



الجمهورية العربية السورية

جامعة تشرين

كلية الهندسة الميكانيكية والكهربائية

قسم هندسة الاتصالات والإلكترونيات

السنة الخامسة: وظيفة 2 برمجة الشبكات

## Second Network Programming Homework

إعداد:

شازان جلال حسن

زينب أحمد سعود

الرقم الجامعي: 2453

الرقم الجامعي : 2890

بإشراف : د. مهند عيسى

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

### **Server tcp:**

```
import socket
import threading

# Bank account details
accounts = {
    '1111':10000,
    '2222':5000,
    '3333':6000,
}

def handle_client(client_socket):
    account_number = client_socket.recv(1024).decode()
    if account_number in accounts:
        client_socket.send(b"Welcome! You have connected to the bank server.")
    else:
        client_socket.send(b"Invalid account number. Connection terminated.")
        client_socket.close()
        return

    while True:
        option = client_socket.recv(1024).decode()
        balance = accounts[account_number]
        if option == '1':
            client_socket.send(f"Your current balance is: {balance}".encode())
        elif option == '2':
            amount = int(client_socket.recv(1024).decode())
            balance += amount
            client_socket.send(f"Deposit successful. Your new balance is: {balance}".encode())
        elif option == '3':
            amount = int(client_socket.recv(1024).decode())
            if balance >= amount:
                balance -= amount
                client_socket.send(f"Withdrawal successful. Your new balance is: {balance}".encode())
            else:
                client_socket.send("Insufficient funds. Withdrawal failed.".encode())
        else:
            break
```

```

client_socket.close()

def start_server():
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind(('127.0.0.1', 4444))
    server.listen(5)
    print("[+] tcp server is waiting for new connection...")

    while True:
        client_socket, address = server.accept()
        print(f"[*] new client is already accepted:{address}")
        client_thread = threading.Thread(target=handle_client,
args=(client_socket,))
        client_thread.start()

start_server()

```

## client tcp:

```

import socket

def main():
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect(('127.0.0.1', 4444))

    account_number = input("Enter your account number: ")
    client.send(account_number.encode())

    print(client.recv(1024).decode())

    while True:
        print("Options:")
        print("1. Check Balance")
        print("2. Deposit")
        print("3. Withdraw")
        print("4. Exit")
        option = input("Enter option: ")

        if option == '4':
            break

        client.send(option.encode())

        if option == '1':
            print(client.recv(1024).decode())

```

```

elif option == '2' or option == '3':
    amount = input("Enter amount: ")
    client.send(amount.encode())
    print(client.recv(1024).decode())

client.close()

if __name__ == "__main__":
    main()

```

## الشرح:

أولا تم إنشاء كود السيرفر باستدعاء مكتبة **socket** و مكتبة **threading** :

تم تعريف قاموس يحوي أرقام 3 حسابات بنك مع الرصيد لكل حساب ثم عرفنا تابع `handle_client` يأخذ بارامتر يعبر عن `client` المتصل عمله : يقوم باستقبال رسالة من ال `client` تحوي على رقم حسابه بعد ان قام المستخدم بإدخاله على الشاشة و يفك ترميز الرسالة باستخدام `decode()` , يدخل رقم المستخدم في حلقة لنتخبره اذا كان موجود في القاموس المخزن سابقا أم لا. ذلك باستخدام حلقة `if` , اذا كان موجود سيرسل السيرفر رسالة الى الكلينت تتضمن ترحيب و اذا لم يكن موجود يخبره بذلك و إنهاء الاتصال.

بعد التأكد من رقم حساب المستخدم نغعل حلقة لانهاية باستخدام `while true` تستقبل رسالة الكلينت متضمنة رقم العملية او الخدمة التي يريدھا و نعرف متغير يعبر عن مبلغ الحساب المطلوب `balance` .

باستخدام الشرط `if` يقدم السيرفر الخدمات للكلينت :

اذا كانت تحوي الرقم 1 تعني الاستعلام عن الرصيد فيتم ارسال رسالة من السيرفر الى الكلينت تتضمن معلومات حول رصيده و اذا كانت تحوي الرقم 2 تعني انه يريد الأيداع في حسابه يدخل المستخدم الكمية و يرسلها الكلينت الى السيرفر بعد استقبالها بشكل `int` من قبل السيرفر يتم إضافتها الى الرصيد و ارسال رسالة للكلينت تتضمن اخباره بنجاح العملية و رصيده الجديد .

