САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №1

по курсу «Алгоритмы и структуры данных»

Тема: Сортировка вставками,выбором,пузырьковая

Выполнил:

Просветова В.Д.

К3141

Проверила:

Афанасьев.А.В.

Санкт-Петербург

2024 г.

# Содержание отчета

Содержание отчета:

Задача №1: Сортировка вставкой

Задача №2: Сортировка вставкой +

Задача №3:Сортировка вставкой по убыванию

Задача №4:Линейный поиск

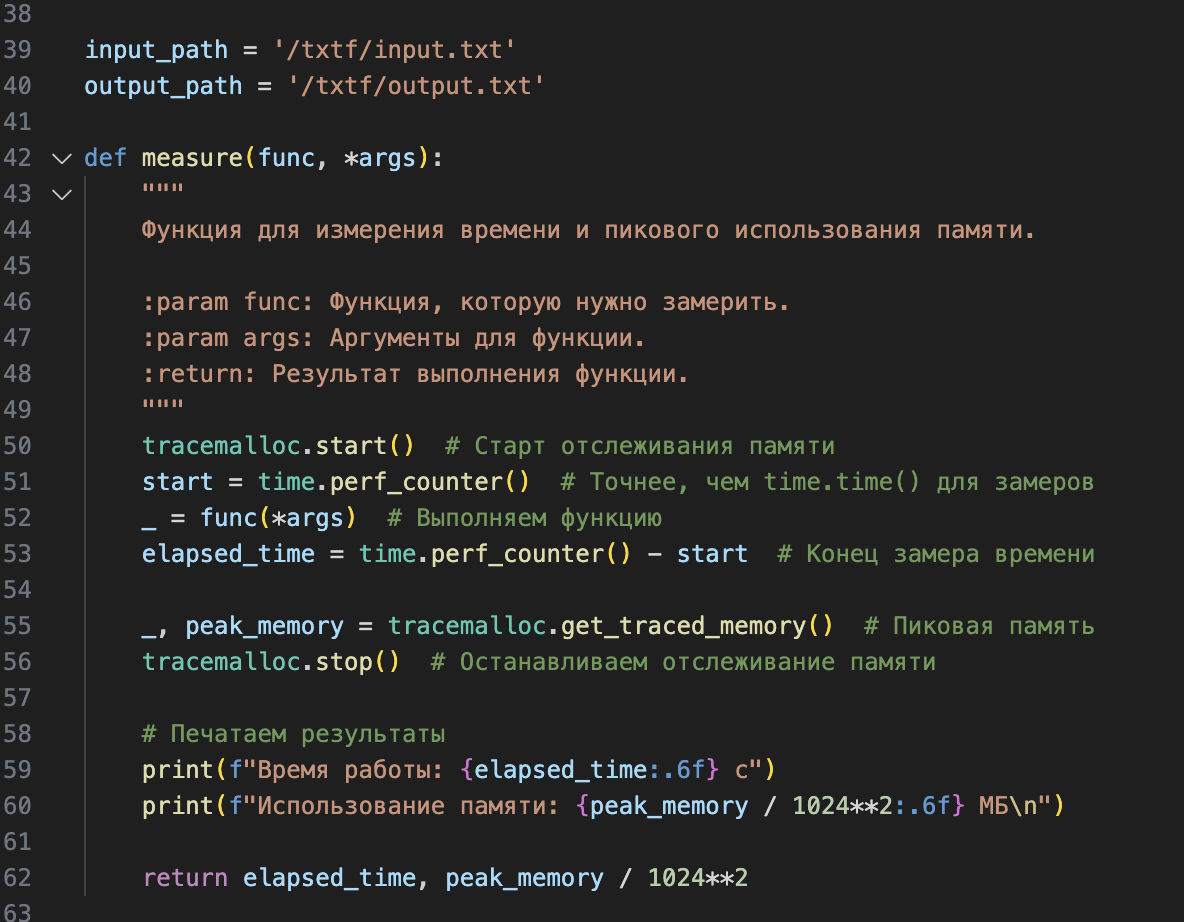
Задача №5:Сортировка выбором

Задача №6:Пузырьковая сортировка

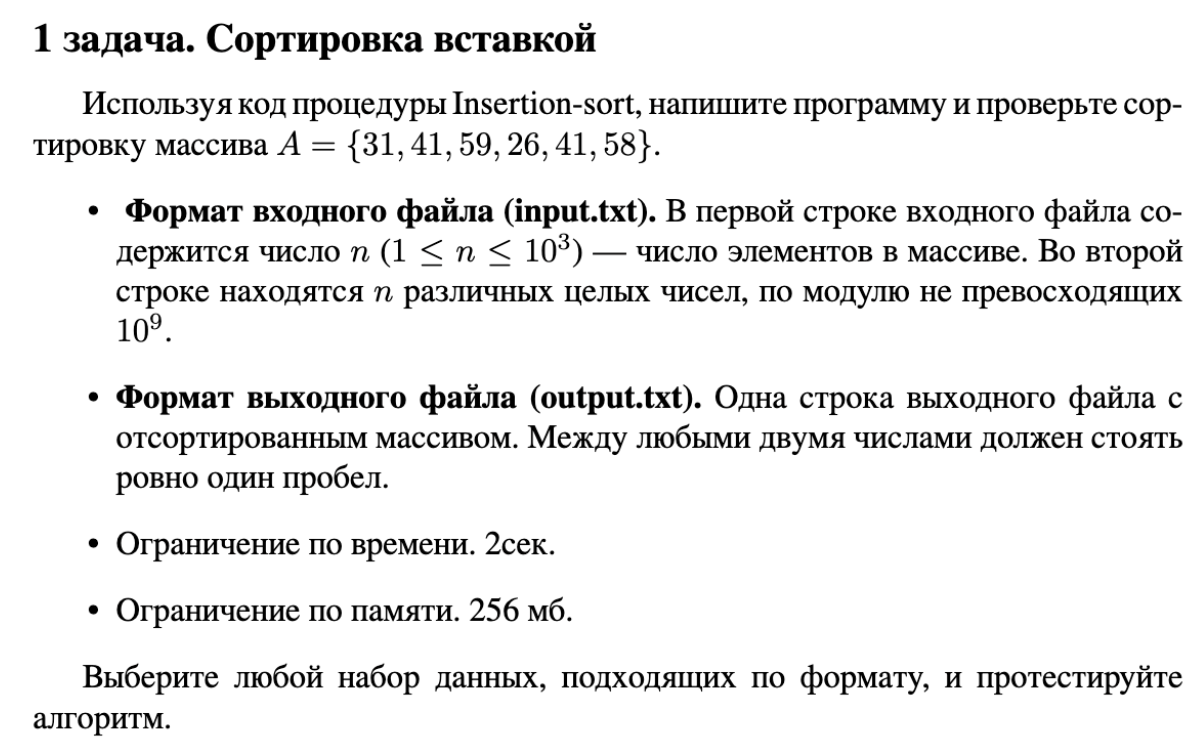
Вывод

Использовала файл utils.py для исключения повторения строчек кода:

