UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO

CARRERA PROFESIONAL DE INGENIERIA INFORMATICA Y DE SISTEMAS

COMPUTACION GRAFICA 2

DOCENTE: M.SC. HECTOR E. UGARTE R.

HOJA DE EJERCICIOS 2: L-SYSTEMS

1. COMPETENCIAS

- El estudiante conoce los sistemas L.
- El estudiante utiliza turtle de Python para graficar.

2. MARCO TEORICO

El concepto de Sistema de Lindenmayer fue concebido por el biólogo y botánico teórico húngaro Aristid Lindenmayer de la Universidad de Utrecht, en 1968. Sin embargo, fueron dos de sus estudiantes, Ben Hesper y Pauline Hogeweg los primeros en darse cuenta del potencial de los sistemas-L para representar plantas (en un principio).

Un sistema-L es un lenguaje, una gramática formal de derivación paralela, un conjunto de reglas y símbolos2 principalmente utilizados para modelar el proceso de crecimiento de las plantas, aunque también puede modelar la morfología de una gran variedad de organismos.

Los sistemas-L también son conocidos como sistemas-L paramétricos, definidos como un conjunto:

$$G = \{V, S, \omega, P\}$$

donde:

V : El alfabeto, es un conjunto de símbolos que pueden ser reemplazados (variables o símbolos no terminales) y se utilizan para componer cadenas.

S : Es un conjunto de símbolos que se mantienen fijos (constantes o símbolos terminales).

ω : El axioma, es la cadena que describe al sistema en su estado inicial, formada por un(os) símbolo(os) de V .

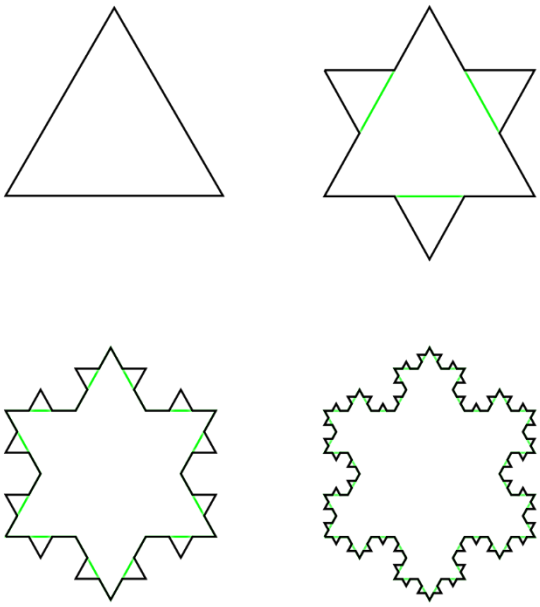
P : Reglas de producción, son las transformaciones que serán aplicadas al axioma y, sucesivamente, a las cadenas generadas. Definen la forma en la que las variables pueden ser reemplazadas por combinaciones de constantes y otras variables. Las reglas de producción generan cadenas formadas únicamente por los símbolos del alfabeto y por lo tanto todas las cadenas pertenecerán al lenguaje definido por el sistema-L.

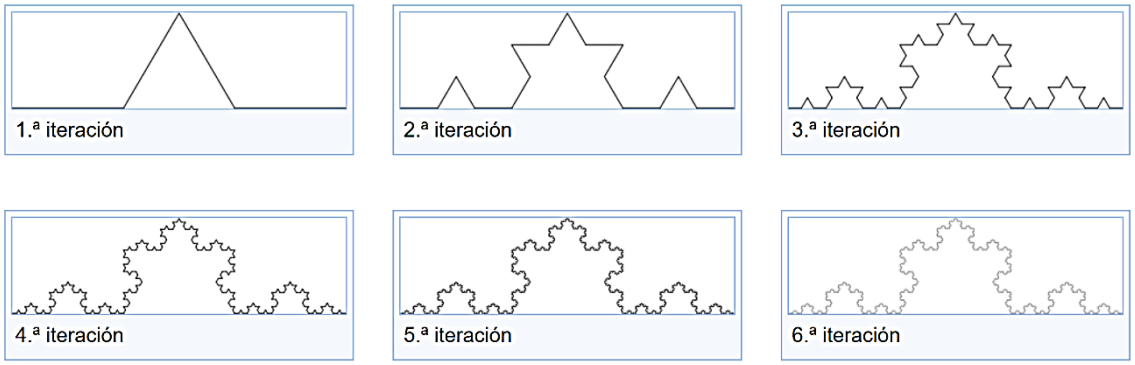
Copo de nieve de Koch

El copo de nieve de Koch, también llamado estrella de Koch o isla de Koch,1 es una curva cerrada continua pero no diferenciable en ningún punto descrita por el matemático sueco Helge von Koch en 1904.

