

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное учреждение  
высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития  
Кафедра инфокоммуникаций

**ОТЧЕТ**  
**ПО ЛАБОРАТОРНОЙ РАБОТЕ №2.16**  
**дисциплины «Основы программной инженерии»**

Выполнил:  
Баратов Семен Григорьевич  
2 курс, группа ПИЖ-б-о-22-1,  
09.03.04 «Программная инженерия»,  
направленность (профиль) «Разработка  
и сопровождение программного  
обеспечения», очная форма обучения

---

(подпись)

Преподаватель:  
Воронкин Р.А., канд. тех. наук, доцент,  
доцент кафедры инфокоммуникаций

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

Ставрополь, 2024 г.

**Тема:** Работа с данными формата JSON в языке Python

**Цель:** приобретение навыков по работе с данными формата JSON с помощью языка программирования Python версии 3.x.

### Результаты выполнения

1. Создали репозиторий с лицензией MIT, добавили в .gitignore необходимые правила для работы с IDE PyCharm, клонировали репозиторий, организовали репозиторий в соответствии с моделью git-flow.

```
Last login: Tue Oct 24 20:29:02 on ttys000
itssyoma@MacBook-Air-Sema Основы программной инженерии % git clone https://github.com/itssyoma/megarepo_21.git
Cloning into 'megarepo_21'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
itssyoma@MacBook-Air-Sema Основы программной инженерии % cd megarepo_21
itssyoma@MacBook-Air-Sema megarepo_21 % git checkout -b develop
Switched to a new branch 'develop'
itssyoma@MacBook-Air-Sema megarepo_21 % git branch release
itssyoma@MacBook-Air-Sema megarepo_21 % git branch develop
fatal: a branch named 'develop' already exists
itssyoma@MacBook-Air-Sema megarepo_21 % git branch hotfix
itssyoma@MacBook-Air-Sema megarepo_21 % git branch feature
itssyoma@MacBook-Air-Sema megarepo_21 %
```

Рисунок 1 – Работа с репозиторием в командной строке.

2. Проработали пример лабораторной работы. Для примера 1 лабораторной работы 2.8 добавьте возможность сохранения списка в файл формата JSON и чтения данных из файла JSON.

Код программы:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# Для примера 1 лабораторной работы 2.8 добавьте возможность
# сохранения списка в файл формата JSON и чтения данных из файла JSON.

import json
import sys
from datetime import date

def get_worker():
    """
    Запросить данные о работнике.
    """
    name = input("Фамилия и инициалы? ")
    post = input("Должность? ")
    year = int(input("Год поступления? "))

    # Создать словарь.
    return {
        'name': name,
        'post': post,
```

```

        'year': year,
    }

def display_workers(staff):
    """
    Отобразить список работников.
    """
    # Проверить, что список работников не пуст.
    if staff:
        # Заголовок таблицы.
        line = '+-{}-+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 8
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^8} |'.format(
                "No",
                "Ф.И.О.",
                "Должность",
                "Год"
            )
        )
        print(line)

        # Вывести данные о всех сотрудниках.
        for idx, worker in enumerate(staff, 1):
            print(
                '| {:>4} | {:<30} | {:<20} | {:>8} |'.format(
                    idx,
                    worker.get('name', ''),
                    worker.get('post', ''),
                    worker.get('year', 0)
                )
            )
            print(line)

    else:
        print("Список работников пуст.")

def select_workers(staff, period):
    """
    Выбрать работников с заданным стажем.
    """
    # Получить текущую дату.
    today = date.today()

    # Сформировать список работников.
    result = []
    for employee in staff:

```

```

        if today.year - employee.get('year', today.year) >= period:
            result.append(employee)

# Возвратить список выбранных работников.
return result

def save_workers(file_name, staff):
    """
    Сохранить всех работников в файл JSON.
    """
    # Открыть файл с заданным именем для записи.
    with open(file_name, "w", encoding="utf-8") as fout:
        # Выполнить сериализацию данных в формат JSON.
        # Для поддержки кириллицы установим ensure_ascii=False
        json.dump(staff, fout, ensure_ascii=False, indent=4)

def load_workers(file_name):
    """
    Загрузить всех работников из файла JSON.
    """
    # Открыть файл с заданным именем для чтения.
    with open(file_name, "r", encoding="utf-8") as fin:
        return json.load(fin)

def main():
    """
    Главная функция программы.
    """
    # Список работников.
    workers = []