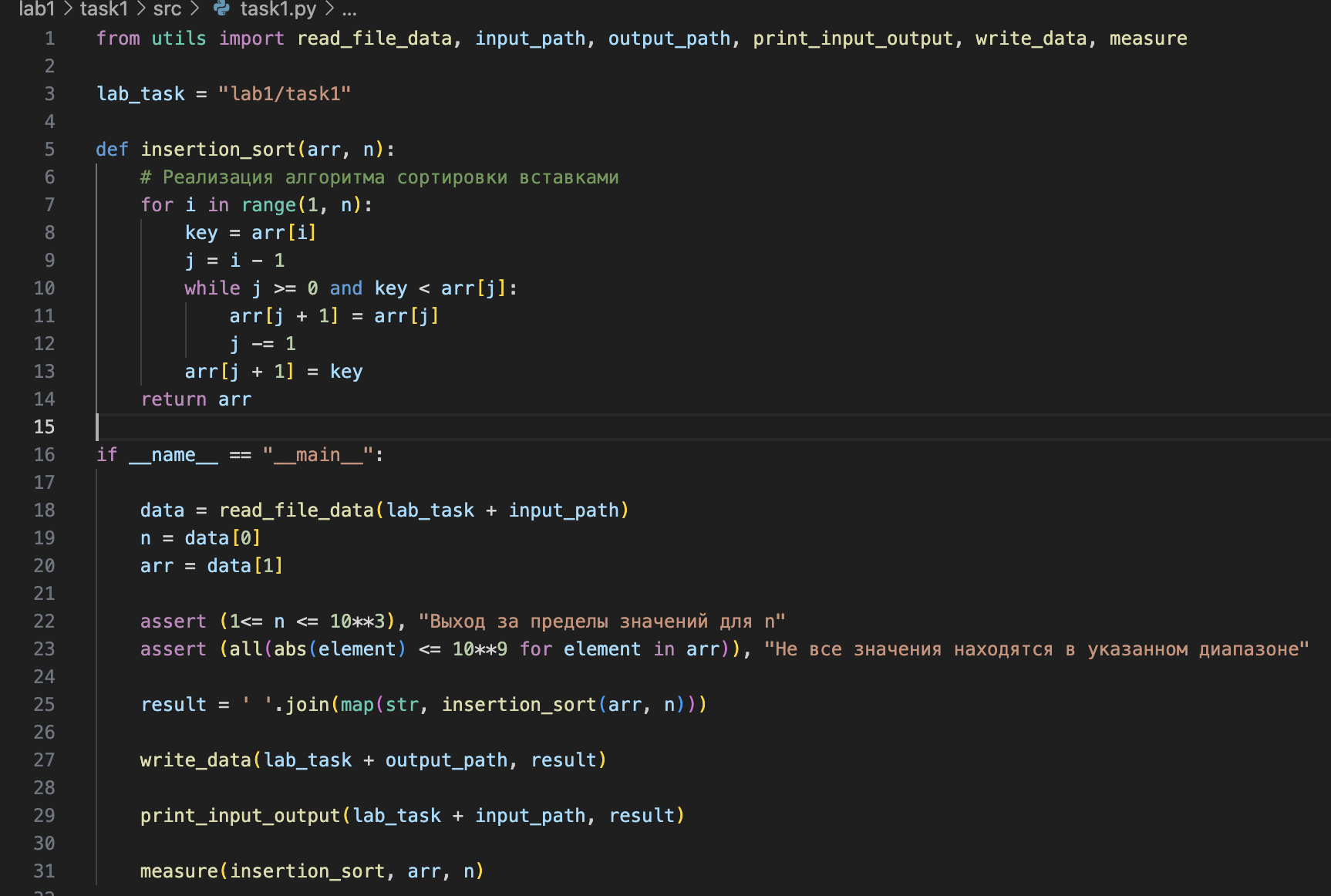




Задание №1.Сортировка вставкой.

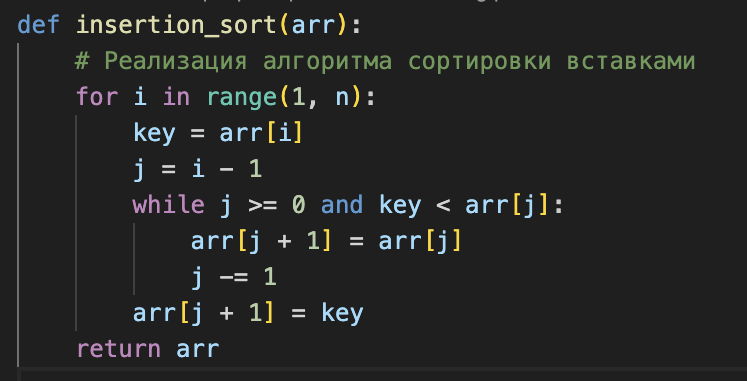


Решение задачи №1:



Этот код реализует алгоритм сортировки вставками (insertion\_sort) для сортировки массива целых чисел, который читается из файла input.txt. Результат сортировки записывается в файл output.txt. Кроме того, код измеряет время работы алгоритма и потребление памяти.

**Определение функции сортировки:**



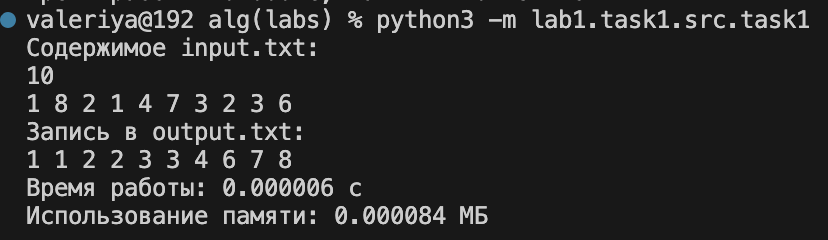
* Функция insertion\_sort принимает массив arr и сортирует его в порядке возрастания.

Алгоритм работает следующим образом:

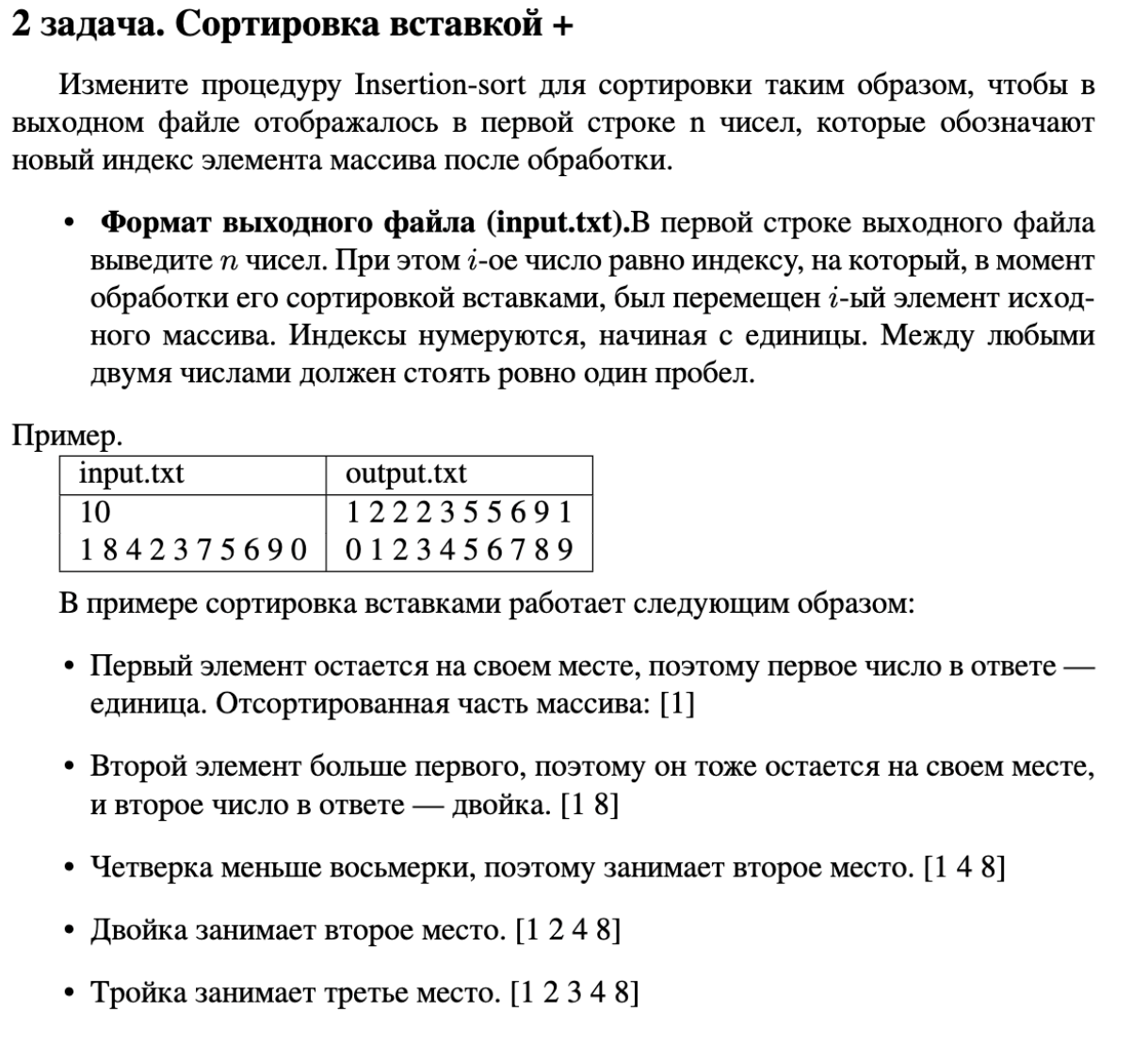
* Цикл начинается со второго элемента массива (i = 1), так как первый элемент уже отсортирован (он единственный элемент в массиве).
* Цикл продолжается до n-1 элемента, где n - длина массива.
* На каждой итерации цикла выбирается текущий элемент массива (arr[i]) как ключевой элемент (key).
* Внутри цикла while мы начинаем с предыдущего элемента (j = i - 1) и движемся влево, пока не найдем правильную позицию для ключевого элемента.
* Если текущий элемент (arr[j]) больше ключевого элемента (key), мы сдвигаем его вправо, копируя его значение в следующую ячейку (arr[j + 1] = arr[j]).
* Мы продолжаем это процесс, пока не найдем элемент, который меньше или равен ключевому элементу, или пока не достигнем начала массива (j >= 0).
* Как только мы нашли правильную позицию для ключевого элемента, мы вставляем его в массив, присваивая его значение ячейке arr[j + 1].
* После того, как цикл завершен, функция возвращает отсортированный массив.

