**ФГАОУ ВО "МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ"**

**Лабораторная работа №8**

Программирование в графическом режиме, модуль turtle

**Вариант №11**

по дисциплине:

Основы программирования

Выполнил

студент 1 курса
группы 211-321
Журавлев Д.А.

Проверил
_____ Никишина И.Н.

**МОСКВА 2021**

**Постановка задачи**

**Задание 1**

Вычислить и вывести на экран или в файл в виде таблицы значения функции, заданной графически, на интервале от Хнач до Хконч с шагом dx. Интервал и шаг должны быть такими что бы проверить все ветви программы. Вывести на окне график полученной функции.

**Задание 2**

Для 10000 выстрелов координаты которых задаются генератором случайных чисел. Вывести на окне график полученной функции.

**Задание 3**

Вычислить и вывести на экран в виде таблицы значения функции заданной с помощью ряда Тейлора, на интервале от Хнач до Хкон с шагом dx и с точностью e. Вывести на окне график полученной функции.

$$. \ Arth = \sum_{n=0}^{\infty} \frac{1}{(2 \cdot n + 1) \cdot x^{2 \cdot n + 1}} = \frac{1}{x} + \frac{1}{3 \cdot x^3} + \frac{1}{5 \cdot x^5} + \ldots \ , \qquad |x| > 1 .$$

**Теоретическая часть**

Для математических вычислений в Python имеются как встроенные, так и дополнительные функции и методы. Для применения дополнительных математических функций необходимо использовать модуль `math`, который подключается с помощью инструкции:

```
import math
```

Для ввода данных используется инструкция `input()`, которая возвращает строку. Введенные значения должны быть преобразованы к числовому формату перед использованием в арифметических выражениях.

Для предотвращения появления ошибок при преобразовании из-за неправильного ввода, а так же предотвращения ошибок из-за деления на ноль используется инструкция `try-except`.

Для вычисления всех значений функции в заданном интервале используется чикл `while`. Циклы позволяют выполнять блок кода до тех пор пока условие цикла не станет ложным.

Вывод данных выполняется инструкцией `print()`, с возможностью форматирования данных.

1. Класс TurtleScreen определяет графические окна как площадку для рисования черепах. Его конструктору требуется аргумент tkinter.Canvas или ScrolledCanvas. Его следует использовать, когда turtle применяется как часть какого-либо приложения.

Функция Screen() возвращает объект одиночку подкласса TurtleScreen. Функцию следует использовать, когда turtle используется как автономный инструмент для создания графики. Т. к. это объект одиночка, наследование от его класса невозможно.

Все методы TurtleScreen/Screen также существуют как функции, т. е. как часть интерфейса, ориентированного на процедуры.

2. RawTurtle (алиас: RawPen) определяет объекты черепахи, которые рисуются на TurtleScreen. Его конструктору требуется в качестве аргумента Canvas, ScrolledCanvas или TurtleScreen, поэтому объекты RawTurtle знают, где рисовать.

Производным от RawTurtle является подкласс Turtle (псевдоним: Pen), который основан на «экземпляре» Screen, создаваемый автоматически, если его ещё не существует.

Все методы RawTurtle/Turtle также существуют как функции, т. е. часть интерфейса, ориентированного на процедуры.

Процедурный интерфейс предоставляет функции, производные от методов классов Screen и Turtle. У них те же имена, что и соответствующие методы. Объект экрана создаётся автоматически при каждом вызове функции, производной от метода Screen. Объект черепахи (без имени) автоматически создаётся при каждом вызове любой функции, производной от метода черепахи.

Для использования нескольких черепах на экране необходимо использовать объектно-ориентированный интерфейс.

**Задание 1**

**Описание программы**

Программа написана на алгоритмическом языке Python 3.9.1, реализована в среде OS Windows 10 и состоит из частей, отвечающих за ввод данных, их преобразования к численному формату, вычисления и представления итоговых данных на экране монитора.

