



الجمهورية العربية السورية

جامعة تشرين

كلية الهندسة الميكانيكية والكهربائية

قسم هندسة الاتصالات والإلكترونيات

السنة الخامسة

وظيفة البرمجة وإدارة الشبكات 2

إعداد الطلاب :

مريانا حسن

نغم جديد

جلنار بدور

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    response = f"Withdrawn {amount:.2f}.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

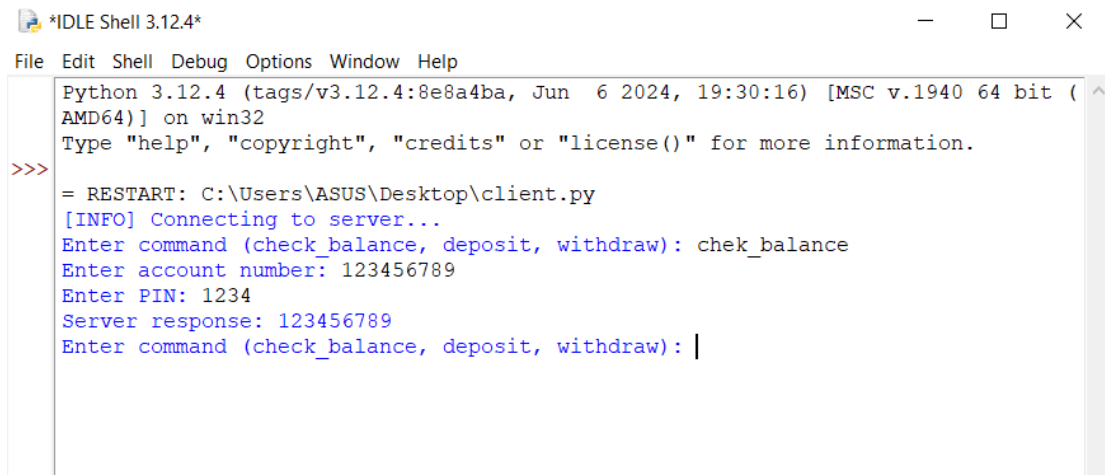
    while True:
        # إرسال طلب إلى الخادم
        command = input("Enter command (check_balance, deposit, withdraw): ")
        account_number = input("Enter account number: ")
        pin = int(input("Enter PIN: "))

        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()
```

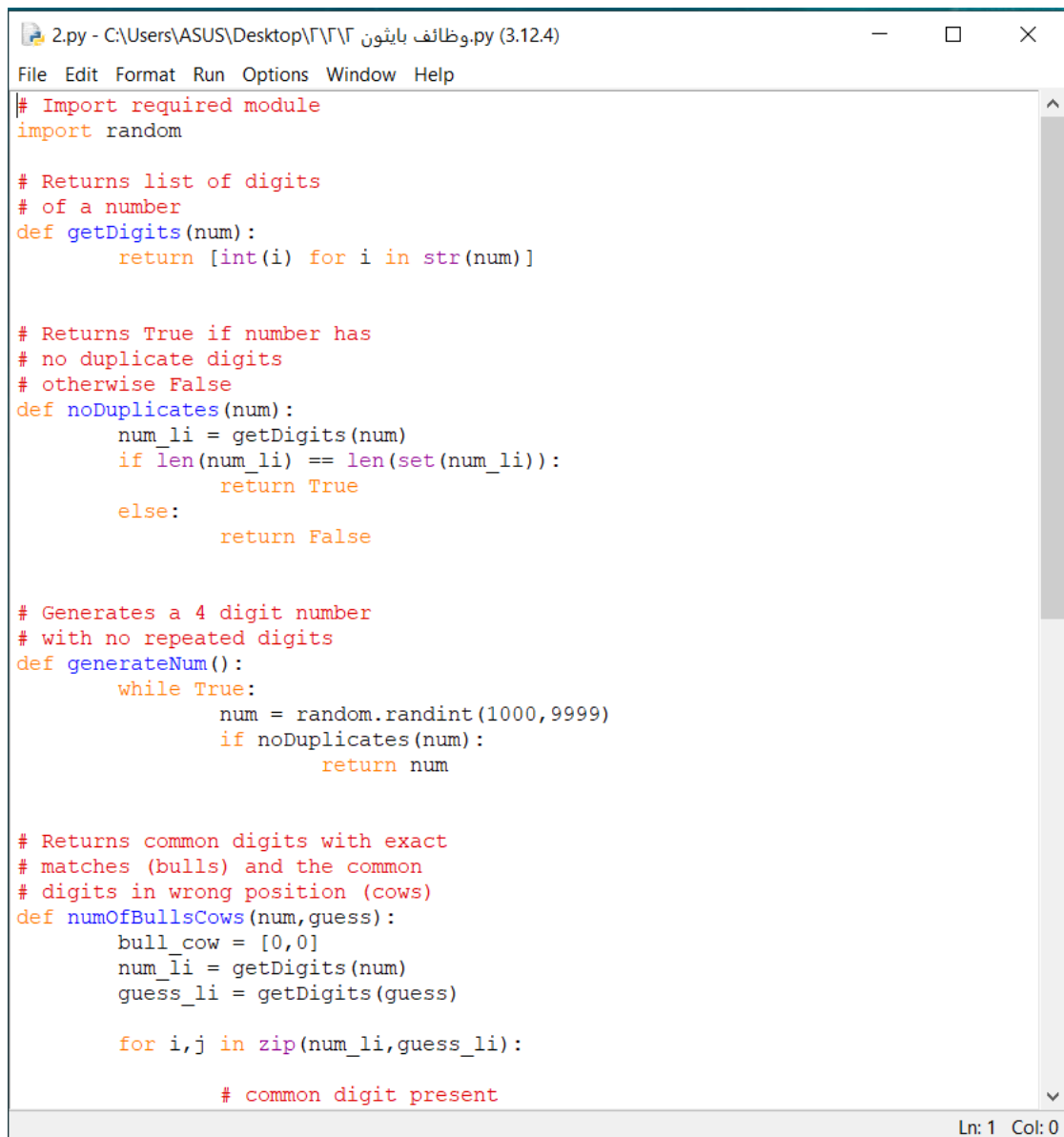
الخرج :