В целом, алгоритм сортировки вставками работает последовательно, просматривая элементы массива и вставляя каждый элемент в правильную позицию в уже отсортированной части массива.

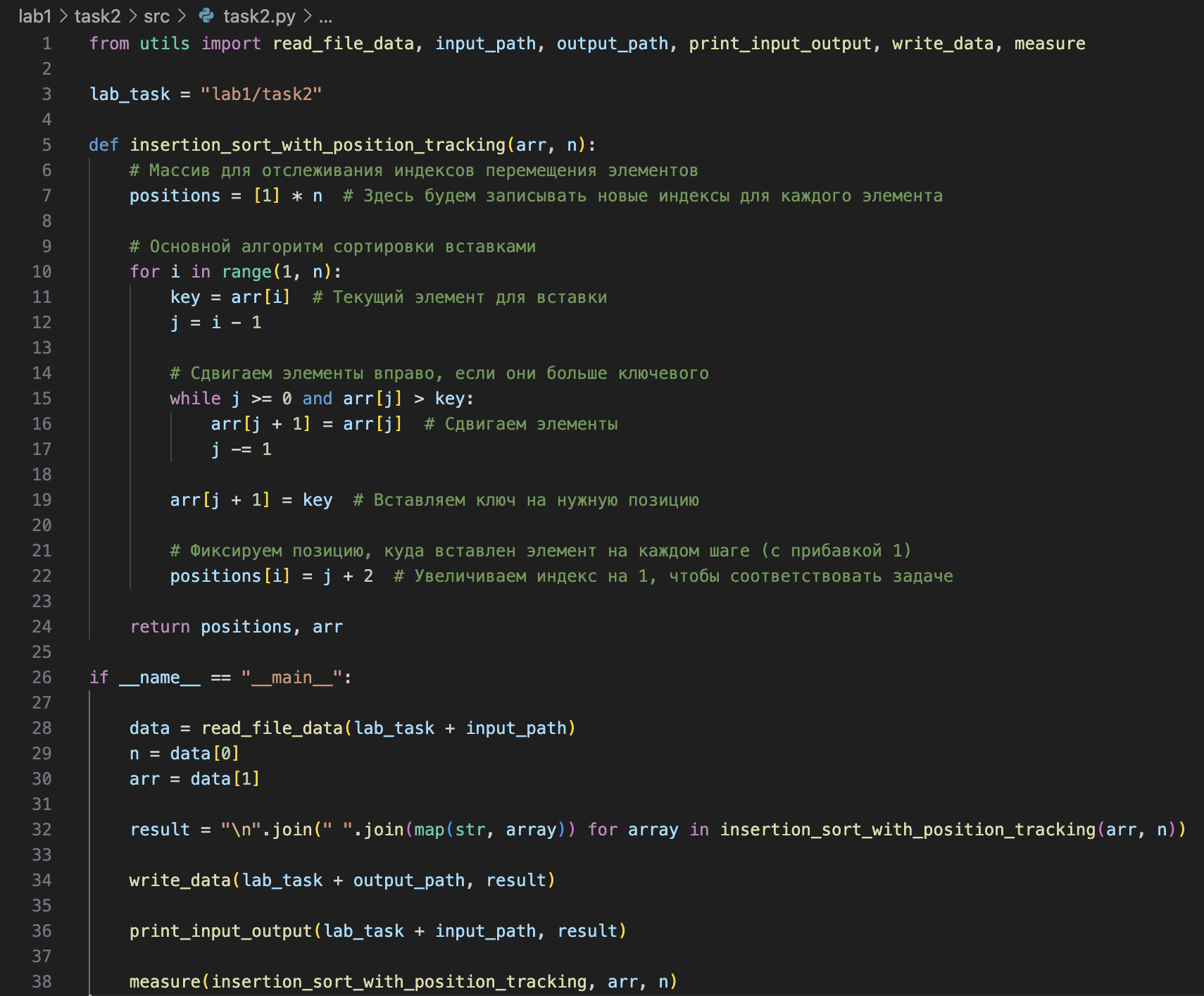
Считываем данные файла с помощью utils и записываем результат:



Задача №2: Сортировка вставкой +



Решение задачи №2:



**Определение функции сортировки**

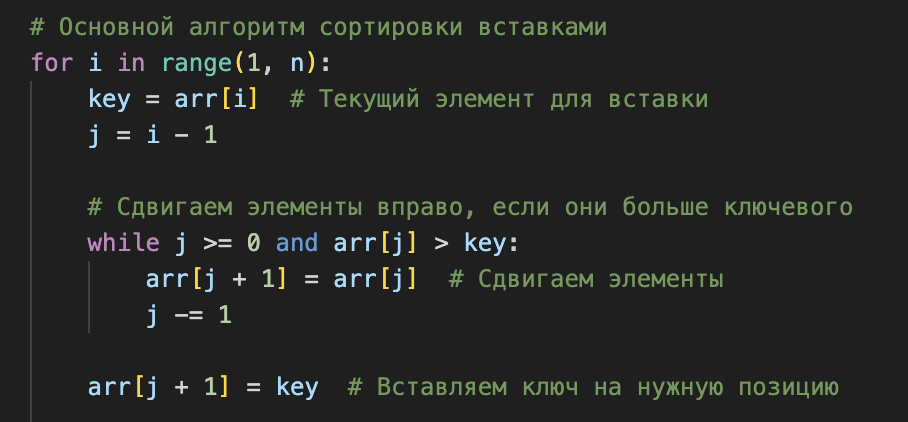
def insertion\_sort\_with\_position\_tracking(arr):- определяется функция insertion\_sort\_with\_position\_tracking, которая принимает массив arr в качестве аргумента.

**Инициализация массива для отслеживания позиций:**

positions = [1] \* n

* Создается массив positions длиной n, где n - количество элементов в массиве arr.
* Массив positions используется для отслеживания новых индексов для каждого элемента после сортировки.

**Основной алгоритм сортировки вставками:**



Этот код реализует алгоритм сортировки вставками.

* На каждой итерации цикла for выбирается текущий элемент key из массива arr.
* Внутри цикла while элементы массива arr сдвигаются вправо, если они больше ключевого элемента key.
* После сдвига элементов ключевой элемент key вставляется в правильную позицию в массиве arr.