**Описание алгоритма**

1. Обернем все тело программы в `try - except` для контроля ошибок приведения и деления на ноль.
2. Ввести значения аргументов `xs xe st`, привести их к типу `float`.
3. Выводим в окне оси сиситемы координат.
4. В цикле вычисляем(функцией func() из прошлой работы) и выводим получившееся значение функции.
5. Увеличиваем счетчик цикла `xs` на шаг `st`.

**Описание входных и выходных данных**

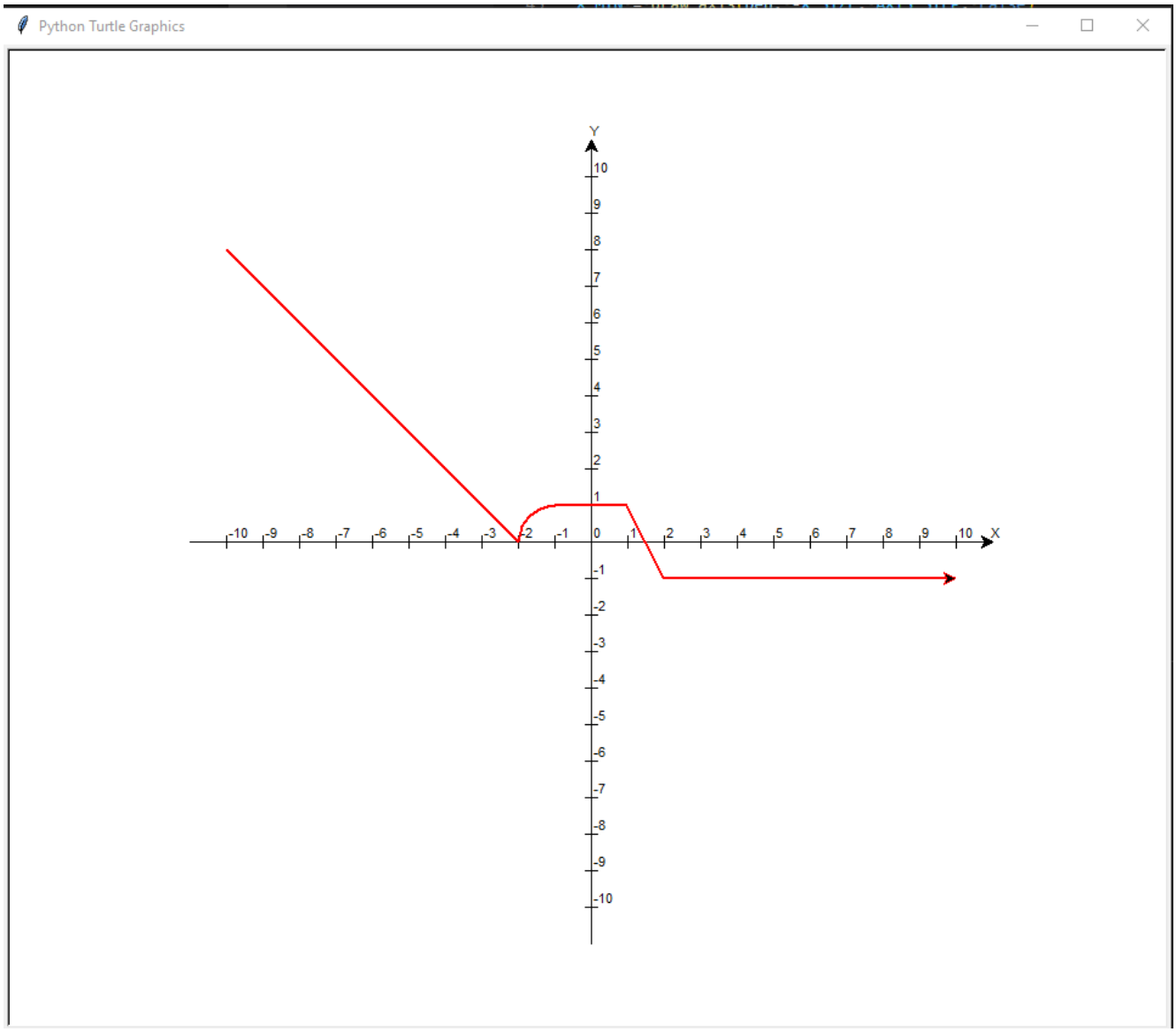
Входные данные поступают с клавиатуры, а выходные – выводятся на монитор для просмотра. Входные имеют тип `float`. Выходные данные являются таблицей значений функции выведенной в терминал.

