САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе № 0

по курсу «Алгоритмы и структуры данных»

Тема: Введение

Вариант: 1

Выполнил:

Дегтярь Г.С

K3141

Проверил(а):

Афанасьев А. В.

Санкт-Петербург

2024 г.

# Содержание отчета

[**Содержание отчета**](#_heading=h.gjdgxs) **2**

[**Задачи по варианту**](#_heading=h.30j0zll) **3**

Задача №1

* Задача № 1.1
* Задача № 1.2
* Задача № 1.3
* Задача № 1.4

Задача № 2

Задача № 3

Задача № 4 4

[**Вывод**](#_heading=h.2et92p0) **5**

# Задачи по варианту

## Задача №1. Ввод – вывод

Текст задачи:

Вам необходимо выполнить 4 следующих задачи:

* **Задача № 1.1. a + b.**

Текст задачи:

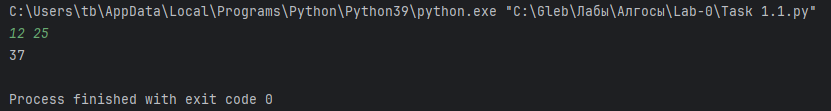
В данной задаче требуется вычислить сумму двух заданных чисел. Вход: одна строка, которая содержит два целых числа a и b. Для этих чисел выполняются условия −10^9 ≤ a, b ≤ 10^9 . Выход: единственное целое число — результат сложения a + b.

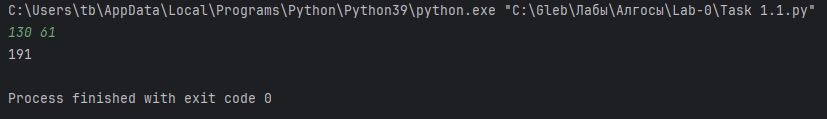
Листинг кода:

| *a***,** b **=** **map(int,input().***split***())**  **print(**a **+** b**)** |
| --- |

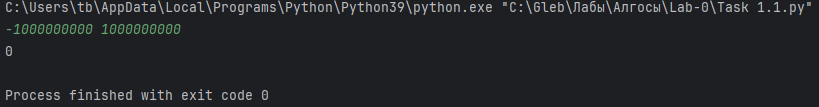
Текстовое объяснение решения:

1. В первой строке мы вводим переменные a, b с клавиатуры, используя функцию input(). Для разделения чисел при вводе мы используем функцию .split(), которая по умолчанию использует символ “ “ (пробел) для разделения ввода. Функцию map мы используем для того, чтоб задать каждой из введенных переменных тип данных: int
2. Во второй строке мы выводим сумму введенных ранее чисел

Результат работы кода на примерах задачи:  




Результат работы кода на максимальных и минимальных значениях:



* **Задача № 1.2. a + b^2.**

Текст задачи:  
В данной задаче требуется вычислить значение a + b 2 . Вход: одна строка, которая содержит два целых числа a и b. Для этих чисел выполняются условия −10^9 ≤ a, b ≤ 10^9 . Выход: единственное целое число — результат сложения a + b 2 .

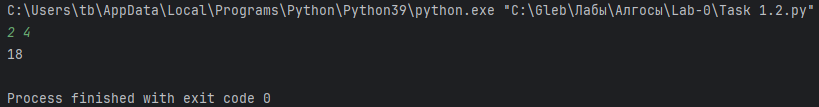
Листинг кода:

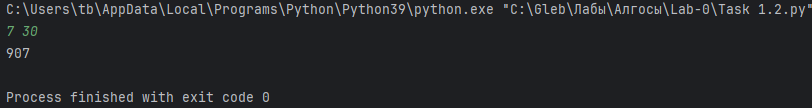
| *a***,** b **=** **map(int,input().***split***())**  **print(**a **+** b**\*\***2**)** |
| --- |

Текстовое объяснение решения:

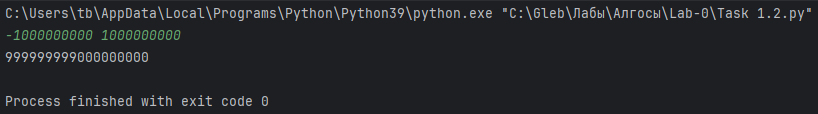
Единственное отличие от задачи № 1.1 заключается в том, что мы перед подсчетом суммы чисел a, b возводим число b в квадрат операцией “\*\*”.

