ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«УФИМСКИЙ УНИВЕРСИТЕТ НАУКИ И ТЕХНОЛОГИЙ»

Институт информатики, математики и робототехники
Кафедра математического и компьютерного моделирования

Лабораторная работа №4: Работа с форматом JSON в Python

ОБУЧАЮЩЕГОСЯ
4 курса группы ПИ-4ИВТ221Б

Санникова Михаила Александровича

Уровень высшего образования:	высшее образование – бакалавриат
Направление подготовки (специальность)	09.03.03 “Прикладная информатика”
Направленность (профиль) программы	Информационные и вычислительные технологии
Дата выполнения	13.10.2025

УФА – 2025

Постановка задачи:

Создать файл students.json

Написать функцию для чтения файла

Добавить в файл students.json:
• Нового студента id: 3, name: «Сидоров Алексей», age: 21, grades: [4, 4, 3, 5], is_active: true
• Обновите статус студента с id=2 на неактивный (is_active: false)
• Добавьте поле «department»: «Информационные системы» на верхний уровень документа

Написать функцию, которая проверяет:
• Наличие обязательных полей у каждого студента (id, name, age, grades, is_active)
• Корректность типов данных (id и age — числа, name — строка, grades — массив чисел)
• Уникальность идентификаторов студентов

Написать функцию, которая возвращает только активных студентов со средним баллом выше заданного значения

Практическая часть:

