Задача 1

Написать функцию `double_this(arr)`, принимающую на вход массив `arr`, состоящий из чисел, и возвращающую массив, полученный удвоением каждого элемента `arr`.

Подсказка: Операции с массивами действуют поэлементно.

Задача 2

Написать функцию `select_even(arr)`, принимающую на вход массив целых чисел `arr` и возвращающую новый массив, который состоит из всех *чётных* элементов `arr`.

Подсказка: напомним, что все арифметические операции, а также операции сравнения, действуют на массивы поэлементно.

Задача 3

Написать функцию `wipe_even(arr, target_value, in_place)`, принимающую на вход массив целых чисел `arr`, и возвращающую массив, полученный из `arr` путём замены всех чётных элементов на `target_value`. Если `target_value` не указано, то оно должно считаться равным числу 0. Если указан параметр `in_place` и он равен `True`, то функция должна менять исходный массив, а если не указан или указан в `False`, то сохранять его неизменным.

Задача 4

Написать функцию `weighted_sum(weights, grades, normalize)`, возвращающую взвешенную сумму оценок, записанных в массив `grades`, в соответствии с весами, записанными в массив `weights`. Например, для `weights = np.array([0.3, 0.3, 0.4])` и `grades = np.array([7, 9, 8])` функция должна вернуть число $0.3 \times 7 + 0.3 \times 9 + 0.4 \times 8 = 8.0$.

Если параметр `normalize` установлен в `True`, а сумма всех весов отличается от 1, то следует умножить все веса на одно и то же число таким образом, чтобы их сумма была равна 1, в противном случае следует использовать веса «как есть», даже если их сумма отличается от 1. Если функция запущена без указания параметра `normalize`, следует считать, что `normalize = False`.

Подсказка: Вам помогут функции `np.dot()` и `np.sum()`. Встроенная функция `sum()` также работает с массивами `numpy`, но гораздо медленнее (проверьте с помощью `%timeit`!)

Задача 5

Написать функцию `mean_by_gender(grades, genders)`, принимающую на вход два массива одинаковой длины: в массиве `grades` записаны оценки некоторых студентов, а в массиве `genders` — их пол в виде строк `male` или `female`. Требуется вернуть словарь, ключами которого будут строки `male` и `female`, а записями — среднее арифметическое оценок студентов соответствующего пола.

Например, если `grades = np.array([5, 4, 3, 5, 2])` и `genders = np.array(["female", "male", "male", "female", "male"])`, функция должна вернуть словарь `{'male': 3.0, 'female': 5.0}`.

Подсказка. Для быстрого вычисления среднего есть функция `np.mean()` или соответствующий метод у объектов типа `numpy.array`.

Метод Монте-Карло

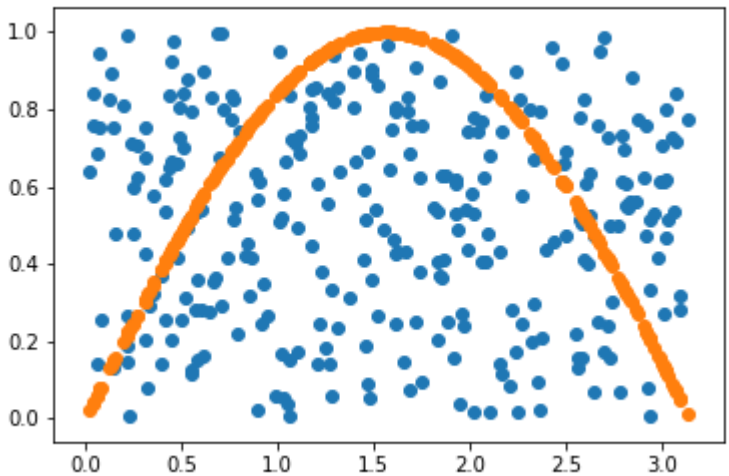
Рассматривается геометрический метод Монте-Карло нахождения площади фигуры. Краткая суть метода:

- фигура ограничивается прямоугольников A с легко определимой площадью S_A;
- в данный прямоугольник случайно "накидываются" точки (из равномерного распределения);
- все точки делятся на два класса: в фигуре и вне её;
- при большом количестве точек N, отношения количества точек внутри фигуры N_{internal} к N стремится к площади фигуры к S_A.

$$\frac{N_{\text{internal}}}{N} \underset{N \to +\infty}{\to} \frac{S}{S_A}$$

$$S \approx \frac{N_{\text{internal}}}{N} S_A$$

Пример "накидывания" точек для $1 < y < sin(x), x \in (0, \pi)$



Задача

Вычислите площадь данной фигуры: $1 < y < sin(x), x \in (0, \pi)$.