



الجمهورية العربية السورية

جامعة تشرين

كلية الهندسة الميكانيكية والكهربائية

قسم هندسة الاتصالات والإلكترونيات

السنة الخامسة

وظيفة البرمجة وإدارة الشبكات 2

إعداد الطلاب :

ماهر يوسف داؤد

سحر حسام الطبق

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    response = f"Withdrawn {amount:.2f}.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

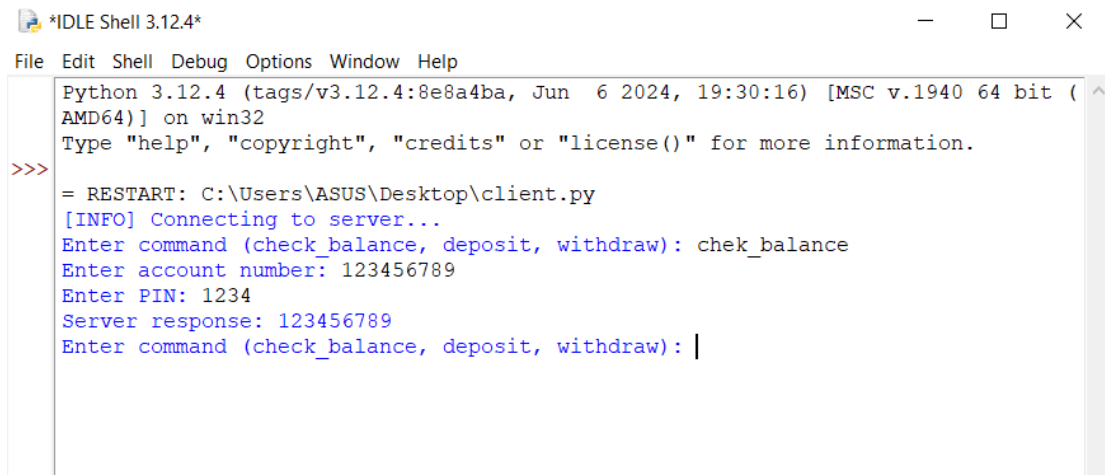
    while True:
        # إرسال طلب إلى الخادم
        command = input("Enter command (check_balance, deposit, withdraw): ")
        account_number = input("Enter account number: ")
        pin = int(input("Enter PIN: "))

