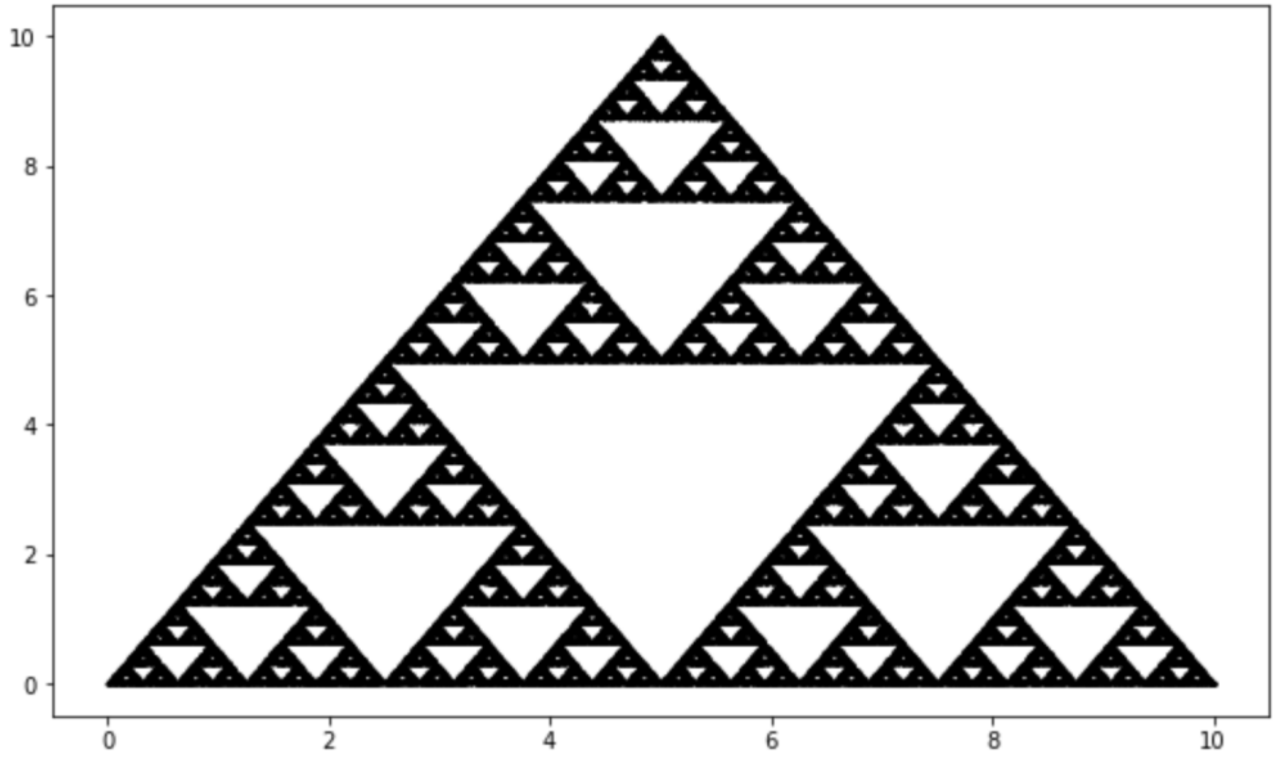
Sierpinski Triangle in Python

ddas.tech/sierpinski-triangle-in-python/

February 4, 2023

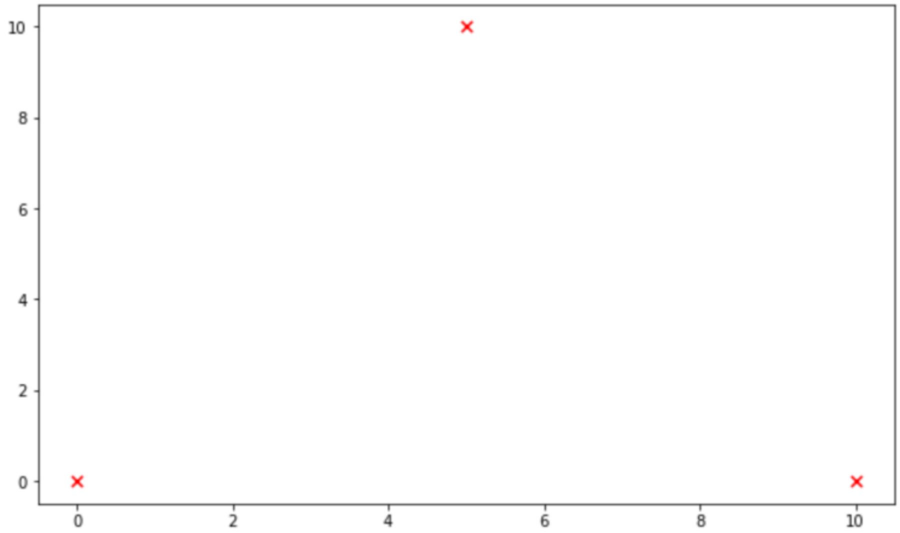
The Sierpinski triangle is a fractal pattern, named after Polish mathematician Waclaw Sierpinski, that is created by recursively dividing a larger triangle into four smaller triangles and removing the middle one. The pattern continues indefinitely, producing a self-similar structure at smaller and smaller scales. The Sierpinski triangle is one of the simplest fractals and has become a popular example of mathematical art. **In this post we will generate a Sierpinski Triangle in Python using random vertices in a 2D Space**



Sierpinski triangle

Generating the three random vertices for a triangle in a 2D space

```
import random
import matplotlib.pyplot as plt
import numpy as np

def generate_random_point(min_x, max_x, min_y, max_y):
  x = random.uniform(min_x, max_x)
  y = random.uniform(min_y, max_y)
  return (x, y)


def generate_three_vertices(min_x, max_x, min_y, max_y):
    p1 = generate_random_point(min_x, max_x, min_y, max_y)
    p2 = generate_random_point(min_x, max_x, min_y, max_y)
    p3 = generate_random_point(min_x, max_x, min_y, max_y)
    return [p1,p2,p3]

min_x = 0
max_x = 10
min_y = 0
max_y = 10

vertices_1 = [(0,0),(10,0),(5,10)]
vertices = generate_three_vertices(min_x,max_x,min_y,max_y)
vertices = vertices_1
# Comment the above line if you want to have random vertices in a 2D Space
print(vertices)
A = np.array(vertices)
fig = plt.figure(figsize = (10, 6))
plt.scatter(
   A[:, 0], A[:, 1],
   marker='x',
   color='red', s=50
)

plt.show()
```