و اذا كانت الرسالة تحوي الرقم 3 يعني ان المستخدم يريد السحب فيرسل الكلينت الكمية المراد سحبها الى السيرفر بعد استقبالها بشكل `int` يقوم السيرفر باختبارها باستخدام `if` اذا كان المبلغ الموجود يكفي للسحب ام لا اذا كان لا يكفي يتم اخبار الكلينت بارسال رسالة له و اذا كانت تكفي يتم سحب المبلغ و ارسالة رسالة تتضمن إتمام العملية بنجاح و اخباره برصيده الحالي بعد السحب.

اذا كانت رسالة الكلينت تحوي أي رقم ماعدا الأرقام السابقة فهذا يعني الخروج من الخدمة

و اغلاق اتصال الكلينت.

نعرف تابع اخر بقصد بناء سيرفر server\_client : يتم انشاء server tcp باستخدام التعليمة:

```
Server = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
```

و إعطائه ip محلي 127.0.0.1 و port للعمل عليه.

ثم تحديد عدد client التي يمكنها الانتظار من خلال التعليمة : server.listen()

و طباعة انه في حالة تنصت على البورت و ثم تفعيل حلقة لانهاية من اجل حالة القبول للاتصال و طباعة رسالة انه تم الاتصال بكلينت جديد و عنوانه....

يعيد بارامترين في الحلقة الأول socket من اجل الكلينت المتصل و الثاني عنوان هذا الكلينت و يتم بناء socket لكل كلينت متصل

أيضا تم بناء threading من التعليمة :

```
Client_thread = threading.thread(target,args)
```

و إعطاء اول بارمتر التابع handle\_client و البارامتر الثاني السوكت التابع للكلينت المتصل و تشغيل ال thread و الغاية منه تمكين السيرفر من تخدم عدة كلينت بنفس الوقت من خلال إنشاء thread لكل كلينت.

و بالنهاية استدعاء التابع start\_server() لبدء العمل.

أما كود client :

استدعاء مكتبة socket و تعريف تابع main() يحوي أولا إنشاء client tcp من خلال التعليمة :

```
client = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
```

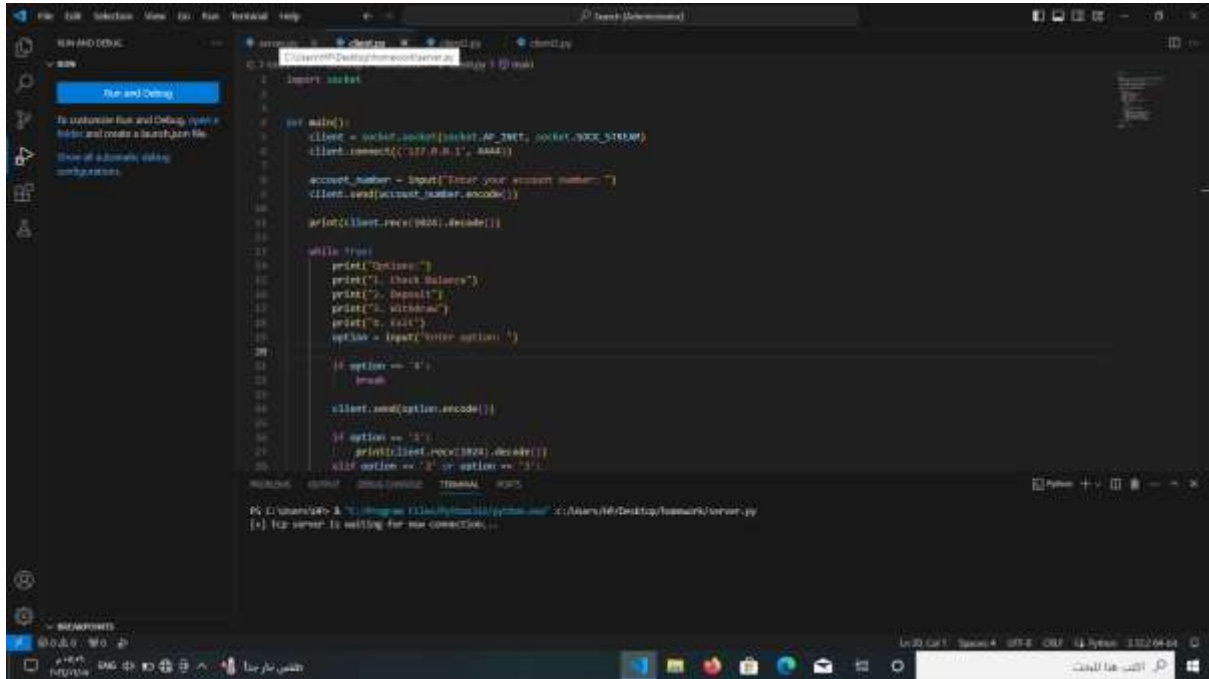
و اتصاله على عنوان السيرفر و البورت الذي يعمل عليه السيرفر , ثم الطلب من المستخدم إدخال رقم الحساب و ارساله الى السيرفر لاختباره ثم يظهر له الخدمات التي يوفرها و عندما يختار رقم الخدمة المرادة يرسلها للسيرفر لمعالجتها و تقديمها.

