

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**  
**дисциплины «Анализ данных»**  
Вариант №2

Выполнила:  
Беседина Инга Олеговна  
2 курс, группа ИВТ-б-о-22-1,  
09.03.01 «Информатика и  
вычислительная техника», очная  
форма обучения

---

(подпись)

Руководитель практики:  
Воронкин Р. А., канд. технических  
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

### Ход работы

Создание виртуального окружения:

```
PS C:\Rep\Data_analysis_2\Project> py -m venv C:\Rep\Data_analysis_2\Project\venv
```

Рисунок 1. Создание виртуального окружения

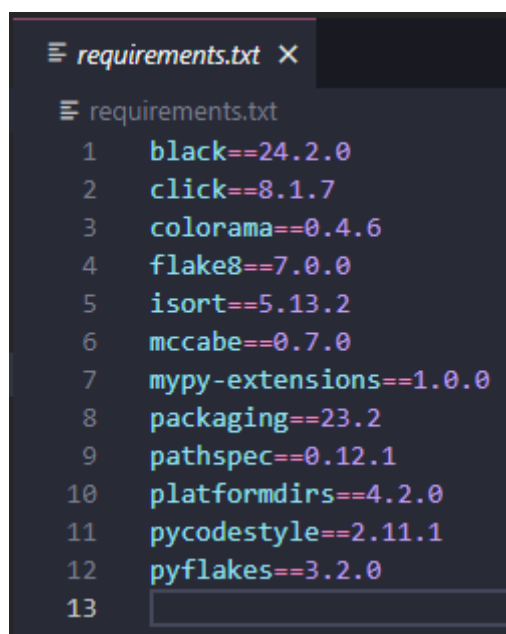
```
PS C:\Rep\Data_analysis_2\Project> venv\Scripts\activate.bat  
PS C:\Rep\Data_analysis_2\Project> venv\Scripts\Activate.ps1  
(venv) PS C:\Rep\Data_analysis_2\Project> |
```

Рисунок 2. Активация виртуального окружения

Установка модулей и создание файла requirements.txt:

```
(venv) PS C:\Rep\Data_analysis_2\Project> pip freeze > requirements.txt
```

Рисунок 3. Создание файла requirements.txt



```
requirements.txt  
1 black==24.2.0  
2 click==8.1.7  
3 colorama==0.4.6  
4 flake8==7.0.0  
5 isort==5.13.2  
6 mccabe==0.7.0  
7 mypy_extensions==1.0.0  
8 packaging==23.2  
9 pathspec==0.12.1  
10 platformdirs==4.2.0  
11 pycodestyle==2.11.1  
12 pyflakes==3.2.0  
13
```

Рисунок 4. Установленные модули

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

```
#!/usr/bin/env python3  
# -*- coding: utf-8 -*-  
  
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(
                "№",
                "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()

```

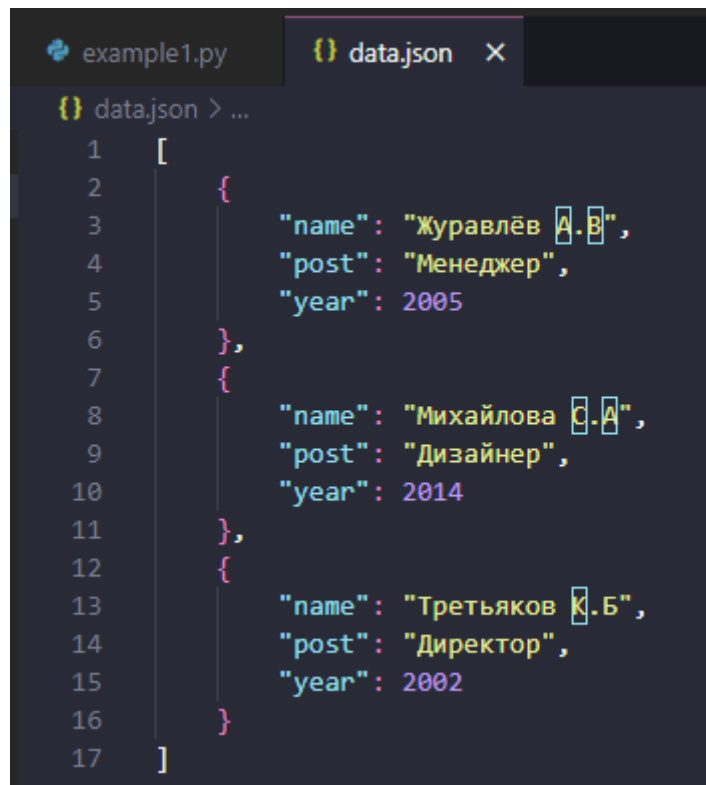
```

>>> 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.
>>> load data.json
>>> add
Фамилия и инициалы? Михайлова С.А
Должность? Дизайнер
Год поступления? 2014
>>> add
Фамилия и инициалы? Журавлёв А.В
Должность? Менеджер
Год поступления? 2005
>>> list
+-----+-----+-----+-----+
| № |           Ф.И.О.           |      Должность      |      Год      |
+-----+-----+-----+-----+
|  1 | Журавлёв А.В              | Менеджер            | 2005          |
|  2 | Михайлова С.А             | Дизайнер            | 2014          |
|  3 | Третьяков К.Б             | Директор            | 2002          |
+-----+-----+-----+-----+
>>> select 15
+-----+-----+-----+-----+
| № |           Ф.И.О.           |      Должность      |      Год      |
+-----+-----+-----+-----+
|  1 | Журавлёв А.В              | Менеджер            | 2005          |
|  2 | Третьяков К.Б             | Директор            | 2002          |
+-----+-----+-----+-----+
>>> save data.json
>>> exit

