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

