Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение  
высшего образования   
«Алтайский государственный технический университет им. И. И. Ползунова»

Факультет информационных технологий

Кафедра прикладной математики

Отчет защищен с оценкой \_\_\_\_\_

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

Отчет

по лабораторной работе № 7

"Вычисление интегралов методом Монте-Карло"

по дисциплине

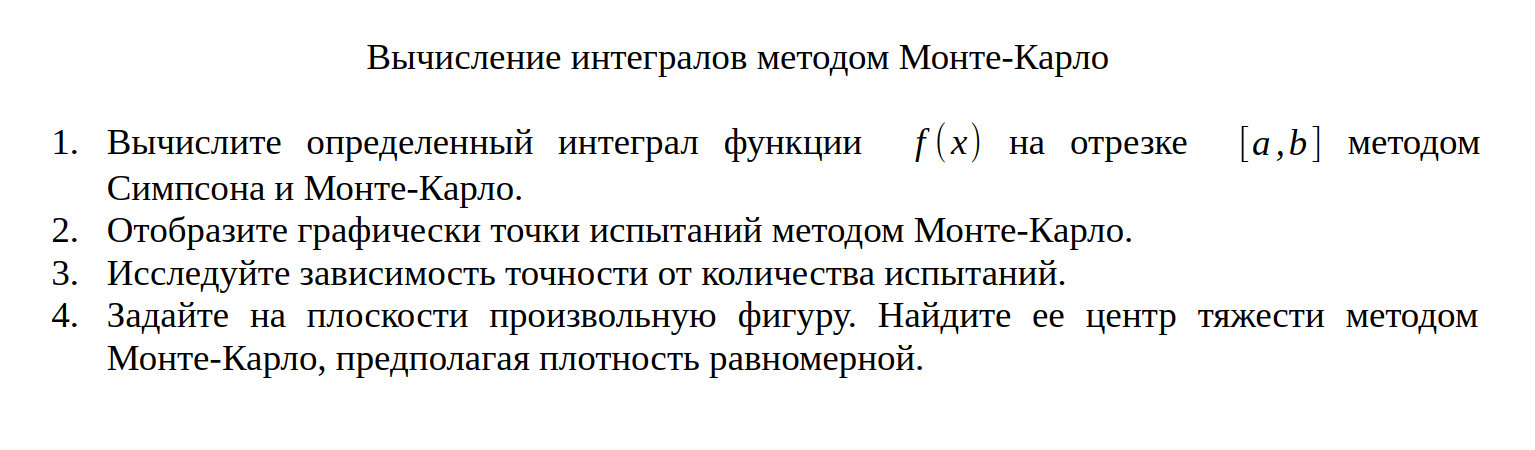
«Вычислительные алгоритмы»

Студенты гр. ПИ-92 Шинтяпин И.И.

Шульпов В.М.

Преподаватель, к.т.н. Проскурин А.В.

Барнаул 2022

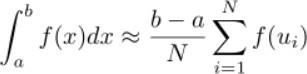


Алгоритм

C:\Users\Victor\AppData\Local\Temp\lu169961e0aop.tmp\lu169961e0apo_tmp_3774237e3f5feb45.pngМетод Симпсона Формула Симпсона:

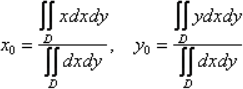
C:\Users\Victor\AppData\Local\Temp\lu169961e0aop.tmp\lu169961e0apo_tmp_5ea0769f83ccee78.jpgC:\Users\Victor\AppData\Local\Temp\lu169961e0aop.tmp\lu169961e0apo_tmp_b1100564b26cd678.pngC:\Users\Victor\AppData\Local\Temp\lu169961e0aop.tmp\lu169961e0apo_tmp_c070268712278d5b.png

Метод Монте-Карло



N – количество испытаний, u – очередное случайное число принадлежащее отрезку [a,b] Поиск центра тяжести прямоугольного треугольника

Координаты центра тяжести находятся по таким формулам:



Кратные интегралы находятся по формуле:

C:\Users\Victor\AppData\Local\Temp\lu169961e0aop.tmp\lu169961e0apo_tmp_27f384735cdce93f.jpg

C:\Users\Victor\AppData\Local\Temp\lu169961e0aop.tmp\lu169961e0apo_tmp_60f59e2305e7ace7.jpg

n – количество испытаний, ni – случайная точка принадлежащая параллелепипеду содержащему треугольник.