    # Организовать бесконечный цикл запроса команд.
    while True:
        # Запросить команду из терминала.
        command = input(">>> ").lower()

        # Выполнить действие в соответствие с командой.
        if command == "exit":
            break

        elif command == "add":
            # Запросить данные о работнике.
            worker = get_worker()
            # Добавить словарь в список.
            workers.append(worker)
            # Отсортировать список в случае необходимости.
            if len(workers) > 1:
                workers.sort(key=lambda item: item.get('name', ''))

        elif command == "list":
            # Отобразить всех работников.

```

```

display_workers(workers)

elif command.startswith("select "):
    # Разбить команду на части для выделения стажа.
    parts = command.split(maxsplit=1)
    # Получить требуемый стаж.
    period = int(parts[1])
    # Выбрать работников с заданным стажем.
    selected = select_workers(workers, period)
    # Отобразить выбранных работников.
    display_workers(selected)

elif command.startswith("save "):
    # Разбить команду на части для выделения имени файла.
    parts = command.split(maxsplit=1)
    # Получить имя файла.
    file_name = parts[1]
    # Сохранить данные в файл с заданным именем.
    save_workers(file_name, workers)

elif command.startswith("load "):
    # Разбить команду на части для выделения имени файла.
    parts = command.split(maxsplit=1)
    # Получить имя файла.
    file_name = parts[1]
    # Сохранить данные в файл с заданным именем.
    workers = load_workers(file_name)

elif command == 'help':
    # Вывести справку о работе с программой.
    print("Список команд:\n")
    print("add – добавить работника;")
    print("list – вывести список работников;")
    print("select <стаж> – запросить работников со стажем;")
    print("help – отобразить справку;")
    print("load – загрузить данные из файла;")
    print("save – сохранить данные в файл;")
    print("exit – завершить работу с программой.")

else:
    print(f"Неизвестная команда {command}", file=sys.stderr)

if __name__ == '__main__':
    main()

```

```

(env) itssyoma@MacBook-Air-Sema megarepo_216 % "/Users/itssyoma/Yandex.Disk.l
sk.localized/Учеба/4 семестр/Основы программной инженерии/megarepo_216/examp
>>> help
Список команд:

add – добавить работника;
list – вывести список работников;
select <стаж> – запросить работников со стажем;
help – отобразить справку;
load – загрузить данные из файла;
save – сохранить данные в файл;
exit – завершить работу с программой.
>>> hello
Неизвестная команда hello
>>> add
Фамилия и инициалы? Баратов С.Г.
Должность? Студент
Год поступления? 2022
>>> add
Фамилия и инициалы? Иванов А.А.
Должность? Студент
Год поступления? 2021
>>> list

```

No	Ф.И.О.	Должность	Год
1	Баратов С.Г.	Студент	2022
2	Иванов А.А.	Студент	2021

```

>>> select 1

```

No	Ф.И.О.	Должность	Год
1	Баратов С.Г.	Студент	2022
2	Иванов А.А.	Студент	2021

```

>>> select 3

```

No	Ф.И.О.	Должность	Год
1	Иванов А.А.	Студент	2021

```

>>> save data.json
>>>

```

Рисунок 2 – Результат выполнения программы, сохранение данных в файл data.json

```

{} data.json > ...
1
2 {
3   "name": "Баратов С.Г.",
4   "post": "Студент",
5   "year": 2022
6 },
7 {
8   "name": "Иванов А.А.",
9   "post": "Студент",
10  "year": 2021
11 }
12 ]

```

Рисунок 3 – JSON-файл с экспортированными данными

```

(env) itssyoma@MacBook-Air-Sema megarepo_216 % "/Users/itssyoma/Yandex.Disk.l
sk.localized/Учеба/4 семестр/Основы программной инженерии/megarepo_216/examp
>>> list
Список работников пуст.
>>> load data.json
>>> list

```

No	Ф.И.О.	Должность	Год
1	Баратов С.Г.	Студент	2022
2	Иванов А.А.	Студент	2021

Рисунок 4 – Импорт данных из JSON-файла

3. Выполнили индивидуальное задание. Для своего варианта лабораторной работы 2.8 необходимо дополнительно реализовать сохранение и чтение данных из файла формата JSON. Необходимо также проследить за тем, чтобы файлы генерируемый этой программой не попадали в репозиторий лабораторной работы.

Предварительно внесем корректировки в файл .gitignore, чтобы файлы, генерируемые программой не попадали в versionный контроль, и, соответственно, в репозиторий лабораторной работы.