Результат работы кода на примерах задачи:





Результат работы кода на максимальных и минимальных значениях:



* **Задача № 1.3. a + b с использованием файлов.**

Текст задачи:  
Имя входного файла: input.txt • Имя выходного файла: output.txt • Формат входного файла. Входной файл состоит из одной строки, которая содержит два целых числа a и b. Для этих чисел выполняются условия −10^9 ≤ a, b ≤ 10^9 . • Формат выходного файла. Выходной файл единственное целое число — результат сложения a + b.

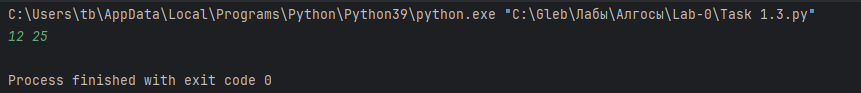
Листинг кода:

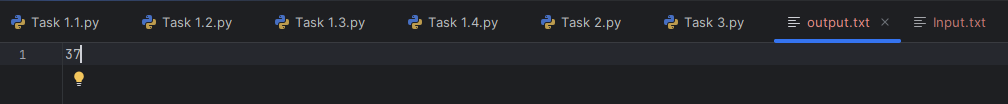
| **with** **open(**"input.txt"**,**"w"**)** **as** k**:**  a**,**b **=** **map(int,** **input().***split***())**  k**.***write***(str(**a**)** **+** " " **+** **str(**b**))**  **with** **open(**"input.txt"**,** "r"**)** **as** f**:**  string **=** f**.***read***().***split***()**  line **=** **[int(**x**)** **for** x **in** string**]**  res **=** line**[**0**]** **+** line**[**1**]**  **with** **open(**"output.txt"**,** "w"**)** **as** g**:**  g**.***write***(str(**res**))** |
| --- |

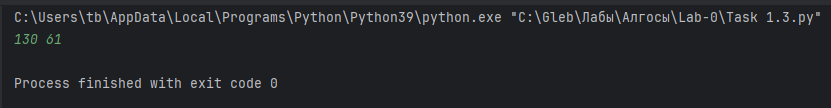
Текстовое объяснение решения:

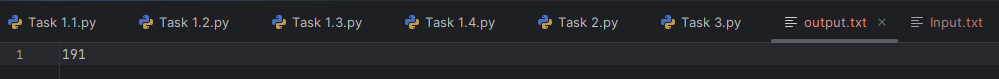
1. В первом абзаце кода мы создаем файл с названием “input.txt” в директории проекта, вводим 2 числа с клавиатуры и записываем их в этот файл.
2. Во втором абзаце кода мы открываем созданный ранее файл, считываем из него два, записанных ранее, числа и создаем из них список string (пока что числа в формате str). Далее с помощью внутреннего цикла мы создаем новый список line, придавая числам формат int. Затем мы находим сумму данных чисел и присваиваем полученный результат переменной res.
3. В третьем абзаце мы создаем файл с названием “output.txt” в директории проекта (он будет хранить наш ответ), записываем в него полученный нами результат

Результат работы кода на примерах из текста задачи:

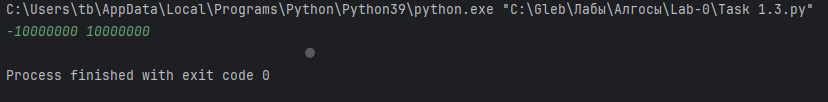
1. 

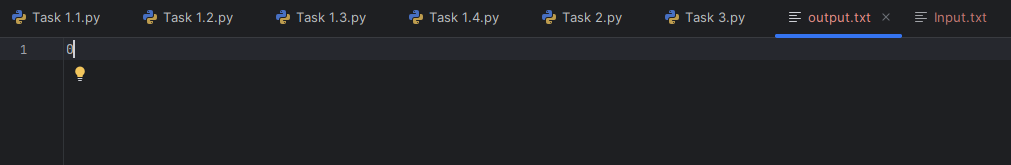


1. 



Результат работы кода на максимальных и минимальных значениях:

****

****

* **Задача № 1.4. a+b^2 с использованием файлов:**

Текст задачи:

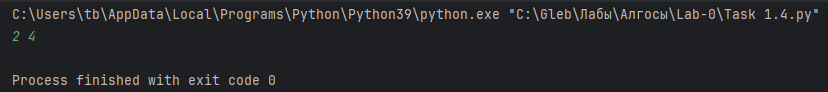
Выполните задачу a+b^2 с использованием файлов аналогично предыдущему пункту.

