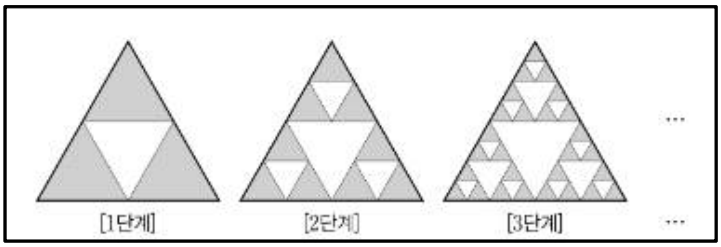
01 프랙탈

기본 개념 프랙탈이란?

일부 작은 조각이 전체와 비슷한 기하학적 형태를 말한다.
이런 특징을 『자기 유사성』이라고 하며 자기 유사성을 갖는 기하학적 구조를 프랙탈 구조라고 한다.
브누아 망델브로가 처음으로 쓴 단어로 어원은 조각났다는 뜻의 라틴어로부터 유래되었다.

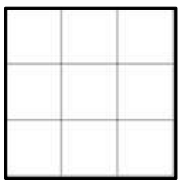
01-1. 프랙탈 기본 내용 정리



프랙탈 차원은 공간에 패턴을 얼마나 조밀하게 채우는지 나타내는 비율이다.
시어핀스키 삼각형의 길이가 1/2인 정삼각형을 각 부분에서 하나씩 제거해 나가는데, 한 번에 2개를 제거한다면 어떨까? 우리 눈에는 구멍이 많이 생겨 더 허전한 도형이 들어올 것이다.
즉, 프랙탈 차원이 클수록 프랙탈 도형이 공간을 점유하는 비율이 높아진다.

01-2. 프랙탈 차원 구하기

프랙탈 차원을 구하는 방법을 알아보자. 먼저 한 변의 길이가 3인 정사각형을 떠올려라. 이 정사각형을 "큰 도형"이라고 하고, 이 정사각형의 각 변을 3등분하여 마주 보는 점끼리 이어서 나타나는 작은 정사각형을 "작은 도형"이라고 하겠다.

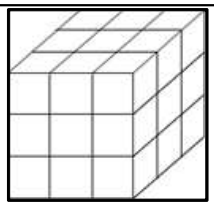


이때 한 변의 길이가 1/3이 되었으므로 3 = m, 작은 도형이 9개 모여 큰 도형이 되므로 9 = N, 구하려는 차원을 d라고 하면 식에 따라 차원을 구할 수 있다. 이 경우 차원(혹은 프랙탈 차원)은 당연히 2이다. 따라서 프랙탈 차원을 구하는 식을 정리하면 다음과 같다.

프랙탈 차원에 관한 식

$$N = m^d, d = \frac{\log N}{\log m}$$

추가



연습을 위해 정육면체의 프랙탈 차원을 구해보자.
이제 한 변의 길이를 3이라고 할 필요 없이, 한 변의 길이가 1/3이 되었으므로 m = 3, 작은 도형이 27개 필요하므로 N = 27이다.

01-3. 실습(파이썬을 이용한 시뮬레이션)

예전에 우리는 액셀을 이용해서 프랙탈을 만들고 그 자기 유사성을 관찰한 적이 있다. 이번에는 파이썬을 이용해서 예전에 만들었던 것을 재현해보자.

실습 패키지 불러오기

```
import numpy as np

import matplotlib.pyplot as plt
```

우선 필요한 패키지들을 불러온다. numpy와 matplotlib라는 패키지를 이용하였다. numpy는 보통 과학 계산을 위한 패키지로 다차원 배열을 처리하는데 유용하게 사용되며 matplotlib는 그래프를 그리는데 주로 이용된다.

import numpy as np에서 as는 numpy라는 문구를 앞으로 np라고 취급하겠다는 의미이다.
보통 파이썬 패키지에서는 "패키지이름.기능1" 이나 "패키지이름.세부목록.기능1" 등으로 함수를 불러오는데 여기서 패키지이름에 들어가는 것이 numpy에서 np로 줄여지는 것이다.

실습 값 정의

```
n = 5  # 초기 입력 값을 정의

width, height = 200, 200 # 만들 프랙탈의 가로, 세로 크기

list1 = np.zeros((width, height))  # n, n칸을 정의1
list2 = np.zeros((2, width*height))  # n^2개의 점을 정의
```

패키지를 불러오는 코드를 작성한 뒤로는 리스트를 2개를 제작한다.
numpy 안에 있는 zeros() 함수를 이용한다.

The function `zeros` creates an array full of zeros, the function `ones` creates an array full of ones, and the function `empty` creates an array whose initial content is random and depends on the state of the memory. By default, the dtype of the created array is `float64`, but it can be specified via the key word argument `dtype`.

```
>>> np.zeros((3, 4))
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
```

공식문서에 의하면 zeros 함수는 내부에 (a, b) 형식으로 값을 전달하면 a x b 크기의 배열을 반환한다.
여기서는 list1 = np.zeros((width, height)) 라는 명령을 통해 반환된 배열을 list1에 할당하고 있다.

실습 그래프 설정

```
# 기본적으로 해주어야 할 것들
figure, axes = plt.subplots()

axes.set_aspect(1)

# 축 범위 설정
plt.xlim(-0.1, 1.1)
plt.ylim(-0.1, 1.1)

# 축 이름
plt.xlabel("X")
plt.ylabel("Y")

# 그래프 이름
plt.title("Fractal1")
```

matplotlib 안에 있는 기능들을 활용해서 그래프를 출력하기 위한 기본 설정들을 해준다. 그래프에는 그래프 이름, 축 범위, 축 이름 등이 필수적으로 들어가야 한다. 위의 내용은 조금씩 바꿀 수는 있지만 기본 골격은 유지될 것이니 외우거나 따로 저장해두어도 좋다.