Листинг кода с комментариями

```
import json
import os
from functools import wraps


def main():
    filename = 'students.json'

    # Задание 1: Создание JSON-файла
    create_initial_json(filename)

    # Задание 2: Чтение JSON-файла
    print("=" * 60)
    print("ЧТЕНИЕ JSON-ФАЙЛА")
    print("=" * 60)
    read_json_file(filename)

    # Задание 3: Модификация JSON-файла
    print("\n" + "=" * 60)
    print("МОДИФИКАЦИЯ JSON-ФАЙЛА")
    print("=" * 60)
    modify_data = modify_json_file(filename)

    # Задание 4: Валидация данных
    print("\n" + "=" * 60)
    print("ВАЛИДАЦИЯ ДАННЫХ")
    print("=" * 60)
    if is_data_valid(modify_data):
        print("✓ Все данные прошли валидацию успешно!")
    else:
        print("✗ Данные содержат ошибки!")

    # Чтение обновленного файла
    print("\n" + "=" * 60)
    print("ОБНОВЛЕННЫЕ ДАННЫЕ")
    print("=" * 60)
    read_json_file(filename)

    # Задание 5: Фильтрация данных
    print("\n" + "=" * 60)
    print("ФИЛЬТРАЦИЯ ДАННЫХ")
    print("=" * 60)
    filtered_students = filter_students(modify_data, min_avg_grade=4.0)
    print(f"Найдено {len(filtered_students)} активных студентов со средним баллом выше 4.0:")
    for student in filtered_students:
        avg_grade = sum(student['grades']) / len(student['grades'])
        print(f"  - {student['name']} (средний балл: {avg_grade:.2f})")


def create_initial_json(filename):
    """Задание 1: Создание начального JSON-файла"""
    data = {
        "students": [
            {
                "id": 1,
```

```
            "name": "Иванов Иван",
            "age": 20,
            "grades": [4, 5, 4, 5],
            "is_active": True
        },
        {
            "id": 2,
            "name": "Петрова Мария",
            "age": 19,
            "grades": [5, 5, 5, 4],
            "is_active": True
        }
    ],
    "group": "ИС-21",
    "year": 2023
}

with open(filename, 'w', encoding='utf-8') as file:
    json.dump(data, file, ensure_ascii=False, indent=2)

print(f"✓ Создан файл {filename}")


def check_file_exists(func):
    @wraps(func)
    def wrapper(file_path):
        if not os.path.exists(file_path):
            print(f"Не существует файла с заданным путем: {file_path}")
            return None
        return func(file_path)

    return wrapper


@check_file_exists
def read_json_file(filename):
    """Задание 2: Чтение JSON-файла"""
    try:
        with open(filename, 'r', encoding='utf-8') as file:
            data = json.load(file)

        print(f"Группа: {data['group']}")
        print(f"Год: {data['year']}\n")

        total_avg = 0
        student_count = len(data['students'])

        print("СТУДЕНТЫ:")
        print("-" * 50)
        for student in data['students']:
            avg_grade = sum(student['grades']) / len(student['grades'])
            total_avg += avg_grade

            status = "активен" if student['is_active'] else "неактивен"
            print(f"{student['name']}, {student['age']} лет")
            print(f"  Средний балл: {avg_grade:.2f}, Статус: {status}")
            print(f"  Оценки: {student['grades']}")
```

```
            print("-" * 30)

        if student_count > 0:
            print(f"ОБЩИЙ СРЕДНИЙ БАЛЛ ГРУППЫ: {total_avg / student_count:.2f}")

        return data

    except Exception as e:
        print(f"Ошибка при чтении файла: {e}")
        return None


def modify_json_file(filename):
    """Задание 3: Модификация JSON-файла"""
    try:
        # Чтение существующих данных
        with open(filename, 'r', encoding='utf-8') as file:
            data = json.load(file)

        # Добавление нового студента
        new_student = {
            "id": 3,
            "name": "Сидоров Алексей",
            "age": 21,
            "grades": [4, 4, 3, 5],
            "is_active": True
        }
        data['students'].append(new_student)
        print("✓ Добавлен новый студент: Сидоров Алексей")

        # Обновление статуса студента с id=2
        for student in data['students']:
            if student['id'] == 2:
                student['is_active'] = False
                print("✓ Обновлен статус студента Петрова Мария на 'неактивен'")

        # Добавление поля department
        data['department'] = "Информационные системы"
        print("✓ Добавлено поле 'department': Информационные системы")

        # Запись обновленных данных
        with open(filename, 'w', encoding='utf-8') as file:
            json.dump(data, file, ensure_ascii=False, indent=2)

        print("✓ Файл успешно обновлен!")
        return data

    except Exception as e:
        print(f"Ошибка при модификации файла: {e}")
        return None


def is_data_valid(data):
    """Задание 4: Валидация данных"""
    errors = []
```

```
    # Проверка обязательных полей верхнего уровня
    required_top_fields = ['students', 'group', 'year']
    for field in required_top_fields:
        if field not in data:
            errors.append(f"Отсутствует обязательное поле: {field}")

    # Проверка студентов
    if 'students' in data:
        student_ids = set()

        for i, student in enumerate(data['students']):
            # Проверка обязательных полей студента
            required_student_fields = ['id', 'name', 'age', 'grades', 'is_active']
            for field in required_student_fields:
                if field not in student:
                    errors.append(f"Студент {i + 1}: отсутствует поле {field}")

            # Проверка уникальности ID
            if 'id' in student:
                if student['id'] in student_ids:
                    errors.append(f"Дублирующийся ID студента: {student['id']}")
                student_ids.add(student['id'])

            # Проверка типов данных
            if 'id' in student and not isinstance(student['id'], int):
                errors.append(f"Студент {student.get('name', i + 1)}: ID должен быть числом")

            if 'name' in student and not isinstance(student['name'], str):
                errors.append(f"Студент {student.get('name', i + 1)}: имя должно быть строкой")

            if 'age' in student and not isinstance(student['age'], int):
                errors.append(f"Студент {student.get('name', i + 1)}: возраст должен быть числом")

            if 'grades' in student:
                if not isinstance(student['grades'], list):
                    errors.append(f"Студент {student.get('name', i + 1)}: оценки должны быть массивом")
                else:
                    for j, grade in enumerate(student['grades']):
                        if not isinstance(grade, (int, float)):
                            errors.append(f"Студент {student.get('name', i + 1)}: оценка {j + 1} должна быть
числом")
                        elif grade < 0 or grade > 5:
                            errors.append(f"Студент {student.get('name', i + 1)}: оценка {grade} вне диапазона 0-
5")

            if 'is_active' in student and not isinstance(student['is_active'], bool):
                errors.append(f"Студент {student.get('name', i + 1)}: is_active должен быть булевым
значением")

    # Вывод результатов валидации
    if errors:
        print("Найдены ошибки:")
        for error in errors:
            print(f"  ✗ {error}")
        return False
    else:
        return True
```

```python
def filter_students(data, min_avg_grade=0):
    """Задание 5: Фильтрация данных"""
    filtered = []

    for student in data['students']:
        # Проверка активности
        if not student['is_active']:
            continue

        # Вычисление среднего балла
        avg_grade = sum(student['grades']) / len(student['grades'])

        # Проверка минимального среднего балла
        if avg_grade >= min_avg_grade:
            filtered.append(student)

    return filtered


def advanced_filter_students(data, min_avg_grade=0, max_age=None, min_age=None):
    """Расширенная фильтрация данных"""
    filtered = []

    for student in data['students']:
        # Проверка активности
        if not student['is_active']:
            continue

        # Вычисление среднего балла
        avg_grade = sum(student['grades']) / len(student['grades'])

        # Проверка среднего балла
        if avg_grade < min_avg_grade:
            continue

        # Проверка возраста (если заданы ограничения)
        if min_age is not None and student['age'] < min_age:
            continue

        if max_age is not None and student['age'] > max_age:
            continue

        filtered.append({
            'student': student,
            'avg_grade': avg_grade
        })

    return filtered


if __name__ == '__main__':
    main()
```

