Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский Томский политехнический Университет»

****

самостоятельная работа № 4

Файловые операции PYthon

курс:

# «Программирование и алгоритмизация на Python»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Исполнитель:** |  | | | | |
| студент группы | 0В21ДзебанА.А |  | ФИО студента |  | 11.12.24 |
|  |  |  |  |  |  |
| **Руководитель:** |  | | | | |
| преподаватель |  |  | Полищук В.Ю. |  |  |
|  |  |  |  |  |  |

Томск – 2024

**Цель работы:**

Обрести навыки работы с файлами.

**Задание:**

На основе самостоятельной работы №3 реализовать программу, так, чтобы в файл формата \*.xlsx, по столбцам, записывались все отсортированные массивы с указанием, в начальных ячейках, времени их сортировки и названия алгоритма сортировки, а в самый первый столбец файла записать неотсортированный массив. При реализации программы все алгоритмы сортировки должны быть представлены в виде функций, а также реализовать функцию записи в файл формата \*.xlsx.

**Ход работы:**

1. Реализация алгоритмов сортировки и создания аналогична 3 лабораторной работе.

# %%

import random

import time

import pandas as pd

# %%

def generate\_array(length, min\_value, max\_value):

    input\_array = []

    for \_ in range(length):

        input\_array.append(random.randint(min\_value, max\_value))

    return input\_array

# %%

def selection\_sort(arr):

    start\_time = time.time()

    n = len(arr)

    for i in range(n):

        min\_index = i

        for j in range(i + 1, n):

            if arr[j] < arr[min\_index]:

                min\_index = j

        arr[i], arr[min\_index] = arr[min\_index], arr[i]

    end\_time = time.time()

    return arr, end\_time - start\_time

# %%

def insertion\_sort(arr):

    start\_time = time.time()

    for i in range(1, len(arr)):

        key = arr[i]

        j = i - 1

        while j >= 0 and arr[j] > key:

            arr[j + 1] = arr[j]

            j -= 1

        arr[j + 1] = key

    end\_time = time.time()

    return arr, end\_time - start\_time

# %%

def bubble\_sort(arr):

    start\_time = time.time()

    n = len(arr)

    for i in range(n):

        for j in range(0, n - i - 1):

            if arr[j] > arr[j + 1]:

                arr[j], arr[j + 1] = arr[j + 1], arr[j]

    end\_time = time.time()

    return arr, end\_time - start\_time

# %%

def shell\_sort(arr):

    start\_time = time.time()

    n = len(arr)

    gap = n // 2

    while gap > 0:

        for i in range(gap, n):

            temp = arr[i]

            j = i

            while j >= gap and arr[j - gap] > temp:

                arr[j] = arr[j - gap]

                j -= gap

            arr[j] = temp

        gap //= 2

    end\_time = time.time()

    return arr, end\_time - start\_time

# %%

def quick\_sort(arr):

    start\_time = time.time()

    def \_quick\_sort(items):

        if len(items) <= 1:

            return items

        pivot = items[len(items) // 2]

        left = [x for x in items if x < pivot]

        middle = [x for x in items if x == pivot]

        right = [x for x in items if x > pivot]

        return \_quick\_sort(left) + middle + \_quick\_sort(right)

    sorted\_arr = \_quick\_sort(arr)

    end\_time = time.time()

    return sorted\_arr, end\_time - start\_time

# %%

length = int(input("Введите длину массива: \n"))

min\_size = int(input("Введите минимально вомзожное значение в массиве: \n"))

max\_size = int(input("Ввведите максимально возможное значение в массиве: \n"))

# %%

inp\_arr = generate\_array(length=length, min\_value=min\_size, max\_value=max\_size)

# %%

algorithms = {

    "Сортировка выбором": selection\_sort,

    "Сортировка вставками": insertion\_sort,

    "Сортировка пузырьковая": bubble\_sort,

    "Сортировка Шелла": shell\_sort,

    "Сортировка быстрая": quick\_sort

}

1. Запишем результаты сортировки в отдельный словарь sorted\_data={key: (sorted\_list, time)}

Для словаря будем создавать значения с ключом, соответствующему названию и времени выполнения, значения для ключа – массив.

sorted\_data = {}

for key, algorithm in algorithms.items():

    output\_array, time\_taken = algorithm(inp\_arr.copy())

    sorted\_data[key] = (output\_array, time\_taken)

    print(f"Выход для алгоритма \"{key}\":", output\_array,f". Время выполнения: {time\_taken}")

1. Определим функцию экспорта в виде файла xlsx (сначала конвертируем данные в pd.DataFrame и переводим в excel)

def export\_as\_xlsx(file\_name, input\_list, sorted\_data):

    initial\_data = {"неотсорт" : input\_list}

    for key, (sorted\_list, time) in sorted\_data.items():

        initial\_data[f"{key.rpartition(' ')[2]}\_{round(time,4)}"] = sorted\_list

    df=pd.DataFrame(initial\_data)

    df.to\_excel(file\_name, index=False)

1. Вводим имя файла и экспортируем:

file\_name = "sorting.xlsx"

export\_as\_xlsx(file\_name=file\_name, input\_list=inp\_arr, sorted\_data=sorted\_data)

print(f"Результаты сохранены в файл {file\_name}")

Файл выглядит следующим образом:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **неотсорт** | **выбором\_0.0246** | **вставками\_0.0201** | **Пузырьковая\_0.0351** | **Шелла\_0.001** | **быстрая\_0.0** |
| 69 | 1 | 1 | 1 | 1 | 1 |
| 11 | 1 | 1 | 1 | 1 | 1 |
| 11 | 1 | 1 | 1 | 1 | 1 |
| 97 | 1 | 1 | 1 | 1 | 1 |
| 29 | 1 | 1 | 1 | 1 | 1 |
| 40 | 1 | 1 | 1 | 1 | 1 |
| 86 | 1 | 1 | 1 | 1 | 1 |
| 6 | 1 | 1 | 1 | 1 | 1 |
| 66 | 1 | 1 | 1 | 1 | 1 |
| 51 | 1 | 1 | 1 | 1 | 1 |
| 78 | 1 | 1 | 1 | 1 | 1 |
| 58 | 1 | 1 | 1 | 1 | 1 |
| 73 | 2 | 2 | 2 | 2 | 2 |
| 56 | 2 | 2 | 2 | 2 | 2 |
| 11 | 2 | 2 | 2 | 2 | 2 |
| 98 | 2 | 2 | 2 | 2 | 2 |
| 74 | 2 | 2 | 2 | 2 | 2 |
| 85 | 2 | 2 | 2 | 2 | 2 |
| 5 | 2 | 2 | 2 | 2 | 2 |
| 21 | 2 | 2 | 2 | 2 | 2 |
| 68 | 2 | 2 | 2 | 2 | 2 |
| 25 | 2 | 2 | 2 | 2 | 2 |
| 3 | 2 | 2 | 2 | 2 | 2 |
| 17 | 2 | 2 | 2 | 2 | 2 |
| 85 | 2 | 2 | 2 | 2 | 2 |
| 21 | 2 | 2 | 2 | 2 | 2 |

**Вывод:** были изучены способы создания файлов в python.