실습 나누기 연산을 통한 프랙탈 구현

```
for ii in range(width-1):
    for jj in range(height-1):
        if ii != 0 and jj != 0:
            list1[ii][jj] = (list1[ii-1][jj] + list1[ii][jj-1] + list1[ii-1][jj-1]) % n  # mod()
        else:
            list1[ii][jj] = 1
```

이 부분이 실제로 중요한 부분이다. 액셀로 했었던 실습을 파이썬으로 다시 구현한 부분이다.

범위

range 형은 숫자의 불변 시퀀스를 나타내며 for 루프에서 특정 횟수만큼 반복하는 데 흔히 사용됩니다.

class range(*stop*)
class range(*start*, *stop*[, *step*])

The arguments to the range constructor must be integers (either built-in int or any object that implements the __index__() special method). If the *step* argument is omitted, it defaults to 1. If the *start* argument is omitted, it defaults to 0. If *step* is zero, ValueError is raised.

for [문자] in range([범위]): 는 범위 내의 정수를 반환하는 함수이며 start가 빠진 경우(실습의 코드에도 start는 없고 stop만 있다) 0부터 정수를 반환한다 명시되어 있다. 즉, for ii in range(width-1): 은 0부터 width-1까지 총 width 개 만큼 반복문(for)를 돌며 코드를 순회한다는 뜻이다.

액셀로 구현한 프랙탈에 대한 기억을 떠올려보면 첫행, 첫열은 전부 1로 채웠으므로 만약 if ii !=0 and jj !=0이 아니면(else), 즉 행이나 열 중 하나라도 0번째라면 1을 채우고 나머지의 경우에는 list1[ii][jj] = (list1[ii-1][jj] + list1[ii][jj-1] + list1[ii-1][jj-1]) % n 연산을 수행한다. 이 또한 액셀로 구현한 프랙탈과 완벽히 동일한 구조이다. n, m번째 칸을 채우는 숫자는 (n-1, m), (n, m-1), (n-1, m-1) 번째 숫자의 합을 n(처음에 받은 숫자)로 나눈 나머지여야 한다는 것을 기억하자.

실습 프랙탈의 크기 줄이기

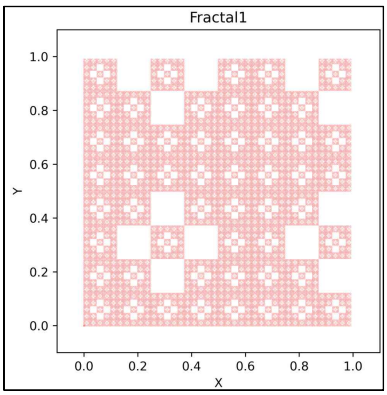
```
count = 0
for ii in range(width-1):
    for jj in range(height-1):
        if list1[ii][jj] != 0:
            list2[0][count] = ii / width
            list2[1][count] = jj / height
            count += 1
```

위의 방식대로 프랙탈을 제작하면 그래프에 점을 찍었을 때 간격이 넓어서 프랙탈이 제대로 안보일 수도 있다. 그럴기에 위와 같은 코드를 입력해서 점 사이의 간격을 좁혀준다. 위의 코드는 이해하지 않아도 상관없다. 이때, list2[0][n]에는 x좌표를, list2[1][n]에는 y좌표를 넣었으며 이는 scatter 함수를 이용하기 위함이다.

실습 출력

```
plt.scatter(list2[0], list2[1], color="red", s=0.01)

plt.show()
```



matplotlib 내의 scatter라는 함수를 이용해서 리스트 내의 좌표에 대한 점을 전부 찍은 결과이다.

함수는 scatter(x좌표의 리스트, y좌표의 리스트, [여러 인자들])로 구성되어 있다.

여기서 plt.show() 대신 plt.savefig('result.png', dpi=600)를 이용하면 파이썬 파일이 있는 경로로 result.png 라는 이름의 이미지 파일이 저장된다.

실습 전체 코드

```
import numpy as np

import matplotlib.pyplot as plt

n = 5  # 초기 입력 값을 정의

width, height = 200, 200 # 만들 프랙탈의 가로, 세로 크기

list1 = np.zeros((width, height))  # n, n칸을 정의1
list2 = np.zeros((2, width*height))  # n^2개의 점을 정의

# 기본적으로 해주어야 할 것들
figure, axes = plt.subplots()

axes.set_aspect(1)

# 축 범위 설정
plt.xlim(-0.1, 1.1)
plt.ylim(-0.1, 1.1)

# 축 이름
plt.xlabel("X")
plt.ylabel("Y")

# 그래프 이름
plt.title("Fractal1")

for ii in range(width-1):
    for jj in range(height-1):
        if ii != 0 and jj != 0:
            list1[ii][jj] = (list1[ii-1][jj] + list1[ii][jj-1] + list1[ii-1][jj-1]) % n  # mod()
        else:
            list1[ii][jj] = 1

count = 0
for ii in range(width-1):
    for jj in range(height-1):
        if list1[ii][jj] != 0:
            list2[0][count] = ii / width
            list2[1][count] = jj / height
        count += 1

plt.scatter(list2[0], list2[1], color="red", s=0.01)

plt.show()
```