ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«УФИМСКИЙ УНИВЕРСИТЕТ НАУКИ И ТЕХНОЛОГИЙ»

Институт информатики, математики и робототехники

Кафедра математического и компьютерного моделирования

**Лабораторная работа №3: Работа с форматом YAML 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 university.yaml

Написать функцию для чтения файла

Добавить в файл university.yaml:

• Новый факультет name: «Кибернетика», head: «Сидоров О.И.», students\_count: 80

• Обновить количество студентов на факультете «Информационные системы» до 170

• Добавить поле address: «ул. Академическая, д.15» в раздел контактов

Написать функцию, которая проверяет:

• Наличие обязательных полей у каждого факультета (name, head, students\_count)

• Корректность типов данных (students\_count — число, email — строка с символом @)

• Уникальность названий факультетов

Реализовать использование анкоров и ссылок

Реализовать поддержку пользовательских тегов

**Практическая часть:**

Листинг кода с комментариями

*import* yaml  
*from* datetime *import* datetime  
*from* typing *import* Dict, Any  
  
  
*class* UniversityYAMLProcessor:  
 *def \_\_init\_\_*(*self*):  
 yaml.add\_constructor('!date', *self*.date\_constructor)  
 yaml.add\_constructor('!datetime', *self*.datetime\_constructor)  
  
 @staticmethod  
 *def* date\_constructor(loader, node):  
 value = loader.construct\_scalar(node)  
 *return* datetime.strptime(value, '%Y-%m-%d').date()  
  
 @staticmethod  
 *def* datetime\_constructor(loader, node):  
 value = loader.construct\_scalar(node)  
 *return* datetime.strptime(value, '%Y-%m-%d %H:%M:%S')  
  
 *def* create\_initial\_yaml(*self*, filename: *str* = 'university.yaml') -> *None*:  
 data = {  
 'university': "Технический Университет",  
 'departments': [  
 {  
 'name': "Информационные системы",  
 'head': "Иванов А.П.",  
 'students\_count': 150  
 },  
 {  
 'name': "Компьютерная инженерия",  
 'head': "Петрова М.В.",  
 'students\_count': 120  
 }  
 ],  
 'academic\_year': "2025-2026",  
 'contact': {  
 'email': "info@university.edu",  
 'phone': "+7-495-123-45-67"  
 }  
 }  
  
 *with open*(filename, 'w', encoding='utf-8') *as* file:  
 yaml.dump(data, file, allow\_unicode=*True*, default\_flow\_style=*False*)  
  
 *print*(f"✓ Создан файл {filename}")  
  
 *def* read\_yaml\_file(*self*, filename: *str* = 'university.yaml') -> Dict[str, Any]:  
 *"""*  
 *Задание 2: Чтение YAML-файла и вывод информации*  
 *"""*  
 *try*:  
 *with open*(filename, 'r', encoding='utf-8') *as* file:  
 data = yaml.safe\_load(file)  
  
 *print*("=" \* 50)  
 *print*("ИНФОРМАЦИЯ ОБ УНИВЕРСИТЕТЕ")  
 *print*("=" \* 50)  
 *print*(f"Университет: {data['university']}")  
 *print*(f"Учебный год: {data['academic\_year']}\n")  
  
 total\_students = 0  
  
 *print*("ФАКУЛЬТЕТЫ:")  
 *print*("-" \* 30)  
 *for* department *in* data['departments']:  
 *print*(f"Факультет: {department['name']}")  
 *print*(f"Заведующий: {department['head']}")  
 *print*(f"Количество студентов: {department['students\_count']}")  
 *print*("-" \* 20)  
 total\_students += department['students\_count']  
  
 *print*(f"\nОБЩЕЕ КОЛИЧЕСТВО СТУДЕНТОВ: {total\_students}")  
 *print*("=" \* 50)  
  
 *return* data  
  
 *except FileNotFoundError*:  
 *print*(f"✗ Ошибка: Файл {filename} не найден")  
 *return* {}  
 *except KeyError as* e:  
 *print*(f"✗ Ошибка: Отсутствует обязательное поле {e}")  
 *return* {}  
 *except Exception as* e:  
 *print*(f"✗ Ошибка при чтении файла: {e}")  
 *return* {}  
  
 *def* modify\_yaml\_file(*self*, filename: *str* = 'university.yaml') -> Dict[str, Any]:  
 *"""*  
 *Задание 3: Модификация YAML-файла*  
 *"""*  
 *try*:  
 *# Чтение существующих данных*  
 *with open*(filename, 'r', encoding='utf-8') *as* file:  
 data = yaml.safe\_load(file)  
  
 *print*("=" \* 50)  
 *print*("МОДИФИКАЦИЯ ДАННЫХ")  
 *print*("=" \* 50)  
  
 *# Добавление нового факультета*  
new\_department = {  
 'name': "Кибернетика",  
 'head': "Сидоров О.И.",  
 'students\_count': 80  
 }  
 data['departments'].append(new\_department)  
 *print*("✓ Добавлен новый факультет: Кибернетика")  
  