        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()
```

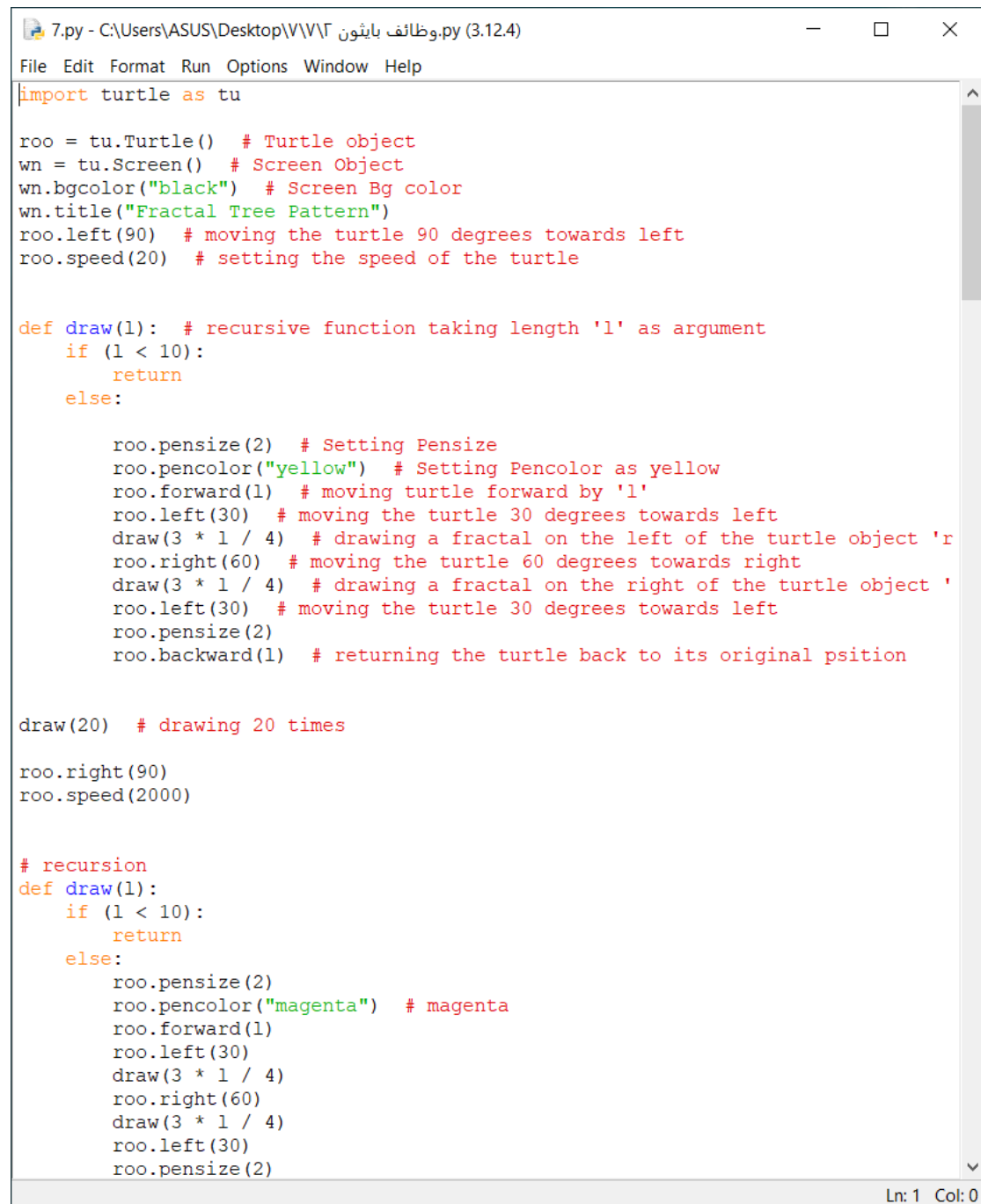
الخرج :



```
*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.

A screenshot of a Python IDE window titled '7.py - C:\Users\ASUS\Desktop\٧\٧\وظائف بايثون.py (3.12.4)'. The window contains a Python script for drawing a fractal tree using the turtle module. The script includes imports, initialization of a turtle and screen, a recursive function 'draw' to create the fractal, and a final call to 'draw(20)'. The script is written in a mix of English and Arabic comments. The IDE interface includes a menu bar (File, Edit, Format, Run, Options, Window, Help) and a status bar at the bottom showing 'Ln: 1 Col: 0'.

```
7.py - C:\Users\ASUS\Desktop\٧\٧\وظائف بايثون.py (3.12.4)
File Edit Format Run Options Window Help
import turtle as tu

roo = tu.Turtle() # Turtle object
wn = tu.Screen() # Screen Object
wn.bgcolor("black") # Screen Bg color
wn.title("Fractal Tree Pattern")
roo.left(90) # moving the turtle 90 degrees towards left
roo.speed(20) # setting the speed of the turtle

def draw(l): # recursive function taking length 'l' as argument
    if (l < 10):
        return
    else:
        roo.pensize(2) # Setting Pensize
        roo.pencolor("yellow") # Setting Pencolor as yellow
        roo.forward(l) # moving turtle forward by 'l'
        roo.left(30) # moving the turtle 30 degrees towards left
        draw(3 * l / 4) # drawing a fractal on the left of the turtle object 'r
        roo.right(60) # moving the turtle 60 degrees towards right
        draw(3 * l / 4) # drawing a fractal on the right of the turtle object '
        roo.left(30) # moving the turtle 30 degrees towards left
        roo.pensize(2)
        roo.backward(l) # returning the turtle back to its original psition

draw(20) # drawing 20 times

roo.right(90)
roo.speed(2000)

# recursion
def draw(l):
    if (l < 10):
        return
    else:
        roo.pensize(2)
        roo.pencolor("magenta") # magenta
        roo.forward(l)
        roo.left(30)
        draw(3 * l / 4)
        roo.right(60)
        draw(3 * l / 4)
        roo.left(30)
        roo.pensize(2)
```