```
.gitignore
1  # For MacOS
2  .DS_Store/
3
4  # JSON
5  *.json
```

Рисунок 5 – Корректировки в файле .gitignore.

Код программы:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sys
import json

def command_add(students):
    # Запросить данные о студенте.
    name = input("Фамилия и инициалы? ")
    group = input("Номер группы? ")
    grade = list(map(int, input("Успеваемость студента? (Пять оценок через пробел) ").split()))
    while True:
        if len(grade) < 5:
            print("Введенное количество оценок меньше 5, введите оценки еще раз: ",
file=sys.stderr)
            grade = list(map(int, input("Успеваемость студента? (Пять оценок через пробел) ").split()))
        else:
            break

    # Создать словарь.
    if sum(grade)/len(grade) >= 4.0:
        student = {
            'name': name,
            'group': group,
            'grade': sum(grade)/len(grade),
```

```

    }
    # Добавить словарь в список.
    students.append(student)

# Отсортировать список в случае необходимости.
if len(students) > 1:
    students.sort(key=lambda item: item.get('name', ''))

return students

def command_list(students):
    if students:
        print("Список студентов с успеваемостью больше 4.0")
        # Заголовок таблицы.
        line = '+--{}--{}--{}--+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} |'.format(
                "No",
                "Ф.И.О.",
                "Группа"
            )
        )
        print(line)

        # Вывести данные о всех сотрудниках.
        for idx, student in enumerate(students, 1):
            print(
                '| {:>4} | {:<30} | {:<20} |'.format(
                    idx,
                    student.get('name', ''),
                    student.get('group', '')
                )
            )
            print(line)

    else:
        print("Студентов с успеваемостью выше 4.0 нет")

def save_students(file_name, staff):
    # Сохранить всех студентов в файл JSON
    # Открыть файл с заданным именем для записи.
    with open(file_name, "w", encoding="utf-8") as fout:
        # Выполнить сериализацию данных в формат JSON.
        # Для поддержки кириллицы установим ensure_ascii=False
        json.dump(staff, fout, ensure_ascii=False, indent=4)

```



```

def load_students(file_name):
    # Загрузить всех студентов из файла JSON.
    # Открыть файл с заданным именем для чтения.
    with open(file_name, "r", encoding="utf-8") as fin:
        return json.load(fin)

if __name__ == '__main__':

    students = []

    # Организовать бесконечный цикл запроса команд.
    while True:
        # Запросить команду из терминала.
        command = input(">>> ").lower()

        # Выполнить действие в соответствие с командой.
        if command == 'exit':
            break
        elif command == 'add':
            students = command_add(students)
        elif command == 'list':
            command_list(students)
        elif command.startswith('save'):
            # Разбить команду на части для выделения имени файла.
            parts = command.split(maxsplit=1)
            # Получить имя файла.
            file_name = parts[1]
            # Сохранить данные в файл с заданным именем.
            save_students(file_name, students)
        elif command.startswith('load'):
            # Разбить команду на части для выделения имени файла.
            parts = command.split(maxsplit=1)
            # Получить имя файла.
            file_name = parts[1]
            # Сохранить данные в файл с заданным именем.
            students = load_students(file_name)

        else:
            print(f"Неизвестная команда {command}", file=sys.stderr)