 *# Обновление количества студентов*  
 *for* department *in* data['departments']:  
 *if* department['name'] == "Информационные системы":  
 department['students\_count'] = 170  
 *print*("✓ Обновлено количество студентов на факультете 'Информационные системы' до 170")  
  
 *# Добавление адреса*  
data['contact']['address'] = "ул. Академическая, д. 15"  
 *print*("✓ Добавлен адрес в контакты")  
  
 *# Запись обновленных данных*  
 *with open*(filename, 'w', encoding='utf-8') *as* file:  
 yaml.dump(data, file, allow\_unicode=*True*, default\_flow\_style=*False*)  
  
 *print*("✓ Файл успешно обновлен!")  
 *print*("=" \* 50)  
  
 *return* data  
  
 *except Exception as* e:  
 *print*(f"✗ Ошибка при модификации файла: {e}")  
 *return* {}  
  
 *def* validate\_university\_data(*self*, data: Dict[str, Any]) -> *bool*:  
 *"""*  
 *Задание 4: Валидация данных университета*  
 *"""*  
errors = []  
  
 *print*("=" \* 50)  
 *print*("ВАЛИДАЦИЯ ДАННЫХ")  
 *print*("=" \* 50)  
  
 *# Проверка обязательных полей университета*  
required\_university\_fields = ['university', 'departments', 'academic\_year']  
 *for* field *in* required\_university\_fields:  
 *if* field *not in* data:  
 errors.append(f"Отсутствует обязательное поле университета: {field}")  
  
 *# Проверка факультетов*  
 *if* 'departments' *in* data:  
 department\_names = *set*()  
  
 *for* i, department *in enumerate*(data['departments']):  
 *# Проверка обязательных полей факультета*  
required\_department\_fields = ['name', 'head', 'students\_count']  
 *for* field *in* required\_department\_fields:  
 *if* field *not in* department:  
 errors.append(f"Факультет {i + 1}: отсутствует поле {field}")  
  
 *# Проверка уникальности названий*  
 *if* 'name' *in* department:  
 *if* department['name'] *in* department\_names:  
 errors.append(f"Дублирующееся название факультета: {department['name']}")  
 department\_names.add(department['name'])  
  
 *# Проверка типа students\_count*  
 *if* 'students\_count' *in* department:  
 *if not isinstance*(department['students\_count'], *int*):  
 errors.append(f"Факультет {department.get('name', i + 1)}: students\_count должен быть числом")  
 *elif* department['students\_count'] < 0:  
 errors.append(  
 f"Факультет {department.get('name', i + 1)}: students\_count не может быть отрицательным")  
  
 *# Проверка email*  
 *if* 'contact' *in* data *and* 'email' *in* data['contact']:  
 email = data['contact']['email']  
 *if* '@' *not in* email:  
 errors.append(f"Некорректный email: {email}")  
  
 *if* errors:  
 *print*("НАЙДЕНЫ ОШИБКИ:")  
 *for* error *in* errors:  
 *print*(f" ✗ {error}")  
 *print*("=" \* 50)  
 *return False*  
 *else*:  
 *print*("Все данные прошли валидацию успешно!")  
 *print*("=" \* 50)  
 *return True*  
  
 *def* create\_yaml\_with\_anchors(*self*, filename: *str* = 'university\_with\_anchors.yaml') -> *None*:  
 yaml\_content = """  
# Определение анкоров  
defaults: &default\_contact  
 email: "info@university.edu"  
 phone: "+7-495-123-45-67"  
 address: "ул. Академическая, д. 15"  
  
base\_department: &base\_dept  
 head: "Иванов А.П."  
 building: "Главный корпус"  
  
university: "Технический Университет"  
academic\_year: 2025-2026  
  
departments:  
 - name: "Информационные системы"  
 <<: \*base\_dept  
 students\_count: 170  
 floor: 3  
  
 - name: "Компьютерная инженерия"  
 <<: \*base\_dept  
 head: "Петрова М.В." # Переопределение  
 students\_count: 120  
 floor: 2  
  
 - name: "Кибернетика"  
 <<: \*base\_dept  
 head: "Сидоров О.И."  
 students\_count: 80  
 floor: 4  
  
contact:  
 <<: \*default\_contact # Использование всех полей из анкора  
"""  
 *with open*(filename, 'w', encoding='utf-8') *as* file:  
 file.write(yaml\_content)  
  
 *print*(f"✓ Создан файл с анкорами: {filename}")  
  
 *def* read\_yaml\_with\_anchors(*self*, filename: *str* = 'university\_with\_anchors.yaml') -> Dict[str, Any]:  
 *"""*  
 *Задание 5: Чтение YAML с анкорами и ссылками*  
 *"""*  
 *try*:  
 *with open*(filename, 'r', encoding='utf-8') *as* file:  
 data = yaml.safe\_load(file)  
  
 *print*("=" \* 50)  
 *print*("ДАННЫЕ С ИСПОЛЬЗОВАНИЕМ АНКОРОВ")  
 *print*("=" \* 50)  
 *print*(f"Университет: {data['university']}")  
  