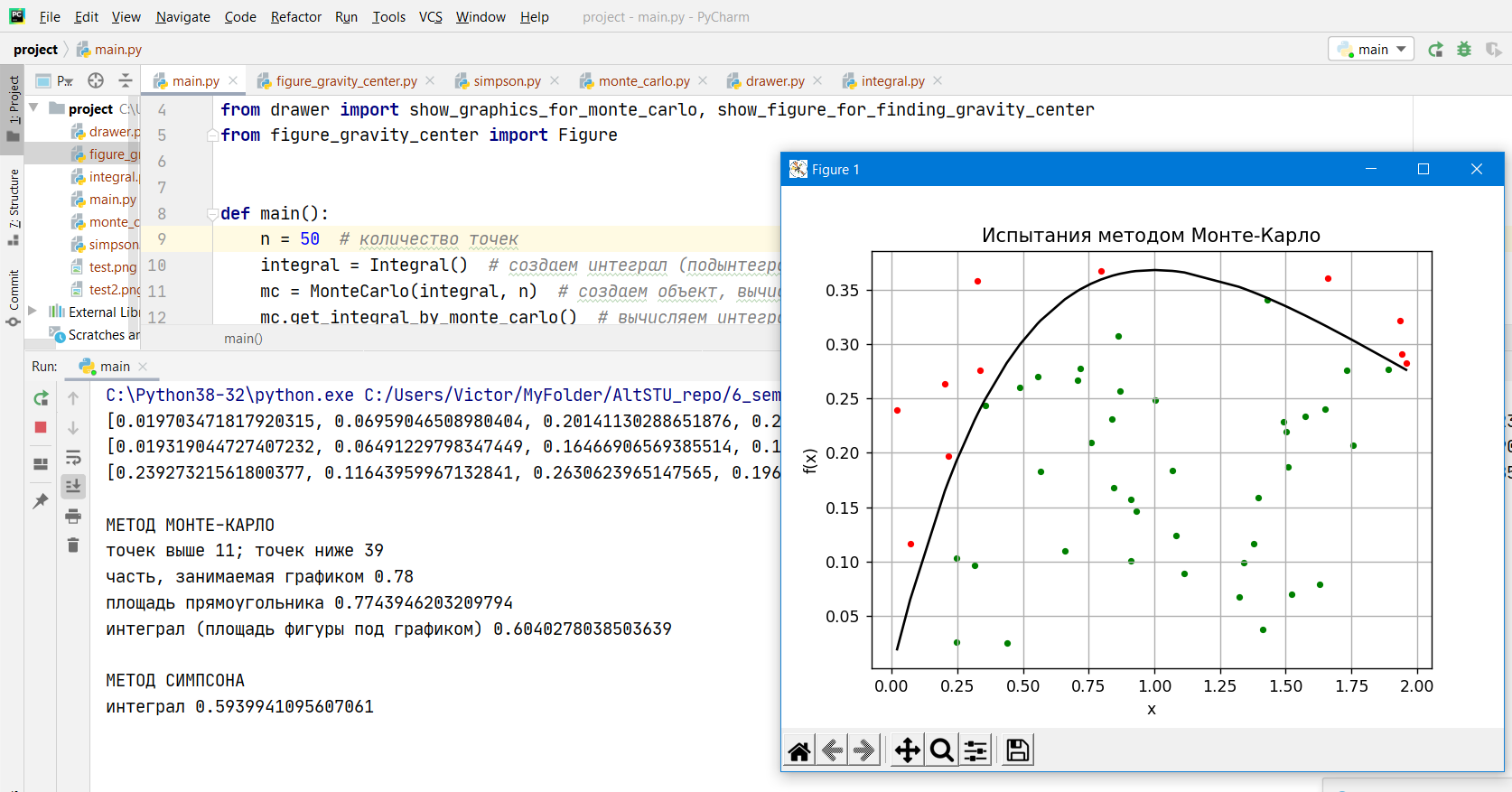
В эту сумму входят только те ni которые принадлежат области фигуры.

Тесты:

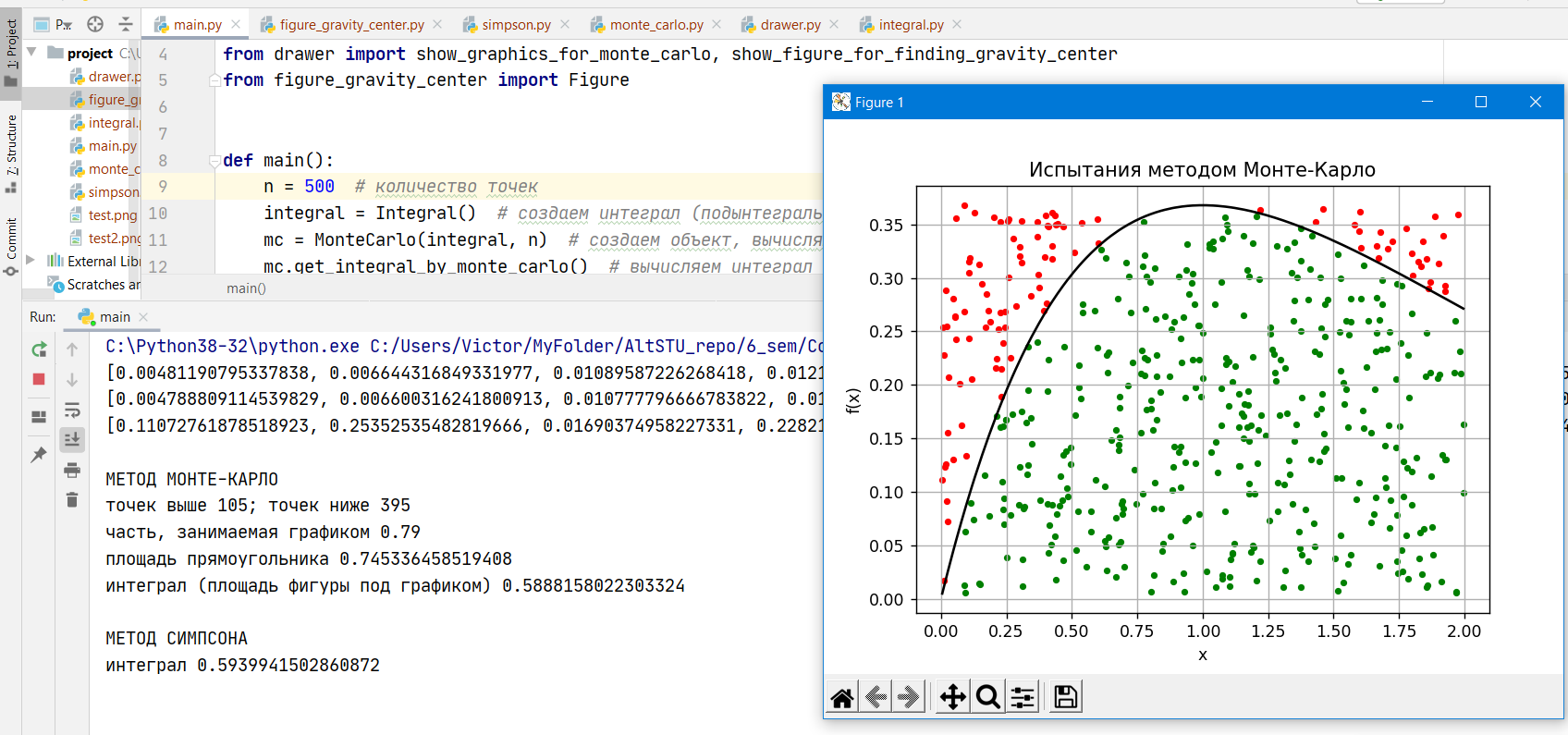
y=x\*e^(-x)

