



الجمهورية العربية السورية

جامعة تشرين

كلية الهندسة الميكانيكية والكهربائية

قسم هندسة الاتصالات والإلكترونيات

السنة الخامسة

وظيفة البرمجة وإدارة الشبكات 2

إعداد الطلاب :

شيماء كيلاني

رہف قصيبة

إشراف :

د. مهند عيسى

العام الدراسي : 2023 - 2024

Question 1: Bank ATM Application with TCP Server/Client and Multi-threading

Project Description:

Build a TCP server and client Bank ATM application using Python. The server should handle multiple client connections simultaneously using multi-threading. The application should allow clients to connect, perform banking operations (such as check balance, deposit, and withdraw), and receive their updated account status upon completion.

كود server :

```
import socket
import threading
import time

host = 'localhost' # عنوان المضيف
port = 11111 # رقم المنفذ
accounts = {
    "123456789": {"balance": 1000, "pin": 1234},
    "987654321": {"balance": 5000, "pin": 4321},
}

def handle_client(client_socket):
    for a in accounts.keys():
        client_socket.send(a.encode())
        # استقبال البيانات من العميل
        data = client_socket.recv(1024).decode().strip()

        # تحليل البيانات وتنفيذ الطلب
        request = data.split()
        command = request[0]
        account_number = request[1]
        pin = request[2] if len(request) > 2 else None

        if command == "check_balance":
            if verify_account(account_number, pin):
                response = f"Your balance is: {accounts[account_number]['balance']}"
            else:
                response = "Invalid account number or PIN."

        elif command == "deposit":
            amount = float(request[3])
            if verify_account(account_number, pin):
                accounts[account_number]["balance"] += amount
                response = f"Deposited {amount:.2f}. Your new balance is {accounts[account_number]['balance']:.2f}"
            else:
                response = "Invalid account number or PIN."

        elif command == "withdraw":
            amount = float(request[3])
```

Ln: 1 Col: 0

```
server.py - C:\Users\ASUS\Desktop\server.py (3.12.4)
File Edit Format Run Options Window Help
amount = float(request[3])
if verify_account(account_number, pin) and accounts[account_number]["balance"] >= amount:
    accounts[account_number]["balance"] -= amount
    response = f"Withdrawn {amount:.2f}. Your new balance is: {accounts[account_number]['balance']:.2f}"
else:
    response = "Insufficient funds."

else:
    response = "Invalid command."

# إرسال الاستجابة إلى العميل
client_socket.sendall(response.encode("utf-8"))

# إغلاق اتصال العميل
client_socket.close()

def verify_account(account_number, pin):
    if account_number not in accounts:
        return False
    if pin is None or accounts[account_number]["pin"] != pin:
        return False
    return True

def start_server():
    server_socket=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind(('localhost', 11111))
    server_socket.listen(5) # عدد اتصالات العملاء المسموح بها في قائمة الانتظار

    while True:
        client_socket, address = server_socket.accept()
        print(f"[INFO] Connected to {address}")

        # إنشاء خيط جديد لكل عميل
        client_thread = threading.Thread(target=handle_client, args=(client_socket,))
        client_thread.start()

if __name__ == '__main__':
    print("[INFO] Starting server...")
    start_server()
```

