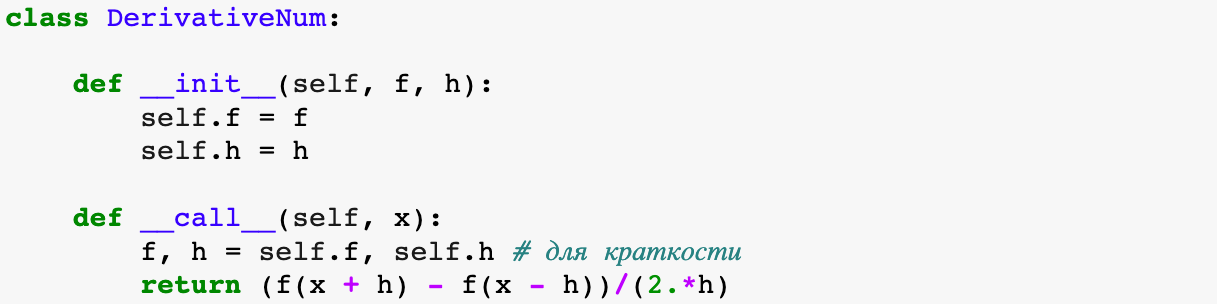
# OOPython

## Задача 2. Численное дифференцирование

### Введение

В лекционном материале **lecture\_5\_classes\_b.ipynb** рассматривалось представление производной  в виде класса, экземпляры которого являются функторами:



В методе **\_\_call\_\_** для приближенного вычисления производной  используется формула центральной разности:



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

* 
* 
* 
* 
* 

Любую из вышеприведенных формул можно записать в общем виде:



где - набор коэффициентов,  - набор точек шаблона, .

### Задание

#### Определение классов

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

Поля:

* дифференцируемая функция 
* шаг численного дифференцирования 
* набор коэффициентов 

Методы:

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

#### Использование классов

Использовать созданные классы для расчета численных производных: для каждой формулы - в точке  построить графики абсолютного значения погрешности в зависимости от шага численного дифференцирования  для следующих функций:



Использовать логарифмический масштаб по обеим осям. Графики погрешностей для каждой функции строить в отдельном окне. Задействовать библиотеку **SymPy** для аналитического вычисления производных. Примеры реализации – см. в **lecture\_5\_classes\_b.ipynb**.