# Лабораторная работа № 1 «Численное интегрирование»

## Инструменты

Использовал интерпретатор, встроенный в Python.

## Задача

Составить программу, которая реализует методы численного интегрирования

с постоянным и переменным шагом для интеграла , и которая считает кратный интеграл .

## Код программы

import math  
  
  
def main\_menu(): # Главное меню  
 print("Главное меню\n1. Методы с постоянным шагом\n2. Методы "  
 "с переменным шагом\n3. Вычисление кратного интеграла\n4. Выйти из программы")  
 user\_input = int(input())  
 if user\_input == 1:  
 constant\_step\_integration\_menu()  
 elif user\_input == 2:  
 variable\_step\_integration\_menu()  
 elif user\_input == 3:  
 multiple\_integral()  
 elif user\_input == 4:  
 pass  
 else:  
 print("Введите число от 1 до 3.")  
 main\_menu()  
  
  
def constant\_step\_integration\_menu(): # Меню "Методы с постоянным шагом"  
 print("Методы с постоянным шагом\n1. Метод правых частей прямоугольника"  
 "\n2. Метод левых частей прямоугольника\n3. Метод трапеций\n4. Ме"  
 "тод парабол\n5. Возврат в главное меню")  
 user\_input = int(input())  
 if user\_input == 1:  
 right\_rectangle\_method(f, a, b, n)  
 elif user\_input == 2:  
 left\_rectangle\_method(f, a, b, n)  
 elif user\_input == 3:  
 trapezoid\_method(f, a, b, n)  
 elif user\_input == 4:  
 parabola\_method(f, a, b, n)  
 elif user\_input == 5:  
 main\_menu()  
 else:  
 print("Введите число от 1 до 5.")  
 constant\_step\_integration\_menu()  
  
  
def variable\_step\_integration\_menu(): # Меню "Методы с переменным шагом"  
 print("Метод с переменным шагом\n1. Метод по 1 алгоритму\n2. Метод по 2 "  
 "алогритму\n3. Возврат в главное меню")  
 user\_input = int(input())  
 if user\_input == 1:  
 double\_counting\_method\_first(f, a, b, n, E)  
 elif user\_input == 2:  
 double\_counting\_method\_second(f, a, b, n, E)  
 elif user\_input == 3:  
 main\_menu()  
 else:  
 print("Введите число от 1 до 3.")  
 variable\_step\_integration\_menu()  
  
  
def right\_rectangle\_method(f, a, b, n): # Метод правых частей прямоугольника  
 r = math.fabs(((b - a) \*\* 2 / (2 \* n)) \* 3.389056)  
 print("Остаточный член R:", r)  
 h = (b - a) / n  
 result = 0  
 while a <= b:  
 a += h  
 result += f(a)  
 result \*= h  
 print("Метод правых частей прямоугольника: " + str(result) + "\n")  
 constant\_step\_integration\_menu()  
  
  
def left\_rectangle\_method(f, a, b, n): # Метод левых частей прямоугольника  
 r = math.fabs(((b - a) \*\* 2 / (2 \* n)) \* 3.389056)  
 print("Остаточный член R:", r)  
 h = (b - a) / n  
 result = 0  
 while a < b:  
 result += f(a)  
 a += h  
 result \*= h  
 print("Метод левых частей прямоугольника: " + str(result) + "\n")  
 constant\_step\_integration\_menu()  
  
  
def trapezoid\_method(f, a, b, n): # Метод трапеций  
 r = math.fabs(((b - a) \*\* 3 / (12 \* n \*\* 2)) \* 7.389056)  
 print("Остаточный член R:", r)  
 h = (b - a) / n  
 result = 0  
 a\_initial = a  
 while a < b - h:  
 a += h  
 result += f(a)  
 result = h \* ((f(a\_initial) + f(b)) / 2 + result)  
 print("Метод трапеций: " + str(result) + "\n")  
 constant\_step\_integration\_menu()  
  
