

Санкт-Петербургский государственный университет

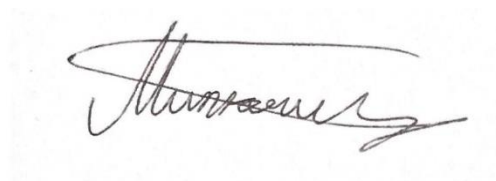
Прикладная математика и информатика

Научно-исследовательская работа

Построение фрактальных множеств на языке Python

Выполнил:

Милюшков Георгий Геннадьевич

A handwritten signature in dark ink, appearing to read 'Милюшков', is centered within a light gray rectangular box.

Научный руководитель:

к.ф.-м.н., PhD, с.н.с.

Мокаев Руслан Назирович

Кафедра прикладной кибернетики

Санкт-Петербург

2021

Введение

Фрактал — это множество со свойствами самоподобия, то есть объект в точности или приближенно совпадающий с частью самого себя. Это понятие появилось в конце 70-х и образовано от латинского слова *fractus* в переводе означающий *состоящий из фрагментов*. В 1975 году это понятие было предложено известным польско-французским математиком Бенуа Мендельбротом для обозначения нерегулярных, но самоподобных структур, которыми он занимался. Широкую известность это слово получило в 1977 году, после выхода его книги «The Fractal Geometry of Nature». Строгого определение этого понятия не существует, хотя Мендельброт определил его следующим образом: «Фракталом называется структура, состоящая из частей, которые в каком-то смысле подобны целому».

Фракталы часто встречаются в повседневной жизни. В природе в большинстве случаев встречаются квазифракталы, то есть фракталы, на которых фрактальная структура пропадает при определённом масштабе. Они не могут быть идеальными фракталами из-за ограничений, накладываемых размерами клеток и молекул. Примером квазифракталов в живой природе являются деревья, листья растений, кораллы, морские звезды и ежи. В неживой природе примером являются береговые линии, горные хребты, снежинки, облака, молнии и кристаллы. Кроме этого, существуют фрактальные антенны в радиотехнике, алгоритм фрактального сжатия изображений в информатике и много других областей, в которых используется фрактальные множества.

Глава 1. Фракталы в математике

Фракталы, удобно представлять в виде математических объектов, задаваемых рекуррентно. В данной работе будут представлены некоторые известные фракталы: Дракон Хартера-Хейтуэя, снежинка Коха, ковер Серпинского, множество Жюлиа и множество Мандельброта.

Снежинка Коха

Снежинка Коха является одним из самых известных геометрических фракталов. Этот фрактал был изобретен известным шведским математиком Нильсом Фабиан Хельге фон Кохом в 1904 году. Он состоит из трех копий *кривой Коха*. Данная кривая была

придумана, как пример непрерывной линии, к которой нельзя провести касательную в любой точке.

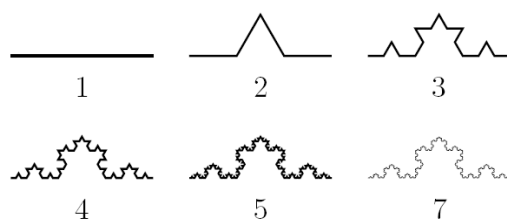


Рисунок 1, Кривая Коха. Итерации 1-7

Рис. 1 иллюстрирует итерации кривой Коха. На первом этапе рисуется непрерывный отрезок, который разделяется на три равные части. Далее, средний интервал удаляется и на его месте строится равносторонний треугольник без этого сегмента. В результате данной операции получается ломанная состоящая из четырех отрезков каждой длины $1/3$. Первый отрезок является $1/3$ изначального отрезка, далее вверх под 120° углом идет новый отрезок, затем под 300° углом вниз идет очередной отрезок и завершается последней $1/3$ изначального отрезка. При последующих итерациях данная операция выполняется на каждой прямой. То есть, на следующем шаге данные «горки» будут нарисованы на каждой из четырех сторон длины $1/3$. При многочисленном выполнении данной операции получается кривая как проиллюстрировано в рисунке 1 в седьмой итерации. Снежинка Коха состоит из трех копий данной кривой. Это означает что на первой итерации снежинка Коха состоит не из прямого отрезка, а из равностороннего треугольника. Далее, при последующих итерациях к каждой стороне треугольника применяются вышеописанные операции одновременно. Программа, отрисовывающая Снежинку Коха, приведена в Приложении А1, а результат работы программы в Приложении А2.