**Отслеживание позиций:**

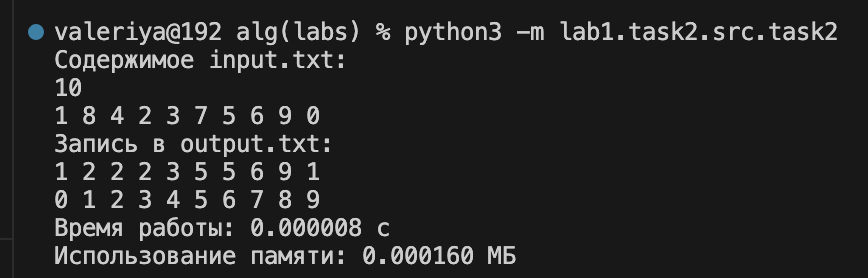
positions[i] = j + 2

* После вставки ключевого элемента в правильную позицию его новый индекс записывается в массив positions.
* Индекс увеличивается на 1, чтобы соответствовать задаче.

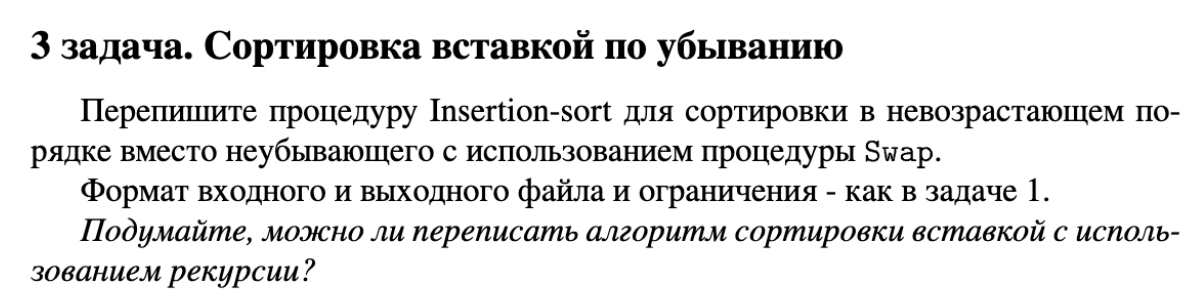
return positions, arr - Функция возвращает два массива: positions и arr.

* Массив positions содержит новые индексы для каждого элемента после сортировки.
* Массив arr содержит отсортированный массив.

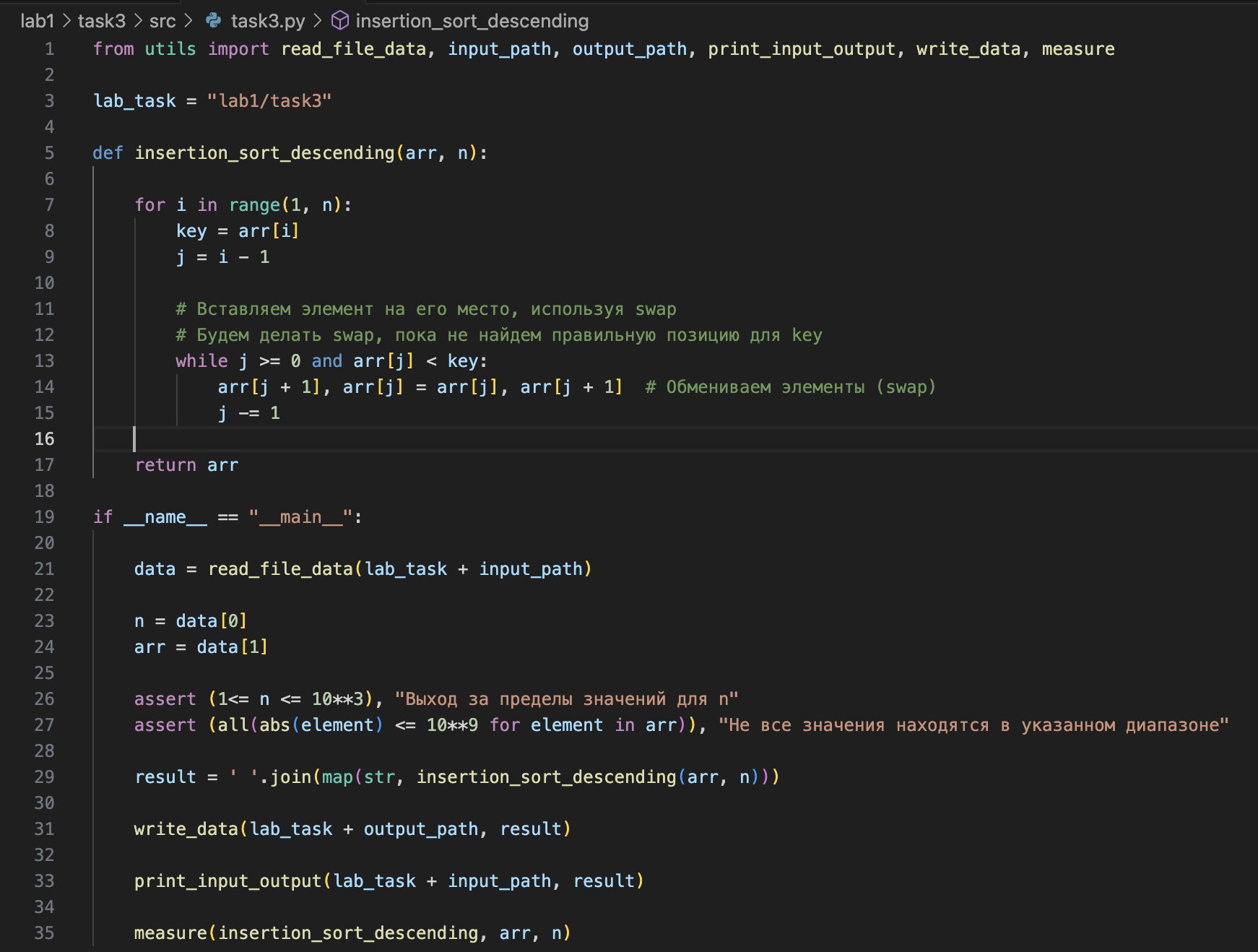
Считываем данные файла с помощью utils и записываем результат:



Задача №3:Сортировка вставкой по убыванию



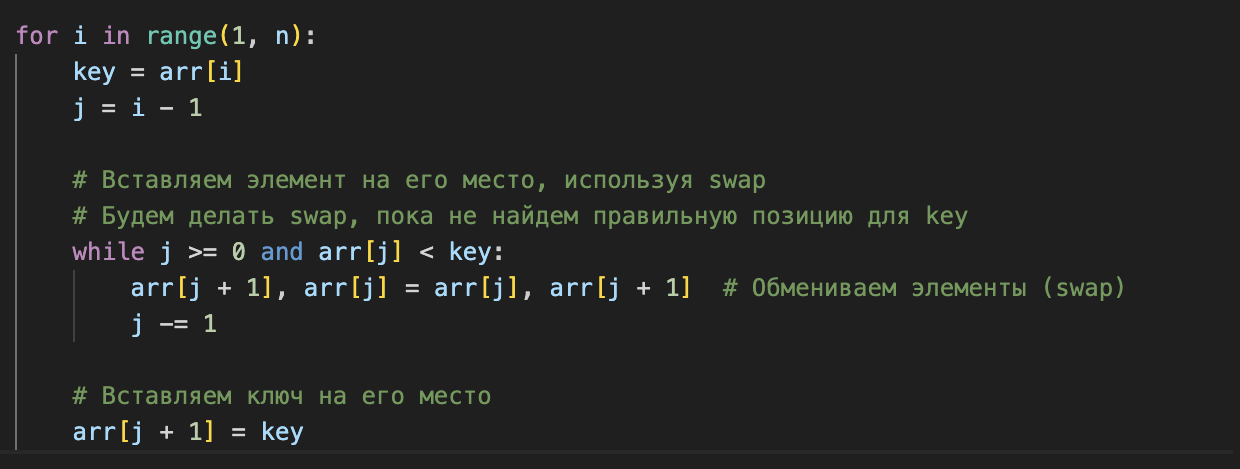
Решение задачи №3:

Этот код реализует алгоритм сортировки вставками в порядке убывания

**Определение функции сортировки:**

def insertion\_sort\_descending(arr): - Определяется функция insertion\_sort\_descending, которая принимает массив arr в качестве аргумента.