ثم يطبع الرسالة المستقبلية من السيرفر التي تحوي الخدمة المطلوبة و ثم إغلاق الاتصال ثم استدعت التابع main()

مع إمكانية اتصال عدة client نجعل الكود نفسه لعدة client و نحاول الاتصال من خلال تشغيل السيرفر أولا ثم ال client

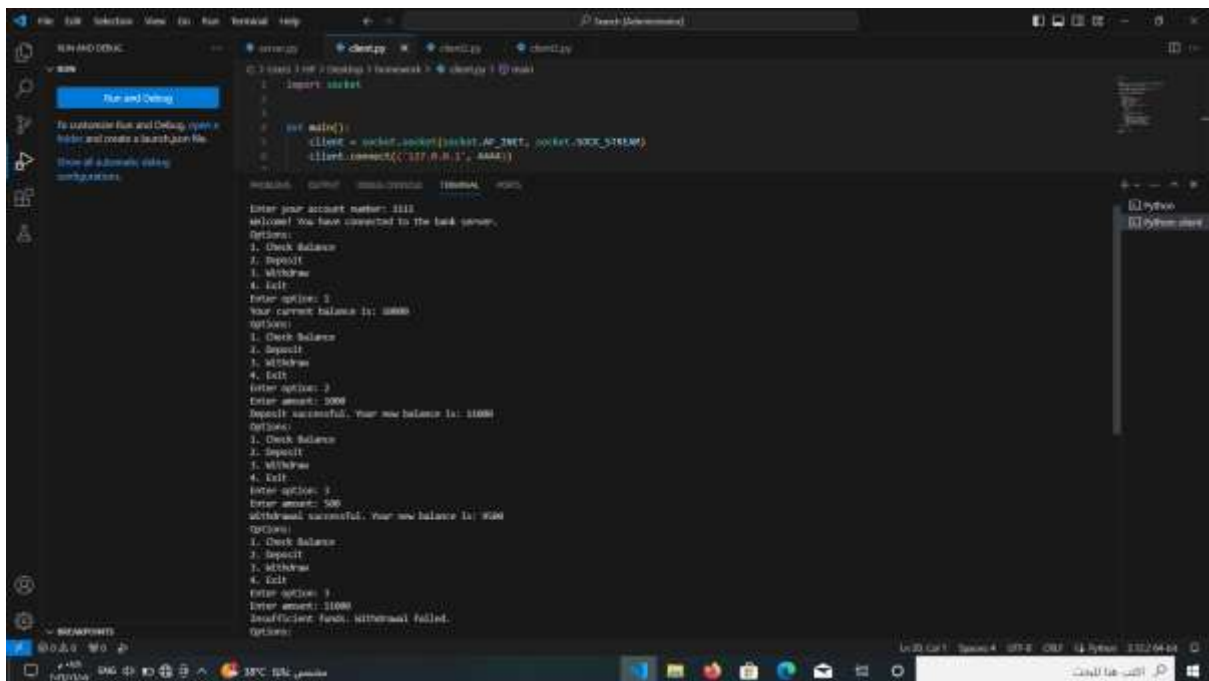
هنا لقطات شاشة توضح التنفيذ:

أولا عند تشغيل السيرفر :