Дракон Хартера-Хейтуэя

Фрактал «Дракон Хартера-Хейтуэя», также известен как «Кривая дракона»-известный фрактал, который получается рекурсивными методами. Этот фрактал был впервые исследован физиками Джоном Хейтуэем, Брюсом Бэнкосом и Вильямом Хартером. Он часто представляется в виде системы Линдемайера, который состоит из *алфавита*, *аксиомы* и множества *правил*. Этот фрактал может быть записан с параметрами:

- Угол равен 90°
- Начальная строка FX

- Правила:
 - $X \rightarrow X+YF$
 - $Y \rightarrow FX-Y$



Рисунок 2, Фрактал «Кривая дракона». Итерации 0-5.

Рис. 2 иллюстрирует первые пять итераций «кривой дракона». Первая итерация получается при добавлении отрезка, поворота его на 90 градусов вокруг одной из вершин, и добавлении полученного отрезка к исходному. Этот алгоритм повторяется при последующих итерациях. Фигура поворачивается на 90 градусов вокруг одной из вершин и добавляется к исходной. При многочисленном выполнении данного алгоритма фигура становится похожей на дракона, именно по этой причине фрактал называется «Кривая дракона» или «Дракон Хартера-Хейтуэя». Программа, отрисовывающая данный фрактал, приведена в Приложении Б1, а результат работы программы в Приложении Б2.

Ковер Серпинского

Этот фрактал часто считается одним из ключевых геометрических фракталов для иллюстрации самоподобия. Это двумерный фрактал, изобретенный польским математиком Вацлавом Серпинским.

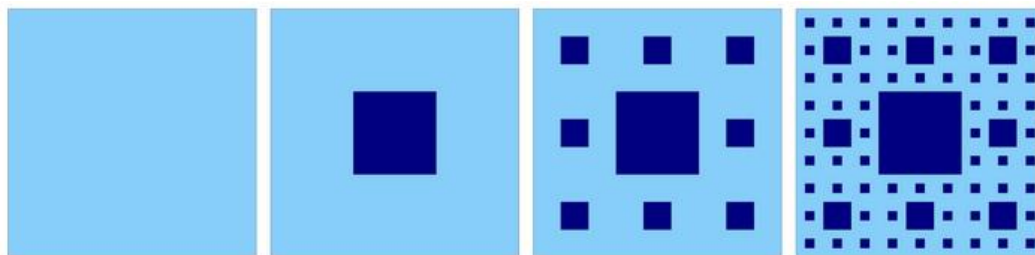


Рисунок 3, Фрактал «Ковер Серпинского». Итерации 0-3.

Первые три итерации данного фрактала проиллюстрированы в Рис. 3. Ковер Серпинского строится рекурсивно. На первом шаге квадрат разделяется на девять равных квадратов (3 x 3). Затем центральный квадрат удаляется. На Рис. 3 это проиллюстрировано вторым изображением, где центральный квадрат имеет другой цвет.

На следующем шаге- каждый из восьми оставшихся квадратов также разделяется на девять других 3×3 , а затем центральные удаляются. Это проиллюстрировано на Рис. 3, третьем изображении, где было перекрашено восемь центральных квадратов. Эта операция может рекурсивно продолжаться бесконечно, где вокруг определенного центрального квадрата строится девять других квадратов меньшего размера, а затем удаляется центральный квадрат. Программа, отрисовывающая Ковер Серпинского, приведена в Приложении В1, а результат работы программы в Приложении В2.

Кроме ковра Серпинского существует также треугольник Серпинского, где вместо квадратов рисуется перевернутые равносторонние треугольники. Алгоритм построения треугольника Серпинского действует похожим образом. Данный фрактал является не менее известным, чем ковер Серпинского. Программа, отрисовывающая данный фрактал приведена в Приложении В3, а результат работы программы в Приложении В4.

Существует также аналог ковра Серпинского в трехмерном пространстве. Такой фрактал называется губка Менгера.

Множество Мандельброта

Этот фрактал был предложен основоположником понятия фрактала — Бенуа Мандельбротом. Данный фрактал отличается от предыдущих тем, что он не является геометрическим. Множество Мандельброта — это множество таких точек t

на комплексной плоскости, для которых рекуррентное соотношение $z_{n+1} = z_n^2 + t$, при $z_0 = 0$ задает ограниченную последовательность.

Возможно рассмотреть пример множества, для которого заданно рекуррентное соотношение $z_{n+1} = z_n^2$. Эта последовательность будет ограничена в том и только том случае если взять z_0 с модулем меньше единицы. При всех остальных случаях данная последовательность будет стремиться к бесконечности.