**Листинг программы**

```
import turtle
from math import *
from random import *

def draw_axis(pen: turtle.Turtle, size, step, arrow) -> None:
    pen.speed(0)
    m = 0
    for i in range(0, size, step * int((size / abs(size)))):
```

```
            m += 1
            pen.fd(step)
            pen.lt(90)
            pen.write(" " + str(m * int((size / abs(size)))))
            pen.fd(5)
            pen.bk(10)
            pen.fd(5)
            pen.rt(90)
        pen.fd(step)
        if (arrow):
            pen.stamp()
            pen.write(arrow)
        pen.home()
        return m * int((size / abs(size)))

X_SIZE = 300
Y_SIZE = 300
AXIS_STEP = 100

pen = turtle.Turtle()
window = turtle.Screen()

try:
    xs = float(window.textinput("Value input", "Input x start"))
    xe = float(window.textinput("Value input", "Input x end"))
    st = float(window.textinput("Value input", "Input dx"))
    es = float(window.textinput("Value input", "Input eps"))
except:
    exit()

X_MAX = draw_axis(pen, X_SIZE, AXIS_STEP, "X")
pen.lt(180)
X_MIN = draw_axis(pen, -X_SIZE, AXIS_STEP, False)
pen.lt(90)
Y_MAX = draw_axis(pen, Y_SIZE, AXIS_STEP, "Y")
pen.lt(270)
Y_MIN = draw_axis(pen, -Y_SIZE, AXIS_STEP, False)
pen.write(" 0")

pen.up()
pen.hideturtle()

while xs <= xe:
    if xs > -1 and xs < 1:
        xs += st
        pen.up()
        continue
    y = 0
    n = 1
    m = 0
    while True:
        de = 1 / (n * xs ** n)
        y += de
        n += 2
        m += 1
        if abs(de) < es:
            break
    pen.goto(xs * AXIS_STEP, y * AXIS_STEP)
    pen.down()
    pen.dot(4, "#227800")
    xs += st

window.mainloop()
```

**Результат работы программы**




**Задание 2**

**Описание программы**

Программа написана на алгоритмическом языке Python 3.9.1, реализована в среде OS Windows 10 и состоит из частей, отвечающих за ввод данных, их преобразования к численному формату, вычисления и представления итоговых данных на экране монитора.

**Описание алгоритма**

1. Выводим в окне оси сиситемы координат.
2. В цикле вычисляем(функцией `shot()` из прошлой работы) и выводим координаты и результат попадания.

