Федеральное агентство связи

Ордена Трудового Красного Знамени

федеральное государственное бюджетное учреждение высшего образования

«Московский технический университет связи и информатики»

Кафедра «Математическая кибернетика и информационные технологии»

Лабораторные работы по дисциплине

«Структуры и алгоритмы обработки данных»

Лабораторная работа №5

«Генерация фрактала»

Выполнила студентка группы БСТ2002

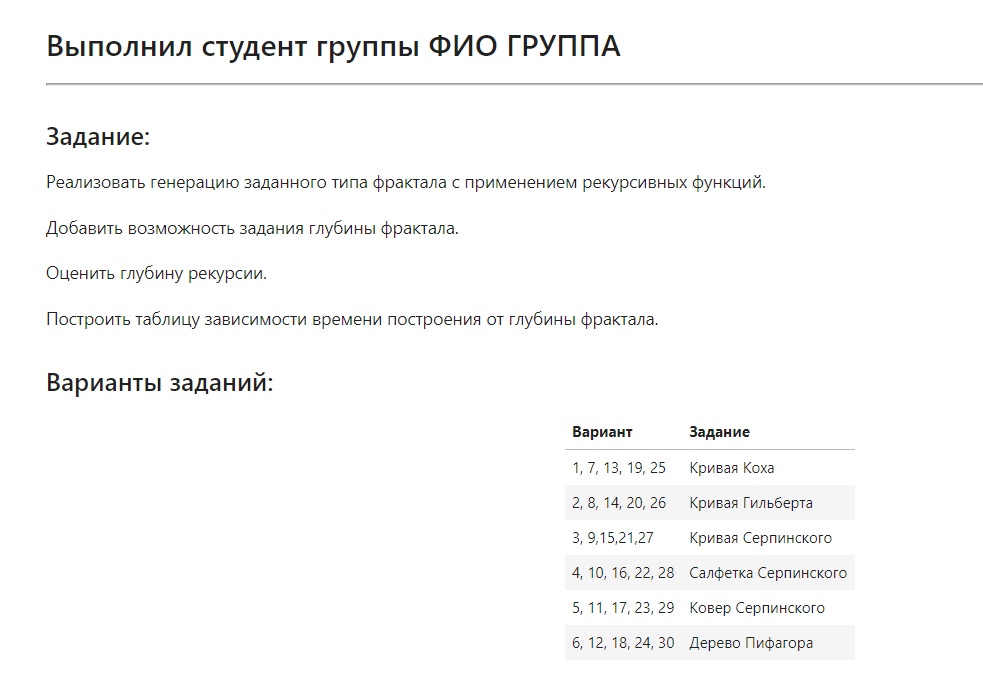
Сергеева А.А.

Вариант №16

Проверил: Аршинов Е.А.