интегрирование от 0 до 2

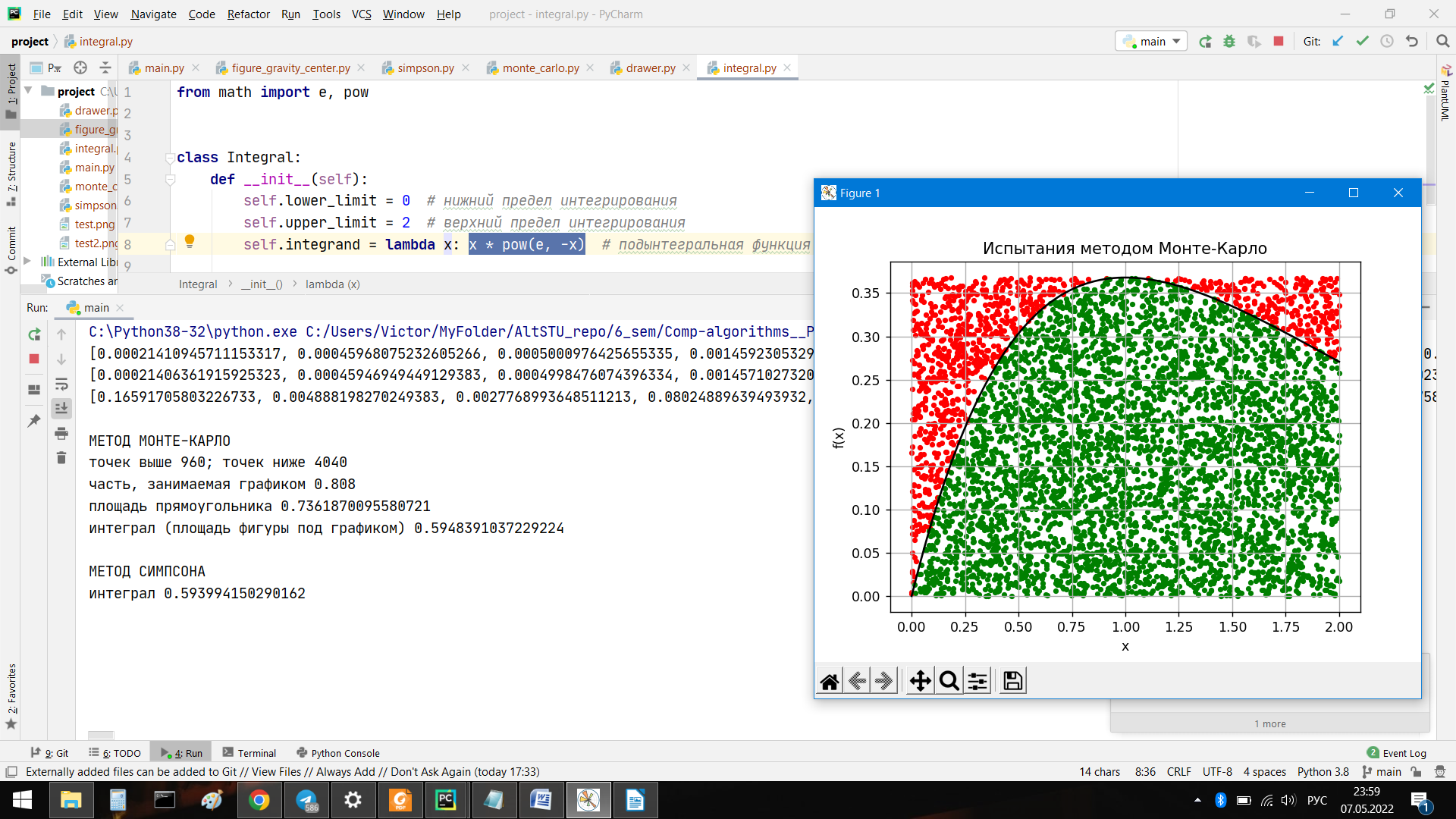
50 точек



500 точек

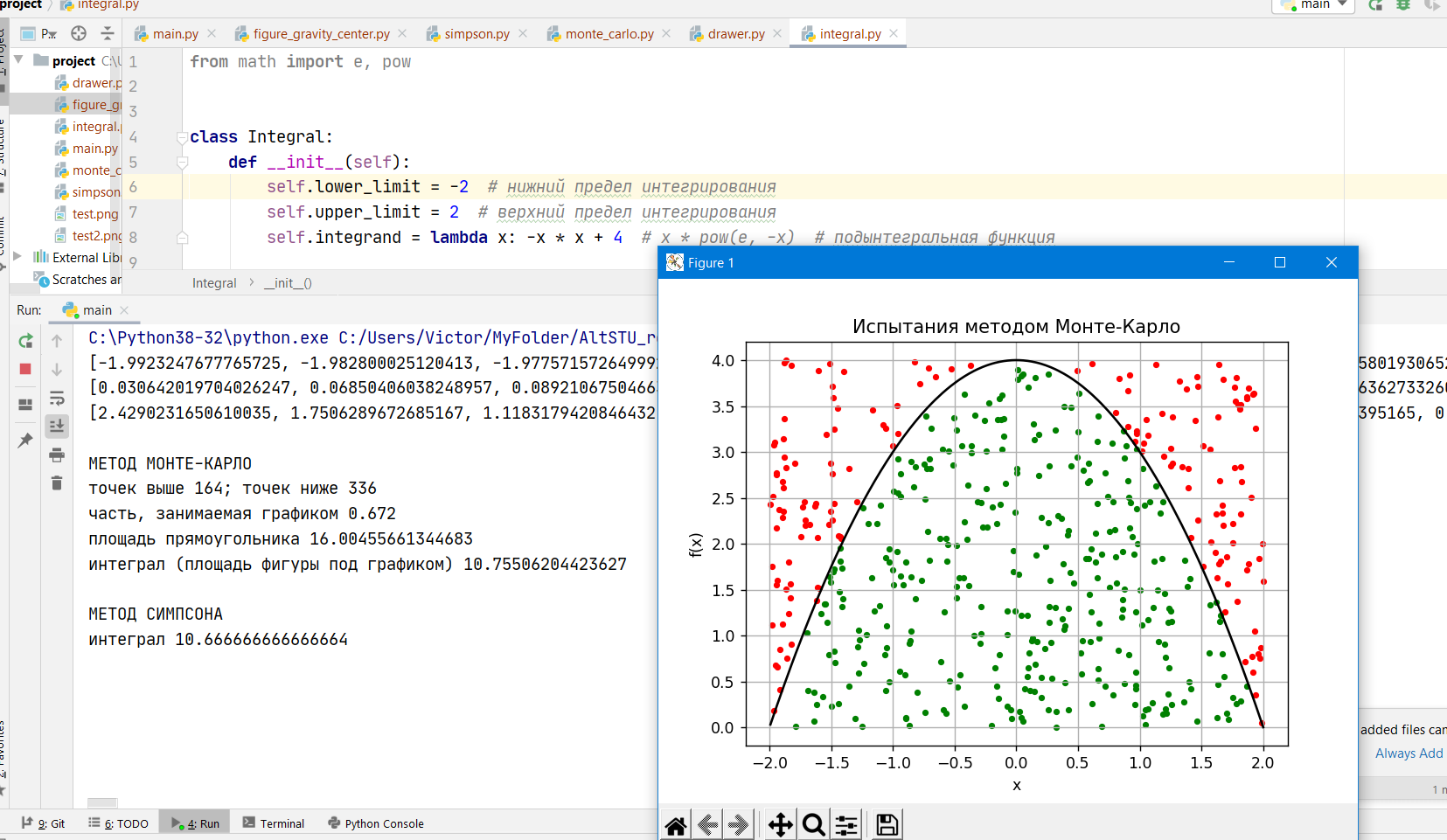


5000 точек