**Описание входных и выходных данных**

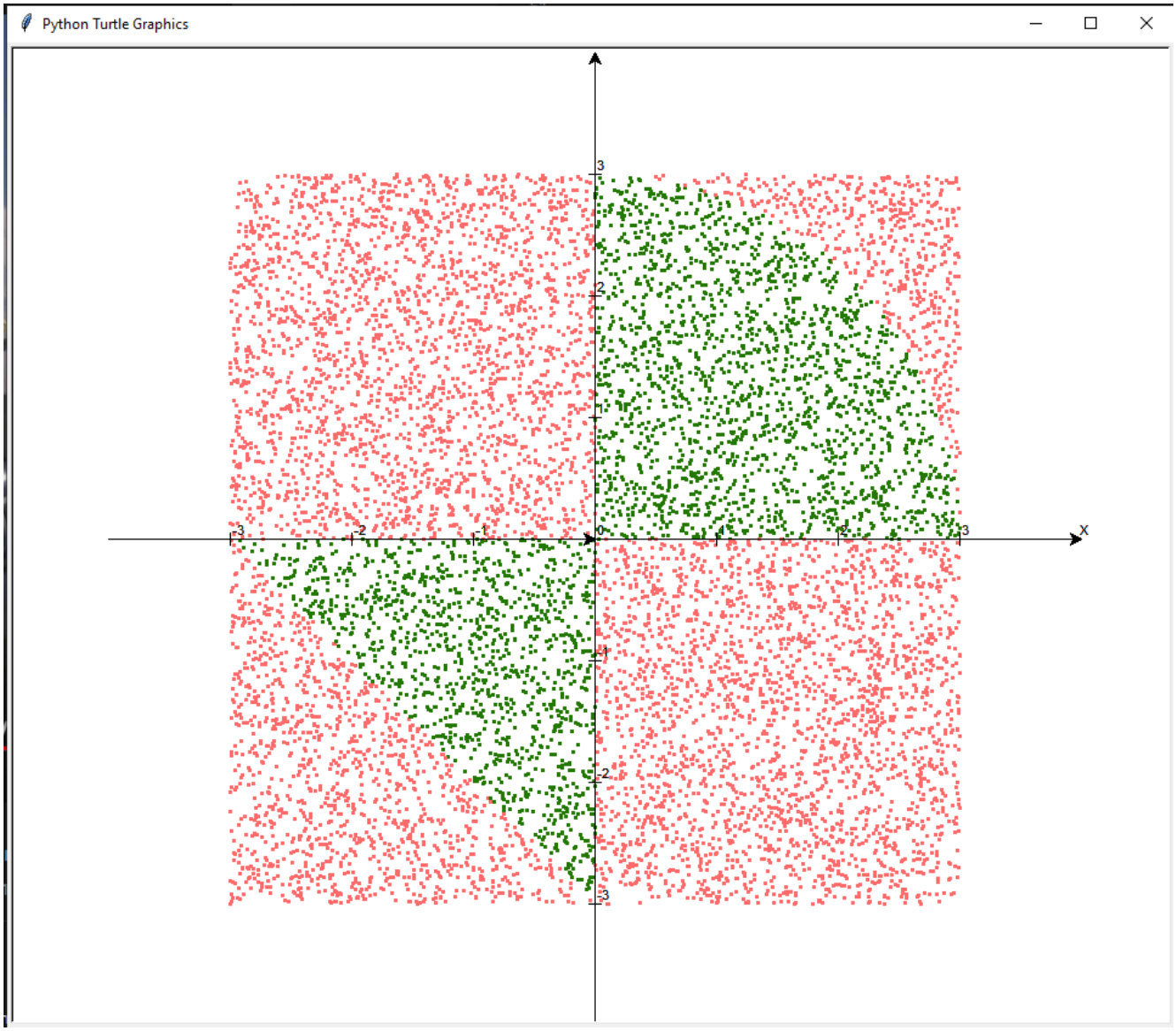
Входные данные поступают с клавиатуры, а выходные – выводятся на монитор для просмотра. Входные имеют тип `float`. Выходные данные являются изображением.

**Листинг программы**

```
import turtle
from math import *
from random import *

def shot(r, x, y):
    if x >= 0 and y >= 0 and sqrt(x*x + y*y) <= r:
        return [x, y, 1]
    elif x <= 0 and y <= 0 and -x - r <= y:
        return [x, y, 1]
    else:
        return [x, y, 0]

X_SIZE = 300
Y_SIZE = 300
AXIS_STEP = 100

pen = turtle.Turtle()
window = turtle.Screen()

def draw_axis(pen: turtle.Turtle, size, step, arrow) -> None:
    pen.speed(0)
    m = 0
    for i in range(0, size, step * int((size / abs(size)))):
        m += 1
        pen.fd(step)
        pen.lt(90)
        pen.write(" " + str(m * int((size / abs(size)))))
        pen.fd(5)
        pen.bk(10)
        pen.fd(5)
        pen.rt(90)
    pen.fd(step)
    if (arrow):
        pen.stamp()
        pen.write(arrow)
    pen.home()
    return m * int((size / abs(size)))

X_MAX = draw_axis(pen, X_SIZE, AXIS_STEP, "X")
pen.lt(180)
X_MIN = draw_axis(pen, -X_SIZE, AXIS_STEP, False)
pen.lt(90)
Y_MAX = draw_axis(pen, Y_SIZE, AXIS_STEP, "Y")
pen.lt(270)
Y_MIN = draw_axis(pen, -Y_SIZE, AXIS_STEP, False)
pen.write(" 0")

pen.penup()
r = min(Y_MAX, X_MAX)
print(r)
for i in range(10000):
    print(i)
    x, y, res = shot(r, uniform(-r, r), uniform(-r, r))
    pen.goto(x * AXIS_STEP, y * AXIS_STEP)
    if res == 1:
        pen.dot(4, "#227800")
    else:
        pen.dot(4, "#ff6b6b")

pen.goto(0, 0)
pen.down()
X_MAX = draw_axis(pen, X_SIZE, AXIS_STEP, "X")
```

```
pen.lt(180)
X_MIN = draw_axis(pen, -X_SIZE, AXIS_STEP, False)
pen.lt(90)
Y_MAX = draw_axis(pen, Y_SIZE, AXIS_STEP, "Y")
pen.lt(270)
Y_MIN = draw_axis(pen, -Y_SIZE, AXIS_STEP, False)
pen.write(" 0")

window.mainloop()
```