```
*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.



```
2.py - C:\Users\ASUS\Desktop\وظائف بايثون.py (3.12.4)
File Edit Format Run Options Window Help
# Import required module
import random

# Returns list of digits
# of a number
def getDigits(num):
    return [int(i) for i in str(num)]

# Returns True if number has
# no duplicate digits
# otherwise False
def noDuplicates(num):
    num_li = getDigits(num)
    if len(num_li) == len(set(num_li)):
        return True
    else:
        return False

# Generates a 4 digit number
# with no repeated digits
def generateNum():
    while True:
        num = random.randint(1000,9999)
        if noDuplicates(num):
            return num

# Returns common digits with exact
# matches (bulls) and the common
# digits in wrong position (cows)
def numOfBullsCows(num,guess):
    bull_cow = [0,0]
    num_li = getDigits(num)
    guess_li = getDigits(guess)

    for i,j in zip(num_li,guess_li):

        # common digit present
```

Ln: 1 Col: 0

```
2.py - C:\Users\ASUS\Desktop\وظائف بايثون.py (3.12.4)
File Edit Format Run Options Window Help
    # common digit present
    if j in num_li:

        # common digit exact match
        if j == i:
            bull_cow[0] += 1

        # common digit match but in wrong position
        else:
            bull_cow[1] += 1

    return bull_cow

# Secret Code
num = generateNum()
tries =int(input('Enter number of tries: '))

# Play game until correct guess
# or till no tries left
while tries > 0:
    guess = int(input("Enter your guess: "))

    if not noDuplicates(guess):
        print("Number should not have repeated digits. Try again.")
        continue
    if guess < 1000 or guess > 9999:
        print("Enter 4 digit number only. Try again.")
        continue

    bull_cow = numOfBullsCows(num,guess)
    print(f"{bull_cow[0]} bulls, {bull_cow[1]} cows")
    tries -=1

    if bull_cow[0] == 4:
        print("You guessed right!")
        break
else:
    print(f"You ran out of tries. Number was {num}")

Ln: 1 Col: 0
```

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\وظائف بايثون.py
Enter number of tries: 5
Enter your guess: 4
Enter 4 digit number only. Try again.
Enter your guess: 5
Enter 4 digit number only. Try again.
Enter your guess: |
```

Cows and Bulls game

بقرة وثيران هي لعبة كسر شفرة ورق وقلم يتم لعبها عادة بين لاعبين. يحاول اللاعب تخمين رقم رمز سري يختاره اللاعب الثاني. القواعد هي كما يلي:

- يقوم لاعب بإنشاء رمز سري، وعادة ما يكون رقمًا مكونًا من 4 خانات. يجب ألا يتكرر أي رقم في هذا الرمز.
- يقوم لاعب آخر بتخمين (رقم مكون من 4 خانات) لكسر الرقم السري. بعد التخمين، سيتم تقديم تلميحين: بقرة وثور.
- تشير الثيران إلى عدد الأرقام الصحيحة في الموضع الصحيح، وتشير الأبقار إلى عدد الأرقام الصحيحة في الموضع الخاطئ. على سبيل المثال، إذا كان الرمز السري هو 1234 وكان الرقم المخمين هو 1246، فلدينا 2 ثور (للمطابقات الدقيقة للأرقام 1 و 2) و 1 بقرة (لمطابقة الرقم 4 في الموضع الخاطئ).
- يستمر اللاعب في التخمين حتى يتم كسر الرمز السري. اللاعب الذي يخمن بأقل عدد من المحاولات يفوز.

شرح الكود :

```
import random
```

يستورد هذا السطر المكتبة العشوائية (random) التي تستخدم لاحقًا لإنشاء رقم سري عشوائي.