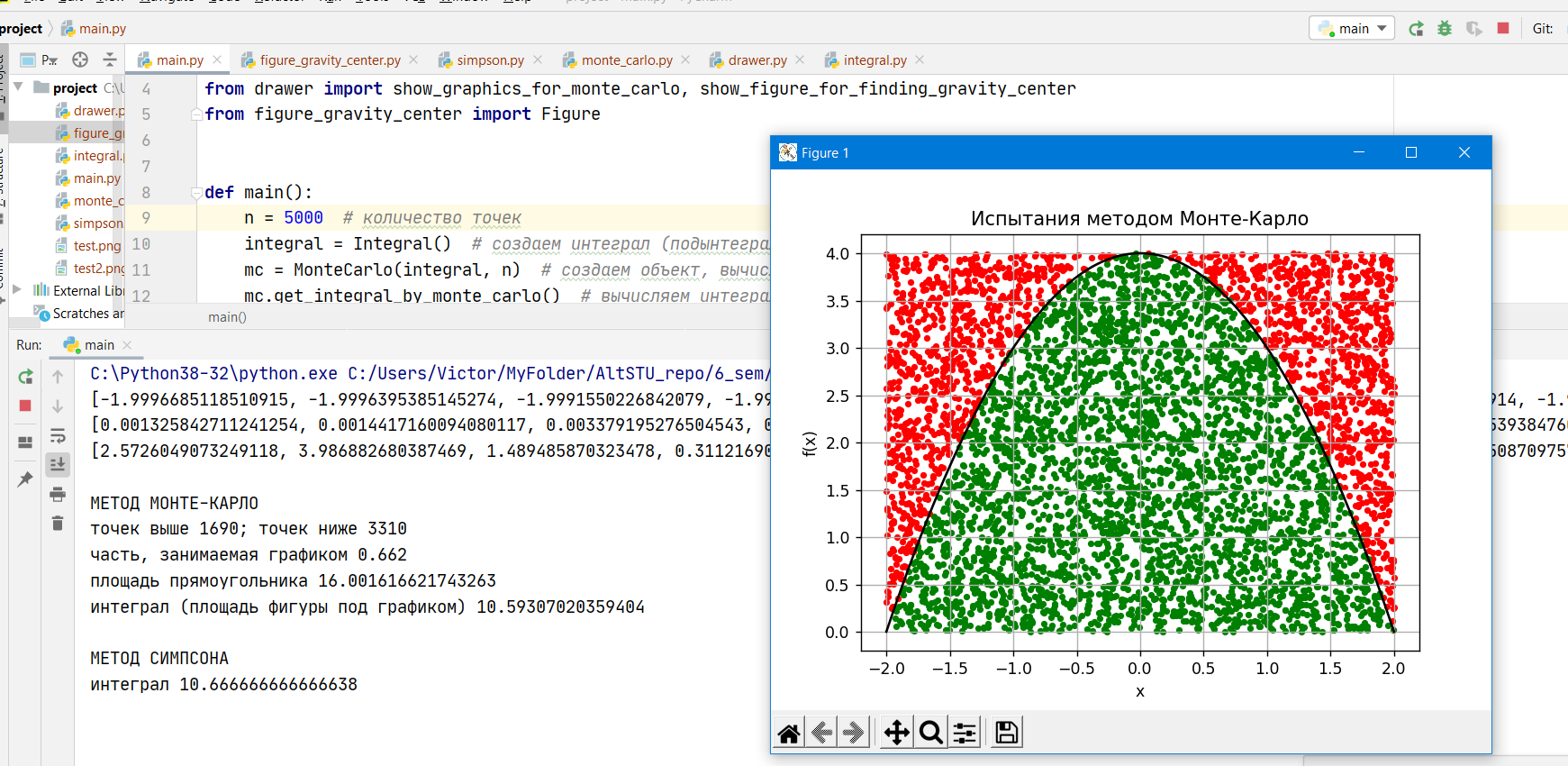


y=-x\*x+4

интегрирование от -2 до 2

500 точек

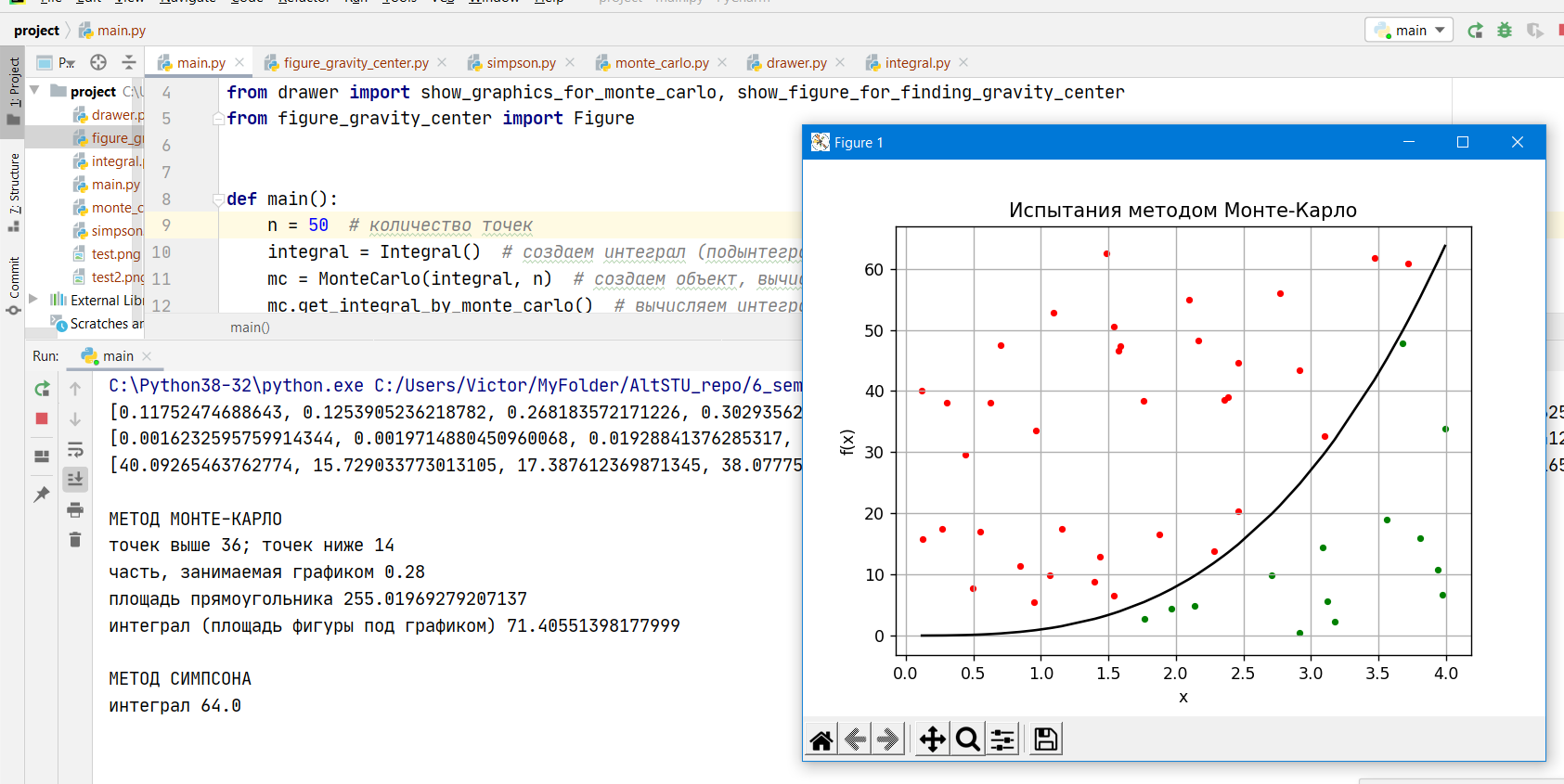
5000 точек



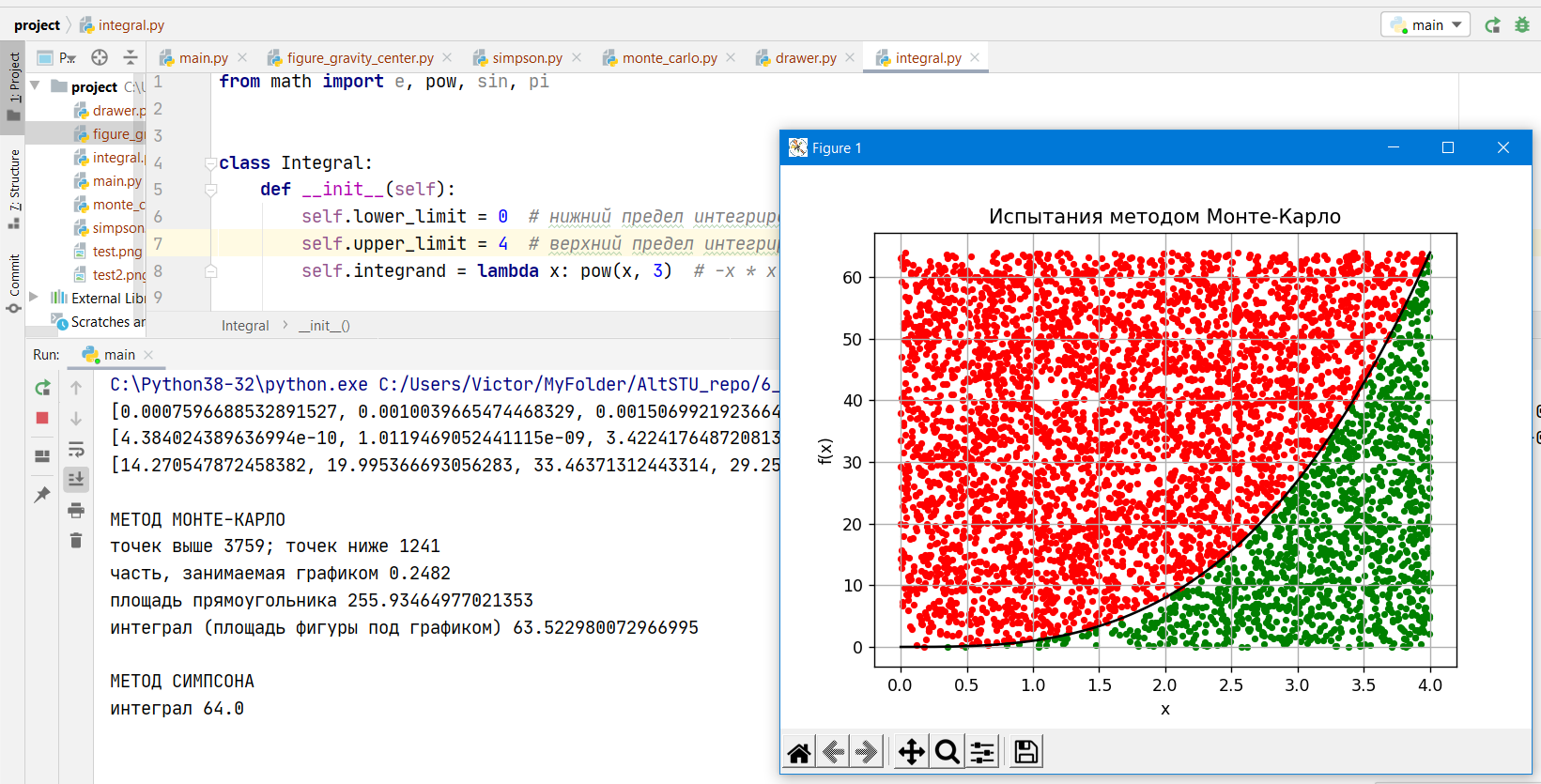