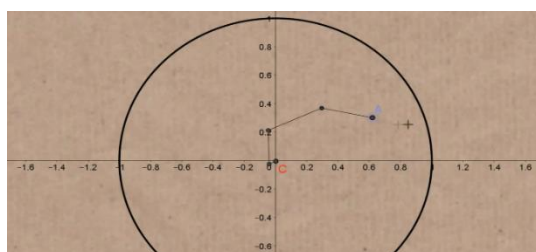


Рисунок 4, границы комплексных чисел при которых последовательность ограничена, и пример точ.А, которая стремиться к 0.

Рис. 4 иллюстрирует границы ограниченной последовательности с рекуррентным соотношением $z_{n+1} = z_n^2$ и пример последовательности точки A которая стремиться к нулю. Данные границы являются окружностью с единичным радиусом. Границы фрактала с рекуррентным соотношением $z_{n+1} = z_n^2 + t$, где $z_0 = 0$, не являются столь очевидными.

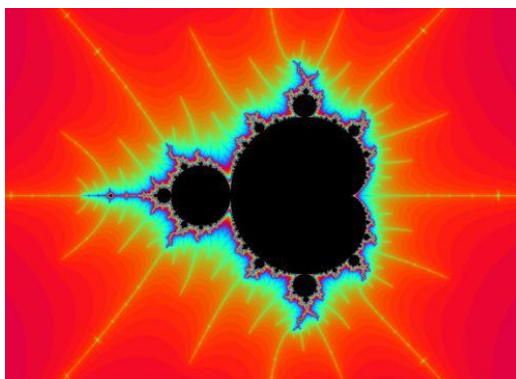


Рисунок 5, Множество Мандельброта

Такие границы иллюстрирует множество Мандельброта, которое представлено на Рис. 5. Оно показывает границы для значения t , таким образом, чтобы последовательность была ограничена. Цвета предназначены для представления на сколько быстро значение последовательности стремится к бесконечности. Данное множество является фракталом, так как при приближении в любую часть этой фигуры, она будет напоминать исходное множество. Этот фрактал является одним из самых известных фракталов, и также является популярным вне математики благодаря своим цветным визуализациям. До сих пор точная площадь данного фрактала неизвестна. Программа, отрисовывающая данный фрактал приведена в Приложении Г1, а результат работы программы в Приложении Г2.

Множество Жюлиа

Множество Жюлиа—фрактал, изобретенный французским математиком Гастоне Жюлиа и Пьером Фату, является очень похожим фракталом на множество Мандельброта. Они очень схожи по принципу построения, хотя иллюстрируют разные вещи. Множество Мандельброта в отличии от множества Жюлиа строит фрактал с рекуррентным соотношением $z_{n+1} = z_n^2 + t$ и в качестве результата рассматриваются границы произвольных комплексных значений t , таковых, чтобы последовательность была ограничена. Эта последовательность начинается со значением $z_0 = 0$. В множестве Жюлиа строится фрактал, который показывает границы ограниченной последовательности для всех z_0 с определенным заданным комплексным числом t . По

этой причине множество Мандельброта представляется единственным образом, в отличие от множества Жюлиа, которое показывает бесконечное количество разных границ для разных комплексных чисел t . Границей при $t = 0$ будет окружность с единичным радиусом. Программа, отрисовывающая множество Жюлиа приведена в Приложении Д1, а результат работы программы в Приложении Д2.

Глава 2. Язык программирования Python

Python является одним из самых известных языков программирования. Данный язык является высокоуровневым объектно-ориентированным языком, разработанным голландским программистом Гвидо ван Россум в 1991 году. Такое название этот язык получил в честь популярного комедийного британского шоу «Monty Python's Flying Circus». Изначально Python был скриптовым языком для одной распределенной операционной системы.

Python является одним из самых используемых языков программирования. Он часто применяется в работе с большими данными, разработке мобильных игр и сайтов, в сложных вычислительных процессах, в машинном обучении и в автоматизации различных процессов. Одним из главных достоинств языка Python является его логичность и простота. Данный язык является интерпретируемым, то есть код интерпретируется без предварительной компиляции. Python используется, как ранее упомянуто, во многих отраслях, но особенно силен данный язык является в работе с данными. Также, из-за его логичности и простоты, данный язык зачастую преподается как первый язык для изучения и знакомством с программированием.

Python имеет свои отличительные особенности, например, кроссплатформенность. Python удачно запускается на любой операционной системе, так как его интерпретаторы существует на многих платформах. К тому же Python успешно работает на многих средах разработки и различных сервисах, что дает программисту больше возможностей. Не менее известная особенность является возможность подключения различных библиотек из языка C.