Triangle vertices in a 2D Space with Max X = 10 and Max Y = 10

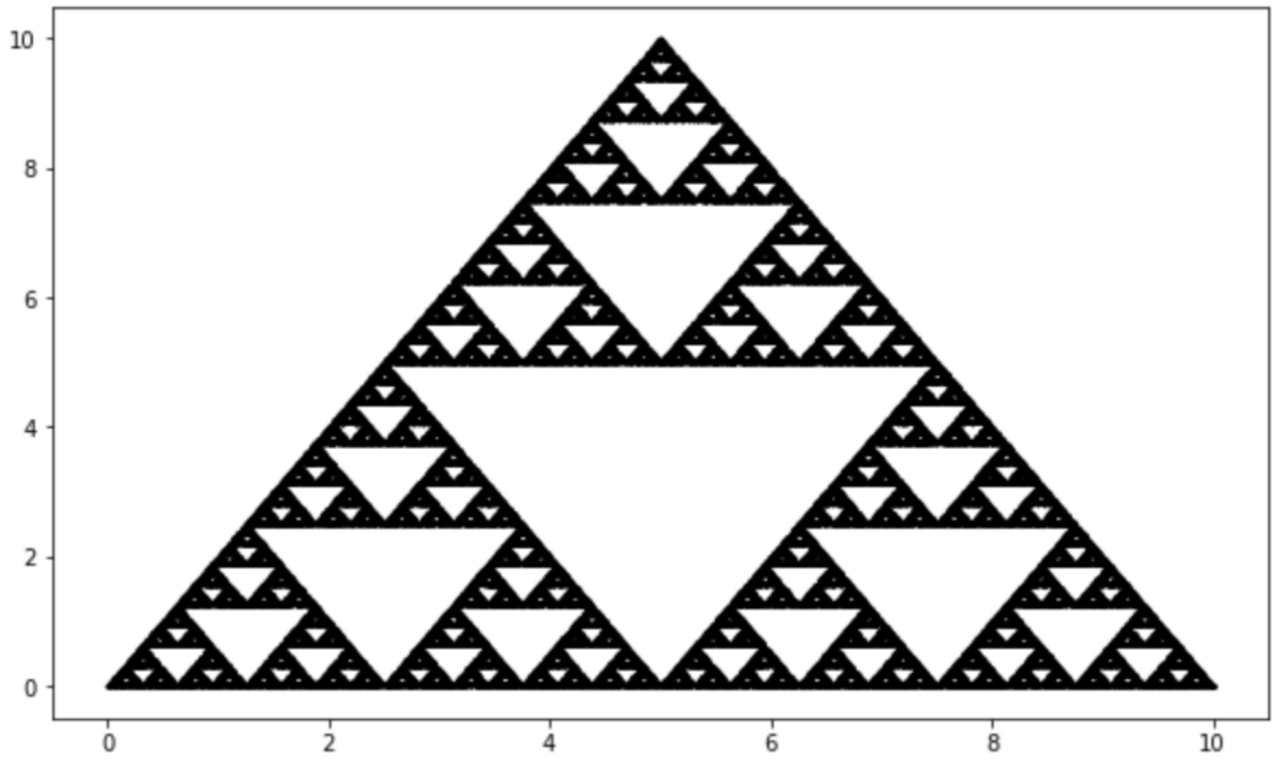
In the below code we will implement the following

- **Generate a random point inside the boundary of a triangle**
- **Find the midpoint given two points**
- **Build the points that form the Sierpinski Triangle**
- **Draw the Sierpinski Triangle**

