Python을 활용한 프랙털(Fractal) 생성 및 프랙털 차원 탐구

대건고등학교 20928 정준영

I. 서론

수학 I 교과서를 읽어보면 I-2.4 ‘로그함수의 활용’ 단원에서 ‘프랙털 차원’(p.60)과 III-1.5 ‘등비수열의 합’ 단원에서 ‘프랙털 카드 만들기’(p.142)를 읽어보면 프랙털의 정의와 여러 흥미로운 사실들을 알 수 있다. 교과서에 따르면 프랙털이란 “전체를 부분으로 나누었을 때 부분 안에 전체의 모습을 갖는 기하학적 모형”이다[1]. 더욱 자세한 정의를 알기 위해 인터넷 사전을 사용하여 추가적인 정의를 알 수 있었다. “언제나 부분이 전체를 닮는 자기 유사성(self-similarity)과 소수(小數) 차원을 특징으로 갖는 형상을 일컫는다. ‘프랙털’이란 이름은 1975년 B.B.만델브로에 의해 지어졌으나, 이러한 형상들에 관한 추상적 논의는 훨씬 이전부터 있었다. 칸토어집합, 코흐눈송이, 시어핀스키삼각형 등이 그 예이다.[2]”. 또한 그 아래에 프랙털의 차원에 대한 정의가 나와 있다. 프랙털 도형을 원래 도형과 닮은 작은 도형으로 분할 할 때, 분할된 닮은 도형의 개수를 N, 원래 도형과 분할된 도형의 닮음비를 1:r이라 할 때, ‘프랙털 차원’ D는 $D = \dfrac{\log N}{\log \dfrac{1}{r}}$ 로 정의한다.

여기서 ‘프랙털 차원은 정확히 무엇을 나타내는 것인가?’라는 의문이 들게 된다. 흔히 ‘차원’이라 하면 3차원, 4차원 등 자연수 차원을 생각할 것이다. 그러나 ‘프랙털 차원’은 정의부터 log와 분수로 이루어져 있다. r과 N의 값에 따라 자연수가 아닌 다른 수들도 가능하다. 그렇다면 일반적인 ‘차원’과 프랙털에서 말하는 ‘차원’은 다른 것일까? 이러한 의문 외에도 추가로 ‘그렇다면 프랙털 모형을 프로그래밍을 통해 직접 그려 볼 수 있을까?’란 의문이 들었다.

이러한 의문들을 해결하기 위해 본문에서 프랙털에 대한 자세한 정의와 프랙털 차원의 의미, 그리고 여러 프랙털의 종류들과 그 프랙털의 원리와 수식을 이용하여 Python으로 프로그래밍하여 직접 프랙털 모형을 그려보고 확대해보며 프랙털 도형이 맞음을 증명해보는 탐구 활동을 진행할 것이다. 탐구할 내용에 아직 이해하기 어려운 수준의 공식과 개념들이 존재하므로 이번 탐구의 목적 중 하나인 프랙털 차원의 이해에 필요한 log가 들어간 식만을 설명하고 이 외의 수학 개념들이 들어가는 수식의 경우 조사만 하고 따로 추가적인 탐구 또는 증명은 하지 않을 것이다.

II. 본문

1. 프랙털과 프랙털 차원

1.1. 프랙털의 정의

서론에서 말했다시피 프랙털이란 전체를 부분으로 나누어도 동일한 모습을 갖는 기하학적 도형이다. 즉 특정 부분으로 확대를 하여도 확대하기 전 전체모습과 같은 모습을 띠게 된다. 이러한 특성을 자기 유사성(self-similarity)이라고 한다.

1.2. 프랙털의 차원

프랙털은 특이한 차원을 가진다. 프랙털의 차원은 일반적으로 생각하는 정수 차원과는 다르게 소수(小數)이다.