У Python есть определенные преимущества относительно других языков программирования. Одним из ключевых преимуществ является легкий синтаксис. Для написания первой программы не нужно изучать синтаксис и особенности Python, в такой степени, как для других языков программирования. Например, в Python отсутствуют

операторные скобки. Сам язык построен очень логически и название различных функций и команд близки к человеческому языку, что позволяет программисту легче запоминать и использовать данные команды. Значительным преимуществом является динамическая типизация. Программисту не нужно указывать тип данных. В Python также удобный возврат нескольких значений функции. Автоматическое выделение памяти ускоряет процесс написания кода, но уменьшает контроль программиста над программой. Нельзя не упомянуть о сборщике мусора, который позволяет оптимизировать использованную память. Еще одним преимуществом является привязка типа данных. Тип данных привязан к значению, а не к переменной, это значительно упрощает код.

Эти преимущества делают данный язык очень удобным для начинающего программиста, но у этих плюсов есть и свои минусы. Полная динамическая типизация и автоматическое управление памятью значительно способствуют замедлению программы. При больших проектах Python не является лидером по скорости работы программы относительно других языков. Также обратной стороной преимуществ Python является меньший контроль программиста над программой.

Python используется почти в любом среднем или крупном проекте. Несмотря на то, что он не новый язык, он постоянно обновляется и способен на решение почти любой задачи.

Глава 3. Библиотеки языка Python

NumPy

NumPy (*Numerical Python*), было изобретено в 2005 году Трависом Олифиантом. Данная библиотека состоит из объектов многомерного массива и наборов процедур для их обработки. Используя NumPy возможно выполнять математические и логические операции над массивами. Также программисту доступны операции преобразования Фурье и процедуры для манипуляции с формой. Кроме этого, NumPy позволяет выполнять операции, связанные с линейной алгеброй. Эта библиотека имеет встроенные функции для линейной алгебры и генерации случайных чисел. Одной из самых используемых функций является `numpy.array()`. Это один из самых простых способов создать массив из обычных списков. Данная функция трансформирует вложенные последовательности в многомерные массивы. При работе с массивами часто используется функция `numpy.arange()`. Данная функция возвращает одномерный массив с равномерно разнесенными значениями внутри заданного интервала. Например,

функция `np.arange(10,30,5)` вернет массив `[10,15,20,25]`. Чаще всего используются малые значения между числами в интервале, чтобы создать много точек для графика.

Matplotlib

При реализации программ, отрисовывающих данные фракталы, основной библиотекой является `matplotlib`. Это библиотека для создания двумерных графиков из данных в массивах. Она является очень большой и состоит из около 70 000 строк кода. `Matplotlib` написан на языке Python и использует `NumPy`. Данная библиотека была написана и поддерживалась в основном Джоном Хантером. Она построена на принципах объектно-ориентированного программирования.

Данная библиотека не единственная графическая библиотека, но у нее есть определенные преимущества, которые ее выделяют на фоне остальных. Она позволяет легко создавать графики программисту, который только начинает знакомиться с языком Python. К тому же выходной результат можно сохранить в высоком качестве в форматах PNG, PDF, SVG, EPS и PGF. Одним из главных преимуществ является то, что фигуры представленные `matplotlib`, очень подходят научным публикациям, так как они легко контролируются и управляются программистом.

Одним из наиболее часто используемых модулей является `pyplot` (часто сокращается и пишется `plt`). Одной из ключевых функций является `plt.plot`. В эту функцию подается два массива со значениями координат `x` и `y`. Часто добавляется третий элемент `'—'`, который означает что между точками проводится непрерывная линия. Python рисует только прямые линии и по этой причине, чтобы представить кривую, надо нарисовать много коротких прямых между ее точками.

Turtle

`Turtle` является менее известной библиотекой, но широко применяется при рисовании различных фигур. Это векторная графика, где выходной график получается при командах в какую сторону передвигать ручку рисования. Пример использования данной библиотеки представлен в приложении Б1. Данную библиотеку не нужно отдельно устанавливать, так как она является частью стандартного пакета Python, доступного «из коробки». Нужно только подключить эту библиотеку к программе командой `import turtle`. В отличие от `matplotlib`, данная библиотека не рисует прямые между определенными точками, а через базовые движения: вперед, назад, влево и вправо. При

фракталах, где тяжело задать все точки, по которым должен строиться фрактал, удобнее использовать turtle.

PIL

PIL(*Python Imaging Library*) является библиотекой в Python для работы с растровой графикой, то есть с изображениями, построенными через многочисленные пиксели. Данная библиотека является активно используемой, но разработка данной библиотеки была прекращена в 2011 году. PIL поддерживает очень многие форматы изображений, что делает эту библиотеку удобной. Код, отрисовывающий множества Мандельброта, представленный в приложении Г1, написан с помощью данной библиотеки для раскраски разных зон фрактала. PIL позволяет почти любые операции связанные с обработкой изображений. Например, попиксельно манипуляции, добавления текста к изображению, многочисленные способы улучшения и редактирования изображений.