**Основной алгоритм сортировки вставками:**

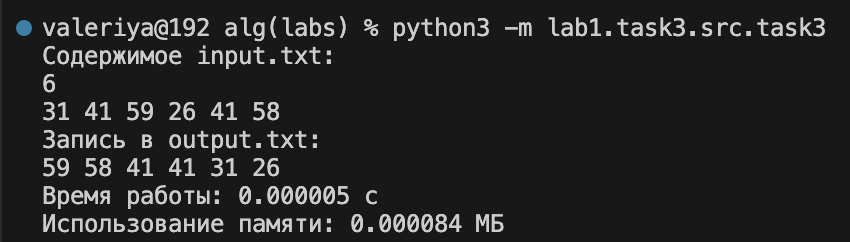
****

* Этот код реализует алгоритм сортировки вставками в порядке убывания.
* На каждой итерации цикла for выбирается текущий элемент key из массива arr.
* Внутри цикла while элементы массива arr обмениваются местами (swap), если они меньше ключевого элемента key.
* После обмена элементами ключевой элемент key вставляется в правильную позицию в массиве arr.

**Возвращение результата:**

return arr - Функция возвращает отсортированный массив arr.

Считываем данные файла с помощью utils и записываем результат:



Можно ли переписать алгоритм сортировки вставкой с использованием рекурсии?

Базовый случай: Если количество элементов n меньше или равно 1, массив уже отсортирован, и функция просто возвращается.

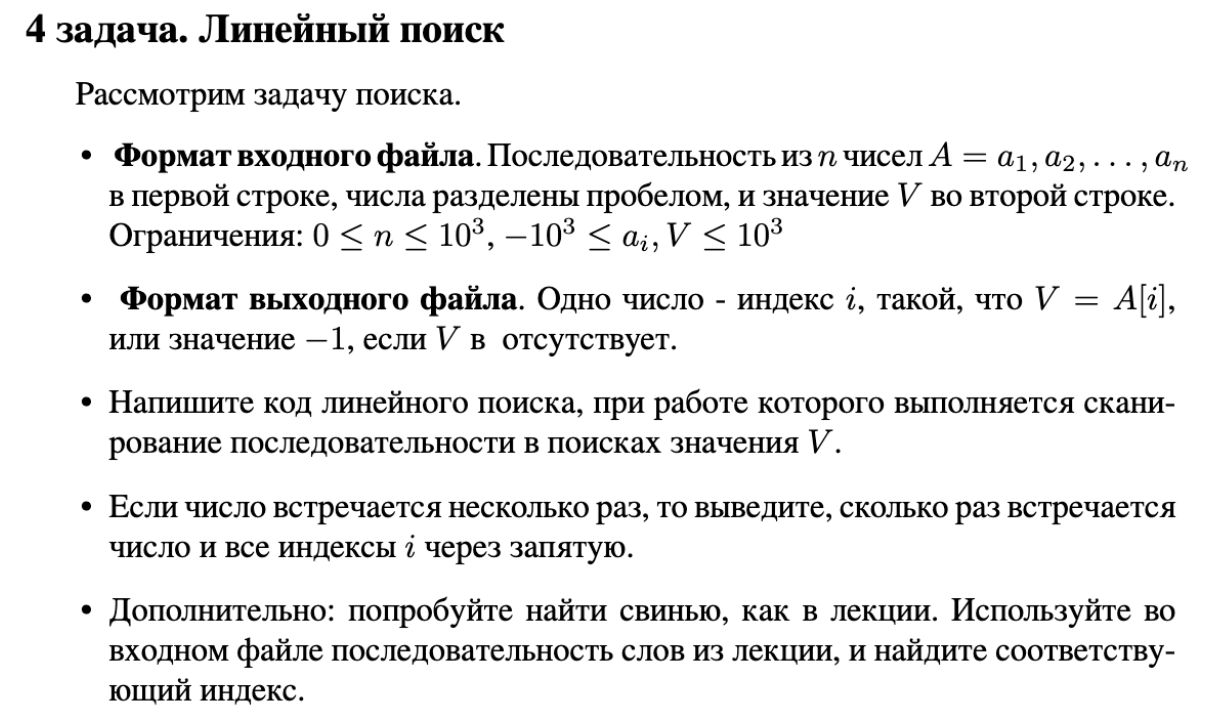
Рекурсивный вызов: Функция вызывает саму себя для сортировки первых n-1 элементов.

Вставка элемента: После сортировки первых n-1 элементов последний элемент вставляется на нужную позицию в отсортированной части массива:

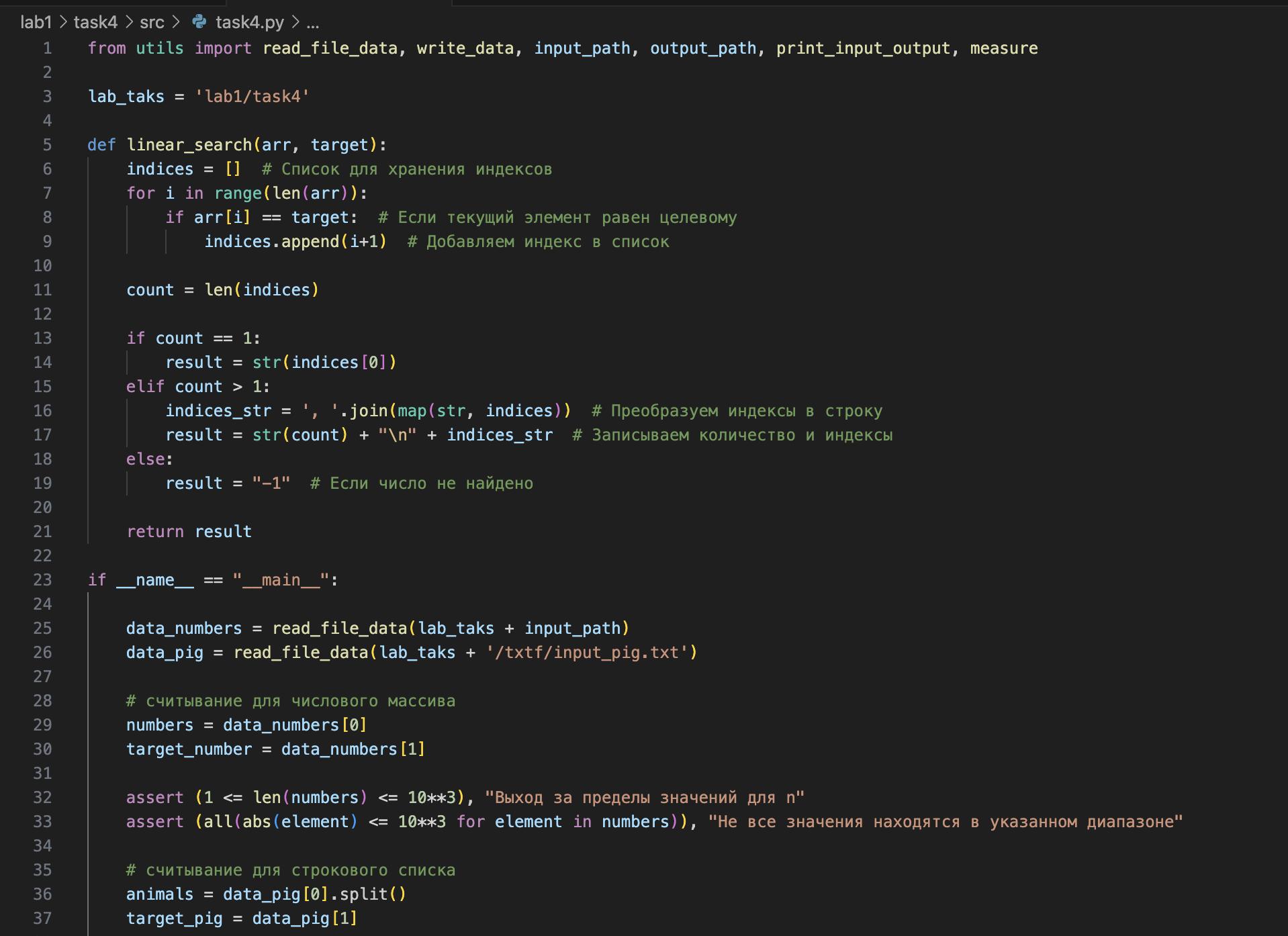
Мы сравниваем последний элемент с предыдущими элементами, и если они меньше, то сдвигаем их вправо.

