



جامعة تشرين

كلية الهندسة الميكانيكية و الكهربائية

قسم هندسة الاتصالات و الالكترونيات

السنة الخامسة

الوظيفة الثانية لمادة برمجة الشبكات

إعداد : إسرائء محمد الحسن 2232

بثينه إبراهيم حمدان 1997

إشراف : د. مهند عيسى

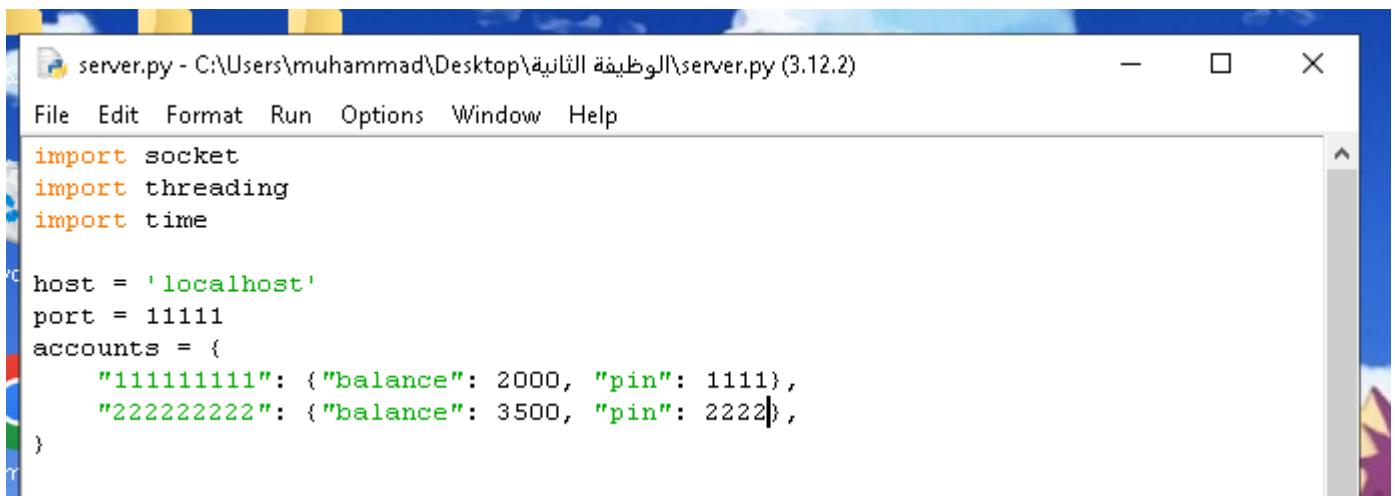
السؤال الأول :

فيما يلي كود سيرفر لصراف ATM يستطيع تخدم عدة زبائن في نفس الوقت  
تم استخدام Socket module لإنشاء اتصال بالتحقق من العنوان 'localhost' و  
port number في حال التطابق يتم الاتصال بنجاح

Thread module لكي يستطيع السيرفر تخدم عدة مستخدمين في نفس الوقت

Time module : تستخدم للتعامل مع الوقت

تم تعريف القاموس accounts{ } لإضافة ارقام الحسابات و تفاصيلها اليه متضمنة  
كلمة المرور PIN و الرصيد



```
server.py - C:\Users\muhammad\Desktop\الوظيفة الثانية\server.py (3.12.2)
File Edit Format Run Options Window Help
import socket
import threading
import time

host = 'localhost'
port = 11111
accounts = {
    "11111111": {"balance": 2000, "pin": 1111},
    "22222222": {"balance": 3500, "pin": 2222},
}
```

التابع handle\_client لمعالجة الطلبات المستقبلية من العملاء على القناة 1024 :

تمرر حلقة لكل عميل و استقبال البيانات منه و التحقق من الامر المطلوب ان كان  
التحقق من الرصيد او ايداع او تحويل

```
server.py - C:\Users\muhammad\Desktop\الوظيفة الثانية\server.py (3.12.2)
File Edit Format Run Options Window Help
"222222222": {"balance": 3500, "pin": 2222},
}

def handle_client(client_socket):
    for a in accounts.keys():
        client_socket.send(a.encode())

    data = client_socket.recv(1024).decode().strip()

    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])
        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."
```

فيما يلي التابع `verify_account()` يقوم بالتحقق من رقم الحساب و كلمة المرور حيث يعيد خطأ في حال كان رقم الحساب المدخل او كلمة المرور غير صحيحة.

- `Start_server()`: تابع ينشئ خادم و يستمع الى اتصالات العملاء
- يتم انشاء `socket.socket` باستخدام التعليمة `socket.socket`
- يتم ربط ال `socket` بعنوان `ip` باستخدام التعليمة `socket.bind`
- فيما يليها تعليمات للاستماع و الاستقبال و الارسال و انشاء خيوط جديدة لاستقبال عملاء جدد.

```
        if verify_account(account_number, pin) and accounts[account_
            accounts[account_number]["balance"] -= amount
            response = f"Withdrawn {amount:.2f}. Your new balance is
        else:
            response = "Insufficient funds."

    else:
        response = "Invalid command."

    client_socket.sendall(response.encode("utf-8"))

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)

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

        client_thread = threading.Thread(target=handle_client, args=(client_sock
        client_thread.start()

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

الخرج :