우리가 일반적으로 생각하는 차원의 정의는 '수학에서, 어떤 대상의 모든 원소들을, 몇 개 (또는 무한대)의 정해진 원소들 $x_1, x_2, \dots, x_n$을 조합해서 모두 나타낼 수 있을 때, 그 정해진 원소들을 기저라고 부르며, 기저 원소의 수를 차원(次元)이라고 한다.[3]' 이다. '기저'라는 말을 더 쉽게 말하자면 "기하학적 도형에서 한 점의 위치를 나타내는 데 필요한 수의 개수[4]" 이다. 우리가 자주 사용했던[i] 차원은 벡터 공간에서의 차원을 의미한다. 기저의 수에 따라 기저가 1개면 1차원, 2개면 2차원, 4개면 4차원이라 부르는 것이다.

프랙털 차원은 이 벡터 차원과는 다르다. 프랙털 차원은 벡터 차원이 아닌 '하우스도르프 차원'이다. 하우스드로프 차원이란 "집합 S와 반지름 r이 주어졌을 때, S를 N(r)개의 공[ii]으로 덮을 수 있다고 할 때 하우스드로프 차원 d는 r이 0으로 갈 때 N(r)가 r^{-d}로 수렴하게 만드는 유일한 실수 d를 말한다.[5]"이다. 즉 하우스드로프 차원 d의 정의는

$$d = \infty\left\{\alpha \geq 0 \middle| \lim_{\delta \to 0} \infty \left(\sum_i (diam(U_i))^{\alpha}\right) = 0\right\}$$

이다[6].

i) 흔히 2차원, 3차원, 4차원이라 부르던 정수 차원

ii) 구의 안쪽, 또는 그 경계인 구까지 포함한 것. (공(기하학), 위키피디아)

프랙털 차원을 쉽게 이해하기 위해 일반적인 차원과 프랙털 차원을 비교하여 생각해보았다. 길이가 1인 선분, 넓이가 1인 정사각형, 부피가 1인 정육면체 그리고 한변의 길이가 1인 시에르핀스키 삼각형이 있다 해보자. 각 모형들의 선분을 $\frac{1}{2}$해 보면 선분의 길이는 $\frac{1}{2}$, 정사각형의 넓이는 $\frac{1}{4}$[i], 정육면체의 부피는 $\frac{1}{8}$, 시에르스핀키 삼각형은 $\frac{1}{3}$이 된다.[ii]. 이번엔 선분의 길이를 $\frac{1}{4}$로 줄여보자. 선분의 길이는 $\frac{1}{4}$, 정사각형의 넓이는 $\frac{1}{16}$, 정육면체의 부피는 $\frac{1}{32}$, 시에르핀스키 삼각형의 넓이는 $\frac{1}{9}$가 된다. 이를 일반화해보면

	선분(1차원)	평면도형(2차원)	입체도형(3차원)
$\frac{1}{2}$	$\left(\frac{1}{2}\right)^1$	$\left(\frac{1}{2}\right)^2$	$\left(\frac{1}{2}\right)^3$
$\frac{1}{4}$	$\left(\frac{1}{4}\right)^1$	$\left(\frac{1}{4}\right)^2$	$\left(\frac{1}{4}\right)^3$
$\frac{1}{n}$	$\left(\frac{1}{n}\right)^1$	$\left(\frac{1}{n}\right)^2$	$\left(\frac{1}{n}\right)^3$

이다. 즉 어떠한 도형의 선분들을 $\frac{1}{n}$ 했을 때 그 도형이 $\left(\frac{1}{n}\right)^d$등분된다면, 해당 도형의 차원은 d이다. 이를 시에르핀스키 삼각형에 적용해본다면, 시에르핀스키 삼각형의 각 변을 $\frac{1}{2}$배 하였을 때 삼각형이 삼등분 즉 처음의 $\frac{1}{3}$배 되므로 시에르핀스키의 차원은

$$\left(\frac{1}{2}\right)^d = \frac{1}{3},\quad d = \log_{\frac{1}{2}}(\frac{1}{3}) = \log_{2^{-1}}(3^{-1}) = -(-\log_2(3)) = \log_2(3) = \frac{\log 3}{\log 2} \approx 1.585$$