Se toma un segmento, se lo divide en tres partes iguales, se remplaza la parte central por dos partes de igual longitud haciendo un ángulo de 60 grados. Luego, con los cuatro segmentos, se procede de la misma manera, lo que da lugar a 16 segmentos más pequeños en la segunda

iteración. Y así sucesivamente. La figura representa las seis primeras etapas de la construcción. La última curva es una buena aproximación de la curva final.





Alfabeto: F

Constantes: +, −

Axioma: F++F++F

Reglas de producción:

F → F−F++F−F

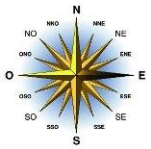
Semántica:

Aquí, F significa «continúa dibujando», + «gira 60 grados a la derecha, y - «gira 60 grados a la izquierda»

Gráficos Tortuga y la librería turtle de Python

Para instalar: pip install PythonTurtle

"Tortuga" (Turtle) es una característica de Python como un tablero de dibujo, que nos permite ordenar a una tortuga que dibuje por todas partes. Podemos usar funciones como turtle.forward(…) y turtle.right(…) que pueden mover la tortuga.



Los métodos de tortuga comúnmente utilizados son:

Método	Parámetro	Descripción
Turtle()	Ninguno	Crea y devuelve un nuevo objeto de tortuga.
forward()	cantidad	Mueve la tortuga hacia adelante en la cantidad especificada
backward()	cantidad	Mueve la tortuga hacia atrás la cantidad especificada
right()	ángulo	Gira la tortuga en el sentido de las agujas del reloj
left()	ángulo	Gira la tortuga en sentido contrario a las agujas del reloj.
penup()	Ninguno	Recoge la pluma de la tortuga.
pendown()	Ninguno	Deja la pluma de la tortuga
up()	Ninguno	Recoge la pluma de la tortuga.
down()	Ninguno	Deja la pluma de la tortuga
color()	Nombre del color	Cambia el color de la pluma de la tortuga.
fillcolor()	Nombre del color	Cambia el color de la tortuga que usará para llenar un polígono
heading()	Ninguno	Devuelve el encabezado actual
position()	Ninguno	Devuelve la posición actual
goto()	x, y	Mueve la tortuga a la posición x,y
begin_fill()	Ninguno	Recuerda el punto de partida de un polígono relleno
end_fill()	Ninguno	Cierre el polígono y rellénelo con el color de relleno actual
dot()	Ninguno	Deja el punto en la posición actual
stamp()	Ninguno	Deja una impresión de una forma de tortuga en la ubicación actual
shape()	nombre de forma	Debería ser 'flecha', 'clásico', 'tortuga' o 'círculo'

3. PRACTICA

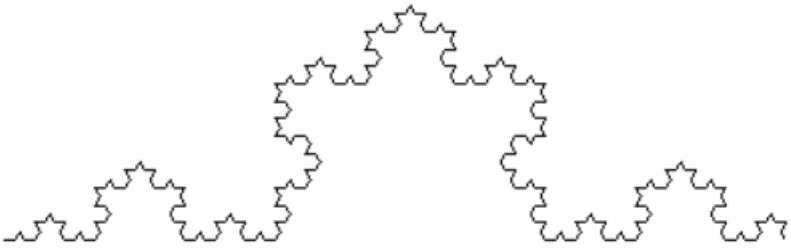
A. Implementemos el copo de nieve de Koch utilizando Python y turtle. Digita el siguiente código, analiza como se producen las cadenas en función de las reglas y verifica su correcto funcionamiento.

```
from turtle import *

def crearSistemaL(numIters,axioma):
    cadenaInicio = axioma
    cadenaFin = ""
    for i in range(numIters):
        cadenaFin = procesarCadena(cadenaInicio)
        cadenaInicio = cadenaFin
    return cadenaFin

def procesarCadena(cadenaVieja):
```

```
    nuevaCadena = ""
    for ch in cadenaVieja:
        nuevaCadena = nuevaCadena + aplicarReglas(ch)
    return nuevaCadena

def aplicarReglas(ch):
    nuevaCadena = ""
    if ch == 'F':
        nuevaCadena = 'F-F++F-F'   # Regla 1
    else:
        nuevaCadena = ch    # No se aplica regla, mantenemos el caracter
    return nuevaCadena

def dibujarSistemaL(aTurtle, instrucciones, angulo, distancia):
    for cmd in instrucciones:
        if cmd == 'F':
            aTurtle.forward(distancia)
        elif cmd == 'B':
            aTurtle.backward(distancia)
        elif cmd == '+':
            aTurtle.right(angulo)
        elif cmd == '-':
            aTurtle.left(angulo)

def main():
    tracer(0, 0)
    inst = crearSistemaL(4, "F")   # crea la cadena
    print(inst)
    t = Turtle()             # crea la tortuga
    wn = Screen()
    t.up()
    t.back(200)
    t.down()
    t.speed(9)
    dibujarSistemaL(t, inst, 60, 5)   # dibuja la imagen
                                  # angulo 60, longitud de segmento 5

main()
```

