Задание 1. Проанализировать скорость и сложность одного любого алгоритма из разработанных в рамках домашнего задания первых трех уроков.
Примечание. Идеальным решением будет:
a. выбрать хорошую задачу, которую имеет смысл оценивать,
b. написать 3 варианта кода (один у вас уже есть),
c. проанализировать 3 варианта и выбрать оптимальный,
d. результаты анализа вставить в виде комментариев в файл с кодом (не забудьте указать, для каких N вы проводили замеры),
e. написать общий вывод: какой из трёх вариантов лучше и почему.
Программа: Найти сумму n элементов следующего ряда чисел: 1, -0.5, 0.25, -0.125,...

Знач	**Рекурсия (les2_task4_ver1.py)**		**Цикл (les2_task4_ver2.py)**		**Исп. functool(les2_task4_ver3.py)**	
	timeit	**cProfile**	**timeit**	**cProfile**	**timeit**	**cProfile**
10	5.56×10^{-6}	11	4.87×10^{-6}	1	283×10^{-9}	11
100	54.9×10^{-6}	101	32.3×10^{-6}	1	287×10^{-9}	101
200	113×10^{-6}	201	67.5×10^{-6}	1	270×10^{-9}	201
500	308×10^{-6}	501	163×10^{-6}	1	-	-
900	581×10^{-6}	901	314×10^{-6}	1	-	-
2000	-	-	692×10^{-6}	1	-	-
					При значении больше 200 программа выдает ошибку	

Вывод: Сложность всех программ одинаковая. Самая быстрая работая программы с использованием functool, но ввод значений ограничевается 200. На втором месте по скорочти работы , программа использующая цикл. Кол-во вводимых значений на порядок больше чем в остальных вариантах этой программы. Оптимальный вариант программы с циклом.

Задание2. Написать два алгоритма нахождения i-го по счёту простого числа. Функция нахождения простого числа должна принимать на вход натуральное и возвращать соответствующее простое число. Проанализировать скорость и сложность алгоритмов.
Первый — с помощью алгоритма «Решето Эратосфена».
Примечание. Алгоритм «Решето Эратосфена» разбирался на одном из прошлых уроков. Используйте этот код и попробуйте его улучшить/оптимизировать под задачу.
Второй — без использования «Решета Эратосфена».
Примечание. Вспомните классический способ проверки числа на простоту.

Знач	**Решето Эрастофена (task2_resheto.py)**		**Моя программа (task2_my.py)**	
	timeit	**cProfile**	**timeit**	**cProfile**
10	425×10^{-6}	91	93×10^{-6}	4
100	149×10^{-3}	1627	13×10^{-3}	4
200	-	3679	57×10^{-3}	4

Ввывод. Сложность и объем алгоритма у этих программ примерно одинаковая. Из-за создания списков программа Решето Эрастофена проигравает моей программе, используйщей циклы в разы. Программу Решето Эрастофена не запускал в тесте timeit с значением 200 – очень долго ждать.