이다[iii]. 즉 각 선분을 n등분하여 전체가 m등분 된 도형의 차원은 $\frac{\log m}{\log n}$이다.[7)]

---

i) 넓이가 1인 1×1 정사각형의 각 변을 $\frac{1}{2}$하면, $\frac{1}{2}\times\frac{1}{2}$ 정사각형이 되므로 넓이는 $\frac{1}{4}$이 된다.

ii) 시에르핀스키 삼각형의 각 변의 길이를 $\frac{1}{2}$하면 삼등분되어 시에르핀스키 삼각형 3개가 나오게 된다. 따라서 하나의 넓이는 $\frac{1}{3}$이 된다. II-2-2.1의 그림에서 두 번째 시에르핀스키 삼각형 참고.

iii) 길이를 $\frac{1}{4}$하여 시에르핀스키 삼각형이 $\frac{1}{9}$ 될 때도 성립한다. $d = \log_{\frac{1}{4}}(\frac{1}{9}) = \log_{2^{-2}}(3^{-2}) = \frac{\log 3^2}{\log 2^2} = \frac{2\log 3}{2\log 2} \approx 1.585$ 으로 길이를 $\frac{1}{2}$ 하였을 때와 같다.

2. 프랙털의 종류

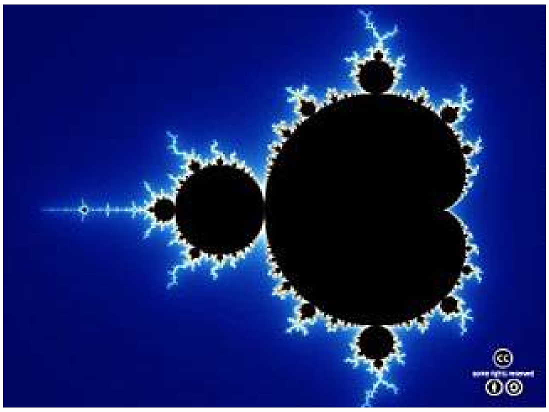
2.1. 시에르핀스키 삼각형(Trójkąt Sierpińskiego)

시에르핀스키 삼각형은 대표적인 프랙털 도형 중 하나이다. 시에르핀스키 삼각형은 내부가 채워진 정삼각형의 세 변의 중점을 연결하여 만들어진 정삼각형 내부를 지우고 다시 남은 세 개의 정삼각형의 각 변 중점을 연결하고 내부를 지우고를 무한히 반복하여 만들 수 있다. 제일 처음 정삼각형의 넓이를 S라 하고 중점을 연결하여 지우는 과정을 한 횟수를 n이라 할 때, n번째 도형의 넓이 S_n은 $(\frac{3}{4})^n S$ 이다. 이 과정을 무한히 반복하면 넓이 $S_\infty = \lim_{n\to\infty}(\frac{3}{4})^n S = 0$이다. 이와 비슷하게 변의 길이의 합을 구해보면 처음 정삼각형의 변 길이 합을 l이라 하면 l_n은 $(\frac{3}{2})^n l$이다. 이 과정을 무한히 반복하면 $l_n = \lim_{n\to\infty}(\frac{3}{2})^n l = \infty$이다. 시에르핀스키 삼각형의 프랙털 차원은 처음과 나중 삼각형의 비가 2:3 이므로 $\frac{\log 3}{\log 2} \approx 1.585$ 차원이다.[8)][9)] 재밌는 사실로는 '파스칼의 삼각형[i)]'의 홀수만 따로 빼보면 시에르핀스키 삼각형과 같다.