 *print*("\nФАКУЛЬТЕТЫ:")  
 *print*("-" \* 30)  
 *for* department *in* data['departments']:  
 *print*(f"Факультет: {department['name']}")  
 *print*(f"Заведующий: {department['head']}")  
 *print*(f"Здание: {department['building']}")  
 *print*(f"Этаж: {department.get('floor', 'не указан')}")  
 *print*(f"Студентов: {department['students\_count']}")  
 *print*("-" \* 20)  
  
 *print*("\nКОНТАКТЫ:")  
 *for* key, value *in* data['contact'].items():  
 *print*(f" {key}: {value}")  
  
 *print*("=" \* 50)  
 *return* data  
  
 *except Exception as* e:  
 *print*(f"✗ Ошибка при чтении файла с анкорами: {e}")  
 *return* {}  
  
 *def* create\_yaml\_with\_custom\_types(*self*, filename: *str* = 'university\_custom.yaml') -> *None*:  
 *"""*  
 *Задание 6: Создание YAML с пользовательскими типами данных*  
 *"""*  
data = {  
 'university': "Технический Университет",  
 'founded': "!date 1990-05-15",  
 'next\_meeting': "!datetime 2025-03-20 14:30:00",  
 'departments': [  
 {  
 'name': "Информационные системы",  
 'established': "!date 2005-09-01",  
 'students\_count': 170  
 },  
 {  
 'name': "Компьютерная инженерия",  
 'established': "!date 2008-03-10",  
 'students\_count': 120  
 }  
 ],  
 'academic\_year': "2025-2026"  
 }  
  
 *with open*(filename, 'w', encoding='utf-8') *as* file:  
 yaml.dump(data, file, allow\_unicode=*True*, default\_flow\_style=*False*)  
  
 *print*(f"✓ Создан файл с пользовательскими типами: {filename}")  
  
 *def* read\_yaml\_with\_custom\_types(*self*, filename: *str* = 'university\_custom.yaml') -> Dict[str, Any]:  
 *"""*  
 *Задание 6: Чтение YAML с пользовательскими типами данных*  
 *"""*  
 *try*:  
 *with open*(filename, 'r', encoding='utf-8') *as* file:  
 content = file.read()  
 data = yaml.load(content, Loader=yaml.Loader)  
  
 *print*("=" \* 50)  
 *print*("ДАННЫЕ С ПОЛЬЗОВАТЕЛЬСКИМИ ТИПАМИ")  
 *print*("=" \* 50)  
 *print*(f"Университет: {data['university']}")  
 *print*(f"Основан: {data['founded']} (тип: {*type*(data['founded']).*\_\_name\_\_*})")  
 *print*(f"Следующее собрание: {data['next\_meeting']} (тип: {*type*(data['next\_meeting']).*\_\_name\_\_*})")  
  
 *print*("\nФАКУЛЬТЕТЫ:")  
 *print*("-" \* 30)  
 *for* department *in* data['departments']:  
 *print*(f"Факультет: {department['name']}")  
 *print*(f"Основан: {department['established']} (тип: {*type*(department['established']).*\_\_name\_\_*})")  
 *print*(f"Студентов: {department['students\_count']}")  
 *print*("-" \* 20)  
  
 *print*("=" \* 50)  
 *return* data  
  
 *except Exception as* e:  
 *print*(f"✗ Ошибка при чтении файла с пользовательскими типами: {e}")  
 *return* {}  
  
  
*def* main():  
 processor = UniversityYAMLProcessor()  
  
 *# Задание 1: Создание начального файла*  
 *print*("\n1. СОЗДАНИЕ НАЧАЛЬНОГО YAML-ФАЙЛА")  
 processor.create\_initial\_yaml()  
  
 *# Задание 2: Чтение файла*  
 *print*("\n2. ЧТЕНИЕ YAML-ФАЙЛА")  
 data = processor.read\_yaml\_file()  
  
 *# Задание 3: Модификация файла*  
 *print*("\n3. МОДИФИКАЦИЯ YAML-ФАЙЛА")  
 modified\_data = processor.modify\_yaml\_file()  
  
 *# Задание 4: Модификация файла*  
 *print*("\n3. ВАЛИДАЦИЯ ДАННЫХ")  
 *if not* processor.validate\_university\_data(modified\_data):  
 *return*  
  
 *# Чтение обновленного файла*  
 *print*("\n5. ЧТЕНИЕ ОБНОВЛЕННОГО ФАЙЛА")  
 processor.read\_yaml\_file()  
  
 *# Задание 5: Работа с анкорами*  
 *print*("\n6. РАБОТА С АНКОРАМИ И ССЫЛКАМИ")  
 processor.create\_yaml\_with\_anchors()  
 processor.read\_yaml\_with\_anchors()  
  
 *# Задание 6: Пользовательские типы данных*  
 *print*("\n7. ПОЛЬЗОВАТЕЛЬСКИЕ ТИПЫ ДАННЫХ")  
 processor.create\_yaml\_with\_custom\_types()  
 processor.read\_yaml\_with\_custom\_types()  
  
  
*if* \_\_name\_\_ == "\_\_main\_\_":  
 main()

Скриншоты выполнения программы

