# Фракталы

**Фракта́л** ([лат.](https://ru.wikipedia.org/wiki/%D0%9B%D0%B0%D1%82%D0%B8%D0%BD%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *fractus* — дроблёный, сломанный, разбитый) — [множество](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE), обладающее свойством [самоподобия](https://ru.wikipedia.org/wiki/%D0%A1%D0%B0%D0%BC%D0%BE%D0%BF%D0%BE%D0%B4%D0%BE%D0%B1%D0%B8%D0%B5) (объект, в точности или приближённо совпадающий с частью себя самого, то есть целое имеет ту же форму, что и одна или более частей).

Термин «фрактал» введён Бенуа Мандельбротом в 1975 году и получил широкую известность с выходом в 1977 году его книги «Фрактальная геометрия природы». Особую популярность фракталы обрели с развитием компьютерных технологий, позволивших эффектно визуализировать эти структуры.

Многие объекты в природе обладают свойствами фрактала. На самом деле природные объекты являются квазифракталами. Они отличаются от идеальных абстрактных фракталов неполнотой и неточностью повторений структуры. Границы облаков, линия берега, деревья, листья растений, кораллы) являются квазифракталами, поскольку на некотором малом масштабе фрактальная структура исчезает. Природные структуры не могут быть идеальными фракталами из-за ограничений, накладываемых размерами живой клетки и, в конечном итоге, размерами молекул.

Классический пример природного фрактала – кочан капусты сорта Романеско( рис\_\_).



Рис.\_\_Фрактальная форма кочана капусты сорта Романеско (Brassica oleracea)

Самые известные самоподобные геометрические фигуры:

* треугольник Серпинского («скатерть»);
* ковёр Серпинского;
* губка Менгера(«ковер серпинского в трехмерном пространстве) ( рис \_\_
* кривая Коха;
* 
* Рис.\_\_ 5 итераций построения губки Менгера.

В данной работе я остановлюсь на треугольнике Серпинского.

# Треугольник Серпинского

Этот фрактал описал в 1915 году польский математик Вацлав Серпинский. Чтобы его получить, нужно взять (равносторонний) треугольник с внутренностью, провести в нём средние линии и вырезать центральный из четырех образовавшихся маленьких треугольников. Дальше эти же действия нужно повторить с каждым из оставшихся трех треугольников, и т. д.

Вырезание центральных треугольников — не единственный способ получить в итоге треугольник Серпинского. Можно строить «наоборот». Берем изначально «пустой» треугольник, затем строим в нём треугольник, образованный средними линиями, затем в каждом из трех угловых треугольников делаем то же самое, и т. д. Поначалу фигуры будут сильно отличаться, но с ростом номера итерации они будут всё больше походить друг на друга.

Для наглядной демонстрации алгоритма я написала программу на Python, реализующую второй алгоритм. Для построения воспользуемся модулем Python Turtle.

Создадим массив массивов. Для простоты назовем его двумерным массивом ( рис.1). В нем мы будем хранить координаты вершин треугольников. Заполним начальными координатами.

|  |  |  |
| --- | --- | --- |
|  | X | Y |
| A | -350 | -250 |
| B | 0 | 350 |
| C | 350 | -250 |

Рис.1.Двумерный массив для хранения координат

Напишем две функции:

getmid(array1 array2) – Получает на вход два массива координат отрезка и возвращает координаты середины отрезка.

triangle(points,depth) – получает на вход массив координат points т глубину вложенности depth и строит по ним треугольник. Далее рекурсивно вызывает саму себя и строит вложенные треугольники.