Y=x^3

Интегрирование от 0 до 4

50 точек

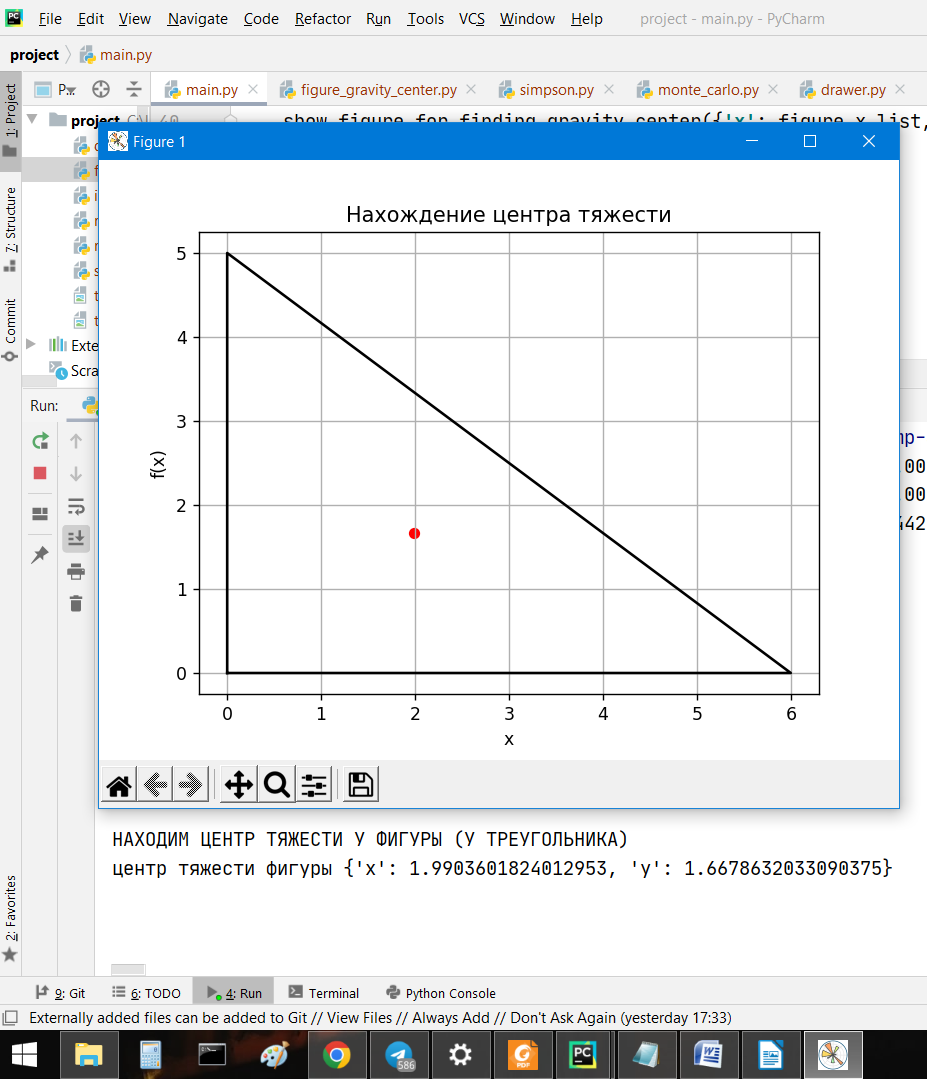


5000 точек



В методе Монте-Карло при увеличении числа точек, увеличивается точность вычисления интеграла. При маленьком количестве точек, мы можем получить весьма неверный результат, так как точки выбираются случайным образом.