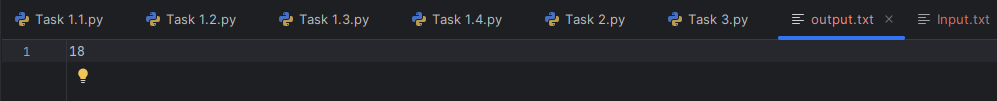
Листинг кода:

| **with** **open(**"input.txt"**,**"w"**)** **as** k**:**  a**,**b **=** **map(int,** **input().***split***())**  k**.***write***(str(**a**)** **+** " " **+** **str(**b**))**  **with** **open(**"input.txt"**,** "r"**)** **as** f**:**  string **=** f**.***read***().***split***()**  line **=** **[int(**x**)** **for** x **in** string**]**  res **=** line**[**0**]** **+** line**[**1**]\*\***2  **with** **open(**"output.txt"**,** "w"**)** **as** g**:**  g**.***write***(str(**res**))** |
| --- |

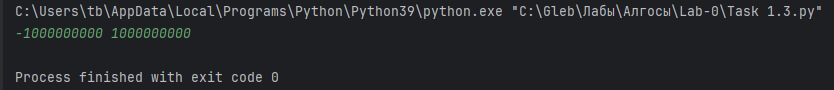
Текстовое объяснение решения:  
Единственным отличием в задаче 1.4 от задачи 1.3 является то, что прежде, чем посчитать сумму, мы возводим второе число во вторую степень line[1]\*\*2

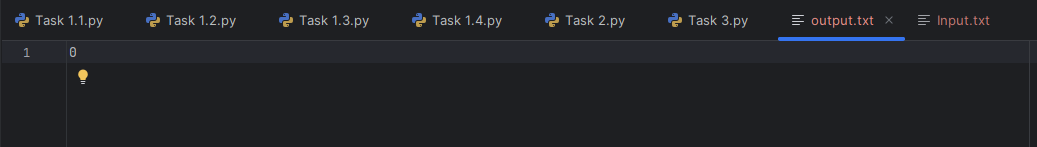
Результат работы кода на примерах из текста задачи:



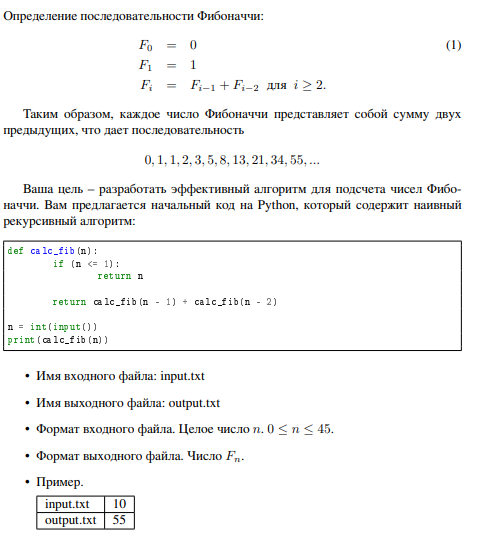


Результат работы кода на максимальных и минимальных значениях:





Вывод по задаче №1: Решать данное задание было очень интересно и полезно, оно хорошо показывает на практике принцип работы с файлами в python.   
  
**Задание № 2. Число Фибоначчи:**

Текст задачи:  
****

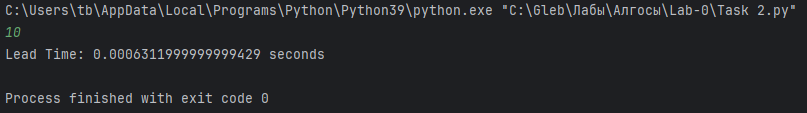
Листинг кода:

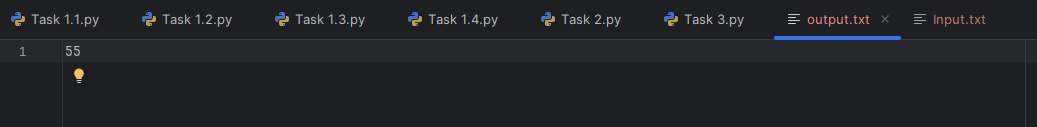
| **import** time  **with** **open(**"input.txt"**,**"w"**)** **as** w**:**  n **=** **input()**  w**.***write***(**n**)**  start **=** time**.***perf\_counter***()**  **with** **open(**"input.txt"**,**"r"**)** **as** f**:**  n **=** f**.***read***()**  n **=** **int(**n**)**  **def** calc\_fib**(**n**):**  **if** n **<=** 1**:**  **return** n  a **=** **[**0**,** 1**]**  **for** i **in** **range(**0**,** n **-** 1**,** 1**):**  j **=** a**[**0**]** **+** a**[**1**]**  a**[**0**]** **=** a**[**1**]**  a**[**1**]** **=** j  **return** a**[**1**]**  **with** **open(**"output.txt"**,**"w"**)** **as** w2**:**  w2**.***write***(str(**calc\_fib**(**n**)))**  end **=** time**.***perf\_counter***()**  **print** **(**"Lead Time: " **+** **str(**end **-** start**)** **+** " seconds"**)** |
| --- |

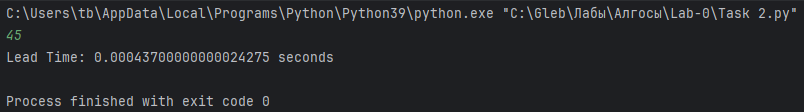
Текстовое объяснение решения:

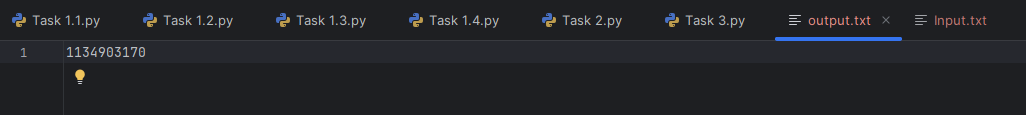
1. Импортируем библиотеку time для дальнейшего подсчета времени выполнения кода
2. Создаем файл “input.txt” и записываем в него номер числа Фибоначчи, которое нам нужно посчитать
3. Начинаем отсчет времени
4. Открываем ранее созданный нами файл и считываем из него входные данные задачи
5. Создаем функцию, которая будет высчитывать число Фибоначчи, на вход для этой функции будет подаваться только одно число – номер числа Фибоначчи. Затем с помощью функции if мы отсеиваем сценарий, при котором нас могли попросить найти нулевое или первое числа Фибоначчи, создаем отдельный вывод для этих частных случаев. Создаем список, содержащий первые два числа Фибоначчи. Создаем цикл от 0 до n-1 с шагом 1, в котором мы складываем числа, содержащиеся в списке, переписываем список так, чтобы полученное число встало под индекс 1, а прошлое число, стоявшее под индексом 1, заменило число под индексом 0. Так получается, что с каждым шагом цикла наш список “плывет”, пошагово увеличивая свои значения. В конце работы списка мы получим число Фибоначчи под номером n.
6. Записываем полученное число Фибоначчи в файл “output.txt”.
7. Заканчиваем отсчет времени (т. к. значительные затраты времени в данном алгоритме приходятся на запись числа ответа в файл “output.txt”, мы должны ставить финальную точку отсчета после указанной операции.)
8. Вычитаем из конечной точки отсчета времени начальную и выводим сколько времени затратила наша программа на выполнение.

Результат работы кода на примерах задачи:

1. 



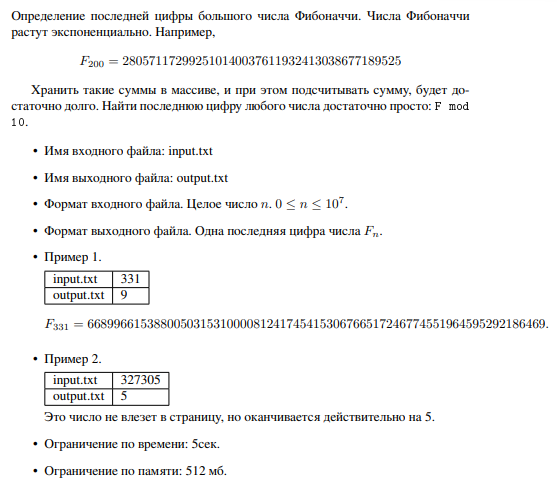
1. 