Список литературы

1. Skrindo Knut, Innføring i Python: учеб.пособие/ 2019, Kopinor-52 с.
2. Saha Amit, Doing math with Python /2015, No starch Press, USA, 265 стр. URL:
<http://index-of.es/Varios-2/Doing%20Math%20with%20Python.pdf>
3. Kenneth Falconer, Fractal Geometry /1990, Sons Ltd, England, West Sussex, 165 стр.
URL: http://inis.jinr.ru/sl/vol2/Mathematics/Falconer,_Fractal_Geometry,1990.pdf

Приложение A1

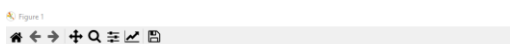
```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 import math
4
5 def koch(a,b,iterations):
6     #Starting point of our original line
7     a1=a[0] #0
8     a2=a[1] #0
9
10    #End points of our original line
11    b1=b[0] #1
12    b2=b[1] #0
13    #pod kakim naklonom stait nasha baza, chtobi potom dobovljat
14    znachenia esli ona pod uglom
15    theta = np.arctan((b2-a2)/(b1-a1))
16    length = np.sqrt((a1-b1)**2+(a2-b2)**2)
17
18    #first nearest point from a
19    c1 = (2*a1+b1)/3.
20    c2 = (2*a2+b2)/3.
21    c = [c1,c2]
22    #second last point from a
23    d1 = (a1+2*b1)/3.
24    d2 = (a2+2*b2)/3.
25    d = [d1,d2]
26
27
28    if c1 >= a1:
29        m1 = c1 + (length/3.)*math.cos(theta+math.pi/3.)
30        m2 = c2 + (length/3.)*math.sin(theta+math.pi/3.)
31    else:
32        m1 = c1 + (length/3.)*math.cos(theta-2*math.pi/3.)
33        m2 = c2 + (length/3.)*math.sin(theta-2*math.pi/3.)
34    m = [m1,m2]
35
36    c = np.array(c)
37    d = np.array(d)
38    m = np.array(m)
39
40    points = []
41
42    if iterations == 0:
43        points.extend([a,b])
44    elif iterations == 1:
45        points.extend([a, c, m, d, b])
46    else:
47        points.extend(koch(a,c,iterations-1)) #Rekursija
48        points.extend(koch(c,m,iterations-1))
49        points.extend(koch(m,d,iterations-1))
50        points.extend(koch(d,b,iterations-1))
```

```

51
52     return points
53
54
55 n=int(input("Глубина рекурсии:"))
56 plt.figure(figsize=(10,7))
57
58 points1 = koch(a=np.array([0, 0]),b=np.array([0.5,1]),iterations=n)
59 ptsx1=[]
60 ptsy1=[]
61 for i in range(len(points1)):
62     ptsx1.append(points1[i][0])
63     ptsy1.append(points1[i][1])
64 plt.plot(ptsx1, ptsy1, '-')
65 plt.axis('equal')
66 plt.axis('off')
67
68 points2 = koch(a=np.array([0.5, 1]),b=np.array([1,0]),iterations=n)
69 ptsx2=[]
70 ptsy2=[]
71 for i in range(len(points2)):
72     ptsx2.append(points2[i][0])
73     ptsy2.append(points2[i][1])
74 plt.plot(ptsx2, ptsy2, '-')
75 plt.axis('equal')
76 plt.axis('off')
77
78 points3 = koch(a=np.array([1, 0]),b=np.array([0,0]),iterations=n)
79 ptsx3=[]
80 ptsy3=[]
81 for i in range(len(points3)):
82     ptsx3.append(points3[i][0])
83     ptsy3.append(points3[i][1])
84 plt.plot(ptsx3, ptsy3, '-')
85 plt.axis('equal')
86     plt.axis('off')

```

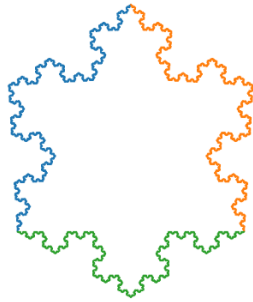
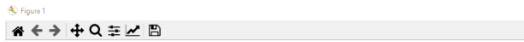
Приложение A2