Нахождение центра тяжести у фигуры (треугольника)



**Код:**

**main.py**

**from** integral **import** Integral  
**from** monte\_carlo **import** MonteCarlo  
**from** simpson **import** Simpson  
**from** drawer **import** show\_graphics\_for\_monte\_carlo, show\_figure\_for\_finding\_gravity\_center  
**from** figure\_gravity\_center **import** Figure  
  
  
**def** main():  
 n = 50 *# количество точек* integral = Integral() *# создаем интеграл (подынтегральная функция, пределы присваиваются в классе)* mc = MonteCarlo(integral, n) *# создаем объект, вычисляющий интеграл методом Монте-Карло* mc.get\_integral\_by\_monte\_carlo() *# вычисляем интеграл* print(mc.x\_list)  
 print(mc.y\_list)  
 print(mc.random\_y\_list)  
 print(**f'\nМЕТОД МОНТЕ-КАРЛО'**)  
 print(**f'точек выше {**mc.over\_points\_num**}; точек ниже {**mc.under\_points\_num**}'**)  
 print(**f'часть, занимаемая графиком {**mc.proportion**}'**)  
 print(**f'площадь прямоугольника {**mc.square\_area**}'**)  
 print(**f'интеграл (площадь фигуры под графиком) {**mc.integral\_value**}'**)  
  
 s = Simpson(integral, n, mc.x\_list, mc.y\_list)*# создаем объект, вычисляющий интеграл методом Симпсона* s.get\_integral\_by\_simpson() *# вычисляем интеграл* print(**f'\nМЕТОД СИМПСОНА'**)  
 print(**f'интеграл {**s.integral\_value**}'**)  
  
 graphic = {**'x'**: mc.x\_list, **'y'**: mc.y\_list}  
 under\_points = {**'x'**: mc.under\_graphic\_x\_list, **'y'**: mc.under\_graphic\_y\_list}  
 over\_points = {**'x'**: mc.over\_graphic\_x\_list, **'y'**: mc.over\_graphic\_y\_list}  
 show\_graphics\_for\_monte\_carlo(graphic, under\_points, over\_points)  
  
 figure = Figure()  
 figure.find\_gravity\_center()  
  
 print(**f'\nНАХОДИМ ЦЕНТР ТЯЖЕСТИ У ФИГУРЫ (У ТРЕУГОЛЬНИКА)'**)  
 print(**f'центр тяжести фигуры {**figure.gravity\_center**}'**)  
  
 show\_figure\_for\_finding\_gravity\_center({**'x'**: figure.x\_list, **'y'**: figure.y\_list}, figure.gravity\_center)  
  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 main()

**integral.py**

**from** math **import** e, pow, sin, pi  
  
  
**class** Integral:  
 **def** \_\_init\_\_(self):  
 self.lower\_limit = 0 *# нижний предел интегрирования* self.upper\_limit = 4 *# верхний предел интегрирования* self.integrand = **lambda** x: pow(x, 3) *# -x \* x + 4 # x \* pow(e, -x) # подынтегральная функция*

**monte\_carlo.py**

**from** random **import** uniform  
  
  
**class** MonteCarlo:  
 *""" метод Монте-Карло """* **def** \_\_init\_\_(self, integral, n):  
 self.integral = integral *# пределы интегрирования и подыинтегральная функция* self.n = n *# количество точек* self.x\_list = [] *# список x-ов* self.y\_list = [] *# список y-ов* self.min\_y = **None** *# минимальный y из списка y\_list* self.max\_y = **None** *# максимальный y из списка y\_list* self.under\_graphic\_x\_list = [] *# координаты x точек находящихся ниже графика* self.under\_graphic\_y\_list = [] *# координаты y точек находящихся ниже графика* self.over\_graphic\_x\_list = [] *# координаты x точек находящихся выше графика* self.over\_graphic\_y\_list = [] *# координаты y точек находящихся выше графика* self.under\_points\_num = 0 *# кол-во точек, которые ниже графика* self.over\_points\_num = 0 *# кол-во точек, которые выше графика* self.square\_area = 0 *# площадь прямоугольника* self.proportion = 0 *# часть графика, которую он занимает в прямоугольнике* self.integral\_value = 0 *# площадь под графиком (значение интеграла)* **def** get\_integral\_by\_monte\_carlo(self):  
 *""" вычисление определнного интеграла методом Монте-Карло на отрезке """* self.\_\_generate\_n\_x()  
 self.\_\_calculate\_n\_y()  
 self.\_\_find\_min\_max\_y()  
 self.\_\_generate\_n\_y()  
 self.\_\_get\_under\_over\_points()  
 self.\_\_get\_square\_area()  
 self.\_\_get\_integral\_value()  
  
 **def** \_\_generate\_n\_x(self):  
 *""" генерация координат x для n точек в интервале от lower\_limit до upper\_limit"""* self.x\_list = [uniform(self.integral.lower\_limit, self.integral.upper\_limit) **for** \_ **in** range(0, self.n)]  
 self.x\_list.sort()  
  
 **def** \_\_calculate\_n\_y(self):  
 *""" подсчет координат y=f(x) для n точек, где f(x) - подынтегральная функция """* self.y\_list = [self.integral.integrand(x) **for** x **in** self.x\_list]  
  
