**1. Importación de librerías**

import matplotlib.pyplot as plt

import numpy as np

* **matplotlib.pyplot:** Se usa para crear gráficos y dibujar figuras.
* **numpy:** Facilita el manejo de arreglos y operaciones matemáticas, en este caso para trabajar con las coordenadas de los puntos del triángulo.

**2. Función recursiva sierpinski\_triangle**

def sierpinski\_triangle(ax, points, depth):

"""Genera recursivamente el Triángulo de Sierpinski"""

if depth == 0:

triangle = plt.Polygon(points, edgecolor="black", facecolor="white")

ax.add\_patch(triangle)

else:

middle = (points[0] + points[1]) / 2

mid2 = (points[1] + points[2]) / 2

mid3 = (points[2] + points[0]) / 2

sierpinski\_triangle(ax, [points[0], middle, mid3], depth - 1)

sierpinski\_triangle(ax, [middle, points[1], mid2], depth - 1)

sierpinski\_triangle(ax, [mid3, mid2, points[2]], depth - 1)

* **Parámetros:**
  + ax: El objeto de ejes de Matplotlib donde se dibujará el triángulo.
  + points: Un arreglo (lista o array de NumPy) con las coordenadas de los tres vértices del triángulo actual.
  + depth: La profundidad de recursión, que determina cuántas subdivisiones se realizan.
* **Caso base (depth == 0):**  
  Si la profundidad es 0, se dibuja un triángulo simple usando plt.Polygon con los puntos dados. Se añade este triángulo al objeto de ejes ax.
* **Caso recursivo (depth > 0):**  
  Se calculan los puntos medios de cada lado del triángulo:
  + middle: Punto medio entre el primer y segundo vértice.
  + mid2: Punto medio entre el segundo y tercer vértice.
  + mid3: Punto medio entre el tercer y primer vértice.

Luego, se realizan **tres llamadas recursivas** a la función, cada una con un triángulo formado por:

* + **Primer triángulo:** Vértices [points[0], middle, mid3].
  + **Segundo triángulo:** Vértices [middle, points[1], mid2].
  + **Tercer triángulo:** Vértices [mid3, mid2, points[2]].

En cada llamada se reduce la profundidad (depth - 1), lo que permite crear la estructura fractal característica del Triángulo de Sierpinski.

**3. Función generate\_sierpinski**

def generate\_sierpinski(depth=5):

"""Configura la figura y genera el Triángulo de Sierpinski"""

fig, ax = plt.subplots()

ax.set\_aspect("equal")

ax.set\_xticks([])

ax.set\_yticks([])

ax.set\_xlim(0, 1)

ax.set\_ylim(0, np.sqrt(3) / 2)

points = np.array([[0.5, np.sqrt(3) / 2], [0, 0], [1, 0]])

sierpinski\_triangle(ax, points, depth)

plt.show()

* **Creación de la figura y ejes:**  
  Se crea una figura y un objeto de ejes usando plt.subplots().
  + ax.set\_aspect("equal"): Asegura que las escalas de los ejes X e Y sean iguales para que el triángulo no se deforme.
  + Se eliminan las marcas de los ejes con ax.set\_xticks([]) y ax.set\_yticks([]).
* **Configuración de límites de la gráfica:**
  + ax.set\_xlim(0, 1): Define el rango del eje X.
  + ax.set\_ylim(0, np.sqrt(3) / 2): Define el rango del eje Y, aprovechando que la altura del triángulo equilátero es np.sqrt(3)/2.
* **Definición de los puntos iniciales:**  
  Se crea un arreglo points que contiene las coordenadas de los tres vértices de un triángulo equilátero:
  + Vértice superior: [0.5, np.sqrt(3) / 2]
  + Vértice inferior izquierdo: [0, 0]
  + Vértice inferior derecho: [1, 0]
* **Generación del Triángulo de Sierpinski:**  
  Se llama a sierpinski\_triangle(ax, points, depth) para iniciar el proceso recursivo.
* **Visualización:**  
  Finalmente, plt.show() muestra la figura resultante.

**4. Ejecución del código**

# Genera el Triángulo de Sierpinski

generate\_sierpinski(5)

* Se llama a la función generate\_sierpinski con una profundidad de 5, lo que define cuántas subdivisiones se realizarán para generar el fractal.

**Resumen**

* **Importación de librerías:** Se cargan Matplotlib para dibujar y NumPy para los cálculos con arreglos.
* **Función recursiva (sierpinski\_triangle):** Divide el triángulo en tres triángulos más pequeños en cada nivel de profundidad, dibujando el triángulo cuando se alcanza el caso base.
* **Configuración y dibujo (generate\_sierpinski):** Prepara el entorno gráfico, define el triángulo inicial y llama a la función recursiva.
* **Ejecución:** Se genera y muestra el Triángulo de Sierpinski con 5 niveles de profundidad.