Salida:

```
F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F-F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F++F-F++
F-F-F-F++F-F++F-F++F-F-F-F++F-F-F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F-F-F++F-F-F-
F++F-F++F-F++F-F-F-F++F-F-F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-
F++F-F++F-F-F-F++F-F-F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F++F-
F++F-F-F-F++F-F-F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F++F-F++F-
F-F-F++F-F-F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F-F-F++F-F-F-F++F-F++F-F++F-F-F-F+
+F-F-F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F-
F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F
```



B. El siguiente sistema L implementa un árbol binario:

Variables: 0, 1

constantes: “[”, “]”

axioma: 0

reglas: (1 → 11), (0 → 1[0]0)

1ra recursividad: 1[0]0

2da recursividad: 11[1[0]0]1[0]0

3ra recursividad: 1111[11[1[0]0]1[0]0]11[1[0]0]1[0]0

…

Semantica:

- 0: dibuja un segmento de línea que termina en una hoja
- 1: dibujar un segmento de línea
- [: posición y ángulo de empuje, gire a la izquierda 45 grados
-]: posición pop y ángulo, gire a la derecha 45 grados

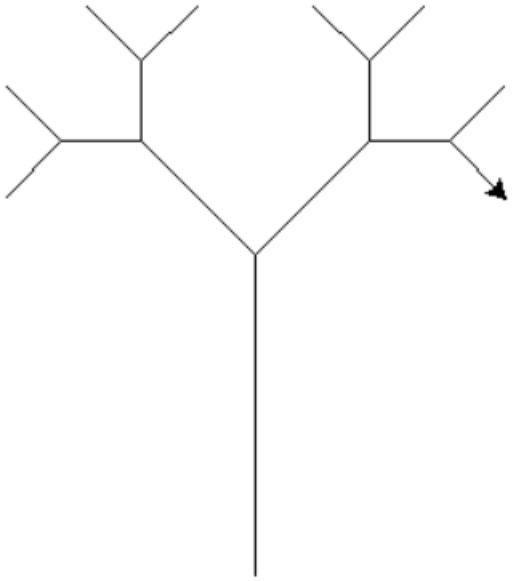
Analiza el código, como se están implementando las reglas y lo mas importante como se utiliza una pila para graficar correctamente:

```
from turtle import *

def crearSistemaL(numIters,axioma):
    cadenaInicio = axioma
    cadenaFin = ""
    for i in range(numIters):
        cadenaFin = procesarCadena(cadenaInicio)
        cadenaInicio = cadenaFin
    return cadenaFin

def procesarCadena(cadenaVieja):
    nuevaCadena = ""
    for ch in cadenaVieja:
        nuevaCadena = nuevaCadena + aplicarReglas(ch)
    return nuevaCadena
```

```
def aplicarReglas(ch):
    nuevaCadena = ""
    if ch == '1':
        nuevaCadena = '11'     # Regla 1
    elif ch == '0':
        nuevaCadena = '1[0]0' # Regla 2
    else:
        nuevaCadena = ch     # No se aplica regla, mantenemos el caracter
    return nuevaCadena

def dibujarSistemaL(aTurtle, instrucciones, angulo, distancia):
    pila = []
    aTurtle.setheading(90)
    for cmd in instrucciones:
        if cmd == '0':
            aTurtle.forward(distancia)
        elif cmd == '1':
            aTurtle.forward(distancia)
        elif cmd == '[':
            pila.append((aTurtle.xcor(),aTurtle.ycor(),aTurtle.heading()))
            aTurtle.left(angulo)
        elif cmd == ']':
            xcor, ycor, heading = pila.pop()
            aTurtle.penup()
            aTurtle.setpos(xcor,ycor)
            aTurtle.pendown()
            aTurtle.setheading(heading)
            aTurtle.right(angulo)

def main():
    inst = crearSistemaL(3, "0")    # crea la cadena
    print(inst)
    t = Turtle()             # crea la tortuga
    wn = Screen()
    t.up()
    t.back(200)
    t.down()
    t.speed(2)
    dibujarSistemaL(t, inst, 45, 40)

main()
```

