



الجمهورية العربية السورية

جامعة تشرين

كلية الهندسة الميكانيكية والكهربائية

قسم هندسة الاتصالات والإلكترونيات

السنة الخامسة

وظيفة البرمجة وإدارة الشبكات 2

إعداد الطلاب :

حلا حسين هولا

نهاد عمار محمود

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    response = f"Withdrawn {amount:.2f}.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)

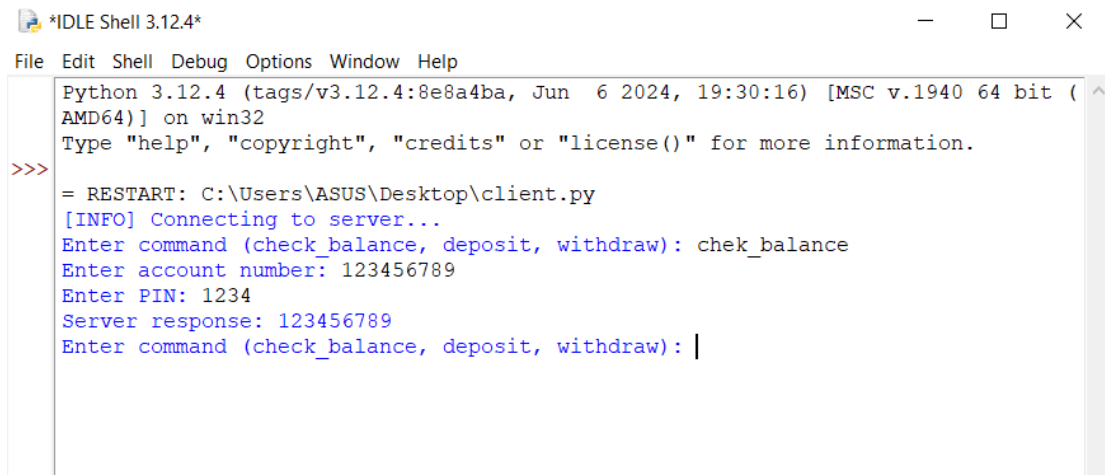
    while True:
        # إرسال طلب إلى الخادم
        command = input("Enter command (check_balance, deposit, withdraw): ")
        account_number = input("Enter account number: ")
        pin = int(input("Enter PIN: "))

        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.

```
6.py - C:\Users\ASUS\Desktop\٦\٦٢ وظائف بايثون.py (3.12.4)
File Edit Format Run Options Window Help
from tkinter import Label, Tk
import time
app_window = Tk()
app_window.title("Digital Clock")
app_window.geometry("420x150")
app_window.resizable(1,1)

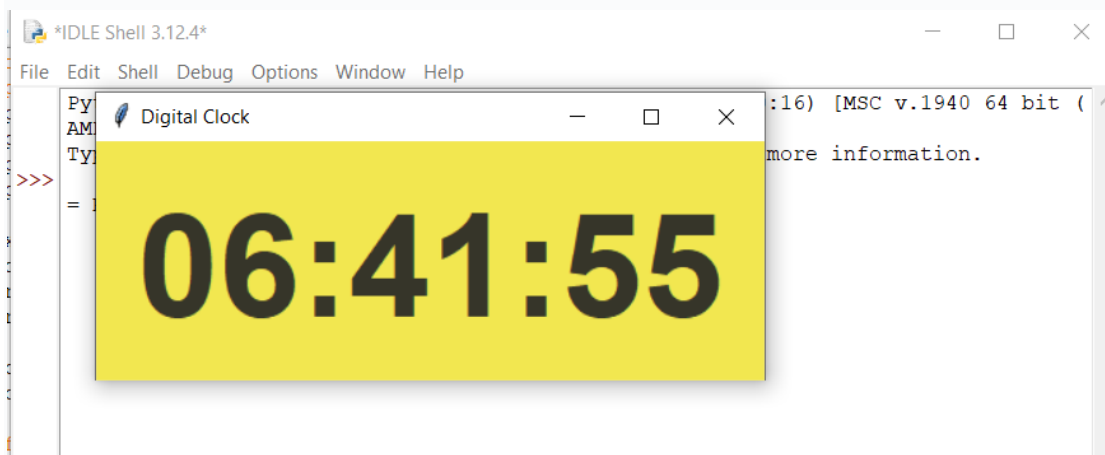
text_font= ("Boulder", 68, 'bold')
background = "#f2e750"
foreground= "#363529"
border_width = 25

label = Label(app_window, font=text_font, bg=background, fg=foreground, bd=border_width)
label.grid(row=0, column=1)

def digital_clock():
    time_live = time.strftime("%H:%M:%S")
    label.config(text=time_live)
    label.after(200, digital_clock)

digital_clock()
app_window.mainloop()
```