|  | Время выполнения | Затраты памяти |
| --- | --- | --- |
| Нижняя граница диапазона значений входных данных из текста задачи | 0.0003828999999999638 seconds |  |
| Пример из задачи | 0.0006269999999999332 seconds |  |
| Пример из задачи | 0.00043700000000024275 seconds |  |
| Верхняя граница диапазона значений входных данных из текста задачи | 0.00043700000000024275 seconds |  |

Вывод по задаче № 2:

Данная задача интересна тем, что обычный рекурсивный алгоритм решения потребует слишком много времени на выполнение кода. Самым оптимальным решением, которое я смог найти, стал линейный алгоритм, который заменят элементы списка, а не хранит их все разом.

**Задача № 3. Еще про числа Фибоначчи:**Текст задачи:  


Листинг кода:

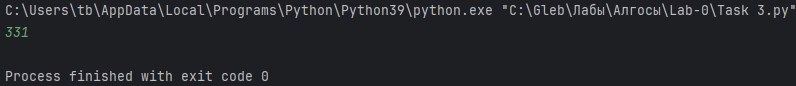
| **import** time  **import** sys  **with** **open(**"C:\Gleb\Лабы\Алгосы\Lab-0\input.txt"**,**"w"**)** **as** w**:**  n **=** **input()**  w**.***write***(**n**)**  start **=** time**.***perf\_counter***()**  **with** **open(**"input.txt"**,**"r"**)** **as** f**:**  n **=** f**.***read***()**  n **=** **int(**n**)**  a **=** **[**0**,**1**]**  **for** i **in** **range** **(**2**,**60**,**1**):**  a**.***append***((**a**[**i **-** 1**]** **+** a**[**i **-** 2**])** **%** 10**)**  **def** calc\_fib**(**n**):**  **if** n **<=** 59**:**  **return** a**[**n**]** **%** 10  ind **=** n **-** **((**n **//** 60**)** **\*** 60**)**  ind **=** **int(**ind**)**  **return** a**[**ind**]** **%** 10  end **=** time**.***perf\_counter***()**  **with** **open(**"output.txt"**,**"w"**)** **as** w2**:**  w2**.***write***(str(**calc\_fib**(**n**))** **+** "\n" **+** "lead time: " **+** **str(**end **-** start**)** **+** " seconds"**)** |
| --- |

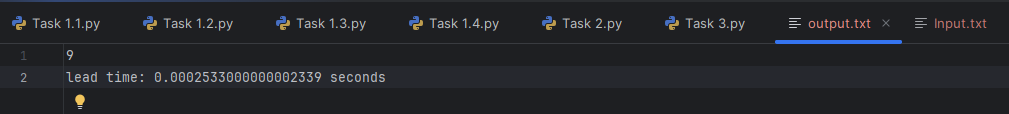
Текстовое объяснение решения:

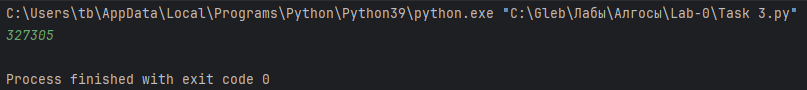
1. Подключаем библиотеку time для подсчета времени выполнения кода
2. Библиотека sys нам не нужна (нужна была бы для работы с памятью, но я забыл ее убрать, когда отказался от идеи подсчета памяти)
3. Привычным образом создаем файл “input.txt” и записываем в него входные данные
4. Начинаем отсчет времени работы программы
5. Считываем из файла “input.txt” входные данные и сразу создаем список, состоящий из первых двух чисел Фибоначчи.
6. Создаем цикл, который с помощью первых двух чисел Фибоначчи высчитываем все числа Фибоначчи до 59го включительно и записывает цифры, которыми они оканчиваются в список a. В данной задаче нам не нужно выводить сами числа Фибоначчи, а лишь их последнюю цифру, при этом мы знаем, что через каждые 60 чисел Фибоначчи цифры на их конце повторяются. Это значит, что мы можем создать формулу и с ее помощью находить последнюю цифру любого нужного нам числа Фибоначчи.
7. Создаем функцию для нахождения последних цифр чисел Фибоначчи. С помощью функции if мы создаем условие, что наш алгоритм получает на вход номер числа Фибоначчи <= 59, он может сразу выдать нужный ответ т.к ранее мы уже создали список, хранящий первые 59 чисел Фибоначчи, достаточно лишь обратиться к этому числу по индексу n и найти его последнюю цифру с помощью операции % 10.
8. Создаем переменную ind, в которую будет записан индекс по которому мы будем обращаться к списку. Для того чтоб вычислить этот индекс я придумал соответствующую формулу: ind = n - ((n // 60) \* 60). Далее мы обращаемся к заранее созданному списку по нашему индексу ind и выводим его.
9. Затем мы заканчиваем отсчет времени (необязательно включать создание файла “output.txt” в промежуток подсчета времени т.к на выходе мы получим лишь одну цифру и создание файла с таким маленьким объемом данных не может занять много времени
10. Создаем файл “output.txt” и записываем в него наш ответ и время