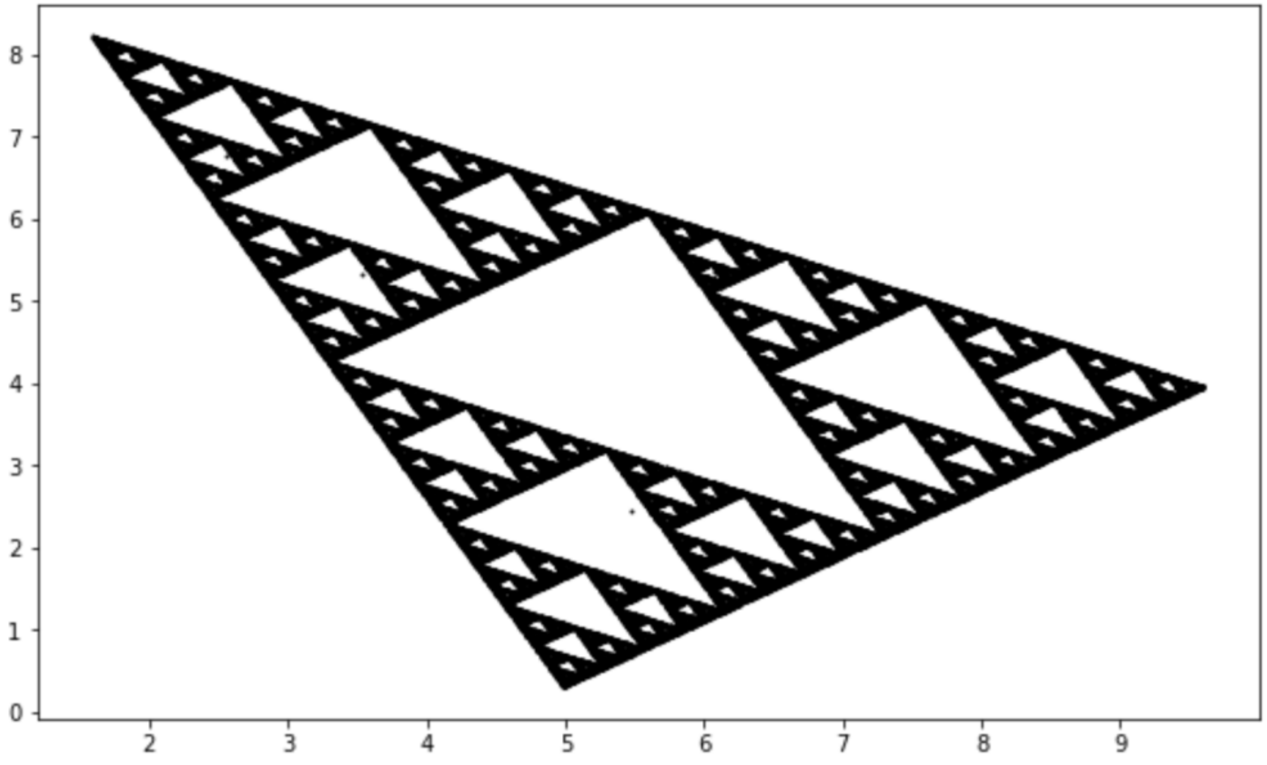
Algorithm used for building the Sierpinski Triangle in Python

1. Generate a triangle vertices in 2D Space
2. Find a random point within the boundaries of the triangle and add it to a list
3. Find the mid point between the random point from step 2 and any of the triangle vertices selected randomly
4. Add the mid point from step 3 to the points list
5. Use the mid point from Step 4 and repeat Step 3 for N number of times.
6. Finally plot the Sierpinski triangle using the points present in the list.

```
# The below function generates a random point in the 2D space
# in the boundary of the three vertices of the triangle
def random_point_in_triangle(vertices):
    u, v = np.random.rand(2)
    if u + v > 1:
        u = 1 - u
        v = 1 - v
    return (
        (1 - u - v) * vertices[0][0] + u * vertices[1][0] + v * vertices[2][0],
        (1 - u - v) * vertices[0][1] + u * vertices[1][1] + v * vertices[2][1],
    )

# Given two points the below function generates the midpoint of the two given points
def midpoint(p1, p2):
    x1, y1 = p1
    x2, y2 = p2
    x_mid = (x1 + x2) / 2
    y_mid = (y1 + y2) / 2
    return (x_mid, y_mid)


def buildSierpinski(random_point, tri_vertices,number_points):
    point_list = []
    point_list += tri_vertices
    point_list.append(random_point)
    initial_point = random_point
    for i in range(number_points):
        vertex = random.choice(tri_vertices)
        mid_point = midpoint(vertex,initial_point)
        point_list.append(mid_point)
        initial_point = mid_point
    return point_list

def draw_sierpenski_triange(points):
    A = np.array(points)
    fig = plt.figure(figsize = (10, 6))
    plt.scatter(
       A[:, 0], A[:, 1],
       marker='o',
       color='black', s=1
    )
    plt.show()


randPoint = random_point_in_triangle(vertices)
numOfSierpenskiPoints = 100000
points = buildSierpinski(randPoint,vertices,numOfSierpenskiPoints)
draw_sierpenski_triange(points)
```



For the below snapshot , we have used random vertices for the triangle



Sierpinski triangle with Random Vertices