 **def** \_\_find\_min\_max\_y(self):  
 *""" нахождение минимального и максимального y в списке y\_list """* self.min\_y = min(self.y\_list)  
 self.max\_y = max(self.y\_list)  
  
 **def** \_\_generate\_n\_y(self):  
 *""" генерация координат y для n точек в интервале от min\_y до max\_y"""* self.random\_y\_list = [uniform(self.min\_y, self.max\_y) **for** \_ **in** self.x\_list]  
  
 **def** \_\_get\_under\_over\_points(self):  
 *""" нахождение точек, которые расположены под графиком """* **for** rand\_y, y, x **in** zip(self.random\_y\_list, self.y\_list, self.x\_list):  
 **if** rand\_y < y:  
 self.under\_graphic\_x\_list.append(x)  
 self.under\_graphic\_y\_list.append(rand\_y)  
 self.under\_points\_num += 1  
 **else**:  
 self.over\_graphic\_x\_list.append(x)  
 self.over\_graphic\_y\_list.append(rand\_y)  
 self.over\_points\_num += 1  
  
 **def** \_\_get\_square\_area(self):  
 *""" подсчет площади прямоугольника, в котором мы генерируем точки """* x = abs(self.integral.lower\_limit) + abs(self.integral.upper\_limit)  
 y = abs(self.min\_y) + (self.max\_y)  
 self.square\_area = x \* y  
  
 **def** \_\_get\_integral\_value(self):  
 *""" получение значения интеграла (площади под графиком) """* self.proportion = self.under\_points\_num / (self.under\_points\_num + self.over\_points\_num)  
 self.integral\_value = self.square\_area \* self.proportion

**simpson.py**

**class** Simpson:  
 *""" метод Симпсона """* **def** \_\_init\_\_(self, integral, n, x\_list, y\_list):  
 self.func = integral.integrand *# подыинтегральная функция* self.min\_x = integral.lower\_limit  
 self.max\_x = integral.upper\_limit *# пределы интегрирования* self.n = n *# количество точек* self.x\_list = x\_list  
 self.y\_list = y\_list  
  
 self.integral\_value = 0  
  
  
 **def** get\_integral\_by\_simpson(self):  
 *""" вычисление определнного интеграла методом Симпсона на отрезке """* h = (self.max\_x - self.min\_x) / self.n *# шаг* \_sum = self.func(self.min\_x) + self.func(self.max\_x)  
 k = 0  
 **for** i **in** range(1, self.n):  
 k = 2 + 2 \* (i % 2)  
 \_sum += k \* self.func(self.min\_x + i \* h)  
 \_sum \*= h/3  
 self.integral\_value = \_sum

**figure\_gravity\_center.py**

**from** random **import** uniform  
  
**class** Figure:  
 *""" фигура, у которой находим центр тяжести """* **def** \_\_init\_\_(self):  
 *# треугольник* self.x\_list = [0, 0, 6, 0]  
 self.y\_list = [0, 5, 0, 0]  
 self.max\_x = max(self.x\_list)  
 self.min\_x = min(self.x\_list)  
 self.max\_y = max(self.y\_list)  
 self.min\_y = min(self.y\_list)  
 self.tries\_num = 100000 *# кол-во испытаний* self.gravity\_center = {**'x'**: 0, **'y'**: 0}  
  
 **def** find\_gravity\_center(self):  
 *""" нахождение центра тяжести """* upx = upy = dow = 0 *# upx - числитель в формуле поиска x центра тяжести, upy - то же самое, но для y,  
 # dow - знаменатель для обеих формул* **for** i **in** range(0, self.tries\_num):  
 x = uniform(self.min\_x, self.max\_x)  
 y = uniform(self.min\_y, self.max\_y)  
 *# проверка того, входит ли очередная точка в область фигуры* **if** self.min\_y < y < -x \* (5/6) + 5:  
 upx += x  
 upy += y  
 dow += 1  
 self.gravity\_center[**'x'**] = upx/dow  
 self.gravity\_center[**'y'**] = upy/dow

**drawer.py**

**import** matplotlib.pyplot **as** plt  
  
  
**def** show\_graphics\_for\_monte\_carlo(graphic, under\_points, over\_points):  
 *""" графическое отображение испытаний методом Монте-Карло """* fig, ax = plt.subplots()  
 ax.plot(graphic[**'x'**], graphic[**'y'**], color=**'black'**)  
 plt.scatter(over\_points[**'x'**], over\_points[**'y'**], color=**'red'**, s=10)  
 plt.scatter(under\_points[**'x'**], under\_points[**'y'**], color=**'green'**, s=10)  
  
 ax.set(xlabel=**'x'**, ylabel=**'f(x)'**,  
 title=**'Испытания методом Монте-Карло'**)  
 ax.grid()  
  
 fig.savefig(**"test.png"**)  
 plt.show()  
  
  
**def** show\_figure\_for\_finding\_gravity\_center(figure, gr\_center\_point):  
 *""" графическое отображение фигуры и ее центра тяжести """* fig, ax = plt.subplots()  
 ax.plot(figure[**'x'**], figure[**'y'**], color=**'black'**)  
 plt.scatter(gr\_center\_point[**'x'**], gr\_center\_point[**'y'**], color=**'red'**, s=30)  
  
 ax.set(xlabel=**'x'**, ylabel=**'f(x)'**,  
 title=**'Нахождение центра тяжести'**)  
 ax.grid()  
  
 fig.savefig(**"test2.png"**)  
 plt.show()

**Вывод:** как видно из тестов, метод Симпсона использовать явно предпочтительней, так как он сходится по времени при увеличении числа процессов так же, как и методы Монте-Карло, при этом точность вычисления остается выше и увеличивается стабильно. Для методов Монте-Карло такого эффекта не наблюдается, поэтому их разумно использовать только для быстрой оценки значения интеграла в особых случаях.