---

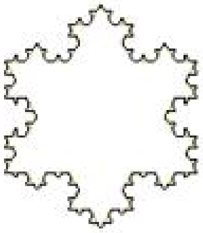
i) 이항계수를 삼각형 모양으로 배열한 것. (파스칼 삼각형, 위키피디아)

2.2. 망델브로 집합(Mandelbrot set)



망델브로 집합이란 복소수 범위에서 z에 대한 점화식[i] $z_0 = 0$ (단 z_n은 복소수), $z_{n+1} = z_n^2 + c$[ii]으로 표현된 수열 $\{z_n\}$의 절댓값이 무한대로 발산하지 않는 복소수 c의 집합이다. 이 집합 c를 복소평면[iii]에 나타내면 위 사진과 같은 도형이 나오게 된다. 망델브로 집합을 그릴 땐 "$|z_n| > 2$면 $\{z_n\}$은 발산한다"는 성질을 이용하여 해당 명제를 만족하는 c를 제외한 모든 복소수 c를 복소평면 위에 그리면 된다.[10][11][12][13] 망델브로 집합의 프랙털 차원은 2차원이라고 한다.[14]

2.3. 코흐 곡선(Koch Curve)



코흐 곡선은 선분을 삼등분하여 가운데 위치한 선분을 삼등분한 선분 길이의 정삼각형으로 만드는 과정을 반복한 프랙털 모형이다. 만약 처음 시작을 선분이 아닌 정삼각형으로 시작한다면 위 사진과 같은 '코흐 눈송이(Koch snowflake)'가 된다. 코흐의 눈송이의 프랙털 차원은 삼등분된 선분이 변환 후 4개가 되므로 $\frac{\log 4}{\log 3} \approx 1.261$이다.[15] 코흐 눈송이의 둘레의 길이는 무한이고, 넓이는 $\frac{\sqrt{3}}{4}\left\{1 + \lim_{n\to\infty}\sum_{k=1}^{n}\frac{3}{4}\times\left(\frac{4}{9}\right)^k\right\} = \frac{2\sqrt{3}}{5}$라고 한다.[16][17]

---

i) 수열의 일반항을 한 개 이상의 앞선 항들을 이용하여 나타낸 식. (점화식, 수학백과)

ii) z_n을 (x_n, y_n)로, c를 (a,b)로 바꾸면 식은 $(x_0, y_0) = (0, 0)$, $x_{n+1} = x_n^2 - y_n^2 + a$, $y_{n+1} = 2x_n y_n^2 + b$ (단, x_n, y_n, a, b는 실수.) 로 될 수 있다. (망델브로 집합, 위키피디아)

iii) 좌표평면의 각 점에 복소수를 대응시킨 것. (복소평면, 수학백과)

3. Python으로 프랙털 그려보기

3.1. 파이썬(Python)과 모듈/라이브러리

파이썬(Python)은 1991년 개발된 객체 지향 언어[i]이자 동적 정형[ii] 프로그래밍 언어이다. 쉬운 난이도와 풍부하고 다양한 라이브러리/모듈 덕분에 많은 곳에서 다양하게 사용되고 있다.

모듈은 파이썬에서 내부 또는 외부에서 불러오는 파이썬 코드 모음 파일이다. 모듈 안에 여러 기능의 함수, 클래스, 변수 등이 담겨있다. 라이브러리는 여러 모듈을 담아놓은 폴더와 비슷한 개념이다. 라이브러리 전체 혹은 라이브러리 속에서 특정 모듈을 불러와 사용할 수 있다.

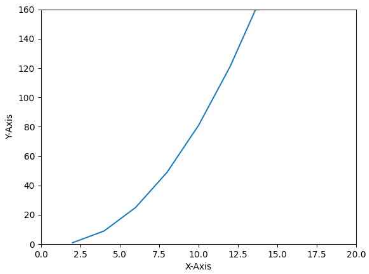
3.2. matplotlib.pyplot

matplotlib.pyplot는 MATLAB[iii]과 비슷하게 명령어로 그래프를 그릴 수 있는 파이썬 외부 모듈이다. 여러 메소드들을 통하여 여러 수치와 값(x, y, 출력 유형, 색 등)을 입력받아 이를 실행하면 그래프를 그려서 출력하게 된다.