**Результат работы программы**



**Задание 3**

**Описание программы**

Программа написана на алгоритмическом языке Python 3.9.1, реализована в среде OS Windows 10 и состоит из частей, отвечающих за ввод данных, их преобразования к численному формату, вычисления и представления итоговых данных на экране монитора.

**Описание алгоритма**

1. Ввести значения аргументов `xs xe st es`, привести их к типу `float`.
2. Выводим в окне оси сиситемы координат.
3. В цикле запускаем цикл для вычисления `Y` при заданном `X`.
4. Во воженном цикле считаем следующий элемент последовательности и складываем с остальными, увеличиваем все счетчики.
5. Выводим получившиеся значения.
6. Увеличиваем счетчик цикла `xs` на шаг `st`.

**Описание входных и выходных данных**

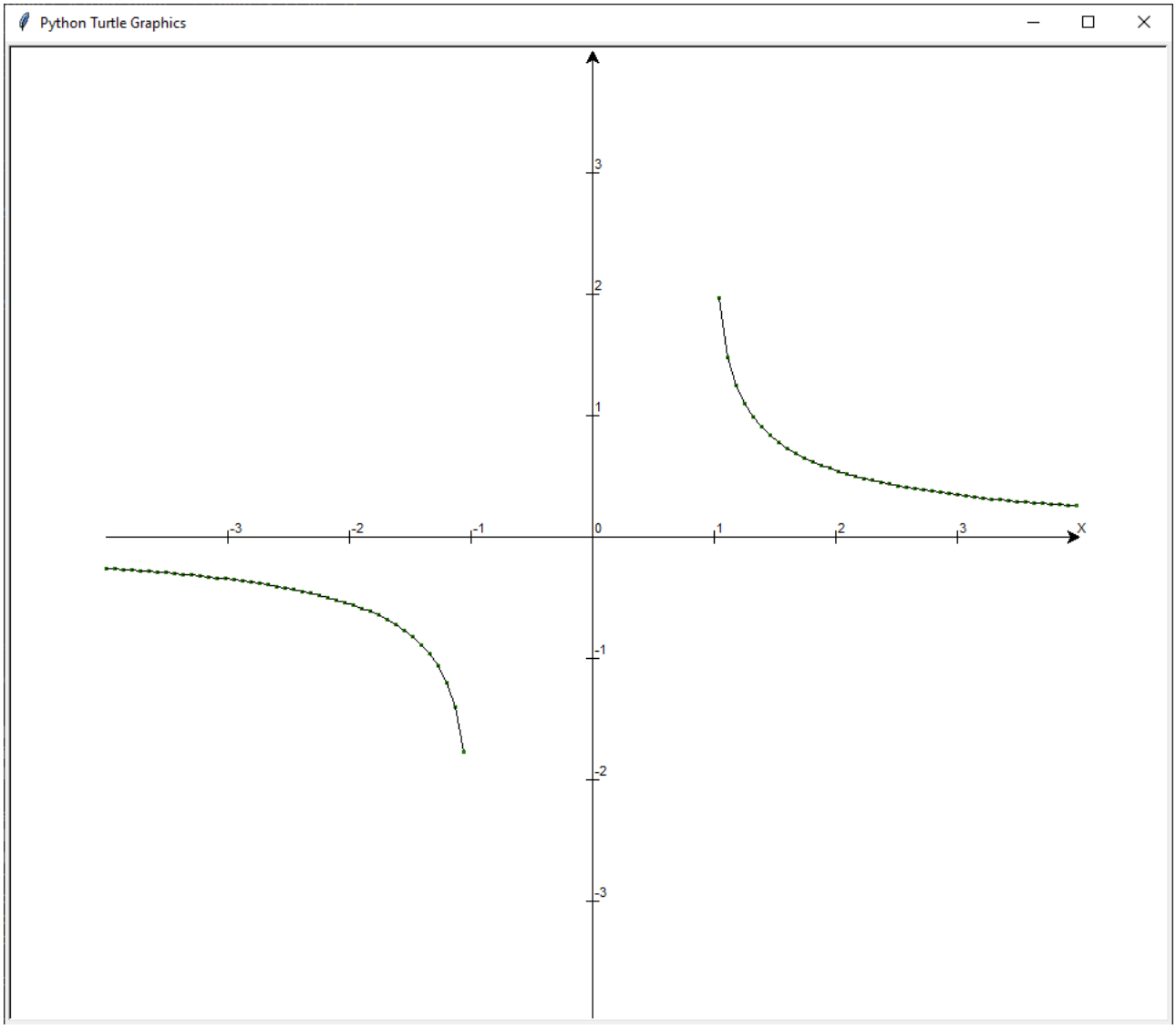
Входные данные поступают с клавиатуры, а выходные – выводятся на монитор для просмотра. Входные имеют тип `float`. Выходные данные являются изображением.