Ln: 1 Col: 0

```
7.py - C:\Users\ASUS\Desktop\٧\٧\وظائف بايثون.py (3.12.4)
File Edit Format Run Options Window Help

    roo.backward(1)

draw(20)

roo.left(270)
roo.speed(2000)

# recursion
def draw(l):
    if (l < 10):
        return
    else:
        roo.pensize(2)
        roo.pencolor("red") # red
        roo.forward(1)
        roo.left(30)
        draw(3 * l / 4)
        roo.right(60)
        draw(3 * l / 4)
        roo.left(30)
        roo.pensize(2)
        roo.backward(1)

draw(20)

roo.right(90)
roo.speed(2000)

# recursion
def draw(l):
    if (l < 10):
        return
    else:
        roo.pensize(2)
        roo.pencolor('#FFF8DC') # white
        roo.forward(1)
        roo.left(30)
        draw(3 * l / 4)
        roo.right(60)
        draw(3 * l / 4)
        roo.left(30)
        roo.pensize(2)
        roo.backward(1)

Ln: 1 Col: 0
```

```
draw(20)

#####

def draw(l):
    if (l < 10):
        return
    else:
        roo.pensize(3)
        roo.pencolor("lightgreen") # lightgreen
        roo.forward(1)
        roo.left(30)
        draw(4 * l / 5)
        roo.right(60)
        draw(4 * l / 5)
        roo.left(30)
        roo.pensize(3)
        roo.backward(1)

draw(40)

roo.right(90)
roo.speed(2000)

# recursion
def draw(l):
    if (l < 10):
        return
    else:
        roo.pensize(3)
        roo.pencolor("red") # red
        roo.forward(1)
        roo.left(30)
        draw(4 * l / 5)
        roo.right(60)
        draw(4 * l / 5)
        roo.left(30)
        roo.pensize(3)
        roo.backward(1)

draw(40)
```

7.py - C:\Users\ASUS\Desktop\٧\٧\وظائف بايثون.py (3.12.4)

File Edit Format Run Options Window Help

```
roo.left(270)
roo.speed(2000)

# recursion
def draw(l):
    if (l < 10):
        return
    else:
        roo.pensize(3)
        roo.pencolor("yellow") # yellow
        roo.forward(1)
        roo.left(30)
        draw(4 * l / 5)
        roo.right(60)
        draw(4 * l / 5)
        roo.left(30)
        roo.pensize(3)
        roo.backward(1)

draw(40)

roo.right(90)
roo.speed(2000)

# recursion
def draw(l):
    if (l < 10):
        return
    else:
        roo.pensize(3)
        roo.pencolor('#FFF8DC') # white
        roo.forward(1)
        roo.left(30)
        draw(4 * l / 5)
        roo.right(60)
        draw(4 * l / 5)
        roo.left(30)
        roo.pensize(3)
        roo.backward(1)

draw(40)
```

Ln: 1 Col: 0

7.py - C:\Users\ASUS\Desktop\٧\٧ وظائف بايثون.py (3.12.4)

File Edit Format Run Options Window Help

```
def draw(1):
    if (1 < 10):
        return
    else:
        roo.pensize(2)
        roo.pencolor("cyan") # cyan
        roo.forward(1)
        roo.left(30)
        draw(6 * 1 / 7)
        roo.right(60)
        draw(6 * 1 / 7)
        roo.left(30)
        roo.pensize(2)
        roo.backward(1)

draw(60)

roo.right(90)
roo.speed(2000)

# recursion
def draw(1):
    if (1 < 10):
        return
    else:
        roo.pensize(2)
        roo.pencolor("yellow") # yellow
        roo.forward(1)
        roo.left(30)
        draw(6 * 1 / 7)
        roo.right(60)
        draw(6 * 1 / 7)
        roo.left(30)
        roo.pensize(2)
        roo.backward(1)

draw(60)

roo.left(270)
roo.speed(2000)

# recursion
```