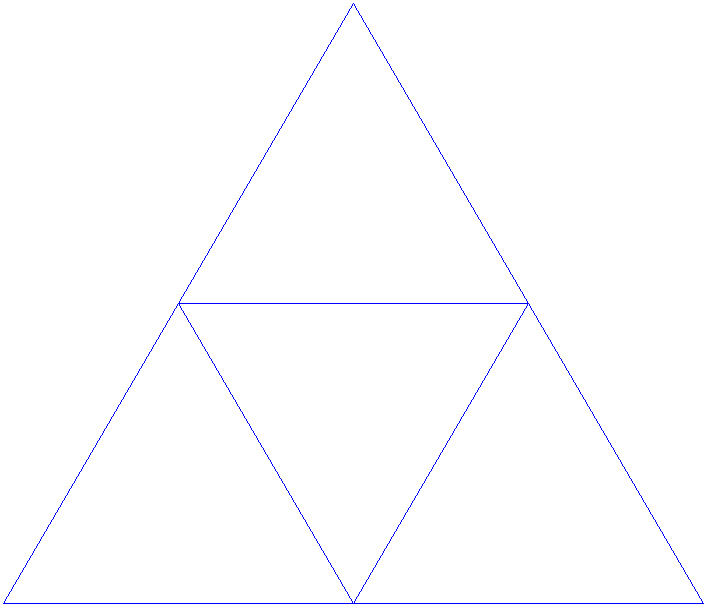


Рис2. Запуск функции triangle c Depth=1

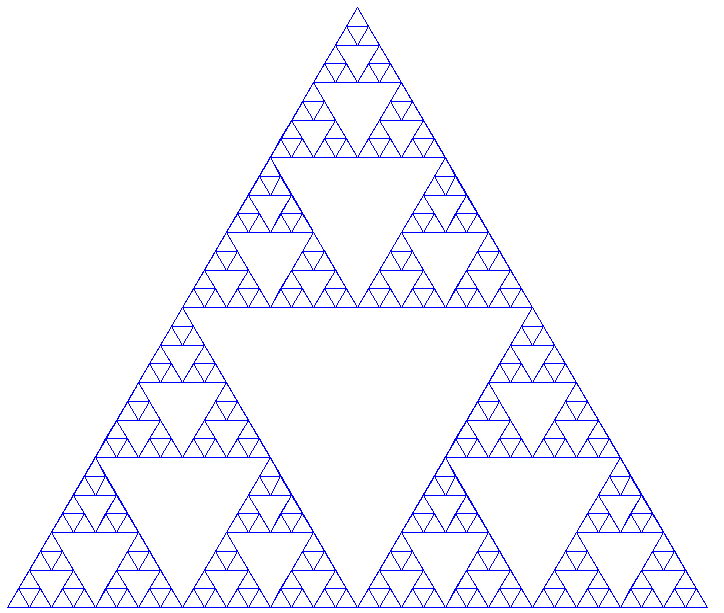


Рис.3.Запуск функции triangle c depth=5

В источнике 1 приведен интересный факт: если в треугольнике Паскаля все нечётные числа окрасить в чёрный цвет, а чётные — в белый, то образуется треугольник Серпинского. Я решила проверить этот факт и написала программу, которая строит треугольник Паскаля.

В программе реализованы 2 функции.

pif(N) – Функция, возвращающая N-строчку треугольника Паскаля.

Pr\_str(row,size) – Функция, рисующая на canvas строчку треугольника Паскаля.

Сама программа в цикле выводит строки треугольника. Листинг программы представлен в приложении 2.



Рис\_\_ Результат вывода программы paskal.pl

На рис.\_\_ представлен вывод программы для 64 строк треугольника Паскаля.

# Источники:

<https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B0%D0%BA%D1%82%D0%B0%D0%BB>

<https://elementy.ru/posters/fractals/Sierpinski>

*А. А. Кириллов.* [Повесть о двух фракталах](http://www.mccme.ru/dubna/2007/notes/kirillov-preprint.pdf). — Летняя школа «Современная математика». — Дубна, 2007.

Приложение 1

# Листинг программы Serp.py

import turtle

pensil = turtle.Turtle()

pensil.ht()

pensil.pencolor('blue')

pensil.speed(250)

points = [[-350,-250],[0,350],[350,-250]] #size of triangle

def getMid(p1,p2):

return ( (p1[0]+p2[0]) / 2, (p1[1] + p2[1]) / 2) #find midpoint

def triangle(points,depth):

pensil.up()

pensil.goto(points[0][0],points[0][1])

pensil.down()

pensil.goto(points[1][0],points[1][1])

pensil.goto(points[2][0],points[2][1])

pensil.goto(points[0][0],points[0][1])

if depth>0:

triangle([points[0],

getMid(points[0], points[1]),

getMid(points[0], points[2])],

depth-1)

triangle([points[1],

getMid(points[0], points[1]),

getMid(points[1], points[2])],

depth-1)

triangle([points[2],

getMid(points[2], points[1]),

getMid(points[0], points[2])],

depth-1)

triangle(points,5)

Приложение 2

# Листинг программы paskal.py

import turtle

def pif(depth):

if depth == 1:

return ["1"]

else:

row=pif(depth-1)

tmp=["1"]

for i in range(0,depth-2):

tmp.append ( str( int(row[i])+int(row[i+1]) ) )

tmp.append("1")

return tmp

def pr\_str(row,fSize):

for i in range(0,len(row)):

if (int(row[i])%2==0):

pensil.pencolor('blue')

else:

pensil.pencolor('red')

pensil.write(row[i], font=("Arial", fSize-2, "normal"))

pensil.up()

pensil.goto(pensil.xcor()+fSize\*2,pensil.ycor())

pensil = turtle.Turtle()

turtle.screensize(800,800)

pensil.ht()

pensil.pencolor('blue')

pensil.speed(250)

for i in range(1,60):

pensil.up()

pensil.goto(0-i\*8,400-i\*10)

pensil.down()

pr\_str(pif(i),8);