Salida:

```
1111[11[1[0]0]1[0]0]11[1[0]0]1[0]0
```



C. De manera similar implementa el siguiente sistema L:

Variables: F

constantes: “+”,”-“

axioma: F+F+F+F

reglas: F -> F+F-F-FF+F+F-F

1 iteración: F+F-F-FF+F+F-F + F+F-F-FF+F+F-F + F+F-F-FF+F+F-F + F+F-F-FF+F+F-F

2 iteración: F+ F-F-FF+ F+ F-F+ F+ F-F-FF+ F+ F-F-F+ F-F-FF+ F+ F-F-F+ F-F-FF+ F+ F-FF+ F-F-FF+ F+ F-F+ F+ F-F-FF+ F+ F-F+ F+ F-F-FF+ F+ F-F-F+ F-F-FF+ F+ F-F+ F+ F-F-FF+ F+ F-F-F+ F-F-FF+ F+ F-FF+ F-F-FF+ F+ F-F+ F+ F-F-FF+ F+ F-F+ F+ F-F-FF+ F+ F-F-F+ F-F-FF+ F+ F-F+ F+ F-F-FF+ F+ F-F+ F+ F-F-FF+ F+ F-F-F+ F-F-FF+ F+ F-FF+ F-F-FF+ F+ F-F+ F+ F-F-FF+ F+ F-F+ F+ F-F-FF+ F+ F-F-F+ F-F-FF+ F+ F-F+ F+ F-F-FF+ F+ F-F+ F+ F-F-FF+ F+ F-F-F+ F-F-FF+ F+ F-FF+ F-F-FF+ F+ F-F+ F+ F-F-FF+ F+ F-F+ F+ F-F-FF+ F+ F-F-F+ F-F-FF+ F+ F-F

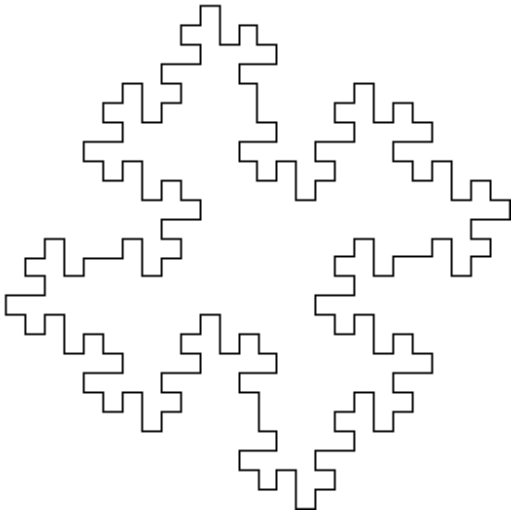
Semántica:

F: mover adelante n unidades dibujando una línea

+: girar a la derecha 90 grados

-: girar a la izquierda 90 grados

Por ejemplo, la segunda iteración se debería ver asi:



D. De manera similar implementar el siguiente sistema L.

Variables: X F

Constantes: +, −, [,]

Axioma: X

Reglas: (X → F+[[X]-X]-F[-FX]+X), (F → FF)

Angulo: 25°

Semántica:

F significa: “dibujar adelante”, - significa “girar a la derecha 25 grados”, + significa “girar a la izquierda 25 grados”, X no se dibuja, se utiliza para controlar la evolución de la curva. [. Se utiliza para guardar los valores actuales de posición y ángulo, los que se restauran cuando se ejecuta].

n=6