**Листинг программы**

```
import turtle
from math import *
from random import *

def draw_axis(pen: turtle.Turtle, size, step, arrow) -> None:
    pen.speed(0)
    m = 0
    for i in range(0, size, step * int((size / abs(size)))):
        m += 1
        pen.fd(step)
        pen.lt(90)
        pen.write(" " + str(m * int((size / abs(size)))))
        pen.fd(5)
        pen.bk(10)
        pen.fd(5)
        pen.rt(90)
    pen.fd(step)
    if (arrow):
        pen.stamp()
        pen.write(arrow)
    pen.home()
    return m * int((size / abs(size)))

X_SIZE = 300
Y_SIZE = 300
AXIS_STEP = 100

pen = turtle.Turtle()
window = turtle.Screen()

try:
    xs = float(window.textinput("Value input", "Input x start"))
```

```
    xe = float(window.textinput("Value input", "Input x end"))
    st = float(window.textinput("Value input", "Input dx"))
    es = float(window.textinput("Value input", "Input eps"))
except:
    exit()

X_MAX = draw_axis(pen, X_SIZE, AXIS_STEP, "X")
pen.lt(180)
X_MIN = draw_axis(pen, -X_SIZE, AXIS_STEP, False)
pen.lt(90)
Y_MAX = draw_axis(pen, Y_SIZE, AXIS_STEP, "Y")
pen.lt(270)
Y_MIN = draw_axis(pen, -Y_SIZE, AXIS_STEP, False)
pen.write(" 0")

pen.up()
pen.hideturtle()

while xs <= xe:
    if xs > -1 and xs < 1:
        xs += st
        pen.up()
        continue
    y = 0
    n = 1
    m = 0
    while True:
        de = 1 / (n * xs ** n)
        y += de
        n += 2
        m += 1
        if abs(de) < es:
            break
    pen.goto(xs * AXIS_STEP, y * AXIS_STEP)
    pen.down()
    pen.dot(4, "#227800")
    xs += st

window.mainloop()
```

**Результат работы программы**




**Список используемой литературы**

1. В.П. Рядченко, Методическое пособие по выполнению лабораторных работ
2. https://pythonworld.ru/