matplotlib.pyplot 사용법을 익히기 위해 약간의 함수들을 사용한 코드를 작성해보았다.[18][19]

```
import matplotlib.pyplot as plt # matplotlib.pyplot 모듈 import

plt.plot([2*m for m in range(1, 20)][:10], [n*n for n in range(1, 20, 2)][:10]) # X: 2~40 사이의 모든 짝수 중 앞 10개 / Y: 1~20 사이의 모든 홀수의 제곱 중 앞 10개
plt.xlabel('X-Axis') # X축의 이름 지정
plt.ylabel('Y-Axis') #Y축의 이름 지정
plt.axis([0, 20, 0, 160]) # [X 최소값, X 최대값, Y최소값, Y최대값]

plt.show() # 그래프 보여주기
```

[iv]

해당 코드의 실행 결과는 

이다.

---

i) 프로그래밍 언어의 형태 중 하나로 일반적인 절차형 프로그래밍 언어(예를 들어 C언어)와 다르게 프로그래밍을 객체들의 조합으로 보아 각 클래스, 객체에 대한 함수들과 메서드로 프로그래밍하고 실행한다. (객체 지향 프로그래밍, 위키피디아)

ii) 컴퓨터에서 데이터를 저장할 때 자료형이 정적 정형과 다르게 각 변수별로 모든 자료형을 가질 수 있는 형태를 말한다. (자료형 체계, 위키피디아)

iii) 매스웍스에서 개발한 수치 해석 및 프로그래밍 환경 제공하는 공학용 소프트웨어. (매트랩, 위키피디아)

iv) 해당 코드는 깃허브(GitHub)에 업로드 하였다.

3.3. 파이썬과 matplotlib으로 시에르핀스키 삼각형 그려보기

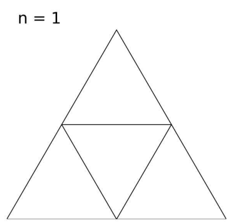
파이썬과 모듈을 사용하여 II-2-2.1 시에르핀스키 삼각형을 그려보았다. 앞서 조사한 시에르핀스키 삼각형의 형성 원리를 이용하여 변환 횟수 n의 값을 조정하며 시에르핀스키 삼각형이 어떻게 그려지는지를 확인해보았다.

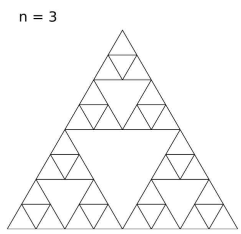
```python
def sierpinski(ax, v, d): # 시에르핀스키 삼각형 생성 함수 매개변수 (ax, v, d) = (축, 초기 정삼각형 각 꼭짓점 좌표(이후 삼각형들의 꼭짓점 좌표 리스트로 활용), 변형한 횟수(재귀 깊이))
    if d == 0: # 만약 변형한 횟수가 0이라면 (= 처음 함수가 실행되었을 때)
        ax.add_patch(plt.Polygon(v, edgecolor='k', fill=None)) # 꼭짓점의 길이 데이터가 v인 도형을 테두리색 검정(k)으로 내부를 채우지 않고 생성하여 축에 추가
    else:
        midpoints = [(v[i] + v[(i + 1) % 3]) / 2 for i in range(3)] # 각 변의 중점을 계산

        sierpinski(ax, [v[0], midpoints[0], midpoints[2]], d - 1)
        sierpinski(ax, [v[1], midpoints[1], midpoints[0]], d - 1)
        sierpinski(ax, [v[2], midpoints[2], midpoints[1]], d - 1) # 초기 삼각형에서 삼등분된 삼각형을 다시 초기값으로 정하여 다시 각각의 삼각형에 대해 시에르핀스키 삼각형 생성 재귀함수 실행
```