Глубина рекурсии 0



Глубина рекурсии 2



Глубина рекурсии 8

Приложение Б1

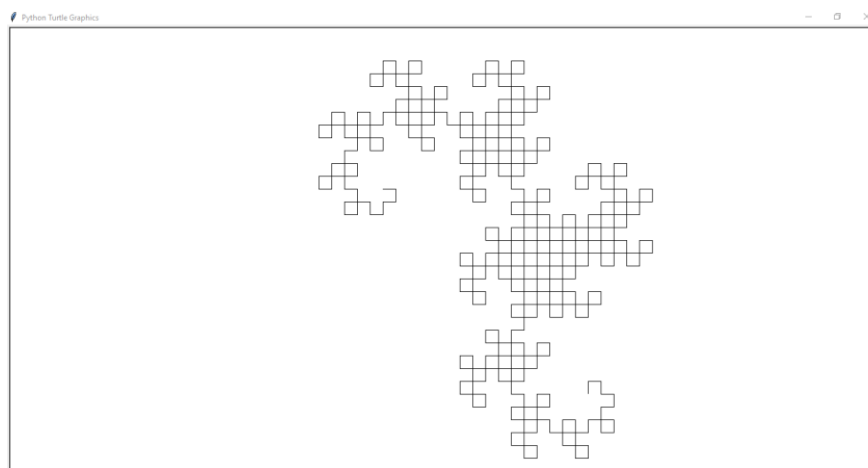
```
1 import turtle
2
3 r = 'r'
4 l = 'l'
5
6 turtle.ht()
7 turtle.speed(0)
8 turtle.penup()
9 turtle.goto(-100,100)
10 turtle.pendown()
11 turtle.color("black")
12 turtle.bgcolor("white")
13
14
15 old = r
16 new = old
17
18 length=20
19 cycle=1
20
21 iteration=int(input('Enter iteration:'))
22
23 while cycle<iteration:
24
25     new = (old) + (r)
26
27     old = old[::-1]
28
29     #probegaem kazhdoe znachenije v perevernutom
30     for char in range(0,len(old)):
31         if old[char] == r:
32             old = (old[:char]) + (l) + (old[char+1:])
33         elif old[char] == l:
34             old = (old[:char]) + (r) + (old[char+1:])
35
36     new = (new) + (old)
```

```

37     old = new
38
39
40     cycle=cycle+1
41
42
43 turtle.forward(length)
44
45 for char in range(0,len(new)):
46     if new[char] == (r):
47         turtle.right(90)
48         turtle.forward(length)
49     elif new[char] == (l):
50         turtle.left(90)
51         turtle.forward(length)
52
53 turtle.exitonclick()

```

Приложение Б2

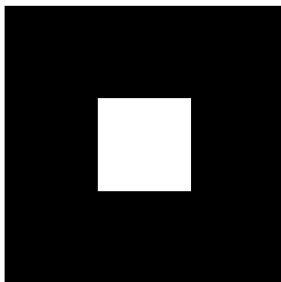
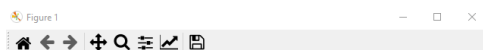


Итерация 9- дракон Хартера Хейтуэя

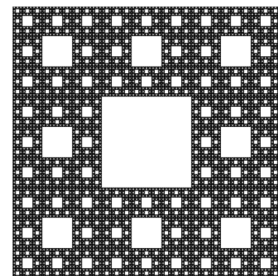
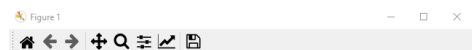
Приложение В1

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4
5 def funcija(x1, y1, size, img):
6     for x in range(x1, x1+size):
7         for y in range(y1, y1+size):
8             img[x, y] = 0
9     return img
10
11
12 glubina= int(input("Level: "))
13 size = 3**glubina
14
15 img = np.ones((size, size), dtype=np.uint8)
16
17 for i in range(1, glubina+1):
18     square_size = int(size/(3**i))
19     for x in range(0, 3**i, 3):
20         x = int((x+1)*square_size)
21         for y in range(0, 3**i, 3):
22             y = int((y+1)*square_size)
23             img = funcija(x, y, square_size, img)
24
25
26 plt.axis('off')
27 plt.imshow(img, cmap='binary')
28 plt.imsave('Kover Serpinskogo', img, cmap='binary')
29 plt.show()
```

Приложение В2



Глубина рекурсии 1



Глубина рекурсии 6

Приложение В3