Ln: 1 Col: 0

```
def draw(l):
    if (l < 10):
        return
    else:
        roo.pensize(2)
        roo.pencolor("magenta") # magenta
        roo.forward(1)
        roo.left(30)
        draw(6 * l / 7)
        roo.right(60)
        draw(6 * l / 7)
        roo.left(30)
        roo.pensize(2)
        roo.backward(1)

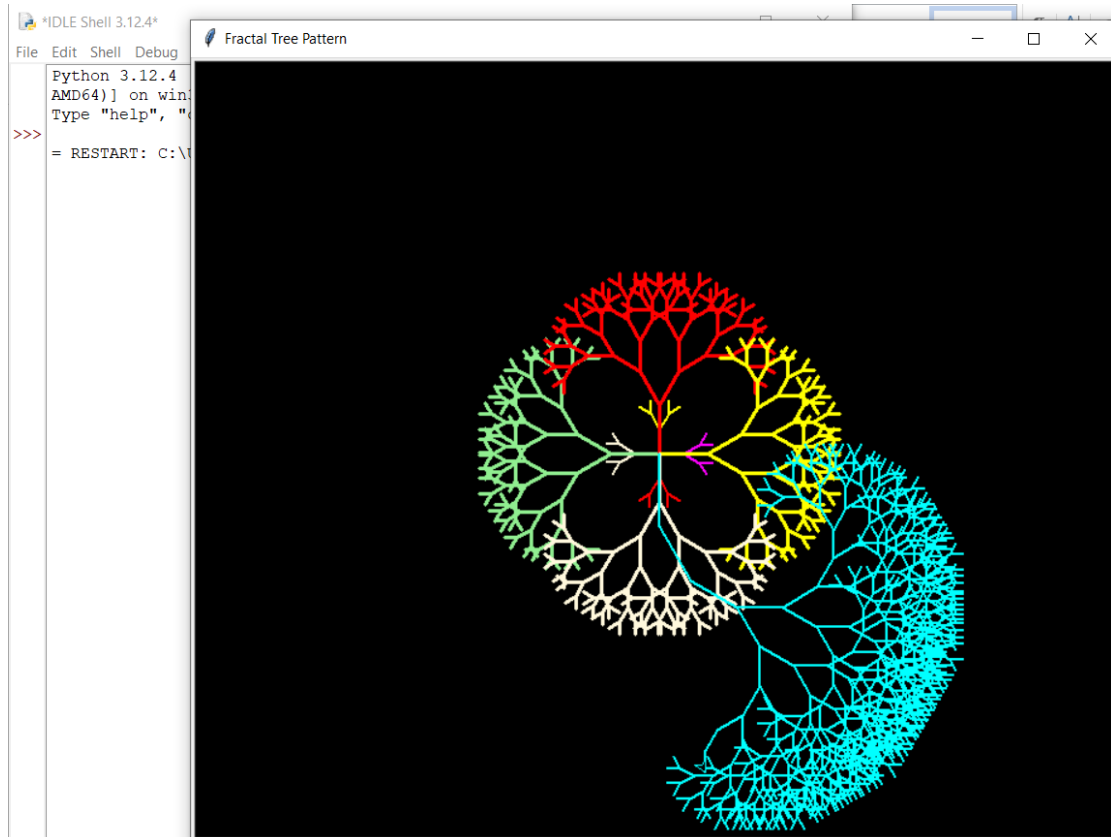
draw(60)

roo.right(90)
roo.speed(2000)

# recursion
def draw(l):
    if (l < 10):
        return
    else:
        roo.pensize(2)
        roo.pencolor('#FFF8DC') # white
        roo.forward(1)
        roo.left(30)
        draw(6 * l / 7)
        roo.right(60)
        draw(6 * l / 7)
        roo.left(30)
        roo.pensize(2)
        roo.backward(1)

draw(60)
wn.exitonclick()
```

الخرج :



Turtle Graphics with Python

شرح الكود :

هذا الكود مكتوب بلغة Python ويستخدم مكتبة Turtle لرسم أشكال فراكتالية على شكل شجرة.

استيراد المكتبات:

• `import turtle as tu:` يتم استيراد مكتبة Turtle وإعطائها الاسم المستعار `tu` لتسهيل الاستخدام.

إنشاء الكائنات:

- `roo = tu.Turtle()`: إنشاء كائن Turtle جديد ويتم تخزينه في المتغير `roo`. هذا الكائن يمثل قلم يرسم على الشاشة.
- `wn = tu.Screen()`: إنشاء كائن Screen جديد ويتم تخزينه في المتغير `wn`. هذا الكائن يمثل النافذة التي ستظهر عليها الرسومات.

تخصيص النافذة:

- `wn.bgcolor("black")`: ضبط لون الخلفية للنافذة إلى الأسود.
- `wn.title("Fractal Tree Pattern")`: عنوان النافذة إلى "Fractal Tree Pattern".