```
*IDLE Shell 3.12.2*
File Edit Shell Debug Options Window Help

Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\muhammad\Desktop\الوظيفة الثانية\server.py
[INFO] Starting server...
[INFO] Connected to ('127.0.0.1', 51119)
Exception in thread Thread-1 (handle_client):
Traceback (most recent call last):
  File "C:\Python\Python312\Lib\threading.py", line 1073, in _bootstrap_inner
    self.run()
  File "C:\Python\Python312\Lib\threading.py", line 1010, in run
    self._target(*self._args, **self._kwargs)
  File "C:\Users\muhammad\Desktop\الوظيفة الثانية\server.py", line 32, in handle_client
    amount = float(request[3])
              ~~~~~^
IndexError: list index out of range

*IDLE Shell 3.12.2*
File Edit Shell Debug Options Window Help

Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\muhammad\Desktop\الوظيفة الثانية\client.py
[INFO] Connecting to server...
Enter command (check_balance, deposit, withdraw): deposit
Enter account number: 111111111
Enter PIN: 1111
Server response: 111111111
Enter command (check_balance, deposit, withdraw): |
```

السؤال الثاني :

فيما يلي كود يطبق خوارزمية مسار Dijkstra لايجاد اقصر مسار بين عقدتين في المخطط الموزون .

تم استيراد المكتبة heapq توفر وظائف للتعامل مع الاكوام الأولية حيث هي مكتبة مخصصة لادارة بنية بيانات شجرية تستخدم لتخزين عناصر مرتبة

Collections.defaultdict : هي نوع بيانات من مكتبة ال collections ينشئ قاموس افتراضي حيث يتم تعيين قيمة افتراضية للقيم الغير موجودة.

- الدالة dijkstra تأخذ 3 بارمترات : الوزن – نقطة البداية- نقطة النهاية.
- انشاء قاموس لتخزين جيران كل عقدة باستخدام الزوج (weight,end)
- Q : جميع العقد التي لم يتم زيارتها
- Visited : العقد التي تمت زيارتها
- 

```
dijkstra.py - C:\Users\muhammad\Desktop\الوظيفة الثانية\dijkstra.py (3.12.2)
File Edit Format Run Options Window Help
from heapq import *
from collections import defaultdict

def dijkstra(edges, strat_node, end_node):
    g = defaultdict(list)
    for start, end, weight in edges:
        g[start].append((weight, end))
    q, visited = [(0, strat_node, ())], set()
```

حلقة while :

لاستخراج العنصر ذو الأقل تكلفة من قائمة الأولوية باستخدام heapop()  
اذا لم زيارة العقدة الحالية فنضيفها الى مجموعة العقد الى العقد المزارة و نحدث المسار بإضافة العقدة الحالية اليه.  
اذا كانت العقدة الحالية هي النقطة النهائية نعيد التكلفة و المسار.

حلقة for :

لكل عقدة v2 متصلة بالعقدة الحالية v1 نتحقق اذا لم يتم زيارة العقدة v2 نضيفها الى heap مع التكلفة و المسار المحدث .

تعليمة : return float("inf")

اذا لم توجد طرق ممكنة من نقطة البداية الى نقطة النهاية نعيد float("inf") و هي تمثل اللانهاية .

الشرط if name == "main"

هذا الجزء ينفذ عند تشغيل برنامج نعرف مجموعة من الحواف بالرسم البياني و  
نشغل دالة dijkstra لحساب اقصر مسار من العقدة A الى العقدة G و يطبع  
التكلفة و المسار.

```
while q:
    (cost,v1,path) = heappop(q)
    if v1 not in visited:
        visited.add(v1)
        path = (v1, path)
        if v1 == end_node:
            return (cost, path)
        for c, v2 in g.get(v1, {}):
            if v2 not in visited:
                heappush(q, (cost+c, v2, path))

    print (q)
return float("inf")

if __name__ == "__main__":

    edges = [
        ("A", "B", 7),
        ("A", "D", 5),
        ("B", "C", 8),
        ("B", "D", 9),
        ("B", "E", 7),
        ("C", "E", 5),
        ("D", "E", 7),
        ("D", "F", 6),
        ("E", "F", 8),
        ("E", "G", 9),
        ("F", "G", 11)
    ]

    print ("=== Dijkstra ===")
    print ("A >> G:")
    print (dijkstra(edges, "A", "G"))
```

---

الخرج :

```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
```

```
Type "help", "copyright", "credits" or "license()" for more information.
```

&gt;&gt;&gt;

```
= RESTART: C:\Users\muhammad\Desktop\الوظيفة الثانية\diijkstra.py
```

```
=== Dijkstra ===
```

```
A >> G:
```

```
[(5, 'D', ('A', ())), (7, 'B', ('A', ()))]  
[(7, 'B', ('A', ())), (12, 'E', ('D', ('A', ()))), (11, 'F', ('D', ('A', ())))]  
[(11, 'F', ('D', ('A', ()))), (12, 'E', ('D', ('A', ()))), (15, 'C', ('B', ('A', ()))), (14, 'E', ('B', ('A', ())))]  
[(12, 'E', ('D', ('A', ()))), (14, 'E', ('B', ('A', ()))), (15, 'C', ('B', ('A', ()))), (22, 'G', ('F', ('D', ('A', ()))))]  
[(14, 'E', ('B', ('A', ()))), (21, 'G', ('E', ('D', ('A', ()))), (15, 'C', ('B', ('A', ()))), (22, 'G', ('F', ('D', ('A', ()))))]  
[(15, 'C', ('B', ('A', ()))), (21, 'G', ('E', ('D', ('A', ()))), (22, 'G', ('F', ('D', ('A', ()))))]  
[(21, 'G', ('E', ('D', ('A', ()))), (22, 'G', ('F', ('D', ('A', ()))))]  
(21, ('G', ('E', ('D', ('A', ())))))
```

&gt;&gt;&gt; |