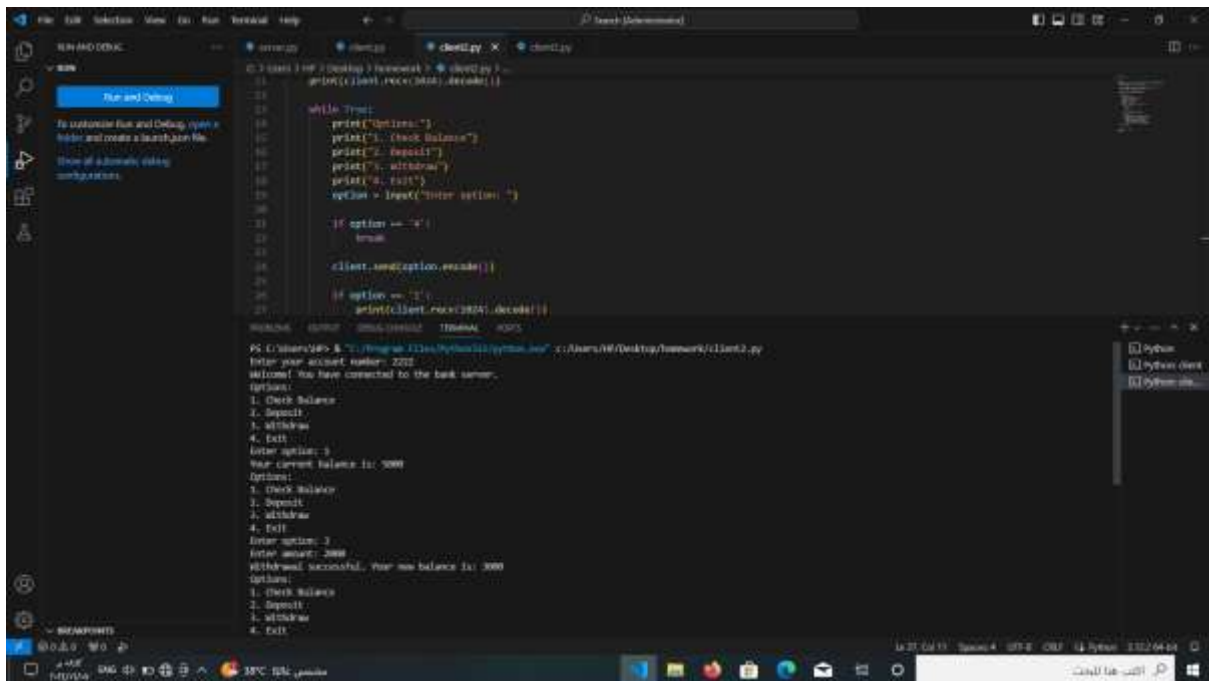
```
1 import socket
2
3 def main():
4     client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
5     client.connect(('127.0.0.1', 5555))
6
7     account_number = input("Enter your account number: ")
8     client.send(account_number.encode())
9
10    print(client.recv(1024).decode())
11
12    while True:
13        print("Options:")
14        print("1. Check Balance")
15        print("2. Deposit")
16        print("3. Withdraw")
17        print("4. Exit")
18        option = input("Enter option: ")
19
20        if option == '1':
21            break
22
23        client.send(option.encode())
24
25        if option == '2':
26            amount = input("Enter amount: ")
27            client.send(amount.encode())
28            print(client.recv(1024).decode())
29        elif option == '3' or option == '4':
30            break
31
32 if __name__ == '__main__':
33     main()
```

ثم عند تشغيل أول client:



```
1 import socket
2
3 def main():
4     client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
5     client.connect(('127.0.0.1', 5555))
6
7     account_number = input("Enter your account number: ")
8     client.send(account_number.encode())
9
10    print(client.recv(1024).decode())
11
12    while True:
13        print("Options:")
14        print("1. Check Balance")
15        print("2. Deposit")
16        print("3. Withdraw")
17        print("4. Exit")
18        option = input("Enter option: ")
19
20        if option == '1':
21            break
22
23        client.send(option.encode())
24
25        if option == '2':
26            amount = input("Enter amount: ")
27            client.send(amount.encode())
28            print(client.recv(1024).decode())
29        elif option == '3' or option == '4':
30            break
31
32 if __name__ == '__main__':
33     main()
```

عند تشغيل client 2 :