تهيئة كائن Turtle:

- `roo.left(90)`: يتم تحريك كائن Turtle بمقدار 90 درجة إلى اليسار.
- `roo.speed(20)`: يتم ضبط سرعة رسم كائن Turtle إلى 20 (يمكن تعديل هذه القيمة للتحكم في سرعة الرسم).

تعريف دالة الرسم: (draw)

- `def draw(l):`: يتم تعريف دالة متكررة تسمى `draw` التي تأخذ طول الفرع `l` كعامل.
- خطوات الدالة:
 1. شرط التوقف: `if (l < 10):` إذا كان طول الفرع أقل من 10، فهذا يعني أننا وصلنا إلى الحد الأدنى للفرع، ويتم التوقف عن الرسم.
 2. رسم الفرع الرئيسي:
 - `roo.pensize(2)`: يتم ضبط سمك قلم الرسم إلى 2.
 - `roo.pencolor("color")`: يتم ضبط لون القلم (هذه القيمة يتم تعويضها لاحقاً بناءً على الاستدعاء).
 - `roo.forward(l)`: يتم تحريك كائن Turtle إلى الأمام برسم خط بطول `l`.
 3. رسم الفروع الجانبية اليسرى:
 - `roo.left(30)`: يتم تحريك كائن Turtle بمقدار 30 درجة إلى اليسار.

▪ `draw(3 * 1 / 4)` : يتم استدعاء دالة `draw` بشكل متكرر لرسم فرع جانبي على اليسار، ولكن بطول يساوي 4/3 من طول الفرع الرئيسي.
4. رسم الفروع الجانبية اليمنى:

▪ `roo.right(60)` : يتم تحريك كائن Turtle بمقدار 60 درجة إلى اليمين.
▪ `draw(3 * 1 / 4)` : يتم استدعاء دالة `draw` بشكل متكرر لرسم فرع جانبي على اليمين، ولكن بطول يساوي 4/3 من طول الفرع الرئيسي.
5. العودة إلى الفرع الرئيسي:

▪ `roo.left(30)` : يتم تحريك كائن Turtle بمقدار 30 درجة إلى اليسار للعودة إلى اتجاه الفرع الرئيسي.
6. رسم خط العودة:

▪ `roo.pensize(2)` : يتم ضبط سمك قلم الرسم إلى 2 مرة أخرى.
▪ `roo.backward(1)` : يتم تحريك كائن Turtle إلى الخلف برسم خط بطول 1 للعودة إلى نقطة بداية الفرع الرئيسي.

استدعاء الدالة ورسم الشجرة:

• `draw(20)` : يتم استدعاء الدالة `draw` للمرة الأولى مع قيمة طول الفرع الابتدائي 20. ويتم تكرار عملية الرسم بشكل متعشع بناءً على شروط التوقف داخل الدالة.

تغيير اللون وسرعة الرسم و تكرار العملية:

يتم تكرار عملية استدعاء الدالة `draw` عدة مرات مع تغيير لون القلم (`pencolor`) وسرعة الرسم (`roo.speed`) وحجم الشجرة (`draw(20)` أو `draw(40)` أو `draw(60)`) لرسم أشجار فراكتلية متعددة الألوان و أحجام مختلفة.

نقطة التوقف:

• `wn.exitonclick()` : هذه السطر من الكود يجعل النافذة تنتظر حتى النقر عليها بالماوس قبل إغلاقها.

ملاحظات:

- يمكن تعديل سرعة الرسم (roo . speed) وحجم الشجرة (draw (20) أو draw (40) أو draw (60)) لتغيير مظهر الشجرة.
- يمكن إضافة المزيد من الألوان أو تغييرها لجعل الشجرة أكثر جمالاً.
- يمكن تعديل زوايا الرسم داخل وظيفة draw لتغيير شكل الشجرة.

ملاحظات إضافية:

- مكتبة Turtle هي أداة بسيطة وفعالة لرسم أشكال هندسية مختلفة.
- يمكن استخدام تقنيات الرسم المتكررة لإنشاء أشكال معقدة مثل أشجار الفراكالية.
- يمكن تعديل الكود بسهولة لتغيير مظهر وخصائص الشجرة.