Ln: 1 Col: 0

كود client :

```
client.py - C:\Users\ASUS\Desktop\client.py (3.12.4)
File Edit Format Run Options Window Help
import socket
import time

host = "0.0.0.0" # عنوان المضيف
port = 11111 # رقم المنفذ

def start_client():
    server_address = ('localhost', 11111)
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect(server_address)

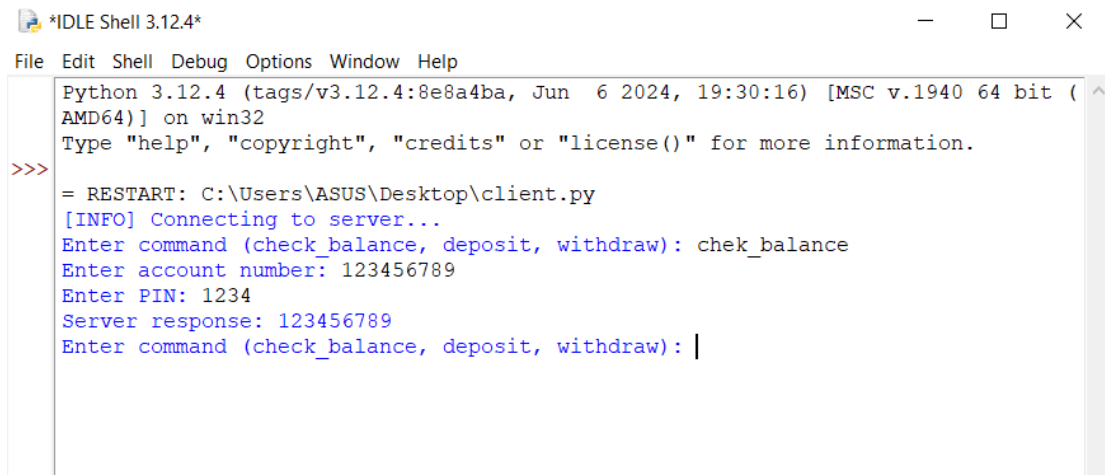
    while True:
        # إرسال طلب إلى الخادم
        command = input("Enter command (check_balance, deposit, withdraw): ")
        account_number = input("Enter account number: ")
        pin = int(input("Enter PIN: "))

        request = f"{command} {account_number} {pin}"
        client_socket.sendall(request.encode("utf-8"))

        # استقبال الاستجابة من الخادم
        response = client_socket.recv(1024).decode()
        print(f"Server response: {response}")

if __name__ == '__main__':
    print("[INFO] Connecting to server...")
    start_client()
```

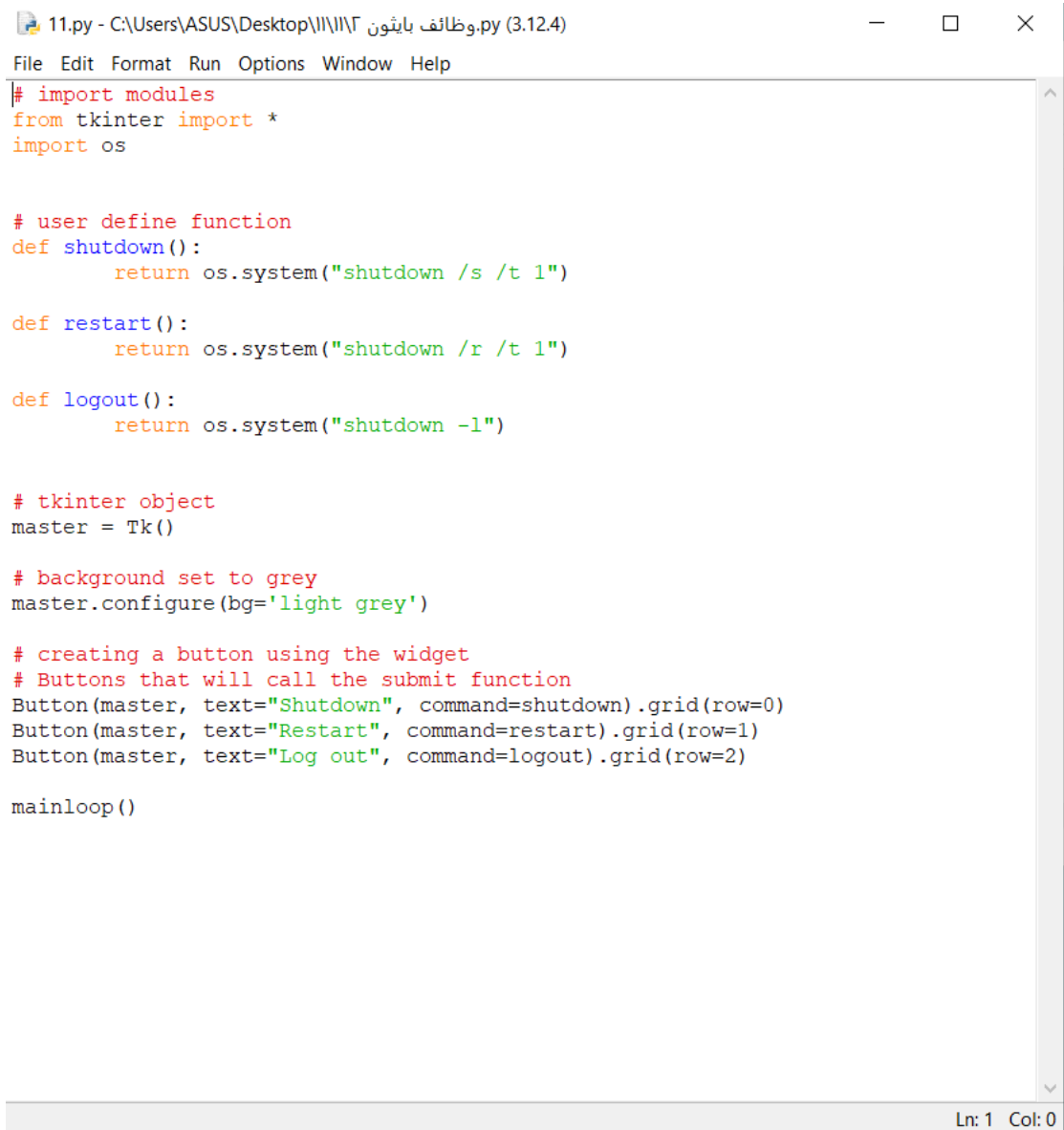
الخرج :



```
*IDLE Shell 3.12.4*
File Edit Shell Debug Options Window Help
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun 6 2024, 19:30:16) [MSC v.1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\ASUS\Desktop\client.py
[INFO] Connecting to server...
Enter command (check_balance, deposit, withdraw): chek_balance
Enter account number: 123456789
Enter PIN: 1234
Server response: 123456789
Enter command (check_balance, deposit, withdraw): |
```

