# OOPython

## Задача 3. Численное интегрирование

### Введение

В лекционном материале **lecture\_6\_numerical\_integration.ipynb** рассматривалось представление первообразной  функции  в виде класса **AntiderivativeNum**, экземпляры которого являются функторами. В методе **\_\_call\_\_** данного класса реализованы вычисления по формуле трапеций для приближенного вычисления первообразной функции через интеграл:



Существует множество формул различных порядков точности для расчета . Ограничимся следующими:

* формула правых прямоугольников: 
* формула левых прямоугольников: 
* формула средних прямоугольников: 
* формула трапеций: 
* формула Симпсона:

Любую из вышеприведенных формул можно записать в общем виде:



где и  - множества коэффициентов и точек расчетной сетки, соответственно.

### Задание

#### Определение классов

Для каждой из формул -(6) реализовать соответствующий класс **Antiderivative***Name*, минимизировав суммарное число строк кода с помощью наследования. Каждый класс должен включать в себя как минимум следующее:

Поля:

* интегрируемая функция 
* левая граница отрезка 
* число отрезков разбиения 

Методы:

* конструктор
* задать коэффициенты  и точки расчетной сетки 
* вычислить значение  по формуле (реализовать через магический метод **\_\_call\_\_**)

#### Использование классов

Использовать созданные классы для приближенного вычисления интегралов на отрезке . Для каждой формулы - построить графики абсолютного значения погрешности в зависимости от числа отрезков интегрирования  для следующих подынтегральных функций:

Использовать логарифмический масштаб по обеим осям. Графики погрешностей для каждой функции строить в отдельном окне. Задействовать библиотеку **SymPy** для аналитического вычисления интегралов. (аналогично **Задаче 2**).

*Подсказка:* для ускорения расчета по формуле (7) вместо цикла можно использовать функцию **numpy.dot(a, b)**, вычисляющую скалярное произведение векторов **a** и **b**.