكود ينشئ تطبيق ويب بسيط باستخدام: ملخص

- بسيطة HTML الصفحة الرئيسية ('/') تعرض صفحة
- تعرض نموذج اختبار يقبل الإجابات من ('/quiz') صفحة الاختبار المستخدم، يعالج الإجابات ويحسب الدرجة، ثم يعرض النتيجة على المستخدم.



Flask Website Home Quiz

Welcome to the Animal Quiz!

Test your knowledge about animals and see how much you know.

Take the Quiz

Activate Windows
Go to Settings to activate Windows.



Animal Quiz

Which bear lives at the North Pole?



Which is the fastest land animal?



Which is the largest animal?



Submit

Activate Windows
Go to Settings to activate Windows.