```
1 import matplotlib.pyplot as plt
2
3
4 def podder(kol,x1,y1,x2,y2,x3,y3,n):
5     plt.plot([x1,x2],[y1,y2], 'k')
6     plt.plot([x1,x3],[y1,y3], 'k')
7     plt.plot([x2,x3],[y2,y3], 'k')
8
9     if(kol<n):
10         podder(
11             kol+1,
12             (x1 + x2) / 2 + (x2 - x3) / 2,
13             (y1 + y2) / 2 + (y2 - y3) / 2,
14             (x1 + x2) / 2 + (x1 - x3) / 2,
15             (y1 + y2) / 2 + (y1 - y3) / 2,
16             (x1 + x2) / 2,
17             (y1 + y2) / 2,
18             n)
19
20         podder(
21             kol+1,
22             (x3 + x2) / 2 + (x2 - x1) / 2,
23             (y3 + y2) / 2 + (y2 - y1) / 2,
24             (x3 + x2) / 2 + (x3 - x1) / 2,
25             (y3 + y2) / 2 + (y3 - y1) / 2,
26             (x3 + x2) / 2,
27             (y3 + y2) / 2,
28             n)
29
30         podder(
31             kol+1,
32             (x1 + x3) / 2 + (x3 - x2) / 2,
33             (y1 + y3) / 2 + (y3 - y2) / 2,
34             (x1 + x3) / 2 + (x1 - x2) / 2,
35             (y1 + y3) / 2 + (y1 - y2) / 2,
36             (x1 + x3) / 2,
37             (y1 + y3) / 2,
38             n)
39
40
41
42 def glavder(x1,y1,x2,y2,x3,y3,n):
43     plt.plot([x1,x2],[y1,y2], 'k')
44     plt.plot([x1,x3],[y1,y3], 'k')
45     plt.plot([x2,x3],[y2,y3], 'k')
46
47     z=1
48     ax=(x1+x2)/2
49     ay=(y1+y2)/2
50     bx=(x1+x3)/2
51     by=(y1+y3)/2
52     cx=(x2+x3)/2
53     cy=(y2+y3)/2
```

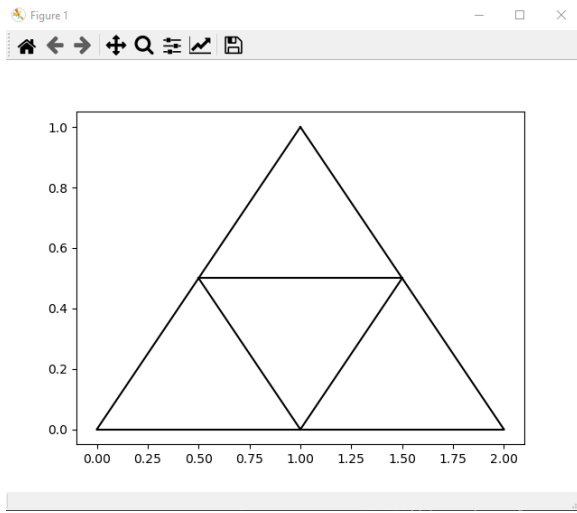


```

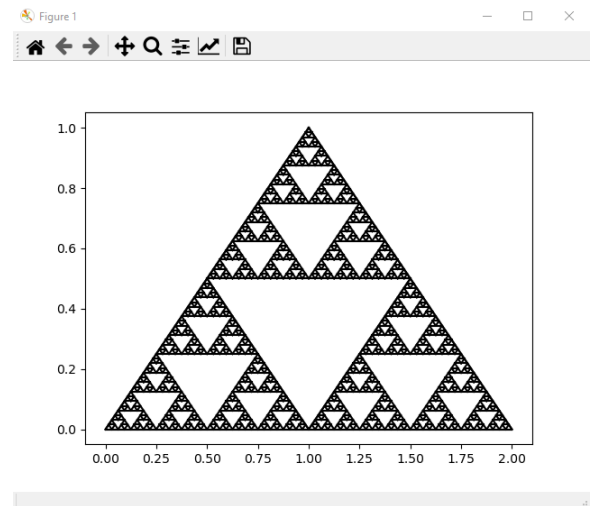
54     return podder(z, ax, ay, bx, by, cx, cy, n)
55
56 n=int(input("Glubina rekursii:"))
57
58
59 glavder(0,0,2,0,1,1,n)

```

Приложение В4



Глубина рекурсии 1



Глубина рекурсии 6

Приложение Г1

```

1  from PIL import Image
2
3  # gde risuem
4  xstart = -2.0
5  xend = 1.0
6  ystart = -1.5
7  yend = 1.5
8
9  # maximalnoe kolichestvo prohodov
10 maxkol = 255
11
12 # razmer fotografii
13 imgx = 500
14 imgy = 500
15 image = Image.new("RGB", (imgx, imgy))
16
17 for y in range(imgy):
18     zy = y * (yend - ystart) / (imgy - 1) + ystart
19     for x in range(imgx):
20         zx = x * (xend - xstart) / (imgx - 1) + xstart
21         z = zx + zy * 1j #complex
22         c = z
23         for i in range(maxkol):
24             if abs(z) > 2.0:
25                 break