```
def getDigits(num):
```

```
    return [int(i) for i in str(num)]
```

هذه الدالة (getDigits) تأخذ رقمًا كمدخل وتعيد قائمة تحتوي على أرقام هذا الرقم بشكل منفصل. على سبيل المثال، إذا تم تمرير الرقم 1234 إلى هذه الدالة، فستعيد القائمة [1, 2, 3, 4].

```
def noDuplicates(num):
```

```

num_li = getDigits(num)

if len(num_li) == len(set(num_li)):

    return True

else:

    return False

```

هذه الدالة (noDuplicates) تأخذ رقما كمدخل وتتحقق مما إذا كان يحتوي على أرقام متكررة. تستخدم الدالة getDigits لاستخراج أرقام الرقم ثم تستخدم مجموعة (set) للتخلص من الأرقام المتكررة. إذا كان طول القائمة الأصلية (مع الأرقام المتكررة) يساوي طول المجموعة (التي تحذف التكرارات)، فهذا يعني أن الرقم لا يحتوي على تكرارات، وتعيد الدالة True.

```

def generateNum():

    while True:

        num = random.randint(1000, 9999)

        if noDuplicates(num):

            return num

```

هذه الدالة (generateNum) تقوم بإنشاء رقم سري عشوائي مكون من 4 خانات ولا يحتوي على أرقام متكررة. تستمر الدالة في إنشاء أرقام عشوائية حتى تجد رقما لا يحتوي على تكرارات باستخدام دالة noDuplicates، ثم تقوم بإعادته.

```

def numOfBullsCows(num, guess):

    bull_cow = [0, 0]

    num_li = getDigits(num)

```



```

guess_li = getDigits(guess)
for i, j in zip(num_li, guess_li):
    if j in num_li:
        if j == i:
            bull_cow[0] += 1
        else:
            bull_cow[1] += 1
return bull_cow

```

هذه الدالة (numOfBullsCows) تأخذ الرقم السري والتخمين كمدخلات وتحسب عدد الثيران (الأرقام الصحيحة في الموضع الصحيح) والأبقار (الأرقام الصحيحة في الموضع الخاطئ). تستخدم دالة getDigits لتحويل الرقمين إلى قوائم من الأرقام الفردية. ثم تتحقق من كل رقم في التخمين (j) للتأكد من وجوده في الرقم السري (num_li). إذا وجد الرقم، يتم التحقق من موقعه (i) إذا كان الرقمان في نفس الموضع، يعتبر ذلك ثور ويتم زيادة قيمة العنصر الأول في قائمة bull_cow. أما إذا وجد الرقم ولكن في موضع خاطئ، يعتبر ذلك بقرة ويتم زيادة قيمة العنصر الثاني في القائمة.

```

num = generateNum()
tries = int(input('Enter number of tries: '))

```

يتم إنشاء رقم سري عشوائي باستخدام generateNum ويطلب من المستخدم إدخال عدد المحاولات المسموح بها.

```

while tries > 0:
    guess = int(input("Enter your guess: "))
    if not noDuplicates(guess):

```

```
print("Number should not have repeated digits. Try again.")
```

```
continue
```

```
if guess < 1000 or guess > 9999:
```

```
print("Enter 4 digit number only. Try again.")
```

```
continue
```

```
bull_cow = numOfBullsCows(num, guess)
```

```
print(f'{bull_cow[0]} bulls, {bull_cow[1]} cows')
```

```
tries -= 1
```

```
if bull_cow[0] == 4:
```

```
print("You
```

يتم هنا التحقق من نتيجة bull_cow. إذا كانت قيمة العنصر الأول (الثيران) تساوي 4، فهذا يعني أن جميع الأرقام في التخمين صحيحة وفي المواضع الصحيحة. لذلك، يتم طباعة رسالة "You guessed right!" ويتم كسر الحلقة (break) لإنهاء اللعبة.

```
else:
```

```
print(f'You have {tries} tries left.')
```

إذا لم يكن التخمين صحيحا (أي لم تكن قيمة الثيران 4)، يدخل الكود في كتلة else. هنا، يتم طباعة رسالة تخبر المستخدم بعدد المحاولات المتبقية باستخدام المتغير tries. تستمر الحلقة وتنتقل إلى التكرار التالي، حيث سيُطلب من المستخدم إدخال تخمين جديد.

```
else:
```

```
print(f'You ran out of tries. Number was {num}')
```

بمجرد انتهاء حلقة `while` (إما بسبب تخمين صحيح أو استنفاد المحاولات)، يتم التحقق من قيمة `tries`. إذا كانت المحاولات صفراً، فهذا يعني أن المستخدم لم يستطع تخمين الرقم السري في الوقت المحدد. لذلك، يتم طباعة رسالتين: الأولى تخبر المستخدم بأنه نفذ المحاولات، والثانية تكشف عن الرقم السري الذي كان يحاول تخمينه. (`num`)