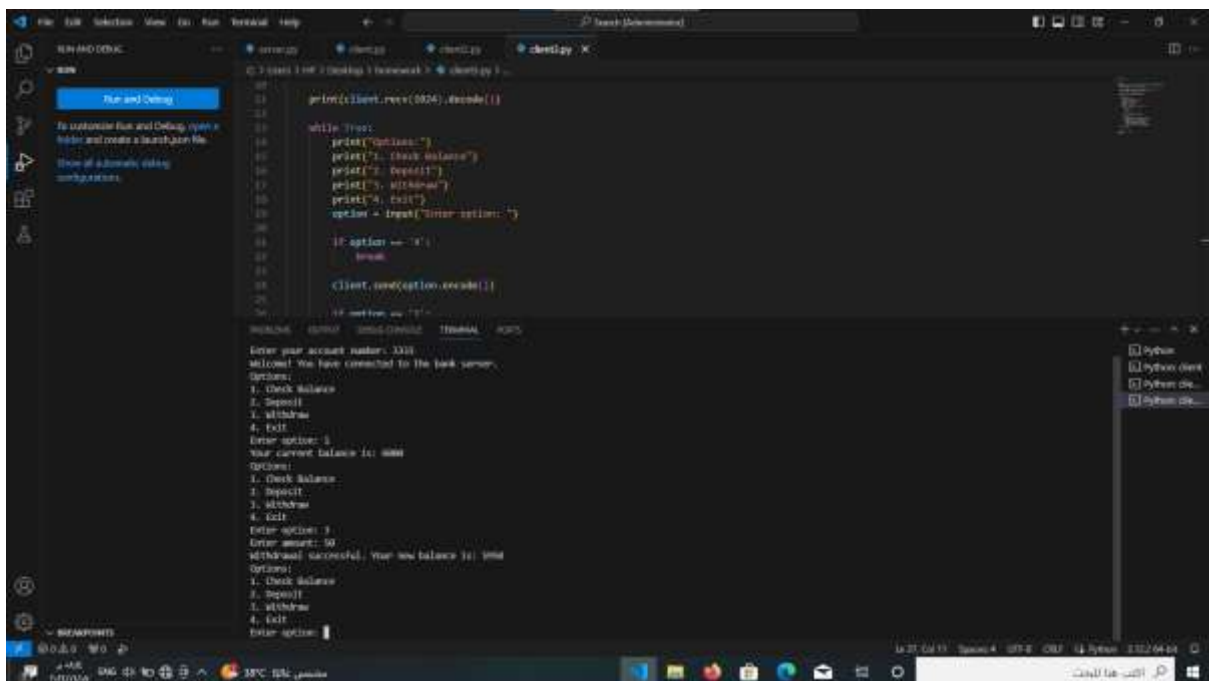
The screenshot shows a VS Code editor with a file named `client2.py` open. The code is a Python script that connects to a bank server and interacts with it. The terminal output shows the script running successfully and displaying a menu of options to the user.

```
1 # client 2 for 2 Desktop 3 framework 4 client.py 5
2 import socket, pickle, pickletools
3
4 while True:
5     print("Options:")
6     print("1. Check Balance")
7     print("2. Deposit")
8     print("3. Withdraw")
9     print("4. Exit")
10    option = input("Enter option: ")
11
12    if option == "1":
13        break
14
15    client.send(option.encode())
16
17    if option == "2":
18        print(client.recv(1024).decode())
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

Terminal Output:

```
K6: C:\Users\MOHAMED > python client2.py
Enter your account number: 2202
Welcome! You have connected to the bank server.
Options:
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter option: 1
Your current balance is: 5000
Options:
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter option: 2
Enter amount: 2000
Withdrawal successful. Your new balance is: 3000
Options:
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter option: 3
```

و تنفيذ client 3 :



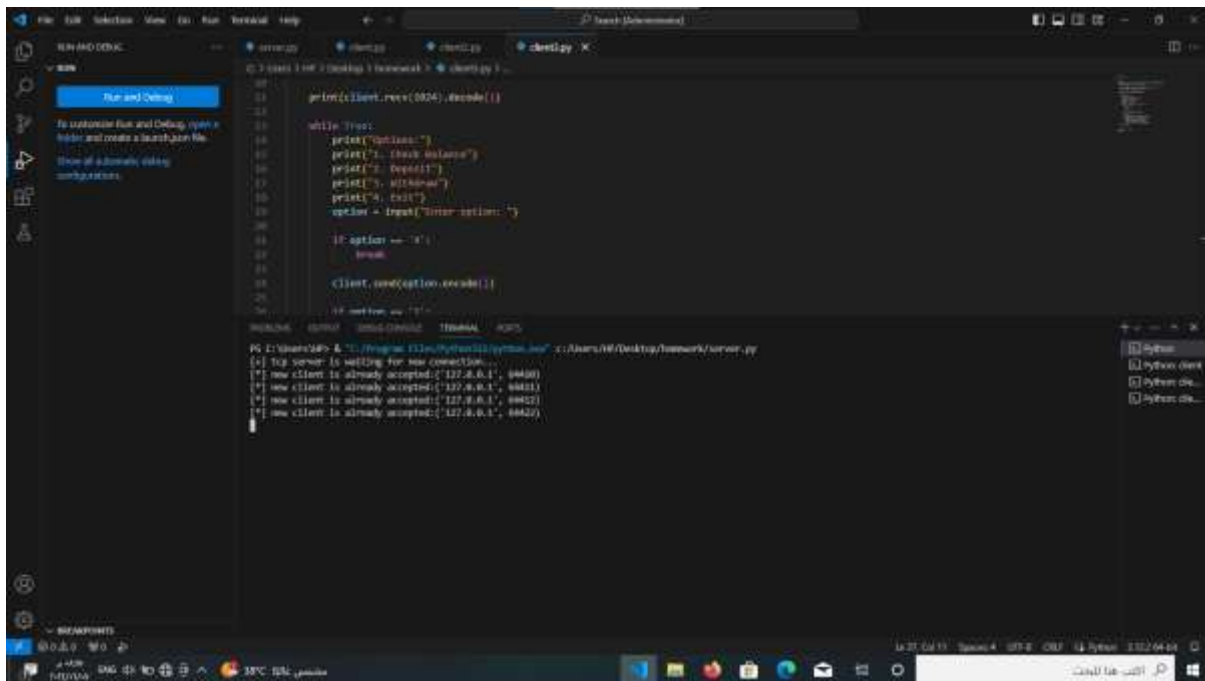
The screenshot shows a VS Code editor with a file named `client3.py` open. The code is a Python script that connects to a bank server and interacts with it. The terminal output shows the script running successfully and displaying a menu of options to the user.

```
1 # client 3 for 3 Desktop 4 framework 5 client.py 6
2 import socket, pickle, pickletools
3
4 while True:
5     print("Options:")
6     print("1. Check Balance")
7     print("2. Deposit")
8     print("3. Withdraw")
9     print("4. Exit")
10    option = input("Enter option: ")
11
12    if option == "1":
13        break
14
15    client.send(option.encode())
16
17    if option == "2":
18        print(client.recv(1024).decode())
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

Terminal Output:

```
Enter your account number: 2202
Welcome! You have connected to the bank server.
Options:
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter option: 1
Your current balance is: 5000
Options:
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter option: 2
Enter amount: 10
Withdrawal successful. Your new balance is: 4990
Options:
1. Check Balance
2. Deposit
3. Withdraw
4. Exit
Enter option: 3
```

وال server :



## Project:

### Age Calculator using Python:

```
import datetime
def ageCalculator(y , m , d):
    today = datetime.datetime.now().date()
    dob = datetime.date(y , m , d)
    age = int((today - dob).days/ 365.25)
    print("your age is:",age)
while True:
    y = int(input("Enter the year:"))
    m = int(input("Enter the month:"))
    d = int(input("Enter the day:"))
    ageCalculator(y , m , d)
```



## الشرح :

تم إنشاء تطبيق حاسبة العمر باستخدام بايثون:

تم استدعاء مكتبة datetime التي توفر وظائف للعمل مع التاريخ و الوقت .

تم تعريف الدالة او التابع ageCalculator يأخذ 3 متغيرات السنة و الشهر و اليوم و ذلك لتمثيل تاريخ الميلاد من أجل حساب العمر.