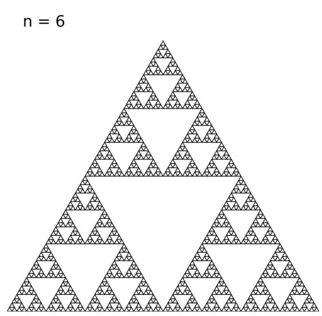
i)

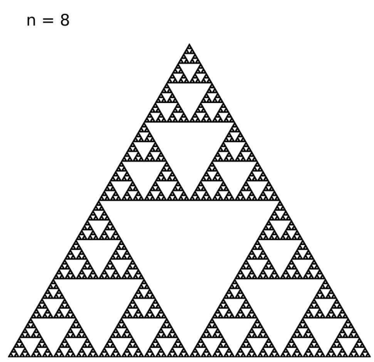
전체 코드는 깃허브ii)에 업로드 하였다.

해당 코드를 n=1 ~ n=10으로 실행한 결과는 다음과 같다:

n = 1:


n = 3:


n = 6:


n = 8:


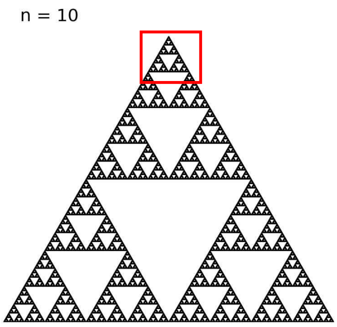
이외 결과는 깃허브에 업로드 해두었다.

i) 해당 코드는 주어진 값들로 재귀함수(자기 자신을 호출하는 함수(재귀함수, 천재학습백과 초등 소프트웨어 용어사전))를 사용하여 시에르핀스키 삼각형을 생성하는 부분이다.

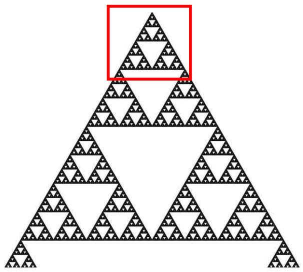
ii) 분산 버전 관리 툴인 깃(Git])(컴퓨터 파일의 변경사항을 추적하고 여러 사용자들 간에 파일들의 작업을 조율하기 위한 분산 버전 관리 시스템(깃 (소프트웨어), 위키피디아)) 저장소 호스팅을 지원하는 웹사이트(깃허브, 위키피디아).

출력된 시에르핀스키 삼각형을 계속 확대하면 다시 같은 모양이 반복되게 된다. 이를 앞서 말한 바와 같이 ‘자기유사성’이라 한다. 이를 확인해보기 위해 n=10 시에르핀스키 삼각형을 계속 확대해보았다. (s는 확대 횟수, 사진 위 빨간 사각형은 다음에 확대할 부분)

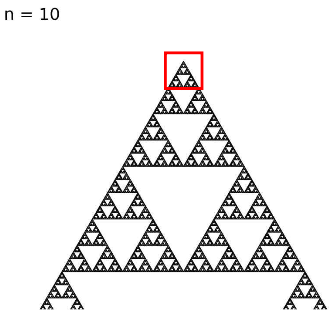
s = 0:



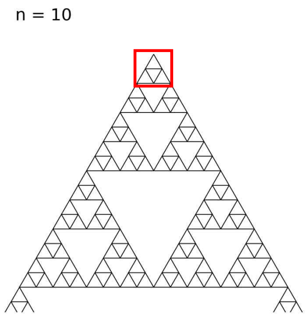
s = 2:



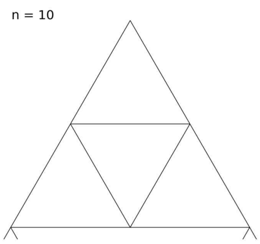
s = 4:



s = 8:



s = 10:



위 그림들과 같이 전체를 삼등분으로 보고 상단 삼각형으로 확대를 할 때마다 n을 1 감소시킨 시에르핀스키 삼각형의 모습과 같아진다. 즉 s가 1 증가할 때마다 n은 1 감소하는 반비례 관계로 시에르핀스키 삼각형은 프랙털의 조건인 자기유사성을 만족하므로 프랙털이 맞음을 확인할 수 있다. 만약 처음 삼각형이 $n=\infty$시에르핀스키 삼각형이라면 해당 삼각형을 무한히 확대할 수 있고 모양은 확대한 횟수와 관계없이 항상 일정할 것이다.

