```
#This is your starting point
seed=(seed_x, seed_y)

#Initialize a list where you will store your points (x_t,y_t), starting with your seed
points = [seed]
```

```
Enter the x-coordinate of the seed point: 0.5

Enter the y-coordinate of the seed point: 0.3

The point is not inside the triangle. Please try again.

Enter the x-coordinate of the seed point: 0.3

Enter the y-coordinate of the seed point: 0.5

Valid seed point entered.
```

[15]:

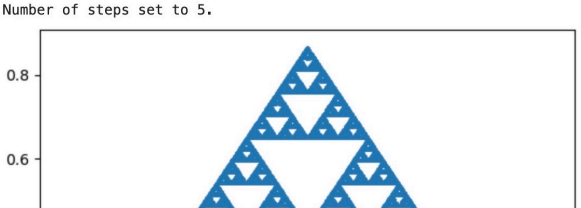
```
# Prompting the user for the number of steps
while True:
    try:
        num_steps = int(input("Enter the number of steps: "))
        if num_steps > 0:
            print(f"Number of steps set to {num_steps}.")
            break
        else:
            print("Please enter a positive integer.")
    except ValueError:
        print("Invalid input. Please enter a positive integer.")

for i in range(num_steps):
    #choose a random vertex to move toward from the list 'vertices'
    # You can use the python code random.randint(0, 2) to
    # choose a random integer between 0 and 2. Then you can use that random integer as
    # your code should look like "next_vertex = vertex[ a random choice of index]"
```

(Pyodide) | Idle Mode: Command Ln 1, Col 1

Code Notebook Py

```
    #choose a random vertex to move toward from the list 'vertices'
    # You can use the python code random.randint(0, 2) to
    # choose a random integer between 0 and 2. Then you can use that random integer as
    # your code should look like "next_vertex = vertex[ a random choice of index]"
    next_vertex=vertices[random.randint(0,2)]

    #create the next point by moving from the last point, i.e. points[-1], to the midpo
    # You may have to look up the formula for the midpoint of a line in the plane.
    next_point=((points[-1][0] + next_vertex[0]) / 2, (points[-1][1] + next_vertex[1]) / 2)

    #add the new point to your list of points
    points.append(next_point)


# Function to plot the solution set
def plot_solution(points):
    plt.scatter([p[0] for p in points], [p[1] for p in points], s=1)
    plt.show()


#plot your points

plot_solution(points)
```

```
Enter the number of steps: 5

Number of steps set to 5.
```

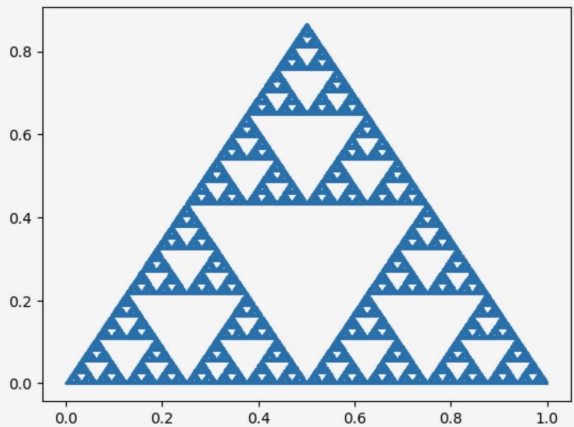


ide) | Idle Mode: Command Ln 1, Col 1

Code

Notebook

Number of steps set to 5.



[19]:

```
import matplotlib.pyplot as plt
import random
import math

#What if we use a unit circle instead of a triangle?

def isInside(x,y):
    return x**2+y**2<1


# Prompting the user for a seed point
while True:
```

dide) | Idle

Mode: Command

Ln 1, Col 1

Code

Notebook

```
def isInside(x,y):
    return x**2+y**2<1


# Prompting the user for a seed point
while True:
    try:
        seed_x = float(input("Enter the x-coordinate of the seed point: "))
        seed_y = float(input("Enter the y-coordinate of the seed point: "))
        if isInside(seed_x, seed_y):
            print("Valid seed point entered.")
            break
        else:
            print("The point is not inside the triangle. Please try again.")
    except ValueError:
        print("Invalid input. Please enter numerical values.")

#This is your starting point
seed=(seed_x, seed_y)

#Initialize a list where you will store your points (x_t,y_t), starting with your seed
points = [seed]
```

```
Enter the x-coordinate of the seed point: 0.5

Enter the y-coordinate of the seed point: 0.5

Valid seed point entered.
```

