```
import math

class NumericalSearchIntegrals:
    def __init__(self, function, function_in_str, left_border, right_border):
        self.function_in_str = function_in_str
        self.f = function
        self.a = left_border
        self.b = right_border

    def left_rectangle_method(self, h):
        return f'∫{self.function_in_str}dx = {self.__left_rect__(h)}'

    def middle_rectangle_method(self, h):
        return f'∫{self.function_in_str}dx = {self.__middle_rect(h)}'

    def simpson_method(self, h):
        return f'∫{self.function_in_str}dx = {self.__simpson__(h)}'

    def __simpson__(self, h):
        m = 1 + (self.b - self.a) // h
        summ = 0
        for j in range(1, int(m)):
            xj = self.a + j * h
            summ += self.f(xj) + 4 * self.f(xj + h / 2) + self.f(self.a + (j + 1) * h)
        return h / 6 * summ

    def __left_rect__(self, h):
        m = 1 + (self.b - self.a) // h
        summ = 0
        for j in range(1, int(m)):
            xj = self.a + j * h
            summ += self.f(xj)
        return h * summ

    def __middle_rect(self, h):
        m = 1 + (self.b - self.a) // h
        summ = 0
        for j in range(1, int(m)):
            xj = self.a + j * h
            summ += self.f(xj + h / 2)
        return h * summ

    def runge_error_estimate(self):
        print(f'Функция f(x) = {self.function_in_str} на отрезке [{self.a}; {self.b}]: ')
        print()
        hs = [0.1, 0.05, 0.025]
        for h in hs:
            print(f'Шаг h = {h}')
            print()
            print(f'Метод левых прямоугольников:')
            print(self.left_rectangle_method(h))
            print(self.runge(self.__left_rect__, 1, h))
            print()
            print(f'Метод средних прямоугольников')
            print(self.middle_rectangle_method(h))
            print(self.runge(self.__middle_rect, 2, h))
            print()
            print(f'Метод Симпсона')
            print(self.simpson_method(h))
            print(self.runge(self.__simpson__, 4, h))
            print()

    def runge(self, method, p, h):
        return f'Погрешность по Рунге = {2 ** p / (2 ** p - 1) * (method(h / 2) - method(h))}'


my_integral = NumericalSearchIntegrals(lambda x: math.sin(x**3), "e^(x^2)", 0, 1)
my_integral.runge_error_estimate()
```