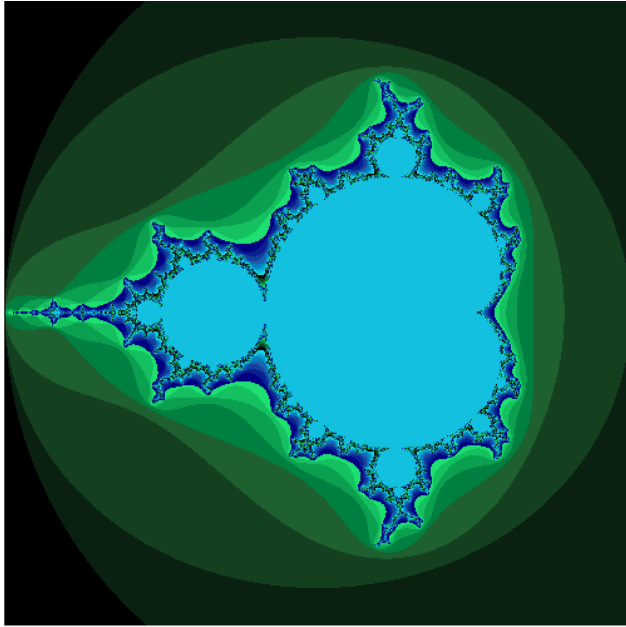
```

```

26         z = z * z + c
27         image.putpixel((x, y), (i % 4 * 10, i % 8 * 32, i % 16 * 16)) #
28 tsvet v kordinate x,y, tsvet v rgb
29
    image.show()

```

Приложение Г2



Множество Мандельброта

Приложение Д1

```

1  import numpy as np
2  import matplotlib.pyplot as plt
3  import matplotlib.cm as cm
4
5  xdlin, yvisot = 200, 200
6
7  t = complex(0.28, 0.008)
8  zabs_max = 10
9  nit_max = 2000
10
11  xmin, xmax = -2, 2
12  xwidth = xmax - xmin
13  ymin, ymax = -2, 2
14  yheight = ymax - ymin
15
16  julia = np.zeros((xdlin, yvisot))
17
18
19  for ix in range(xdlin):
20      for iy in range(yvisot):
21          kol = 0

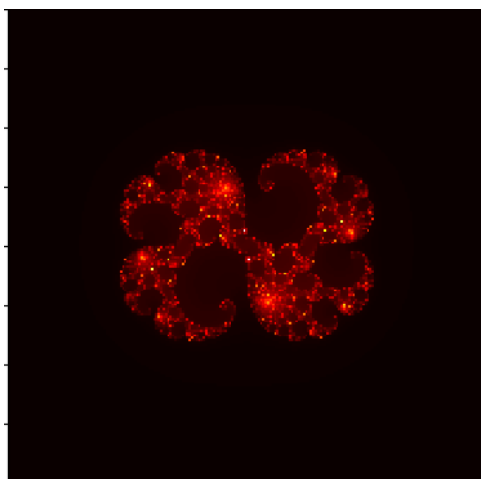
```

```

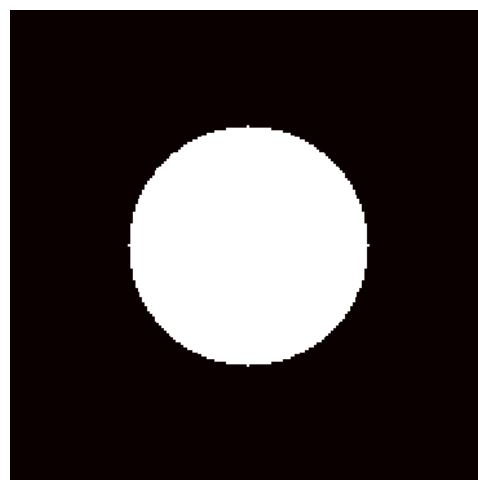
22     z = complex(ix / xdlin * xwidth + xmin,
23                 iy / yvisot * yheight + ymin)
24     while abs(z) <= zabs_max and kol < nit_max:
25         z = z**2 + t
26         kol += 1
27     ot= 1-np.sqrt(kol / nit_max)
28     otno = kol / nit_max
29     julia[ix,iy] = otno
30
31 fig, ax = plt.subplots()
32 ax.imshow(julia, interpolation='nearest', cmap=cm.hot)
33
34
35 plt.show()

```

Приложение Д2



Множество Жюлиа $t = 0.28 + 0.008i$



Множество Жюлиа $t = 0$