Скриншоты выполнения программы

```
✓ Создан файл students.json
============================================================
ЧТЕНИЕ JSON-ФАЙЛА
============================================================
Группа: ИС-21
Год: 2023

СТУДЕНТЫ:
--------------------------------------------------
Иванов Иван, 20 лет
  Средний балл: 4.50, Статус: активен
  Оценки: [4, 5, 4, 5]
------------------------------
Петрова Мария, 19 лет
  Средний балл: 4.75, Статус: активен
  Оценки: [5, 5, 5, 4]
------------------------------
ОБЩИЙ СРЕДНИЙ БАЛЛ ГРУППЫ: 4.62

============================================================
МОДИФИКАЦИЯ JSON-ФАЙЛА
============================================================
✓ Добавлен новый студент: Сидоров Алексей
✓ Обновлен статус студента Петрова Мария на 'неактивен'
✓ Добавлено поле 'department': Информационные системы
✓ Файл успешно обновлен!

============================================================
ВАЛИДАЦИЯ ДАННЫХ
============================================================
✓ Все данные прошли валидацию успешно!
```

```
============================================================
ОБНОВЛЕННЫЕ ДАННЫЕ
============================================================
Группа: ИС-21
Год: 2023

СТУДЕНТЫ:
--------------------------------------------------
Иванов Иван, 20 лет
  Средний балл: 4.50, Статус: активен
  Оценки: [4, 5, 4, 5]
------------------------------
Петрова Мария, 19 лет
  Средний балл: 4.75, Статус: неактивен
  Оценки: [5, 5, 5, 4]
------------------------------
Сидоров Алексей, 21 лет
  Средний балл: 4.00, Статус: активен
  Оценки: [4, 4, 3, 5]
------------------------------
ОБЩИЙ СРЕДНИЙ БАЛЛ ГРУППЫ: 4.42

============================================================
ФИЛЬТРАЦИЯ ДАННЫХ
============================================================
Найдено 2 активных студентов со средним баллом выше 4.0:
  - Иванов Иван (средний балл: 4.50)
  - Сидоров Алексей (средний балл: 4.00)
```