# OOPython

## Задача 5. Численное решение ОДУ

### Введение

Будем рассматривать следующие методы численного решения автономного ОДУ:

1. явный метод Эйлера 1-го порядка точности (реализован в **lecture\_9\_scalar\_ode.ipynb**)
2. метод Эйлера с пересчетом 2-го порядка точности (реализован в **lecture\_9\_scalar\_ode.ipynb**)
3. *любой* явный метод Рунге-Кутты 3-го порядка точности
4. явный метод Рунге-Кутты 4-го порядка точности (реализован в **lecture\_9\_scalar\_ode.ipynb**)
5. неявный метод трапеций 2-го порядка точности

### Задание

#### Определение классов

Для каждого из методов (1)-(5) реализовать соответствующий класс *MethodName*, минимизировав суммарное число строк кода с помощью наследования. Каждый класс должен включать в себя как минимум следующее:

**Поля**:

* функция правой части ОДУ
* начальное условие
* множество точек сетки и ее параметры: кол-во точек, отрезков, шаг сетки
* массив для хранения значений численного решения
* начальный и конечный моменты времени

**Методы:**

* конструктор
* сеттеры для функции правой части, начального условия, параметров сетки
* решить ОДУ (timestepping - цикл по точкам сетки)
* построить график численного решения

#### Использование классов

Каждым из методов провести численное решение логистического уравнения и построить графики соответствующих решений.

Параметры функции правой части логистического уравнения:

* 
* 

Параметры расчетной сетки:

* число отрезков разбиения 
* 
* 

Начальное условие:

* 

При реализации неявного метода трапеций критерий остановки итераций метода Ньютона для решения нелинейного уравнения: .