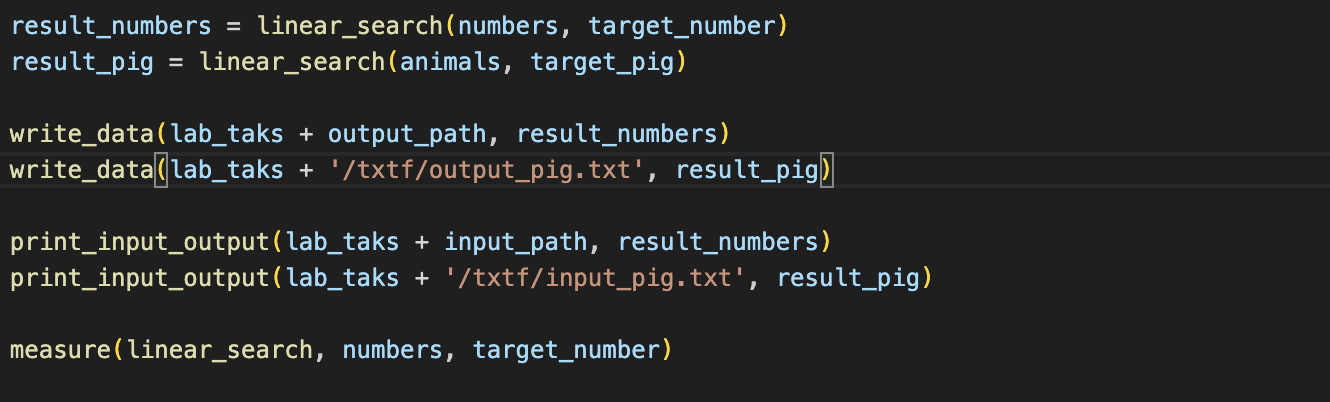
Затем вставляем последний элемент в правильную позицию.

Задача №4:Линейный поиск



Решение задачи №4:





Этот код представляет собой функцию linear\_search, которая принимает два параметра:

* arr: Массив целых чисел.
* target: Целевое число, которое нужно найти в массиве.

Функция возвращает строку, содержащую информацию о найденных индексах целевого числа в массиве.

**Шаг 1: Инициализация списка для хранения индексов**

Первой строкой кода является инициализация пустого списка indices для хранения индексов целевого числа в массиве.

**Шаг 2: Обход массива и поиск целевого числа**

В цикле for функция обходит каждый элемент массива arr. Если текущий элемент arr[i] равен целевому числу target, функция добавляет индекс i+1 в список indices. Индекс увеличивается на 1, поскольку в задаче используется 1-индексация.

**Шаг 3: Подсчет количества найденных индексов**

После обхода массива функция подсчитывает количество найденных индексов с помощью функции len.

**Шаг 4: Формирование результата**

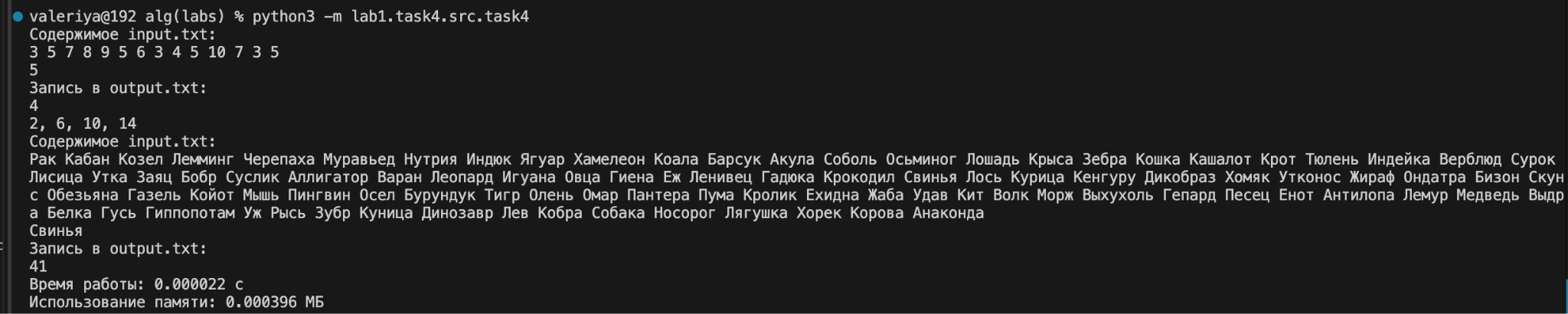
В зависимости от количества найденных индексов функция формирует результат:

* Если найден только один индекс, функция возвращает строку с этим индексом.
* Если найдено несколько индексов, функция возвращает строку, содержащую количество найденных индексов и сами индексы, разделенные запятыми.
* Если целевое число не найдено, функция возвращает строку "-1".

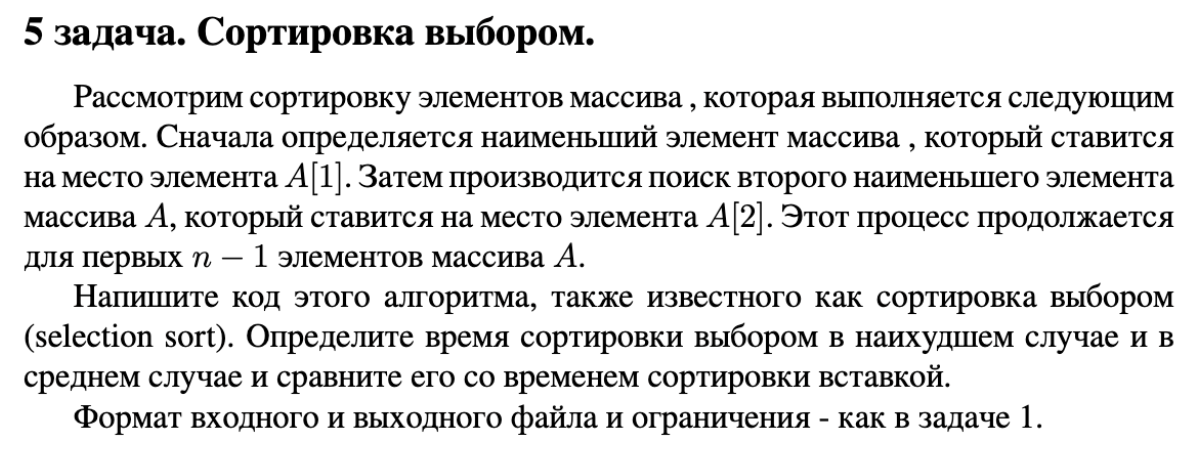
**Шаг 5: Возвращение результата**

После формирования результата функция возвращает его.