Question 2: Simple Website Project with Python Flask Framework (you have choice to use Django or any Other Deferent Useful Python Project “from provide Project Links”)

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



The image shows a screenshot of a Python IDE window titled "11.py - C:\Users\ASUS\Desktop\11\11\وظائف بايثون.py (3.12.4)". The window has a menu bar with "File", "Edit", "Format", "Run", "Options", "Window", and "Help". The code editor contains the following Python code:

```
# import modules
from tkinter import *
import os

# user define function
def shutdown():
    return os.system("shutdown /s /t 1")

def restart():
    return os.system("shutdown /r /t 1")

def logout():
    return os.system("shutdown -l")

# tkinter object
master = Tk()

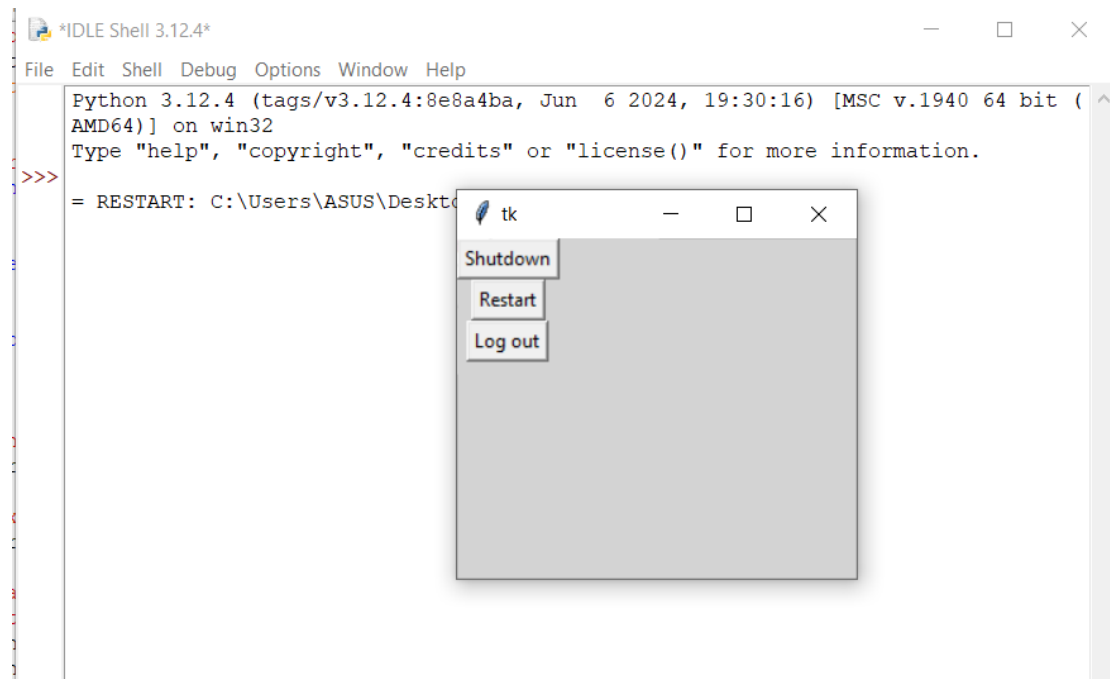
# background set to grey
master.configure(bg='light grey')

# creating a button using the widget
# Buttons that will call the submit function
Button(master, text="Shutdown", command=shutdown).grid(row=0)
Button(master, text="Restart", command=restart).grid(row=1)
Button(master, text="Log out", command=logout).grid(row=2)

mainloop()
```