Москва 2022

**Задание**



**Ход работы**

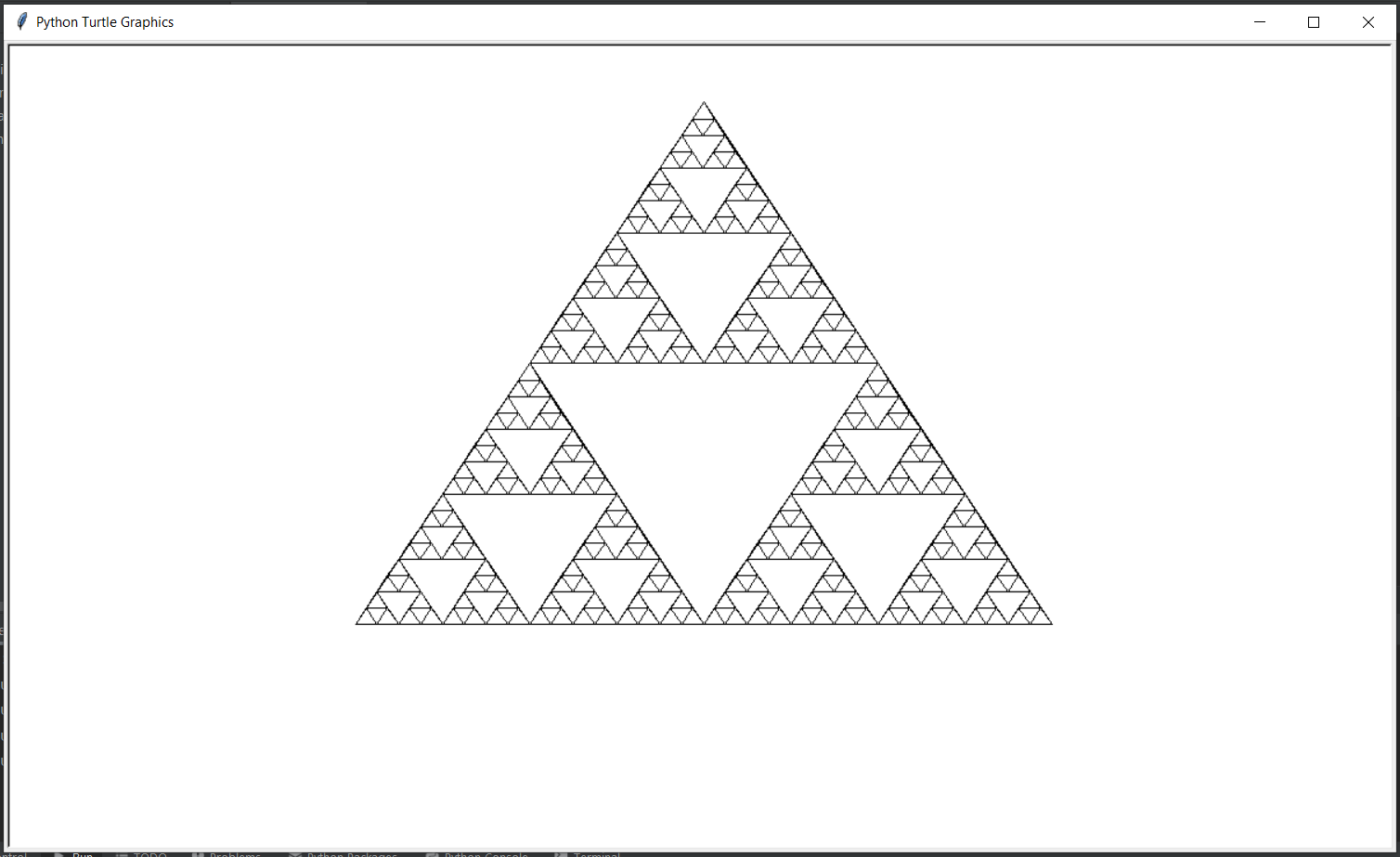
Буду генерировать фрактал под названием «СалфеткаСерпинского». Построение фрактала: Середины сторон [равностороннего треугольника](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B2%D0%BD%D0%BE%D1%81%D1%82%D0%BE%D1%80%D0%BE%D0%BD%D0%BD%D0%B8%D0%B9_%D1%82%D1%80%D0%B5%D1%83%D0%B3%D0%BE%D0%BB%D1%8C%D0%BD%D0%B8%D0%BA)  {\displaystyle T\_{0}} соединяются [отрезками](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D1%80%D0%B5%D0%B7%D0%BE%D0%BA). Получаются 4 новых треугольника. Из исходного треугольника удаляется внутренность [срединного треугольника](https://ru.wikipedia.org/wiki/%D0%A1%D1%80%D0%B5%D0%B4%D0%B8%D0%BD%D0%BD%D1%8B%D0%B9_%D1%82%D1%80%D0%B5%D1%83%D0%B3%D0%BE%D0%BB%D1%8C%D0%BD%D0%B8%D0%BA).