الخرج :



Digital Clock with Python

شرح الكود :

هذا الكود مكتوب بلغة Python ويستخدم مكتبة Tkinter لإنشاء واجهة مستخدم رسومية بسيطة تعرض ساعة رقمية.

استيراد المكتبات:

- `from tkinter import Label, Tk:` يتم استيراد فئتي `Label` لإنشاء عنصر نصي و `Tk` لإنشاء النافذة الرئيسية.
- `import time:` يتم استيراد مكتبة `time` للوصول إلى وظائف التعامل مع الوقت.

إنشاء النافذة الرئيسية:

- `app_window = Tk():` يتم إنشاء نافذة رئيسية باستخدام `Tk()`.
- **تخصيص النافذة:**
 - `app_window.title("Digital Clock"):` ضبط عنوان النافذة إلى "Digital Clock".
 - `app_window.geometry("420x150"):` عرض وارتفاع النافذة إلى 420 و 150 وحدة على التوالي.
 - `app_window.resizable(1,1):` (يمكن تكبيرها وتصغيرها) السماح بتغيير حجم النافذة.

تحديد تنسيق النص والواجهة:

- `text_font= ("Boulder", 68, 'bold'):` تعريف متغير `text_font` لتخزين تنسيق الخط المستخدم لعرض الوقت.
- `background = "#f2e750":` `background` لتخزين لون الخلفية (باستخدام رمز لوني).
- `foreground= "#363529":` `foreground` لتخزين لون النص (باستخدام رمز لوني).
- `border_width = 25:` `border_width` تعريف متغير `border_width` لتخزين عرض حواف العنصر النصي.

إنشاء العنصر النصي: (Label)

```
label = Label(app_window,
font=text_font, bg=background,
fg=foreground, bd=border_width):
عنصر نصي من نوع Label ضمن النافذة الرئيسية.
font: ضبط تنسيق الخط باستخدام المتغير text_font.
bg: ضبط لون الخلفية باستخدام المتغير background.
fg: ضبط لون النص باستخدام المتغير foreground.
bd: ضبط عرض حواف العنصر النصي باستخدام المتغير border_width.
label.grid(row=0, column=1): وضع العنصر
النصي label على النافذة الرئيسية باستخدام grid.
row=0: يوضع العنصر النصي في الصف الأول (0).
column=1: يوضع العنصر النصي في العمود الثاني (1).
```

تعريف دالة تحديث الوقت:

```
def digital_clock():
digital_clock.
خطوات الدالة:
time_live = 1
time.strftime("%H:%M:%S"):
الحالي بتنسيق "ساعات:دقائق:ثواني" باستخدام
time.strftime("%H:%M:%S").
2. label.config(text=time_live): يتم تحديث
نص العنصر النصي label بالوقت المستخرج.
3. label.after(200, digital_clock): يتم
استخدام after لتأخير تنفيذ الدالة digital_clock
أخرى لمدة 200 مللي ثانية (أي تحديث الوقت كل 200 مللي ثانية).
```

تشغيل الساعة الرقمية:

```
digital_clock(): يتم استدعاء دالة digital_clock
للمرة الأولى لعرض الوقت الابتدائي.
```

• `app_window.mainloop()`: Tkinter تشغيل حلقة الرئيسية، مما يسمح بتحديث العرض باستمرار وفقاً للتأخير المحدد في الدالة `digital_clock`.