```

```

o (env) itssyoma@MacBook-Air-Sema megarepo_216 % "/Users/itssyoma/Ya
sk.localized/Учеба/4 семестр/Основы программной инженерии/megarepo
>>> add
Фамилия и инициалы? Баратов С.Г.
Номер группы? 221
Успеваемость студента? (Пять оценок через пробел) 5 5 5 5 5
>>> add
Фамилия и инициалы? Иванов А.А.
Номер группы? 222
Успеваемость студента? (Пять оценок через пробел) 5 5 5 5 5
>>> list
Список студентов с успеваемостью больше 4.0
+-----+-----+-----+
| No | Ф.И.О. | Группа |
+-----+-----+-----+
| 1 | Баратов С.Г. | 221 |
| 2 | Иванов А.А. | 222 |
+-----+-----+-----+
>>> save data.json
>>> 

```

Рисунок 6 – Результат работы программы

```
{ } data.json > ...  
1  
2 {  
3     "name": "Баратов С.Г.",  
4     "group": "221",  
5     "grade": 5.0  
6 },  
7 {  
8     "name": "Иванов А.А.",  
9     "group": "222",  
10    "grade": 5.0  
11 }  
12
```

Рисунок 7 – Содержимое JSON-файла

```
o (env) itssyoma@MacBook-Air-Sema megarepo_216 % "/Users/itssyoma/Yan  
sk.localized/Учеба/4 семестр/Основы программной инженерии/megarepo_  
>>> list  
Студентов с успеваемостью выше 4.0 нет  
>>> load data.json  
>>> list  
Список студентов с успеваемостью больше 4.0  
+-----+-----+-----+  
| No | Ф.И.О. | Группа |  
+-----+-----+-----+  
| 1 | Баратов С.Г. | 221 |  
| 2 | Иванов А.А. | 222 |  
+-----+-----+-----+  
>>>
```

Рисунок 8 – Результат импорта данных из JSON-файла

4. Выполнили задание повышенной сложности. Очевидно, что программа в примере 1 и в индивидуальном задании никак не проверяет правильность загружаемых данных формата JSON. В следствие чего, необходимо после загрузки из файла JSON выполнять валидацию загруженных данных. Валидацию данных необходимо производить с использованием спецификации JSON Schema, описанной на сайте <https://json-schema.org/>. Одним из возможных вариантов работы с JSON Schema является использование пакета `jsonschema`, который не является частью стандартной библиотеки Python. Таким образом, необходимо реализовать валидацию загруженных данных с помощью спецификации JSON Schema.

Код программы:

```
#!/usr/bin/env python3  
# -*- coding: utf-8 -*-  
  
import sys  
import json
```

```

from jsonschema import validate
from jsonschema.exceptions import ValidationError

def validation(instance):
    schema = {
        "type": "array",
        "items": {
            "type": "object",
            "properties": {
                "name": {"type": "string"},
                "group": {"type": "string"},
                "grade": {"type": "number"},
            },
            "required": ["name", "group", "grade"],
        },
    }

    try:
        validate(instance, schema=schema)
        return True
    except ValidationError as err:
        print(err.message)
        return False

def command_add(students):
    # Запросить данные о студенте.
    name = input("Фамилия и инициалы? ")
    group = input("Номер группы? ")
    grade = list(map(int, input("Успеваемость студента? (Пять оценок через пробел) ").split()))
    while True:
        if len(grade) < 5:
            print("Введенное количество оценок меньше 5, введите оценки еще раз: ",
file=sys.stderr)
            grade = list(map(int, input("Успеваемость студента? (Пять оценок через пробел) ").split()))
        else:
            break

    # Создать словарь.
    if sum(grade)/len(grade) >= 4.0:
        student = {
            'name': name,
            'group': group,
            'grade': sum(grade)/len(grade),
        }
        # Добавить словарь в список.
        students.append(student)

    # Отсортировать список в случае необходимости.
    if len(students) > 1:
        students.sort(key=lambda item: item.get('name', ''))