```

Рисунок 5. Результат работы программы



```
example1.py data.json X
{} data.json > ...
1  [
2      {
3          "name": "Журавлёв А.В.",
4          "post": "Менеджер",
5          "year": 2005
6      },
7      {
8          "name": "Михайлова С.А.",
9          "post": "Дизайнер",
10         "year": 2014
11     },
12     {
13         "name": "Третьяков К.Б.",
14         "post": "Директор",
15         "year": 2002
16     }
17 ]
```

Рисунок 6. Файл json

### Индивидуальное задание:

- Использовать словарь, содержащий следующие ключи: фамилия и инициалы; номер группы; успеваемость (список из пяти элементов). Написать программу, выполняющую следующие действия: ввод с клавиатуры данных в список, состоящий из словарей заданной структуры; записи должны быть упорядочены по возрастанию среднего балла; вывод на дисплей фамилий и номеров групп для всех студентов, имеющих оценки 4 и 5; если таких студентов нет, вывести соответствующее сообщение.

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

import json
import sys

def get_student():
    """
    Запросить данные о студенте.
    """
    surname = input("Введите фамилию и инициалы: ")
    group_num = input("Введите номер группы: ")
    print('Введите оценки: ')
    grades = [int(n) for n in input().split()]

    student = {
        'name': surname,
        'group_number': group_num,
```

```

        'grades': grades
    }

    return student

def display_students(staff):
    """
    Отобразить список студентов.
    """
    # Проверить, что список студентов не пуст.
    if staff:
        # Заголовок таблицы.
        line = '+--{}--{}--{}--{}--+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20,
            '-' * 14
        )
        print(line)
        print(
            '| {:^4} | {:^30} | {:^20} | {:^14} |'.format(
                "№",
                "Ф.И.О.",
                "Группа",
                "Оценки"
            )
        )
        print(line)

        # Вывести данные о всех студентах.
        for idx, student in enumerate(staff, 1):
            print(
                '| {:>4} | {:<30} | {:<20} | {:>14} |'.format(
                    idx,
                    student.get('name', ''),
                    student.get('group_number', ''),
                    ', '.join(str(el) for el in student.get('grades'))
                )
            )
            print(line)

        else:
            print("Список студентов пуст.")

def select_students(staff):
    # Сформировать список студентов, имеющих оценки 4 и 5.
    result = []
    for student in staff:
        if all(grade >= 4 for grade in student['grades']):

```



```

        result.append(student)

    # Возвратить список выбранных студентов.
    return result

def save_students(file_name, staff):
    """
    Сохранить всех учеников 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 учеников 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 студентов.
    students = []

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

        # Выполнить действие в соответствии с командой.
        if command == 'exit':
            break

        elif command == 'add':
            # Запросить данные о работнике.
            student = get_student()

            # Добавить словарь в список.
            students.append(student)
            # Сортировка по возрастанию среднего балла
            students.sort(key=lambda x: sum(x['grades']) / 5)

```

```
elif command == 'list':
    # Отобразить всех студентов.
    display_students(students)

elif command == 'select':
    selected = select_students(students)
    # Отобразить выбранных студентов.
    display_students(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_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)

elif command == 'help':
    # Вывести справку о работе с программой.
    print("Список команд:\n")
    print("add - добавить студента;")
    print("list - вывести список студентов;")
    print("select - запросить студентов, имеющих оценки 4, 5;")
    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()
```