The status bar at the bottom right indicates "Ln: 1 Col: 0".

الخرج :



GUI to Shutdown, Restart and Logout from the PC using Python

شرح الكود :

هذا الكود مكتوب بلغة Python ويستخدم مكتبة Tkinter لإنشاء واجهة مستخدم رسومية بسيطة للتحكم بنظام التشغيل.

استيراد المكتبات:

- `from tkinter import *` : يتم استيراد جميع عناصر واجهة المستخدم المستخدمة الرسومية من مكتبة Tkinter.
- `import os` : يتم استيراد مكتبة os التي توفر وظائف للتفاعل مع نظام التشغيل.

تعريف الدوال: (User Defined Functions)

- shutdown () : تقوم هذه الدالة بإيقاف تشغيل النظام بعد تأخير قدره ثانية واحدة. (/t 1)
- تستخدم الدالة وظيفة os.system من مكتبة os لتنفيذ أمر إيقاف التشغيل. (shutdown /s /t 1)
- restart () : تقوم هذه الدالة بإعادة تشغيل النظام بعد تأخير قدره ثانية واحدة. (/t 1)
- تستخدم الدالة وظيفة os.system من مكتبة os لتنفيذ أمر إعادة التشغيل. (shutdown /r /t 1)
- logout () : تقوم هذه الدالة بتسجيل خروج المستخدم الحالي من الجلسة.
- تستخدم الدالة وظيفة os.system من مكتبة os لتنفيذ أمر تسجيل الخروج. (shutdown -l)

إنشاء نافذة Tkinter الرئيسية:

- master = Tk() : يتم إنشاء نافذة رئيسية باسم master باستخدام مكتبة Tkinter.
- master.configure(bg='light grey') : يتم ضبط لون خلفية النافذة إلى رمادي فاتح.

إنشاء عناصر واجهة المستخدم (Buttons):

- يتم إنشاء ثلاثة عناصر من نوع Button لعرض خيارات إيقاف التشغيل وإعادة التشغيل وتسجيل الخروج.
- يتم ضبط خاصية text لكل زر لتحديد النص المعروض عليه (Shutdown, Restart, Log out).
- يتم ضبط خاصية command لكل زر لتحديد الدالة التي يتم تنفيذها عند الضغط عليه. (shutdown, restart, logout)
- يتم استخدام طريقة grid لترتيب الأزرار في صفوف (row=0, row=1, row=2).

تشغيل واجهة المستخدم:

- .mainloop () : يتم تنفيذ حلقة Tkinter الرئيسية، مما يسمح للمستخدم بالتفاعل مع واجهة المستخدم الرسومية.