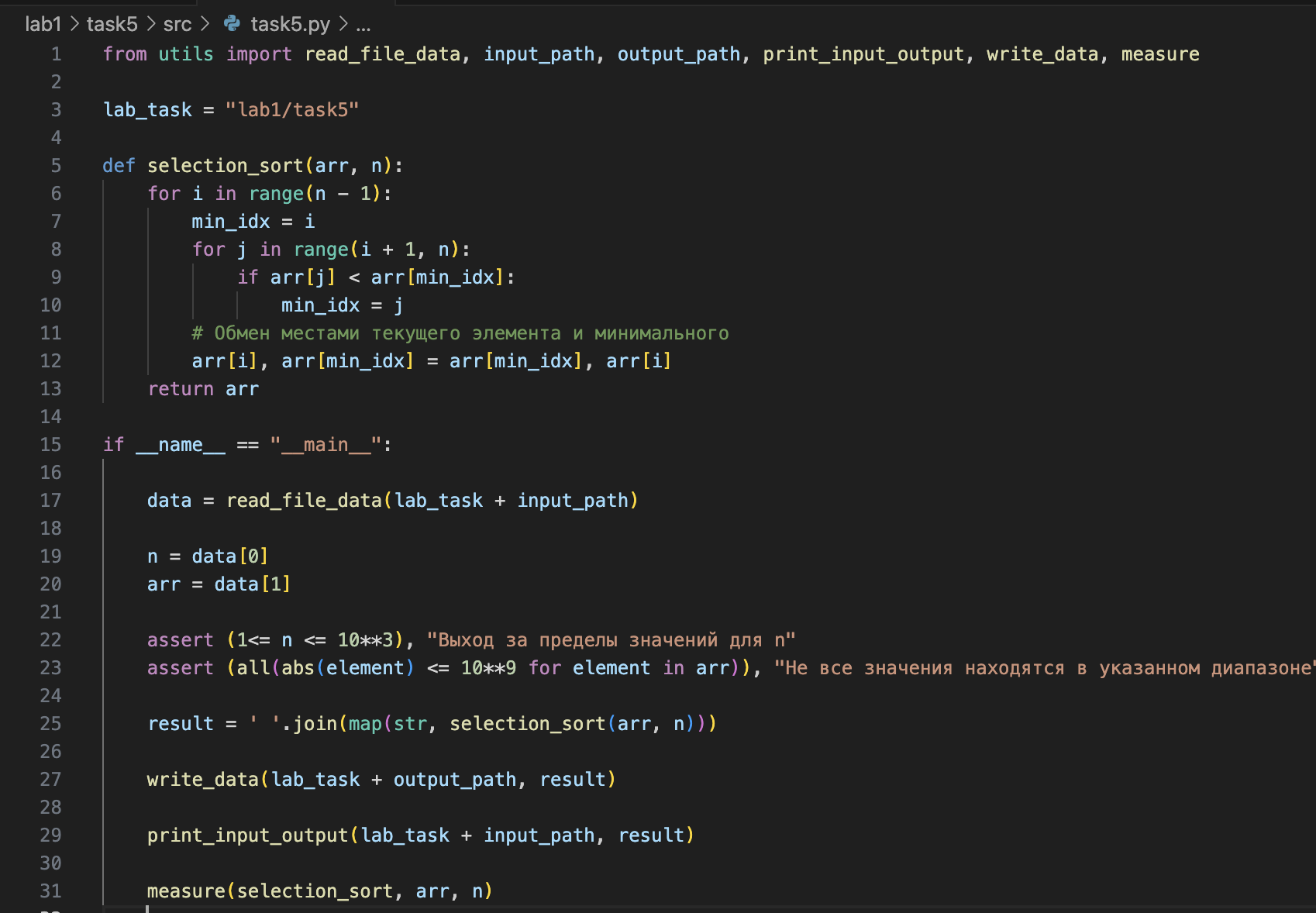
Считываем данные файла с помощью utils и записываем результат:



Задача №5:Сортировка выбором

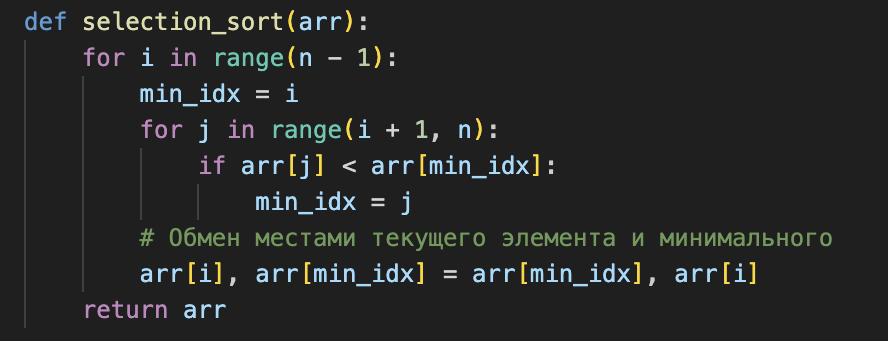


Решение задачи №5:



Данный код на Python реализует сортировку выбором (selection sort) для массива целых чисел, который читается из файла input.txt. После сортировки результат записывается в файл output.txt. Кроме того, код измеряет время выполнения программы и количество используемой памяти.

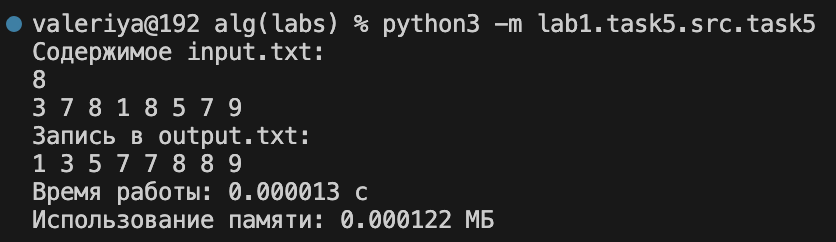
**Функция сортировки выбором:**



Эта функция принимает на вход массив arr и выполняет сортировку выбором. Алгоритм работает следующим образом:

1. Внешний цикл for проходит по массиву от первого элемента до предпоследнего (n - 1).
2. Внутренний цикл for проходит по массиву от текущего элемента (i + 1) до последнего элемента (n).
3. Если элемент arr[j] меньше текущего минимального элемента arr[min\_idx], то индекс минимального элемента обновляется (min\_idx = j).
4. После внутреннего цикла текущий элемент и минимальный элемент меняются местами (arr[i], arr[min\_idx] = arr[min\_idx], arr[i]).
5. Функция возвращает отсортированный массив.

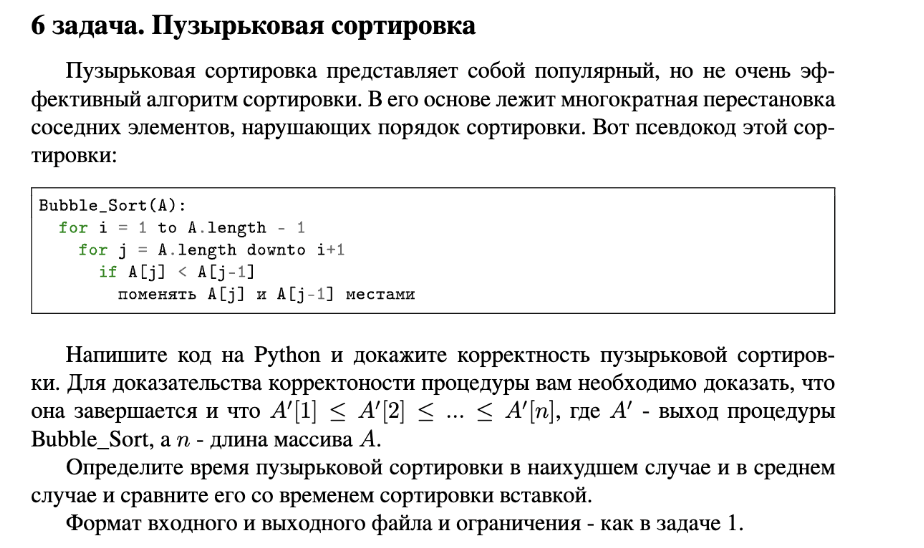
Считываем данные файла с помощью utils и записываем результат:

****

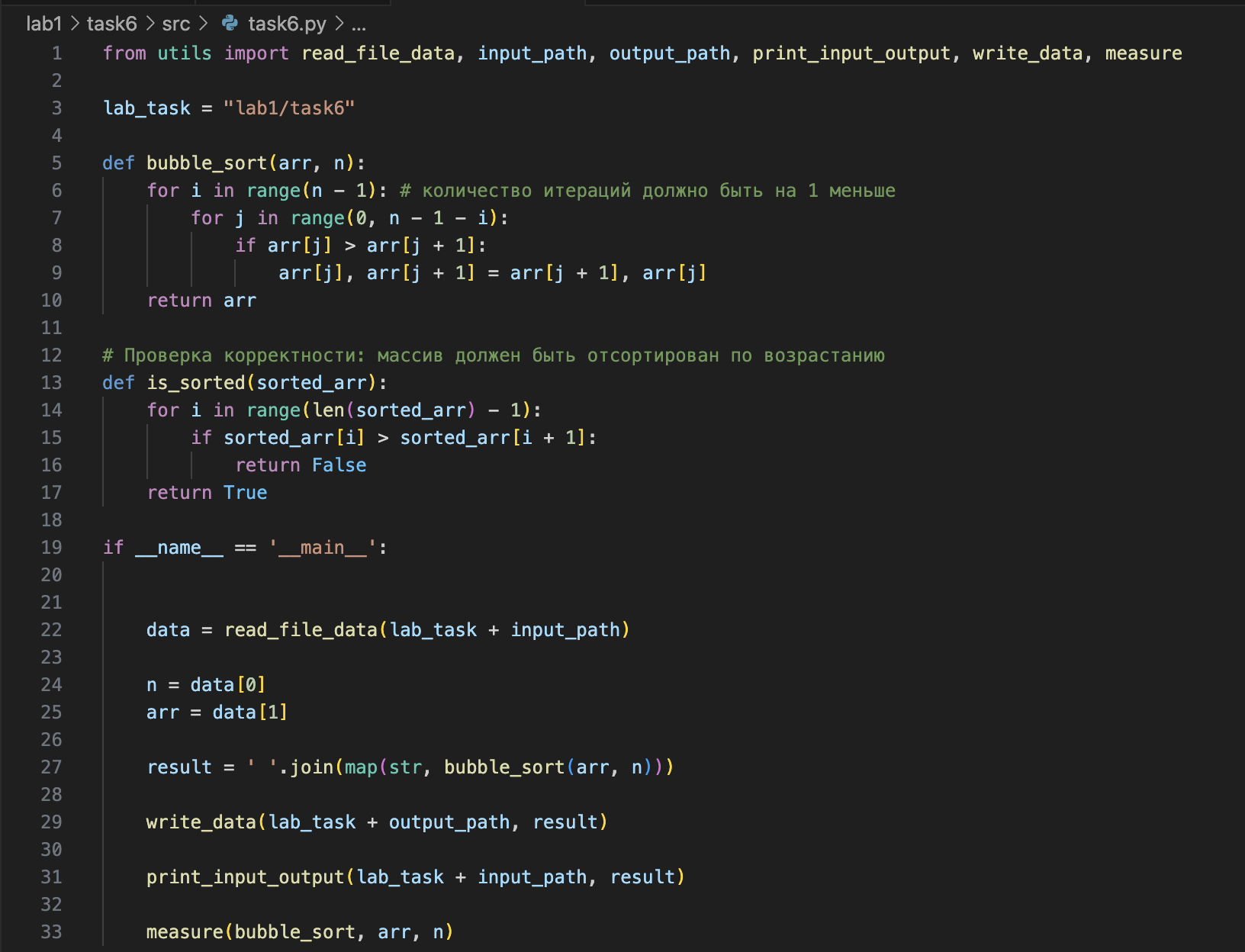
**Сравнение времени сортировкой вставкой и сортировкой выбором:**

O(n^2) - квадратичная функция от n, где n - размер массива (для наихудшего случая сортировки вставкой и сортировки выбором).В наилучшем случае O(n) - для сортировки вставкой, сортировка выбором - не меняется, остается O(n^2).

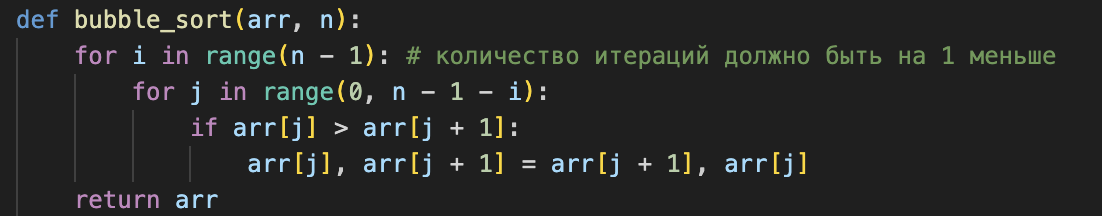
Задача №6:Пузырьковая сортировка



Решение задачи №6:



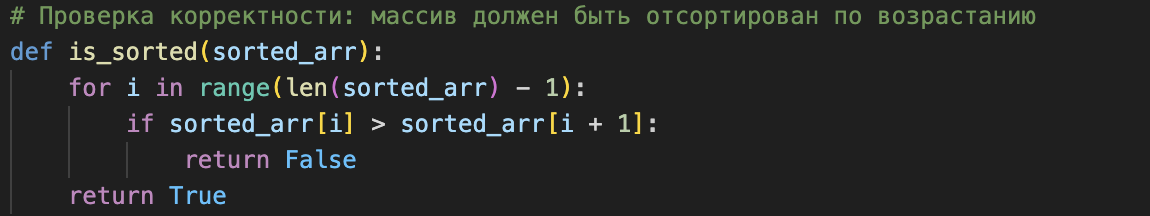
Этот код представляет собой реализацию алгоритма сортировки пузырьком (bubble sort) и функцию проверки корректности отсортированного массива.

****

Эта функция принимает на вход массив arr и его длину n. Функция сортирует массив по возрастанию, используя алгоритм пузырьком.

Алгоритм работает следующим образом:

1. Внешний цикл for i in range(n - 1): выполняется n-1 раз, где n - длина массива.
2. Внутренний цикл for j in range(0, n - 1 - i): выполняется n-1-i раз, где i - текущая итерация внешнего цикла.
3. Внутри внутреннего цикла проверяется, если текущий элемент arr[j] больше следующего элемента arr[j + 1]. Если это так, то элементы меняются местами.
4. После окончания внутреннего цикла, наибольший элемент в массиве "всплывает" в конец массива.
5. Внешний цикл повторяется, пока не будут отсортированы все элементы массива.

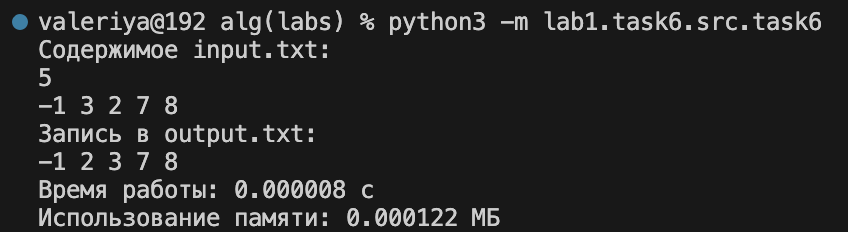
****

Эта функция принимает на вход отсортированный массив sorted\_arr. Функция проверяет, если массив действительно отсортирован по возрастанию.

Алгоритм работает следующим образом:

1. Цикл for i in range(len(sorted\_arr) - 1): выполняется len(sorted\_arr) - 1 раз, где len(sorted\_arr) - длина массива.
2. Внутри цикла проверяется, если текущий элемент sorted\_arr[i] больше следующего элемента sorted\_arr[i + 1]. Если это так, то функция возвращает False, указывая на то, что массив не отсортирован.
3. Если цикл завершается без обнаружения несортированных элементов, то функция возвращает True, указывая на то, что массив отсортирован.

Считываем данные файла с помощью utils и записываем результат:

****

Вывод:

Цель лабораторной работы достигнута. А именно, я познакомилась и отработала различные виды сортировки : сортировку вставкой, сортировку вставкой +, сортировку вставкой по убыванию, линейный поиск, сортировку выбором. При выполнении работы столкнулась с трудностями, но к счастью, успешно их решила.