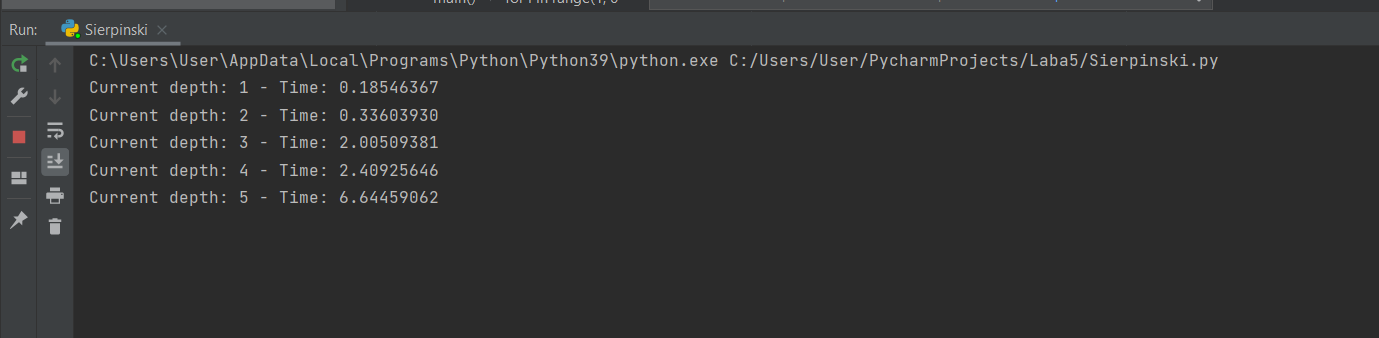
Для рисования фрактала необходимо импортировать модуль turtle. В функции draw\_triangle строим треугольник по 3- точкам. Функция get\_mid находит середину каждой стороны и возвращает ее. В функции sierpinski рисуем треугольник по точкам, и если глубина больше, то рекурсивно вызываем саму же функцию для 3-х треугольников. В функции main мы создаем нашу черепашку, для отрисовки, устанавливаем параметры окна, задаем список координат, по которым строится наш 1-й треугольник, задаем глубину и вызываем функцию sierpinski. Ниже в цикле создаем табличку зависимости времени от глубины фрактала.

**Код программы:**

import turtle  
import time  
  
  
# Draw triangle between the three points in the points list  
def draw\_triangle(points, t):  
 t.up()  
 t.goto(points[0][0], points[0][1])  
 t.down()  
 t.goto(points[1][0], points[1][1])  
 t.goto(points[2][0], points[2][1])  
 t.goto(points[0][0], points[0][1])  
  
  
def get\_mid(p1, p2):  
 return (p1[0] + p2[0]) // 2, (p1[1] + p2[1]) // 2  
  
  
def sierpinski(points, depth, t):  
 draw\_triangle(points, t)  
 if depth > 0:  
 sierpinski([points[0],  
 get\_mid(points[0], points[1]),  
 get\_mid(points[0], points[2])],  
 depth - 1, t)  
 sierpinski([points[1],  
 get\_mid(points[0], points[1]),  
 get\_mid(points[1], points[2])],  
 depth - 1, t)  
 sierpinski([points[2],  
 get\_mid(points[2], points[1]),  
 get\_mid(points[0], points[2])],  
 depth - 1, t)  
  
  
def main():  
 t = turtle.Turtle()  
 turtle.tracer(10, 0)  
 t.ht()  
  
 width = 1200  
 height = 700  
 screen = turtle.Screen()  
 screen.setup(width, height, 0, -50)  
  
 points\_list = [[-300, -150], [0, 300], [300, -150]]  
 depth = 3  
 sierpinski(points\_list, depth, t)  
  
 # Building up a table  
 for i in range(1, 4):  
 turtle.TurtleScreen.\_RUNNING = True  
 turtle\_time = turtle.Turtle()  
 turtle\_time.ht()  
 turtle.tracer(2, 0)  
  
 start = time.time()  
 sierpinski(points\_list, i, turtle\_time)  
 finish = time.time() - start  
 print("Current depth: {} - Time: {}".format(i, '%8.8f' % finish))  
  
 screen.exitonclick()  
  
  
main()

**Результаты работы программы для глубины 5:**





**Вывод**

В данной лабораторной работе научились генерировать фрактал «СалфеткаСерпинского».