```
# Generating a Fractal - general usage

import matplotlib.pyplot as plt
import numpy as np

class Vector:                       # REMINDER: class --> defines the attributes and bahaviors of the vector
    def __init__(self, x, y):      # REMINDER: init function --> special method in a class that is called when
        self.x = x                 # instance variable for x
        self.y = y                 # instance variable for y

class AffineTransform:
    def __init__(self, a, b, c, d, e, f):   # initializes object's attributes
        self.a = a    # instance variables
        self.b = b
        self.c = c
        self.d = d
        self.e = e
        self.f = f

def recursivePlot(vector, transforms, num_transforms, current_depth, max_depth, ax):  # recursive plot functi
    if current_depth == max_depth:          # base case: current recursion depth equals maximum recursion dep
        ax.plot(vector.x, vector.y, 'ro')   # if true --> recursivePlot will plot the Vector with a call to a
        return

# apply affine transformations to the vector
    for transform in transforms:            # loops through all the transformations in the array
        new_x = transform.a * vector.x + transform.b * vector.y + transform.e
        new_y = transform.c * vector.x + transform.d * vector.y + transform.f
        new_vector = Vector(new_x, new_y)   # calculates a new vector for each transformation
        recursivePlot(new_vector, transforms, num_transforms, current_depth + 1, max_depth, ax)

def drawFractal(startPoint, transforms, max_depth):   # draws the fractal w/ specified params
```

```
    fig, ax = plt.subplots()                          # ax plot allows the function to plot on the same set o
    num_transforms = len(transforms)                  # the number of transforms is equal to the amount/lengt
    recursivePlot(startPoint, transforms, num_transforms, 0, max_depth, ax)
    ax.set_xlabel('X')
    ax.set_ylabel('Y')
    ax.set_title('Sierpinski triangle')
    ax.grid(True)
    plt.savefig('fractal.svg', format='svg')          # saves the fractal as an svg file
```

```
# Example usage - using the Sierpinski triangle for this example

# Starting point of the fractal - from user input
def user_input():
    while True:
        try:
            x = float(input("Enter the x-coordinate of the starting point (between 0 and 12): "))
            y = float(input("Enter the y-coordinate of the starting point (between 0 and 18): "))
            if 0 <= x <= 12 and 0 <= y <= 18:
                return Vector(x, y)
            else:
                print("Error: Coordinates out of bounds. Please enter coordinates within the range [0, 12] fo
        except ValueError:
            print("Error: Invalid input. Please enter numerical values for coordinates.")

if __name__ == "__main__":
    startPoint = user_input()

    # Define transformations for Sierpinski triangle
    transforms = [
        AffineTransform(0.5, 0, 0, 0.5, 0, 0),   # Scale down by 1/2
        AffineTransform(0.5, 0, 0, 0.5, 0.5, 0), # Scale down by 1/2 and move right by 0.5
        AffineTransform(0.5, 0, 0, 0.5, 0.25, 0.5) # Scale down by 1/2, move right by 0.25, and up by 0.5
```

```
    AffineTransform(0.5, 0, 0, 0.5, 0.25, 0.5) # Scale down by 1/2, move right by 0.25, and up by 0.5
]

# Calculate maximum recursion depth
max_depth = 5

# Execute the draw fractal function
drawFractal(startPoint, transforms, max_depth)
```