```

```

return students

def command_list(students):
    if students:
        print("Список студентов с успеваемостью больше 4.0")
        # Заголовок таблицы.
        line = '+--{}--{}--{}--+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} |'.format(
                "No",
                "Ф.И.О.",
                "Группа"
            )
        )
        print(line)

        # Вывести данные о всех сотрудниках.
        for idx, student in enumerate(students, 1):
            print(
                '| {:>4} | {:<30} | {:<20} |'.format(
                    idx,
                    student.get('name', ''),
                    student.get('group', '')
                )
            )

            print(line)

    else:
        print("Студентов с успеваемостью выше 4.0 нет")

def save_students(file_name, staff):
    # Сохранить всех студентов в файл JSON
    # Открыть файл с заданным именем для записи.
    with open(file_name, "w", encoding="utf-8") as fout:
        # Выполнить сериализацию данных в формат JSON.
        # Для поддержки кириллицы установим ensure_ascii=False
        json.dump(staff, fout, ensure_ascii=False, indent=4)

def load_students(file_name):
    # Загрузить всех студентов из файла JSON.
    # Открыть файл с заданным именем для чтения.
    with open(file_name, "r", encoding="utf-8") as fin:
        data = json.load(fin)

```

```

if validation(data):
    return data

if __name__ == '__main__':

    students = []

    # Организовать бесконечный цикл запроса команд.
    while True:
        # Запросить команду из терминала.
        command = input(">>> ").lower()

        # Выполнить действие в соответствие с командой.
        if command == 'exit':
            break
        elif command == 'add':
            students = command_add(students)
        elif command == 'list':
            command_list(students)
        elif command.startswith('save'):
            # Разбить команду на части для выделения имени файла.
            parts = command.split(maxsplit=1)
            # Получить имя файла.
            file_name = parts[1]
            # Сохранить данные в файл с заданным именем.
            save_students(file_name, students)
        elif command.startswith('load'):
            # Разбить команду на части для выделения имени файла.
            parts = command.split(maxsplit=1)
            # Получить имя файла.
            file_name = parts[1]
            # Сохранить данные в файл с заданным именем.
            students = load_students(file_name)

    else:
        print(f"Неизвестная команда {command}", file=sys.stderr)

```

```

○ (env) itssyoma@MacBook-Air-Sema megarepo_216 % "/Users/itssyoma/Yi
sk.localized/Учеба/4 семестр/Основы программной инженерии/мегагери
>>> list
Студентов с успеваемостью выше 4.0 нет
>>> load data.json
>>> list
Список студентов с успеваемостью больше 4.0

```

No	Ф.И.О.	Группа
1	Баратов С.Г.	221
2	Иванов А.А.	222

Рисунок 9 – Успешный ввод данных

```
○ (env) itssyoma@MacBook-Air-Sema megarepo_2
sk.localized/Учеба/4 семестр/Основы програ
>>> list
Студентов с успеваемостью выше 4.0 нет
>>> load data.json
'5.0' is not of type 'number'
>>> list
Студентов с успеваемостью выше 4.0 нет
>>> █
```

Рисунок 10 – Ввод данных, не прошедший валидацию

## Ответы на контрольные вопросы

Конечно, я готов помочь с ответами на вопросы о JSON:

1. JSON (JavaScript Object Notation) используется для обмена данными между сервером и клиентом веб-приложений.
2. Типы значений, используемые в JSON, включают строки, числа, логические значения (true/false), массивы, объекты, null.
3. Работа со сложными данными в JSON осуществляется путем организации данных в виде вложенных объектов и массивов.
4. Формат данных JSON5 является расширением формата JSON и добавляет поддержку комментариев, одиночных кавычек для строк и другие улучшения.
5. Для работы с данными в формате JSON5 на языке Python можно использовать библиотеку json5.
6. В Python для сериализации данных в формате JSON используется модуль json.
7. Функция json.dump() используется для записи данных в файл, а json.dumps() для преобразования данных в строку.
8. Для десериализации данных из формата JSON в Python используются функции json.load() и json.loads().

9. Для работы с данными формата JSON, содержащими кириллицу, необходимо убедиться, что данные правильно кодируются и декодируются с учетом кодировки Unicode.

10. JSON Schema – это спецификация для описания структуры данных в формате JSON. Схема данных определяет правила для валидации и проверки данных в формате JSON.

Пример схемы данных для примера №1 лабораторной работы:

```
schema = {  
  "type": "array",  
  "items": {  
    "type": "object",  
    "properties": {  
      "name": {"type": "string"},  
      "post": {"type": "string"},  
      "year": {"type": "integer"},  
    },  
  },  
  "required": ["name", "post", "year"],  
}
```