III. 결론

이번 탐구에서 프랙털과 프랙털 차원의 정의, 프랙털 차원을 log를 통해 구하는 방법, 프랙털의 다양한 예시 그리고 대표적인 프랙털 도형인 시에르핀스키 삼각형을 Python과 matplotlib.pyplot 모듈을 이용하여 직접 생성해보고 프랙털의 성질인 자기유사성을 직접 생성된 도형을 확대해보며 확인하여 시에르핀스키 삼각형이 프랙털이 맞다는 사실을 알아냈다.

프랙털은 전체를 부분으로 나누어도 같은, 즉 계속 특정 부분을 확대하여도 계속 같은 모양을 유지하게 되는 도형을 말한다. 이러한 특성을 자기유사성이라 하고 시에르핀스키 삼각형을 직접 확대해보며 자기유사성을 확인한 것과 같이 다른 모든 프랙털 도형도 똑같은 방법으로 특정 부분을 확대하면 무한히 같은 모양을 유지할 것이다.

프랙털 차원은 우리가 평소에 사용하는 차원과는 약간 다르다는 것을 알 수 있었다. 그러나 전체적인 그 원리 자체는 결국 같으며 프랙털 차원 계산식은 log의 성질을 이용하여 풀면 프랙털 차원 계산 과정을 이해할 수 있었다.

이번 탐구를 통해 앞서 말한 많은 것들을 새롭게 배울 수 있었으며 차원의 정의와 개념을 이용하여 프랙털 차원 계산식 $\frac{\log m}{\log n}$을 log의 성질을 이용하여 직접 유도해봤다. 또한 시에르핀스키 삼각형은 프랙털 도형이 맞다는 사실을 프랙털의 정의인 자기유사성을 통해 증명해보았다.

추후 더 많은 수학 개념들을 배우고 난 후 유도나 증명해내지 못한 식들 또한 직접 풀어보고 망델브로 집합, 코흐 곡선을 비롯한 프랙털 도형들을 시에르핀스키 삼각형처럼 직접 Python으로 그려보고 확대하여 자기유사성을 확인해보고 싶다. 또한 만약 기회가 된다면 나만의 프랙털 도형을 수학적 원리를 통하여 만들어 보고 싶다.

Ⅳ. 참고문헌 (각주와 미주로 대체)

1) 수학 Ⅰ p.60 “프랙털 차원”, 금성출판사
2) 프랙탈, 두산백과
3) 차원, 위키피디아
4) 차원, 수학백과
5) 하우스드로프 차원, 위키피디아`
6) "하우스드로프 차원", ChatGPT
7) 프랙털 차원, 수학백과
8) 시에르핀스키 삼각형, 수학백과
9) 시에르핀스키 삼각형, 위키피디아
10) 만델브로 집합, 수학산책
11) 망델브로 집합, 위키피디아
12) 망델브로 집합, 박경미의 수학콘서트 플러스
13) 줄리아 집합, 박경미의 수학콘서트 플러스
14) The Hausdorff dimension of the boundary of the Mandelbrot set and Julia sets, Mitsuhiro Shishikura, Tokyo Institute of Technology & State University of New York at Stony Brook, 12 Apr 1991
15) 코흐 곡선, 두산백과
16) 코크 곡선, 위키피디아
17) Agnes Scott College
18) Matplotlib Tutorial - 파이썬으로 데이터 시각화하기, wikidocs (세부 링크 총 40개)
19) matplotlib.pyplot, matplotlib