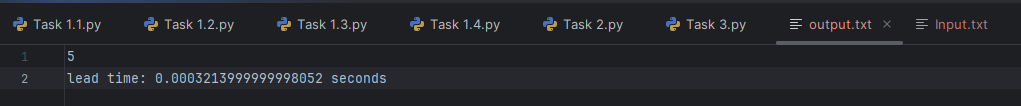
выполнения кода.

Результат работы кода на примерах из текста задачи:

1. 

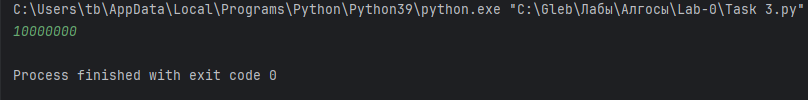


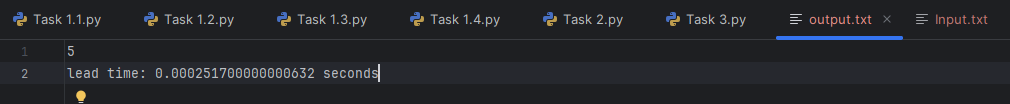
1. 



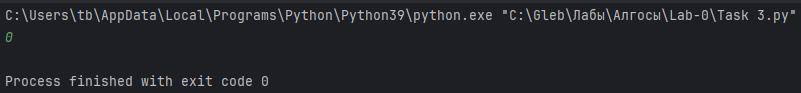
Результат работы кода на максимальных и минимальных значениях:

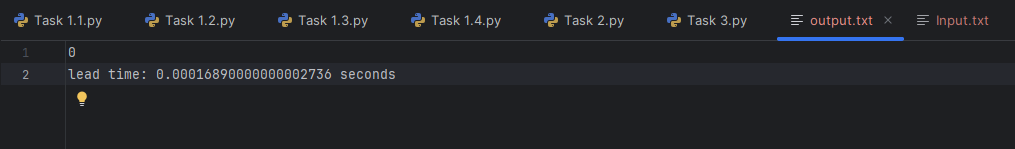
Max:





Min:





|  | Время выполнения | Затраты памяти |
| --- | --- | --- |
| Нижняя граница диапазона значений входных данных из текста задачи | 0.00041389999999996707 seconds |  |
| Пример из задачи | 0.00019699999999978068 seconds |  |
| Пример из задачи | 0.00017399999999989646 seconds |  |
| Верхняя граница диапазона значений входных данных из текста задачи | 0.00017140000000015476 seconds |  |

Вывод о задаче: Данная задача потребовала намного более нестандартного подхода, чем предыдущие. На мой взгляд ее нельзя решить никаким линейным алгоритмом т. к. при больших входящих данных программа слишком долго выполняет код. Самым оптимальным решением, как я считаю, станет создание формулы для прямого нахождения нужного числа Фибоначчи (или, как в нашем случае, его последней цифры)

**Задание № 4. Тестирование ваших алгоритмов:**

Текст задачи: Задача: вам необходимо протестировать время выполнения вашего алгоритма в Задании 2 и Задании 3.

Вывод о задаче:   
Благодаря использованию линейного алгоритма в второй задаче и формулы в третьей, мне удалось сократить время выполнения данных программ до минимальных значений. При использовании иных решений код выполнялся либо более 5 секунд, либо зависал вовсе.

# Вывод :

Лабораторная работа направлена на изучение основных методов работы с файлами в python, тестирования программы, создания наиболее эффективного алгоритма. Данная работа помогла мне освоить перечисленные навыки и вспомнить основные метода работы с языком программирования python.