```

>>> help
Список команд:

add - добавить студента;
list - вывести список студентов;
select - запросить студентов, имеющих оценки 4, 5;
help - отобразить справку;
load - загрузить данные из файла;
save - сохранить данные в файл;
exit - завершить работу с программой.
>>> add
Введите фамилию и инициалы: Журавлёв А.В
Введите номер группы: 1
Введите оценки:
5 5 5 4 5
>>> add
Введите фамилию и инициалы: Михайлова С.А
Введите номер группы: 2
Введите оценки:
3 3 4 3 4
>>> add
Введите фамилию и инициалы: Третьяков К.Б
Введите номер группы: 3
Введите оценки:
4 4 4 5 4
>>> list
+-----+-----+-----+-----+
| № | Ф.И.О. | Группа | Оценки |
+-----+-----+-----+-----+
| 1 | Михайлова С.А | 2 | 3, 3, 4, 3, 4 |
| 2 | Третьяков К.Б | 3 | 4, 4, 4, 5, 4 |
| 3 | Журавлёв А.В | 1 | 5, 5, 5, 4, 5 |
+-----+-----+-----+-----+
>>> select
+-----+-----+-----+-----+
| № | Ф.И.О. | Группа | Оценки |
+-----+-----+-----+-----+
| 1 | Третьяков К.Б | 3 | 4, 4, 4, 5, 4 |
| 2 | Журавлёв А.В | 1 | 5, 5, 5, 4, 5 |
+-----+-----+-----+-----+
>>> save indiv_data.json
>>> exit

```

Рисунок 7. Результат работы программы

```

[
  {
    "name": "Михайлова С.А",
    "group_number": "2",
    "grades": [
      3,
      3,
      4,
      3,
      4
    ]
  }
]

```

```

    },
    {
      "name": "Третьяков К.Б",
      "group_number": "3",
      "grades": [
        4,
        4,
        4,
        5,
        4
      ]
    },
    {
      "name": "Журавлёв А.В",
      "group_number": "1",
      "grades": [
        5,
        5,
        5,
        4,
        5
      ]
    }
  ]
}
]

```

## 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.

Схема:

```

{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "items": {
    "type": "object",
    "properties": {
      "name": {
        "type": "string"
      },
      "group_number": {
        "type": "string",
        "format": "integer"
      }
    }
  }
}

```

```

    },
    "grades": {
        "type": "array",
        "items": {
            "type": "integer"
        }
    }
},
"additionalProperties": false,
"required": [
    "name",
    "group_number",
    "grades"
]
}
}

```

Валидация данных при загрузке:

```

def load_students(file_name):
    """
    Загрузить всех учеников из файла JSON.
    """
    # Загрузка схемы.
    with open("student-schema.json", "r") as s:
        schema = json.load(s)

    try:
        validate(instance=file_name, schema=schema)
        print("Validation succeeded!")
        with open(file_name, "r", encoding="utf-8") as fin:
            return json.load(fin)
    except ValidationError as e:
        print("Validation failed!")
        print(f"Error message: {e.message}")

```

```

>>> load indv_data.json
Validation succeeded!
>>> list

```

№	Ф.И.О.	Группа	Оценки
1	Михайлова С.А	2	3, 3, 4, 3, 4
2	Третьяков К.Б	3	4, 4, 4, 5, 4
3	Журавлёв А.В	1	5, 5, 5, 4, 5

```

>>> exit

```

Рисунок 8. Результат работы программы

### **Контрольные вопросы:**

1. Формат JSON подходит для сериализации сложных структур. Применяется в веб-приложениях как для обмена данными между браузером и сервером (AJAX), так и между серверами (программные HTTP-сопряжения)
2. JSON значения могут быть одним из шести типов данных: строкой, числом, объектом, массивом, булевым значением или null
3. Сложные данные в JSON передаются, как значения назначенные ключам и представляют собой связку ключ-значение
4. JSON5 - расширение JSON, которое добавляет некоторые новые функции и улучшения для удобства чтения и написания. Основные отличия JSON5 и JSON: JSON5 поддерживает однострочные комментарии, позволяет опускать запятые после последнего элемента в объекте или массиве, дополнительные типы данных, такие как NaN, infinity
5. Для работы с данными в формате JSON5 в языке Python можно использовать библиотеку JSON5, предназначенную для чтения и записи данных в формате JSON5
6. Библиотека json; json.dump(); json.dumps()
7. json.dump() - конвертировать python объект в json и записать в файл; json.dumps() - тоже самое, но в строку
8. json.load() - прочитать json из файла и конвертировать в python объект json.loads() - тоже самое, но из строки с json (s на конце от string/строка)
9. Для работы с данными формата JSON, содержащими кириллицу, устанавливают ensure\_ascii=False
10. Схема JSON - это инструмент для проверки структуры данных JSON

**Вывод:** в ходе выполнения лабораторной работы были приобретены навыки по работе с данными формата JSON с помощью языка программирования Python версии 3.x