24]:

```
# Prompting the user for the number of steps
while True:
    try:
        num_steps = int(input("Enter the number of steps: "))
        if num_steps > 0:
            print(f"Number of steps set to {num_steps}.")
```

| Idle

Mode: Command

Ln 1, Col 1

Code

Notebook

[24]:

```
# Prompting the user for the number of steps
while True:
    try:
        num_steps = int(input("Enter the number of steps: "))
        if num_steps > 0:
            print(f"Number of steps set to {num_steps}.")
            break
        else:
            print("Please enter a positive integer.")
    except ValueError:
        print("Invalid input. Please enter a positive integer.")

for i in range(num_steps):
    #choose a random vertex to move toward from the list 'vertices'
    # You can use the python code random.randint(0, 2) to
    # choose a random integer between 0 and 2. Then you can use that random integer as
    # your code should look like "next_vertex = vertex[ a random choice of index]"

    # Generate a random angle between 0 and 2π
    theta = random.uniform(0, 2 * math.pi)

    # Calculate the coordinates
    x = math.cos(theta)
    y = math.sin(theta)
    next_vertex=(x,y)

    #create the next point by moving from the last point, i.e. points[-1], to the midpo
    # You may have to look up the formula for the midpoint of a line in the plane.
    next_point=((points[-1][0] + next_vertex[0]) / 2, (points[-1][1] + next_vertex[1]) / 2)

    #add the new point to your list of points
    points.append(next_point)


# Function to plot the solution set
def plot_solution(points):
```

lide) | Idle

Mode: Command

Ln 1, Col 1

[13]:

```
import matplotlib.pyplot as plt
import random
# TODO: Implement the Chaos Game algorithm
# 1. Initialize a random seed point within the triangle
# 2. Roll the die and move the point half the distance to the chosen vertex.
# 3. Repeat and collect the points in a list
# (Remember to start plotting points after a dozen rolls)
# Initialize the vertices of the triangle and the plot
# Vertices of an equilateral triangle
vertices = [(0, 0), (1, 0), (0.5, 0.866)]
# A function to check whether point (x, y)
# lies inside the triangle formed by
# A0, 0), (1, 0) and (0.5, 0.866)
def isInside(x1, y1, x2, y2, x3, y3, x, y):
    def area(x1, y1, x2, y2, x3, y3):
        return abs((x1*(y2 - y3) + x2*(y3 - y1) + x3*(y1 - y2)) / 2.0)
    A = area(x1, y1, x2, y2, x3, y3)
    A1 = area(x, y, x2, y2, x3, y3)
    A2 = area(x1, y1, x, y, x3, y3)
    A3 = area(x1, y1, x2, y2, x, y)
    return A == A1 + A2 + A3

# Prompting the user for a seed point
while True:
    try:
        seed_x = float(input("Enter the x-coordinate of the seed point: "))
        seed_y = float(input("Enter the y-coordinate of the seed point: "))
        if isInside(0, 0, 1, 0, 0.5, 0.866, seed_x, seed_y):
            print("Valid seed point entered.")
            break
        else:
            print("The point is not inside the triangle. Please try again.")
    except ValueError:
        print("Invalid input. Please enter numerical values.")
```

e) | Idle

Mode: Command

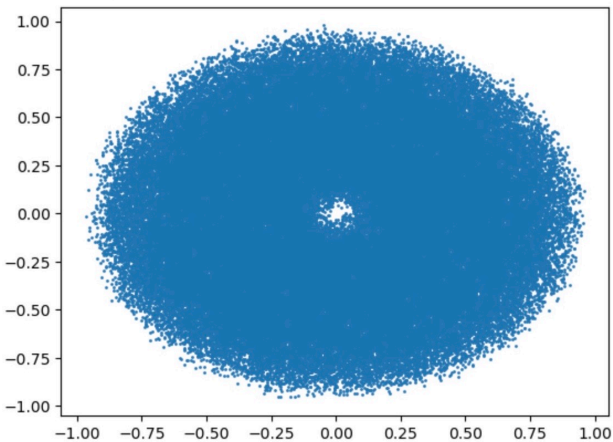
Ln 1, Col 1

Sierpinski.ipynb

0

```
# Function to plot the solution set
def plot_solution(points):
    plt.scatter([p[0] for p in points], [p[1] for p in points], s=1)
    plt.show()


#plot your points

plot_solution(points)
```

```
Enter the number of steps: 100000

Number of steps set to 100000.
```



de) | Idle Mode: Command Ln 1, Col 1