تم استخدام التعليمة التالية :

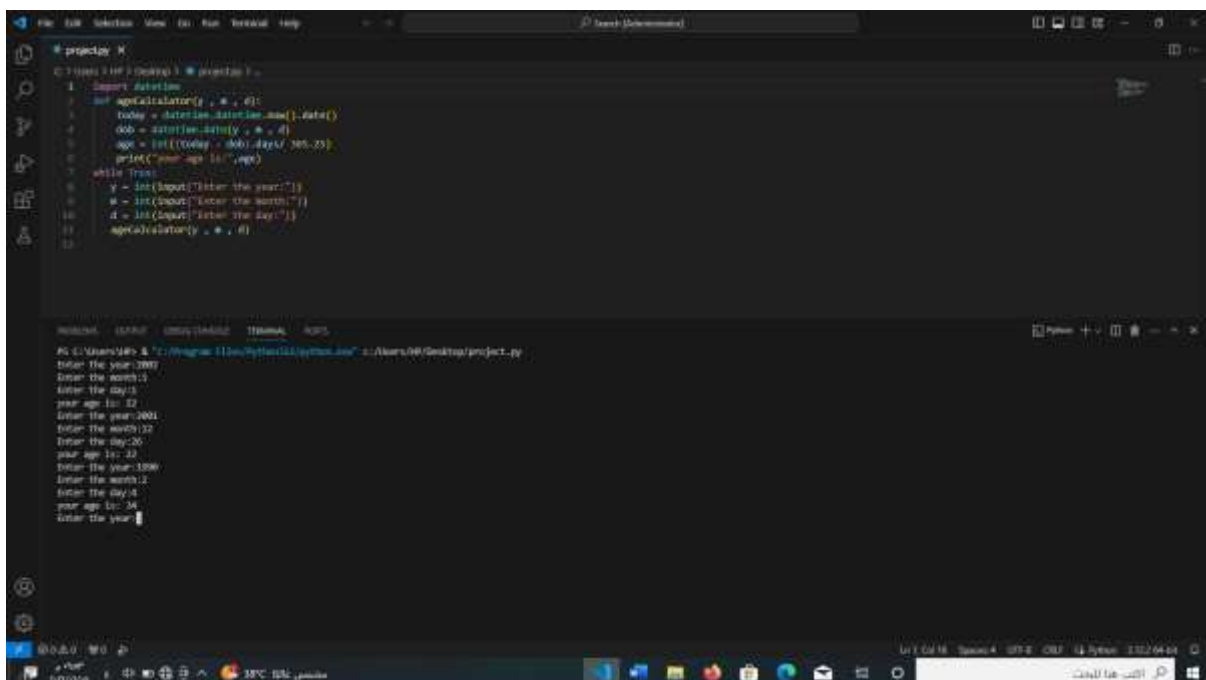
`Datetime.datetime.now().date()`

و تخزينها ضمن متغير و ذلك لحساب التاريخ الحالي دون الساعة ثم إنشاء تاريخ الميلاد باستخدام التعليمة : `Datetime.date(y,m,d)`

و تم حساب العمر من خلال معادلة رياضية و تخزينها في متغير بشكل int عدد صحيح , تم طرح تاريخ الميلاد من التاريخ الحالي فيعطي فارق الأيام و للتحويل من الأيام الى السنوات يتم قسمة عدد الأيام على 365.25 لأخذ السنوات الكبيسة بعين الاعتبار (كل أربع سنوات يكون هناك يوم إضافي).

ثم طباعة العمر الناتج و تم تعريف حلقة لانهائية من اجل المستخدم تطلب منه إدخال سنة ميلاده و اليوم و الشهر و استدعاء التابع السابق من اجل إدخال المستخدم.

لقطة الشاشة توضح التنفيذ:



```
File Edit Shell View Run Terminal Help
Search (Automatically)

project M
C:\Users\user\Documents\project M
1. Import datetime
2. def ageCalculator(y, m, d):
3.     today = datetime.datetime.now().date()
4.     dob = datetime.datetime(y, m, d)
5.     age = int((today - dob).days / 365.25)
6.     print("your age is: ",age)
7. while True:
8.     y = int(input("Enter the year:"))
9.     m = int(input("Enter the month:"))
10.    d = int(input("Enter the day:"))
11.    ageCalculator(y, m, d)
```

```
Python 3.12.0 Shell
C:\Users\user> python C:\Users\user\Documents\project M\project M.py
Enter the year:2007
Enter the month:3
Enter the day:5
your age is: 17
Enter the year:2001
Enter the month:12
Enter the day:20
your age is: 23
Enter the year:1990
Enter the month:1
Enter the day:8
your age is: 34
Enter the year:
```