  
def parabola\_method(f, a, b, n): # Метод парабол  
 r = math.fabs(((b - a) \*\* 5 / (720 \* n \*\* 2)) \* 7.389056)  
 print("Остаточный член R:", r)  
 h = (b - a) / n  
 result = 0  
 result\_odd = 0  
 result\_even = 0  
 a\_initial = a  
 a += 2 \* h  
 while a <= b - h: # Нечётные  
 result\_odd += f(a)  
 a += 2 \* h  
 a = a\_initial + h  
 while a <= b - h: # Чётные  
 result\_even += f(a)  
 a += 2 \* h  
 result = h / 3 \* (f(a\_initial) + f(b) + 4 \* result\_odd + 2 \* result\_even)  
 print("Метод парабол: " + str(result) + "\n")  
 constant\_step\_integration\_menu()  
  
  
def double\_counting\_method\_first(f, a, b, n, E): # Метод двойного пересчёта (первый)  
 r = math.fabs(((b - a) \*\* 2 / (2 \* n)) \* 3.389056)  
 print("Остаточный член R:", r)  
 h = E \*\* (1 / 2)  
 inn = 0  
 inn2 = 0  
 while math.fabs(inn - inn2) <= E:  
 inn = 0  
 inn2 = 0  
 i = a + h  
 while i <= b:  
 inn += (f(i) + f(i - h)) / 2  
 i += h  
 inn \*= h  
 h /= 2  
 i = a + h  
 while i <= b:  
 inn2 += (f(i) + f(i - h)) / 2  
 i += h  
 inn2 \*= h  
 print("Метод двойного пересчёта (первый): " + str(inn) + "\n")  
 variable\_step\_integration\_menu()  
  
  
def double\_counting\_method\_second(f, a, b, n, E): # Метод двойного пересчёта (второй)  
 r = math.fabs(((b - a) \*\* 2 / (2 \* n)) \* 3.389056)  
 print("Остаточный член R:", r)  
 hv = E \*\* (1 / 2)  
 hs = 0  
 inn = 0  
 inn2 = 0  
 while math.fabs(inn2 - inn) <= E:  
 amount = 0  
 amount2 = 0  
 hd = hv  
 i = a + hs  
 while i < b:  
 amount += (f(i) + f(i + hv)) / 2  
 i += hd  
 inn = amount \* hv  
 hs = hv / 2  
 i = a + hs  
 while i < b:  
 amount2 += (f(i) + f(i + hv)) / 2  
 i += hd  
 inn2 = amount2 \* hv  
 hs = hv / 2  
 hv /= 2  
 print("Метод двойного пересчёта (второй): " + str(inn) + ", " + str(inn2) + "\n")  
 variable\_step\_integration\_menu()  
  
  
def multiple\_integral(): # Вычисление кратного интеграла  
 a = 0  
 b = 3.141592/2  
 c = 0  
 d = 3.141592/4  
 nx = 10000  
 ny = 10000  
 hx = (b - a) / nx  
 hy = (d - c) / ny  
 print("hx = " + str(hx) + ", hy = " + str(hy))  
 sx = 0  
 x = a  
 while x <= b - hx:  
 sy = 0  
 x += hx  
 y = c  
 while y <= d - hy:  
 sy += math.fabs(math.sin(x + y))  
 y += hy  
 iy = hy \* sy  
 sx += iy  
 ix = hx \* sx  
 print("Результат: " + str(ix) + "\n")  
 variable\_step\_integration\_menu()  
  
  
f = lambda x: math.e \*\* x - x \*\* 2 # Подынтегральная функция  
a = 1 # Нижний предел интегрирования  
b = 2 # Верхний предел интегрирования  
n = 100 # Количество разбиений  
E = 0.00001 # Заданная точность  
  
main\_menu()

## Результаты для табличного интеграла













## Результаты для индивидуального интеграла (19 вариант)













## Результат вычисления кратного интеграла



## Анализ результатов

Самый точный метод для моего табличного интеграла — метод трапеции, так как график интеграла очень близок по форме к трапеции. Точность метода можно увеличить за счёт увеличения количества разбиений (например, до 500), но после определённого значения (например, 1000) точность начнёт ухудшаться.